PROGRAMMING

FR-80

INFORMATION INTERNATIONAL 12435 WEST OLYMPIC BLVD. LOS ANGELES, CA. 90064

8211

PROGRAMMER'S MANUAL

FOR THE

FR 80 COMPUTER OUTPUT MICROFILM RECORDER

(Second Edition)

July 1971

Published by

Information International, Inc.
12435 West Olympic Boulevard
Los Angeles, California 90064

CHANGE RECORD

Change No.	Date of Issue	Signature of Person Entering Change in This Book	Date of Entry

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1
INTRODUCTION

The FR 80 Programmer's Manual is provided for use by experienced assembly language programmers who are modifying FR 80 software or writing new FR 80 programs. It is not intended for the "open shop" user. The FR 80 User's Manual is provided for that purpose.

This chapter is concerned with the overall aspects of FR 80 software, such as philosophy, maintenance, etc. Chapter 2 describes the available host computer software. Chapter 3 outlines the standard FR 80 software, which is explained in greater detail in the FR 80 Operator's Manual and the FR 80 User's Manual. Chapters 4, 5, and 6 discuss operating systems, utility programs and the writing of new software for the FR 80. These three chapters are essential to the programmer who is going to work with FR 80 assembly language. This material is not covered in any other Information International publication.

1.1 THE FR 80 SYSTEM

The FR 80 is designed and manufactured by Information International, Inc. (hereinafter called III). The system normally operates off-line. By reading digital data from magnetic tape, the system processes the information and records characters and vectors on a high-precision, cathode-ray tube. A special camera system photographs the face of the tube. The resulting film is processed into either a positive or a negative image. Film duplicates, hardcopy, or printing plates can be made from the original film.

The FR 80 is organized into four functional sections:

1. Input section
2. Processor
3. Data translator
4. Recording unit

1.1.1 Input Section

Standard FR 80 input consists of 7- or 9-track magnetic tape units, a master tape controller, Teletype, and paper tape reader. The master tape controller may be expanded to four magnetic tape units and provides switch selection of the desired input. The input section controls the flow of data to the processor at a nominal transfer rate of 18,000 18-bit words per second. The Teletype and paper tape reader serve as a 10-character-per-second auxiliary communications link with the processor unit.

1.1.2 Processor

The processor is an 18-bit binary computer, with a 4096-word memory. Serving as the central control unit of the system, the processor comfines operating data and plotting instructions for routing to the data translator. Under program control, the processor instructs the data translator to generate the alphanumerics, vectors, and special forms required. A display monitor provides the operator with a window into the system. An 8-inch by 10-inch cathode-ray display tube is driven directly from the precision light source to provide an accurate view of the recorded image.

1.1.3 Data Translator

The high precision and versatility of the FR 80 is determined by the function generators and control circuitry contained in the data translator, which is subdivided into a vector generator, character generator, point plot circuitry, and control circuits for the monitor and recording section. Upon command from the processor, the digital data received by the translator is converted to analog signals that control the precision light source deflection beam. The deflection drive signals are corrected for linearity, focus, and astigmatism before routing to the light source deflection coils. Control signals from the data translator maintain control of the camera and monitor functions.

1.1.4 Recording Unit

Electrical signals are converted into a recorded film image in the recording unit, which comprises a precision light source, optics, and microfilm camera. Electromagnetic deflection is used to position the light source beam and achieve the best possible image quality. The created image is focused by the optical system and recorded by the microfilm camera. The recording cameras are available with incremental pulldown. The flexibility of the FR 80 permits the addition of Miracode and other retrieval codes to the microfilm record.

1.2 SOFTWARE DESIGN PHILOSOPHY

The Information International FR 80 carefully balances hardware and software to provide the greatest flexibility to the user.

The FR 80 includes a programmable controller and display monitor. All the software is capable of creation on the FR 80 itself. Insurance against obsolescence is assured by maintaining a high degree of flexibility through software control of all major functions. In other words, rather than fixed hardware capable only of accepting a limited number of formats, the FR 80 system is organized to accept virtually any formatted tape.

This allows the FR 80 to replace any existing COM installation without change of host computer software, tape formats, or microform requirements. The result is a higher quality, more uniform, more readable image. In addition, a broad range of features and capabilities are available for future applications growth. A further feature is a continuing extension of FR 80 capability through maintenance and additions to distributed software.

1.3 MAINTENANCE OF FR 80 SOFTWARE

All of III's software is maintained at Los Angeles on a 16K FR 80 with the disk option. There are four basic FR 80 software systems: SC 4020 Simulator, META Interpreter, Print Simulator, and CalComp Simulator. Recognizing that there is an almost unlimited number of possible combinations for an FR 80 (such as two operating systems, 4K memory, 8K memory, disk, several types of magnetic tape formats, at least seven standard camera options, three character fonts, more than ten character sets, plus a long list of features from which a user can choose), III has designed each software system so that an III operator can enter the hardware configuation and software features desired by a given customer for a particular application and assemble the appropriate binary programs These programs are distributed to the FR 80 user on program system tapes, and are subsequently updated with additional distributions incorporating improved performance, additional features, etc. Each system tape distribution contains a list of the binary programs plus an information bulletin outlining changes and improvements from prior distributions.

A copy of the system tape for each FR 80 user is maintained at III headquarters in Los Angeles for use in duplicating identical conditions to those encountered in the field.

Should a "bug" or an unusual condition develop at the user's site, III will test the same condition and give a "patch" over the telephone.

1.4 III POLICY ON NEW SOFTWARE

The software systems provided for the FR 80 are very flexible. The built-in features allow the operator to adapt to almost any requirement. Should a particular requirement develop that is not within the scope of the standard software, there are several alternatives open:

1. If the requirement appears to be of general use, III will add it to its software development plan and will provide this as part of the standard software at no charge. If the user cannot wait for the normal distribution of an additional feature, III will quote the charge for a temporary solution.
2. If the application is of a special nature peculiar to one customer, III will provide the required program at time-and-material rates. Price and delivery date will be quoted on request.
3. Should the user desire to develop such programs internally, III offers, at no charge, its proprietary subroutine package together with documentation and training to enable the user to write his own software.

1.5 USER PROGRAMMING AIDS

Many FR 80 users have program staffs already established for their computer installation. They have become accustomed to modifying vendor-distributed software to meet their own specific requirements. In order to do that, they normally request symbolic listings of all software provided by the vendor. In the case of the FR 80, the software distributed on system tapes is in binary form. Due to the complexity of options offered in so small a system, the symbolics appropriate to a particular distributed program are not available. The basic software systems that produce these binaries are proprietary to III.

Practical manipulation of this software even within the expanded system available at III necessitates operation with the barest minimum of included comments. For reasons of both proprietary protection as well as the impracticality of customer modification, this software is not offered with the FR 80.

In lieu of this, to permit an FR 80 user to write his own programs, III offers a set of documented FR 80 subroutines. These are the basic subroutines III uses to write special applications programs not covered by the standard software. Access to these programs is subject to execution of a satisfactory agreement protecting III's proprietary interests.

Chapter 2

HOST COMPUTER SOFTWARE

III makes available free of charge to FR 80 users the symbolics and user manuals for several of the more common host computer software systems.

On request, III will provide any FR 80 customer with the symbolics on magnetic tape and furnish a set of user manuals without charge.

These systems are the current production systems of the companies that furnish them for distribution and are essentially error free. However, neither the contributing company nor III assumes responsibility for bugs which may be uncovered in your application.

2.1 SC 4020 ROUTINES

Developed by North American Rockwell Corporation, these routines are written in COBOL with some 360 assembly language. The current version is operating under OS/MVT. Output is a tape formatted for the SC 4020. The III FR 80 reads this tape directly.

2.2 FRESCO

This is a host computer package particularly suited for use in installations where the users are familiar with the SC 4020 syntax. FRESCO was also developed by North American Rockwell Corporation and is an expansion of their SC 4020 routines package to take advantage of the additional features of the FR 80. Written in COBOL with some 360 assembly language, it is implemented under OS/MVT. The output of FRESCO is in the FR 80 data format. FRESCO is provided through the courtesy of North American Rockwell Corporation.

2.3 IGS (INTEGRATED GRAPHICS SYSTEM)

Developed by the RAND Corporation, IGS was designed to provide a universal higher level language that would produce tapes for recording on any graphics recorder. Through the courtesy of the RAND Corporation, III is able to provide a version of IGS tailored to produce META output for the SD 4060. This same package can be modified by the user to provide a more efficient META output with extended features for the FR 80.

2.4 3D PLOTS

Through the courtesy of Idaho Nuclear Corporation, III has available for distribution listings of routines for producing 3D plots. The routines are written in FORTRAN.

2.5 OTHER HOST COMPUTER SOFTWARE PACKAGES

Additional host computer software packages of general interest will be distributed to FR 80 users if the contributer will furnish III a magnetic tape with the symbolics and a copy of the user's manual.

III will also furnish technical assistance to FR 80 users who would like to convert their host computer systems to FR 80 data format output. Such conversion has resulted in significant reductions in host computer processing time, higher information density on the output magnetic tape and, in many cases, substantially increased recording speed.

Chapter 3

STANDARD FR 80 SOFTWARE

3.1 SIMULATORS

There are four standard simulator systems:

1. SC 4020 simulator
2. Meta interpreter (SD 4060)
3. Print simulators
4. CalComp simulator

For detailed discussions of these simulator systems, refer to the FR 80 Operator's Manual and the FR 80 User's Manual.

3.2 FORMS COMPILER

The FR 80 utilizes forms stored in core rather than a hardware forms flash. Refer to the FR 80 User's Manual and the FR 80 Operator's Manual for details.

3.3 FR 80 DATA FORMAT

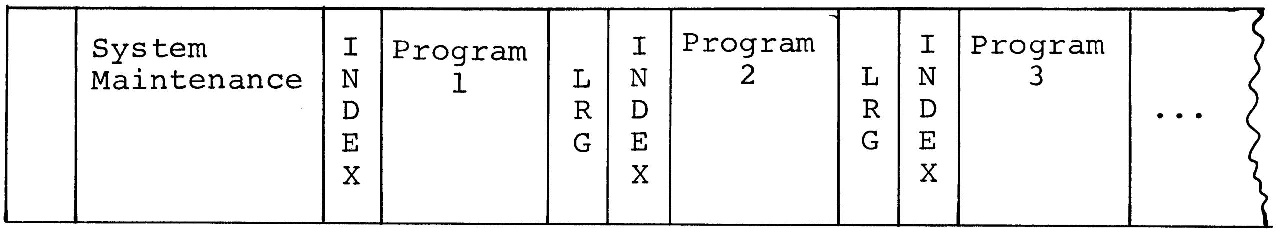
There is a preferred FR 80 format for users who wish to generate tapes on the host computer and are not restricted by format. The FR 80 data format is described in detail in the FR 80 User's Manual.

Chapter 4

FR 80 OPERATING SYSTEMS

4.1 TAPE OPERATING SYSTEM

An FR 80 with this configuration stores all the operating programs on a specially prepared magnetic tape supplied by III. The structure of the tape is shown in Figure 4-1.



LRG = Long Record Gap

Figure 4-1. FR 80 Magnetic Tape Structure.

To begin operation of the FR 80, a paper tape bootstrap is loaded in the reader of the Teletype, the address switches are set to 40_8 and the hardware read-in switch is depressed. The system tape is expected to be on Unit 3. The System Maintenance Program is loaded into the computer and started. The program responds by outputting *MONITOR to the Teletype.

The functions within the System Maintenance Program are:

1. System Tape Copy - If two tape units are executing on the FR 80, the copy function will copy the system tape onto Unit 3 from Unit 1.
2. Index - This function will search the entire tape and print out the names of each program and the magnetic tape buffer length of each program.

3. Abut - This function will produce test frames of film for precisely determining the abutment number for each camera.

With a tape operating system, it is often necessary to input all the operational parameters and forms into the program. Instead of repeating this every time the program is run, the tape system allows for a program to be dumped back onto the system tape. The dump can be with a new name or over the same program. Upon distribution of new and updated programs from III, the user can combine system tapes by utilizing this dump feature.

4.2 DISK OPERATING SYSTEM

When the FR 80 is configured with the disk option, the immediate differences are faster program setup, forms storage, and a rapid programming system. The disk utilized is a fixed-head-per-track, fast-access disk containing 262,143 18-bit words. The average access time is approximatel 16ms. The disk is structured into blocks of 256 words each.

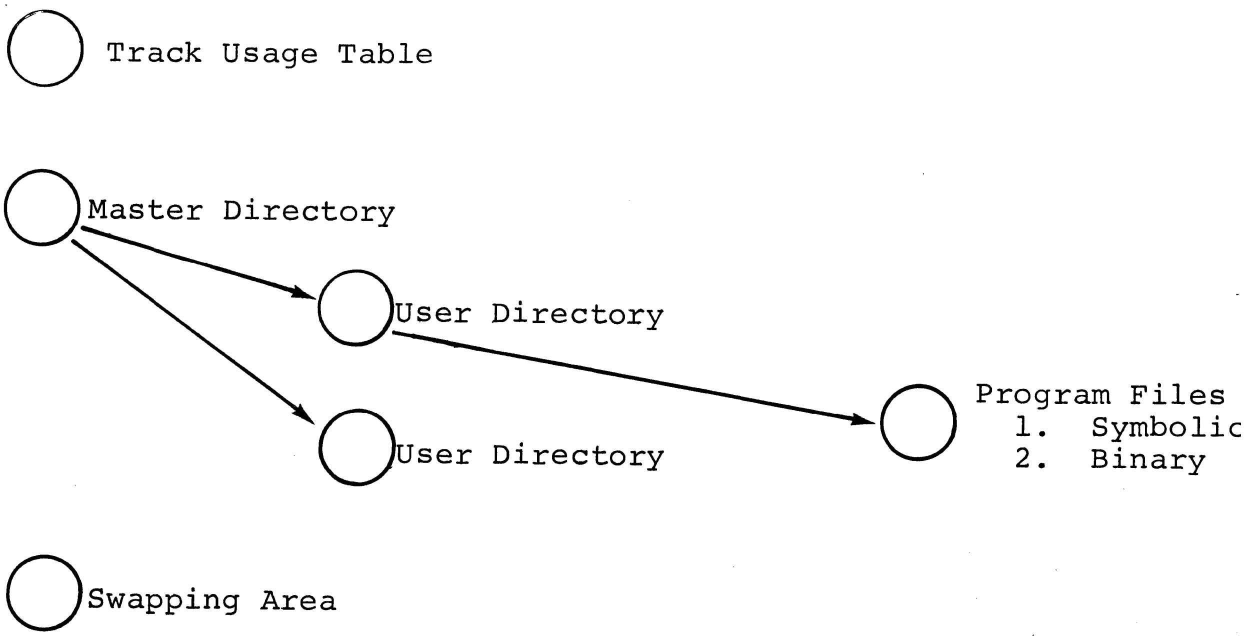


Figure 4-2. Disk Data Structure.

See Figure 4-2. The Track Usage Table (TUT) is a bit-encoded table indicating used or working block availability. The master directory contains the names and pointers to each of the user directories. The user directories give the names and pointers to each of the symbolic or binary files within that directory. The swapping area is used to retain disk debug or a core image if debug is running.

4.2.1 Debug

Debug is the basic operating system within the disk structure. Debug has the capability to:

1. Load binary programs from disk.
2. Modify the binary image.
3. File changes back on the disk.
4. Set break points for debugging binary programs.

Debug is retained in the swapping area on disk, and a program bootstrap located at -42_8 performs the swapping of the core image and debug. Once debug is running, the monitor display shows the core image that is currently in the swapping area.

Chapter 5
UTILITY PROGRAMS

5.1 TEXT EDITOR

This program is operable on both disk and tape operating systems. The only difference is the output medium. On the tape system, the output is paper tape. On the disk system, all storage is on the disk. The text editor program is used to create symbolic files of forms or new programs. After the file has been created, the text editor can be utilized to correct and update the file.

5.2 ASSEMBLER

This utility program is a powerful two-pass macro-assembler that accepts a high-level programming syntax developed by III. The assembler produces binary files on either paper tape or disk, depending on the machine configuration.

5.3 DISK UTILITY PROGRAMS

5.3.1 Tape Dump Reloader - This program loads symbolic or binary files from magnetic tape and transfers them to the disk.

5.3.2 Disk Dumper - This program dumps programs from the disk to magnetic tape.

5.3.3 Magnetic Tape Display - This program is used to display on the monitor the binary image on either 7- or 9-track tapes. The data can also be output to the Teletype.

5.3.4 Disk Audit - This program reads every block on the disk and outputs to the Teletype any hardware errors that are detected. The program also displays on the monitor the disk allocation by directories. Disk audit is the means for deleting files from the disk or renaming files.

Chapter 6
NEW APPLICATIONS

6.1 CREATING NEW SOFTWARE FOR THE FR 80

The FR 80 comes with a powerful programming system for both tape and disk operating configurations. The first step in preparing a new program is the editing and updating of the source symbolics. These symbolics consist of standard subroutines and the application package utilizing all the necessary routines to perform some specific task. The user prepares a symbolic file using the Symbolic Text Editor and has either a disk image or a paper tape of the symbolics, depending on the FR 80 configuration. This file can be modified and updated using the text editor.

Next is the assembly process. The FR 80 has a two-pass macro-assembler for processing the source file and producing a disk or paper tape binary program. At this point, the user can add this new program to the library of existing FR 80 programs either on disk or magnetic tape.

To provide the user with this programming capability, III supplies the source to produce custom application programs. These subroutines include:

1. Operating monitor
2. Vector routines
3. Character sets
4. Character routines
5. Magnetic tape routines

Given these basic routines, the user can put together an application package with a minimum of programming effort.

6.2 STANDARD SUBROUTINES

6.2.1 Standard Subroutine Parameters (III-109)

This subroutine is a set of definitions, including:

1. Machine configuration (core, tape, disk)
2. Types of cameras available
3. Special options
4. Macro definitions for standard programming conventions

6.2.2 Operating Monitor

Five files comprise the operational monitor to interface between the human operator and the applications program:

1. III-166 INVAR
2. III-166
3. III-161
4. III-161 GO
5. III-183

Included in these files are the Teletype input/output routines and basic FR 80 functions. The FR 80 functions are:

1. Beam positioning
2. Beam parameters
3. Camera advance
4. Monitor magnetic tape I/O
   a. Density select
   b. Drive select

c. Space forward

d. Space backward

e. Error retry

5. Teletype command decoding

6. Error responses

7. Image rotation

8. Focus program

9. Program load and dump

Monitor subroutines III-161 and III-161 GO are files that contain:

1. Teletype input/output formatting

2. Focus pattern for setting exposure level

3. Tape operating utility routines

   a. Program dumper

   b. Program loader

The Disk I/O file, III-183, contains routines to create, read, and write files formatted within the disk operating structure.

6.2.3 Vector Routines (III-162)

This file contains a general set of vector drawing routines. There is a routine to set the head and tail coordinates of a vector; after these are set, three different vector routines can be called.

1. Solid vector

2. Dotted vector

3. Dashed vector

6.2.4 Character Sets

6.2.4.1 Character Dispatch Tables (III-164) - This file contains dispatch tables for different character sets. These are defined using the conditional assembly features. Some of the sets available are:

1. III master set
2. BCD
3. EBCDIC
4. EBCDIC with lower case
5. CDC
6. Univac
7. Honeywell
8. G.E.
9. SC 4020
10. SC 4060
11. SC 4400

6.2.4.2 Character Descriptors (III-164 FILM) - This file contains the character descriptors for the character generator hardware. There are two other fonts available from III: the NMA Microfont (III-164 Micro) and OCR-B (III-164 OCR-B).

6.2.5 Character Routines (III-147)

This file contains routines to accept magnetic tape characters and convert them into FR 80 internal codes using III-164; the routine then performs the I/O instruction to start the character generator plotting the specified character.

6.2.6 Magnetic Tape Routines (III-163)

This file contains routines to read 7- or 9-track tape in binary or character data. The data on tape can also be accessed in bit or word formats. The file also contains routines to reposition the tape for processing nested repeats of data on tape.

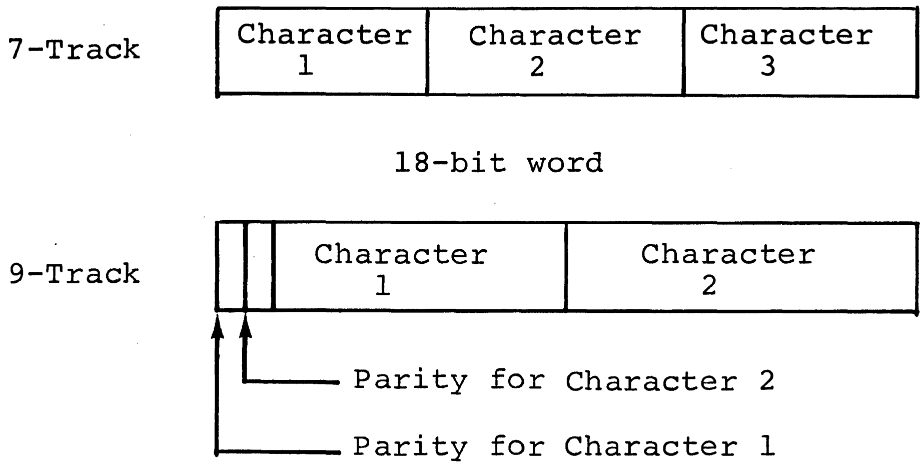


Figure 6-1. Internal Data Formats.

6.3 FR 80 COMMANDS

6.3.1 Setup Commands

RST (706002) RESET

This command stops the vector and character generators and blanks the PLS (Precision Light Source), i.e., disables intensification of the recording tube. The PDP instruction CAF will also do a RESET in addition to its other functions. This command should be done at the beginning of every program and at the detection of certain error conditions.

BLNK (706064) BLANK

Blank the PLS (Precision Light Source) but not the monitor. This command should be issued at the beginning of a program sequence which is meant to intensify only the monitor; for example, during periods of operator communication when it is not desired to make film, such as in the Text Editor or Monitor. (This command can also be used to draw dashed vectors See LDL (706164)in par. 6.3.4.

UNBL (706062) UNBLANK

Enable intensification of the PLS. All light-producing commands following this will intensify both the monitor and the PLS.

LSPS (706344) LOAD SPOT SIZE

The low-order 3 bits of the accumulator are put in the PLS spot size register (the high-order 15 bits are ignored). All subsequent points, characters, and vectors will be drawn with the corresponding spot size. This size varies linearly over a range of 1 to 5 from the smallest (0, with a maximum size of 4 scope points) to the largest (7). The monitor is not affectec by this command. A delay of at least 100 microseconds should occur after this command is given before the first plotting command will utilize the required spot size.

LBRT (706364) LOAD BRIGHTNESS

The low-order 3 bits of the accumulator are put in the PLS brightness register. All subsequent points, characters, and vectors will be drawn at the corresponding brightness. The relation of the 8 levels (0-dimmest to 7-brightest) varies greatly, depending on the setting of the PLS intensity knob. The monitor is not affected by this command. No delay is required after setting brightness.

6.3.2 Positioning Commands

The positioning of the PLS and monitor beams is accomplished through the use of four 14-bit registers: the X-buffer register, the X-DAC register, the Y-buffer register, and the Y-DAC register. (DAC means Digital-to-Analog Converter and refers to the device which converts a 14-bit number to a voltage which deflects the beam to the proper position.) When a number is put into one of the buffers, the beam is not deflected. When a number is put into one of the DACs, the beam is deflected. The proper use of the positioning commands follows under pars. 6.3.3, 6.3.4, and 6.3.5.

LXB (706204) LOAD X-BUFFER

Load the X-buffer register with the low-order 14 bits of the accumulator. (The high-order 4 bits are ignored.)

LYB (706004) LOAD Y-BUFFER

Load the Y-buffer with the low-order 14 bits of the accumulator.

LXD (706124) LOAD X-DAC

Load the X-buffer and X-DAC registers with the low-order 14 bits of the accumulator, and load the Y-DAC register from the Y-buffer register. This command deflects the beam.

LYD (706224) LOAD Y-DAC

Load the Y-buffer and Y-DAC registers with the low-order 14 bits of the accumulator, and load the X-DAC register from the X-buffer register. This command also deflects the beam.

6.3.3 Point Plotting Commands

INTS (706022) INTENSIFY SPOT

This command causes intensification at the current beam position on the monitor and, if an unblank command is currently in effect, on the PLS. The intensification lasts for 2 microseconds, and the size and intensity of the spot are determined by the most recent LSPS and LBRT commands.

LXDI (706244) LOAD X-DAC AND INTENSIFY

This command is equivalent to doing an LXD instruction followed by an INTS instruction, with the exception that an automatic delay of 35 microseconds occurs after the loading of the DACs and before the intensification. The central processor does not stop processing subsequent commands during this 35 microseconds. Hence an SPNB command should be given if any conflicting FR 80 commands are given soon after this one. (See par. 6.3.6.)

LYDI (706264) LOAD Y-DAC AND INTENSIFY

This command is the same as LXDI with an LYD substituted for the LXD.

6.3.4 Vector Generation Commands

The FR 80 must be given the following items of information in order for it to draw a vector:

Item 1. Ratio of Smaller over Larger . . this is the 12-bit fraction arrived at by dividing the smaller component by the larger. (The longest vector that can be drawn is quarter screen, 4096 points; hence, the 12 bits instead of 14.)

Item 2. The sign of the smaller component.

Item 3. A bit telling which axis (X or Y) has the larger component.

Item 4. The sign of the larger component.

Item 5. The magnitude of the larger component. (This is also 12 bits long.)

The vector will be drawn from the current beam position to the position determined by these five quantities.

LSL (706104)   LOAD SMALLER OVER LARGER

This command loads Items 1-3 as follows:

1. The ratio of the smaller over the larger is in the low-order 12 bits of the accumulator (AC6-AC17).

2. The sign of the smaller is in AC bit 5 (0 for plus, 1 for minus).

3. AC bit 4 is a 0 if X is the larger, and a 1 if Y is the larger. AC bits 0-3 are ignored.

LDL (706164)   LOAD LARGER AND GO

Items 4 and 5 are loaded as follows:

4. The larger component is in AC bits 6-17.
5. The sign of the larger is in AC bit 5.

This command also initiates drawing of the vector. The vector always appears on the PLS if it has been unblanked. (Dashed vectors are drawn on the PLS by giving alternate BLNK and UNBL commands while a vector is going.)

The vector generator requires the use of the X or Y buffer in order to achieve minimal timing and optimal end point match. This is illustrated by the following situation:

A vector is to be drawn from point (X_1, Y_1) to point (X_2, Y_2), and then another vector is to be drawn from

(X_2, Y_2) to (X_3, Y_3). After the first vector has been drawn, the DAC registers still contain the numbers X_1 and Y_1 but the beam is actually positioned at (X_2, Y_2). The difficulty arises when the DACs are updated to begin the second vector. If an LXD is given with X_2 in the accumulator, the beam will begin moving to the position (X_2, Y_1). Then, several microseconds later, when an LYD is given with Y_2 in the accumulator, the beam will have to reverse direction and move to (X_2, Y_2). This is avoided by giving an LXB with X_2 in the accumulator (this command does not cause the beam to move), followed by an LYD with Y_2 in the accumulator. This sequence will update the DACs to the proper position without the undesirable beam excursions.

6.3.5 Character Generation Commands

The character generator allows for 64 sizes, 8 rotations, variable spacing and line feed values, and a range of character coding and styling limited only by the memory capacity and resolution of the system. This versatility in coding and styling is accomplished by having the character graphic representations stored in core memory encoded in a language very much like that used by incremental pen plotters. In addition, there is a table of pointers to these representations, where the position within the table represents the octal code for the character. To display a character, one puts the pointer in location 37_8 and does a CHGO. (Location 37_8 is the fourth PDP Data Channel pointer. Location 36_8, the counter, is incremented but not tested. See the PDP manual for a description of Data Channel Operations.

LSIZ (706404)   LOAD SIZE

The low-order 6 bits of the accumulator are put in the size register. (The high-order 12 bits are ignored.) All subsequent characters are plotted at the corresponding size, from 0 (smallest) to 77_8 (largest).

LCDX (706464)   LOAD CHARACTER DELTA-X

Load the delta-X register with the low-order 14 bits of the accumulator. This number is used for spacing or carriage return, depending on the value of the rotation register (see below).

LCDY (706444)   LOAD CHARACTER DELTA-Y

Load the delta-Y register with the low-order 14 bits of the accumulator. This number is used for spacing or carriage return, depending on the value of the rotation register (see below).

LROT (706424)   LOAD ROTATION

This command loads the 5 low-order accumulator bits into the rotation register. The low-order 3 bits select one of 8 rotations, where 0 means upright and 1 through 7 are successive rotations of 45 degrees counterclockwise. The other 2 bits (AC 13 and 14) are interpreted as follows:

AC13 refers to X-delta

AC14 refers to Y-delta

a 1-bit refers to spacing
a Ø-bit refers to carriage return

Thus, if AC13 is a 1, the X-DAC will be incremented by delta-X when spacing occurs (see below). If it is a Ø, the X-DAC will be incremented by delta-X when carriage return occurs. If AC14 is a 1, the Y-DAC will be incremented by delta-Y when spacing occurs; if it is a Ø, the Y-DAC will be incremented by delta-Y when carriage return occurs. There are four possibilities:

00 -	update both X and Y for carriage return and neither for space
01 -	update X for carriage return and Y for space
10 -	update X for space and Y for carriage return
11 -	update both for space and neither for carriage return

CHGO (706324) CHARACTER GO

The character generator will display the character whose incremental codes are stored beginning in the location following the one pointed at by location 37_8. (See the PDP manual discussion of Data Channel Operations.) When the character has been displayed, the X and Y DACs (but not the buffers) will be incremented by the deltas (i.e., spacing will occur) depending on the value of the high-order 2 bits of the rotation register.

SPC (706024) SPACE

No character will be displayed but spacing will occur, i.e., the X and Y DACs will be incremented by the deltas according to the value of the 2 high-order bits of the rotation register.

CRT (706042)   CARRIAGE RETURN

This command does two things:

1. The DAC which has been spacing (see LROT) will be replaced by the contents of the corresponding buffer register. This may be either, neither, or both, depending on the contents of the rotation register.

2. The DAC which has been selected for carriage return by the LROT command will be incremented by the corresponding delta register. This may also be either, neither, or both.

6.3.6 Skip and Other Commands

SPNB (742000) for 9/L SKIP IF PLOTTING NOT BUSY
SPNB (706041) for 9/L or 15

Skip the next instruction if the character, vector, or point plotting hardware is not busy; if one of them is busy, go to the next instruction.

ADV (706304)   ADVANCE FILM

Advance the camera the number of pulldowns specified by the low-order 3 bits of the accumulator, where zero means 8 pulldowns and 1 through 7 mean 1 through 7 pulldowns.

SFNA (706061)   SKIP IF FILM NOT ADVANCING

Skip the next instruction if the film is not advancing; go to the next instruction if it is.

SM1Ø (706021)   SKIP IF MORE THAN 10 FEET

Skip the next instruction if there are more than 10 feet of film in the supply magazine; otherwise, go to the next instruction.

SFLM (706001)   SKIP IF FILM

Skip the next instruction if there is film in the supply magazine; go to the next instruction if the supply magazine is empty.

APPENDIX A

III PROGRAM LIBRARY

III-126D; DEBUG (DISK)

2. ABSTRACT

2.1 DEBUG is a general purpose "invisible" symbolic debugging program, in that the program image being dealt with is resident on the disk until swapped in. The program is allowed to use locations 0 through 17730. A bootstrap swapper above those locations exchanges the DEBUG core image with the program core image.

3. REQUIREMENTS

None.

4. USAGE

To restart, START at 17777. If that fails to work, read in DISK UNSAVE (program number III-165) at address 17735. Should that fail also, a fresh copy of DISK DEBUG may be loaded from the disk by reading in LOAD DISK DEBUG (program number III-150) at 17755.

4.1 Available Commands

(In the following, a and b stand for any legal expression and s stands for any legal symbol.)
"$" may be entered either by typing "$" (shift 4), or ALT MODE.

4.1.1 Initializing Commands

COMMAND	ACTION
$K	Kill all but permanent symbols.
$$Z	Zero memory up to upper search limit.
$$K	Kill symbols then zero memory ($$K = $K, $$Z).
a<$$Z	Zero memory from a up to upper search limit.
a<b$$Z	Zero memory from a to b inclusive.

The double $ is always required on the zeroing command to avoid inadvertent loss of information.

4.1.2 Program Loading

The following commands are used for loading or comparing standard III programs from disk or paper tape. (The $H command is useful only for making a magnetic tape system from a disk.) When there are more symbols that can be fitted into the allocated storage in DEBUG, the definitions for excess symbols will be omitted.

NOTE: In the following list, the term "name" is a binary disk file name or "0" meaning paper tape. "name" applies only to load and file commands. The file name will be assumed to be in the default directory, or, if not there, in the SYS directory. With DEBUG, the default directory name may be changed by typing a "directory-name;".

COMMAND	ACTION
name$L	Clears core, deletes symbols, and loads program with symbols.
name$$L	Does not clear core, does not delete old symbols; loads program with symbols.
name$J	Used to load a system program. Clears all of core but the first 100 cells, loads the program, and starts it at the normal starting address.
name$$J	Does not clear core, does not delete symbols for the previous program; loads program and starts it running at its normal address.
name$G	Clears core except first 100 cells, loads program without symbols, starts at its normal starting address plus 1.
name$$G	Does not clear core, does not delete symbols; loads program without symbols, and starts at its normal starting address plus 1.
name$H	Loads program with paper tape DEBUG for creation of mag tape operating program.
name$S	Does not clear core, does not delete old symbols, does not load core; loads symbols only.
name$T	Compare core with binary program file in the range between the lower search limit and the upper search limit. Differences between them (applying the search mask) will be typed out, unless the word in the program file is a 0 or XX.

4.1.3 Program Deletion

name$$D will delete a binary disk file.

4.1.4 Storing Programs on Disk or Punching Programs Onto Paper Tape

Programs may be stored on the disk or punched onto paper tape by means of the following command:

```
name$F
```

As before, if "name" is "0", the program will be punched on paper tape.

If "name" already exists on the disk, the program will be replaced; otherwise a new program will be created. A ? indicates that the disk is full, or the default directory is full, and the program was not saved on the disk.

4.1.5 Mode Control

COMMAND	ACTION
$S	Print words as symbolic commands.
$C	Print words as constants.
$R	Print addresses in relative symbolic.
$0	Print addresses as constants.
n$R	Change output number base to n (n>1).
n.	Interpret n as a decimal number.
=	Print last quantity in current mode.
:	Print last quantity in opposite mode.

The initial modes are S and R, and the initial output base is 8. (The input base is always either octal or decimal, depending on the absence or presence of the decimal point.)

In symbolic mode the listing is made to look as much as possible like input to the assembler. Labels are printed when there is an exact match (in which case they are followed by a comma), or when the location counter increases by more than 1 (in which case the label is followed by a /).

In patch mode, addresses are always printed and contents are never printed. (Patch mode is entered with left parenthesis and exited with slash.) Initially, DEBUG is in non-patch mode.

4.1.6 Arithmetic

SYMBOL	DEFINITION
+ or space	Plus
-	Minus
*	Times
%	Divide
!	Inclusive OR

4.1.7 Examination

COMMAND	ACTION
a/	Exit patch mode and open a in current mode.
a\	Open a in opposite mode.
/	Exit patch mode and open addressed register in current mode.
\	Open addressed register in opposite mode.
carriage return	Modify, close, and exit patch mode.
line feed	Modify and open next.
↑	Modify and open previous.
)	Modify and open addressed.
TAB	Modify, open addressed, and change sequence.
a(	Enter patch mode and open a.
(	Enter patch mode and open addressed register.

4.1.8 Registers

REGISTER	
$A	Accumulator
$L	Link
$Q	MQ
$F	Bottom of symbol table
$J	Location at which program starts on '
$J+1	Location at which program starts on "
$J+2	Contents of shift counter
$J+3	Status of extend mode
$M	Mask for searches
$M+1	Lower search limit

REGISTER	
$M+2	Upper search limit
.	Current location
#	Value of last quantity typed

All of the above are input symbols only - they do not print out as addresses.

4.1.9 Running

COMMAND	ACTION
'	Go to location contained in $J
a'	Go to a
"	Go to location contained in $J+1
a"	Go to a and put a in $J+1

$J and $J+1 initially contain halts. If ' or " are used without arguments before changing $J or $J+1, an error is indicated.

When a program is loaded, its start or pause address is put in $J.

A running program may JMS to 17776 to store into $J+1 so that a " (SHIFT 2) will return control to that location with the status restored.

NOTE: If it is desired that DEBUG type the return address for such an entry, the JMS 17776 should be preceded by DZM 17775.

COMMAND	ACTION
$B	Remove breakpoint
a$B	Put breakpoint at a
	The interrupt status is saved and restored at breakpoints. Interrupt Status: Contents of AC, contents of MQ, link, shift counter, and PIE.
$P	Proceed after breakpoint stop.
	If not in DEBUG as a result of a breakpoint, then $P will act as a " and proceed to the address stored in $J+1. WARNING: An attempt to proceed from a breakpoint that is replacing a CAL will not operate properly.
n$P	Proceed and break on the n'th time
a$X	Execute the command a
	If the execution of the command results in a skip, the bell will ring.

4.1.10 Symbol Definition

COMMAND	ACTION
a<s>	Define the symbol s as the quantity a.
s,	Define the open location as s. If there is no open location, an error is indicated.

4.1.11 Search and Replacement

COMMAND	ACTION
a$W	Search for and print all locations equal to a.
a$N	Search for locations not equal to a.
a$A	Search for locations with address a.
a$E	Search for locations with effective address a.
a←	Repeat last search and replace masked bits with a.

Searches may be stopped by typing anything. On replacement, the ALT MODE key stops the printing, but continues the replacement; any other key stops both the replacement and the printing.

If two replacements are attempted without an intervening search, or if the search was an E-type, an error will be indicated.

Searches and replacements are masked by ($M). Replacement after an A search will be address only unless the replacement word contains bits outside the address part, in which case it will be a word replacement masked by ($M).

A search begins at ($M+1) unless the command was preceded by b<, in which case it starts at b, as does the ensuing replacement, if any. Searching ends at ($M+2). At the end of either a search or a replacement, . is equal to the address of the last match. If no argument is given (e.g., $N), zero is assumed.

4.1.12 Program Calls to DEBUG

DEBUG may be entered from a program so as to load and start another program, or to load a program with symbols.

To call for a load, JMP 17776 after setting 17775 to one of the following:

> A 13-bit ADDRESS pointing to the directory name and file name of the program to be loaded. That program will be loaded with symbols as if "name$L" had been typed.
>
> A file name first-word; if the file name is more than three characters, I-4 must contain a file name second-word. That program will be loaded without symbols and started at its normal starting address plus 1, as if "name$G" had been typed.

4.2 Display Feature and Operating Instructions

Figure 1 shows a typical display in Symbolic mode; that is, with $S in effect.* The top line is a display of the various machine registers. From left to right, we see the LINK (0 in this case), the symbolic contents of the ACCUMULATOR, the symbolic contents of the MQ, and -- in this case -- the contents of $J+1. (If the content of the upper right field is preceded by a right arrow,→, it indicates that the program is interrupted by a breakpoint, and that the breakpoint is at the location indicated by the entry following the →.)

The second line is the absolute address of the first location displayed. Each subsequent line of the display is one location higher in memory. (The second line may be absent if there is no reasonable symbolic equivalent for that address, in which case the number will appear on the following line followed by the /.)

The third line indicates the symbolic address of the first location displayed, and to the right of it is the content of that location.

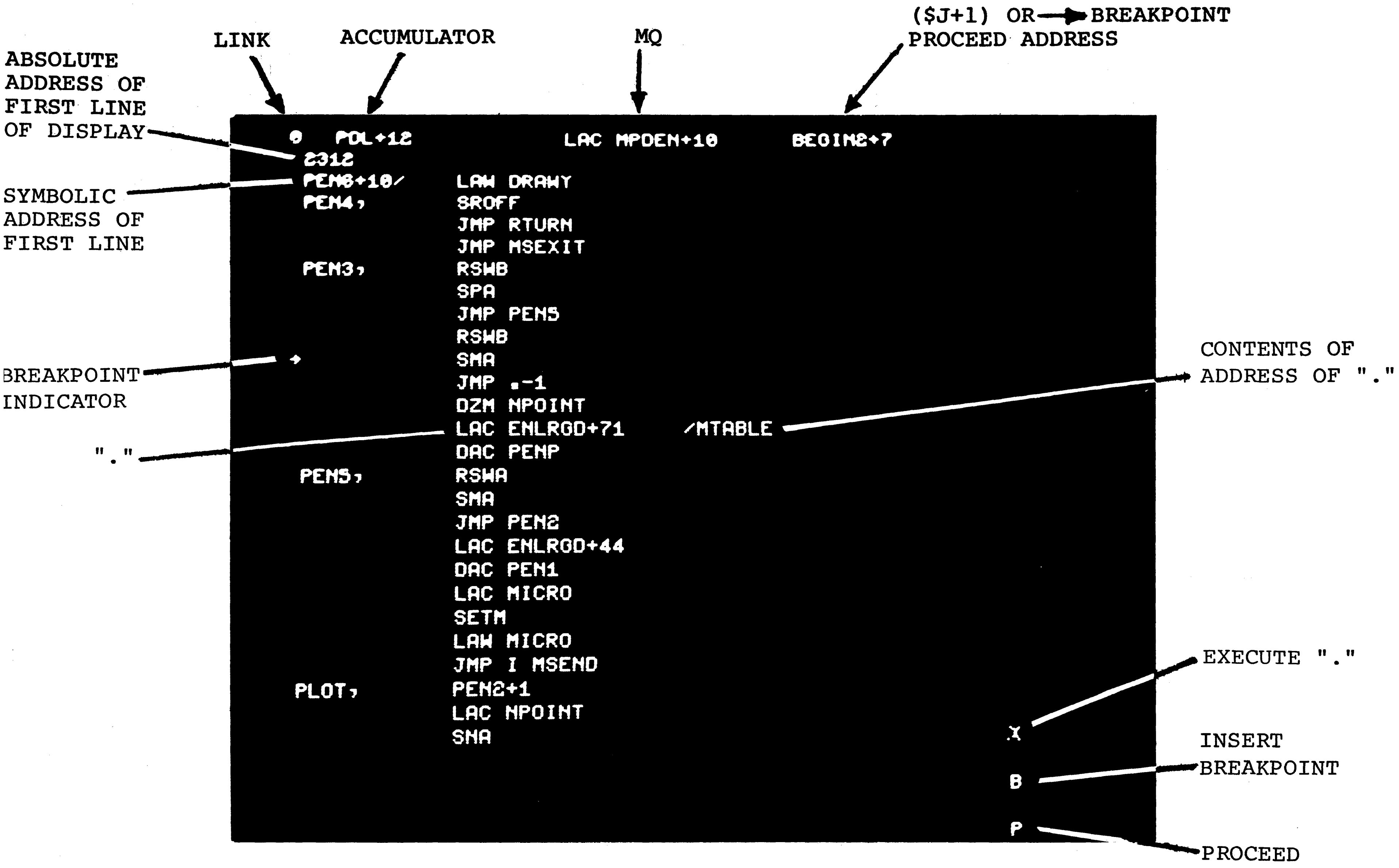
The fourth line, PEN4, indicates that there is a symbolic tag corresponding to the location whose contents are being displayed on that line.

The right arrow at the left side of the screen indicates where a breakpoint is set.

If the open register "." is on the screen, it is indicated by having the contents of its address, preceded by a /, to the right of its own contents. (The presence of this information in the middle of the screen tells the user where the current location counter for DEBUG is.)

For installations having a light pen, the three characters in the lower right corner of the screen will appear for use in conjunction with the light pen. The three characters have the following significance: Pointing at the X will execute the contents of the currently open register. Pointing at the B will cause a breakpoint to be placed at the currently open register. Pointing at the P will cause the program to act as though a $P were typed. When any characters are

* Since the symbol table for the display features is derived from the normal DEBUG symbol table, storage limitations in DEBUG may prevent all symbols from being available for display in large programs.



```
 0   POL+12                 LAC MPDEN+10      BEGIN2+7
 2312
 PEN6+10/       LAW DRAWY
 PEN4,          SROFF
                JMP RTURN
                JMP MSEXIT
 PEN3,          RSWB
                SPA
                JMP PEN5
                RSWB
 →              SMA
                JMP .-1
                DZM NPOINT
                LAC ENLRGD+71      /MTABLE
                DAC PENP
 PEN5,          RSWA
                SMA
                JMP PEN2
                LAC ENLRGD+44
                DAC PEN1
                LAC MICRO
                SETM
                LAW MICRO
                JMP I MSEND
 PLOT,          PEN2+1
                LAC NPOINT
                SNA
                                                                X

                                                                B

                                                                P
```

Figure 1. Typical Display in Symbolic Mode.

typed after carriage return, these three characters will be extinguished before any command is completed, and the light pen feature will be disabled to avoid ambiguity of action.

Use of the light pen doesn't open any registers. It just changes the value of ".". In fact, if the pen points to a cell when another register is open, the action of the pen will close that register without modification and change". ". to the new value.

Other things the light pen may do:

1. Pointing with the light pen to one of the displayed cells on the screen will cause that to become the open register, and it will be flagged by the presence of the / followed by the contents of its address in the middle of the screen.

2. Pointing at the value in the middle of the screen (the contents of the addressed register) will perform the same function as a tab typed on the Teletype, and will open or change "." to that location.

3. Similarly, one may point to the address in the upper right corner to see the code in the neighborhood where the program will return. [Whenever a cell is selected by some means other than the pen, the selected cell will become the eighth line on the screen, so that the cells above and below may be served with it. However, if the selected cell was gotten to by a simulated tab from the eighth line on the screen, it will become the first line on the screen (to prevent infinite tabbing).]

4. Pointing at the AC will effect a tab to that location.

5. Pointing at the right arrow at the left side of the screen will remove a breakpoint.

If your machine has parameter knobs, the display may be moved to adjacent cells by turning the right A parameter knob. For machines without parameter knobs, set data switches 16 and 17 to perform the same function. Switch 17 down moves slowly to successively higher core locations, and switch 16 down moves slowly to successively lower core locations.

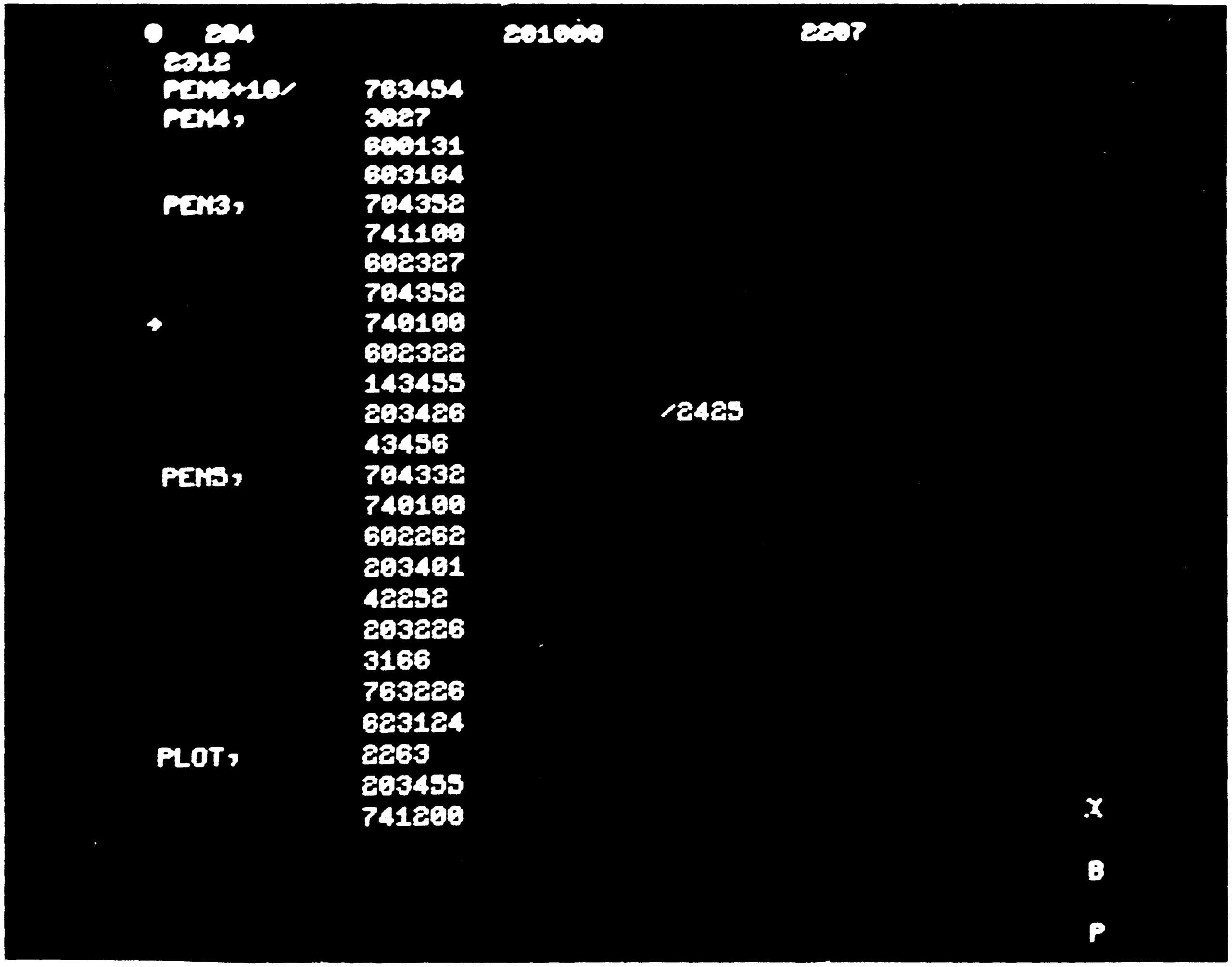
To change the open register without modification, turn the left A parameter knob, or turn on data switches 14 or 15.

The Display Feature is disabled by moving sense switch 1 down (which corresponds to setting AC switch 12 to 1 on machines without sense switches).

Holding the left foot pedal down suppresses all Teletype output from DEBUG. (On machines without foot pedals, the left pedal is simulated by AC switch 3.)

The left foot pedal also changes the display from current mode to the opposite mode, i.e., symbolic to octal or vice versa. Figure 2 shows the sample display in Octal mode. The radix for all display items is 8, regardless of the Teletype radix setting.

In essence, the right foot pedal undoes tabbing operations. Toggling the right foot pedal will switch the display back to the last open cell not in view of the display. [Any operation that opens a cell not on the screen will store its value in a table of previous cells. Toggling the right foot pedal will take you to the previous entry in that table. (The table has 8 entries treated cyclically.)]



```
●  204                      201000         2207
 2012
 PEN6+10/     763454
 PEN4,        3027
              600131
              603164
 PEN3,        704352
              741100
              602327
              704352
+             740100
              602322
              143455
              203426                /2425
              43456
 PEN5,        704332
              740100
              602262
              203401
              42252
              203226
              3166
              763226
              623124
 PLOT,        2263
              203455
              741200                                X

                                                    B

                                                    P
```

Figure 2. Sample Display in Octal Mode.

COMMAND SUMMARY - DEBUG

$K	Kill Symbols
$$Z	Zero memory
$$K	$K, $$Z
a<$$Z	Zero above a
a<b$$Z	Zero a to b
name$L	Clear & load w/symbols
name$$L	Load with symbols
name$J	Clear, load without symbols, and start
name$$J	Load without symbols, and start.
name$G	Clear, load without symbols & start at ($J)+1
name$$G	Load without symbols, and start at ($J)+1
name$H	Clear, load w/paper tape DEBUG and symbols
name$T	Compare program with binary file
name$S	Load symbols only
name$$D	Delete binary program
name$F	Save core as program
$S	Symbolic
$C	Constant
$R	Relative
$O	Octal
n$R	Set base n
n.	n is decimal
=	Equals, current mode
:	Equals, opposite mode
+	Plus
-	Minus
*	Times
%	Divide
!	Or
/	Open, current mode
\	Open, opposite mode
c.r.	Close
l.f.	Open next
↑	Open previous
)	Open addressed
TAB	New sequence, addressed
(	Open, patch mode

$A	Accumulator
$L	Link
$Q	MO
$F	Symbol table
$J	Start address
$J+1	Return address
$M	Search mask
$M+1	Lower limit
$M+2	Upper limit
.	Current address
#	Last value typed
'	Start
"	Return
$B	Breakpoint
$P	Proceed
$X	Execute
a<s>	Define a as s
s,	Define current as s
$W	Word search
$N	Not-word search
$A	Address search
$E	Effective address search
←	Replace

EXTENDED MEMORY FEATURES IN DEBUG

For machines with extended memory (more than 8K), there is one additional command that relates to the mode in which addresses are accepted. This command determines the masking to be used for address interpretation.

The "at" sign, @, when used without argument, means that the addresses will be interpreted as the hardware interprets indirect addresses, i.e., 14 bits on a 16K machine, and 15 bits on a larger machine. The display will wrap from bank to bank when a core bank boundary is within the field of the display.

Typing @ preceded by an argument which is the core bank number, will select that core bank and use only a 13-bit addressing field. The display will correspondingly wrap from beginning to end of that bank if that area is on the field of the screen.

In addition to being able to type this command, there will be displayed in the lower right corner of the screen a 0, 1, 2, and so on through the number of core banks available, followed by an @.

Pointing at the @ will extinguish it and select the total core mode as though an @ were typed. Pointing at one of the numbers will extinguish the number and select that bank as the only addressable area.

<u>APPENDIX B</u>

III PROGRAM LIBRARY

III-125; SYMBOLIC TEXT EDITOR

CONTENTS

2. ABSTRACT

2.1 This program provides for the creation or modification of symbolic text files. Input or output may be paper tape, disk, or teletype.

3. REQUIREMENTS

None.

4. USAGE

4.1 Introduction

When debugging, perhaps the most tedious job is that of making a corrected symbolic program. It involves seemingly endless duplication, listing, splicing, and close attention. The Symbolic Text Editor lightens the task and speeds the process of correcting programs by using the computer to perform the drudgery.

In brief, the Editor reads a section of symbolic text into core memory, where it is available for examination and correction. The corrected text can then be output to a new file. Text may also be entered directly from the keyboard for original text preparation. The Editor accepts tape input and provides output in ASCII* code. Keyboard communication with the Editor may be accomplished on the Teletype Model 33KSR. The information to be corrected is stored in a text buffer, which occupies all of memory not taken up by the Editor itself.

* The code herein referred to as ASCII is actually #33 code or ASCII with the eighth bit punched on tape.

4.1.1 How the Editor Works

By convention, paper tape information is organized into various-sized blocks. The larger blocks are called pages and are separated by form feed codes on paper tape. Each page is divided into smaller blocks called lines, which are separated by carriage return-line feed pairs. A terminating carriage return-line feed pair is part of the line that precedes it.

In the text buffer, lines are implicitly numbered decimally in sequence beginning with 1. Form feeds are not stored in memory; hence there are no page divisions in the text buffer, and the entire contents of the Editor's buffer are treated as a single page. The user may organize his output into pages if he wishes.

4.1.1.1 Operating Modes

In order to distinguish between commands to itself and text to be entered into the buffer, the Editor operates in one of three modes. In Command mode, typed input is interpreted as directions to the Editor to perform some operation. In Text mode, all typed input is taken as text to be inserted in or appended to the contents of the text buffer. The Character mode allows each character of a designated line to be referenced.

4.1.1.2 Commands

A command directs the Editor to perform some operation. A command consists of a single letter preceded by zero, one, or two arguments. If we let E represent any command letter, the three ways of constructing a command are:

No arguments	E↲*
One argument	nE↲
Two arguments	m,nE↲

* The nonprinting "carriage return" and "tab" characters will be represented hereafter by ↲ and ⇥ respectively.

Note that two arguments must be separated by a comma, but that no comma separation is allowed between the argument(s) and the command.

To be executed, a command must be followed by a carriage return. This is the signal to the Editor to process the information just typed. The Editor responds with a line feed as soon as it has processed the command and begun the operation. If a mistake is made while typing a command, the entire line may be ignored by typing Line Feed.

4.1.1.3 Arguments

An argument may be any decimal integer, special symbol, or arithmetic expression, consisting of decimal integers or special symbols separated by the addition (space) or subtraction (minus sign) operators.

Examples:

```
      256
      /-39
      25-3
   12 48-7
      . 4
```

The following two characters are special symbols in the Editor.

The (/) symbol represents the total number of lines in the text buffer. If there are 100 lines in the text buffer, the symbol / has a value of 100. It is used primarily for referencing the last line in the buffer.

Example:

/L↵ This will cause the last line in the text buffer to be listed

The (.) symbol has a value equal to the number of the last line involved in an Editor operation. The value of . will be equal to the number of the last line referenced or typed in. In each command description that follows, the value of . after the completion of the operation is stated.

Example:

23,.C↵ Lines 23 to the current line are changed.

4.1.1.4 Special Functions

Certain keys have special operating functions. The first three below are nonprinting, so the symbol in parentheses is used to indicate the operation of the associated key.

Carriage Return (↵)

In both Command and Text modes, this is the signal for the Editor to process the information just typed. In Command mode, the operation specified is to be performed. In Text mode, it means that the preceding line of text is to be placed in the buffer. In Character mode, operation of this key causes the line to be stored back in the buffer with the corrections made, and an exit from Character mode back to the Editor's Command mode.

Line Feed (↓)

This character has two meanings depending on when it is used. If it is struck after some information has been typed, it causes that information to be deleted. Used thus in Command or Text mode, it has the effect of erasing mistakes. When it has processed the line feed, the Editor responds with a carriage return. In Character mode, it causes the original line to be stored back in the buffer with no corrections.

Rub Out (RO)

Typing RO in Command mode will cause the next line of text to be printed.

Pressing RO in Text mode will cause the last character of an incomplete line of text to be deleted from the input buffer. Continued striking of this key will cause successive characters to be deleted one by one, working from the end of the line back to the beginning. In this way, a mistake can be corrected without having to retype the whole line.

Example: Instead of DAC PTEM, the following line was typed:

DAC CTE

To correct the line, RO is struck three times, erasing the last three letters in succession, E, T, and C. The correct text is then typed in.

In Text mode, the rub out key has another function. Typed immediately after a carriage return, it signals the Editor to return to Command mode. If one deletes all the characters in an incomplete line and then strikes RO one more time, the Editor will also return to Command mode.

In Character mode, RO allows the user to move the pointer to the left, and if in Character Insert mode, to delete that character.

Colon (:) When this symbol is typed in Command mode, the Editor will print the decimal value of the argument that precedes it followed by a carriage return. It is frequently used for determining the number of lines of text in the buffer.

Example: /: 57

or in determining the number of the current line:

Example: .: 32

4.1.2 The Editor's Command Repertoire

4.1.2.1 Command Mode Errors

If a command requires one or two arguments, they must be provided. If an argument is missing, the Editor types the error message, ARG MISSING.

The Editor types ? if:

1. A command is given too many arguments.

2. An argument is zero, negative, or greater than /.

3. The second argument is less than the first (except for the "X" command).

4. The command is illegal.

5. No match could be found when searching.

6. Attempting to get next page of input when no more input exists.

The Editor types BUFFER ALMOST FULL if there are less than 100 (octal) free words in memory.

The Editor types BUFFER FULL if the buffer is full. No more text will be accepted.

If the keyboard input buffer becomes full while the Editor is typing, all further input is lost, and the bell is rung once for each character.

The commands are grouped under four headings:

1. Input Commands

2. Editing Commands

3. Output Punching Commands

4. String Search Commands

4.1.2.2 Input Commands

In these commands, the form feed and the physical end of the input tape are both end-of-page indicators. If an end-of-tape is encountered before the completion of a command, the operation is terminated.

NOTE: Commands followed by an * are included to allow the user to know what the commands will do with all possible combinations of arguments. They are not recommended for general use. All combinations not listed are illegal.

COMMAND		ACTION
R		Read one page of text and append to buffer (.=1). The Editor will read information from the input file until a form feed or the physical end of input is encountered. The incoming text is appended to the contents of the buffer; no information in the buffer is lost. The form feed is not entered into the buffer.
nR		Read n lines (not pages) of text and append to buffer (.=1). The Editor will read n lines and append them to the contents of the buffer. Reading will cease if a form feed or the end of tape is encountered before n lines have been read.
m,nR	*	Equivalent to R.
m,nX		Read the next m lines of text from the input tape and insert them after line n. (.= last line entered).
Z		Skip one page of text. The input tape will be moved forward until a form feed is encountered. The contents of the buffer are unaffected.
nZ		Skip n pages of text. The contents of the text buffer are unaffected.
m,nZ	*	Equivalent to Z.

4.1.2.3 Editing Commands

The following commands permit the alteration of text in the Editor's buffer:

COMMAND		ACTION
D		Equivalent to .D.
nD		Delete line n. Line n is removed from the text buffer. The numbers of all lines following it are reduced by one, as is the line count. (.=n or /, whichever is smaller.)
m,nD		Delete lines m through n, inclusive. The line following line n becomes the new line m. (.=m or /, whichever is smaller.)
A		Enter Text mode and append to buffer. The Editor enters Text mode upon processing this command, and the user may then type in any number of lines of text. These will be appended to the end of the text in the buffer or placed into a previously empty buffer. The line count is increased accordingly. This command may be given with an empty buffer to enter text. (.=1)
m,nA	*	Equivalent to A.
I		Equivalent to .I.
nI		Insert text before line n. The Editor enters Text mode to accept input. No information is lost. The first line typed becomes the new line n. The numbers of all lines following the insertion, as well as the line count, are increased by the number of lines inserted. (.= last line entered.)
m,nI	*	Equivalent to nI.
C		Equivalent to .C.

COMMAND		ACTION
nC		Change line n. Line n is deleted, and the Editor enters Text mode to accept input. The user may now insert as many lines of text as he wishes in place of the deleted line. If more than one line is inserted, subsequent lines will be renumbered. (.= last line entered.)
m,nC		Change lines m through n. These lines (inclusive) are deleted. The user may insert any number of lines (or none at all) in their place. (.= number of last line typed in.)
J		Equivalent to .+1J.
nJ		Open line n to ↵ or second tab and enter Text mode for end of line correction or comment insertion. Any comment already in line is lost. When ↵ is typed at the end of the line, the Editor returns to Control mode. NOTE: Line Feed deletes entire line and returns to Control mode. (.=n)
m,nJ	*	Equivalent to nJ.
K		Kill the buffer. The contents of the buffer are completely erased. The values of / and . are set to zero. (. and / = 0)

The following commands will cause the printout of all or any part of the contents of the text buffer. Printing may be stopped at any time by the use of AC switch 0. Normally, this switch is down. If it is turned on then off at any time during a printout, the operation will stop immediately. The line being printed is unaffected in the buffer. If this occurs in the middle of a line, the user must type a ↵ to restore the

Editor to normal Command mode operation. Execution of any of the following commands may be halted in this manner. The contents of the text buffer are unaffected by the following operations unless specified.

COMMAND		ACTION
L		Equivalent to .L.
nL		Print line n. This line will be typed out, followed by a ↵ and a Line Feed. (.=n whether printing is stopped or not.)
m,nL		Print lines m through n, inclusive. (.=n whether printing is stopped or not.)
W		Write entire buffer. This causes the Editor to print the entire text contained in the buffer. The buffer remains intact. (.=/)
nW		Write n pages. The text buffer is cleared, and the Editor reads n pages from tape, printing each one on a separate page, spacing accross page perforations automatically. This command is equivalent to executing K, followed by the sequence, R, W, K, performed n times or until a physical end of tape is encountered. The buffer will be empty upon completion of this command.
m,nW	*	Equivalent to W. (.=/=0)
H		Same as for W except line numbering is forced.
nH		Same as for W except line numbering is forced.
m,nH	*	Same as for W except line numbering is forced.
Q		Uncommented print. The Editor will print the entire contents of the buffer, suppressing all text on each line after the second tabulation. Normally, this has the effect of suppressing comments in the program. (.=/ after execution)

COMMAND		ACTION
nQ		Print line n uncommented. Line n will be printed up to the second tabulation; all information following this is not printed. (.=n after execution)
m,nQ		Print lines m through n up to second tabulation. (.=n after execution)
G		Equivalent to .G.
nG		Get the first line after line n which contains a tag. The Editor will scan the text beginning with line n until it encounters a line beginning with a character other than ↵, ↓, ⇥, or/. This line is printed. (.= number of the line printed)
m,nG	*	Equivalent to nG.
B		Back up and print. Line .-1 will be printed. (.=.-1 after execution)
E[1]		Read entire input tape and print the first line of each page.
U		Equivalent to 1U.
nU		Move display pointer up n lines. This command supplies its own ↵, if in half duplex. (.= .-n or 1 if current line < n.) Pointer will not move past line 1.
m,nU	*	Equivalent to nU.
〈alt mode〉		Equivalent to 1 〈alt mode〉.
n 〈alt mode〉		Move display pointer down n lines. This command supplies its own ↵ if in half duplex. The pointer will not move past line /. [.=.+n or / if (/ - current line) < n]

[1] The E command applies to Editor assembled for paper tape only.

COMMAND		ACTION
M		Equivalent to .+1M. (.= line found)
nM		String search within buffer. (.= line found) (See paragraph 4.1.2.5, page 16.)
m,nM	*	Equivalent to nM.
Y		String search entire input. (See paragraph 4.1.2.5, page 15.)
V		Turn on line numbering.
nV		Turn off line numbering. (1V is recommended)
m,nV	*	Turn off line numbering.

4.1.2.4 Output-Punching Commands

The following commands provide for the output of corrected text or for the duplication of pages of the input tape. Punching can be halted by the use of AC switch 0. In this case, however, the switch must be kept up until processing of the punch command is finished.

COMMAND	ACTION
P	Punch the entire contents of the text buffer. This is preceded by a short length of tape feed. The contents of the text buffer are unaffected by this command.
nP	Punch line n. This is preceded by a short length of tape feed. (.=n after execution)
m,nP	Punch lines m through n, inclusive. (.=n after execution)

COMMAND		ACTION
S		Punch a form feed. This is preceded and followed by a short length of tape feed.
nS	*	Equivalent to S.
m,nS	*	Equivalent to S.
F		Equivalent to 1F. (Ignored for disk output.)
nF		Punch n lines of blank tape. (Ignored for disk output.)
m,nF	*	Equivalent to 1F. (Ignored for disk output.)
O		Punch one page. The Editor will punch the contents of the text buffer followed by a form feed, and then clear the buffer. This command is equivalent to the sequence: P, S, K.
N		Punch, then read the next page. The Editor will punch the contents of the buffer and a form feed, clear the buffer, and read the next page of tape into the buffer. This command is equivalent to the sequence: O, R.
nN		Punch, duplicate, and read. The Editor will punch the contents of the buffer, punch a form feed, clear the buffer, duplicate n-1 pages of tape, and then read the nth page into the text buffer. This command is equivalent to the sequence: P, S, (n-1)T, R.
m,nN	*	Equivalent to N.
T		Equivalent to 1T.
nT		Tape duplication. The Editor will clear the buffer, then read and punch n pages of tape. The buffer is empty upon completion of this command, which is equivalent to the sequence: K, n(R, P, S, K). If, within the range of this command, the Editor encounters two form feeds with nothing more than tape feed between them, only one will be punched. However, the Editor will count the space between them as a separate page in reading the input tape.

COMMAND		ACTION
m,nT	*	Equivalent to 1T.

4.1.2.5 String Search Commands

The string search commands enable the user to search either the buffer or the input tape for a specified string of characters. The string may consist of any characters except RO, ↵, and ↓. The string will be found if it is surrounded by delimiters. The delimiters are ⟨space⟩ ⟨tab⟩ ↵ - (, / = # ! $) + and the first character of a page is assumed to have a delimiter in front of it although there really is not. Thus, if the string is JMP ABC, the search will find ⟨tab⟩ JMP ABC↵, but not ⟨tab⟩ JMP ABCD↵. After typing the command, the string to be searched for is entered on the next line (following the ↵). If a mistake is made, the string can be deleted with a ↓ and re-entered. RO negates the search command, and returns to Command mode. Also, A#BC is not the same as #ABC. The string is terminated by a ↵, and the search begins immediately. If nothing is typed but a ↵, the last string entered will be used. If no string has ever been entered, or it has been deleted by ↓↵, a ? is typed. The maximum size of the string is 16 (octal) characters, after which the Editor types ↵ and continues.

If the command is Y, the entire input is searched, starting with the next line of input. When a page boundary is encountered, the buffer is output, and replaced by the following page. (If there is no more input, a ? is typed, and control is returned to Command mode.) Searching continues from the first line of the new page. When a match is found, . is equal to the line in which the match occurs. A ? is typed at the end of input. This command is useful for locating typing errors revealed by the assembler, and for finding references to a given variable which is being changed.

If the command is nM, the contents of the buffer are searched starting at line n. The pointer is moved to the first occurrence of the string or the end of the buffer. To continue the search with the next line, type M⤸⤸. If the Editor reaches the end of the buffer without finding the string, a ? is typed.

4.2 Character Mode

All of the editing operations described previously have a line as the smallest unit of text that can be referenced. The command described below puts the Editor into Character mode, allowing each character of the designated line to be referenced.

4.2.1 The Command

;	Equivalent to .;
n;	Enter Character mode to edit line n.
m,n;	Equivalent to n;

4.2.2 The Pointer

When the Editor enters Character mode, only the designated line is displayed, preceded by a right arrow. The Character mode pointer is an up-arrow (except in Overwrite mode) which is positioned either under the right arrow, or one of the characters. The pointer is allowed to be under the arrow for inserting before the first character of the line. If, while in this position, any other action is requested, the pointer is moved under the first character before taking the desired action.

4.2.3 Character Mode Instructions

When in Character mode, instructions to its control routine are not followed by a ↵. Each one is executed as soon as it is typed. If half duplexing, this means that the characters will run across the page.

COMMAND	ACTION
Carriage Return (↵)	Causes the line to be stored back in the buffer with the corrections made and an exit from Character mode back to the Editor's Command mode.
Line Feed (↓)	Line Feed causes the original line to be stored back in the buffer with no corrections.
⟨space⟩	Equivalent to 1⟨space⟩.
n⟨space⟩	Moves the pointer n characters to the right. The pointer will not move past the last character of the line.
Rub Out (RO)	Equivalent to 1RO.
nRO	Moves the pointer n characters to the left. The pointer will not move past the right arrow.
C	Equivalent to 1C.
nC	Deletes n characters as nD would and enters insert mode to replace those characters.
D	Equivalent to 1D.
nD	Deletes n characters starting with the character pointed at and going to the right. If there are less than n characters there, the rest of the line is deleted. After deletion, the pointer is under the character which was to the right of the last character deleted. If there is no such character, the pointer is under the last remaining character of the line.
I	Enter Insert mode. A space appears to the right of the character pointed at, and the pointer moves up part way into the space. Each character typed thereafter, except control characters, is inserted into the line above the pointer and the pointer (and blank) moved to the right of the new character. If the pointer was pointing at the last character of the line, the characters are appended onto the back of the line.

COMMAND	ACTION
O	Enter Overwrite mode. The pointer changes from an up-arrow to an underbar. Each character typed thereafter, except control characters, is substituted for the character above the pointer, and the pointer moved to the next character to the right. If there are no more characters to the right, the Editor automatically enters Insert mode.

All other characters are ignored, and a bell is rung.

4.2.4 Control Characters for Insert and Overwrite Modes

↓	Same as for the Character mode control routine.
↲	Same as for the Character mode control routine.
⟨alt mode⟩	Exit from Overwrite or Insert mode and return to the Character mode control routine.
RO	Allows the user to change the character to the left of the pointer if he made a mistake in typing. In Overwrite mode, the pointer is moved one character to the left. In Insert mode, the character to the left is deleted and the pointer moved. If the leftmost character of the line is being pointed at, the control character is ignored.

All other characters are entered into the line as text.

4.2.5 Error Messages

When in Character mode, the line being edited is unpacked in the area above the used portion of the text buffer. If the buffer is almost full, the messages "BUFFER ALMOST FULL" or "BUFFER FULL" may be typed when entering Character mode, or while inserting text. The first message is a warning and does not affect the buffer. The second message causes an immediate exit from Character mode, eliminating any changes made in the line.

4.3 Disk Editor Commands

The following commands are available when the Editor is part of a disk operating system. The commands typed-in are always echoed on the teleprinter, even when in full duplex mode.

Those commands not requiring a file-name do not require a ↲. If a file-name is called for, that name is as specified in the Disk Operating System description. If the file-name is not typed before the ↲, the last name typed will be used (if no name has been typed, the default file-name from the disk will be used.)

Any command needing a file-name may be aborted by typing ↓ or RO before the ↲ is typed.

Any command naming the output file will create a file by that name on the disk if none previously existed, or replace a previously existing file.

COMMAND	ACTION
EB	Initialize input from paper tape instead of disk.
EE	Resume input from disk.
EP	Initialize output to paper tape punch.
ES	Initialize or resume output to disk.
ER filename↲	Initialize reading from disk file, and read first page. Select disk as output device.
EI filename↲	Take subsequent input from file specified. When the end of this file is reached, input will be resumed from file specified before the EI was given. EI commands may be nested.
ET	Generate a scratch copy of the file being edited on the disk as "TEMP FILE", and read the first page of that file. The default file-name will not be changed. This command allows one to return to an earlier page in the file being edited easily, and provides an extra copy of the file being edited.

COMMAND	ACTION
EF filename↵	Name the output so far put on the disk and initialize for more disk output. Establish default file-name on disk.
EC filename↵	Copy the input to the end, and name the resultant output. Initialize for more disk output. Establish default file-name on disk.
EG filename↵	Same as EC, except that control is passed to the assembler to assemble the file. If the assembler detects an error, control will be returned to the Editor with this file-name. If no assembly errors were noted, the resulting program will be loaded.

4.4 Editor Command Summary

COMMAND	ARGUMENTS	FUNCTION
A	0	Enter text mode and append to buffer.
m,nA	2	Equivalent to A.
B	0	Back up and print.
C	0	Equivalent to .C.
nC	1	Change line n.
m,nC	2	Change lines m through n.
nD	1	Delete line n.
m,nD	2	Delete lines m through n.
E*	0	Read entire input paper tape and print first line of each page.

* The E Command applies to Editor assembled for paper tape only.

COMMAND	ARGUMENTS	FUNCTION
EB	0	Initialize input from paper tape instead of disk.
EE	0	Resume input from disk.
EP	0	Initialize output to paper tape punch.
ES	0	Initialize or resume output to disk.
ER filename↵	0	Initialize reading from disk file, and read first page. Select disk as output device.
EI filename↵	0	Take subsequent input from file specified. When the end of this file is reached, input will be resumed from file specified before the EI was given. EI commands may be nested.
ET	0	Generate a scratch copy of the file being edited on the disk as "TEMP FILE", and read the first page of that file. The default file-name will not be changed This command allows you to return to an earlier page in the file being edited easily, and provides an extra copy of the file being edited.
EF filename↵	0	Name the output so far put on the disk, and initialize for more disk output. Establish default file-name on disk.
EC filename↵	0	Copy the input to the end, and name the resultant output. Initialize for more disk output. Establish default file-name on disk.
EG filename↵	0	Same as EC, except that control is passed to the assembler to assemble the file. If the assembler detects an error, control will be returned to the Editor with this file-name. If no assembly errors were noted, the resulting program will be loaded.

COMMAND	ARGUMENTS	FUNCTION
F	0	Equivalent to 1F. (Ignored for disk output.)
nF	1	Feed n lines of blank tape. (Ignored for disk output.)
m,nF	2	Equivalent to 1F. (Ignored for disk output.)
G	0	Equivalent to .G.
nG	1	Get next location tag after line n.
m,nG	2	Equivalent to nG.
H	0	Same as W, except line numbering is forced.
nH	1	Same as W, except line numbering is forced.
m,nH	2	Same as W, except line numbering is forced.
I	0	Equivalent to .I.
nI	1	Insert text before line n.
m,nI	2	Equivalent to nI.
J	0	Equivalent to .+1J.
nJ	1	Open line n and, without display, type and enter Text mode for end-of-line corrections or comment insertion.
m,nJ	2	Equivalent to nJ.
K	0	Kill the buffer.
L	0	Equivalent to .L.
nL	1	Print line n.
m,nL	2	Print lines m through n.

COMMAND	ARGUMENTS	FUNCTION
M	0	Equivalent to .+1M.
nM	1	String search within buffer.
m,nM	2	Equivalent to nM.
N	0	Punch, then read next page. Equivalent to O, R.
nN	1	Punch, duplicate, and read. Equivalent to P, S, (n-1)T, R.
m,nN	2	Equivalent to N.
O	0	Punch one page. Equivalent to P, S, K.
P	0	Punch the contents of the buffer.
nP	1	Punch line n.
m,nP	2	Punch lines m through n.
Q	0	Print entire buffer -- uncommented.
nQ	1	Print line n uncommented.
m,nQ	2	Print lines m through n up to second tab.
R	0	Read one page of text and append to buffer.
nR	1	Read n lines of text and append to buffer.
m,nR	2	Equivalent to R.
S	0	Punch form feed.
nS	1	Equivalent to S.
m,nS	2	Equivalent to S.
T	0	Equivalent to 1T.
nT	1	Duplicate n pages of tape. Equivalent to K, n(R, P, S, K).
m,nT	2	Equivalent to 1T.

COMMAND	ARGUMENT	FUNCTION
U	0	Equivalent to 1U.
nU	1	Move display pointer up n lines.
m,nU	2	Equivalent to nU.
V	0	Turn on line numbering.
nV	1	Turn off line numbering (1V is recommended).
m,nV	2	Turn off line numbering.
W	0	Write the entire buffer.
nW	1	Write n pages. Equivalent to K, n(R, W, K).
m,nW	2	Equivalent to W.
m,nX	2	Read next m lines of text from input tape and insert them after line n.
Y	0	String search for entire tape.
Z	0	Skip one page of text.
nZ	1	Skip n pages of text.
m,nZ	2	Equivalent to Z.
⟨alt mode⟩		Equivalent to 1⟨alt mode⟩.
n⟨alt mode⟩		Move display pointer down n lines. This command supplies its own ↓, if in half duplex. The pointer will not move past line /. [.=.+n or / if (/ - current line)<n]
;		Enter Character mode on line n.

4.5 Summary of Editor Operations

If a graphics console is available, the parameter windows will display the page and line number respectively in the A and B windows. Alternatively, the page number may be displayed in the data buffer lights of the mag tape controller.

The A parameter knobs may be used for positioning within the buffer. The right knob, moving vertically from line to line, as ALT mode or U from the Teletype. The left knob will cause entering Character mode, and allow positioning as ⟨space⟩ or RO. The right knob may be used to exit Character mode.

Foot pedals are used in the following way:

> Toggling the left pedal will act in the same manner as typing Y↲↲; that is, repeating the previous search.
>
> Toggling the right pedal performs the same function as typing N↲, going to the next page.

On the FR-80, the left foot pedal is replaced by AC switch 3, the right foot pedal by AC switch 4. The sense switches are replaced by AC switches as follows:

Sense Switch	AC Switch
1	12
2	13
3	14
4	15
5	16
6	17

4.5.1 AC Switch Settings

SWITCH	FUNCTION	
0	Down	Normal operation.
	Up	Stop printing or punching.
1	Down	Simulate full duplex.
	Up	Simulate half duplex.

4.5.2 Special Characters

CHARACTER	FUNCTION
/	Equals number of lines in the buffers.
.	Equals number of last line referenced.
:	Types value of expression preceding it.
⤸	In Command mode: Execute preceding command. In Text mode: Put last line typed in buffer. In Character mode: Exit Character mode to Command mode.
↓	Delete all input since last ⤸ in Text or Command mode.
RO	In Command mode: Print next line (.+1). In Text mode: Leave Text mode if no characters in current line. Otherwise, erase the last character. In Y or M Command mode: When typing string for search match, returns system to Command mode and negates command.

4.6 Duplexing

The Editor can simulate either full or half duplexing according to AC1. In full duplex, the characters typed in are not typed back except for ⤸ and ↓, both of which respond with ⤸. If line numbering is on and the Editor is in Text mode, a ↓ is typed with each ⤸.

In half duplex, all legal characters are typed back, except RO, and either ⤸ or ↓ is answered with both characters. The type out is done at the time the Editor's Teletype input routine takes the character from the input buffer. Thus, characters typed while the

Editor is busy will not be typed out until later. Typing in while the Editor is printing will not bother it.

4.7 CRT Display

Provision is made for displaying the contents of the buffer on the CRT. A right arrow in the first column points to the line equal to ".". This pointer changes automatically when the value of . changes, and can also be moved by U and ⟨alt mode⟩ as described earlier. The display attempts to display as much of the buffer as possible. Whenever the pointer is within four lines of text of the top or bottom of the screen, it is moved to the center, except when that would move the first or last line of the page away from the edge of the screen. The parameters used to do this are recomputed when the display, using the current parameters, runs over the bottom of the screen. Therefore, changes can be forced by setting the pointer to the first line of the page if the page will not completely fit on the screen.

When the Editor is in Text mode, eight points are displayed around the edges of the screen. Note that after each ↲ in Text mode, the pointer moves down and inserts a blank line. If the user returns to Control mode, this line disappears and the pointer moves up one line.

When editing the first line of the buffer, the pointer may disappear under certain conditions, because the Editor has set it equal to line zero.

When sense switch no. 1 is up, each line is terminated at the edge of the screen if it is too long. When sense switch no. 1 is down, lines which are too long are continued on the next line, with an up arrow in the first column. With sense switch no. 1 up, the display runs faster and is neater looking, as the display parameters are adjusted for it.

The display is only on when the Editor is waiting for Teletype input. The display stops as soon as a character is typed. When the user is typing fast, a few lines at the bottom will not be displayed.

4.8 Line Numbering

Another option available with the Editor is line numbering on teletype output. When the numbering is turned on by means of a V, all lines printed by the B, E, G, H, J, L, M, Q, W, and Y commands will be preceded by their line number in the buffer. The numbers are mod 100 and are two characters long followed by two spaces. These numbers do not affect the tabs, which take the first column of the text as column one. The H command always numbers lines. When executing A, C, or I, the number of the line to be entered is typed after each ↲. If a RO is then typed to return to Control mode, the last number typed minus one, is equal to . and is the last line entered. Note that J will also type the next line number after the ↲, even though it is already in Control mode.

4.9 Disk Editor Usage

1. To create a user file:

   A. Using DEBUG, load in the Editor by typing E$J

   B. When Editor is loaded, type:

      EF (User Name); (Program Name)↲

      Example: To create the program name RADAR in WLJ's file, type: EFWLJ;RADAR↲

2. To call a program from the Disk:

   Type ER (User Name); (Program Name)↲

   Example: To call WLJ RADAR from the disk, type: ERWLJ;RADAR↲

   The first page of RADAR will be displayed on the monitor scope.

3. To punch paper tape:

   Type EP

   Pages of text will be punched on paper tape instead of being stored on the disk.

   By typing ES , all pages will be stored on the disk.

4. To read from paper tape, put tape in Reader, then do R's.

   NOTE: If no ER has been done upon loading the Editor, it will default to paper tape input and output. If ER's have been done, and paper tape reading is desired, type:

   EB (read from paper tape)

   To stop paper tape reading type:

   EE

5. To store a program onto the disk after editing, type:

   EC (User Name); (Program Name)↵

   Example: ECWLJ;RADAR↵

   Thus, the steps required for creation and storage of a program on the disk are the following:

   A. Get Editor Program - (E$J)

   B. Create program name - EF (User) ; (Program Name)↵

   C. Call Program from Disk - ER (User) ; (Program Name)↵

   D. Type in program (refer to Editor command set)

   E. Store completed program on disk - EC (User); (Program Name)↵

6. The user may combine editing, assembling, and loading of his program with the following command - EG↵

   After creating or modifying a program by using the ER command, the user, instead of doing an EC, will type EG↵

   This command will store his program onto the disk (EC), call in the assembler, assemble the program, store it in his file as (Program Name) BINARY, and load the program into memory ready to run under DEBUG control. If assembler errors were detected, the process is interrupted, and control is returned to the Editor with the first page of the symbolic program displayed on the monitor screen. The user may now correct any program errors and try the EG command again.

7. The user may combine programs with the use of the EI command.

Example: The user wishes to insert the first five pages of Program X into Program Y. He wishes to insert these pages before page 5 of Program Y. Program X is 10 pages long. Therefore, the last 5 pages of Program X are not wanted. The following commands will accomplish this:

```
ER (User); Y↵

5N↵

EI (User); X↵
```

NOTE: At this point, the monitor is still displaying page 5 of Program Y.

Any editing commands operate on Program X.

Append first 5 pages of X to Y.

```
5N↵
```

Skip last 5 pages of X

```
5Z↵
```

Store Program away

```
EC↵
```

The structure of Program Y now is as follows:

(Original Y)

Page 1_Y
Page 2_Y
Page 3_Y
Page 4_Y
Page 5_Y

(First 5 pages of X)

Page 1_X
Page 2_X
Page 3_X
Page 4_X
Page 5_X

(Last 5 pages of original Y)

Page 6_Y
Page 7_Y
Page 8_Y
Page 9_Y
Page 10_Y

4.10 Using the Editor

4.10.1 Example 1

The following detailed example of the editing of a page of text will familiarize the reader with the basic operations of the Editor. All of the commands are described in full after the example, and the sample page of text is reproduced in Figure 1.

The Editor starts automatically with cleared buffers. All subsequent operations, including loading of the symbolic tape to be edited, are performed through the Editor from the teleprinter keyboard. Our sample text is a page from the symbolic tape of the Editor itself; it is punched in ASCII format.

```
SYMBOLIC TAPE EDITOR PART 2 MAIN SUBROUTINES

X1 = 10
X2 = 11

APPEND,     0
            CHKONE

            SKP
            JMP CONERR
            LAC LASIN
            DAC THISN
            LAC LAST

            DAC THIS
            LAC (NOP

            DAC TISW          /TELETYPE INPUT SWITCH
            DZM LSTCHR        /STORAGE FOR LAST CHARACTER SEEN
            LAC (JMS TONE     /SET SWITCH IN PACK ROUTINE

            DAC ON            /FROM TELETYPE
            JMS A             /APPEND ALL LINES AND RESET POINTERS

            JMP I APPEND

A,          0
            JMS PACK          /ADD ONE LINE
            JMP .-1
            JMP I A

TONE,       0                 /TYPE IN
            TTI
            ISZ TONE          /INDEX RETURN IF NOT DONE
            DAC LSTCRR        /SAVE CHARACTER FOR CHECK IF BLANK KEY TYPED
            JMP I TONE



BLANK,      LAC LSTCHR
            SAD (77
            JMP I TONE        /NO DELETE IF LAST CHARACTER WAS CR
            JMS DECR          /OTHERWISE ERASE ONE CHARACTER
            JMP TONE + 1

DECR,       0
            LAC PCNI          /IF CHARACTER CNT IS -2 MUST BACK UP 1 WORD
            SAD (-2
            JMP FUR
            ADD (-1           /OTHERWISE DECREMENT CHARACTER COUNT BY 1
            DAC PCNI
            LAC PIEM          /GET PARTIAL ACCUMULATION
DE1,        AND (777700       /AND OUT LAST CHARACTER
            CLL
            RTR       RTR     RTR
            DAC PIEM          /ROTATE WORD BACK ONE CHARACTER
            JMP I DECR
```

Figure 1. Editor Part 2 Main Subroutines.

We are now ready to read in our symbolic text. The command for this operation is:

```
       R↵
```

This causes the Editor to read the input tape until a form feed or the physical end of tape is encountered.

Once the text is in the buffer, we might wish to find out how many lines there are in the page, as follows:

```
       /:  57
```

With this information, we can find our way about the page. Note that blank lines are included in the count.

The first task is to insert a comment before the line that begins with the symbol A. To do this, we must find the number of that line. This can be done by asking the Editor to print out a line with the number we estimate is the correct one. We can see that the line is a little less than halfway down the page, and we know that there are 57 lines on the page, so we ask for line 25 using the command L, as follows:

```
       25L↵
              JMP I APPEND
```

Now we see from the program copy that the line we want is really two lines further on.

To insert text before the desired line, we will use the command 27I. This causes the Editor to enter Text mode. As many lines of text as we wish are then entered into the text before line 27. Having done so, we strike the RO key to return to Command mode. The result of our work:

```
       27I↵
        →|  /THE FOLLOWING SUBROUTINES HANDLE↵
        →|  /INCOMING DATA.↵
   (RO)
```

If we wish, we can now verify our work by asking for a printout of the lines including the new text. Bear in mind that the inserted lines have been added; the buffer now contains 59 lines; and all lines after the insertion have had their numbers increased by two. Our verification would look like this:

```
      25,29L↵
                JMP I APPEND
                /THE FOLLOWING SUBROUTINES HANDL
                /INCOMING DATA.
     A,              0
```

Our next correction is to change the name of the subroutine PACK to LPAC. To do this, we use the Change command, C. From the previous printout, we know that the line beginning with the symbol A is now line 29, so the one we wish to change is now line 30. The operation is completed as follows:

```
      30C↵
             ⇥JMS LPAC     /ADD ONE LINE↵
```

If we have no further corrections to make, we are ready to punch out the corrected text. The following sequence of commands causes the contents of the buffer to be punched, followed by some tape feed and a stop code. Finally, the contents of the buffer are totally erased.

```
      P↵
      S↵
      K↵
```

We are now ready to read the next page of the input tape. Figure 2 shows the whole of our operations for this example; the comments in parentheses summarize what happens.

4.10.2 Rearranging Text on Output

Text may be rearranged on output by using the nP and m,nP commands to punch lines or groups of lines from the Editor's buffer in the order of their selection by the user rather than their order in the buffer. The buffer contents are unaffected by this process.

For example, if we wished to relocate the six lines beginning at line 40 in the sample text at the end of the page, we would type:

1,39P↵	to punch the first section of text
46,/P↵	to punch the final section of text
40,45P↵	to relocate the desired lines to the end of the tape being punched
S↵	to complete the page with a stop code and tape feed

```
   R↵                                                  (read one page of input tape)
   /:     57                                           (find number of lines)
   25L↵                                                (print line 25)
          JMP I APPEND
   27I↵                                                (insert text before line 27)
        /THE FOLLOWING SUBROUTINES HANDLE↵
        /INCOMING DATA.↵
(RO)25,29L↵                                            (return to Command mode and
          JMP I APPEND                                 verify previous insertion)
        /THE FOLLOWING SUBROUTINES HANDLE
        /INCOMING DATA
   A,     0
   30C↵                                                (change line 30)
          JMS LPAC     /ADD ONE LINE?↵                 (continuation; remain in
          JMP .-1      /GET NEXT LINE↵                 text mode)
(RO)P↵                                                 (return to Command mode and
   S↵                                                  punch out corrected text.)
   K↵.                                                 (erase the text)
```

Figure 2. Editing Example 1.

4.10.3 Example 2

This example illustrates some of the more subtle commands available to the Editor user. It is suggested that this section not be studied until the user has become acquainted with the fundamental aspects of the Editor illustrated in Example 1, preceding page.

To begin, we return to the sample text given in Figure 1, and read the page into the buffer.

```
R↲
```

Suppose now we wish to make an addition to a line following symbolic location BLANK. To find its line number in the buffer, we will use the nG command. This command searches from line n until a line is found which does not begin with a ⇥, ↲, or /. To avoid stopping at earlier symbolic addresses, we choose n as 35.

```
35G↲
BLANK,   LAC LSTCHR
```

To find the line number, we use the character period.

```
.:40
```

We know the line we are concerned with is two numbers further, that is line 42. In line 42, we wish to change the instruction JMP I TONE to JMS TONE deleting the I and changing the P to S. To accomplish this, we use the command n; to enter Character mode.

```
42↲

      4 ⟨space⟩ DDOS↲
```

"4 ⟨space⟩ " positions the Character mode pointer under the P. The first "D" deletes the P. The second "D" deletes the space that follows it. The pointer is now under the "I", which should be changed to "S". Typing "O" enters Overstrike mode. Typing "S" changes the "I" to an "S".

To inspect the altered line without bothering to print the comments, we could use the command nQ. This suppresses all text following the second tab on each line.

```
        42Q↵

                JMS TONE
```

Now suppose we want to punch the corrected text but move the instructions beyond the address DECR to the next page of our tape. We can punch the first portion of text down to line 46, then delete it moving DECR to line 1.

```
        1,45P↵
        S↵
        1,45D↵
        1L↵
        DECR,↵                  0
```

We may now read the next page of tape into the buffer following the instruction JMP I DECR. Figure 3 summarizes this example.

```
R↲                (read 1 page of input tape)
35G↲              (print first symbolic addressed line after 35)
BLANK,   LAC LSTCHR
.:40              (print the number of the current line)
42;               (enter Character mode for corrections)
         4<space>DDOS↲
42L↲              (check correction made)
         JMS TONE /NO DELETE IF LAST CHARACTER WAS CR
42Q↲              (check corrections, suppress comments)
         JMS TONE
1,45P↲            (punch lines 1 through 45)
S↲                (punch a stop code)
1,45D↲            (delete lines 1 through 45)
1L↲               (check the location of DECR)
DECR,    0
```

Figure 3. Editing Example 2.

APPENDIX C

III PROGRAM LIBRARY

III-101D; ASSEMBLER (FOR DISK SYSTEMS)

2. ABSTRACT

2.1 Purpose

To assemble symbolic source programs into binary program images.

4. USAGE

4.1 Source Language

CHARACTER SET - The set consists of:

1) letters A through Z
2) digits Ø through 9
3) the following special characters:

CHARACTER	MEANING
Space	
+	plus
-	minus
*	times
:	divided by
!	inclusive OR
←	shift
&	logical AND
<	arithmetic left delimiter
>	arithmetic right delimiter
(	constant left delimiter
)	constant right delimiter
[	text left delimiter
]	text right delimiter
#	variable indicator
'	text concatenation
"	7-bit ASCII

CONT.

4.1 CONT.

CHARACTER	MEANING
,	symbolic definition, or argument delimiter
.	current location, symbol constituent, or decimal indicator
/	location setter, or comment delimiter
\	macro argument delimiter
=	symbol assignment
cr	word delimiters
tab	
$	extended address indicator

NUMBERS

A number is a sequence of digits whose value is determined by the radix established by the OCTAL or DECIMAL pseudo-op. A sequence of digits including a period is always evaluated as a decimal integer.

ASCII CHARACTERS

The character double-quote (") is used to cause evaluation of the 7 bit ASCII code for the character which immediately follows. Thus the value of "A is 101_8.

LOCATION COUNTER

The assembler begins assembling code at location 100_8 and increases the location counter as it assembles. The current location is referenced using: . (period). The location counter may be changed by using / preceded by an expression, all of whose constituents have been previously defined.

SYMBOLS

A symbol is any set of up to 6 characters chosen from the letters, digits and period; at least one of which is a

CONT.

4.1 CONT.

letter. (More than 6 characters may be used, but the assembler ignores all after the sixth.) Interspersed with these characters, but not included in the symbol, may be the special characters "$" and "#".

All standard operation codes and assembler pseudo-ops are defined within the assembler.

EXTENDED MEMORY REFERENCES

Ordinarily, quantities which represent addresses; such as ".", variables, and symbols defined with a comma are treated as 13-bit integers. To refer to the full 15-bit quantity when assembling for a machine with extended memory, a "$" is placed before or after the period or symbol.

SYMBOL ASSIGNMENTS

A symbol followed by a comma causes the symbol to be given the current value of the location counter. A symbol followed by = followed by an expression causes the symbol to be given the value of the expression. Symbols which are defined by using "=" may be redefined as desired, but errors will result from inconsistent definitions of symbols defined by any other way.

(A statement containing two consecutive equals signs assembles the same as one with a single equals sign, except that the symbol assignment will not be included in the binary output file.)

VARIABLES

The assembler will assign storage locations to all symbols which contain #.

OPERATORS

The following characters are infix operators, with the following meanings:

4.1 CONT.

+ or space	e_1+e_2 or $e_1\ e_2$	has the value of the sum of e_1 and e_2.
-	e_1-e_2	has the value of the difference of e_1 minus e_2.
*	e_1*e_2	has the value of the product of e_1 and e_2.
:	$e_1:e_2$	has the value of the quotient of e_1 divided by e_2.

The value obtained from any of the previous operators yields one's complement form negative numbers.

!	$e_1!e_2$	has the value of the 18-bit inclusive-OR of e_1 and e_2.
&	$e_1\&e_2$	has the value of the 18-bit logical-AND of e_1 and e_2.
←	$e_1\leftarrow e_2$	has the value of e_1 shifted left e_2 places. If e_2 is negative, the value is that of e_1 shifted right $-e_2$ places. In all shifts, zeroes are shifted in.

If two or more operators appear together, the rightmost is used.

If e_1 is not present, 0 is assumed.

CONT.

4.1 CONT.

EXPRESSIONS

An expression is one of the following; followed by a tab, carriage return, or with some pseudo-ops, a comma.

- .   (location counter)
- a symbol
- a number
- an ASCII character
- expression operator expression
- <expression>

If hierarchy delimiters ("<" and ">") are not used, expressions having more than one operator will be evaluated from left to right.

The hierarchy delimiters < and > are used as (and) are used in conventional algebraic notation.

COMMENTS

A slash immediately preceded by a word delimiter causes everything from the slash to the next carriage return to be ignored by the assembler. The first line of input is treated as a comment and also printed.

CONSTANTS

Constants may be referenced by surrounding them with parentheses. This causes the assembler to store the constant in a memory location and substitute the address of that location for the expression within the parentheses.

CONT.

4.1 CONT.

PSEUDO-OPS

Note: Only the first 6 characters of any pseudo-op need be used.

OCTAL - Appearance of this pseudo-op causes all succeeding numbers in the source code to be interpreted as decimal until the next occurrence of OCTAL.

Note: The initial mode of the assembler is OCTAL.

CONSTANTS - This causes currently unassigned constant locations to be assigned.

VARIABLES - This causes currently unassigned variables (symbols used with #) to be assigned.

START - If this is followed by a location, the object program will start at that location when loaded. Otherwise, the program will halt when loaded.

.PRINT - This pseudo-op will cause the character string following it to be printed at assembly time. It has no effect on the object program. The first non-blank character after the .PRINT pseudo-op will not be printed, but used as a terminator for the string.

CONT.

4.1 CONT.

.INSERT - Causes insertion of the file name that follows on machines with mass storage, if that file has not previously been inserted. This pseudo-op and the remainder of its line is ignored by the paper tape assembler.

.RPT

Any text string may be repeated in the source code by preceding it with .RPT e, where e is any legal expression. The text following the comma will be repeated a number of times given by the value of e. (If this value is negative or zero, the text will not be examined by the assembler.) The text may be enclosed in brackets, in which case it will be repeated verbatim; or it may be ended by a carriage return, in which case it will be repeated up to and including the carriage return. Within the text to be repeated, the symbol .RPCNT has the value of the number of completed repeats so far.

IFS

A set of conditional assembly pseudo-ops exist which allow skipping or assembling text based on evaluation of an expression. The format for the text of the statement is as specified for repeats. (i.e. either terminated by a carriage return or included within brackets.)

CONT.

4.1 CONT.

The pseudo-ops and their meanings are:

.IFZ e, --- if e=0 or e=-0 assemble text

.IFNZ e, --- if e=0 or e=-0 skip text

.IFP e, --- if e>0 assemble text

.IFNP e, --- if e>0 skip text

.IFM e, --- if e<0 assemble text

.IFNM e, --- if e<0 skip text

.IFD s, --- (where s is a symbol)
if s has previously been encountered in assembled text, assemble text.

.IFND s, --- if s has previously been encountered in assembled text, skip text.

The inclusive OR of several conditions may be expressed by separating the conditions with semicolons. For example,

.IFZ e_1;.IFND s;.IFM e_2,e_0

will assemble e_0 if e_1 is zero, or s has not been encountered, or if e_2 is negative.

MACROS

A macro definition is specified by the pseudo-operation .DEF followed by a name, followed by a list of dummy variables, followed by the text of the macro. The name may be any legal symbol. The arguments are separated by commas, slashes or back slashes and the list is ended by a carriage return. One back-slash may be inserted to separate the argument list. Those arguments following the back-slash will have a generated symbol of the form Gxxxxx (where the x's are octal digits) if that argument is not supplied by a macro call. Alternatively, one slash may be inserted to separate the argument list. This does the same as a back-slash except that the only terminator

CONT.

4.1 CONT.

recognized for the argument preceeding the slash in a call for this macro will be a carriage return. Since this also terminates the macro call, subsequent arguments will have generated symbols substitued. The text of the macro is ended by the pseudo-op .TERM. Within the text of the macro, character concatenation is specified by an apostrophe. The macro is called by using its name followed by a list of arguments separated by commas. The macro call is terminated by a tab or carriage return. Missing arguments will have an appropriate null substitued, unless a generated symbol is called for. If an argument contains a tab, space, comma, or carriage return it should be enclosed in brackets. If an argument is an expression to be evaluated, it should be enclosed in back-slashes (\).

4.2 ERROR MESSAGES

The possible error messages and their meanings are:

SCE - Symbol core exceeded - the program has too many symbols.

CLD - Constants location disagrees.

MDV - Multiply defined variable - a symbol has been defined twice, once with the #.

ILF - Illegal format - special characters have been used the wrong way.

VLD - Variables location disagrees.

CONT.

4.3 OPERATING INSTRUCTIONS FOR DISK ASSEMBLER

The assembler is kept on the disk as SYS; A BINARY.

Typing A$J to DEBUG loads and starts the assembler. It is then waiting for a file name to be typed. Unless another directory name is typed the default directory name will be used. If the file is not found the error message: FNF "filename" will be typed. The name may then be retyped. When processing .INSERT pseudo-ops, if the directory name is not specified the default directory name will be used. If the file is not found in that directory, a search will be made in the mandatory directory "SYS". If the file is not found there, the error message FNF "filename" will be typed and control returned to DEBUG.

If A$G is typed to DEBUG, or the assembler is called automatically by the EDITOR, the following occurs: The assembler uses the default directory and file name as the program file to assemble. When the assembly is complete, if errors were typed, the EDITOR is loaded and started with the default file opened. If no errors were typed, the binary program file just created is loaded.

The output from the assembler is a file with the same first name as the input program and a second name of "BINARY". This file will either replace a previously existing file or be created as a new file.

CONT.

5. PROPERTIES

5.1 USEFUL PARAMETERS

MOVEDN To produce programs with a loader which does not occupy the highest core locations set MOVEDN to the number of words to leave free in upper core.

CORES To utilize the additional storage afforded by extended memory set CORES to the number of 8192 word core banks available.

APPENDIX D

III PROGRAM LIBRARY

III-139, TAPE DUMP RELOADER

2. ABSTRACT

2.1 This program provides for the replacement or creation of disk files as copied from magnetic tape. Single files, single directories, or the entire magnetic tape can be reloaded under operator command. The disk file system may be initialized with this program.

3. REQUIREMENTS

None.

4. USAGE

4.1 When started, the program will type the identification information from the first record on the tape (if there is no tape ready on Unit #1, the monitor will display the "HANG A TAPE" message).

The tape format from which files are to be reloaded is given in the program description for III-138, Disk Dumper. If the tape is not in this format, the error "INVALID FORMAT!" is typed and displayed.

The error, "BAD TRACK USAGE TABLE", may be typed and displayed on the monitor. If this happens, the only command that may be entered is "WIPE)" to initialize the disk.

When a file from magnetic tape is copied to the disk, it will replace an existing file with the same name, if one exists.

Each time a file is loaded onto the disk, date informatio from the description record is stored on the disk to be displayed by III-148, Disk Audit, when that directory is being displayed.

The commands which may be typed to this program are:

COMMAND	ACTION
ALL↵	Copy all files after the current position of the tape to the disk, and rewind the tape.
Bfilename↵	Same as Ffilename BINARY↵
Ddirectory name↵	Copy all files on the tape after the current position of the tape with the specified directory-name. (If no directory-name is specified, the directory-name from the first file on the tape will be used.) The tape will be rewound at completion of this command. The directory searched for becomes the disk default directory-name. If no files were found on the tape with that directory-name, "FNF" will be typed back. Successful completion of this command is indicated by typing "OK" on the teleprinter.
Ddirectory name← new name↵	Same as Ddirectory name↵ except the tape will be searched for files having the specified directory name and stored in by the same name in the new directory.
E	Search for description record (date) change. The new description record information and the filename will by typed when encountered. The tape will be positioned to read that file next.
Ffilename↵	Locate "filename" and copy it on to the disk. (If no filename is typed, the previously typed filename will be used. If none was previously typed, the name from the first file on the tape will be used.) The directory name of the file searched for becomes the disk default directory name.

COMMAND	ACTION
Ffilename←new name↵	Search for "filename" on tape and store it as new name on disk.
I	Generate index of tape to display on the screen. The index will portray each filename in a two-column format. Following each filename, is the number of disk blocks required to hold the file. If the blocking factor on the tape is less than 8, the blocking factor used will appear in parenthesis after the block count. Consecutive files on the tape will be in alternate left and right columns. At any file, if the description record changes, its contents will be put on the next line and the filename for that file will be in the left column of the following line. The tape will be rewound at completion of this command.
N	Initialize for new tape.
R	Rewind the tape.
T	Performs all the functions of the "I" command as an index in the preceding format is typed.
WIPE↵	Initialize the disk. Blocks 40_8-1777_8 are filled with zeros; a master directory with one entry, "SYS", is created; the "SYS" directory has no files; a track usage table with all allocatable blocks indicated available is written.

4.2 Error Messages:

When loading files onto the disk, the message, "DISK FULL", may be typed and displayed on the screen. This means that insufficient blocks are available on the disk to hold the file being loaded. If "DIRECTORY FULL" is typed and displayed, either the directory for the file being loaded had no room for a new entry (each directory may have a maximum of 50 files), or there were no more directories assignable (a maximum of 15 directories may exist).

"CKSM ERR BLOCK n ddd;ffffff" means that there was a disk checksum error in that block when the file was copied to magnetic tape. This is only a warning message; the file will be loaded normally.

A bell is the response to an illegal command type-in.

5. PROPERTIES

By assembling with TAPMON defined, the code for the FR 80 Monitor will be inserted.

APPENDIX E

III PROGRAM LIBRARY

III-138; DISK DUMPER

2. ABSTRACT

2.1 To dump any file, any user directory, the entire disk, or files from paper tape onto magnetic tape.

3. REQUIREMENTS

None.

4. USAGE

4.1 When started, the program will request that the operator enter the date. Any characters may be entered into the data except line-feed and rub-out. Line-feed will renew the request for the date. Rub-out will erase the previous character entered. The date is entered when a carriage return is typed.

The following commands may now be typed:

ALL ↵ - Rewinds the tape, dumps the entire disk file system onto tape, and rewinds the tape.

B - Backs over one file and types its name.

C - Search the tape for a different date.

D directory-name↵ - Dumps all files in one directory (leaves tape positioned for further dumps). If no directory name typed, the default user on disk will be used.

D directory-name←new name
or
D←new name↵ - Tape is written with new directory name in place of specified or default directory name.

E - Positions tape at end of previously written dumps in preparation for dumps onto the end of the tape.

F file-name↵ or
F file-name←new name - Dumps one file onto the tape (leaves tape positioned for further dumps)

N - Allows entry of a new date.

P file-name↵ - Starts reading paper tape and copying it to magnetic tape (file name must have a valid directory name). The copying will continue until the end of the paper tape is reached. If the second name of file-name is "BINARY" file marks are written and control is returned to the next command. If the second name is not "BINARY" either a "C" or "S" must be typed. "C" indicates another paper tape is to be read to continue the file. "S" indicates the last paper tape has been read in; file marks will be written and control will be returned to the next command.

Q file-name↵ - Same as "P" except the second name is forced to "BINARY".

R - Rewinds the tape.

Sn↵ - Tape blocking factor (1 to 8)

T - Lists contents of tape in INDEX form.

V - Verifies a file from paper tape. The program will type back the file-name against which the tape will be compared. At the end of the paper tape the actions are as described in "P". An unsuccessful comparison types "COMP ERR" and returns to DEBUG.

WEØT↵ - Write an END-OF-FILE mark on the tape.

X - Provides typeout of file names in "D" and "ALL" dumps (initial mode)

Z - Suppresses typeout of file names in "D" and "ALL" dumps

At interrupt time, ↑S, ↑K, and ↑D (control-S, control-K, and control-D) are recognized and operate as specified in the Keyboard Routines (III-113).

Unrecognized commands will ring the bell.

Successful completion of a command is indicated by the teletype typing 'OK'.

CONT.

5. PROPERTIES

5.1 All information dumped will be verified by read-compare with disk data.

5.2 MAG TAPE FORMAT:

Each disk file will be stored on magnetic tape as one physical file. (Two consecutive file marks terminate the tape.)

Each file will consist of one label record and one or more blocked data records. The format of the label record is as follows:

400_8 words;

0 - Directory name (6-bit ASCII-40_8, 3 characters/word)

1-4 - File name " " " " " "

5 - Zero for 7 track 556BPI, negative for 9 track core dump mode, positive for 7 track 800 BPI

6-377 - Date (8-bit ASCII, 1 character/word)

The format of the blocked data records is as follows:

One to eight 400_8 word data blocks. Each data block except the last contains in its first word a serial number (beginning with 1). The first word of the last block contains a zero. (If there was a check sum error on the disk block from which a data block was copied, the sign bit of the first word of that data block will be set to flag the suspect data.) The remainder of the data block is copied directly from the disk (376_8 words of data followed by an unused word).

CONT.

Symbolic files will contain two characters per word. They are 8-bit Teletype characters right-justified in each half word field. The terminal character of the file is 141_8.

Binary files (identified by words 3 & 4 of the label record being "BINARY") contain data in the format described in the "DISK DEBUG" writeup (III-126D).

If the last data block does not complete the tape record, words of -0 will be used to fill out the record.

5.3 If default SELECW is altered, it must be done prior to starting execution so that the sixth word of the header is correctly generated.

6. FILE NAME SPECIFICATION FORMAT

Where file-name is required in the input, the following formats are valid:

↵ - Use default file name

directory name;file name↵

file name↵ - Use default directory name

Where a new file name is input, the same formats are valid.

7. NOTES

7.1 ERRORS

BADMAS - Bad master directory on disk, no dumping can be done.

BADDIR - Bad user directory prevents dumping any files from that directory

CKSM ERR - A checksum error was found on the disk in the indicated block of the indicated file

WRITE ERR - Bad tape prevents dumping - exits to Debug

COMP ERR - Bad tape prevents dumping - exits to Debug

NEED RING - Attempting to write on write protected tape - put tape unit off-line, put ring in tape, and put unit back on line.

APPENDIX F

III PROGRAM LIBRARY

III-123; MAGNETIC TAPE DISPLAY

2. ABSTRACT

2.1 This program provides for recovery of information from magnetic tape.

Any record on magnetic tape at 200, 556, or 800 bits per inch in either BCD (even) or BINARY (odd) parity may be examined through the first 4000_8 18-bit words. (6144_{10} characters)

Any portion of the record may be displayed on the PFR-3 monitor or teletype. Data in BINARY records may be displayed as 19-bit signed decimal integers or 18-bit octal numbers. Data in BCD records may be displayed as 18-bit octal numbers or 6-bit characters. Octal numbers may be displayed in either signed or unsigned integer form. Any integer may be displayed with leading zeroes printed or suppressed.

Optionally, repeating sequences of words can be displayed in a compacted format.

A brief command summary is displayed on the monitor under user command (space bar).

CONT.

3. REQUIREMENTS

None.

4. USAGE

4.1 When the program is started a summary of the commands is displayed on the monitor scope.

4.2 Commands

4.2.1 Commands are entered at the teletype keyboard and are echoed on the printer as they are interpreted. A command is a single character immediately preceded by 0, 1, or 2 parameters. A parameter is an unsigned number. (Two parameters are separated by a comma) Unexpected parameters are ignored. Unrecognized commands clear the teletype buffers

CONT.

4.2.1 cont.

and print a question mark. Multi-digit parameters are interpreted as octal or decimal depending on the current mode.

The interrupt characters detected by the teletype routines are effective without interference to the command sequence. The characters and their consequences are described in the description of the Keyboard Routines (III-113).

4.2.2 Mode Commands (Initial state underscored)

4.2.2.1 S,U - Signed or Unsigned

In displaying data in octal mode they will be represented as signed or unsigned quantities.

4.2.2.2 Z,X - Suppress or Print Leading Zeroes

In displaying numerical data words, leading zeroes will be replaced with blanks if Z.

4.2.2.3 D,O - Decimal or Octal

Parameters input, and all numerical data output will be treated as decimal or octal integers.

4.2.2.4 P,Q - Page or Line

In Line mode, the display will consist of only one line of data and the record header. (The record header only prints on TTY when the first word of the record is being printed.)

In Page mode, display continues consecutively from the first word displayed until the monitor screen is filled, the end of record is reached, or teletype output is interrupted. Teletype display can be interrupted at the end of any data line by entering any character (except the interrupt characters) while the line is printing. The characters entered will be interpreted when the line has finished printing. (ALT-MODE may be used to resume printing where broken off)

4.2.2.5 M,T - Monitor or Teletype

In Monitor mode the display will be on the PFR-3 monitor. In Teletype mode, entering space on the

cont.

4.2.2.5 (cont.)

keyboard will restart printing the display on the teleprinter.

4.2.2.6 2B, 5B, 8B - 200, 556, or 800 bits per inch

Select magnetic tape unit to read at this density.

4.2.2.7 Space - Change or Initiate Display

In Monitor mode, if command summary is being displayed, change to data display, else revert to command summary display.

In Teletype mode, restart display.

4.2.2.8 nC - Compact n word groups

If n is omitted or 0, no compaction will occur. (The program is initially in this state.)

If n $\geq$ (NUMBER OF WORDS DISPLAYED PER LINE), then (WORDS-PER-LINE - 1) will be used instead of n. This number will be used as a maximum span in compacting the display, repeating word sequences. (i.e., n=1 searches only for repeating words;
n=2 searches also for repeating pairs, etc.)

If compacting of the sequence could result in the display of fewer lines the compacted data is displayed as a word count followed by the repeating sequence.

e.g. the display:

```
10 202020  202020  202020
13 202020  202020  777777
```

would be displayed in compacted form as:

```
10#  5* 202020
15  777777
```

4.3 Positioning Commands

4.2.3.1 L - Rewind Tape

Position tape unit at load point.

cont.

4.2.3.2 m,nR - Read Magnetic Tape

If m and n are omitted the next record on the tape will be read.

If m is omitted the nth. record in the current file will be read.

The nth. record of the mth. file will be read. (If there are fewer than n records in the file, the last record in the file will be read.)

Display of the record is automatic.

Unreadable records will have a question mark inserted into the header.

4.2.3.3 nW - Display Starting with Word n

If the record does not contain a word n, a question mark will be returned and no action will be taken.

Display begins automatically.

4.2.3.4 nALT-MODE - Display Next

n omitted or 0 interpreted as 1

Positions display start at first word after current display (unless beyond record).

If $n > 1$ each subsequent value attempts to position 8 words later, refusing to go beyond record end. (i.e., if large n entered, last words of record displayed)

4.2.3.5 n= - Scan Forward for Equal

Relationship is exact equality with comparand.

A minus sign entered before, after, or between parameter digits negates the value of the parameter.

Beginning with first word being displayed, apply relationship to each word in remainder of record until it applies. (-0<0) When it applies, initiate display at this point. If it does not apply leave display as it was and print a question mark.

4.2.3.6 n< - Scan Forward for Greater Number

Relationship is n< comparand.

Otherwise same as =.

cont.

4.2.3.7 n > - Scan Forward for Smaller Number

Relationship is n > comparand.

Otherwise same as =.

4.2.4 Other Commands

4.2.4.1 CR - New Line

Can be used to cancel parameter, or stop printing without side effect.

5. PROPERTIES

5.1 Output Format.

5.1.1 When tape is at load point:

'TAPE AT LOAD POINT'

5.1.2 When an end of file is read after file n:

'END OF FILE n'

5.1.3 When a record has been read: the header will display:

```
nR     mF     x W   BIN
↑      ↑     ↗ ↑ ↑   ↑
1      2     3 4 5   6
```

1 The record number, right justified appears here

2 The file number, right justified appears here

3 The number of words read appears here

4 Blank if record was not longer than buffer else '+'

5 Blank if record read without parity error else '?'

6 BIN if odd parity tape else 'BCD'

5.2.4 Subsequent lines will display contents of the record in one of the following forms:

cont.

5.2.4 cont.

5.2.4.1 Normal Form

$w \quad X_w \quad \cdots \quad X_{w+\ell-1}$

w is the word location in the record

X_w is the contents of that word

ℓ is the number of words/line*

5.2.4.2 Compacted Form

$w\# \quad n* \quad \underbrace{X_w \cdots}_{\text{Repeating sequence}}$

w is the first word location

X_w is the contents of first word of repeating sequence

n is the number of words in repeating sequence

5.2.5 SAMPLE OUTPUT WITH EXPLANATION:

(See Pages 9 and 10)

* For monitor display WPL contains the number of words/line that will be used in two's complement form (Initially 8)

cont.

```
 (1) •
 (2) L
 (3) 5B
 (4) 2C
 (5) UO
 (6) XT
 (7) 7777R     103 R       1 F    1000 W     BIN


          Ø    ØØØ3Ø4  ØØØ24Ø  ØØØ33Ø  ØØØ3Ø3  ØØØ3Ø3  ØØØ315  ØØØ3Ø1  ØØØ21
         1Ø    ØØØ212  ØØØ33Ø  ØØØ3Ø3  ØØØ317  ØØØ3Ø4  ØØØ3Ø5  ØØØ264  ØØØ25
         2Ø    ØØØ211  ØØØ33Ø  ØØØ33Ø  ØØØ211  ØØØ257  ØØØ323  ØØØ324  ØØØ3Ø
         3Ø    ØØØ323  ØØØ31Ø  ØØØ24Ø  ØØØ3Ø3  ØØØ3Ø1  ØØØ311  ØØØ24Ø  ØØØ31
         4Ø    ØØØ316  ØØØ324  ØØØ317  ØØØ24Ø  ØØØ324  ØØØ3Ø5  ØØØ315  ØØØ32
         5Ø    ØØØ24Ø  ØØØ3Ø3  ØØØ3Ø5  ØØØ314  ØØØ314  ØØØ215  ØØØ212  ØØØ21
         6Ø    ØØØ312  ØØØ315  ØØØ32Ø  ØØØ24Ø  ØØØ33Ø  ØØØ3Ø3  ØØØ317  ØØØ31
 (8) 65ØW
        65Ø    ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØ37Ø  77777
        66Ø #     12Ø*  777777


 (9) M WØ= T
         73    ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØØØØ  ØØØ214  ØØØØØ
(10) Q
        1Ø3 #     14Ø*  ØØØØØØ
     DZ
        163       175     21Ø     197     193     196     16Ø     177     16
     -
        163       175     21Ø     197     193     196     16Ø     177     16

     Q
        243       257     322     3Ø5     3Ø1     3Ø4     24Ø     261     24
     777W
        777    777777
(11) 1ØØØW?
     MW-Ø=T
        657 #     121*  777777
(12) MWT325<
          2       33Ø     3Ø3     3Ø3     315     3Ø1     215     212     33
     REND OF FILE      1
     L TAPE AT LOAD POINT


     R       1 R       1 F    144Ø W     BCD


          Ø    2Ø3447  516563  464447  2Ø3422  656323  714645  2Ø223Ø  22204
(13) 2BLR       1 R       1 F     621 W?    BCD


          Ø    345767  466736  677347  223266  636374  2Ø2Ø2Ø  2Ø2Ø2Ø  2Ø2Ø2
(14) 5BDP1R       1 R       1 F      743 W     BCD


          Ø  PDOK %LABEL PDOK<<
          8 #      12*
         2Ø  ØØØØ3ØØ↑      %ENTRY PDO
         28 K1 %LABEL PDOK1<<
         36 #      11*
         47 ØØØØ4ØØ↑      GO %BUS %D.
         55   1<<
```

cont.

5.2.5 cont.

(1) Tells DEBUG to start program

(2) Rewind the tape

(3) Setup to read 556 bits/inch

(4) Compact word pairs when found

(5) Display as unsigned octal numbers

(6) Print leading zeroes on teletype

(7) Read last record of the file.

The last record of the first file was number 103.

Its 1000 words were read without error in binary mode.

(8) Output is interrupted by request to display the record from word 650 onward. The output shows that all the words from 657 to the end of the record contain 777777.

(9) The display is shifted to the monitor.

W moves the display to the top of the record.

0= searches for first word in the record containing zero.

T changes the display to the teletype and a blank starts it printing. The first zero was found at word 73.

(10) Output was interrupted with a request to display only one line at a time.

ALT-MODE (not printing) and space request display of next line.

(11) Request for non-existent word returns question mark.

(12) Display first word in record greater than 325

(13) Set density to 200 bpi, rewind and read.

The question mark after the W in the header indicates that the record could not be read without error.

(14) The density is changed to 556 bpi, Display mode changed to Decimal, meaning character representation for a BCD record. Printing of the whole record is selected and it is re-read. Word 8 begins a sequence of 12 words of blanks.

cont.

6. NARRATIVE

6.1 Method

Tape read errors cause a re-read in the opposite parity mode to be attempted. After a total of five reads the record is accepted and an error flag is inserted to the display header.

COMMAND SUMMARY - MAGNETIC TAPE DISPLAY

MODES

(initial state underscored)

S,U	-	Signed or Unsigned
Z,X	-	Suppress or Print Zeroes
D,O	-	Decimal or Octal
P,Q	-	Page or Line
M,T	-	Monitor or Teletype
2B, 5B, 8B	-	200, 556, or 800 bits per inch
Space	-	Change or Initiate Display
nC	-	Compact n word groups (Init n=0)

POSITIONING

L	-	Rewind Tape
R	-	Read next
nR	-	Read nth record of current file
m,nR	-	Read nth record of mth file
nW	-	Display starting with word n of record
$	-	Display next part of record (See note below)
n$	-	Skip 8(n-1) words into next part (See note below)
n=	-	Scan forward and display from first word to equal n
n <	-	Scan forward and display from first word greater than n
n >	-	Scan forward and display from first word less than n

OTHER

CR	-	Gives new line and cancels parameter

INTERRUPTS

Control-D	-	Enter DEBUG at 15000 (RETURN$X to resume)
Control-S	-	Stop printing (kill TTY output buffer)
Control-K	-	Kill TTY input buffer

[illegible]

APPENDIX G

III PROGRAM LIBRARY

III-148; DISK AUDIT

2. ABSTRACT

2.1 This program detects and corrects errors in the disk file system, displays the contents of the disk, and provides for deletion and renaming of files.

3. REQUIREMENTS

None.

4. USAGE

4.1 The program initially reads every track on the disk to accumulate chain data. If any errors are detected in the information stored on the disk, or checksum errors in any block, appropriate messages will be typed describing the error. When done, the program will be in "directory mode".

4.2 Directory-mode. In directory-mode a summary of disk utilization broken down by users will be displayed on the monitor. A sample display follows:

```
$T
FREE BLOCKS 138

SYS  171                       AFS  0
SYM  0                         ENG  0
FR8  631                       PWC  0
RPH  0

SYSTEM MESSAGE
```

Free blocks refers to the number of blocks in the file system area not currently in use by any file. The numbers adjacent to each user name refers to the number of blocks in use by all files for that user.

The following inputs will be accepted from the keyboard:

CONT.

Carriage-return - Switches to "File-mode" using the default user name.

$RESTORE ↲ (1) - Restores all directories to the state they were in when the program was started. (Will not work if operator has returned to DEBUG since the program was started).

User-name ↲ - Switches to "File-mode" in the directory of user-name and establishes user-name as the default user-name on the disk. If the user-name typed did not previously exist, a response of "NEW DIRECTORY?" will be typed. Any character typed in other than carriage return will abort the request and return to directory mode. If a carriage return is entered, an empty directory will be created for the user. (Unless no more master directory space is available)

$T ↲ - Types the summary being displayed on the monitor.

$CLEAN ↲ - In all directories deletes those files whose second name is "BINARY" if there exists within the same directory a symbolic file with the same first name.

Deletes all files with the name "TEMP∧∧∧ FILE".

$ZEROoooo ↲ - Writes a block containing all zeroes to the specified disk address.

@ - Enter or exit mode where every directory entry, including unnamed ones, is displayed. Unnamed entries will show as "---".

#n ↲ - Switch to "File mode" in directory n. ("SYS" is directory #1.)

$U ↲ - Displays a table showing the usage of all disk blocks. A sample display is shown on Page 4.

$M ↲message↓ - Stores system message. The system message may contain up to 254 characters and will be displayed every time this program is run.

(1) Throughout this document "$" refers to either ALT-MODE or the dollar sign.

CONT.

4.2 (cont.)

```
ØØØØ   WWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWW
ØØ4Ø   WWWWWWWWWWWWWWWW WWWWWWWWWWWWWWWW
Ø1ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø14Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø2ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø24Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø3ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø34Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø4ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø44Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø5ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø54Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø6ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø64Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø7ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
Ø74Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
1ØØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
1Ø4Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
11ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
114Ø   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
12ØØ   XXXXXXXXXXXXXXXX XXXXX-----------
124Ø   ---------------- XXXXXXXXXXXXXXXX
13ØØ   XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
134Ø   XXXXXXXXXXXXXXXX XXXXXXXXX-------
14ØØ   ---------------- ---XXXXXXXXXXXXX
144Ø   XXXXXXXXXX------ --------XXXXXXXX
15ØØ   XXXXXXXXXXXXXXXX XXXXXXXXX-------
154Ø   -------X-------- ----XXXXXXXXXXXX
16ØØ   XX---XXXXXXX---- ----------------
164Ø   ---------------- ----------------
17ØØ   ---------------- ----------------
174Ø   ------------MDDD DDDDDDDDDDDDTUUU
```

The left column gives octal disk addresses. Each letter in the table describes the usage of the block represented by its position.

W means Working block

X allocated file block

- available file block

M Master directory

D user directory

T Track usage table

U Unused blocks

CONT.

4.2 (cont.)

Typing a carriage-return returns control to "Directory Mode".

$UT ↲ - Types the table displayed above before displaying it. Typing a carriage-return returns control to "Directory Mode".

$Soooo ↲ - Shows contents of block oooo in symbolic format. The top line of the display gives the block number and the contents of its chain word.

Typing a carriage-return returns to "Directory Mode".

Typing "N" shows the contents of the block to which the current one chains.

Typing space shows the contents of the next consecutive block.

Typing "A" shows contents of next available block.

$SBoooo ↲ - Shows contents of block oooo in binary format. The display shows 100_8 words in octal unsigned form. To display the next 100_8 words of data, type in $ (alt. mode). Other inputs are as for $Soooo.

4.3 <u>File Mode</u> - In File Mode, a listing of all files and their lengths within one directory is displayed. A sample display follows:

```
 $T
 FREE BLOCKS 135

 SYS   174

 D       BINARY 23          E       BINARY 27
 F       BINARY 23          A       BINARY 20
 M       BINARY 19          R       BINARY 21
 S       BINARY 17          AA      BINARY 21
*SKY            3

 18 JULY III123 MASTER
```

Last line of sample display indicates tape from which most recently loaded. Free blocks are the same as in directory mode. Numbers next to file names refer to the number of disk blocks used by that file. The asterisk next to a file name, if it appears, means that that name is the default file name.

The following inputs will be accepted from the keyboard:

carriage-return - Returns to directory mode.

$RESTORE ↵ - As in directory mode. If successful, returns to directory mode.

$DELETE ↵ - Delete all files from directory. If no files were in the directory the name will be removed from the master directory and control returns to directory mode.

file-name$D ↵ - Delete "file-name"*

file-name$B ↵ - Delete "file-name BINARY"

$T ↵ - Type contents of this directory as displayed on the monitor.

file-name-1←file-name-2↵ - Change the name of "file-name-1" to "file-name-2"

$CLEAN ↵ - Deletes files within this directory with qualifications given under $CLEAN in directory mode.

file-name $S ↵ - Gives display of first block of file-name* in form described under $Soooo ↵. Except that carriage-return returns to "file mode"; the description for $Soooo ↵ applies.

file-name $SB ↵ - Gives binary format display of first block of file-name.* Except that carriage-return returns to "File mode", the description for $SBoooo ↵ applies.

file-name = oooo ↵- Makes a new entry in current user directory with oooo as the first data block.

This feature should be used with caution as it is possible to introduce Y file errors to the file system.

The purpose of this feature is to allow recovery of files lost through typing errors to the editor.

@ - Enters or exits mode which displays every directory entry, whether used or not. Unused directory entries will be represented by "---".

* file name may be "#n" to operate on the n'th

CONT

4.4 ERROR MESSAGES:

o=Octal digit	d=Decimal number
u=User	f=File

? "bell"

Unacceptable command typed in, no action taken.

INVALID DEFAULT USER NAME

Default user name on disk not in master directory. The default user name is changed to "SYS".

LOGICAL CHECKSUM IN OLD TUT

Checksum of track usage table entries on disk is invalid. A new TUT is always written.

UNPROTECTED BLOCKS IN OLD TUT

Track usage table on disk indicates one or more blocks in file area are free when some file pointer chains to them. A new TUT is written.

UNAVAILABLE BLOCKS IN OLD TUT

One or more blocks are marked as unavailable in track usage table on disk when they are not part of any file. A new TUT is written.

```
              [ WORKING BLOCK
              [ AVAILABLE BLOCK
oooo CHECKSUM [ MASTER DIRECTORY
              [ TUT
              [ uuu USER DIRECTORY
              [ BLOCK d of uuu;ffffff ffffff
```

A hardware checksum error was found when block oooo was read from the disk. The checksum error persisted in re-reading the block 3 times but the data was consistent. The block was rewritten to eliminate the error.

"WORKING BLOCK" refers to the low-address blocks on the disk which are used for the working copy of DEBUG and the current program.

"AVAILABLE BLOCK" refers to an unused block in the file area of the disk.

"TUT" refers to the track usage table block on the disk.

```
                [ WORKING BLOCK
                [ AVAILABLE BLOCK
oooo DATA ERROR [ MASTER DIRECTORY
                [ TUT
                [ uuu USER DIRECTORY
                [ BLOCK d of uuu;ffffff ffffff
```

4.4 ERROR MESSAGES (cont.)

A hardware checksum error was detected all four times this block was read. Some inconsistancy in the data was found so no corrective action was taken.

oooo TRANSIENT CHECKSUM ERROR

A hardware checksum error was detected but a subsequent re-read occurred with no error.

oooo HIGH ORDER BITS IN CHAIN BLOCK d OF uuu;ffffff ffffff

The chain word of block oooo had bits on that were not part of a disk address. The block was rewritten with the high order bits removed from the chain word.

CIRCULAR FILE uuu;ffffff ffffff

A chain word was found that pointed to a previous block in the same file. The offending chain word was set to zero and the block was rewritten.

```
oooo BLOCK d OF uuu;ffffff ffffff
INVALID CHAIN POINTS TO oooo   [WORKING BLOCK
                               |MASTER DIRECTORY
                               |uuu USER DIRECTORY
                               [TUT
```

A chain word was found which contained a disk address not in the file area of the disk. The offending chain word was set to zero and the block rewritten.

```
uuu;ffffff ffffff Y'ED WITH uuu;ffffff ffffff (TAIL IS[BINARY)
                                                      [SYM.)oo
```

Two chain words from different files were found to be pointing to the same block. If a head had a name indicating its type is different from the tail, it was truncated at the point of convergence. If two binary heads chain to a symbolic tail, the address of the first block of the deleted tail is given as oooo.

APPENDIX H

III PROGRAM LIBRARY

III-109; STANDARD SUBROUTINE PARAMETERS

III-109

ABSTRACT

III-109 contains subroutine parameters that specify the FR 80 hardware and software configuration at assembly time.

GENERAL

Code will be assembled depending upon the definition and value of parameters in III-109. A parameter is defined by removing the slash in front of it; parameters with slashes are processed as comments by the assembler.

The following are the most commonly changed assembly parameters in III-109:

7TRACK - A value of zero means a 7-track drive is not available; a value of one means a 7-track drive is available.

9TRACK - A value of zero means a 9-track drive is not available; a value of one means a 9-track drive is available.

CAMNUM - The value of CAMNUM determines the camera to be used.

CAMNUM Value	Camera
1	35mm Unperforated
2	16mm Unperforated
3	35mm Perforated
4	16mm Perforated
5	Not used at this time
6	105mm Fiche

NUMCAM - The value of NUMCAM determines number of cameras that may be used. A number greater than one causes code to be assembled to facilitate camera changes at run time.

If the parameter MUMBLE has been defined before the assembler processes III-109, a summary of the hardware and software configuration will be printed on the Teletype.

APPENDIX I

III PROGRAM LIBRARY

OPERATING MONITOR:

III-166 INVAR

III-166

III-161

III-161 GO

APPENDIX I

III-166 INVAR

2. ABSTRACT

III-166 INVAR is inserted by III-166 and is part of MONITOR. The routines and their respective core positions are invarient; they remain the same for all application programs and are completely independent of machine configuration.

3. GENERAL

The invariant area starts at location 40 and is several hundred words long. This area is protected and is not reloaded when the operator loads a new program from a system tape. An example is ("LOAD/META ↓ ").

These routines and their absolute core positions are necessary to MONITOR. For example, the final tape read routines are located in INVAR to protect them from being stepped on by the incoming program.

III-166
III-161
III-161 GO

2. ABSTRACT

III-166, III-161, and III-161 GO are a set of subroutines making up the operating system called "MONITOR" which is resident in all application programs. These routines interpret and execute operator commands.

3. GENERAL

III-166 creates the MONITOR display of commands, communicates to the operator, and dispatches to the appropriate routines after interpreting a valid operator command.

Included in III-166 are general purpose routines that simplify FR 80 application programming. These routines allow numeric conversions, teletype communication, frame and page advance control, and initialization and resetting of FR 80 registers.

4. SUBROUTINES

4.1 Numeric and Teletype Routines

4.10 MOOUT

MOOUT converts the contents of the AC to octal and outputs the octal number to the teletype.

MOOUT is a conditional subroutine that exists only if BIGBUF=0. MOOUT calls MTTOUT which normally outputs to the teletype. (See MTTOUT description to display the octal number on the Monitor and PLS.)

Calling Sequence:

```
        .
        .
        .
    LAC NUMBER
    JMS MOOUT
        .
        .
        .
```

4.11 MDOUT

MDOUT converts the contents of the AC to decimal and outputs the decimal number to the teletype.

If BIGBUF≠0, then MDOUT = MNOUT. (See MOOUT and MNOUT descriptions.)

Calling Sequence:

```
      .
      .
      .
  LAC NUMBER
  JMS MDOUT
      .
      .
      .
```

4.12 MNOUT

MNOUT is an internal subroutine that converts the contents of the AC to radix n and outputs the number to the teletype.

If the programmer calls MNOUT, with BIGBUF=0, the following instruction must be a LAC (n) where $1 \geq n \geq 10.$ is the radix number.

MNOUT will only output a maximum of six digits to the teletype unless special assembly parameters are changed. For example, if the programmer wished to output the contents of the AC in binary (n=2) he should give MDNUMB the two's complement of the number of digits to be printed and increase MDUBUF table to accommodate the digits. (MDNUMB, 18. and MDOBUF, .+18./).

Failure to increase the table will result in inadvertently destroying subsequent code.

To output unsigned numbers, MNOUT+1 should be changed to a CLL.

To prevent zero suppression, MRADIX+5 should be changed to a NOP.

Calling Sequence (If BIGBUF=0)

```
      .
      .
      .
LAC NUMBER
JMS MNOUT
LAC (n)           /normally 8 or 10.
      .
      .
      .
```

Calling Sequence (if BIGBUF ≠0)

```
      .
      .
      .
LAC NUMBER
JMS MNOUT         /assumes n=10.
```

4.13 GETNUM

GETNUM is used in conjunction with other teletype and internal monitor routines. GETNUM converts numbers from the teletype buffer in six-bit character code format to octal and decimal numbers.

The procedure for entering MONITOR commands is to type the command, a slash (/), the necessary parameters separated by commas, and then a carriage return (↵). GETNUM assumes numerical parameters.

The internal MONITOR routines convert the typed parameters into 6-bit teletype (TTY) codes and then stores these codes into the teletype buffer area.

For example, the operator types "SETSIZE/1200, 6↵ ". Each digit of the parameter would be converted to TTY code with the TTY code for a comma separating both character streams. The final delimiter would be a binary zero in the buffer area.

GETNUM converts these character streams into a signed binary number and exits on a nonnumerical character code delimiter. GETNUM can convert both octal and decimal digits to a signed binary number. After the exit from GETNUM, OCTNUM contains the octal number and DECNUM contains the decimal number. The last character code that caused the exits from GETNUM (i.e., nonnumerical parameter) is in the AC.

The programmer should enter his MONITOR command in the monitor dispatch table (using the specified format). The program will jump to the programmer's subroutine after the carriage return is typed.

Calling Sequence:

```
        .
        .
        .
TEST,   JMS GETNUM
        LAC DECNUM      /assuming decimal input
        DAC SIZE
        .
        .
        .
```

GETNUM is called by GETINM and GETANM

4.14 MYESNO

MYESNO outputs a "YES" message to the teletype (TTY), if the link is clear, and a "NO" message if the link is set.

Calling Sequence:

```
       .
       .
 RAL
 JMS    MYESNO
       .
       .
```

4.15 KYBLIS

KYBLIS is the keyboard listen routine. JMS KYBLIS is usually inserted in the program at a recurring program loop to allow operation intervention with the running program.

If no character has been typed from the teletype (i.e., keyboard flag not set) then KYBLIS exits immediately. When the keyboard flag is set, there is a test to see if it is a control character. If DEBUG is defined, a control D, octal code 204, will cause a jump to DEBUG. A control I, octal code 211, causes the program to go to MONITOR. A control A, octal 201, will cause the program to go to MONITOR at the next frame advance. Any other characters are ignored by KYBLIS.

Typing "CONTINUE /↵ " causes the program to continue program execution. If in DEBUG, typing "JMP 1 KYBLIS $ X" continues the program.

Calling Sequence:

```
       .
       .
 LOOP,
       .
       .
       JMS KYBLIS
       .
       .
       JMP LOOP
```

4.16 ACCTG

ACCTG is called at BEGIN time and whenever MONITOR interrupts program execution.

ACCTG outputs a teletype carriage return and linefeed, the time, and the frame and picture numbers.

Calling Sequence:

```
        .
        .
        .
JMS ACCTG
        .
        .
        .
```

4.17 MCRLF

MCRLS outputs a teletype carriage return and linefeed.

Calling Sequence:

```
        .
        .
        .
JMS MCRLF
        .
        .
        .
```

4.18 TIMOUT

TIMOUT outputs a time message to the teletype. Cells 6 and 7 are used by our internal MONITOR routines to store and increment time data. The outputted time message gives hour, minute, and second information.

Calling Sequence:

```
        .
        .
        .
JMS TIMOUT
        .
        .
        .
```

4.2 FR 80 REGISTER ROUTINES

4.20 PSTLL

PSTLL remains in a two instruction loop until the character, vector, and point plotting generators are not busy. This prevents jamming the FR 80 registers before the previous FR 80 command has finished.

```
   Calling Sequence:
             .
             .
        PSTLL
             .
             .
```

4.21 MNSPOT

MNSPOT loads the FR 80 spot size register. The AC must contain the desired spot size (0-7) in the low order three bits.

If the spot size is different then a delay loop of 50 milliseconds is initiated to allow sufficient time for the beam to change size.

```
   Calling Sequence:
             .
             .
      LAC SIZE
       MNSPOT
             .
             .
```

4.22 SETOMU

This subroutine loads certain FR 80 registers to initialize the optical mechanical unit. SETOMU executes a PSTLL, does a RST (RESETT), loads spot size, intensity, rotation, character spacing (horizontal and vertical), and character size registers with programmer determined values.

The contents of RECSPT, RECPIN, CHRROT, CHDELX, CHDELY, and CHRSIZ are used to load the spot, intensity, rotation, x spacing, y spacing, and size registers respectively.

```
   Calling Sequence:
               .
               .
         SETOMU
               .
               .
```

4.23 SETPLS

SETPLS calls SETOMU and unblanks the PLS (precision light source), if PLSON contains a NOP. If BIGBUF≠0, SETPLS always unblanks the PLS.

SETPLS should be called at BEGIN time, and whenever the program needs to reinitialize the FR 80 hardware registers.

```
   Calling Sequence:
               .
               .
  BEGIN,
               .
               .
          SETPLS
               .
               .
```

4.24 SETXY (DAC SETTING ROUTINE)

There are three DAC setting routines which differ by their respective delay loops. SETXY sets the X and Y DAC's with no delay, SETXYF has a 30 microsecond delay, and SETXYS has a 120 microsecond delay. The SEXTY routines set the DAC's by using two internal routines, XXXXXX and YYYYYY, which are initially set at assembly time, and changed by calling ROTATR at run time. Thus a programmer can initially set up ROTCOM, call ROTATR, and then ignore rotation when setting the X and Y coordinates.

These routines assume that the desired X and Y coordinates immediately follow the call to SETXY in this specified format.

```
Calling Sequence:
        .
        .
        .
 JMS SETXY         /or JMS SETXYS or JMS SETXYF
 LAC XCORD         /14 low order bits used
 LAC YCORD
        .
        .
        .
```

Use of a delay ensures that the electron beam has sufficient time to settle at the new coordinates before a VGO or CHGO is initiated.

4.25 ROTATR

ROTATR is a subroutine that resets CHRROT (CHRROT's contents are loaded into the rotation register) and resets the DAC setting routines.

ROTCOM must contain an OPR for comic mode and a SKP for cine mode.

Calling Sequence:

```
         .
         .
         .
LAC (OPR)          /or LAC (SKP)
DAC ROTCOM
JMS ROTATR
         .
         .
         .
```

4.26 INTENS

INTENS is an intensify point subroutine. INTENS assumes that the X and Y DAC registers have been set to the desired coordinates.

This routine calls PSTLL and then executes an INTS instruction. The beam is intensified on the monitor screen, and on the PLS if previously enabled by an UNBL command.

The beam uses current spot size and intensity values and intensifies for two microseconds. The one's complement of the number of hits (intensifications) is stored in PTHITS.

Calling Sequence:

```
         .
         .
         .
 LAM-10            /10 hits
DAC PTHITS
  INTENS
         .
         .
         .
```

4..3 ADVANCE ROUTINES

4.30 ADVSYS

ADVSYS advances the camera the number of increments that is contained in the AC. ADVSYS first calls PSTLL to ensure that the vector and character generator are not busy.

ADVSYS advances the camera in increments, where m= contents of the AC. If m> n (n=maximum number of increments allowable for each camera advance for the specified camera), then ADVSYS advances n increments at a time until m increments are executed. PULMAX contains n which is specified at assembly time. n=4 for sprocketed camera and 8 for unsprocketed cameras.

After each camera advance the SFNA IOT is executed to ensure that the film is not moving before further program processing.

Calling Sequence:

```
      .
      .
      .
   LAC (12.)        /m=12.
   ADVSYS
      .
      .
      .
```

4.31 ADVANN

ADVANN is a general purpose camera advance routine for advancing the film n increments, where n is contained in the AC.

After checking that there is more than ten feet of film left in the supply magazine, ADVANN calls ADVSYS.

If there is insufficient film left, ADVANN goes to MONITOR to wait for further operation instructions.

Calling Sequence:

```
       .
       .
    LAC (n)
   ADVANN
       .
       .
```

4.32 ADVAN

ADVAN is a general purpose camera advance routine which advances the camera one pulldown. A pulldown is defined as n increments, where n is large enough to advance the exposed film past the PLS.

PULLNO contains n and is automatically set at assembly time to give the correct advance. PULLNO can be changed in MONITOR by selecting a new camera or by typing "PULLDOWN/n ↵ ". "n" should be ≥ 3 for an unsprocketed camera and ≥ 4 (must be a multiple of 4) for a sprocketed camera.

ADVAN increments PICNUM (if MANYUP is not defined) and FRAMNM. ADVAN calls ADVANN and ADVANF.

Calling Sequence:

```
       .
       .
   ADVAN
       .
       .
```

4.33 ADVANF

ADVANF is the accounting advance routine that is called by ADVAN to update the frame number (FRAMNM).

ADVANF also increments NUMFRM to see if "frames are done." If NUMFRM becomes zero then ADVANF outputs a "FRAMES DONE" message and goes to MONITOR.

If a "control A" was typed, ADVANF also goes to MONITOR. Typing "CONTINUE /↗ " continues program execution in both cases.

4.34 CLEAR

CLEAR advances the exposed film past the film gate to ready the takeup magazine for film processing. CLEAR calls ADVSYS.

```
Calling Sequence:
          .
          .
          .
      CLEAR
          .
          .
          .
```

4.35 FRSPIC

FRSPIC is always defined but is relevant only if MANYUP or STRIPM is defined. FRSPIC is usually called at BEGIN time to initialize parameters necessary for multiple images per frame. (In stripfiche mode each strip is considered a frame.)

FRSPIC forces a frame advance (strip advance in stripfiche mode), with appropriate reinitializations, upon the first call to NEXPIC.

4.36 NEXPIC

NEXPIC is a general purpose page and frame advance routine. NEXPIC is always defined but degenerates to an ADVAN if MANYUP or STRIPM is not defined.

NEXPIC updates X and Y images per frame counters to determine the appropriate action to take on each page and frame advance. When doing multiple images per frame, the camera is advanced only after the multiple image requirement is satisfied. For example, the operator specifies with MONITOR commands two images in X and two images in Y. FRSPIC will initialize NEXPIC to advance the film and position the DAC's for the first image. The second call will update the X coordinate, the third call updates the Y coordinate and reinitializes the X coordinate. The fourth call will update the X coordinate. The fifth call will cause a frame advance and reset both X and Y coordinates for a new frame, etc.

EXAMPLE

```
At begin time       FRSPIC
                    NEXPIC          /frame advance
                    DATA FRAME
                    NEXPIC          /2nd call
                    DATA FRAME
                    NEXPIC          /3rd call
                    DATA FRAME
                    NEXPIC          /4th call
                    DATA FRAME
                    NEXPIC          /5th call - frame
                                       advance
                    etc.
```

Calling Sequence:

```
          .
          .
          .
      NEXPIC
          .
          .
          .
```

APPENDIX J

III PROGRAM LIBRARY

III-183; DISK I/O

DISK I/O III-183

ABSTRACT

III-183 is a set of subroutines which provide capability for disk input/output. A push down stack is provided for nested reads.

GENERAL

The disk organizatio1 is described completely in the "Disk Operating System" documentation. The disk has 1024. blocks of 256. 18-bit words. Resident on the disk is a master directory which indexes up to 15. user directories, and a track usage table (TUT) which indicates the state (used or unused of all blocks on the disk.

Data is read from the disk or written to the disk in blocks of 256. 18-bit words; however the disk buffering routines make the block structure invisible to the user.

A disk file is referenced by a directory name and a file name. The subroutines in III-183 assume that the directory name is in location DKFILE and that the file name is in locations DKFILE+1 through DKFILE+4.

SUBROUTINES

DKRINI - This subroutine initializes the system to read from the file specified by DKFILE through DKFILE+4.

Calling Sequence:

```
      .
      .
      .
  DKRINI
 RETURN          /DKRINI returnes here if the file
                  doesn't exist.
 RETURN+1        /normal return for DKRINI
      .
      .
      .
```

DKWINI - This subroutine initializes the system to write on the disk by finding an available block, setting up the buffer, and storing the default directory and file name into locations DKFILE through DKFILE+4. Control is returned to DEBUG if the disk is full.

Calling Sequence:

```
   .
   .
   .
DKWINI
   .
   .
   .
```

DKREAD - This subroutine returns to next 9 bit byte in the low order AC from the file opened by DKRINI.

Calling Sequence:

```
   .
   .
   .
DKREAD
SAD (EOFCHR         /check for EOF character
```

DKRDWD - This subroutine returns the next 18 bit word in the AC from the file opened by DKRINI.

Calling Sequence:

```
   .
   .
   .
DKRDWD
   .
   .
   .
```

DKWRIT - This subroutine writes the 9 low order bits of the AC to the disk.

Calling Sequence:

```
   .
   .
   .
DKWRIT
   .
   .
   .
```

DKWRWD - This subroutine writes the contents of the AC to the disk.

Calling Sequence:

```
      .
      .
      .
 DKWRWD
      .
      .
      .
```

DKNAME - This subroutine names all disk output since the last DKWINI. Locations DKFILE through DKFILE+4 specify the directory name and file name.

Calling Sequence:

```
        .
        .
        .
DKNAME
RETURN          /DKNAME returns here if there is no
                /such directory and DKNWSR is not
                /defined, or if the master directory is
                /full.
RETURN+1        /DKNAME returns here if the user
                /directory is full.
RETURN+2        /normal return for DKNAME.
```

DKPUSH - This subroutine saves all relevant information about the last file opened by DKRINI. This allows reading from a new file and later reopening the old file by doring a DKPOP. Extra core locations of interest to the user program may be pushed onto the stack by defining DKPNUM to be the number of extra words desired. The extra core locations are DKPBLK to DKPBLK+ DKPNUM, for non-zero DKPNUM.

Calling Sequence:

```
         .
         .
         .
  DKPUSH
  RETURN                  /push down stack is full.
 RETURN+1                 /normal return for DKPUSH
         .
         .
         .
  DKRINI                  /open a new file.
         .
         .
         .
```

DKPOP - This subroutine reopens the last file processed by DKPUSH. The read routines will continue where they left off in the file.

Calling Sequence:

```
         .
         .
         .
    DKPOP
    RETURN                /push down stack is empty
    RETURN +1             /normal return for DKPOP.
         .
         .
         .
```

The following subroutines are called by those listed above.

DKRTUT - This subroutine reads the track usage table into core and stores the default directory name and file name in DKFILE through DKFILE+4.

DKCRUS - This subroutine creates a new directory with the name in DKFILE. A new master directory is written on the disk. The routine skips on a normal return.

DKRDMD - This subroutine reads the master directory into core.

DKDRFN - This subroutine searches the master directory for the name in DKFILE. The routine skips on a normal return.

DKFFIL - This subroutine searches the user directory for the name in DKFILE+1 through DKFILE+4. The routine skips on a normal return with a pointer in the AC to the first block of the file.

DKDLET - This subroutine deletes the file named in DKFILE through DKFILE+4 from the disk.

DKGET - This subroutine searches the track usage table for an available block. Control is returned to DEBUG if the disk is full.

DKINIT - This subroutine initializes to read into core the block whose number is in the AC.

DKRDSK - This subroutine handles the actual data transfer from disk to core.

DKWRBK - This subroutine handles the actual data transfer from core to disk.

ASSEMBLY PARAMETERS

NODKWT - When NODKWT is defined, it is not possible to write on the disk.

DKNWSR - When DKNWSR is defined, it allows new directories to be created.

NODKRD - When NODKRK is defined, it is not possible to read files from the disk.

DKREPL - When DKREPL is defined, it allows file replacement.

DKPNUM - When DKPNUM is defined, it allows "push-reads." Define it as the number of extra words (usually zero) to be pushed.

DKCHAN - When DKCHAN is defined, it allows a new TUT to be built when using subroutine DKNAME. Store a LAM (-0) in location DKCHAN when calling DKNAME to cause a new TUT to be built.

ADDITIONAL NOTES

III-182 reads a file name and formats it correctly for use with III-183.

A useful macro for producing file names in the proper format directly is.

```
N X, Y, Z - macro to pack characters X, Y, Z into name format.
. DEF N X, Y, Z
"X+40&77←6!<"Y+40&77>←6!<"Z+40&77>
. TERM
```

APPENDIX K

III PROGRAM LIBRARY
III-162; VECTOR ROUTINES

VECTOR ROUTINES III-162

ABSTRACT

III-162 provides subroutines for drawing solid vectors, dashed vectors, and dotted vectors.

GENERAL

The starting point of a vector is referred to as the "head"; the end point is called the "tail." The macro SETHD is used to set the coordinates of the head; the macro SETTL sets the coordinates of the tail. Specification of a null (zero-length) vector results in the intensification of a single point.

For a vector from (XHD, YHD) to (XTL, YTL) the macro coding should be:

```
SETHD XHD, YHD
SETTL XTL, YTL
```

The following subroutines assume that the head and tail coordinates have been set with the SETHD, SETTL macros.

SOLID VECTORS

Subroutine DRWVEC uses the hardware vector generator to produce solid vectors. Location VECHIT contains the 1's complement of the number of hits for each vector.

Calling Sequence:

```
     .
     .
     .
JMS DRWVEC
     .
     .
     .
```

DASHED VECTORS

Subroutine DRWVDS uses the hardware vector generator to produce dashed vectors. Dashes are made by blanking and unblanking the PLS while the vecotr generator is drawing. The dashed line format is specified by a call to COMDSH before calling DRWVDS.

COMDSH Calling Sequence:

```
STL or CLL
JMS COMDSH               /compile dashed line code.
LAC "SMALLER LENGTH"     / argument 1
LAC "LARGER LENGTH"      / argument 2
```

If the link is zero, then argument 1 refers to the length of the dash and argument 2 refers to the length of the space.

If the link is one, then argument 1 refers to the length of the space and argument 2 refers to the length of the dash.

It is necessary to call COMDSH only once for a particular dashed line format.

Any change in format requires another call to COMDSH.

A call to DRWVDS utilizes the output of COMDSH and draws the dashed vector. Location VECHIT contains the 1's complement of the number of hits for each vector.

Calling Sequence

```
    .
    .
    .
JMS DRWVDS
    .
    .
    .
```

DOTTED VECTORS

Subroutine DRWDOT produces dotted vectors without the use of the hardware vector generator. The assembly parameter DSH determines the spacing between dots. The number of scope points between dots, as measured on the axis of the larger component, is 2^{DSH}. Location DOTHIT contains the 1's complement of the number of hits for each vector.

Calling Sequence:

```
    .
    .
    .
JMS DRWDOT
    .
    .
    .
```

ASSEMBLY PARAMETERS

DASHED - When DASHED is defined, routines will be assembled to produce dashed vectors.

DOTVEC - When DOTVEC is defined, routines will be assembled to produce dotted vectors.

DSH - When DOTVEC is defined, DSH determines the number of scope points, X, between dots, as measured on the axis of the larger component. $X = 2^{DSH}$

APPENDIX L

III PROGRAM LIBRARY

III 163; MAGNETIC TAPE ROUTINES

MAGNETIC TAPE ROUTINES III-163

2. ABSTRACT

This insert file contains subroutines for initialization, reading, writing, or repositioning of single or multiple tape units.

3. GENERAL

The read routines can access data in groups of bits or in words. The write routines take words and transfer the data to tape. There is also a set of subroutines to save the magnetic tape status and at a later time reposition the tape back to that point. The reposition routines can handle nested calls for backing up the tape. By assembly option the read routines can be single or double buffered.

4. CONDITIONAL PARAMETERS

1. Single or Double Buffering. If the symbolic parameter TWOBUF==1 is defined, the read routines will be double buffered; if it is not defined,the read routines will be single buffered.

2. Get Bits Subroutine (GETT see Para. 6). This routine will not be assembled if the symbolic parameter MTWRDS==1 is defined.

3. Use of an Auto-Index Register for the Read Routine. If the symbolic parameter MTPTR is not defined, the read routines will use a core location for indexing through the data buffer. If the user desires a faster access time and can afford the dedicated use of an auto-index register, define MTPTR==n

where n= 10 through 17.

4. Repositioning Routines. If these routines are desired, define MTRPT==m where n= the maximum level of nesting.

5. Write Routines. If the user desires write routines, two symbolic parameters must be defined:

```
MTWRIT==1
MTMANY==n          n= number of tape units -1
```

5. INITIALIZATION OF READ ROUTINES

a. Single Buffered

```
LAC     (Buffer-1
DAC     MTAREA
LAC     LENGTH          /buffer length in two's complement
DAC     PBUFSZ
JMS     MTRINI
```

b. Double Buffered

```
LAC     (BUFFER-1
DAC     CURBUF#
LAC     LENGTH          /1/2 total buffer length in two's
DAC     PBUFSZ          complement
CMA
ADD     CURBUF
DAC     NEXBUF#
JMS     MTRINI
```

If both 7 and 9-track drives are available, the location MT9SW will contain a NOP if the data was read from a 9-track drive.

6. GET BIT ROUTINE

By utilizing the macro instruction

```
        GETT    n 1≤n≤18.
```

The user can access data in a bit by bit basis. If the data is known to cross record boundaries,MTBYSW should contain a SKP. If not, the partial word at the end of the record will be ignored.

The GETT n macro expands to the following:

```
        LAW     n
        JMS     MTBYTE
        AND     (1←<N-1>-1+<1--<N-1>>)
```

7. GET WORD ROUTINE

If the user desires only to access data in 18-bit word formats, this subroutine will access data sequentially from the tape buffer.

```
        JMS     MTLAC
```

NOTE: The GETT macro and the word routine can not be intermixed.

If this method is utilized, the symbolic parameter MTWRDS==1 should be defined to save program space.

8. REPOSITIONING ROUTINES

These routines consist of a push-down list and backup routines to process nested repeats of data from mag tape.

1. Push List

```
        JMS             MTPUSH
```

This saves the current status of the mag tape pointers.

2. Pop List

```
        JMS             MTPOP
```

This removes the top mag tape pointers from the push down list.

3. Backup Tape

```
        JMS             MTREPO
```

This routine backs the tape up to the previous push position.

9. WRITE ROUTINES

If these routines are utilized the mag tape read routines should not be double buffered. The initialization sequence changes when the write routines are included.

Initialization:

```
LAC     WRUNIT          /write unit number
JMS     MTWINI
LAC     LENGTH          /length of write buffer two's comple-
                        ment
LAC     (WRBUF-1)       /buffer address -1
LAC     RDUNIT          /read unit number
JMS     MTRINI
LAC     LENGTH          /length of read buffer two's comple-
                        ment
LAC     (RDBUF-1)
```

Unit Select:

```
LAC     UNIT            /see figure below
JMS     MTSEL
```

		Not Used
0 1 2	3 4 5 6 7 8 9 10 11	12 13 14 15 16 17

Bits	0-2	Unit Number	
	3	Parity	0=even 1=odd
	4	Core Dump	
	5	Long Gap	
	6-8	Command	
	9	Interrupt Enable	
	10-11	Density	

Command	
0	NOP
1	Rewind
2	Read
3	Read/Compare
4	Write
5	Write EOF
6	Space Forward
7	Space Reverse

Density	
0	200 BPI
1	556 BPI
2	800 BPI
3	800 9-Track

10. Write

To place a word in the write buffer

```
        JMS         MTDAC
```

will transfer the contents of the accumulator into the write buffer, when the buffer is full it will be written to the selected unit.