PCP-129

GE-PAC* 30
CONTROL COMPUTER

PROGRAMMING MANUAL

GENERAL ELECTRIC

PCP-129 B
8/70 (300)

GE-PAC 30
CONTROL COMPUTER

PROGRAMMING MANUAL

GENERAL ELECTRIC

GE 29-013R03

REVISION RECORD

PUBLICATION: PCP-129B
GE-PAC 30 PROGRAMMING MANUAL

REVISION OR PAGE NO.	DATE
Under INTRODUCTION Tab	8/70
All Pages	
Under DEVICE DESCRIPTION Tab	
Selector Channel Programming Manual Section	
Page 5	
Eliminate Page 6	
Eliminate Appendix 1	
Under LOADER Tab	
All Sections and Appendices	
Under TEST PROGRAMS Tag	
GE-PAC Model 30-1 Test Program Description and Operating Instructions	
Page 1	
Page 2	
Operating Instructions For the ASR 33 and ASR 35 Teletypewriter Test Program	
Page 1	
SECTIONS ADDED:	
Operating Instructions For the Teletypewriter/Terminate Test Programs	
Appendix 1	
Operating Instruction for the High Speed Paper Tape Reader Test Program	
Appendix 1	
High Speed Paper Tape Punch Test Program	
Appendix 1	
Card Reader Test	
Appendix 1	

CONTENTS

TABLE OF CONTENTS

GENERAL DESCRIPTION

1. INTRODUCTION

This manual describes standard programs available from GE-PAC* 30, and how to use these programs. The manual is intended as a reference for programmers who are familiar with GE-PAC 30 Digital System, and as an introduction for programmers who have not used GE-PAC 30 Systems previously. Sections which describe system operation procedures, a typical programming sequence, and listings of which programs may be used with which equipment complement are provided before the actual program descriptions. Note that programs described in this manual are normally supplied with the manual in the form of punched paper tapes.

2. HEXADECIMAL NOTATION

GE-PAC 30 documentation uses hexadecimal notation extensively. The letter X denotes that the following alphanumeric characters, enclosed in single quote marks, form a hexadecimal number. Thus, X'50' indicates 50_{16}. Table 1 lists the decimal and binary equivalents for each valid hexadecimal character.

In some contexts, hexadecimal notation is used exclusively. For example, CLUB, the interactive debug program, uses hexadecimal numbers only. Also, a program listing as generated by the assembler, describes the binary form of the program in hexadecimal. In these cases, the X'---' notation is not used, and all numbers are assumed to be hexadecimal. In general, memory locations and program starting addresses are also defined in hexadecimal.

3. GLOSSARY OF TERMS

This section explains some terms and concepts used in GE-PAC 30 programs and program documentation. The terms are arranged in alphabetical order for easy reference.

50 Sequence — The 50 Sequence resides in core memory from X'50' to X'7F' and contains an 8-bit loader and a Device Definition Table. These 24 halfwords must be manually entered into memory. This area of core memory should be reserved for the 50 Sequence; once keyed into memory, this sequence normally remains there, available for use.

68 Sequence — The 68 Sequence (o GE-PAC 30-2 only) is a short form of the 50 Sequence. It makes use of the 30-2

TABLE 1. HEXADECIMAL NOTATION

Hex Character	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

* Registered Trademark of General Electric Company

Autoload instruction.

absolute — Programs designed to occupy a fixed set of locations in the core memory are called absolute programs. For example, an absolute program designed for bytes 80-99 in memory will not execute correctly if moved (relocated) to bytes 180-199 in memory.

assembler — The assembler program translates the source form of a program into a form which can be conveniently loaded into the system by a loader program. GE-PAC 30 provides an assembler program which converts assembly - language tapes into binary object tapes. See object and source.

bias — The base value used by the REL or General Loader to load a relocatable program is called the bias. The bias value is added to all relocatable quantities during the loading process. See General Loader.

bootstrap tapes — Certain program tapes are provided with the appropriate loaders on the tape itself. These tapes are loaded into memory using the 50 Sequence Loader or 68 Sequence Loader. All bootstrap tapes have a part number with an M10 designation. See 50 Sequence, 68 Sequence and Fast Format.

editor — An editor is a program which manipulates symbolic or textual information. It facilitates the creation, examination, and modification of character oriented data. Such a program is useful for the creation and editing of source tapes. See source.

Fast Format — Bootstrap tapes for large absolute programs, such as FORTRAN or the Assembler, employ a Fast Format for data organization. This format is essentially an 8-bit format which minimizes loading time on slow devices. Fixed length records are used, however, to facilitate checksum procedures. A transfer address is specified in the first record of a Fast Format tape. See bootstrap.

firmware — Micro-programs which are written for a Read-Only-Memory (ROM) are called firmware, as opposed to conventional machine language programs which are called software. See micro-program.

floating-point — A method of representing numbers with a mantissa or fraction and an exponent or characteristic. For example, in the number $.5 \times 10^3$, the .5 is the mantissa and the 3 is the exponent of the base 10. GE-PAC 30 systems represent floating-point numbers in a floating hexadecimal format using a 24 bit fraction and a 7 bit exponent of the base 16.

FORTRAN — The FORTRAN language permits the statement of arithmetic problems in an algebraic-type format. The GE-PAC 30 FORTRAN system is an interpreter which permits problems to be created and executed in an inter-active manner.

General Loader — The General Loader in the largest and most comprehensive of the GE-PAC 30 loaders. This loader handles absolute or relocatable programs with external program linkages and forward reference definitions, which occur on object tapes from one-pass assemblies. The loader bias and error messages are printed on the teletypewriter for operator convenience. See one-pass and bias.

listing — The assembler inputs a source tape and generates an object tape and a listing. The object tape contains the binary information to be loaded into memory. The listing is a printed record which shows each source statement, and the binary information generated for that statement. The binary information is represented in hexadecimal form.

loader — A loader is a type of program which, when executed by the machine, reads information from a peripheral device and loads the core memory with instructions and data.

micro-programs
GE-PAC 30 machines involve a Read-Only-Memory (ROM) used to control basic Processor operations. The sequence of commands which reside in the ROM is called a micro-program. See firmware.

object
Object tapes are binary tapes produced by the assembler. For each source tape assembled, there is an object tape. A loader reads the object tapes and places the corresponding instructions and data in core memory. See assembler, loader, and source.

one-pass
The assembler takes one, two, or three passes across the source tape to complete an assembly. The number of passes is controlled by an option control statement in the source program. When so directed, the assembler will make an assembly - complete with listing and object tape - in one pass. In this case, the resulting object tape must be loaded by the General Loader. See General Loader.

part number
Each program is identified by a part number which defines the type of program, the revision level, and the tape format. For example, the part number for the assembler is 03-001R01M10. In this number, the 03-001 identifies the assembler, the R01 indicates the revision level, and the M10 indicates that the program is available in bootstrap form. In general, an M08 designation means a relocatable program in standard binary object format. An M09 designation means an absolute program in standard binary object format. The M08 and M09 tapes require the REL or General Loader to be loaded into memory. Any bootstrap tape with designation M10 is loaded using the 8-bit loader at 50 or 68. Tapes with other designations may require special loading procedures.

program
A program is a set of machine instructions which, when executed by the machine, performs some useful function.

relocatable
Programs designed to be loaded anywhere in core memory are called relocatable. For example, a program which occupies 26 bytes could be loaded into X'80'-X'99' or X'180' - X'199' and executed from either location.

Relocating Loader
The REL Loader is appropriate for loading absolute or relocatable binary object tapes on which all data is defined. This loader is not appropriate for linking to external programs or for loading object tapes from one-pass assemblies. See relocatable and one-pass.

source
A source is a mnemonic or easy-to-read representation of a program. Assembly language or FORTRAN can be used to generate a source form of a program. The source is often prepared as a source paper tape, or source deck of punched cards. See object.

4. THE 50 AND 68 SEQUENCES

The 50 Sequence in the name of the basic 8-bit loader and Device Definition Table which resides in memory from X'50' to X'7F'. The 50 Sequence can be used on any GE-PAC 30 Processor. The 68 Sequence performs the same function as the 50 Sequence, but uses the 30-2 Autoload instruction, thus requiring less care space (X'68' to X'7F'). It has one other advantage: The 50 Sequence requires tapes to be placed in the reader exactly on the first character. The 68 Sequence bypasses leading blank tape, so does not have this requirement. One of these Sequences must be manually entered into memory when a Processor is first turned on. The Sequences serve two basic functions:

1. The 8-bit loader is used to pull bootstrap, or self-loading, tapes into memory.

2. The Device Definition Table is used to provide a limited degree of device independence by specifying which devices are to be used by standard programs.

Listings of the two sequences are shown in Table 2. Note that the 8-bit loader portion is standard for all memory sizes. The Device Table, from X'78' to X'7F' is changed according to the device configuration at hand. The sequences shown in

TABLE 2. 50, 68 SEQUENCE

```
                *
                *
                *    50 SEQUENCE LOADER
                *    FOR ALL GE-PAC 30 PROCESSORS
                *
0050                   ORG    X'50'
0050   C820     LOAD   LHI    2,X'80'         LOADS TAPE FROM X'80'
       0080
0054   C830            LHI    3,1             THRU X'CF'
       0001
0058   C840            LHI    4,X'CF'
       00CF
005C   D3A0            LB     10,BINDV        NOTE THAT LOCATION X'5A'
       0078
0060   DEA0            OC     10,BINDV+1      MUST BE CHANGED FOR ALL
       0079
0064   9DAE     SENSE  SSR    10,14           M14 TEST PROGRAM TAPES.
0066   083E            LHR    14,14
0068   4230            BTC    3,SENSE
       0064
006C   DBA2            RD     10;0(2)
       0000
0070   C120            BXLE   2,SENSE
       0064
0074   4300            B      X'80'
       0080
0078   0294     BINDV  DC     X'0294'         DEVICE DEFINITIONS ARE
007A   0298     BOUTDV DC     X'0298'         FOR TTY
007C   0294     SINDV  DC     X'0294'
007E   0298     SOUTDV DC     X'0298'
                *
                *
                *    68 SEQUENCE LOADER
                *    FOR 30-2 PROCESSORS ONLY
                *
0068                   ORG    X'68'
0068   C830            LHI    3,1             LOADS TAPE FROM X'80' THRU
       0001
006C   D3A0            LB     10,BINDV        X'CF'.  LOCATION X'72'
       0078
0070   D500            AL     0,X'CF'         MUST BE CHANGED FOR ALL
       00CF
0074   4300            B      X'80'           M14 TEST PROGRAM TAPES
       0080
                *    DEVICE DEFINITIONS ARE SAME AS FOR 50 SEQUENCE
                *
                *

                * HIGH SPEED PAPER TAPE READER= NN99 (BINDV,SINDV)
                * HIGH SPEED PAPER TAPE PUNCH= NN9A (BOUTDV)
                * CARD READER= NN20 (SINDV)
                *    WHERE NN= DEVICE NUMBER
                *
```

Table 2 are appropriate for teletypewriter input/output.

Note that these sequences are also used to load device test programs. When this happens, the following locations must be changed as indicated in the test program descriptions:

X'5A'	50 Sequence
X'72'	68 Sequence

The 8-bit loader stores 8-bit data bytes into memory from X'80' to X'CF' and transfers to X'80'. This loader, and all GE-PAC 30 loaders, use the binary input device as defined in X'78'. When using the 8-bit loader at X'50', three special steps are required:

1. The first data character on the tape must be placed over the read fingers or the photo diodes of the tape reader. Failure to observe this first-character restriction will cause the wrong data to be stored into memory.

2. When loading from a teletypewriter the tape motion must be started manually. After the loader is started at X'50', with an ASR-33 toggle the reader switch to START. With an ASR-35, put the reader switch in RUN with the Teletypewriter Mode Switch in the KT position.

When using the 8-bit loader at X'68', only step 2 is required.

The device Definition Table contains four halfwords. Each halfword specifies one device as follows:

0 7	8 15
Device No.	Output Cmnd.

The left byte contains the device number. The right byte contains the output command required to start that device. The four half-words are used as follows:

X'78'	BINDV	Binary Input	Used by loaders to select the load device
X'7A'	BOUTDV	Binary Output	Used by assembler to select the punch device.
X'7C'	SINDV	Source Inputs	Used by assembler to select the source input device.
X'7E'	LISTDV	Symbolic Outputs	Used by assembler to select the list device.

During loading operations, therefore, only BINDV at X'78' is used. During assembly operations, the other three halfwords are used. As other programs are created or revised by GE-PAC 30, they also will reference the appropriate halfword in the Device Table for device selection.

The range of devices that can be used may vary with different programs. One rule applies to all programs: device number 2 implies a teletypewriter. Special steps are taken, whenever device number 2 is specified, to handle the idiosyncrasies of a teletypewriter. Device Table entries for other devices are shown in Table 3. The device numbers shown are the ones normally assigned to these devices.

5. PROGRAM PREPARATION

The steps required to prepare a program are summarized in this Section. The first step in implementing a program is to write the program using assembly language statements as described in the Assembler section of this manual. Given a symbolic description of a program, the preparation process involves 10 steps as follows:

1. Enter the 50 or 68 Sequence into memory if necessary.
2. Load the REL Loader or General Loader.
3. Load the TIDE (Editor) program.
4. Use the TIDE to prepare program source tapes.
5. Load the appropriate Assembler program.
6. Assemble the source tapes.
7. Re-load the REL or General Loader.
8. Load the object tapes.
9. Load Hex Debug (CLUB).
10. Test the object program and punch new tape if desired.

These 10 steps are summarized in Figure 2. These steps are discussed in the following paragraphs.

Step	Comments
1	If the 50 Sequence is not already in core, it must be manually entered as follows:

Program	Function	Table Location	Device	Table Entry
Loaders	Loading	BINDV at X'78'	Teletypewriter High Speed Tape Reader	0294 0399
Assembler	Punching Object	BOUTDV at X'7A' H	Teletypewriter High Speed Tape Punch	0298 139A
Assembler	Reading Source	SINDV at X'7C'	Teletypewriter High Speed Tape Reader Card Reader	0294 0399 0420
Assembler	Listing	LISTDV at X'7E'	Teletypewriter High Speed Tape Punch	0298 139A

TABLE 3. DEVICE DEFINITION TABLE ENTRIES

Step	Comments
	A. Set the Data/Address switches to X'50', set the MODE CONTROL to ADRS, and depress EXECUTE.
	B. Set the MODE CONTROL to MEMW.
	C. Set the Data/Address switches to X'C820', and depress EXECUTE. This enters the first halfword into memory.
	D. Set the Data/Address switches to X'0080', and depress EXECUTE. This enters the second halfword into memory.
	E. Continue this process until all 24 halfwords of the 50 Sequence, as described in Section 5, have been entered into memory.
2	In order to get the TIDE (Editor) Program into memory, the REL Loader, Part Number 06-024M10, is required. The loader is entered into memory as follows:
	A. Put the proper loader tape in the tape reader, observing the first character allignment.
	B. Set the Data/Address switches to X'50' or X'68', set the MODE CONTROL to ADRS, and depress EXECUTE.
	C. Depress INITIALIZE.

Step	Comments
	D. Set the MODE CONTROL to RUN, and depress EXECUTE.
	E. If a teletypewriter is in use, start the tape moving by toggling the reader switch to Start or Run.
	F. If errors are detected during the load, the tape will stop. In this case, reposition the tape to the previous record gap and depress EXECUTE. Refer to the Loader Descriptions for details on error recovery procedures.
	G. The load is complete when the tape has been read to the end, and the Processor halts with the EXECUTE light on.
3	Now that the loader is in memory, the TIDE program can be loaded. This is done as follows:
	A. Put the TIDE tape in the tape reader, placing the read fingers under any portion of the blank loader.
	B. Depress EXECUTE. This should start the tape moving.
	C. If errors are detected during the load, the tape will stop. In this case, reposition the tape to the previous record gap and depress EXECUTE.

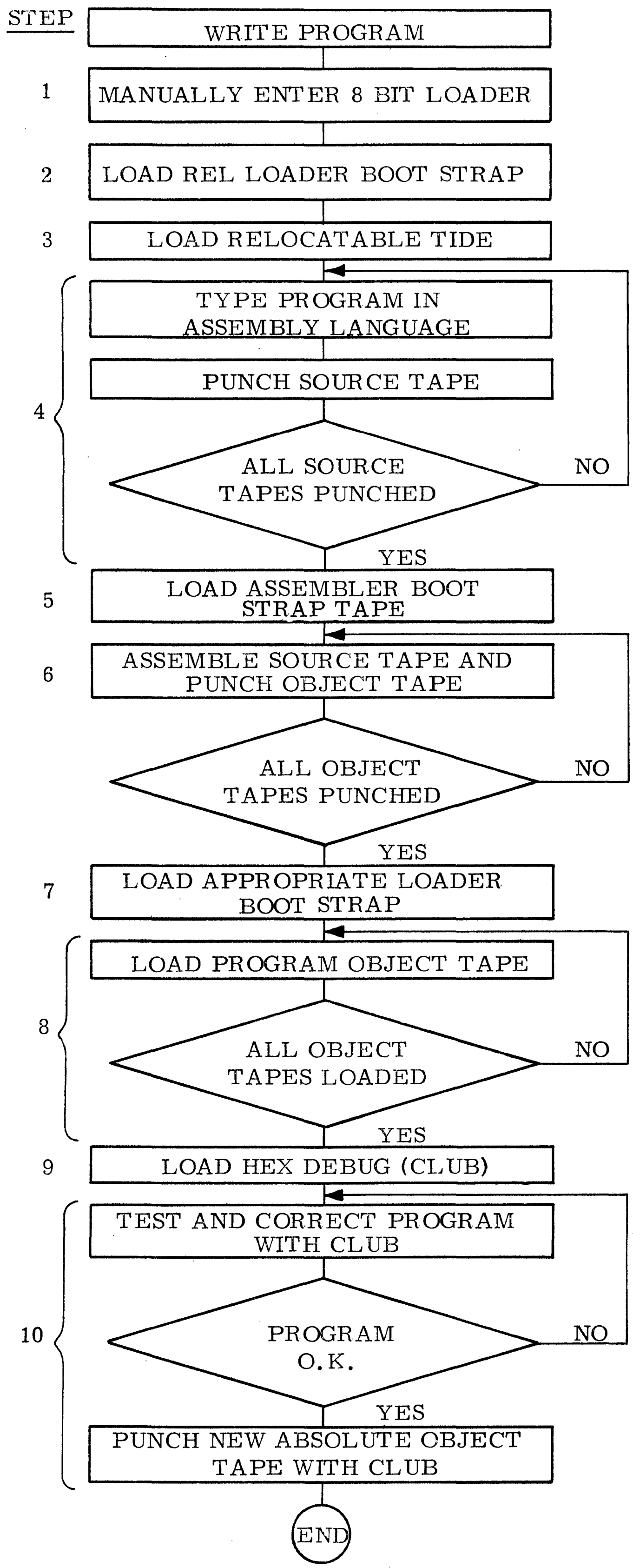


Fig. 2 Program Preparation Sequence

Step	Comments
	D. The load is complete when the tape has been read to the end, and the Processor halts with the EXECUTE light illuminated.
4	Now that TIDE is in memory, the source tapes can be prepared. For details on the use of TIDE refer to the Editor section of this manual.
5	Given the source tapes, as prepared by TIDE, the next step is to assemble the tapes. For this step, the TAPE Assembler, Part Number 03-001M10, must be loaded into memory as follows:

A. Put the assembler tape into the tape reader, observing the first character allignment.

B. Set the Data/Address switches to X'50' or X'68'. Set the MODE CONTROL to ADRS, and depress EXECUTE.

C. Depress INITIALIZE.

D. Set the MODE CONTROL to RUN, and depress EXECUTE.

E. If a teletypewriter is in use, manually start the tape moving.

F. If errors are detected, the tape will stop. In this case, reposition the tape to the previous record gap and depress EXECUTE.

G. The load in complete when the tape has been read to the end, and the Processor halts with the EXECUTE light illuminated.

Step	Comments
6	The use of the Assembler is described in the Assembler Section of this manual.
7	For each source tape assembled, the Assembler generates an object tape. These object tapes must now be loaded into memory. The object tapes may be absolute or relocatable, depending on how the program was written. If the programs involve ENTRY's or EXTRN's, the General Loader, Part Number 06-025M10, is required. Object tapes from 1-pass assemblies also require the General Loader. If the object tapes are relocatable or absolute either the REL Loader or the General Loader can be used.

Step	Comments
	The version of Hex Debug to be used also affects the choice of loader. See Step 9. The proper loader should be entered into memory, using the same procedure as described under Step 2.
8	With the loader in memory, the object tapes can be loaded. The loading process is described in detail in the Loader Descriptions section of this manual.
9	If it is necessary to test or debug the programs just loaded, the Hex Debug Program (CLUB) may be useful. Three versions of CLUB are available. They are:

A. Relocatable with output, 03-002M08.

B. Relocatable without output, 03-003M08.

C. Relocatable with output and disassembly, 03-M08.

The output feature in CLUB enables a portion of memory to be punched as an absolute binary object tape. If it is not necessary to punch a new tape, the CLUB without output can be used since it requires less memory space. Note that CLUB loaded at X'80' occupies memory from X'80' to X'5B8' without output, from X'80' to X'7D2' with output, and from X'80' to X'A4E' with output and disassembly. If other programs need memory in this area, relocatable CLUB can be loaded into some other area of core memory. The relocatable CLUB tapes require the REL or General Loader for loading. The use of CLUB should be considered in Step 7 when selecting the loader to be used.

Step	Comments
10	The use of CLUB is described in the Debug section of this manual.

6. PROGRAMMING CONVENTIONS

This Section summarizes some of the conventions used in programs and documentation supplied by GE-PAC 30.

Convention	Comments
hex notation	Hexadecimal notation is used exentsively. This notation is explained in Section 3.
ASCII codes	Standard programs supplied by GE-PAC 30 represent characters in ASCII code. This character code is defined in the GE-PAC 30 Reference Manual (PCP-125A). The ASCII code represents each character with 8-bits in which the high-order bit is available for parity. The parity bit may change according to the input/output devices in use. Internal to the Processor, most programs mask the high order bit to zero, and handle the character in terms of the 7-bit codes. The assembler generates the 7-bit form of ASCII codes for characters.
low core	The general registers in a 30-01 Processor reside in core memory. Refer to the GE-PAC 30 Reference Manual for details. GE-PAC 30 programs, as a rule, never refer to general registers by their absolute address in core memory. This rule is essential for program capability between the GE-PAC 30-01 and 30-02 Processors.
tape formats	GE-PAC 30 programs are normally supplied as binary object tapes in one of four formats. These formats, and the corresponding part number designations are:

M08 - Relocatable
M09 - Absolute
M10 - Bootstrap
M14 - 8-Bit Binary

The M08 and M09 tapes are never generated from 1 - pass assemblies, and require the General Loader only when ENTRY'S or EXTRN'S are involved. The Math Library tapes do require the General Loader since they use ENTRY'S. Other M08 and M09 tapes are loadable by the ABS or REL Loaders. The M10 and M14 tapes are loaded by the 8-bit loader at X'50'.

Convention	Comments
device number 2	Some GE-PAC 30 programs, such as CLUB and the test programs assume that device number 2 is a teletypewriter. This device is used for keyboard inputs and message printouts. The ABS, REL, and General Loader take special steps when device number 2 is specified in the Binary Device Definition at X'78'. These special steps involve XON and XOFF characters which control tape motion.

GE-PAC 30-01

CORE MEMORY ALLOCATION FOR GENERAL REGISTERS AND PROGRAM STATUS WORDS

Hexadecimal Memory Address	Register Assignment
General Registers	
00 - 01	R0
02 - 03	R1
04 - 05	R2
06 - 07	R3
08 - 09	R4
0A - 0B	R5
0C - 0D	R6
0E - 0F	R7
10 - 11	R8
12 - 13	R9
14 - 15	R10
16 - 17	R11
18 - 19	R12
1A - 1B	R13
1C - 1D	R14
1E - 1F	R15
Micro-Processor Registers	
20 - 21	Instruction Register
22 - 23	Instruction Address Register
24 - 25	Current PSW: Status and Condition Code
26 - 27	Current PSW: Instruction Address Counter
28 - 2F	Reserved for Micro-Processor
Program Status Words	
30 - 33	Old PSW: Illegal Instruction Interrupt
34 - 37	New PSW: Illegal Instruction Interrupt
38 - 3B	Old PSW: Machine Malfunction Interrupt
3C - 3F	New PSW: Machine Malfunction Interrupt
40 - 43	Old PSW: External Device Interrupt
44 - 47	New PSW: External Device Interrupt
48 - 4B	Old PSW: Divide Fault Interrupt
4C - 4F	New PSW: Divide Fault Interrupt
50	First user available memory location

CORE MEMORY ALLOCATION FOR REGISTERS AND PROGRAM STATUS WORDS

Hexadecimal Memory Address	Register Assignment
Floating-Point Registers	
00 - 03	R0
04 - 07	R2
08 - 0B	R4
0C - 0F	R6
10 - 13	R8
14 - 17	R10
18 - 1B	R12
1C - 1F	R14
Micro-Processor Registers	
20 - 21	High Speed Interrupt Pointer
22 - 23	Register Save Pointer
24 - 25	Current PSW: Status and Condition Code
26 - 27	Current PSW: Location Counter
Program Status Words	
28 - 2B	Old PSW Flp Divide Fault Interrupt
2C - 2F	New PSW Flp Divide Fault Interrupt
30 - 33	Old PSW Illegal Instruction Interrupt
34 - 37	New PSW Illegal Instruction Interrupt
38 - 3B	Old PSW Machine Malfunction Interrupt
3C - 3F	New PSW Machine Malfunction Interrupt
40 - 43	Old PSW External Device Interrupt
44 - 47	New PSW External Device Interrupt
48 - 4B	Old PSW Fix Divide Fault Interrupt
4C - 4F	New PSW Fix Divide Fault Interrupt
50	First User Available Memory Location

PSW STATUS FIELD ASSIGNMENTS

Bit Set	Meaning
0	Wait State
1	External Device Interrupt
2	Machine Malfunction Interrupt
3	Fixed-Point Divide Fault Interrupt
4	Reserved
5	Floating-Point Divide Fault Interrupt
6 through 11	Unassigned

TABLE OF CONTENTS

TELETYPEWRITER OPERATION AND PROGRAMMING MANUAL

1. DEVICE DESCRIPTION

Model Numbers	- ASR-33 and ASR-35
Data Rate	- 10 characters per sec.
Printer Width	- 72 characters max.
Character Set	- see Appendix 2
Paper Feed	- pin feed

Note that this description applies to teletypewriter with Teletypewriter Controller, Part Number 32-062. Figure 1 shows the teletypewriter keyboard layout.

2. POWER CONTROL

A three position power switch is located to the right and below the keyboard. When rotated left to the position marked LINE, power is applied to the teletypewriter and the device is logically corrected to the Processor. When rotated to the right to the position marked LOCAL, the unit is powered, but disconnected from the Processor.

3. STATUS AND COMMANDS

Table 1 illustrates the teletypewriter status and command byte coding.

A Sense Status Instruction (SS or SSR) is used to transfer the status byte from the device controller to the Processor. The least significant four bits (4 - 7) of the status byte are copied into the condition code during the Sense Status operation. Branch instructions can test these four bits directly.

Note that the status byte from teletypewriter with the Controller, Part Number 32-004, involved a DRR bit in bit position 3. Programming appropriate to one teletypewriter controller may not be appropriate to another. For example, during READ operations, it is necessary to test the DRR and BSY bits in one case, while the BSY bit is sufficient in the other case. In general, testing the status byte for all bits zero during READ operations is compatible with all teletypewriter controllers.

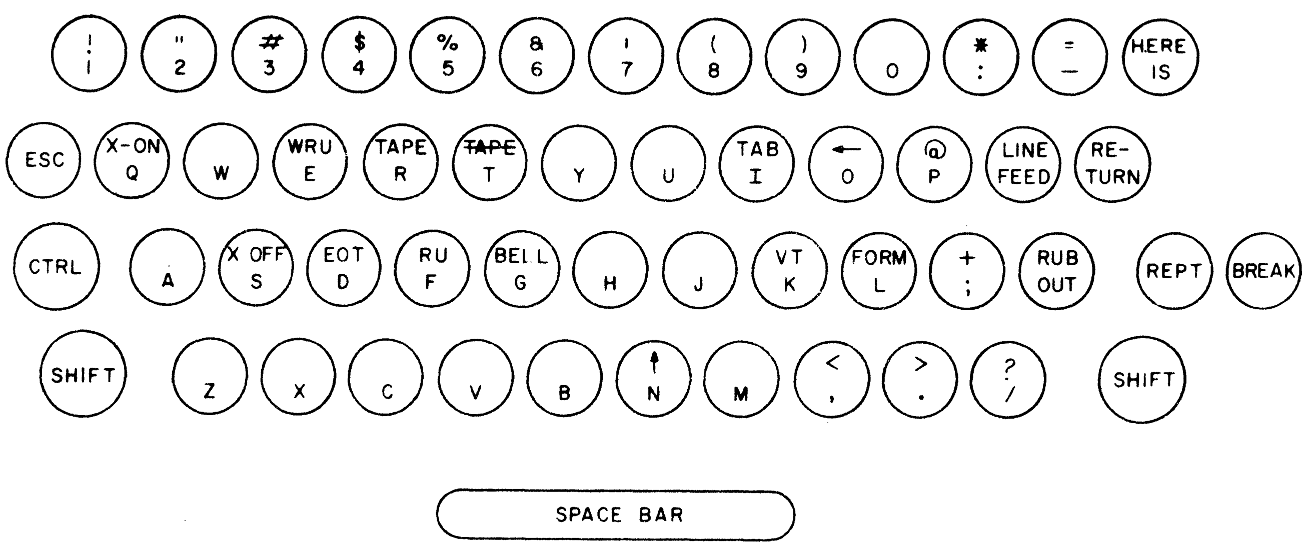


Figure 1. Teletypewriter Keyboard Layout

DISPLAY PANEL PROGRAMMING MANUAL

1. INTRODUCTION

This document pertains to the General Purpose Display Panel and Display Panel Controller, Part Number 32-061. The General Purpose Display Panel facilitates console operation and operator interaction with the machine. The console operating procedures are discussed in Chapter 3 of the GE-PAC 30 Systems Reference Manual, Publication Number 29-004.

In addition to its role as a console control panel, the General Purpose Display Panel can be programmed like a typical peripheral device. This discussion describes the programming aspects of the device.

2. OPERATOR CONTROLS

The Display Panel includes six distinct elements:

1. Control Switches: POWER, INITIALIZE, and EXECUTE.
2. MODE CONTROL rotary switch.
3. SPEED CONTROL rotary switch.
4. REGISTER DISPLAY rotary switch.
5. Data/Address switches.
6. Two 16-bit halfword display registers.

With normal input-output instructions, a program can sense the state of the MODE CONTROL rotary switch and the REGISTER DISPLAY rotary switch, read the Data/Address switches, and output data to the display registers.

3. DEVICE NUMBER

The Device Number of the Display Panel is 1. Unlike most peripheral device controllers, this Device Number is hard-wired and cannot be changed.

4. STATUS FORMAT

The status of the Display Panel can be determined with an SS or SSR instruction. The meaning of each bit in the 8-bit status byte is shown in Figure 2.

5. MODE COMMAND

Within the Display Panel Interface, Part Number 32-061, are two counters which control the data transfer to and from the Display Panel. The output counter, which is 2 bits in length, determines which byte of the Register Display is the destination for data bytes transferred to the display. The input counter, which is only 1 bit, controls which half of the Data/Address switches is used when data is read from the display. The order of bytes transferred to or from the display registers and switches is shown in Figure 1. The appropriate counter is incremented following each transfer to or from the Display Panel.

There are two modes associated with the Display Panel: the Normal Mode and the Incremental Mode. In the Normal Mode, the control counters in the interface are cleared (set to zero) every time the Display Panel (Device Number 1) is addressed. In this mode, only byte 0 of the registers or switches is accessible by program since each Write Data (WD or WDR) or Read Data (RD or RDR) instruction addresses the specified device every time the instruction is executed. The re-addressing of Device Number 1 keeps the control counters at zero in this mode.

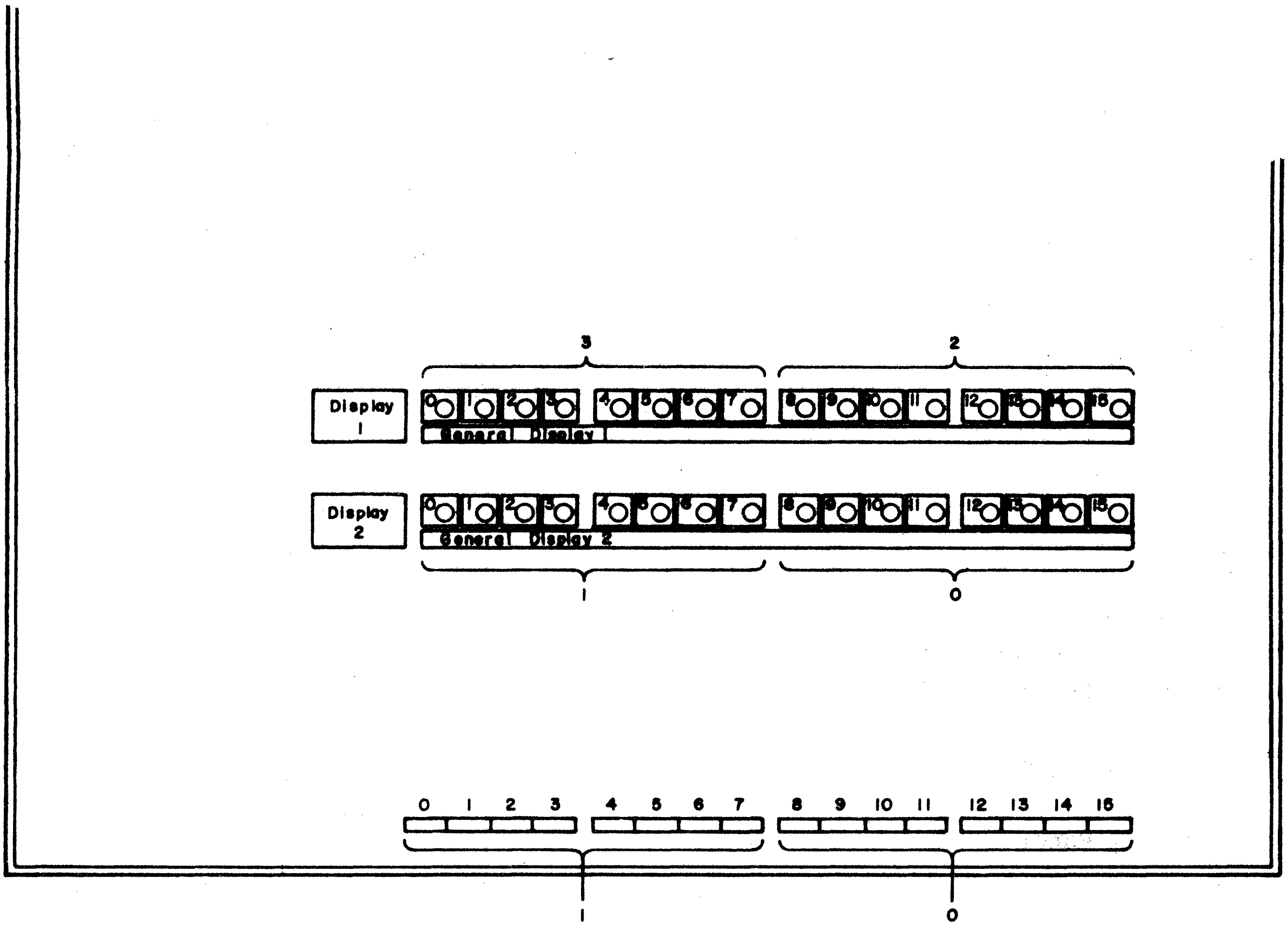


FIGURE 1. ORDER OF BYTE TRANSFER

In the Incremental Mode, the control counters are cleared when the mode is selected, but not cleared when the device is addressed. This mode allows subsequent Read Data or Write Data instructions to increment the Control counters, and access subsequent bytes of the registers or switches.

The Output Command (OC or OCR) instruction is used to control the mode. The command byte is as follows:

0	1	2	3	4	5	6	7
Norm	Incr						

Norm = Select Normal Mode
Incr = Select Incremental Mode

The remaining bits of the command byte are not used. The control counters are cleared following a command to select the Incremental Mode.

6. DATA INPUT

One byte of the Data/Address switches, as specified by the input control counter, is read when an RD or RDR instruction is executed for Device Number 1. A bit of the data byte is a 1 if the corresponding switch is depressed.

0	1	2	3	4	5	6	7
MODE				REG			

MODE CONTROL Switch

Model 3	Model 4				
VARI	VARI FIX	0	1	0	0
	VARI FLT	0	1	1	0
HALT	HALT FIX	1	1	0	0
	HALT FLT	1	1	1	0
RUN	RUN	1	0	0	0
ADRS	ADRS	0	0	1	1
MEMR	MEMR	0	0	1	0
MEMW	MEMW	0	0	0	1

REGISTER DISPLAY Switch

0	0	0	0	OFF
0	0	0	1	Register Display
0	0	1	0	INST
0	1	0	0	PSW
1	0	0	0	R0/1
1	0	0	1	R2/3
1	0	1	0	R4/5
1	0	1	1	R6/7
1	1	0	0	R8/9
1	1	0	1	R10/11
1	1	1	0	R12/13
1	1	1	1	R14/15

Figure 2. Status Byte

7. DATA OUTPUT

One 8-bit data byte is transferred to that section of the display registers, as specified by the output control counter, when a WD or WDR instruction is executed to Device Number 1. The indicator in that register section is lit if the corresponding bit of the data byte is a 1. The remaining sections of the display registers are unchanged.

8. INTERRUPTS

There are no interrupts associated with the Display Panel.

9. INITIALIZATION

Depressing the INITIALIZE button on the Display Panel puts the Display Panel interface in the Normal Mode.

NOTE

After data transfers to or from the Display Panel in the Incremental Mode, the program should return the Display Panel to the Normal Mode. The micro-program which supports the Display Panel in most GE-PAC 30 Processors assumes that the Display Panel is in the Normal Mode.

TABLE 1
TELETYPEWRITER STATUS AND COMMAND BYTE DATA
HEX ADDRESS 02

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE			BRK		BSY	EX		DU
COMMAND BYTE	DISABLE	ENABLE	UNBLOCK	BLOCK	WRT	READ	PWR ON	PWR OFF

BRK — The Break bit is set when the Break key on the Teletypewriter is depressed, or the Teletypewriter is logically disconnected from the Processor.

BSY — The significance of the Busy bit depends upon whether a Read or a Write operation is in progress. During Write mode, BSY is normally low, and goes high only while data is being received by the device. During Read mode, BSY is normally high, and goes low only when data has been received from the device, but not yet been transferred to the Processor. During Read mode, BSY goes high again as soon as the Processor accepts the data.

EX — The Examine bit is set whenever BRK is set.

DU — The Device Unavailable bit is set whenever the Teletypewriter power is off, the Teletypewriter is in LOCAL mode, or power is not connected to the Teletypewriter.

DISABLE — This command disables the Device Interrupt to the Processor from the Device Controller.

ENABLE — This command enables the Device Interrupt to the Processor from the Device Controller.

UNBLOCK — This command enables the printer to print data entered via either the keyboard or the tape reader.

BLOCK — This command disables the feature described above.

WRT
READ — The Write and Read commands are used to define the significance of the BSY bit.

PWR ON
PWR OFF — The Power On and Power Off commands are significant only with those Teletypewriters provided with an optional Power Control Box. The option permits the Teletypewriter power to be enabled or disabled under program control.

4. DEVICE NUMBER

Teletypewriters are normally assigned Device Number 2. The device number can be changed, or additional teletypewriters added, as needed by a minor change to the teletypewriter controller. Refer to the Maintenance Manual for details.

5. INTERRUPTS

The interrupt associated with the teletypewriter is a data-ready, or ready-to-transfer interrupt. That is, when enabled, an interrupt is generated by the teletypewriter controller whenever it is ready to execute a data transfer with the Processor. In the WRITE mode, an interrupt occurs when the controller is ready for another character to send to the teletypewriter. In the READ mode, an interrupt occurs when the controller has assembled a character for transfer to the Processor.

Note that when changing from the READ mode to the WRITE mode with interrupts enabled, an interrupt occurs as soon as the controller is ready to receive the first character for output.

6. INITIALIZATION

When the INITIALIZE button on the Processor Display Panel is depressed, the following occurs:

1. The DSBL, BLK, and READ command functions are set.
2. The BSY status bit is set.
3. The BRK, EX, and DU status bits are reset.
4. Any pending interrupts are cleared.

7. ASR-35 FEATURES

The ASR-35 is a ruggedized or heavy-duty version of the ASR-33. The programming of an ASR-35 is identical to an ASR-33. The operation of an ASR-35 is similar to an ASR-33 with the following exceptions:

1. The Tape Reader and Tape Punch controls are different as explained later in this description.
2. The ASR-35 has a mode control switch to the left of the keyboard. The meaning of the 5 positions of this switch is illustrated in Appendix 3.
3. Several additional keys, such as Local Line Feed, are provided. The meaning of these keys is self-explanatory.

8. PAPER TAPE READER

The tape reader is controlled by a three-position switch on the reader. The three positions are START, STOP, and FREE. When the switch is moved to the START position, any tape in the reader is advanced continuously at a 10 character per second rate. Tape motion continues until the reader switch is moved to the STOP position.

In the STOP position, tape motion can be controlled by program, assuming the teletypewriter power switch is in the LINE position. The specific control characters which affect the reader are X-ON (X'91') which starts the tape motion, and X-OFF (X'93') which stops tape motion.

The programmed steps required to start the reader are as follows:

```
        OC    DEV, WRITE   Set Write Mode
WAIT1   SS    DEV,STATUS   Wait for BSY = 0
        BTC   BSY,WAIT1
        WD    DEV,XON      Start Tape
        OC    DEV,READ     Set Read Mode
WAIT2   SS    DEV,STATUS   Wait for BSY = 0
        BTC   BSY,WAIT2
        RD    DEV,DATA     Read a Char
```

The programmed steps required to stop the reader are as follows:

```
        OC    DEV,WRITE    Set Write Mode
WAIT3   SS    DEV,STATUS   Wait for BSY = 0
        BTC   BSY,WAIT3
        WD    DEV,XOFF     Stop the Tape
```

Note that when stopping the tape reader under program control, the tape may advance 1 or 2 characters between the time the XOFF character is issued and the tape comes to a complete halt. Similarly on starting a tape, 1 or 2 characters may be missed before synchronization is attained. Therefore, tapes to be read under program control should be formatted to account for the start/stop characteristics of the reader.

The above procedures apply to both the ASR-33 and the ASR-35 teletypewriters. The ASR-35 tape reader is enabled, however, only when the ASR-35 Mode Switch is in the KT, T, or TTS positions. See Appendix 3 for details.

9. PAPER TAPE PUNCH

The ASR-33 tape punch can be manually turned on at any time by depressing the LOCK "ON" button on the punch unit. Once turned on in this fashion, all data output to the teletypewriter is unconditionally printed and punched. Note that non-printing characters transferred to the teletypewriter will be punched, but no image will be printed, and the printer carriage will not advance. To manually turn the punch off on an ASR-33, the following steps are required:

1. Turn the power switch to LOCAL mode.

2. Depress the UNLOCK key on the tape punch.

3. Strike the ~~TAPE~~ key while the CTRL key is depressed.

If the ASR-33 is not in a LOCK "ON" mode (depress UNLOCK to release the LOCK "ON" mode), and the power switch is in the LINE position, the tape punch can be controlled via program.

The specific control characters which affect the punch are TAPE ON (X'92'), which starts the punch, and TAPE OFF(X'94'), which stops the punch. The punch controls are achieved by outputting the appropriate character to the teletypewriter. Note that the TAPE OFF character will get punched on the tape.

The tape punch can be manually started in an alternate way. If the punch is not already on, strike the TAPE key with the CTRL key depressed, and the power switch in LOCAL mode. This technique is equivalent to transferring a TAPE ON character to the teletype from the Processor.

The ASR-35 tape punch is enabled only when the ASR-35 Mode Switch is in the KT or TTR position. In these modes, the punch is controlled via TAPE and ~~TAPE~~ keys, and TAPE ON and TAPE OFF characters as described above. Refer to Appendix 3 for details. Following the program transfer of a TAPE ON character to start the ASR-35 tape punch, the program should output 2 or 3 rubout characters (X'FF') to achieve data synchronization prior to punching the data.

10. DATA FORMATS

The format of data transferred to and from the teletypewriter is as follows:

1. When reading data from a tape, each 8-bit tape character is transferred to the Processor as one 8-bit data byte.

2. When reading data from the keyboard, one data byte is transferred for each key depressed. The data is the ASCII code for the particular character, in which the most significant bit for the character is a one or zero to achieve even parity for that character. In general, programs which read teletypewriter data mask the most significant bit to zero, and manipulate 7-bit ASCII codes in memory.

3. When transferring characters from the Processor to the teletypewriter printer, the most significant bit of each character can be either one or zero since it has no effect on the teletypewriter.

4. When transferring characters from the Processor to the teletypewriter punch, each 8-bit data byte is punched as one tape character. The most significant bit is punched as specified in the data byte.

11. PROGRAM EXAMPLES

Typical routines for transferring data to and from the teletypewriter are shown in Appendix 1.

APPENDIX 1
SAMPLE PROGRAM LISTINGS

```
                              OPT   PASS2,PRINT,NOPNCH,STOP
                        *INPUT
                        *
                        *
                        *A BYTE WILL BE INPUT TO R4 FROM THE TELETYPEWRITER
                        *REGISTERS R3,R4,R15 WILL BE USED
                        *THE CALL IS BAL R15,INPUT
                        *
                        *
 0000R                          ENTRY INPUT
 0000R C830             INPUT   LHI   R3,DEVNO            LOAD DEVICE NUMBER
       0002
 0004R DE30                     OC    R3,UNBLOK           SET DEVICE MODE
       0012R
 0008R 9D34             SENS    SSR   R3,R4               INPUT STATUS
 000AR 42F0                     BTC   X'F',SENS           LOOP IF NOT READY
       0008R
 000ER 9B34                     RDR   R3,R4               INPUT BYTE
 0010R 030F                     BR    R15                 RETURN TO CALL
 0003                   R3      EQU   3
 0004                   R4      EQU   4
 000F                   R15     EQU   15
 0012R A400             UNBLOK  DC    X'A400'             DISABLE,UNBLOK,READ
 0002                   DEVNO   EQU   2
 0014R                          END

 DEVNO  0002
*INPUT  0000R
 R15    000F
 R3     0003
 R4     0004
 SENS   0008R
 UNBLOK 0012R
```

```
                          OPT    PASS2,PRINT,NOPNCH,STOP
                  *OUTPUT
                  *
                  *
                  *A BYTE WILL BE OUTPUT FROM R4 TO THE TELETYPEWRITER
                  *REGISTERS R3,R4,R5,R15 WILL BE USED
                  *THE CALL IS BAL R15,OUTPUT
                  *
                  *
 0000R                    ENTRY  OUTPUT
 0000R C830       OUTPUT  LHI    R3,DEVNO            LOAD DEVICE NUMBER
       0002
 0004R DE30               OC     R3,BLOCK            SET DEVICE MODE
       0012R
 0008R 9D35       SENS    SSR    R3,R5               INPUT STATUS
 000AR 42F0               BTC    X'F',SENS           LOOP IF NOT PDY
       0008R
 000ER 9A34               WDR    R3,R4               OUTPUT BYTE
 0010R 030F               BR     R15                 RETURN TO CALL
 0003             R3      EQU    3
 0004             R4      EQU    4
 0005             R5      EQU    5
 000F             R15     EQU    15
 0002             DEVNO   EQU    2
 0012R 9800       BLOCK   DC     X'9800'             DISABLE,BLOCK,WRITE
 0014R                    END

  BLOCK    0012R
  DEVNO    0002
* OUTPUT   0000R
  R15      000F
  R3       0003
  R4       0004
  R5       0005
  SENS     0008R
```

```
                        OPT   PASS2,PRINT,NOPNCH,STOP
                *TTY OUTPUT EXAMPLE DISABLE,BLOCK,WRITE
                *TYPOUT
                *
                *
                *A SERIES OF BYTES WILL BE OUTPUT
                *AS DETERMINED BY THE CALLING SEQUENCE
                *
                *REGISTERS R3,R4,R5,R6,R15,WILL BE USED
                *THE CALLING SEQUENCE IS
                * BAL R15,TYPOUT
                * DC A(MESS) STARTING ADDRESS
                * DC A(END) ENDING ADDRESS+1
                *
                *
ØØØØR                   ENTRY TYPOUT
ØØØØR 486F      TYPOUT  LH    R6,Ø(R15)        GET STARTING ADRS
      ØØØØ
ØØØ4R 485F              LH    R5,2(R15)        GET ENDING ADRS
      ØØØ2
ØØØ8R C83Ø              LHI   R3,DEVNO         LOAD DEVICE NUMBER
      ØØØ2
ØØØCR DE3Ø              OC    R3,BLOCK         SET MODE
      ØØ28R
ØØ1ØR 9D34      SENS    SSR   R3,R4            INPUT STATUS
ØØ12R 42FØ              BTC   X'F',SENS        TEST STATUS
      ØØ1ØR
ØØ16R DA36              WD    R3,Ø(R6)         OUTPUT DATA
      ØØØØ
ØØ1AR CA6Ø              AHI   R6,1             INCREMENT ADRS
      ØØØ1
ØØ1ER Ø565              CLHR  R6,R5            TEST FOR END
ØØ2ØR 428Ø              BTC   X'8',SENS        LOOP IF NOT
      ØØ1ØR
ØØ24R 43ØF              BFC   Ø,4(R15)         RETURN TO CALL
      ØØØ4
ØØØ3            R3      EQU   3
ØØØ4            R4      EQU   4
ØØØ5            R5      EQU   5
ØØØ6            R6      EQU   6
ØØØF            R15     EQU   15
ØØØ2            DEVNO   EQU   2
ØØ28R 98ØØ      BLOCK   DC    X'98ØØ'
ØØ2AR                   END

 BLOCK    ØØ28R
 DEVNO    ØØØ2
 R15      ØØØF
 R3       ØØØ3
 R4       ØØØ4
 R5       ØØØ5
 R6       ØØØ6
 SENS     ØØ1ØR
*TYPOUT   ØØØØR
```

APPENDIX 2. TELETYPEWRITER/ASCII/HEX CONVERSION TABLE

HEX (MSD) →					8	9	A	B	C	D	E	F
(LSD) ↓	Teletype-writer Tape Channels →			8	Depends upon parity							
				7	0	0	0	0	1	1	1	1
				6	0	0	1	1	0	0	1	1
				5	0	1	0	1	0	1	0	1
	4	3	2	1								
Ø	0	0	0	0	NULL	DC_0	SPACE	0	@	P		
1	0	0	0	1	SUM	X-ON	!	1	A	Q		
2	0	0	1	0	EOA	TAPE ON	"	2	B	R		
3	0	0	1	1	EOM	X-OFF	#	3	C	S		
4	0	1	0	0	EOT	TAPE OFF	$	4	D	T		
5	0	1	0	1	WRU	ERR	%	5	E	U		
6	0	1	1	0	RU	SYNC	&	6	F	V		
7	0	1	1	1	BELL	LEM	'	7	G	W		
8	1	0	0	0	FE_0	S_0	(	8	H	X		
9	1	0	0	1	HT/SK	S_1	)	9	I	Y		
A	1	0	1	0	LF	S_2	*	:	J	Z		
B	1	0	1	1	VT	S_3	+	;	K	[		
C	1	1	0	0	FF	S_4	,	<	L	\		ACK
D	1	1	0	1	CR	S_5	-		M	]		ALT. MODE
E	1	1	1	0	SO	S_6	.	>	N	↑		ESC
F	1	1	1	1	SI	S_7	/	?	O	←		DEL

APPENDIX 3. 35 ASR OPERATING MODES

MODE	LINE	LOCAL
K (Keyboard)	KB Printer Computer	KB Printer
KT (Keyboard Tape)	Reader KB Printer Punch Computer	Reader KB Printer Punch
T (Tape)	Reader Printer Computer KB Punch Off Line	Reader Printer KB Punch
TTs (Tape to Tape Send) Non ASCII Tapes	Reader Computer KB Punch Off Line Printer Not Used	Not Applicable
TTr (Tape to Tape Rev) Non ASCII Tapes	Punch Computer Reader KB Printer Not Used	Not Applicable

```
                    OPT   PASS2,PRINT,NOPNCH,STOP
                *
                * THIS SUBROUTINE STARTS THE TAPE MOVING AND READS
                *
                * UNTIL CHARACTER X'FE' IS DETECTED
                *
                * CALL BAL R15,SEARCH
                *
0000R C830      SEARCH  LHI   R3,3              LOAD DEVICE NUMBER
      0003
0004R DE30              OC    R3,RUN            START READER
      001ER
0008R 9D34      SENSE   SSR   R3,R4             OBTAIN STATUS
000AR 42F0              BTC   X'F',SENSE        LOOP UNTIL CHAR AVAILABL
      0008R
000ER 9B34              RDR   R3,R4             INPUT CHARACTER
0010R C540              CLHI  R4,X'00FE'        TEST FOR STOP CHARACTER
      00FE
0014R 4230              BNE   SENSE
      0008R
0018R DE30              OC    R3,STOP           STOP READER
      001FR
001CR 030F              BR    R15               RETURN TO CALLING ROUTIN
0003            R3      EQU   3
0004            R4      EQU   4
000F            R15     EQU   15
001ER 9520      RUN     DC    X'9520'           DISABLE,RUN,SLEW,FWD
001FR           STOP    EQU   RUN+1             STOP
                *
                *
                *
                *
                *
                *
                *
                *
                *                               APPENDIX A  PAGE 1
0020R                   END

R15    000F
R3     0003
R4     0004
RUN    001E
SEARCH0000
SENSE  0008
STOP   001F
```

HIGH SPEED PAPER TAPE READER/PUNCH OPERATION AND PROGRAMMING

Publication Number 29-016R01

1. INTRODUCTION

This manual provides information on the operation and programming of the High Speed Paper Tape Reader, the High Speed Paper Tape Punch, and the Combination High Speed Reader/Punch. Included in this document are a general description, a table of status and command bytes, and sample programs for each device.

Note, that with the Combination Reader/Punch, since there is only one device controller, the devices cannot be used simultaneously. To read and punch tapes at the same time, it is necessary to use separate device controllers for each device.

2. GENERAL DESCRIPTION

Table 1 lists general characteristics of the Reader and Punch.

TABLE 1. READER AND PUNCH CHARACTERISTICS

Characteristics	Reader	Punch
Type	photo-electric	electro-mechanical
Tape Width	adjustable tape guides for $\frac{1}{2}$", 11/16", and 1" tape	fixed width of 1"
Speed	maximum of 300 characters-per-second	maximum of 63.3 characters-per-second
Tape handling	paper, paper-mylar, and mylar	same as the Reader
Stop time	capable of stopping on a character (approximately 1 millisecond)	will punch the next character and stop
Read/Load Lever	allows loading or changing of tapes of varying widths	does not apply
Power Switch	applies AC power to Reader motor	applies AC power to Punch motor
Remote Switch	does not apply	puts the Punch on-line with the Processor

3. STATUS AND COMMAND

Table 2 provides status and command byte data for the HSPTR/P.

4. INTERRUPTS

When enabled in the READ Mode, the device controller generates an external device interrupt when a data character is present in the controller, waiting to be transferred to the Processor. When enabled, in the WRITE Mode, the device controller generates an external device interrupt when the controller is ready for another character to be punched.

5. INITIALIZATION

When the INITIALIZE pushbutton on the Processor is depressed, the following occurs:

1. Interrupts of all kinds are disabled.

2. The BSY, NMTN, and EX status bits are set.

3. The DISABLE, STOP, INCR, and READ command functions are set.

4. The punch power is turned off unless the POWER pushbutton is depressed (locked on).

6. PUNCH POWER CONTROLS

There are two pushbuttons on the front panel of the High Speed Paper Tape Punch. The top button, labeled REMOTE, determines the state of the Punch in reference to the Processor. If this pushbutton is released, the Punch is off-line with the Processor and is said to be in a LOCAL Mode. If the button is depressed, the Punch is on-line with the Processor and is said to be in a REMOTE Mode. The POWER pushbutton is located directly below the REMOTE pushbutton. With the POWER Switch released, and the REMOTE Switch depressed, Punch power may be turned on by the program via the command RUN bit. The power may also be turned off by the program via the command STOP bit. Figure 1 illustrates when the power is on/off as a function of programmed and manual controls. The letter B represents the POWER button depressed, $\bar{B}$ means the POWER button is released. The letter P represents program-controlled power on. $\bar{P}$ means program-controlled power off.

7. MODE SWITCHING

With a Combination Reader/Punch, care must be used when switching modes. The following is an example of switching from the WRITE to the READ Mode.

```
        .
        .
        .
        .
        WD    DEVICE, BUFFER
        .
        .
        .
        .
WAIT    SSR   DEVICE,STATUS
        BTC   8, WAIT
        OC    DEVICE, READ
        .
        .
        .
        .
```

	$\bar{B}$ $\bar{P}$	$\bar{B}$ P	B P	B $\bar{P}$
REMOTE	off	on	on	on
LOCAL	off	off	on	on

Figure 1. Punch Power

TABLE 2. READER/PUNCH STATUS AND COMMAND BYTE FORMAT

STATUS BYTE	OV			NMTN	BSY	EX		DU
BIT NUMBER	0	1	2	3	4	5	6	7
COMMAND BYTE	DISABLE	ENABLE	STOP	RUN	INCR	SLEW	WRITE	READ

STATUS BIT DESCRIPTIONS

BIT	READER	PUNCH
OV	The Overflow bit is set when the Buffer Register is loaded from the Reader before the previous character has been transferred. This condition can only happen in the SLEW mode.	The Overflow bit is always in a reset condition in the WRITE mode.
NMTN	The No-Motion bit is set when the Reader has been issued a STOP command and the tape has stopped on the next character.	The No-Motion bit is always in a reset condition in the WRITE mode.
BSY	The BUSY bit is set when the Buffer Register is empty, waiting for a character from the Reader.	The BUSY bit is set when the Buffer is full, waiting to transfer to the Punch.
EX	The Examine bit is set whenever OV=1 or NMTN=1.	The Examine bit is always reset in the WRITE mode.
DU	The Device Unavailable bit is set when the power to the Reader motor is off, or the Read/Load lever is in the Load position (straight up).	The Device Unavailable bit is set when the power to the Punch motor is off, or the REMOTE switch is released, or a low tape condition exists on the tape reel. There is no low tape sensor on the fan fold bins.

TABLE 2. READER/PUNCH STATUS AND COMMAND BYTE FORMAT (Continued)

COMMAND BIT DESCRIPTION

BIT	READER	PUNCH
DISABLE	This command inhibits Interrupts from the Device Controller from interrupting the Processor.	Same as the Reader.
ENABLE	This command permits Interrupts from the Device Controller to interrupt the Processor.	Same as the Reader.
STOP	This command halts the motion of the tape after the next character has been read. The next character to be read is positioned over the sense lights when the tape stops.	This command halts the tape after the next character is punched. The Punch motor is also turned off, unless the Power Switch on the panel is depressed.
RUN	This command starts the tape moving and leaves the controller in the RUN mode.	This commands starts the Punch motor, unless the REMOTE Switch on the panel is released.
INCR	In this mode of operation, the tape is advanced one character when the controller is in the RUN mode and BSY=1. The tape stops after reading one character. The tape remains stopped until a Read Data instruction is issued to the Processor, which will reset BSY and start the tape moving.	Not used.
SLEW	In this mode of operation, the tape is advanced, reading continuous characters, until stopped.	Not used.
WRITE		Designates the High Speed Paper Tape Punch.
READ	Designates the High Speed Paper Tape Reader.	

The sense loop is required to insure that the last character in the buffer register is punched prior to issuing the Output Command READ. If the READ command is given too soon, the last character is interferred with. This is because the command READ causes the character on the tape, under the sense lights to be strobed into the buffer register. The logic behind this is that when the Reader has been issued a command STOP (Output Command WRITE causes a stop action also),the tape stops with the next character to be read under the sense lights. Thus, a RUN/STOP action will not cause a skipping of characters on tape. Because of this feature, there is no need to sense status and check for BSY = 1 in switching from the READ Mode to the WRITE Mode.

Appendix 1 shows a sample program for the Combination Reader/Punch which reads a block of characters and punches a block of characters.

8. DEVICE NUMBER

The High Speed Reader/Punch device controller is normally assigned address X'03' if using a Reader only. If using a Punch only, or both a Reader and a Punch, address X'13' is normally assigned. These device numbers are easily changed by a minor modification to the device controllers.

9. SAMPLE PROGRAMS

Appendix 2 is a sample program using the High Speed Paper Tape Reader in the Incremental Mode. The Output command for this mode of operation is X'99'.

Appendix 3 is a sample program using the High Speed Paper Tape Reader in the Slew Mode. The Output command for this mode of operation is X'95'.

Appendix 4 is a sample program using the High Speed Paper Tape Punch. The Output command for this operation is X'92'.

APPENDIX 1
SAMPLE PROGRAM
READER/PUNCH COMBINATION

```
                        OPT    PASS2,NOPNCH,PRINT,STOP
               *
               * SAMPLE PROGRAM USING THE HSPTR/HSPTP IN MODE SWITCHING
               *
0000R C840     START    LHI    DEVICE,3          SET DEVICE NUMBER
      0003
0004R C830              LHI    3,100             SET HIGH LIMIT AND
      0064
0008R C820              LHI    2,1               LOW LIMITS OF THE
      0001
000CR 0711     INPUT    XHR    1,1               BUFFER AREA
000ER DE40              OC     DEVICE,READ       READ MODE
      0042R
0012R 9D40     INP1     SSR    DEVICE,           STATUS
0014R 4280              BTC    8,INP1
      0012R
0018R DB41              RD     DEVICE,BUFFER(1)    INPUT CHAR.
      0048R
001CR C110              BXLE   1,INP1
      0012R
               *
               *
0020R 0711     OUTPUT   XHR    1,1               CLEAR LOW LIMITS
0022R DE40              OC     DEVICE,WRITE      WRITE MODE
      0044R
0026R 9D45     OUT1     SSR    DEVICE,STATUS
0028R 4280              BTC    8,OUT1
      0026R
002CR DA41              WD     DEVICE,BUFFER(1)    OUTPUT CHAR.
      0048R
0030R C110              BXLE   1,OUT1
      0026R
0034R 9D45     WAIT     SSR    DEVICE,STATUS     WAIT UNTIL LAST
0036R 4280              BTC    8,WAIT            CHAR. IS PUNCHED
      0034R
003AR DE40              OC     DEVICE,STOP       TURN PUNCH OFF
      0046R
003ER 4300              B      INPUT
      000CR
0042R 9999     READ     DC     X'9999'
0044R 9292     WRITE    DC     X'9292'
0046R 2020     STOP     DC     X'2020'
0004           DEVICE   EQU    4
0005           STATUS   EQU    5
0048R          BUFFER   DS     100
00ACR                   END

BUFFER   0048R
DEVICE   0004
INP1     0012R
INPUT    000CR
OUT1     0026R
OUTPUT   0020R
READ     0042R
```

```
START   0000R
STATUS  0005
STOP    0046R
WAIT    0034R
WRITE   0044R
```

APPENDIX 2
SAMPLE PROGRAM
READER-INCREMENTAL MODE

```
                          OPT    PASS2,NOPNCH,PRINT,STOP
                   *
                   * SAMPLE PROGRAM FOR HSPTR (INCREMENTAL MODE)
                   *
                   *
0000R D310         INPUT  LB     DEV,BINDV          SELECT DEVICE NUMBER
      0078
0004R DE10                OC     DEV,BINDV+1        ISSUE OUTPUT COMMAND
      0079
0008R 9D12         SENSE  SSR    DEV,STATUS         GET STATUS OF READER
000AR 4250                BTC    5,TROBLE           DU OR EX=1; ERROR
      0018R
000ER 4280                BTC    8,SENSE            BSY=1; WAIT
      0008R
0012R 9B13                RDR    DEV,TEMP           READ ONE CHAR. FROM TAPE
0014R 4300                B      PROCES             PROCESS THIS CHARACTER
      0020R
                   *
0018R C200         TROBLE LPSW   STOP               TROUBLE CORRECTED; RETURN
      001CR
001CR 8000         STOP   DC     X'8000',A(INPUT)     TO INPUT ROUTINE
      0000R
                   *
0001               DEV    EQU    1                  DEVICE NUMBER X'13'
0002               STATUS EQU    2                  HOLDS STATUS BITS
0003               TEMP   EQU    3                  TEMPORARY STORAGE
                   *
0078               BINDV  EQU    X'78'              OUTPUT COMMAND IS X'99'
                   *
0020R              PROCES EQU    *
0020R                     END

BINDV    0078
DEV      0001
INPUT    0000R
PROCES   0020R
SENSE    0008R
STATUS   0002
STOP     001CR
TEMP     0003
TROBLE   0018R
```

APPENDIX 3
SAMPLE PROGRAM
READER-SLEW MODE

```
                        OPT    PASS2,NOPNCH,PRINT,STOP
               *
               *SAMPLE PROGRAM FOR HSPTR (SLEW MODE)
               * READ TAPE UNTIL X'FF' CHARACTER IS ENCOUNTERED
               *
0000R 0744              XHR    INDEX,INDEX          ZERO INDEX
0002R D310     INPUT    LB     DEV,BINDV            SELECT DEVICE NUMBER
      0078
0006R DE10              OC     DEV,BINDV+1          ISSUE OUTPUT COMMAND
      0079
000AR 9D12     SENSE    SSR    DEV,STATUS           CHECK STATUS
000CR 4250              BTC    5,TROBLE             DU OR EX=1; ERROR
      002CR
0010R 4280              BTC    8,SENSE              BSY=1; WAIT
      000AR
0014R 9B13              RDR    DEV,CHECK            READ FIRST CHAR FROM TAPE
0016R C530              CLHI   CHECK,X'FF'          IS IT DELIMITING CHARACTER
      00FF
001AR 4330              BE     STOP                 YES, STOP TAPE MOTION
      0026R
001ER 4034              STH    CHECK,BUFFER(INDEX)  STORE& GET NEXT CH
      0034R
0022R C140              BXLE   INDEX,SENSE          READ TAPE UNTIL DELIMITOR
      000AR
0026R DE10     STOP     OC     DEV,DONE             OUTPUT COMMAND X'20'
      1034R
002AR 0307              BR     RETURN               GO BACK TO MAIN PROGRAM
002CR C200     TROBLE   LPSW   HALT                 TROUBLE CORRECTED; RETURN
      0030R
0030R 8000     HALT     DC     X'8000',A(INPUT)     TO INPUT ROUTINE
      0002R
               *
0001           DEV      EQU    1                    DEVICE NUMBER X'13'
0002           STATUS   EQU    2                    HOLDS STATUS BITS
0003           CHECK    EQU    3                    REG USED TO CHECK FOR DELI
0004           INDEX    EQU    4                    INDEX VALUE
0007           RETURN   EQU    7
0078           BINDV    EQU    X'78'                OUTPUT COMMAND IS X'95'
0034R          BUFFER   DS     4096                 BUFFER SIZE
1034R 2020     DONE     DC     X'2020'              STOP COMMAND
1036R                   END

BINDV    0078
BUFFER   0034R
CHECK    0003
DEV      0001
DONE     1034R
HALT     0030R
INDEX    0004
INPUT    0002R
RETURN   0007
SENSE    000AR
STATUS   0002
STOP     0026R
```



```
TROBLE   002CR
```

APPENDIX 4
SAMPLE PROGRAM
PUNCH

```
                              OPT     PASS2,NOPNCH,PRINT,STOP
                    *
                    *  SAMPLE PROGRAM FOR THE HIGH SPEED PAPER TAPE PUNCH
                    *
0000R 0722                    XHR     COUNT,COUNT          ZERO LOW LIMITS
0002R D300          OUTPUT    LB      DEV,BOUTDV           SELECT DEVICE NUMBER
      007A
0006R DE00                    OC      DEV,BOUTDV+1         ISSUE OUTPUT COMMAND
      007B
000AR 9D01          SENSE     SSR     DEV,STATUS           CHECK STATUS
000CR 4210                    BTC     1,TROBLE             DU=1; STOP
      0026R
0010R 4280                    BTC     8,SENSE              BSY=1; WAIT
      000AR
0014R DA02                    WD      DEV,BUFFER(COUNT)       OUTPUT CHARACTER
      002ER
0018R C120                    BXLE    COUNT,SENSE          DO UNTIL DONE
      000AR
001CR 9D01          WAIT      SSR     DEV,STATUS           WAIT FOR LAST
001ER 4280                    BTC     8,WAIT               CHAR TO BE OUTPUT
      001CR
0022R DE00                    OC      DEV,OFF              TURN PUNCH MOTOR OFF
      102ER
0026R C200          TROUBLE LPSW STOP                      TROUBLE CORRECTED; RETURN
      002AR
002AR 8000          STOP      DC      X'8000',A(OUTPUT)    TO OUTPUT ROUTINE
      0002R
002ER               BUFFER    DS      4096
                    *
0000                DEV       EQU     0                    DEV. NUM. X'13'
0001                STATUS    EQU     1                    HOLDS STATUS OF THE PUNCH
0002                COUNT     EQU     2                    HOLDS LOW LIMIT OF BUFFER
0005                RETURN    EQU     5                    RETURN TO MAIN PROG.
007A                BOUTDV    EQU     X'7A'                OUT. COMD. X'92'
102ER 2020          OFF       DC      X'2020'
1030R                         END

BOUTDV 007A
BUFFER 002ER
COUNT  0002
DEV    0000
OFF    102ER
OUTPUT 0002R
RETURN 0005
SENSE  000AR
STATUS 0001
STOP   002AR
TROBLE 0026R
WAIT   001CR
```

CARD READER OPERATION AND PROGRAMMING MANUAL

1. GENERAL DESCRIPTION

The 7-510 Card Reader employs a photoelectric read station and a vacuum throat feed assembly. A special "wide strobe" read technique is used to preclude loss of data, even on cards which have been mispunched by as much as plus or minus one-half column.

The card read rate is in excess of 200 cards per minute with a 500 card capacity for both the input hopper and the output stacker. Throughout the read operation light current checks, dark current checks, and card motion checks are continuously performed to verify the performance of the Card Reader.

2. OPERATOR CONTROLS

2.1 POWER

The lighted POWER pushbutton applies AC power to all circuits. The pushbutton is lit when the power is on.

2.2 MOTOR Start

The lighted MOTOR pushbutton starts the drive motor if no error indicator lights are lit. The pushbutton is lit when the drive motor is running.

2.3 Read START

The lighted START pushbutton clears all error indicators and advances the Card Reader to the "ready" state to begin a read cycle upon receipt of the proper signal. The pushbutton is lit when the switch is depressed and no errors have been detected.

2.4 Read STOP

The lighted STOP pushbutton inhibits further read cycles until Read START is again depressed. Read STOP action is delayed until the current read cycle is completed. The pushbutton is lit when the switch is depressed, or if the Card Reader is stopped due to an error detection.

3. STATUS INDICATOR LIGHTS

3.1 Power On

The indicator on the POWER Switch is illuminated when power is applied to the Card Reader.

3.2 Motor On

The MOTOR Switch indicator is illuminated when the motor is running.

3.3 Read Start

The START Switch is illuminated when the switch is depressed and no malfunctions have been detected.

3.4 Read Stop

The STOP Switch is illuminated when the switch is depressed or the Card Reader has stopped due to a trouble detection, as described in the following paragraphs.

3.5 PICK FAIL

If a card fails to be picked upon command, the PICK FAIL indicator is illuminated.

TABLE 1
CARD READER STATUS AND COMMAND BYTE DATA
(HEX ADDRESS 04)

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE	EOV	TBL	HE	NMTN	BSY	EX	EOM	DU
COMMAND BYTE	DISABLE	ENABLE	FEED					

EOV — The EOV bit is set when the data is not taken from the Device Controller buffer before the next column of data arrives from the read station. This bit is reset by a FEED Command.

TBL/DU — These bits are set when the Card Reader fails to pick a card upon command, or when an error condition occurs in the Card Reader. The error conditions are:

1. Card Motion Error
2. Light Current Error
3. Dark Current Error

These error conditions prevent the reading of any more cards until manually reset by the operator.

HE — The HE bit is set when the last card in the input hopper has been read. When HE sets, NMTN is set. The HE bit must be manually reset by the operator.

NMTN — The NMTN is set except for the time between a FEED command and the time it takes for a card to pass through the read station.

BSY — The BSY bit is set while the Device Controller is awaiting data from the Card Reader. It resets when the data is available to be transferred.

EX — The EX bit sets when any one of the upper four (4) bits of the Status byte is set.

EOM — The EOM bit is set whenever NMTN is set, and when the input hopper becomes empty.

DISABLE — This command disables the Card Reader Device Interrupt.

ENABLE — This command enables the Card Reader Device Interrupt.

FEED — This command initiates a new card feed cycle; however, no action occurs if TBL, DU, or HE is set.

3.6 CARD MOTION Error

If the interval between the time the selected card enters the read station and the time the card leaves, does not correspond to 85 ± 1/3 columns (the total card width), the CARD MOTION indicator illuminates.

3.7 LIGHT CURRENT Error

When all photo-read-cells do not conduct whenever a card is not in the read station, the LIGHT CURRENT indicator illuminates.

3.8 DARK CURRENT Error

The DARK CURRENT indicator illuminates if all photo-read-cells do not go dark for some instant between the beginning of the card and column 1, or between column 80 and the end of the card.

4. STATUS AND COMMAND BYTES

Table 1 illustrates the status and command byte coding for the Card Reader.

5. DATA FORMAT

A card Feed command causes the card to move over the photo-read-cells column by column, starting with column 1. Every column read (blank columns are read as all bits zero) generates a data strobe for that column and initiates a data transfer cycle. The first Read Data instruction reads the top six rows of the column; the second Read Data instruction reads the bottom six rows of that column. Figure 1 is an example of the data byte format.

6. INTERRUPTS

When enabled (Bit 1 of the COMMAND byte set), the Card Reader Device Controller generates an external device interrupt for each column read. The interrupt indicates to the Processor that data is available for transfer.

7. INITIALIZATION

When the INITIALIZE pushbutton on the Processor is depressed, the following occurs:

1. The NMTN and EOM bits are set.
2. The EOV bit is reset.
3. The BSY and EX bits are set.

8. OPERATOR PROCEDURES

After applying power to the Card Reader, allow it a few minutes to warm up. Cards should be placed face down in the hopper with the 12-edge toward the operator. Additional cards may be added to the hopper without interferring with the operation.

ROW NUMBER			12	11	0	1	2	3	First Data Byte
BIT NUMBER	0	1	2	3	4	5	6	7	
ROW NUMBER			4	5	6	7	8	9	Second Data Byte

NOTE: Bit numbers 0 and 1 should always be zero.

Figure 1. Data Byte Format

9. PROGRAMMING

A sample card input routine is shown in Appendix 1. In the sample program, note that the HE bit (hopper empty) is checked before other bits. This bit does not become set until the last card is read. If 80 columns are not read from each card, there is a Card Reader malfunction, as all blank columns should be read as zeros.

Code conversion is required when reading conventional Hollerith cards. A GE-PAC 30 subroutine (HTASCV, Program Number 07-019) is available for this purpose. The subroutine converts the 12 bits of binary data from one column to the corresponding 7-bit ASCII code.

Appendix 2 provides a table of the Hollerith punched-card codes for the ASCII character set.

APPENDIX 1
SAMPLE PROGRAM

```
                       *
                              OPT    PASS2,PRINT,PUNCH,STOP
                       *
                       *  NON-INTERRUPT CARD READER ROUTINE
                       *
0001                   DEVNUM EQU    1
0002                   STATUS EQU    2
0003                   INDEX  EQU    3
0004                   INCR   EQU    4
0005                   LIMIT  EQU    5
000F                   RETURN EQU    15
                       *
0000R 0733             READ   XHR    INDEX,INDEX       ZERO INDEX VALUE
0002R C840                    LHI    INCR,2            SET UP INCREMENT
      0002
0006R C850                    LHI    LIMIT,158         SET UP LIMIT
      009E
000AR D310                    LB     DEVNUM,SINDV      SET DEVICE NUMBER
      007C
000ER 9D12             WAIT   SSR    DEVNUM,STATUS
0010R 4320                    BFC    2,WAIT            IF EOM NOT SET-HANG
      000ER
0014R C420                    NHI    STATUS,X'20'      HOPPER EMPTY CHECK
      0020
0018R 4230                    BNZ    EMPTY
      0038R
                       *
001CR DE10             FEED   OC     DEVNUM,SINDV+1
      007D
0020R 9D12             SENSE  SSR    DEVNUM,STATUS
0022R 4270                    BTC    7,ERROR           BITS SHOULD NOT BE SET
      0038R
0026R 4280                    BTC    8,SENSE           BUSY BIT SET-HANG
      0020R
002AR DB13                    RD     DEVNUM,BUFFER(INDEX)    FIRST CHAR. (ROWS 1
      0038R
002ER DB13                    RD     DEVNUM,BUFFER+1(INDEX)  SECOND CHAR (ROWS
      0039R
0032R C130                    BXLE   INDEX,SENSE       80 COLUMNS READ
      0020R
                       *
                       *  DO HOLLERITH TO ASCII CONVERSION ROUTINE
                       *
0036R 030F                    BR     RETURN
                       *
007C                   SINDV  EQU    X'7C'             50 SEQUENCE
                       *
0038R                  EMPTY  EQU    *                 INPUT HOPPER EMPTY
                       *
0038R                  ERROR  EQU    *                 ERROR ROUTINE
                       *
0038R                  BUFFER DS     160
00D8R                         END
```

```
BUFFER  0038R
DEVNUM  0001
EMPTY   0038R
ERROR   0038R
FEED    001CR
INCR    0004
INDEX   0003
LIMIT   0005
READ    0000R
RETURN  000F
SENSE   0020R
SINDV   007C
STATUS  0002
WAIT    000ER
```

APPENDIX 2
ASCII TO CARD CODE CONVERSION

GRAPHIC	8-BIT ASCII CODE	7-BIT ASCII CODE	CARD CODE	GRAPHIC	8-BIT ASCII CODE	7-BIT ASCII CODE	CARD CODE
SPACE	A0	20	0-8-2	@	C0	40	8-4
!	A1	21	12-8-7	A	C1	41	12-1
"	A2	22	8-7	B	C2	42	12-2
#	A3	23	8-3	C	C3	43	12-3
$	A4	24	11-8-3	D	C4	44	12-4
%	A5	25	0-8-4	E	C5	45	12-5
&	A6	26	12	F	C6	46	12-6
'	A7	27	8-5	G	C7	47	12-7
(	A8	28	12-8-5	H	C8	48	12-8
)	A9	29	11-8-5	I	C9	49	12-9
*	AA	2A	11-8-4	J	CA	4A	11-1
+	AB	2B	12-8-6	K	CB	4B	11-2
,	AC	2C	0-8-3	L	CC	4C	11-3
-	AD	2D	11	M	CD	4D	11-4
.	AE	2E	12-8-3	N	CE	4E	11-5
/	AF	2F	0-1	O	CF	4F	11-6
0	B0	30	0	P	D0	50	11-7
1	B1	31	1	Q	D1	51	11-8
2	B2	32	2	R	D2	52	11-9
3	B3	33	3	S	D3	53	0-2
4	B4	34	4	T	D4	54	0-3
5	B5	35	5	U	D5	55	0-4
6	B6	36	6	V	D6	56	0-5
7	B7	37	7	W	D7	57	0-6
8	B8	38	8	X	D8	58	0-7
9	B9	39	9	Y	D9	59	0-8
:	BA	3A	8-2	Z	DA	5A	0-9
;	BB	3B	11-8-6	[	DB	5B	12-8-2
<	BC	3C	12-8-4	\	DC	5C	11-8-1
=	BD	3D	8-6	]	DD	5D	11-8-2
>	BE	3E	0-8-6	↑	DE	5E	11-8-7
?	BF	3F	0-8-7	←	DF	5F	0-8-5

HOLLERITH TO ASCII CONVERSION PROGRAM DESCRIPTION

1. INTRODUCTION

The Hollerith to ASCII Conversion (HTASCV) subroutine is used to convert 12-bits of binary data read from one column of a punched card to the corresponding 7-bit ASCII code.

2. PROGRAM TAPE

The tape for HTASCV (07-019M08) is relocatable, and can be loaded with the General Loader (06-025M10). The subroutine requires X'CE' bytes of memory.

3. CALLING SEQUENCE

The calling sequence required to use this routine is as follows:

```
        EXTRN  HTASCV

        LH     10, DATA      USE R10 FOR
                             ARG
LOC     BAL    15,HTASCV     USE R15 FOR
                             RETURN
        B      ERROR         RETURN HERE
                             ON ERROR
        STB    10, RESULT    RETURN HERE
                             IF OK
```

Control returns to LOC + 4 if the binary data presented in register 10 does not correspond to a proper card code. On the error return, register 10 holds X'20', the ASCII code for a space character. On a normal return, register 10 holds the ASCII code for the proper character.

4. OPERATION

The HTASCV routine uses registers 10 -15. These registers are not restored to their initial state on return to the calling program. The card code specified in register 10 is converted with a table look-up procedure. The table used is shown on Table 1.

Card codes for unspecified entries in the table (e.g. 1-8) are converted to ASCII spaces. These table entries could be defined with other codes if needed for special applications.

Refer to Table 2 for a definition of the card code for all ASCII characters.

5. TIMING

The code conversion requires the execution of 19 -43 instructions. Assuming a uniform distribution of card codes, the average code conversion requires approximately 30 instruction times.

TABLE 1. CONVERSION TABLE

	Rows 12, 11, 0, 8							
Rows 1-7, 9	blank	0	8	0-8	11	12	11-8	12-8
blank	b̸	0	8	Y	—	&	Q	H
1	1	/			J	A	\	
2	2	S	:	b̸	K	B	]	[
3	3	T	#	,	L	C	&	.
4	4	U	@	%	M	D	*	<
5	5	V	'	←	N	E	)	(
6	6	W	=	>	O	F	;	+
7	7	X	"	?	P	G	↑	!
9	9	Y			R	I		

b̸ = ASCII space character (X'20')

TABLE 2
ASCII TO CARD CODE CONVERSION

GRAPHIC	8-BIT ASCII CODE	7-BIT ASCII CODE	CARD CODE	GRAPHIC	8-BIT ASCII CODE	7-BIT ASCII CODE	CARD CODE
SPACE	A0	20	0-8-2	@	C0	40	8-4
!	A1	21	12-8-7	A	C1	41	12-1
"	A2	22	8-7	B	C2	42	12-2
#	A3	23	8-3	C	C3	43	12-3
$	A4	24	11-8-3	D	C4	44	12-4
%	A5	25	0-8-4	E	C5	45	12-5
&	A6	26	12	F	C6	46	12-6
'	A7	27	8-5	G	C7	47	12-7
(	A8	28	12-8-5	H	C8	48	12-8
)	A9	29	11-8-5	I	C9	49	12-9
*	AA	2A	11-8-4	J	CA	4A	11-1
+	AB	2B	12-8-6	K	CB	4B	11-2
,	AC	2C	0-8-3	L	CC	4C	11-3
-	AD	2D	11	M	CD	4D	11-4
.	AE	2E	12-8-3	N	CE	4E	11-5
/	AF	2F	0-1	O	CF	4F	11-6
0	B0	30	0	P	D0	50	11-7
1	B1	31	1	Q	D1	51	11-8
2	B2	32	2	R	D2	52	11-9
3	B3	33	3	S	D3	53	0-2
4	B4	34	4	T	D4	54	0-3
5	B5	35	5	U	D5	55	0-4
6	B6	36	6	V	D6	56	0-5
7	B7	37	7	W	D7	57	0-6
8	B8	38	8	X	D8	58	0-7
9	B9	39	9	Y	D9	59	0-8
:	BA	3A	8-2	Z	DA	5A	0-9
;	BB	3B	11-8-6	[	DB	5B	12-8-2
<	BC	3C	12-8-4	\	DC	5C	11-8-1
=	BD	3D	8-6	]	DD	5D	11-8-2
>	BE	3E	0-8-6	↑	DE	5E	11-8-7
?	BF	3F	0-8-7	←	DF	5F	0-8-5

SELECTOR CHANNEL PROGRAMMING MANUAL

1. INTRODUCTION

The Selector Channel controls the transfer of data between I/O devices and core memory at rates of up to 500K bytes per second. Up to 25 I/O devices can be connected to the Selector Channel, but only one device can transfer data at a time. The advantage gained in using the Selector Channel is that other program processing can occur simultaneously with the transfer of data between the I/O device and core. This is accomplished by allowing the Selector Channel and the Processor to access memory on a cycle-stealing basis. In some instances, the execution times of the program in process will be affected, while in others, the effect will be negligible. This depends upon which model GE-PAC 30 Processor is in use and the rate at which the Selector Channel and Processor both compete for access to memory. A GE-PAC 30 sales engineer can supply exact details upon request.

This description applies to Selector Channel Controller Boards, Part Numbers 32-030 and 32-031. Figure 1 is a block diagram which shows the incorporation of the Selector Channel into the GE-PAC 30 peripheral system.

2. PROGRAMMING CONSIDERATIONS

Programming a device on the Selector Channel consists of setting up the device, setting up the Selector Channel, and sending a GO command to the Selector Channel. When all devices on the Selector Channel are idle, the Selector Bus becomes a part of the Multiplexor Bus. This provides the path to set up the device and the Selector Channel. The last device addressed prior to sending the GO command is the device the Selector Channel controls, assuming that the device is connected to the Selector Channel. The program must, therefore, address the desired Selector Channel device, set up the Selector Channel, and then send the Go command before addressing any other devices.

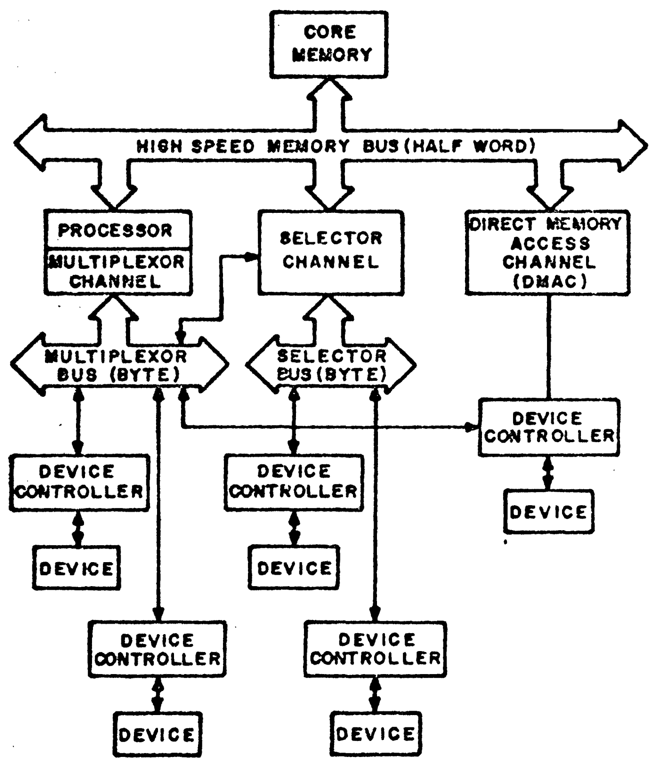


Figure 1 Systems Interface, Block Diagram

During the data transfer, the Selector Channel provides a direct data path between the device and core memory. Until the transfer is completed, no I/O instructions can be issued to any device on the Selector Channel, including the device transferring data.

If devices on the Selector Channels are referenced while the Channel is busy, the False Sync (V condition code) bit will be set. The setting up or the initialization of the device is accomplished by executing an Output Command (OC or OCR) instruction. Refer to the Programming Manual for the device to be controlled for the bit configuration of the Output Command. Note that the Selector Channel has a unique device number just like all other I/O devices. Output Commands, as with all input/output instructions, affect only the device addressed.

The Selector Channel has a 16-bit incrementing address register and a 16-bit final address register. The user program loads the starting core address into the incrementing register and the final core address into the final address register. Transfer is completed when the incrementing address register matches the final address register. The address limits are expressed inclusively; transfers begin and end on the addresses placed in the starting and final address registers.

Core memories in most GE-PAC 30 Processors are addressed on halfword boundaries; that is, each time memory is accessed two bytes or a halfword are obtained. A sixteen bit address register is used, with the least significant bit, bit 15, being used to determine the byte desired. See Figure 2.

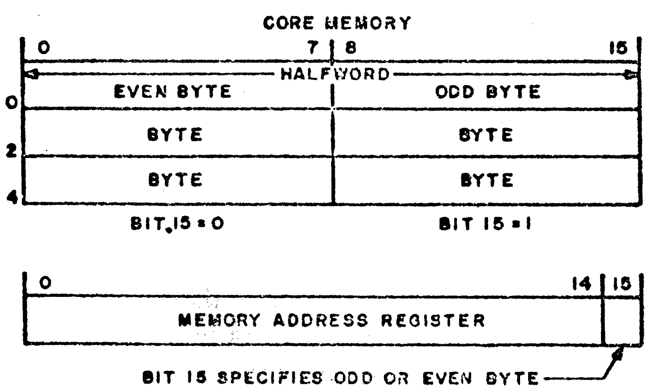


Figure 2. Memory Addressing

Each time the Selector Channel accesses core memory, two bytes (a halfword) are transmitted. It is mandatory that data transfers begin on a halfword boundary and end on either halfword or byte boundaries. This is accomplished by setting bit 15 to zero in the starting register and bit 15 to a one in the final address register for halfword boundaries and zero for byte boundaries. The following will result if data transfers are ended on byte boundaries:

1. Write Mode (Core to Device) - End on byte boundary (bit 15 = 0) - no effect

2. Read Mode (Device to Core) - End on byte boundary (bit 15 = 0) - The previous contents of the last odd byte in core is written into the current odd byte in core. See Figure 3.

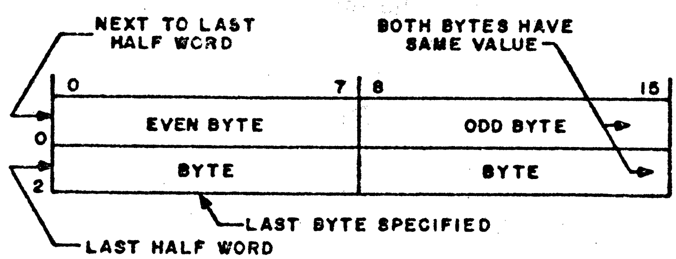


Figure 3. Core Memory Configuration, End on Byte Boundary

The user program specifies the mode, either Read or Write, and gives the GO command. The following sections provide details for programming the Selector Channel.

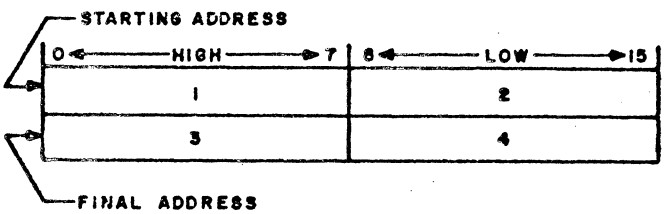
NOTE

When executing programs that involve the use of the Selector Channel, the Processor may not be run in the Variable Mode.

3. SPECIFICS

3.1 Transmission of Starting and Final Addresses

Four successive bytes are required to specify the starting and final addresses. Either the Write Data (WD or WDR), or Write Block (WB or WBR) instructions may be used to send the starting and final addresses to the Selector Channel Controller. Figure 4 illustrates the meaning of four bytes used for addressing.



1. Starting Address High (bits 0-7)
2. Starting Address Low (bits 8-15)
3. Final Address High (bits 0-7)
4. Final Address Low (bits 8-15)

Figure 4. Meaning of the Data Bytes When Setting Start and Final Address

3.2 Status and Commands

Table 1 illustrates the Selector Channel Status and Command byte coding. A Sense Status instruction (SS or SSR) is used to transfer the status byte from the Selector Channel Device Controller to the Processor. The least significant four bits (4-7) of the status byte are copied into the condition code during the Sense Status operation. Branch instructions can test these four bits directly.

The Output Command instruction (OC or OCR) is used for transmitting the command byte to the Selector Channel Controller. Table 1 also describes the command byte.

3.3 Termination

Data transmission between the Selector Channel and the Device presently connected to it will be halted if any of the following conditions occur:

1. The starting address matches the final address. This would be considered a normal termination.

2. The starting (incrementing) address goes from all ones to zero (maximum count). In this case, no match occurred and this would be considered an abnormal termination.

3. Any of the DU, EOM, or EX status bits of the device presently connected to the Selector Channel changes to a ONE. This is also an abnormal termination.

4. A STOP command is sent to the Selector Channel Controller via a user program.

The termination condition is determined one of two ways: by a status scan, or by the interrupt method. The methods are described in the following paragraphs.

1. Status Scan. The status of the Selector Channel Controller may be examined by issuing a Sense Status (SS or SSR) instruction. The Busy Bit is a 1 while transmission is in process, and zero when no transmission is in effect. One method of testing for termination would be to continually or periodically test the Busy Bit of the Selector Channel Controller. The change from one to zero would then indicate the termination of a data transfer. When the Selector Channel is busy, only the busy bit

TABLE 1
SELECTOR CHANNEL STATUS AND COMMAND BYTE DATA

BIT NUMBER	0	1	2	3	4	5	6	7
STATUS BYTE					BSY			
COMMAND BYTE			READ	GO	STOP			

BSY This bit is set when the Selector Channel is in the process of transferring data.

READ This command changes the mode of the Selector Channel from WRITE to READ. In the READ mode, data is transmitted from the active device on the Selector Channel and written into core memory. Whenever a data transmission has been completed, the Selector Channel is placed in the WRITE mode. Each time a READ operation is required, a READ Command must be issued.

GO This command initiates a data transmission. This command can be issued at the same time the READ/WRITE mode is established.

STOP This command halts any data transmission in process and initializes the Selector Channel for starting a new operation.

(bit 4) is present in the status byte and all other bits are zero. At termination, the status of the device is presented in the status byte, except for the Busy Bit which is zero.

2. Interrupt Method. When data transmission is initiated on the Selector Channel, the interrupt is effectively enabled. If external device interrupts are permitted (bit 1 of the PSW set) to enter the Processor, at termination, the Processor will be interrupted. The acknowledge interrupt (AI, AIR) instruction will cause the device number of the Selector Channel (normally X'FO') and status of the peripheral device to be brought into the Processor, and also clear the interrupt for the Processor. The Busy Bit is treated in the manner described previously for Status Scan.

3.4 Reading the Final Address

The last Processor core location either written into or read from amy be obtained by executing a pair of Read Data (RD or RDR) instructions or a Read Block (RB or RBR) instruction. This information will permit a user program to verify a successful data transmission or determine at what address termination occurred.

Figure 5 illustrates the meaning of the order in which the data is read into the Processor.

4. DEVICE NUMBER

The Selector Channel is normally assigned device number X'FO', but may easily be changed by a minor wiring modification on the Selector Channel device controller board. Refer to the Maintenance Manual for specific details.

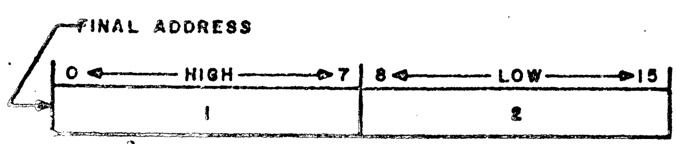
5. INITIALIZATION

Whenever the INITIALIZE pushbutton on the Processor is depressed, the following actions occur:

1. Any data transmission in process is halted and the stop mode is effected.
2. The Selector Channel is placed in the Write Mode.
3. The Selector Channel is made idle.
4. The Selector Channel interrupt is reset.

6. SAMPLE PROGRAM

Appendix 1 presents a sample program for a magnetic tape unit connected to the Selector Channel. The purpose of this sample program is to show the program instruction used to control the Selector Channel and the order in which they may be executed.



1. Final Address High (bits 0-7)
2. Final Address Low (bits 8-15)

Figure 5. Order in Which Read Data Instructions are Executed

APPENDIX 1

```
                              OPT    PASS2,PRINT,PUNCH,STOP,LAB=MTSMP3
                    *     SAMPLE PROGRAM FOR MAGNETIC TAPE
                    *     EOF/WRITE/BACKSPACE/READ/COMPARE- 9 TRACK @ 800 BPI
                    *     USES SELECTOR CHANNEL
                    *
                    *     GENERATES AN END-OF-FILE MARK, WRITES THE
                    *     CONTENTS OF CORE (X'500' TO X'FFF') ON THE TAPE,
                    *     BACKSPACES 1 RECORD, REREADS THE RECORD AND
                    *     COMPARES IT WITH WHAT WAS THERE.
                    *     IF THE RECORD IS CORRECT, ANOTHER
                    *     END-OF-FILE MARK IS WRITTEN.
                    *
                    *     ASSUMES TAPE WILL NOT ENCOUNTER BOT OR EOT
                    *
0000                MT       EQU    0
0001                SC       EQU    1
0003                STAT     EQU    3
0004                INDEX    EQU    4
0005                INCR     EQU    5
0006                LIMIT    EQU    6
0007                TEMP     EQU    7
0008                RETURN   EQU    8
                    *
                    *     WRITE CORE ON TAPE
0000R C800          DUMP     LHI    MT,X'85'           MAG TAPE DEV NO
      0085
0004R C810                   LHI    SC,X'F0'           SEL CHNL DEV NO
      00F0
0008R 41F0                   BAL    15,WAIT            WAIT FOR NMTN=1
      00D4R
000CR DE00                   OC     MT,WRTEOF          EOF COMMAND
      0106R
0010R 9D13          BSY      SSR    SC,STAT            WAIT FOR SEL CHNL NOT BUSY
0012R 4280                   BTC    8,BSY
      0010R
0016R DE10                   OC     SC,STOP            RESET SEL CHNL ADDRESSES
      0105R
001AR DA10                   WD     SC,WLIMS           INIT SEL CHNL ADDRESSES
      00F8R
001ER DA10                   WD     SC,WLIMS+1
      00F9R
0022R DA10                   WD     SC,WLIMS+2
      00FAR
0026R DA10                   WD     SC,WLIMS+3
      00FBR
002AR 41F0                   BAL    15,WAIT            WAIT FOR NMTN=1
      00D4R
002ER DE00                   OC     MT,WRITE           MAG TAPE TO WRITE
      0100R
0032R DE10                   OC     SC,GOWRT           SEL CHNL TO 'GO'
      0102R
0036R 9D13          WAIT1    SSR    SC,STAT            WAIT FOR SEL CHNL NOT BUSY
0038R 4280                   BTC    8,WAIT1
      0036R
```

```
003CR 9D03      WAIT2    SSR   MT,STAT          WAIT FOR EOM=1
003ER 4320               BFC   2,WAIT2
      003CR
0042R C430               NHI   STAT,X'C1'       ERR,EOF,DU MUST =0
      00C1
0046R 4230               BNZ   SUBR1
      00E0R
004AR DB10               RD    SC,HIADRS        READ BACK SEL CHNL ADDR
      0108R
004ER DB10               RD    SC,LOADRS
      0109R
0052R 4870               LH    TEMP,HIADRS
      0108R
0056R 4570               CLH   TEMP,WLIMS+2
      00FAR
005AR 4230               BNE   SUBR2            HIADRS BAD
      00E4R
                *    BACKSPACE 1 RECORD
005ER 41F0               BAL   15,WAIT          WAIT FOR NMTN=1
      00D4R
0062R DE00               OC    MT,BKSP
      0104R
                *    READ THE RECORD JUST WRITTEN
0066R DE10               OC    SC,STOP          INIT SEL CHNL
      0105R
006AR DA10               WD    SC,RLIMS         SET UP SEL CHNL ADDRESSES
      00FCR
006ER DA10               WD    SC,RLIMS+1
      00FDR
0072R DA10               WD    SC,RLIMS+2
      00FER
0076R DA10               WD    SC,RLIMS+3
      00FFR
007AR 41F0               BAL   15,WAIT          WAIT FOR NMTN=1
      00D4R
007ER DE00               OC    MT,READ          MAG TAPE TO READ
      0101R
0082R DE10               OC    SC,GORD          SEL CHNL TO 'GO'
      0103R
0086R 9D13      WAIT3    SSR   SC,STAT          WAIT FOR SEL CHNL NOT BUSY
0088R 4280               BTC   8,WAIT3
      0086R
008CR 9D03      WAIT4    SSR   MT,STAT          WAIT FOR EOM=1
008ER 4320               BFC   2,WAIT4
      008CR
0092R C430               NHI   STAT,X'C1'       ERR,EOF,DU MUST=0
      00C1
0096R 4230               BNZ   SUBR3
      00E8R
009AR DB10               RD    SC,HIADRS        READ BACK SEL CHNL ADDR
      0108R
009ER DB10               RD    SC,LOADRS
      0109R
00A2R 4870               LH    TEMP,HIADRS
      0108R
```

```
00A6R 4570              CLH    TEMP,RLIMS+2
      00FER
00AAR 4230              BNE    SUBR4              HIADRS BAD
      00ECR
                *       COMPARE READ/WRITE BUFFERS
00AER 0744              XHR    INDEX,INDEX
00B0R C850              LHI    INCR,2
      0002
00B4R C860              LHI    LIMIT,X'AFF'
      0AFF
00B8R 4874      COMPAR  LH     TEMP,WBUFF(INDEX)
      0500
00BCR 4574              CLH    TEMP,RBUFF(INDEX)
      1000
00C0R 4230              BNE    SUBR5              DOES NOT MATCH
      00F0R
00C4R C140              BXLE   INDEX,COMPAR       LOOP
      00B8R
                *       WRITE END-OF-FILE MARK
00C8R 41F0              BAL    15,WAIT            WAIT FOR NMTN=1
      00D4R
00CCR DE00              OC     MT,WRTEOF          END-OF-FILE COMMAND
      0106R
00D0R 4300              B      DUMP               RESTART PROGRAM
      0000R
                *
                *
                *
00D4R 9D03      WAIT    SSR    MT,STAT
00D6R C430              NHI    STAT,X'10'
      0010
00DAR 4330              BZ     WAIT               WAIT FOR NMTN=1
      00D4R
00DER 030F              BR     15                 RETURN
                *       ENTRIES FOR ERROR LOGIC
00E0R 0000      SUBR1   DC     0,0                WRITE ERROR
      0000
00E4R 0000      SUBR2   DC     0,0                SC WRT ADRS BAD
      0000
00E8R 0000      SUBR3   DC     0,0                READ ERROR
      0000
00ECR 0000      SUBR4   DC     0,0                SC READ ADRS BAD
      0000
00F0R 0000      SUBR5   DC     0,0                COMPARE ERROR
      0000
00F4R 0000      SUBR6   DC     0,0                DEVICE UNAVAILABLE
      0000
                *       CONSTANTS, TEMPORARY STORAGE
00F8R 0500      WLIMS   DC     WBUFF
00FAR 0FFF              DC     WBUFF+X'AFF'
00FCR 1000      RLIMS   DC     RBUFF
00FER 1AFF              DC     RBUFF+X'AFF'
0500            WBUFF   EQU    X'500'
1000            RBUFF   EQU    X'1000'
```

```
0100R A2A1     WRITE   DC    X'A2A1'      DIS,FWD,WR
0101R          READ    EQU   *-1          DIS,FWD,RD
0102R 1030     GOWRT   DC    X'1030'      WRITE SEL CHNL
0103R          GORD    EQU   *-1          READ SEL CHNL
0104R 9108     BKSP    DC    X'9108'      BACKSPACE COMMAND
0105R          STOP    EQU   *-1          INIT SEL CHNL
0106R 3030     WRTEOF  DC    X'3030'      END-OF-FILE COMMAND
0108R          HIADRS  EQU   *
0109R          LOADRS  EQU   HIADRS+1
0108R                  END

BKSP     0104R
BSY      0010R
COMPAR   00B8R
DUMP     0000R
GORD     0103R
GOWRT    0102R
HIADRS   0108R
INCR     0005
INDEX    0004
LIMIT    0006
LOADRS   0109R
MT       0000
RBUFF    1000
READ     0101R
RETURN   0008
RLIMS    00FCR
SC       0001
STAT     0003
STOP     0105R
SUBR1    00E0R
SUBR2    00E4R
SUBR3    00E8R
SUBR4    00ECR
SUBR5    00F0R
SUBR6    00F4R
TEMP     0007
WAIT     00D4R
WAIT1    0036R
WAIT2    003CR
WAIT3    0086R
WAIT4    008CR
WBUFF    0500
WLIMS    00F8R
WRITE    0100R
WRTEOF   0106R
```

TABLE OF CONTENTS

LOADER DESCRIPTIONS

1. INTRODUCTION

Two loaders are available for loading standard format binary object tapes as generated by the Assembler or Hex Debut Program (CLUB). The three loaders are:

Loader	Program Part No.
The Relocating (REL) Loader	06-024
The General Loader	06-025

These loaders are compatible, but vary in size and the number of features provided. The General Loader is the most comprehensive, with facility for ENTRY and EXTRN handling, forward reference definitions, label processing, relocation, etc. The REL Loader handles program relocation, but all data must be defined; i.e. no ENTRY's, EXTRN's, or object tapes from 1-pass assemblies are handled. Tapes with undefined data can be loaded since the REL Loader will skip the appropriate items. Standard binary object tapes supplied by GE-PAC 30 have an M08 part designation for relative tapes, and an M09 part designation for absolute tapes.

Appendix 1 provides a summary of the important loader features.

2. FEATURES IN COMMON

The following features are common to both loaders.

1. All loaders are provided in relocatable bootstrap form (M10 part designation). These tapes are loaded with the 50 loader.

   NOTE

   The 30-2 instruction set includes the Autoload instruction which allows the use of the 68 Loader (a shorter form of the 50 Loader which allows leading blank tape to be bypassed). Appendix 2 contains listings of both loaders.

   Each loader tape contains a relocating bootstrap sequence followed by the actual loader in normal relocatable object format. The relocating bootstrap sequence causes the REL or General Loader to be loaded into the top of available core memory.

2. The input device for loading is definced by the Binary Input Device in the Device Definition Table in low core. Specifically, the halfword at X'78' is interpreted as follows:

78	0 7	8 15
	Dev No	Command

   This halfword for various devices is shown below:

Teletypewriter	0294
High Speed Paper Tape Reader	0399

3. When reading binary data from tape, blank tape and illegal characters are skipped. The set of legal tape characters is defined in Section 3.

4. Checksums and sequence numbers are checked after each binary record is read. Appropriate error halts are used when errors are detected.

5. The first location (ORG) of each loader is the starting location. Starting procedures are discussed further in Section 4.

6. While a tape is being read, the loaders output the data bytes to the console lights for confirmation of loader operations.

7. The console lights are used to identify the meaning of loader halts. The light patterns used are:

XX00	for normal end
XX0F	for checksum or sequence number error
XXFn	for improper loader control item where n is the 4-bit item

   Refer to Appendix 1 for a summary of improper control items

8. The loaders transfer to the program loaded, if specified on the object tape.

3. OBJECT TAPE FORMAT

Standard format binary object tapes are divided into records: records are separated by 12 blank characters. Each record contains 108 bytes of information. The first four bytes are organized as follows:

	0 15
Byte 1 and 2	Sequence Number
Byte 3 and 4	Checksum

The sequence numbers are negative integers -1, -2, -3, etc. represented in two's complement form. The first record in a program must have sequence number -1. Subsequent records must be in proper order to be loaded.

The checksum is an odd parity Exclusive OR sum of all words in the record, except itself, plus a word of all ONE's. When a checksum error is detected during input, the loaders halt with XX0F indicated on the console lights.

The remainder of the record is a sequence of items; an item is 4-bits or a half-byte. There are two types of items: control items and data items. There are 16 different control items, each of which is followed by a certain number (which might be zero) of data items.

The control items, and their meaning are listed on Table 1.

Each character punched on paper tape represents one item of information. The least significant four bits of the row are used for data, the most significant four bits control the ASR 33 zones. The zones have been selected to produce a non-printing set of ASCII characters and to avoid the characters XON, XOFF', TON, TOFF and WRU. Since each record consists of 108 bytes of memory and control data, there are 216 rows punched in the tape for each record. The rows punched and their hexadecimal equivalent are as listed on Table 2.

TABLE 1. CONTROL ITEM DEFINITIONS

Control Item	Meaning	number of data items following
0	Read next record	0
1	End of program	0
2	Define chain	0
3	Toggle abs/rel mode	0
4	Load transfer address	4
5	Load program address	4
6	Load reference address	4
7	Load definition address	4
8	Data, 2 bytes absolute	4
9	Data, 2 bytes relative	4
A	Data, 4 bytes absolute	8
B	Data, 4 bytes relative	8
C	Symbol, reference	12
D	Symbol, definition	12
E	Unused	0
F	Program Label	12

TABLE 2. TAPE CODES

ZONE	DATA	HEX DATA
1001	0000	0
1000	0001	1
1000	0010	2
1000	0011	3
1000	0100	4
1001	0101	5
1001	0110	6
1001	0111	7
1001	1000	8
1001	1001	9
1001	1010	A
1001	1011	B
1001	1100	C
1001	1101	D
1001	1110	E
1001	1111	F

The tape therefore appears as shown on Figure 1.

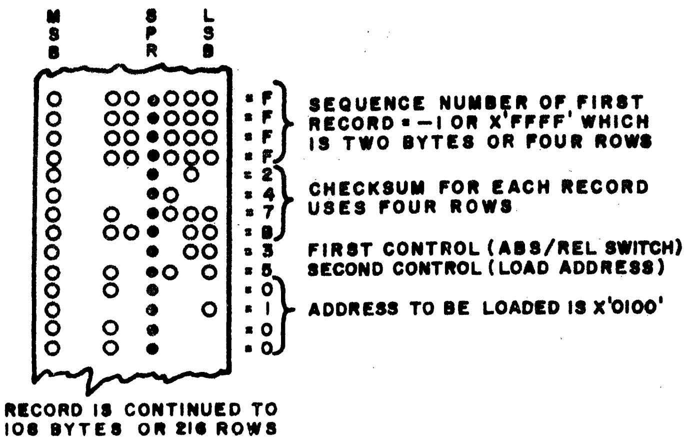


FIGURE 1. TAPE FORMAT

4. PROGRAM RELOCATION

The binary tapes generated by the Assembler can be absolute or relocatable, which can be loaded by either the REL Loader or the General Loader. With these loaders, a pointer called the BIAS identifies the first location (lowest address) to be used for a relocatable program.

When the REL or General Loader is executed at its starting address (ORG), the BIAS value is set to X'80'. This BIAS value is used during program loading to adjust any relocatable data values. Note that absolute data is stored at the absolute location specified for the data; absolute programs have no effect on BIAS. Relocatable programs are stored from the location indicated by BIAS upward into memory. After a program has been loaded, the BIAS value is adjusted to point to the next available location. To indicate that the load is complete, the loader halts with the Wait light illuminated, and with XX00 contained in the display register. At this time, the adjusted BIAS value is held in Register 0. This register can be examined by rotating the MODE CONTROL switch to HALT, rotating the REGISTER DISPLAY switch to R0/R1, and depressing EXECUTE.

If more programs are to be loaded, place the next tape in the reader, put the MODE CONTROL switch in RUN, and depress EXECUTE. This procedure starts the loader executing at ORG + 1A, which is the Continue Location. The continue operation uses the current value of BIAS, and does not reset it to X'80'. Multiple relocatable tapes are thus loaded one after another into adjacent areas of core memory.

If it is desired to load a relocatable program at an arbitrary point in core memory, it is necessary to redefine the BIAS value. To adjust the BIAS pointer, use the following procedure.

1. Change the halfword at ORG + A in the loader to the desired BIAS value.

2. Start the loader executing at ORG + 8, rather than the normal start or continue location.

Note that the value at ORG + A remains until changed to a new value. The loader can always be restarted at ORG + 8 which resets the BIAS to the value contained in ORG + A.

5. GENERAL LOADER FEATURES

In addition to the capabilities already discussed, the General Loader provides various features not available with the REL Loader.

5.1 BIAS Printout

At the start of every load operation, the General Loader types on the teletypewriter the current value of the BIAS pointer. This printout occurs prior to reading the first record of a new program, and the message is of the form

BIAS = XXXX

where the XXXX represents the current BIAS value in hexadecimal form.

5.2 Messages

Other messages which are typed on the teletypewriter are as indicated in Table 3.

5.3 ENTRY/EXTRN Handling

Programs generated by the assembler can use ENTRY's or EXTRN's to achieve cross-referencing and linkage with external programs. In this case, the object tape for these programs contains the symbolic names declared as ENTRY's or EXTRN's. The General Loader uses a symbol table to remember these names when a program is loaded. This symbol table builds downward in core memory from the origin (ORG) of the loader. Each entry in the loader symbol table requires 8 bytes of memory.

Since the loader symbol table is building downward into memory, and the programs being loaded are building upward into memory, the loader checks to see that the loading program does not over-write the symbol table. If the loading program requires data stored above the current bottom of the symbol table, a memory full message is generated, and the loader halts.

When the General Loader is initially entered into memory, the symbol table contains 3 entries which are global symbols relevant to the General Loader itself. These global symbols and their meanings are:

LOAD	This symbol represents the origin of the General Loader. Given this symbol, an external program can determine the start, continue, and bias-redefinition locations.
BIAS	This symbol represents the halfword in the loader which contains the current BIAS value.
CRNT	This symbol represents the halfword in the loader which contains a pointer to the current bottom of the symbol table. Given this pointer, an external program can test and/or alter the size of the loader symbol table. To clear the table, a program should load the halfword CRNT with the value LOAD -8. To clear the table of all symbols except the 3 global symbols, a program should load the halfword CRNT with the value LOAD -X'20'.

Note that no program can define an ENTRY point with the name LOAD, BIAS, or CRNT, because such a definition would conflict with the global symbols in the General Loader.

When the General Loader is executed at its start location (ORG) or its bias - redefinition location (ORG + 8), the symbol table is cleared of all names except the 3 global symbols. Executing the General Loader at its continue location (ORG + 1A) does not change the state of the symbol table.

TABLE 3. ERROR MESSAGES

Message	Meaning
CKSM ERR	A checksum error was detected after reading the previous record. Reposition the tape to the beginning of the record and push EXECUTE to reread the record.
SEQ NUM ERR	A sequence number error was detected after reading the previous error. Reposition the tape to the proper record and push EXECUTE to try again. This error usually occurs when the tape is improperly positioned following a checksum error.
MEMORY FULL	This message is caused by a conflict between the General Loader and the loading program. The program being loaded has not been loaded to conclusion. The alternatives are: A. Load fewer programs B. Make absolute tapes of the programs to be loaded and then use REL Loader which requires much less memory. C. Eliminate some EXTRNS and ENTRYS to reduce size of symbol table. D. Purchase more memory. Note that the General Loader cannot load programs above itself in memory.
NORMAL END	This case occurs when a program has successfully loaded and no END transfer address has been specified or if undefined external references remain. All undefined external references will be listed on the printer preceded by a U prior to printing the NORMAL END message. If a transfer address is specified and no undefined symbols remain, the Loader transfers directly to the address specified, and no NORMAL END message occurs.

TABLE 3. ERROR MESSAGES (Continued)	
Message	Meaning
LOAD ERR	This message results if a control item E is detected during load. Push EXECUTE to ignore the control item and proceed with the load. Note that this control item should not occur in general. This message, therefore, may be indicative that something is wrong. In this case, it is recommended that the loading procedure be restarted.

At the end of each program load, the symbol table is scanned for undefined symbols. Any undefined symbols are typed in the form

U XXXXXX

where XXXXXX is the symbol name. All such undefined names are printed preceding the normal end message. An undefined symbol results from the fact that the symbol was declared as an EXTRN in some program, and no program yet loaded has declared that same symbol as an ENTRY. As soon as some loading program declares that symbol as an ENTRY, the symbol becomes defined. If more than one program declares a symbol an ENTRY, the message

M XXXXXX

where XXXXXX is the symbol name, is typed at the time the multiple definition occurs. In this case, the first value defined remains in the symbol table, and the second definition value is ignored.

At the end of each program load, the loader transfers immediately to the program loaded, only if a transfer address is specified on the tape, and if the symbol table contains no undefined symbols. If any symbols in the table are undefined at the end of a load, those symbols are listed, NORMAL END is printed, and the loader halts, waiting to load the next program.

5.4 Forward Reference Definitions

Program object tapes generated by 1-pass assemblies involve forward references to symbols which are defined later in the program. The General Loader uses a chaining procedure for satisfying any forward references at the time the symbol definition is encountered. Therefore, 1-pass assemblies are possible, provided the General Loader is used to load the object tape. Note that the REL Loader cannot perform this forward reference definition function.

An example of a forward reference in a program is:

```
        OPT     PASS1, PUNCH
        LH      3, SAM
        BR      5
SAM     DC      3
        END
```

In this case, the reference to SAM occurs before SAM is defined. There are several restrictions on the use of forward references during 1-pass assemblies, and on the use of symbols which are ENTRY's or EXTRN's for the program to be loaded properly. The restrictions are:

1. Such symbols must not be combined in arithmetic expressions such as

```
          LH      3,SAM+2
```

2. Such symbols must not be used in the R1 or R2 field for an instruction such as

```
          LH      3,2(SAM)
```

3. Such symbols must not be used with assembler pseudo-ops such as DO, EQU, END, etc; for example

```
          DO      SAM
```

5.5 Label Handling

Programs generated by the assembler can be labelled through the use of the OPT Command such as:

```
OPT       PASS2,PUNCH,LAB=ABCDEF
```

The program label can be up to 6 characters. The first character must be a letter; subsequent characters can be letters or digits. The object tape, in this case, contains the program label in symbolic form. When the General Loader detects a program label, the label is typed in the form

```
LABEL = ABCDEF
```

If object tapes which contain labels are loaded by the REL Loader, an error halt occurs with XXFF on the Display Panel. In this case, push EXECUTE to proceed with the load.

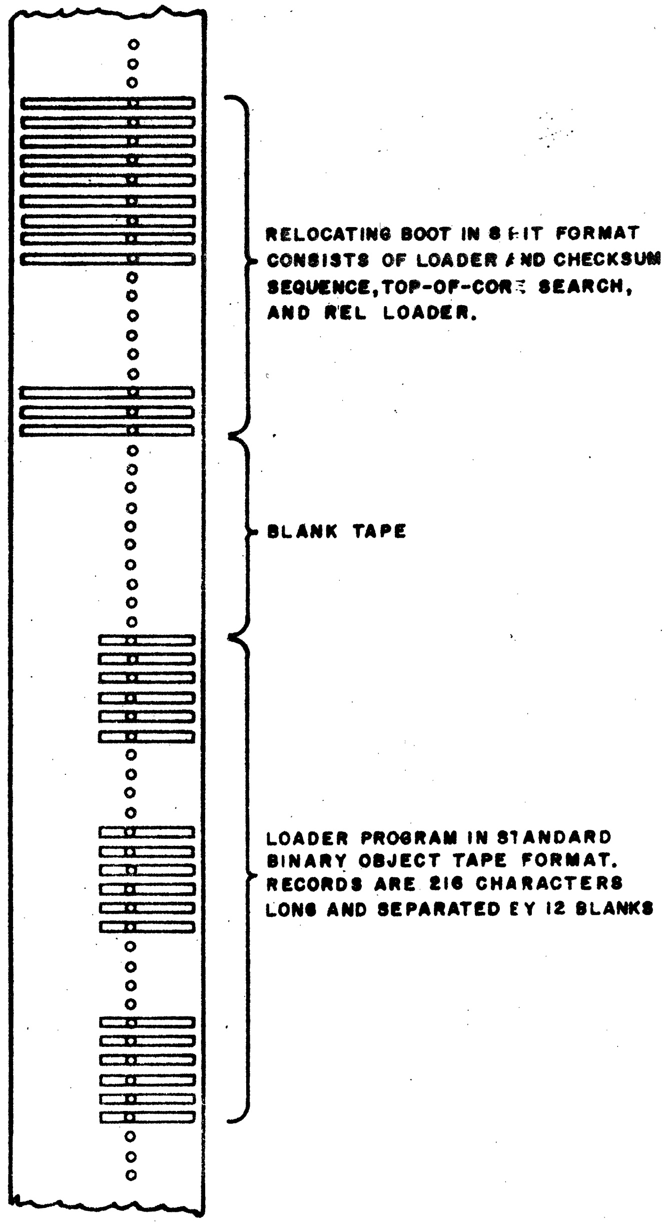


FIGURE 2. LOADER TAPE FORMAT

6. LOADER TAPE FORMAT

The loaders are provided in a relocating bootstrap form. The format of the tapes is illustrated in Figure 2. The tapes consist of two segments: the boot portion in 8-bit format, and the actual loader in standard binary object tape format. When the tape is loaded using the 8-bit loader at X'50', the following sequence of events takes place.

1. The 8-bit loader at X'50' reads another loader into X'80' to X'CF' and transfers to X'80'.

2. The program at X'80' reads the balance of the 8-bit data into X'DO' to X'34F', which includes a REL Loader.

3. An arithmetic checksum on the information from X'DO' to X'34F' is then tested. If the checksum is correct, the process continues. If the checksum is not correct, the tape is stopped and the program halts.

4. The top-of-memory is then determined with a search technique, and the REL Loader BIAS is set a fixed distance from the top-of-core. The REL Loader is placed X'300' from the top-of-core. The General Loader is placed X'600' from the top.

5. The REL Loader then reads the loader program, which is in relocatable form, and relocates it into the top portion of core memory.

6. The REL Loader computes checksums on each record, and halts whenever a checksum error is detected. In this case, reposition the tape to the previous record gap and push EXECUTE to re-read the record.

7. When the entire tape has been loaded, the Processor halts with the Wait light illuminated. Press EXECUTE to transfer control to the loader just loaded.

This sequence requires that the proper 50 Sequence is used, including the Binary Input Device Definition in X'78'. The 50 Sequence is shown in Appendix 2. Listings for the relocating boot sequence, including the REL Loader, are shown in Appendix 3.

Since the loader portion of each tape is a relocatable object tape, it is possible to put the loaders anywhere in memory. This can be done by using a bootstrap load to get the REL or General Loader into the top of memory. The BIAS can then be adjusted and any loader can then be relocated to any arbitrary point in memory. Once relocated, CLUB can be used to dump an absolute tape of the loader in that location.

CAUTION

> Note that when loading the bootstrap loader tapes, memory from X'80' to X'3BF' is used. Any programs in this area of memory will be overwritten.

7. OPERATION

The steps required to load and operate the loaders are summarized below.

1. Manually enter the 50 sequence into memory if it is not already there. Specify the device to be used for loading X'78', the Binary Input Device definition. See Appendix 2 for a listing of the 50 sequence.

2. Place the loader tape in the tape reader, with the first data character over the read fingers, or photo diodes. If program linkage is required, or one-pass object tapes are to be loaded, the General Loader must be used. Otherwise, the REL Loader can be used.

3. Enter X'0050' into the console switches, select ADRS Mode and depress EXECUTE.

4. Depress INITIALIZE. Select RUN Mode, and depress EXECUTE.

5. If an ASR 33 Teletypewriter is being used as the input device, it is necessary to toggle the reader switch to START, which starts the tape moving. If an ASR 35 Teletypewriter is in use, the mode switch should be in the T position, and the reader switch should be put in RUN to start the tape. If a high speed paper tape reader is in use, the tape will start by itself.

6. If no input errors occur, the entire tape will be read to the end, at which time the Processor will halt with XX00 in the console lights.

7. If checksum errors are detected during tape input, the tape will stop and the Processor will halt with XX0F contained in the console lights. When this occurs, reposition the tape to the previous record gap, and push EXECUTE to re-read the record. If the error halt occurs after the first record on the tape, restart the entire load procedure.

8. Put the tape to be loaded into the tape reader. If the tape to be loaded is relocatable, and a specific BIAS value is required, enter the BIAS value into ORG + A, and set the starting address to ORG + 8. If the tape to be loaded is absolute, or if the current BIAS value is satisfactory, set the starting address to ORG. Depress INITIALIZE. Select RUN Mode and depress EXECUTE.

9. If improper control items are detected during the load, the tape will stop, and the Processor will halt with XXFn contained in the console lights where n is the bad control item. When this occurs, it must be determined if the right loader is being used. That is, if the object tape involves ENTRY's or EXTRN's or forward references, the General Loader must be used. If the loader is appropriate, and the tape is proper, push EXECUTE to skip the improper data and proceed with the load.

10. If checksum errors are detected during the load, the tape will stop and the Processor will halt with the Wait light illuminated and XX0F contained in the console lights. Reposition the tape to the previous EXECUTE to re-read the record.

11. When the load is complete, the tape will stop. If no transfer address is specified on the tape, the Processor will halt with the XX00 contained in the display register. If a transfer address is specified, the REL Loader will transfer directly to the location specified. The General Loader transfers only if the symbol table contains no undefined symbols.

12. If more tapes are to be loaded, return to step 8 and repeat the process. This loading process is summarized in Figure 3.

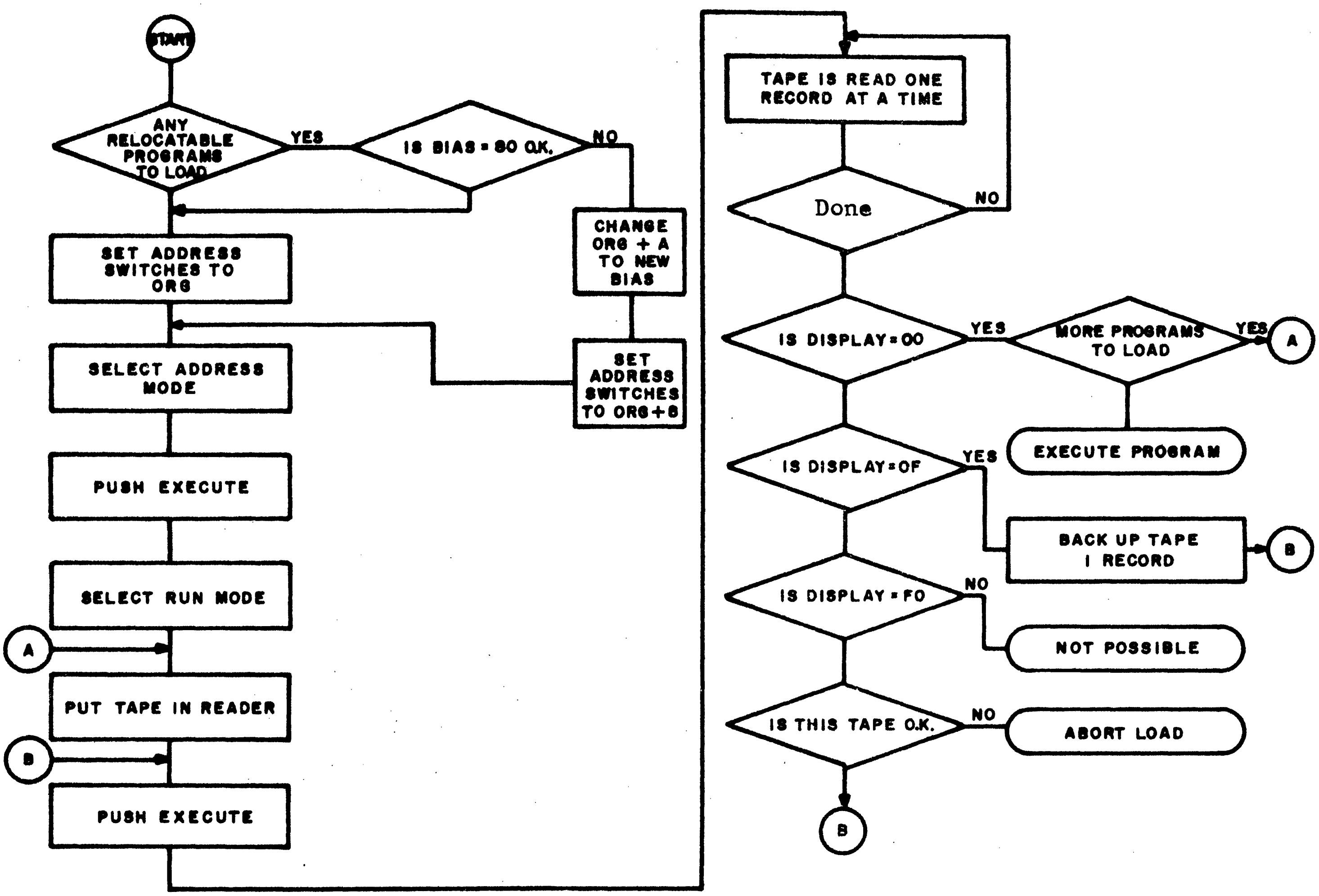


FIGURE 3. OPERATING PROCEDURE

BOOTSTRAP PROGRAMS AND PROCEDURES

1. INTRODUCTION

Certain absolute programs, such as FORTRAN and the Assembler are provided in absolute bootstrap form (M10 designation). This tape format provides the following features.

1. The tapes are self-loading, requiring only the 50 or 68 Sequence to load the tape into memory.
2. The loading time is minimized.
3. The tape is organized in blocks. This format enables error checks to be made while the tape loads.
4. At the completion of the load, control is transferred to the loaded program.

This tape format is appropriate for absolute programs only. Relocatable programs, such as the standard loaders, are provided in other formats which permit relocatability. This document discusses programs and procedures associated with absolute bootstrap tapes.

2. GENERAL DESCRIPTION

Absolute bootstrap tapes consist of two major segments:

1. The bootstrap portion, in 8-bit format, which contains a front-end intermediate loader, and a Fast Format Loader.
2. The actual program to be loaded, represented in fast format.

The two major segments of the tape are separated by several inches of blank tape. See Figure 1.

When an absolute bootstrap tape is loaded, the following sequence of events takes place.

1. The 8-bit loader at X'50' or X'68' reads the front-end intermediate loader into locations from X'80' to X'CF', and then transfers to X'80'.
2. The front-end intermediate loader at X'80' reads the Fast Format Loader, which is the balance of the 8-bit data, into locations from X'1D00' to X'1E01', and computes an arithmetic checksum in the process.
3. The arithmetic checksum on the information from X'1D00' to X'1E01' is then tested. If the checksum is not correct, the tape is stopped and the program halts. If the checksum is correct, the process continues, and control is transferred to X'1D00', the starting location of the Fast Format Loader.

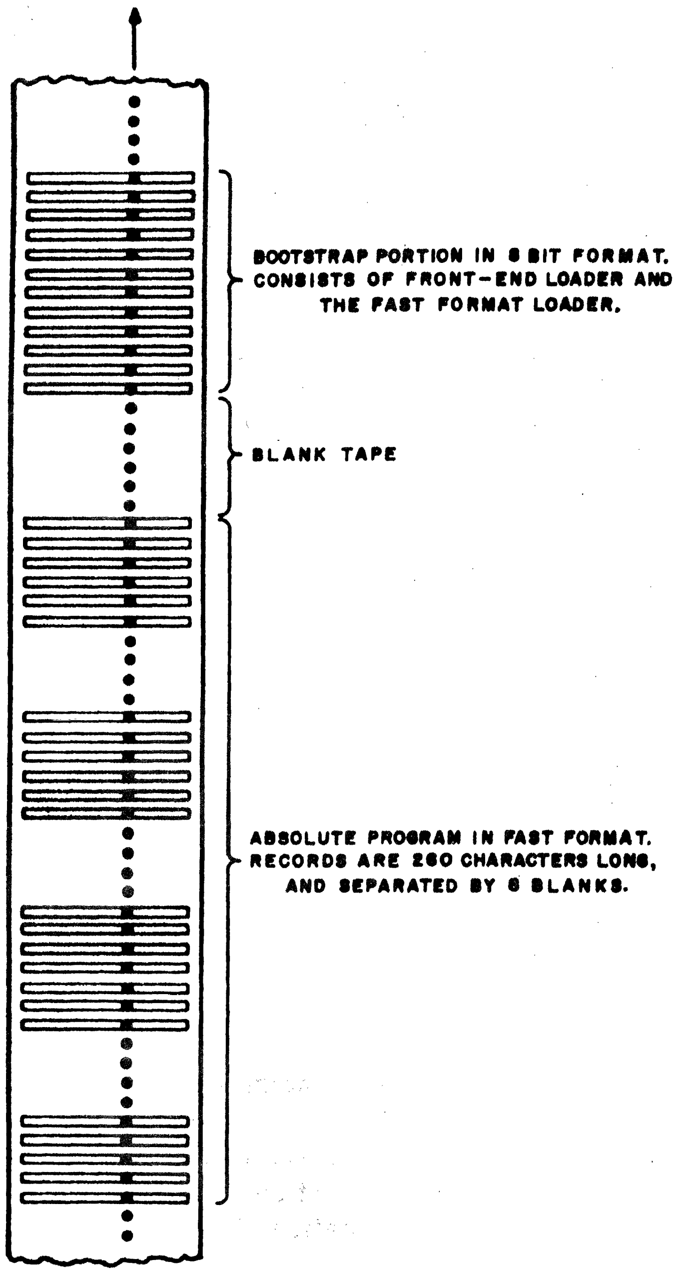


Figure 1. Absolute Bootstrap Format

4. The Fast Format Loader then reads the program. At the completion of the load, the Fast Format Loader transfers to the loaded program.

This operation requires that the proper 50 Sequence is used, including the Binary Device Definition at X'78'. Listing of both sequences are shown in Appendix 1. A listing for the front-end routine is in Appendix 2. Fast Format Loader and Puncher listings are in Appendices 3 and 4.

3. FAST FORMAT

Fast format tapes are organized into 260 character records, separated by 8 blank characters. Each character on the tape contains 8-bits, or 1 byte of information. The first two characters in each record define a sequence number. Sequence numbers are negative integers -1, -2, -3, -4, etc, represented in two's complement binary form. The third and fourth characters in each record define an arithmetic checksum. This checksum is generated when the tape is punched, and checked when the tape is loaded.

The first record of a fast format tape contains 3 addresses immediately following the checksum characters. Each address takes 2 characters. The addresses, in the order in which they appear on the tape, are as follows:

1. Starting (lowest) address for the program.

2. Final (highest) address for the program.

3. Transfer address at the end-of-load.

The remainder of the first record, and the entire contents of all subsequent records, are absolute 8-bit data bytes for the program.

The Fast Format Loader checks the checksum during loading. If an error is detected, the tape is stopped, and the Processor halts with XX0F displayed on the console lights. In this case, reposition the tape to the previous record gap and depress EXECUTE to reread the previous record.

A listing for the Fast Format Loader is provided in Appendix 3.

4. TAPE PREPARATION

Bootstrap tapes are produced in two steps: the first step generates the bootstrap portion, which includes the Fast Format Loader; the second step reduces the desired program to fast format.

The first step is accomplished as follows:

1. Load the Absolute Boot Front End from X'80' to X'CF'.
2. Load the Fast Format Loader in the appropriate location.
3. Compute the arithmetic checksum of the Fast Format Loader and enter the computed checksum into the location at X'A2'.
4. Adjust the PLOW, PHGM, and PSTRT values in the Front-End routine.
5. Punch the Front-End routine and the Fast Format Loader in 8-bit format in one consecutive block on tape, using the CLUB Q directive.

The second step includes the following:

1. Load the object program into memory.
2. Load the Fast Format Puncher.
3. Set up the necessary parameters for the Fast Format Puncher program. (See Appendix 4.)
4. Punch the program in fast format.

APPENDIX 1
SUMMARY

The following Console indications are common to all loaders.

Console Lights	Condition	Comment
XX00	Normal End	Load complete
XX0F	Input Error	A checksum or sequence number error was detected after reading the last record. Reposition tape and push EXECUTE to reread.
XXFn	Load Error	Improper control item detected where n is the bad item. Push EXECUTE to ignore the data and continue. Refer to Section 3 for a definition of loader control items.

	REL Loader	General Loader
Starting address after boot load*	nD00	nA00
Restart address in general	ORG	ORG
Bias define address	ORG + 8	ORG + 8
Bias definition value	ORG + A	ORG + A
Continue address	ORG + 1A	ORG + 1A
Loader size	X'2B6'	X'538'
Illegal control items	2, 6, 7, C, D, E, F	E
Ignored control items	--------	------

*n = 0, 1, 2, 3, ... for memory sizes 4K, 8K, 12K, 16K, etc.

General Loader Restart - sets BIAS to X'80'
- clears symbol table of all but LOAD, BIAS, CRNT
- clears any transfer address

APPENDIX 2
50, 68 SEQUENCE

```
                      *
                      *
                      *    50 SEQUENCE LOADER
                      *    FOR ALL GE-PAC 30 PROCESSORS
                      *
0050                          ORG    X'50'
0050   C820           LOAD    LHI    2,X'80'          LOADS TAPE FROM X'80'
       0080
0054   C830                   LHI    3,1              THRU X'CF'
       0001
0058   C840                   LHI    4,X'CF'
       00CF
005C   D3A0                   LB     10,BINDV         NOTE THAT LOCATION X'5A'
       0078
0060   DEA0                   OC     10,BINDV+1       MUST BE CHANGED FOR ALL
       0079
0064   9DAE           SENSE   SSR    10,14            M14 TEST PROGRAM TAPES
0066   08EE                   LHR    14,14
0068   4230                   BTC    3,SENSE
       0064
006C   DBA2                   RD     10,0(2)
       0000
0070   C120                   BXLE   2,SENSE
       0064
0074   4300                   B      X'80'
       0080
0078   0294           BINDV   DC     X'0294'          DEVICE DEFINITIONS ARE
007A   0298           BOUTDV  DC     X'0298'          FOR TTY
007C   0294           SINDV   DC     X'0294'
007E   0298           SOUTDV  DC     X'0298'
                      *
                      *
                      *    68 SEQUENCE LOADER
                      *    FOR 30-2 PROCESSORS ONLY
                      *
0068                          ORG    X'68'
0068   C830                   LHI    3,1              LOADS TAPE FROM X'80' THRU
       0001
006C   D3A0                   LB     10,BINDV         X'CF'.  LOCATION X'72'
       0078
0070   D500                   AL     0,X'CF'          MUST BE CHANGED FOR ALL
       00CF
0074   4300                   B      X'80'            M14 TEST PROGRAM TAPES
       0080
                      *    DEVICE DEFINITIONS ARE SAME AS FOR 50 SEQUENCE
                      *
                      *
                      *
                      * HIGH SPEED PAPER TAPE READER= NN99 (BINDV,SINDV)
                      * HIGH SPEED PAPER TAPE PUNCH= NN9A (BOUTDV)
                      * CARD READER= NN20 (SINDV)
                      *    WHERE NN= DEVICE NUMBER
                      *
```

APPENDIX 3

LISTING OF THE RELOCATING BOOTSTRAP

This Appendix consists of two listings:

1. The Bootstrap Front End at X'80'.

2. The REL Loader at X'108'.

```
                        OPT        PASS2, PRINT, PUNCH, STOP

                  *
                  * BOOTSTRAP FRONT END
                  * CONSISTS OF 8-BIT LOADER AND CHECKSUM LOOP.
                  * APPEARS IN 8-BIT FORMAT.
                  * ASSUMES THAT-REG 3= 1
                  * ASSUMES THAT-REG 1Ø = DEV NUMBER
                  * ASSUMES THAT-TAPE IS MOVING
                  * ASSUMES THAT-ALL CHARS ON TAPE ARE LEGAL
                  * IF CHECKSUM ERROR DETECTED, LOADER STOPS
                  * THE TAPE AND HALTS. AFTER PROCESSOR HALTS.
                  * RESTART BOOTSTRAP LOAD, OR PUSH EXECUTE TO
                  * IGNORE CHECKSUM ERROR AND CONTINUE EXECUTION.
                  *
                  *
ØØ8Ø                        ORG        X'8Ø'
                  *
                  *
ØØDØ              PLOW      EQU        X'DØ'
Ø34F              PHGH      EQU        X'34F'
ØØDØ              PSTRT     EQU        X'DØ'
C493              CKSUM     EQU        X'C493'
                  *
                  *
ØØ8Ø    C82Ø      START     LHI        2, PLOW        SET PROGRAM LIMITS
        ØØDØ
ØØ84    C84Ø                LHI        4, PHGH        R3 MUST HOLD 1
        Ø34F
ØØ88    Ø755                XHR        5, 5           CLEAR R5 FOR CHECKSUM
ØØ8A    9DAE      SENSE     SSR        1Ø, 14         SENSE STATUS
ØØ8C    Ø8EE                LHR        14, 14         TAPE SHOULD BE MOVING
ØØ8E    423Ø                BNZ        SENSE
        ØØ8A
ØØ92    9BAB                RDR        1Ø, 11         READ ONE CHAR
ØØ94    ØA5B                AHR        5, 11          BUMP CHECKSUM
ØØ96    9A3B                WDR        3, 11          DISPLAY CHAR
ØØ98    D2B2                STB        11, Ø (2)      STORE CHAR
        ØØØØ
ØØ9C    C12Ø                BXLE       2, SENSE
        ØØ8A
                  *
                  *
ØØAØ    C55Ø                CLHI       5, CKSUM       CHECK THE CHECKSUM
        C493
ØØA4    433Ø                BE         PSTRT          GO TO PROGRAM IF OK
        ØØDØ
ØØA8    DEAØ                OC         1Ø, STOP       STOP TAPE IF NOT OK
        ØØCB
ØØAC    C5AØ                CLHI       1Ø, 2
        ØØØ2
ØØBØ    423Ø                BNE        HALT
        ØØC2
ØØB4    OOCH                OC         1Ø, TWRT       IF DEV NO = 2
```

```
ØØB8    9DAE              SSR     1Ø,14          PUT TTY IN WRITE MODE
ØØBA    429Ø              BTC     9,*-2          AND ISSUE XOFF
        ØØB8
ØØBE    DAAØ              WD      1Ø,XOFF
        ØØCD
ØØC2    C2ØØ    HALT      LPSW    *+4            HALT
        ØØC6
ØØC6    8ØØØ              DC      X'8ØØØ',PSTRT  GO TO PROGRAM ON EXECUTE
        ØØØDØ
                *
                *
ØØCA    98A9    TWRT      DC      X'98A9'        WRITE,STOP COMMANDS
ØØCB            STOP      EQU     *-1
ØØCC    9193    XON       DC      X'9193'        XON,XOFF CHARACTERS
ØØCD            XOFF      EQU     *-1
ØØCE    ØØØØ              DC      Ø              FILLER
                *
                *
                * TOP-OF-CORE SEARCH
                * REQUIRES LENGTH PARAMETER
                * FOR REL LOADER SET-UP
                *
ØØDØ    C81Ø    SRCH      LHI     1,X'4ØØ'       SET PNTR TO 1K
        Ø4ØØ
ØØD4    C82Ø              LHI     2,X'FFFF'      TEST DATA
        FFFF
ØØD8    4831    SCAN      LH      3,Ø(1)         SAVE CURRENT DATA
        ØØØØ
ØØDC    4Ø21              STH     2,Ø(1)         WRITE TEST DATA
        ØØØØ
ØØEØ    4841              LH      4,Ø(1)         READ TEST DATA
        ØØØØ
ØØE4    433Ø              BZ      FOUND          IF ZERO, TOP IS FOUND
        ØØFC
ØØE8    4Ø31              STH     3,Ø(1)         RESTORE CURRENT DATA
        ØØØØ
ØØEC    CA1Ø              AHI     1,X'4ØØ'       BUMP PNTR BY 1K
        Ø4ØØ
ØØFØ    423Ø              BNZ     SCAN
        ØØD8
ØØF4    C2ØØ              LPSW    LOST           PNTR ZERO = TROUBLE
        ØØF8
ØØF8    8ØØØ    LOST      DC      X'8ØØØ',A(SRCH)
        ØØDØ
ØØFC    CB1Ø    FOUND     SHI     1,LENGTH       ADJUST FOR PROG LENGTH
        Ø3ØØ
Ø1ØØ    4Ø1Ø              STH     1,RELORG+1Ø    SET UP REL LOADER
        Ø112
Ø1Ø4    43ØØ              B       RELORG+8       JMP TO REL LOADER
        811Ø
                *
                *
Ø3ØØ            LENGTH    EQU     X'3ØØ'
Ø1Ø8            RELORG    EQU     *
Ø1Ø8                      END
```

CKSUM	C493
FOUND	ØØFC
HALT	ØØC2
LENGTH	Ø3ØØ
LOST	ØØF8
PHGH	Ø34F
PLOW	ØØDØ
PSTRT	ØØDØ
RELORG	Ø1Ø8
SCAN	ØØD8
SENSE	ØØ8A
SRCH	ØØDØ
START	ØØ8Ø
STOP	ØØCB
TWRT	ØØCA
XOFF	ØØCD
XON	ØØCC

```
                         OPT   PASS2,PRINT,PUNCH,STOP
                *
                * BASIC REL LOADER
                * 06-024
                *
0000            R0       EQU   0
0001            R1       EQU   1
0002            R2       EQU   2
0003            R3       EQU   3
0004            BYTE     EQU   4
0005            PICK     EQU   5
0006            SEQNUM   EQU   6
0007            ONE      EQU   7
0008            TWO      EQU   8
0009            FOUR     EQU   9
000A            A        EQU   10
000B            B        EQU   11
000C            C        EQU   12
000D            D        EQU   13
000E            E        EQU   14
000F            ABSF     EQU   15
0078            BINDV    EQU   X'78'
                *
                *
0000R C8A0      START    LHI   A,X'80'             INITIALIZE LOC,BIAS
      0080
0004R 4300               B     *+8
      000CR
0008R C8A0      REDEF    LHI   A,X'80'             BIAS REDEFINITION
      0080
000CR 40A0               STH   A,LOC
      0242R
0010R 40A0               STH   A,BIAS
      0246R
0014R 0BAA               SHR   A,A                 CLEAR EXECUTE ADRS
0016R 40A0               STH   A,LOCX
      0240R
001AR 0B66      CONT     SHR   SEQNUM,SEQNUM       CLEAR SEQNUM
001CR 0BFF               SHR   ABSF,ABSF           SET REL MODE
001ER C870               LHI   ONE,1               SET CONSTANTS 1,2,4
      0001
0022R C880               LHI   TWO,2
      0002
0026R C890               LHI   FOUR,4
      0004
002AR 0B67      NEXT     SHR   SEQNUM,ONE          DECR SEQ COUNT
002CR 4120               BAL   R2,INPUT            INPUT ONE RECORD
      01A4R
0030R C8C0               LHI   C,X'FFFF'
      FFFF
0034R 47C0               XH    C,BUFF
      0248R
0038R C8A0               LHI   A,102
```

```
       0066
003CR  47CA    CKIT    XH     C,BUFF+4(A)      COMPUTE CHECKSUM
       024CR
0040R  0BA8            SHR    A,TWO
0042R  4380            BNL    CKIT
       003CR
0046R  45C0            CLH    C,BUFF+2         COMPARE CHECKSUM
       024AR
004AR  4230            BNE    ERROR
       0098R
004ER  4560            CLH    SEQNUM,BUFF      COMPARE TO SEQ NUM
       0248R
0052R  4230            BNE    ERROR
       0098R
0056R  C850            LHI    PICK,BUFF+4      ADJUST PICK,BYTE
       024CR
005AR  C840            LHI    BYTE,12
       000C
               *
005ER  C550    LOOP    CLHI   PICK,BUFF+108    TEST IF RECORD DONE
       02B4R
0062R  4380            BNL    NEXT
       002AR
0066R  48A5            LH     A,0(PICK)        EXTRACT NEXT COMMAND
       0000
006AR  4110            BAL    R1,EXTR
       018ER
006ER  08EA            LHR    E,A              SAVE CONTROL BYTE
0070R  0AAA            AHR    A,A
0072R  48BA            LH     B,JUMP(A)        GO TO COMMAND ROUTINE
       0078R
0076R  030B            BR     B
               *
0078R  002AR   JUMP    DC     NEXT,END,ERR0,FLIP
       00A6R
       00DAR
       00E8R
0080R  0100R           DC     LDX,LDL,ERR1,ERR1
       010CR
       00D6R
       00D6R
0088R  0118R           DC     UNAB,UNRL,DUAB,DURL
       0120R
       013AR
       0150R
0090R  00CER           DC     ERR3,ERR3,ERR0,ERR3
       00CER
       00DAR
       00CER
               *
0098R  C8A0    ERROR   LHI    A,X'000F'        DISPLAY X'0F' TO
       000F
009CR  9A7A            WDR    ONE,A            SHOW INPUT ERROR
009ER  C200            LPSW   *+4
```

```
        00A2R
00A2R 8000               DC     X'8000',A(NEXT+2)
      002CR
              *
00A6R 48A0   END         LH     A,LOC            END OF PROGRAM
      0242R
00AAR 08FF               LHR    ABSF,ABSF        UPDATE BIAS WITH
00ACR 4330               BZ     FIXB             THE REL LOC COUNTER
      00B4R
00B0R 48A0               LH     A,LOC+2
      0244R
00B4R 40A0   FIXB        STH    A,BIAS
      0246R
00B8R 48A0               LH     A,LOCX           JUMP TO PROGRAM
      0240R
00BCR 423A               BNZ    0(A)             IF LOCX IS NOT ZERO
      0000
00C0R 9A7A               WDR    ONE,A            DIAPLAY 00 TO
00C2R 4800               LH     R0,BIAS          SHOW NORMAL END
      0246R
00C6R C200               LPSW   *+4              LEAVE BIAS IN R0
      00CAR
00CAR 8000               DC     X'8000',A(CONT)  HALT
      001AR
              *
00CER 4120   ERR3        BAL    R2,WORD          SKIP OVER ANY
      0176R
00D2R 4120   ERR2        BAL    R2,WORD          REF,DEF,CHAIN,ETC
      0176R
00D6R 4120   ERR1        BAL    R2,WORD
      0176R
00DAR C6E0   ERR0        OHI    E,X'F0'          HALT AND DISPLAY
      00F0
00DER 9A7E               WDR    ONE,E            BAD CONTROL BYTE
00E0R C200               LPSW   *+4
      00E4R
00E4R 8000               DC     X'8000',LOOP
      005ER
              *
00E8R C7F0   FLIP        XHI    ABSF,X'FFFF'     FLIP THE ABS FLAG
      FFFF
00ECR 48A0               LH     A,LOC            FLIP LOC COUNTERS
      0242R
00F0R 48B0               LH     B,LOC+2
      0244R
00F4R 40A0               STH    A,LOC+2
      0244R
00F8R 40B0               STH    B,LOC
      0242R
00FCR 4300               B      LOOP
      005ER
              *
0100R 4130   LDX         BAL    R3,GETT          SET EXECUTION ADRS
      0166R
```

```
0104R 40D0             STH    D,LOCX
      0240R
0108R 4300             B      LOOP
      005ER
                 *
010CR 4130       LDL   BAL    R3,GETT            SET LOAD LOCATION
      0166R
0110R 40D0             STH    D,LOC
      0242R
0114R 4300             B      LOOP
      005ER
                 *
0118R 4120       UNAB  BAL    R2,WORD            LOAD 2 BYTES ABS
      0176R
011CR 4300             B      UNRX
      0128R
0120R 4120       UNRL  BAL    R2,WORD            LOAD 2 BYTES REL
      0176R
0124R 4AD0             AH     D,BIAS
      0246R
0128R 48C0       UNRX  LH     C,LOC
      0242R
012CR 40DC             STH    D,0(C)
      0000
0130R 0AC8             AHR    C,TWO              BUMP LOAD LOCATION
0132R 40C0             STH    C,LOC
      0242R
0136R 4300             B      LOOP
      005ER
                 *
013AR 4120       DUAB  BAL    R2,WORD            LOAD 4 BYTES ABS
      0176R
013ER 48C0             LH     C,LOC
      0242R
0142R 40DC             STH    D,0(C)
      0000
0146R 0AC8             AHR    C,TWO
0148R 40C0             STH    C,LOC
      0242R
014CR 4300             B      UNAB
      0118R
                 *
0150R 4120       DURL  BAL    R2,WORD            LOAD 4 BYTES REL
      0176R
0154R 48C0             LH     C,LOC
      0242R
0158R 40DC             STH    D,0(C)
      0000
015CR 0AC8             AHR    C,TWO
015ER 40C0             STH    C,LOC
      0242R
0162R 4300             B      UNRL
      0120R
                 *
```

```
Ø166R 412Ø      GETT     BAL    R2,WORD          GET 2 BYTES OF DATA
      Ø176R
Ø16AR Ø8FF               LHR    ABSF,ABSF        AND ADD BIAS TO IT
Ø16CR 4233               BNZ    Ø(R3)            IF IN REL MODE
      ØØØØ
Ø17ØR 4ADØ               AH     D,BIAS
      Ø246R
Ø174R Ø3Ø3               BR     R3
                *
Ø176R Ø8C9      WORD     LHR    C,FOUR           ASSEMBLE 1 WORD OR
Ø178R 48A5      WORD1    LH     A,Ø(PICK)        TWO BYTES OF DATA
      ØØØØ
Ø17CR 411Ø               BAL    R1,EXTR          INTO REG D.
      Ø18ER
Ø18ØR CDDØ               SLHL   D,4
      ØØØ4
Ø184R Ø6DA               OHR    D,A
Ø186R ØBC7               SHR    C,ONE
Ø188R 423Ø               BNZ    WORD1
      Ø178R
Ø18CR Ø3Ø2               BR     R2
                *
Ø18ER CCA4      EXTR     SRHL   A,Ø(BYTE)        EXTRACT ONE FOUR BIT
      ØØØØ
Ø192R C4AØ               NHI    A,X'F'           BYTE FROM THE DATA
      ØØØF
Ø196R ØB49               SHR    BYTE,FOUR        IN REG A.
Ø198R 4311               BNM    Ø(R1)
      ØØØØ
Ø19CR C84Ø               LHI    BYTE,12          UPDATE PICK AND BYTE
      ØØØC
Ø1AØR ØA58               AHR    PICK,TWO
Ø1A2R Ø3Ø1               BR     R1
                *
Ø1A4R D3DØ      INPUT    LB     D,BINDV          PICK UP DEV NUMBER
      ØØ78
Ø1A8R Ø5D8               CLHR   D,TWO
Ø1AAR 423Ø               BNE    IN2
      Ø1BCR
Ø1AER DEDØ               OC     D,TWRT           IF TTY,SET WRITE MODE
      Ø2Ø2R
Ø1B2R 9DDE      IN1      SSR    D,E              AND OUTPUT XON
Ø1B4R 429Ø               BTC    9,IN1
      Ø1B2R
Ø1B8R DADØ               WD     D,XON
      Ø2Ø4R
Ø1BCR DEDØ      IN2      OC     D,BINDV+1        START DEVICE
      ØØ79
Ø1CØR C8AØ               LHI    A,BUFF           SET BUFF POINTER
      Ø248R
Ø1C4R 411Ø      IN3      BAL    R1,CHAR          GET 2 CHARS AND
      Ø2Ø6R
Ø1C8R CDBØ               SLHL   B,4              ASSEMBLE 8-BIT BYTE
      ØØØ4
```

```
01CCR 08CB              LHR   C,B
01CER 4110              BAL   R1,CHAR
      0206R
01D2R C4B0              NHI   B,X'F'
      000F
01D6R 06CB              OHR   C,B
01D8R D2CA              STB   C,0(A)          STORE BYTE
      0000
01DCR 0AA7              AHR   A,ONE
01DER C5A0              CLHI  A,BUFF+108      READ 108 BYTES
      02B4R
01E2R 4280              BL    IN3
      01C4R
01E6R 05D8              CLHR  D,TWO           TEST IF TTY
01E8R 4330              BE    IN4
      01F2R
01ECR DED0              OC    D,STOP          STOP DEVICE
      0203R
01F0R 0302              BR    R2
01F2R DED0        IN4   OC    D,TWRT          SET WRITE MODE AND
      0202R
01F6R 9DDE        IN5   SSR   D,E             ISSUE XOFF
01F8R 4290              BTC   9,IN5
      01F6R
01FCR DAD0              WD    D,XOFF
      0205R
0200R 0302              BR    R2
                  *
0202R 98A9        TWRT  DC    X'98A9'
0203R             STOP  EQU   TWRT+1
0204R 9193        XON   DC    X'9193'
0205R             XOFF  EQU   XON+1
0206R 9DDE        CHAR  SSR   D,E             READ ONE CHAR
0208R 08EE              LHR   E,E
020AR 4230              BNZ   CHAR            ACCEPT CHAR IF HEX
      0206R
020ER 9BDB              RDR   D,B             1-4,10,15-1F
0210R 9A7B              WDR   ONE,B           DISPLAY DATA
0212R C4B0              NHI   B,X'7F'
      007F
0216R 4330              BZ    CHAR            SKIP ALL OTHER CHARS
      0206R
021AR C5B0              CLHI  B,X'20'
      0020
021ER 4380              BNL   CHAR
      0206R
0222R C5B0              CLHI  B,X'15'
      0015
0226R 0381              BFCR  8,R1
0228R C5B0              CLHI  B,X'11'
      0011
022CR 4380              BNL   CHAR
      0206R
0230R C5B0              CLHI  B,X'10'
```

```
      0010
0234R 0331              BFCR  3,R1
0236R C5B0              CLHI  B,X'05'
      0005
023AR 4380              BNL   CHAR
      0206R
023ER 0301              BR    R1
0240R 0000      LOCX    DC    0
0242R 0080      LOC     DC    X'80'
0244R 0000              DC    0
0246R 0080      BIAS    DC    X'80'
0248R           BUFF    DS    108
02B4R                   DS    2
02B6R                   END

A        000A
ABSF     000F
B        000B
BIAS     0246R
BINDV    0078
BUFF     0248R
BYTE     0004
C        000C
CHAR     0206R
CKIT     003CR
CONT     001AR
D        000D
DUAB     013AR
DURL     0150R
E        000E
END      00A6R
ERR0     00DAR
ERR1     00D6R
ERR2     00D2R
ERR3     00CER
ERROR    0098R
EXTR     018ER
FIXB     00B4R
FLIP     00E8R
FOUR     0009
GETT     0166R
IN1      01B2R
IN2      01BCR
IN3      01C4R
IN4      01F2R
IN5      01F6R
INPUT    01A4R
JUMP     0078R
LDL      010CR
LDX      0100R
LOC      0242R
LOCX     0240R
LOOP     005ER
NEXT     002AR
```

ONE	0007
PICK	0005
R0	0000
R1	0001
R2	0002
R3	0003
REDEF	0008R
SEQNUM	0006
START	0000R
STOP	0203R
TWO	0008
TWRI	0202R
UNAB	0118R
UNRL	0120R
UNRX	0128R
WORD	0176R
WORDI	0178R
XOFF	0205R
XON	0204R

APPENDIX 4

ABS BOOT FRONT END LISTING

ABS BOOTSTRAP
FRONT END
5/6/68

```
                              OPT    PASS2,PRINT,PUNCH,STOP
                      *
                      * BOOTSTRAP FRONT END
                      * CONSISTS OF 8-BIT LOADER AND CHECKSUM LOOP.
                      * APPEARS IN 8-BIT FORMAT.
                      * ASSUMES THAT-REG 3= 1
                      * ASSUMES THAT-REG 10= DEV NUMBER
                      * ASSUMES THAT-TAPE IS MOVING
                      * ASSUMES THAT-ALL CHARS ON TAPE ARE LEGAL
                      * IF CHECKSUM ERROR DETECTED, LOADER STOPS
                      * THE TAPE AND HALTS. AFTER PROCESSOR HALTS.
                      * RESTART BOOTSTRAP LOAD, OR PUSH EXECUTE TO
                      * IGNORE CHECKSUM ERROR AND CONTINUE EXECUTION.
                      *
                      *
0080                          ORG    X'80'
                      *
                      *
1D00                  PLOW    EQU    X'1D00'
1E01                  PHGH    EQU    X'1E01'
1D00                  PSTRT   EQU    X'1D00'
6001                  CKSUM   EQU    X'6001'
                      *
                      *
0080   C820           START   LHI    2,PLOW           SET PROGRAM LIMITS
       1D00
0084   C840                   LHI    4,PHGH           R3 MUST HOLD 1
       1E01
0088   0755                   XHR    5,5              CLEAR R5 FOR CHECKSUM
008A   9DAE           SENSE   SSR    10,14            SENSE STATUS
008C   08EE                   LHR    14,14            TAPE SHOULD BE MOVING
008E   4230                   BNZ    SENSE
       008A
0092   9BAB                   RDR    10,11            READ ONE CHAR
0094   0A5B                   AHR    5,11             BUMP CHECKSUM
0096   9A3B                   WDR    3,11             DISPLAY CHAR
0098   D2B2                   STB    11,0(2)          STORE CHAR
       0000
009C   C120                   BXLE   2,SENSE
       008A
                      *
                      *
00A0   C550                   CLHI   5,CKSUM          CHECK THE CHECKSUM
       6001
00A4   4330                   BE     PSTRT            GO TO PROGRAM IF OK
       1D00
00A8   DEA0                   OC     10,STOP          STOP TAPE IF NOT OK
       00CB
00AC   C5A0                   CLHI   10,2
       0002
00B0   4230                   BNE    HALT
       00C2
00B4   DEA0                   OC     10,TWRT          IF DEV NO = 2
       00CA
```

```
00B8  9DAE            SSR    10,14            PUT TTY IN WRITE MODE
00BA  4290            BTC    9,*-2            AND ISSUE XOFF
      00B8
00BE  DAA0            WD     10,XOFF
      00CD
00C2  C200    HALT    LPSW   *+4              HALT
      00C6
00C6  8000            DC     X'8000',PSTRT    GO TO PROGRAM ON EXECUTE
      1D00
              *
              *
00CA  98A9    TWRT    DC     X'98A9'          WRITE,STOP COMMANDS
00CB          STOP    EQU    *-1
00CC  9193    XON     DC     X'9193'          XON,XOFF CHARACTERS
00CD          XOFF    EQU    *-1
00CE  0000            DC     0                FILLER
00D0                  END

CKSUM    6001
HALT     00C2
PHGH     1E01
PLOW     1D00
PSTRT    1D00
SENSE    008A
START    0080
STOP     00CB
TWRT     00CA
XOFF     00CD
XON      00CC
```

FAST FORMAT LOADER LISTING

```
                                OPT    PASS2,PRINT,PUNCH,STOP
                   *
                   * FAST FORMAT LOADER
                   *
0000               R0       EQU  0
0001               R1       EQU  1
0002               R2       EQU  2
0003               R3       EQU  3
0004               LOC      EQU  4
0005               LAST     EQU  5
0006               GOTO     EQU  6
0007               SEQNUM   EQU  7
0008               ONE      EQU  8
0009               TWO      EQU  9
000A               A        EQU  10
000B               B        EQU  11
000C               C        EQU  12
000D               D        EQU  13
000E               E        EQU  14
000F               F        EQU  15
0078               BINDV    EQU  X'78'
0104               LENGTH   EQU  260
                   *
                   *
0000R 0B77         CONT     SHR  SEQNUM,SEQNUM       CLEAR SEQNUM
0002R C880                  LHI  ONE,1               SET CONSTANTS 1,2
      0001
0006R C890                  LHI  TWO,2
      0002
000AR 0B78         NEXT     SHR  SEQNUM,ONE          NEXT BLOCK
000CR D3D0         INPUT    LB   D,BINDV             PICK UP DEV NO
      0078
0010R 05D9                  CLHR D,TWO
0012R 4230                  BNE  IN2
      0024R
0016R DED0                  OC   D,TWRT              IF TTY, SET WRITE MODE
      00FER
001AR 9DDE         IN1      SSR  D,E                 AND OUTPUT XON
001CR 4290                  BTC  9,IN1
      001AR
0020R DAD0                  WD   D,XON
      0100R
0024R DED0         IN2      OC   D,BINDV+1           START DEVICE
      0079
0028R C8A0                  LHI  A,BUFF              SET BUFF POINTER
      0102R
002CR 4110         IN3      BAL  R1,CHAR
      00ECR
0030R 4330                  BZ   IN3                 SKIP LEADER
      002CR
0034R D2BA         IN31     STB  B,0(A)
      0000
0038R 0AA8                  AHR  A,ONE               READ 260 CHARS
003AR C5A0                  CLHI A,BUFF+LENGTH       STARTING AT FIRST
```

```
       0206R
003ER  4380           BNL   IN32           NON-BLANK
       004AR
0042R  4110           BAL   R1,CHAR
       00ECR
0046R  4300           B     IN31
       0034R
004AR  05D9    IN32   CLHR  D,TWO
004CR  4330           BE    IN4
       0058R
0050R  DED0           OC    D,STOP         STOP DEVICE
       00FFR
0054R  4300           B     CSUM
       0066R
0058R  DED0    IN4    OC    D,TWRT         IF TTY, SET WRITE MODE
       00FER
005CR  9DDE    IN5    SSR   D,E            AND OUTPUT XOFF
005ER  4290           BTC   9,IN5
       005CR
0062R  DAD0           WD    D,XOFF
       0101R
               *
0066R  D3C0    CSUM   LB    C,BUFF         COMPUTE CHECKSUM
       0102R
006AR  D3B0           LB    B,BUFF+1
       0103R
006ER  0ACB           AHR   C,B
0070R  C8A0           LHI   A,BUFF+4
       0106R
0074R  D3BA    CKIT   LB    B,0(A)         CHECKSUM IS ARITHMETIC
       0000
0078R  0ACB           AHR   C,B            SUM OF ALL BYTES EXCEPT
007AR  0AA8           AHR   A,ONE          THE CHECKSUM AT BUFF+2
007CR  C5A0           CLHI  A,BUFF+LENGTH
       0206R
0080R  4280           BL    CKIT
       0074R
0084R  45C0           CLH   C,BUFF+2
       0104R
0088R  4230           BNE   ERROR
       00DER
008CR  4570           CLH   SEQNUM,BUFF
       0102R
0090R  4230           BNE   ERROR
       00DER
               *
0094R  C8A0    STORE  LHI   A,BUFF+4       STORE DATA
       0106R
0098R  C570           CLHI  SEQNUM,-1
       FFFF
009CR  4230           BNE   LOOP
       00B0R
00A0R  4840           LH    LOC,BUFF+4     IF FIRST BLOCK
       0106R
```

```
00A4R 4850              LH    LAST,BUFF+6        SET BEG,END,AND
      0108R
00A8R 4860              LH    GOTO,BUFF+8        TRANSFER ADDRESS
      010AR
00ACR C8A0              LHI   A,BUFF+10
      010CR
00B0R 48BA      LOOP    LH    B,0(A)             STORE HALFWORD
      0000
00B4R 40B4              STH   B,0(LOC)
      0000
00B8R 0AA9              AHR   A,TWO
00BAR 0545              CLHR  LOC,LAST           STOP WHEN LAST
00BCR 4380              BNL   END                HALFWORD FILLED
      00CER
00C0R 0A49              AHR   LOC,TWO
00C2R C5A0              CLHI  A,BUFF+LENGTH      READ NEXT BLOCK
      0206R
00C6R 4280              BL    LOOP               WHEN BUFF EMPTY
      00B0R
00CAR 4300              B     NEXT
      000AR
                *
00CER 0866      END     LHR   GOTO,GOTO
00D0R 0236              BTCR  3,GOTO
00D2R 07CC              XHR   C,C                DISPLAY X'00'
00D4R 9A8C              WDR   ONE,C              FOR NORMAL END
00D6R C200      HALT    LPSW  *+4
      00DAR
00DAR 8000              DC    X'8000',A(CONT)
      0000R
                *
00DER C8C0      ERROR   LHI   C,X'F'             DISPLAY X'0F' TO
      000F
00E2R 9A8C              WDR   ONE,C              SHOW INPUT ERROR
00E4R C200              LPSW  *+4
      00E8R
00E8R 8000              DC    X'8000',A(INPUT)
      000CR
                *
00ECR 9DDE      CHAR    SSR   D,E                READ ONE CHAR
00EER 08EE              LHR   E,E                AND DISPLAY IT
00F0R 4230              BNZ   CHAR
      00ECR
00F4R 9BDB              RDR   D,B
00F6R 9A8B              WDR   ONE,B
00F8R C4B0              NHI   B,X'FF'
      00FF
00FCR 0301              BR    R1
                *
00FER 98A9      TWRT    DC    X'98A9'
00FFR           STOP    EQU   TWRT+1
0100R 9193      XON     DC    X'9193'
0101R           XOFF    EQU   XON+1
                *
```

```
0102R           BUFF    DS   LENGTH
0206R                   DS   2
0208R                   END

A        000A
B        000B
BINDV    0078
BUFF     0102
C        000C
CHAR     00EC
CKIT     0074
CONT     0000
CSUM     0066
D        000D
E        000E
END      00CE
ERROR    00DE
F        000F
GOTO     0006
HALT     00D6
IN1      001A
IN2      0024
IN3      002C
IN31     0034
IN32     004A
IN4      0058
IN5      005C
INPUT    000C
LAST     0005
LENGTH   0104
LOC      0004
LOOP     00B0
NEXT     000A
ONE      0008
R0       0000
R1       0001
R2       0002
R3       0003
SEQNUM   0007
STOP     00FF
STORE    0094
TWO      0009
TWRT     00FE
XOFF     0101
XON      0100
```

APPENDIX 6

FAST FORMAT PUNCHER LISTING

```
                                 OPT   PASS2,PRINT,PUNCH,STOP
                            *
                            *  FAST FORMAT PUNCHER
                            *
                            *  THIS PROGRAM PUNCHES A BLOCK OF CORE
                            *  IN FAST FORMAT.  BLANK LEADER AND
                            *  TRAILER IS PUNCHED.  THE PROGRAM HALTS
                            *  PRIOR TO AND AFTER PUNCHING TO ALLOW
                            *  THE PUNCH TO BE TURNED ON AND OFF.
                            *
                            *  ORG    = LOW LIMIT
                            *  ORG+2  = HIGH LIMIT (LAST HALFWORD IN BLOCK)
                            *  ORG+4  = TRANSFER ADRS (MAKE Ø FOR NO TRANSFER)
                            *  ORG+6  = UNUSED
                            *  ORG+8  = STARTING LOCATION
                            *
                            *  THE DEVICE NUMBER IS TAKEN FROM 7A
                            *  THE OUTPUT COMMAND IS TAKEN FROM 7B
                            *
ØØØ1                        R1        EQU    1
ØØØ2                        R2        EQU    2
ØØØ3                        LOC       EQU    3
ØØØ4                        SEQNUM    EQU    4
ØØØA                        A         EQU    1Ø
ØØØB                        B         EQU    11
ØØØC                        C         EQU    12
ØØØD                        CHAR      EQU    13
ØØØE                        DEV       EQU    14
ØØØF                        STAT      EQU    15
ØØ7A                        BOUTDV    EQU    X'7A'
Ø1Ø4                        LENGTH    EQU    26Ø
ØØØ8                        GAP       EQU    8
                            *
ØØØØR ØØØØ                  LOWADR    DC     Ø
ØØØ2R ØØØØ                  HGHADR    DC     Ø
ØØØ4R ØØØØ                  XFRADR    DC     Ø
ØØØ6R ØØØØ                            DC     Ø
ØØØ8R C2ØØ                  STRT      LPSW   WAIT
      ØØØCR
ØØØCR 8ØØØ                  WAIT      DC     X'8000',A(GOGO)
      ØØ1ØR
ØØ1ØR 412Ø                  GOGO      BAL    R2,LEADER          PUNCH LEADER
      ØØA8R
ØØ14R ØB44                            SHR    SEQNUM,SEQNUM      CLEAR SEQNUM
ØØ16R 483Ø                            LH     LOC,XFRADR
      ØØØ4R
ØØ1AR 4Ø3Ø                            STH    LOC,BUFF+8         SET XFER ADRS
      ØØD6R
ØØ1ER 483Ø                            LH     LOC,HGHADR
      ØØØ2R
ØØ22R 4Ø3Ø                            STH    LOC,BUFF+6         SET HIGH LIMIT
      ØØD4R
ØØ26R 483Ø                            LH     LOC,LOWADR
      ØØØØR
```

```
ØØ2AR 4Ø3Ø              STH   LOC,BUFF+4          SET LOW LIMIT
      ØØD2R
ØØ2ER C8AØ              LHI   A,BUFF+10
      ØØD8R
ØØ32R CB4Ø      BLOK    SHI   SEQNUM,1            DECR SEQNUM
      ØØØ1
ØØ36R 4Ø4Ø              STH   SEQNUM,BUFF
      ØØCER
ØØ3AR 48B3      MOVE    LH    B,Ø(LOC)            MOVE DATA INTO
      ØØØØ
ØØ3ER 4ØBA              STH   B,Ø(A)              BUFFER UNTIL THE
      ØØØØ
ØØ42R CA3Ø              AHI   LOC,2               BUFFER IS FULL
      ØØØ2
ØØ46R CAAØ              AHI   A,2
      ØØØ2
ØØ4AR C5AØ              CLHI  A,BUFF+LENGTH
      Ø1D2R
ØØ4ER 428Ø              BL    MOVE
      ØØ3AR
                *
ØØ52R D3CØ      CSUM    LB    C,BUFF              COMPUTE
      ØØCER
ØØ56R D3BØ              LB    B,BUFF+1            ARITHMETIC
      ØØCFR
ØØ5AR C8AØ              LHI   A,BUFF+4            CHECKSUM
      ØØD2R
ØØ5ER ØACB              AHR   C,B
ØØ6ØR D3BA      CKIT    LB    B,Ø(A)
      ØØØØ
ØØ64R ØACB              AHR   C,B
ØØ66R CAAØ              AHI   A,1
      ØØØ1
ØØ6AR C5AØ              CLHI  A,BUFF+LENGTH
      Ø1D2R
ØØ6ER 428Ø              BL    CKIT
      ØØ6ØR
ØØ72R 4ØCØ              STH   C,BUFF+2            PUT CSUM INTO BUFF+2
      ØØDØR
                *
ØØ76R C8AØ              LHI   A,BUFF
      ØØCER
ØØ7AR D3DA      GETS    LB    CHAR,Ø(A)           PUNCH BUFFER
      ØØØØ
ØØ7ER 411Ø              BAL   R1,PUNCH            IN 8 BIT BYTES
      ØØBCR
ØØ82R CAAØ              AHI   A,1
      ØØØ1
ØØ86R C5AØ              CLHI  A,BUFF+LENGTH+GAP
      Ø1DAR
ØØ8AR 428Ø              BL    GET8
      ØØ7AR
ØØ8ER 48AØ              LH    A,HGHADR
      ØØØ2R
```

```
ØØ92R Ø5A3               CLHR A, LOC          TERMINATE WHEN
ØØ94R 428Ø               BL     TERM          ALL DATA PUNCHED
      ØØAØR
ØØ98R C8AØ               LHI    A,BUFF+4
      ØØD2R
ØØ9CR 43ØØ               B      BLOK
      ØØ32R
              *
ØØAØR 412Ø    TERM       BAL    R2,LEADER
      ØØA8R
ØØA4R 43ØØ               B      STRT
      ØØØ8R
              *
ØØA8R Ø7DD    LEADER     XHR    CHAR,CHAR     PUNCH LEADER
ØØAAR C8AØ               LHI    A,1ØØ
      ØØ64
ØØAER 411Ø               BAL    R1,PUNCH
      ØØBCR
ØØB2R CBAØ               SHI    A,1
      ØØØ1
ØØB6R 423Ø               BNZ    *-8
      ØØAER
ØØBAR Ø3Ø2               BR     R2
ØØBCR D3EØ    PUNCH      LB     DEV,BOUTDV    PUNCH CHAR
      ØØ7A
ØØCØR DEEØ               OC     DEV,BOUTDV+1
      ØØ7B
ØØC4R 9DEF               SSR    DEV,STAT
ØØC6R 429Ø               BTC    9,*-2
      ØØC4R
ØØCAR 9AED               WDR    DEV,CHAR
ØØCCR Ø3Ø1               BR     R1
ØØCER         BUFF       DS     LENGTH
Ø1D2R                    DO     GAP
Ø1D2R ØØØØ               DC     Ø
Ø1D4R ØØØØ
Ø1D6R ØØØØ
Ø1D8R ØØØØ
Ø1DAR ØØØØ
Ø1DCR ØØØØ
Ø1DER ØØØØ
Ø1EØR ØØØØ
Ø1E2R                    END

A     ØØØA
B     ØØØB
BLOK  ØØ32
BOUTDVØØ7A
BUFF  ØØCE
C     ØØØC
CHAR  ØØØD
CKIT  ØØ6Ø
CSUM  ØØ52
DEV   ØØØE
```

GAP	ØØØ8
GET8	ØØ7A
GOGO	ØØ1Ø
HGHADR	ØØØ2
LEADER	ØØA8
LENGTH	Ø1Ø4
LOC	ØØØ3
LOWADR	ØØØØ
MOVE	ØØ3A
PUNCH	ØØBC
R1	ØØØ1
R2	ØØØ2
SEQNUM	ØØØ4
STAT	ØØØF
STRT	ØØØ8
TERM	ØØAØ
WAIT	ØØØC
XFRADR	ØØØ4

TABLE OF CONTENTS

ASSEMBLER PROGRAM MANUAL

1. INTRODUCTION

GE-PAC 30 Digital Systems involve Processors which can be programmed to solve a wide range of problems. The program to be executed by a Processor consists of binary coded instructions and data which are stored in a core memory. The instructions, and their binary code, which are recognized by the GE-PAC 30 Processors, are defined in the GE-PAC 30 Reference Manual, Publication Number 29-004.

To assist the process of defining and generating a program, the user can write his program in a symbolic way, using what is called assembly language. In the assembly language, programs are represented using symbols and mnemonic abbreviations for the instructions and data in the program. The statements in the assembly language which represent the program constitute the source form of the program. Table 1 is an example of an assembly language program that searches an area of core memory for the first occurrence of the number 15.

The translation from the symbolic source program to the binary object program is done by the assembler. The assembler reads the source program, statement by statement, from punched paper tape or cards. As the statements are read, a symbol table is accummulated. This table contains every symbol and the value of the location counter where the symbol was encountered. For the previous example, the symbol table after reading the source program once, would be as shown in Table 2.

The assembler generates both an object tape and a listing. The object tape contains the binary information to be loaded into memory. The listing is a printed record which shows each source statement and the binary information generated for that statement. The binary information on a listing is always represented in hexadecimal form.

TABLE 1. TYPICAL SOURCE PROGRAM

Name	Operation	Operand	Comment
	ORG	X'100'	SET THE LOCATION COUNTER
BEGIN	LHI	2, TOP	TOP OF DATA TABLE
	LHI	3, 2	HALFWORD INCREMENT
	LHI	4, BOTTOM	BOTTOM OF DATA TABLE
	LHI	10, 15	SEARCH VALUE OF 15
LOOK	CLH	10, 0(2)	COMPARE
	BE	FINI	BRANCH ON EQUAL TO FINI
	BXLE	2, LOOK	NOT FOUND GO LOOK FURTHER
FINI	LPSW	WAIT	STOP THE PROGRAM
WAIT	DC	X'8000', A(BEGIN)	
TOP	DS	1000	
BOTTOM	DS	2	
	END	BEGIN	

TABLE 2. TYPICAL SYMBOL TABLE

Symbol	Value (in hexadecimal)
BEGIN	0100
LOOK	0110
FINI	011C
WAIT	0120
TOP	0124
BOTTOM	050C

It is important to note that this assembler processes assembly language statements for user programs which are to reside in core memory. Assemblers which process micro-programs for a Read-Only Memory, which are related to the internal structure of the GE-PAC 30 Processor, are discussed in other publications, such as the GE-PAC 30-1 Micro-Programming Manual, Publication Number 29-021.

There are a number of versions of the Assembler Program available, each tailored to specific machine configurations. Certain versions of the assembler read source statements from a card reader, and other versions read source statements from tape devices such as paper tape or magnetic tape. Only certain versions of the assembler provide means for generating floating-point instructions and data. For a description of the various versions, and the differences between them, refer to the Assembler Operating Procedures, Publication Number 03-001A16.

2. ASSEMBLY PROCEDURES

The assembler takes one, two, or three passes across the source tape to complete the assembly. The number of passes is controlled by an option control statement in the source program. Refer to the OPT pseudo operation for details. When so directed, the assembler makes an assembly -- complete with listing and object tape -- in one pass. In this case the assembly time is minimized, but the resulting object tape must be loaded with the General Loader (part number 06-025).

With two pass-assemblies, the first pass is devoted to development of a symbol table. On the second pass, the listing is printed and the object tape is punched. Assemblies are normally performed using two passes. The two-pass procedure is appropriate except where the input-output device configuration prohibits punching and printing on the same pass. In this case, the three-pass assembly can be used. With a three-pass assembly, the symbol table is built on pass one, the listing is printed on pass 2, and the object tape is punched on pass 3.

The assembly listing is produced as part of the assembly process. The listing contains the source statements and the data generated from each statement. Table 3 indicates the assembly listing for the previous example.

The first four hexadecimal digits represent the value of the location counter or values of symbols resulting from EQU assembler statements. The next four hexadecimal digits represent the data generated by the assembler from the source statement.

Error flags may precede the location counter values. These flags indicate that an error was encountered in interpreting the statement. The meaning of each flag is as follows:

TABLE 3. SAMPLE ASSEMBLY LISTING

Location	Data	Name	Operation	Operand	Comments
0100			ORG	X'100'	SET THE LOCATION COUNTER
0100	C820	BEGIN	LHI	2, TOP	TOP OF DATA TABLE
	0124				
0104	C830		LHI	3, 2	HALFWORD INCREMENT
	0002				
0108	C840		LHI	4, BOTTOM	BOTTOM OF DATA TABLE
	050C				
010C	C8A0		LHI	10, 15	SEARCH VALUE OF 15
	000F				
0110	45A3	LOOK	CLH	10, 0(2)	COMPARE
	0000				
0114	4330		BE	FINI	BRANCH ON EQUAL TO FINI
	011C				
0118	C120		BXLE	2, LOOK	NOT FOUND GO LOOK FURTHER
	0110				
011C	C200	FINI	LPSW	WAIT	STOP THE PROGRAM
	0120				
0120	8000	WAIT	DC	X'8000', A(BEGIN)	
	0100				
0124		TOP	DS	1000	
050C		BOTTOM	DS	2	
050E			END	BEGIN	
BEGIN	0100				
BOTTOM	050C				
FINI	011C				
LOOK	0110				
TOP	0124				
WAIT	0120				

F	Format error
M	Multiple defined symbol
O	Operation mnemonic invalid
T	Truncation error, a constant or expression has overflowed the specified limits
R	Relocation error, a meaningless combination of relocatable symbols in an expression
S	Symbol table overflow
U	Undefined symbol

Whenever an invalid op error (O) occurs, the assembler always advances the location counter by four bytes so that the program can be patched easily for debugging.

A flag immediately following the data generated by the assembler indicates whether the data is relocatable, absolute, or a forward reference. The flags are:

BLANK	Absolute data
R	Relocatable data
F	Forward reference data

The symbol table that was accummulated during PASS1 is printed following the END assembly pseudo-op. Any statements containing symbols preceded by an error flag of U (Undefined symbol) can be corrected at this time and repeat PASS1 of the assembly process.

The symbol table is again printed following the END assembly pseudo-op after PASS2. The symbols are listed alphabetically with their values. If the symbol is defined, the value is followed by an R if that value is relocatable. If the symbol is undefined, the last value of the location counter for a statement referencing the undefined symbol is printed.

Preceding each symbol is a field for error flags. These flags are as follows:

*	-	Externally defined symbol
U	-	Undefined symbol

3. THE ASSEMBLER LANGUAGE

3.1 Source Statements

There are two basic kinds of source statements, instruction statements and comment statements. Instruction statements are used for machine instructions and assembler instructions. The instruction statements may have the following information fields:

- Name
- Operation
- Operand
- Comments

Comment statements, which begin with *, should not be confused with the comment field of the instruction statement. Comment statements can occupy the entire statement line.

3.1.1 Instruction Statements. The comment and instruction statements are written by the programmer on a coding form that has the various fields clearly marked. See Figure 1. This form, when filled out, is used to generate the source paper tape or source cards that are read by the assembler during the assembly process. As shown on Figure 1, the Name begins in column 1, the Operation begins in column 10, the Operand begins in column 16, and Comments are usually in 35-60. The fixed field positions are a convenience for the programmer only, and are not required by the assembler. The assembler simply requires that fields be separated by one or more spaces. The fields are described in the following paragraphs.

Name

A name is from one to six characters in length. The name must be written with the first character in column 1, and it must not contain any blanks. Names are used by the programmer to identify data and instructions in the program. The first character must be a letter; the remaining five can be letters or numbers. Typical names are:

Name	Operation
START	
ARG1	
LOOP2	
GO	

Operation

The operation field specifies a machine instruction mnemonic that is translated by the assembler to machine code, or it specifies an assembler instruction mnemonic to control the assembly process. An operation is always required in an instruction statement, and should be written on the coding form beginning in column 10. No blanks may be used within the operation. Typical operations are:

Name	Operation	Operand
	ORG	
	LHI	
	AHR	
	DC	

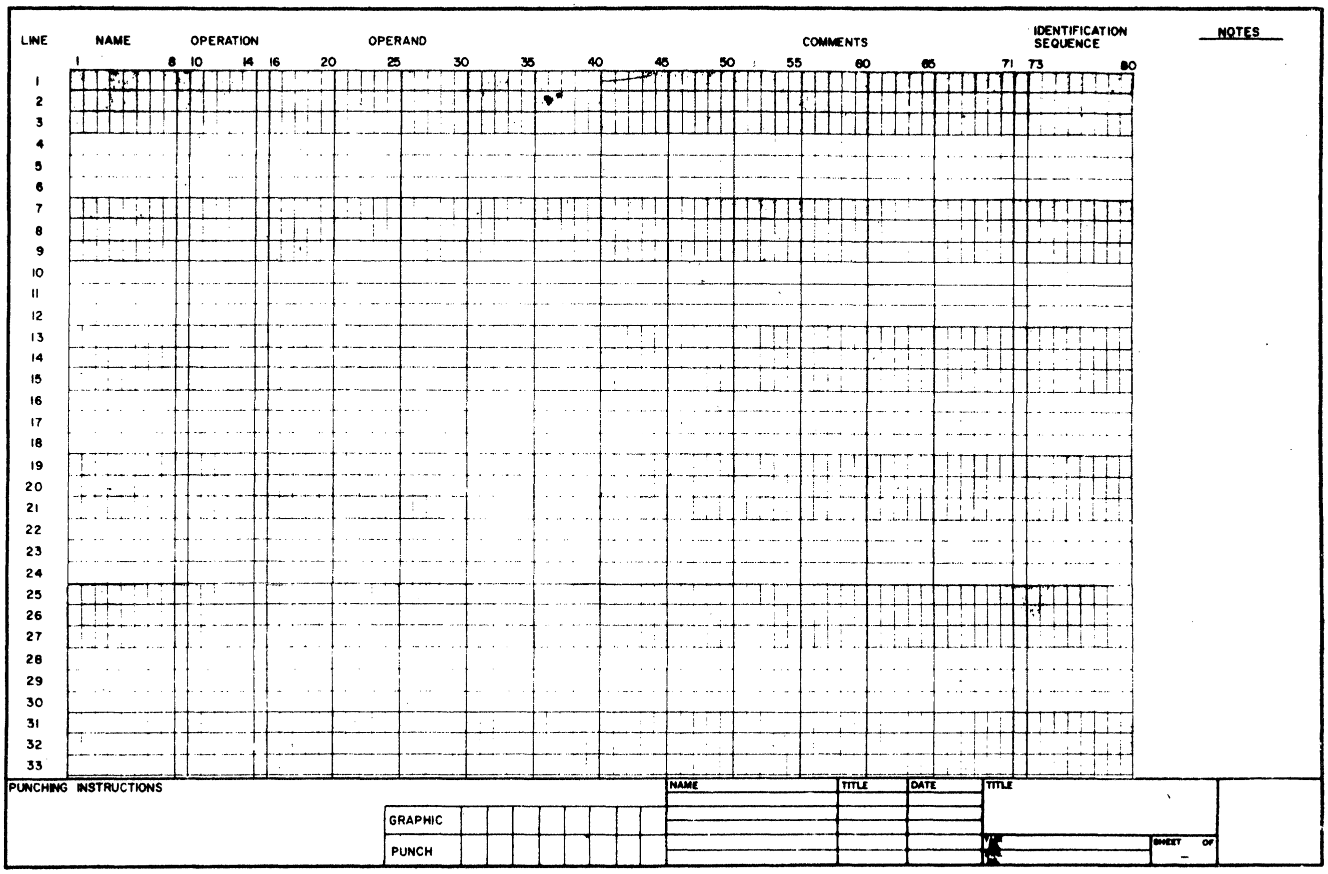


Figure 1. Coding Form

Operand

Operands identify the data to be used by the instruction. The type of operand and the number of operands required depend on the particular instruction appearing as the operation. No blanks may appear within, or between, operands. Typical operands are:

Name	Operation	Operand
	AH	R6, TEMP
	BL	OUT
	STH	R6, TABLE(R5)
	B	IN

Comment

Comments are descriptive text. Comments are printed on the assembly listing, along with the name, operation, and operand of the source statement. Comments are written beginning after the first blank in the operand, and can contain 26 characters. In general, columns 36-61 are used. Typical comments are:

Operand	Comment
R6, TEMP	FETCH FIRST VALUE
A(START)	TABLE AREA
STOP	ERROR STOP

3.1.2 Comment Statements. Comment statements are descriptive text that can occupy the entire source statement line. Comment statements are written with an asterisk (*) in column 1, followed by any descriptive text the programmer desires and can contain 55 characters in addition to the *. Any number of comment statements may be used at any place in a program. Comment statements do not produce binary object information and are used only as documenting aids. Several comment statements are:

```
* THIS IS A COMMENT STATEMENT,
* IT CAN BE USED ANYWHERE IN A
* PROGRAM AS A PROGRAMMER
* AND DOCUMENTATION AID
*
* X < OR = Y/Z? IF SO, GO ON
```

3.1.3 Character Set. All source statements are written using the following characters:

Alphabetics	A through Z
Numerics	0 through 9
Special characters	+ - , = * ' ( ) blank and all characters printable on a teletypewriter

3.2 Assembler Language Structure

The source instruction statement consists of:

- A name
- An operation
- An operand
- A comment

Each entry in a source statement may be composed of one or more items depending on the kind of source statement being written.

- A name, when present, must be a symbol
- An operand may be composed of one or more expressions, which in turn are composed of symbols, constants and arithmetic combinations of symbols and constants
- An operation, always present, must be a machine instruction Mnemonic or an assembler instruction Mnemonic

3.2.1 Symbols. A symbol is used as a name or as an operand. In either case, symbols consist of from one to six characters. The first character must be alphabetic. The characters that can be used for a symbol are:

Alphabetics	A through Z
Numerics	0 through 9

The following symbols are valid and could be used as a name or as an operand.

T2
LOOP25
N
STOP

The following symbols are invalid for the reasons given:

2TOP	First character is not alphabetic
COMMAND	More than 6 characters
A to D	Contains a blank
X4.2	Contains a special character, a period

3.2.2 Instruction Constants. Instruction constants appear as an operand for both machine instructions and assembler instructions. An instruction constant can be one of three types:

- Decimal
- Hexadecimal
- Character

In general, instruction constants define 16-bits or a halfword of information. The type of constant is identified by a prefix code.

Code	Constant Type
None	Decimal
H	Halfword Decimal
X	Hexadecimal
C	Character

Decimal constants can be from one to five decimal digits, not to exceed 32,767 maximum or -32,768 minimum, and are written as:

```
125
32765
-15
etc.
```

Hexadecimal constants can be from one to four digits. The hexadecimal digits are:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

The hexadecimal constant must be enclosed in single quotation marks and be preceded by the letter X. Leading zeros are not necessary. The hexadecimal constants are right justified to form 16-bit halfwords. Examples are:

X'F'	or	X'000F'
X'D4E'	or	X'0D4E'
X'030'	or	X'0030'

Character constants used in the operand field of an instruction statement can be from one to two characters. The permissable characters are:

Alphabetics	A through Z
Numerics	0 through 9
Special characters	+, -, = * ( ) blank and all ASCII coded characters printable on the teletypewriter except the single quote (').

The character constant must be enclosed in single quotation marks and be preceded by the letter C. A single character within the quotes is right justified to form a 16-bit halfword. Each character is translated into a byte of seven bit ASCII code.

C'*'	generates	X'002A'
C'12'	generates	X'3132'
C'XY'	generates	X'5859'

3.2.3 Expressions. An expression is a symbol, a constant, or a series of such items separated by the arithmetic operations + (addition), and - (subtraction). Examples of valid expressions are:

```
SAM
5
LOOP+4
TABLE+X'12A'
STOP-GO+2
C'A'+1
-FROG
```

3.2.4 Relocatable and Absolute Expressions. An expression is absolute if its value is absolute. Similarly, an absolute expression does not change as a function of the physical location of the program in the machine. The value of a relocatable expression does change when the location of the program changes. The relocatable value will change by the difference in byte locations between the originally assigned area of storage and the newly assigned area of storage.

An expression, when evaluated, produces a value which is considered absolute or relocatable according to the rules outlined in Table 4.

3.2.5 Location Counter. The value of the location counter can be referenced by using an *, which means "current value of location counter". Addressing relative to the location counter is on a byte basis. To specify an address that is one RX instruction forward, the correct expression would be *+4.

TABLE 4. ABSOLUTE AND RELOCATABLE EXPRESSION RULES

	A+B	A-B
A is absolute, B is absolute	Absolute	Absolute
A is absolute, B is relocatable	Relocatable	Invalid
A is relocatable, B is absolute	Relocatable	Relocatable
A is relocatable, B is relocatable	Invalid	Absolute

In both examples below, the Branch instruction transfers to the instruction labelled LOOP25.

```
         B      *+8
         LHI    R6,0
LOOP25   LB     R5,TABLE(R6)

         B      *+6
         SHR    R6,R6
LOOP25   LB     R5,TABLE(R6)
```

The proper alignment of the location counter to halfword memory boundaries is provided by the assembler. If a character data constant specification is followed by an instruction or halfword data, halfword alignment is forced. The value of the location counter is absolute or relocatable depending on the operand entry of the assembler ORG instruction. If the expression appearing as the operand is relocatable, the location counter value (*) is relocatable; if the expression is absolute, the location counter value is absolute. If no ORG is specified in a program, the location counter starts at relocatable zero.

4. MACHINE INSTRUCTIONS FORMAT

The assembler provides the facility for representing all the machine instruction operation codes with mnemonics. The binary instruction is generated by the assembler from the operation mnemonic and the operand. Table 5 summarizes the formats used.

The mnemonic in the operation field specifies the desired function, i.e., Add. Each instruction has a unique mnemonic that is used as the operation. These mnemonics and their meanings are listed in Appendix 1. Some instruction examples are:

RR - Format Instructions

Name	Operation	Operand
GO	LHR	1,2
	BALR	R15,R12
LOOP12	AHR	3,3
	DHR	DEND,ISOR

RX - Format Instructions

Name	Operation	Operand
TEST1	STH	R7,TEMP
	MH	13,TABLE (3)
	LH	TWELV,0(X7)
	AH	X'B',TOP+4(5)

RS - Format Instructions

Name	Operation	Operand
	LHI	0,X'9DAE'
FINI	AHI	R7,1
	BXLE	R4,LAST1
	SLHL	R12,8

5. ASSEMBLER INSTRUCTIONS (PSEUDO-OPS)

Assembler instructions are used to control the assembly process, define symbols, and generate data. Assembler instruction statements do not always generate data as the machine instruction statements do. The following paragraphs describe the assembler instructions.

TABLE 5. INSTRUCTION FORMAT SUMMARY

Machine Format					Assembly Format		Applicable Instructions
Bits	8	4	4	16	OPERATION	OPERAND	
	OP	R1	R2		OP	R1, R2	All RR except branches
	OP	M1	R2		OP	M1, R2	BTCR or BFCR
	OP		R2		OP	R2	BR or NOPR
	OP	R1	X2	A	OP	R1, A(X2)	All RX except branches
	OP	M1	X2	A	OP	M1, A(X2)	BTC or BFC
	OP		X2	A	OP	A(X2)	B or NOP or Extended Branch Mnemonics
	OP	R1	X2	A	OP	R1, A(X2)	All RS except LPSW
	OP		X2	A	OP	A(X2)	LPSW

5.1 Symbol Definition Instructions

5.1.1 EQU - Equate Symbol

Name	Operation	Operand
A symbol required	EQU	an expression

The EQU assembler instruction is used to equate a symbol to the value of an expression. Symbols used in the expression must be previously defined. The value of the symbol is relocatable or absolute as determined by the expression.

The EQU assembler instruction is used to equate symbolic General Register names to their appropriate value.

Name	Operation	Operand
R6	EQU	6
R7	EQU	7
	•	
	•	
	•	
	LHR	R6, R7

5.1.2 ENTRY - Identify Entry-Point Symbol

Name	Operation	Operand
Not used	ENTRY	One or more symbols separated by commas

The utility of symbolic register designations is in the ease with which registers can be reassigned without extensive recoding. To change from General Register 6 to General Register 1 requires changing only the R6 EQU 6 assembler statement.

Other examples are:

Name	Operation	Operand
LOOP	EQU	LOOP1
TOP	EQU	END-64
DELTA	EQU	BOTTOM-TOP
HERE	EQU	*
START	EQU	X'01FE'
BLANKS	EQU	C'

The ENTRY assembler instruction identifies symbols that are defined in this program and may be used by some other program. This permits programs that are assembled separately to communicate with each other. Only those symbols identified as entry symbols are available to other separately assembled programs. All ENTRY statements must precede any symbol definitions in the program.

An example is:

Name	Operation	Operand
	ENTRY	SIN, COSIN
	•	
	•	
SIN	LHI	R7, TEMP2
	•	
	•	
COSIN	LHI	R8, TEMP3
	•	
	•	
	END	

5.1.3 EXTRN - Identify External Symbol

Name	Operation	Operand
Not used	EXTRN	One or more symbols separated by commas

The EXTRN assembler instruction identifies symbols that are defined in another program that will be referenced by this program. This permits programs that are assembled separately to communicate with each other. Only those symbols identified as ENTRY symbols in another program should be identified as externally defined in this program. All EXTRN statements must precede any symbol definitions in the program.

An example is:

Name	Operation	Operand
	EXTRN	SIN, COSIN
	•	
	•	
	BAL	R15, SIN
	•	
	•	
	BAL	R15, COSIN
	•	
	•	
	END	

Any symbols declared as EXTRN's must be used with the following restrictions:

1. EXTRN symbols must not be combined in arithmetic expressions; i.e.

   LH 3, SIN+2

2. EXTRN symbols must not be used in the R1 or R2 field of an instruction; i.e.

   LH 3, 2(SIN)

3. EXTRN symbols must not be used with assembler pseudo-ops such as DO, EQU, END, etc.

The utility of the ENTRY, EXTRN assembler instructions is realized when subroutines are written. Rather than having to assemble the main program and its subroutines at the same time in order to establish correct communication, the ENTRY EXTRN permits the main program to be assembled and then loaded with the previously assembled subroutines. The symbols identified by ENTRY and EXTRN statements are then linked at load time by the General Loader. The ability to assemble and debug the subroutines and main programs in a modular fashion is very convenient.

Consider the following two hypothetical programs:

```
Name      Operation     Operand

*         MAIN  PROGRAM
*
*
          EXTRN         RIPLE,DABB
START     LHI           R7,7
          LHI           R15,X'F0F0'
          •
          •
          STH           R1,DABB
          BAL           R7,RIPLE
          B             START
          END
*         SUBROUTINE    RIPLE
*
          ENTRY         RIPLE,DABB
START     LH            R2,DABB
          AHI           R2,C'*-'
          BR            R7
DABB      DS            2
RIPLE     EQU           START
          END
```

The symbols RIPLE and DABB are used by the main program, but their values are not known at assembly time. Since they are defined as EXTRN's, the symbols RIPLE and DABB and the location at which they are referenced in the main program, are punched on the object tape or cards along with the rest of the assembled main program. In a similar fashion, when the subroutine is assembled, the symbols RIPLE and DABB and their values are punched along with the subroutine.

As the main program and subroutines are loaded, the loader accumulates a table of references to symbols and their values. This information is used by the loader to link the main program and subroutine by replacing every reference to RIPLE and DABB by the values passed on from the subroutine by the ENTRY assembler instruction. Note that the General Loader must be used to load the object tape for any program involving ENTRY's or EXTRN's.

5.2 Data Definition Instructions

There are two data definition instructions, the DC and the DS. These assembler instructions provide a convenient means to define and reserve data storage.

5.2.1 DC - Define Constant

Name	Operation	Operand
A symbol optional	DC	One or more operands separated by commas

The DC assembler instruction is used to define constants and generate actual data. These constants may be hexadecimal, decimal, character, address, or floating-point constants. The type of constant is indicated by a prefix code.

Code	Constant Type	Machine Format
C	Character	8-bit character code
X	Hexadecimal	16-bit binary
H	Decimal	16-bit binary
A	Address	16-bit binary
E	Floating-Point	32-bit binary

5.2.1.1 C - Character Constant

The character constant can be any length. It must be enclosed in single quotation marks and preceded by a C.

```
Name      Operation     Operand

MESG1     DC            C'LOAD THE TAPE'
          DC            C'EXECUTE AT 19FE'
```

Each character is translated into one 8-bit byte of storage. If an odd number of characters is specified, a blank character is automatically appended. This maintains halfword boundary alignment for any following machine instructions. If only one character appears between the quote marks, the 8-bit byte is left justified in the halfword, with the code for blank X'20' in the right half. In general, all characters are translated into 7-bit ASCII code, with the most significant bit zero. As an example of this alignment, process, the following two data definition instructions are equivalent. Each instruction generates 14 bytes of data.

Name	Operation	Operand
	DC	C'AN ODD NUMBER'
	DC	C'AN EVEN NUMBER'

5.2.1.2 X - Hexadecimal Constant

A hexadecimal constant can be from one to four digits. The hexadecimal digits are:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

The hexadecimal constants must be enclosed in single quotation marks and preceded by an X. Examples are:

Name	Operation	Operand
DATA1	DC	X'1FE'
	DC	X'C800'

The hexadecimal constant is converted to a properly aligned 16-bit halfword. If fewer than four digits are specified, the digits are right justified and leading zeros generated. For example, the following data constants are equivalent and result in a 16-bit data constant.

Name	Operation	Operand
	DC	X'1C'
	DC	X'01C'
	DC	X'001C'

5.2.1.3 H - Halfword Decimal Constants

A decimal constant can be from one to five digits plus sign. They cannot exceed +32,767 maximum or -32,768 minimum. The decimal digits are enclosed in single quotation marks and preceded by the letter H.

DC - H'-792', H'-30000'

The decimal constant is converted to a properly aligned, right justified 16-bit integer.

5.2.1.4 A - Address Constant

An address constant is a storage address that is translated into a constant. It is a relocatable or absolute constant as determined by the combination of symbols and constants in the expression. Unlike other constants, the address constant is enclosed in parentheses and preceded by the letter A.

DC	A(LOOP+2)
DC	A(TABLE)
DC	A(TOP-BOTTOM)

The constant stored is relocatable or absolute as determined by the rules given in Table 4.

The following examples show how a single DC instruction can be used to define different types of data. Each operand is separated from the next with a comma.

Name	Operation	Operand
DATUM1	DC	X'0F00',C'ABCD'
MSG2	DC	C'A MESSAGE', H'132'
	DC	A(ARGA1),A(HEX-16),X'39'

Decimal constants and address constants can be created without the H' ' and A( ) notation if desired. For example:

DC	123,H'123'
DC	SAM,A(SAM)
DC	TOP+39,X-Y

5.2.1.5 E - Floating Point Constant

The floating-point constant consists of a decimal number, as formatted below, enclosed in single quotes and preceded by an E. The format of the decimal number is as follows:

1. An optional leading plus sign or a minus sign.

2. One or more decimal digits that may include a decimal point.

3. An optional E character followed by an optional leading plus sign or a minus sign and one or two decimal digits, denoting a power of ten.

If, however, more than six digits are specified, (for example E'1234567E3'), the proper order of magnitude will result, but only six digits of precision are maintained. That is, numbers in the range from approximately 5.4 x 10^{-79} to 7.2 x 10^{75} can be represented in the above format with six digits of precision.

Each floating-point DC data constant entry is translated by the assembler into a floating-point number in a specified binary representation requiring two halfwords. Refer to the **GE-PAC 30** Reference Manual, Publication Number 29-004 for a detailed explanation of the floating-point binary representation generated.

There cannot be any blanks within or between E constants, and they must be separated from each other with a comma as in the last example below.

Examples:			
	DC	E'7.2E+75'	approximate maximum
	DC	E'5.4E-79'	approximate minimum
	DC	E'7,1E+75'	
	DC	E'5.5E-79'	
	DC	E'+127.47E-45'	
	DC	E'-4.007E0'	
	DC	E'123456'	
	DC	E'.123456E6'	
	DC	E'1E-74', E'1E-75'	

The assembler will produce an error flag when any of the following occur:

Error Flag	Error
F	Multiple decimal points occur before the number is terminated, or an E is encountered.
F	Any decimal point occurs after the E is encountered.
T	The specified power of ten is not in the range -99 to 99.

Illegal Examples:		Error Flag
DC	E'10.03.49'	F (Format Error)
DC	E'10.03E4.0'	F
DC	E'2DA8E-30'	F
DC	E'10,000'	F
DC	E'1E-100'	T (Truncation Error)
CD	E'478E+100'	T

Numbers whose magnitude exceeds the largest possible number are converted to the largest floating-point number which is X'7FFF', X'FFFF' for positive values, and X'FFFF', X'FFFF' for negative values. Numbers whose magnitude is less than the smallest possible number are converted to true floating-point zero, which is X'0000', X'0000'.

Examples:		
	DC	E'7.3E+75'
	DC	E'5.3E-79'
	DC	E'1E-99'
	DC	E'1E+99'
	DC	E'73E+76'
	DC	E'123456E83'

5.2.2 DS - Define Storage

Name	Operation	Operand
A symbol optional	DS	An expression

The DS assembler instruction is used to reserve storage areas. The value of the expression in the operand entry determines the number of bytes reserved. If a symbol appears as a name, the value of the symbol is the location of the first byte reserved. No data is generated and the storage area reserved is not set to zero.

Example:

Name	Operation	Operand
INAREA	DS	80
OUTPUT	DS	TABLE1-TABLE2

5.3 Assembler Control Instructions

Assembler control instructions are used to control the location counter, the number of passes, between pass stops, printing and punching, conditional assembly, and assembly termination. None of these assembler instructions generate machine code instructions or constants in the object program.

5.3.1 OPT - Specify Options

Name	Operation	Operand
Not used	OPT	One or more operands separated by commas

The OPT statement must be the first statement in the program. The OPT assembler instruction is used to specify the following assembly options.

- Number of Passes: PASS1, PASS2, PASS3
- Printing: PRINT, NOPRNT
- Punching: PUNCH, NOPNCH
- Between Pass Stop: STOP, GO
- Program Label: LAB=ABCDEF

The options can appear in any order in the OPT statement.

If no OPT statement or specification of a particular option appears, the assumed options are as follows:

- PASS1
- PRINT
- NOPNCH
- STOP

Typical OPT statements might be:

Name	Operation	Operand
	OPT	PASS2, PUNCH, GO
	OPT	PUNCH, NOPRNT, PASS1
	OPT	STOP, PRINT, PASS3, PUNCH
	OPT	PUNCH, LAB=PROG3

-PASS1, One-Pass assembly option.

Specifying the PASS1 option, causes the source tape or cards to be read by the assembler once. The printed assembly listings and punched object tape are produced in accordance with the punching and printing options that have been specified.

For example:

Name	Operation	Operand
	OPT	PUNCH, PRINT, PASS1

will produce an assembly listing, punch the object code, and halt after one pass over the source statements. Object tapes from one-pass assemblies must be loaded by the General Loader. Therefore, PASS1 assemblies should only be specified when it is feasible to use the General Loader at load time.

-PASS2, Two-Pass assembly option.

Specifying the PASS2 option causes the source tape or cards to be read by the assembler twice. The printed assembly listing and punched object are produced during the second pass. As with the PASS1 option, they are produced in accordance with the punching and printing options that have been specified.

- PASS3, Three-Pass assembly option.

Specifying the PASS3 option causes the source tape or cards to be read by the assembler three times. The principle use of the PASS3 option is to produce two pass assemblies of a program using the teletypewriter. The three pass assembly is identical to the two pass except that the assembly listing is produced during the second pass and the object punched during the third pass.

- PRINT, Print assembly listing option

Specifying the PRINT option will cause the assembly listing to be printed during:

- the first pass of a one-pass assembly
- the second pass of a two-pass assembly
- the second pass of a three-pass assembly

- NOPRNT, No printing option

Specifying the NOPRNT option suppresses any printing of the assembly listing.

- PUNCH, Punch object option.

Specifying the PUNCH option causes the object program to be punched during:

- the first pass of a one-pass assembly
- the second pass of a two-pass assembly
- the third pass of a three-pass assembly

- NOPNCH, No punching option

Specifying the NOPNCH option suppresses punching of the object program.

- STOP, Stop after each pass option.

Specifying the STOP option causes the assembler to stop after each pass of the assembly.

- GO, Go to the next pass option.

Specifying the GO option causes the assembler to go immediately to the next pass of the assembly without operator intervention. This is useful when batching assemblies.

- LAB = nnnnnn

A program label can be 1 to 6 characters: the first character must be a letter, subsequent characters can be letters or digits. The program label is punched on the object tape in symbolic form. When using the General Loader, program labels are typed at load time. Note that program labels are appropriate only when the General Loader is used.

5.3.2 ORG - Set Location Counter

Name	Operation	Operand
Not used	ORG	A relocatable or absolute expression

The ORG assembler instruction is used to control the location counter. The ORG causes the location counter to be set to the value of the expression in the operand entry. The value is relocatable or absolute as determined by the expression.

The location counter is initialized to zero before each assembly. If no ORG assembler instruction appears at the beginning of the program, the location counter will begin at relocatable zero.

Symbols appearing in the operand of the ORG must be previously defined.

The ORG assembler instruction assures proper halfword alignment for any following machine instructions by always forcing the value of the location counter to be even. For example, the following two ORG statements produce a location counter value of X'019C'.

Name	Operation	Operand
	ORG	X'019D'
	ORG	X'019C'

To obtain a relocatable program, no ORG statement is necessary. A program can be made absolute at any time by using an ORG with an absolute operand, like ORG X'100'. Once a program is absolute, it can be made relocatable again by referring to a previously defined relocatable symbol. For example:

```
      OPT
START EQU  *            REL PORTION
      ORG  X'1000'      ABS PORTION
      ORG  START+100    REL PORTION
      END
```

5.3.3 DO - Conditional Assembly

Name	Operation	Operand
A symbol option	DO	A single expression

The DO assembler instruction causes the statement immediately following the DO statement to be processed as many times as specified by the value of the expression in the operand entry. If the value is zero, the next statement is skipped. The conditional assembly of instructions and generation of data is often used to configure standard programs at assembly time.

For example:

```
•
•
DO     CNFGR1
BAL    R15,SUBR1
DO     1-CNFGR1
BAL    R1,SUBR3
•
•
```

If CNFGR1 has a value of 1, the branch to SUBR1 will be generated. If the value of CNFGR1 is 0, the branch to SUBR3 will be generated.

5.3.4 END - End Assembly

Name	Operation	Operand
A symbol optional	END	An absolute or relocatable expression (optional)

The END assembler instruction terminates the assembly of the program. The value of the expression, if present, designates the place in the program where control is transferred after the program has been loaded. If an expression is not present, no automatic transfer of control takes place after loading.

An example follows:

Name	Operation	Operand
	ORG	100
PLACE1	LHI	R3, DATA2
	•	
	•	
	•	
	•	
LAST	END	PLACE1

The optional symbol, LAST, points to the next sequential halfword address beyond the object program. After loading this example program, Processor control is automatically transferred to location X'0100' (PLACE1).

APPENDIX 1

SUMMARY OF MACHINE INSTRUCTIONS

INSTRUCTION	TYPE	MNEMONIC	OPERAND FORMAT	OP CODE
Acknowledge Interrupt	RR	AIR	R1, R2	9F
Acknowledge Interrupt	RX	AI	R1, A(X2)	DF
Add Halfword	RR	AHR	R1, R2	0A
Add Halfword	RX	AH	R1, A(X2)	4A
Add Halfword Immediate	RS	AHI	R1, A(X2)	CA
Add with Carry Halfword	RR	ACHR	R1, R2	0E
Add with Carry Halfword	RX	ACH	R1, A(X2)	4E
AND Halfword	RR	NHR	R1, R2	04
AND Halfword	RX	NH	R1, A(X2)	44
AND Halfword Immediate	RS	NHI	R1, A(X2)	C4
Autoload**	RX	AL	R1, A(X2)	D5
Branch and Link	RR	BALR	R1, R2	01
Branch and Link	RX	BAL	R1, A(X2)	41
Branch on False Condition	RR	BFCR	M1, R2	03
Branch on False Condition	RX	BFC	M1, A(X2)	43
Branch on True Condition	RR	BTCR	M1, R2	02
Branch on True Condition	RX	BTC	M1, A(X2)	42
Branch on Index Low or Equal	RS	BXLE	R1, A(X2)	C1
Branch on Index High	RS	BXH	R1, A(X2)	C0
Branch Unconditional*	RR	BR	M1, R2	030
Branch Unconditional*	RX	B	A(X2)	430
Branch on Overflow*	RX	BO	A(X2)	424
Branch on Zero*	RX	BZ	A(X2)	433
Branch on Not Zero*	RX	BNZ	A(X2)	423
Branch on Equal*	RX	BE	A(X2)	433
Branch on Not Equal*	RX	BNE	A(X2)	423
Branch on Plus*	RX	BP	A(X2)	422
Branch on Not Plus*	RX	BNP	A(X2)	432

*Extended Branch Mnemonics

**GE-PAC 30-2 Instruction Only

APPENDIX 1

(Continued)

INSTRUCTION	TYPE	MNEMONIC	OPERAND FORMAT	OP CODE
Branch on Low*	RX	BL	A(X2)	428
Branch on Not Low*	RX	BNL	A(X2)	438
Branch on Minus*	RX	BM	A(X2)	421
Branch on Not Minus*	RX	BNM	A(X2)	431
Branch on Carry*	RX	BC	A(X2)	428
Compare Logical Halfword	RR	CLHR	R1, R2	05
Compare Logical Halfword	RX	CLH	R1, A(X2)	45
Compare Logical Halfword Immediate	RS	CLHI	R1, A(X2)	C5
Divide Halfword	RR	DHR	R1, R2	0D
Divide Halfword	RX	DH	R1, A(X2)	4D
Exclusive OR Halfword	RR	XHR	R1, R2	07
Exclusive OR Halfword	RX	XH	R1, A(X2)	47
Exclusive OR Halfword Immediate	RS	XHI	R1, A(X2)	C7
Floating-Point Add**	RR	AER	R1, R2	2A
Floating-Point Add**	RX	AE	R1, A(X2)	6A
Floating-Point Compare**	RR	CER	R1, R2	29
Floating-Point Compare**	RX	CE	R1, A(X2)	69
Floating-Point Divide**	RR	DER	R1, R2	2D
Floating-Point Divide**	RX	DE	R1, A(X2)	6D
Floating-Point Load**	RR	LER	R1, R2	28
Floating-Point Load**	RX	LE	R1, A(X2)	68
Floating-Point Multiply**	RR	MER	R1, R2	2C
Floating-Point Multiply**	RX	ME	R1, A(X2)	6C
Floating-Point Store**	RX	STE	R1, A(X2)	60
Floating-Point Subtract**	RR	SER	R1, R2	2B
Floating-Point Subtract**	RX	SE	R1, A(X2)	6B
Load Byte	RR	LBR	R1, R2	93
Load Byte	RX	LB	R1, A(X2)	D3

*Extended Branch Mnemonics
**GE-PAC 30-2 Instruction Only

APPENDIX 1
(Continued)

INSTRUCTION	TYPE	MNEMONIC	OPERAND FORMAT	OP CODE
Load Halfword	RR	LHR	R1, R2	08
Load Halfword	RX	LH	R1, A(X2)	48
Load Halfword Immediate	RS	LHI	R1, A(X2)	C8
Load Multiple**	RX	LM	R1, A(X2)	D1
Load Program Status Word	RX	LPSW	A(X2)	C2
Multiply Halfword	RR	MHR	R1, R2	0C
Multiply Halfword	RX	MH	R1, A(X2)	4C
No Operation*	RR	NOPR	R2	020
No Operation*	RX	NOP	A(X2)	420
OR Halfword	RR	OHR	R1, R2	06
OR Halfword	RX	OH	R1, A(X2)	46
OR Halfword Immediate	RS	OHI	R1, A(X2)	C6
Output Command	RR	OCR	R1, R2	9E
Output Command	RX	OC	R1, A(X2)	DE
Read Block	RR	RBR	R1, R2	97
Read Block	RX	RB	R1, A(X2)	D7
Read Data	RR	RDR	R1, R2	9B
Read Data	RX	RD	R1, A(X2)	DB
Sense Status	RR	SSR	R1, R2	9D
Sense Status	RX	SS	R1, A(X2)	DD
Shift Left Arithmetic	RS	SLHA	R1, A(X2)	CF
Shift Left Logical	RS	SLHL	R1, A(X2)	CD
Shift Right Arithmetic	RS	SRHA	R1, A(X2)	CE
Shift Right Logical	RS	SRHL	R1, A(X2)	CC
Store Byte	RR	STBR	R1, R2	92
Store Byte	RX	STB	R1, A(X2)	D2

*Extended Branch Mnemonics
**GE-PAC 30-2 Instruction only

APPENDIX 1
(Continued)

INSTRUCTION	TYPE	MNEMONIC	OPERAND FORMAT	OP CODE
Store Halfword	RX	STH	R1, A(X2)	40
Store Multiple**	RX	STM	R1, A(X2)	D0
Subtract Halfword	RR	SHR	R1, R2	0B
Subtract Halfword	RX	SH	R1, A(X2)	4B
Subtract Halfword Immediate	RS	SHI	R1, A(X2)	CB
Subtract with Carry Halfword	RR	SCHR	R1, R2	0F
Subtract with Carry Halfword	RX	SCH	R1, A(X2)	4F
Unchain**	RR	UNCH	R1, R2	90
Write Block	RR	WBR	R1, R2	96
Write Block	RX	WB	R1, A(X2)	D6
Write Data	RR	WDR	R1, R2	9A
Write Data	RX	WD	R1, A(X2)	DA

**GE-PAC 30-2 Instruction Only

APPENDIX 2
SUMMARY OF ASSEMBLER INSTRUCTIONS

Symbol Definition Instructions

EQU	Equate Symbol
ENTRY	Identify Entry-Point Symbol
EXTRN	Identify External Symbol

Data Definition Instructions

DC		Define Constant, used to specify the following data types
	-C	Character Constant
	-X	Hexadecimal Constant
	-A	Address Constant
	-H	Halfword Decimal Constant
	-E	Floating-Point Constant
DS		Define Storage

Assembler Control Instructions

OPT	Specify Options	
	- PASS1	One Pass Assembly
	- PASS2	Two Pass Assembly
	- PASS3	Three Pass Assembly
	- PUNCH	Punch Object Tape
	- NOPNCH	No Punching of Object Tape
	- PRINT	Print Assembly Listing
	- NOPRNT	No Printing of Assembly Listing
	- STOP	Stop After Each Pass
	- GO	Go, After Each Pass, to the next Pass
ORG	Set Location Counter	
DO	Conditional Assembly	
END	End Assembly	

ASSEMBLER OPERATING INSTRUCTIONS

1. GENERAL DESCRIPTION

The Assembler accepts source statements as described in the Assembler Manual, Publication Number 03-001R03A12. Refer to the Assembler Manul for an explanation of the source language. This document describes the operation procedures for the Tape Assembler, which accepts source statements from either a teletypewriter or a high speed tape reader. The operation of the Card Assembler is the same, except that source statements must be entered through a card reader.

2. CONFIGURATION

Both the Tape and Card Assemblers run on any GE-PAC 30 Processor with 8K or more of core memory. The Tape Assembler operates with either teletypewriters or high speed paper tape equipment for input-output. The Card Assembler requires a card reader for source inputs. The High Speed Arithmetic instruction repertoire is not required.

3. TAPE FORMAT

The Assemblers are provided as bootstrap tapes, as indicated by the M10 designation in the object tape part number. These tapes are loaded using the 8-bit loader at X'50'. Refer to the Bootstrap Programs and Procedures, Publication Number 06-030A12, for an explanation of the tape organization and loading sequence.

4. ASSEMBLER TAPES

There are four variations of the Assembler as follows:

Name	Tape Number
Tape Assembler/30-1	03-001R03M10
Card Assembler/30-1	03-004R03M10
Tape Assembler/30-2	03-008R03M10
Card Assembler/30-2	03-009R03M10

The Tape Assemblers read source input from a tape device: teletypewriter or high-speed paper tape reader. The Card Assemblers read source input from a card reader and are appropriate only with a Soroban Column-strobing card reader.

The 30-1 suffix implies that only 30-1 mnemonic op-codes, as defined in the Assembler Manual, Publication Number 03-001R03A12, are recognized. However, both 30-1 Assemblers can run on a 30-2 Processor.

The 30-2 suffix implies that in addition to recognizing the 30-1 mnemonic op-codes, the 30-2 Assemblers also recognize 30-2 mnemonic op-codes. See Appendix 1 for a summary of mnemonic op-codes that are pertinent only for 30-2. The 30-2 Assemblers also recognize the Assembler pseudo op Floating-Point Data Constant (DC); for example, DC E'789.163E-56'. Because of the memory required to expand the 30-2 Assembler's capability, the 30-2 Assembler Symbol Table is proportionately smaller than those of the 30-1 Assemblers. Refer to Table 2 under Section 9. Both 30-2 Assemblers run on a 30-1 Processor.

NOTE

Both 30-1 Assemblers and both 30-2 Assemblers run on any standard GE-PAC 30 Processor with 8K or more memory. The High Speed option is not required.

5. LOADING PROCEDURES

The Assembler bootstrap tapes should be loaded with the 8-bit loader at X'50'. The 50 Sequence, which includes this loader, is discussed in the first part of the Programming Manual, Publication Number 29-013. Prior to loading, the 50 Sequence must be entered into memory. Also the Binary Input Device Definition at X'78' must be set to select the desired loading device. Given the 50 Sequence in memory, the steps required to load the Assemblers are:

1. Place the bootstrap tape in the tape reader with the first

character over the read fingers or just preceding the photo diodes.

2. Set Data/Address Switches to X'50', set MODE CONTROL to ADRS, and depress EXECUTE.

3. Depress INITIALIZE.

4. Set the MODE CONTROL to RUN, and depress EXECUTE.

5. If a teletypewriter is being used as the load device, manually start the tape motion by moving the reader switch to Start or Run. When approximately a foot of tape has been read, the lower half of Display Register 2 flashes to indicate that the tape is actually loading. If this does not occur, check to see that the loading procedures were followed correctly.

6. If the reader stops and the Processor halts before the end of the tape is reached, an error has been detected. In this case, reposition the tape for the previous record gap and push EXECUTE to reread the previous record.

7. When all of the program has been loaded, the tape will stop, and Processor control is transferred directly to the Assembler at X'80'. The Assembler then halts, with the Wait light illuminated.

Note that during the bootstrap loading process, memory locations from X'1D00' to X'1F07' are used.

6. DEVICE SELECTION

The Assemblers use three halfwords in the Device Definition Table as follows:

Name	Location	Used For
BOUTDV	X'7A'	selection of the punch device
SINDV	X'7C'	selection of the source input device
LISTDV	X'7E'	selection of the list device

These halfwords must be set up prior to starting the assembler. These halfwords should contain information in the form:

0 7	8 15
Device No.	Output Command

The appropriate halfwords for various devices are shown below.

Teletypewriter Input	0294
Teletypewriter Output	0298
High Speed Paper Tape Input	0399
High Speed Paper Tape Output	039A
Card Input	04A0
Line Printer	0780

Various configurations are as follows:

X'7A'	0298	Teletypewriter punch device
X'7C'	0294	Teletypewriter source input device

X'7E'	0298	Teletypewriter list device
X'7A'	039A	High Speed tape punch device
X'7C'	0399	High Speed tape input device
X'7E'	0780	Line Printer
X'7A'	039A	High Speed tape punch device
X'7C'	04A0	Card reader source input device
X'7E'	0298	Teletypewriter list device

In the third configuration above, one of the Card Assemblers is required to handle source inputs from a card reader.

7. SOURCE TAPE FORMAT

Source statements can be any number of characters followed by a carriage return character. The characters should be represented in ASCII code, as defined in the Reference Manual, Publication Number 29-004. The most significant bit of each character is ignored by the assembler; therefore, either the 7-bit or 8-bit form of ASCII is acceptable.

The carriage return character terminates each statement. Line feed characters, and any non-printing characters are ignored by the assembler. Statements longer than 60 characters are truncated on input. That is, all characters between the sixieth character and the terminating carriage return are ignored. However, due to the listing format, no more than 56 characters per source statement are printed.

Statements should be separated on the source tape by at least 5 or 6 non-printing characters. This statement separation is required due to the start/stop characteristics of a teletypewriter tape reader. Source tapes typically used 10-12 rubout characters between records. Actually, any non-printing character other than carriage return will suffice.

The first statement in a program should be an option control (OPT) statement. This statement defines options from the following list:

PASS1, PASS2, PASS3
PRINT, NOPRNT
PUNCH, NOPNCH
STOP, GO
LAB XXXXXX

If no OPT statement is provided, the Assembler assumes PASS1, PRINT, NOPNCH, STOP, and provides no program label.

The last statement in a program must be an END statement. When operating, the Assembler reads the source tape until an END statement is encountered.

8. OPERATING PROCEDURES

Following the load of a bootstrap Assembler tape, control is transferred directly to the Assembler. The Assembler in this case performs some initialization and halts; push EXECUTE on the Display Panel to proceed with the assembly.

If the Assembler needs to be restarted, use the following procedure:

1. Set the Data/Address switches to X'80'.
2. Set the MODE CONTROL to ADRS.
3. Depress EXECUTE.
4. Set the MODE CONTROL to RUN.
5. Depress EXECUTE. The Assembler will halt.
6. Depress EXECUTE again.

When started, the Assembler prints the message

PASS1

on the list device, advances the paper several lines, and halts. Push EXECUTE to proceed with the assembly. Once the assembly is started, the procedures vary according to the number of passes specified in the OPT statement. Theses procedures are summarized in Figures 1, 2, and 3.

Prior to each pass, the Assembler types a message to identify which pass is next; also the message PREPARE PUNCH occurs preceding the punch pass. After printing these messages, the Assembler advances the paper several lines and halts. At this point, the operator should place the source tape in the reader, adjust the list device to top-of-form if printing is going to occur, prepare the punch if punching is going to occur, and depress EXECUTE on the Display Panel.

Operator	Assembler
Start at X'80'	
	Halts
Depress EXECUTE	
	Clears symbol table, types PASS 1 and halts
Bring paper to top-of-form, prepare punch device, put source tape in reader, and depress EXECUTE	
	Reads source tape and does 1-pass operation until END statement, prints out symbol table then

Figure 1. Operating Procedure – 1 Pass Assembly

NOTE

All punch devices are actually turned on under program control of the Assembler. The message PREPARE PUNCH implies only that the punch device be prepared. That is, the punch device must have tape inserted, and ample leader must be punched, etc.

The Assembler then reads the source tape and performs the operations appropriate to the current pass, as shown on Table 1.

Each pass proceeds until an END statement is encountered. When this occurs, the current pass is completed. When the END statement is read during PASS1, the Assembler prints out the Symbol Table. All undefined symbols are preceded by an error flag of U. The source program can be corrected at this time and a PASS1 restarted.

If an excess number of symbols is read by the Assembler, Symbol Table overflow is shown by listing the source statement that caused the overflow. These source statements are flagged with an S error flag. The printout on overflow occurs during PASS1 and also the print pass, when the PRINT option has been specified in the OPT statement.

If the current pass was not the final pass of an assembly, the next pass is identified with a message, and the assembler halts. The above procedures are repeated for each pass. If the completed pass was the final pass, the Assembler halts. In this case, if EXECUTE is depressed, the symbol table is cleared, the Assembler prepares itself for another assembly, prints PASS 1 and halts. Rather than proceed with another program, however, the Assembler can be restarted on PASS 2 or PASS 3 if desired. The restart addresses are as follows:

PASS 1	X'80'
PASS 2 of 2	X'A6'
PASS 2 of 3	X'AE'
PASS 3 of 3	X'C6'

TABLE 1. ASSEMBLER OPERATIONS

PASS NUMBER	ASSEMBLY TYPE		
	PASS 1	PASS 2	PASS 3
1	READ SOURCE PRINT LISTING PUNCH TAPE	READ SOURCE	READ SOURCE
2		READ SOURCE PRINT LISTING PUNCH OBJECT	READ SOURCE PRINT LISTING
3			READ SOURCE PUNCH OBJECT

Operator	Assembler
Start at X'80'	
	Halts
Depress EXECUTE	
	Clears symbol table, types PASS 1, and halts
Put source tape in reader, and depress EXECUTE	Reads source tape until END, prints out symbol table, types PASS 2, types PREPARE PUNCH, and halts.
Bring paper to top-of-form, prepare punch device, put source tape in reader, and depress EXECUTE	
	Reads source tape, does 2nd pass operations until END statement, then

Figure 2. Operating Procedure - 2 Pass Assembly

Operator	Assembler
Start at X'80'	
	Halts
Depress EXECUTE	
	Clears symbol table, types PASS 1, and halts
Put source tape in reader, and depress EXECUTE	
	Reads source tape until END, prints out symbol table, types PASS 2, and halts.
Bring paper to top-of-form, put source tape in reader and depress EXECUTE	Reads source tape, does 2nd pass operations until END, types PASS 3, PREPARE PUNCH, and halts
Prepare punch device, put source tape in reader, and depress EXECUTE	
	Reads source tape, does 3rd pass operations until END statement, then

Figure 3. Operating Procedure - 3 Pass Assembly

Restarting a pass may be appropriate if the tape breaks during punching, the paper jams during printing, etc.

9. SYMBOL TABLE SIZE

The symbol table limits are defined as follows:

X'82'	contains a pointer to the top of the symbol table, normally X'1FFF'
X'8A'	contains a pointer to the bottom of the symbol table

With all versions of the Assembler, the top of the symbol table is defined to be X'1FFF', which is the maximum address in an 8K memory. The bottom of the symbol table varies with each version of the Assembler as indicated in Table 2.

To change the symbol table limits, load the halfwords at X'82' and X'8A' with the desired limits and restart the Assembler at X'80'. The bottom limit of the table should not be lowered, since the Assemblers require from X'80' to the bottom limit shown in Table 2.

The maximum number of symbols that can be defined depends on the length of the symbols. One or two character symbols require 6 bytes of table space, three or four character symbols require 8 bytes, and five or six character symbols require 10 bytes.

10. ASSEMBLED OBJECT TAPE FORMAT

The assembled object tape is in standard loader format. Refer to <u>Loader Descriptions</u> Publication Number 06-025A12, under OBJEC TAPE FORMAT for a detailed description of standard loader format. The tape is absolute when an ORG statement is present with an absolute argument; otherwise the tape is relocatable.

TABLE 2

Version	Symbol Table Bottom Limit	Approximate Number of 4 character symbols permissable
Tape Assembler/30-1 03-001R03	X'1800'	$(256)_{10}$
Card Assembler/30-1 03-004R03	X'1940'	$(216)_{10}$
Tape Assembler/ 30-2 03-008R03	X'1A10'	$(190)_{10}$
Card Assembler/ 30-2 03-009R03	X'1B40'	$(152)_{10}$

APPENDIX 1
INSTRUCTIONS RECOGNIZED ONLY BY 30-2 ASSEMBLERS

Instruction	Type	Mnemonic	Op-Code
Autoload	RX	AL	D5
Floating-Point Add	RR	AER	2A
Floating-Point Add	RX	AE	6A
Floating-Point Compare	RR	CER	29
Floating-Point Compare	RX	CE	69
Floating-Point Divide	RR	DER	2D
Floating-Point Divide	RX	DE	6D
Floating-Point Load	RR	LER	28
Floating-Point Load	RX	LE	68
Floating-Point Multiply	RR	MER	2C
Floating-Point Multiply	RX	ME	6C
Floating-Point Store	RX	STE	60
Floating-Point Subtract	RR	SER	2B
Floating-Point Subtract	RX	SE	6B
Load Multiple	RX	LM	D1
Store Multiple	RX	STM	D0

APPENDIX 2
PROCEDURES FOR USER-DEFINED MNEMONICS

A feature has been added to the assembler that permits the user to define his own mnemonics for machine op-codes. This feature is especially useful for those users who have generated the micro-programming necessary for developing new machine instructions for the GE-PAC 30 Processors. This feature also permits the user to assign different mnemonics to already existing machine op-codes.

The method used to define new mnemonics to the assembler is the EQU statement. The format of the statement is as follows:

Name	Operation	Operand
New Mnemonic	EQU	A Constant

The name field of the EQU statement contains the user's desired new mnemonic. The new mnemonic may then be used in the operation field of any succeeding instruction statements. The user's new mnemonic may consist of from one to five characters, the first of which must be a letter and the others must be either letters or numbers. It cannot contain any special characters or blanks between characters.

The operand field of the EQU statement contains a constant, which when interpreted by the assembler, must have a 16-bit halfword value of the form (in hexadecimal):

n n x y

where nn = hexidecimal digits of an op-code

and	x = 0, y = 8	for one word (RR) instruction
or	x = 0, y = 2	for two word (RX or RS) instruction,
or	y = C	for one word extended (RR) instruction in which x is the condition code,
or	y = 3	for two word extended (RX or RS) instruction in which x is the condition code.

NOTE

It is suggested that the choice of nn = F0 be restricted to allow compatibility with the HEX-DEBUG programs' use of F000 for breakpoints.

Legal Examples

Name	Operation	Operand	Generates in Object Program
LOOP1	EQU	X'2208'	
	LOOP1	5,6	X'2256'
MOVE	EQU	X'3302'	
	MOVE	4,3(7)	X'3347',X'0003'
OP	EQU	X'89AC'	
	OP	4	X'89A4'
LINK	EQU	X'41F3'	
	LINK	2	X'41F0',X'0002'
UNCH	EQU	X'900C'	
	UNCH	0	X'9000'

NOTE

The UNCH instruction associated with the High Speed Interrupt Option in the 30-2 is not included in the Assembler Op-Code Table. In order to use the UNCH mnemonic, it must be defined by the user in an EQU statement as in the last example above.

Illegal Examples			Because
SAM	EQU	X'1238'	Third hex digit not 0.
	SAM	5,6	
FROG	EQU	X'5555'	Fourth hex digit not legal.
	FROG	4,3(7)	
OP	EQU	X'89AC'	Too many arguments.
	OP	4,5	
CALL	EQU	X'41F3'	Index value greater than 15.
	CALL	SAM (100)	

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER 1

GENERAL DESCRIPTION

1.1 INTRODUCTION

This manual describes the interactive FORTRAN system which operates on GE-PAC 30 digital systems. The system provides a direct mode for on-line evaluation of arithmetic expressions, and an editing mode for the creation and manipulation of stored programs. The system combines the convenience of a desk calculator with the programming power of FORTRAN.

In many FORTRAN systems designed for small computers, the programming is simple, but the mechanics of program preparation are complex. This is especially true of paper tape oriented machines for which it is necessary to go through the phases of compiling, object tape loading, and system subroutine loading. The handling of the paper tapes becomes laborious, when the whole procedure needs repeating for every program correction. In contrast, the GE-PAC 30 system needs no program preparation other than the entry of the source information; corrections can be made while the programs remain in core memory.

1.2 FEATURES

The system provides a set of FORTRAN operations and a set of system commands for editing, debugging, and control. The FORTRAN set is chosen to provide the greatest computational power for the least core space required for its implementation. The system is designed expressly for online use. The user communicates with the system through a teletypewriter. Features which assist the interaction between man and machine are:

1. Single character indications which request input and reflect the mode of the system are as follows:

←	direct-mode input request
*	edit-mode input request
=	data input request

2. All inputs are terminated by a carriage return. Until the terminating carriage return is received, no processing takes place, and the input line can be corrected or changed at will.

3. Command directives and FORTRAN operations can be abbreviated during input to minimize typing.

4. Commands are provided for listing all defined variables and for listing the names of all defined programs.

5. Commands for program creation, editing, and execution can be freely intermixed.

6. Error messages indicate the point in a program at which an error occurred during execution.

7. A performance improvement feature called FREEZE is provided. This operation alters the stored programs so that they become "more compiled", and results in a substantial decrease in execution time.

   The system is designed to operate in systems with 8K bytes or more of core memory, sufficient working space is available for the user to create and execute a FORTRAN program with 50-100 statements and 10-50 variables. Any available memory above 8K is used to expand the user's working space for more programs and data.

1.3 SCOPE OF MANUAL

This manual describes the GE-PAC 30 Interactive FORTRAN system for programmers who are familiar with FORTRAN. (For a tutorial discussion of FORTRAN in general, refer to one of the references listed in Appendix 3 of this manual.) For general information on programming GE-PAC 30 Digital Systems, refer to the Reference Manual (Publication Number 29-004) and the Programming Manual (Publication Number 29-013R01). Chapter 2 of this manual describes the various elements of this FORTRAN system. Chapter 3 explains the FORTRAN operations available. Chapter 4 describes the use of the system. Appendices provide a summary of instructions and some typical programs.

CHAPTER 2

ELEMENTS OF THE SYSTEM

2.1 INTRODUCTION

This Chapter describes the basic elements provided in the GE-PAC 30 Interactive FORTRAN System. A thorough knowledge of these elements is required in order to appreciate the Chapters which follow on FORTRAN Operations and Use of the System.

2.2 NUMBERS

The principal advantage of a FORTRAN system is its ability to manipulate numbers. The FORTRAN system described here uses real numbers only. There is no distinction made between integer, real, or complex-type numbers, and no special type-declarations are required.

Real numbers appear in various forms in this FORTRAN system. First there is the internal form that the computer uses. This affects speed and accuracy, but does not directly concern the programmer. Secondly, there is the external form used for input. This is the form that the programmer must use whenever he explicitly enters numbers. Finally, there is the external form used for output. This is the form that the system uses whenever it generates a number for the outside world. The external form for output is a subset of the external form for input. The forms of number representation are described in the following paragraphs.

2.2.1 External Number Representation For Input

Externally, numbers are represented in decimal in FORTRAN 'E format'. A decimal input number consists of an optional sign, up to six decimal digits that may include a decimal point, and an optional character 'E' followed by one or two decimal digits denoting a positive power of ten, or a minus sign and one or two decimal digits denoting a negative power of ten. A number field is terminated by any character than cannot legitimately occur in that field.

With this representation, the programmer has great freedom in specifying numbers. Typical numbers for input are:

-27

3.426

.00097

45

78.0

36.29E04 meaning $36.29*10^4$
-.0056E-8 meaning $-.0056*10^{-8}$

2.2.2 External Number Representation For Output

On output, numbers are also represented in decimal using 'E format'. However, for each internal number, there is one single output representation, and the rules for forming it are as follows:

1. A minus sign is written for a negative number. Nothing is written to denote the sign of a positive number.

2. If the number is in the range .1 through 10^6, the number is represented without an 'E'.

3. The number so represented is written using six or fewer digits, with a decimal point between the appropriate digits.

4. Trailing zeros are suppressed, as is a trailing decimal point.

5. For numbers outside the range .1 through 10^6, an 'E' is used. The number is expressed in the range .1 through 1, with a multiplying power of ten. In this case the part other than the power of ten is represented as indicated in Rule 3.

Some examples are:

15	Decimal point and trailing zeros suppressed.
-27.32	Two trailing zeros suppressed.
219000	Decimal point suppressed.
.123037	
-406.561	
.127E-21	Three zeros suppressed and 'E' used.
.231427E17	'E' used.

2.2.3 Internal Number Representation

All numbers are represented internally in floating hexadecimal form. This form uses a sign-and-magnitude representation, with a 24 bit fraction and a seven bit exponent. This number representation gives between six and seven decimal digits of precision.

The range of numbers stored internally is approximately $10^{\pm 76}$. Whenever the result of a computation exceeds the largest possible number, that result is forced to $\pm.723704 \times 10^{+76}$, which is the largest possible number. Whenever the result becomes less than the smallest possible number, that result is set to zero.

2.3 VARIABLES

The term variable is used in FORTRAN to denote a quantity that is referred to by name rather than be the appearance of a specific number. The value of a variable can be changed and is not restricted to one value. All variables -- like all numbers used in this system -- are real, and there is no distinction between real variables, integer variables, complex variables, etc.

The names of variables can be chosen by the programmer, but the following rules must be obeyed.

1. Variable names can be 1 or 2 characters in length.

2. The first character must be a letter.

3. The second character can be a letter or a digit.

4. A variable name should not duplicate an array name. Arrays pertain to subscripted variables and are discussed later.

Examples of proper variable names are:

X
A1
FF
R9

Variables are given a specific value with an assignment statement. For example:

A=1.5
B=-3E6

Assignment statements are discussed in Chapter 3. Once a variable has been assigned a value, that variable can be used to represent that value whenever it is needed. For example, if the variable A were assigned a value as above, the assignment

C=A

makes the variable C have the same value.

Note that the two character restriction on variable names is to conserve memory space. If a name longer than two characters is used, that name is truncated on input and only the first two characters are used. Names longer than two characters may be used therefore, providing no two names have the same first pair of characters.

2.4 EXPRESSIONS

The five basic arithmetic operations are represented with the symbols:

addition	+
subtraction	-
multiplication	*
division	/
exponentiation	**

These operation symbols can be used in combination with numbers, variables, and parentheses to form expressions. Examples of expressions are:

A+5.1
37*(X-3)
N**-3
(-X*4)**(4/(A±B))

Parentheses can be used to denote groupings and to define the order of operations to be performed. The meaning of the parentheses conforms to ordinary mathematical usage. For example, 2-3+4 = 3 and 2-(3+4) = -5.

Note that the minus sign can be used without a preceding operand or immediately following another operation symbol. This use of the minus is called unary minus since it operates on only one operand. The unary minus has the same effect as a multiplication of the operand by a negative one. For example:

3*-4 = -12
-8/-2 = 4
4**-1 = .25

When the order of operations is not completely defined by parentheses, unary minus operations occur first, followed by exponentiations, then all multiplications and divisions, and lastly additions and subtractions. Within a sequence of consecutive multiplications and divisions, or additions and subtractions, in which the order is not fully defined by parentheses, the operations are performed from left to right. For example:

2**3*4 = 32
2**(3*4) = 4096
3-4/2+2 = 3
3-4/(2+2) = 2
3-(4/2+2) = -1
(3-4)/2+2 = 1.5
(3-4)/(2+2) = -.25

Some special cases are as follows:

1. Dividing any number by zero will result in +.723704E76, the largest possible number.

2. Raising zero to any power will result in zero.

3. Raising a negative value to a power

   -V**P where V > 0

   results in V**P if P is within .5 of an even integer, and -(V**P) if P is within .5 of an odd integer. Note that this is an approximation, since a negative number raised to a non-integer power mathematically can yield a complex result.

2.5 ARRAYS

Arrays are used in FORTRAN to manipulate vectors and matrices. An array has a name by which it is referenced, and a set of values called elements of the array. Each element is identified by a number called a subscript. For this reason, arrays are often called subscripted variables. Arrays with this FORTRAN can use 1 or 2 subscripts; that is they can have one or two dimensions.

A typical reference to an array element is:

X(1,2)

where X is the name of the array. The numbers 1 and 2 are subscripts, and the presence of two subscripts indicates that X is a two dimensional array. Note that no space character or operator symbol appears between X and the following left parenthesis. The juxtaposition of the name and the left parenthesis is significant, and cannot be overlooked. For example X*(2) is an expression while X(2) is a reference to the 2nd element of the one dimensional array named X.

The rules for naming arrays are similar to the rules for variable names. They are:

1. Array names can be 1 or 2 characters in length.

2. The first character must be a letter.

3. The second character can be a letter or a digit.

4. Array names should not duplicate variable names or function names. Functions are discussed later.

All arrays in this FORTRAN are real, and there is no distinction between floating-point, integer, or complex-type arrays. Array elements are given a specific value with an assignment statement. For example:

X(1,2) = 45.3

The purpose of array elements is so that a single program or process can be repeated for many data values by putting the program inside a loop. Using arrays, different data values can be referenced simply by changing the value of the subscripts.

Note that subscript values can be specified with any expression. Typical references to array elements are

C5(N)
PQ(I-1,J+2)
Z(A*B+C,4)

As these examples suggest, it is the ability to identify subscripts with symbolic references or general expressions that makes arrays useful and convenient.

Array subscripts are, in general, integers. If the value specified is not an integer, however, the system rounds it to the nearest integer before it is used. Each resulting integer must be in the range of 1 to N, where N is the upper limit for that subscript defined with a DIMENSION statement. No array element can be defined or referenced until the size of the array has been specified. A typical DIMENSION statement is

```
DIMENSION    A(2,4), B(50)
```

in which the array A is defined as two-dimentional, with 8 elements, and subscript limits 2 and 4; the array B is defined as one-dimensional with 50 elements and subscripts in the range of 1 to 50. Note that the first subscript value is always 1, and the DIMENSION statement defines only the upper limit. DIMENSION statements are discussed further in Chapter 3.

2.6 FUNCTIONS

FORTRAN includes some built-in routines for the evaluation of certain mathematical functions. The functions can be utilized by referring to the name of a specific function and specifying an argument enclosed in parentheses. For example

```
COS(.5)
```

refers to the Cosine function and specifies the value .5 as the argument of the function. Note that the name must correspond exactly to the FORTRAN name for the given function; also, no spaces or operation symbols can appear between the name and the left parenthesis. The argument, however, can be specified by a single number, a symbolic variable, or any expression. The expressions in fact, can contain other references to functions. In other words, it is possible to nest function references. For example

```
COS(-.5)
COS(3*A+B)
COS(2*PI-COS(.3))
```

The functions provided in this FORTRAN are as follows:

FORTRAN Name	Function
SIN	Sine, Argument in Radians
COS	Cosine, Argument in Radians
ATN	Arctangent, Result in Radians
LOG	Natural Logarithm
EXP	Exponential to Base e

Note that:

SIN(X) and COS(X) should be avoided for X > 1000.

LOG(X) is illegal for negative X, and will result in an error message.

EXP(X) is evaluated as .7E76, the largest positive number, for all X greater than 174.

Note that no explicit square root function is provided. The square root of a number N can be computed by N**.5, or by using the expression

```
EXP(LOG(N)/2)
```

In general, the Rth root of N can be computed by

```
EXP(LOG(N)/R)
```

2.7 STATEMENTS

In FORTRAN, the unit of expression is the statement. There are two basic types of statements: system command and FORTRAN operations. The system commands are directives associated with program editing, debugging, and general use of the system. These statements are discussed in Chapter 4. FORTRAN operation statements are concerned with numbers, variables, arrays and expressions as discussed in this Chapter. The specific FORTRAN operations are discussed in the next Chapter.

Both types of statements have certain properties in common as follows:

1. Statements consist of a string of characters; the character set is that found on a teletypewriter keyboard.

2. Statements are of variable length, the end of the statement being indicated by a carriage return character (RETURN key on the keyboard).

3. The maximum length of a statement is 50 characters including the terminating carriage return, but excluding leading spaces; no means for statement continuation is provided.

4. Any statement beginning with the letter C followed by a blank character (space bar on the teletypewriter) is treated as a comment and is not processed in any way. The comment statement allows the programmer to write helpful remarks. Comments are of value in those cases where programs are prepared off-line. The system ignores comments, and comment statements are neither stored internally, nor subsequently listed.

5. The use of blank characters (spaces) is significant in the system, and attention must be paid to their use. Note that a string of consecutive blanks is always treated the same as a single blank. In general, leading blanks and trailing blanks are permitted, and have no affect on the processing of a statement. Blanks must be used following FORTRAN operation names, statement numbers, and within certain operation statements. In general, blanks should only be used where called for by the format of a specific command or operation. Refer to Chapter 3 and 4 for details.

A FORTRAN operation statement can include a statement number, which serves as a label so that other statements can refer to it. The cross-reference between statements is important for the transfer of control within a FORTRAN program. Programs are discussed in the next section. Note that the statement number is optional. When used, statement numbers must have the following properties:

1. Statement numbers can have one or two characters. Both characters should be decimal digits. The number 00 should not be used.

2. The statement number must appear first in the statement and be followed by one or more blanks. Blanks also can precede the statement number.

3. No two statements in a program should have the same number. Note that 0N is equivalent to N. Also, there is no sequencing implied by the statement number.

Some examples of statements with statement numbers are:

```
31    X=5+N
 7    DIMENSION    A(5)
99    A(3)=2
 9    N=N+1
```

System Command statements never use statement numbers.

2.8 PROGRAMS

A program is a set or series of FORTRAN operation statements. In this system, statements can be arranged and stored in groups called subroutines; the terms program and subroutine are equivalent. Each subroutine in the system has a name. Subroutines are identified by the SUBROUTINE system command. For example:

```
SUBROUTINE  AB
```

The argument of the SUBROUTINE command, in this case AB, is the program name. Program names must adhere to the following rules:

1. Program names can be 1 or 2 characters in length.
2. The first character must be a letter.
3. The second character can be a letter or a digit.
4. No two programs should have the same name.

A program can be of any length; that is a program can contain any number of FORTRAN operation statements. The end of a program is identified by the system command END. Details of a program creation and system commands are discussed in Chapter 4. A basic assumption in program organization is that statements are executed sequentially unless the flow of control is specifically changed.

Figure 2-1 shows a sample program which provides the general solution to two simultaneous linear equations as follows:

$$AX + BY = C$$
$$DX + EY = F$$

where X and Y are the unknowns. In the program, if there is no solution for the values given (if AE - BD = 0), the program will input a new set of values. The meaning of each FORTRAN operation is discussed in the following chapter.

```
     SUBROUTINE Q1
 3   ACCEPT  A,B,C
     ACCEPT  D,E,F
     N=A*E-B*D
     IF (N) 5,67,5
67   TYPE 'NO SOLUTION, TRY AGAIN'
     GO TO 3
 5   X=(C*E-B*F)/N
     Y=(A*F-C*D)/N
     TYPE 'X=', X, ' Y=', Y
     END
```

FIGURE 2-1. SAMPLE FORTRAN PROGRAM

CHAPTER 3

FORTRAN OPERATIONS

3.1 INTRODUCTION

This chapter describes the FORTRAN operations provided with this system. There are four types of operations: control, declarations, assignments, and input-output transfers. Table 3-1 summarizes the operation names in each class.

These operations are discussed in detail in the sections that follow. Arguments for these operations might be expressions, variable names, array names, program names, or statement numbers. It is important to note for each operation what type of arguments are appropriate.

This system provides two distinct modes of operation. In the direct mode, statements are immediately evaluated and the specified operation takes place. All system commands and some FORTRAN operations can be performed in the direct mode. In the edit mode, FORTRAN statements are not executed, but rather are stored for later execution. The edit mode is explained more fully in Section 4.2. The description of each operation states whether that operation can be executed in the direct mode.

TABLE 3-1. FORTRAN OPERATIONS

Type	FORTRAN Name	Purpose
Control	CALL	Execute a Program
	RETURN	Exit From a Program
	GO TO	Transfer To a Statement
	IF	Compare an Expression to Zero
	DO	Define a Set of Statements to Execute Repeatedly in a Loop
	CONTINUE	Define End of a "DO Loop"
Declaration	DIMENSION	Define Name and Size of Arrays
Assignment	Name=Value	Assign a Value to Named Variable or Array Element
Input-Output	TYPE	Print Values or Character Strings on the Teletypewriter Printer
	ACCEPT	Input Numbers From Teletypewriter Keyboard and Assign to Variables or Array Elements.
	WRITE	Transfer to assembly language output routine
	READ	Transfer to assembly language input routine
	FUNCTION	Transfer to assembly language function routine

3.2 CONTROL OPERATIONS

3.2.1 CALL P

This causes the subroutine named P to be executed. The word CALL must be followed by a blank. The argument P must be the name of a defined subroutine. If the subroutine named is not defined, the CALL operation will not be executed. If the called program executes a RETURN operation, control returns to the statement immediately following the CALL statement.

The CALL operation may be used in direct mode, and is the means for starting program execution. A RETURN in the program called from direct mode causes control to return to the user at the keyboard. The system will then type ← to indicate it is ready for new direct mode commands.

The CALL operation can refer to subroutines which themselves call subroutines. This technique is known as nesting. Subroutines can be nested to a level of 5, which means 5 successive CALL operations can be executed before a RETURN is required. An example of nesting is shown in Figure 3-1.

3.2.2 RETURN

This operation, as suggested previously, terminates the execution of the current subroutine, and causes control to return to the point from which the subroutine was called. The RETURN operation requires no arguments. There is an implicit RETURN statement following the last statement of every stored program. A RETURN statement is necessary, therefore, only when it is desired to exit from a subroutine at some place other than the last statement. RETURN should never be used in direct mode.

3.2.3 GO TO N

This causes control to transfer to statement N in the present program. The words GO and TO must be followed by a blank. The argument N must be a proper statement number, and that statement number must appear in the program If the statement number referenced does not appear in the program, the system will not execute the program. Specific error messages are discussed in Chapter 4. If the referenced statement number appears more than once in a program, the first one in the program will be utilized. The GO TO operation should never be used in direct mode. A typical GO TO statement is:

```
GO TO 37
```

```
←    CALL A          SUBROUTINE A
←                    X=X+1
                     CALL B             SUBROUTINE
                     Z=Y-1              Y=Z*X-1
                     RETURN             RETURN
```

FIGURE 3-1. NESTING EXAMPLE

3.2.4 GO TO (N1,N2,...), X

This statement, known as a computed GO TO, causes control to transfer to the statement indicated by N1 if the rounded value of the expression X has the value of 1. The words GO and TO must be followed by blanks. The arguments N1, N2, etc. must be proper and defined statement numbers. The argument X can be any arithmetic expression. If the rounded value of X is less than 1, or larger than the number of statement labels provided, the operation is not performed and an error message results. The computed GO TO statement should never be used in direct mode. A sample program using the GO TO operation is shown below.

```
  N=N+1
  GO TO (3,4,5),N
3 R=N
  GO TO 9
4 R=3*N-1
  GO TO 9
5 R=5*N**2-4*N+3
9 T=R/2
```

3.2.5 IF (X) N1,N2,N3

This operation compares the value of the expression X to zero. If the value is less than zero, the statement indicated by N1 is executed next; if the value is equal to zero, statement N2 is executed next; if the value is greater than zero, statement N3 is executed next. The word IF and the expression in parentheses must be followed by blanks. The argument X can be any expression. The arguments N1, N2, and N3 must be proper and defined statement numbers. If the statement numbers referenced do not appear in the program, the system will not execute the program. If the referenced statement numbers appear more than once in the program, the first one in the program will be utilized. The IF statement should never be used in direct mode. A program using IF is as follows:

```
92 N=N+1
   IF (N-6)   92,92,3
 3 K=N**2
```

3.2.6 DO N V=L1, L2

This statement defines a set of statements, from the one immediately following this statement down to and including the statement indicated by N, to be executed repeatedly in a loop. The statement numbered N must be a CONTINUE statement. The number of times to repeat the loop is defined by assigning to the index variable V a lower limit L1 and an upper limit L2. For the first execution of the statements, V has the value L1; for each succeeding iteration of the loop, V is incremented by 1. The loop continues until the statements have been executed, with V having the largest value satisfying the expression:

$$V \leq L2+.5.$$

where V and L2 are rounded to the nearest integer for purposes of the comparison.

The word DO and the statement number N must be followed by blanks. The argument N must be a proper and defined statement number. The argument V can be any proper variable name or array element. The arguments L1 and L2 can be any expression. The argument L2 must be less than 65,536.

An example using the DO operation is shown below.

```
  DO   3   I=1,5
  A(I)=I**2
3 CONTINUE
```

In the example, the loop is repeated for I=1,2,....,5. Observe that the index variable (in this case I) can be referenced within the loop. It is this feature that makes the DO statement very powerful. Another example is:

```
   DO 81 B=1.3,9.6
   X=X+B
81 CONTINUE
```

In this example, the loop is repeated for B=1.3, 2.3, ..., 10.3. As this example shows, while the increment value is always the integer 1, the lower and upper limits do not need to be integer values. Another example is:

```
     N=3
     DO 17 P= -2, N+1
     X=X*P
     N=N+1
  17 CONTINUE
```

As seen above, the limit values can be positive or negative. In this case, the loop is repeated for P= -2, -1, 0, ... , 4. Note that the limit values are calculated and fixed when the DO statement is first encountered. Even though the variable N changes its value in the loop, the upper limit remains at 4. Also, if the termination condition is immediately satisfied when the DO statement is first encountered, the statements following it are executed once.

DO statements may be nexted to a maximum of four levels. A nested DO is one that lies wholly within the range of another DO loop. This configuration is very common when programming matrix operations. It is illegal, however, to have an inner DO whose terminating CONTINUE statement lies beyond the corresponding statement of the outer DO. It is also illegal to transfer control into range of a DO from outside, unless control got outside in the first place by means of a transfer from within the range of the DO. A single CONTINUE statement can terminate a set of nested DO loops. An example is:

```
  DO    3    I=1,4             I Loop
  DO    3    J=1,4             J Loop
  N=4*(I-1)+J
  DO    6    K=1,3             K Loop
  A(I,J)=A(I,J)+B(K)
6 CONTINUE
  C(N)=A(I,J)
3 CONTINUE
```

3.2.7 CONTINUE

This operation, as shown above, defines the end of a DO loop. No arguments are required with this statement. CONTINUE can be used any place in a program, independent of DO loops. When not associated with any DO loops, the statement is treated as a null operator, and it has no effect on the program. A CONTINUE statement can be identified with a statement number just like any other statement. Neither CONTINUE nor DO should be used in direct mode.

3.3 DECLARATIONS

3.3.1 DIMENSION A(L), B(M, N), . . .

This operation defines the names of arrays, their dimensions, and the number of elements per dimension for each named array. The word DIMENSION must be followed by a blank, and no other blanks should be used with this statement. The names A, B, etc. must be proper array names; the arguments L, M, N, etc. can be any expression. The system will round the value of the expressions to the nearest integer when the statement is executed. The rounded values must be equal to or greater than one. Arrays can be either one or two dimensional.

Some examples are:

```
DIMENSION R(35)
DIMENSION AR(3, N+6), BR(X*N)
```

When the DIMENSION statement is executed, the system checks to see whether the arrays referenced are already defined. If the referenced array is not defined, the system determines if enough space is available in memory to store all the elements of that array. If insufficient memory is available, the necessary memory space is allocated for the array and the system then repeats the process for the next array named.

If the named array is already defined when DIMENSION is executed, the current memory allocation is not changed. If more elements are indicated than the system has space for, an error message results. If the specified number of elements fit into the allocated space, memory is unchanged and no message occurs.

To change the size of a defined array, first ERASE the entire array, and then DIMENSION it to the new size. The ERASE operation is discussed in Section 4.3. The DIMENSION statement can be used in direct mode.

3.3.2 Other Declarations

No other declarations are required in this FORTRAN system. The declarations

```
REAL ----
INTEGER ----
COMMON ----
FORMAT ----
```

which are required in some FORTRAN systems are ignored if they occur, and have no effect on the system. This means that a FORTRAN program written for a more comprehensive system could be executed on this system as well, providing no conflicts exist with variable names.

3.4 ASSIGNMENTS

Assignment statements have the form

```
V=X
```

where V is the name of a variable or array element, and X is any arithmetic expression. This statement tells the system to replace the value of the variable named on the left with the value of the expression on the right. No blanks should be used in the arithmetic statement. A blank character following the name V or the equals sign will cause the assignment to be improper. Typical assignment statements are:

```
X3=2*3.14/N
A(3)=H+SIN(X)
Q(I,J-1)=I*J-3
```

An array should be defined with a DIMENSION statement prior to assigning a value to any elements of the array. If an array was previously defined, the system checks the subscript to see if it is within the defined size of the array. If it is not, no assignment is made and the user is informed with an error message. If the named array was not defined prior to the assignment operation, the system will then define an array with the name and size indicated. Note that after an array A is defined, referring to A without subscripts implies the first element of that array. That is:

```
A=A(1)=A(1,1)
```

Assignments can be made any time in direct mode or in stored programs. When an assignment statement is used in a stored program, however, the specified assignment is not made until the program is executed; the assignment does not occur when the statement is first entered into the program.

An alternate way of assigning values to variables or array elements is with the ACCEPT statement which reads numbers from the teletypewriter keyboard. This operation is discussed in the next section.

3.5 INPUT AND OUTPUT

Input and output for the teletypewriter is performed using the statements ACCEPT and TYPE. For all other devices, the statements READ, WRITE, or FUNCTION can be used for linking to assembly language routines which drive the devices.

Input and output is performed without the use of FORMAT statements. Input formats are free, and output formats are implied. For compatibility, FORMAT statements which may occur because a program has been run on another system are skipped over, and ignored.

The FORTRAN system is constructed so that as peripheral units are attached to the computer, they can be operated with READ, WRITE, or FUNCTION statements. These units may be conventional peripheral devices such as magnetic tape units, or they may be A-to-D converters, multiplexors, etc. The basic FORTRAN, however, does not contain any of the required I/O driver subroutines.

Note that there are several versions of FORTRAN available, each tailored to certain machine configurations. The capability for linkage to user-supplied assembly language routines via READ, WRITE, or FUNCTION statements is not available in all versions of FORTRAN. This capability is provided by a supplement called the RWF Expansion. For a definition of the various versions of FORTRAN, refer to the Operating Instructions for Interactive FORTRAN, Publication Number 03-005A16. For a discussion of the details of the assembly language linkage, refer to Operation Procedures for FORTRAN W/RWF Expansion, Publication Number 03-011A16.

All input-output operations can be performed in direct mode except where noted.

3.5.1 TYPE A, B, . . .

This operation causes one line of information to be printed on the teletypewriter. Each line is followed by a carriage return and line feed. The word TYPE should be followed by a blank. The arguments A, B, etc. can be expressions or character strings enclosed in quote (') marks. For example:

```
TYPE    X,2*N,'FT PER SEC'
```

For each expression, the system types out the value of the expression. For each character string, the system types the characters exactly as they appear between the quotes.

Each value to be printed is allocated a field of 18 spaces on the teletypewriter printer. Each value appears in decimal output format as defined in Section 2.1.2; the decimal format requires from 1 to 12 spaces depending on the value. For example, an integer printout

```
1
```

requires only once space, while the number

```
-.12345E-12
```

requires 12 spaces. After the value is typed, and if another value is to be typed immediately following, the system spaces over to satisfy the 18 space field width. If the next argument of the TYPE statement is a character string, however, no spaces follow the value and the character string begins immediately.

For example, the statement

```
TYPE    2*2,1/3,4-6
```

yields the output

```
4                 .333333           -2
```

with 17 spaces between the first two values and 11 spaces between the last two values. The statement

```
TYPE    2*2,'ABC',1/3,4-6
```

yields

```
4ABC.333333           -2
```

with no spaces after the 4, no spaces after the C, and 11 spaces after the last 3.

A typical type statement, where X=5, is:

```
TYPE    'X=',X,' UNITS'
```

which results in the print out:

```
X= 5 UNITS
```

Some restrictions on the TYPE statement are:

1. TYPE with character string arguments cannot be used in direct mode.

2. If only expression arguments are used, no more than four arguments can appear with one TYPE statement since the teletypewriter printer is 72 characters wide.

3. If expression and character string arguments are mixed, no more than seven arguments can be used with one TYPE statement.

4. A TYPE statement with no arguments should be avoided. Use TYPE ' ' to achieve a blank line print out.

3.5.2 ACCEPT A, B, ...

This operation causes the system to read one line of data from the teletypewriter. The numbers specified by the data are assigned to the variables A, B, etc. The word ACCEPT must be followed by a blank. The arguments A, B, etc. can be variable or array element names. When the ACCEPT statement is executed, the system types an equal sign (=) at the left margin of the teletypewriter to indicate that one line of data is needed. The data can contain one or more numbers; numbers must be separated by a blank or a comma. In general, the data entered from the keyboard should correspond to the number of arguments in the ACCEPT statement. That is, if the ACCEPT statement specifies 4 arguments, the data line should include 4 numbers. Up to seven arguments can be used with each ACCEPT statement. As an example, when the statement

```
ACCEPT X, P(1)
```

is executed, the teletypewriter entry

```
54.5, -.2E6
```

followed by a carriage return results in setting the variable X to 54.5 and the array element P(1) to -.2E6.

A TYPE statement can be used preceding an ACCEPT statement to identify the variable name. For example, the statements

```
TYPE 'DEFINE N'
ACCEPT N
```

would appear on the teletypewriter as

```
DEFINE N
=
```

after which the value for N could be typed.

An array should be defined with a DIMENSION statement prior to referencing any array elements with an ACCEPT statement. No provision is made for reading all the elements of an array with one statement. A DO loop must be used for this purpose. For example, the program

```
  DIMENSION P(10)
  DO 3 I=1,10
  TYPE 'P(',I,')'
  ACCEPT P(I)
3 CONTINUE
```

reads 10 values and assigns them to the elements of the array P.

3.5.3 WRITE X, A, B

The WRITE statement is similar to the TYPE statement, except the output is performed by an assembly language driver routine supplied by the user. The argument X is used as a switch or device number that the user's program decodes. A buffer of ASCII characters is generated for the values of the arguments A, B, ... The argument X must be an expression with a numeric value. The arguments A, B, ... can be symbol names, numeric literals, expressions, or character strings.

When the WRITE statement is executed, FORTRAN generates a buffer of characters for the arguments A, B, etc. The data format conforms exactly to that of the TYPE statement. The value of the argument X is evaluated and integerized to facilitate testing by the user. FORTRAN then gives control to an assembly language routine whose address is contained in ORG + X'E', where ORG is the first location of FORTRAN. The user supplies this program and sets this address.

When the user-supplied program is entered, all the parameters of interest are supplied in the machine registers. Refer to Publication Number 03-011A16 for details.

3.5.4 READ X,A,B

The READ statement is similar to the ACCEPT statement, except the input is performed by an assembly language routine supplied by the user. The argument X is used as a switch or device number that the user decodes. The data read is stored as values of the arguments A, B, etc. The arguments X, A, B, etc. must all be variable names or references to array elements; they cannot be literals, expressions, or character strings.

When the READ statement is executed, the value of X is integerized to facilitate testing by the user. FORTRAN then gives control to the assembly language routine whose address is contained in ORG + X'C', where ORG is the first location of FORTRAN. The user supplies this program and sets this address. The user-written routine should read a record of information into a buffer. This data will be processed exactly the same as with an ACCEPT statement. When the user program is entered, all parameters of interest are supplied in machine registers. Refer to Publication Number 03-011A16 for details.

3.5.5 FUNCTION (X,A,B,...),C,D,...

The FUNCTION statement is for use in linking to any general purpose machine language routine. The first group of arguments (X, A, B,...) must be variable names or references to array elements. The second group of arguments C, D,... can be any expression - variable names, numeric literals, character strings, etc. If the first group contains only one argument, no parentheses are required.

Examples of proper FUNCTION statements are:

```
FUNCTION  (X, BF),3.5,SIN(2)
FUNCTION  (X, V1, V2),4*N
FUNCTION  X,3.5,'HELP'
```

When a FUNCTION statement is executed, all arguments are processed, and an argument list is generated according to the arguments provided. Each argument from the first group is converted to an address which locates the specified variable or array element in memory. Each argument from the second group is evaluated and converted to a value. FORTRAN then gives control to the assembly language routine whose address is in ORG + X'10', where ORG is the first location of FORTRAN. The user supplies this program and sets this address. When the user program is entered, all parameters of interest are supplied in machine registers. Refer to Publication Number 03-011A16 for details.

CHAPTER 4

USE OF THE SYSTEM

4.1 INTRODUCTION

This interactive FORTRAN is operated and controlled from the teletypewriter keyboard. The system presents two distinct modes to the user. The direct mode, which is characterized by the arrow character (←) in the left margin, permits on-line assignment of variables, evaluation of expressions, etc. The edit mode, which is characterized by an asterisk (*) in the left margin, allows the creation and modification of stored programs. When the system is started, the user is given control at the keyboard in direct mode.

The user converses with the system in statements - either FORTRAN operation statements or system commands. Until the RETURN key, which terminates a statement, is depressed, no processing takes place, and the input line can be corrected or changed at will. The left arrow (←) key can be used anytime to erase the last character in the line. The RUB OUT key can be used anytime to erase the current line. When RUB OUT is depressed, the system types a hash mark (#) to confirm the erasure, and advances one line so that the current statement can be retyped. This chapter discusses the details of program editing, system command, error messages, etc.

4.2 PROGRAM EDITING

The commands used for program editing are:

SUBROUTINE	Define a new subroutine or refer to existing subroutine.
OPEN	Refer to a specific statement of referenced subroutine.
LIST	List either entire subroutine or one statement of a subroutine
DELETE	Delete either entire subroutine or one statement of a subroutine
END	Terminate editing sequence

These commands, except for SUBROUTINE, can be abbreviated to the first two characters to minimize typing. The SUBROUTINE command can be abbreviated to the first four characters (SUBR).

The basic procedure for creating a new program of name AB is to enter the command:

```
SUBROUTINE    AB
```

This command defines a new program with name AB, and puts a single RETURN statement in the program. The RETURN statement is then established as the <u>open</u> statement, which places the system in the edit mode. The system then types * to indicate the edit mode. Note that the edit mode implies that some statement in a stored program is open.

In the edit mode, the insertion of statements is implicit. That is, whenever a FORTRAN operation statement is entered during edit mode, that statement is inserted immediately <u>before</u> the open one. Further, the same statement remains the open one, allowing a set of statements to be inserted at the chosen place. The END command returns the system to direct mode.

A typical editing sequence is

```
← SUBROUTINE  AB
*  5  ACCEPT   N
*  X=3+EXP(N)
*  TYPE  X
*  END
←
```

The final ← typed by the system implies direct mode which means no statement is open. To list the subroutine, the command

```
SUBROUTINE  AB
```

should be used to identify the program of interest. This time, since a program of name AB already exists, a new definition is not necessary, and no statement is opened. Rather, the subroutine as a whole is considered open. The command LIST then lists the entire program.

```
←  SUBROUTINE  AB
←  LIST
            SUBROUTINE  AB
          5 ACCEPT  N
            X=3+EXP(N)
            TYPE X
            END
←
```

Observe that the last RETURN statement is listed as END. After the LIST operation, the entire program is still considered open. A specific statement can be opened with the command

```
OPEN  5
```

which opens statement 5. The following sequence shows how to insert a new statement after the TYPE statement.

```
← OPEN     5,3
                   END
*  GO TO 5
*  END
←
```

In this sequence, the OPEN statement specifies the 3rd statement after statement 5. The system listed this statement accordingly, and left the statement open, placing the system in edit mode. The GO TO statement is then inserted, and the END command brings the system back to direct mode.

To change a statement, it must be opened and then deleted. The new statement can then be inserted in its place. For example

```
←SUBROUTINE   AB
←OPEN   5,1
               X=3+EXP(N)
*  DELETE
               TYPE X
*  X=EXP(N+3)
*  END
←
```

Again, the OPEN statement specifies the first statement after statement 5. This statement is listed by the system and it remains open. The DELETE command then deletes the open statement, after which the system opens the next statement and lists it. The new assignment statement is inserted and the END command restores the direct mode. The resulting subroutine is listed as follows:

```
←      SUBROUTINE   AB
←      LIST
            SUBR     AB
          5 ACCE       N
            X=EXP(N+3)
            TYPE X
            GO TO 5
            END
←      DELETE
←
```

After the LIST operation, the whole subroutine is open. In this case, the DELETE command, as shown, deletes the whole program from memory.

Table 4-1 shows a sequence of editing operations. Each of the editing commands is described in later paragraphs.

4.2.1 SUBROUTINE A

The word SUBROUTINE must be followed by a blank. The argument A must be a proper name. The rules governing program names are discussed in Section 2.8. The effect of this command depends on whether program A already exists.

TABLE 4-1. SAMPLE EDITING SEQUENCE

Editing Sequence	Explanations
← C SAMPLE PROGRAM	Comments are Ignored.
← SUBROUTINE A	Define Program A.
* END	Return to Direct Mode.
← SUBROUTINE A	List Program A.
← LIST	
SUBR A	
END	
← OPEN 0	Open First Statement.
END	
* X=X+1	Insert 2 Statements.
* TYPE X	
* OPEN 0	Open First Statement.
X=X+1	
* DELETE	Delete It.
TYPE X	
* X=0	Insert 2 More.
* 3 X=X+1	
* LIST	List Statement After the Open Statement.
END	
* IF (X-10) 3,3,4	Insert 2 More.
* 4 TYPE 'OK'	
* END	Return to Direct Mode.
← SUBROUTINE A	
← LIST	List Complete Program.
SUBR A	
X=0	
3 X=X+1	
TYPE X	
IF (X-10) 3,3,4	
4 TYPE 'OK'	
END	
←	

If no such program exists, a program with name A is defined. In the new program, a single RETURN statement is established. This RETURN statement is always shown as END when the program is listed. The RETURN statement is made the open statement, and the system types * to reflect the edit mode.

If a program named A already exists, the system remains in the direct mode, but the whole program is opened. This particular state of the system, when an entire program is open rather than any particular stored statement, can be thought of as a special case of the direct mode. In this state:

LIST	lists the entire program and leaves the state of the system unchanged.
OPEN	opens a particular statement within the open program and puts the system into the edit mode.
DELETE	deletes the entire program and leaves the system in direct mode with no program open.
END	closes the open program and leaves the system in direct mode with no program open.

4.2.2 OPEN N

This statement opens the statement labeled N in the open program. The opened statement is listed on the teletypewriter and the system is left in the edit mode. The OPEN command can be used in the edit mode, or in the direct mode if a whole program has been opened with a SUBROUTINE statement. The word OPEN should be followed by a blank. The argument N should be a proper statement number. The statement OPEN 0 will open the first unlabeled statement in the open program following the SUBROUTINE statement. An alternate form of the statement is

```
OPEN  N,X
```

in which X can be any expression. The value of X, which is rounded by the system to the nearest integer, must be positive or zero. This command opens the Xth statement past that which has the statement number N.

For example,

```
OPEN  67,2
```

opens the second statement past the one with label 67.

When a statement is open, the system is in the edit mode, as shown by the * character printed on the teletypewriter. Any FORTRAN operation statements typed in this mode are inserted into the stored program immediately before the open statement.

4.2.3 LIST

The effect of this command depends on the state of the system. If a whole program has been opened with a SUBROUTINE statement, that program is listed in its entirety. The state of the system is unchanged after the LIST operation. If the system is in the edit mode with a particular statement open, the next statement is opened and listed on the teletypewriter. In the edit mode, therefore, a succession of LIST commands causes successive statements of the program to be listed.

When statements are listed, they are moved over several spaces and lined up for readability. The listed version of a statement may differ from its input form in the following ways:

1. All variable names, array names, and statement numbers are truncated to two characters.

2. All numbers are expressed in the output format (see Section 2.2.2) which, in some cases, differs from the input format. For example, the entered number 5.6E7 will be listed as .56E8.

3. The FORTRAN operation names may be abbreviated to save time and space.

The form in which statements are listed is suitable for input. When an entire program is listed, the first line is a SUBROUTINE statement, and the last line is an END statement. The system also accepts the abbreviated form of FORTRAN operation names. Listing a program with the teletypewriter punch turned on will produce a paper-tape copy of the program. The paper-tape can be used later to reload the program into memory. Refer to Section 4.4.4 for details.

4.2.4 DELETE

The effect of this command also depends on the state of the system. If a whole program has been opened with a SUBROUTINE statement, the entire program is deleted. The system is left in direct mode with no program open. If the system is in edit mode with a particular statement open, the open statement is deleted. The next statement is then opened and listed. In the edit mode, therefore, a succession of DELETE commands causes successive statements of the program to be deleted.

4.2.5 END

This command terminates an editing sequence. Any open programs or statements are closed, and the system returns to direct mode. If there is ever any doubt as to the current state of the system, the END command can be used to unconditionally restore the direct mode.

4.3 SYSTEM COMMANDS

The previous section discusses those system commands associated with program editing. Other system commands are:

CLEAR	Delete all stored programs, arrays, variables, etc.
PROGRAMS	List the names of all programs currently in memory.
VARIABLES	List the names and values of all defined variables and arrays.
ERASE	Delete a specific variable or array from memory.
FREEZE	Specify "Freeze" mode for faster program execution.
UNFREEZE	Specify normal "unfrozen" mode for program debugging convenience.

These commands are recognized in either direct or edit mode. The names of these system commands can be abbreviated to the first two characters to minimize typing.

4.3.1 CLEAR

This command deletes all programs, variables, and arrays from memory, and returns the system to an initialized state in direct mode.

4.3.2 PROGRAMS

This command lists the names of all programs currently in memory. The names are listed in alphabetical order. If no programs are defined, nothing is printed. This operation does not change the state of the system. That is, if this command is used in edit mode, the names are listed, and the system remains in the edit mode. For example

```
← PROGRAMS
AB
K9
X
←
```

4.3.3 VARIABLES

This command lists the names and values of all defined variables in alphabetical order, one per line. For arrays, the array name is repeated for each element of the array. The names and values are spaced over on the page for readability. For example:

```
← VARIABLES
            A=45.6
            B3=-2
←           RT=.345E6
```

If no variables are defined, nothing is printed. This operation does not change the state of the system. If the command is used in edit mode, the names and values are listed and the system remains in the edit mode.

4.3.4 ERASE A

This command deletes the variable or array named A from memory. The word ERASE should be followed by a blank. The argument A should be a proper variable or array name. If the referenced variable or array is not defined when this command is used, the memory is unchanged. Only one argument can be specified with each ERASE command. The system always returns to direct mode after an ERASE operation.

Note than when A is an array name, the entire array is deleted, not just one element of the array. This operation is useful if it is necessary to change the size of a defined array. In this case, the array first must be deleted with an ERASE command, and then redefined with a DIMENSION statement. This command also is useful for regaining memory space used by variables and arrays that are no longer needed.

4.3.5 FREEZE/UNFREEZE

A special mode known as the Freeze mode, in which program execution is speeded up, is provided. Since programs are stored in symbolic form, all names and statement numbers have to be repeatedly looked up in memory during program execution in the normal "unfrozen" mode.

When programs are executed with the Freeze mode in effect, however, they are scanned and all symbolic references are replaced with address references. The resulting program can be executed at a much faster rate. When the execution terminates, the symbolic references are automatically re-established. To the user, the program always appears in symbolic form.

When the system is initialized, or when the CLEAR command is used, the mode is set to normal. The FREEZE command puts the system into the Freeze mode. The UNFREEZE command restores the normal mode. Note that the FREEZE command simply defines the mode. The program is not actually altered (frozen) until execution begins. Thus, programs can be edited freely even though the Freeze mode prevails.

To execute a program in Freeze mode, all variables and arrays must be defined prior to the CALL operation which starts the execution. For arrays, the DIMENSION statement must be executed, but all the elements need not be defined. If all variables and arrays are not defined, an error message will result from the CALL statement, and the program will not be executed. Refer to Section 4.5 for details on error messages. One way to define all variables and arrays used within a program is to execute the program. Another way would be to use DIMENSION and assignment statements in direct mode before executing the program. In general, the best strategy is to create and debug programs in the normal mode. Once a program is operational, use FREEZE for faster execution.

4.4 CONSOLE PROCEDURES

This FORTRAN is designed especially for on-line use. The primary advantage of an interactive system is the ability to intermix the creation, debugging, and execution phases of a programming task.

With the features provided, the user can converse freely without fear of making mistakes. When a typing error is made, the user can correct it easily. When programming errors occur, the user is informed with an error message. In no case can a programmer cause loss of control or interfere with the integrity of the system. Details of the user-machine interaction are discussed in this section.

4.4.1 Teletypewriter Features

Single character indicators are used to request input and reflect the mode of the system. They are:

←	direct mode input
*	edit mode input
=	data input

Statements, which can be variable length, are terminated with a carriage return. Until the RETURN key is depressed, no processing takes place and any typing errors can be corrected.

The left arrow (←) key can be used any time to erase the last character. The RUB OUT key can be used any time to erase the current line. The system will type # in response to RUB OUT to confirm that the line was erased.

The LINE FEED character is always ignored on input. The LINE FEED can be used anytime for page formatting without ill effect. Similarly, a null statement (single RETURN only) is ignored by the system.

Should an illegal character (such as $ or @ be used by accident, the system will detect this when the line is processed and inform the user by typing a question mark (?).

Many commands can be abbreviated to minimize typing. Acceptable abbreviations are:

DIME	for	DIMENSION
CONT		CONTINUE
RETU		RETURN
GO		GO TO
ACCE		ACCEPT
WRIT		WRITE
FUNC		FUNCTION
SUBR	for	SUBROUTINE
OP		OPEN
LI		LIST
DE		DELETE
EN		END
CL		CLEAR
VA		VARIABLES
PR		PROGRAMS
ER		ERASE
FR		FREEZE
UN		UNFREEZE

4.4.2 Direct Mode Operations

Certain of the FORTRAN operations can be used in direct mode as well as in stored programs. This feature effectively provides a desk calculator with the arithmetic power of the FORTRAN language. The FORTRAN operations that can be used in direct mode are:

CALL	execute a program
DIMENSION	define arrays
Name=Value	variable or array assignment
TYPE	output values
ACCEPT	input numbers
WRITE	execute assembly language output routine
READ	execute assembly language input routine
FUNCTION	execute assembly language function routine

Note that character string arguments are not appropriate with TYPE operations in direct mode. Examples of direct mode operations are:

```
←    X=2.4
←    TYPE   X*X-2*X+3
     3.96
←    CALL   P
←    TYPE   N
     453.2
←
```

All variables and arrays in this system are global; that is, the variables or arrays created in one program or in direct mode can be referenced by any other program. This fact means that variables used within a program can be set to initial values in direct mode prior to execution. Similarly, after program execution, the status of the program variables can be interrogated with TYPE or VARIABLES operations in direct mode. The global nature of variables can be very helpful during program debugging.

4.4.3 Program Execution

Before creating any stored programs, the CLEAR operation should be used to erase the previous user's programs from memory. After a program has been entered into memory, it can be executed with a CALL statement. When the CALL occurs in direct mode, the system scans the programs in memory for certain logical errors. The specific errors it looks for are:

1. A stored CALL statement which refers to an undefined program.

2. A stored GO TO, IF or DO statement which refers to an undefined statement.

If any errors of this sort are found, the system responds with an error message and will not execute the program specified. Details of the error message are discussed in Section 4.5. Note that the system scans all programs in memory for these errors, not just the one called for execution.

When a program is executing, the user can interrupt at any time to regain control at the keyboard. The procedure for interrupting a program execution is to depress Switch 15 on the Processor Display Panel. The system tests for this switch at certain points in the execution cycle. When the system senses that Switch 15 is set, execution terminates, an error message is typed, and control returns to direct mode. After regaining control, Switch 15 should be released.

4.4.4 Paper Tape Operations

Some teletypewriter terminals have a paper tape reader and punch. With this type of terminal, programs can be saved on paper tape, and later reloaded from the tape. The procedure for making a tape is:

1. Identify the program with a SUBROUTINE statement.

2. Type LIST and before depressing the RETURN key, turn on the tape punch.

3. Depress RETURN. The program will then be listed and punched.

4. When the operation is complete, tear off the tape, and turn off the punch.

Tape input is possible because whenever the system seeks input, it attempts to start the reader. Therefore, whenever a tape is put in the teletypewriter reader, it will be read as soon as the system seeks input. The program loading procedures are:

1. Make sure no program of the same name is currently defined.

2. When the system is in direct mode, and after the ← has been typed, put the program tape in the reader.

3. Momentarily put the reader switch into the start position, which starts the tape moving.

4. After the program has been read, remove the tape from the reader.

Data inputs can also be entered from tape using a similar procedure.

4.5 ERROR MESSAGES

Error messages can occur at the following times during system use: after direct or edit mode entries, at CALL time, or during program execution.

Direct mode errors are indicated by a question mark (?). The statement causing the error is not processed, and the system remains in direct mode. For example:

```
←  TYPE   LOG(-1)
?
←
```

In this case the error resulted because the LOG function is undefined for negative arguments.

Errors within a stored program are indicated by a question mark (?) followed by the name of the program. On the next line, the specific offending statement is listed. The system then returns to direct mode with the indicated program open. In this state, the LIST operation can be used to examine the program, or OPEN can be used to examing a single statement within the program. For example:

```
←  CALL   P
?  P
                DIMENSION   (A(-3)
←  LIST
                SUBROUTINE   P
                DIMENSION   A(-3)

                END
←
```

In this case the error occurred because the array definition involved a negative value. The LIST operation caused all of P to be printed.

Inserting a statement into a stored program, assigning a value to a variable, or defining an array, requires a sufficient amount of core memory. Whenever these operations are called for, if enough memory is not available, the system types an exclamation point (!). If the operation was within a stored program, the program name and statement are also listed. For example:

```
←  CALL   Q
!  Q
             N=3
←
```

In this case, since the assignment statement was the culprit, there was not enough memory available to store away the name and value of the variable N.

Some special cases are:

1. When CALL is used in direct mode to start execution, all programs in memory are scanned for improper program name and statement number references. If any reference is made to an undefined name, the system types an error message and lists the offending statement. For example:

```
← CALL  P
? Q
            DO  3  I=1,5
←
```

   This message implies that statement 3 is undefined in program Q.

   If the FREEZE mode is in effect, the system also checks for defined variables.

2. The DIMENSION operation checks that sufficient memory space is available for the array. For example:

```
      DIMENSION   A(100)
!
```

   This message says that not enough memory is available. When this happens, however, a variable named A may get defined. In this case, the variable should be erased before another DIMENSION operation is attempted for the array A.

3. When program execution is interrupted by the user depressing Switch 15, the system types an error message indicating the statement at which the break occurred. In this case, there is no actual error as the message might suggest.

Some sources of error messages are summarized below:

1. Illegal characters used, such as @, $, etc.

2. Input statement too long. Limit is 54 characters.

3. Undefined variable or array referenced.

4. Expressions improper or too complex due to nested parentheses, etc. Limit is 14 nested explicit or implicit parentheses.

5. DO loops nested too deeply. Limit is 4.

6. Subroutines nested too deeply. Limit is 5.

7. Improper number of arguments used, such as DO 5 I=1,2,3.

8. LOG used with a negative argument.

9. SIN or COS used with argument greater than 1000.

10. TYPE 'CHARACTERS' used in direct mode.

11. Edit commands such as OPEN, LIST, or DELETE used when no program is open.

12. OPEN N used and the open program contains no statement with label N.

13. CALL P used with P not a proper or defined name.

14. The statements DO, IF, GO use improper statement number arguments.

15. A computed GO TO executed with the index value out of range.

16. READ, WRITE operations attempted with no driver routines available.

4.6 SYSTEM CAPACITY

This system operates in 8K bytes or more of core memory. The FORTRAN processor itself occupies approximately 6.5K bytes of memory. In an 8K memory, this leaves a 1.5K working space for user's stored programs and data. Any available memory above 8K can be used to expand the working space.

Working space in memory is used as follows:

1. Stored statements require 20 bytes per average statement.

2. Defined variables require 6 bytes each.

3. Defined arrays require 6+4N bytes where N is the number of elements in the array.

Each 1000 bytes of working space an hold 50 average statements, over 150 variables, or a 15 X 15 two dimensional array.

APPENDIX 1
INTERACTIVE FORTRAN SUMMARY

←	direct mode input
*	edit mode input
=	data input
?	error
!	memory full

Numbers - real only, E format
- precision 6-7 digits
- range ± 76

Variables - 2 char. names, letter first
- global, real only

Arrays - 2 char. names, letter first
- 1 or 2 dimensions
- global, real only
- subscripts 1, 2, ... , N

Expressions - use (,), +, -, *, /, **

Functions - SIN, COS, EXP, LOG, ATN

Statements - 2 digit statement numbers
- terminate with RETURN char.
- 54 char. max, no continuations

Programs - 2 char. names, letter first
- nesting to level of 5
- implicit RETURN at end

Operations - CALL P
- RETURN
- GO TO N
- GO TO (N1, N2, . . .), X
- IF (X) N1, N2, N3
- DO N V=L1, L2
- CONTINUE
- DIMENSION A(L), . . .
- V=X
- TYPE A, B, . . .
- ACCEPT A, B, . . .
- WRITE X, A, B, . . .
- READ X, A, B, . . .
- FUNCTION (X, A, B, . . .), C, D, . . .

Erase previous character with ←
Erase line with RUB OUT
Interrupt execution with Switch 15

Editing Commands

SUBR	or	SUBROUTINE A
OP		OPEN N, X
LI		LIST
DE		DELETE
EN		END

System Commands

CL	or	CLEAR
PR		PROGRAMS
VA		VARIABLES
ER		ERASE A
FR		FREEZE
UN		UNFREEZE

Direct Mode Operations
CALL P
DIMENSION A(L), . . .
V=X
TYPE A, B, . . .
ACCEPT A, B, . . .
WRITE X, A, B, . . .
READ X, A, B, . . .
FUNCTION (X, A, B, . . .), C, D, . . .

Ignored
C COMMENTS
REAL
INTEGER
COMMON
FORMAT

Memory Usage
20 bytes per statement
6 bytes per variable
6+4N bytes per array

Features
2 character names
Free format input/output
2 dimensional arrays
All values real and global
FREEZE mode for speed-up

APPENDIX 2
SAMPLE INTERACTIVE FORTRAN PROGRAM

```
C  PROGRAM TO SHOW ARRAYS AND DO
   SUBROUTINE T1
   DIMENSION  A(5,7), B1(10,7)
   L=7
   DO 20 I=1,5
   DO 20 J=1,L
   A(I,J)=10*I+J
   GO TO 9
20 CONTINUE
   DO  35  I=L-6, 2*L-4
   DO  36  J=1,7
   TYPE  B(I,J)
36 CONTINUE
35 CONTINUE
C  EXAMPLE OF EXPRESSION AS SUBSCRIPT
   A(2,3)=5
   B(1,1)=999
   TYPE  'ELEMENT VALUE IS', B(A(2,3)-4, A(2,3)*A(2,3)/6-3)
   RETURN
C  EXAMPLE OF JUMP OUT OF DO
 9 B(2*I-1,J)=A(I,J)
   B(2*I,J)=10*(A(I,J)*A(I,J))/A(I,J)
   GO TO 20
   END
```

APPENDIX 3
FORTRAN REFERENCES

Refer to the following publications for a description of FORTRAN in general:

1. Farina, Mario V., FORTRAN IV Self Taught, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1966.

2. Golden, James T., FORTRAN IV Programming, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1965.

3. Jamison, Robert V., FORTRAN Programming, McGraw-Hill Book Company, New York, 1966.

4. McCracken, Daniel D., A Guide to FORTRAN Programming, John Wiley & Sons, New York, 1966.

OPERATING INSTRUCTIONS FOR INTERACTIVE FORTRAN

1. GENERAL DESCRIPTION

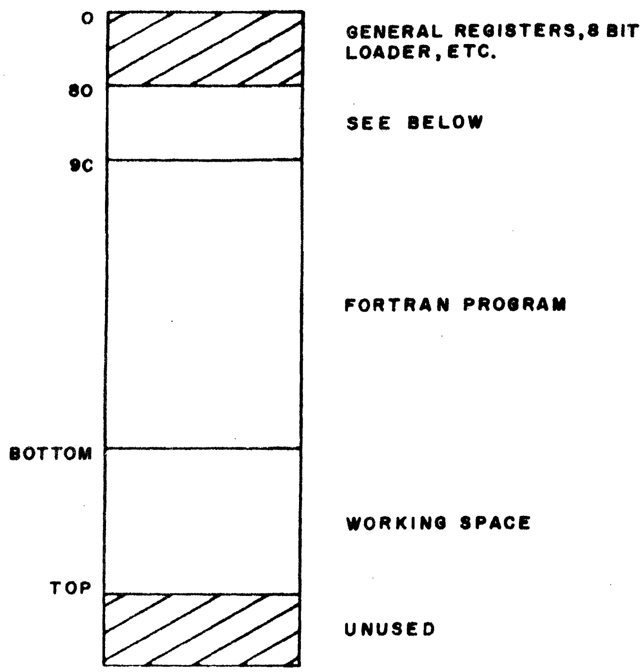
The FORTRAN system is discussed in detail in the User's Manual For Interactive FORTRAN, Publication Number 29-014. There are several versions of FORTRAN, each tailored to specific machine configurations. One of the versions involves a READ-WRITE-FUNCTION Expansion for linking assembly language routines to FORTRAN. Details of this Expansion are discussed in Operating Procedures For FORTRAN With The RWF Expansion, Publication Number 03-011A16. This document describes the procedure for loading, starting, and using the various FORTRAN tapes.

2. VARIATIONS

There are four variations of FORTRAN as follows:

Number	Name	
03-005R02M10	FORTRAN	- requires high speed option - provides no RWF Expansion
03-006R02M10	FORTRAN W/TRAP	- no high speed option required - no RWF Expansion
03-007M10	FORTRAN/ 30-2	- requires high speed option - requires floating-point option - provides RWF Expansion
03-011M10	FORTRAN W/RWF	- requires high speed option - provides RWF Expansion

All variations require at least 8K bytes of core memory. All variations assume a teletypewriter is interfaced to the Processor as Device Number 2. The high speed option, mentioned above, is the High Speed Arithmetic Instruction Repertoire. The floating-point option is the High Speed Arithmetic Instruction Repertoire, which is available on the 30-02 only. Note that the amount of working space available for user programs and data varies with each of the above. Refer to Figure 1 for details. The FORTRAN W/TRAP includes the multiply and divide TRAP subroutine for machines without the high speed option. Since the TRAP arithmetic operations require more execution time than hardware instructions, this version of FORTRAN runs slower than the others. Also, due to the size of the TRAP routine, the working space is considerably reduced, and this version requires more than 8K of memory to be generally useful.



PROGRAM NUMBER	BOTTOM VALUE	TOP VALUE
03-005R02	X'1B70'	X'1FFE'
03-006R02	X'1D80'	X'1FFE'
03-007	X'19A0'	X'1FFE'
03-011	X'1C00'	X'1FFE'

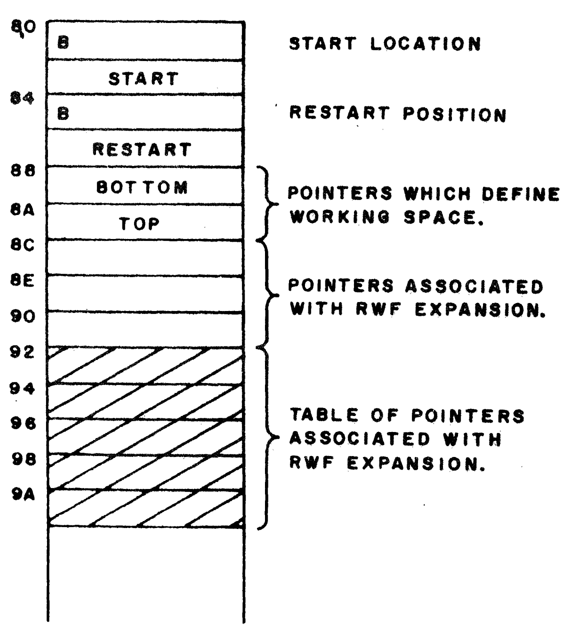


Figure 1. Memory Allocation

3. TAPE FORMAT

The FORTRAN tapes are bootstrap tapes. Refer to Bootstrap Programs and Procedures, Publication Number 06-030A12, for an explanation of the tape organization. The bootstrap tape is loaded by using the eight-bit loader at X'50'. Note that memory locations from X'1D00' to X'1FFF' are used during the bootstrap loading process.

New features in the above versions of FORTRAN are as follows:

1. A problem concerning referencing of undefined subroutines is corrected.

2. The left arrow character (←) is recognized during keyboard inputs for purposes of deleting the last character in the line.

3. The starting location is X'80'. A restart location is provided at X'84'. Refer to Section 5.

4. The Illegal instruction interrupt pointer is set when FORTRAN is started. See Section 5.

4. LOADING PROCEDURES

FORTRAN tapes are loaded using the eight-bit loader at X'50'. The 50 Sequence, which includes the eight-bit loader, is described in the first section of the Programming Manual, Publication Number 29-013. Assuming that the 50 Sequence has been entered into memory, FORTRAN is loaded as follows:

1. Put the tape into the tape reader. Be sure the first data character is over the read fingers or photo diodes.

2. Set the Data/Address Switches to X'50', set the MODE CONTROL to ADRS, and depress EXECUTE.

3. Depress INITIALIZE.

4. Set the MODE CONTROL to RUN, and depress EXECUTE.

5. If a teletypewriter is in use as the input device, manually start the tape by moving the reader switch to Start or Run.

6. The tape should be read until the end. If errors are detected on input, the tape will stop, and the Processor will halt with the Wait light lit. In this case, reposition the tape to the previous record gap and depress EXECUTE to reread the record.

7. When the tape has been entirely loaded, control is automatically transferred to FORTRAN. FORTRAN indicates it is ready for use by printing ← on the teletypewriter.

5. STARTING PROCEDURES

Set the display switches to X'0080'. Select ADRS mode and depress EXECUTE. Select RUN mode and depress EXECUTE. The system will type an arrow (←) to indicate it is ready for commands from the keyboard.

When execution is started at X'0080', the system is initialized as follows:

1. The Illegal instruction PSW in locations X'34' - X'37' is set.

2. The limits of the working space are established, and the working space is cleared which erases any stored programs and variables.

3. The system is set to unfrozen state in Direct mode.

To restart the FORTRAN program without clearing the working space, start execution at X'0084'. Refer to Figure 2.

6. MEMORY ALLOCATION

Figure 1 shows a memory map of the system. The limits of the working space, as shown, are established whenever the program is started at X'0080', or the CLEAR operation is used. After the FORTRAN tape has been loaded, the limits of memory available for working space can be changed as follows:

1. The halfword at X'0088' contains the lower limit. This limit is the address of the first halfword within the working space. This limit can be changed with memory write (MEMW) operations on the display panel to any desired value.

2. The halfword at X'008A' contains the upper limit. This limit, which is the address of the last halfword within the working space, is defined as X'1FFE' when the tape is loaded. This limit can be changed with memory write (MEMW) operations on the display panel to any desired value.

3. After changing the upper or lower limits, start execution at X'0080' as described previously in Section 5.

Redefinition of the working space limits may be desirable if more than 8K of memory is available or if it is necessary to keep other programs in memory with FORTRAN.

7. PAPER TAPE PROCEDURES

Whenever FORTRAN seeks input from the teletypewriter, a single character is typed to reflect the type of input needed. The characters are:

←	for direct mode input
*	for edit mode input
=	for data input

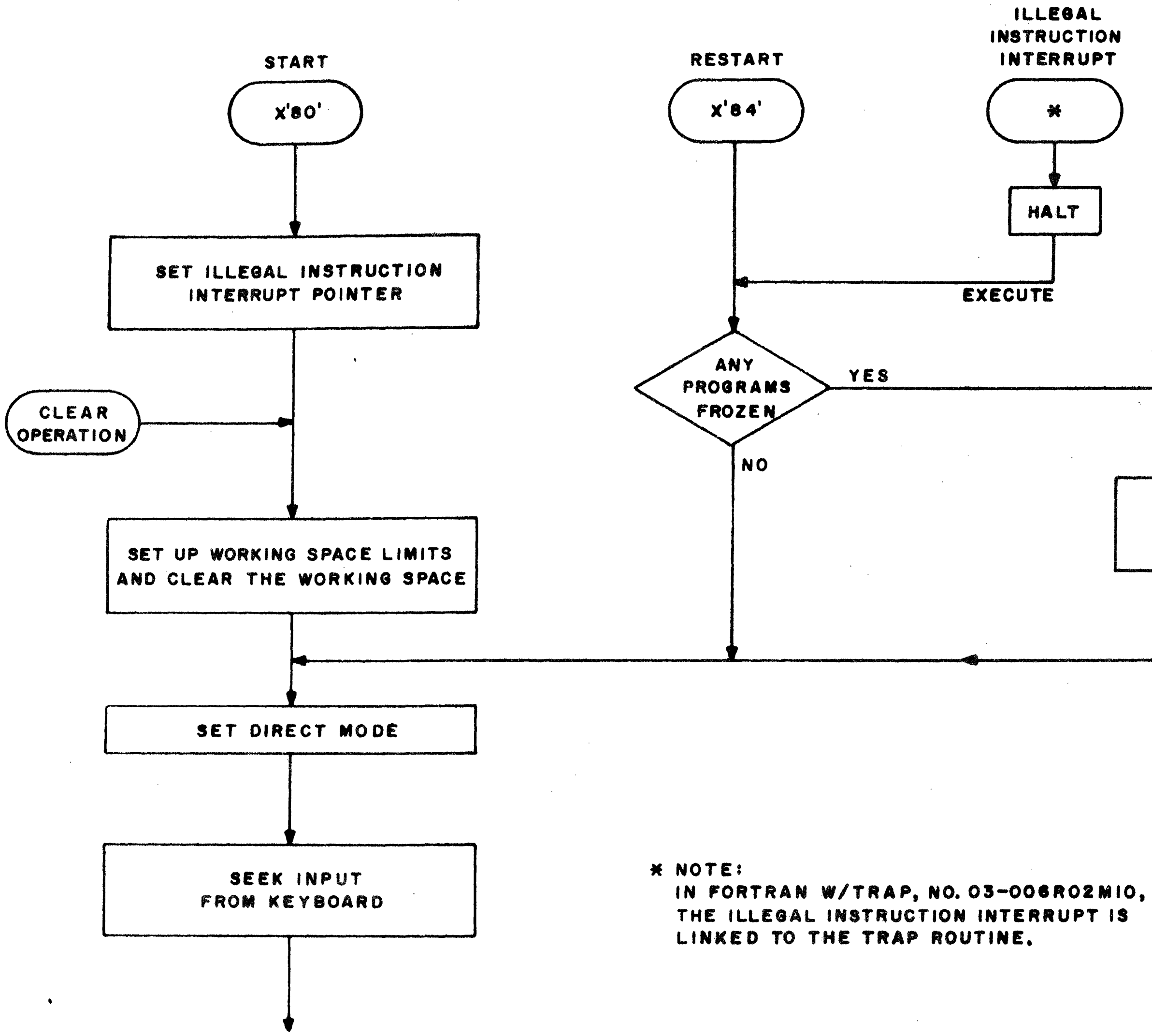


Figure 2. Starting Procedures

Following the character, the program issues an XON character, which starts the tape reader if it contains a tape. This means that whenever a tape is in the Reader, and the program seeks input, the tape will advance and be read.

All inputs to the system are terminated by a carriage return (RETURN key). Whenever the system detects a RETURN character, it issues an XOFF character, which stops the tape reader. Note that a tape may move 1 or 2 characters after XOFF is issued. Similarly, a tape may move 1 or 2 characters after XON is issued before proper synchronization is obtained. For this reason, a number of blank or space characters must separate each line on the paper tape.

When no tape is in the Reader, and inputs are given through the keyboard, the extraneous XON and XOFF characters will make a slight click, but have no other effect.

To punch a tape, the tape punch on the teletypewriter must be turned on manually. While the tape punch is turned on, all characters typed are also punched. By doing a LIST operation with the punch turned on, therefore, the program listed will also be punched. Recall that statements are indented several spaces when listed by the system. These leading spaces serve to separate properly the lines on paper tape.

The procedure for reading a program tape is as follows:

1. Make sure the system contains no programs with the same name as those to be loaded.

2. After the system types ← to request another direct mode input, put the tape in the reader. The leading spaces prior to the first statement should be placed over the read fingers.

3. Momentarily push the reader switch to the Start position. The tape will be read, stopping and starting again after each line.

4. When all of the program has been read, remove the tape from the reader.

To read a tape with a Model ASR-35 Teletypewriter, it is necessary to use the KT Teletypewriter mode.

OPERATING PROCEDURES FOR FORTRAN WITH RWF EXPANSION

Publication Number 03-011A16

1. GENERAL DESCRIPTION

FORTRAN with RWF Expansion, Program Number 03-011, is equivalent to FORTRAN, 03-005R02, supplemented with a READ-WRITE-FUNCTION interface. This version of FORTRAN runs on any GE-PAC 30 Processor with 8K bytes or more of core memory and the high-speed arithmetic instruction repertoire. FORTRAN, with the READ-WRITE-FUNCTION Expansion, allows users to expand the FORTRAN program with their own routines for input, output, and arithmetic functions. The program interface provided is sufficiently general that, with the proper precautions, any assembly language routine can be successfully linked with FORTRAN.

It should be noted that FORTRAN with RWF Expansion takes more core memory than other versions of FORTRAN. While the program runs in 8K, therefore, very little space remains for user's programs. More memory is required, therefore, for this program to be used effectively.

2. TAPE DESCRIPTION

FORTRAN with RWF Expansion is available in two tape formats.

1. Tape 03-011M10 is a bootstrap tape, and is loaded using the eight-bit loader at X'50'. Refer to Publication Number 06-030A12 for details.
2. Relocatable tapes for FORTRAN with RWF are available. Several tapes are required, and they must be loaded and linked together by the General Loader (06-025). Because of the loaders required, it is essential to have more than 8K of memory to use these tapes.

3. FEATURES

This program involves several pointers at the beginning of the program as follows:

ORG	+ 0	B START	Start here and clear memory
	+ 4	B RESTART	Start here and preserve memory
	+ 8	A(BOTTOM)	Pointer to bottom of working space
	+ A	A(TOP)	Pointer to top of working space
	+ C	A(READ)	Pointer to READ routine
	+ E	A(WRITE)	Pointer to WRITE routine
	+ 10	A(FUNC)	Pointer to FUNCTION routine

The working space pointers are initially set to:

A(BOTTOM) = X'1COO'
A(TOP) = X'1FFE'

The READ, WRITE, FUNC pointers initially point to the error routine in the FORTRAN program. User-written routines are linked to FORTRAN by adjusting these pointers accordingly.

Another feature in this version of FORTRAN is the use of the left arrow character (←) during user inputs for single character deletes. That is, the user typing

X-←= 2

followed by a carriage return causes the system to delete the minus character (-) and perform X=2.

The FORTRAN statements relevant to the READ, WRITE, and FUNC operations are described below. Note that this description, while not always in agreement with the FORTRAN User's Manual, Publication Number 03-005A12, tells it like it really is with this particular version of FORTRAN.

4. READ STATEMENT

READ X, A, B, C, ...

The READ statement provides all the facilities of the ACCEPT statement but, in addition, it permits use of devices other than the Teletypewriter. Data is read into FORTRAN by user-written device driver routines.

The data read is stored as the values of A, B, C, etc. X is used as a switch or a device number that the user's program decodes. This allows READ to operate with several input devices that the user interfaces.

X, A, B, C, ... are all elements. This means that they can be symbol names or references to array elements; they cannot be literals or expressions. This should be noted particularly with respect to X.

When the READ statement is executed, the FORTRAN gives control to that program whose address is in ORG + X'C', where ORG is the first location of FORTRAN. The user supplies this program and sets this address.

The program must be written to read a record of information into the FORTRAN buffer. This record will subsequently be processed exactly as a teletypewriter line is processed following an ACCEPT statement.

When the program is called, all the parameters of interest are supplied in the machine registers. This allows the user to write the program without knowledge of the FORTRAN system.

The execution of READ causes numbers that have been read into the FORTRAN buffer by the user's program to be stored as values of the symbols A, B, C, ... etc.

X, which must be given as an "element" in the READ statement, is fetched from the symbol table, integerized, and placed in a convenient register. The user's input program may use this integer to determine which of several possible input routines should be used. This allows the READ statement to seek input from one of several devices that the user has interfaced.

The user supplies characters to the buffer in eight-bit ASCII code (high-order bit set), and terminates the record by placing a carriage return character at the end of the data. The rules concerning what characters are used to represent numbers and their separators, are exactly the rules that apply to the ACCEPT statement. The user must not enter more than 70 characters.

When control is given to the user-written program, the registers contain their values and addresses that are indicated in Table 1.

TABLE 1

REGISTER ALLOCATION
ON TRANSFER TO USER PROGRAM

Register Number	Mnemonic Name	Contents
0	SIX	6
1	FOUR	4
2	TWO	2
3	ONE	1
4	LOC	Address of first entry in list containing high-order part of arguments.
5	OP	Address of first entry in list containing low-order part of arguments.
6	AHI	Pointer to character position in buffer where first character is stored. Subsequent characters are stored in succeeding locations.
7	ALO	Undefined.
8	BHI	Rounded integerized value of X, expressed as an integer.
9	BLO	Undefined.
A	SIZE	Address of a table containing addresses of useful subroutines.
B	CUR	(a) Count of characters for output in WRITE program (N characters mean count is set at N+2). (b) Count of arguments for FUNC program (N arguments mean count is set at 2*N).
C	Z1	Rounded integerized value of X, expressed in floating-point; high-order half.
D	Z2	Rounded integerized value of X, expressed in floating-point; low-order half.
E	BACK	Return address for returning to FORTRAN system.
F	ERROR	Address of universal error routine in FORTRAN system.

5. WRITE STATEMENT

WRITE X, A, B, C, ...

The WRITE statement is for use by a user who wishes to exercise the facilities of the TYPE statement, but who wishes to have the output performed on a device whose driver program he wishes to write. This statement allows output of data from the FORTRAN system to devices that the user interfaces for himself.

The data written are the values of the variables A, B, C, ... etc. X is used as a switch or device number that the user's program decodes. This allows WRITE to operate with several output devices that the user interfaces.

X, A, B, C, ... are all "values". This means they can be symbol names, numeric literals, expressions, or character strings. Note that X should not be a character string.

When the WRITE program is executed, the FORTRAN first generates a buffer full of characters. It then gives control to that program whose address is in ORG + X'E', where ORG is the first location of FORTRAN. The user supplies this program and sets this address.

The function of the program must be to write out a record of information for the FORTRAN buffer to the user's device. The record that is written, is exactly the same as the record that is written to the teletypewriter by the TYPE statement, and printed as a line. The information in the buffer does not include any carriage return or line feed characters.

When the program is entered, all the parameters of interest are supplied in the machine registers. This allows the user to write programs without knowledge of the FORTRAN system.

X is given as a "value". The WRITE statement causes it to be integerized and placed in a convenient register. The user's output program may use this integer to determine which of several possible output routines should be used.

The characters in the buffer are in eight-bit ASCII code (high-order bit set). The format of the numbers and characters in the buffer is exactly the same as for the TYPE statement.

When control is given to the user-written program, the registers contain those values that are indicated in Table 1. The number of characters to be output (the exact number plus two) is contained in register CUR. The exact number will never exceed 72.

6. FUNC STATEMENT

FUNC (X, Y, ...), A, B, ...

The FUNC statement is for use by a user who wishes to write his own machine-language function program, and insert it into the FORTRAN system.

The arguments are a group of "elements" and a group of "values". The "elements" are passed to the user-written program as addresses, and the "values" are passed as hexadecimal numbers. The "elements", the first arguments in the statement, are indicated by being surrounded by parantheses There must be at least one "element". If there is exactly one "element", no parentheses are required. Examples of proper FUNC statements are:

```
FUNC    X, 3.5, SIN(2)
FUNC    (X, BF), 4*N
FUNC    (X, Y, Z), 1, EXP(P), 'HELP'
```

The user provides a program whose address must be put in location ORG + X'10', where ORG is the first location of FORTRAN. When FUNC is executed, control is passed to the user's program. All the information the user needs is passed in the registers as is shown in Table 1. The arguments are passed in two tables; the first holds the high-order parts, and the second the low-order parts. The starting locations of this table are held in registers LOC and OP. The number of arguments (the exact number multiplied by two) is held in register CUR.

The rounded and integerized value of X is held in register BHI. It may be used by the user to steer the function program to a number of other subprograms. X however, is also held in the argument tables as the first argument.

7. REGISTER ALLOCATION

The constants 1, 2, 4, 6 are supplied in four registers for the user's convenience, and they may be used freely. However, these registers must be properly restored by the user if he alters them.

The address in register ERROR is the address of a FORTRAN routine for handling errors. It the user's program is written to test for errors, their occurrence should cause control to go to that place. This register must also be saved and restored if it is used by the user program.

Arguments that were specified with the FUNC statement, occur as four byte items. The high-order two bytes occur in a table whose first address is held in LOC: the low-order two bytes occur in a table whose first address is held in OP. Successive arguments are in successive halfwords.

If an argument is a "value", the four bytes that represent it are its floating hexadecimal value, providing the argument is numeric.

The value may, however, be a character string. In this case, the first two bytes consist of X'F000' and the second two bytes are the address of the first character of the character string in core. The character string is preceded by a halfword containing the character count plus four, followed by the ASCII characters of the string. See Figure 1.

If an argument is an "element", the two high-order bytes that represent it, form the address of the location in core where the value of the argument is stored. This value is numeric and consists of four bytes. The address that is pointed to, is the one that contains the low-order half of the value. See Figure 2.

(In the event that an array name is used as an argument, the address obtained is the address of the location where the first element of the array is stored.)

If an argument is merely a name, the two low-order bytes that represent it, contain auxiliary information. The name may be the name of a variable or of an array. The low-order part contains the address of the first byte in the symbol table that represents the variable or array. (A knowledge of the symbol table structure shows that the address in the high and low part differ by ten for arrays and by four for variables. See Figure 3.

If, however, the argument is an element of an array, the two low-order bytes that represent it contain zero.

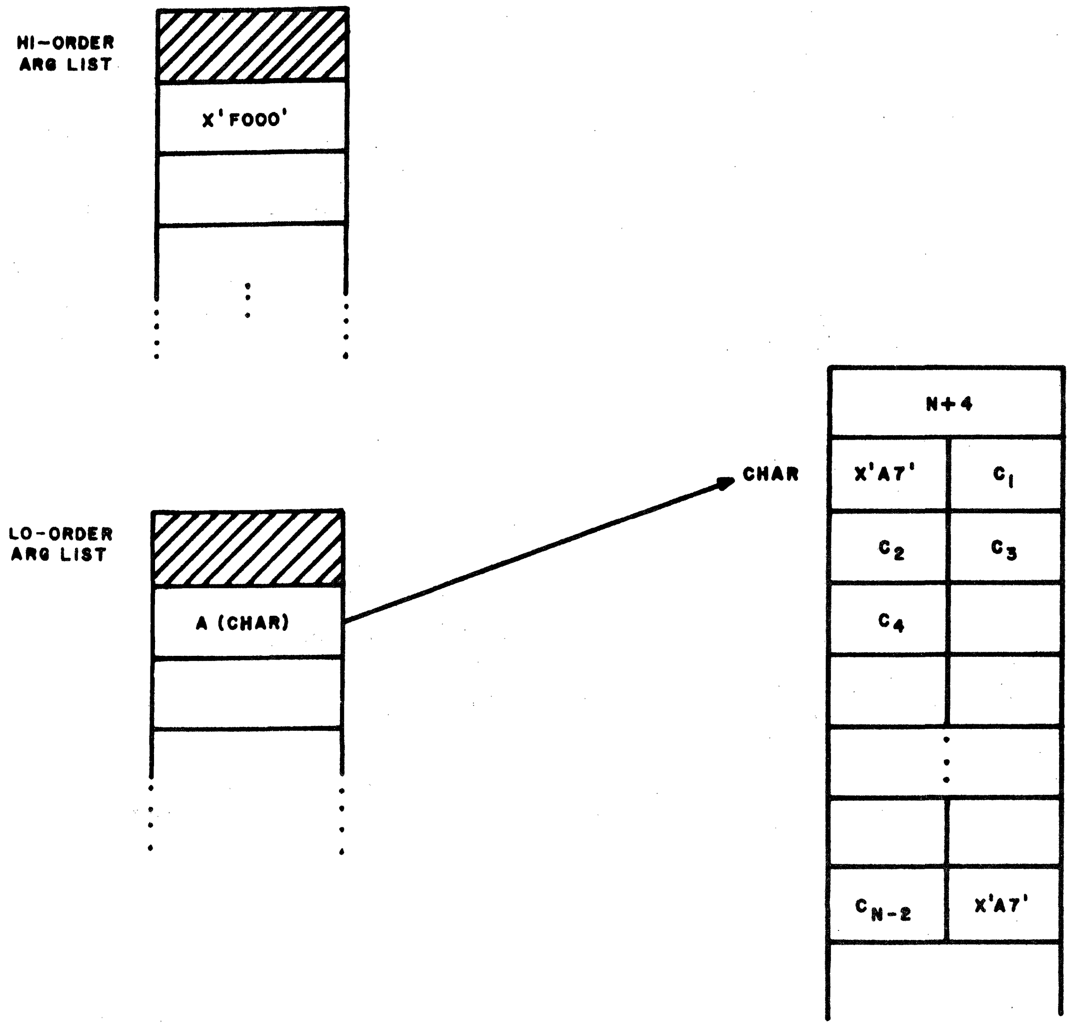
8. OPERATION

It is the responsibility of the user to generate each assembly language routine appropriate to his needs. Each routine must be loaded into memory in some locations that do not conflict with FORTRAN itself. If need be, the BOTTOM AND TOP pointers which define the working space, should be adjusted.

When FORTRAN transfers control to a user-provided routine, care should be taken to restore the appropriate registers prior to returning control to FORTRAN.

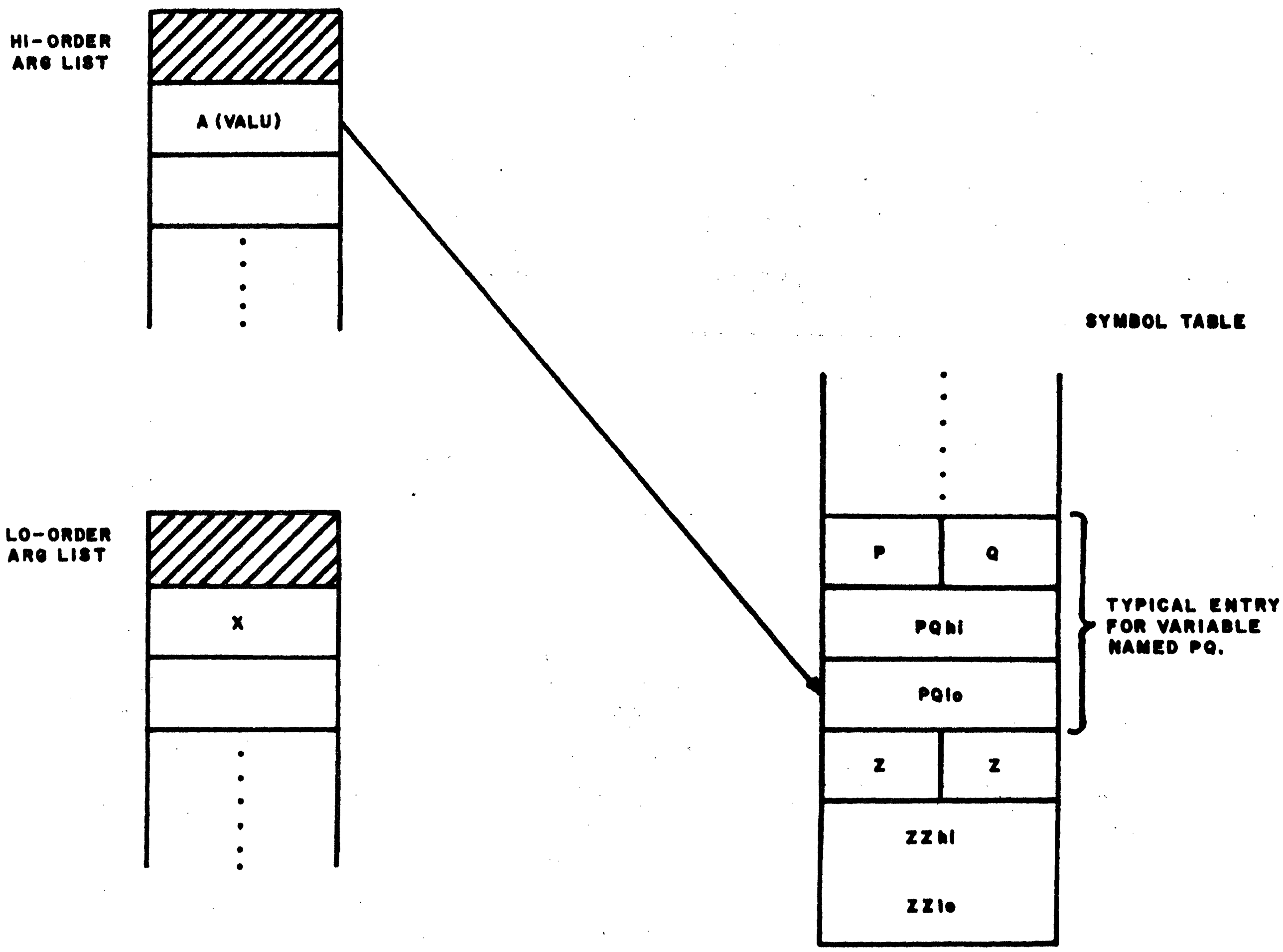
For a discussion of the Floating-Point data format used within FORTRAN, refer to Publication Number 07-020A12.

Good luck!



The halfword preceding CHAR contains the value N+4, where N = the number of characters in the character string including the apostrophe (') characters. Note the X'A7' is the character code for apostrophe (').

Figure 1. Character String Argument



If the argument in the statement is an element, the high-order argument list contains a pointer to the symbol table entry for that element. In the case shown above, the element is a variable named PQ, and the pointer points to the low-order portion of the value of PQ. In this case, X = A(VALU) - 4.

Figure 2. Variable Arguments

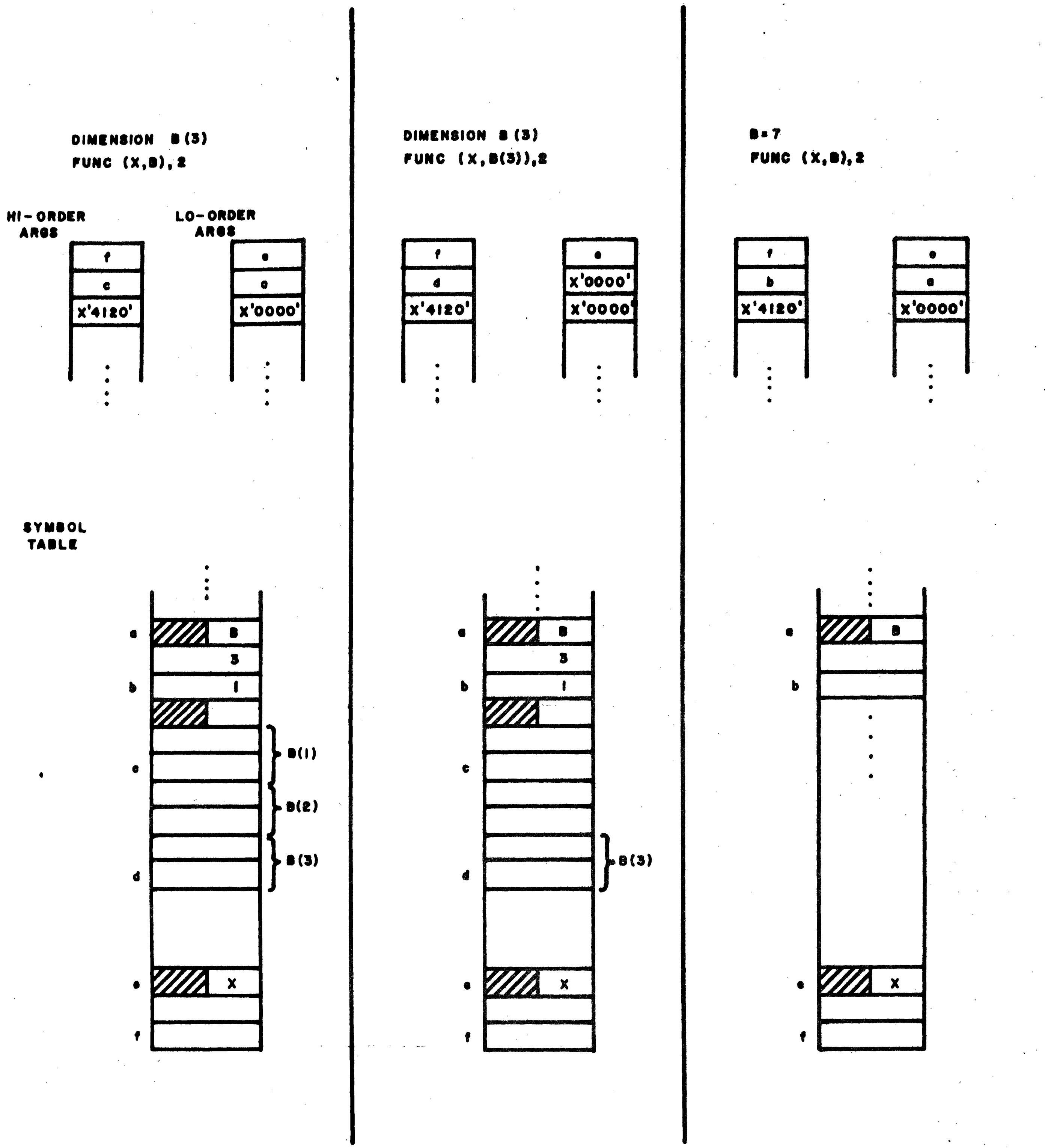


Figure 3. Sample FUNC Statements

TABLE OF CONTENTS

EDITOR (TIDE) PROGRAM MANUAL

Publication Number 29-082

1. INTRODUCTION

TIDE, Program Number 06-014, is an interactive text editor program. It is designed to create and modify character-oriented text material which is stored on paper tape, or input through the teletypewriter keyboard. The text may be an assembly language program, a FORTRAN program, or any text in the literal sense.

TIDE is directed by an operator through the keyboard of a teletypewriter terminal. Upon receiving a keyboard input directive, the editor will read text from a specified input device into a designated area of core memory. The editor allows the user to examine, change, delete, and/or modify the text while it remains in core memory. When the editor receives a keyboard output directive, the revised text can then be output to the specified output device.

2. PROGRAM STRUCTURE

2.1 Operating Modes

TIDE has two modes of operation: Command and Edit. The program indicates the current mode by printing, in column one on the teletypewriter, a left (←) for the Command Mode or an asterisk (*) for the Edit Mode. In the Command Mode, the program accepts keyboard commands which specify an editing procedure, or which specify a text input or output operation. From an edit command, TIDE enters the Edit Mode. Edit Mode allows the user to insert, append, or modify the text, after which control returns to the Command Mode.

2.2 Basic Unit

The basic unit of the stored text is a variable length line from 1 to 67 ASCII code characters long including the line terminating carriage return (CR). Each line of input is stored in a line buffer until the CR terminates the input. The <u>line buffer</u> contents are then moved to the <u>text buffer</u>. If the text buffer contains a symbolic source program, each source statement is one line of text. The statements, or lines, have unique decimal addresses which are sequenced in ascending order; the first statement in the buffer has address number one (1). This allows editing of any statement by line address rather than core location address.

2.3 Line Addressing

A specific line can be referenced by its decimal number address n. To examine line n, type the decimal number followed by a carriage return. The teleprinter will list:

```
n ZZZ...Z
```

where n is the line number, and Z the text contained in line n. This becomes the line currently available for examination or modification and is called the open line. All forms of line examinations are exclusive to the Command Mode (←) and will never cause transfer to the Edit Mode (*). However, any attempt to examine a line not contained in the buffer, or to reference a nonexistent line, will result in an error message (see Section 2.7).

The execution of some TIDE commands will change the position of a line in the text buffer, and consequently change the line number. This line number change does not affect the contents of the line. (See Section 2.6, Command Examples.)

To facilitate line addressing and determine the number of bytes used in the text buffer, the symbols in Table 1 have been implemented. A CR delimiter following the symbol is used as described for decimal number addressing. Editing convention, however, eliminates the CR after a Line Feed (LF) or CR itself. All of the symbols with the exception of the arrow (↑) cause the listed line to become the open line.

2.4 Command Formats

Commands are entered through the keyboard in one of the following general formats:

FORMAT	TERMINATING CHARACTER	DESCRIPTION
X	CR	Editor performs Command X.
X n	CR	Editor performs Command X on n lines.
X a,b	CR	Editor performs Command X on lines a through b inclusive. Both a and b must be positive with b not less than a.

where n, a, and b are decimal numbers hereafter called arguments, and Command X directs the editor to perform the specific operations described later in the manual in Table 3. All command formats are terminated by a CR. If there is an error in the command format (Section 2.7), no action is taken and program control returns to the Command Mode (←). One or more spaces should separate the arguments from the command.

TABLE 1. LINE ADDRESSING

SYMBOL	TERMINATING CHARACTER	FUNCTION
n	CR	Opens and lists line number n (a decimal number).
Carriage Return	None	Opens and lists the line preceding the current open line.
Line Feed	None	Opens and lists the line following the current open line.
* (asterisk key)	CR	Lists the current open line.
. (period key)	CR	Lists the last line in the text buffer.
↑ (upper arrow key)	CR	Lists the byte count of the contents of the text buffer. The count is shown in decimal.

Line addressing (Section 2.3) and command arguments can involve arithmetic using addition and subtraction. Some sample formats are shown in Table 2. When using line arithmetic with two argument commands (general format X a, b), the rules for a and b still apply.

2.5 Commands

The three main functions of the editor are input, modification, and output. The input commands are used to enter text into the buffer. The modify commands direct various manipulations of the text stored in the text buffer. The output commands produce a hard copy of the text on the command-specified output device. Table 3 contains the definitions for the command repertoire. The Tabulate (T) command causes print and list operations to use a format similar to the Assembler. The command-specified output devices will be explained in Section 3.2. Following all of the output commands except List (L), the computer stops to allow time to prepare output device; to continue depress the EXECUTE Switch. Commands which are terminated by the break key (BK) may also be halted by setting the Display Panel Data/Address Switch 15.

2.6 Command Examples

See Table 3 for command definitions.

a. Append

```
←  A(CR)²
*  APPEND LINES TO (CR)
*  THE TEXT BUFFER (CR)
*  (BK)
←
```

b. Print - Print lines one through five without tabs.

```
← P  1,5(CR)
LINE ONE STILL LINE NUMBER ONE
LINE TWO DELETED IN EXAMPLE C
LINE 3 NUMBER WILL CHANGE
LINE 4 NUMBER WILL CHANGE
LINE 5 NUMBER WILL CHANGE
←
```

TABLE 2. LINE ARITHMETIC

SAMPLE FORMAT	TERMINATING CHARACTER	DESCRIPTION
2+7	CR	Lists line number 9.
.-8	CR	Lists the eighth line before the last line in the text buffer.
.-*	CR	Lists the line indicated by subtracting the decimal number address of the current open line from the decimal number address of the last line in the text buffer.
X *+3	CR	Editor performs command X on the third line following the current open line.
X *+3,.-6	CR	Editor performs command X on lines (*+3) through (.-6), inclusive.

2 (XX) Refers to a non-printing keyboard input.

TABLE 3
COMMAND REPERTOIRE

FUNCTION	KEYBOARD INPUT COMMAND	RESPONSE	DEFINITION	DESCRIPTION
Input	A	*	Append	The editor enters the Edit Mode (*) and accepts input from the keyboard. The typed text line is appended following the last line of text (if any) in the buffer. Each line of input is terminated with a CR. After an * response, the Append operation is terminated by depressing BK. After termination, the last line input, now the open line, and its decimal number address are printed. TIDE returns to the Command Mode (←).
Input	A n	none	Append n lines	TIDE enters the Edit Mode after which n lines are read from the Source Input Device, specified in location X'7C'[1]. This read operation may be halted when an end of line is reached by depressing BK. After either manual or normal termination, this command continues as the A command.
Input	I	*	Insert	The editor enters the Edit Mode (*) and accepts text lines from the keyboard to be inserted preceding the open line. Insertions are made in the order in which lines are input. Each line is terminated by depressing CR and the operation is terminated by depressing BK. Upon termination, the open line with its corrected decimal address will be printed and control goes to the Command Mode (←).
Input	I n	none	Insert n lines	Control transfers to the Edit Mode after which n lines of text are read from the Source Input Device specified in location X'7C'[1], and are inserted into the text buffer as described for the I command. The insertion may be terminated when the end of a line is reached by depressing BK. Upon either manual or normal termination, the open line with its corrected line number will be printed and control goes to the Command Mode (←).

TABLE 3 (Continued)

FUNCTION	KEYBOARD INPUT COMMAND	RESPONSE	DEFINITION	DESCRIPTION
Modify	D	none	Delete	The editor deletes the current open line. The line following the deleted line is now the open line and will be printed along with its corrected line number. Control returns to the Command Mode (←). If the last line in the buffer is the open line and it is deleted, the new last line is now the open line and it is listed.
Modify	D a, b	none	Delete lines a through b	Lines a through b inclusive are deleted. Line b+1 becomes the open line and is printed along with its corrected decimal address. TIDE remains in the Command Mode (←). If line b + 1 is non-existent, the last line is opened and listed.
Modify	C	*	Change	The open line is deleted. Control then goes to the Edit Mode (*) through which insertions can be made as in the I Command. After insertions are made, the command is terminated by depressing BK. The line following the deleted line becomes the open line and is printed along with its corrected decimal number address. Control transfers to the Command Mode (←). If the last line in the buffer is the open line and it is changed without insertion, the new last line is now the open line and it is listed.
Modify	C a, b	*	Change lines a through b	This command deletes lines a through b inclusive then continues as the C command.
Output	P	Processor Halt	Print	This command must be preceded by the T or N command.[3] The P command prints the contents of the text buffer in T or N unnumbered line format on the device specified in location X'7E'[1]. Printing may be terminated when the end of a line is reached by depressing BK. Upon manual termination, the open line and its decimal address are printed. The editor remains in the Command Mode (←).

TABLE 3 (Continued)

FUNCTION	KEYBOARD INPUT COMMAND	RESPONSE	DEFINITION	DESCRIPTION
Output	P a, b	Processor Halt	Prints lines a through b	Lines a through b inclusive are printed as in the P Command.
Output	O	Processor Halt	Output punched tape	The entire text buffer is punched in the standard source tape format [2] on the device specified in location X'7A'[1]. Output may be halted at the end of a line by depressing BK. Upon manual termination, TIDE prints the open line and its decimal address. TIDE remains in the Command Mode (←).
Output	O a, b	Processor Halt	Output lines a through b on punched tape	This command punches lines a through b inclusive and continues as the O Command.
Output	L	none	List	The contents of the text buffer are printed in T or N numbered line format on the teleprinter. Printing may be terminated by depressing BK after which the open line and its decimal address are printed. The editor remains in the Command Mode (←).
Output	L a, b	none	List lines a through b	Lines a through b inclusive are printed and terminated as in the L Command.
Setup	T	←	Tabulate output	Sets the tabulate flag for format control. Output will be similar to the Assembler format.
Setup	N	←	Untabulated output	Resets the tabulate flag for no format control. Output spacing will be as in the text buffer. This flag state is N when TIDE is started at the origin.
Setup	K	TIDE ←	Kill text buffer	This command starts TIDE at the ORG as described in Section 3.3, Starting Location.

TABLE 3 (Continued)

FUNCTION	KEYBOARD INPUT COMMAND	RESPONSE	DEFINITION	DESCRIPTION
Other	R	Processor Halt	Reproduce a punched tape	This command duplicates a punched paper tape. The input device is specified in location X'7C'[1] and the output device in location X'7A'[1]. No information from the tape will enter the text buffer. Duplication may be terminated when the end of a line is reached by depressing BK. After manual halt, the open line and its line number will be printed. After the entire tape is reproduced, the last line reproduced is printed. TIDE remains in the Command Mode (←). ASCII codes X'00' through X'1F' are not duplicated.
Other	R n	Processor Halt	Duplicate n lines of punched tape.	The number of lines specified in the argument are duplicated on punched paper tape. The I/O devices and break key (BK) termination are as stated for the R command. No information from the tape enters the text buffer. If n lines are reproduced, the last line punched is printed. TIDE remains in the Command Mode (←).
Other	S	none	Skip	The device specified in location X'7C'[1] will advance until halted, when the end of a line is reached, by depressing BK. After manual halt, the open line and its number are printed. The editor remains in the Command Mode (←).
Other	S n	none	Skip n lines	The number of lines specified in the argument are skipped as in the S Command. If n lines are skipped, the last line skipped is printed. If the operation is terminated with a BK, the open line and its number are listed.

1. See Section 3.2,I/O Device Selection
2. See Section 3.4, Tape Format
3. See Table 3 T, N

where:

n a decimal number

a first argument, a decimal number

b second argument, not less than a, a decimal number.

c. Change - From example b the open line is number two. Insert two lines. Open line number four is printed.

```
← C (CR)
*  INSERT A LINE (CR)
*  INSERT ANOTHER LINE (CR)
*  (BK)
4  LINE 3 NUMBER WILL CHANGE
←
```

d. List - List lines one through five for example c after C Command execution. Note the change in line numbers.

```
← L 1,5(CR)
  1 LINE ONE STILL LINE NUMBER ONE
  2 INSERT A LINE
  3 INSERT ANOTHER LINE
  4 LINE 3 NUMBER WILL CHANGE
  5 LINE 4 NUMBER WILL CHANGE
←
```

2.7 Errors

The error message for an improper TIDE command entry or for a line of text (from any input device) which exceeds the character limit, is the question mark (?). If a command entry error is made, no action is taken upon the information in the text buffer, TIDE responds with an error message (?), and remains in the Command Mode (←). If the text line exceeds 67 characters, the error message (?) will be printed and program control is transferred to the Command Mode (←). None of the characters in the line will be entered into the text buffer. If the error occurs in either the command or text entry, and is discovered before depressing the CR, the mistake may be corrected. Corrections are made by typing a left arrow (←) which deletes the last character in the line, or by typing a Rubout (RO) which deletes the entire line. The editor responds to the RO with a # symbol and control remains in the current mode of operation.

Another TIDE error flag is the exclamation point (!) which means the text buffer has overflowed. When this happens, the line which caused overflow is not entered into the text buffer and the text buffer is unchanged. TIDE returns to the Command Mode (←). To enter more information, it is necessary to delete one or more lines from the buffer. To reread the line which caused overflow, it is necessary to backspace the input tape one line, and adjust the buffer contents to make more room.

Section 3.4, Tape Formats, explains errors caused by incorrect input tape formats.

3. OPERATING PROCEDURES

3.1 Loading

TIDE, Program Number 06-014M08, is a relocatable program which requires X'0E56' bytes of memory including a text buffer of 1000 bytes. To load TIDE, use the Relocatable Loader, Program Number 06-024, or the General Loader, Program Number 06-025. Refer to Loader Descriptions, Publication Number 06-025A12 for use of these loaders.

3.2 I/O Device Selection

Prior to TIDE execution, the appropriate halfwords in the 50 Sequence device definition table should be set in the following format:

0 7	8 15
Device Number	Output Command

The device definition halfwords are:

Teletypewriter inputs no printing	X'0294'
Teletypewriter inputs with printing	X'02A4'
Teletypewriter outputs	X'0298'
High Speed Paper Tape input	X'0399'
High Speed Paper Tape output	X'0392'
Line Printer	X'0780'

Device selection locations in the 50 Sequence device definition table approppropriate to TIDE are:

Location	Symbol	Use With TIDE Operations[1]
X'7A'	BOUTDV	Reproduce, Output
X'7C'	SINDV	Append, Insert, Reproduce, Skip
X'7E'	LISTDV	Print

3.3 Starting Location

If the execution of TIDE is started at the origin (ORG), the text and line buffer pointers are set to the first locations of the buffers. Registers and appropriate locations are initialized, and the message TIDE along with the Command Mode (←) are printed.

If TIDE is started at location ORG + 4, no initialization occurs, the current state of the buffers and pointers is unchanged, and the Command Mode (←) is printed.

1. Refer to Table 3

3.4 Tape Format

Punched tapes produced by the Output (O) command of TIDE are always in the standard source tape format. Each line of text is punched followed by a CR, LF, and eight rubouts (RO). The format for each character is seven bit ASCII code except for the RO which is eight bit ASCII code.

Input tapes for TIDE should be in the standard source tape format; however, the minimum tape format requirements are that each line must be terminated by a carriage return. and contain no more than 67 characters including the carriage return. In addition, successive statements must be separated by at least five or six non-printing characters. If a line length error (?) (Section 2.7) occurs, the tape must be manually adjusted to the next line. If the text buffer capacity is exceeded (error !), manually backspace the tape one line. In the latter case, editing can be done on the text already in the text buffer.

3.5 Text Buffer Size

When loaded, TIDE provides a text buffer for 1000 characters. The user may adjust the size of the text buffer by inserting the beginning and ending absolute addresses into locations X'0008'R and X'000A'R respectively. The text buffer may be located anywhere in core providing it does not overwrite TIDE or the I/O device selection addresses. When started at ORG, TIDE tests the text buffer limits and if they overwrite TIDE, forces them to the locations originally specified in the editor. The first text buffer location in the editor is X'0A6E'R and the last location is X'0E54'R.

APPENDIX 1
TIDE RESPONSES

SYMBOL	DEFINITION
←	Command Mode
*	Edit Mode
?	Error Message
#	Rubout Acknowledgement
!	Text Buffer Overflow

TELETYPE TIDE CONTROL

KEY[1]	DEFINITION/ACTION
*	Lists the open line.
.	Lists the last line in the test buffer.
↑	Lists the byte count of the current contents of the text buffer.
LF	Opens and lists the line following the current open line.
CR	Opens and lists the line preceding the current open line.
RO	Erase the line just typed from the keyboard, cancels an incorrect command.
←	Deletes the last character typed.
BK	Halts I/O. See Table 4.

1. Section 2.3 describes these symbols as used in line addressing.

SPECIAL TIDE ADDRESSES

HEXADECIMAL LOCATION	DEFINITION
0000 + Bias	Starting location. Program will initialize and reset buffer pointers.
0004 + Bias	Restart location. Program will not initialize and buffer pointers are not reset.
0008 + Bias	Location which defines the first address of the text buffer. This address must not overwrite the TIDE program.

APPENDIX 1 (Continued)

HEXADECIMAL LOCATION	DEFINITION
000A + Bias	Location which defines the last address of the text buffer. This address must be within core limits and greater than the starting address of the text buffer.
X'0A6E' + Bias	TIDE defined first address of text buffer.
X'0E54' + Bias	TIDE defined last address of text buffer.

TABLE OF CONTENTS

MATH ROUTINE LIBRARY ABSTRACTS AND DESCRIPTIONS

1. INTRODUCTION

This publication provides programming information on the GE-PAC 30 Math Routine Library. An abstract and a brief program description are provided for each subroutine. To aid in locating data on a particular subroutine, the following index is provided.

INDEX

2. 7-001 BINARY TO BCD (INTEGER) - SINGLE PRECISION

2.1 Abstract

This subroutine converts a halfword binary integer number, in two's complement form, to its BCD equivalent (ASCII) sign plus five digit form.

The argument is divided by 10, and the quotient forms one of the decimal digits. The remainder is again divided by 10 to form another decimal digit. This process is continued until five decimal digits have been formed.

The conversion introduces no errors in the result.

The average execution time of this subroutine is 3.3 milliseconds.

The subroutine occupies X'C8' bytes of memory.

2.2 Description

2.2.1 Calling Sequence.

```
BAL 15,SIBTOD
DC  A(ARG)
DC  A(RESULT)
```

A(ARG) is the address that contains the halfword argument to be converted. A negative number must be represented in two's complement form. The argument is assumed to be an integer number.

A(RESULT) is the starting address in which the resultant binary coded decimal number is stored. Six bytes must be reserved for storage of the result. The first byte contains the sign, the next byte contains the most significant decimal digit, etc. Five BCD digits plus sign are developed. All resultant digits and sign are in Teletypewriter ASCII code.

Upon completion of the subroutine, control is returned to the first instruction following the calling sequence.

2.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

2.2.3 Algorithm. The algorithm consists of continued division of the argument by 10^n. The quotient formed after each division is equal to the binary coded decimal digit. The decimal position of the digit is denoted by the exponent of 10 used in the division. The exponent of 10 is decremented by one, and divided into the remainder to form the next BCD digit. This is repeated until five BCD digits have been formed. If the argument is negative, the two's complement of the number is formed before conversion is attempted. A negative sign is then affixed to the resultant BCD number. Positive arguments result in a plus sign being affixed to the resultant BCD number.

2.2.4 Accuracy. The conversion introduces no error. Five decimal digits plus sign are developed.

2.2.5 Timing. The conversion requires 3.2 milliseconds for a positive argument, and 3.4 milliseconds for a negative argument.

2.2.6 Size. This subroutine requires X'C8' bytes of memory.

3. 7-002 BINARY TO BCD (FRACTIONAL) - SINGLE PRECISION

3.1 Abstract

This subroutine converts a halfword binary fractional number, in two's complement form, to its BCD equivalent (ASCII) sign plus five digit form.

The argument is multiplied by 10 and the integer portion of the product (formed in the most significant halfword) forms one of the decimal digits. The least significant half of the product is again multipled by 10 to form another decimal digit. This process is continued until five decimal digits are formed.

The result of the conversion is accurate to five decimal digits plus or minus one (±1) in the least significant digit. This is because the decimal equivalent of a rational binary number may be irrational. Truncation is performed, rather than rounding.

The average execution time of this subroutine is 2.8 milliseconds.

The subroutine occupies X'C0' bytes of memory.

3.2 Description

3.2.1 <u>Calling Sequence</u>.

```
BAL 15, SFBTOD
DC A(ARG)
DC A(RESULT)
```

A (ARG) is the address that contains the halfword argument to be converted. A negative number must be represented in two's completment form. The argument is assumed to be a fractional number.

A (RESULT) is the starting address in which the resultant binary coded decimal number is stored. Seven bytes must be reserved for storage of the result. The first byte contains the sign, the next byte contains a decimal point followed by the most significant decimal digit, etc. Five BCD digits plus sign and decimal point are developed. All characters are in Teletypewriter ASCII code.

Upon completion of the subroutine, control is returned to the first instruction following the calling sequence.

3.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

3.2.3 Algorithm. The algorithm consists of continued multiplication of the argument by 10^n. When the argument is multiplied by 10, the most significant half of the product forms the most significant decimal digit. The least significant half of the product is again multiplied by 10 to form the next decimal digit. This process is continued until five decimal digits have been formed. If the argument is negative, the two's complement of the number is formed before conversion is attempted. A negative sign is then affixed to the resultant BCD number. Positive arguments result in a plus sign being affixed to the resultant BCD number. In any case, a decimal point follows the sign.

3.2.4 Accuracy. The results of the conversion are accurate to five decimal digits plus or minus one (± 1) in the least significant digit. This error occurs when the argument to be converted is irrational in the base 10 system.

3.2.5 Timing. Conversion requires 2.62 milliseconds for a positive argument and 2.85 milliseconds for a negative argument.

3.2.6 Size. The subroutine occupies X'C0' bytes of memory.

4. 7-003 BCD TO BINARY (INTEGER) - SINGLE PRECISION

4.1 Abstract

This subroutine converts a BCD integer number, in sign magnitude form, to its binary equivalent in two's complement form. The absolute magnitude of the BCD number may not exceed +32,767 or -32,768, since these are the largest numbers which can be expressed in 16 bits.

The algorithm used in the conversion is successive multiplication and addition. Let ABCDE be five decimal digits, and Y be equal to the binary equivalent. Then:

$$Y_2 = E_{10} + D_{10}(10_2)1 + C_{10}(10_2)^2 + B_{10}(10_2)^3 + A_{10}(10_2)^4$$

The conversion introduces no errors in the result. The average execution time of this program is 3.2 milliseconds.

The subroutine occupies X'E4' bytes of memory.

4.2 Description

4.2.1 Calling Sequence.

```
BAL  15,SIDTOB
DC   A(ARG)
DC   A(RESULT)
```

A(ARG) is the starting address of a sign plus five binary coded decimal (BCD) integer digit argument. Leading zeros, if any, must appear. The argument occupies six consecutive bytes. Negative numbers are represented in sign magnitude form. The Teletypewriter (ASCII) minus character is used to denote negative arguments. The argument appears as ±NNNN.

A(RESULT) is the address that contains the result of the conversion. The result is one halfword (2 bytes) long. Negative results are expressed in two's complement form.

4.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

4.2.3 Algorithm. The algorithm consists of successive multiplication and addition. The most significant digit of the argument is multiplied by 10^4, the next most significant digit by 10^3, etc. The products are then added to form the equivalent binary number. If the argument was negative, the two's complement of the result is formed. Since the result must be represented by a maximum of 16 bits, an attempt to convert a BCD number outside the range -32,768 to +32,767 results in a register overflow and the operation is not performed. This is determined, in the algorithm, by testing the result for negative values (before complementing). If the value is negative, an overflow has occurred and the subroutine exits without modifying the cells at ∧(RESULT).

4.2.4 Accuracy. The conversion algorithm introduces no errors. If the range of the argument is -32,768 to +32,767, 5 decimal digits are converted to 16 binary bits with no error. If the number is outside this range, no equivalent binary number is generated.

4.2.5 Timing. This subroutine requires 3.1 milliseconds for a positive argument, and 3.25 milliseconds for a negative argument.

4.2.6 Size. This subroutine requires X'E4' bytes of memory.

5. 7-004 BCD TO BINARY (FRACTIONAL) - SINGLE PRECISION

5.1 Abstract

This subroutine converts a BCD fractional number, in sign magnitude form, to its binary equivalent in two's completment form.

The algorithm used in the conversion is successive division and addition. Let Y(10)=.ABCDE. Then:

$$Y_{(2)} = \frac{A}{(10)_{10}} + \frac{B}{(10)^{2}_{10}} + \frac{C}{(10)^{3}_{10}} + \frac{D}{(10)^{4}_{10}} + \frac{1/8E}{(10)^{4}_{10}}$$

The error introduced by this approximation is no more than +1 in the least significant digit. A cancelling error of 1 in the least significant digit may occur in trunction error when the last remainder is ignored. The total error generated by this conversion is no more than ± 1 in the least significant bit.

The average execution time of this program is 3.8 milliseconds.

The subroutine occupies X'FC' bytes of memory.

5.2 Description

5.2.1 Calling Sequence.

```
BAL 15, SFDTOB
DC      A(ARG)
DC      A(RESULT)
```

A(ARG) is the starting address of a sign plus five binary coded decimal (BCD) fractional digit argument. Leading zeros, if any, must appear. The argument occupies six consecutive bytes. Negative numbers are represented in sign magnitude form. The Teletypewriter (ASCII) minus character is used to denote negative arguments. The argument appears as ±NNNNN.

A(RESULT) is the address that contains the result of the conversion. The result is one halfword (2bytes) long. Negative results are expressed in two's complement form.

Upon completion of the subroutine, control is returned to the first instruction following the calling sequence.

5.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

5.2.3 Algorithm. The algorithm consists of successive division and addition. The BCD digits are aligned before division so that the first digit of the result, be it zero or one, will occur in the 2^{14} binary position. The most significant BCD digit is then divided by 10 and the quotient is stored. The next BCD digit is fetched, the remainder from the previous division is added to the new digit, and the result is properly aligned. The shifted result is then divided by 10^2 etc. The last digit should be divided by 10^5, however, this can not be represented in 16 bits. The last digit is effectively shifted right 3 places which divides it by 8 so that it will be divided by 80,000, rather than 100,000 in converting the last digit. This results in an answer that is larger than it should be (but will differ only by the least significant bit). The error is partially cancelled by truncation of the last remainder, which tends to make the result smaller than it should be. When all five digits have been divided, the resulting quotients are added and the result is a 16 bit binary representation of the BCD fractional number. If this BCD number was negative, the two's complement of the result is taken.

5.2.4 Accuracy. The total error is no more than ± 1 in the least significant bit.

5.2.5 Timing. This subroutine requires 3.74 milliseconds for a positive argument, and 3.92 milliseconds for a negative argument.

5.2.6 Size. This subroutine requires X'FC' bytes of memory.

6. 7-005 MULTIPLY - SINGLE PRECISION

6.1 Abstract

This subroutine performs a binary multiplication of two halfword (2 byte) operands and yields a fullword (4 byte) result.

Negative operands must be in two's complement form. Negative results are in two's complement form.

The multiplier (OP1) is shifted right. If the bit shifted out is a one, the multiplicand is added to the partial product and the result is shifted right once. If the bit shifted out is a zero, the partial product is right shifted once without adding. This is repeated until all bits of the multiplier have been scanned. Upon completion of the scan, the result is in the registers which have been accumulating the partial product.

The two's complement of negative operands is taken. If the result is to be negative, the two's complement of the result is taken.

The subroutine introduces no errors in the result.

The average execution time of the subroutine is 4.4 milliseconds.

The subroutine occupies X'106' bytes of memory.

6.2 Description

6.2.1 Calling Sequence.

```
BAL   15,SMULT
DC    A(OP1)
DC    A(OP2)
DC    A(RESULT)
```

A(OP1) is the address of the first operand (multiplier).

A(OP2) is the address of the second operand (multiplicand). The operands are halfword (2 byte) arguments which, when negative, are represented in two's complement form.

A(RESULT) is the address in which the result will be stored. The result is a fullword (4 bytes). Negative results are represented in two's complement form.

6.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

6.2.3 Algorithm. Multiplication is performed by successive addition and shifting according to the basic definition of multiplication. The multiplier (OP1) is shifted right once. If the bit shifted out is a one, the multiplicand is added to the partial product and the result is shifted right once. If the bit shifted out is a zero, the partial product is right shifted once without adding. This process is repeated 16 times. At this point all bits of the multiplier have been scanned. Operands which are negative are complemented. If the result is to be negative, the two's complement of the result is taken.

The algorithm introduces no error. A 32 bit (fullword) product is developed.

6.2.4 Timing. The average execution time of the subroutine is 4.4 milliseconds. The worst case execution time is 5.60 milliseconds. The best case time is 3.8 milliseconds.

6.2.5 Size. The subroutine occupies X'106' bytes of memory.

7. 7-006 DIVIDE - SINGLE PRECISION

7.1 Abstract

This subroutine performs a binary division of a full word dividend (OP1) by a halfword divisor (OP2) and yields a halfword quotient and a halfword remainder.

Negative operands must be in two's complement form. Negative results are in two's complement form.

The divisor is subtracted from the dividend. If the subtraction can be made without generating an overflow (sign of the result changes) a one is placed in the quotient register and the difference and quotient registers are left shifted once. If the subtraction results in an overflow the dividend is restored, a zero is placed in the quotient register and both are shifted left once. This process is repeated 16 times. Upon completion, the 32 bit dividend has been replaced by a 16 bit quotient and a 16 bit remainder.

The two's complement of negative operands is taken. If the result is negative, the two's complement of the quotient is taken. If the dividend was negative, the two's complement of the remainder is taken.

If the arguments are such that the quotient will exceed register limits, eg: $Q > 2^{15} - 1$ or $Q < -2^{15}$, an error return is made.

The subroutine introduces no errors in the result.

The average execution time of the subroutine is 7.0 milliseconds.

The subroutine occupies X'164' bytes of memory.

7.2 Description

7.2.1 <u>Calling Sequence.</u>

```
        BAL  15,SDIV
        DC   A(OP1)
        DC   A(OP2)
        DC   A(RESULT)
ERRRTN  DS   4
```

A(OP1) is the address of the first operand (dividend). The dividend is a fullword (4 byte) argument.

A(OP2) is the address of the second operand (divisor). The divisor is a halfword (2 byte) argument. Both arguments, if negative, must be in two's complement form.

A(RESULT) is the address in which the quotient and remainder are stored. Since both the quotient and remainder are two bytes long, four bytes (a fullword) must be reserved. The remainder is stored first, followed by the quotient. In the event of a divide overflow, no result is stored.

ERRRTN is the address which the subroutine returns to if the quotient can not be represented in 16 bits (divide overflow). Generally, the programmer uses these 4 bytes to store a branch or BAL so that some correction to the operands may be made.

7.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

7.2.3 Algorithm. Division is performed by successive subtraction and shifting according to the basic definition of division. The divisor is subtracted from the dividend. If the subtraction can be made without generating an overflow (sign of the result changes) a one is placed in the quotient register and both are shifted left once. This process is repeated 16 times. Upon completion, the 32 bit dividend has been replaced by a 16 bit quotient and a 16 bit remainder.

Operands which are negative are two's complemented. If the result is to be negative, the two's complement of the quotient is taken. If the dividend was negative, the two's complement of the remainder is taken.

If the arguments are such that the quotient will exceed the register limits; eg: $Q > 2^{15}-1$ or $Q < -2^{15}$, to an error return is made.

The algorithm is the implementation of the basic definition of divide.

7.2.4 Accuracy. The subroutine introduces no errors if Q lies in the range $-215 \leq Q < 2^{15}$. A 16 bit quotient and a 16 bit remainder are developed.

7.2.5 <u>Timing.</u> The average execution time of this subroutine is 7.0 milliseconds. The worst case execution time is 7.81 milliseconds. The best case execution time is 6.5 milliseconds.

7.2.6 <u>Size.</u> The subroutine occupies X'164' bytes of memory.

8. 7-007 SQUARE ROOT - SINGLE PRECISION

8.1 Abstract

This subroutine extracts the square root of a halfword argument. Negative arguments are complemented before root extraction.

The algorithm used in this conversion is the Newton-Raphson approximation. The argument is prescaled so that it lies in the range $1 > X \geq 1/4$. Upon completion, the resultant root is postscaled to its correct value.

The maximum error generated by this subroutine is two places in the last decimal digit. The maximum relative error is .00003.

The average execution time of this subroutine is 3.1 milliseconds.

The subroutine occupies X'F0' bytes of memory.

8.2 Description

8.2.1 Calling Sequence.

```
BAL  15,SSQRT
DC   A(ARG)
DC   A(RESULT)
```

A(ARG) is the address of the halfword argument. The argument is considered to be a fractional halfword (16 bit) binary number. Negative numbers, in two's complement form, are treated as positive quantities and a positive real root is extracted.

A(RESULT) is the address where the resultant root will be stored. The root is one halfword in length.

8.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

8.2.3 Algorithm. The argument is prescaled by the subroutine so that it lies in the range $1 > X \geq 1/4$. The argument is scaled within this range so that the scale factor is even. If $X = A2^s$ then $\sqrt{X} = \sqrt{A}\,2^{s/2}$ (where s is the scale factor). If the argument is negative, its two's complement is taken before root extraction is attempted.

Three successive Newton-Raphson approximations are used to determine the root. A trial root is first developed using the equation: $Zo = 1/2 + 1/2X$ where X is the argument. Three successive approximations are then made using the equation: $Zi + 1 = 1/2\ (X/Zi - Zi) + Zi$.

8.2.4 Timing. The average execution time of the subroutine is (3.1 + .1695L) milliseconds. The worst case execution time is (3.3 + .1695L) milliseconds. The best case execution time is (2.9 + .1695L) milliseconds. Where L is the number of shifts necessary to scale the number to the range $1 < X \leq 1/4$.

8.2.5 Size. The subroutine occupies X'F0' bytes of memory.

9. 7-008 LOG BASE 2, E, 10 - SINGLE PRECISION

9.1 Abstract

This subroutine calculates the log to base two, epsilon or 10 of a halfword positive argument scaled as a fraction with an exponent. The base and exponent is supplied to the subroutine in the calling sequence.

Negative arguments cause the subroutine to make an error exit and the operands are not changed.

The result of this subroutine is a halfword negative (in two's complement form) log properly scaled as determined by the fractional argument.

The log is calculated by means of a polynomial expansion employing Chebyshev coefficients.

The maximum error generated by this subroutine is five places in the last decimal digit. The maximum relative error is .00005.

The average execution time of this subroutine is 2.75 milliseconds.

The subroutine occupies X'16A' bytes of memory.

9.2 Description

9.2.1 Calling Sequence.

```
BAL 15, SLOG
     DC A(ARG)
     DC A(EXP)
     DC A(BASE)
     DC A(RESULT)
     DS 4                    ERROR RETURN
```

A(ARG) is the address of the halfword argument. The argument is considered to be a fractional positive halfword (16 bit) binary number. Negative arguments cause an error return.

A(EXP) is the address of the halfword which contains the exponent of the argument so that any number in the range $2^{15}-1$ to $2^{-15}-1$ may be represented. For example, proper scaling will cause, the arguments 0.7, 7.0, 70.0 etc, to appear in A(ARG) as X'599A' = 0.7. A(EXP) would contain 0 for 0.7, 1 for 7.0, 2 for 70.0 etc. Negative exponents must be in two's complement notation.

A(BASE) is the address of a halfword whose contents determine to which base the log will be taken. For base two, A(BASE) must contain 0000. For base epsilon A(BASE) must contain X'0004'. For base ten A(BASE) must contain X'0002'.

A(RESULT) is the halfword address which contains the resultant log. A(RESULT) contains the characteristic and the next halfword A(RESULT)+2 contains the mantissa. Negative logs are represented in two's complement notation.

9.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

9.2.3 Algorithm. The argument is first prescaled so that it lies in the range. $1 > X \geq 1/2$. This is done by shifting the argument left and accumulating the number of shifts as a scale factor 'b'. The scale factor is then added to the exponent. The effective argument (U) is then formed.

$$U = \frac{X + \sqrt{1/2}}{X - \sqrt{1/2}}$$

The log to the base two is then approximated by means of a polynomial expansion employing Chebyshev coeffieients.

$$LOG_2\ X = \left[\sum_{1}^{2} C_{2k-1}\ U^{2k-1}\right] - 1/2 \text{ where } C_1 = 2.88523,\ C_3 = 0.98353$$

The log is then converted to its proper base by use of the identity.

$$LOG_b\ M = (LOG_c\ M)(LOG_b\ C)$$

The negative log is then added to the sum of the scale factor and exponent to form the true mantissa and characteristic.

9.2.4 Accuracy. The relative accuracy of this subroutine is $\pm$ .00005 where $E_{rel} = \frac{\text{APPROX-FUNCT}}{\text{FUNCTION}}$ The last decimal digit may be in error by five places.

9.2.5 Timing. The average execution time of the subroutine is 2.75 milliseconds plus scaling time for arguments less than 1/2 (log base 10). Scaling time is equal to 120N microseconds where N is equal to the number of shifts necessary to make the argument fall in the range $1 > X \geq 1/2$. Worst case execution time is 3.45 milliseconds plus scaling time (log base epsilon). Best case time is 2.5 milliseconds plus scaling time (log base 2).

9.2.6 Size. The subroutine occupies X'16A' bytes of memory.

10. 7-010 SINE/COSINE - SINGLE PRECISION

10.1 Abstract

This subroutine generates the sine and cosine of an argument angle. The argument is supplied to the subroutine in fractions of a degree, eg. each degree is broken into one hundred parts.

The sine and cosine are generated by polynomial approximation employing Chebyshev coefficients.

$$\mathrm{SIN}\,\frac{\pi}{2}X = C_1X + C_3X^3 + C_5X^5 \quad \text{where } -1<X<1$$

and $C_1 = 1.57063$

$C_3 = -.64323$

$C_5 = .07271$

$$\mathrm{Cos}\,\frac{\pi}{2}X = \mathrm{Sin}\,\frac{\pi}{2}(90-X)$$

The maximum relative error introduced by this subroutine is $\pm.00004$. The result will be in error by no more than 4 in the fifth decimal digit.

The average execution time is 2.3 milliseconds.

The subroutine occupies X'100' bytes of memory.

10.2 Description

10.2.1 Calling Sequence.

```
BAL  15,SINE              BAL  15,COSINE
DC   A(ARG)        or     DC   A(ARG)
DC   A(RESULT)            DC   A(RESULT)
```

A(ARG) is the address that contains the halfword argument angle. The angle is expressed as an integer in hundredths of a degree, for example 131.24 degrees = 13124×10^{-2} = X'3344'. The hex number X'3344' would be the argument for 131.24 degrees. Arguments greater than 180 degrees are expressed as negative angles less than 180 degrees, for example 228.76 degrees = -131.24 degrees = X'CCBC'. The two's complement hex number X'CCBC' would be the argument for -131.24 degrees.

A(RESULT) is the address that contains the halfword result. Negative values are expressed in two's complement form.

10.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

10.2.3 Algorithm. A polynomial approximation using Chebyshev coefficients is used to calculate the sine. If the cosine function is called for, the complement of the argument angle is taken as the effective argument to the sine routine. The argument is reduced to an angle in the range $-1 < X < 1$ if necessary by subtraction of 90 degrees.

$$\mathrm{SIN}\ \frac{\pi}{2}X = C_1X + C_3X^3 + C_5X^5 \text{ where } -1 < X < 1$$

and $C_1 = 1.57063$

$C_3 = -.64323$

$C_5 = .07271$

10.2.4 Accuracy. The maximum relative error (ER) introduced by the subroutine is $\pm.0004$. ER = (APPROXIMATION-FUNCTION)/FUNCTION The result will be in error by no more than 4 in the fifth decimal digit.

10.2.5 Timing. The average execution time of this subroutine is 2.3 milliseconds. The worst case time is 2.6 milliseconds. The best case time is 2.0 milliseconds.

10.2.6 Size. The subroutine occupies X'100' bytes of memory.

11. 7-011 ARCTANGENT - SINGLE PRECISION, RESTRICTED

11.1 Abstract

This subroutine calculates the arctangent for any tangent argument which lies in the range $1 > X \geq -1$. The resultant angle lies in the range $\frac{\pi}{2} > 0 \geq -\frac{\pi}{2}$.

For arguments $-1 > X \geq 1$, the unrestricted arctangent (7-018) subroutine must be used. (See Section 13.)

Both the argument and the result are one halfword in length (16 bits). Negative numbers are represented in two's complement notation.

The algorithm used in this subroutine is approximation by polynomial expansion using Chebyshev coefficients.

The relative accuracy of this subroutine is $\pm.00005$ which provides four and one half decimal digits of accuracy.

The execution time of this subroutine is 1.95 milliseconds.

The subroutine occupies X'82' bytes of memory.

11.2 Description

11.2.1 Calling Sequence.

```
BAL  15, ARCTAN
DC   A(ARG)
DC   A(RESULT)
```

A(ARG) is the halfword address which contains the tangent argument. The argument (X) must lie in the range $1 > X \geq -1$. Negative values are represented in two's complement notation.

A(RESULT) is the halfword address which contains the resultant angle expressed in radians. Negative results are in two's complement notation.

11.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

11.2.3 Algorithm. The algorithm used in this subroutine is approximation by polynomial expansion using Chebyshev coefficients.

$$\text{ARCTAN } X = \sum_{i=0}^{i=3} C_{2i+1} X^{2i+1} \quad \text{where } 1 > X \geq -1$$

$$C_1 = .999215$$
$$C_3 = -.321182$$
$$C_5 = +.146277$$
$$C_7 = -.038993$$

11.2.4 Accuracy. The relative error (ER) of this subroutine is $\pm .00004$ where $ER = \frac{\text{APPROX-FUNCTION}}{\text{FUNCTION}}$. This subroutine provides four and one half decimal digits of accuracy.

11.2.5 Timing. The execution time of this subroutine is 1.95 milliseconds.

11.2.6 Size. The subroutine occupies X'82' bytes of memory.

12. 7-012 ANGLE CONVERSION - SINGLE PRECISION

12.1 Abstract

This subroutine converts an angle expressed in radians to decimal degrees for use in the sine/cosine subroutine.

The argument is a compound number, the integer portion occupies a halfword and the fraction the next halfword. Negative arguments are not defined, and result in an incorrect result. The argument must lie in the principal range $2\pi > X \geq 0$.

The subroutine converts the argument to decimal degrees. Angles larger than 180 degrees are represented as negative angles less than 180. Negative angles are represented in two's complement form. The result occupies one halfword (16 bits).

The subroutine has a relative accuracy of $\pm .00003$.

The average execution time of the subroutine is 2.5 milliseconds.

The subroutine occupies X'B6' bytes of memory.

12.2 Description

12.2.1 Calling Sequence.

```
BAL 15, RADDEG
     DC A(ARG)
     DC A(RESULT)
```

A(ARG) is the halfword address which contains the integer part of the radian argument. The next halfword A(ARG)+2, contains the fractional part of the radian argument. Negative arguments are not defined, and result in an incorrect answer. The argument must be a principal value, eg. $2\pi > X \geq 0$.

A(RESULT) is the halfword address which contains the argument expressed in decimal degrees. Arguments greater than π result in negative angles less than 180 degrees. For example $3\pi/2$ converts to -90.00 degrees. The result is of proper form for entry to the sine or cosine routines.

12.2.2 Algorithm. The argument, if greater than π, is first scaled by subtracting π so that it lies in the range $\pi > X \geq 0$. The scaled radian argument is then multiplied by 57.2958 to convert to degrees. If the original argument was greater than π, the result is subtracted from 360 degrees and the two's complement formed. The result is packed to a halfword such that $18000 >$ ANGLE IN HUNDREDTHS ≥ 0.

12.2.3 Accuracy. The relative error of this subroutine is $\pm .00003$ where $ER = \dfrac{\text{APPROX-FUNCTION}}{\text{FUNCTION}}$

12.2.4 Timing. The best case execution time of this subroutine is 2.3 milliseconds. The average execution time is 2.5 milliseconds. The worst case execution time is 2.9 milliseconds.

12.2.5 Size. The subroutine occupies X'B6' bytes of memory.

13. 7-018 ARCTANGENT - SINGLE PRECISION, UNRESTRICTED

13.1 Abstract

This subroutine calculates the arctangent for any ratio (R) such that $2^{15}-1 \geq R > -2^{15}$. The resultant angle ($\theta$) is such that $2\pi > \theta \geq 0$.

The arguments are one halfword (16 bits) in length and if negative are represented in two's complement notation.

The result is one fullword in length. The integer and fractional portions of the result each occupy one halfword.

The algorithm used in this subroutine is approximation by polynomial expansion using Chebyshev coefficients.

The relative accuracy of this subroutine is $\pm.00005$ which provides four and one half decimal digits of accuracy.

The average execution time of this subroutine is 3.3 milliseconds.

The subroutine occupies X'1DC' bytes of memory.

13.2 Description

13.2.1 Calling Sequence.

```
BAL  15,  ATANUR
DC    A   (Y)
DC    A   (X)
DC    A   (RESULT)
```

A (Y) is the halfword address that contains the Y value (DIVIDEND) of the tangent ratio. Negative values are represented in two's complement form.

A (X) is the halfword address that contains the X value (DIVISOR) of the tangent ratio. Negative values are represented in two's complement notation.

A(RESULT) is the halfword address that contains the integer portion of the resultant angle. A (RESULT) +2 is the halfword address that contains the fractional portion of the resultant angle. The angle is expressed in radians and will lie in the range $2\pi > \theta \geq 0$.

13.2.2 Register And Processor Status. The General Registers are not affected. The condition code of the PSW is modified.

13.2.3 Algorithm. The Y ordinate is first compared to the X ordinate. If Y>X the cotangent is formed by taking the ratio X/Y rather than the tangent Y/X. The signs of X and Y are noted to determine which sector the angle will lie in. The negative arguments, if any, are then complemented so that the effective tangent (U) lies in the range $1 > U \geq 0$.

The effective arctangent is then calculated by means of a polynomial approximation using Chebyshev coefficients.

$$\text{ARCTAN } U = \sum_{i=0}^{i=3} C_{2i+1} U^{2i+1} \quad \text{where } U = \frac{|X|}{|Y|} \text{ or } \frac{|Y|}{|X|}$$

such that $1 > U \geq 0$

$C_1 = .999215$
$C_3 = -.321182$
$C_5 = +.146277$
$C_7 = -.038993$

The angle is then relocated to its correct sector and subtracted from $\frac{\pi}{2}$ if the cotangent was formed.

13.2.4 Accuracy. The relative error (ER) of this subroutine is $\pm .00005$ where $ER = \frac{\text{APPROX-FUNCT}}{\text{FUNCTION}}$. The subroutine provides four and one half decimal digits of accuracy.

13.2.5 Timing. The worst case execution time of this subroutine is 3.8 milliseconds. The average execution time is 3.3 milliseconds. The best case execution time is 3.1 milliseconds.

13.2.6 Size. The subroutine occupies X'1DC' bytes of memory.

FIXED-POINT MULTIPLY/DIVIDE TRAP SUBROUTINE

1. INTRODUCTION

This subroutine, which is intended for linkage to the illegal instruction interrupt, is appropriate for those GE-PAC 30 processors without fixed-point multiply/divide instructions. The High Speed option provides Multiply, Divide, Read Block, and Write Block instructions. In processors without the High Speed Option, the execution of a Multiply or Divide instruction causes an Illegal Instruction Interrupt. This TRAP Subroutine, when linked to the Illegal Instruction Interrupt, performs the multiply or divide operation as specified by the particular instruction executed.

2. INSTRUCTION FORMATS

The TRAP subroutine implements Multiply Halfword and Divide Halfword instructions as described in the following paragraphs.

2.1 Multiply Halfword

MHR R1, R2 [RR]

0 7	8 11	12 15
ØC	R1	R2

MH R1, A(X2) [RX]

0 7	8 11	12 15	16 31
4C	R1	X2	A

The 16-bit second operand is multiplied with the General Register specified by R1 + 1. The first operand, R1, must specify an even numbered register. The resulting 32-bit product is contained in R1 and R1 + 1, an even-odd pair; the second operand is unchanged. The sign of the product is determined by the rules of algebra.

(R1, R1 + 1) ⟵ (R1 + 1) * (R2) [RR]

(R1, R1 + 1) ⟵ (R1 + 1) * (A + (X2)) [RX]

Resulting Condition Code:

Unchanged.

Programming Note:

After multiplication, the most significant 15 bits with sign are contained in R1. The least significant 16 bits are contained in R1 + 1.

2.2 Divide Halfword

DHR R1, R2 [RR]

0 7	8 11	12 15
ØD	R1	R2

DH R1, A(X2) [RX]

0 7	8 11	12 15	16 31
4D	R1	X2	A

The 16-bit second operand is divided into the 32-bit dividend contained in the General Register specified by R1 and R1 + 1. The first operand, R1, must specify an even numbered register. The resulting 15-bit quotient with sign is contained in R1 + 1; a 15-bit remainder with sign is contained in R1, the second operand is unchanged. The sign of the result is determined by the rules of algebra; the sign of the remainder is the same as the sign of the dividend.

(R1 + 1) ⟵ (R1, R1 + 1)/(R2) [RR]
(R1) ⟵ Remainder

(R1 + 1) ⟵ (R1, R1 + 1)/(A + (X2)) [RX]
(R1) ⟵ Remainder

Resulting Condition Code:

Unchanged.

Programming Note:

A quotient which cannot be expressed in 16 bits will cause an Arithmetic Fault interrupt if enabled by bit 3 of the Program Status Word. The operands will remain unchanged.

3. OPERATION

The subroutine is organized as shown in Figure 1, and contains seven principal parts. These parts are used as follows:

1. The first part consists of the first four halfwords in the subroutine. These halfwords contain pointers which are useful for expanding the subroutine to handle other than multiply and divide operations. Expansion for non-multiply divide ops is discussed in a later section.

2. The second part begins at the fifth halfword (SAVE) which is the entry point for the subroutine. This part saves all registers in a data area called BLOK, and sets up some useful constants.

3. The next part of the subroutine picks up the operands for the instruction which caused the Illegal Instruction Interrupt and tests the illegal op-code. If the op-code is not a multiply or divide instruction, the subroutine executes a LPSW TRAP instruction, which loads the Current PSW with the first two halfwords in the subroutine. Unless changed to some other value, these halfwords contain 8FFF/FFFF and the Processor halts since the Wait Bit in the Current PSW gets set.

4. The Multiply routine is used when the illegal op-code specifies a multiply operation.

5. The Divide routine is used when the illegal op-code specifies a divide operation.

6. The EXIT part of the subroutine places the return address (RETN) which is in Register 0, into the halfword at X'32', restores all registers from the data area (BLOK) and executes a LPSW X'30'.

7. The last part, the data area (BLOK), is used to save all 16 General Registers.

The operation of the subroutine is illustrated in Figure 2.

During the divide operation, if an overflow occurs, the subroutine performs the appropriate Divide-Fault Interrupt if specified in the Old PSW at location X'30'. If the Divide-Fault Interrupt is not enabled, the control is returned to the point immediately following the divide instruction with the operands unchanged.

4. NON-MULTIPLY/DIVIDE INSTRUCTIONS

The TRAP Subroutine is designed for expansion to handle other than multiply and divide instructions. To facilitate this expansion, the symbolic name of the first location in the subroutine (TRAP) is declared as an ENTRY. This means that other programs can refer to the name TRAP and the loader will link the programs accordingly. The two halfwords at TRAP (8FFF/FFFF) are used by the subroutine (LPSW TRAP) when a non-multiply/divide op is detected. In addition, the next two halfwords contain pointers to the restore routine (EXIT) and the data area (BLOK).

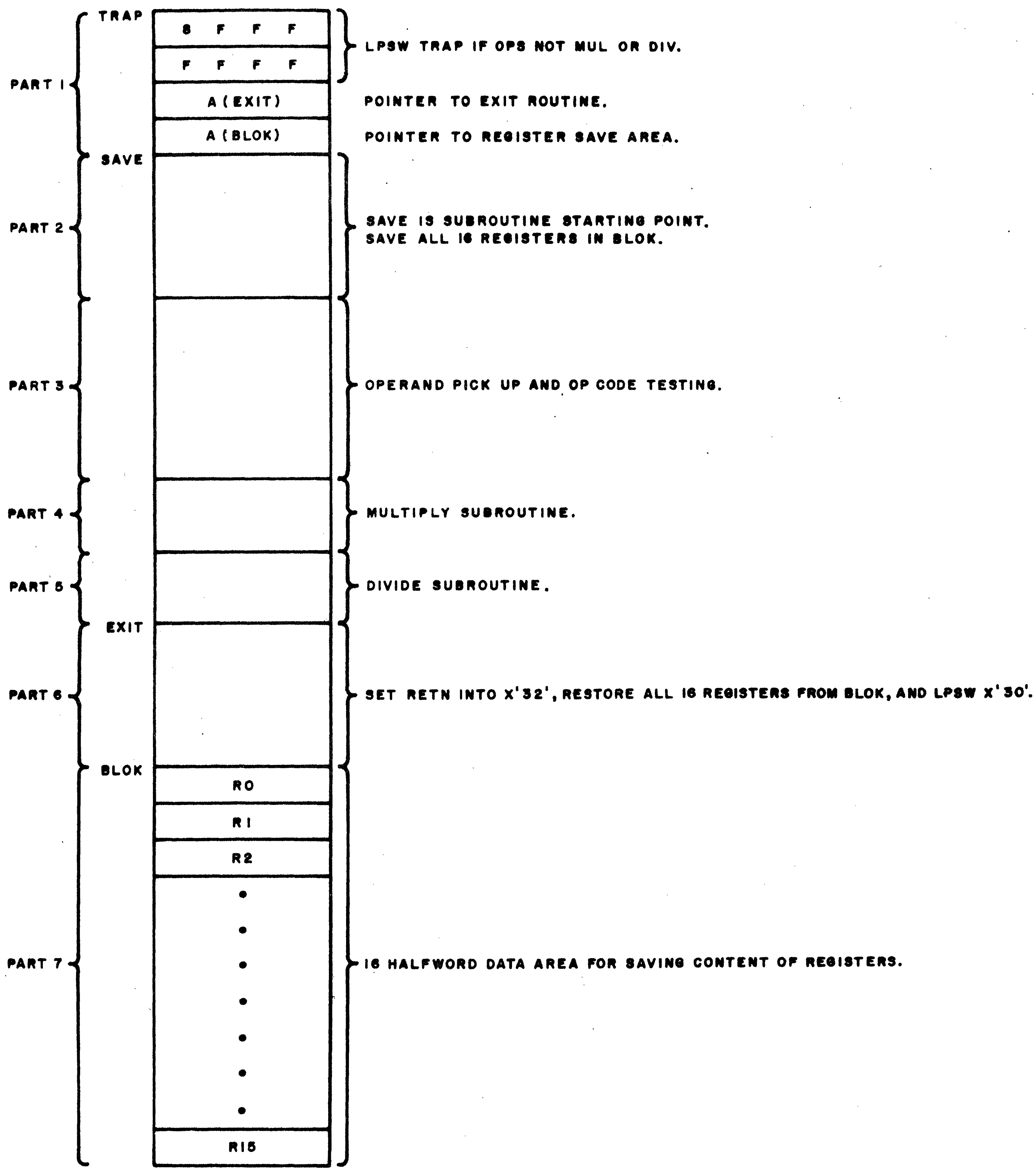


Figure 1. Organization of Fixed-Point Multiply/Divide Trap Subroutine

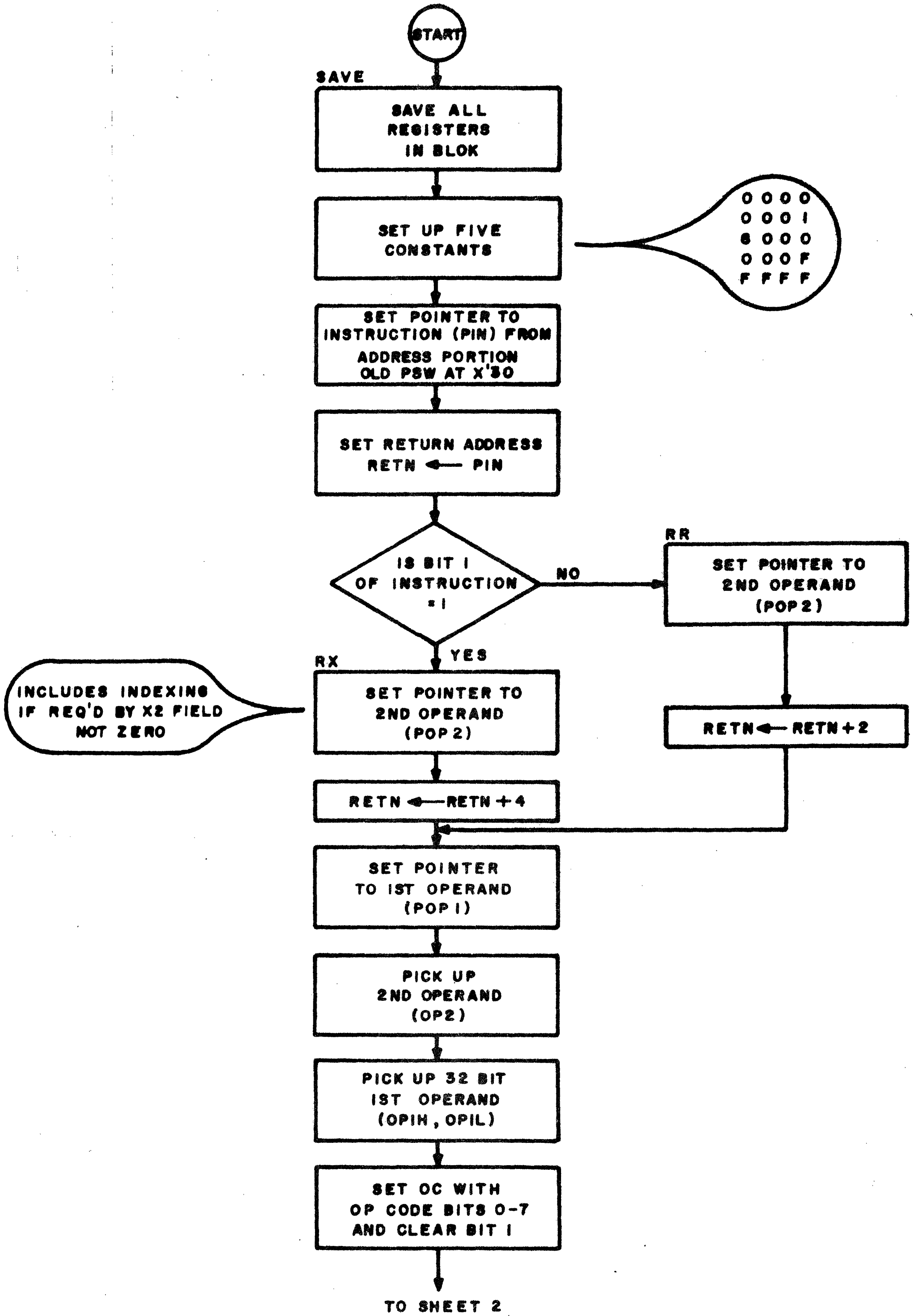


Figure 2. Operation of Fixed-Point Multiply/Divide Trap Subroutine (Sheet 1 of 2)

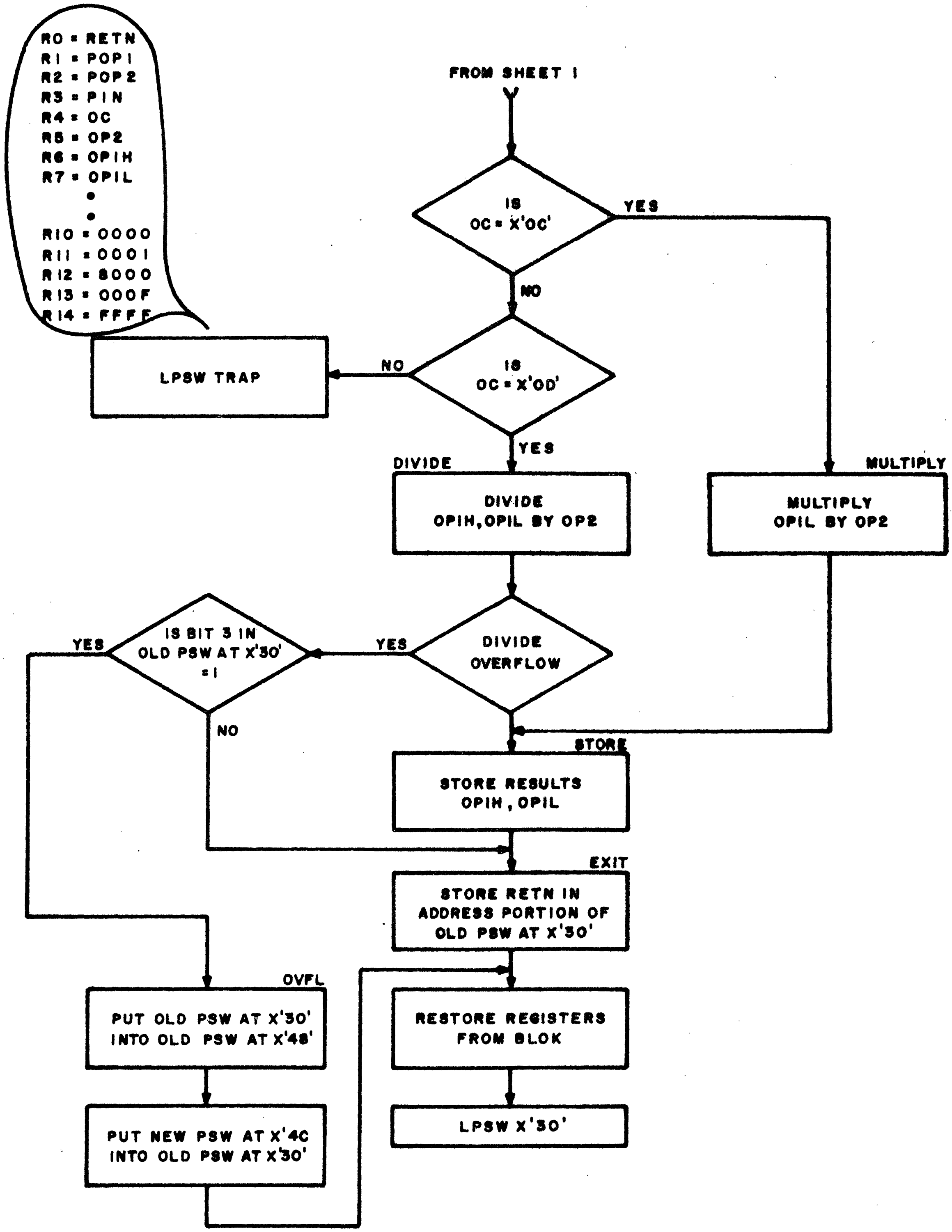


Figure 2. Operation of Fixed-Point Multiply/Divide Trap Subroutine (Sheet 2 of 2)

To extend the TRAP routine, the first two halfwords (8FFF/FFFF) in the subroutine should be set to enable transfer to some routine which further tests the op-code. At the time the LPSW TRAP instruction is executed, the register contents are as shown in Table 1.

Note that the instruction is assumed RR type if bit 1 = 0, and RX type otherwise. Therefore, in the op-code (OC), bit 1 is cleared, since it is no longer needed. If RS instructions are handled, the pointer to operand 2 (POP2), which is the address of the operand in RR and RX instructions, contains the actual value of the operand.

5. USE OF TRAP SUBROUTINE

The program tape 07-021M08 is a relocatable tape and must be loaded with either the Relocating Loader 06-024 or the General Loader 06-025. Since the name TRAP is declared as an ENTRY point, the Relocating Loader will halt at one point with the display panel lights indicating an Improper Control Item. This error halt can be ignored. (See Loader Descriptions, Publication Number 06-025A12 for details).

To use the TRAP subroutine, the New PSW location for illegal instructions at X'34' must be set to point to the TRAP subroutine entry point (TRAP + 8). This PSW is established when the TRAP routine is loaded. During program debugging, it may be necessary to re-establish this PSW pointer if it gets changed.

Note that the TRAP routine is designed with a single-level save/restore mechanism. This means that any processing of illegal instructions, external to the TRAP routine, should not involve other illegal instructions unless the NEW PSW for the ILLEGAL INSTRUCTION INTERRUPT (X'34') is changed. If this pointer is not changed, the Illegal Instruction Interrupt will cause program control to be transferred to location SAVE in the TRAP routine, and the register contents in BLOK will be destroyed.

TABLE 1. REGISTER CONTENTS

Register	Name	Contents
0	RETN	Return Address
1	POP1	Pointer to Operand 1
2	POP2	Pointer to Operand 2
3	PIN	Pointer to Illegal Instruction
4	OC	Op-Code Right Justified with Bit 1 = 0
5	OP2	The Second Operand
6	OP1H	The First Operand, High Order
7	OP1L	The First Operand, Low Order
8	R8	Undefined
9	R9	Undefined
10	0000	Constant = 0
11	0001	Constant = 1
12	8000	Constant = bit 0 only
13	000F	Constant = 15
14	FFFF	Constant = -1
15	R15	Undefined

Note that CLUB, the debug program, also uses the PSW location for illegal instructions. Care should be exercised, therefore, when using CLUB in memory at the same time as the TRAP subroutine. It may be necessary to repeatedly set the PSW location to point to TRAP whenever transferring control out of CLUB to another program. This procedure will inhibit the use of conventional CLUB breakpoints.

The 30-01 execution times required for the TRAP routine are as follows:

Multiply ops	7.3 msec.
Divide ops	8.2 msec.
Other ops	1.5 msec.

The TRAP routine requires 210_{16} or 528_{10} bytes of core memory.

FLOATING POINT PACKAGE DESCRIPTION

1. INTRODUCTION

The Floating Point Package, 07-020, provides in subroutine form, the basic programs required for the manipulation of floating-point numbers in any **GE-PAC 30** system with the high speed option, but without any floating-point instructions per se. The Floating Point Package provides four arithmetic operations: addition, subtraction, multiplication, and division; and four data conversion operations: fixed to floating-point, floating-point to fixed, decimal to binary, and binary to decimal. All operations are performed by calling an appropriate subroutine, where the name of each subroutine is identified as an EXTRN. The Floating Point Package, therefore, can be linked to an assembly language program by using the General Loader.

The basic approach to the use of this package is to load the operands of interest into specific general registers, and to call the appropriate subroutine. On exit from the subroutine, the result is left in specified general registers; other general registers are restored to their original value.

The GE-PAC 30 format for single-precision, floating-point data represents numbers in the range from 5.4×10^{-79} to 7.2×10^{75}, with six digits of precision.

For systems without the high-speed option, the Floating Point Package can be used by employing software multiply/divide subroutines which are linked to the illegal instruction trap in the machine.

2. DATA FORMAT

A floating-point number consists of a signed exponent and a signed fraction. The quantity expressed by this number is the product of the fraction and the number 16 raised to the power of the exponent. The exponent is expressed in excess 64 binary notation; the fraction is expressed as a hexadecimal number having a radix point to the left of the high order digit.

Each floating point value requires two halfwords. The floating point format is:

0	1 ... 7	8 ... 15
S	X	A
B		

where S = sign of the fraction
X = exponent, in excess 64 code
AB = fraction

Sign and magnitude representation is used, in which the sign bit S is zero for positive values, and one for negative values. The fraction AB contains six hexadecimal digits as shown below.

0	1 ... 7	8	... 15
S	X	F_1	F_2
F_3	F_4	F_5	F_6

The value of a floating point fraction can be expressed as:

$$F_1 \cdot 16^{-1} + F_2 \cdot 16^{-2} + F_3 \cdot 16^{-3} + \cdots + F_6 \cdot 16^{-6}$$

Sample values are shown in Table 1.

TABLE 1. SAMPLE FLOATING-POINT VALUES

Data in Hexadecimal	Value
4110 0000	1.0
C110 0000	-1.0
4198 0000	9.5
C080 0000	- .5
0000 0000	0
FFFF FFFF	$-(1-16^{-6}) \cdot 16^{63}$
8010 0000	-16^{-65}
4019 999A	$.1 + 16^{-6}$

A normalized floating-point number has a non-zero, high-order hexadecimal fraction digit (F_1). If one or more high-order fraction digits ($F_1 F_2 \cdots$) are zero, the number is said to be unnormalized. The range of the magnitude (M) of a normalized floating-point number is:

$$16^{-65} \leq M \leq (1 - 16^{-6}) \cdot 16^{63}$$

or approximately

$$5.4 \cdot 10^{-79} \leq M \leq 7.2 \cdot 10^{75}$$

The floating-point value in which all data bits are zero is called true zero. A true zero may arise as the result of an arithmetic operation due to exponent underflow, or when a result fraction is zero due to loss of significance. In general, zero values participate as normal numbers in all arithmetic operations. If the resultant exponent of an addition, subtraction, multiplication, or division overflows, all bits of the exponent and fraction are set, and the correct sign is generated.

3. ARITHMETIC ROUTINES

The subroutine names for arithmetic operations are:

FADD	addition
FSUB	subtraction
FMUL	multiplication
FDIV	division

Each of these subroutines expects two operands, A and B, and generates the result B = A op B, where op = +, -, *, or / depending on which subroutine was used. Prior to calling the subroutine, the A operand should be placed in Registers 6 and 7, and the B operand should be placed in Registers 8 and 9. The subroutine, when called, computes A op B, and leaves the result in place of the B value in Registers 8 and 9. All other registers, including the A value in Registers 6 and 7, are unchanged. The condition code may be changed, however, although the flags are not set in any meaningful way.

The subroutine for the arithmetic operations expect normalized operands, and generate normalized results. If the data is not normalized prior to calling the subroutine, the result may be incorrect.

The arithmetic subroutines will generate true zero on exponent underflow and the maximum value on exponent overflow. No flags are set to indicate that overflow or underflow occurred. There are no error returns associated with arithmetic subroutines.

The calling sequence for each subroutine follows. The EXTRN declarations should appear once at the beginning of the program.

```
        EXTRN       FADD
*
*     PUT A VALUE IN REGISTERS 6, 7
*     PUT B VALUE IN REGISTERS 8, 9
*

        BAL         15, FADD          B <- A+B
*
*     RESULT LEFT IN REGISTERS 8, 9
*     ALL OTHER REGISTERS UNCHANGED
*
*

        EXTRN       FSUB
*
*     PUT A VALUE IN REGISTERS 6, 7
*     PUT B VALUE IN REGISTERS 8, 9
*

        BAL         15, FSUB          B <- A-B
*
*     RESULT LEFT IN REGISTERS 8, 9
*     ALL OTHER REGISTERS UNCHANGED
*
*

        EXTRN       FMUL
*
*     PUT A VALUE IN REGISTERS 6, 7
*     PUT B VALUE IN REGISTERS 8, 9
*

        BAL         15, FMUL          B <- A*B
*
*     RESULT LEFT IN REGISTERS 8, 9
*     ALL OTHER REGISTERS UNCHANGED
*
*

        EXTRN       FDIV
*
*     PUT A VALUE IN REGISTERS 6, 7
*     PUT B VALUE IN REGISTERS 8, 9
*

        BAL         15, FDIV          B <- A/B
*
*     RESULT LEFT IN REGISTERS 8, 9
*     ALL OTHER REGISTERS UNCHANGED
*
*
```

The execution times for these routines are as follows.

Subroutine	30-01	30-02
FADD	4.4 msec	.55 msec
FSUB	4.4	.55
FMUL	4.9	.65
FDIV	7.0	.95

The execution times for FADD and FSUB are average figures, since these times depend significantly on the amount of pre-equalization and post-normalization involved. The times for FMUL and FDIV are quite constant since these operations depend very little on the data involved.

4. CONVERSION ROUTINES

The subroutine names for the conversion operations are:

FFIX	floating-point to fixed-point
FFLOAT	fixed-point to floating-point
FDBCNV	floating decimal to binary
FBDCNV	floating binary to decimal

The FFIX routine converts the floating-point value (B) in Registers 8, 9 to a fixed-point 16 bit integer in two's complement notation. The fixed-point result is left in Register 8. A floating-point form of the integer value is returned in Register 12, 13. An error return is made if the fixed-point integer will not fit in 16 bits.

The FFLOAT routine converts a fixed-point, two's complement integer in Register 8 to a floating-point quantity. The floating-point result is left in Registers 8, 9.

The FDBCNV routine converts a string of ASCII characters which define a decimal number to the corresponding floating-point value. The decimal number is located by loading Register 11 with a pointer to the first character in the ASCII string. The characters can be represented in either 7-bit or 8-bit ASCII code. The floating-point result is left in Registers 8, 9.

The format of the decimal numbers is that of the "E" format as used in FORTRAN. The decimal format consists of the following.

1. An optional leading plus or minus sign
2. Up to six digits that may include a decimal point
3. An optional E character followed by an optional plus or minus sign and one or two decimal digits, denoting a power of ten.

The decimal number is terminated by any of the following:

1. Any character not a digit, decimal point, E, plus sign, or minus sign.
2. The second letter E encountered.
3. Any plus sign or minus sign that is not either first, or immediately following the first E character.

Examples of numbers in decimal format are:

-27
3.426
.00097
45.
-36.2E-3
+.07E27
1.5E+6

If more than six digits are specified, the proper order of magnitude will result, but only six digits of precision are maintained. An error return is provided with the decimal to floating conversion. The error exit is taken when any of the following occur:

1. Multiple decimal points occur before the number is terminated, or an E is encountered.

2. Any decimal point occurs after an E is encountered, but before the number is terminated.

3. The specified power of ten is not in the range -99 to +99.

When an error return is taken, the pointer in Register 11 indicates the character which caused the error. On a normal return, the pointer indicates the character which terminates the number.

The FBDCNV routine converts a floating-point value in Registers 8, 9 to the corresponding decimal number as an ASCII character string. The destination for the ASCII characters is indicated by Register 11, which contains a pointer to the first character position in memory. The subroutine generates 7-bit ASCII characters in which the most significant bit is zero.

The format of the decimal number is similar to the "E" format as used in FORTRAN. The number of characters generated varies from 1 to 12. The shortest number is a simple integer, like 3, and the longest number is of the form -.123456E-12. The rules for generating the decimal number are as follows:

1. A minus sign is generated for a negative number, no sign for a positive number.

2. If the number is in the range .1 to 10^6, the number is represented without an E. Six or fewer digits are generated, with the decimal point between the appropriate digits. Trailing zeros and decimal point are suppressed.

3. For numbers outside the range .1 to 10^6, an E is used. The number is expressed in the range .1 to 1 with a multiplying power of ten. In this case, trailing zeros are also suppressed.

Some examples are:

15
-27.32
219000
.123037
-406.56
.127E-21
.231427E17
0

There is no error return from the FBDCNV routine. On return from the subroutine, Register 11 points to the character position immediately following the last character generated. Registers 8, 9 are not preserved.

The calling sequences for each subroutine are as follows. The EXTRN declarations should appear once at the beginning of the program.

```
          EXTRN     FFIX
*
*    PUT FLOATING VALUE B IN REGISTER 8, 9
*

          BAL       15,FFIX        FIX B
          B         ERROR          ERROR RETURN
          B         OK             NORMAL RETURN
*
*    2'S COMPLEMENT, FIXED-POINT INTEGER LEFT IN REGISTER 8
*    FLOATING-POINT INTEGER LEFT IN REGISTERS 12, 13
*    ALL OTHER REGISTERS UNCHANGED
*
*

          EXTRN     FFLOAT
*
*    PUT 2'S COMPLEMENT, FIXED-POINT INTEGER IN REGISTER 8

          BAL       15,FFLOAT      FLOAT B
*
*    NO ERROR RETURN
*    FLOATING-POINT RESULT LEFT IN REGISTERS 8, 9
*    ALL OTHER REGISTERS UNCHANGED
*
*
```

```
         EXTRN    FDBCNV
*
*   PUT POINTER TO ASCII CHAR STRING IN REGISTER 11
*

         BAL      15,FDBCNV     DECIMAL TO FLOATING
         B        ERROR         ERROR RETURN
         B        OK            NORMAL RETURN
*
*   FLOATING-POINT RESULT LEFT IN REGISTERS 8, 9
*   REGISTER 11 POINTS TO TERMINATING CHARACTER
*   ALL OTHER REGISTERS UNCHANGED
*
*

         EXTRN    FBDCNV
*
*   PUT POINTER TO CHAR BUFFER IN REGISTER 11
*   ALLOW SPACE FOR 12 ASCII CHARACTERS
*   PUT FLOATING VALUE IN REGISTER 8, 9
*

         BAL      15,FBDCNV     FLOATING TO DECIMAL
*
*   NO ERROR RETURN
*   REGISTER 11 POINTS TO LAST GENERATED CHAR +1
*   REGISTERS 8, 9 DESTROYED
*   ALL OTHER REGISTERS UNCHANGED
*
*
```

The execution times for the routines are:

Subroutine	30-01	30-02
FFIX	3.6 msec	.45 msec
FFLOAT	3.5 msec	.44 msec
FDBCNV	20-100-500 msec	2.5-12.-62. msec
FBDCNV	20-100-500 msec	2.5-12.-62. msec

The execution times for FDBCNV and FBDCNV are minimum - average - maximum figures. The variation depends on the number of digits involved, and the magnitude of the floating-point exponent.

5. OPERATION

The Floating Point Package, 07-020, is provided as a relocatable tape with part number 07-020M08. This tape includes the names FADD, FSUB, FMUL, FDIV, FFIX, FFLOAT FDBCNV, and FBDCNV, which are declared as ENTRY points. This tape therefore, must be loaded with the General Loader, 06-025M1(This tape is self-contained, and requires no additional routines or tapes.

The Floating Point Package requires $6B0_{16}$ or 1712_{10} bytes of core memory.

TABLE OF CONTENTS

GE-PAC MODEL 30-1

TEST PROGRAM DESCRIPTION AND OPERATING INSTRUCTIONS

1. INTRODUCTION

The function of the 30-01 Test Program is to determine whether the Processor is capable of executing all instructions properly. Each instruction is exercised and the result is compared to an expected result.

If no failures are detected, the program prints out "GE-PAC MODEL 30-1 IS A OK" at the conclusion of the tests and halts. Depress EXECUTE to begin the test again, starting at location X'80'. If a failure is encountered, the testing is halted and an attempt is made to print out "FAILURE" and the hexadecimal number of the test that failed.

The program is divided into twenty sections. Each section is designated by a hexadecimal number from zero to fourteen. Each section tests all formats within a given instruction type. For example, in the test of the OR instruction, both the RR and RX formats are tested.

When a failure occurs, the Processor is placed in the Wait state. Standard maintenance procedures can be used to isolate and remedy the source of the failure.

Note that the starting location of the test is X'80'. If it is desired, the test will perform a continuous loop if it is started at location X'80', but the I/O instructions will not be tested if started at this location.

TABLE 1. LOADER PROGRAM

Location	Numbers to Insert			Program		
ØØ5Ø	C82Ø	ØØ8Ø		LHI	2, X'8Ø'	START
ØØ54	C83Ø	ØØØ1		LHI	3, 1	INCRE
ØØ58	C84Ø	Ø6F7		LHI	4, X'6F7'	END
ØØ5C	D3AØ	ØØ78		LB	1Ø, BINDV	DEVNUM
ØØ6Ø	DEAØ	ØØ79		OC	1Ø, BINDV+1	COMD
ØØ64	9DAE			SSR	1Ø, 14	STATUS
ØØ66	Ø8EE			LHR	14, 14	
ØØ68	423Ø	ØØ64		BTC	3, X'64'	TEST
ØØ6C	DBA2	ØØØØ		RD	1Ø, Ø (2)	
ØØ7Ø	C12Ø	ØØ64		BXLE	2, X'64'	
ØØ74	43ØØ	ØØ8Ø		B	X'8Ø	
ØØ78	Ø294 (TTY)		BINDV	DC	X'Ø298'	TTY
ØØ78	Ø399 (HSPTR)			DC	X'Ø399'	HSPTR

A listing of the 30-01 Test Program is provided later in this publication.

2. OPERATING INSTRUCTIONS

1. Manually insert the loader program listed on Table 1 beginning at location X'50'.

2. Verify that the program was correctly inserted by examining each core location that was written.

3. On the Teletypewriter, place the LINE-OFF-LOCAL rotary switch in the LINE position. On the Model ASR 35, place the MODE selector in the KT position.

4. Place the Test Program paper tape in the reader, being careful to place the first character over the sensing fingers.

5. If the High Speed Paper Tape Reader (X'399') is used, remember that the first character must be placed over the photo diodes, also.

6. Initialize and Address the Processor to location X'50', the first address of the loader program.

7. Start the 30-01 Processor running by selecting the RUN mode and pushing the EXECUTE button. Then operate the START-STOP-FREE switch to start the paper tape advancing through the reader.

8. After the last character has been loaded, the program is executed.

9. If the Processor is functioning correctly, the program will print out "GE-PAC MODEL 30-1 IS A OK". If the Processor is equipped with multiply and divide, either the OK printout or the W, R, U printout occurs about 30 seconds from the time the program is executed. If the High Speed Arithmetic Option is not installed, the printout should occur almost immediately after the program is executed.

10. If the Processor is not functioning the program will attempt to print out "FAILURE" and the hexadecimal number of the failing test.

3. DESCRIPTION OF TESTS

3.1 Condition Code Bit Test
Section 0

The three forms of the LH instruction (LH, LHI and LHR) are executed to test setting and resetting condition code bits G and L. The Branch instructions verify the correct setting of the condition code bits. The test section number is placed into a memory location referred to as FAILNU. It is the contents of this location that is printed if a machine instruction fails.

3.2 Add Instruction Test
Section 1

Each of the five add instructions (AH, AHR, AHI, ACHR, and ACH) are exercised by this section. After each add instruction is executed, the condition code bits are tested using branch instructions. Add operations that generate overflow and carry are included. A final test involves an add with carry instruction using two registers. The condition code bits are set to reflect the number contained in both registers.

3.3 Subtract Instruction Test
Section 2

Each of the five subtraction instructions (SHR, SH, SHI, SCHR, and SCH) is exercised by this section. After each subtract instruction is executed, the condition code bits are tested using Branch instructions. A final test involves a subtract with carry instruction using two registers. The condition code bits are set to reflect the number contained in both registers.

3.4 Exclusive OR Instruction
Section 3

This section tests the operation of the three forms of the Exclusive OR Instruction (XH, XHR, and XHI). Each of the three forms of Exclusive OR Instruction is tested employing various constants. After each execution, the condition code bits reflect the result of the operation. The Branch instructions verify the setting of the condition code bits.

3.5 Logical AND Instruction Test
Section 4

This section tests the operation of the three forms of the Logical AND Instruction (NH, NHR, and NHI). Each of the three forms of Logical AND Instruction is tested employing various constants. After each execution, the condition code bits reflect the result of the operation. The Branch instructions verify the setting of the condition code bits.

3.6 Logical OR Instruction Test
Section 5

This section tests the operation of the three forms of the Logical OR Instruction (OR, OHR, and OHI). Each of the three forms of Logical OR Instructions is tested employing various constants. After each execution, the condition code bits reflect the result of the operations. The Branch instructions verify the setting of the condition code bits.

3.7 Branch Instruction Test
Section 6

This section tests the Branch instruction's ability to act as an unconditional Branch (BR) and to serve as a Non-Operation (NOPR) Instruction. A failure in the machine's interpretation of the unconditional Branch instruction leads to the error routine for all other test sections. The error subroutine prints out the message "FAILURE" and the section number of the test that fails. It then enters the Wait state.

3.8 Branch Instruction Test
Section 7

This section test the RR format of the Branch instruction's ability to act as an unconditional Branch. A failure in the machine's interpretation of the Branch instructions results in branching to the error subroutine.

3.9 Compare Logical Instruction Test
Section 8

This section tests the operation of the three forms of the Logical Compare instruction (CHL, CLRH, and CLHI). Each of the Compare instructions uses an address in the test program to compare against. The results of the comparisons are reflected in the setting of the condition code bits which the Branch instructions test.

3.10 Store Byte Instruction Test
Section 9

Both forms of the Store Byte instruction (STBR and STB) are tested. The destination location or registers as specified in the Store Byte instruction are tested to insure that after the execution of this instruction, bits 8-15 are unchanged. This section also checks that this instruction's execution did not affect the setting of the condition code bits.

3.11 Shift Instruction Test
Section 10

The four shift instructions: Shift Left and Right Arithmetic, and Shift Left and Right Logical, are examined. The propagation of ones to the right using the Arithmetic Shift, the shift into the carry bit from either end of a register, and a shift of 16 are checked. The condition code bits are tested after each shift operation using Branch instructions.

3.12 Load Byte Memory Instruction Test - Section 11

In this section, the RX form of the Load Byte (LB) instruction is tested. The instruction's action of zeroing bits 0 through 7 in the Destination register as specified by the instruction, is verified. In addition, the instruction's action of not affecting the condition code bits is also checked.

3.13 Load Byte Register Instruction Test - Section 12

In this section, the RR form of the Load Byte (LBR) instruction is tested. The test operates in a similar manner to that described above for the RX form of the Load Byte instruction.

3.14 Load Program Status Word Instruction - Section 13

The specialized action of the Load Program Status Word (LPSW) instruction is tested by this section. If the Processor fails to execute this instruction porperly, a transfer to the error subroutine is executed.

3.15 Branch and Link Instruction Test
Section 14

Both forms of the Branch and Link instruction (BAL and BALR) are tested. The loading of the designated link register with the correct link address is verified. In addition, the Processor's ability to branch to the specified location is also checked.

3.16 Branch on Index High, Low, or Equal Instruction Test - Section 15

The two Branch on Index Instructions, Branch on Index High (BXH) and Branch Low or Equal (BXLE) are tested. The three required registers are set to a value and the BXH and BXLE instructions executed. An improper execution of the BXH and BXLE leads to the error subroutine.

3.17 Index Instruction Test
Section 16

A Load instruction, indexed, is used to test indexing. The contents of the indexed value is then compared to a known value to verify that the indexing operation was properly completed.

3.18 Illegal Instruction Test
Section 17

In this test, an illegal instruction is executed. The Illegal Instruction New PSW is set to an address in this test section. After the illegal instruction is executed, the address in the Old Illegal Instruction PSW is tested to check if it contains the address of the illegal instruction.

3.19 Multiply and Divide Instruction Test - Section 18

As some 30-01's are not equipped with the High Speed Option (Multiply, Divide, Read Block or Write Block), the Illegal Instruction Interrupt New PSW is first set to test program address "TWENTY1". When a Multiply instruction is attempted, program control will branch to location "TWENTY1", thus by-passing all of Section 18 and 19.

The multiply and divide test consists of a loop in which that same numbers are multiplied together (squared). The product is then divided into the multiplier. The result of the division yields the original number. All integers from 1 to 65, 534, progressing one unit at a time, are multiplied and divided.

The second part tests the signs obtained from the multiplication and division of all the possible combinations of signed operands.

The third part tests the Divide Fault Interrupt. Two numbers are selected such that the answer of the division cannot be expressed in a 16 bit register. A verification is provided to check that both the Divide Fault Interrupt occurs and that the divisor remains unchanged.

3.20 Read Block/Write Block Instruction Test - Section 19

The Write Block instruction will cause a message type out on the teleprinter which says "DEPRESS KEYS W, R, U". The Read Block instruction accepts these inputs (W, R, U) and then compares the data output to the user input. If the user does not input correctly (W comma R comma U), the 30-01 test program will respond with the error message. It is important that five (5) characters be input or the Processor will hang in a Read Block loop.

3.21 Input/Output Instructions, Acknowledge Interrupt, and Test for False Sync Test - Section 20

Part one (1) of this test follows the same procedure as described in Section 19.

Part two (2) tests the Acknowledge Interrupt instruction without an interrupt pending. When this happens, R1 should contain zeros and R2 or A+(X2) should contain a four (0004) which is the False Sync bit.

APPENDIX 1 LISTING

```
                                  OPT   PASS2,PRINT,PUNCH,STOP
                       *
                       *
                       * Ø6-ØØ5RØ2 GE-PAC 30-01 TEST PROGRAM
                       *
ØØ8Ø                           ORG    X'8Ø'                BEGIN EXECUTION AT LOC 8Ø
                       *
                       * WITH A LITTLE LUCK THE POINTER WILL SET TO ZERO
                       *
                       * Ø  Ø  Ø  Ø  Ø  CONDITION CODE BIT TEST
                       *
                       *
ØØ8Ø   43ØØ                    B      ROUND                ENTER HERE FOR EXECUTING
       ØØ8C
ØØ84   42ØØ                    NOP    Ø                    RB & WR INSTRUCTIONS
       ØØØØ
ØØ88   43ØØ                    B      SKIPEN               ENTER HERE FOR LOOPING
       ØØA2
ØØ8C   DEAØ    ROUND           OC     R1Ø,WDATA            THIS SUBROUTINE IS
       Ø6B4
ØØ9Ø   9DAE    RND             SSR    R1Ø,R14              USED TO ISSUE A XOFF
ØØ92   428Ø                    BTC    8,RND                CHARACTER TO STOP TAPE REA
       ØØ9Ø
ØØ96   DAAØ                    WD     R1Ø,XOFF
       Ø68C
ØØ9A   C82Ø    ZERO            LHI    R2,X'FFFF'
       FFFF
ØØ9E   43ØØ                    B      ZEROA
       ØØA6
ØØA2   C82Ø    SKIPEN          LHI    R2,Ø
       ØØØØ
ØØA6   4Ø2Ø    ZEROA           STH    R2,SKIPTS
       Ø6FA
ØØAA   C8ØØ                    LHI    RO,ERRORA
       Ø18C
ØØAE   4ØØØ                    STH    RO,X'36'             STORE ERROR ADDR IN ILLPSW
       ØØ36
ØØB2   C82Ø                    LHI    R2,Ø                 CLEAR FAIL REGISTER PTR
       ØØØØ
ØØB6   4Ø2Ø                    STH    R2,X'34'
       ØØ34
ØØBA   4Ø2Ø                    STH    R2,FAILNU            STORE FAIL TEST NUMBER
       Ø6F8
ØØBE   483Ø                    LH     R3,FOXES             LOADS 'FFFF' INTO R3
       06A6
ØØC2   Ø31Ø                    BFCR   1,ERROR              TEST COND. CODE L=1
ØØC4   Ø2EØ                    BTCR   X'E',ERROR           TEST COND.CODE C, V, G=Ø
ØØC6   C81Ø                    LHI    R1,X'1'              PLACE 1 IN R1
       ØØØ1
ØØCA   Ø32Ø                    BFCR   2,ERROR              TEST COND. CODE G=1
ØØCC   Ø2DØ                    BTCR   X'D', ERROR          TEST COND. CODE C,V,L=Ø
ØØCE   Ø833                    LHR    R3,R3                USED TO SET COND. CODE
ØØDØ   Ø31Ø                    BFCR   1,ERROR              TEST COND. CODE L=1
ØØD2   Ø2EØ                    BTCR   X'E',ERROR           TEST COND. CODE C,V,G=Ø
```

```
ØØD4  C83Ø           LHI   R3,Ø           USED TO SET COND. CODE
      ØØØØ
ØØD8  Ø2FØ           BTCH  X'F',ERROR     TEST COND. CODE C.V.G,L=Ø
                *
                *
                *  ERROR POINTER NOW SET TO 1
                *
                *  1  1  1  1  1      ADD INSTRUCTION TEST
                *
                *
ØØDA  ØA21      ONE  AHR   R2,R1          SETS ERROR POINTER TO 1
ØØDC  4Ø2Ø           STH   R2,FAILNU      STORE FAIL TEST NUMBER
      Ø6F8
ØØEØ  Ø2DØ           BTCR  X'D',ERROR     TEST COND. CODE G=1
ØØE2  Ø32Ø           BFCR  2,ERROR        TEST COND. CODE C,V,L=Ø
ØØE4  CA1Ø           AHI   R1,X'7FFF'     CHANGES R1FROM '7FFF' TO
      7FFF
                *                         "8ØØØ'
ØØE8  Ø35Ø           BFCR  5,ERROR        TEST COND CODE L,V=1
ØØEA  Ø2AØ           BTCR  X'A',ERROR     TEST COND CODE C,G=Ø
ØØEC  4A1Ø           AH    R1,EIGTH       CHANGE R1 FROM '8ØØØ' TO
      Ø6A4
                *                         'ØØØØ' WITH CARRY
ØØFØ  Ø3CØ           BFCR  X'C',ERROR     TEST COND CODE V,C=1
ØØF2  Ø23Ø           BTCR  3,ERROR        TEST COND CODE L, G=0
ØØF4  ØE11           ACHR  R1,R1          CHANGE R1 FROM 'ØØØØ' TO
                *                         'ØØØ1'
ØØF6  Ø32Ø           BFCR  2,ERROR        TEST COND CODE G=1
ØØF8  Ø2DØ           BTCR  X'D',ERROR     TEST COND CODE C,V,L=Ø
ØØFA  4E3Ø           ACH   R3,NADA        TESTS COND CODE REFLECTS
      Ø6A8
                *                         ANSWER OF R3 AND R1
ØØFE  Ø32Ø           BFCR  2,ERROR        TEST COND CODE G=1
Ø1ØØ  Ø2DØ           BTCR  X'D',ERROR     TEST COND CODE C,V,L=Ø
Ø1Ø2  Ø833           LHR   R3,R3          TESTS R3 REMAINED ZERO
Ø1Ø4  Ø2FØ           BTCR  X'F',ERROR     TEST COND CODE C,V,G,L=Ø
                *
                *
                *  ERROR POINTER NOW SET TO 2
                *
                *  2  2  2  2  2   SUBTRACTION INSTRUCTION TEST
                *
                *
Ø1Ø6  ØA21      TWO  AHR   R2,R1          SETS ERROR POINTER TO 2
Ø1Ø8  4Ø2Ø           STH   R2,FAILNU      STORE FAIL TEST NUMBER
      Ø6F8
Ø1ØC  CB3Ø           SHI   R3,1           CHANGES R3 TO 'FFFF'
      ØØØ1
Ø11Ø  Ø39Ø           BFCR  9,ERROR        TEST COND CODE L,C=1
Ø112  Ø26Ø           BTCR  6,ERROR        TEST COND CODE V,C=Ø
Ø114  ØB33           SHR   R3,R3          CHANGES R3 TO Ø
Ø116  Ø2FØ           BTCR  X'F',ERROR     TEST COND CODE C,V,G,L=Ø
Ø118  4B3Ø           SH    R3,FOXES       CHANGES R3 TO 1
      Ø6A6
```

```
Ø11C  Ø3AØ              BFCR  X'A',ERROR        TEST COND CODEC,G=1
Ø11E  Ø25Ø              BTCR  5,ERROR           TEST COND CODE V,L=Ø
Ø12Ø  ØF33              SCHR  R3,R3             CHANGES R3 TO'FFFF'
Ø122  Ø39Ø              BFCR  9,ERROR           TEST COND CODE C,L=1
Ø124  Ø26Ø              BTCR  6,ERROR           TEST COND CODE V,G=Ø
Ø126  4F3Ø              SCH   R3,ALMFX          CHANGES R3 TO ZERO
      Ø6AØ
Ø12A  Ø2DØ              BTCR  X'D',ERROR        TEST COND CODE C,V,L=Ø
Ø12C  Ø32Ø              BFCR  2,ERROR           TEST COND CODE G REFLECT
                *                               SUBTRACT WITH CARRY
Ø12E  Ø833              LHR   R3,R3             R3 REMAINS ZERO
Ø13Ø  Ø23Ø              BTCR  3,ERROR           TEST COND CODE G,L=Ø
                *
                *
                *  ERROR POINTER NOW SET TO 3
                *
                *
                *  3  3  3  3  3   EXCLUSIVE OR INSTRUCTION
                *
Ø132  ØA21      THREE   AHR   R2,R1             SETS ERROR POINTER TO 3
Ø134  4Ø2Ø              STH   R2,FAILNU         STORE FAIL TEST NUMBER
      Ø6F8
Ø138  C73Ø              XHI   R3,X'FFFF'        CHANGES R3 to 'FFFF'
      FFFF
Ø13C  Ø31Ø              BFCR  1,ERROR           TEST COND CODE L=1
Ø13E  Ø2EØ              BTCR  X'E',ERROR        TEST COND CODE C,V.G=Ø
Ø14Ø  Ø733              XHR   R3,R3             CHANGES R3 TO ZERO
Ø142  Ø2FØ              BTCR  X'F',ERROR        TEST COND CODE C,V,G,L=Ø
Ø144  473Ø              XH    R3,ALT            CHANGES R3 TO '5A5A'
      Ø6A2
Ø148  Ø32Ø              BFCR  2,ERROR           TEST COND CODE G=1
                *
                *
                *  ERROR POINTER NOW SET TO 4
                *
                *  4  4  4  4    LOGICAL AND INSTRUCTION TEST
                *
                *
Ø14A  ØA21      FOUR    AHR   R2,R1             SETS ERROR POINTER TO FOUR
Ø14C  4Ø2Ø              STH   R2,FAILNU         STORE FAIL TEST NUMBER
      Ø6F8
Ø15Ø  C43Ø              NHI   R3,X'A5A5'        CHANGES R3 TO ZERO
      A5A5
Ø154  Ø2FØ              BTCR  X'F',ERROR        TEST COND CODE C,V,G,L=Ø
Ø156  C73Ø              XHI   R3,X'FFFF'        CHANGES R3 TO 'FFFF'
      FFFF
Ø15A  Ø433              NHR   R3,R3             R3 REMAINS WITH 'FFFF'
Ø15C  Ø31Ø              BFCR  1,ERROR           TEST COND CODE L=1
Ø15E  Ø2EØ              BTCR  X'E',ERROR        TEST COND CODE C,V,G=Ø
Ø16Ø  443Ø              NH    P3,NADA           CHANGES R3 TO ZERO
      Ø6A8
Ø164  Ø2FØ              BTCR  X'F',ERROR        TEST COND CODE C,V,G,L=Ø
                *
                *
```

```
                   *  ERROR POINTER NOW SET TO 5
                   *
                   *  5  5  5  5       LOGICAL OR INSTRUCTION TEST
                   *
                   *
Ø166  ØA21         FIVE      AHR   R2,R1               SETS ERROR POINTER TO 5
Ø168  4Ø2Ø                   STH   R2,FAILNU           STORE FAIL TEST NUMBER
      Ø6F8
Ø16C  Ø633                   OHR   R3,R3               R3 REMAINS ZERO
Ø16E  Ø2FO                   BTCR  X'F',ERROR          TEST COND CODE C,V,G,L=Ø
Ø17Ø  C63Ø                   OHI   R3,X'A5A5'          CHANGES R3 TO 'A5A5'
      A5A5
Ø174  Ø31Ø                   BFCR  1,ERROR             TEST COND CODE L=1
Ø176  Ø2EØ                   BTCR  X'E',ERROR          TEST COND CODE C,V,G=Ø
Ø178  463Ø                   OH    R3,ALT              CHANGES R3 TO 'FFFF"
      Ø6A2
Ø17C  Ø31Ø                   BFCR  1,ERROR             TEST COND CODE L=1
Ø17E  ØA31                   AHR   R3,R1               CHANGES R3 TO ZERO
Ø18Ø  Ø27Ø                   BTCR  X'7',ERROR          TEST COND CODE V,G,L=Ø
                   *
                   *
                   *  ERROR POINTER NOW SET TO 6
                   *
                   *  6  6  6  6    BRANCH INSTRUCTION TEST
                   *
                   *  LET US SEE IF THE BRANCH INSTRUCTIONS HAVE BEEN WORKING
                   *
                   *
Ø182  ØA21         SIX       AHR   R2,R1               SETS ERROR POINTER TO 6
Ø184  4Ø2Ø                   STH   R2,FAILNU           STORE FAIL TEST NUMBER
      Ø6F8
Ø188  43ØØ                   BFC   Ø,AROUND            TEST OF UNCONDITIONAL
      Ø23A
                   *                                   BRANCH INSTRUCTION
Ø18C  4Ø1Ø         ERRORA    STH   R1,SAVE
      Ø71C
Ø19Ø  4Ø3Ø                   STH   P3,SAVE+2
      Ø71E
Ø194  4Ø4Ø                   STH   R4,SAVE+4
      Ø72Ø
Ø198  4Ø5Ø                   STH   R5,SAVE+6


      Ø722
Ø19C  4Ø9Ø                   STH   R9,SAVE+8
      Ø724
Ø1AØ  4ØAØ                   STH   R10,SAVE+10
      Ø726
Ø1A4  4ØFØ                   STH   R15,SAVE+12
      Ø728
Ø1A8  ØB55                   SHR   R5,R5
Ø1AA  C83Ø         CONVRT    LHI   R3,4
      ØØØ4
Ø1AE  484Ø                   LH    R4,FAILNU
      Ø6F8
Ø1B2  CC43                   SRHL  R4,Ø(3)
```

```
       ØØØØ
ØlB6   C44Ø                 NHI    R4,X'F'
       ØØØF
ØlBA   C54Ø                 CLHI   R4,X'A'
       ØØØA
ØlBE   428Ø                 BL     *+8
       ØlC6
ØlC2   CA4Ø                 AHI    R4,7
       ØØØ7
ØlC6   CA4Ø                 AHI    R4,X'3Ø'
       ØØ3Ø
ØlCA   D245                 STB    R4,TESTNU(5)
       Ø6C8
ØlCE   CA5Ø                 AHI    R5,1
       ØØØ1
ØlD2   CB3Ø                 SHI    R3,4
       ØØØ4
ØlD6   431Ø                 BNM    CONVRT+4
       ØlAE
ØlDA   C8AØ                 LHI    R1Ø,2              LOAD TTY DEVICE NUM
       ØØØ2
ØlDE   C81Ø                 LHI    R1,1
       ØØØ1
ØlE2   C8FØ                 LHI    R15,MESS           START OF FAILURE MSG
       Ø6BE
ØlE6   DEAØ      SENS       OC     R1Ø,WDATA          TTY TO WRITE MODE
       Ø6B4
ØlEA   9DAE                 SSR    R1Ø,R14            TEST STATUS BYTE OF TTY
ØlEC   42FØ                 BTC    X'F',SENS          WHEN BUSY IS ZERO
       ØlE6
ØlFØ   DAAF                 WD     R1Ø,Ø(R15)         SEND CHARACTER TO TTY
       ØØØØ
ØlF4   ØAF1                 AHR    R15,R1             INCREMENT INDEX
ØlF6   C5FØ                 CLHI   R15,MESS1          TEST SENT LAST CHARACTER
       Ø6CA
ØlFA   428Ø                 BTC    8,SENS             RETURN TO SENSE STATUS
       ØlE6
ØlFE   489Ø                 LH     R9,SKIPTS          TEST START LOCATION
       Ø6FA
Ø2Ø2   433Ø                 BZ     RETRN3
       Ø2ØE
Ø2Ø6   C89Ø                 LHI    R9,ZERO
       ØØ9A
Ø2ØA   43ØØ                 B      RETRN4
       Ø212
Ø2ØE   C89Ø      RETRN3     LHI    R9,SKIPEN
       ØØA2
Ø212   4Ø9Ø      RETRN4     STH    R9,ERRWAT+2
       Ø238
Ø216   481Ø                 LH     R1,SAVE
       Ø71C
Ø21A   483Ø                 LH     R3,SAVE+2
       Ø71E
Ø21E   484Ø                 LH     R4,SAVE+4
```

```
        0720
0222    4850            LH      R5,SAVE+6
        0722
0226    4890            LH      R9,SAVE+8
        0724
022A    48A0            LH      R10,SAVE+10
        0726
022E    48F0            LH      R15,SAVE+12
        0728
0232    C200            LPSW    ERRWAT               THIS STOPS PROGRAM
        0236
0236    8000    ERRWAT  DC      X'8000',A(ZERO) SHOULD JAM IF IT DOES NOT
        009A
                *                                    STOP
023A    0200    AROUND  BTCR    0,ERROR              TEST OF A NOP
023C    0832            LHR     R3,R2                CHANGES R3 TO 6
023E    4220            BTC     2,BY1                TEST COND CODE G=1
        0244
0242    0300            BFCR    0,ERROR              THIS BRANCHES TO ERROR
0244    43D0    BY1     BFC     X'D', BY2            TEST COND CODE C,V,L=0
        024A
0248    0300            BFCR    0,ERROR              THIS BRANCHES TO ERROR
024A    02D0    BY2     BTCR    X'D',ERROR           TEST COND CODE C,V,L=0
                *
                *
                *  MUST HAVE DONE SOMETHING RIGHT
                *
                *  ERROR POINTER NOW SET TO 7
                *
                *  7  7  7  7   BRANCH INSTRUCTION TEST (RR FORM)
                *
                *
024C    0A21    SEVEN   AHR     R2,R1                SETS ERROR POINTER TO 7
024E    4020            STH     R2,FAILNU            STORE FAIL TEST NUMBER
        06F8
0252    C830            LHI     R3,THERE             LOAD ADDR 'THERE' INTO R3
        025A
0256    0303            BFCR    0,R3                 TESTS UNCONDITIONAL BRANCH
                *                                    TO THERE
0258    0300            BFCR    0,ERROR              THIS BRANCHES TO ERROR
025A    C830    THERE   LHI     R3,EIGHT             LOADS ADDR 'EIGHT'INTO
        0262
                *                                    R3
025E    0223            BTCR    2,R3                 TEST COND CODE G=1
0260    0300            BFCR    0,ERROR              THIS BRANCHES TO ERROR
                *
                *
                *  ERROR POINTER NOW SET TO 8
                *  8  8  8  8   COMPARE LOGICAL INSTRUCTION TEST
                *
0262    0A21    EIGHT   AHR     R2,R1                SETS ERROR POINTER TO 8
0264    4020            STH     R2,FAILNU            STORE FAIL TEST NUMBER
        06F8
0268    4530            CLH     R3,THERE+2           COMPARES CONTENTS OF R3
```

```
        Ø25C
                 *                                    WITH CONTENTS OF LOC
                 *                                    THERE +2, SHOULD BE EQUAL
Ø26C    Ø2FØ            BTCR    X'F',ERROR            TEST COND CODE C,V,G,L=Ø
Ø26E    C53Ø            CLHI    R3,EIGHT +1           COMPARES CONTENTS OF R3
        Ø263
                 *                                    WITH THE VALUE 'EIGHT +1'
                 *                                    R3 SHOULD BE LESS
Ø272    Ø38Ø            BFCR    8,ERROR               TEST COND CODE C=1
Ø274    C84Ø            LHI     R4,EIGHT-1            LOADS THE VALUE 'EIGHT-1'
        Ø261
                 *                                    INTO R4
Ø278    Ø534            CLHR    R3,R4                 COMPARES R3 WITH R4
Ø27A    Ø2DØ            BTCR    X'D',ERROR            TEST COND CODE C,V,L=Ø
                 *
                 *
                 *  ERROR POINTER NOW SET TO 9
                 *
                 *  9  9  9  9     STORE BYTE INSTRUCTION TEST
                 *
                 *
Ø27C    ØA21            AHR     R2,R1                 SETS ERROR POINTER TO 9
Ø27E    4Ø2Ø            STH     R2,FAILNU             STORE FAIL TEST NUMBER
        Ø6F8
Ø282    4Ø4Ø     NINE   STH     R4,TEMP               STORE CONTENTS R4 IN
        Ø6FC
                 *                                    MEMORY
Ø286    483Ø            LH      R3,TEMP               LOAD SAME LOC INTO R3
        Ø6FC
Ø28A    Ø543            CLHR    R4,R3                 TEST R4 CONTAINS SAME AS
                 *                                    R3
Ø28C    Ø2FØ            BTCR    X'F',ERROR            TEST COND CODE C,V,G,L=Ø
Ø28E    ØB33            SHR     R3,R3                 CHANGES R3 TO ZERO
Ø29Ø    4Ø3Ø            STH     R3,TEMP               STORES R3 IN MEMORY
        Ø6FC
Ø294    484Ø            LH      R4,TEMP               LOAD SAME LOC INTO R4
        Ø6FC
Ø298    Ø2FØ            BTCR    X'F',ERROR            TEST COND CODE C,V,G,L=Ø
Ø29A    C83Ø            LHI     R3,X'5AA5'            CHANGES R3 TO '5AA5'
        5AA5
Ø29E    D23Ø            STB     R3,TEMP               STORE BITS Ø-7 OF R3 INTO
        Ø6FC
                 *                                    MEMORY
Ø2A2    484Ø            LH      R4,TEMP               CHANGES R4 TO 'A5ØØ'
        Ø6FC
Ø2A6    Ø31Ø            BFCR    1,ERROR               TEST COND CODE L=1
Ø2A8    D24Ø            STB     R4,TEMP               STORE BITS Ø-7 OF R4 INTO
        Ø6FC
                 *                                    MEMORY NOW CONTAINS ZERO
Ø2AC    484Ø            LH      R4,TEMP               CHANGES R4 TO ZERO
        Ø6FC
Ø2BØ    Ø2FØ            BTCR    X'F',ERROR            TEST COND CODE C,V,G,L=Ø
Ø2B2    D23Ø            STB     R3,TEMP+1             STORE BITS Ø-7 OF R3
        Ø6FD
```

```
                    *                                     ('A5') INTO BITS 8-15 OF
                    *                                     MEMORY
                    *                                     ('A5') INTO BITS 8-15 OF
                    *                                     MEMORY
Ø2B6    484Ø                 LH      R4,TEMP              CHANGES R4 TO 'ØØA5'
        Ø6FC
Ø2BA    Ø2DØ                 BTCR    X'D',ERROR           TEST COND CODE  C,V,L=Ø
Ø2BC    Ø32Ø                 BFCR    2,ERROR              TEST COND CODE G=1
Ø2BE    ØB44                 SHR     R4,R4                CHANGES R4 TO ZERO
Ø2CØ    9243                 STBR    R4,R3                CHANGES R4 TO ØØA5
Ø2C2    Ø833                 LHR     R3,R3                USED TO SET COND CODE
Ø2C4    Ø32Ø                 BFCR    2,ERROR              TEST COND CODE G=1
Ø2C6    C73Ø                 XHI     R3,X'5A00'           CHANGES R3 TO 'ØØØØ'
        5AØØ
Ø2CA    Ø2FØ                 BTCR    X'F',ERROR           TEST COND CODE C,V,G,L=Ø
                    *
                    *
                    * ERROR POINTER NOW SET TO A
                    *
                    *  A  A  A  A    SHIFT INSTRUCTION TEST
                    *
                    *
Ø2CC    ØA21        TEN      AHR     R2,R1                SETS ERROR POINTER TO 'A'
Ø2CE    4Ø2Ø                 STH     R2,FAILNU            STORE FAIL TEST NUMBER
        Ø6F8
Ø2D2    ØA31                 AHR     R3,R1                CHANGES R3 TO 'ØØØ1'
Ø2D4    CD3Ø                 SLHL    R3,15                CHANGES R3 TO '8ØØØ'
        ØØØF
Ø2D8    Ø31Ø                 BFCR    1,ERROR              TEST COND CODE L=1
Ø2DA    Ø2EØ                 BTCR    X'E',ERROR           TEST COND CODE C,V,G=Ø
Ø2DC    CE3Ø                 SRHA    R3,15                CHANGES R3 TO 'FFFF'
        ØØØF
Ø2EØ    Ø31Ø                 BFCR    1,ERROR              TEST COND CODE L=1
Ø2E2    Ø2EØ                 BTCR    X'E',ERROR           TEST COND CODE C,V,G=Ø
Ø2E4    CF3Ø                 SLHA    R3,15                CHANGES R3 TO '8ØØØ'
        ØØØF
Ø2E8    Ø39Ø                 BFCR    9,ERROR              TEST COND CODE C,L=1
Ø2EA    Ø26Ø                 BTCR    6,ERROR              TEST COND CODE V,G=Ø
Ø2EC    CF3Ø                 SLHA    R3,1                 CHANGES R3 TO '8ØØØ'
        ØØØ1
Ø2FØ    Ø31Ø                 BFCR    1,ERROR              TEST COND CODE L=1
Ø2F2    Ø2EØ                 BTCR    X'E',ERROR           TEST COND CODE C,V,G=Ø
Ø2F4    CD3Ø                 SLHL    R3,0                 NO CHANGE TO R3 = '8ØØØ'
        ØØØØ
Ø2F8    Ø31Ø                 BFCR    1,ERROR              TEST COND CODE L=1
Ø2FA    Ø2EØ                 BTCR    X'E',ERROR           TEST COND CODE C,V,G=Ø
Ø2FC    CC3Ø                 SRHL    R3,15                CHANGES R3, TO 'ØØØ1'
        ØØØF
Ø3ØØ    Ø32Ø                 BFCR    2,ERROR              TEST COND CODE G=1
Ø3Ø2    Ø2DØ                 BTCR    X'D',ERROR           TEST COND CODE C,V,L=Ø
Ø3Ø4    CC3Ø                 SRHL    R3,1                 CHANGES R3 TO 'ØØØØ'
        ØØØ1
Ø3Ø8    Ø38Ø                 BFCR    8,ERROR              TEST COND CODE C=1
Ø3ØA    Ø27Ø                 BTCR    7,ERROR              TEST COND CODE V,G,L=Ø
```

```
Ø3ØC    ØA31                AHR     R3,R1            CHANGES R3 TO 'ØØØ1'
Ø3ØE    CE3Ø                SRHA    R3,1             CHANGES R3 TO 'ØØØØ'
        ØØØ1
Ø312    Ø38Ø                BFCR    8,ERROR          TEST COND CODE C=1
Ø314    Ø27Ø                BTCR    7,ERROR          TEST COND CODE V,G,L=Ø
Ø316    ØE33                ACHR    R3,R3            CHANGES Q3 TO 'ØØØ1'
Ø318    CF3Ø                SLHA    R3,14            CHANGES R3, TO '4ØØØ'
        ØØØE
Ø31C    Ø32Ø                BFCR    2,ERROR          TEST COND CODE G=1
Ø31E    Ø2DØ                BTCR    X'D',ERROR       TEST COND CODE C,V,L=Ø
Ø32Ø    CD3Ø                SLHL    R3,2             CHANGES R3,TO 'ØØØØ'
        ØØØ2
Ø324    Ø38Ø                BFCR    8,ERROR          TEST COND CODE C=1
Ø326    Ø27Ø                BTCR    7,ERROR          TEST COND CODE V,G,L=Ø
Ø328    ØE33                ACHR    R3,R3            CHANGES R3  TO 'ØØØ1'
Ø32A    CC3Ø                SRHL    R3,16            P3 REMAINS 'ØØØ1'
        ØØ1Ø
Ø32E    Ø32Ø                BFCR    2,ERROR          TEST COND CODE G=1
Ø33Ø    Ø2DØ                BTCR    X'D',ERROR       TEST COND CODE C,V,L=Ø
                    *
                    *
                    *  ERROR POINTER NOW SET TO B
                    *
                    *  B  B  B  B                    LOAD BYTE INSTRUCTION
                    *
                    *
Ø332    ØA21        ELEVEN  AHR     R2,R1            SETS ERROR POINTER TO 'B'
Ø334    4Ø2Ø                STH     R2,FAILNU        STORE FAIL TEST NUMBER
        Ø6F8
Ø338    D33Ø                LB      R3,ONES          CHANGES R3 TO 'ØØFF'
        Ø356
Ø33C    Ø2DØ                BTCR    X'D',ERROR       TEST COND CODE C,V,L=Ø
Ø33E    Ø32Ø                BFCR    2,ERROR          TEST COND CODE G=1
Ø34Ø    CD3Ø                SLHL    R3,8             CHANGES R3 TO 'FFØØ'
        ØØØ8
                    *                                NOTE-THIS CHANGES COND
                    *                                CODE
Ø344    D33Ø                LB      R3,ONES1+1       CHANGES R3 RO 'ØØFF'
        Ø359
                    *                                LB INSTRUCTION ZERO'S BITS
                    *                                Ø-8
Ø348    Ø2EØ                BTCR    X'E',ERROR       TEST COND CODE C,V,C=Ø
Ø34A    Ø31Ø                BFCR    1,ERROR          TEST COND CODE L=1
Ø34C    C53Ø                CLHI    R3,X'FF'         TEST R3 FOR HAVING 'ØØFF'
        ØØFF
Ø35Ø    Ø2FØ                BTCR    X'F',ERROR       TEST COND CODE C,V,G,L=Ø
Ø352    43ØØ                B       TWELV            GO TO NEXT TEST
        Ø35A
Ø356    FFØØ        ONES    DC      X'FFØØ'          USED IN ABOVE TEST
Ø358    ØØFF        ONES1   DC      X'ØØFF'          USED IN ABOVE TEST
                    *
                    *
                    *  ERROR POINTER NOW SET TO C
                    *
```

```
                *  C  C  C  C                   LOAD BYTE REG INSTRUCTION TESTS
                *
                *
Ø35A   ØA21     TWELV   AHR     R2,R1           SETS ERROR POINTER TO 'C'
Ø35C   4Ø2Ø             STH     R2,FAILNU       STORE FAIL TEST NUMBER
       Ø6F8
Ø36Ø   CD3Ø             SLHL    R3,8            CHANGES R3 TO 'FFØØ'
       ØØØ8
                *                               THIS INSTRUCTION SETS COND
                *                               CODE
Ø364   9333             LBR     R3,R3           CHANGES R3 TO  ØØØØ
Ø366   Ø31Ø             BFCR    1,ERROR         TESTS COND CODE L=1
Ø368   Ø833             LHR     3,3             USED TO SET COND CODE
Ø36A   Ø2FØ             BICR    X'F',ERROR      TEST COND CODE C,V,G,L=Ø
                *
                *  ERROR POINTER NOW SET TO D
                *
                *  D  D  D  D                   LOAD PSW INSTRUCTION S
                *
Ø36C   ØA21     THIRT   AHR     R2,R1           SETS ERROR POINTER TO 'D'
Ø36E   4Ø2Ø             STH     R2,FAILNU       STORE FAIL TEST NUMBER
       Ø6F8
Ø372   C2ØØ             LPSW    GO              PROG SHOULD BRANCH TO GO1
       Ø378
Ø376   Ø3ØØ             BFCR    Ø,ERROR         LAND HERE IF PREVIOUS
                *                               INSTRUCTION FALSE
Ø378   ØØØF     GO      DC      X'ØØØF'         USED FOR LPSW INSTRUCTION
Ø37A   Ø37C             DC      A(GO1)          USED FOR LPSW INSTRUCTION
Ø37C   42FØ     GO1     BTC     X'F',*+6        UNCONDITION BRANCH TO NEXT
       Ø382
                *                               TEST
Ø38Ø   Ø3ØØ             BFCR    Ø,ERROR         LAND HERE IF PREVIOUS
                *                               INSTRUCTION FAILS
                *
                *
                *  ERROR POINTER NOW SET TO E
                *
                *  E  E  E  E   BRANCH AND LINK INSTRUCTION TEST
                *                               INSTRUCTION TEST
                *
Ø382   ØA21     FOURT   AHR     R2,R1           SET ERROR POINTER TO 'E'
Ø384   4Ø2Ø             STH     R2,FAILNU       STORE FAIL TEST NUMBER
       Ø6F8
Ø388   C83Ø             LHI     R3,BRANCH       LOAD ADDR 'BRANCH'INTO R3
       Ø39Ø
Ø38C   Ø143             BALR    R4,R3           BRANCH TO ADDR IN R3
Ø38E   Ø3ØØ             BFCR    Ø,ERROR         LAND HERE ONLY ON ERROR
Ø39Ø   C54Ø     BRANCH CLHI     R4,BRANCH-2     TEST LINK ADDRESS IN Ø4
       Ø38E
                *                               SAME AS 'BRANCH -4'
Ø394   Ø2FØ             BTCR    X'F',ERROR      TEST COND CODE C,V,G,L=Ø
Ø396   414Ø             BAL     R4,BRAN2        BRANCH TO 'BRAN 2'
       Ø39C
```

```
Ø39A   Ø3ØØ            BFCR    Ø,ERROR          LAND HERE ONLY ON ERROR
Ø39C   C54Ø    BRAN2   CLHI    R4,BRAN2-2       TEST LINK ADDR IN R4
       Ø39A
               *                                SAME AS 'BRAN2-4'
Ø3AØ   Ø2FØ            BTCR    X'F',ERROR       TEST COND CODE BITS
               *                                C,V,G,L=Ø
               *
               *
               * ERROR POINTER NOW SET TO F
               *
               *  F  F  F  F                    BXLE&BXH INSTRUCTION T
               *
               *
Ø3A2   ØA21            AHR     R2,R1            SETS ERROR POINTER TO 'F'
Ø3A4   4Ø2Ø            STH     R2,FAILNU        STORE FAIL TEST NUMBER
       Ø6F8
Ø3A8   C83Ø            LHI     R3,2             CHANGE R3 TO 'ØØØ2'
       ØØØ2
Ø3AC   C84Ø            LHI     R4,X'FFFF'       INCREMENT FOR BXLE+BXH
       FFFF
               *                                INSTRUCTION
Ø3BØ   C85Ø            LHI     R5,Ø             END VALUE FOR BXLE+BXH
       ØØØØ
               *                                INSTRUCTION
Ø3B4   C13Ø            BXLE    R3,ERRORA        SHOULD NOT BRANCH, R3
       Ø18C
               *                                CHANGE TO '0001'
Ø3B8   CØ3Ø            BXH     R3,ERROR         CHANGE R3 TO 'ØØØØ'
       Ø18C
Ø3BC   ØA31            AHR     R3,R1            CHANGE R3 TO 'ØØØ1
Ø3BE   C13Ø            BXLE    R3,*+6           SHOULD BRANCH CHANGE R3
       Ø3C4
               *                                TO 'ØØØØ'
Ø362   Ø3ØØ            BFCR    Ø,ERROR          LAND HERE IN ERROR
Ø3C4   CØ3Ø            BXH     R3,*+6           SHOULD BRANCH CHANGE P3
       Ø3CA
               *                                TO 'FFFF'
Ø3C8   Ø3ØØ            BFCR    Ø,ERROR          LAND HERE IN ERROR
               *
               *
               *  ERROR POINTER SET TO 1Ø
               *
               * 1Ø  1Ø  1Ø  1Ø    INDEXING INSTRUCTION TEST
               *
               *          1Ø  1Ø  1Ø  1Ø        TEST INDEXING ON RS RX
               *
               *
Ø3CA   ØA21            AHR     R2,R1
Ø3CC   4Ø2Ø            STH     R2,FAILNU        STORE FAIL TEST NUMBER
       Ø6F8
Ø3DØ   C83Ø    LOC1    LHI     R3,LOC1          TEST RS WITHOUT INDEXING
       Ø3DØ
Ø3D4   C53Ø            CLHI    R3,LOC1          CHANGES R3 TO VALUE LOC1
       Ø3DØ
```

```
Ø3D8   Ø23Ø              BTCR   3,ERROR
Ø3DA   C831              LHI    R3,LOC1(R1)       TEST RS WITHOUT INDEXING
       Ø3DØ
Ø3DE   C53Ø              CLHI   R3,LOC1+1         CHANGES R3 TO VALUE LOC1+1
       Ø3D1
Ø3E2   Ø23Ø              BTCR   3,ERROR
Ø3E4   483Ø              LH     R3,DIAGN          TEST RX WITHOUT INDEXING
       Ø68E
Ø3E8   C53Ø              CLHI   R3,1              CHANGES R3 TO OCNTENTS OF
       ØØØ1
Ø3EC   Ø23Ø              BTCR   3,ERROR           LOC DIAGN
Ø3EE   4832              LH     R3,DIAGN(R2)      TEST RX WITH INDEXING
       Ø68E
Ø3F2   C53Ø              CLHI   R3,X'100'         CHANGES R3 TO CONTENTS OF
       Ø1ØØ
Ø3F6   Ø23Ø              BTCR   3,ERROR           OF LOC DIAGN+R2
                  *
                  *
                  *
                  *  ERROR POINTER NOW SET TO 11
                  *
                  *  11  11  11  11     ILLEGAL INSTRUCTION TEST
                  *
                  *  FOR TRAP,OTPSW SHOULD CONTAIN ADRS OF ILL
                  *
Ø3F8   ØA21       OK1    AHR    R2,R1             SETS ERROR POINTER TO '11'
Ø3FA   4Ø2Ø              STH    R2,FAILNU         STORE FAIL TEST NUMBER
       Ø6F8
Ø3FE   C83Ø              LHI    R3,Ø              SET CONDITION CODE PORTION
       ØØØØ
                  *                               OF
Ø4Ø2   4Ø3Ø              STH    R3,X'34'          ILLEGAL INSTR INTERRUPT
       ØØ34
                  *                               NEW PSW TO ZERO
Ø4Ø6   C83Ø              LHI    R3,NEXTH          LOADS ADDR INTO ADDR PART
       Ø41Ø
Ø4ØA   4Ø3Ø              STH    R3,X'36'          OF NEWPSW ILLEGAL
       ØØ36
                  *                               INSTRUCTION INTERRPUT
Ø4ØE   FØØØ       ILL    DC     X'F000'           THIS ILLEGAL INSTRUCTION
                  *                               CAUSES AN INTERRUPT
Ø41Ø   483Ø       NEXTH  LH     R3,X'32'
       ØØ32
Ø414   C53Ø              CLHI   R3,ILL            TEST ADDR OF ILLEGAL
       Ø4ØE
                  *                               INSTRUCTION PLACED IN OLD
                  *                               PSW ILLEGAL INTERRUPT
Ø418   433Ø              BFC    3,EIGHTN          ADDRESS IS CORRECT GO
       Ø41E
Ø41C   Ø3ØØ              BFCR   Ø,ERROR           ON TO NEXT TEST
Ø41E   C83Ø       EIGHTN LHI    R3,ERRORA
       Ø18C
Ø422   4Ø3Ø              STH    R3,X'36'          SET ADDR TO ERROR
       ØØ36
```

```
                    *
                    *
                    *
                    *  12  12  12  12     MULTIPLY & DIVIDE INSTRUCTION TEST
                    *
                    *  MULTIPLY & DIVIDE TEST:  MULTIPLIER AND DIVIDEND IN R6
                    *
                    *
                    *
                    *
Ø426    ØA21                 AHR     R2,R1              SETS ERROR POINTER TO 12
Ø428    4Ø2Ø                 STH     R2,FAILNU          STORE FAIL TEST NUM
        Ø6F8
Ø42C    C83Ø                 LHI     R3,TWNTY1          STORE ADDR OF NEXT
        Ø5A4
Ø43Ø    4Ø3Ø                 STH     R3,X'36'           TEST IN NEW PSW ILLEGAL
        ØØ36
                    *                                   INSTRUCTION
Ø434    ØB33                 SHR     R3,R3              BECAUSE ALL PROCESSORS
                    *                                   DO NOT
Ø436    4Ø3Ø                 STH     R3,X'34'           HAVE MULT AND DIVIDE
        ØØ34
                    *                                   INSTRUCTIONS
Ø43A    C86Ø        MUDVT    LHI     R6,1               LOADS MULTIPLIER
        ØØØ1
Ø43E    C89Ø                 LHI     R9,1               LOADS MULTIPLICAND
        ØØØ1
Ø442    ØC86        LOOP1    MHR     R8,R6              FORM X SQUARED
Ø444    ØD86                 DHR     R8,R6              DIVIDE PREVIOUS STEP
Ø446    Ø888                 LHR     R8,R8              CHECK FOR ZERO REMAINDER
Ø448    Ø2FØ                 BTCR    X'F',ERROR
Ø44A    Ø596                 CLHR    R9,R6              COMPAR DIVIDEND WITH
                    *                                   MULTIPLIER
Ø44C    433Ø                 BFC     3,OK               SHOULD BE EQUAL
        Ø452
Ø45Ø    Ø3ØØ                 BFCR    Ø,ERROR            NOT, GO TO ERROR
Ø452    CA6Ø        OK       AHI     R6,1               INCREMENT MULTIPLIER
        ØØØ1
Ø456    CA9Ø                 AHI     R9,1               INCREMENT
        ØØØ1
Ø45A    C56Ø                 CLHI    R6,X'FFFF'         TEST IF FINISHED
        FFFF
Ø45E    433Ø                 BFC     3,FINT1            IF SO, JUMP TO NEXT PART
        Ø466
Ø462    43ØØ                 BFC     Ø,LOOP1            NOT, CONTINUE THIS PART
        Ø442
                    *  THIS TEST MULTIPLIES AND DIVIDES POSITIVE
                    *  AND NEGATIVE NUMBERS AND CHECKS THE SIGNS
                    *  OF THE RESULTING VALUES.
Ø466    485Ø        FINT1    LH      R5,PLUS1
        Ø6AA
Ø46A    4ØØØ                 STH     PØ,X'36'           STORE ERROR ADDR IN ILLPSW
        ØØ36
Ø46E    4C4Ø                 MH      R4,PLUS1           MULT PLUS 1 TIMES PLUS 1
```

```
          Ø6AA
Ø472      455Ø          CLH    R5,PLUS1       R5 SHOULD HAVE PLUS 1
          Ø6AA
Ø476      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT QESPON
Ø478      Ø844          LHR    R4,R4          R4 SHOULD BE ZERO
Ø47A      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø47C      485Ø          LH     R5,MINUS1
          Ø6A6
Ø48Ø      4C4Ø          MH     R4,PLUS1       MULT PLUS 1 TIMES MINUS 1
          Ø6AA
Ø484      454Ø          CLH    R4,MINUS1
          Ø6A6
Ø488      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø48A      455Ø          CLH    R5,MINUS1
          Ø6A6
Ø48E      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø49Ø      485Ø          LH     R5,PLUS1
          Ø6AA
Ø494      4C4Ø          MH     R4,MINUS1      MULT MINUS1 TIMES PLUS1
          Ø6A6
Ø498      454Ø          CLH    R4,MINUS1      R4 SHOULD HAVE MINUS 1
          Ø6A6
Ø49C      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø49E      455Ø          CLH    R5,MINUS1      R5 SHOULD HAVE MINUS1
          Ø6A6
Ø4A2      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø4A4      485Ø          LH     R5,MINUS1
          Ø6Ø6
Ø4A8      4C4Ø          MH     R4,MINUS1      MULT MINUS1 TIMES MINUS1
          Ø6A6
Ø4AC      455Ø          CLH    R5,PLUS1       R5 SHOULD HAVE PLUS 1
          Ø6AA
Ø4BØ      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø4B2      Ø844          LHR    R4,R4          R4 SHOULD HAVE ZERO
Ø4B4      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø4B6      ØB44          SHR    R4,R4
Ø4B8      485Ø          LH     R5,PLUS3       R4+R5=+3
          Ø6BØ
Ø4BC      4D4Ø          DH     R4,PLUS2       DIVIDE POSITIVE INTO
          Ø6AC
                  *                           POSITIVE NUMBER
Ø4CØ      454Ø          CLH    R4,PLUS1       TEST FOR POSITIVE REMAINDE
          Ø6AA
Ø4C4      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø4C6      455Ø          CLH    R5,PLUS1       TEST FOR POSITIVE QUOTIENT
          Ø6AA
Ø4CA      Ø23Ø          BTCR   3,ERROR        TESTING FOR CORRECT RESPON
Ø4CC      484Ø          LH     R4,MINUS1
          Ø6A6
Ø4DØ      485Ø          LH     R5,MINUS3      R4+R5=-3
          Ø6B2
Ø4D4      4D4Ø          DH     R4,PLUS2       DIVIDE POSITIVE INTO
          Ø6AC
                  *                           NEGATIVE NUMBER
```

```
Ø4D8    454Ø           CLH     R4,MINUS1          TEST FOR NEGATIVE REMAINDE
        Ø6A6
Ø4DC    Ø23Ø           BTCR    3,ERROR            TESTING FOR CORRECT RESPON
Ø4DE    455Ø           CLH     R5,MINUS1          TEST FOR NEGATIVE QUOTIENT
        Ø6A6
Ø4E2    Ø23Ø           BTCR    3,ERROR            TESTING FOR CORRECT RESPON
Ø4E4    ØB44           SHR     R4,R4
Ø4E6    485Ø           LH      R5,PLUS3           R4+R5=+3
        Ø6BØ
Ø4EA    4D4Ø           DH      R4,MINUS2          DIVIDE NEGATIVE INTO
        Ø6AE
Ø4EE    454Ø           CLH     R4,PLUS1           TEST FOR POSITIVE REMAINDE
        Ø6AA
Ø4F2    Ø23Ø           BTCR    3,ERROR            TESTING FOR CORRECT RESPON
Ø4F4    455Ø           CLH     R5,MINUS1          TEST FOR NEGATIVE QUOTIENT
        Ø6A6
Ø4F8    Ø23Ø           BTCR    3,ERROR            TESTING FOR CORRECT RESPON
Ø4FA    484Ø           LH      R4,MINUS1
        Ø6A6
Ø4FE    485Ø           LH      R5,MINUS3          R4+R5=-3
        Ø6B2
Ø5Ø2    4D4Ø           DH      R4,MINUS2          DIVIDE NEGATIVE INTO
        Ø6AE
                *                                 NEGATIVE NUMBER
Ø5Ø6    454Ø           CLH     R4,MINUS1          TEST FOR NEGATIVE REMAINDE
        Ø6A6
Ø5ØA    Ø23Ø           BTCR    3,ERROR            TESTING FOR CORRECT RESPON
Ø5ØC    455Ø           CLH     R5,PLUS1           TEST FOR POSITIVE QUOTIENT
        Ø6AA
Ø51Ø    Ø23Ø           BTCR    3,ERROR            TESTING FOR CORRECT RESPON
                *  THIS SECTION TESTS DIVIDE FAULT
Ø512    C86Ø           LHI     R6,Ø               LOADS ZERO IN REG6
        ØØØØ
Ø516    ØB77           SHR     R7,R7
Ø518    4Ø6Ø           STH     R6,X'4C'           STORE VALUE IN CC PART OF
        ØØ4C
                *                                 NEW PSW DIVIDE FAULT
                *                                 INTERRUPT
Ø51C    C86Ø           LHI     R6,OVREC           STORE INTERRUPT ADDR IN
        Ø538
Ø52Ø    4Ø6Ø           STH     R6,X'4E'           NEW PSW DIVIDE FAULT
        ØØ4E
                *                                 INTERRUPT
Ø524    C2ØØ           LPSW    ENABLE             ALLOW DIVIDE FAULT
        Ø528
                *                                 INTERRUPT TO OCCUR
Ø528    1ØØØ    ENABLE DC      X'1000'            ENABLE FOR DIV
Ø52A    Ø52C           DC      A(HERE)
Ø52C    C86Ø    HERE   LHI     R6,X'2ØØØ'         LOADS DIVIDEND
        2ØØØ
Ø53Ø    C88Ø           LHI     R8,X'4ØØØ'         LOADS DIVISOR
        4ØØØ
Ø534    ØD68           DHR     R6,R8              DIVIDE SHOULD CAUSE
                *                                 INTERRUPT
```

```
Ø536   Ø3ØØ               BFCR    Ø,ERROR          NOT, GO TO ERROR
Ø538   C56Ø     OVREC     CLHI    R6,X'2ØØØ'       TEST OPERANDS HAVE NOT
       2ØØØ
                *                                  CHANGED
Ø53C   Ø2FØ               BTCR    X'F',ERROR       IF SO, GO TO ERROR
Ø53E   Ø877               LHR     R7,R7            TEST NO REMAINDER
                *                                  GENERATED
Ø54Ø   Ø2FØ               BTCR    X'F',ERROR
Ø542   ØD68               DHR     R6,R8            TEST DIVIDING BY ZERO DOES
                *                                  NOT CAUSE PROCESSOR TO
                *                                  LOOP HERE
                *
Ø544   4ØØØ               STH     RØ,X'4E'
       ØØ4E
                *
                *
                *  ERROR POINTER SET TO 13
                *
                *  13  13  13
                *
                *
Ø548   ØA21     NINTN     AHR     R2,R1
Ø54A   4Ø2Ø               STH     R2,FAILNU        STORE FAIL TEST NUMBER
       Ø6F8
                * READ BLOCK-WRITE BLOCK INSTRUCTION TEST
Ø54E   483Ø               LH      R3,SKIPTS
       Ø6FA
Ø552   433Ø               BZ      TWNTY
       Ø5A6
Ø556   C8CØ     WRTBLK    LHI     R12,2            LOADS TTY DEVICE NUMBER
       ØØØ2
Ø55A   DECØ               OC      R12,WDATA
       Ø6B4
Ø55E   D6CØ               WB      R12,FSTLOC       TEST WB INSTRUCTION
       Ø6B6
Ø562   Ø2FØ               BTCR    X'F',ERROR
Ø564   C84Ø               LHI     R4,DATA3
       Ø6EE
Ø568   C85Ø               LHI     P5,DATA4
       Ø6F7
Ø56C   96C4               WBR     R12,P4           TEST WBR INSTRUCTION
Ø56E   DECØ               OC      R12,RDATA
       Ø6B5
Ø572   D7CØ               RB      R12,THDLOC       TEST RB INSTRUCTION
       Ø6BA
Ø576   Ø2FØ               BTCR    X'F',ERROR
Ø578   C84Ø               LHI     R4,TEMP+3
       Ø6FF
Ø57C   C85Ø               LHI     R5,TEMP+4
       Ø7ØØ
Ø58Ø   97C4               RBR     R12,R4           TEST RBR INSTRUCTION
Ø582   Ø2FØ               BTCR    X'F',ERROR
Ø584   C84Ø               LHI     R4,X'7F'
       ØØ7F
```

```
Ø588  C85Ø            LHI   R5,-5              INITIALIZE LOOP COUNT
      FFFB
Ø58C  D365   TESTA    LB    R6,TEMP+5(R5)      LOADS  TTY CHAR IN Q6
      Ø7Ø1
Ø59Ø  D375            LB    R7,DATA3+1Ø(R5)    TEST CHARS RECEIVED
      Ø6F8
Ø594  Ø464            NHR   R6,R4              STRIP PARITY BIT
Ø596  Ø567            CLHR  R6,R7              MATCH CHARS IN TABLE
Ø598  Ø23Ø            BTCR  3,ERROR
Ø59A  ØA51            AHR   R5,R1              INCREMENT LOOP COUNT
Ø59C  423Ø            BNZ   TESTA
      Ø58C
Ø5AØ  43ØØ            B     TWNTY
      Ø5A6
             *
             *
             *ERROR POINTER NOW SET TO FOURTEEN
             *
             * INPUT/OUTPUT, ACKNOWLEDGE INTERRUPT, AND FALSE SYNC
             * 14    14     14      TEST WD,WDR,RD,RDR,SS,SSR
             *                        OC,OCR  INSTRUCTIONS
             *
Ø5A4  ØA21   TWNTY1   AHR   R2,R1              INCREMENT ERROR POINTER BY
Ø5A6  ØA21   TWNTY    AHR   R2,R1
Ø5A8  4Ø2Ø            STH   R2,FAILNU          STORE FAIL TEST NUMBER
      Ø6F8
Ø5AC  483Ø            LH    R3,SKIPTS
      Ø6FA
Ø5BØ  433Ø            BZ    END
      Ø65A
Ø5B4  C8CØ            LHI   R12,2              LOAD TTY DEVICE NUMBER
      ØØØ2
Ø5B8  DECØ            OC    R12,WDATA          TEST OC INSTRUCTION
      Ø6B4
Ø5BC  C83Ø            LHI   R3,DATA1
      Ø6E4
Ø5CØ  C84Ø            LHI   R4,1
      ØØØ1
Ø5C4  C85Ø            LHI   R5,DATA4
      Ø6F7
Ø5C8  9DC6   STAT4    SSR   R12,R6             TEST SSR INSTRUCTION
Ø5CA  42FØ            BTC   X'F',STAT4
      Ø5C8
Ø5CE  DAC3            WD    R12,Ø(R3)          TEST OF WD INSTRUCTION
      ØØØØ
Ø5D2  DDCØ   STAT5    SS    R12,TEMP           TEST SS INSTRUCTION
      Ø6FC
Ø5D6  42FØ            BTC   X'F',STAT5
      Ø5D2
Ø5DA  ØA31            AHR   R3,R1
Ø5DC  D363            LB    R6,Ø(R3)
      ØØØØ
Ø5EØ  9AC6            WDR   R12,R6             TEST WDR INSTRUCTION
Ø5E2  C13Ø            BXLE  R3,STAT4
```

```
         05C8
 05E6    D330              LB      R3,RDATA
         06B5
 05EA    9EC3              OCR     R12,R3           TEST OF OCR INSTRUCTION
 05EC    C830              LHI     P3,TEMP
         06FC
 05F0    9DC6      STAT6   SSR     R12,R6
 05F2    42F0              BTC     X'F',STAT6
         05F0
 05F6    DBC3              RD      R12,0(R3)        TEST RD INSTRUCTION
         0000
 05FA    C530              CLHI    R3,TEMP+4
         0700
 05FE    4380              BNL     CHART
         0616
 0602    0A31              AHR     R3,R1
 0604    9DC6      STAT7   SSR     R12,R6
 0606    42F0              BTC     X'F',STAT7
         0604
 060A    9BC6              RDQ     R12,R6           TEST RDR INSTRUCTION
 060C    D263              STB     R6,0(R3)
         0000
 0610    0A31              AHR     R3,R1
 0612    4300              B       STAT6
         05F0
 0616    C840      CHART   LHI     R4,X'7F'         MASK FOR PARITY
         007F
 061A    C850              LHI     R5,-5            INITIALIZE LOOP COUNT
         FFFB
 061E    D365      TESTB   LB      R6,TEMP+5(R5)    LOADS  TTY CHAR IN R6
         0701
 0622    D375              LB      R7,DATA3+10(R5)  TEST CHARS QECEIVED
         06F8
 0626    0464              NHR     R6,R4            STRIP PARITY BIT
 0628    0567              CLHR    R6,R7            MATCH CHARS IN TABLE
 062A    0230              BTCR    3,ERROR
 062C    0A51              AHR     R5,R1            INCREMENT LOOP COUNT
 062E    4230              BNZ     TESTB
         061E
                   * THIS IS A COMBINED TEST OF THE AIR AND AI
                   * INSTRUCTIONS AND A TEST OF THE FALSE SYNC
 0632    9F34              AIR     R3,R4            REMOVE PENDING INTERRUPTS
 0634    C830              LHI     R3,X'FFFF'       SET R3 BITS TO ALL ONES
         FFFF
 0638    C840              LHI     R4,X'FFFF'       SET R4 BITS TO ALL ONES
         FFFF
 063C    DF30              AI      R3,TEMP+1        THIS GENERATES FALSE SYNC
         06FD
 0640    D340              LB      R4,TEMP+1        BIT 13 SHOULD GET SET
         06FD
 0644    C540              CLHI    R4,4             TEST FOR THIS
         0004
 0648    0230              BTCR    3,ERROR
 064A    C830              LHI     R3,X'FFFF'       SET R3 BITS TO ALL ONES
```

```
        FFFF
 064E   C840            LHI    R4,X'FFFF'        SET R4 BITS TO ALL ONES
        FFFF
 0652   9F34            AIR    R3,R4             THIS GENERATES FALSE SYNC
 0654   C540            CLHI   R4,4              BIT 13 SHOULD GET SET
        0004
 0658   0230            BTCR   3,ERROR
                *
 065A   C830    END     LHI    R3,MESS1
        06CA
 065E   C840            LHI    R4,1
        0001
 0662   C850            LHI    R5,MESS2-1
        06E3
 0666   C8C0            LHI    R12,2             LOAD DEV NUN OF TTY
        0002
 066A   DEC0    STAT3   OC     R12,WDATA         TTY TO WRITE MODE
        06B4
 066E   9DCE            SSR    R12,R14           TEST STATUS BYTE OF TTY
 0670   42F0            BTC    X'F',STAT3
        066A
 0674   DAC3            WD     R12,0(3)          OUTPUT OK MESSAGE
        0000
 0678   C130            BXLE   R3,STAT3
        066A
 067C   4820            LH     R2,SKIPTS
        06FA
 0680   4330            BZ     SKIPEN
        00A2
 0684   C200            LPSW   OKWAIT
        0688
 0688   8000    OKWAIT  DC     X'8000',A(ZERO)
        009A
 068C   9393    XOFF    DC     X'9393'
 068E   0001    DIAGN   DC     X'1'              THIS TABLE IS USED TO
                *                                PROPAGTE
 0690   0002            DC     X'2'              A ONE THRU A FIELD OF ZERO
                *                                AND
 0692   0004            DC     X'4'              A ZERO THRU A FIELD OF ONE
 0694   0008            DC     X'8'
 0696   0010            DC     X'10'
 0698   0020            DC     X'20'
 069A   0040            DC     X'40'
 069C   0080            DC     X'80'
 069E   0100            DC     X'100'
 06A0   FFFE    ALMFX   DC     X'FFFE'
 06A2   5A5A    ALT     DC     X'5A5A'
 06A4   8000    EIGTH   DC     X'8000'
 06A6   FFFF    FOXES   DC     X'FFFF'
 06A8   0000    NADA    DC     0
 06AA   0001    PLUS1   DC     1
 06AC   0002    PLUS2   DC     2
 06AE   FFFE    MINUS2  DC     -2
 06B0   0003    PLUS3   DC     3
```

```
Ø6B2   FFFD    MINUS3   DC    -3
Ø6A6           MINUS1   EQU   FOXES
Ø6B4   98A4    WDATA    DC    X'98A4'
Ø6B5           RDATA    EQU   WDATA+1
Ø6B6   06E4    FSTLOC   DC    DATA1
Ø6B8   06ED    SECLOC   DC    DATA2
Ø6BA   Ø6FC    THDLOC   DC    TEMP
Ø6BC   Ø6FE    FORLOC   DC    TEMP+2
Ø6BE   8D8A    MESS     DC    X'8D8A',C'FAILURE'
       4641
       494C
       5552
       452Ø
Ø6C8           TESTNU   DS    2
Ø6CA   8D8A    MESS1    DC    X'8D8A'
Ø6CC   4745             DC    C'GE-PAC MOD     EL 30-1  IS AOK'
       2D50
       4143
       204D
       4F44
       454C
       2033
       302D
       3120
       4953
       2Ø41
       4F4B
Ø6E4           MESS2    EQU   *
Ø6E4   8D8A    DATA1    DC    X'8D8A',C'DEPRESS'
       4445
       5Ø52
       4553
       532Ø
Ø6ED           DATA2    EQU   *-1
Ø6EE   4B45    DATA3    DC    C'KEYS          W,R,U'
       5953
       2Ø57
       2C52
       2C55
Ø6F7           DATA4    EQU   *-1
Ø6F8           FAILNU   DS    2
Ø6FA           SKIPTS   DS    2
Ø6FC           TEMP     DS    32
Ø71C           SAVE     DS    32
ØØØØ           ERROR    EQU   Ø
ØØØØ           RØ       EQU   Ø
ØØØ1           R1       EQU   1
ØØØ2           R2       EQU   2
ØØØ3           R3       EQU   3
ØØØ4           R4       EQU   4
ØØØ5           R5       EQU   5
ØØØ6           R6       EQU   6
ØØØ7           R7       EQU   7
ØØØ8           R8       EQU   8
```

```
0009            R9       EQU    9
000A            R10      EQU    X'A'
000B            R11      EQU    X'B'
000C            R12      EQU    X'C'
000D            R13      EQU    X'D'
000E            R14      EQU    X'E'
000F            R15      EQU    X'F'
0044            NIPSW    EQU    X'44'
0030            OTRPSW   EQU    X'30'
073C                     END

ALMEX    06A0
ALT      06A2
AROUND   023A
BRAN2    039C
BRANCH   0390
BY1      0244
BY2      024A
CHART    0616
CONVRT   01AA
DATA1    06E4
DATA2    06ED
DATA3    06EE
DATA4    06F7
DIAGN    068E
EIGHT    0262
EIGHTN   041E
EIGTH    06A4
ELEVEN   0332
ENABLE   0528
END      065A
ERROR    0000
ERRORA   018C
ERRWAT   0236
FAILNU   06F8
FINT1    0466
FIVE     0166
FORLOC   06BC
FOUR     014A
FOURT    0382
FOXES    06A6
FSTLOC   06B6
GO       0378
GO1      037C
HERE     052C
ILL      040E
LOC1     03D0
LOOP1    0442
MESS     06BE
MESS1    06CA
MESS2    06E4
MINUS1   06A6
MINUS2   06AE
MINUS3   06B2
```

	MUDVT	Ø43A
U	N4	ØØØØ
	NADA	Ø6A8
	NEXTH	Ø41Ø
	NINE	Ø282
	NINTM	Ø548
	NIPSW	ØØ44
	OK	Ø452
	OKI	Ø3F8
	OKWAIT	Ø688
	ONE	ØØDA
	ONES	Ø356
	ONESI	Ø358
	OTRPSW	ØØ3Ø
	OVREC	Ø538
	PLUS1	Ø6AA
	PLUS2	Ø6AC
	PLUS3	Ø6BØ
	RØ	ØØØØ
	R1	ØØØ1
	R1Ø	ØØØA
	R11	ØØØB
	R12	ØØØC
	R13	ØØØD
	R14	ØØØE
	R15	ØØØF
	R2	ØØØ2
	R3	ØØØ3
	R4	ØØØ4
	R5	ØØØ5
	R6	ØØØ6
	R7	ØØØ7
	R8	ØØØ8
	R9	ØØØ9
	RDATA	Ø6B5
	RETRN3	Ø2ØE
	RETRN4	Ø212
	RND	ØØ9Ø
	ROUND	ØØ8C
	SAVE	Ø71C
	SECLOC	Ø6B8
	SENS	Ø1E6
	SEVEN	Ø24C
	SIX	Ø182
	SKIPEN	ØØA2
	SKIPTS	Ø6FA
	STAT3	Ø66A
	STAT4	Ø5C8
	STAT5	Ø5D2
	STAT6	Ø5FØ
	STAT7	Ø6Ø4
	TEMP	Ø6FC
	TEN	Ø2CC
	TESTA	Ø58C

TESTB	Ø61E
TESTNU	Ø6C8
THDLOC	Ø6BA
THERE	Ø25A
THIRT	Ø36C
THREE	Ø132
TWELV	Ø35A
TWNTY	Ø5A6
TWNTY1	Ø5A4
TWO	Ø1Ø6
WDATA	Ø6B4
WRTBLK	Ø556
XOFF	Ø68C
ZERO	ØØ9A
ZEROA	ØØA6

GE-PAC 30-2 TEST PROGRAM DESCRIPTION AND OPERATING INSTRUCTIONS

Publication Number 06-036R03A12

1. INTRODUCTION

The function of the 30-2 Test Program is to determine whether the Processor is capable of executing all instructions properly. Each instruction is exercised and the result is compared to an expected result.

If no failures are detected, the program print out "GE-PAC MODEL 30-2 IS A OK" at the conclusion of the tests. If a failure is encountered, the testing is halted and an attempt is made to print out "FAILURE" and the hexadecimal number of the test that failed.

The program is divided into twenty-three sections. Each section is designated by a hexadecimal number from zero to seventeen. Each section tests all formats within a given instruction type. For example, in the test of the OR instruction, both the RR and RX formats are tested.

When a failure occurs, the Processor is placed in the Wait state. Standard maintenance procedures can be used to isolate and remedy the source of the failure.

Note that the starting location of the test is X'80'. If it is desired, the test will perform a continuous loop if it is started at location X'88', but the I/O instructions will not be tested if started at this location.

TABLE 1. LOADER PROGRAM

Location	Numbers to Insert		Program			
ØØ5Ø	C82Ø	ØØ8Ø		LHI	2,X'80'	START
ØØ54	C83Ø	ØØØ1		LHI	3,1	INCRE
ØØ58	C84Ø	ØA6D		LHI	4,A6D	END
ØØ5C	D3AØ	ØØ78		LB	1Ø, BINDV	DEVNUM
ØØ6Ø	DEAØ	ØØ79		OC	1Ø, BINDV+1	COMD
ØØ64	9DAE			SSR	1Ø,14	STATUS
ØØ66	Ø8EE			LHR	14,14	
ØØ68	423Ø	ØØ64		BTC	3,X'64'	TEST
ØØ6C	DBA2	ØØØØ		RD	1Ø,Ø(2)	
ØØ7Ø	C12Ø	ØØ64		BXLE	2,X'64'	
ØØ74	43ØØ	ØØ8Ø		B	X'8Ø'	
ØØ78	Ø294 (TTY)		BINDV	DC	X'Ø294'	TTY
ØØ78	Ø399 (HSPTR)			DC	X'Ø399'	HSPTR

A listing of the 30-2 Test Program is provided later in this publication.

2. OPERATING INSTRUCTIONS

1. Manually insert the loader program listed on Table 1 beginning at location X'50'.

2. Verify that the program was correctly inserted by examining each core location that was written.

3. On the Teletypewriter, place the LINE-OFF-LOCAL rotary switch in the LINE position. On the Model ASR 35, place the MODE selector in the TTr position.

4. Place the Test Program paper tape in the reader, being careful to place the first character over the sensing fingers.

5. If the High Speed Paper Tape Reader (X'0399') is used, remember that the first character must be placed over the photo diodes, also.

6. Initialize and Address the Processor to location X'50', the first address of the loader program.

7. Start the 30-2 Processor running by selecting the RUN mode and pushing the EXECUTE button. Then operate the START-STOP-FREE switch to start the paper tape advancing through the reader.

8. After the last character has been loaded, the program is executed. The START-STOP-FREE lever switch on the paper tape reader should be operated to FREE and then returned to STOP to stop the tape.

9. If the Processor is functioning correctly, the program will print out "GE-PAC MODEL 30-2 IS A OK". If the Processor is equipped with multiply and divide, either the OK printout or the W, R, U printout occurs about 30 seconds from the time the program is executed. If the High Speed Arithmetic Option is not installed, the printout should occur almost immediately after the program is executed.

10. If the Processor is not functioning, the program will attempt to print out "FAILURE", and the hexadecimal number of the failing test.

3. DESCRIPTION OF TESTS

3.1 Condition Code Bit Test Section 0

The three forms of the LH instruction (LH, LHI, and LHR) are executed to test setting and resetting condition code bits G and L. The Branch instructions verify the correct setting of the condition code bits. The test section number is placed into a memory location referred to as FAILNU. It is the contents of this location that is printed if a machine instruction fails.

3.2 Add Instruction Test Section 1

Each of the five add instructions (AH, AHR, AHI, ACHR, and ACH) are exercised by this section. After each add instruction is executed, the condition code bits are tested using branch instructions. Add operations that generate overflow and carry are included. A final test involves an add with carry instruction using two registers. The condition code bits are set to reflect the number contained in both registers.

3.3 Subtract Instruction Test Section 2

Each of the five subtraction instructions (SHR, SH, SHI, SCHR, and SCH) is exercised by this section. After each subtract instruction is executed, the condition code bits are tested using Branch instructions. A final test involves a subtract with carry instruction using two registers. The condition code bits are set to reflect the number contained in both registers.

3.4 Exclusive OR Instruction Section 3

This section tests the operation of the three forms of the Exclusive OR Instruction (XH, XHR, and XHI). Each of the three forms of Exclusive OR Instruction is tested employing various constants. After each execution, the condition code bits reflect the result of the operation. The Branch instructions verify the setting of the condition code bits.

3.5 Logical AND Instruction Test Section 4

This section tests the operation of the three forms of the Logical AND Instruction (NH, NHR, and NHI). Each of the three forms of Logical AND Instruction is tested employing various constants. After each execution, the condition code bits reflect the result of the operation. The Branch instructions verify the setting of the condition code bits.

3.6 Logical OR Instruction Test Section 5

This section tests the operation of the three forms of the Logical OR Instruction (OR, OHR, and OHI). Each of the three forms of Logical OR Instructions is tested employing various constants. After each execution, the condition code bits reflect the result of the operation. The Branch instructions verify the setting of the condition code bits.

3.7 Branch Instruction Test Section 6

This section tests the Branch instruction's ability to act as an unconditional Branch (BR) and to serve as a Non-Operation (NOPR) Instruction. A failure in the machine's interpretation of the unconditional Branch instruction leads to the error routine which serves as the common error return for all other test sections. The error subroutine prints out the message "FAILURE" and the section number of the test that fails. It then enters the Wait state.

3.8 Branch Instruction Test Section 7

This section tests the RR format of the Branch instruction's ability to act as an unconditional Branch. A failure in the machine's interpretation of the Branch instruction results in branching to the error subroutine.

3.9 Compare Logical Instruction Test Section 8

This section tests the operation of the three forms of the Logical Compare instruction (CHL, CLHR, and CLHI). Each of the Compare instructions uses an address in the test program to compare against. The results of the comparisons are reflected in the setting of the condition code bits which the Branch instructions test.

3.10 Store Byte Instruction Test Section 9

Both forms of the Store Byte instruction (STBR and STB) are tested. The destination location or registers as specified in the Store Byte instruction are tested to insure that after the execution of this instruction, bits 8-15 are unchanged. This section also checks that this instruction's execution did not affect the setting of the condition code bits.

3.11 Shift Instruction Test Section 10

The four shift instructions: Shift Left and Right Arithmetic, and Shift Left and Right Logical, are examined. The propagation of ones to the right using the Arithmetic Shift, the shift into the carry bit from either end of a register, and a shift of 16 are checked. The condition code bits are tested after each shift operation using Branch instructions.

3.12 Load Byte Memory Instruction Test - Section 11

In this section, the RX form of the Load Byte (LB) instruction is tested. The instruction's action of zeroing bits 0 through 7 in the Destination register as specified by the instruction, is verified. In addition, the instruction's action of not affecting the condition code bits is also checked.

3.13 Load Byte Register Instruction Test - Section 12

In this section, the RR form of the Load Byte (LBR) instruction is tested. The test operates in a similar manner to that described above for the RX form of the Load Byte instruction.

3.14 Load Program Status Word Instruction - Section 13

The specialized action of the Load Program Status Word (LPSW) instruction is tested by this section. If the Processor fails to execute the instruction properly, a transfer to the error subroutine is executed.

3.15 Branch and Link Instruction Test Section 14

Both forms of the Branch and Link instruction (BAL and BALR) are tested. The loading of the designated link register with the correct link address is verified. In addition, the Processor's ability to branch to the specified location is also checked.

3.16 Branch on Index High, Low, or Equal Instruction Test - Section 15

The two Branch on Index Instructions, Branch on Index High (BXH) and Branch Low or Equal (BXLE), are tested. The three required registers are set to a value and the BXH and BXLE instructions executed. An improper execution of the BXH and BXLE leads to the error subroutine.

3.17 Index Instruction Test Section 16

A Load instruction, indexed, is used to test indexing. The contents of the indexed value is then compared to a known value to verify that the indexing operation was properly completed.

3.18 Illegal Instruction Test Section 17

In this test, an illegal instruction is executed. The illegal instruction New PSW is set to an address in this test section. After the illegal instruction is executed, the address in the old illegal instruction PSW is tested to check if it contains the address of the illegal instruction.

3.19 Multiply and Divide Instruction Test - Section 18

As some 30-02 's are not equipped with the High Speed Option (Multiply, Divide, Read Block or Write Block), the Illegal Instruction Interrupt New PSW is first set to test program address "TWENTY1". When a Multiply instruction is attempted, program control will branch to location "TWENTY1", thus by-passing all of Section 18 and 19.

The multiply and divide test consists of a loop in which the same numbers are multiplied together (squared). The product is then divided into the multiplier. The result of the division yields the original number. All integers from 1 to 65,534, progressing one unit at a time, are multiplied and divided.

The second part tests the signs obtained from the multiplication and division of all the possible combinations of signed operands.

The third part tests the divide fault interrupt. Two numbers are selected such that the answer of the division cannot be expressed in a 16 bit register. A verification is provided to check that both the divide fault interrupt occurs and that the divisor remains unchanged.

3.20 Read Block/Write Block Instruction Test - Section 19

The Write Block instruction will cause a message type out on the teleprinter which says "DEPRESS KEYS W, R, U". The Read Block instruction accepts these inputs (W, R, U) and then compares the data output to the user input. If the user does not input correctly (W comma R comma U), the 30-2 test program will respond with the error message. It is important that five (5) characters be input or the Processor will hang in a Read Block loop.

3.21 Input/Output Instructions, Acknowledge Interrupt, and Test for False Sync Test - Section 20

Part one (1) of this test follows the same procedure as described in Section 19.

Part two (2) tests the Acknowledge Interrupt instruction without an interrupt pending. When this happens, R1 should contain zeros and R2 or A+(X2) should contain a four (0004) which is the False Sync bit.

3.22 Load and Store Multiple Instruction Test - Section 21

In this test all registers (except R1 and R2) are loaded with their own number value. The Store Multiple instruction stores the content of all registers in successive memory locations starting at address TEMP. All the registers (except R1 and R2) are then zeroed and a Load Multiple instruction is executed. Each register is checked to determine if the Store and Load Multiple were performed correctly.

3.23 Source and Destination Register Test - Section 22

This checks to see that each register can be used as a Source register as well as a Destination register.

3.24 Register Bit Test Section 23

Every bit of every register is alternately set and reset. After successfully completing the register test, the program prints "GE-PAC MODEL 30-2 IS A OK". The test program returns to either location X'80' or X'88' depending upon where the program was started.

```
                                   OPT    PASS2,PRINT,PUNCH,STOP
                         *
                         *                                          APPENDIX 1
                         *06-036   30-2   TEST PROGRAM               LISTING
                         *
0080                               ORG    X'80'            BEGIN EXECUTION AT LOC 80
                         *
                         * WITH A LITTLE LUCK THE POINTER WILL SET TO ZERO
                         *
                         *  0  0  0  0  0   CONDITION CODE BIT TEST
                         *
                         *
0080   4300                        B      ROUND            ENTER HERE FOR EXECUTING
        008C
0084   4200                        NOP    0                RD & WB  INSTRUCTIONS
        0000
0088   4300                        B      SKIPEN           ENTER HERE FOR LOOPING
        00A2
008C   DEA0      ROUND             OC     R10,WDATA        THIS SUBROUTINE IS
        0A2A
0090   9DAE      RND               SSR    R10,R14          USED TO ISSUE A XOFF
0092   4280                        BTC    8,RND            CHARACTER TO STOP TAPE REA
        0090
0096   DAA0                        WD     R10,XOFF
        09D6
009A   C820      ZERO              LHI    R2,X'FFFF'       ENTER HERE FOR EXECUTING
        FFFF
009E   4300                        B      ZEROA            RB & WB INSTRUCTIONS
        00A6
00A2   C820      SKIPEN            LHI    R2,0             ENTER HERE FOR LOOPING
        0000
00A6   4020      ZEROA             STH    R2,SKIPTS        CONTINOUSLY WITHOUT STOPIN
        0A70
00AA   C800                        LHI    R0,ERRORA
        018C
00AE   4000                        STH    R0,X'36'         STORE ERROR ADDR IN ILLPSW
        0036
00B2   C820                        LHI    R2,0             CLEAR FAIL REGISTER PTR
        0000
00B6   4020                        STH    R2,FAILNU        STORE FAIL TEST NUMBER
        0A6E
00BA   4020                        STH    R2,X'22'         MODEL 4 REGISTER POINTER
        0022
00BE   4830                        LH     R3,FOXES         LOADS 'FFFF' INTO R3
        0A1C
00C2   0310                        BFCR   1,ERROR          TEST COND. CODE L=1
00C4   02E0                        BTCR   X'E',ERROR       TEST COND CODE C,V,G=0
00C6   C810                        LHI    R1,X'1'          PLACE 1 IN R1
        0001
00CA   0320                        BFCR   2,ERROR          TEST COND. CODE G=1
00CC   02D0                        BTCR   X'D',ERROR       TEST COND. CODE C,V,L=0
00CE   0833                        LHR    R3,R3            USED TO SET COND. CODE
00D0   0310                        BFCR   1,ERROR          TEST COND. CODE L=1
00D2   02F0                        BTCR   X'E',ERROR       TEST COND. CODE C,V,G=0
00D4   C830                        LHI    R3,0             USED TO SET COND. CODE
        0000
00D8   02F0                        BTCR   X'F',ERROR       TEST COND. CODE C,V,G,L=0
                         *
                         *
                         * ERROR POINTER NOW SET TO 1
                         *
                         * 1  1  1  1  1    ADD INSTRUCTION TEST
```

```
                  *
                  *
00DA   0A21    ONE      AHR    R2,R1            SETS ERROR POINTER TO 1
00DC   4020             STH    R2,FAILNU        STORE FAIL TEST NUMBER
        0A6E
00E0   02D0             BTCR   X'D',ERROR       TEST COND. CODE G=1
00E2   0320             BFCR   2,ERROR          TEST COND. CODE C,V,L=0
00E4   CA10             AHI    R1,X'7FFF'       CHANGES R1 FROM '7FFF' TO
        7FFF
                  *                             '8000'
00E8   0350             BFCR   5,ERROR          TEST COND CODE L,V=1
00EA   02A0             BTCR   X'A',ERROR       TEST COND CODE  C,G=0
00EC   4A10             AH     R1,EIGTH         CHANGE R1 FROM '8000' TO
        0A1A
                  *                             '0000' WITH CARRY
00F0   03C0             BFCR   X'C',ERROR       TEST COND CODE V,C=1
00F2   0230             BTCR   3,ERROR          TEST COND CODE L,G=0
00F4   0E11             ACHR   R1,R1            CHANGE R1 FROM '0C00' TO
                  *                             '0001'
00F6   0320             BFCR   2,ERROR          TEST COND CODE G=1
00F8   02D0             BTCR   X'D',ERROR       TEST COND CODE C,V,L=0
00FA   4E30             ACH    R3,NADA          TESTS COND CODE REFLECTS
        0A1E
                  *                             ANSWER OF R3 AND R1
00FE   0320             BFCR   2,ERROR          TEST COND CODE G=1
0100   02D0             BTCR   X'D',ERROR       TEST COND CODE C,V,L=0
0102   0833             LHR    R3,R3            TESTS R3 REMAINED ZERO
0104   02F0             BTCR   X'F',ERROR       TEST COND CODE C,V,G,L=0
                  *
                  *
                  *ERROR POINTER NOW SET TO 2
                  *
                  *  2  2  2  2  2   SUBTRACTION INSTRUCTION TEST
                  *
                  *
0106   0A21    TWO      AHR    R2,R1            SETS ERROR POINTER TO 2
0108   4020             STH    R2,FAILNU        STORE FAIL TEST NUMBER
        0A6E
010C   CB30             SHI    R3,1             CHANGES R3 TO 'FFFF'
        0001
0110   0390             BFCR   9,ERROR          TEST COND CODE L,C=1
0112   0260             BTCR   6,ERROR          TEST COND CODE V,G=0
0114   0B33             SHR    R3,R3            CHANGES R3 TO 0
0116   02F0             BTCR   X'F',ERROR       TEST COND CODE C,V,G,L=0
0118   4B30             SH     R3,FOXES         CHANGES R3 TO 1
        0A1C
011C   03A0             BFCR   X'A',ERROR       TEST COND CODEC,G=1
011E   0250             BTCR   5,ERROR          TEST COND CODE V,L=0
0120   0F33             SCHR   R3,R3            CHANGES R3 TO 'FFFF'
0122   0390             BFCR   9,ERROR          TEST COND CODE C,L=1
0124   0260             BTCR   6,ERROR          TEST COND CODE V,G=0
0126   4F30             SCH    R3,ALMFX         CHANGES R3 TO ZERO
        0A16
012A   02D0             BTCR   X'D',ERROR       TEST COND CODE C,V,L=0
012C   0320             BFCR   2,ERROR          TEST COND CODE G REFLECT
                  *                             SUBTRACT WITH CARRY
012E   0833             LHR    R3,R3            R3 REMAINS ZERO
0130   0230             BTCR   3,ERROR          TEST COND CODE G,L=0
                  *
                  *
                  *  ERROR POINTER NOW SET TO 3
                  *
```

```
                *
                *  3  3  3  3  3   EXCLUSIVE OR INSTRUCTION
                *
0132  0A21      THREE   AHR    R2,R1            SETS ERROR POINTER TO 3
0134  4020              STH    R2,FAILNU        STORE FAIL TEST NUMBER
       0A6E
0138  C730              XHI    R3,X'FFFF'       CHANGES R3 TO'FFFF'
       FFFF
013C  0310              BFCR   1,ERROR          TEST COND CODE L=1
013E  02E0              BTCR   X'E',ERROR       TEST COND CODE C,V,G=0
0140  0733              XHR    R3,R3            CHANGES R3 TO ZERO
0142  02F0              BTCR   X'F',ERROR       TEST COND CODE C,V,G,L=0
0144  4730              XH     R3,ALT           CHANGES R3 TO '5A5A'
       0A18
0148  0320              BFCR   2,ERROR          TEST COND CODE G=1
                *
                *
                *  ERROR POINTER NOW SET TO 4
                *
                *  4  4  4  4      LOGICAL AND INSTRUCTION TEST
                *
                *
014A  0A21      FOUR    AHR    R2,R1            SETS ERROR POINTER TO FOUR
014C  4020              STH    R2,FAILNU        STORE FAIL TEST NUMBER
       0A6E
0150  C430              NHI    R3,X'A5A5'       CHANGES R3 TO ZERO
       A5A5
0154  02F0              BTCR   X'F',ERROR       TEST COND CODE C,V,G,L=0
0156  C730              XHI    R3,X'FFFF'       CHANGES R3 TO 'FFFF'
       FFFF
015A  0433              NHR    R3,R3            R3 REMAINS WITH'FFFF'
015C  0310              BFCR   1,ERROR          TEST COND CODE L=1
015E  02E0              BTCR   X'E',ERROR       TEST COND CODE C,V,G=0
0160  4430              NH     R3,NADA          CHANGES R3 TO ZERO
       0A1E
0164  02F0              BTCR   X'F',ERROR       TEST COND CODE C,V,G,L=0
                *
                *
                *  ERROR POINTER NOW SET TO 5
                *
                *5  5  5  5      LOGICAL OR INSTRUCTION TEST
                *
                *
0166  0A21      FIVE    AHR    R2,R1            SETS ERROR POINTER TO 5
0168  4020              STH    R2,FAILNU        STORE FAIL TEST NUMBER
       0A6E
016C  0633              OHR    R3,R3            R3 REMAINS ZERO
016E  02F0              BTCR   X'F',ERROR       TEST COND CODE C,V,G,L=0
0170  C630              OHI    R3,X'A5A5'       CHANGES R3 TO 'A5A5'
       A5A5
0174  0310              BFCR   1,ERROR          TEST COND CODE L=1
0176  02E0              BTCR   X'E',ERROR       TEST COND CODE C,V,G=0
0178  4630              OH     R3,ALT           CHANGES R3 TO'FFFF'
       0A18
017C  0310              BFCR   1,ERROR          TEST COND CODE L=1
017E  0A31              AHR    R3,R1            CHANGES R3 TO ZERO
0180  0270              BTCR   X'7',ERROR       TEST COND CODE V,G,L=0
                *
                *
                *  ERROR POINTER NOW SET TO 6
                *
                *6  6  6  6      BRANCH INSTRUCTION TEST
```

```
                  *
                  * LET US SEE IF THE BRANCH INSTRUCTIONS HAVE BEEN WORKIN
                  *
                  *
0182  0A21        SIX      AHR   R2,R1             SETS ERROR POINTER TO 6
0184  4020                 STH   R2,FAILNU         STORE FAIL TEST NUMBER
       0A6E
0188  4300                 BFC   0,AROUND          TEST OF UNCONDITIONAL
       020A
                  *                                BRANCH INSTRUCTION
018C  D000        ERRORA   DC    X'D000',A(SAVE) MEANS STOR MULTIPLE
       0A92
0190  0855                 SHR   R5,R5
0192  C830        CONVRT   LHI   R3,4
       0004
0196  4840                 LH    R4,FAILNU
       0A6E
019A  CC43                 SRHL  R4,0(3)
       0000
019E  C440                 NHI   R4,X'F'
       000F
01A2  C540                 CLHI  R4,X'A'
       000A
01A6  4280                 BL    *+8
       01AE
01AA  CA40                 AHI   R4,7
       0007
01AE  CA40                 AHI   R4,X'30'
       0030
01B2  D245                 STB   R4,TESTNU(5)
       0A3E
01B6  CA50                 AHI   R5,1
       0001
01BA  CB30                 SHI   R3,4
       0004
01BE  4310                 BNM   CONVRT+4
       0196
01C2  C8A0                 LHI   R10,2             LOAD TTY DEVICE NUM
       0002
01C6  C810                 LHI   R1,1
       0001
01CA  C8F0                 LHI   R15,MESS          START OF FAILURE MSG
       0A34
01CE  DEA0        SENS     OC    R10,WDATA         TTY TO WRITE MODE
       0A2A
01D2  9DAE                 SSR   R10,R14           TEST STATUS BYTE OF TTY
01D4  42E0                 BTC   X'E',SENS         WHEN BUSY IS ZERO
       01CE
01D8  DAAF                 WD    R10,0(R15)        SEND CHARACTER TO TTY
       0000
01DC  0AF1                 AHR   R15,R1            INCREMENT INDEX
01DE  C5F0                 CLHI  R15,MESS1         TEST SENT LAST CHARACTER
       0A40
01E2  4280                 BTC   8,SENS            RETURN TO SENSE STATUS
       01CE
01E6  4890                 LH    R9,SKIPTS         TEST START LOCATION
       0A70
01EA  4330                 BZ    RETRN3
       01E6
01EE  C890                 LHI   R9,ZERO
       009A
01F2  4300                 B     RETRN4
```

```
        01FA
01F6   C890     RETRN3   LHI    R9,SKIPEN
        00A2
01FA   4090     RETRN4   STH    R9,ERRWAT+2
        0208
01FE   D100              DC     X'D100',A(SAVE) MEANS LOAD MULTIPLE
        0A92
0202   C200              LPSW   ERRWAT          THIS STOPS PROGRAM
        0206
0206   8000     ERRWAT   DC     X'8000',A(ZERO) SHOULD JAM IF IT DOES NOT
        009A
                *                               STOP
020A   0200     AROUND   BTCR   0,ERROR         TEST OF A NOP
020C   0832              LHR    R3,R2           CHANGES R3 TO 6
020E   4220              BTC    2,BY1           TEST COND CODE G=1
        0214
0212   0300              BFCR   0,ERROR         THIS BRANCHES TO ERROR
0214   43D0     BY1      BFC    X'D',BY2        TEST COND CODE C,V,L=0
        021A
0218   0300              BFCR   0,ERROR         THIS BRANCHES TO ERROR
021A   02D0     BY2      BTCR   X'D',ERROR      TEST COND CODE C,V,L=0
                *
                *
                *  MUST HAVE DONE SOMETHING RIGHT
                *
                *  ERROR POINTER NOW SET TO 7
                *
                *  7  7  7  7      BRANCH INSTRUCTION TEST (RR FORM)
                *
                *
021C   0A21     SEVEN    AHR    R2,R1           SETS ERROR POINTER TO 7
021E   4020              STH    R2,FAILNU       STORE FAIL TEST NUMBER
        0A6E
0222   C830              LHI    R3,THERE        LOAD ADDR 'THERE' INTO R3
        022A
0226   0303              BFCR   0,R3            TESTS UNCONDITIONAL BRANCH
                *                               TO THERE
0228   0300              BFCR   0,ERROR         THIS BRANCHES TO ERROR
022A   C830     THERE    LHI    R3,EIGHT        LOADS ADDR 'EIGHT' INTO
        0232
                *                               R3
022E   0223              BTCR   2,R3            TEST COND CODE G=1
0230   0300              BFCR   0,ERROR         THIS BRANCHES TO ERROR
                *
                *
                *  ERROR POINTER NOW SET TO 8
                *
                *  8  8  8  8      COMPARE LOGICAL INSTRUCTION TEST
                *
0232   0A21     EIGHT    AHR    R2,R1           SETS ERROR POINTER TO 8
0234   4020              STH    R2,FAILNU       STORE FAIL TEST NUMBER
        0A6E
0238   4530              CLH    R3,THERE+2      COMPARES CONTENTS OF R3
        022C
                *                               WITH CONTENTS OF LOC
                *                               THERE +2, SHOULD BE EQUAL
023C   02F0              BTCR   X'F',ERROR      TEST COND CODE C,V,G,L=0
023E   C530              CLHI   R3,EIGHT+1      COMPARES CONTENTS OF R3
        0233
                *                               WITH THE VALUE 'EIGHT +1'
                *                               R3 SHOULD BE LESS
0242   0380              BFCR   8,ERROR         TEST COND CODE C=1
```

```
0244  C840          LHI   R4,EIGHT-1     LOADS THE VALUE 'EIGHT-1'
       0231
                *                        INTO R4
0248  0534          CLHR  R3,R4          COMPARES R3 WITH R4
024A  02D0          BTCR  X'D',ERROR     TEST COND CODE C,V,L=0
                *
                *
                *  ERROR POINTER NOW SET TO 9
                *
                *  9  9  9  9      STORE BYTE INSTRUCTION TEST
                *
                *
024C  0A21          AHR   R2,R1          SETS ERROR POINTER TO 9
024E  4020          STH   R2,FAILNU      STORE FAIL TEST NUMBER
       0A6E
0252  4040    NINE  STH   R4,TEMP        STORE CONTENTS R4 IN
       0A72
                *                        MEMORY
0256  4830          LH    R3,TEMP        LOAD SAME LOC INTO R3
       0A72
025A  0543          CLHR  R4,R3          TEST R4 CONTAINS SAME AS
                *                        R3
025C  02F0          BTCR  X'F',ERROR     TEST COND CODE C,V,G,L=0
025E  0B33          SHR   R3,R3          CHANGES R3 TO ZERO
0260  4030          STH   R3,TEMP        STORES R3 IN MEMORY
       0A72
0264  4840          LH    R4,TEMP        LOAD SAME LOC INTO R4
       0A72
0268  02F0          BTCR  X'F',ERROR     TEST COND CODE C,V,G,L=0
026A  C830          LHI   R3,X'5AA5'     CHANGES R3 TO '5AA5'
       5AA5
026E  D230          STB   R3,TEMP        STORE BITS 0-7 OF R3 INTO
       0A72
                *                        MEMORY
0272  4840          LH    R4,TEMP        CHANGES R4 TO 'A500'
       0A72
0276  0310          BFCR  1,ERROR        TEST COND CODE L=1
0278  D240          STB   R4,TEMP        STORE BITS 0-7 OF R4 INTO
       0A72
                *                        MEMORY NOW CONTAINS ZERO
027C  4840          LH    R4,TEMP        CHANGES R4 TO ZERO
       0A72
0280  02F0          BTCR  X'F',ERROR     TEST COND CODE C,V,G,L=0
0282  D230          STB   R3,TEMP+1      STORE BITS 0-7 OF R3
       0A73
                *                        ('A5') INTO BITS 8-15 OF
                *                        MEMORY
                *                        ('A5') INTO BITS 8-15 OF
                *                        MEMORY
0286  4840          LH    R4,TEMP        CHANGES R4 TO '00A5'
       0A72
028A  02D0          BTCR  X'D',ERROR     TEST COND CODE C,V,L=0
028C  0320          BFCR  2,ERROR        TEST COND CODE G=1
028E  0B44          SHR   R4,R4          CHANGES R4 TO ZERO
0290  9243          STBR  R4,R3          CHANGES R4 TO  00A5
0292  0833          LHR   R3,R3          USED TO SET COND CODE
0294  0320          BFCR  2,ERROR        TEST COND CODE G=1
0296  C730          XHI   R3,X'5A00'     CHANGES R3 TO '0000'
       5A00
029A  02F0          BTCR  X'F',ERROR     TEST COND CODE C,V,G,L=0
                *
                *
```

```
                    *ERROR POINTER NOW SET TO A
                    *
                    *  A  A  A  A      SHIFT INSTRUCTION TEST
                    *
                    *
029C   0A21   TEN      AHR    R2,R1         SETS ERROR POINTER TO 'A'
029E   4020            STH    R2,FAILNU     STORE FAIL TEST NUMBER
        0A6F
02A2   0A31            AHR    R3,R1         CHANGES R3 TO '0001'
02A4   CD30            SLHL   R3,15         CHANGES R3 TO '8000'
        000F
02A8   0310            BFCR   1,ERROR       TEST COND CODE L=1
02AA   02E0            BTCR   X'E',ERROR    TEST COND CODE C,V,G=0
02AC   CE30            SRHA   R3,15         CHANGES R3 TO 'FFFF'
        000F
02B0   0310            BFCR   1,ERROR       TEST COND CODE L=1
02B2   02E0            BTCR   X'E',ERROR    TEST COND CODE C,V,G=0
02B4   CF30            SLHA   R3,15         CHANGES R3 TO '8000'
       000F
02B8   0390            BFCR   9,ERROR       TEST COND CODE C,L=1
02BA   0260            BTCR   6,ERROR       TEST COND CODE V,G=0
02BC   CF30            SLHA   R3,1          CHANGES R3 TO '8000'
        0001
02C0   0310            BFCR   1,ERROR       TEST COND CODE L=1
02C2   02E0            BTCR   X'E',ERROR    TEST COND CODE C,V,G=0
02C4   CD30            SLHL   R3,0          NO CHANGE TO R3 ='8000'
        0000
02C8   0310            BFCR   1,ERROR       TEST COND CODE  L=1
02CA   02E0            BTCR   X'E',ERROR    TEST COND CODE  C,V,G=0
02CC   CC30            SRHL   R3,15         CHANGES R3 TO '0001'
        000F
02D0   0320            BFCR   2,ERROR       TEST COND CODE G=1
02D2   02D0            BTCR   X'D',ERROR    TEST COND CODE C,V,L=0
02D4   CC30            SRHL   R3,1          CHANGES R3 TO '0000'
        0001
02D8   0380            BFCR   8,ERROR       TEST COND CODE C=1
02DA   0270            BTCR   7,ERROR       TEST COND CODE V,G,L=0
02DC   0A31            AHR    R3,R1         CHANGES R3 TO '0001'
02DE   CE30            SRHA   R3,1          CHANGES R3 TO '0000'
        0001
02E2   0380            BFCR   8,ERROR       TEST COND CODE C=1
02E4   0270            BTCR   7,ERROR       TEST COND CODE V,G,L=0
02E6   0E33            ACHR   R3,R3         CHANGES R3 TO '0001'
02E8   CF30            SLHA   R3,14         CHANGES R3 TO '4000'
        000E
02EC   0320            BFCR   2,ERROR       TEST COND CODE G=1
02EE   02D0            BTCR   X'D',ERROR    TEST COND CODE C,V,L=0
02F0   CD30            SLHL   R3,2          CHANGES R3 TO '0000'
        0002
02F4   0380            BFCR   8,ERROR       TEST COND CODE C=1
02F6   0270            BTCR   7,ERROR       TEST COND CODE V,G,L=0
02F8   0E33            ACHR   R3,R3         CHANGES R3 TO '0001'
02FA   CC30            SRHL   R3,16         R3 REMAINS '0001'
        0010
02FE   0320            BFCR   2,ERROR       TEST COND CODE G=1
0300   02D0            BTCR   X'D',ERROR    TEST COND CODE C,V,L=0
                    *
                    *
                    *ERROR POINTER NOW SET TO B
                    *
                    *  B  B  B  B              LOAD BYTE INSTRUCTION
                    *
```

```
                *
0302  0A21     ELEVEN   AHR    R2,R1           SETS ERROR POINTER TO 'B'
0304  4020              STH    R2,FAILNU       STORE FAIL TEST NUMBER
      0A6E
0308  D330              LB     R3,ONES         CHANGES R3 TO '00FF'
      0326
030C  02D0              BTCR   X'D',ERROR      TEST COND CODE C,V,L=0
030E  0320              BFCR   2,ERROR         TEST COND CODE G=1
0310  CD30              SLHL   R3,8            CHANGES R3 TO 'FF00'
      0008
                *                              NOTE-THIS CHANGES COND
                *                              CODE
0314  D330              LB     R3,ONES1+1      CHANGES R3 TO '00FF'
      0329
                *                              LB INSTRUCTION ZERO'S BITS
                *                              0-8
0318  02E0              BTCR   X'E',ERROR      TEST COND CODE C,V,G=0
031A  0310              BFCR   1,ERROR         TEST COND CODE L=1
031C  C530              CLHI   R3,X'FF'        TEST R3 FOR HAVING '00FF'
      00FF
0320  02F0              BTCR   X'F',ERROR      TEST COND CODE C,V,G,L=0
0322  4300              B      TWELV           GO TO NEXT TEST
      032A
0326  FF00     ONES     DC     X'FF00'         USED IN ABOVE TEST
0328  00FF     ONES1    DC     X'00FF'         USED IN ABOVE TEST
                *
                *
                *ERROR POINTER NOW SET TO C
                *
                *C C C C            LOAD BYTE REG INSTRUCTION TESTS
                *
                *
032A  0A21     TWELV    AHR    R2,R1           SETS ERROR POINTER TO 'C'
032C  4020              STH    R2,FAILNU       STORE FAIL TEST NUMBER
      0A6E
0330  CD30              SLHL   R3,8            CHANGES R3 TO 'FF00'
      0008
                *                              THIS INSTRUCTION SETS COND
                *                              CODE
0334  9333              LBR    R3,R3           CHANGES R3 TO  0000
0336  0310              BFCR   1,ERROR         TESTS COND CODE L=1
0338  0833              LHR    3,3             USED TO SET COND CODE
033A  02F0              BTCR   X'F',ERROR      TEST COND CODE C,V,G,L=0
                *
                *
                *ERROR POINTER NOW SET TO D
                *D D D D                       LOAD PSW INSTRUCTION S
                *
                *                              NEGATIVE NUMBER
033C  0A21     THIRT    AHR    R2,R1           SETS ERROR POINTER TO 'D'
033E  4020              STH    R2,FAILNU       STORE FAIL TEST NUMBER
      0A6E
0342  C200              LPSW   GO              PROG SHOULD BRANCH TO GO1
      0348
0346  0300              BFCR   0,ERROR         LAND HERE IF PREVIOUS
                *                              INSTRUCTION FALSE
0348  000F     GO       DC     X'000F'         USED FOR LPSW INSTRUCTION
034A  034C              DC     A(GO1)          USED FOR LPSW INSTRUCTION
034C  42F0     GO1      BTC    X'F',*+6        UNCONDITION BRANCH TO NEXT
      0352
                *                              TEST
0350  0300              BFCR   0,ERROR         LAND HERE IF PREVIOUS
```

```
                      *                                       INSTRUCTION FAILS
                      *
                      *
                      *ERROR POINTER NOW SET TO E
                      *
                      * E E E E         BRANCH AND LINK INSTRUCTION TEST
                      *                                       INSTRUCTION TEST
                      *
0352   0A21           FOURT    AHR     R2,R1                  SET ERROR POINTER TO 'E'
0354   4020                    STH     R2,FAILNU              STORE FAIL TEST NUMBER
       0A6E
0358   C830                    LHI     R3,BRANCH              LOAD ADDR 'BRANCH' INTO R3
       0360
035C   0143                    BALR    R4,R3                  BRANCH TO ADDR IN R3
035E   0300                    BFCR    0,ERROR                LAND HERE ONLY ON ERROR
0360   C540           BRANCH   CLHI    R4,BRANCH-2            TEST LINK ADDRESS IN R4
       035E
                      *                                       SAME AS 'BRANCH -4'
0364   02F0                    BTCR    X'F',ERROR             TEST COND CODE C,V,G,L=0
0366   4140                    BAL     R4,BRAN2               BRANCH TO 'BRAN 2'
       036C
036A   0300                    BFCR    0,ERROR                LAND HERE ONLY ON ERROR
036C   C540           BRAN2    CLHI    R4,BRAN2-2             TEST LINK ADDR IN R4
       036A
                      *                                       SAME AS 'BRAN2-4'
0370   02F0                    BTCR    X'F',ERROR             TEST COND CODE BITS
                      *                                       C,V,G,L=0
                      *
                      *
                      *ERROR POINTER NOW SET TO F
                      *
                      * F F F F                                     BXLE&BXH INSTRUCTION T
                      *
                      *
0372   0A21                    AHR     R2,R1                  SETS ERROR POINTER TO 'F'
0374   4020                    STH     R2,FAILNU              STORE FAIL TEST NUMBER
       0A6F
0378   C830                    LHI     R3,2                   CHANGE R3 TO '0002'
       0002
037C   C840                    LHI     R4,X'FFFF'             INCREMENT FOR BXLE+BXH
       FFFF
                      *                                       INSTRUCTION
0380   C850                    LHI     R5,0                   END VALUE FOR BXLE+BXH
       0000
                      *                                       INSTRUCTION
0384   C130                    BXLE    R3,ERRORA              SHOULD NOT BRANCH, R3
       018C
                      *                                       CHANGE TO '0001'
0388   C030                    BXH     R3,ERRORA              CHANGE R3 TO '0000'
       018C
038C   0A31                    AHR     R3,R1                  CHANGE R3 TO '0001'
038E   C130                    BXLE    R3,*+6                 SHOULD BRANCH CHANGE R3
       0394
                      *                                       TO '0000'
0392   0300                    BFCR    0,ERROR                LAND HERE IN ERROR
0394   C030                    BXH     R3,*+6                 SHOULD BRANCH CHANGE R3
       039A
                      *                                       TO 'FFFF'
0398   0300                    BFCR    0,ERROR                LAND HERE IN ERROR
                      *
                      *
                      *ERROR POINTER SET TO 10
```

```
               *
               *10 10 10 10        INDEXING INSTRUCTION TEST
               *
               *      *10  1C  10  10         TEST INDEXING ON RS RX
               *
               *
039A  0A21              AHR    R2,R1
039C  4020              STH    R2,FAILNU        STORE FAIL TEST NUMBER
       0A6E
03A0  C830     LOC1     LHI    R3,LOC1          TEST RS WITHOUT INDEXING
       03A0
03A4  C530              CLHI   R3,LOC1          CHANGES R3 TO VALUE LOC1
       03A0
03A8  0230              BTCR   3,ERROR
03AA  C831              LHI    R3,LOC1(R1)      TEST RS WITHOUT INDEXING
       03A0
03AE  C530              CLHI   R3,LOC1+1        CHANGES R3 TO VALUE LOC1+1
       03A1
03B2  0230              BTCR   3,ERROR
03B4  4830              LH     R3,DIAGN         TEST RX WITHOUT INDEXING
       09D4
03B8  C530              CLHI   R3,1             CHANGES R3 TO  OCNTENTS OF
       0001
03BC  0230              BTCR   3,ERROR          LOC DIAGN
03BE  4832              LH     R3,DIAGN(R2)     TEST RX WITH INDEXING
       09D4
03C2  C530              CLHI   R3,X'0080'       CHANGES R3 TO CONTENTS OF
       0080
03C6  0230              BTCR   3,ERROR          OF LOC DIAGN+R2
               *
               *
               *
               * ERROR POINTER NOW SET TO 11
               *
               *11 11 11 11        ILLEGAL INSTRUCTION TEST
               *
               *FOR TRAP,OTPSW SHOULD CONTAIN ADRS OF ILL
               *
03C8  0A21     OK1      AHR    R2,R1            SETS ERROR POINTER TO '11'
03CA  4020              STH    R2,FAILNU        STORE FAIL TEST NUMBER
       0A6E
03CE  C830              LHI    R3,0             SET CONDITION CODE PORTION
       0000
               *                                OF
03D2  4030              STH    R3,X'34'         ILLEGAL INSTR INTERRUPT
       0034
               *                                NEW PSW TO ZERO
03D6  C830              LHI    R3,NEXTH         LOADS ADDR INTO ADDR PART
       03E0
03DA  4030              STH    R3,X'36'         OF NEWPSW ILLEGAL
       0036
               *                                INSTRUCTION INTERRUPT
03DE  E000     ILL      DC     X'E000'          THIS ILLEGAL INSTRUCTION
               *                                CAUSES AN INTERRUPT
03E0  4830     NEXTH    LH     R3,X'32'
       0032
03E4  C530              CLHI   R3,ILL           TEST ADDR OF ILLEGAL
       03DE
               *                                INSTRUCTION PLACED IN OLD
               *                                PSW ILLEGAL INTERRUPT
03E8  4330              BFC    3,EIGHTN         ADDRESS IS CORRECT GO
       03EE
```

```
03EC   0300             BFCR   0,ERROR          ON TO NEXT TEST
03EE   C830    EIGHTN   LHI    R3,ERRORA
        018C
03F2   4030             STH    R3,X'36'         SET ADDR TO ERROR
        0036
                *
                *
                *
                *12  12  12  12    MULTIPLY & DIVIDE INSTRUCTION TEST
                *
                * MULTIPLY & DIVIDE TEST: MULTIPLIER AND DIVIDEND IN R6
                *
                *
                *
                *
03F6   0A21             AHR    R2,R1            SETS ERROR POINTER TO 12
03F8   4020             STH    R2,FAILNU        STORE FAIL TEST NUM
        0A6E
03FC   C830             LHI    R3,TWNTY1        STORE ADDR OF NEXT
        0574
0400   4030             STH    R3,X'36'         TEST IN NEW PSW ILLEGAL
        0036
                *                               INSTRUCTION
0404   0B33             SHR    R3,R3            BECAUSE ALL PROCESSORS
                *                               DO NOT
0406   4030             STH    R3,X'34'         HAVE MULT AND DIVIDE
        0034
                *                               INSTRUCTIONS
040A   C860    MUDVT    LHI    R6,1             LOADS MULTIPLIER
        0001
040E   C890             LHI    R9,1             LOADS MULTIPLICAND
        0001
0412   0C86    LOOP1    MHR    R8,R6            FORM X  SQUARED
0414   0D86             DHR    R8,R6            DIVIDE PREVIOUS STEP
0416   4000             STH    RC,X'36'         STORE ERROR ADDR IN ILLPSW
        0036
041A   0888             LHR    R8,R8            CHECK FOR ZERO REMAINDER
041C   02F0             BTCR   X'F',ERROR
041E   0596             CLHR   R9,R6            COMPAR DIVIDEND WITH
                *                               MULTIPLIER
0420   4330             BFC    3,OK             SHOULD BE EQUAL
        0426
0424   0300             BFCR   0,ERROR          NOT, GO TO ERROR
0426   CA60    OK       AHI    R6,1             INCREMENT MULTIPLIER
        0001
042A   CA90             AHI    R9,1             INCREMENT MULTIPLICAND
        0001
042E   C560             CLHI   R6,X'FFFF'       TEST IF FINISHED
        FFFF
0432   4330             BFC    3,FINT1          IF SO, JUMP TO NEXT PART
        043A
0436   4300             BFC    0,LOOP1          NOT, CONTINUE THIS PART
        0412
                * THIS TEST MULTIPLIES AND DIVIDES POSITIVE
                * AND NEGATIVE NUMBERS AND CHECKS THE SIGNS
                * OF THE RESULTING VALUES.
043A   4850    FINT1    LH     R5,PLUS1
        0A20
043E   4C40             MH     R4,PLUS1         MULT PLUS 1 TIMES PLUS 1
        0A20
0442   4550             CLH    R5,PLUS1         R5 SHOULD HAVE PLUS 1
        0A20
```

```
0446   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
0448   0844        LHR    R4,R4         R4 SHOULD BE ZERO
044A   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
044C   4850        LH     R5,MINUS1
        0A1C
0450   4C40        MH     R4,PLUS1      MULT PLUS 1 TIMES MINUS 1
       0A20
0454   4540        CLH    R4,MINUS1
        0A1C
0458   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
045A   4550        CLH    R5,MINUS1
        0A1C
045E   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
0460   4850        LH     R5,PLUS1
       0A20
0464   4C40        MH     R4,MINUS1     MULT MINUS1 TIMES PLUS1
       0A1C
0468   4540        CLH    R4,MINUS1     R4 SHOULD HAVE MINUS 1
       0A1C
046C   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
046E   4550        CLH    R5,MINUS1     R5 SHOULD HAVE MINUS 1
       0A1C
0472   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
0474   4850        LH     R5,MINUS1
       0A1C
0478   4C40        MH     R4,MINUS1     MULT MINUS1 TIMES MINUS1
       0A1C
047C   4550        CLH    R5,PLUS1      R5 SHOULD HAVE PLUS 1
       0A20
0480   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
0482   0844        LHR    R4,R4         R4 SHOULD HAVE ZERO
0484   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
0486   0B44        SHR    R4,R4
0488   4850        LH     R5,PLUS3      R4+R5=+3
       0A26
048C   4D40        DH     R4,PLUS2      DIVIDE POSITIVE INTO
       0A22
             *                          POSITIVE NUMBER
0490   4540        CLH    R4,PLUS1      TEST FOR POSITIVE REMAINDE
       0A20
0494   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
0496   4550        CLH    R5,PLUS1      TEST FOR POSITIVE QUOTIENT
       0A20
049A   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
049C   4840        LH     R4,MINUS1
       0A1C
04A0   4850        LH     R5,MINUS3     R4+R5=-3
       0A28
04A4   4D40        DH     R4,PLUS2      DIVIDE POSITIVE INTO
       0A22
             *
04A8   4540        CLH    R4,MINUS1     TEST FOR NEGATIVE REMAINDE
       0A1C
04AC   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
04AE   4550        CLH    R5,MINUS1     TEST FOR NEGATIVE QUOTIENT
       0A1C
04B2   0230        BTCR   3,ERROR       TESTING FOR CORRECT RESPON
04B4   0B44        SHR    R4,R4
04B6   4850        LH     R5,PLUS3      R4+R5=+3
       0A26
04BA   4D40        DH     R4,MINUS2     DIVIDE NEGATIVE INTO
       0A24
```

```
04BE   4540             CLH    R4,PLUS1          TEST FOR POSITIVE REMAINDE
        0A20
04C2   0230             BTCR   3,ERROR           TESTING FOR CORRECT RESPON
04C4   4550             CLH    R5,MINUS1         TEST FOR NEGATIVE QUOTIENT
        0A1C
04C8   0230             BTCR   3,ERROR           TESTING FOR CORRECT RESPON
04CA   4840             LH     R4,MINUS1
        0A1C
04CE   4850             LH     R5,MINUS3         R4+R5=-3
        0A28
04D2   4D40             DH     R4,MINUS2         DIVIDE NEGATIVE INTO
        0A24
                 *                               NEGATIVE NUMBER
04D6   4540             CLH    R4,MINUS1         TEST FOR NEGATIVE REMAINDE
        0A1C
04DA   0230             BTCR   3,ERROR           TESTING FOR CORRECT RESPON
04DC   4550             CLH    R5,PLUS1          TEST FOR POSITIVE QUOTIENT
        0A20
04E0   0230             BTCR   3,ERROR           TESTING FOR CORRECT RESPON
                 * THIS SECTION TESTS DIVIDE FAULT
04E2   C860             LHI    R6,0              LOADS ZERO IN REG6
        0000
04E6   0B77             SHR    R7,R7
04E8   4060             STH    R6,X'4C'          STORE VALUE IN CC PART OF
        004C
                 *                               NEW PSW DIVIDE FAULT
                 *                               INTERRUPT
04EC   C860             LHI    R6,OVREC          STORE INTERRUPT ADDR IN
        0508
04F0   4060             STH    R6,X'4E'          NEW PSW DIVIDE FAULT
        004E
                 *                               INTERRUPT
04F4   C200             LPSW   ENABLE            ALLOW DIVIDE FAULT
        04F8
                 *                               INTERRUPT TO OCCUR
04F8   1000    ENABLE   DC     X'1000'           ENABLE FOR DIV
04FA   04FC             DC     A(HERE)
04FC   C860    HERE     LHI    R6,X'2000'        LOADS DIVIDEND
        2000
0500   C880             LHI    R8,X'4000'        LOADS DIVISOR
        4000
0504   0D68             DHR    R6,R8             DIVIDE SHOULD CAUSE
                 *                               INTERRUPT
0506   0300             BFCR   0,ERROR           NOT, GO TO ERROR
0508   C560    OVREC    CLHI   R6,X'2000'        TEST OPERANDS HAVE NOT
        2000
                 *                               CHANGED
050C   02F0             BTCR   X'F',ERROR        IF SO, GO TO ERROR
050E   0877             LHR    R7,R7             TEST NO REMAINDER
                 *                               GENERATED
0510   02F0             BTCR   X'F',ERROR
0512   0D68             DHR    R6,R8             TEST DIVIDING BY ZERO DOES
                 *                               NOT CAUSE PROCESSOR TO
                 *                               LOOP HERE
                 *
0514   4000             STH    R0,X'4E'
        004E
                 *
                 *
                 *ERROR POINTER SET TO 13
                 *
                 * 13 13 13
```

```
                   *
                   *
0518   0A21       NINTN   AHR    R2,R1
051A   4020               STH    R2,FAILNU          STORE FAIL TEST NUMBER
       0A6E
                   * READ BLOCK-WRITE BLOCK INSTRUCTION TEST
051E   4830               LH     R3,SKIPTS
       0A70
0522   4330               BZ     TWNTY
       0576
0526   C8C0       WRTBLK  LHI    R12,2              LOADS TTY DEVICE NUMBER
       0002
052A   DEC0               OC     R12,WDATA
       0A2A
052E   D6C0               WB     R12,FSTLOC         TEST WB INSTRUCTION
       0A2C
0532   02F0               BTCR   X'F',ERROR
0534   C840               LHI    R4,DATA3
       0A64
0538   C850               LHI    R5,DATA4
       0A6D
053C   96C4               WBR    R12,R4             TEST WBR INSTRUCTION
053E   DEC0               OC     R12,RDATA
       0A2B
0542   D7C0               RB     R12,THDLOC         TEST RB INSTRUCTION
       0A30
0546   02F0               BTCR   X'F',ERROR
0548   C840               LHI    R4,TEMP+3
       0A75
054C   C850               LHI    R5,TEMP+4
       0A76
0550   97C4               RBR    R12,R4             TEST RBR INSTRUCTION
0552   02F0               BTCR   X'F',ERROR
0554   C840               LHI    R4,X'7F'
       007F
0558   C850               LHI    R5,-5              INITIALIZE LOOP COUNT
       FFFB
055C   D365       TESTA   LB     R6,TEMP+5(R5)      LOADS  TTY CHAR IN R6
       0A77
0560   D375               LB     R7,DATA3+10(R5) TEST CHARS RECEIVED
       0A6E
0564   0464               NHR    R6,R4              STRIP PARITY BIT
0566   0567               CLHR   R6,R7              MATCH CHARS IN TABLE
0568   0230               BTCR   3,ERROR
056A   0A51               AHR    R5,R1              INCREMENT LOOP COUNT
056C   4230               BNZ    TESTA
       055C
0570   4300               B      TWNTY
       0576
                   *
                   *
                   *ERROR POINTER NOW SET TO FOURTEEN
                   *
                   * INPUT/OUTPUT, ACKNOWLEDGE INTERRUPT, AND FALSE SYNC



                   * 14    14     14       TEST WD,WDR,RD,RDR,SS,SSR
                   *                       OC,OCR  INSTRUCTIONS
```

```
0574  0A21    TWNTY1  AHR   R2,R1        INCREMENT ERROR POINTER BY
0576  0A21    TWNTY   AHR   R2,R1
0578  4020            STH   R2,FAILNU    STORE FAIL TEST NUMBER
       0A6E
057C  4830            LH    R3,SKIPTS
       0A70
0580  4330            BZ    FIFTEN
       0632
0584  C8C0            LHI   R12,2        LOAD TTY DEVICE NUMBER
       0002
0588  DEC0            OC    R12,WDATA    TEST OC INSTRUCTION
       0A2A
058C  C830            LHI   R3,DATA1
       0A5A
0590  C840            LHI   R4,1
       0001
0594  C850            LHI   R5,DATA4
       0A6D
0598  9DC6    STAT4   SSR   R12,R6       TEST SSR INSTRUCTION
059A  42F0            BTC   X'F',STAT4
       0598
059E  DAC3            WD    R12,0(R3)    TEST OF WD INSTRUCTION
       0000
05A2  DDC0    STAT5   SS    R12,TEMP     TEST SS INSTRUCTION
       0A72
05A6  42F0            BTC   X'F',STAT5
       05A2
05AA  0A31            AHR   R3,R1
05AC  D363            LB    R6,0(R3)
       0000
05B0  9AC6            WDR   R12,R6       TEST WDR INSTRUCTION
05B2  C130            BXLE  R3,STAT4
       0598
05B6  D330            LB    R3,RDATA
       0A2B
05BA  9EC3            OCR   R12,R3       TEST OF OCR INSTRUCTION
05BC  C830            LHI   R3,TEMP
       0A72
05C0  9DC6    STAT6   SSR   R12,R6
05C2  42F0            BTC   X'F',STAT6
       05C0
05C6  DBC3            RD    R12,0(R3)    TEST RD INSTRUCTION
       0000
05CA  C530            CLHI  R3,TEMP+4
       0A76
05CE  4380            BNL   CHART
       05E6
05D2  0A31            AHR   R3,R1
05D4  9DC6    STAT7   SSR   R12,R6
05D6  42F0            BTC   X'F',STAT7
       05D4
05DA  9BC6            RDR   R12,R6       TEST RDR INSTRUCTION
05DC  D263            STB   R6,0(R3)
       0000
05E0  0A31            AHR   R3,R1
05E2  4300            B     STAT6
       05C0
05E6  C840    CHART   LHI   R4,X'7F'     MASK FOR PARITY
       007F
05EA  C850            LHI   R5,-5        INITIALIZE LOOP COUNT
       FFFB
```

```
05EE  D365     TESTB   LB     R6,TEMP+5(R5)    LOADS  TTY CHAR IN R6
       0A77
05F2  D375             LB     R7,DATA3+10(R5)  TEST CHARS RECEIVED
       0A6E
05F6  0464             NHR    R6,R4            STRIP PARITY BIT
05F8  0567             CLHR   R6,R7            MATCH CHARS IN TABLE
05FA  0230             BTCR   3,ERROR
05FC  0A51             AHR    R5,R1            INCREMENT LOOP COUNT
05FE  4230             BNZ    TESTB
       05EE
               * THIS IS A COMBINED TEST OF THE AIR AND AI
               * INSTRUCTIONS AND A TEST OF THE FALSE SYNC
0602  9F34             AIR    R3,R4            REMOVE PENDING INTERRUPTS
0604  C830             LHI    R3,X'FFFF'       SET R3 BITS TO ALL ONES
       FFFF
0608  C840             LHI    R4,X'FFFF'       SET R4 BITS TO ALL ONES
       FFFF
060C  DF30             AI     R3,TEMP+1        THIS GENERATES FALSE SYNC
       0A73
0610  D340             LB     R4,TEMP+1        BIT 13 SHOULD GET SET
       0A73
0614  C540             CLHI   R4,4             TEST FOR THIS
       0004
0618  0230             BTCR   3,ERROR
061A  0833             LHR    R3,R3            R3 SHOULD BE SET TO ZERO
061C  0230             BTCR   3,ERROR
061E  C830             LHI    R3,X'FFFF'       SET R3 BITS TO ALL ONES
       FFFF
0622  C840             LHI    R4,X'FFFF'       SET R4 BITS TO ALL ONES
       FFFF
0626  9F34             AIR    R3,R4            THIS GENERATES FALSE SYNC
0628  C540             CLHI   R4,4             BIT 13 SHOULD GET SET
       0004
062C  0230             BTCR   3,ERROR
062E  0833             LHR    R3,R3            R3 SHOULD BE SET TO ZERO
0630  0230             BTCR   3,ERROR
               *
               * ERROR POINTER NOW SET TO FIFTEEN
               *
               * 15   LOAD AND STORE MULTIPLE INSTRUCTION TEST
0632  0A21     FIFTEN  AHR    R2,R1            SETS ERROR POINTER TO 15
0634  4020             STH    R2,FAILNU        STORE FAIL TEST NUMBER
       0A6E
0638  C800             LHI    R0,2             INITIALIZES R0 TO 2
       0002
063C  C830             LHI    R3,3             INITIALIZES R3 TO 3
       0003
0640  C840             LHI    R4,4             INITIALIZES R4 TO 4
       0004
0644  C850             LHI    R5,5             INITIALIZES R5 TO 5
       0005
0648  C860             LHI    R6,6             INITIALIZES R6 TO 6
       0006
064C  C870             LHI    R7,7             INITIALIZES R7 TO 7
       0007
0650  C880             LHI    R8,8             INITIALIZES R8 TO 8
       0008
0654  C890             LHI    R9,9             INITIALIZES R9 TO 9
       0009
0658  C8A0             LHI    R10,10           INITIALIZES R10 TO 10
       000A
065C  C8B0             LHI    R11,11           INITIALIZES R11 TO 11
```

```
         000B
0660    C8C0            LHI    R12,12           INITIALIZES R12 TO 12
         000C
0664    C8D0            LHI    R13,13           INITIALIZES R13 TO 13
         000D
0668    C8E0            LHI    R14,14           INITIALIZES R14 TO 14
         000E
066C    C8F0            LHI    R15,15           INITIALIZES R14 TO 15
         000F
                   *     STM    0,TEMP              TEST STORE MULTIPLE
                   *                            INSTRUCTION
0670    D000            DC     X'D000',A(TEMP)  TEMPORARY FOR STM INSTRUCT
         0A72
0674    0800            SHR    R0,R0            THIS SETS R0 TO ALL ZEROS
0676    0810            LHR    R1,R0            THIS SETS R1 TO ALL ZEROS
0678    0821            LHR    R2,R1            THIS SETS R2 TO ALL ZEROS
067A    0832            LHR    R3,R2            THIS SETS R3 TO ALL ZEROS
067C    0843            LHR    R4,R3            THIS SETS R4 TO ALL ZEROS
067E    0854            LHR    R5,R4            THIS SETS R5 TO ALL ZEROS
0680    0865            LHR    R6,R5            THIS SETS R6 TO ALL ZEROS
0682    0876            LHR    R7,R6            THIS SETS R7 TO ALL ZEROS
0684    0887            LHR    R8,R7            THIS SETS R8 TO ALL ZEROS
0686    0898            LHR    R9,R8            THIS SETS R9 TO ALL ZEROS
0688    08A9            LHR    R10,R9           THIS SETS R10 TO ALL ZEROS
068A    08BA            LHR    R11,R10          THIS SETS R11 TO ALL ZEROS
068C    08CB            LHR    R12,R11          THIS SETS R12 TO ALL ZEROS
068E    08DC            LHR    R13,R12          THIS SETS R13 TO ALL ZEROS
0690    08ED            LHR    R14,R13          THIS SETS R14 TO ALL ZEROS
0692    08FE            LHR    R15,R14          THIS SETS R15 TO ALL ZEROS
                   *     LM     0,TEMP              TEST LOAD MULTIPLE
                   *                            INSTRUCTION
0694    D100            DC     X'D100',A(TEMP)  TEMPORARY FOR LM INSTRUCTI
         0A72
0698    C500            CLHI   R0,2             TEST R0 RESTORED CORRECTLY
         0002
069C    4230            BNE    ERRORA
         018C
06A0    C510            CLHI   R1,1             TEST R1 RESTORED CORRECTLY
         0001
06A4    4230            BNE    ERRORA
         018C
06A8    C520            CLHI   R2,X'15'         TEST R2 RESTORED CORRECTLY
         0015
06AC    4230            BNE    ERRORA
         018C
06B0    C530            CLHI   R3,3             TEST R3 RESTORED CORRECTLY
         0003
06B4    4230            BNE    ERRORA
         018C
06B8    C540            CLHI   R4,4             TEST R4 RESTORED CORRECTLY
         0004
06BC    4230            BNE    ERRORA
         018C
06C0    C550            CLHI   R5,5             TEST R5 RESTORED CORRECTLY
         0005
06C4    4230            BNE    ERRORA
         018C
06C8    C560            CLHI   R6,6             TEST R6 RESTORED CORRECTLY
         0006
06CC    4230            BNE    ERRORA
         018C
06D0    C570            CLHI   R7,7             TEST R7 RESTORED CORRECTLY
```

```
       0007
06D4   4230         BNE    ERRORA
       018C
06D8   C580         CLHI   R8,8          TEST R8 RESTORED CORRECTLY
       0008
06DC   4230         BNE    ERRORA
       018C
06E0   C590         CLHI   R9,9          TEST R9 RESTORED CORRECTLY
       0009
06E4   4230         BNE    ERRORA
       018C
06E8   C5A0         CLHI   R10,10        TEST R10 RESTORED CORRECTL
       000A
06EC   4230         BNE    ERRORA
       018C
06F0   C5B0         CLHI   R11,11        TEST R11 RESTORED CORRECTL
       000B
06F4   4230         BNE    ERRORA
       018C
06F8   C5C0         CLHI   R12,12        TEST R12 RESTORED CORRECTL
       000C
06FC   4230         BNE    ERRORA
       018C
0700   C5D0         CLHI   R13,13        TEST R13 RESTORED CORRECTL
       000D
0704   4230         BNE    ERRORA
       018C
0708   C5E0         CLHI   R14,14        TEST R14 RESTORED CORRECTL
       000E
070C   4230         BNE    ERRORA
       018C
0710   C5F0         CLHI   R15,15        TEST R15 RESTORED CORRECTL
       000F
0714   4230         BNE    ERRORA
       018C
            *
            * ERROR POINTER NOW SET TO 16
            *
            * 16 16 16 16 SOURCE AND DESTINATION TEST
            *
0718   C820  SIXTEN LHI    R2,X'16'      TEST NUMBER OF THIS SECT
       0016
071C   4020         STH    R2,FAILNU     STORE FAIL TEST NUMBER
       0A6E
0720   C800         LHI    0,X'1'        THE PURPOSE OF THIS TEST
       0001
0724   C810         LHI    1,X'2'        IS TO TEST THAT EACH REG
       0002
0728   C820         LHI    2,X'4'        CAN BE USED BOTH AS SOURCE
       0004
072C   C830         LHI    3,X'8'        AND DESTINATION REGISTER
       0008
0730   C840         LHI    4,X'10'
       0010
0734   C850         LHI    5,X'20'
       0020
0738   C860         LHI    6,X'40'
       0040
073C   C870         LHI    7,X'80'
       0080
0740   C880         LHI    8,X'100'
       0100
```

```
0744   C890           LHI    9,X'200'
        0200
0748   C8A0           LHI    10,X'400'
        0400
074C   C8B0           LHI    11,X'800'
        0800
0750   C8C0           LHI    12,X'1000'
        1000
0754   C8D0           LHI    13,X'2000'
        2000
0758   C8E0           LHI    14,X'4000'
        4000
075C   C8F0           LHI    15,X'8000'
        8000
0760   C500           CLHI   0,X'1'
        0001
0764   4230           BNE    ERRORA
        018C
0768   C510           CLHI   1,X'2'
        0002
076C   4230           BNE    ERRORA
        018C
0770   C520           CLHI   2,X'4'
        0004
0774   4230           BNE    ERRORA
        018C
0778   C530           CLHI   3,X'8'
        0008
077C   4230           BNE    ERRORA
        018C
0780   C540           CLHI   4,X'10'
        0010
0784   4230           BNE    ERRORA
        018C
0788   C550           CLHI   5,X'20'
        0020
078C   4230           BNE    ERRORA
        018C
0790   C560           CLHI   6,X'40'
        0040
0794   4230           BNE    ERRORA
        018C
0798   C570           CLHI   7,X'80'
        0080
079C   4230           BNE    ERRORA
        018C
07A0   C580           CLHI   8,X'100'
        0100
07A4   4230           BNE    ERRORA
        018C
07A8   C590           CLHI   9,X'200'
        0200
07AC   4230           BNE    ERRORA
        018C
07B0   C5A0           CLHI   10,X'400'
        0400
07B4   4230           BNE    ERRORA
        018C
07B8   C5B0           CLHI   11,X'800'
        0800
07BC   4230           BNE    ERRORA
        018C
```

```
07C0  C5C0          CLHI   12,X'1000'
      1000
07C4  4230          BNE    ERRORA
      018C
07C8  C5D0          CLHI   13,X'2000'
      2000
07CC  4230          BNE    ERRORA
      018C
07D0  C5E0          CLHI   14,X'4000'
      4000
07D4  4230          BNE    ERRORA
      018C
07D8  C5F0          CLHI   15,X'8000'
      8000
07DC  4230          BNE    ERRORA
      018C
07E0  06F0          OHR    15,0
07E2  06E1          OHR    14,1
07E4  06D2          OHR    13,2
07E6  06C3          OHR    12,3
07E8  06B4          OHR    11,4
07EA  06A5          OHR    10,5
07EC  0696          OHR    9,6
07EE  0687          OHR    8,7
07F0  0678          OHR    7,8
07F2  0669          OHR    6,9
07F4  065A          OHR    5,10
07F6  064B          OHR    4,11
07F8  063C          OHR    3,12
07FA  062D          OHR    2,13
07FC  061E          OHR    1,14
07FE  060F          OHR    0,15
0800  050F          CLHR   0,15
0802  4230          BNE    ERRORA
      018C
0806  051E          CLHR   1,14
0808  4230          BNE    ERRORA
      018C
080C  052D          CLHR   2,13
080E  4230          BNE    ERRORA
      018C
0812  053C          CLHR   3,12
0814  4230          BNE    ERRORA
      018C
0818  054B          CLHR   4,11
081A  4230          BNE    ERRORA
      018C
081E  055A          CLHR   5,10
0820  4230          BNE    ERRORA
      018C
0824  0569          CLHR   6,9
0826  4230          BNE    ERRORA
      018C
082A  0578          CLHR   7,8
082C  4230          BNE    ERRORA
      018C
              *
              * ERROR POINTER NOW SET TO 17
              *
              *
              * 17 17 17 17 SET AND RESET EVERY BIT IN EVERY REG. TEST
0830  C820    SEVTEN  LHI    R2,X'17'        SETS ERROR POINTER TO 17
```

```
       0017
08 34  4020              STH    R2,FAILNU          STORE FAIL TEST NUMBER
       0A6E
08 38  C810              LHI    1,-64              THIS LOOP IS USED TO TEST
       FFC0
                 *                                 REG 0
083C   4801      REGOT   LH     0,DIAGN+64(1)
       0A14
0840   4501              CLH    0,DIAGN+64(1)
       0A14
0844   4230              BNE    ERRORA
       018C
0848   CA10              AHI    1,2
       0002
084C   4230              BNZ    REGOT
       083C
0850   C820              LHI    2,-64              THIS LOOP IS USED TO TEST
       FFC0
0854   4812      REG1T   LH     1,DIAGN+64(2)      R 1
       0A14
0858   4512              CLH    1,DIAGN+64(2)
       0A14
085C   4230              BNE    ERRORA
       018C
0860   CA20              AHI    2,2
       0002
0864   4230              BNZ    REG1T
       0854
0868   C810              LHI    1,-64              THIS LOOP IS USED TO TEST
       FFC0
                 *                                 REG 2
086C   4821      REG2T   LH     2,DIAGN+64(1)
       0A14
0870   4521              CLH    2,DIAGN+64(1)
       0A14
0874   4230              BNE    ERRORA
       018C
0878   CA10              AHI    1,2
       0002
087C   4230              BNZ    REG2T
       086C
0880   C820              LHI    2,2
       0002
0884   C810              LHI    1,-64              THIS LOOP IS USED TO TEST
       FFC0
                 *                                 REG3
0888   4831      REG3T   LH     3,DIAGN+64(1)
       0A14
088C   4531              CLH    3,DIAGN+64(1)
       0A14
0890   4230              BNE    ERRORA
       018C
0894   0A12              AHR    1,2
0896   4230              BNZ    REG3T
       0888
089A   C810              LHI    1,-64              THIS LOOP IS USED TO TEST
       FFC0
                 *                                 REG4
089E   4841      REG4T   LH     4,DIAGN+64(1)
       0A14
08A2   4541              CLH    4,DIAGN+64(1)
       0A14
```

```
08A6  4230             BNE   ERRORA
       018C
08AA  0A12             AHR   1,2
08AC  4230             BNZ   REG4T
       089E
08B0  C810             LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
08B4  4851   REG5T     LH    5,DIAGN+64(1)    REG5
       0A14
08B8  4551             CLH   5,DIAGN+64(1)
       0A14
08BC  4230             BNE   ERRORA
       018C
08C0  0A12             AHR   1,2
08C2  4230             BNZ   REG5T
       08B4
08C6  C810             LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
08CA  4861   REG6T     LH    6,DIAGN+64(1)    REG6
       0A14
08CE  4561             CLH   6,DIAGN+64(1)
       0A14
08D2  4230             BNE   ERRORA
       018C
08D6  0A12             AHR   1,2
08D8  4230             BNZ   REG6T
       08CA
08DC  C810             LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
08E0  4871   REG7T     LH    7,DIAGN+64(1)    REG 7
       0A14
08E4  4571             CLH   7,DIAGN+64(1)
       0A14
08E8  4230             BNE   ERRORA
       018C
08EC  0A12             AHR   1,2
08EE  4230             BNZ   REG7T
       08E0
08F2  C810             LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
08F6  4881   REG8T     LH    8,DIAGN+64(1)    REG 8
       0A14
08FA  4581             CLH   8,DIAGN+64(1)
       0A14
08FE  4230             BNE   ERRORA
       018C
0902  0A12             AHR   1,2
0904  4230             BNZ   REG8T
       08F6
0908  C810             LHI   1,-64            THIS LOOP US USED TO TEST
       FFC0
090C  4891   REG9T     LH    9,DIAGN+64(1)    REG 9
       0A14
0910  4591             CLH   9,DIAGN+64(1)
       0A14
0914  4230             BNE   ERRORA
       018C
0918  0A12             AHR   1,2
091A  4230             BNZ   REG9T
       090C
091E  C810             LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
```

```
0922  48A1    REG10T  LH    10,DIAGN+64(1)   REG 10
       0A14
0926  45A1            CLH   10,DIAGN+64(1)
       0A14
092A  4230            BNE   ERRORA
       018C
092E  0A12            AHR   1,2
0930  4230            BNZ   REG10T
      0922
0934  C810            LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
0938  48B1    REG11T  LH    11,DIAGN+64(1)   REG 11
       0A14
093C  45B1            CLH   11,DIAGN+64(1)
       0A14
0940  4230            BNE   ERRORA
       018C
0944  0A12            AHR   1,2
0946  4230            BNZ   REG11T
       0938
094A  C810            LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
094E  48C1    REG12T  LH    12,DIAGN+64(1)   R12
       0A14
0952  45C1            CLH   12,DIAGN+64(1)
       0A14
0956  4230            BNE   ERRORA
       018C
095A  0A12            AHR   1,2
095C  4230            BNZ   REG12T
       094E
0960  C810            LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
0964  48D1    REG13T  LH    13,DIAGN+64(1)   R13
       0A14
0968  45D1            CLH   13,DIAGN+64(1)
       0A14
096C  4230            BNE   ERRORA
       018C
0970  0A12            AHR   1,2
0972  4230            BNZ   REG13T
       0964
0976  C810            LHI   1,-64
       FFC0
097A  48E1    REG14T  LH    14,DIAGN+64(1)   THIS LOOP IS USED TO TEST
       0A14
097E  45E1            CLH   14,DIAGN+64(1)   R14
       0A14
0982  4230            BNE   ERRORA
       018C
0986  0A12            AHR   1,2
0988  4230            BNZ   REG14T
       097A
098C  C810            LHI   1,-64            THIS LOOP IS USED TO TEST
       FFC0
0990  48F1    REG15T  LH    15,DIAGN+64(1)   R15
       0A14
0994  45F1            CLH   15,DIAGN+64(1)
       0A14
0998  4230            BNE   ERRORA
       018C
099C  0A12            AHR   1,2
```

```
099E   4230               BNZ    REG15T
        0990
09A2   C830               LHI    R3,MESS1
        0A40
09A6   C840               LHI    R4,1
        0001
09AA   C850               LHI    R5,MESS2-1
        0A59
09AE   C8C0               LHI    R12,2          LOAD DEV NUN OF TTY
        0002
09B2   DEC0     STAT3     OC     R12,WDATA      TTY TO WRITE MODE
        0A2A
09B6   9DCE               SSR    R12,R14        TEST STATUS BYTE OF TTY
09B8   42F0               BTC    X'F',STAT3
        09B2
09BC   DAC3               WD     R12,0(3)       OUTPUT OK MESSAGE
        0000
09C0   C130               BXLE   R3,STAT3
        09B2
09C4   4820               LH     R2,SKIPTS
        0A70
09C8   4330               BZ     SKIPEN
        00A2
09CC   C200               LPSW   OKWAIT
        09D0
09D0   8000     OKWAIT    DC     X'8000'
09D2   009A               DC     A(ZERO)
09D4   0001     DIAGN     DC     X'1'           THIS TABLE IS USED TO
09D6   9393     XOFF      DC     X'9393'
                *                               PROPAGTE
09D8   0002               DC     X'2'           A ONE THRU A FIELD OF ZERO
                *                               AND
09DA   0004               DC     X'4'           A ZERO THRU A FIELD OF ONE
09DC   0008               DC     X'8'
09DE   0010               DC     X'10'
09E0   0020               DC     X'20'
09E2   0040               DC     X'40'
09E4   0080               DC     X'80'
09E6   0100               DC     X'100'
09E8   0200               DC     X'200'
09EA   0400               DC     X'400'
09EC   0800               DC     X'800'
09EE   1000               DC     X'1000'
09F0   2000               DC     X'2000'
09F2   4000               DC     X'4000'
09F4   8000               DC     X'8000'
09F6   FFFE               DC     X'FFFE'
09F8   FFFD               DC     X'FFFD'
09FA   FFFB               DC     X'FFFB'
09FC   FFF7               DC     X'FFF7'
09FE   FFEF               DC     X'FFEF'
0A00   FFDF               DC     X'FFDF'
0A02   FFBF               DC     X'FFBF'
0A04   FF7F               DC     X'FF7F'
0A06   FEFF               DC     X'FEFF'
0A08   FDFF               DC     X'FDFF'
0A0A   FBFF               DC     X'FBFF'
0A0C   F7FF               DC     X'F7FF'
0A0E   EFFF               DC     X'EFFF'
0A10   DFFF               DC     X'DFFF'
0A12   BFFF               DC     X'BFFF'
0A14   7FFF               DC     X'7FFF'
```

```
0A16  FFFE    ALMFX    DC    X'FFFE'
0A18  5A5A    ALT      DC    X'5A5A'
0A1A  8000    EIGTH    DC    X'8000'
0A1C  FFFF    FOXES    DC    X'FFFF'
0A1E  0000    NADA     DC    0
0A20  0001    PLUS1    DC    1
0A22  0002    PLUS2    DC    2
0A24  FFFE    MINUS2   DC    -2
0A26  0003    PLUS3    DC    3
0A28  FFFD    MINUS3   DC    -3
0A1C          MINUS1   EQU   FOXES
0A2A  98A4    WDATA    DC    X'98A4'
0A2B          RDATA    EQU   WDATA+1
0A2C  0A5A    FSTLOC   DC    DATA1
0A2E  0A63    SECLOC   DC    DATA2
0A30  0A72    THDLOC   DC    TEMP
0A32  0A74    FORLOC   DC    TEMP+2
0A34  8D8A    MESS     DC    X'8D8A',C'FAILURE'
       4641
       494C
       5552
       4520
0A3E          TESTNU   DS    2
0A40  8D8A    MESS1    DC    X'8D8A'
0A42  4745             DC    C'GE-PAC MOD     EL 30-2  IS AOK'
       2D50
       4143
       204D
       4F44
       454C
       2033
       302D
       3220
       2041
       4F4B
0A5A          MESS2    EQU   *
0A5A  8D8A    DATA1    DC    X'8D8A',C'DEPRESS'
       4445
       5052
       4553
       5320
0A63          DATA2    EQU   *-1
0A64  4B45    DATA3    DC    C'KEYS             W,R,U'
       5953
       2057
       2C52
       2C55
0A6D          DATA4    EQU   *-1
0A6E          FAILNU   DS    2
0A70          SKIPTS   DS    2
0A72          TEMP     DS    32
0A92          SAVE     DS    32
0000          ERROR    EQU   0
0000          R0       EQU   0
0001          R1       EQU   1
0002          R2       EQU   2
0003          R3       EQU   3
0004          R4       EQU   4
0005          R5       EQU   5
0006          R6       EQU   6
0007          R7       EQU   7
```

```
0008          R8      EQU    8
0009          R9      EQU    9
000A          R10     EQU    X'A'
000B          R11     EQU    X'B'
000C          R12     EQU    X'C'
000D          R13     EQU    X'D'
000E          R14     EQU    X'E'
000F          R15     EQU    X'F'
0044          NIPSW   EQU    X'44'
0030          OTRPSW  EQU    X'30'
0AB2                  END

 ALMFX    0A16
 ALT      0A18
 AROUND   020A
 BRAN2    036C
 BRANCH   0360
 BY1      0214
 BY2      021A
 CHART    05E6
 CONVRT   0192
 DATA1    0A5A
 DATA2    0A61
 DATA3    0A64
 DATA4    0A60
 DIAGN    09D4
 EIGHT    0232
 EIGHTN   03EE
 EIGTH    0A1A
 ELEVEN   0302
 ENABLE   04F8
 ERROR    0000
 ERRORA   018C
 ERRWAT   0206
 FAILNU   0A6E
 FIFTEN   0632
 FINT1    043A
 FIVE     0166
 FORLOC   0A32
 FOUR     014A
 FOURT    0352
 FOXES    0A1C
 FSTLOC   0A2C
 GO       0348
 GO1      034C
 HERE     04FC
 ILL      03DE
 LOC1     03A0
 LOOP1    0412
 MESS     0A34
 MESS1    0A40
 MESS2    0A5A
 MINUS1   0A1C
 MINUS2   0A24
 MINUS3   0A28
 MUDVT    040A
 NADA     0A1E
 NEXTH    03E0
 NINE     0252
 NINTN    0518
 NIPSW    0044
 OK       0426
```

OK1	03C8
OKWAIT	09D0
ONE	00DA
ONES	0326
ONES1	0328
OTRPSW	0030
OVREC	0508
PLUS1	0A20
PLUS2	0A22
PLUS3	0A26
R0	0000
R1	0001
R10	000A
R11	000B
R12	000C
R13	000D
R14	000E
R15	000F
R2	0002
R3	0003
R4	0004
R5	0005
R6	0006
R7	0007
R8	0008
R9	0009
RDATA	0A2B
REG0T	083C
REG10T	0922
REG11T	0938
REG12T	094E
REG13T	0964
REG14T	097A
REG15T	0990
REG1T	0854
REG2T	086C
REG3T	0888
REG4T	089E
REG5T	08B4
REG6T	08CA
REG7T	08E0
REG8T	08F6
REG9T	090C
RETRN3	01F6
RETRN4	01FA
RND	0090
ROUND	008C
SAVE	0A92
SECLOC	0A2E
SENS	01CE
SEVEN	021C
SEVTEN	0830
SIX	0182
SIXTEN	0718
SKIPEN	00A2
SKIPTS	0A70
STAT3	09B2
STAT4	0598
STAT5	05A2
STAT6	05C0
STAT7	05D4
TEMP	0A72

TEN	029C
TESTA	055C
TESTB	05EE
TESTNU	0A3E
THDLOC	0A30
THERE	022A
THIRT	033C
THREE	0132
TWELV	032A
TWNTY	0576
TWNTY1	0574
TWO	0106
WDATA	0A2A
WRTBLK	0526
XOFF	09D6
ZERO	009A
ZEROA	00A6

FLOATING-POINT TEST PROGRAM OPERATION MANUAL

Publication Number 06-046A12

1. INTRODUCTION

The purpose of the Floating-Point Test Program, Program Number 06-046, is to test all the floating-point instructions. Refer to the Reference Manual, Publication Number 29-004, for an explanation of the floating-point instructions. Each instruction is exercised thoroughly and the results obtained are compared to a table of expected results.

If there are no failures, the program causes a "FLOATING-POINT OPERATIONS OK" message to be printed. Should a failure be encountered, an error message of "FAILURE IN TEST NUMBER" is printed and the Processor is halted with the number of the test that failed in Display Register 2 (rightmost eight bits).

The test program is divided into six general sections. The sections are subdivided into specific tests that check the various aspects and formats of a particular instruction. This is done by manipulating the signs, exponents, and magnitudes of the operands. Upon completion of each floating-point operation, the Condition Code bits are inspected as a final validity check.

2. OPERATING INSTRUCTIONS

The test program (06-046M09) is an Absolute tape and may be loaded with the Absolute, Relocating, or General Loader. The program occupies memory from X'80' to X'AAA', with the starting location being X'80'. For more detailed information on tape loading and execution, refer to <u>Loader Description Manual</u>, Publication Number 06-025A12.

The input device, used in loading the test program object tape, is taken from BINDV (X'78') in the table in the 50 Sequence; the output device used in printing the message is taken from LISTDV (X'7E'). These locations must be manually set up by the user.

After either the error message or the "OK" message has been typed, the Processor halts with the address portion of the PSW loaded with X'80', the starting address of the test. The execution time for the test is a few seconds.

3. SECTION AND TEST DESCRIPTIONS

3.1 Section 1 (Test 0 - Test 10)

The two forms of Load (LE/LER) and the Store (STE) instructions are executed to test for proper normalization of floating-point operands. The G, L, and V bits of the Condition Code are used as further criteria in checking the integrity of these instructions. The test number is placed in General Register 2 (ERR) and it is the contents of this register that is printed if an error is encountered.

3.2 Section 2 (Test 20 - Test 2F)

The two forms of Addition (AE/AER) are tested in this Section. After each Add operation, the Condition Code bits are tested. The sum is then tested by comparing it to an expected result.

3.3 Section 3 (Test 30 - Test 34)

Both forms of Subtract (SE/SER) are tested in this Section. After each Subtract operation, the bits of the Condition Code are checked. The difference obtained is then compared to a table of expected results.

3.4 Section 4 (Test 40 - Test 52)

The RR and RX forms of Multiply (ME/MER) are checked in this Section. The G, L, and V flags of the Condition Code are checked upon completion of the Multiply instruction. The product is then compared to an expected result. The normalization before rounding feature is also tested.

3.5 Section 5 (Test 60 - Test 6A)

The two forms of Divide (DE/DER) and the Divide Fault Interrupt are exercised in this Section. The Condition Code is examined by Branch instruction as a supplementary check.

3.6 Section 6 (Test 70 - Test 74)

Both forms of floating-point Compare (CE/CER) are tested in this Section. The LOAD/STORE and Arithmetic results are checked by the fixed-point Compare instructions (CLH/CLHR). The floating-point Compare instructions are tested by using Branch instructions to determine the bits of the Condition Code. If this Section has been successfully completed, the program prints "FLOATING-POINT OPERATIONS OK".

4. ILLEGAL INSTRUCTION INTERRUPT

If an Illegal instruction is decoded, the Processor will halt with the Instruction Address Counter set to X'80'. The instruction that caused the failure will be stored in Old PSW: Illegal Instruction Interrupt (X'30' - X'33').

Refer to Appendix 1 for the test numbers and the restart address of each test. The restart address should not be used to initiate execution unless the test program has been previously started from X'80', which sets up appropriate registers. A listing of the test with the operands and the expected results is provided in Appendix 2.

APPENDIX 1
FLOATING-POINT TEST PROGRAM

Test Number	Restart Location	Test Number	Restart Location	
ØØ	98	34	428	
Ø1	A6			
Ø2	B6	4Ø	44C	
Ø3	C6	41	464	
Ø4	E2	42	478	
Ø5	F2	43	48C	
Ø6	1Ø2	44	4A2	
Ø7	11E	45	4B6	
Ø8	12E	46	4CA	
Ø9	13E	47	4DE	
ØA	14C	48	4F2	
ØB	15A	49	5Ø6	
ØC	16A	4A	51C	
ØD	17A	4B	52E	
ØE	194	4C	542	
ØF	1A4	4D	556	ØØ - 1Ø = LE/LER/STE
1Ø	1B4	4E	566	
		4F	576	2Ø - 2F = AE/AER
2Ø	1C8	5Ø	586	
21	1EE	51	59A	3Ø - 34 = SE/SER
22	212	52	5AE	
23	236			4Ø - 52 = ME/MER
24	25A	6Ø	5C6	
25	26E	61	5DE	6Ø - 6A = DE/DER
26	282	62	5F4	
27	296	63	6Ø8	7Ø - 74 = CE/CER
28	2A8	64	61C	
29	2BC	65	63Ø	
2A	2EØ	66	646	
2B	3Ø4	67	65C	
2C	328	68	67Ø	
2D	34A	69	684	
2E	37Ø	6A*	69A	*DVD FAULT INTERRUPT
2F	384			
		7Ø	6DC	
3Ø	398	71	6F4	
31	3BC	72	7Ø2	
32	3EØ	73	7ØC	
33	4Ø4	74	71E	

APPENDIX 2
Floating-Point Test
Program Listing
November 1968

```
                    *
0080                          ORG     X'80'
                    *
                    * TEST FLOATING POINT INSTRUCTIONS:
                    *         06-046
                    *
                    *                   LE/LER
                    *                   STE
                    *                   CE/CER
                    *                   AE/AER
                    *                   SE/SER
                    *                   ME/MER
                    *                   DE/DER
                    *
0001                ONE       EQU     1
0002                ERR       EQU     2
0003                TOT       EQU     3
0004                INDEX     EQU     4
0005                INCR      EQU     5
0006                LIMIT     EQU     6
0007                OUT       EQU     7
0008                DEV       EQU     8
0009                STATUS    EQU     9
000A                HELP      EQU     10
                    *
                    * LOAD/STORE CHECK AND
                    * CONDITION CODE CHECK
                    *
0080  4820                    LH      ERR,STOP          SET UP NEW PSW OF ILL.
       076A
0084  4020                    STH     ERR,X'34'         INST. INTERRUPT. TO
       0034
0088  4820                    LH      ERR,STOP+2        HALT IF EVER ENCOUNTERED
       076C
008C  4020                    STH     ERR,X'36'
       0036
0090  C810                    LHI     ONE,1
       0001
0094  C820                    LHI     ERR,ERROR
       076E
0098  0733                    XHR     TOT,TOT           ZERO ERROR TOTAL COUNT
009A  6800                    LE      0,D0              DATA0 TO REG. 0
       083E
009E  02F2                    BTCR    X'F',ERR          COND. CODE = 0
00A0  41F0                    BAL     15,COMP0
       0798
00A4  0842                    DC      R0                RESULT0 NORMALIZED
                    *
00A6  0A31                    AHR     TOT,ONE           ADD ONE TO ERROR COUNT
00A8  6810                    LE      1,D1              DATA1 TO REG. 0 & REG1
       0846
00AC  0322                    BFCR    2,ERR             C.C. = 2
00AE  02D2                    BTCR    X'D',ERR
00B0  41F0                    BAL     15,COMP0
       0798
00B4  084A                    DC      R1                RESULT1 NORMALIZED
                    *
00B6  0A31                    AHR     TOT,ONE           TOT+1 > TOT
00B8  6820                    LE      2,D2              D2 > R2
       084E
00BC  0312                    BFCR    1,ERR             CC = 1
```

```
00BE  02E2            BTCR  X'E',ERR
00C0  41F0            BAL   15,COMP2
        07A0
00C4  0852            DC    R2              R2 NORMALIZED
                  *
00C6  0A31            AHR   TOT,ONE         BUMP ERR COUNT
00C8  6830            LE    3,D3
        0856
00CC  0322            BFCR  2,ERR           CC =2
00CE  02D2            BTCR  X'D',ERR
00D0  41F0            BAL   15,COMP2
        07A0
00D4  085A            DC    R3              NORMALIZED
00D6  2802            LER   0,2             R2& R3 TO R0&R1
00D8  0322            BFCR  2,ERR           CC =2
00DA  02D2            BTCR  X'D',ERR
00DC  41F0            BAL   15,COMP0
        0798
00E0  085A            DC    R3              NORMALIZED
                  *
00E2  0A31            AHR   TOT,ONE         TOT = 4
00E4  6840            LE    4,D4
        085F
00E8  0322            BFCR  2,ERR           CC =2
00EA  02D2            BTCR  X'D',ERR
00EC  41F0            BAL   15,COMP4
        07A8
00F0  0862            DC    R4              UNNORMALIZED
                  *
00F2  0A31            AHR   TOT,ONE         TOT =5
00F4  6850            LE    5,D5            DATA5 TO R4 & R5
        0866
00F8  0322            BFCR  2,ERR           CC =2
00FA  02D2            BTCR  X'D',ERR
00FC  41F0            BAL   15,COMP4
        07A8
0100  086A            DC    R5
                  *
0102  0A31            AHR   TOT,ONE         TOT =6
0104  6860            LE    6,D6
        086E
0108  0312            BFCR  1,ERR           CC =1
010A  02E2            BTCR  X'E',ERR
010C  41F0            BAL   15,COMP6
        07B0
0110  0872            DC    R6              UNNORMALIZED
0112  2857            LER   5,7             R6&R7 TO R4&R5
0114  0312            BFCR  1,ERR           CC =1
0116  02E2            BTCR  X'E',ERR
0118  41F0            BAL   15,COMP4
        07A8
011C  0872            DC    R6              NORMALIZED
                  *
011E  0A31            AHR   TOT,ONE         TOT =7
0120  6870            LE    7,D7            D7 TO R6&R7
        0876
0124  0322            BFCR  2,ERR           CC =2
0126  02D2            BTCR  X'D',ERR
0128  41F0            BAL   15,COMP6
        07B0
012C  087A            DC    R7              UNNORMALIZED
                  *
```

```
012E  0A31          AHR   TOT,ONE       TOT =8
0130  6880          LE    8,D8
       087E
0134  0312          BFCR  1,ERR         CC =1
0136  02E2          BTCR  X'E',ERR
0138  41F0          BAL   15,COMP8
       07B8
013C  0882          DC    R8
              *
013E  0A31          AHR   TOT,ONE       TOT =9
0140  6890          LE    9,D9
       0886
0144  02F2          BTCR  X'F',ERR      CC = 0
0146  41F0          BAL   15,COMP8      POSITIVE
       07B8
014A  088A          DC    R9            ILLEGAL ZERO
              *
014C  0A31          AHR   TOT,ONE       TOT =A
014E  68A0          LE    10,D10
       088E
0152  02F2          BTCR  X'F',ERR      CC = 0
0154  41F0          BAL   15,COMP10     NEGATIVE
       07C0
0158  0892          DC    R10           ILLEGAL ZERO
              *
015A  0A31          AHR   TOT,ONE       TOT =B
015C  68B0          LE    11,D11
       0896
0160  0342          BFCR  4,ERR         CC = 4
0162  02B2          BTCR  X'B',ERR
0164  41F0          BAL   15,COMP10     POSITIVE
       07C0
0168  089A          DC    R11           UNDERFLOW
              *
016A  0A31          AHR   TOT,ONE       TOT =C
016C  68C0          LE    12,D12
       089E
0170  0342          BFCR  4,ERR         CC =4
0172  02B2          BTCR  X'B',ERR
0174  41F0          BAL   15,COMP12     POSITIVE
       07C8
0178  08A2          DC    R12           UNDER FLOW
              *
017A  0A31          AHR   TOT,ONE       TOT =D
017C  68D0          LE    13,D13
       08A6
0180  0342          BFCR  4,ERR         CC =4
0182  02B2          BTCR  X'B',ERR
0184  41F0          BAL   15,COMP12     POSITIVE
       07C8
0188  08AA          DC    R13           UNDERFLOW
018A  28DD          LER   13,13
018C  02F2          BTCR  X'F',ERR      CC =0
018E  41F0          BAL   15,COMP12
       07C8
0192  08AA          DC    R13
              *
0194  0A31          AHR   TOT,ONE       TOT =E
0196  68E0          LE    14,D14
       08AE
019A  0342          BFCR  4,ERR         CC =4
019C  02B2          BTCR  X'B',ERR
```

```
019E  41F0              BAL   15,COMP14      NEGATIVE
        07D0
01A2  08B2              DC    R14            UNDERFLOW
               *
01A4  0A31              AHR   TOT,ONE        TOT =F
01A6  68F0              LE    15,D15
        08B6
01AA  0342              BFCR  4,ERR          CC =4
01AC  02B2              BTCR  X'B',ERR
01AE  41F0              BAL   15,COMP14      NEGATIVE
        07D0
01B2  08BA              DC    R15            UNDERFLOW
               *
01B4  0A31              AHR   TOT,ONE        TOT =10
01B6  68E0              LE    14,D16
        08BE
01BA  0342              BFCR  4,ERR          CC =4
01BC  02B2              BTCR  X'B',ERR
01BE  41F0              BAL   15,COMP14      NEGATIVE
        07D0
01C2  08C2              DC    R16            UNDERFLOW
               *
01C4  4300              B     MATH
        01C8
               *
01C8  C830     MATH     LHI   TOT,X'20'      SET ERR TOTAL TO 20, HEX.
        0020
01CC  6800              LE    0,A0           DATA0 > R0&R1
        08C6
01D0  6820              LE    2,A1           DATA 1 > R2&R3
        08CA
01D4  2A02              AER   0,2            DATA 0 + DATA 1 > R0&R1
01D6  0322              BFCR  2,ERR          CC =2, GREATER THAN ZERO
01D8  02D2              BTCR  X'D',ERR
01DA  41F0              BAL   15,COMP0       DO A FIX POINT COMPAR
        0798
01DE  091E              DC    E0             ADDRESS OF CORRECT ANS.
01E0  6A20              AE    2,A0           DATA0 + DATA 1 > R2&R3
        08C6
01E4  0322              BFCR  2,ERR          CC =2
01E6  02D2              BTCR  X'D',ERR
01E8  41F0              BAL   15,COMP2
        07A0
01EC  091E              DC    E0
               *
01EE  0A31              AHR   TOT,ONE        TOT =21
01F0  6820              LE    2,A2
        08CE
01F4  6840              LE    4,A3
        08D2
01F8  2A24              AER   2,4            SUM > R2&R3
01FA  0312              BFCR  1,ERR          CC =1, LESS THAN ZERO
01FC  02E2              BTCR  X'E',ERR
01FE  41F0              BAL   15,COMP2
        07A0
0202  0922              DC    E1
0204  6A40              AE    4,A2
        08CE
0208  0312              BFCR  1,ERR
020A  02E2              BTCR  X'E',ERR
020C  41F0              BAL   15,COMP4
        07A8
```

```
0210   0922              DC     E1
                    *
0212   0A31              AHR    TOT,ONE          TOT =22
0214   6840              LE     4,A4
        08D6
0218   6860              LE     6,A5
        08DA
021C   2A46              AER    4,6
021E   0322              BFCR   2,ERR            CC =2
0220   02D2              BTCR   X'D',ERR
0222   41F0              BAL    15,COMP4
        07A8
0226   0926              DC     E2
0228   6A60              AE     6,A4
        08D6
022C   0322              BFCR   2,ERR            CC =
022E   02D2              BTCR   X'D',ERR
0230   41F0              BAL    15,COMP6
        07B0
0234   0926              DC     E2
                    *
0236   0A31              AHR    TOT,ONE          TOT =23
0238   6860              LE     6,A6
        08DE
023C   6880              LE     8,A7
        08E2
0240   2A68              AER    6,8
0242   0312              BFCR   1,ERR            CC =1
0244   02E2              BTCR   X'E',ERR
0246   41F0              BAL    15,COMP6
        07B0
024A   092A              DC     E3
024C   6A80              AE     8,A6
        08DE
0250   0312              BFCR   1,ERR            CC =1
0252   02E2              BTCR   X'E',ERR
0254   41F0              BAL    15,COMP8
        07B8
0258   092A              DC     E3
                    *
025A   0A31              AHR    TOT,ONE          TOT = 24
025C   68A0              LE     10,A8
        08E6
0260   6AA0              AE     10,A9
        08EA
0264   0322              BFCR   2,ERR            CC =2
0266   02D2              BTCR   X'D',ERR
0268   41F0              BAL    15,COMP10
        07C0
026C   092E              DC     E4
                    *
026E   0A31              AHR    TOT,ONE          TOT =25
0270   68C0              LE     12,A2
        08CE
0274   6AC0              AE     12,A9
        08EA
0278   0312              BFCR   1,ERR            CC =1
027A   02E2              BTCR   X'E',ERR
027C   41F0              BAL    15,COMP12
        07C8
0280   0932              DC     E5
                    *
```

```
0282  0A31        AHR   TOT,ONE       TOT =26
0284  68E0        LE    14,A4
       08D6
0288  6AE0        AE    14,A7
       08E2
028C  0312        BFCR  1,ERR         CC =1
028E  02E2        BTCR  X'E',ERR
0290  41F0        BAL   15,COMP14
       07D0
0294  0932        DC    E5
          *
0296  0A31        AHR   TOT,ONE       TOT =27
0298  6880        LE    8,A8
       08E6
029C  2A88        AER   8,8
029E  0322        BFCR  2,ERR         CC=2
02A0  02D2        BTCR  X'D',ERR
02A2  41F0        BAL   15,COMP8
       07B8
02A6  0936        DC    E6
          *
02A8  0A31        AHR   TOT,ONE       TOT =28
02AA  6860        LE    6,A8
       08E6
02AE  6A60        AE    6,A8
       08E6
02B2  0322        BFCR  2,ERR         CC =2
02B4  02D2        BTCR  X'D',ERR
02B6  41F0        BAL   15,COMP6
       07B0
02BA  0936        DC    E6
          *
02BC  0A31        AHR   TOT,ONE       TOT =29
02BE  6840        LE    4,A10
       08EE
02C2  2A44        AER   4,4           OVERFLOW (+)
02C4  0362        BFCR  6,ERR         CC=6
02C6  0292        BTCR  9,ERR
02C8  41F0        BAL   15,COMP4
       07A8
02CC  093A        DC    E7
02CE  6840        LE    4,A10
       08EE
02D2  6A40        AE    4,A10         OVERFLOW (+)
       08EE
02D6  0362        BFCR  6,ERR
02D8  0292        BTCR  9,ERR
02DA  41F0        BAL   15,COMP4
       07A8
02DE  093A        DC    E7
          *
02E0  0A31        AHR   TOT,ONE       TOT =2A
02E2  6820        LE    2,A12
       08F2
02E6  2A22        AER   2,2           OVERFLOW (-)
02E8  0352        BFCR  5,ERR         CC=5
02EA  02A2        BTCR  X'A',ERR
02EC  41F0        BAL   15,COMP2
       07A0
02F0  093E        DC    E8
02F2  6820        LE    2,A12
       08F2
```

```
02F6   6A20          AE     2,A12          OVERFLOW (-)
        08F2
02FA   0352          BFCR   5,ERR          CC=5
02FC   02A2          BTCR   X'A',ERR
02FE   41F0          BAL    15,COMP2
        07A0
0302   093E          DC     E8
               *
0304   0A31          AHR    TOT,ONE        TOT =2B
0306   6800          LE     0,A14
        08F6
030A   6820          LE     2,A15
        08FA
030E   2A02          AER    0,2            UNDERFLOW
0310   0232          BTCR   3,ERR          CC=0
0312   41F0          BAL    15,COMP0
        0798
0316   0942          DC     E9
0318   6820          LE     2,A15
        08FA
031C   6A20          AE     2,A14          UNDERFLOW
        08F6
0320   0232          BTCR   3,ERR          CC=0
0322   41F0          BAL    15,COMP2
        07A0
0326   0942          DC     E9
               *
0328   0A31          AHR    TOT,ONE        TOT = 2C
032A   6840          LE     4,A16
        08FF
032E   6860          LE     6,A17
        0902
0332   2A46          AER    4,6            SUM =0
0334   0232          BTCR   3,ERR          CC=0
0336   41F0          BAL    15,COMP4
        07A8
033A   0942          DC     E9
033C   6840          LE     4,A16
        08FE
0340   2A64          AER    6,4            SUM =0
0342   0232          BTCR   3,ERR          CC=0
0344   41F0          BAL    15,COMP6
        07B0
0348   0942          DC     E9
               *
034A   0A31          AHR    TOT,ONE        TOT =2D
034C   6880          LE     8,A18
        0906
0350   6A80          AE     8,A19          ESTABLISH BOUNDRIES
        090A
0354   0322          BFCR   2,ERR          CC=2
0356   02D2          BTCR   X'D',ERR
0358   41F0          BAL    15,COMP8
        07B8
035C   0946          DC     E10
035E   6880          LE     8,A19
        090A
0362   6A80          AE     8,A18          ESTABLISH BOUNDRIES
        0906
0366   0322          BFCR   2,ERR          CC =2
0368   02D2          BTCR   X'D',ERR
036A   41F0          BAL    15,COMP8
```

```
      07B8
036E  0946              DC    E10
            *
0370  0A31              AHR   ITCT,ONE          TOT =2E
0372  68A0              LE    10,A22
      090E
0376  6AA0              AE    10,A23
      0912
037A  0322              BFCR  2,ERR             CC=2
037C  02D2              BTCR  X'D',ERR
037E  41F0              BAL   15,COMP10
      07C0
0382  094A              DC    E12
            *
0384  0A31              AHR   TOT,ONE           TOT =2F
0386  68C0              LE    12,A24
      0916
038A  6AC0              AE    12,A25
      091A
038E  41F0              BAL   15,COMP12
      07C8
0392  094E              DC    E13
            *
0394  4300              B     MATH1
      0398
            *
            *
0398  0A31        MATH1 AHR   TOT,ONE           TOT=30 FOR SUBTRACT
039A  6800              LE    0,T0              ROUTINE
      0952
039E  6820              LE    2,T1
      0956
03A2  2B02              SER   0,2
03A4  0322              BFCR  2,ERR
03A6  02D2              BTCR  X'D',ERR
03A8  41F0              BAL   15,COMP0
      0798
03AC  096A              DC    S0
03AE  6B20              SE    2,T0
      0952
03B2  0312              BFCR  1,ERR
03B4  02E2              BTCR  X'E',ERR
03B6  41F0              BAL   15,COMP2
      07A0
03BA  096E              DC    S1
            *
03BC  0A31              AHR   TOT,ONE           TOT =31
03BE  6840              LE    4,T2
      095A
03C2  6860              LE    6,T3
      095E
03C6  2B46              SER   4,6
03C8  0322              BFCR  2,ERR
03CA  02D2              BTCR  X'D',ERR
03CC  41F0              BAL   15,COMP4
      07A8
03D0  0972              DC    S2
03D2  6B60              SE    6,T2
      095A
03D6  0312              BFCR  1,ERR
03D8  02E2              BTCR  X'E',ERR
03DA  41F0              BAL   15,COMP6
```

```
        07B0
03DE    0976            DC      S3
                *
03E0    0A31            AHR     TOT,ONE         TOT =32
03E2    6880            LE      8,T4
        0962
03E6    68A0            LE      10,T5
        0966
03EA    2B8A            SER     8,10
03EC    0322            BFCR    2,ERR
03EE    02D2            BTCR    X'D',ERR
03F0    41F0            BAL     15,COMP8
        07B8
03F4    0972            DC      S2
03F6    6BA0            SE      10,T4
        0962
03FA    0312            BFCR    1,ERR
03FC    02E2            BTCR    X'E',ERR
03FE    41F0            BAL     15,COMP10
        07C0
0402    0976            DC      S3
                *
0404    0A31            AHR     TOT,ONE         TOT =33
0406    68C0            LE      12,T3
        095E
040A    68E0            LE      14,T5
        0966
040E    2BCE            SER     12,14
0410    0322            BFCR    2,ERR
0412    02D2            BTCR    X'D',ERR
0414    41F0            BAL     15,COMP12
        07C8
0418    097A            DC      S4
041A    6BE0            SE      14,T3
        095E
041E    0312            BFCR    1,ERR
0420    02E2            BTCR    X'E',ERR
0422    41F0            BAL     15,COMP14
        07D0
0426    097E            DC      S5
                *
0428    0A31            AHR     TOT,ONE         TOT =34
042A    68A0            LE      10,T1
        0956
042E    6880            LE      8,T1
        0956
0432    2BAA            SER     10,10
0434    0232            BTCR    3,ERR           CC=0
0436    41F0            BAL     15,COMP10
        07C0
043A    0982            DC      S6
043C    6B80            SE      8,T1
        0956
0440    0232            BTCR    3,ERR           CC=0
0442    41F0            BAL     15,COMP8
        07B8
0446    0982            DC      S6
                *
0448    4300            B       MATH2
        044C
                *
                *
```

```
044C  C830      MATH2   LHI     TOT,X'40'         MULTIPLY ROUTINE
      0040
0450  6800              LE      0,M0
      0986
0454  6820              LE      2,M1
      098A
0458  2C02              MER     0,2
045A  0362              BFCR    6,ERR             CC=6
045C  0292              BTCR    9,ERR             OVERFLOW (+)
045E  41F0              BAL     15,COMP0          FIXED POINT COMPAR
      0798
0462  09F2              DC      P1                ADRS. OF THE PRODUCT
                *
0464  0A31              AHR     TOT,ONE           TOT=41
0466  6800              LE      0,M0
      0986
046A  6C00              ME      0,M2
      098E
046E  0352              BFCR    5,ERR             CC=5
0470  02A2              BTCR    X'A',ERR          OVERFLOW (-)
0472  41F0              BAL     15,COMP0
      0798
0476  09F6              DC      P2
                *
0478  0A31              AHR     TOT,ONE           TOT=42
047A  6820              LE      2,M3
      0992
047E  6C20              ME      2,M2
      098E
0482  0362              BFCR    6,ERR             CC=6
0484  0292              BTCR    9,ERR             OVERFLOW (+)
0486  41F0              BAL     15,COMP2
      07A0
048A  09F2              DC      P1
                *
048C  0A31              AHR     TOT,ONE           TOT=43
048E  6840              LE      4,M4
      0996
0492  6860              LE      6,M5              ROUNDING CHECK
      099A
0496  2C46              MER     4,6
0498  0322              BFCR    2,ERR             CC=2
049A  02D2              BTCR    X'D',ERR
049C  41F0              BAL     15,COMP4
      07A8
04A0  09FA              DC      P3
                *
04A2  0A31              AHR     TOT,ONE           TOT=44
04A4  6860              LE      6,M6
      099E
04A8  6C60              ME      6,M7
      09A2
04AC  0322              BFCR    2,ERR             CC=2
04AE  02D2              BTCR    X'D',ERR          ROUNDING CHECK
04B0  41F0              BAL     15,COMP6
      07B0
04B4  09FE              DC      P4
                *
04B6  0A31              AHR     TOT,ONE           TOT=45
04B8  6880              LE      8,M8
      09A6
04BC  6C80              ME      8,M9
```

```
         09AA
04C0     0312              BFCR   1,ERR             CC=1
04C2     02E2              BTCR   X'E',ERR
04C4     41F0              BAL    15,COMP8
         07B8
04C8     0A02              DC     P5
                   *
04CA     0A31              AHR    TOT,ONE           TOT=46
04CC     68A0              LE     10,M10
         09AE
04D0     6CA0              ME     10,M9
         09AA
04D4     0312              BFCR   1,ERR             CC=1
04D6     02E2              BTCR   X'E',ERR          NEGATIVE PRODUCT
04D8     41F0              BAL    15,COMP10
         07C0
04DC     0A06              DC     P6
                   *
04DE     0A31              AHR    TOT,ONE           TOT=47
04E0     68C0              LE     12,M12
         09B6
04E4     6CC0              ME     12,M11
         09B2
04E8     0322              BFCR   2,ERR             CC=2
04EA     02D2              BTCR   X'D',ERR          POSITIVE PRODUCT
04EC     41F0              BAL    15,COMP12
         07C8
04F0     0A0A              DC     P7
                   *
04F2     0A31              AHR    TOT,ONE           TOT=48
04F4     68E0              LE     14,M14
         09BE
04F8     6CE0              ME     14,M13
         09BA
04FC     0342              BFCR   4,ERR             CC=4
04FE     02B2              BTCR   X'B',ERR          UNNORMALIZED UNDERFLOW
0500     41F0              BAL    15,COMP14
         07D0
0504     0A0E              DC     P8
                   *
0506     0A31              AHR    TOT,ONE           TOT=49
0508     68C0              LE     12,M15
         09C2
050C     68E0              LE     14,M16
         09C6
0510     2CCE              MER    12,14
0512     0342              BFCR   4,ERR             CC=4
0514     02B2              BTCR   X'B',ERR
0516     41F0              BAL    15,COMP12         ZERO PRODUCT
         07C8
051A     0A0E              DC     P8
                   *
051C     0A31              AHR    TOT,ONE           TOT=4A
051E     68E0              LE     14,M18
         09CE
0522     6CE0              ME     14,M17
         09CA
0526     02F2              BTCR   X'F',ERR          CC=0
0528     41F0              BAL    15,COMP14         ZERO PRODUCT
         07D0
052C     0A0E              DC     P8
                   *
```

```
052E   0A31          AHR    TOT,ONE        TOT=4B
0530   6800          LE     0,M19
        09D2
0534   6C00          ME     0,M20
        09D6
0538   0312          BFCR   1,ERR          CC=1
053A   02E2          BTCR   X'E',ERR       NEGATIVE PRODUCT
053C   41F0          BAL    15,COMPO
       0798
0540   0A12          DC     P9
                *
0542   0A31          AHR    ITCT,ONE       TOT=4C
0544   6820          LE     2,M21
        09DA
0548   6C20          MF     2,M22
        09DE
054C   0312          BFCR   1,ERR          CC=1
054E   02E2          BTCR   X'E',ERR       NEGATIVE PRODUCT
0550   41F0          BAL    15,COMP2
        07A0
0554   0A16          DC     P10
                *
0556   0A31          AHR    TOT,ONE        TOT=4D
0558   6C00          ME     0,M21
        09DA
055C   0322          BFCR   2,ERR          CC=2
055E   02D2          BTCR   X'D',ERR       POSITIVE PRODUCT
0560   41F0          BAL    15,COMPO
        0798
0564   0A1A          DC     P11
                *
0566   0A31          AHR    ITCT,ONE       TOT=4E
0568   6C20          ME     2,M20
        09D6
056C   0312          BFCR   1,ERR          CC=1
056E   02E2          BTCR   X'E',ERR       NEGATIVE PRODUCT
0570   41F0          BAL    15,COMP2
        07A0
0574   0A1E          DC     P12
                *
0576   0A31          AHR    TOT,ONE        TOT=4F
0578   6C00          ME     0,M22
        09DF
057C   0322          BFCR   2,ERR          CC=2
057E   02D2          BTCR   X'D',ERR       POS. PRO.
0580   41F0          BAL    15,COMPO
        0798
0584   0A22          DC     P13
                *
0586   0A31          AHR    TOT,ONE        TOT=50
0588   6C20          ME     2,M19
        09D2
058C   0322          BFCR   2,ERR          CC=2
058E   02D2          BTCR   X'D',ERR
0590   41F0          BAL    15,COMP2
        07A0
0594   0A22          DC     P13
0596   2B02          SER    0,2
0598   0232          BTCR   3,ERR          CC=0
                *
059A   0A31          AHR    ITOT,ONE       TOT=51
059C   6840          LE     4,M23
```

AP 12

```
         09E2
05A0   6C40                 ME      4,M24
         09E6
05A4   0322                 BFCR    2,ERR             CC=2
05A6   02D2                 BTCR    X'D',ERR          POS. PRO. FOR CROSS MLT.
05A8   41F0                 BAL     15,COMP4
         07A8
05AC   0A26                 DC      P14
05AE   0A31                 AHR     TOT,ONE           TOT=52
05B0   6840                 LE      4,M25
         09EA
05B4   6C40                 ME      4,M26             UNNORMALIZED/ROUNDING
         09EE
05B8   0322                 BFCR    2,ERR
05BA   02D2                 BTCR    X'D',ERR
05BC   41F0                 BAL     15,COMP4
         07A8
05C0   0A2A                 DC      P15
                  *
05C2   4300                 B       MATH3
         05C6
                  *
                  *
05C6   C830       MATH3     LHI     TOT,X'60'         SET TOT TO 60
         0060
05CA   6800                 LE      0,B0
         0A2E
05CE   6820                 LE      2,B1
         0A32
05D2   2D02                 DER     0,2               +/+
05D4   0322                 BFCR    2,ERR             CC=2
05D6   02D2                 BTCR    X'D',ERR
05D8   41F0                 BAL     15,COMP0
         0798
05DC   0A6E                 DC      C0
                  *
05DE   0A31                 AHR     TOT,ONE           TOT=61
05E0   6840                 LE      4,B3
         0A3A
05E4   6860                 LE      6,B2
         0A36
05E8   2D64                 DER     6,4               -/+
05EA   0312                 BFCR    1,ERR             CC=1
05EC   02E2                 BTCR    X'E',ERR
05EE   41F0                 BAL     15,COMP6
         07B0
05F2   0A72                 DC      C1
                  *
05F4   0A31                 AHR     TOT,ONE           TOT=62
05F6   6880                 LE      8,B4
         0A3E
05FA   6D80                 DE      8,B5              +/-
         0A42
05FE   0312                 BFCR    1,ERR             CC=1
0600   02E2                 BTCR    X'E',ERR
0602   41F0                 BAL     15,COMP8
         07B8
0606   0A76                 DC      C2
                  *
0608   0A31                 AHR     TOT,ONE           TOT=63
060A   68A0                 LE      10,B6
         0A46
```

```
060E  6DA0            DE    10,B7         -/-
        0A4A
0612  0322            BFCR  2,ERR         CC=2
0614  02D2            BTCR  X'D',ERR
0616  41F0            BAL   15,COMP10
        07C0
061A  0A7A            DC    C3
                *
061C  0A31            AHR   TOT,ONE       TOT=64
061E  68C0            LE    12,B1
        0A32
0622  6DC0            DE    12,B2         +A7-A
        0A36
0626  0312            BFCR  1,ERR         CC=1
0628  02E2            BTCR  X'E',ERR
062A  41F0            BAL   15,COMP12
        07C8
062E  0A7E            DC    C4
                *
0630  0A31            AHR   TOT,ONE       TOT=65
0632  68E0            LE    14,B8
        0A4E
0636  68C0            LE    12,B9
        0A52
063A  2DEC            DER   14,12         + OVERFLOW
063C  0362            BFCR  6,ERR         CC=6
063E  0292            BTCR  9,ERR
0640  41F0            BAL   15,COMP14
        07D0
0644  0A82            DC    C5
                *
0646  0A31            AHR   TOT,ONE       TOT=66
0648  6830            LE    3,B10
        0A56
064C  6850            LE    5,B9
        0A52
0650  2D24            DER   2,4           - OVERFLOW
0652  0352            BFCR  5,ERR         CC=5
0654  02A2            BTCR  X'A',ERR
0656  41F0            BAL   15,COMP2
        07A0
065A  0A86            DC    C6
                *
065C  0A31            AHR   TOT,ONE       TOT=67
065E  6860            LE    6,B10
        0A56
0662  6D60            DE    6,B11         + OVERFLOW
        0A5A
0666  0362            BFCR  6,ERR         CC=6
0668  0292            BTCR  9,ERR
066A  41F0            BAL   15,COMP6
        07B0
066E  0A82            DC    C5
                *
0670  0A31            AHR   TOT,ONE       TOT=68
0672  6880            LE    8,B12
        0A5E
0676  6D80            DE    8,B13         UNDERFLOW
        0A62
067A  0342            BFCR  4,ERR         CC=4
067C  02B2            BTCR  X'B',ERR
067E  41F0            BAL   15,COMP8
```

```
        07B8
0682    0A8A                DC      C7
                 *
0684    0A31                AHR     TOT,ONE           TOT=69
0686    68A0                LE      10,B14
         0A66
068A    68C0                LE      12,B15
         0A6A
068E    2DAC                DER     10,12             NORMALIZED UNDERFLOW
0690    0342                BFCR    4,ERR             CC=4
0692    02B2                BTCR    X'B',ERR
0694    41F0                BAL     15,COMP10
         07C0
0698    0A8A                DC      C7
                 *
069A    0A31                AHR     TOT,ONE           TOT=6A
069C    C200                LPSW    BIT5UP            BIT 5 WILL BE SET
         0836
06A0    C8A0     DVD        LHI     HELP,DVDFIA
         06BA
06A4    40A0                STH     HELP,X'2E'        FAULT INTERRUPT ADRS
         002E
06A8    07AA                XHR     HELP,HELP         ZERO OTHER BITS OF
06AA    40A0                STH     HELP,X'2C'        NEW PSW DFI
         002C
06AE    6860                LE      6,B13
         0A62
06B2    6880                LE      8,C7              C7 IS 0
         0A8A
06B6    2D68                DER     6,8
06B8    0302                BFCR    0,ERR
06BA    41F0     DVDFIA     BAL     15,COMP6          OPERANDS SHOULD NOT
         07B0
06BE    0A62                DC      B13               HAVE BEEN CHANGED
06C0    2888                LER     8,8               SHOULD BE ZERO
06C2    0232                BTCR    3,ERR             SHOULD BE ZERO
06C4    C200                LPSW    BIT5NO            RESET BIT5 OF PSW
         083A
06C8    4020     DVD1       STH     ERR,X'2E'         STOR ERR ADRS IN DFI
         002E
06CC    2D68                DER     6,8               A/0
06CE    41F0                BAL     15,COMP6          NO CHANGE IN OPERANDS
         07B0
06D2    0A62                DC      B13
06D4    2888                LER     8,8               R8=0
06D6    0232                BTCR    3,ERR
                 *
06D8    4300                B       CECER             FLOATING PT. COMPARE
         06DC
                 *
06DC    C830     CECER      LHI     TOT,X'70'         TOT=70
         0070
06E0    6800                LE      0,F0
         0A8E
06E4    6820                LE      2,F1
         0A92
06E8    2920                CER     2,0               2>0
06EA    0322                BFCR    2,ERR             CC=2
06EC    02D2                BTCR    X'D',ERR
06EE    2902                CER     0,2               0?2
06F0    0392                BFCR    9,ERR             CC=9
06F2    0262                BTCR    6,ERR
```

```
06F4  0A31           AHR   TOT,ONE              TOT=71
06F6  6840           LE    4,F2
      0A96
06FA  6940           CE    4,F1                 4>F1
      0A92
06FE  0392           BFCR  9,ERR                CC=9
0700  0262           BTCR  6,ERR
                *
0702  0A31           AHR   TOT,ONE              TOT=72
0704  6860           LE    6,F3
      0A9A
0708  2966           CER   6,6                  6=6
070A  02F2           BTCR  X'F',ERR             CC=0
                *
070C  0A31           AHR   TOT,ONE              TOT =73
070E  6960           CE    6,F4
      0A9E
0712  0322           BFCR  2,ERR                CC=2
0714  02D2           BTCR  X'D',ERR
0716  6880           LE    8,F4
      0A9E
071A  2986           CER   8,6                  8>6
071C  0392           BFCR  9,ERR                CC=9
                *
071E  0A31           AHR   TOT,ONE              TOT=74
0720  68A0           LE    10,F5
      0AA2
0724  68C0           LE    12,F6
      0AA6
0728  29AC           CER   10,12                10=12
072A  02F2           BTCR  X'F',ERR             CC=0
072C  29CA           CER   12,10                12=10
072E  02F2           BTCR  X'F',ERR             CC=0
0730  69A0           CE    10,F6
      0AA6
0734  0322           BFCR  2,ERR
0736  02D2           BTCR  X'D',ERR
0738  69C0           CE    12,F5
      0AA2
073C  0392           BFCR  9,ERR
073E  29AA           CER   10,10
0740  0200           NOPR
0742  0262           BTCR  6,ERR
0744  02F2           BTCR  X'F',ERR             CC=0
                * "AOK" OUTPUT ROUTINE
                *
0746  C860           LHI   LIMIT,MESN1-MESB1   HOW MANY BYTES?
      001F
074A  0744           XHR   INDEX,INDEX          0 INDEX
074C  0851           LHR   INCR,ONE             1 INCR
074E  D380           LB    DEV,LISTDV           SELECT DEVICE
      007E
0752  DE80           OC    DEV,LISTDV+1         OUTPUT COMMAND
      007F
0756  9D89      SENSE  SSR   DEV,STATUS
0758  42F0           BTC   X'F',SENSE
      0756
075C  D374      LB1  LB    OUT,MESB1(INDEX)   LOAD BYTES INTO OUT
      07F8
0760  9A37           WDR   DEV,OUT

0762  C140           BXLE  INDEX,SENSE
```

```
      0756
0766  C200                LPSW   STOP             HLT.
      076A
076A  8000    STOP        DC     X'8000',X'80'
      0080
              *
              * ERROR OUTPUT ROUTINE
              *
076E  C860    ERROR       LHI    LIMIT,MESN2-MESB2   HOW MANY BYTES?
      0018
0772  0744                XHR    INDEX,INDEX      ZERO INDEX REG.
0774  0851                LHR    INCR,ONE         1 INCR
0776  D380                LB     DEV,LISTDV       SELECT DEVICE
      007E
077A  DE80                OC     DEV,LISTDV+1     OUTPUT COMMAND
      007F
077E  9D89    SEN         SSR    DEV,STATUS
0780  42F0                BTC    X'F',SEN
      077E
0784  D374    LB2         LB     OUT,MESB2(INDEX)   LOAD BYTES INTO OUT
      0818
0788  9A87                WDR    DEV,OUT
078A  C140                BXLE   INDEX,SEN
      077E
078E  0873                LHR    OUT,TOT          OUTPUT TEST NUMBER
0790  0881                LHR    DEV,ONE          ADRS. DISPLAY REGISTER
0792  9A87                WDR    DEV,OUT          THAT FAILED
0794  C200                LPSW   STOP             DEPRESS EX. TO BEGIN OVER
      076A
              *
              *
0798  6000    COMP0       STE    0,COM            THIS ROUTINE WILL STORE
      0832
079C  4300                B      COMPAR           THE 32 BIT RESULT
      07D8
              *                                   IN 4 SUCCESSIVE
07A0  6020    COMP2       STE    2,COM            BYTES IN MEMORY
      0832
07A4  4300                B      COMPAR
      07D8
              *
07A8  6040    COMP4       STE    4,COM
      0832
07AC  4300                B      COMPAR
      07D8
              *
07B0  6060    COMP6       STE    6,COM
      0832
07B4  4300                B      COMPAR
      07D8
              *
07B8  6080    COMP8       STE    8,COM
      0832
07BC  4300                B      COMPAR
      07D8
              *
07C0  60A0    COMP10      STE    10,COM
      0832
07C4  4300                B      COMPAR
      07D8
              *
07C8  60C0    COMP12      STE    12,COM
```

```
        0832
07CC    4300            B       COMPAR
        07D8
                *
07D0    60E0    COMP14  STE     14,COM
        0832
07D4    4300            B       COMPAR
        07D8
                *
07D8    48EF    COMPAR  LH      14,0(15)        C(R15) R14
        0000
07DC    48CE            LH      12,0(14)        C(R14) R12
        0000
07E0    48DE            LH      13,2(14)        C(R14+2) R13
        0002
07E4    45C0            CLH     12,COM          C(R12):C(COM)
        0832
07E8    4230            BNE     ERROR
        076E
07EC    45D0            CLH     13,COM+2        C(R13):C(COM+2)
        0834
07F0    4230            BNE     ERROR
        076E
07F4    430F            B       2(15)
        0002
                *
                *
07F8    0D0A    MESB1   DC      X'0D0A'
07FA    464C            DC      C'FLOATING      POINT OPERATIONS OK'
        4F41
        5449
        4E47
        2050
        4F49
        4E54
        204F
        5045
        5241
        5449
        4F4E
        5320
        4F4B
0816    2020    MESN1   DC      X'2020'
                *
0818    0D0A    MESB2   DC      X'0D0A'
081A    4641            DC      C'FAILURE       IN TEST NUMBER'
        494C
        5552
        4520
        494E
        2054
        4553
        5420
        4E55
        4D42
        4552
0830    2020    MESN2   DC      X'2020'
                *
007E            LISTDV  EQU     X'7E'
                *
0832            COM     DS      4
                *
```

A2-18

```
0836  0400     BIT5UP  DC   X'0400',A(DVD)  ENABLE DVD FAULT INTRUPT
       06A0
083A  0000     BIT5NO  DC   0,A(DVD1)       DISABLE DVD FLT. INTRUPT
       06C8
               *
               * DATA AND RESULT TABLE FOR LOAD AND STORE
               *
083E  0000     D0      DC   0,0             CONDITION CODE = 0
       0000
0842  0000     R0      DC   0,0             ZERO VALUE
       0000
0846  0010     D1      DC   X'0010',0       CC = 2
       0000
084A  0010     R1      DC   X'0010',0       POSITIVE NORMALIZED
       0000
084E  FF10     D2      DC   X'FF10',0       CC = 1
       0000
0852  FF10     R2      DC   X'FF10',0       NEGATIVE NORMALIZED
       0000
0856  7F10     D3      DC   X'7F10',0       CC = 2
       0000
085A  7F10     R3      DC   X'7F10',0       PN
       0000
085E  0101     D4      DC   X'0101',0       CC = 2
       0000
0862  0010     R4      DC   X'0010',0       POSITIVE UNNORMALIZED
       0000
0866  4200     D5      DC   X'4200',X'1000' CC = 2
       1000
086A  4010     R5      DC   X'4010',0       PUN
       0000
086E  F300     D6      DC   X'F300',X'01FF' CC = 1
       01FF
0872  F01F     R6      DC   X'F01F',X'F000' NEGATIVE UNNORMALIZED
       F000
0876  4400     D7      DC   X'4400',X'00F8' CC = 2
       00F8
087A  40F8     R7      DC   X'40F8',0       PUN
       0000
087E  C500     D8      DC   X'C500',1       CC = 1
       0001
0882  C010     R8      DC   X'C010',0       NUN
       0000
0886  4600     D9      DC   X'4600',0       CC = 0
       0000
088A  0000     R9      DC   0,0             POSITIVE ILLEGAL ZERO
       0000
088E  C600     D10     DC   X'C600',0       CC = 0
       0000
0892  0000     R10     DC   0,0             NEGATIVE ILLEGAL ZERO
       0000
0896  0001     D11     DC   1,0             CC = 4
       0000
089A  0000     R11     DC   0,0             POSITIVE UNDERFLOW
       0000
089E  0100     D12     DC   X'0100',X'1000' CC = 4
       1000
08A2  0000     R12     DC   0,0             PU
       0000
08A6  0300     D13     DC   X'0300',X'0010' CC = 4
       0010
08AA  0000     R13     DC   0,0             PU
```

```
         0000
08AE     8008     D14      DC     X'8008',0          CC = 4
         0000
08B2     0000     R14      DC     0,0                NEGATIVE UNDERFLOW
         0000
08B6     8200     D15      DC     X'8200',X'0800'    CC = 4
         0800
08BA     0000     R15      DC     0,0                NU
         0000
08BE     8400     D16      DC     X'8400',8          CC = 4
         0008
08C2     0000     R16      DC     0,0                NU
         0000
                  *
                  * DATA AND RESULT TABLE FOR ADD
                  *
08C6     0640     A0       DC     X'0640',X'00A0'    DATA TABLE
         00A0
08CA     0630     A1       DC     X'0630',X'00A1'
         00A1
08CE     C080     A2       DC     X'C080',X'00A2'
         00A2
08D2     4050     A3       DC     X'4050',X'00A3'
         00A3
08D6     C050     A4       DC     X'C050',X'00A4'
         00A4
08DA     4080     A5       DC     X'4080',X'00A5'
         00A5
08DE     4050     A6       DC     X'4050',X'00A6'
         00A6
08E2     C080     A7       DC     X'C080',X'00A7'
         00A7
08E6     4080     A8       DC     X'4080',X'00A8'
         00A8
08EA     C050     A9       DC     X'C050',X'00A9'
         00A9
08EE     7F80     A10      DC     X'7F80',0
         0000
08F2     FF80     A12      DC     X'FF80',0
         0000
08F6     0080     A14      DC     X'0080',1
         0001
08FA     8080     A15      DC     X'8080',0
         0000
08FE     0580     A16      DC     X'0580',0
         0000
0902     8580     A17      DC     X'8580',0
         0000
0906     0680     A18      DC     X'0680',0
         0000
090A     0180     A19      DC     X'0180',0
         0000
090E     05FF     A22      DC     X'05FF',X'FFFF'
         FFFF
0912     06F0     A23      DC     X'06F0',1
         0001
0916     0010     A24      DC     X'0010',X'8000'
         8000
091A     0010     A25      DC     X'0010',X'8000'
         8000
091E     0670     E0       DC     X'0670',X'0141'    SUM TABLE A0+A1
         0141
```

```
0922  C02F   E1      DC    X'C02F',X'FFFF'  A2+A3
       FFFF
0926  4030   E2      DC    X'4030',X'0001'  A4+A5
       0001
092A  C030   E3      DC    X'C030',X'0001'  A6+A7
       0001
092E  402F   E4      DC    X'402F',X'FFFF'  A8+A9
       FFFF
0932  C0D0   E5      DC    X'C0D0',X'014B'  A2+A9,A4+A7
       014B
0936  4110   E6      DC    X'4110',X'0015'  A8+A8
       0015
093A  7FFF   E7      DC    X'7FFF',X'FFFF'
       FFFF
093E  FFFF   E8      DC    X'FFFF',X'FFFF'  A12+A12
       FFFF
0942  0000   E9      DC    0,0              14+15,16+17
       0000
0946  0680   E10     DC    X'0680',X'0008'  A18+A19
       0008
094A  0710   E12     DC    X'0710',0        A22+A23
       0000
094E  0021   E13     DC    X'0021',0        A24+A25
       0000
             *
             * DATA AND RESULT TABLE FOR SUBTRACT
             *
0952  0640   T0      DC    X'0640',0        DATA TABLE
       0000
0956  0620   T1      DC    X'0620',0
       0000
095A  4080   T2      DC    X'4080',0
       0000
095E  C060   T3      DC    X'C060',0
       0000
0962  4060   T4      DC    X'4060',0
       0000
0966  C080   T5      DC    X'C080',0
       0000
096A  0620   S0      DC    X'0620,0         DIFFERENCE TABLE T0-T1
       0000
096E  8620   S1      DC    X'8620',0        T1-T0
       0000
0972  40E0   S2      DC    X'40E0',0        T2-T3,T4-T5
       0000
0976  C0E0   S3      DC    X'C0E0',0        T5-T4,T3-T2
       0000
097A  4020   S4      DC    X'4020',0        T3-T5
       0000
097E  C020   S5      DC    X'C020',0        T5-T3
       0000
0982  0000   S6      DC    0,0              T1-T1
       0000
             *
             * DATA AND RESULT TABLE FOR MULTIPLY
             *
0986  7F98   M0      DC    X'7F98',X'7654'  DATA TABLE
       7654
098A  4212   M1      DC    X'4212',X'3456'
       3456
098E  C212   M2      DC    X'C212',X'3456'
       3456
```

```
0992  FF98     M3     DC     X'FF98',X'7654'
      7654
0996  36FF     M4     DC     X'36FF',X'FFFF'
      FFFF
099A  2210     M5     DC     X'2210',0
      0000
099E  4711     M6     DC     X'4711',X'1111'
      1111
09A2  42F0     M7     DC     X'42F0',0
      0000
09A6  2210     M8     DC     X'2210',0
      0000
09AA  A080     M9     DC     X'A080',0
      0000
09AE  4020     M10    DC     X'4020',0
      0000
09B2  C060     M11    DC     X'C060',0
      0000
09B6  8440     M12    DC     X'8440',0
      0000
09BA  8210     M13    DC     X'8210',0
      0000
09BE  0220     M14    DC     X'0220',0
      0000
09C2  8662     M15    DC     X'8662',X'1058'
      1058
09C6  8497     M16    DC     X'8497',X'6543'
      6543
09CA  4010     M17    DC     X'4010',1
      0001
09CE  0200     M18    DC     X'0200',X'F'
      000F
09D2  C410     M19    DC     X'C410',0
      0000
09D6  4320     M20    DC     X'4320',0
      0000
09DA  8230     M21    DC     X'8230',0
      0000
09DE  4640     M22    DC     X'4640',0
      0000
09E2  4380     M23    DC     X'4380',X'8000'
      8000
09E6  4280     M24    DC     X'4280',X'8000'
      8000
09EA  3455     M25    DC     X'3455',X'5550'
      5550
09EE  2730     M26    DC     X'2730',X'0003'
      0003
               *
               * RESULT TABLE
09F2  7FFF     P1     DC     X'7FFF',X'FFFF'  + OVERFLOW 0*1
      FFFF
09F6  FFFF     P2     DC     X'FFFF',X'FFFF'  - OVERFLOW 0*2
      FFFF
09FA  17FF     P3     DC     X'17FF',X'FFFF'  ROUNDING PROBLEM 4*5
      FFFF
09FE  48FF     P4     DC     X'48FF',X'FFFF'  ROUNDING PROBLEM 6*7
      FFFF
0A02  8180     P5     DC     X'8180',0        + PRODUCT M8*M9
      0000
0A06  A010     P6     DC     X'A010',0        - PRODUCT M9*M10
      0000
```

```
0A0A  0418     P7     DC     X'0418',0        + PRODUCT M11*M12
       0000
0A0E  0000     P8     DC     0,0              UNDERFLOW 13*14,15*16,17*1
       0000
0A12  C620     P9     DC     X'C620',0        M19*M20
       0000
0A16  87C0     P10    DC     X'87C0',0        M21*M22
       0000
0A1A  0760     P11    DC     X'0760,0         P9*M21
       0000
0A1E  8A18     P12    DC     X'8A18',0        P10*M20
       0000
0A22  0D18     P13    DC     X'0D18',0        P11*M22,P12*M19
       0000
0A26  4540     P14    DC     X'4540',X'8040'  M23*M24, CROSS MULTIPLY
       8040
0A2A  1B10     P15    DC     X'1B10',0
       0000
               *
               * DATA AND RESULT TABLE FOR DIVIDE
               *
0A2E  2480     B0     DC     X'2480',0        DATA TABLE
       0000
0A32  2020     B1     DC     X'2020',0
       0000
0A36  A020     B2     DC     X'A020',0
       0000
0A3A  2480     B3     DC     X'2480',0
       0000
0A3E  2420     B4     DC     X'2420',0
       0000
0A42  A080     B5     DC     X'A080',0
       0000
0A46  A080     B6     DC     X'A080',0
       0000
0A4A  A420     B7     DC     X'A420',0
       0000
0A4E  4680     B8     DC     X'4680',0
       0000
0A52  0240     B9     DC     X'0240',0
       0000
0A56  C680     B10    DC     X'C680',0
       0000
0A5A  8240     B11    DC     X'8240',0
       0000
0A5E  0080     B12    DC     X'0080',0
       0000
0A62  7F80     B13    DC     X'7F80',0
       0000
0A66  0300     B14    DC     X'0300',X'1000'
       1000
0A6A  4240     B15    DC     X'4240,0
       0000
               *
0A6E  4540     C0     DC     X'4540',0        B0/B1
       0000
0A72  BC40     C1     DC     X'BC40',0        B2/B3
       0000
0A76  C440     C2     DC     X'C440',0        B4/B5
       0000
0A7A  3D40     C3     DC     X'3D40',0        B6/B7
       0000
```

```
0A7E  C110      C4      DC      X'C110',0          B1/B2
        0000
0A82  7FFF      C5      DC      X'7FFF',X'FFFF'    B8/B9,B10/B11
        FFFF
0A86  FFFF      C6      DC      X'FFFF',X'FFFF'    B10/B9
        FFFF
0A8A  0000      C7      DC      0,0                B12/B13,B14/B15
        0000
                *
                * TABLE USED IN FLOATING POINT COMPARE
                *
0A8E  0010      F0      DC      X'0010',0
        0000
0A92  0010      F1      DC      X'0010',1
        0001
0A96  8010      F2      DC      X'8010',1
        0001
0A9A  4140      F3      DC      X'4140',0
        0000
0A9E  C140      F4      DC      X'C140',0
        0000
0AA2  0800      F5      DC      X'0800',0
        0000
0AA6  8800      F6      DC      X'8800',0
        0000
                *
0AAA                    END

 A0       08C6
 A1       08CA
 A10      08EE
 A12      08F2
 A14      08F6
 A15      08FA
 A16      08FE
 A17      0902
 A18      0906
 A19      090A
 A2       08CE
 A22      090E
 A23      0912
 A24      0916
 A25      091A
 A3       08D2
 A4       08D6
 A5       08DA
 A6       08DE
 A7       08E2
 A8       08E6
 A9       08EA
 B0       0A2E
 B1       0A32
 B10      0A56
 B11      0A5A
 B12      0A5E
 B13      0A62
 B14      0A66
 B15      0A6A
 B2       0A36
 B3       0A3A
 B4       0A3E
 B5       0A42
```

B6	0A46
B7	0A4A
B8	0A4E
B9	0A52
BIT5NO	083A
BIT5UP	0836
C0	0A6E
C1	0A72
C2	0A76
C3	0A7A
C4	0A7E
C5	0A82
C6	0A86
C7	0A8A
CECER	06DC
COM	0832
COMPC	0798
COMP10	07C0
COMP12	07C8
COMP14	07D0
COMP2	07A0
COMP4	07A8
COMP6	07B0
COMP8	07B8
COMPAR	07D8
D0	083E
D1	0846
D10	088E
D11	0896
D12	089E
D13	08A6
D14	08AE
D15	08B6
D16	08BE
D2	084E
D3	0856
D4	085E
D5	0866
D6	086E
D7	0876
D8	087E
D9	0886
DEV	0008
DVD	06A0
DVD1	06C8
DVDFIA	06BA
E0	091E
E1	0922
E10	0946
E12	094A
E13	094E
E2	0926
E3	092A
E4	092E
E5	0932
E6	0936
E7	093A
E8	093E
E9	0942
ERR	0002
ERROR	076E
F0	0A8E

F1	0A92
F2	0A96
F3	0A9A
F4	0A9E
F5	0AA2
F6	0AA6
HELP	000A
INCR	0005
INDEX	0004
LB1	075C
LB2	0784
LIMIT	0006
LISTDV	007E
M0	0986
M1	098A
M10	09AE
M11	09B2
M12	09B6
M13	09BA
M14	09BE
M15	09C2
M16	09C6
M17	09CA
M18	09CE
M19	09D2
M2	098E
M20	09D6
M21	09DA
M22	09DE
M23	09E2
M24	09E6
M25	09EA
M26	09EE
M3	0992
M4	0996
M5	099A
M6	099E
M7	09A2
M8	09A6
M9	09AA
MATH	01C8
MATH1	0398
MATH2	044C
MATH3	05C6
MESB1	07F8
MESB2	0818
MESN1	0816
MESN2	0830
ONE	0001
OUT	0007
P1	09F2
P10	0A16
P11	0A1A
P12	0A1E
P13	0A22
P14	0A26
P15	0A2A
P2	09F6
P3	09FA
P4	09FE
P5	0A02
P6	0A06

P7	0A0A
P8	0A0E
P9	0A12
R0	0842
R1	084A
R10	0892
R11	089A
R12	08A2
R13	08AA
R14	08B2
R15	08BA
R16	08C2
R2	0852
R3	085A
R4	0862
R5	086A
R6	0872
R7	087A
R8	0882
R9	088A
S0	096A
S1	096E
S2	0972
S3	0976
S4	097A
S5	097E
S6	0982
SEN	077E
SENSE	0756
STATUS	0009
STOP	076A
T0	0952
T1	0956
T2	095A
T3	095E
T4	0962
T5	0966
TOT	0003

MEMORY TEST PROGRAM
OPERATION MANUAL

1. INTRODUCTION

This manual describes the operation of the Memory Test for standard GE-PAC 30 Processors equipped with a display panel.

The Memory Test writes various test patterns into each memory cell and compares the readout to the pattern written. If a failure occurs, the program prints the address of the failing location on the teletypewriter, followed by the data transmitted to memory, followed by the data actually received from memory. If no failures are encountered, the program performs the next test specified.

The Memory Test Program Tape (Part Number 06-003R04M14) is to be loaded using the basic 50 Loader described in Appendix 1.

2. OPERATION

The following paragraphs provide operating procedures for loading the Memory Test. Select the appropriate paragraph depending upon the equipment complement of the Digital System to be tested.

2.1 Teletypewriter Reader

Use the following procedure on systems with a Teletypewriter Tape Reader as the input device.

1. Manually insert the 50 Loader or 68 Loader. See Appendix 1.
2. Set location X'78 to X'0294'.
3. Select address X'50' or X'68'.
4. Load the Memory Test Tape on the reader with the first character over the read fingers.
5. Depress the INITIALIZE pushbutton.
6. Rotate the MODE CONTROL Switch to the RUN position, and depress the EXECUTE Switch.
7. Set the Teletypewriter to the Remote Mode.
8. Start the tape reader running by moving the switch lever to START. On the Model 35 TTY, the MODE Switch should be set to the TTr position in addition.
9. Stop the reader once blank tape begins to pass over the read fingers.

As soon as the last character on tape is read, the test program is automatically executed.

2.2 High Speed Paper Tape Reader

Use the following procedure on systems with a High Speed Paper Tape Reader as the input device.

1. Manually insert the 50 Loader or 68 Loader. See Appendix 1.

2. Set location X'78' to X'0399'.

3. Select address X'50' or X'68'.

4. Load the Memory Test Tape in the High Speed Paper Tape Reader such that the blank frame in front of the first character holes on the tape rest over the photo-diodes.

5. Rotate the latching lever to lock the paper tape in place.

6. Rotate the MODE CONTROL Switch to the HALT position and momentarily depress first the EXECUTE pushbutton and then the INITIALIZE pushbutton.

7. Rotate the MODE CONTROL Switch to the RUN position and depress the EXECUTE Switch.

The above step starts the tape moving through the High Speed Paper Tape Reader, loading the test program into memory. As soon as the last character is read, execution of the Memory Test is immediately begun, and the Tape Reader is stopped.

3. PROGRAM DESCRIPTION

Immediately after loading the tape, or if execution is started at location X'80', the Memory Test will determine the size of the specific memory under test, generate the following printout, and HALT.

MEMORY SIZE nnK - SET DATA SW

If this printout does not occur immediately, the memory test is not functioning properly. Using the listing provided in Appendix 2, verify the first few locations to insure they were loaded correctly.

Starting execution of the Memory Test at location X'84' will cause the memory test to be relocated to location X'500' so that the locations originally occupied by the memory test (X'80' - X'4FF') can be tested. After the test has been relocated, the following printout is generated and the program halts.

LOW CORE TEST - SET DATA SW

The restart location for the memory test after it has been relocated is X'500'.

In either case, after the program has halted, Data Switches 8 - 14 on the display must be set to indicate which tests are to be performed (see Figure 1). If all tests are to be performed, depress all Data Switches 8 - 14. Data Switch 15 should not be depressed if all messages are to be printed. To continue, depress the EXECUTE pushbutton. The data switches can be changed during the running of the test. These changes in the data switches are detected by the program which will stop or begin executing the tests specified by the data switch settings.

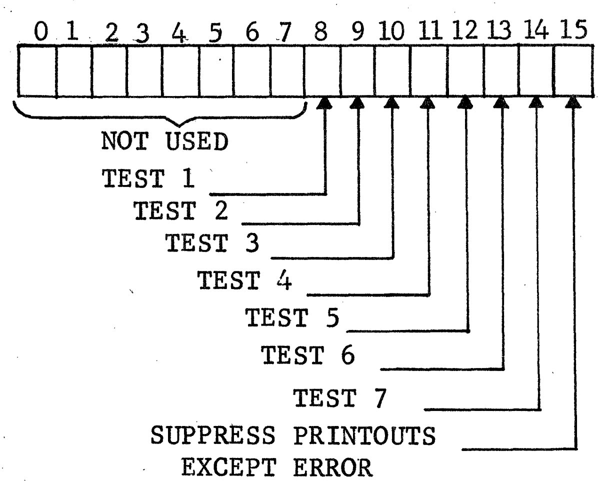


Fig. 1. Data Switch Meanings

If Data Switch 15 is not depressed, the following printout is generated <u>prior</u> to each test being performed:

TEST n

where "n" is the number of the test to be performed. At the completion of the last test, the data switches are read and the specified tests are executed.

If during the execution of a test a defective memory cell (sixteen-bit halfword) is found, an error count in Register 11 is incremented, and a message in the following format is typed:

FAILURE 0300 0001 0000

The above message indicates that at location X'300', the memory test attempted to write the value X'0001'. During the reading of this location, however, the data actually read was X'0000'. This shows that bit 15 is the bit in error and was the reason for the error printout. It is possible that the expected result and actual result will be the same. This is due to the fact that the memory cell is read a second time for the printout, and the second reading may be successful. This should still be regarded as a problem, as it indicates a marginal memory cell.

Depressing Data Switch 15 causes all test identification printout to be suppressed and the error count to be compared with a limit. If the error count is equal to or greater than the limit, error message printouts will be suppressed. The count of errors will be maintained in Register 11 (maximum is 32,768). This feature is particularly useful when it is desired to test memory over night or over a weekend. The limit is normally set to X'50', but can be modified by changing location X'044C'. The error count is cleared whenever the test is started at X'80' or X'500'.

To suppress all printouts, turn the teletypewriter off. An error count is maintained in Register 11.

4. TEST DESCRIPTIONS

The following is a description of the tests contained in the Memory Test Program and which data switch is associated with which test.

TEST 1 - Depressing Data Switch 8 will cause Test 1 to be executed. This test loads the address of each location into itself until memory is full; i.e., location X'100' is loaded with X'0100'; location X'102' is loaded with X'0102'; location X'104' is loaded with X'0104'; etc. until the top-of-core is reached. Then each location is checked to see if any errors have occurred. When an error is detected, the error count is incremented by one and an error message printout is generated.

TEST 2 - Depressing Data Switch 9 will cause Test 2 to be executed. This test clears memory and walks a word of all ones (X'FFFF') through memory. After each cell has been tested, it is cleared and tested to insure it is zero before going to the next cell. For each error detected, the error count is incremented by one and an error message printout is generated.

TEST 3 - Depressing Data Switch 10 will cause Test 3 to be executed. This test clears memory and walks a test word through memory. The test word contains all zeros except for one bit which is shifted left one bit position starting at the least significant bit position each time through memory; i.e., each halfword is tested with X'0001', then X'0002', then X'0004', to X'8000'. Each cell is cleared and tested to insure it is zero after it has been tested. When an error is detected, the error count is incremented by one and an error message printout is generated.

TEST 4 - Depressing Data Switch 11 will cause Test 4 to be executed. This test fills memory with all ones (X'FFFF') and walks a word of all zeros through memory. Each cell is again filled with X'FFFF' and checked after it has been tested. For each error detected, the error count is incremented by one and an error message printout is generated.

TEST 5 - Depressing Data Switch 12 will cause Test 5 to be executed. This test fills memory with all ones (X'FFFF') and shifts a zero bit, starting at the least significant bit position, through a test word of all ones. The test word is then walked through memory. Each cell is then restored to all ones (X'FFFF') and checked. When an error occurs, the error count is incremented and an error message printout is generated.

TEST 6 - Depressing Data Switch 13 will cause Test 6 to be executed. This test fills memory with X'CCCC' and walks a test word of X'3333' through memory. Each cell is returned to X'CCCC' and checked after it has been tested. For each error detected, the error count is incremented by one and an error message printout is generated.

TEST 7 - Depressing Data Switch 14 will cause Test 7 to be executed. This test fills memory with X'3333' and walks a test word of X'CCCC' through memory. Each cell is returned to X'3333' and checked after it has been tested. For each error detected, the error count is incremented by one and an error message is generated.

APPENDIX 1

50 LOADER FOR LOADING MEMORY TEST

5Ø	C82Ø	LOAD	LHI	2,X'8Ø'
52	ØØ8Ø			
54	C83Ø		LHI	3,1
56	ØØØ1			
58	C84Ø		LHI	4,X'44D'
5A	044D			
5C	D3AØ		LB	1Ø, BINDV
5E	ØØ78			
6Ø	DEAØ		OC	1Ø, BINDV+1
62	ØØ79			
64	9DAE	SENSE	SSR	1Ø,14
66	Ø8EE		LHR	14,14
68	423Ø		BTC	3,SENSE
6A	ØØ64			
6C	DBA2		RD	1Ø,Ø(2)
6E	ØØØØ			
7Ø	C12Ø		BXLE	2,SENSE
72	ØØ64			
74	43ØØ		B	X'8Ø'
76	ØØ8Ø			
78	Ø294	BINDV	DC	X'Ø294'

68 Loader For Loading Memory Test
(For GE-PAC 30-2's only).
(Note that the 68 Loader will bypass leading blanks.)

68	C830		LHI	3,1
6A	0001			
6C	D3A0		LB	10, BINDV
6E	0078			
70	D500		AL	0,X'44D'
72	044D			
74	4300		B	X'80'
76	0080			
78	0294	BINDV	DC	X'0294

The device definitions above are for a Teletype with Device Number 2. For other input devices, use the following:

High Speed Paper Tape Input	Ø399
Mini Tape Input	Ø595

```
                        OPT   PASS2,PRINT,PUNCH,STOP
0080                    ORG   X'80'
                  ********************************************************
                  *
                  *      MEMORY TEST
                  *
                  *      06-003R04A13
                  *
                  *      JUNE 23, 1969
                  *
                  ********************************************************
                  *
                  *
0003              BIAS    EQU   3
0006              TSTWD   EQU   6                  TEST WORD
0008              TTY     EQU   8
0009              HCLIM   EQU   9                  HIGH CORE LIMIT
000B              CERR    EQU   11                 COMPARE ERROR CNTR
000C              DISPLY  EQU   12
000D              BEGIN   EQU   13                 ADDR TESTING TO START
000C              ONE     EQU   DISPLY
0008              TWO     EQU   TTY
0500              START   EQU   X'500'
                  *
                  *
0080  4300                B     MENTRY
      012E
                  *
                  *
                  ********************************************************
                  *
                  *  THIS ROUTINE RELOCATES THE BODY OF THE MEMORY TEST
                  *  TO LOC X'0500' AND HALTS PRIOR TO EXECUTING THE
                  *  TESTS SPECIFIED IN DATA SW 8-14
                  *
                  ********************************************************
0084  C830        RELOC   LHI   BIAS,START-EXEC
      036C
0088  0B77                SHR   7,7
008A  C890                LHI   9,TERM-EXEC
      026A
008E  4847        RPT     LH    4,EXEC(7)
      0194
0092  4047                STH   4,START(7)
      0500
0096  4857                LH    5,EXEC+2(7)
      0196
009A  0804                LHR   0,4
009C  C440                NHI   4,X'D000'          ISOLATE OP-CODE
      D000
00A0  C540                CLHI  4,X'C000'          IS INSTR RS?
      C000
00A4  4330                BE    RSCHK
      0100
```

```
00A8  C440           NHI    4,X'4000'          IS INSTR RR?
      4000
00AC  4330           BZ     RR
      00B8
00B0  0A53   RX      AHR    5,BIAS             MUST BE RX
00B2  0A78   RS      AHR    7,8                INCR INDEX BY 2
00B4  4057           STH    5,START(7)
      0500
00B8  C170   RR      BXLE   7,RPT
      008E
00BC  C890           LHI    9,LIMIT-EXEC
      02B8
00C0  4847   DATA1   LH     4,EXEC(7)
      0194
00C4  4047           STH    4,START(7)
      0500
00C8  C170           BXLE   7,DATA1
      00C0
00CC  4813           LH     1,LOOP+6(BIAS)
      0210
00D0  4013           STH    1,C1+2(BIAS)
      01FE
00D4  4013           STH    1,LOOP+2(BIAS)
      020C
00D8  4013           STH    1,Z+2(BIAS)
      0398
00DC  C8D0           LHI    BEGIN,X'80'-START
      FB80
00E0  0BD3           SHR    BEGIN,BIAS
00E2  C89D           LHI    HCLIM,X'47E'(BEGIN)
      047E
00E6  0BBB           SHR    CERR,CERR
00E8  C810           LHI    1,LOWMSG
      0110
00EC  C830           LHI    3,LOWEND
      012D
00F0  41E0           BAL    14,TYPEIT
      0366
00F4  C200           LPSW   WAIT
      00F8
00F8  8000   WAIT    DC     X'8000',A(*+2)
      00FC
00FC  4300           B      START
      0500
0100  C400   RSCHK   NHI    0,X'DF00'
      DF00
0104  C500           CLHI   0,X'C300'
      C300
0108  4280           BTC    8,RX
      00B0
010C  4300           B      RS
      00B2
             *
0110  4C4F   LOWMSG  DC     C'LOW              CORE TEST -- SET DATA SW'
```

```
      5720
      434F
      5245
      2054
      4553
      5420
      2D2D
      2053
      4554
      2044
      4154
      4120
      5357
012C  0D0A              DC     X'0D0A'
012D         LOWEND     EQU    *-1
             *
             *
             *
012E  C880   MENTRY     LHI    TTY,2
      0002
0132  C8C0              LHI    DISPLY,1
      0001
0136  082C              LHR    2,ONE
0138  0BDD              SHR    BEGIN,BEGIN
013A  C200              LPSW   *+4
      013E
013E  0000              DC     0,A(*+2)          CLEAR CURRENT PSW INTINB
      0142
0142  0BBB              SHR    CERR,CERR         CLEAR ERROR CNTR
0144  0B99              SHR    HCLIM,HCLIM
0146  0B33              SHR    3,3               CLEAR REG 3
0148  C860              LHI    6,X'B0B0'         MEM SIZE PRINTOUT-00
      B0B0
014C  0A3C   NT         AHR    3,12              COUNT OF K'S OF MEMORY
014E  CA90              AHI    HCLIM,X'400'      INDEX IN MULTS OF 1024
      0400
0152  4280              BC     FOUND             FOR 64 K OF MEMORY BUG OUT
      0162
0156  4099              STH    HCLIM,0(HCLIM)    STORE TEST WORD
      0000
015A  4859              LH     5,0(HCLIM)        TEST IF DATA RETURNS
      0000
015E  4230              BNZ    NT                NO -- TOP-OF-CORE FOUND
      014C
0162  CB90   FOUND      SHI    HCLIM,X'503'
      0503
0166  CB30   CONV       SHI    3,10              SUBT OUT TENS
      000A
016A  4210              BM     UNITS             BRANCH IF GONE NEG
      0176
016E  CA60              AHI    6,X'100'          INCR TENS DIGIT
      0100
0172  4300              B      CONV              CHECK FOR MORE TENS
      0166
```

```
0176  CA30    UNITS   AHI    3,10              BACK POSITIVE
      000A
017A  0663            OHR    6,3               INSERT UNITS DIGIT
017C  4060            STH    6,NUMBER          STORE INTO PRINTOUT
      041A
0180  C810            LHI    1,MESS
      040C
0184  C830            LHI    3,END
      042F
0188  41E0            BAL    14,TYPEIT         PRINT OUT MEM SIZE
      0366
018C  C200            LPSW   HALT              HALT SO SWITCHES CAN BE
      0190
0190  8000    HALT    DC     X'8000',A(*+2)    SET UP
      0194
              ******************************************************
              *
              *   EXECUTIVE -- DETERMINES WHAT TESTS ARE TO BE
              *                PERFORMED AND CALLS THOSE TESTS
              *
              ******************************************************
0194  9BC0    EXEC    RDR    DISPLY,0          READ SW 8-15 FOR TESTS
0196  CD00            SLHL   0,9               TO BE PERFORMED
      0009
019A  4380            BFC    8,*+8
      01A2
019E  41A0            BAL    10,TEST1          SW8 SET, GOTO TEST 1
      01EE
01A2  CD00            SLHL   0,1
      0001
01A6  4380            BFC    8,*+8
      01AE
01AA  41A0            BAL    10,TEST2          SW9 SET, GOTO TEST 2
      0220
01AE  CD00            SLHL   0,1
      0001
01B2  4380            BFC    8,*+8
      01BA
01B6  41A0            BAL    10,TEST3          SW10 SET, GOTO TEST 3
      0248
01BA  CD00            SLHL   0,1
      0001
01BE  4380            BFC    8,*+8
      01C6
01C2  41A0            BAL    10,TEST4          SW11 SET, GOTO TEST 4
      0276
01C6  CD00            SLHL   0,1
      0001
01CA  4380            BFC    8,*+8
      01D2
01CE  41A0            BAL    10,TEST5          SW12 SET, GOTO TEST 5
      029E
01D2  CD00            SLHL   0,1
      0001
```

```
01D6  4380            BFC    8,*+8
      01DE
01DA  41A0            BAL    10,TEST6          SW13 SET, GOTO TEST 6
      02D4
01DE  CD00            SLHL   0,1
      0001
01E2  4380            BFC    8,*+8
      01EA
01E6  41A0            BAL    10,TEST7          SW14 SET, GOTO TEST 7
      02FE
01EA  4300            B      EXEC
      0194
                ********************************************************
                *
                *  TEST 1 -- LOADS EACH MEM LOC WITH IT'S ADDR AND
                *            CHECKS EACH LOC.
                *
                ********************************************************
01EE  C810      TEST1 LHI    1,C'              1'
      2031
01F2  4010            STH    1,TSTNO
      0408
01F6  41E0            BAL    14,PRNT           PRINT TEST ID
      0356
01FA  087D            LHR    7,BEGIN           START ADDR OF TEST AREA
01FC  C867      C1    LHI    TSTWD,START(7)
      0500
0200  4067            STH    TSTWD,START(7)
      0500
0204  C170            BXLE   7,*-8
      01FC
0208  087D            LHR    7,BEGIN           START ADDR OF TEST AREA
020A  C867      LOOP  LHI    TSTWD,START(7)
      0500
020E  4567            CLH    TSTWD,START(7)
      0500
0212  4330            BE     *+8
      021A
0216  41F0            BAL    15,ERR
      0380
021A  C170            BXLE   7,LOOP
      020A
021E  030A            BR     10                RETURN TO EXEC
                ********************************************************
                *
                *  TEST 2 -- CORE IS CLEARED AND A WORD OF ONES IS
                *            WALKED THRU MEMORY
                *
                ********************************************************
0220  C810      TEST2 LHI    1,C'              2'
      2032
0224  4010            STH    1,TSTNO
      0408
0228  41E0            BAL    14,PRNT           PRINT TEST ID
```

```
        0356
 022C   0B55                SHR    5,5              CLEAR REG 5
 022E   087D                LHR    7,BEGIN          START ADDR OF TEST AREA
 0230   4057                STH    5,START(7)
        0500
 0234   C170                BXLE   7,*-4            CLEAR MEMORY
        0230
 0238   C860                LHI    TSTWD,X'FFFF'
        FFFF
 023C   087D                LHR    7,BEGIN          START ADDR OF TEST AREA
 023E   41F0                BAL    15,TEST          TEST LOCATION
        0328
 0242   C170                BXLE   7,*-4
        023E
 0246   030A                BR     10               RETURN TO EXEC
                  *******************************************************
                  *
                  *  TEST 3 -- CORE IS CLEARED AND A ONE BIT IS SHIFIED
                  *            THRU A WORD OF ZEROS AND THE WORD IS
                  *            WALKED THRU MEMORY
                  *
                  *******************************************************
 0248   C810      TEST3     LHI    1,C'             3'
        2033
 024C   4010                STH    1,TSTNO
        0408
 0250   41E0                BAL    14,PRNT          PRINT TEST ID
        0356
 0254   0B55                SHR    5,5
 0256   087D                LHR    7,BEGIN          START ADDR OF TEST AREA
 0258   4057                STH    5,START(7)
        0500
 025C   C170                BXLE   7,*-4            CLEAR MEMORY
        0258
 0260   086C                LHR    TSTWD,12         LOAD TEST WD WITH X'0001'
 0262   087D      REPEAT    LHR    7,BEGIN          START ADDR OF TEST AREA
 0264   41F0                BAL    15,TEST          TEST LOCATION
        0328
 0268   C170                BXLE   7,*-4
        0264
 026C   CD60                SLHL   TSTWD,1          SHIFT 1 THRU WD OF ZERO
        0001
 0270   4380                BFC    8,REPEAT         CONT UNTIL CARRY OCCURS
        0262
 0274   030A                BR     10               RETURN TO EXEC
                  *******************************************************
                  *
                  *            OF ZEROS IS WALKED THRU MEMORY
                  *
                  *******************************************************
 0276   C810      TEST4     LHI    1,C'             4'
        2034
 027A   4010                STH    1,TSTNO
        0408
```

```
027E  41E0            BAL   14,PRNT          PRINT TEST ID
      0356
0282  087D            LHR   7,BEGIN          START ADDR OF TEST AREA
0284  C850            LHI   5,X'FFFF'
      FFFF
0288  4057            STH   5,START(7)
      0500
028C  C170            BXLE  7,*-4            FILL CORE WITH ONES
      0288
0290  087D            LHR   7,BEGIN          START ADDR OF TEST AREA
0292  0B66            SHR   TSTWD,TSTWD
0294  41F0            BAL   15,TEST
      0328
0298  C170            BXLE  7,*-4
      0294
029C  030A            BR    10               RETURN TO EXEC
                ******************************************************
                *
                *  TEST 5 -- CORE IS FILLED WITH ALL ONES AND A ZERO
                *            BIT IS SHIFTED THRU A WORD OF ONES AND
                *            THE WORD IS WALKED THRU MEMORY
                *
                ******************************************************
029E  C810      TEST5 LHI   1,C'             5'
      2035
02A2  4010            STH   1,TSTNO
      0408
02A6  41E0            BAL   14,PRNT          PRINT TEST ID
      0356
02AA  C850            LHI   5,X'FFFF'
      FFFF
02AE  087D            LHR   7,BEGIN          START ADDR OF TEST AREA
02B0  4057            STH   5,START(7)
      0500
02B4  C170            BXLE  7,*-4            FILL CORE WITH ONES
      02B0
02B8  C860            LHI   TSTWD,X'FFFE'    ZERO BIT IN WD OF ONES
      FFFE
02BC  087D      AGIN  LHR   7,BEGIN          START ADDR OF TEST AREA
02BE  41F0            BAL   15,TEST          TEST LOCATION
      0328
02C2  C170            BXLE  7,*-4
      02BE
02C6  CD60            SLHL  TSTWD,1          SHIFT ZERO THRU ONES
      0001
02CA  038A            BFCR  8,10
02CC  C660            OHI   TSTWD,X'0001'    SET BIT 15 = 1
      0001
02D0  4300            B     AGIN
      02BC
                ******************************************************
                *
                *  TEST 6 -- CORE IS FILLED WITH X'CCCC' AND A WORD OF
                *            X'3333' IS WALKED THRU MEMORY
```

```
                 *
                 ***********************************************************
02D4  C810       TEST6    LHI    1,C'              6'
      2036
02D8  4010                STH    1,TSTNO
      0408
02DC  41E0                BAL    14,PRNT           PRINT TEST ID
      0356
02E0  C850                LHI    5,X'CCCC'
      CCCC
02E4  087D                LHR    7,BEGIN           START ADDR OF TEST AREA
02E6  4057                STH    5,START(7)
      0500
02EA  C170                BXLE   7,*-4             FILL CORE WITH X'CCCC'
      02E6
02EE  087D                LHR    7,BEGIN           START ADDR OF TEST AREA
02F0  C860                LHI    TSTWD,X'3333'
      3333
02F4  41F0                BAL    15,TEST           TEST LOCATION
      0328
02F8  C170                BXLE   7,*-4
      02F4
02FC  030A                BR     10                RETURN TO EXEC
                 ***********************************************************
                 *
                 *  TEST 7 -- CORE IS FILLED WITH X'3333' AND A WORD OF
                 *            X'CCCC' IS WALKED THRU MEMORY
                 *
                 ***********************************************************
02FE  C810       TEST7    LHI    1,C'              7'
      2037
0302  4010                STH    1,TSTNO
      0408
0306  41E0                BAL    14,PRNT           PRINT TEST ID
      0356
030A  C850                LHI    5,X'3333'
      3333
030E  087D                LHR    7,BEGIN           START ADDR OF TEST AREA
0310  4057                STH    5,START(7)
      0500
0314  C170                BXLE   7,*-4             FILL CORE WITH X'3333'
      0310
0318  087D                LHR    7,BEGIN           START ADDR OF TEST AREA
031A  C860                LHI    TSTWD,X'CCCC'
      CCCC
031E  41F0                BAL    15,TEST           TEST LOCATION
      0328
0322  C170                BXLE   7,*-4
      031E
0326  030A                BR     10                RETURN TO EXEC
                 ***********************************************************
                 *
                 *  TEST -- THE CONTENTS OF TSTWD IS STORED INTO THE
                 *          TEST LOC & READ FOR CHECKING
```

```
                 *
                 ****************************************************
0328  0B44       TEST     SHR    4,4
032A  D240                STB    4,STATUS
      0402
032E  4067                STH    TSTWD,START(7)  STORE TEST WORD
      0500
0332  4567                CLH    TSTWD,START(7)  READ & COMPARE
      0500
0336  4230                BNE    FAIL
      0348
033A  D2C0                STB    ONE,STATUS
      0402
033E  4057                STH    5,START(7)      RESTORE LOCATION
      0500
0342  4557                CLH    5,START(7)      RESTORE CORRECTLY?
      0500
0346  033F                BFCR   3,15            YES - EXIT
0348  40F0       FAIL     STH    15,RETURN       SAVE TEST RETURN ADDR
      0400
034C  41F0                BAL    15,ERR          GO TO ERROR ROUTINE
      0380
0350  48F0                LH     15,RETURN       RESTORE RETURN ADDR
      0400
0354  030F                BR     15              RETURN TO CALLING PROG
                 *
                 *
0356  9BC1       PRNT     RDR    DISPLY,1        READ BITS 8-15
0358  041C                NHR    1,ONE           BIT 15 SET
035A  4230                BNZ    EXIT            YES-EXIT
      037E
035E  C810                LHI    1,TSTMSG
      0404
0362  C830                LHI    3,TSTEND
      040B
0366  DE80       TYPEIT   OC     TTY,TYPEO       CMD TTY TO WRITE
      0403
036A  DD80       STAT1    SS     TTY,STATUS
      0402
036E  4210                BTC    1,EXIT          DEV UNAVAILABLE-EXIT
      037E
0372  4280                BTC    8,STAT1         DEV BUSY-LOOP
      036A
0376  DA81                WD     TTY,0(1)        OUTPUT CHAR
      0000
037A  C110                BXLE   1,STAT1
      036A
037E  030E       EXIT     BR     14              RETURN TO CALLING PROG
                 ****************************************************
                 *
                 *  ERR -- DATA FOR THE ERROR MSG PRINTOUT IS ASSEMBLED
                 *         AND CONVERTED TO ASCII
                 *
                 ****************************************************
```

```
0380  08BB   ERR     LHR   CERR,CERR
0382  021F           BTCR  1,15
0384  0ABC           AHR   CERR,ONE       INCR ERROR CNTR BY 1
0386  9BC1           RDR   DISPLY,1
0388  041C           NHR   1,ONE
038A  4330           BZ    *+10           BIT 15 SET - NO BR
      0394
038E  45B0           CLH   CERR,LIMIT     ERR CNTR >     LIMIT
      044C
0392  038F           BFCR  8,15           YES - EXIT
0394  0B22           SHR   2,2
0396  C837   Z       LHI   3,START(7)     LOAD FAILING ADDRESS
      0500
039A  41E0           BAL   14,CONVP       CONV TO ASCII
      03D2
039E  0836           LHR   3,TSTWD
03A0  D340           LB    4,STATUS       RESTORE FAILURE?
      0402
03A4  0844           LHR   4,4
03A6  4330           BZ    *+6            YES
      03AC
03AA  0835           LHR   3,5
03AC  41E0           BAL   14,CONVP1      CONV TSTWD TO ASCII
      03C8
03B0  4837           LH    3,START(7)
      0500
03B4  41E0           BAL   14,CONVP1      CONV FAILED DATA TO ASCII
      03C8
03B8  082C           LHR   2,ONE          RESTORE CONSTANTS
03BA  C810           LHI   1,CMPERR
      0430
03BE  C830           LHI   3,ENDCE
      044B
03C2  41E0           BAL   14,TYPEIT      PRINT ERROR MSG
      0366
03C6  030F           BR    15             RETURN TO CALLING PROG
             *******************************************************
             *
             *  THE CONTENTS OF REG 3 ARE CONVERTED TO ASCII AND
             *  STORED INTO DATA AREA OF THE ERROR MSG
             *
             *******************************************************
03C8  C840   CONVP1  LHI   4,X'2020'
      2020
03CC  4042           STH   4,DATA(2)      STORE SPACES
      043A
03D0  0A28           AHR   2,8            INCR INDEX BY 2
03D2  C810   CONVP   LHI   1,12           SHIFT COUNT
      000C
03D6  0843           LHR   4,3
03D8  CC41           SRHL  4,0(1)
      0000
03DC  C440           NHI   4,X'F'         MASK DIGIT
      000F
```

```
03E0  C540          CLHI   4,X'A'          CHAR < X'A' ?
      000A
03E4  4280          BL     *+8             YES - DO NOT ADD 7
      03EC
03E8  CA40          AHI    4,7             ADD 7 TO CHAR
      0007
03EC  CA40          AHI    4,X'30'         ADD X'30' TO MAKE ASCII
      0030
03F0  D242          STB    4,DATA(2)       STORE INTO DATA AREA
      043A
03F4  0A2C          AHR    2,ONE           INCR INDES BY ONE
03F6  CB10          SHI    1,4             DECR SHIFT COUNT BY 4
      0004
03FA  4310          BNM    CONVP+4
      03D6
03FE  030E   TERM   BR     14              RETURN TO CALLING PROG
             *
             *
             *
0400  0000   RETURN DC     X'0000'
0402  0098   STATUS DC     X'0098'
0403         TYPEO  EQU    *-1
0404  5445   TSTMSG DC     C'TEST'
      5354
0408  0000   TSTNO  DC     X'0000'
040A  0D0A          DC     X'0D0A'
040B         TSTEND EQU    *-1
040C  0D0A   MESS   DC     X'0D0A'
040E  4D45          DC     C'MEMORY        SIZE '
      4D4F
      5259
      2053
      495A
      4520
041A  3030   NUMBER DC     C'00K           --  SET DATA SW'
      4B20
      202D
      2D20
      2053
      4554
      2044
      4154
      4120
      5357
042E  0D0A          DC     X'0D0A'
042F         END    EQU    *-1
0430  4641   CMPERR DC     C'FAILURE       '
      494C
      5552
      4520
      2020
043A         DATA   DS     16
044A  0D0A          DC     X'0D0A'
044B         ENDCE  EQU    *-1
```

```
044C  0050     LIMIT    DC     X'50'
044E                    END
```

AGIN	02BC
BEGIN	000D
BIAS	0003
C1	01FC
CERR	000B
CMPERR	0430
CONV	0166
CONVP	03D2
CONVP1	03C8
DATA	043A
DATA1	00C0
DISPLY	000C
END	042F
ENDCE	044B
ERR	0380
EXEC	0194
EXIT	037E
FAIL	0348
FOUND	0162
HALT	0190
HCLIM	0009
LIMIT	044C
LOOP	020A
LOWEND	012D
LOWMSG	0110
MENTRY	012E
MESS	040C
NT	014C
NUMBER	041A
ONE	000C
PRNT	0356
RELOC	0084
REPEAT	0262
RETURN	0400
RPT	008E
RR	00B8
RS	00B2
RSCHK	0100
RX	00B0
START	0500
STAT1	036A
STATUS	0402
TERM	03FE
TEST	0328
TEST1	01EE
TEST2	0220
TEST3	0248
TEST4	0276
TEST5	029E
TEST6	02D4
TEST7	02FE

TSTEND	040B
TSTMSG	0404
TSTNO	0408
TSTWD	0006
TTY	0008
TWO	0008
TYPEIT	0366
TYPEO	0403
UNITS	0176
WAIT	00F8
Z	0396

OPERATING INSTRUCTIONS FOR THE ASR 33 AND ASR 35

TELETYPEWRITER TEST PROGRAM 06-004R03

1. FUNCTION

This program stores teletypewriter characters that have been inputted from the keyboard, checks the parity on each character, punches the characters on tape, and then reads back the tape comparing each character with the original input. The action of the BREAK key is also tested. A detected error causes the program to halt and Data/Address lamps 8 through 15 on the Display Panel light.

2. PROGRAM TAPE

The teletypewriter (TTY) test program tape (part number 06-004R03M14) is punched in Binary and is loaded using the basic loader listed in Table 1.

3. LOADING TEST TAPE

1. Manually insert the basic loader beginning at location X'50'. When finished, reselect address X'50'. Set mode to ADR. Push EXECUTE.
2. Load the teletypewriter test tape on the reader with the first character over the read fingers. On the ASR 35 set the MODE Switch to KT.
3. Rotate the MODE CONTROL to the HALT position and depress the INITIALIZE pushbutton.
4. Set Mode to RUN push EXECUTE.

TABLE 1. BASIC LOADER

LOCATION	OP-CODE	ADDRESS	NAME	OPERATION	OPERAND	COMMENTS
0050	C820	0080		LHI	2,X'0080'	BEGINNING LOAD ADDRESS
0054	C830	0001		LHI	3,1	
0058	C840	034D		LHI	4,X'34D'	FINAL LOAD ADDRESS
005C	D3A0	0078		LB	10, BINDV	DEVICE NUMBER
0060	DEA0	0079		OC	10, BINDV+1	OUTPUT COMMAND, MOVE
0064	9DAE		SENSE	SSR	10,14	SENSE STATUS
0066	08EE			LHR	14,14	
0068	4230	0064		BTC	3, SENSE	BUSY
006C	DBA2	0000		RD	10,0(2)	NO, READ A CHARACTER
0070	C120	0064		BXLE	2, SENSE	INCREMENT LOAD ADDRESS
0074	4300	0080		B	X'0080'	BRANCH TO PROGRAM
0078	0294		BINDV*	DC	X'0294'	

*BINDV TTY X'0294'
HSPTR X'0399'

5. Start the tape reader running by moving the switch lever on the reader to the START position.

6. Stop the reader once blank tape begins to pass over the read fingers.

As soon as the last character on tape is read, the loader will transfer to the start of the test program and halt with the Wait light lit.

4. TEST PROGRAM DESCRIPTION

The TTY Test Program is divided into five sections. Each section is described in the following paragraphs.

Section 1 stores characters that are inputted from the keyboard in the interrupt mode. Before each character is stored, it is checked for even parity. The device number and status are also tested. The inputting of a carriage return, or more than 72 characters, causes the test to print out "PUNCH TAPE" and the system halts with the Wait light illuminated. This allows the operator time to load blank tape.

Section 2 punches a tape (with blank leader and trailer) containing the characters received in Section 1. This operation is also done under interrupt control. The device number and status are again tested. After all the characters have been punched, the program prints "READ TAPE" and halts with the Wait light illuminated. This allows the operator to remove the tape from the punch, and place it in the reader.

Section 3 reads the characters from the punched paper tape and compares them with the character that was originally inputted. A mismatch causes a branch to error. In this section and the next, the status bits are used to decide when to read and write teletype characters.

Section 4 prints the characters read from the tape, allowing the operator to inspect the printout.

Section 5 tests the operation of the break key. The message "DEPRESS BRK" instructs the operator to depress the BREAK key. When the BREAK key is released, the program prints "BRK OK".

At the end of the test, the program prints "END".

5. OPERATOR INSTRUCTIONS

On the ASR 35, set the MODE Switch to K. After the Wait light comes on, rotate the MODE CONTROL to HALT and depress the INITIALIZE pushbutton. Next, rotate the MODE CONTROL to RUN, and depress the EXECUTE pushbutton. If the teletypewriter status byte reflects a correct initialization pattern, the program outputs a carriage return and line feed. If not correctly initialized, the program halts and Data/Address lamps 8 through 15 are lit. In each instance where the program halts due to error, Register 15 contains the failing address plus four bytes. Therefore, to obtain the actual failing program address, subtract four from the address in Register 15.

Type in each character until all letters, numbers and symbols (both upper and lower case) have been entered. Do not hit the BREAK, HERE IS, or any of the Tape control keys (X-ON, X-OFF, TAPE ON, TAPE OFF). The receipt of 72 characters, or a carriage return terminates the test section. The program outputs the message "PUNCH TAPE" and halts with the Wait light on.

Next load a supply of perforator tape. The test program will provide a leader and trailer on the test tape.

On the ASR 35, set the MODE Switch to KT. Depress the EXECUTE pushbutton. All characters that were inputted in Test Section 1 will be punched on the tape and also typed on the paper copy. After all the characters are punched on the tape, the program types out the message "READ TAPE". The test program then halts and the Wait light is turned on.

Load the test tape in the reader in the blank area. On the ASR 35, set the MODE Switch to T. Depress the EXECUTE pushbutton to start the reader. After the tape is read, the test program prints the contents of the tape. This allows the operator to compare the results with the characters which were originally inputted. If an error is detected, the program prints "TAPE ERROR" and halts with the Wait lamp lit. Depressing the EXECUTE pushbutton causes the program to proceed to the next test.

This completes all tests. The message "END" is typed, and the program halts with the Wait light lit.

Upon completing the reader test, the test program instructs the operator to "DEPRESS BRK". Upon releasing the BREAK key, the message "BRK OK" is typed. If the BREAK key is inoperative, the program hangs up in a loop between locations X'28E' and X'294'.

To repeat the teletypewriter test, depress the EXECUTE pushbutton.

The Starting address of the teletypewriter test program is X'80'. When executing the program from this location, it is necessary to depress the EXECUTE pushbutton twice.

The TTY Test program assumes that the TTY device controller is wired as device number 2. If the TTY controller is wired for another device number, write the device number of the TTY controller under test, in hexadecimal, into location X'8A'.

A listing of the teletypewriter test program is provided in Appendix 1.

MARK III MEMORY TEST PROGRAM OPERATION MANUAL

1. INTRODUCTION

This manual describes the operation of the Mark III Memory Test for standard GE-PAC 30 Processors, as well as Processors with the Autoload option. (GE-PAC 30 Processors equipped with the Autoload feature are not normally equipped with a Display Panel.) This memory test also tests Processors equipped with the optional parity check circuitry.

The Mark III Memory Test writes various test patterns into each memory cell and compares the readout to the patterns written. If a failure occurs, the program prints the address of the failing location on the teletypewriter, followed by the data transmitted to memory, followed by the data acutally received. If no failures are encountered, the program prints that the memory was found to be in good operating condition. On Processors that are equipped with the optional parity detection circuitry, the program also prints those locations where parity failures occur.

2. PROGRAM TAPES

Two Mark III Memory Program Test Tapes are available. The standard Memory Test Tape (Part Number 06-003R03M14) is to be loaded using the basic 50 loader described in Appendix 1. A special memory test tape (Part Number 06-034M14) is also available for those machines that have the Autoload feature, and hence do not require an additional loader.

3. OPERATION

The following paragraphs provide operating procedures for the Memory Test. Select the appropriate paragraph depending upon the equipment complement of the Digital System to be tested.

3.1 Standard Tape, Teletypewriter Reader

Use the following procedure on systems with a Display Panel, when the Teletype Tape Reader is to be used as the input device.

1. Manually insert the 50 Loader beginning at location X'50'. See Appendix 1.
2. Set location X'78' to X'0294'.
3. Select address X'50'.
4. Load the Memory Test Tape on the reader with the first character over the read fingers.
5. Depress the INITIALIZE pushbutton.
6. Rotate the MODE CONTROL Switch to the RUN position, and depress the EXECUTE Switch.
7. Set the Teletypewriter to the Remote Mode.
8. Start the tape reader running by moving the switch lever to START. On the Model 35 TTY, the MODE Switch should be set to TTr position in addition.
9. Stop the reader once blank tape begins to pass over the read fingers.

As soon as the last character on tape is read, the test program is automatically executed.

3.2 Standard Tape, High Speed Paper Tape Reader

Use the following procedure on systems with a Display Panel, when a High Speed Paper Tape Reader is to be used as the input device.

1. Manually insert the 50 Loader beginning at location X'50'. See Appendix 1.
2. Set location X'78' to X'0395'.
3. Select address X'50'.
4. Load the Memory Test Tape in the High Speed Paper Tape Reader such that the left edge of the first character holes on the tape rest over the photo-diodes.
5. Rotate the latching lever to lock the paper tape in place.
6. Rotate the MODE CONTROL Switch to the HALT position and momentarily depress first the EXECUTE pushbutton and then the INITIALIZE pushbutton.
7. Rotate the MODE CONTROL Switch to the RUN position and depress the EXECUTE Switch.

The above step starts the tape moving through the High Speed Paper Tape Reader, loading the test program into memory. As soon as the last character is read, execution of the Memory Test is immediately begun, and the Tape Reader is stopped.

3.3 Autoload

Use the following procedure on Autoload systems (no Display).

1. Load the Autoload Memory Test Tape on the reader with the first character over the read fingers.
2. Depress the INITIALIZE pushbutton.
3. Depress the AUTOLOAD latching pushbutton.
4. Depress the EXECUTE pushbutton.
5. Start the tape reader running by moving the switch lever to START.
6. Stop the reader once blank tape begins to pass over the read fingers.
7. Depress the INITIALIZE pushbutton.
8. Depress the AUTOLOAD pushbutton, releasing it from its operated position.

The memory test is then started by momentarily depressing the EXECUTE pushbutton.

4. INDICATIONS

The first action of the Mark III Memory Test is to determine the memory size of the specific Processor under test, and to generate the following printout:

MEMORY SIZE ___K.

If this printout does not occur immediately, the memory test is not functioning properly. Verify, using the listing provided in Appendix 2 or Appendix 3, that the first few program locations were loaded correctly.

After the memory size printout is finished being typed, on those Processors equipped with a Display Panel, lamps 8 through 14 of General Display 2 being to flicker. If the memory test fails to start, or stops, some of the lamps will be lit at normal intensity.

If all of the memory cells are found acceptable, a printout occurs. The time from execution to printout varies depending upon the size of the memory that is being tested. The message printed is as follows:

GE-PAC 30 MEMORY OK

The Processor is then halted and the Wait Lamp is illuminated. The test can be repeated by depressing the EXECUTE button again. The memory size, however, is not typed out on subsequent passes.

The Autoload Memory Test does not stop after the ok message is typed out. The test is repeated continuously with the ok message being typed each time the end is reached.

If a defective memory cell (16 Bit Halfword) is found, a message in the following format is typed:

FAILURE ø3øø øøø1 øøøø

The above message indicates that at location X'300', the memory test attempted to write the value , X'0001'. During the reading of this location, however, the data actually read was X'0000'. This shows that bit 15 is the bit in error and was the reason for the error printout. It is possible that the expected result and actual result will be the same. This is due to the fact that the memory cell is read a second time for the printout, and the second reading may be successful. This should still be regarded as a trouble, as it indicates a marginal memory cell.

On those Processors equipped with the memory parity detection circuits, a printout in the following format occurs if a memory reading resulting in a parity error occurs.

PARITY FAILURE 0300 0001 0000

The printout has the same meaning as described in the preceding paragraphs, except that the error was detected by the parity check circuit.

The memory test prints all errors it encounters until all possible memory locations have been tested. After all locations have been checked once, the ok message is typed as an indication that the test is complete. Of course, if any error messages precede the ok message, the memory failed the test.

5. CONTINUOUS RUNNING FEATURE

The standard Mark III Memory Test, 06-003R02, can be used to continuously test the Processor for extended periods (for example, overnite) and allow all other peripherals to be turned off. This feature can be enabled anytime by releasing all Data/Address switches 8 through 15. In this mode, any failures that occur are tallied in Register 11, and any parity failures that occur are tallied in Register 4. The maximum count stored in either register is 32,767 errors. Once all switches are released, local power can be removed from the teletypewriter.

To again restore the printout, turn local power on the teletypewriter, and depress Switch 8 with the program running. After the program prints the ok message, the Processor is stopped and registers 4 and 11 can be read on the Display Panel.

Appendix 2 contains a listing of the standard Mark III Memory Test. Appendix 3 contains a listing of the Autoload Mark III Memory Test.

APPENDIX 1

50 LOADER FOR LOADING MEMORY TEST

```
5Ø    C82Ø    LOAD    LHI    2,X'8Ø'
52    ØØ8Ø
54    C83Ø            LHI    3,1
56    ØØØ1
58    C84Ø            LHI    4,X'2D1'
5A    02D1
5C    D3AØ            LB     1Ø, BINDV
5E    ØØ78
6Ø    DEAØ            OC     1Ø, BINDV+1
62    ØØ79
64    9DAE    SENSE   SSR    1Ø,14
66    Ø8EE            LHR    14,14
68    423Ø            BTC    3,SENSE
6A    ØØ64
6C    DBA2            RD     1Ø,Ø(2)
6E    ØØØØ
7Ø    C12Ø            BXLE   2,SENSE
72    ØØ64
74    43ØØ            B      X'8Ø'
76    ØØ8Ø
78    Ø294    BINDV   DC     X'Ø294'
```

The device definitions above are for a Teletypewriter with device number 2. For other input devices, use the following:

High Speed Paper Tape Input	Ø395
Card Input	Ø42Ø

APPENDIX 2

STANDARD MARK III MEMORY TEST LISTING

```
                              OPT    PASS2,PRINT,PUNCH,STOP
                      *
                      * 06-003R03 MARK 3 MEMORY TEST
                      * NEW MEMORY TEST INCLUDES-
                      * 1. PRINOUT ON FAILURE
                      * 2 ENABLES MACHINE MALFUNCTIONS INTERRUPT FOR PARITY
                      * FAILURE DECTION
                      * 3. IMPOSES A MORE SEVERE TEST PATTERN
                      *
0080                          ORG    X'80'              PROG ORIGINED AT X'80'
                      *
                      *
0080   C8D0           MENTRY  LHI    13,TWO             DEVICE NUM OF TTY FOR MSG
       0002
0084   C8C0                   LHI    12,ONE             DEVICE NUM OF DISPLAY
       0001
0088   0B44                   SHR    4,4                CLEAR FOR PARITY INT CNT
008A   4040                   STH    4,X'3C'
       003C
008E   4040                   STH    4,INTFLG           CLEAR INTERRUPT FLAG
       02CE
0092   C800                   LHI    0,INTSER           STORE ADDRS OF INTR SERVIC
       0236
0096   4000                   STH    0,X'3E'            ROUTINE
       003E
009A   C200                   LPSW   *+4
       009E
009E   0000                   DC     0,A(*+2)           CLEAR CURRENT PSW INTINB
       00A2
00A2   0BBB                   SHR    11,11              CLEAR ERROR COUNTER
00A4   C860                   LHI    6,X'B0B0'          MEM SIZE PRINTOUT-OO
       B0B0
00A8   C890                   LHI    9,X'400'           INDEX IN MULTS OF 1024
       0400
00AC   0B33                   SHR    3,3                CLEAR REG 3
00AE   0A3C           NT      AHR    3,12               COUNT OF K'S OF MEMORY
00B0   4099                   STH    9,0(9)             STORE TEST WORD
       0000
00B4   4829                   LH     2,0(9)             TEST IF DATA RETURNS
       0000
00B8   4330                   BZ     FOUND              NO, FOUND LAST ADDR
       00C8
00BC   CA90                   AHI    9,X'400'           ADD 1024 TO COUNT
       0400
00C0   4280                   BC     FOUND              FOR 64 K OF MEMORY BUG OUT
       00C8
00C4   4300                   B      NT
       00AE
00C8   0B9D           FOUND   SHR    9,13               RESULTS LAST MEMORY ADR
00CA   CB30           CONV    SHI    3,10               SUBTRACT OUT TENS
       000A
00CE   4210                   BM     UNITS              BRANCH IF GONE NEGATIVE
       00DA
00D2   CA60                   AHI    6,X'100'           ADD TO TENS DIGIT
```

```
        0100
00D6    4300            B       CONV                FIND ANY MORE TENS
        00CA
00DA    CA30    UNITS   AHI     3,10                BACK TO POSITIVE
        000A
00DE    0B3C            SHR     3,12                SUBTRACT OUT UNITS
00E0    4210            BM      PRNTS               BRANCH IF GONE NEGATIVE
        00EA
00E4    0A6C            AHR     6,12                ADD 1 TO UNITS
00E6    4300            B       UNITS+4             FIND ANY MORE UNITS
        00DE
00EA    4060    PRNTS   STH     6,NUMBER            STORE TO PRINT OUT
        02C6
00EE    C810            LHI     1,MESS3             PRINT OUT MEM SIZE
        02B8
00F2    C830            LHI     3,NUMBER+3
        02C9
00F6    41E0            BAL     14,PRNT
        0220
00FA    088D            LHR     8,13                CONSTANTS FOR LOOP
                *THIS TESTS A ONE THRU A FIELD OF ZEROS
00FC    C200    FIRST   LPSW    *+4
        0100
0100    2000            DC      X'2000',A(*+2)      ALLOW MATCHINE MALEFUNCTIO
        0104
                *                                       INTERRUPTS
0104    085C            LHR     5,12                FIRST TEST WORD
0106    41A0    CWIT0   BAL     10,TESTS            EXECUTE 1 THRU ZEROS
        012A
010A    4280            BC      NEXT1               TEST IF FINISHED
        0112
010E    4300            B       CWIT0               START ACTUAL TESTING
        0106
                *THIS TESTS A ZERO THRU A FIELD OF ONES
0112    C850    NEXT1   LHI     5,X'FFFE'           FIRST TEST
        FFFE
0116    41A0    CW0T1   BAL     10,TESTS            EXECUTE TEST
        012A
011A    4E50            ACH     5,ZERO              ADD LEST SIGNIFICANT BIT B
        02CC
011E    C550            CLHI    5,X'FFFE'           TEST IF FINISHED
        FFFE
0122    4330            BE      NEXT2               GO TO NEXT TEST
        0148
0126    4300            B       CW0T1               START ACTUAL TESTING
        0116
012A    0865    TESTS   LHR     6,5                 LOAD TEST WORD IN WORK REG
012C    C870            LHI     7,START             ADDR OF FIRST AVAIL LOC
        02D2
0130    41F0    STORE   BAL     15,TEST             TEST MEMORY SUBR
        025A
0134    CD60            SLHL    6,1
        0001
0138    4E60            ACH     6,ZERO              REPLACE SHIFTED OUT ONE
```

```
        02CC
013C    9AC7    JUMP2   WDR     12,7            FLASH LAMPS ON DISPLAY
013E    C170            BXLE    7,STORE         NOT FINISHED TEST NEXT LOC
        0130
0142    CD50            SLHL    5,1             SHIFT BIT TO NEXT POSITION
        0001
0146    030A            BR      10              RETURN TO CALLING ROUTINE
                *
0148    C870    NEXT2   LHI     7,START
        02D2
014C    C860            LHI     6,X'FFFF'
        FFFF
0150    41F0    ZAOT    BAL     15,TEST         TEST MEMORY SUBR
        025A
0154    0766            XHR     6,6
0156    41F0            BAL     15,TEST         TEST MEMORY SUBR
        025A
015A    C860            LHI     6,X'FFFF'
        FFFF
015E    41F0            BAL     15,TEST         TEST MEMORY SUBR
        025A
0162    9AC7            WDR     12,7
0164    C170            BXLE    7,ZAOT
        0150
                *THE FOLLOWING TEST WRITES THE ADDRESS OF EACH LOCATION
                *IN THAT LOCATION
                *
0168    C870            LHI     7,START
        02D2
016C    0867    STORE2  LHR     6,7
016E    41F0            BAL     15,TEST         TEST MEMORY SUBR
        025A
0172    9AC7    JUMP    WDR     12,7            FLASH LAMPS ON DISPLAY
0174    C170            BXLE    7,STORE2        TEST IF FINISHED
        016C
0178    9BC0    OKMSG   RDR     12,0            ANY DATA/ADDR SWITCHES
017A    0800            LHR     0,0             OPERATED, NOT CONTINUE
017C    4330            BZ      FIRST+8
        0104
0180    C810            LHI     1,MESS1         START OF OK MSG
        0274
0184    C830            LHI     3,EMES1-1       E11
        0289
0188    41E0            BAL     14,PRNT         GO TO PRINT ROUTINE
        0220
018C    C200            LPSW    OKWT            THEN HALT
        0190
0190    8000    OKWT    DC      X'8000'
0192    00FC            DC      A(FIRST)
0194    9BC0    ERROR   RDR     12,0
0196    0800            LHR     0,0             ANY SWITCHES OPERATED
0198    4230            BNZ     ERRCON          ON DISPLAY PANEL
        01A6
019C    08BB            LHR     11,11           TEST IF SIGN BIT SET
```

```
019E  4210          BM    OVER           MEANS REACHED MAX COUNT
      01A4
01A2  0ABC          AHR   11,12
01A4  030F   OVER   BR    15             RETURN TO TEST ROUTINE
01A6  C810   ERRCON LHI   1,MESS2
      028A
01AA  C830          LHI   3,EMESS2-1
      0295
01AE  41E0   CTYPE  BAL   14,PRNT
      0220
01B2  0B22          SHR   2,2
01B4  0837          LHR   3,7
01B6  41E0          BAL   14,CONVP       PRINT OUT ADDRESS
      01F2
01BA  0836          LHR   3,6
01BC  41E0          BAL   14,CONVP1      CONVERTS NUMBER TO ASCII
      01E8
01C0  4837          LH    3,0(7)
      0000
01C4  41E0          BAL   14,CONVP1      CONVERTS NUMBER TO ASCII
      01E8
01C8  C810          LHI   1,TEMP
      0296
01CC  C830          LHI   3,TEMP+17
      02A7
01D0  41E0          BAL   14,PRNT
      0220
01D4  4830          LH    3,INTFLG
      02CE
01D8  4330          BZ    TIP
      01E6
01DC  0B33          SHR   3,3
01DE  4030          STH   3,INTFLG
      02CE
01E2  C200          LPSW  X'38'
      0038
01E6  030F   TIP    BR    15
01E8  C800   CONVP1 LHI   0,X'2020'
      2020
01EC  4002          STH   0,TEMP(2)
      0296
01F0  0A2D          AHR   2,13
01F2  C810   CONVP  LHI   1,12
      000C
01F6  0803          LHR   0,3
01F8  CC01          SRHL  0,0(1)
      0000
01FC  C400          NHI   0,X'F'
      000F
0200  C500          CLHI  0,X'A'
      000A
0204  4280          BL    *+8
      020C
0208  CA00          AHI   0,7
```

```
        0007
020C    CA00            AHI     0,X'30'
        0030
0210    D202            STB     0,TEMP(2)
        0296
0214    0A2C            AHR     2,12            ADDS ONE TO REG 2
0216    CB10            SHI     1,4
        0004
021A    4310            BNM     CONVP+4
        01F6
021E    030E            BR      14
0220    082C    PRNT    LHR     2,12            LOAD ONE INTO REG 2
0222    DED0            OC      13,TYPEO
        02D0
0226    9DD0    STATI   SSR     13,0            TTY READY TO RECEIVE
0228    42F0            BTC     X'F',STATI      NEXT CHARACTER
        0226
022C    DAD1            WD      13,0(1)         SEND CHARACTER TO TTY
        0000
0230    C110            BXLE    1,STATI         HAVE SENT ALL CHARACTERS
        0226
0234    030E            BR      14              RETURN TO CALLING PROG
0236    9BC0    INTSER  RDR     12,0            EXAMINE FRONT PANEL
                *                               SWITCHES
0238    0800            LHR     0,0             ANY SWITCHES OPERATED
023A    4230            BNZ     PARERR
        024A
023E    0844            LHR     4,4             TEST IF COUNTER OVERFLOW
0240    4210            BM      OVERI
        0246
0244    0A4C            AHR     4,12            INCREMENT COUNTER
0246    C200    OVERI   LPSW    X'38'           RETRN TO PLACE BEFORE INTR
        0038
024A    4070    PARERR  STH     7,INTFLG
        02CE
024E    C810            LHI     1,MESS4
        02A6
0252    C830            LHI     3,EMESS4-1
        02B7
0256    4300            B       CTYPE
        01AE
025A    4067    TEST    STH     6,0(7)          SEND OUT TEST TO MEMORY
        0000
025E    4567            CLH     6,0(7)          TEST IF REPLY MATCHES
        0000
0262    4330            BE      *+16
        0272
0266    40F0            STH     15,PETURN       SAVE RETURN ADDR BEFORE
        02CA
026A    41F0            BAL     15,ERROR        BRANCHING TO ERROR
        0194
026E    48F0            LH      15,RETURN       RESTORE RETURN ADDR
        02CA
0272    030F            BR      15
```

```
0274  8D8A    MESS1   DC    X'8D8A'
0276  4745            DC    C'GE-PAC 30       MEMORY OK '
      2D50
      4143
      2033
      3020
      4D4F
      5259
      204F
      4B20
028A          EMES1   EQU   *
028A  8D8A    MESS2   DC    X'8D8A'
028C  4641            DC    C'FAILURE         '
      494C
      5552
      4520
      2020
0296          EMESS2  EQU   *
0296          TEMP    DS    16
02A6  8D8A    MESS4   DC    X'8D8A'
02A8  5041            DC    C'PARITY          FAILURE  '
      5249
      5459
      2046
      4149
      4C55
      5245
      2020
02B8          EMESS4  EQU   *
02B8  8D8A    MESS3   DC    X'8D8A',C'MEMORY SIZE'
      4D45
      4D4F
      5259
      2053
      495A
      4520
02C6          NUMBER  DS    2                WHERE MEM SIZE IS STORED
02C8  A0CB            DC  @ X'A0CB'          PRINTS SPACE K
02CA          RETURN  DS    2
02CC  0000    ZERO    DC    0
02CE  0000    INTFLG  DC    0                STORAGE LOC FOR INT FLG
0001          ONE     EQU   1
0002          TWO     EQU   2
02D0  9898    TYPEO   DC    X'9898'
02D2          START   END

CONV     00CA
CONVP    01F2
CONVP1   01E8
CTYPE    01AE
CW0TI    0116
CW1T0    0106
EMES1    028A
```

EMESS2	0296
EMESS4	02B8
ERRCON	01A6
ERROR	0194
FIRST	00FC
FOUND	00C8
INTFLG	02CE
INTSER	0236
JUMP	0172
JUMP2	013C
MENTRY	0080
MESS1	0274
MESS2	028A
MESS3	02B8
MESS4	02A6
NEXT1	0112
NEXT2	0148
NT	00AE
NUMBER	02C6
OKMSG	0178
OKWT	0190
ONE	0001
OVER	01A4
OVER1	0246
PARERR	024A
PRNT	0220
PRNTS	00EA
RETURN	02CA
START	02D2
STAT1	0226
STORE	0130
STORE2	016C
TEMP	0296
TEST	025A
TESTS	012A
TIP	01E5
TWO	0002
TYPEO	02D0
UNITS	00DA
ZAOT	0150
ZERO	02CC

APPENDIX 3

AUTOLOAD MARK III MEMORY TEST LISTING

```
                        OPT     PASS2,PRINT,PUNCH,STOP
                *
                * 06-034 MEMORY TEST FOR AUTOLOAD PROCESSOR
                * NEW MEMORY TEST INCLUDES-
                * 1. PRINOUT ON FAILURE
                * 2 ENABLES MACHINE MALFUNCTIONS INTERRUPT FOR PARITY
                * FAILURE DECTION
                * 3. IMPOSES A MORE SEVERE TEST PATTERN
                *
0050                    ORG     X'50'           FOR AUTOLOAD PROCESSOR
                *
                *
0050  C8D0      MENTRY  LHI     13,TWO          DEVICE NUM OF TTY FOR MSG
      0002
0054  C8C0              LHI     12,ONE          DEVICE NUM OF DISPLAY
      0001
0058  0B44              SHR     4,4             CLEAR FOR PARITY INT CNT
005A  4040              STH     4,X'3C'
      003C
005E  C800              LHI     0,INTSER        STORE ADDRS OF INTR SERVIC
      01DE
0062  4000              STH     0,X'3E'         ROUTINE
      003E
0066  0BBB              SHR     11,11           CLEAR ERROR COUNTER
0068  C860              LHI     6,X'B0B0'       MEM SIZE PRINTOUT-00
      B0B0
006C  C890              LHI     9,X'400'        INDEX IN MULTS OF 1024
      0400
0070  0B33              SHR     3,3             CLEAR REG 3
0072  0A3C      NT      AHR     3,12            COUNT OF K'S OF MEMORY
0074  4099              STH     9,0(9)          STORE TEST WORD
      0000
0078  4829              LH      2,0(9)          TEST IF DATA RETURNS
      0000
007C  4S30              BZ      FOUND           NO, FOUND LAST ADDR
      008C
0080  CA90              AHI     9,X'400'        ADD 1024 TO COUNT
      0400
0084  4280              BC      FOUND           FOR 64 K OF MEMORY BUG OUT
      008C
0088  4300              B       NT
      0072
008C  0B9D      FOUND   SHR     9,13            RESULTS LAST MEMORY ADR
008E  CB30      CONV    SHI     3,10            SUBTRACT OUT TENS
      000A
0092  4210              BM      UNITS           BRANCH IF GONE NEGATIVE
      009E
0096  CA60              AHI     6,X'100'        ADD TO TENS DIGIT
      0100
009A  4300              B       CONV            FIND ANY MORE TENS
      008E
009E  CA30      UNITS   AHI     3,10            BACK TO POSITIVE
      000A
00A2  0B3C              SHR     3,12            SUBTRACT OUT UNITS
```

```
00A4   4210                 BM     PRNTS              BRANCH IF GONE NEGATIVE
       00AE
00A8   0A6C                 AHR    6,12               ADD 1 TO UNITS
00AA   4300                 B      UNITS+4            FIND ANY MORE UNITS
       00A2
00AE   4060       PRNTS     STH    6,NUMBER           STORE TO PRINT OUT
       0256
00B2   C810                 LHI    1,MESS3            PRINT OUT MEM SIZE
       0248
00B6   C830                 LHI    3,NUMBER+3
       0259
00BA   41E0                 BAL    14,PRNT
       01C8
00BE   088D                 LHR    8,13               CONSTANTS FOR LOOP
                  *THIS TESTS A ONE THRU A FIELD OF ZEROS
00C0   C200       FIRST     LPSW   *+4
       00C4
00C4   2000                 DC     X'2000',A(*+2)     ALLOW MATCHINE MALEFUNCTIO
       00C8
                  *                                      INTERRUPTS
00C8   085C                 LHR    5,12               FIRST TEST WORD
00CA   41A0       CWITØ     BAL    10,TESTS           EXECUTE 1 THRU ZEROS
       00EE
00CE   4280                 BC     NEXT1              TEST IF FINISHED
       00D6
00D2   4300                 B      CWITØ              START ACTUAL TESTING
       00CA
                  *THIS TESTS A ZERO THRU A FIELD OF ONES
00D6   C850       NEXT1     LHI    5,X'FFFE'          FIRST TEST
       FFFE
00DA   41A0       CWØTI     BAL    10,TESTS           EXECUTE TEST
       00EE
00DE   4E50                 ACH    5,ZERO             ADD LEST SIGNIFICANT BIT B
       025C
00E2   C550                 CLHI   5,X'FFFE'          TEST IF FINISHED
       FFFE
00E6   4330                 BE     NEXT2              GO TO NEXT TEST
       010C
00EA   4300                 B      CWØTI              START ACTUAL TESTING
       00DA
00EE   0865       TESTS     LHR    6,5                LOAD TEST WORD IN WORK REG
00F0   C870                 LHI    7,START            ADDR OF FIRST AVAIL LOC
       0260
00F4   41F0       STORE     BAL    15,TEST            TEST MEMORY SUBR
       01EA
00F8   CD60                 SLHL   6,1
       0001
00FC   4E60                 ACH    6,ZERO             REPLACE SHIFTED OUT ONE
       025C
0100   9AC7       JUMP2     WDR    12,7               FLASH LAMPS ON DISPLAY
0102   C170                 BXLE   7,STORE            NOT FINISHED TEST NEXT LOC
       00F4
0106   CD50                 SLHL   5,1                SHIFT BIT TO NEXT POSITION
       0001
```

```
010A  030A               BR     10             RETURN TO CALLING ROUTINE
                   *
010C  C870       NEXT2   LHI    7,START
      0260
0110  C860               LHI    6,X'FFFF'
      FFFF
0114  41F0       ZAOT    BAL    15,TEST        TEST MEMORY SUBR
      01EA
0118  0766               XHR    6,6
011A  41F0               BAL    15,TEST        TEST MEMORY SUBR
      01EA
011E  C860               LHI    6,X'FFFF'
      FFFF
0122  41F0               BAL    15,TEST        TEST MEMORY SUBR
      01EA
0126  9AC7               WDR    12,7
0128  C170               BXLE   7,ZAOT
      0114
                 *THE FOLLOWING TEST WRITES THE ADDRESS OF EACH LOCATION
                 *IN THAT LOCATION
                 *
012C  C870               LHI    7,START
      0260
0130  0867       STORE2  LHR    6,7
0132  41F0               BAL    15,TEST        TEST MEMORY SUBR
      01EA
0136  9AC7       JUMP    WDR    12,7           FLASH LAMPS ON DISPLAY
0138  C170               BXLE   7,STORE2       TEST IF FINISHED
      0130
013C  C810       OKMSG   LHI    1,MESS1        START OF OK MESSAGE
      0204
0140  C830               LHI    3,EMES1-1      E11
      0219
0144  41E0               BAL    14,PRNT        GO TO PRINT ROUTINE
      01C8
0148  C200               LPSW   OKWT           THEN HALT
      014C
014C  0000       OKWT    DC     X'0000'
014E  00C0               DC     A(FIRST)
0150  C810       ERRCON  LHI    1,MESS2
      021A
0154  C830               LHI    3,EMESS2-1
      0225
0158  41E0       CTYPE   BAL    14,PRNT
      01C8
015C  0B22               SHR    2,2
015E  0837               LHR    3,7
0160  41E0               BAL    14,CONVP       PRINT OUT ADDRESS
      019A
0164  0836               LHR    3,6
0166  41E0               BAL    14,CONVP1      CONVERTS NUMBER TO ASCII
      0190
016A  4837               LH     3,0(7)
      0000
```

```
016E  41E0            BAL   14,CONVP1         CONVERTS NUMBER TO ASCII
      0190
0172  C810            LHI   1,TEMP
      0226
0176  C830            LHI   3,TEMP+17
      0237
017A  41E0            BAL   14,PRNT
      01C8
017E  4830            LH    3,X'24'
      0024
0182  C430            NHI   3,X'2000'
      2000
0186  4230            BNZ   TIP
      018E
018A  C200            LPSW  X'38'
      0038
018E  030F    TIP     BR    15
0190  C800    CONVP1  LHI   0,X'2020'
      2020
0194  4002            STH   0,TEMP(2)
      0226
0198  0A2D            AHR   2,13
019A  C810    CONVP   LHI   1,12
      000C
019E  0803            LHR   0,3
01A0  CC01            SRHL  0,0(1)
      0000
01A4  C400            NHI   0,X'F'
      000F
01A8  C500            CLHI  0,X'A'
      000A
01AC  4280            BL    *+8
      01B4
01B0  CA00            AHI   0,7
      0007
01B4  CA00            AHI   0,X'30'
      0030
01B8  D202            STB   0,TEMP(2)
      0226
01BC  0A2C            AHR   2,12              ADDS ONE TO REG 2
01BE  CB10            SHI   1,4
      0004
01C2  4310            BNM   CONVP+4
      019E
01C6  030E            BR    14
01C8  082C    PRNT    LHR   2,12              LOAD ONE INTO REG 2
01CA  DED0            OC    13,TYPEO
      025E
01CE  9DD0    STAT1   SSR   13,0              TTY READY TO RECEIVE
01D0  42F0            BTC   X'F',STAT1        NEXT CHARACTER
      01CE
01D4  DAD1            WD    13,0(1)           SEND CHARACTER TO TTY
      0000
01D8  C110            BXLE  1,STAT1           HAVE SENT ALL CHARACTERS
```

```
      01CE
01DC  030E                BR    14                  RETURN TO CALLING PROG
01DE  C810     PARERR     LHI   1,MESS4
      0236
01E2  C830                LHI   3,EMESS4-1
      0247
01E6  4300                B     CTYPE
      0158
01EA  4067     TEST       STH   6,0(7)              SEND OUT TEST TO MEMORY
      0000
01EE  4567                CLH   6,0(7)              TEST IF REPLY MATCHES
      0000
01F2  4330                BE    *+16
      0202
01F6  40F0                STH   15,RETURN           SAVE RETURN ADDR BEFORE
      025A
01FA  41F0                BAL   15,ERROR            BRANCHING TO ERROR
      0150
01FE  48F0                LH    15,RETURN           RESTORE RETURN ADDR
      025A
0202  030F                BR    15
0204  8D8A     MESS1      DC    X'8D8A'
0206  4745                DC    C'GE-PAC 30         MEMORY OK '
      2D50
      4143
      2033
      3020
      4D4F
      5259
      204F
      4B20
021A           EMES1      EQU   *
021A  8D8A     MESS2      DC    X'8D8A'
021C  4V41                DC    C'FAILURE                    '
      494C
      5552
      4520
      2020
0226           EMESS2     EQU   *
0226           TEMP       DS    16
0236  8D8A     MESS4      DC    X'8D8A'
0238  5041                DC    C'PARITY            FAILURE   '
      5249
      5459
      2046
      4149
      4C55
      5245
      2020
0248           EMESS4     EQU   *
0248  8D8A     MESS3      DC    X'8D8A',C'MEMORY SIZE'
      4D45
      4D4F
```

```
             5259
             2053
             495A
             4520
0256                NUMBER   DS    2           WHERE MEM SIZE IS STORED
0258  A0CB                   DC    X'A0CB'     PRINTS SPACE K
025A                RETURN   DS    2
025C  0000          ZERO     DC    0
0001                ONE      EQU   1
0002                TWO      EQU   2
0150                ERROR    EQU   ERRCON
01DE                INTSER   EQU   PARERR
025E  9898          TYPEO    DC    X'9898'
0260                START    END
```

```
CONV      008E
CONVP     019A
CONVP1    0190
CTYPE     0158
CW0TI     00DA
CW1T0     00CA
EMES1     021A
EMESS2    0226
EMESS4    0248
ERRCON    0150
ERROR     0150
FIRST     00C0
FOUND     008C
INTSER    01DE
JUMP      0136
JUMP2     0100
MENTRY    0050
MESS1     0204
MESS2     021A
MESS3     0248
MESS4     0236
NEXT1     00D6
NEXT2     010C
NT        0072
NUMBER    0256
OKMSG     013C
OKWT      014C
ONE       0001
PARERR    01DE
PRNT      01C8
PRNTS     00AE
RETURN    025A
START     0260
STAT1     01CE
STORE     00F4
STORE2    0130
TEMP      0226
TEST      01EA
TESTS     00EE
```

TIP	018E
TWO	0002
TYPEO	025E
UNITS	009E
ZAOT	0114
ZERO	025C

GE-PAC 30
OPERATING INSTRUCTIONS FOR THE
TELETYPEWRITER/TERMINET TEST PROGRAMS

1. DESCRIPTION

The KSR programs store characters that have been input from the keyboard, check the parity on each character, and print each character. The action of the BREAK/INTERRUPT KEY is also tested. The ASR versions also punch the input data on tape and read back the punched tape, comparing each character with the original input.

2. LOADING INFORMATION

All tapes are M14 tapes which are loaded with the 50 Sequence Loader or the 68 Sequence Loader. For the KSR programs, change location X'5A' or X'72' to X'200'. For the ASR programs, change location X'5A' or X'72' to X'34D'.

The TermiNet 300 keyboard must be set up as follows:

ONLINE, READY pushbuttons on (lights lit)
TRANSPARENCY switch to ON.
INHIBIT switch to PRINT.
RATE switch to 30.
LINE FEED to 1.
AUTO LF to OFF.

For an ASR 35 Teletypewriter, set the MODE switch to KT.

Load the program tape as follows:

1. Load the tape on the reader with the first character over the read fingers:

06-004M14	ASR Teletypewriter Test
96-004M14	ASR TermiNet Test
06-083M14	KSR Teletypewriter Test
96-083M14	KSR TermiNet Test

2. Enter X'50' or X'68' on the console switches, set the MODE CONTROL dial to ADDRESS, depress the EXECUTE button.

3. Depress the INITIALIZE button.

4. Set the MODE CONTROL dial to RUN and depress the EXECUTE button.

5. Start the Teletypewriter reader by moving the START-STOP-FREE switch to START. Start the TermiNet 300 reader by depressing the RUN button.

6. Stop the reader once blank tape begins to pass over the read fingers.

As soon as the last character on the tape is read, the loader will transfer to the start of the test program and halt.

Set the TERMINET 300 TRANSPARENCY switch to OFF.

3. OPERATING INSTRUCTIONS

Whenever an error occurs, the program will halt with console lights 8 through 15 lit. Register 15 contains a program address. This address minus 4 bytes points to the location in the program where the error occurred (Appendix 1 contains listings of the test programs).

1. Rotate the MODE CONTROL dial to HALT and depress the INITIALIZE button.

2. Rotate the MODE CONTROL dial to RUN and depress the EXECUTE button. The program outputs a carriage return and line feed.

3. Enter characters on the keyboard. Do not hit the BREAK, HERE IS, or any of the tape control keys. The receipt of 73 characters or a carriage return causes the program to terminate keyboard input.

The KSR test programs will print the characters just read in from the keyboard, then go to Step 6.

4. The program outputs the message "PUNCH TAPE" and halts. Depress the EXECUTE button. The characters just read in from the keyboard will be punched on tape and printed. Leader and trailer tape will also be punched on the tape.

5. The program outputs the message "READ TAPE" and halts. Load the tape just punched in the reader, with leader tape over the read fingers. Depress the EXECUTE button. The tape will be read and its contents printed. If an error is detected, the program outputs the message "TAPE ERROR" and halts. Depressing the EXECUTE button will cause the program to proceed to Step 6.

6. The program outputs the message "DEPRESS BRK". Depress the BREAK/INTERRUPT button. The message "BRK OK" will be printed. If the BREAK/INTERRUPT button is not functioning properly, the program will hang up in a Sense Status loop.

7. To repeat the test, depress the EXECUTE button. Or restart the program at X'80'.

The test programs assume a device number of X'02'. Location X'8A' contains this number and must be changed if the device being tested is not X'02'.

NOTE:

To read the program tape on the TermiNet 300 at 120 characters/ second, the following changes apply (see LOADING INFORMATION):

Set the RATE switch to 120 (the keyboard will automatically go to STAND-BY).

Change location X'0078' (BINDV) to X'0297'.

When the tape stops, set the TRANSPARENCY switch to OFF, the RATE switch to 30, and depress the ONLINE button.

```
                          OPT    PASS2,PRINT,NOPNCH,STOP
                   *
                   *
                   *      50 SEQUENCE LOADER
                   *      FOR ALL GE-PAC 30 PROCESSORS
                   *
0050                      ORG    X'50'
0050  C820         LOAD   LHI    2,X'80'          LOADS TAPE FROM X'80'
      0080
0054  C830                LHI    3,1              THRU X'CF'
      0001
0058  C840                LHI    4,X'CF'
      00CF
005C  D3A0                LB     10,BINDV         NOTE THAT LOCATION X'5A'
      0078
0060  DEA0                OC     10,BINDV+1       MUST BE CHANGED FOR ALL
      0079
0064  9DAE         SENSE  SSR    10,14            M14 TEST PROGRAM TAPES
0066  08EE                LHR    14,14
0068  4230                BTC    3,SENSE
      0064
006C  DBA2                RD     10,0(2)
      0000
0070  C120                BXLE   2,SENSE
      0064
0074  4300                B      X'80'
      0080
0078  0294         BINDV  DC     X'0294'          DEVICE DEFINITIONS ARE
007A  0298         BOUTDV DC     X'0298'          FOR TTY
007C  0294         SINDV  DC     X'0294'
007E  0298         SOUTDV DC     X'0298'
                   *
                   *
                   *      68 SEQUENCE LOADER
                   *      FOR 30-2 PROCESSORS ONLY
                   *
0068                      ORG    X'68'
0068  C830                LHI    3,1              LOADS TAPE FROM X'80' THRU
      0001
006C  D3A0                LB     10,BINDV         X'CF'.  LOCATION X'72'
      0078
0070  D500                AL     0,X'CF'          MUST BE CHANGED FOR ALL
      00CF
0074  4300                B      X'80'            M14 TEST PROGRAM TAPES
      0080
                   *      DEVICE DEFINITIONS ARE SAME AS FOR 50 SEQUENCE
                   *
                   *
                   * DEFINITIONS FOR TERMINET ARE SAME AS FOR TTY
                   * HIGH SPEED PAPER TAPE READER= NN99 (BINDV,SINDV)
                   * HIGH SPEED PAPER TAPE PUNCH= NN9A (BOUTDV)
                   * CARD READER= NN20 (SINDV)
                   *      WHERE NN= DEVICE NUMBER
                   *
```

APPENDIX 1

```
                         OPT     PASS2,PRINT,PUNCH,STOP
                  *         ASR TELETYPEWRITER TEST PROGRAM
                  *
                  * 06-004R03
                  *
                  * COPPER TELE TYPEWRITER INTERFACE CONTROLLER
                  *
                  *
0080                     ORG     X'80'              START PROGRAM AT LOC 80
                  *
0080   C200       START  LPSW    *+4                HALT UPON ENTERING
       0084
0084   8000              DC      X'8000',A(*+2)     PROGRAM
       0088
0088   C8D0              LHI     13,2               LOAD TTY DEVICE NUMBER
       0002
008C   0B11              SHR     1,1                CLEAR CHAR COUNTER REGISTE
008E   4010              STH     1,X'44'            TURN-OFF EXT PSW INTERRUPT
       0044
0092   C820              LHI     2,ERRORA           CONTAINS ADDR OF ERROR ROU
       02BA
0096   C8B0              LHI     11,1               R11 ALWAYS CONTAINS 1
       0001
009A   9DD0              SSR     13,0               TEST IF TTY AVAILABLE
009C   C700              XHI     0,8                CORRECTLY INITIALIZED
       0008
00A0   4330              BZ      *+6                ONLY BSY STATUS BIT =1
       00A6
00A4   01F2              BALR    15,ERROR
00A6   C800              LHI     0,ERRORA           LOAD NEW EXT PSW INT
       02BA
00AA   4000              STH     0,X'46'            WITH ADDR OF ERROR
       0046
00AE   C8F0              LHI     15,NOINT           ERROR ADDR FOR NO INT
       00BE
00B2   DED0              OC      13,WRTBLK          WRITE,BLOCK,DISABLE
       0342
00B6   C200              LPSW    INIT
       00BA
00BA   4000       INIT   DC      X'4000',A(*+2)
       00BE
00BE   DAD0       NOINT  WD      13,TAPEOF          SEND TAPE OFF TO TTY
       0347
                  *
                  *                                 AND SHOULD NOT GENERATE
                  *                                 EXT PSW INTERRUPT
00C2   C8A0              LHI     10,MESS1           SENDS CR AND LF TO TTY
       0300
00C6   C8C0              LHI     12,MMESS1          END ADDR OF MSG
       0301
00CA   41F0              BAL     15,TYPEMG          SUBROUTINE TO TYPE MSG
       02CA
00CE   C800              LHI     0,CLEAR            BRANCH LOC AT INTERRUPT
       00DC
```

```
00D2  4000         STH    0,X'46'         STORE IN NEW EXT PSW INT
      0046
00D6  DED0         OC     13,RECV         ENABLE,UNBLK,READ
      0343
00DA  01F2         BALR   15,ERROR        NO INT,GO TO ERROR
00DC  9F00  CLEAR  AIR    0,0             CLEAR ATTEN FLOP
00DE  C800         LHI    0,STORE         BRANCH LOC AT INTERRUPT
      00F2
00E2  4000         STH    0,X'46'         STORE IN NEW EXT PSW INT
      0046
00E6  C200         LPSW   *+4             ALLOW EXT INTERRUPTS
      00EA
00EA  4000         DC     X'4000',A(*+2)
      00EE
00EE  4300         B      *               WAIT HERE FOR INTERRUP
      00EE
00F2  9F0C  STORE  AIR    0,12            ACKNOWLEDGE INTERRUPT
00F4  050D         CLHR   0,13            TEST FOR SAME DEVICE NUMBE
00F6  4330         BE     *+6             R11 IS LOADED WITH DEVICE
      00FC
00FA  01F2         BALR   15,ERROR
00FC  08CC         LHR    12,12           TEST FOR STATUS BYTE ZERO
00FE  4330         BZ     *+6             NOT,GO TO ERROR
      0104
0102  01F2         BALR   15,ERROR
0104  9BD0         RDR    13,0            READ TTY CHARACTER
            *
            *FOLLOWING LOOP TESTS FOR EVEN PARITY
            *
0106  08C0         LHR    12,0            LOAD CHAR IN TEST REGISTER
0108  0BEE         SHR    14,14           PARITY COUNTER-EVEN
010A  C840         LHI    4,-9            INITIALIZE LOOP COUNT
      FFF7
010E  0A4B  NXTBIT AHR    4,ONEREG        DECREMENT LOOP COUNT
0110  4330         BZ     TESTP           FINISHED WHEN ZERO
      0122
0114  CCC0         SRHL   12,1            TEST BIT FOR ONE OR ZERO
      0001
0118  4380         BFC    8,NXTBIT        BRANCH IF ONE
      010E
011C  07EB         XHR    14,ONEREG       ADD ONE TO PARITY COUNT
011E  4300         B      NXTBIT
      010E
0122  08EE  TESTP  LHR    14,14           TEST FINAL PARITY COUNT
0124  4330         BZ     *+6             EVEN PARITY WILL MAKE
      012A
0128  01F2         BALR   15,ERROR        R14 EQUAL TO ZERO
            *
012A  C500         CLHI   0,X'8D'         TEST FOR CARRIAGE RETURN
      008D
012E  4330         BE     PNCH            SIGNALS END OF INPUT
      0144
0132  D201         STB    0,BUFR(1)       STORE CHARACTER
      0352
```

```
0136  0A1B           AHR   1,ONEREG            INCREMENT REGISTER
0138  C510           CLHI  1,72                TEST FOR MAX OF 72 CHARS
      0048
013C  4380           BNL   PNCH                YES GO TO NEXT ROUTINE
      0144
0140  C200           LPSW  X'40'               RETURN TO WAIT LOOP
      0040
0144  C8A0    PNCH   LHI   10,MESS1            STRT ADDR OF PNCH TAPE MSG
      0300
0148  C8C0           LHI   12,EMES1            END ADDR OF MSG
      030B
014C  DED0           OC    13,WRITE            WRITE,BLOCK,ENABLE
      0344
0150  41F0           BAL   15,TYPEMG           SUBROUTINE TO TYPE MSG
      02CA
0154  0B1B           SHR   1,ONEREG            ADJUST CHAR COUNT
0156  9F00           AIR   0,0                 ALLOWS WAIT LAMP COME ON
0158  C200           LPSW  LOC1
      015C
015C  8000    LOC1   DC    X'8000',A(NEXT) WAIT FOR NEXT DECISION
      0160
              *
              *
              *THIS SECTION PUNCHES TAPE
              *
0160  C800    NEXT   LHI   0,PCHAR             LOAD NEXT EXT PSW INT
      0182
0164  4000           STH   0,X'46'             WITH ADDR PCHAR
      0046
              *
0168  DED0           OC    13,WRITE            WRITE,BLOCK,ENABLE
      0344
016C  DAD0           WD    13,TAPEON           SEND TAPE ON TO TTY
      0346
0170  41F0           BAL   15,LEADR            PUNCH BLANKS ON TAPE
      02E0
0174  0BAA           SHR   10,10               CLEAR CHAR COUNT REGISTER
0176  C200           LPSW  LOC2
      017A
017A  4000    LOC2   DC    X'4000',A(*+2)      ENABLE EXT DEV INT
      017E
017E  4300           B     *                   WAIT HERE FOR INTRUPT
      017E
0182  9F0C    PCHAR  AIR   0,12                ACKNOWLEDGE INTERRUPT
0184  050D           CLHR  0,13                TEST IF DEVICE NUMBER SAME
0186  4330           BE    *+6                 R11 IS LOADED WITH DEVICE
      018C
018A  01F2           BALR  15,ERROR
              *
018C  DADA           WD    13,BUFR(10)         PUNCH NEXT CHAR
      0352
0190  9DD0           SSR   13,0                OBTAIN STATUS BYT FROM TTY
0192  C500           CLHI  0,8                 TEST BUSY BIT SET
      0008
```

```
0196  4330            BE     *+6               IF NOT SET
      019C
019A  01F2            BALR   15,ERROR          GO TO ERROR
019C  05A1            CLHR   10,1              TEST IF ALL CHARS SENT
019E  4380            BNL    PUNOFF            FINISHED GO TO PUNOFF
      01A8
01A2  0AAB            AHR    10,ONEREG         INCREMENT COUNTER
01A4  C200            LPSW   X'40'             RETURN TO WAIT LOOP
      0040
01A8  41F0    PUNOFF  BAL    15,LEADR          PUNCH BLANKS ON TAPE
      02E0
01AC  DAD0            WD     13,TAPEOF         SEND TAPE OFF TO TTY
      0347
01B0  C8A0            LHI    10,MESS2
      030C
01B4  C8C0            LHI    12,EMES2          END ADDR OF MSG
      0317
01B8  41F0            BAL    15,TYPEMG         SUBROUTINE TO TYPE MSG
      02CA
01BC  9F00            AIR    0,0               ALLOWS WAIT LAMP COME ON
01BE  C200            LPSW   LOC3
      01C2
01C2  8000    LOC3    DC     X'8000',A(*+2)    THIS HALTS PROG TEST
      01C6
              *
              *THIS SECTION READS PUNCHED TAPE
              *
              *AND PRINTS THE CONTENTS OF THE TAPE
              *
              ****PLACE PUNCH TAPE IN READER***
              *
01C6  0BAA    RDTAPE  SHR    10,10             CLEAR COUNTER REGISTER
01C8  40A0            STH    10,ECFLAG         CLEAR ERROR FLAG
      034E
01CC  DED0            OC     13,WRITE          WRITE,BLOCK,ENABLE
      0344
01D0  DAD0            WD     13,XON            SEND XON TO TTY
      0348
              *
01D4  DED0            OC     13,RTAPE          READ,BLOCK,ENABLE
      0345
01D8  C8E0            LHI    14,TEST           SET SWITCH TO TEST
      020C
01DC  40E0            STH    14,THRU+2         FOR LEADING ZERO CHARS
      01EC
01E0  9DDC    RETRN1  SSR    13,12             ACCEPT CHAR WHEN
01E2  08CC            LHR    12,12             BUSY IS EQUAL TO ZERO
01E4  4230            BNZ    RETRN1
      01E0
01E8  9BD0            RDR    13,0
01EA  4300    THRU    B      TEST
      020C
01EE  05A1            CLHR   10,1              TEST IF ALL CHARS SENT
01F0  4380            BNL    FINT              BRANCH WHEN ALL CHAR READ
```

```
          022A
01F4      D3EA                LB      14,BUFR(10)
          0352
01F8      050E                CLHR    0,14
01FA      4330                BE      *+8            TEST FOR CORRECT CHAR
          0202
01FE      41F0                BAL     15,ERRFLG
          02FA
0202      D20A                STB     0,BUFR(10)
          0352
0206      0AAB                AHR     10,ONEREG
0208      4300                B       RTCHAR
          021E
020C      0800      TEST      LHR     0,0            TEST FOR FIRST NON-ZERO CH
020E      4330                BZ      RTCHAR
          021E
0212      C8E0                LHI     14,THRU+4
          01EE
0216      40E0                STH     14,THRU+2
          01EC
021A      4300                B       THRU+4
          01EE
021E      9DDC      RTCHAR    SSR     13,12          AS SOON AS
0220      C5C0                CLHI    12,8           RD COMMAND IS SENT
          0008
0224      4330                BE      RETRN1         BUSY SHOULD GO TO A ONE
          01E0
0228      01F2                BALR    15,ERROR
022A      DED0      FINT      OC      13,WRITE       WRITE,BLOCK,ENABLE
          0344
022E      9DD0      SENS1     SSR     13,0           READ STATUS BYTE FROM TTY
0230      4290                BTC     9,SENS1        LOOP UNTIL BUSY=0
          022E
0234      DAD0                WD      13,XOFF        SEND XOFF TO TTY
          0349
0238      C8A0                LHI     10,MESS1       SENDS CR AND LF TO TTY
          0300
023C      C8C0                LHI     12,MMESS1      END ADDR OF MSG
          0301
0240      41F0                BAL     15,TYPEMG      SUBROUTINE TO TYPE MSG
          02CA
0244      0BAA                SHR     10,10          CLEAR BUFR COUNTER
0246      DADA      TYCHAR    WD      13,BUFR(10)
          0352
024A      9DDC      NEXT2     SSR     13,12
024C      4290                BTC     9,NEXT2
          024A
0250      08CC                LHR     12,12          TEST FOR BUSY=0
0252      4330                BZ      *+6
          0258
0256      01F2                BALR    15,ERROR
0258      C8AA                LHI     10,1(10)       INCREMENT BYTE COUNT
          0001
025C      051A                CLHR    1,10
```

```
025E  4380            BNL   TYCHAR          TEST FOR ALL CHARS PRINTED
      0246
0262  4800            LH    0,ECFLAG        TEST FOR ERROR IN
      034E
0266  4330            BZ    BREAK           READING TAPE
      0278
026A  C8A0            LHI   10,TRMSG
      0336
026E  C8C0            LHI   12,ETRMSG       END ADDR OF MSG
      0341
0272  41F0            BAL   15,TYPEMG       SUBROUTINE TO TYPE MSG
      02CA
0276  01F2            BALR  15,ERROR
                *
                *THE FOLLOWING TESTS THE OPERATION OF THE BREAK KEY
                *
0278  C8A0      BREAK LHI   10,MESS3
      0318
027C  C8C0            LHI   12,EMES3        END ADDR OF MSG
      0327
0280  41F0            BAL   15,TYPEMG       SUBROUTINE TO TYPE MSG
      02CA
0284  9DD0      SENS2 SSR   13,0            READ STATUS BYTE FROM TTY
0286  C700            XHI   0,X'24'         TEST EXAM AND BREAK
      0024
028A  4230            BNZ   SENS2           LOOP UNTIL BRK=1
      0284
028E  9DD0      SENS3 SSR   13,0            READ STATUS BYTE FROM TTY
0290  C700            XHI   0,X'24'         TEST EXAM AND BREAK
      0024
0294  4330            BZ    SENS3           LOOP UNTIL BRK=0
      028E
0298  C8A0            LHI   10,MESS4        THIS ROUTINE PRINTS
      0328
029C  C8C0            LHI   12,EMES4        END ADDR OF MSG
      032F
02A0  41F0            BAL   15,TYPEMG       SUBROUTINE TO TYPE MSG
      02CA
02A4  C8A0            LHI   10,MESS5        THIS ROUTINE PRINTS
      0330
02A8  C8C0            LHI   12,EMES5        END ADDR OF MSG
      0335
02AC  41F0            BAL   15,TYPEMG       SUBROUTINE TO TYPE MSG
      02CA
02B0  DED0            OC    13,RTAPE        CHANGE MODE TO READ
      0345
02B4  9F00            AIR   0,0             RESET ATTEN FLOP
02B6  4300            B     START
      0080
02BA  9F44     ERRORA AIR   4,4             CLEAR ATTEN FLOP
02BC  DAB0            WD    ONEREG,ERDSY    DISPLAY ONES ON
      034A
02C0  C200            LPSW  HALT            PANEL WHEN ERROR OCCURS
      02C4
```

```
02C4   8000     HALT     DC     X'8000,A(NEXT5)
       02C8
02C8   030F     NEXT5    BR     15
02CA   9DD0     TYPEMG   SSR    13,0            READ STATUS
02CC   4290              BTC    9,TYPEMG        TEST BUSY AND DU
       02CA
                *                                  TTY READ WHEN BSY=0
02D0   DADA              WD     13,0(10)        SENDS ONE CHAR TO TTY
       0000
02D4   C1A0              BXLE   10,TYPEMG       TEST FOR END OF MSG
       02CA
02D8   9DD0     ENDMG    SSR    13,0            READ STATUS - WAITING FOR
02DA   4290              BTC    9,ENDMG         LAST CHAR TO BE DIGESTED
       02D8
02DE   030F              BR     15              RETURN TO CALLING ROUTINE
02E0   0BAA     LEADR    SHR    10,10
02E2   C8C0              LHI    12,40           ALLOWS FOR FORTY BLANKS
       0028
02E6   9DD0     LEADR1   SSR    13,0
02E8   4290              BTC    9,LEADR1
       02E6
02EC   C0A0              BXH    10,OVER
       02F8
02F0   DAD0              WD     13,ZERO
       034C
02F4   4300              B      LEADR1
       02E6
02F8   030F     OVER     BR     15
02FA   40E0     ERRFLG   STH    14,ECFLAG
       034E
02FE   030F              BR     15
0300   8D8A     MESS1    DC     X'8D8A'
0301            MMESS1   EQU    *-1
0302   5055              DC     C'PUNCH         TAPE'
       4E43
       4820
       5441
       5045
030B            EMES1    EQU    *-1
030C   8D8A     MESS2    DC     X'8D8A'
030E   5245              DC     C'READ          TAPE'
       4144
       2054
       4150
       4520
0317            EMES2    EQU    *-1
0318   8D8A     MESS3    DC     X'8D8A'
031A   8D8A              DC     X'8D8A'
031C   4445              DC     C'DEPRESS       BRK'
       5052
       4553
       5320
       4252
       4B20
```

```
0327            EMES3    EQU   *-1
0328    8D8A    MESS4    DC    X'8D8A'
032A    4252             DC    C'BRK          OK'
        4B20
        4F4B
032F            EMES4    EQU   *-1
0330    8D8A    MESS5    DC    X'8D8A'
0332    454E             DC    C'END'
        4420
0335            EMES5    EQU   *-1
0336    8D8A    TRMSG    DC    X'8D8A'
0338    5441             DC    C'TAPE         ERROR'
        5045
        2045
        5252
        4F52
0341            ETRMSG   EQU   *-1
0342    9864    WRTBLK   DC    X'9864'        WRITE,BLOCK,DISABLE
0343            RECV     EQU   WRTBLK+1
0344    5854    WRITE    DC    X'5854'
0345            RTAPE    EQU   WRITE+1
0346    1214    TAPEON   DC    X'1214'
0347            TAPEOF   EQU   TAPEON+1
0348    1113    XON      DC    X'1113'
0349            XOFF     EQU   XON+1
034A    FFFF    ERDSY    DC    X'FFFF'
034C    0000    ZERO     DC    0
000B            ONEREG   EQU   11
0002            ERROR    EQU   2
034E            ECFLAG   DS    2
0350            STATUS   DS    2
0352            BUFR     DS    72
039A                     END

BREAK     0278
BUFR      0352
CLEAR     00DC
ECFLAG    034E
EMES1     030B
EMES2     0317
EMES3     0327
EMES4     032F
EMES5     0335
ENDMG     02D8
ERDSY     034A
ERRFLG    02FA
ERROR     0002
ERRORA    02BA
ETRMSG    0341
FINT      022A
HALT      02C4
INIT      00BA
LEADR     02E0
LEADR1    02E6
```

LOC1	015C
LOC2	017A
LOC3	01C2
MESS1	0300
MESS2	030C
MESS3	0318
MESS4	0328
MESS5	0330
MMESS1	0301
NEXT	0160
NEXT2	024A
NEXT5	02C8
NOINT	00BE
NXTBIT	010E
ONEREG	000B
OVER	02F8
PCHAR	0182
PNCH	0144
PUNOFF	01A8
RDTAPE	01C6
RECV	0343
RETRN1	01E0
RTAPE	0345
RTCHAR	021E
SENS1	022E
SENS2	0284
SENS3	028E
START	0080
STATUS	0350
STORE	00F2
TAPEOF	0347
TAPEON	0346
TEST	020C
TESTP	0122
THRU	01EA
TRMSG	0336
TYCHAR	0246
TYPEMG	02CA
WRITE	0344
WRTBLK	0342
XOFF	0349
XON	0348
ZERO	034C

```
                              OPT    PASS2,PRINT,PUNCH,STOP,LAB=ASRTRM
                      *
                      * ASR TERMINET TEST PROGRAM
                      *
                      * 96-004R03       AUGUST, 1970
                      *
                      * COPPER TELETYPEWRITER INTERFACE CONTROLLER
                      *
0080                          ORG    X'80'
000B                  ONEREG  EQU    11
0002                  ERROR   EQU    2
0080  C200            START   LPSW   *+4              HALT UPON ENTERING PROGRAM
      0084
0084  8000                    DC     X'8000',A(*+2)
      0088
0088  C8D0                    LHI    13,2             LOAD TTY DEVICE NUMBER
      0002
008C  0B11                    SHR    1,1              CLEAR CHAR COUNT
008E  4010                    STH    1,X'44'          NEW EXT INT PSW
      0044
0092  C820                    LHI    2,ERRORA         ADDRESS OF ERROR ROUTINE
      02BA
0096  C8B0                    LHI    11,1             R11= CONSTANT 1
      0001
009A  9DD0                    SSR    13,0             TEST IF TTY AVAILABLE
009C  C700                    XHI    0,X'18'          CORRECTLY INITIALIZED
      0018
00A0  4330                    BZ     *+6              ONLY BUSY BIT SET
      00A6
00A4  01F2                    BALR   15,ERROR
00A6  C800                    LHI    0,ERRORA         NEW EXT INT PSW
      02BA
00AA  4000                    STH    0,X'46'
      0046
00AE  C8F0                    LHI    15,NOINT         NO INT ERR ADDRESS
      00BE
00B2  DED0                    OC     13,WRTBLK        WRITE,BLOCK,DISABLE
      0342
00B6  C200                    LPSW   INIT             NO INT SHOULD BE GENERATED
      00BA
00BA  4000            INIT    DC     X'4000',A(*+2)
      00BE
00BE  DAD0            NOINT   WD     13,TAPEOF
      0347
                      *
00C2  C8A0                    LHI    10,MESS1         OUTPUT CR,LF
      0300
00C6  C8C0                    LHI    12,MMESS1
      0301
00CA  41F0                    BAL    15,TYPEMG
      02CA
00CE  C800                    LHI    0,CLEAR          NEW EXT INT PSW
      00DC
00D2  4000                    STH    0,X'46'
```

```
         0046
00D6     DED0              OC     13,RECV           ENABLE,UNBLOCK,READ
         0343
00DA     01F2              BALR   15,ERROR          NO INT- ERROR
00DC     9F00     CLEAR    AIR    0,0               CLEAR ATN
00DE     C800              LHI    0,STORE           NEW EXT INT PSW
         00F2
00E2     4000              STH    0,X'46'
         0046
00E6     C200              LPSW   *+4               WAIT FOR INTERRUPTS
         00EA
00EA     4000              DC     X'4000',A(*+2)    FROM OPERATOR TYPING
         00EE
00EE     4300              B      *
         00EE
00F2     9F0C     STORE    AIR    0,12              ACKNOWLEDGE, CHECK DEVICE
00F4     050D              CLHR   0,13
00F6     4330              BZ     X'104'            OK
         0104
00FA     01F2              BALR   15,ERROR          BAD
00FC     08CC              LHR    12,12             STATUS BYTE ZERO?
00FE     4330              BZ     *+6               YES
         0104
0102     01F2              BALR   15,ERROR          NO
0104     9BD0              RDR    13,0              READ THE CHARACTER
                  *
                  * TEST CHARACTER FOR EVEN PARITY
                  *
0106     08C0              LHR    12,0              R12= CHARACTER
0108     0BEE              SHR    14,14             PARITY COUNTER
010A     C840              LHI    4,-9              LOOP COUNT
         FFF7
010E     0A4B     NXTBIT   AHR    4,ONEREG          DECR LOOP COUNT
0110     4330              BZ     TESTP             DONE WHEN ZERO
         0122
0114     CCC0              SRHL   12,1              TEST BIT FOR 1 OR 0
         0001
0118     4380              BFC    8,NXTBIT          BRANCH IF ONE
         010E
011C     07EB              XHR    14,ONEREG         INCR PARITY COUNTER
011E     4300              B      NXTBIT
         010E
0122     08EE     TESTP    LHR    14,14             TEST FINAL PARITY COUNT
0124     4330              BZ     *+6               OK
         012A
0128     01F2              BALR   15,ERROR          BAD
012A     C500              CLHI   0,X'8D'           IS CHAR CR?
         008D
012E     4330              BE     PNCH              YES
         0144
0132     D201              STB    0,BUFR(1)         NO- STORE IN BUFFER
         0352
0136     0A1B              AHR    1,ONEREG          INCR CHAR COUNT
0138     C510              CLHI   1,72              72 CHAR?
```

```
        0048
013C    4380                BNL    PNCH              YES
        0144
0140    C200                LPSW   X'40'             NO- WAIT FOR NEXT CHAR
        0040
                *
                * PUNCH DATA ON TAPE
                *
0144    C8A0    PNCH        LHI    10,MESS1          OUTPUT MESSAGE
        0300
0148    C8C0                LHI    12,EMES1
        030B
014C    DED0                OC     13,WRITE          WRITE,BLOCK,ENABLE
        0344
0150    41F0                BAL    15,TYPEMG
        02CA
0154    0B1B                SHR    1,ONEREC          ADJUST CHAR COUNT
0156    9F00                AIR    0,0               CLEAR ATN
0158    C200                LPSW   LOC1              WAIT FOR OPERATOR
        015C
015C    8000    LOC1        DC     X'8000',A(NEXT)
        0160
                *
0160    C800    NEXT        LHI    0,PCHAR           NEW EXT INT PSW
        0182
0164    4000                STH    0,X'46'
        0046
0168    DED0                OC     13,WRITE          WRITE,BLOCK,ENABLE
        0344
016C    DAD0                WD     13,TAPEON         TURN ON PUNCH
        0346
0170    41F0                BAL    15,LEADR          PUNCH LEADER
        02E0
0174    0BAA                SHR    10,10             CLEAR CHAR COUNT
0176    C200                LPSW   LOC2              PERMIT EXT INT
        017A
017A    4000    LOC2        DC     X'4000',A(*+2)    AND WAIT FOR INTERRUPT
        017E
017E    4300                B      *
        017E
0182    9F0C    PCHAR       AIR    0,12              ACK AND CHECK DEV NO
0184    050D                CLHR   0,13
0186    4330                BE     *+6               OK
        018C
018A    01F2                BALR   15,ERROR
018C    DADA                WD     13,BUFR(10)       PUNCH NEXT CHAR
        0352
0190    9DD0                SSR    13,0              CHECK STATUS
0192    C500                CLHI   0,X'18'
        0018
0196    4330                BE     *+6
        019C
019A    01F2                BALR   15,ERROR          BAD
019C    05A1                CLHR   10,1              PUNCHING DONE?
```

```
  019E   4380            BNL    PUNOFF          YES
         01A8
  01A2   0AAB            AHR    10,ONEREG       NO
  01A4   C200            LPSW   X'40'           GO WAIT FOR NEXT CHAR
         0040
  01A8   41F0    PUNOFF  BAL    15,LEADR        PUNCH TRAILER
         02E0
  01AC   DAD0            WD     13,TAPEOF       TURN OFF PUNCH
         0347
                 *
                 * READ THE PUNCHED TAPE
                 *
  01B0   C8A0            LHI    10,MESS2        OUTPUT MESSAGE
         030C
  01B4   C8C0            LHI    12,EMES2
         0317
  01B8   41F0            BAL    15,TYPFMG
         02CA
  01BC   9F00            AIR    0,0             CLEAR ATN
  01BE   C200            LPSW   LOC3            WAIT FOR OPERATOR
         01C2
  01C2   8000    LOC3    DC     X'8000',A(*+2)
         01C6
  01C6   0BAA    RDTAPE  SHR    10,10           CLEAR CHAR COUNT
  01C8   40A0            STH    10,ECFLAG       AND ERROR FLAG
         034E
  01CC   DED0            OC     13,WRITE        WRITE,BLOCK,ENABLE
         0344
  01D0   DAD0            WD     13,XON          TURN ON READER
         0348
  01D4   DED0            OC     13,RTAPE        READ,BLOCK,ENABLE
         0345
  01D8   C8E0            LHI    14,TEST         SET LEADING ZERO SWITCH
         020C
  01DC   40E0            STH    14,THRU+2
         01EC
  01E0   9DDC    RETRN1  SSR    13,12           WAIT FOR BUSY =0
F 01E2   0200            NOPR   0,0
  01E4   4290            BTC    9,RETRN1
         01E0
  01E8   9BD0            RDR    13,0            AND READ 1 CHAR
  01EA   4300    THRU    B      TEST            LEADING ZERO SWITCH
         020C
  01EE   05A1            CLHR   10,1            TAPE READING DONE?
  01F0   4380            BNL    FINT            YES
         022A
  01F4   D3EA            LB     14,BUFR(10)     NO
         0352
  01F8   050E            CLHR   0,14            TEST FOR CORRECT CHAR
  01FA   4330            BE     *+8
         0202
  01FE   41F0            BAL    15,ERRFLG
         02FA
  0202   D20A            STB    0,BUFR(10)
```

```
         0352
0206     0AAB                 AHR     10,ONEREG          INCR CHAR COUNT
0208     4300                 B       RTCHAR
         021E
                    *
020C     0800       TEST      LHR     0,0                TEST FOR FIRST
020E     4330                 BZ      RTCHAR             NON-ZERO CHAR
         021E
0212     C8E0                 LHI     14,THRU+4          FOUND- CHANGE
         01EE
0216     40E0                 STH     14,THRU+2          LEADING ZERO SWITCH
         01EC
021A     4300                 B       THRU+4
         01EE
                    *
021E     9DDC       RTCHAR    SSR     13,12              BUSY SHOULD BE 1
0220     C5C0                 CLHI    12,X'18'
         0018
0224     4330                 BE      RETRN1             IT IS
         01E0
0228     01F2                 BALR    15,ERROR
                    *
022A     DED0       FINT      OC      13,WRITE           WRITE,BLOCK,ENABLE
         0344
022E     9DD0       SENS1     SSR     13,0               WAIT FOR BUSY=0
0230     4290                 BTC     9,SENS1
         022E
0234     DAD0                 WD      13,XOFF            TURN OFF READER
         0349
                    *
                    * PRINT DATA READ FROM TAPE
                    *
0238     C8A0                 LHI     10,MESS1           OUTPUT CR, LF
         0300
023C     C8C0                 LHI     12,MMESS1
         0301
0240     41F0                 BAL     15,TYPEMG
         02CA
0244     0BAA                 SHR     10,10
0246     DADA       TYCHAR    WD      13,BUFR(10)        OUTPUT 1 CHAR FROM BUFFER
         0352
024A     9DDC       NEXT2     SSR     13,12              WAIT FOR BUSY= 0
024C     4290                 BTC     9,NEXT2
         024A
0250     0200                 NOPR
0252     4300                 B       *+6
         0258
0256     01F2                 BALR    15,ERROR           ERROR
0258     C8AA                 LHI     10,1(10)           INCR BYTE COUNT
         0001
025C     05A1                 CLHR    10,1               AND TEST FOR DONE
025E     4280                 BL      TYCHAR             NOT DONE
         0246
0262     4800                 LH      0,ECFLAG           TEST FOR READ ERROR
```

```
       034E
0266   4330                BZ     BREAK              NO
       0278
026A   C8A0                LHI    10,TRMSG           YES
       0336
026E   C8C0                LHI    12,ETRMSG
       0341
0272   41F0                BAL    15,TYPEMG
       02CA
0276   01F2                BALR   15,ERROR
                    *
                    * TEST OPERATION OF BREAK KEY
                    *
0278   C8A0         BREAK  LHI    10,MESS3           OUTPUT MESSAGE
       0318
027C   C8C0                LHI    12,EMES3
       0327
0280   41F0                BAL    15,TYPEMG
       02CA
0284   9DD0         SENS2  SSR    13,0               WAIT FOR BREAK BIT=1
0286   C700                XHI    0,X'34'
       0034
028A   4230                BNZ    SENS2
       0284
028E   9DD0         SENS3  SSR    13,0               WAIT FOR BREAK BIT=0
0290   C700                XHI    0,X'34'
       0034
0294   4330                BZ     SENS3
       028E
0298   C8A0                LHI    10,MESS4           OUTPUT MESSAGE
       0328
029C   C8C0                LHI    12,EMES4
       032F
02A0   41F0                BAL    15,TYPEMG
       02CA
                    *
                    * END OF TEST
                    *
02A4   C8A0                LHI    10,MESS5
       0330
02A8   C8C0                LHI    12,EMES5
       0335
02AC   41F0                BAL    15,TYPEMG
       02CA
02B0   DED0                OC     13,RTAPE           CHANGE MODE TO READ
       0345
02B4   9F00                AIR    0,0                RESET ATN
02B6   4300                B      START              GO WAIT FOR OPERATOR
       0080
                    *
                    *
                    *
02BA   9F44         ERRORA AIR    4,4                CLEAR ATN
02BC   DAB0                WD     ONEREG,ERDSY       DISPLAY ONES
```

```
        034A
02C0    C200               LPSW   HALT                AND STOP
        02C4
02C4    8000    HALT       DC     X'8000',A(NEXT5)
        02C8
02C8    030F    NEXT5      BR     15                  CONTINUE
                *
02CA    9DD0    TYPEMG     SSR    13,0                WAIT FOR BUSY= 0 AND
02CC    4290               BTC    9,TYPEMG            OUTPUT 1 CHAR
        02CA
02D0    DADA               WD     13,0(10)            FROM R10
        0000
02D4    C1A0               BXLE   10,TYPEMG           LOOP
        02CA
02D8    9DD0    ENDMG      SSR    13,0                WAIT FOR LAST CHAR DONE
02DA    4290               BTC    9,ENDMG
        02D8
02DE    030F               BR     15                  RETURN
                *
                * PUNCH LEADER, TRAILER TAPE
                *
02E0    0BAA    LEADR      SHR    10,10
02E2    C8C0               LHI    12,128              ALLOWS FOR 128 BLANKS
        0080
02E6    9DD0    LEADR1     SSR    13,0
02E8    4290               BTC    9,LEADR1
        02E6
02EC    C0A0               BXH    10,OVER
        02F8
02F0    DAD0               WD     13,ZERO
        034C
02F4    4300               B      LEADR1
        02E6
02F8    030F    OVER       BR     15
                *
02FA    40E0    ERRFLG     STH    14,ECFLAG
        034E
02FE    030F               BR     15
                *
                *
                *
                *
0300    8D8A    MESS1      DC     X'8D8A'
0301            MMESS1     EQU    *-1
0302    5055               DC     C'PUNCH             TAPE'
        4E43
        4820
        5441
        5045
030B            EMES1      EQU    *-1
030C    8D8A    MESS2      DC     X'8D8A'
030E    5245               DC     C'READ              TAPE'
        4144
        2054
```

```
        4150
        4520
0317                EMES2   EQU     *-1
0318    8D8A        MESS3   DC      X'8D8A'
031A    8D8A                DC      X'8D8A'
031C    4445                DC      C'DEPRESS          BRK'
        5052
        4553
        5320
        4252
        4B20
0327                EMES3   EQU     *-1
0328    8D8A        MESS4   DC      X'8D8A'
032A    4252                DC      C'BRK              OK'
        4B20
        4F4B
032F                EMES4   EQU     *-1
0330    8D8A        MESS5   DC      X'8D8A'
0332    454E                DC      C'END'
        4420
0335                EMES5   EQU     *-1
0336    8D8A        TRMSG   DC      X'8D8A'
0338    5441                DC      C'TAPE             ERROR'
        5045
        2045
        5252
        4F52
0341                ETRMSG  EQU     *-1
0342    9864        WRTBLK  DC      X'9864'
0343                RECV    EQU     *-1
0344    5854        WRITE   DC      X'5854'
0345                RTAPE   EQU     *-1
0346    1214        TAPEON  DC      X'1214'
0347                TAPEOF  EQU     *-1
0348    1113        XON     DC      X'1113'
0349                XOFF    EQU     *-1
034A    FFFF        ERDSY   DC      X'FFFF'
034C    0000        ZERO    DC      0
034E                ECFLAG  DS      2
0350                STATUS  DS      2
0352                BUFR    DS      72
039A                        END

BREAK     0278
BUFR      0352
CLEAR     00DC
ECFLAG    034E
EMES1     030B
EMES2     0317
EMES3     0327
EMES4     032F
EMES5     0335
ENDMG     02D8
ERDSY     034A
```

ERRFLG	02FA
ERROR	0002
ERRORA	02BA
ETRMSG	0341
FINT	022A
HALT	02C4
INIT	00BA
LEADR	02E0
LEADR1	02E6
LOC1	015C
LOC2	017A
LOC3	01C2
MESS1	0300
MESS2	030C
MESS3	0318
MESS4	0328
MESS5	0330
MMESS1	0301
NEXT	0160
NEXT2	024A
NEXT5	02C8
NOINT	00BE
NXTBIT	010E
ONEREG	000B
OVER	02F8
PCHAR	0182
PNCH	0144
PUNOFF	01A8
RDTAPE	01C6
RECV	0343
RETRN1	01E0
RTAPE	0345
RTCHAR	021E
SENS1	022E
SENS2	0284
SENS3	028E
START	0080
STATUS	0350
STORE	00F2
TAPEOF	0347
TAPEON	0346
TEST	020C
TESTP	0122
THRU	01EA
TRMSG	0336
TYCHAR	0246
TYPEMG	02CA
WRITE	0344
WRTBLK	0342
XOFF	0349
XON	0348
ZERO	034C

```
                                 OPT    PASS2,PRINT,PUNCH,STOP,LAB=KSRTTY
                         *
                         * KSR TELETYPEWRITER TEST PROGRAM
                         *
                         * 06-083R00       AUGUST, 1970
                         *
                         * COPPER TELETYPEWRITER INTERFACE CONTROLLER
                         *
0080                             ORG    X'80'
000B                     ONEREG  EQU    11
0002                     ERROR   EQU    2
0080   C200              START   LPSW   *+4                  HALT UPON ENTERING PROGRAM
       0084
0084   8000                      DC     X'8000',A(*+2)
       0088
0088   C8D0                      LHI    13,2                 LOAD TTY DEVICE NUMBER
       0002
008C   0B11                      SHR    1,1                  CLEAR CHAR COUNT
008E   4010                      STH    1,X'44'              NEW EXT INT PSW
       0044
0092   C820                      LHI    2,ERRORA             ADDRESS OF ERROR ROUTINE
       01B0
0096   C8B0                      LHI    11,1                 R11= CONSTANT 1
       0001
009A   9DD0                      SSR    13,0                 TEST IF TTY AVAILABLE
009C   C400                      NHI    0,7                  CORRECTLY INITIALIZED
       0007
00A0   4330                      BZ     *+6                  ONLY BUSY BIT SET
       00A6
00A4   01F2                      BALR   15,ERROR
00A6   C800                      LHI    0,ERRORA             NEW EXT INT PSW
       01B0
00AA   4000                      STH    0,X'46'
       0046
00AE   C8F0                      LHI    15,NOINT             NO INT ERR ADDRESS
       00BE
00B2   DED0                      OC     13,WRTBLK            WRITE,BLOCK,DISABLE
       01F6
00B6   C200                      LPSW   INIT                 NO INT SHOULD BE GENERATED
       00BA
00BA   4000              INIT    DC     X'4000',A(*+2)
       00BE
                         *
00BE   C8A0              NOINT   LHI    10,MESS1             OUTPUT CR, LF
       01D6
00C2   C8C0                      LHI    12,MMESS1
       01D7
00C6   41F0                      BAL    15,TYPEMG
       01C0
00CA   C800                      LHI    0,CLEAR              NEW EXT INT PSW
       00D8
00CE   4000                      STH    0,X'46'
       0046
00D2   DED0                      OC     13,RECV              ENABLE,UNBLOCK,READ
```

```
        01F7
00D6    01F2              BALR   15,ERROR            NO INT- ERROR
00D8    9F00    CLEAR     AIR    0,0                 CLEAR ATN
00DA    C800              LHI    0,STORE             NEW EXT INT PSW
        00EE
00DE    4000              STH    0,X'46'
        0046
00E2    C200              LPSW   *+4                 WAIT FOR INTERRUPTS
        00E6
00E6    4000              DC     X'4000',A(*+2)      FROM OPERATOR TYPING
        00EA
00EA    4300              B      *
        00EA
00EE    9F0C    STORE     AIR    0,12                ACKNOWLEDGE, CHECK DEVICE
00F0    050D              CLHR   0,13
00F2    4330              BE     *+6                 OK
        00F8
00F6    01F2              BALR   15,ERROR            BAD
00F8    08CC              LHR    12,12               STATUS BYTE ZERO?
00FA    4330              BZ     *+6                 YES
        0100
00FE    01F2              BALR   15,ERROR            NO
0100    9BD0              RDR    13,0                READ THE CHARACTER
                *
                * TEST CHARACTER FOR EVEN PARITY
                *
0102    08C0              LHR    12,0                R12= CHARACTER
0104    0BEE              SHR    14,14               PARITY COUNTER
0106    C840              LHI    4,-9                LOOP COUNT
        FFF7
010A    0A4B    NXTBIT    AHR    4,ONEREG            DECR LOOP COUNT
010C    4330              BZ     TESTP               DONE WHEN ZERO
        011E
0110    CCC0              SRHL   12,1                TEST BIT FOR 1 OR 0
        0001
0114    4380              BFC    8,NXTBIT            BRANCH IF ONE
        010A
0118    07EB              XHR    14,ONEREG           INCR PARITY COUNTER
011A    4300              B      NXTBIT
        010A
011E    08EE    TESTP     LHR    14,14               TEST FINAL PARITY COUNT
0120    4330              BZ     *+6                 OK
        0126
0124    01F2              BALR   15,ERROR            BAD
0126    C500              CLHI   0,X'8D'             IS CHAR CR?
        008D
012A    4330              BE     PRINT               YES
        0140
012E    D201              STB    0,BUFR(1)           NO- STORE IN BUFFER
        0200
0132    0A1B              AHR    1,ONEREG            INCR CHAR COUNT
0134    C510              CLHI   1,72                72 CHAR?
        0048
0138    4380              BNL    PRINT               YES
```

```
        0140
013C    C200           LPSW    X'40'           NO- WAIT FOR NEXT CHAR
        0040
               *
               * PRINT CHAR IN BUFFER
               *
0140    DED0   PRINT   OC      13,WRITE        WRITE,BLOCK,ENABLE
        01F8
0144    C8A0           LHI     10,MESS1        OUTPUT CR, LF
        01D6
0148    C8C0           LHI     12,MMESS1
        01D7
014C    41F0           BAL     15,TYPEMG
        01C0
0150    0BAA           SHR     10,10
0152    DADA   TYCHAR  WD      13,BUFF(10)     OUTPUT 1 CHAR FROM BUFFER
        0200
0156    9DDC   NEXT2   SSR     13,12           WAIT FOR BUSY= 0
0158    4290           BTC     9,NEXT2
        0156
015C    08CC           LHR     12,12
015E    4330           BZ      *+6
        0164
0162    01F2           BALR    15,ERROR        ERROR
0164    C8AA           LHI     10,1(10)        INCR BYTE COUNT
        0001
0168    05A1           CLHR    10,1            AND TEST FOR DONE
016A    4280           BL      TYCHAR          NOT DONE
        0152
               *
               * TEST OPERATION OF BREAK KEY
               *
016E    C8A0   BREAK   LHI     10,MESS1        OUTPUT MESSAGE
        01D6
0172    C8C0           LHI     12,EMES3
        01E7
0176    41F0           BAL     15,TYPEMG
        01C0
017A    9DD0   SENS2   SSR     13,0            WAIT FOR BREAK BIT=1
017C    C700           XHI     0,X'24'
        0024
0180    4230           BNZ     SENS2
        017A
0184    9DD0   SENS3   SSR     13,0            WAIT FOR BREAK BIT=0
0186    C700           XHI     0,X'24'
        0024
018A    4330           BZ      SENS3
        0184
018E    C8A0           LHI     10,MESS4        OUTPUT MESSAGE
        01E8
0192    C8C0           LHI     12,EMES4
        01EF
0196    41F0           BAL     15,TYPEMG
        01C0
```

```
                  *
                  * END OF TEST
                  *
019A  C8A0               LHI   10,MESS5
      01F0
019E  C8C0               LHI   12,EMESS5
      01F5
01A2  41F0               BAL   15,TYPEMG
      01C0
01A6  DED0               OC    13,RTAPE          CHANGE MODE TO READ
      01F9
01AA  9F00               AIR   0,0               RESET ATN
01AC  4300               B     START             GO WAIT FOR OPERATOR
      0080
                  *
                  *
                  *
01B0  9F44        ERRORA AIR   4,4               CLEAR ATN
01B2  DAB0               WD    ONEREG,ERDSY      DISPLAY ONES
      01FA
01B6  C200               LPSW  HALT              AND STOP
      01BA
01BA  8000        HALT   DC    X'8000',A(NEXT5)
      01BE
01BE  030F        NEXT5  BR    15                CONTINUE
                  *
01C0  9DD0        TYPEMG SSR   13,0              WAIT FOR BUSY= 0 AND
01C2  4290               BTC   9,TYPEMG          OUTPUT 1 CHAR
      01C0
01C6  DADA               WD    13,0(10)          FROM R10
      0000
01CA  C1A0               BXLE  10,TYPEMG         LOOP
      01C0
01CE  9DD0        ENDMG  SSR   13,0              WAIT FOR LAST CHAR DONE
01D0  4290               BTC   9,ENDMG
      01CE
01D4  030F               BR    15                RETURN
                  *
                  *
                  *
01D6  8D8A        MESS1  DC    X'8D8A'
01D7              MMESS1 EQU   *-1
01D8  8D8A        MESS3  DC    X'8D8A'
01DA  8D8A               DC    X'8D8A'
01DC  4445               DC    C'DEPRESS         BRK'
      5052
      4553
      5320
      4252
      4B20
01E7              EMES3  EQU   *-1
01E8  8D8A        MESS4  DC    X'8D8A'
01EA  4252               DC    C'BRK             OK'
      4B20
```

```
        4F4B
01EF              EMES4    EQU    *-1
01F0    8D8A      MESS5    DC     X'8D8A'
01F2    454E               DC     C'END'
        4420
01F5              EMES5    EQU    *-1
01F6    9864      WRTBLK   DC     X'9864'
01F7              RECV     EQU    *-1
01F8    5854      WRITE    DC     X'5854'
01F9              RTAPE    EQU    *-1
01FA    FFFF      ERDSY    DC     X'FFFF'
01FC    0000      ZERO     DC     0
01FE              STATUS   DS     2
0200              BUFR     DS     72
0248                       END

BREAK     016E
BUFR      0200
CLEAR     00D8
EMES3     01E7
EMES4     01EF
EMES5     01F5
ENDMG     01CE
ERDSY     01FA
ERROR     0002
ERRORA    01B0
HALT      01BA
INIT      00BA
MESS1     01D6
MESS3     01D8
MESS4     01E8
MESS5     01F0
MMESS1    01D7
NEXT2     0156
NEXT5     01BE
NOINT     00BE
NXTBIT    010A
ONEREG    000B
PRINT     0140
RECV      01F7
RTAPE     01F9
SENS2     017A
SENS3     0184
START     0080
STATUS    01FE
STORE     00EE
TESTP     011E
TYCHAR    0152
TYPEMG    01C0
WRITE     01F8
WRTBLK    01F6
ZERO      01FC
```

```
                              OPT    PASS2,PRINT,PUNCH,STOP,LAB=KSRTRM
                      *
                      * KSR TERMINET TEST PROGRAM
                      *
                      * 96-083R00       AUGUST, 1970
                      *
                      * COPPER TELETYPEWRITER INTERFACE CONTROLLER
                      *
0080                          ORG    X'80'
000B                  ONEREG  EQU    11
0002                  ERROR   EQU    2
0080   C200           START   LPSW   *+4                HALT UPON ENTERING PROGRAM
       0084
0084   8000                   DC     X'8000',A(*+2)
       0088
0088   C8D0                   LHI    13,2               LOAD TTY DEVICE NUMBER
       0002
008C   0B11                   SHR    1,1                CLEAR CHAR COUNT
008E   4010                   STH    1,X'44'            NEW EXT INT PSW
       0044
0092   C820                   LHI    2,ERRORA           ADDRESS OF ERROR ROUTINE
       01B0
0096   C8B0                   LHI    11,1               R11= CONSTANT 1
       0001
009A   9DD0                   SSR    13,0               TEST IF TTY AVAILABLE
009C   C400                   NHI    0,X'E7'            CORRECTLY INITIALIZED
       00E7
00A0   4330                   BZ     *+6                ONLY BUSY BIT SET
       00A6
00A4   01F2                   BALR   15,ERROR
00A6   C800                   LHI    0,ERRORA           NEW EXT INT PSW
       01B0
00AA   4000                   STH    0,X'46'
       0046
00AE   C8F0                   LHI    15,NOINT           NO INT ERR ADDRESS
       00BE
00B2   DED0                   OC     13,WRTBLK          WRITE,BLOCK,DISABLE
       01F6
00B6   C200                   LPSW   INIT               NO INT SHOULD BE GENERATED
       00BA
00BA   4000           INIT    DC     X'4000',A(*+2)
       00BE
                      *
00BE   C8A0           NOINT   LHI    10,MESS1           OUTPUT CR, LF
       01D6
00C2   C8C0                   LHI    12,MMESS1
       01D7
00C6   41F0                   BAL    15,TYPEMG
       01C0
00CA   C800                   LHI    0,CLEAR            NEW EXT INT PSW
       00D8
00CE   4000                   STH    0,X'46'
       0046
00D2   DED0                   OC     13,RECV            ENABLE,UNBLOCK,READ
```

```
        01F7
00D6    01F2                BALR   15,ERROR            NO INT- ERROR
00D8    9F00     CLEAR      AIR    0,0                 CLEAR ATN
00DA    C800                LHI    0,STORE             NEW EXT INT PSW
        00EE
00DE    4000                STH    0,X'46'
        0046
00E2    C200                LPSW   *+4                 WAIT FOR INTERRUPTS
        00E6
00E6    4000                DC     X'4000',A(*+2)      FROM OPERATOR TYPING
        00EA
00EA    4300                B      *
        00EA
00EE    9F0C     STORE      AIR    0,12                ACKNOWLEDGE, CHECK DEVICE
00F0    050D                CLHR   0,13
00F2    4330                BZ     X'100'              OK
        0100
00F6    01F2                BALR   15,ERROR            BAD
00F8    08CC                LHR    12,12               STATUS BYTE ZERO?
00FA    4330                BZ     *+6                 YES
        0100
00FE    01F2                BALR   15,ERROR            NO
0100    9BD0                RDR    13,0                READ THE CHARACTER
                 *
                 * TEST CHARACTER FOR EVEN PARITY
                 *
0102    08C0                LHR    12,0                R12= CHARACTER
0104    0BEE                SHR    14,14               PARITY COUNTER
0106    C840                LHI    4,-9                LOOP COUNT
        FFF7
010A    0A4B     NXTBIT     AHR    4,ONEREG            DECR LOOP COUNT
010C    4330                BZ     TESTP               DONE WHEN ZERO
        011E
0110    CCC0                SRHL   12,1                TEST BIT FOR 1 OR 0
        0001
0114    4380                BFC    8,NXTBIT            BRANCH IF ONE
        010A
0118    07EB                XHR    14,ONEREG           INCR PARITY COUNTER
011A    4300                B      NXTBIT
        010A
011E    08EE     TESTP      LHR    14,14               TEST FINAL PARITY COUNT
0120    4330                BZ     *+6                 OK
        0126
0124    01F2                BALR   15,ERROR            BAD
0126    C500                CLHI   0,X'8D'             IS CHAR CR?
        008D
012A    4330                BE     PRINT               YES
        0140
012E    D201                STB    0,BUFR(1)           NO- STORE IN BUFFER
        0200
0132    0A1B                AHR    1,ONEREG            INCR CHAR COUNT
0134    C510                CLHI   1,72                72 CHAR?
        0048
0138    4380                BNL    PRINT               YES
```

```
         0140
 013C    C200                LPSW   X'40'               NO- WAIT FOR NEXT CHAR
         0040
                    *
                    * PRINT CHAR IN BUFFER
                    *
 0140    DED0       PRINT    OC     13,WRITE            WRITE,BLOCK,ENABLE
         01F8
 0144    C8A0                LHI    10,MESS1            OUTPUT CR, LF
         01D6
 0148    C8C0                LHI    12,MMESS1
         01D7
 014C    41F0                BAL    15,TYPEMG
         01C0
 0150    0BAA                SHR    10,10
 0152    DADA       TYCHAR   WD     13,BUFR(10)         OUTPUT 1 CHAR FROM BUFFER
         0200
 0156    9DDC       NEXT2    SSR    13,12               WAIT FOR BUSY= 0
 0158    4290                BTC    9,NEXT2
         0156
 015C    0200                NOPR
 015E    4300                B      *+6
         0164
 0162    01F2                BALR   15,ERROR            ERROR
 0164    C8AA                LHI    10,1(10)            INCR BYTE COUNT
         0001
 0168    05A1                CLHR   10,1                AND TEST FOR DONE
 016A    4280                BL     TYCHAR              NOT DONE
         0152
                    *
                    * TEST OPERATION OF BREAK KEY
                    *
 016E    C8A0       BREAK    LHI    10,MESS1            OUTPUT MESSAGE
         01D6
 0172    C8C0                LHI    12,EMES3
         01E7
 0176    41F0                BAL    15,TYPEMG
         01C0
 017A    9DD0       SENS2    SSR    13,0                WAIT FOR BREAK BIT=1
 017C    C700                XHI    0,X'34'
         0034
 0180    4230                BNZ    SENS2
         017A
 0184    9DD0       SENS3    SSR    13,0                WAIT FOR BREAK BIT=0
 0186    C700                XHI    0,X'34'
         0034
 018A    4330                BZ     SENS3
         0184
 018E    C8A0                LHI    10,MESS4            OUTPUT MESSAGE
         01E8
 0192    C8C0                LHI    12,EMES4
         01EF
 0196    41F0                BAL    15,TYPEMG
         01C0
```

```
                        *
                        * END OF TEST
                        *
019A  C8A0                      LHI   10,MESS5
      01F0
019E  C8C0                      LHI   12,EMES5
      01F5
01A2  41F0                      BAL   15,TYPEMG
      01C0
01A6  DED0                      OC    13,RTAPE           CHANGE MODE TO READ
      01F9
01AA  9F00                      AIR   0,0                RESET ATN
01AC  4300                      B     START              GO WAIT FOR OPERATOR
      0080
                        *
                        *
                        *
01B0  9F44              ERRORA  AIR   4,4                CLEAR ATN
01B2  DAB0                      WD    ONEREG,ERDSY       DISPLAY ONES
      01FA
01B6  C200                      LPSW  HALT               AND STOP
      01BA
01BA  8000              HALT    DC    X'8000',A(NEXT5)
      01BE
01BE  030F              NEXT5   BR    15                 CONTINUE
                        *
01C0  9DD0              TYPEMG  SSR   13,0               WAIT FOR BUSY= 0 AND
01C2  4290                      BTC   9,TYPEMG           OUTPUT 1 CHAR
      01C0
01C6  DADA                      WD    13,0(10)           FROM R10
      0000
01CA  C1A0                      BXLE  10,TYPEMG          LOOP
      01C0
01CE  9DD0              ENDMG   SSR   13,0               WAIT FOR LAST CHAR DONE
01D0  4290                      BTC   9,ENDMG
      01CE
01D4  030F                      BR    15                 RETURN
                        *
                        *
                        *
01D6  8D8A              MESS1   DC    X'8D8A'
01D7                    MMESS1  EQU   *-1
01D8  8D8A              MESS3   DC    X'8D8A'
01DA  8D8A                      DC    X'8D8A'
01DC  4445                      DC    C'DEPRESS          BRK'
      5052
      4553
      5320
      4252
      4B20
01E7                    EMES3   EQU   *-1
01E8  8D8A              MESS4   DC    X'8D8A'
01EA  4252                      DC    C'BRK              OK'
      4B20
```

```
          4F4B
   01EF              EMES4    EQU     *-1
   01F0   8D8A       MESS5    DC      X'8D8A'
0         454E                DCC'END'
          4420
   01F5              EMES5    EQU     *-1
   01F6   9864       WRTBLK   DC      X'9864'
   01F7              RECV     EQU     *-1
   01F8   5854       WRITE    DC      X'5854'
   01F9              RTAPE    EQU     *-1
   01FA   FFFF       ERDSY    DC      X'FFFF'
   01FC   0000       ZERO     DC      0
   01FE              STATUS   DS      2
   0200              BUFR     DS      72
   0248                       END

   BREAK    016E
   BUFR     0200
   CLEAR    00D8
   EMES3    01E7
   EMES4    01EF
   EMES5    01F5
   ENDMG    01CE
   ERDSY    01FA
   ERROR    0002
   ERRORA   01B0
   HALT     01BA
   INIT     00BA
   MESS1    01D6
   MESS3    01D8
   MESS4    01E8
   MESS5    01F0
   MMESS1   01D7
   NEXT2    0156
   NEXT5    01BE
   NOINT    00BE
   NXTBIT   010A
   ONEREG   000B
   PRINT    0140
   RECV     01F7
   RTAPE    01F9
   SENS2    017A
   SENS3    0184
   START    0080
   STATUS   01FE
   STORE    00EE
   TESTP    011E
   TYCHAR   0152
   TYPEMG   01C0
   WRITE    01F8
   WRTBLK   01F6
   ZERO     01FC
```

OPERATING INSTRUCTIONS FOR THE HIGH SPEED PAPER TAPE READER TEST PROGRAM

Publication Number 06-016A12

1. FUNCTION

The test program checks the validity and accuracy of the High Speed Paper Tape Reader.

2. TAPE FORMAT

The object tape (Program Number 06-016 R01M09) is in an absolute format. For easier handling, the specially prepared test tape is included on the same tape as the object program. They are separated by two fanfolds of leader.

3. LOADING PROCEDURE

The object tape may be loaded using the Absolute, Relocatable, or General Loaders. Control is automatically transferred to the starting address (X'80') of the test upon completion of the load.

4. PROGRAM DESCRIPTION

The program reads the specially prepared test tape and makes a comparison of what was read from the tape to known values. The leading or trailing zeros are ignored. If an error is made in reading the tape, the program halts with the erroneous character read just past the sense lights. Register 9 contains the character that was read and Register 8 contains what should have been read. To continue reading test tape,depress EXECUTE. The tape motion will halt just prior to reaching the end of the tape. This is because an erroneous character was placed there to cause this stop action.

Appendix 1 is a listing of the program.

```
                  *             OPT      PASS2, PRINT, PUNCH
                  *
                  * HSPTR TEST PROGRAM      06-016R01          12-26-68
                  *
0080                            ORG      X'80'
                  *
0080     0B66     RSTRT         SHR      6,6              INITIALIZE
0082     C830                   LHI      3,1              BXLE INCREMENT
         0001
0086     C840                   LHI      4,7              BXLE LIMIT
         0007
008A     C8A0                   LHI      10,3             HSPTR DEVICE NUMBER
         0003
008E     DEA0                   OC       10, MOVE
         00D8
0092     0B22     START1        SHR      2,2              BXLE INDEX
0094     0A63     START         AHR      6,3              INCREM SHIFT REGISTER
0096     9DAE     SENSE         SSR      10,14
0098     08EE                   LHR      14,14
009A     4230                   BTC      3, SENSE
         0096
009E     9BA9                   RDR      10,9             DATA IN R9
00A0     0899                   LHR      9,9              IGNORE LEADING ZEROES
00A2     4330                   BZ       SENSE
         0096
00A6     9386                   LBR      8,6              ONLY RIGHT MOST BYTE
00A8     0589                   CLHR     8,9              COMPARE
00AA     4230                   BTC      3, ERROR         ERROR IF NOT EQUAL
         00C4
00AE     CD60     CONTIN        SLHL     6,1              SHIFT PATTERN LEFT ONE
         0001
00B2     C120                   BXLE     2, SENSE         REPEAT SEVEN TIMES
         0096
00B6     CE60                   SRHA     6,7              RESTART PATTERN +1
         0007
00BA     4220                   BTC      2, START1
         0092
00BE     0B66                   SHR      6,6
00C0     4300                   BFC      0, START1        FINISH ONE CYCLE
         0092
00C4     DEA0     ERROR         OC       10, STOP         STOP READER
         00D9
00C8     C200                   LPSW     WAIT             HALT
         00CC
00CC     8000     WAIT          DC       X'8000', OUTCOM
         00D0
00D0     DEA0     OUTCOM        OC       10, MOVE         RE-ISSUE OUTPUT COMMAND
         00D8
00D4     4300                   B        CONTIN           CONTINUE TEST
         00AE
00D8     99A9     MOVE          DC       X'99A9'
00D9              STOP          EQU      MOVE+1
00DA                            END      RSTRT
```

CONTIN	00AE
ERROR	00C4
MOVE	00D8
OUTCOM	00D0
RSTRT	0080
SENSE	0096
START	0094
START1	0092
STOP	00D9
WAIT	00CC

HIGH SPEED PAPER TAPE PUNCH TEST PROGRAM

Publication Number 06-037A12

1. PURPOSE

The Punch Test Program (Program Number 06-037) is designed to test the operation of the High Speed Paper Tape Punch.

2. TAPE FORMAT

The test program is in an absolute format with the origin designated as X'80'. The program occupies 580_{10} or $1EA_{16}$ bytes of memory. A listing of the program is provided in Appendix 1.

3. LOADING PROCEDURE

The object tape is loaded into memory using the Absolute Loader (Tape Number 06-023M10). If further information is needed, refer to Loader Descriptions, Publication Number 06-025A12.

4. PROGRAM OPERATION

The program is designed to perform four separate tests. It has four different starting locations which determine which test is to be performed. Table 1 is a description of each test and its corresponding starting address.

TABLE 1.

Test Number	Starting Location	Description
#1	X'80'	Reads and punches whatever data is expressed by the rightmost eight Data/Address switches (8-15) on the Display Panel. The test runs until manually stopped by the operator. As long as the test is running, the switches may be changed to form any combination.
#2	X'84'	Same as Test 1 except Interrupts are used instead of Sense Status to control data outputs to the punch.
#3	X'88'	Punches six distinct patterns allowing a visual confirmation as to the correctness of the punch. Each pattern is repeated 20 times; however, if SW15 is depressed, the current pattern is repeated until the switch is released. After all six patterns are completed, the program halts. To repeat Test 3, depress the EXECUTE button. Figure 1 is an example of the six patterns.

TABLE 1
(Continued)

Test Number	Starting Location	Description
#4	X'8C'	Punches all sixty-four ASCII characters followed by a Carriage Return, Line Feed and ten blanks. This test also repeats 20 times and will repeat indefinitely as long as SW15 is depressed. Upon completion of this test, the program halts, but the test may be repeated by pushing the EXECUTE button.

The power to run the High Speed Punch is turned on by program control. The only obligation of the user is to be sure power has been supplied to the punch motor.

The device number and output commands for the punch are taken from X'7A', the Binary Output Device (BOUTDV) specification in the 50 Sequence. The GE-PAC 30 high-speed punch is normally assigned Device Number 3. To use the high-speed punch, the location at X'7A' should be set with the following:

BOUTDV spec	Meaning
X'0392'	Device Number 3, disable, run, and write. This command is appropriate to Test 1, 3, and 4, in which interrupts are not required.
X'0352'	Device Number 3, enable, run, and write. This command is required for Test 2 which uses interrupts.

This test can be used with a teletypewriter to exercise the teletypewriter punch unit. In this case Test 3 should be avoided since the data contains control characters, such as form feeds and vertical tabs, which wreak havoc with the teletypewriter. Teletypewriters are conventionally assigned Device Number 2. The device table codes required for the teletypewriter are:

BOUTDV spec	Meaning
X'0298'	Device Number 2, disable, block, and write.
X'0258'	Device Number 2, enable, block and write.

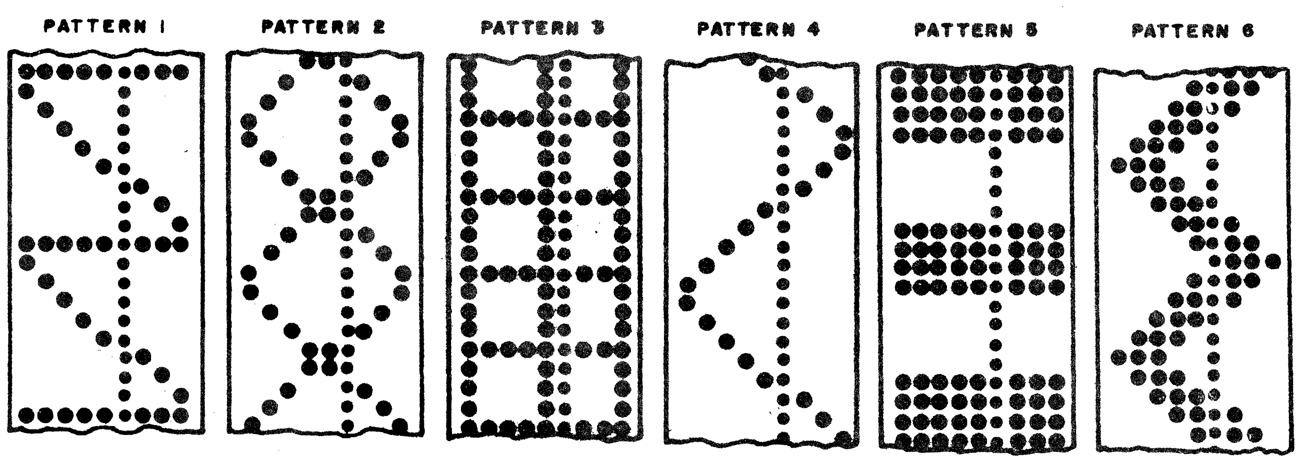


Figure 1. Tape Patterns

Appendix 1.
Program Listing

```
                         OPT    PASS2,PRINT,PUNCH,STOP
                  *
                  * HIGH SPEED PUNCH TEST PROGRAM
                  * BINARY OUTPUT DEV IN X'7A' DEFINES DEVICE NUMBER
                  * 06-037
                  *
0000              SDEV     EQU    0
0001              SCNT     EQU    1
0002              DEV      EQU    2
0003              ONE      EQU    3
0004              STAT     EQU    4
0005              DATA     EQU    5
0006              NREP     EQU    6
0007              R7       EQU    7
0008              R8       EQU    8
0009              R9       EQU    9
000A              A        EQU    10
000B              B        EQU    11
000C              C        EQU    12
000D              D        EQU    13
000E              E        EQU    14
000F              F        EQU    15
007A              BOUTDV   EQU    X'7A'
0080                       ORG    X'80'
0080  4300                 B      TEST1
      00B0
0084  4300                 B      TEST2
      00CA
0088  4300                 B      TEST3
      0110
008C  4300                 B      TEST4
      0204
                  *
007B              DSABLE   EQU    X'7B'
007B              ENABLE   EQU    X'7B'
0090  A200        STOP     DC     X'A200'
                  *
0092  9D24        PUNCH    SSR    DEV,STAT          PUNCH DATA
0094  42F0                 BTC    F,PUNCH
      0092
0098  4200                 NOP
      0000
009C  9A25                 WDR    DEV,DATA
009E  030F                 BR     F
                  *
00A0                       DO     4
00A0  4200                 NOP
      0000
00A4  4200
      0000
00A8  4200
      0000
00AC  4200
      0000
                  *
                  *
                  * TEST1 READS DATA FROM DISPLAY PANEL SWITCHES
                  * AND PUNCHES THE DATA, USING SENSE STATUS LOOP
                  *
00B0  D320        TEST1    LB     DEV,BOUTDV
      007A
00B4  DE20                 OC     DEV,DSABLE
```

```
        007B
 00B8   C830            LHI    ONE,1
        0001
 00BC   9D24    T1      SSR    DEV,STAT
 00BE   42F0            BTC    F,T1
        00BC
 00C2   9B35            RDR    ONE,DATA
 00C4   9A25            WDR    DEV,DATA
 00C6   4300            B      T1
        00BC
                *
                * TEST2 READS DATA FROM DISPLAY PANEL SWITCHES
                * AND PUNCHES THE DATA, USING INTERRUPTS
                *
 00CA   D320    TEST2   LB     DEV,BOUTDV
        007A
 00CE   DE20            OC     DEV,ENABLE
        007B
 00D2   C830            LHI    ONE,1
        0001
 00D6   0B44            SHR    STAT,STAT
 00D8   4040            STH    STAT,X'44'          SET NEW EXT PSW
        0044
 00DC   C850            LHI    DATA,T2I
        00F4
 00E0   4050            STH    DATA,X'46'
        0046
 00E4   0B11            SHR    SCNT,SCNT
 00E6   0B00            SHR    SDEV,SDEV
 00E8   9A34    T2      WDR    ONE,STAT            DISPLAY BAD STAT
 00EA   9A25    T2E     WDR    DEV,DATA            AND PUNCH DATA
 00EC   C200            LPSW   *+4                 ENABLE EXT INT
        00F0
 00F0   C000            DC     X'C000',T2
        00E8
                *
 00F4   9F54    T2I     AIR    DATA,STAT           ACK INT
 00F6   0552            CLHR   DATA,DEV            TEST IF PUNCH
 00F8   4230            BNE    T2P
        0108
 00FC   0844            LHR    STAT,STAT           TEST IF STATUS OK
 00FE   4230            BNZ    T2
        00E8
 0102   9B35            RDR    ONE,DATA            IF SO, READ SWITCHES
 0104   4300            B      T2E
        00EA
                *
 0108   0805    T2P     LHR    SDEV,DATA           REMEMBER SPURIOUS DEV
 010A   0A13            AHR    SCNT,ONE            BUMP SPURIOUS COUNT
 010C   4300            B      T2
        00E8
                *
                * TEST3 PUNCHES 6 VISUAL PATTERNS
                * FOR VISUAL VERIFICATION.
                * EACH PATTERN IS PUNCHED 20 TIMES.
                * THE PROGRAM PUNCHES THE CURRENT PATTERN
                *CONTINUOUSLY IF SWITCH 15 IS DEPRESSED.
                *
 0110   C8A0    TEST3   LHI    A,TABLE
        016E
 0114   48BA    T3A     LH     B,0(A)              SET PATTERN POINTER
        0000
```

```
0118  4330          BZ     T3END
      0162
011C  D320          LB     DEV,BOUTDV
      007A
0120  DE20          OC     DEV,DSABLE
      007B
0124  C860          LHI    NREP,20
      0014
0128  C830          LHI    ONE,1
      0001
012C  48CB  T3LOOP  LH     C,0(B)          SET BYTE COUNT
      0000
0130  C8DB          LHI    D,2(B)
      0002
0134  9D24  T3TEST  SSR    DEV,STAT
0136  42F0          BTC    F,T3TEST
      0134
013A  DA2D          WD     DEV,0(D)
      0000
013E  0AD3          AHR    D,ONE
0140  0BC3          SHR    C,ONE
0142  4230          BNZ    T3TEST
      0134
0146  0866          LHR    NREP,NREP
0148  4330          BZ     T3R
      0152
014C  0B63          SHR    NREP,ONE
014E  4230          BNZ    T3LOOP
      012C
0152  9B35  T3R     RDR    ONE,DATA        READ SWITCH
0154  0453          NHR    DATA,ONE
0156  4230          BNZ    T3LOOP
      012C
015A  CAA0          AHI    A,2
      0002
015E  4300          B      T3A
      0114
            *
0162  DE20  T3END   OC     DEV,STOP        POWER DOWN
      0090
0166  C200          LPSW   *+4             WAIT
      016A
016A  8000          DC     X'8000',TEST3   REPEAT ON EXECUTE
      0110
            *
016E  018C  TABLE   DC     PAT1,PAT2,PAT3
      01A0
      01B2
0174  01C4          DC     PAT4,PAT5,PAT6,0
      01DE
      01F0
      0000
017C                DS     16
018C  0009  PAT1    DC     9               TRIANGLES
018E  FF01          DC     X'FF01',X'0204'
      0204
0192  0810          DC     X'0810',X'2040',X'8000'
      2040
      8000
0198                DS     8
01A0  0008  PAT2    DC     8               DIAMONDS
01A2  1824          DC     X'1824,X'4281'
```

A1-3.

```
      4281
01A6  8142                DC    X'8142',X'2418'
      2418
01AA                      DS    8
01B2  0008        PAT3    DC    8                   SQUARES
01B4  FF89                DC    X'FF89',X'8989'
      8989
01B8  FF89                DC    X'FF89',X'8989'
      8989
01BC                      DS    8
01C4  0010        PAT4    DC    16                  SAWTEETH
01C6  0102                DC    X'0102',X'0408'
      0408
01CA  1020                DC    X'1020',X'4080'
      4080
01CE  8040                DC    X'8040',X'2010'
      2010
01D2  0804                DC    X'0804',X'0201'
      0201
01D6                      DS    8
01DE  0008        PAT5    DC    8                   BLOCKS
01E0  FFFF                DC    X'FFFF',X'FFFF'
      FFFF
01E4  0000                DC    0,0
      0000
01E8                      DS    8
01F0  000A        PAT6    DC    10                  WORMS
01F2  070E                DC    X'070E',X'1C38'
      1C38
01F6  70E0                DC    X'70E0',X'7038',X'1C0E'
      7038
      1C0E
01FC                      DS    8
                  *
                  * TEST4 PUNCHES THE ASCII CHAR SET
                  * FOLLOWED BY CR/LF AND BLANK TAPE.
                  * THE SEQUENCE IS PUNCHED 20 TIMES
                  * AND THE PROGRAM HALTS.
                  * THE PROGRAM PUNCHES CONTINUOUSLY
                  * IF SWITCH 15 IS DEPRESSED.
                  * THE RESULTING TAPE CAN BE LISTED ON A TTY.
                  *
0204  D320        TEST4   LB    DEV,BOUTDV
      007A
0208  DE20                OC    DEV,DSABLE
      007B
020C  C830                LHI   ONE,1
      0001
0210  C860                LHI   NREP,20
      0014
0214  C850        T4LOOP  LHI   DATA,X'A0'
      00A0
0218  41F0        T4TEST  BAL   F,PUNCH
      0092
021C  0A53                AHR   DATA,ONE
021E  C550                CLHI  DATA,X'E0'
      00E0
0222  4280                BL    T4TEST
      0218
0226  C850                LHI   DATA,X'8D'
      008D
022A  41F0                BAL   F,PUNCH
```

A1-4.

```
      0092
022E  C850            LHI   DATA,X'8A'
      008A
0232  41F0            BAL   F,PUNCH
      0092
0236  0855            SHR   DATA,DATA
0238  C8C0            LHI   C,10
      000A
023C  41F0   T4LEAD   BAL   F,PUNCH
      0092
0240  08C3            SHR   C,ONE
0242  4230            BNZ   T4LEAD
      023C
             *
0246  0866            LHR   NREP,NREP
0248  4330            BZ    T4R
      0252
024C  0863            SHR   NREP,ONE
024E  4230            BNZ   T4LOOP
      0214
0252  9B35   T4R      RDR   ONE,DATA
0254  0453            NHR   DATA,ONE
0256  4230            BNZ   T4LOOP
      0214
             *
025A  DE20            OC    DEV,STOP          POWER DOWN
      0090
025E  C200            LPSW  *+4               WAIT
      0262
0262  8000            DC    X'8000',TEST4     REPEAT ON EXECUTE
      0204
             *
0266                  END

A        000A
B        000B
BOUTDV   007A
C        000C
D        000D
DATA     0005
DEV      0002
DSABLE   007B
E        000E
ENABLE   007B
F        000F
NREP     0006
ONE      0003
PAT1     018C
PAT2     01A0
PAT3     01B2
PAT4     01C4
PAT5     01DE
PAT6     01F0
PUNCH    0092
R7       0007
R8       0008
R9       0009
SCNT     0001
SDEV     0000
STAT     0004
STOP     0090
T1       00BC
```

T2	00EB
T2E	00EA
T2I	00FA
T2P	010B
~~T3A~~	~~0114~~
T3END	0162
T3LOOP	012C
T3R	0152
T3TEST	0134
T4LEAD	023C
~~T4LOOP~~	~~0214~~
T4R	0252
T4TEST	0218
TABLE	016E
TEST1	00B0
TEST2	00CA
~~TEST3~~	~~0110~~
TEST4	0204

CARD READER TEST

Publication Number 06-038A12

1. INTRODUCTION

The purpose of this test (Program Number 06-038) is to inform the user or serviceman when the Soroban (SCCR) Card Reader is not operating correctly. This is determined by reading a card from the test deck and then comparing what was read to a predetermined master image which is stored in an area in memory referred to as the TABLE. The TABLE consists of 160_{10} bytes of Hollerith coded data, or 2 bytes per each column. The contents of the card read is stored in an area in memory called the Buffer. So it is actually the contents of the Buffer that is compared to the image contained in the TABLE. Because blank columns on the card are read as zero data, there should always be 160 bytes of data stored in the Buffer. It is important to note that each card of the test deck is punched exactly the same, which is a necessity since the master image cannot be changed.

2. TAPE FORMAT

The object tape (Part Number 06-038M09) is an absolute tape with the starting location designated as X'80'. The program requires 844_{16} or $2,116_{10}$ bytes of memory.

3. LOADING PROCEDURE

To load the Card Reader Test program, the 50 sequence must be manually entered into memory. The 50 sequence is then used to load the Absolute Loader, Part Number 06-023M10, which in turn loads the object tape. For further information on tape loading, refer to Loader Descriptions, Publication Number 06-025A12.

4. SWITCH OPTIONS

The Card Reader Test program is designed to sense designated Data/Address switches on the front of the Display Panel and initiate special operations based on the switch settings. The switches are sensed after each card has been read. Thus, the switch settings may be changed while the test is in operation. Data/Address switches 12 through 15 are sensed. Table 1 is a list of the switch coding.

5. SWITCH PRIORITY

Because more than one switch may be depressed at any given time, a priority of switches has been set up within the program. Table 2 is a list of switch priorities and their uses. Figure 1 is a flow chart which illustrates the operation of the program.

TABLE 1

Depressed Switch	Meaning	Depressed Switch	Meaning
Normal (No Switch Set)	Compares each card read to the master image and prints all error messages	Switch 14	The first card read after SW14 has been depressed is stored into an area in memory referred to as IMTABL (New Image Table). As long as SW14 is depressed, every card read is compared to the new image instead of the master image. If a comparison error is detected, an error message is printed and a list of that card is automatically given. No further comparison will be made with that card. The new image formed does not destroy the master image with which the normal test deck is compared. As soon as SW14 is released, the next card is compared to the master image.
Switch 12	Does not make any comparison of cards read. Every printable character read from each card is output to the device specified by the contents of location X'7E' (LISTDV) of the 50 sequence. This option allows any form of punched card to be listed. Note that blanks on the card are typed as spaces and non-printable characters are skipped.		
Switch 13	Suppresses all error message printouts while the test deck is being read. When the input hopper becomes empty, messages are typed to indicate any errors encountered.	Switch 15	Prints a listing of the card after the comparison test.

TABLE 2

Switch Number	Meaning	Switch Number	Meaning
SW12	SW12 has the highest priority. Regardless of what other switches are depressed, a listing of each card is the only function performed.		Note, this will not suppress a listing caused by SW12 or SW15.
SW13	SW13 surpresses all error message printouts during the reading of the test deck.	SW15	After a comparison has been made and all error messages printed, this setting causes the card to be listed in its entirety.

6. ERROR MESSAGES AND THEIR MEANINGS

There are seven error messages. Four messages will be typed during the reading of the test deck, if errors are found, and three will be typed when the card reader input hopper becomes empty. If no errors are detected in running the test, a message of "A OK" will be printed. Table 3 is a list of error messages and their meanings. Appendix 1 provides a listing of the test program.

TABLE 3

Error Message	Meaning
Column Error	If 80 columns of data are not read from each card, this message is typed. No comparison is made in this case.
New Image Comparison Errors	When SW14 is depressed, and a comparison mismatch is found, this message is printed. A listing is automatically given and no further comparison of the card to the new image is made. SW13 suppresses this form of error listing.
Was Read As..	If SW14 is not depressed, and there is an error in comparing a card from the normal test deck to the master image, this message occurs. Following this message is a printout of the number of each row (12, 11, 0, 1 ... 9) where the card reader read a punch.

TABLE 3
(Continued)

Error Message	Meaning
Should Have Been ...	This message follows the "Was Read As ..." message and prints the row number where punches should have been read.
Hopper Empty Messages	
Error In Bit 0, 1, 8, and/ or 9	Bits 0, 1, 8, and 9 of the 16 bit halfword formed from each column of Hollerith data should always be zero. If while performing the test, one of these four bits is sensed as a "1", it will be masked to a zero. A comparison is made of the entire card, but this message is typed upon completion of the test to indicate the malfunction.
Column Error	If 80 columns of data were not read from every card of the test deck, this message occurs when the test is completed.
Data Comparison Errors	If at any time a mismatch is found when comparing the cards from the test deck to the master image, or the new image, this message is typed when the input hopper becomes empty.

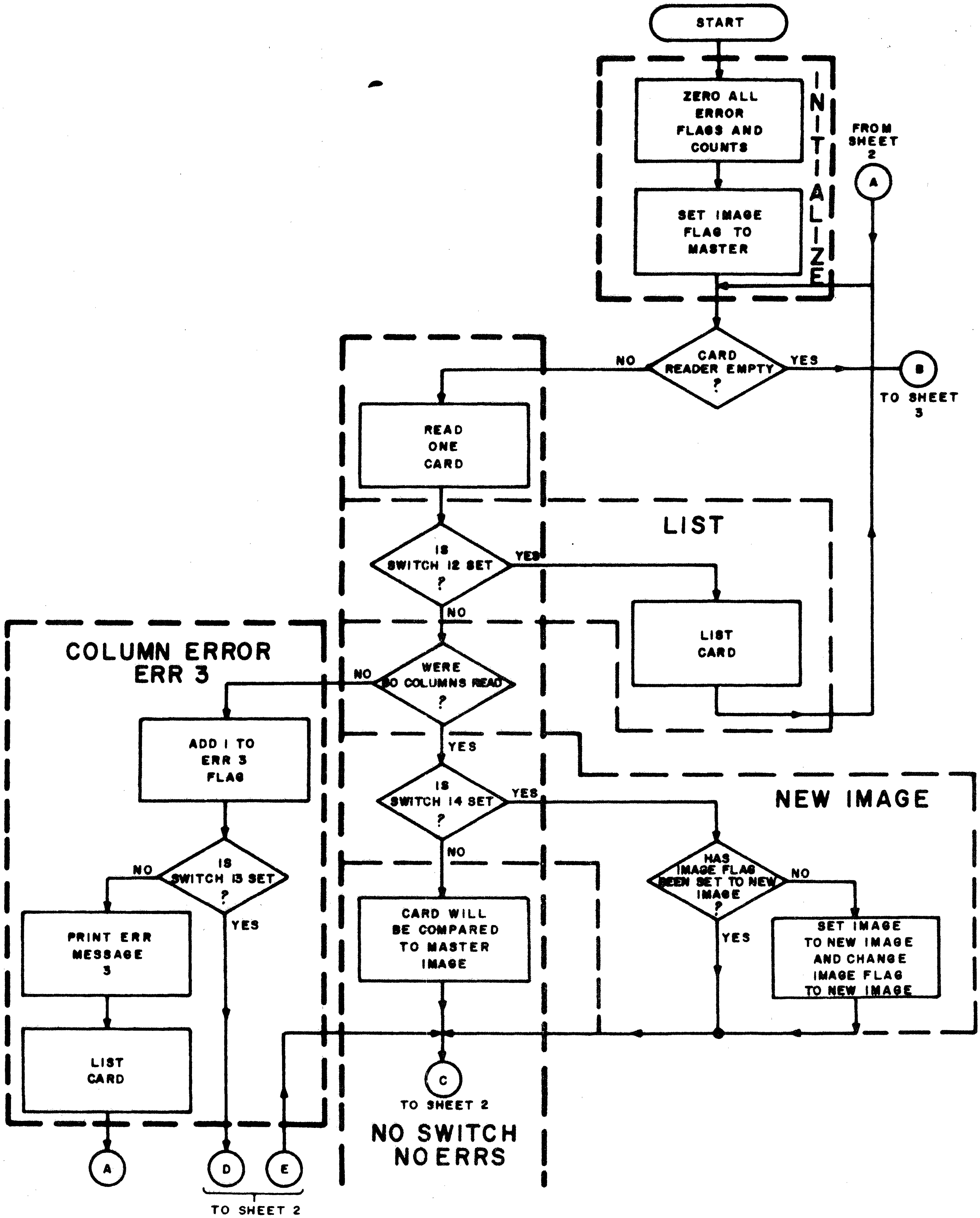


FIGURE 1. TEST FLOW CHART, SHEET 1 OF 4

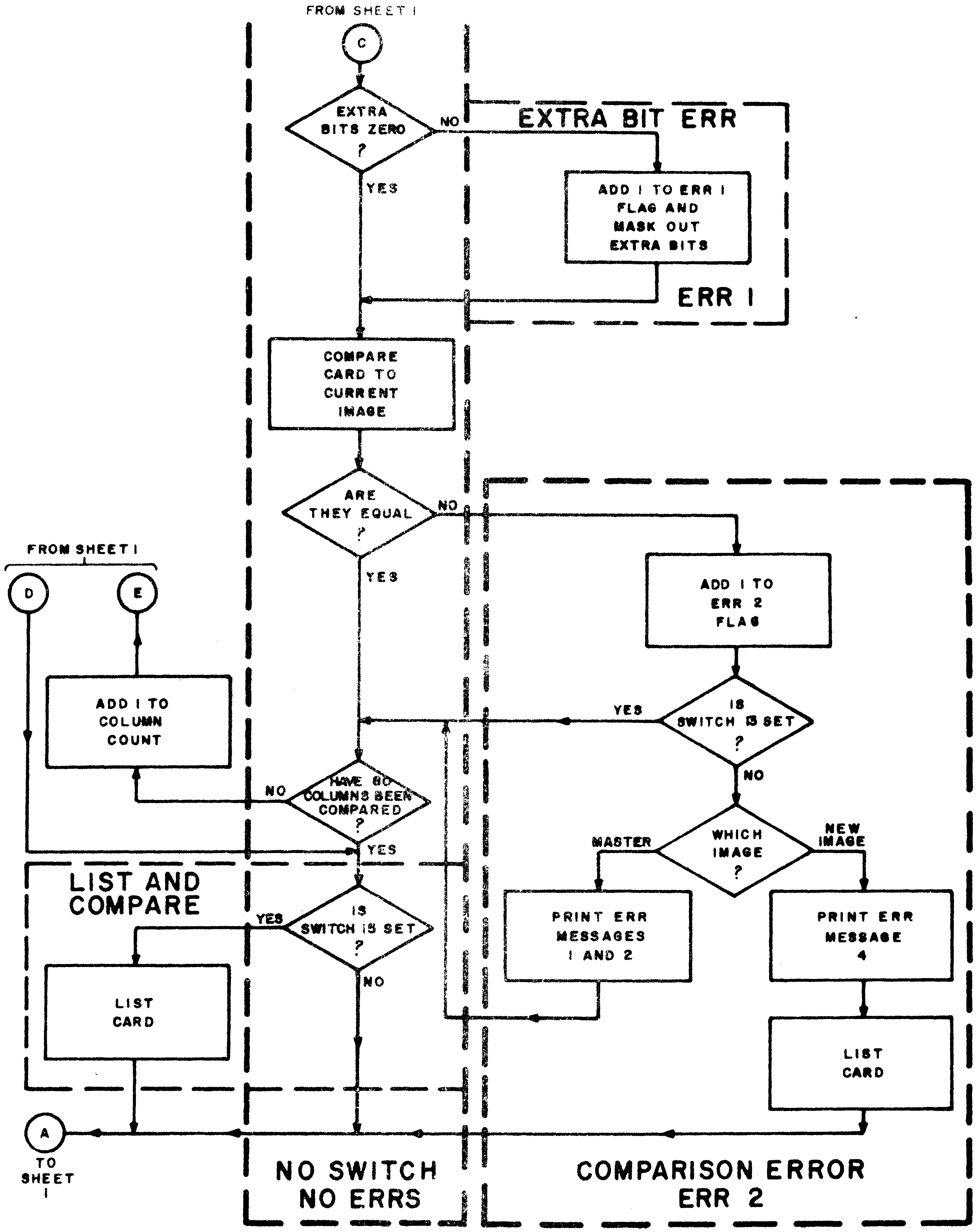


FIGURE 1. TEST FLOW CHART, SHEET 2 OF 4

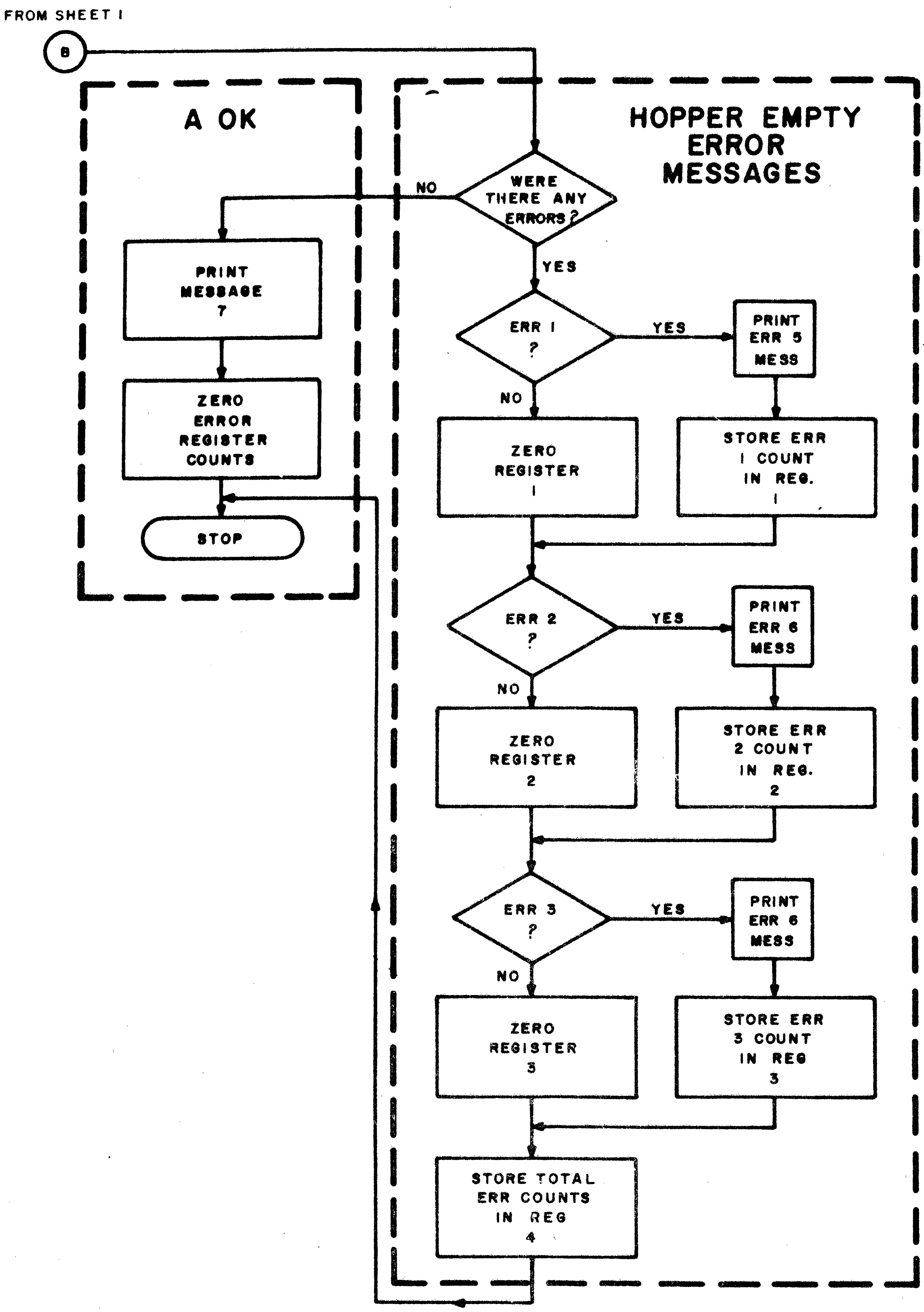


FIGURE 1. TEST FLOW CHART, SHEET 3 OF 4

NOTES

MESS 1 - "SHOULD HAVE BEEN" (WILL PRINT ROW NUMBERS WHERE PUNCHES SHOULD HAVE BEEN READ)

MESS 2 - "WAS READ AS" (WILL PRINT ROW NUMBERS WHERE PUNCHES WERE READ)

MESS 3 - "COLUMN ERROR" (8Ø COLUMNS NOT READ)

MESS 4 - "NEW IMAGE COMPARISON ERROR"

MESS 5 - "ERROR IN BITS Ø, 1, 8 AND/OR 9"

MESS 6 - "DATA COMPARISON ERRORS"

MESS 7 - "A OK" (NO ERRORS)

HOPPER* EMPTY MESS

ERR 1 - MESS 5*

ERR 2 - MESS 2 - MESS 1 - MESS 4 - MESS 6*

ERR 3 - MESS 3*

NO ERRORS - MESS 7*

UPON COMPLETION

REG. 1 - EXTRA BIT ERR COUNT

REG. 2 - COMPARISON ERR COUNT

REG. 3 - COLUMN ERR COUNT

REG. 4 - TOTAL NUMBER OF ERR COUNTS

MEANING OF SWITCHES

SW12 - UNCONDITIONAL CARD LIST (NO TEST)

SW13 - SUPPRESSES ALL ERROR MESSAGE PRINTOUTS UNTIL HOPPER IS EMPTY

SW14 - ESTABLISHES A NEW IMAGE AND MAKES A COLUMN COMPARISON

SW15 - CAUSES LIST OF CARDS TO BE GIVEN UPON COMPLETION OF COMPARISON

NO SW- COMPARES AND PRINTS ALL ERROR MESSAGES

FIGURE 1. TEST FLOW CHART, SHEET 4 OF 4

7. ERROR COUNTS

While reading the test deck, a count of each of the three types of errors is recorded if any should occur. Upon completion of the test, these counts are stored in General Registers for easy accessability by the Serviceman or user. Table 4 explains this feature.

8. USER NOTE

It is suggested that if this test is being used to test the validity of the Card Reader, that No Switches be depressed. This will speed the process up if there are no errors and it will also give the serviceman a good working guide from the error message printouts, should there be errors.

After verifying that the Card Reader is operating properly, this same test, with SW12 set, can be used to check the specified list device (X'78').

9. TEST DECK

The cards required for the test deck are as shown in Figure 2.

TABLE 4

General Register	Error
Register 1	"Error in Bits 0, 1, 8 and/ or 9"
Register 2	"Data Comparison Errors" NOTE: If SW14 is depressec and SW13 is released, this count will not be entirely accurate. As soon as the first comparison error is encountered, the message "NEW IMAGE COMPARISON ERROR" will be typed and a listing of the card will be printed. The program has been designed so that no further comparison of that card is made.
Register 3	"Column Error"
Register 4	This register will contain the total count of all errors.

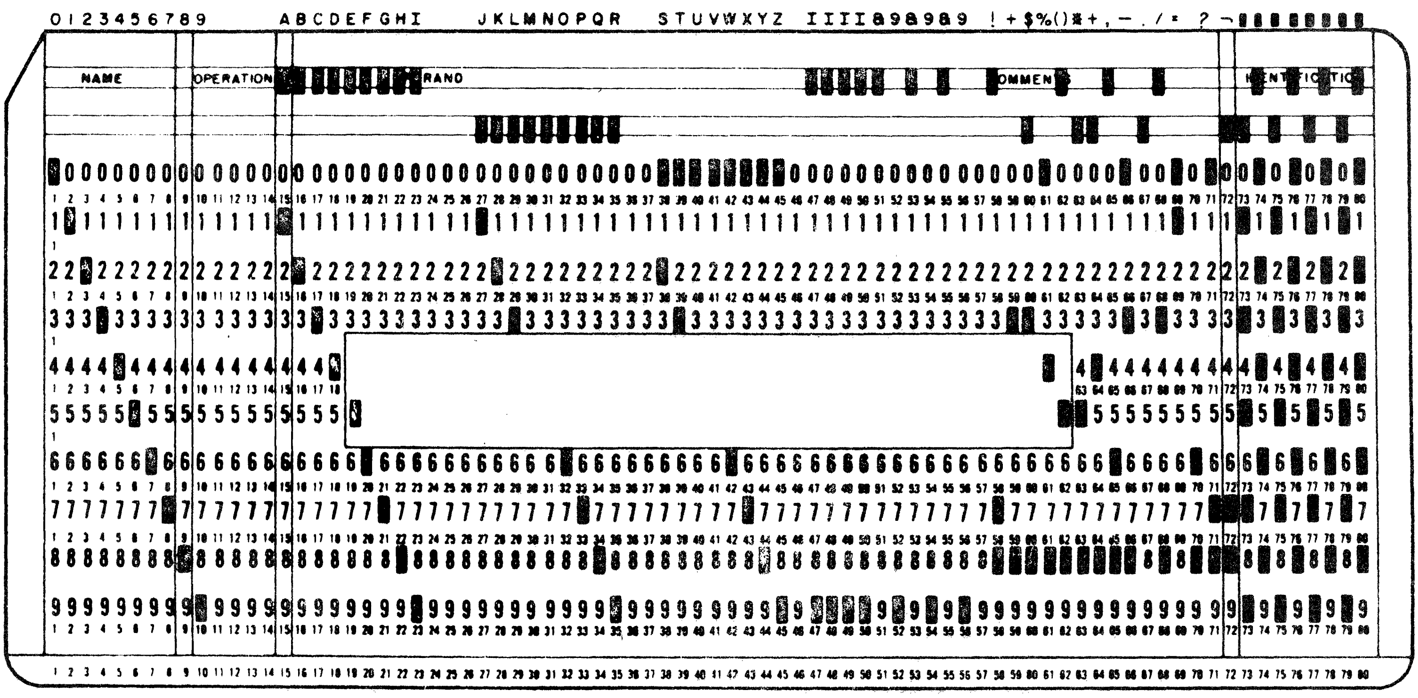


FIGURE 2. CARD IMAGE FOR TEST DECK

APPENDIX 1
CARD READER TEST PROGRAM LISTING

```
                    OPT    PASS2,PRINT,PUNCH,STOP
              *
              * CARD QEADER TEST WITH SWITCH OPTIONS
              *
0080                ORG    X'80'
              *
0000          DEVNUM EQU   0
0001          ONE    EQU   1
0002          INDEX  EQU   2
0003          INCR   EQU   3
0004          LIMIT  EQU   4
0005          STAT   EQU   5
0006          TEMP   EQU   6
0007          ERRCNT EQU   7
0008          CHECK  EQU   8
0009          R9     EQU   9
000A          INDEX2 EQU   10
000B          INCR2  EQU   11
000C          LIMIT2 EQU   12
000D          OUT    EQU   13
000E          R14    EQU   14
000F          R15    EQU   15
000B          ERRINC EQU   INCR2
0004          R4     EQU   4
0003          R3     EQU   3
0002          R2     EQU   2
0001          R1     EQU   1
000A          R10    EQU   10
              *
              * SET NEW IMAGE FLAG TO -1
              *
              * ROUTINE TO ZERO ERR FLAGS,
              *
              * SET UP CONSTANTS
              *
0080  C820    START  LHI   INDEX,-1
      FFFF
0084  4020           STH   INDEX,IMFLAG
      0528
0088  0B77           SHR   ERRCNT,ERRCNT
008A  4070           STH   ERRCNT,ERR1FL
      0522
008E  4070           STH   ERRCNT,ERR2FL
      0524
0092  4070           STH   ERRCNT,ERR3FL
      0526
0096  C840           LHI   LIMIT,158
      009E
009A  C830           LHI   INCR,2
      0002
              *
              * CARD INPUT ROUTINE
              *
009E  0B22    NXTCRD SHR   INDEX,INDEX
```

```
00A0  C8B0           LHI    INCR2,1
      0001
00A4  C800           LHI    DEVNUM,4
      0004
00A8  DE00           OC     DEVNUM,FEED
      052A
00AC  9D05   SENSE   SSR    DEVNUM,STAT
00AE  0815           LHR    ONE,STAT
00B0  C410           NHI    ONE,X'20'
      0020
00B4  4230           BNZ    EMPTY
      0334
00B8  0855           LHR    STAT,STAT
00BA  4230           BNZ    SENSE
      00AC
00BE  DB02           RD     DEVNUM,BUFF(INDEX)
      03E2
00C2  DB02           RD     DEVNUM,BUFF+1(INDEX)
      03E3
00C6  C120           BXLE   INDEX,SENSE
      00AC
             *
             * ROUTINE TO CHECK SW12 FOR UNCONDITIONAL LIST
             *
00CA  080B           LHR    DEVNUM,INCR2
00CC  9B05           RDR    DEVNUM,STAT
00CE  0865           LHR    TEMP,STAT
00D0  C450           NHI    STAT,8
      0008
00D4  4230           BNZ    SW2
      010E
             *
             * ROUTINE TO INDICATE COLUMN ERRORS
             *
00D8  C520           CLHI   INDEX,160
      00A0
00DC  4230           BNE    ERR3
      00E4
00E0  4300           B      SWCH
      0140
             *
             * MESS3 IS ERROR MESSAGE WHEN 80 COLUMNS
             * WERE NOT READ FROM INCOMING CARD
             *
             *
00E4  4870   ERR3    LH     ERRCNT,ERR3FL
      0526
00E8  0A7B           AHR    ERRCNT,ERRINC
00EA  4070           STH    ERRCNT,ERR3FL
      0526
00EE  0856           LHR    STAT,TEMP
00F0  C450           NHI    STAT,4
      0004
00F4  4230           BNZ    SW15
```

```
       014E
00F8   4300          B      SW13NO
       00FC
              *
              * 80/80  LISTING
              * AND ROUTINE TO INDICATE COLUMN ERRORS
              *
00FC   C8C0   SW13NO LHI    LIMIT2,MESN3-MESS3
       000E
0100   0BAA          SHR    INDEX2,INDEX2
0102   D3DA   SW1    LB     OUT,MESS3(INDEX2)
       084E
0106   4190          BAL    R9,STATUS
       03CC
010A   C1A0          BXLE   INDEX2,SW1
       0102
010E   C8D0   SW2    LHI    OUT,X'0D0D'
       0D0D
0112   4190          BAL    R9,STATUS
       03CC
0116   C8D0          LHI    OUT,X'0A0A'
       0A0A
011A   4190          BAL    R9,STATUS
       03CC
011E   0842          LHR    LIMIT,INDEX
0120   0B22          SHR    INDEX,INDEX
0122   48A2   SW3    LH     R10,BUFF(INDEX)
       03E2
0126   41F0          BAL    R15,HTASCV
       0286
012A   4300          B      SW4
       0134
012E   08DA          LHR    OUT,R10
0130   4190          BAL    R9,STATUS
       03CC
0134   C120   SW4    BXLE   INDEX,SW3
       0122
0138   C840          LHI    LIMIT,158
       009E
013C   4300          B      NXTCRD
       009E
              *
              * SWITCH CHECKING ROUTINE
              *
0140   0856   SWCH   LHR    STAT,TEMP
0142   C450          NHI    STAT,2
       0002
0146   4230          BNZ    IMAGE
       025E
014A   4300          B      COMPAR
       015C
              *
              * 80/80 LISTING
              *
```

```
014E  0856   SW15    LHR   STAT,TEMP
0150  C450           NHI   STAT,1
      0001
0154  4330           BZ    NXTCRD
      009E
0158  4300           B     SW2
      010E
             *
             * COMPARES CARDS TO MASTER TO DETERMINE ERRORS
             *
015C  C850   COMPAR  LHI   STAT,-1
      FFFF
0160  D250           STB   STAT,IMFLAG
      0528
0164  C850           LHI   STAT,TABLE
      052C
0168  4050           STH   STAT,MASTER+2
      017E
016C  0B22   C1      SHR   INDEX,INDEX
016E  4882   C2      LH    CHECK,BUFF(INDEX)
      03E2
0172  0858           LHR   STAT,CHECK
0174  C450           NHI   STAT,X'C0C0'
      C0C0
0178  4230           BNZ   ERR1
      018C
             *
             * COMPARES CARD READ,COLUMN BY COLUMN, TO CURRENT
             * IMAGE TO DETERMINE IF THERE WERE ANY READ ERRORS
             *
017C  4582   MASTER  CLH   CHECK,TABLE(INDEX)
      052C
0180  4230           BNE   ERR2
      019E
0184  C120   M1      BXLE  INDEX,C2
      016E
0188  4300           B     SW15
      014E
             *
             * ROUTINE TO INDICATE EXTRA BIT ERRORS
             *
018C  4870   ERR1    LH    ERRCNT,ERR1FL
      0522
0190  0A7B           AHR   ERRCNT,ERRINC
0192  4070           STH   ERRCNT,ERR1FL
      0522
0196  C480           NHI   CHECK,X'3F3F'
      3F3F
019A  4300           B     MASTER
      017C
             *
             * ERROR MESSAGE OUTPUT FOR CARDS THAT DO NOT COMPAR
             * TO THE CURRENT IMAGE
             *
```

```
019E  4870   ERR2     LH     ERRCNT,ERR2FL
      0524
01A2  0A7B            AHR    ERRCNT,ERRINC
01A4  4070            STH    ERRCNT,ERR2FL
      0524
01A8  0856            LHR    STAT,TEMP
01AA  C450            NHI    STAT,4
      0004
01AE  4230            BNZ    M1
      0184
01B2  4850            LH     STAT,IMFLAG
      0528
01B6  4210            BM     MES2IN
      01BE
01BA  4300            B      MES4IN
      0248
             *
             * MESS2 IS FOR MASTER IMAGE COMPARISON ERRORS
             *
             * "READ AS"
             *
01BE  C8C0   MES2IN   LHI    LIMIT2,MESN2-MESS2
      000E
01C2  0BAA            SHR    INDEX2,INDEX2
01C4  D3DA   MEST2    LB     OUT,MESS2(INDEX2)
      083E
01C8  4190            BAL    R9,STATUS
      03CC
01CC  C1A0            BXLE   INDEX2,MEST2
      01C4
             *
             * ROUTINE THAT DETERMINES IN WHAT ROWS PUNCHES WERE READ
             *
01D0  C8E0            LHI    R14,X'2000'
      2000
01D4  0B11            SHR    ONE,ONE
01D6  08FE   MORE     LHR    R15,R14
01D8  04F8            NHR    R15,CHECK
01DA  4230            BNZ    PUNCH
      01F4
01DE  0A1B   FIN      AHR    ONE,INCR2
01E0  CCE0            SRHL   R14,1
      0001
01E4  4380            BFC    8,MORE
      01D6
01E8  4850            LH     STAT,IMFLAG
      0528
01EC  4210            BM     MES1IN
      0218
01F0  4300            B      DONE
      0240
             *
01F4  08D1   PUNCH    LHR    OUT,ONE
01F6  0ADD            AHR    OUT,OUT
```

```
01F8  48DD             LH    OUT,HOLE(OUT)
      05CC
01FC  08FD             LHR   R15,OUT
01FE  CCD0             SRHL  OUT,8
      0008
0202  4190             BAL   R9,STATUS
      03CC
0206  08DF             LHR   OUT,R15
0208  4190             BAL   R9,STATUS
      03CC
020C  C8D0             LHI   OUT,X'2D'
      002D
0210  4190             BAL   R9,STATUS
      03CC
0214  4300             B     FIN
      01DE
                 *
                 * MESS1 IS FOR MASTER IMAGE COMPARISON ERROR3
                 *
                 * "SHOULD HAVE BEEN"
                 *
0218  0BAA       MES1IN  SHR   INDEX2,INDEX2
021A  C8C0             LHI   LIMIT2,MESN1-MESS1
      0014
021E  D3DA       MEST1   LB    OUT,MESS1(INDEX2)
      0828
0222  4190             BAL   R9,STATUS
      03CC
0226  C1A0             BXLE  INDEX2,MEST1
      021E
                 *
                 * ROUTINE TO INDICATE IN WHAT ROWS PUNCHES SHOULD HAVE B
                 *
022A  08A2             LHR   INDEX2,INDEX
022C  CDA0             SLHL  INDEX2,2
      0002
0230  C8CA             LHI   LIMIT2,7(INDEX2)
      0007
0234  D3DA       LODBYT  LB    OUT,VALUE+2(INDEX2)
      05EA
0238  4190             BAL   R9,STATUS
      03CC
023C  C1A0             BXLE  INDEX2,LODBYT
      0234
0240  C120       DONE    BXLE  INDEX,C2
      016E
0244  4300             B     NXTCRD
      009E
                 *
                 * MESS4 IS FOR NEW IMAGE COMPARISON ERRORS
                 *
                 * "NEW IMAGE COMPARISON ERRORS"
                 *
0248  0BAA       MES4IN  SHR   INDEX2,INDEX2
```

```
024A  C8C0          LHI   LIMIT2,MESN4-MESS4
      001E
024E  D3DA  MEST4   LB    OUT,MESS4(INDEX2)
      085E
0252  4190          BAL   R9,STATUS
      03CC
0256  C1A0          BXLE  INDEX2,MEST4
      024E
025A  4300          B     SW2
      010E
            *
            * CARDS WILL BE COMPARED TO A NEW IMAGE
            *
025E  4850  IMAGE   LH    STAT,IMFLAG
      0528
0262  0A5B          AHR   STAT,INCR2
0264  4050          STH   STAT,IMFLAG
      0528
0268  4230          BNZ   C1
      016C
026C  0B22          SHR   INDEX,INDEX
026E  4852  CHANGE  LH    STAT,BUFF(INDEX)
      03E2
0272  4052          STH   STAT,IMTABL(INDEX)
      0482
0276  C120          BXLE  INDEX,CHANGE
      026E
027A  C850          LHI   STAT,IMTABL
      0482
027E  4050          STH   STAT,MASTER+2
      017E
0282  4300          B     SW15
      014E
            *HOLLERITH TO ASCII CONVERSION ROUTINE
            *R10 IS COLUMN BINARY TO BE CONVERTED
            *R15 IS RETURN ADDRESS
0286        HTASCV  EQU   *
000A        RA      EQU   10
000B        RB      EQU   11
000C        RC      EQU   12
000D        RD      EQU   13
000E        RE      EQU   14
            *
0286  C4A0          NHI   RA,X'3F3F'       MASK DATA BITS
      3F3F
028A  08BA          LHR   RB,RA
028C  C8D0          LHI   RD,2             SET BXLE INCR
      0002
0290  CCB0          SRHL  RB,11
      000B
0294  D3BB          LB    RB,ZTAB(RB)      GET COUNT FROM ZONE TABLE
      02D2
0298  C5B0          CLHI  RB,X'FF'
      00FF
```

```
029C  4330          BE    ERROR                 FF MEANS ILLEGAL ZONE
      02BC
02A0  08CD          LHR   RC,RD
02A2  04CA          NHR   RC,RA
02A4  06BC          OHR   RB,RC                 OR ROW 8 INTO ZONE COUNT
02A6  C4A0          NHI   RA,X'073D'
      073D
02AA  0BCC          SHR   RC,RC                 SET BXLE LIMITS
02AC  C8E0          LHI   RE,16
      0010
02B0  45AC   SCAN   CLH   RA,DTAB(RC)           SCAN DIGIT TABLE
      02DA
02B4  4330          BE    MATCH
      02C4
02B8  C1C0          BXLE  RC,SCAN
      02B0
02BC  C8A0   ERROR  LHI   RA,X'2A'              ASTERIK CHAR IN RA
      002A
02C0  430F          B     0(15)                 ERROR RETURN
      0000
02C4  CDC0   MATCH  SLHL  RC,2                  FORM INDEX FROM ZONE
      0002
02C8  0ACB          AHR   RC,RB                 AND DIGIT COUNTS
02CA  D3AC          LB    RA,ATAB(RC)           PICK UP ASCII FROM TABLE
      02EC
02CE  430F          B     4(15)                 NORMAL RETURN
      0004
             *
             * TABLE USED TO CONVERT FROM HOLLERITH TO ASCII
             * FOR LISTINGS
             *
02D2  0001   ZTAB   DC    X'0001',X'04FF',X'05FF',X'FFFF'
      04FF
      05FF
      FFFF
02DA  0000   DTAB   DC    X'0000',X'0400',X'0200',X'0100'
      0400
      0200
      0100
02E2  0020          DC    X'0020',X'0010',X'0008',X'0004',X'0001'
      0010
      0008
      0004
      0001
02EC  2030   ATAB   DC    X'2030',X'3859',X'2D26',X'5148'
      3859
      2D26
      5148
02F4  312F          DC    X'312F',X'2020',X'4A41',X'5C20'
      2020
      4A41
      5C20
02FC  3253          DC    X'3253',X'3A20',X'4B42',X'5D5B'
      3A20
```

```
        4B42
        5D5B
 0304   3354              DC      X'3354',X'232C',X'4C43',X'242E'
        232C
        4C43
        242E
 030C   3455              DC      X'3455',X'4025',X'4D44',X'2A3C'
        4025
        4D44
        2A3C
 0314   3556              DC      X'3556',X'275F',X'4E45',X'2928'
        275F
        4E45
        2928
 031C   3657              DC      X'3657',X'3D3E',X'4F46',X'3B2B'
        3D3E
        4F46
        3B2B
 0324   3758              DC      X'3758',X'223F',X'5047',X'5E21'
        223F
        5047
        5E21
 032C   395A              DC      X'395A',X'2020',X'5249',X'2020'
        2020
        5249
        2020
                *
                * WHEN HOPPER IS EMPTY THIS ROUTINE WILL STORE
                * ERR1 COUNT IN R1
                * ERR2 COUNT [redacted] IN R2
                * ERR3 COUNT IN R3
                * TOTAL ERRS COUNT IN R4
                * AND PRINT ERROR MESSAGES FOR EACH TYPE ERROR
                * OR A OK MESSAGE IF THERE WERE NO ERRORS
                *
 0334   4870    EMPTY     LH      ERRCNT,ERR1FL
        0522
 0338   4230              BNZ     MES5IN
        0382
 033C   0B11              SHR     R1,R1
 033E   4870    E1        LH      ERRCNT,ERR2FL
        0524
 0342   4230              BNZ     MES6IN
        039C
 0346   0B22              SHR     R2,R2
 0348   4870    E2        LH      ERRCNT,ERR3FL
        0526
 034C   4230              BNZ     MES3IN
        0368
 0350   0B33              SHR     R3,R3
 0352   0B44    E3        SHR     R4,R4
 0354   4A40              AH      R4,ERR1FL
        0522
 0358   4A40              AH      R4,ERR2FL
```

```
        0524
035C    4A40             AH     R4,ERR3FL
        0526
0360    4330             BFC    3,MES7IN
        03B6
0364    C200             LPSW   STOP
        03DE
                 *
                 * ROUTINE TO INDICATE THAT THERE ARE COLUMN ERRORS
                 *
0368    0BAA     MES3IN  SHR    INDEX2,INDEX2
036A    C8C0             LHI    LIMIT2,MESN3-MESS3
        000E
036E    D3DA     MEST3   LB     OUT,MESS3(INDEX2)
        084E
0372    4190             BAL    R9,STATUS
        03CC
0376    C1A0             BXLE   INDEX2,MEST3
        036E
037A    4830             LH     R3,ERR3FL
        0526
037E    4300             B      E3
        0352
                 *
                 * MESS5 IS FOR ERROR IN BITS 0,1,8AND/OR 9 OF
                 * HOLLERITH CODE, 16 BIT HALF WORD PER COLUMN
0382    0BAA     MES5IN  SHR    INDEX2,INDEX2
0384    C8C0             LHI    LIMIT2,MESN5-MESS5
        0020
0388    D3DA     MEST5   LB     OUT,MESS5(INDEX2)
        087E
038C    4190             BAL    R9,STATUS
        03CC
0390    C1A0             BXLE   INDEX2,MEST5
        0388
0394    4810             LH     R1,ERR1FL
        0522
0398    4300             B      E1
        033E
                 *
                 * MESS6 IS ERROR MESSAGE TO INDICATE THAT
                 * THERE HAVE BEEN COMPARISON ERRORS.
                 *
039C    0BAA     MES6IN  SHR    INDEX2,INDEX2
039E    C8C0             LHI    LIMIT2,MESN6-MESS6
        0018
03A2    D3DA     MEST6   LB     OUT,MESS6(INDEX2)
        08A0
03A6    4190             BAL    R9,STATUS
        03CC
03AA    C1A0             BXLE   INDEX2,MEST6
        03A2
03AE    4820             LH     R2,ERR2FL
        0524
```

```
03B2  4300            B      E2
      0348
                *
                * MESS7 IS A OK MESSAGE FOR NO ERRORS DETECTED
                *
03B6  0BAA      MES7IN  SHR    INDEX2,INDEX2
03B8  C8C0              LHI    LIMIT2,MESN7-MESS7
      0008
03BC  D3DA      MEST7   LB     OUT,MESS7(INDEX2)
      08BA
03C0  4190              BAL    R9,STATUS
      03CC
03C4  C1A0              BXLE   INDEX2,MEST7
      03BC
03C8  C200              LPSW   STOP
      03DE
                *
                * OUTPUT  ROUTINE
                *
03CC  D300      STATUS  LB     DEVNUM,LISTDV
      007E
03D0  DE00              OC     DEVNUM,LISTDV+1
      007F
03D4  9D05              SSR    DEVNUM,STAT
03D6  42F0              BTC    X'F',STATUS
      03CC
03DA  9A0D              WDR    DEVNUM,OUT
03DC  0309              BR     R9
                *
03DE  8000      STOP    DC     X'8000',X'80'
      0080
                *
                * ALL DATA READ FROM CARDS IS STORED HERE
03E2            BUFF    DS     160
                *
                * THE NEW IMAGE IS STORED HERE
0482            IMTABL  DS     160
                *
                * ERR1 -- BITS 0,1,8 AND/OR 9 NOT ZERO
0522            ERR1FL  DS     2
                *
                * "DATA COMPARISON ERRORS"
0524            ERR2FL  DS     2
                *
                * "COLUMN ERROR"
0526            ERR3FL  DS     2
                *
0528            IMFLAG  DS     2
                *
052A  2020      FEED    DC     X'2020'
                *
007E            LISTDV  EQU    X'7E'
                *
                *
```

```
                    *PUNCHES IN ORDER EXPECTED ON CARD
                    * FOR THE MASTER IMAGE COMPARISON
                    *
052C                TABLE    EQU     *
                    *
052C  0800                   DC      X'0800'          0
052E  0400                   DC      X'0400'          1
0530  0200                   DC      X'0200'          2
0532  0100                   DC      X'0100'          3
0534  0020                   DC      X'0020'          4
0536  0010                   DC      X'0010'          5
0538  0008                   DC      X'0008'          6
053A  0004                   DC      X'0004'          7
053C  0002                   DC      X'0002'          8
053E  0001                   DC      X'0001'          9
0540  0000                   DC      X'0000'
0542  0000                   DC      X'0000'    } BLANKS
0544  0000                   DC      X'0000'
0546  0000                   DC      X'0000'
0548  2400                   DC      X'2400'          A
054A  2200                   DC      X'2200'          B
054C  2100                   DC      X'2100'          C
054E  2020                   DC      X'2020'          D
0550  2010                   DC      X'2010'          E
0552  2008                   DC      X'2008'          F
0554  2004                   DC      X'2004'          G
0556  2002                   DC      X'2002'          H
0558  2001                   DC      X'2001'          I
055A  0000                   DC      X'0000'
055C  0000                   DC      X'0000'    } BLANKS
055E  0000                   DC      X'0000'
0560  1400                   DC      X'1400'          J
0562  1200                   DC      X'1200'          K
0564  1100                   DC      X'1100'          L
0566  1020                   DC      X'1020'          M
0568  1010                   DC      X'1010'          N
056A  1008                   DC      X'1008'          O
056C  1004                   DC      X'1004'          P
056E  1002                   DC      X'1002'          Q
[redacted] 0570  [redacted] 1001   DC      X'1001'          R
0572  0000                   DC      X'0000'
0574  0000                   DC      X'0000'    } BLANKS
0576  0A00                   DC      X'0A00'          S
0578  0900                   DC      X'0900'          T
057A  0820                   DC      X'0820'          U
057C  0810                   DC      X'0810'          V
057E  0808                   DC      X'0808'          W
0580  0804                   DC      X'0804'          X
0582  0802                   DC      X'0802'          Y
[redacted] 0584  [redacted] 0801   DC      X'0801'          Z
0586  0000                   DC      X'0000'  - BLANK
0588  2001                   DC      X'2001'          I
058A  2001                   DC      X'2001'          I
058C  2001                   DC      X'2001'          I
```

```
058E  2001          DC    X'2001'          I
0590  2000          DC    X'2000'          &
0592  0001          DC    X'0001'          9
0594  2000          DC    X'2000'          &
0596  0001          DC    X'0001'          9
0598  2000          DC    X'2000'          &
059A  0001          DC    X'0001'          9
059C  0000          DC    X'0000'
059E  2006          DC    X'2006'          !
05A0  0102          DC    X'0102'          #
05A2  1102          DC    X'1102'          $
05A4  0822          DC    X'0822'          %
05A6  2012          DC    X'2012'          (
05A8  1012          DC    X'1012'          )
05AA  1022          DC    X'1022'          *
05AC  200A          DC    X'200A'          +
05AE  0902          DC    X'0902'          )
05B0  1000          DC    X'1000'          -
05B2  2102          DC    X'2102'          .
05B4  0C00          DC    X'0C00'          /
05B6  000A          DC    X'000A'          =
05B8  0806          DC    X'0806'          ?
05BA  1006          DC    X'1006'          ↑
05BC  1515          DC    X'1515'
05BE  2A2A          DC    X'2A2A'
05C0  1515          DC    X'1515'
05C2  2A2A          DC    X'2A2A'
05C4  1515   O.K.   DC    X'1515'   NONPRINTABLE  CHARACTERS
05C6  2A0A          DC    X'2A2A'
05C8  1515          DC    X'1515'
05CA  2A2A          DC    X'2A2A'
              *
              *
              * ASCII CODE OF PUNCHES READ
              *  FOR MASTER IMAGE COMPARISON TEST
              *
              *
05CC  3132    HOLE  DC    X'3132'          12
05CE  3131          DC    X'3131'          11
05D0  0030          DC    X'30'            0
05D2  0031          DC    X'31'            1
05D4  0032          DC    X'32'            2
05D6  0033          DC    X'33'            3
05D8  0000          DC    0
05DA  0000          DC    0
05DC  0034          DC    X'34'            4
05DE  0035          DC    X'35'            5
05E0  0036          DC    X'36'            6
05E2  0037          DC    X'37'            7
05E4  0038          DC    X'38'            8
05E6  0039          DC    X'39'            9
              *
              *
              * ASCII CODE OF PUNCHES THAT SHOULD HAVE READ
```

```
                    * FOR MASTER IMAGE COMPARISON TEST
                    *
                    *
05E8                VALUE   EQU    *
                    *
05E8  8D8A                  DC     X'8D8A',X'2020',X'2030',X'2020' 0
      2020
      2030
      2020
05F0  8D8A                  DC     X'8D8A',X'2020',X'2031',X'2020' 1
      2020
      2031
      2020
05F8  8D8A                  DC     X'8D8A',X'2020',X'2032',X'2020' 2
      2020
      2032
      2020
0600  8D8A                  DC     X'8D8A',X'2020',X'2033',X'2020' 3
      2020
      2033
      2020
0608  8D8A                  DC     X'8D8A',X'2020',X'2034',X'2020' 4
      2020
      2034
      2020
0610  8D8A                  DC     X'8D8A',X'2020',X'2035',X'2020' 5
      2020
      2035
      2020
0618  8D8A                  DC     X'8D8A',X'2020',X'2036',X'2020' 6
      2020
      2036
      2020
0620  8D8A                  DC     X'8D8A',X'2020',X'2037',X'2020' 7
      2020
      2037
      2020
0628  8D8A                  DC     X'8D8A',X'2020',X'2038',X'2020' 8
      2020
      2038
      2020
0630  8D8A                  DC     X'8D8A',X'2020',X'2039',X'2020' 9
      2020
      2039
      2020
0638  0D0A                  DC     X'0D0A',X'2020',X'2020',X'2000'      BLANK
      2020
      2020
      2000
0640  0D0A                  DC     X'0D0A',X'2020',X'2020',X'2000'      BLANK
      2020
      2020
      2000
0648  0D0A                  DC     X'0D0A',X'2020',X'2020',X'2000'      BLANK
```

```
         2020
         2020
         2000
0650     0D0A        DC    X'0D0A',X'2020',X'2020',X'2000'     BLANK
         2020
         2020
         2000
0658     8D8A        DC    X'8D8A',X'3132',X'2D31',X'2020'  A
         3132
         2D31
         2020
0660     8D8A        DC    X'8D8A',X'3132',X'2D32',X'2020'  B
         3132
         2D32
         2020
0668     8D8A        DC    X'8D8A',X'3132',X'2D33',X'2020'  C
         3132
         2D33
         2020
0670     8D8A        DC    X'8D8A',X'3132',X'2D34',X'2020'  D
         3132
         0D34
         2020
0678     8D8A        DC    X'8D8A',X'3132',X'2D35',X'2020'  E
         3132
         2D35
         2020
0680     8D8A        DC    X'8D8A',X'3132',X'2D36',X'2020'  F
         3132
         2D36
         2020
0688     8D8A        DC    X'8D8A',X'3132',X'2D37',X'2020'  G
         3132
         2D37
         2020
0690     8D8A        DC    X'8D8A',X'3132',X'2D38',X'2020'  H
         3132
         2D38
         2020
0698     8D8A        DC    X'8D8A',X'3132',X'2D39',X'2020'  I
         3132
         2D39
         2020
06A0     0D0A        DC    X'0D0A',X'2020',X'2020',X'2000'     BLANK
         2020
         2020
         2000
06A8     0D0A        DC    X'0D0A',X'2020',X'2020',X'2000'     BLANK
         2020
         2020
         2000
06B0     0D0A        DC    X'0D0A',X'2020',X'2020',X'2000'     BLANK
         2020
         2020
```

```
      2000
06B8  8D8A          DC    X'8D8A',X'3131',X'2D31',X'2020' J
      3131
      2D31
      2020
06C0  8D8A          DC    X'8D8A',X'3131',X'2D32',X'2020' K
      3131
      2D32
      2020
06C8  8D8A          DC    X'8D8A',X'3131',X'2D33',X'2020' L
      3131
      2D33
      2020
06D0  8D8A          DC    X'8D8A',X'3131',X'2D34',X'2020' M
      3131
      2D34
      2020
06D8  8D8A          DC    X'8D8A',X'3131',X'2D35',X'2020' N
      3131
      2D35
      2020
06E0  8D8A          DC    X'8D8A',X'3131',X'2D36',X'2020' O
      3131
      2D36
      2020
06E8  8D8A          DC    X'8D8A',X'3131',X'2D37',X'2020' P
      3131
      2D37
      2020
06F0  8D8A          DC    X'8D8A',X'3133',X'2D38',X'2020' Q
      3133
      2D38
      2020
06F8  8D8A          DC    X'8D8A',X'3131',X'2D39',X'2020' R
      3131
      2D39
      2020
0700  0D0A          DC    X'0D0A',X'2020',X'2020',X'2000'     BLANK
      2020
      2020
      2000
0708  0D0A          DC    X'0D0A',X'2020',X'2020',X'2000'     BLANK
      2020
      2020
      2000
0710  8D8A          DC    X'8D8A',X'2030',X'2D32',X'2020' S
      2030
      2D32
      2020
0718  8D8A          DC    X'8D8A',X'2030',X'2D33',X'2020' T
      2030
      2D33
      2020
0720  8D8A          DC    X'8D8A',X'2030',X'2D34',X'2020' U
```

```
      2030
      2D34
      2020
0728  8D8A          DC     X'8D8A',X'2030',X'2D35',X'2020'  V
      2030
      2D35
      2020
0730  8D8A          DC     X'8D8A',X'2030',X'2D36',X'2020'  W
      2030
      2D36
      2020
0738  8D8A          DC     X'8D8A',X'2030',X'2D37',X'2020'  X
      2030
      2D37
      2020
0740  8D8A          DC     X'8D8A',X'2030',X'2D38',X'2020'  Y
      2030
      2D38
      2020
0748  8D8A          DC     X'8D8A',X'2030',X'2D39',X'2020'  Z
      2030
      2D39
      2020
0750  0D0A          DC     X'0D0A',X'2020',X'2020',X'2000'      BLANK
      2020
      2020
      2000
0758  0D0A          DC     X'0D0A',X'3132',X'2D39',X'2020'     I
      3132
      2D39
      2020
0760  0D0A          DC     X'0D0A',X'3132',X'2D39',X'2020'     I
      3132
      2D39
      2020
0768  0D0A          DC     X'0D0A',X'3132',X'2D39',X'2020'     I
      3132
      2D39
      2020
0770  0D0A          DC     X'0D0A',X'3132',X'2D39',X'2020'     I
      3132
      2D39
      2020
0778  0D0A          DC     X'0D0A',X'2020',X'3132',X'2020'     &
      2020
      3132
      2020
0780  0D0A          DC     X'0D0A',X'2020',X'2039',X'2020'     9
      2020
      2039
      2020
0788  0D0A          DC     X'0D0A',X'2020',X'3132',X'2020'     &
      2020
      3132
```

```
       2020
0790   0D0A       DC    X'0D0A',X'2020',X'2039',X'2020'    9
       2020
       2039
       2020
0798   8D8A       DC    X'8D8A',X'2020',X'3132',X'2020' &
       2020
       3132
       2020
07A0   0D0A       DC    X'0D0A',X'2020',X'2039',X'2020'    9
       2020
       2039
       2020
07A8   0D0A       DC    X'0D0A',X'2020',X'2020',X'2000'    BLANK
       2020
       2020
       2000
07B0   0D0A       DC    X'0D0A',X'3132',X'2D38',X'2D37'    !
       3132
       2D38
       2D37
07B8   0D0A       DC    X'0D0A',X'2038',X'2D33',X'2020'    #
       2038
       2D33
       2020
07C0   0D0A       DC    X'0D0A',X'3131',X'2D38',X'2D33'    $
       3131
       2D38
       2D33
07C8   0D0A       DC    X'0D0A',X'2030',X'2D38',X'2D34'    %
       2030
       2D38
       2D34
07D0   8D8A       DC    X'8D8A',X'3132',X'2D35',X'2D38' (
       3132
       2D35
       2D38
07D8   8D8A       DC    X'8D8A',X'3131',X'2D35',X'2D38' )
       3131
       2D35
       2D38
07E0   8D8A       DC    X'8D8A',X'3131',X'2D34',X'2D38' *
       3131
       2D34
       2D38
07E8   8D8A       DC    X'8D8A',X'3132',X'2D36',X'2DB8' +
       3132
       2D36
       2DB8
07F0   8D8A       DC    X'8D8A',X'2030',X'2D38',X'2D33' ,
       2030
       2D38
       2D33
07F8   8D8A       DC    X'8D8A',X'2020',X'3131',X'2020' -
```

```
       2020
       3131
       2020
0800   8D8A              DC     X'8D8A',X'3132',X'2D38',X'2D33'  .
       3132
       2D38
       2D33
0808   8D8A              DC     X'8D8A',X'20B0',X'2D31',X'2020'  /
       20B0
       2D31
       2020
0810   8D8A              DC     X'8D8A',X'2036',X'2D38',X'2020'  =
       2036
       2D38
       2020
0818   0D0A              DC     X'0D0A',X'2030',X'2D38',X'2D37'     ?
       2030
       2D38
       2D37
0820   0D0A              DC     X'0D0A',X'3131',X'2D38',X'2D37'     ↑
       3131
       2D38
       2D37
              *
0828   0D0A   MESS1      DC     X'0D0A'
082A   5348              DC     C'SHOULD          HAVE BEEN'
       4F55
       4C44
       2020
       4841
       5645
       2042
       4545
       4E20
083C   2020   MESN1      DC     X'2020'
              *
083E   0D0A   MESS2      DC     X'0D0A'
0840   5741              DC     C'WAS             READ AS'
       5320
       5245
       4144
       2041
       5320
084C   2020   MESN2      DC     X'2020'
              *
084E   0D0A   MESS3      DC     X'0D0A'
0850   434F              DC     C'COLUMN          ERROR'
       4C55
       4D4E
       2045
       5252
       4F52
085C   2020   MESN3      DC     X'2020'
              *
```

```
085E  0D0A    MESS4   DC    X'0D0A'
0860  4E45            DC    C'NEW              IMAGE COMPARISON ERRORS'
      5720
      494D
      4147
      4520
      434F
      4D50
      4152
      4953
      4F4E
      2045
      5252
      4F52
      5320
087C  2020    MESN4   DC    X'2020'
              *
087E  0D0A    MESS5   DC    X'0D0A'
0880  4552            DC    C'ERROR            IN BITS 0,1,8, AND/OR 9'
      524F
      5220
      494E
      2042
      4954
      5320
      302C
      312C
      382C
      2041
      4E44
      2F4F
      5220
      3920
089E  2020    MESN5   DC    X'2020'
08A0  0D0A    MESS6   DC    X'0D0A'
08A2  4441            DC    C'DATA             COMPARISON ERRORS'
      5441
      2043
      4F4D
      5041
      5249
      534F
      4E20
      4552
      524F
      5253
08B8  2020    MESN6   DC    X'2020'
              *
08BA  0D0A    MESS7   DC    X'0D0A'
08BC  2041            DC    C'                 A   OK'
      2020
      4F4B
08C2  2020    MESN7   DC    X'2020'
              *
```

```
08C4                    END

ATAB     02EC
BUFF     03E2
C1       016C
C2       016E
CHANGE   026E
CHECK    0008
COMPAR   015C
DEVNUM   0000
DONE     0240
DTAB     02DA
E1       033E
E2       0348
E3       0352
EMPTY    0334
ERR1     018C
ERR1FL   0522
ERR2     019E
ERR2FL   0524
ERR3     00E4
ERR3FL   0526
ERRCNT   0007
ERRINC   000B
ERROR    02BC
FEED     052A
FIN      01DE
HOLE     05CC
HTASCV   0286
IMAGE    025E
IMFLAG   0528
IMTABL   0482
INCR     0003
INCR2    000B
INDEX    0002
INDEX2   000A
LIMIT    0004
LIMIT2   000C
LISTDV   007E
LODBYT   0234
M1       0184
MASTER   017C
MATCH    02C4
MES1IN   0218
MES2IN   01BE
MES3IN   0368
MES4IN   0248
MES5IN   0382
MES6IN   039C
MES7IN   03B6
MESN1    083C
MESN2    084C
MESN3    085C
MESN4    087C
```

MESN5	089E
MESN6	08B8
MESN7	08C2
MESS1	0828
MESS2	083E
MESS3	084E
MESS4	085E
MESS5	087E
MESS6	08A0
MESS7	08BA
MEST1	021E
MEST2	01C4
MEST3	036E
MEST4	024E
MEST5	0388
MEST6	03A2
MEST7	03BC
MORE	01D6
NXTCRD	009E
ONE	0001
OUT	000D
PUNCH	01F4
R1	0001
R10	000A
R14	000E
R15	000F
R2	0002
R3	0003
R4	0004
R9	0009
RA	000A
RB	000B
RC	000C
RD	000D
RE	000E
SCAN	02B0
SENSE	00AC
START	0080
STAT	0005
STATUS	03CC
STOP	03DE
SW1	0102
SW13NO	00FC
SW15	014E
SW2	010E
SW3	0122
SW4	0134
SWCH	0140
TABLE	052C
TEMP	0006
VALUE	05E8
ZTAB	02D2

READER COMMENTS

The General Electric Company solicits your comments on publications covering Process Computer equipment. Please explain any "no" responses in the COMMENTS section. Your comments and suggestions become the property of the General Electric Company.

- Name of Manual: ______________________
- What is your computer application: ______________________
  ____
- How is this publication used:

  Familiarization ☐ Reference ☐

  Training ☐ Maintenance ☐

  Other (Explain) ______________________

	YES	NO
• Does this publication meet your requirements	☐	☐
• Is the material:	☐	☐
1) Presented in clear text	☐	☐
2) Conveniently organized	☐	☐
3) Adequately detailed	☐	☐
4) Adequately illustrated	☐	☐
5) Presented at appropriate technical level	☐	☐

- Please provide specific text references (page number, line, etc.) with your comments.

NAME ______________________ DATE ______________________

TITLE ______________________

COMPANY NAME ______________________

AND ADDRESS ______________________

COMMENTS:

No postage necessary if mailed in the U. S. A.

Staple

Communications concerning Technical Publications should be directed to:

Manager, Technical Publications
GE Process Computer Department
2255 West Desert Cove Road
Phoenix, Arizona 85029

Fold — Fold

FIRST CLASS
Permit No. 4091
Phoenix, Arizona

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

GENERAL ELECTRIC COMPANY
PROCESS COMPUTER DEPARTMENT
2255 West Desert Cove Road
Phoenix, Arizona 85029

Attention: Technical Publications

Cut Along Line

Fold — Fold

Additional Comments: