MAGNETIC TAPE SYSTEM

HEWLETT hp PACKARD

11000 Wolfe Road
Cupertino, Calif. 95014

HP 02116-91752

April, 1970

First Edition, Aug. 1969
Revised, April 1970

Printed in the U.S.A.

Second Edition

PREFACE

The Hewlett-Packard Magnetic Tape System (MTS) provides a flexible framework for the usage of absolute and relocatable programs stored on magnetic tape. The use and operating procedures for an existing Magnetic Tape System are described in this book, but the steps required to create MTS are described in a companion volume, *PREPARE TAPE SYSTEM*.

This publication, in conjunction with the *PREPARE TAPE SYSTEM* manual (02116-91751), supersedes the MAGNETIC TAPE SYSTEM manual which was published in August 1969. Information from the preceding manual which concerned system preparation has been moved to *PREPARE TAPE SYSTEM;* all other information has been incorporated into this publication. The new chaining feature is described in Section IV.

Topics covered in this text are:

Introduction - MTS Hardware/Software

Section I - Organizational Overview of MTS and its Elements

Section II - Operating Procedures for an Existing MTS

Section III - User Requests to MTS

Section IV - Absolute, relocatable, and conversational programming; MTS interface with ALGOL, FORTRAN, BASIC, and Assembly Language; Editing

Section V - MTS Usage

Appendix A - Samples of Prepare Tape System and Prepare Control System

Appendix B - Programming Techniques

Appendix C - Stand-Alone Environment

Glossary

Index

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Certain conventions have been used to increase readability:

italics are used for symbolic items (in format descriptions) and for emphasis.

[ ] brackets are used to enclose optional items in format descriptions.

The reader should also be familiar with other software systems that he plans to include in the Magnetic Tape System. These specific systems have been documented in other Hewlett-Packard manuals:

Title	HP Number
FORTRAN	02116-9015
HP BASIC	02116-9077
ALGOL	02116-9072
ASSEMBLER	02116-9014
SYMBOLIC EDITOR	02116-9016
BASIC CONTROL SYSTEM	02116-9017
PROGRAM LIBRARY	02116-9032

CONTENTS

INTRODUCTION

The Hewlett-Packard Magnetic Tape System (MTS) provides a simple vehicle for quickly loading software programs such as the FORTRAN Compiler or BCS Relocating Loader into core memory. The Magnetic Tape System is created by transferring software programs from paper tape to a magnetic tape. In the magnetic tape environment, programs are loaded into core automatically by a supervisory program, .IPL., that operates in response to user requests.

SOFTWARE IN AN MTS-ENVIRONMENT

The following HP software is able to operate in the MTS environment without modification:

- FORTRAN Compiler
- ALGOL Compiler
- Symbolic Editor
- BASIC Interpreter
- Extended Assembler
- Cross-Reference Symbol Table Generator
- BASIC Control System
  - .IOC. (non-buffered)
  - Relocating Loader
  - BCS Drivers
- Relocatable Program Library
- SIO Drivers

MTS is capable of carrying out standard programming operations easily and efficiently. The programmer may edit source programs to magnetic or paper tape, then compile the source programs into relocatable or absolute object programs. Absolute object programs can be added to the Magnetic Tape System before exectuion. Relocatable object programs are executed by loading them with the BASIC Control System (in or out of MTS). The BASIC Control

System can load programs directly into core for immediate execution or can produce an absolute tape.

SYSTEM GENERATION

The Magnetic Tape System is generated using three other software programs:

- PTS (Prepare Tape System), a file generator program that creates the magnetic tape containing the HP software programs (and any user programs).
- .IPL. (Inter-Pass Loader), a supervisory program that controls loading of programs from magnetic tape into core.
- MTS Bootstrap, an independent program which consists of a standard input/output module (S.SIO), a magnetic tape SIO driver, and MTS Boot. The Bootstrap initiates operation of MTS.

Once MTS is configured, it consists of two parts: a bootstrap paper tape and a system magnetic tape. The magnetic tape is organized into program files:

- File 1 contains absolute programs, such as FORTRAN or user programs, that are loaded into core by .IPL., and
- File 2 contains subroutines, such as those of the Relocatable Program Library, that may be linked by the Relocating Loader to any relocatable user program that requires them.
- File 3 scratch file area.

The balance of the magnetic tape is available to executing programs for storage of temporary data and scratch use.

HARDWARE ENVIROMENT

MTS requires the following minimum hardware equipment:

- HP computer with 8K memory,
- HP Magnetic Tape Unit:
  HP 2Ø2Ø, or
  HP 3Ø3Ø (~~requires 2-channel DMA~~ DRIVER ASSUMES DMA channel; not available on 2114 computer),
- System console:
  HP 2752A Buffered Teleprinter, or
  HP 2754B Buffered Teleprinter

A Batch Input Device is required for Batch Processing:

HP 2761A-Ø7 Mark Sense Card Reader (recommended), or
HP 2737A Punched Tape Reader, or
HP 2748A Punched Tape Reader, or
HP 2758A Punched Tape Reader

The following devices may be added to increase operating flexibility, convenience, and speed:

HP 2737A Punched Tape Reader (in addition to card reader), or
HP 2748A Punched Tape Reader, or
HP 2758A Punched Tape Reader,
HP 2753A High-Speed Tape Punch,
HP 2778A Line Printer,
Additional 8K of core memory (only on 2116 computer).

SECTION I

SYSTEM ORGANIZATION

A configured Magnetic Tape System (MTS) consists of a magnetic tape divided into two program files and a scratch area; and a control area of core divided into an I/O control area and .IPL., an inter-pass loader. (See Figure 1-1.) The area of core up to 15777_8 (8K) or 35777_8 (16K) is available for programs.

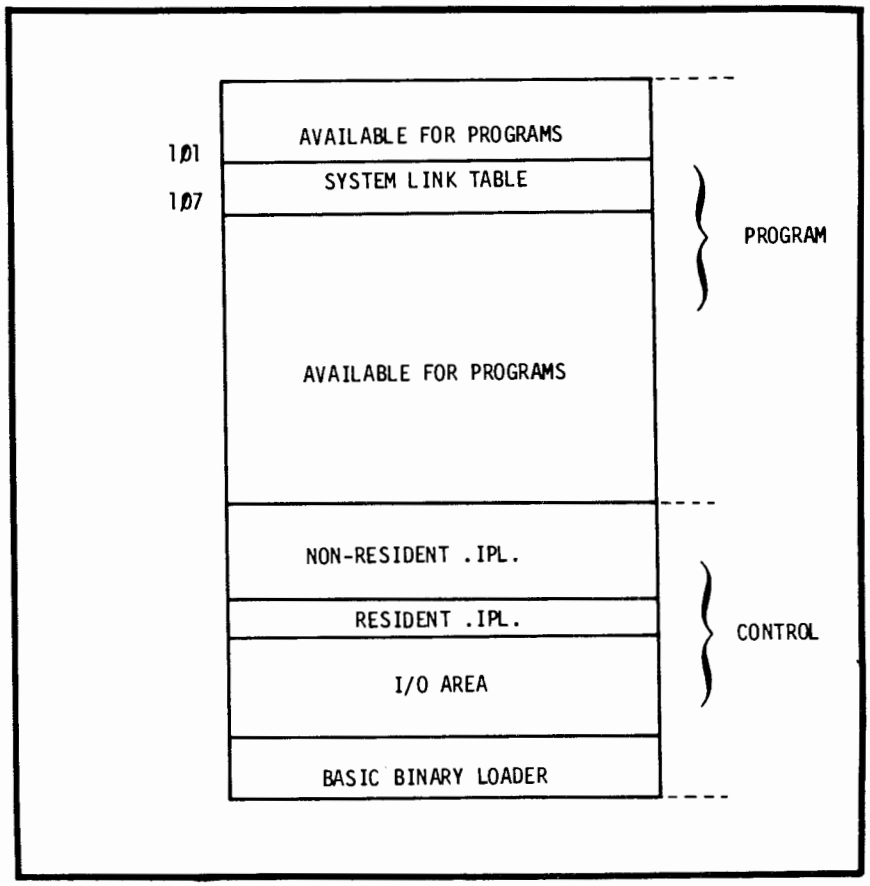

Figure 1-1. General View of MTS Core Memory

INTER-PASS LOADER

The inter-pass loader (.IPL.) is divided into two parts: a small, core-resident portion and a larger, tape-resident portion. The basic function of .IPL. is to load one or more programs from File 1 of the magnetic tape into core memory. The programs are specified by the user through identifiers assigned when MTS is generated.

As it loads the programs, .IPL. examines their starting addresses and links the programs back to .IPL.. (See Section IV.) .IPL. transfers control to the starting address of the first program specified. When the program has run to completion, it makes an .IPL. call (Section IV), which either handles the next user request or loads another program that is specified in the call. In this way, programs can be chained together without operator or programmer intervention.

I/O ORGANIZATION

The I/O control area of core is divided into a core-resident SIO magnetic tape driver and an SIO module area. (See Figure 1-2.) The magnetic tape driver handles input from and output to the magnetic tape unit.

The SIO (Software Input/Output) module area is an overlay area for tape-resident SIO modules. An SIO module is a combination of up to four SIO drivers which resides in File 1 as an absolute program (and has an identifying name). SIO drivers are available for the teleprinter, card reader, line printer, high-speed punch, and paper tape reader.

SIO modules always include drivers to handle keyboard input, list output, paper tape or card input, and tape punch output. Each SIO module is constructed starting with the teleprinter driver. Then, if a line printer driver is added, it replaces a portion of the teleprinter driver for the list and punch output function. If a paper tape reader driver is added, it replaces the paper tape input of the teleprinter. If a card reader is added, paper tape input is eliminated and card input is added. If a high-speed tape punch driver is added, it replaces the teleprinter's tape punch function. However, even when all possible drivers are included, the keyboard input function of the teleprinter remains in every SIO module.

When MTS is configured, an SIO module must be chosen as the standard SIO module and named S.SIO. It cannot contain a line printer driver. It is loaded into core by .IPL. between every user request. (See Section III.) All other SIO modules included in the system are non-standard SIO modules and are loaded into core only when specifically requested by name.

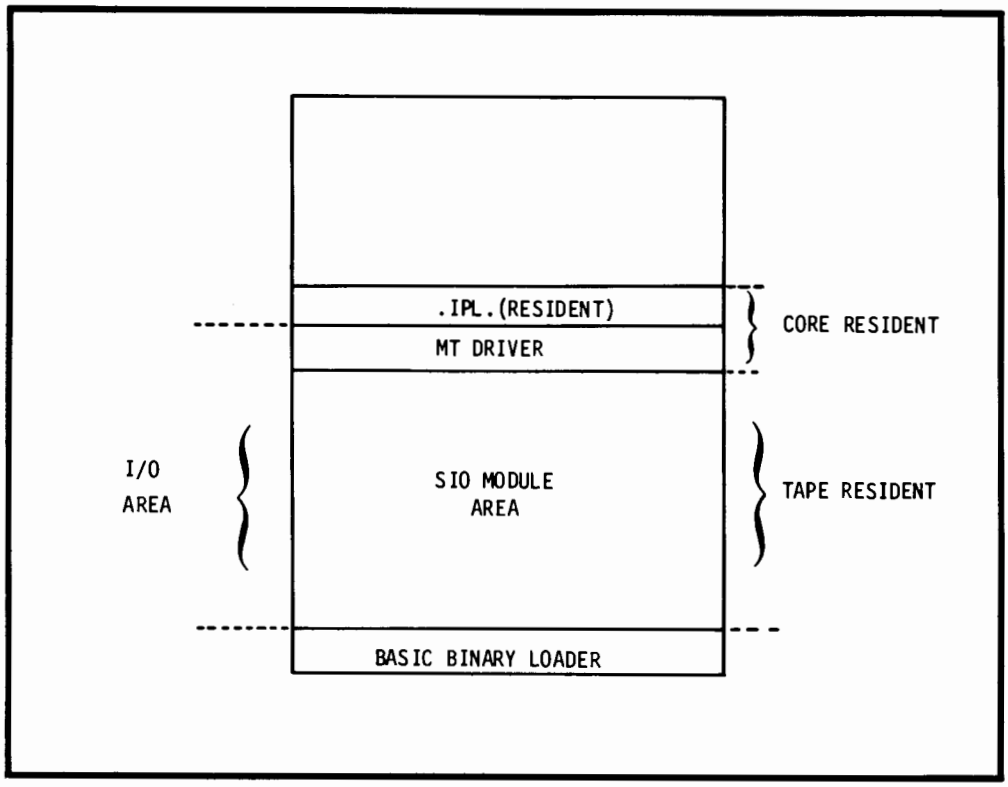

Figure 1-2. SIO Modules

System Link Table

Locations 101_8 through 107_8 comprise the system link table. During operation, this table includes pointers to the entry points of the SIO drivers currently in core, for keyboard input (always the teleprinter), tape punch output, paper tape input or card input, and list output. In addition, it has pointers to the magnetic tape driver and the first and last word addresses of available memory. (See Figure 1-3.)

Absolute Program Input/Output

Those absolute programs from File 1 which are generated by the Assembler use the system link table to make I/O requests of the SIO drivers currently in core. Each driver controls one device of one type on one or more I/O channels.

By chaining, absolute programs may replace the SIO module currently in core with another one from File 1. In this way, programs can modify their I/O capabilities dynamically and in particular, one program may use both card and tape input (which is not possible with only one SIO module). Section IV describes chaining in detail.

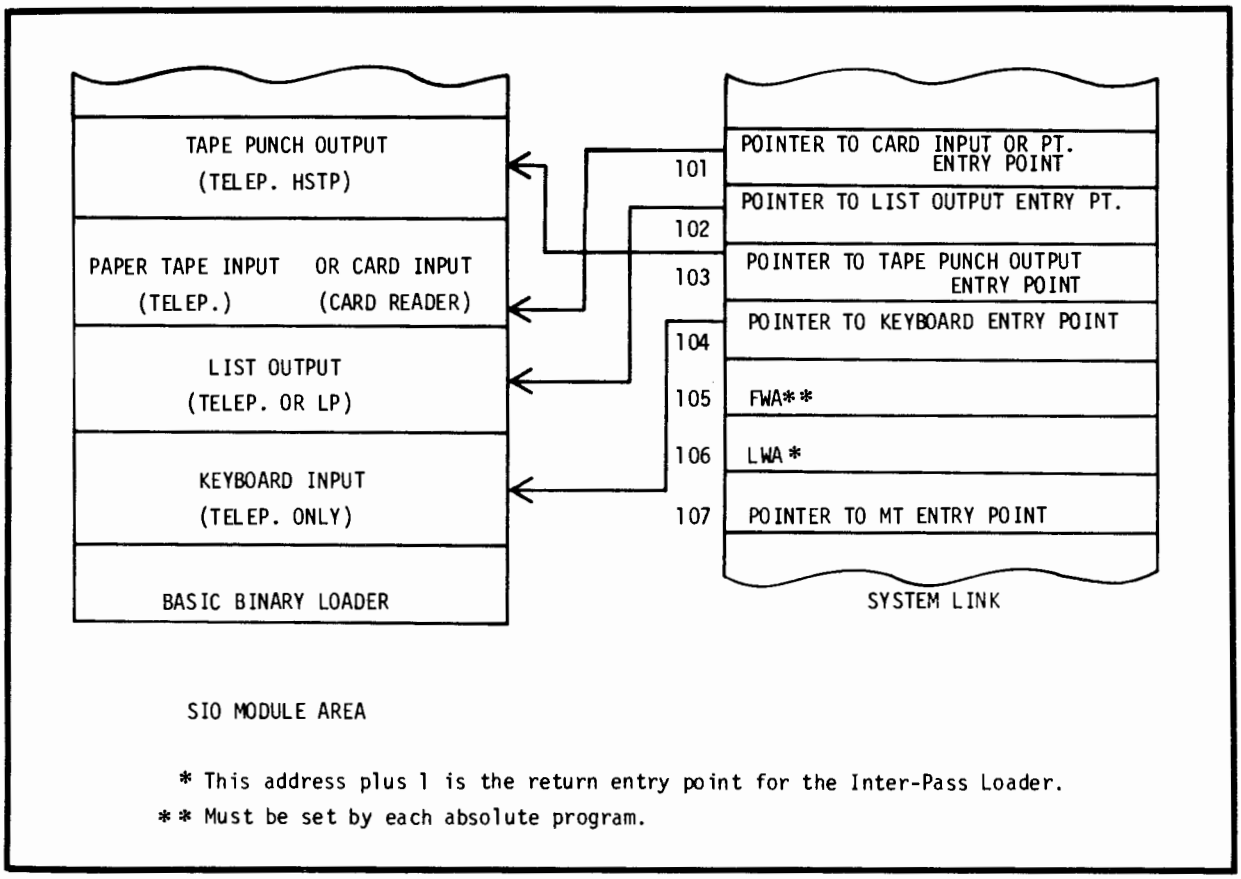

Figure 1-3. System Link Table

Relocatable Program Input/Output

Relocatable object programs produced by the Assembler, FORTRAN compiler, or ALGOL compiler must run under control of the Basic Control System. BCS relocates the programs into fixed locations and links them to its BCS drivers. Thus, relocatable programs do not use SIO drivers, but BCS drivers which occupy the core area below .IPL..

Figure 1-4 shows how .IPL., the I/O control area, the system link table, and the absolute program area are interrelated in core memory.

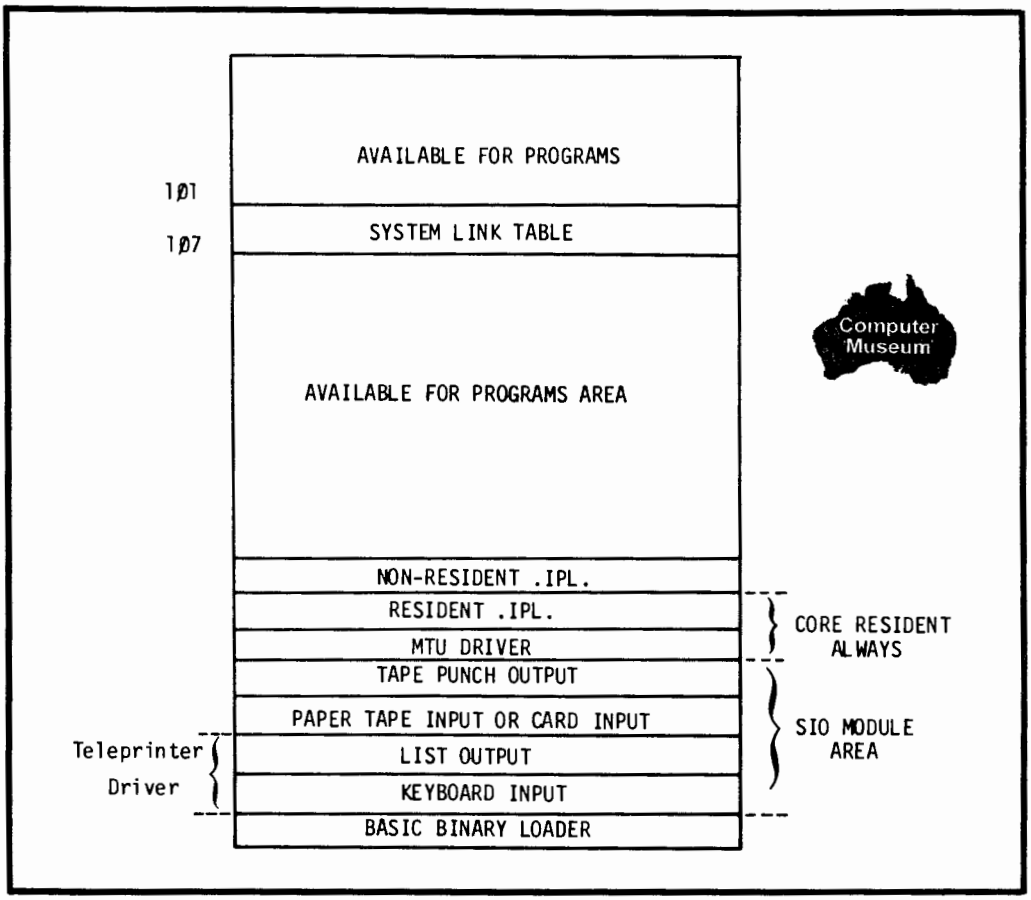

Figure 1-4. Detailed View of MTS Core Memory

USE OF FILE 1

File 1 of the magnetic tape always contains S.SIO, the standard SIO Module and .IPL., the inter-pass loader. In addition, it may contain any of the HP software programs mentioned in the Introduction, user absolute programs (generated by the Assembler or the BCS Relocating Loader), and non-standard SIO modules.

Programs in File 1 are loaded and run by .IPL. in response to user directives (these directives are explained in Section III). Figure 1-5 shows the sequence of operation:

1. A directive within the input stream is accepted by .IPL.;

2. .IPL. picks up the program identifiers in the directive;

3. .IPL. locates the programs in File 1 of the magnetic tape;

4. The programs are loaded into memory in the order they are requested, and .IPL. transfers control to the starting address of the first program loaded;

5. The program reads in data from the input stream;

6. The magnetic tape scratch area is available for temporary storage of data;

7. The program may produce some paper tape or list output; and

8. The program makes an .IPL. call to terminate.

For example, the directive may specify the ALGOL compiler and a non-standard SIO module to be loaded. The ALGOL Compiler reads in the source program and outputs a relocatable binary object program on paper tape.

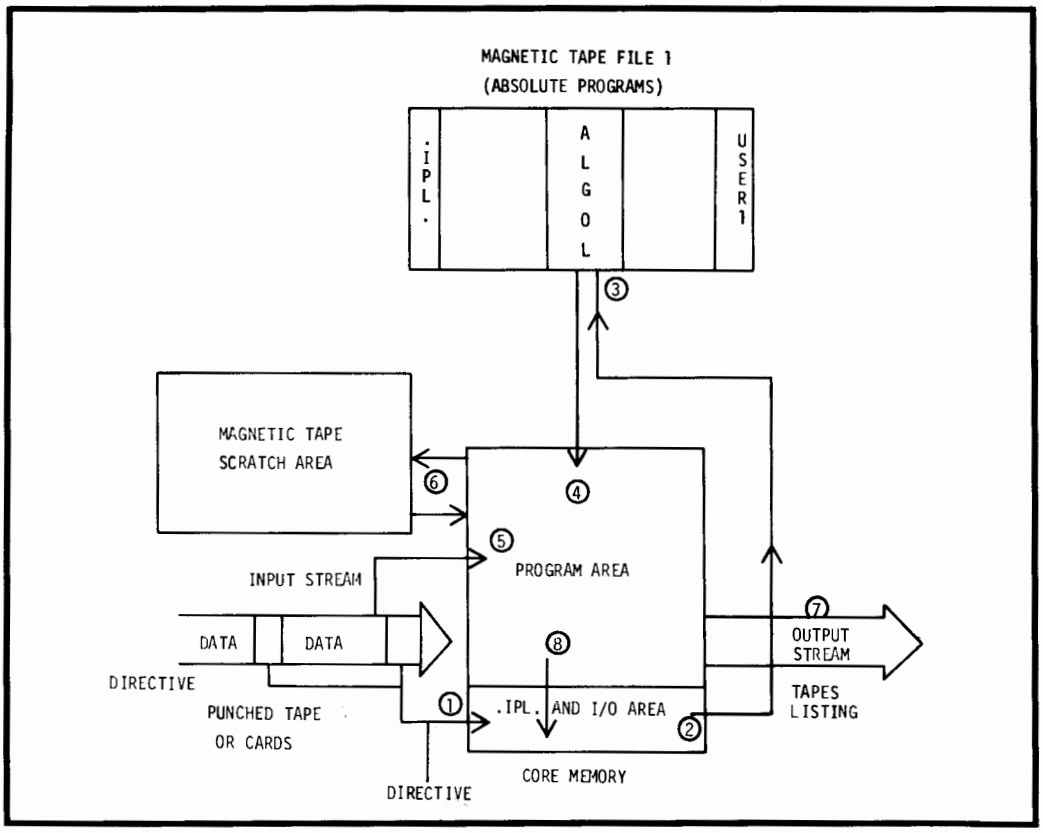

Figure 1-5. Execution of Programs in File 1

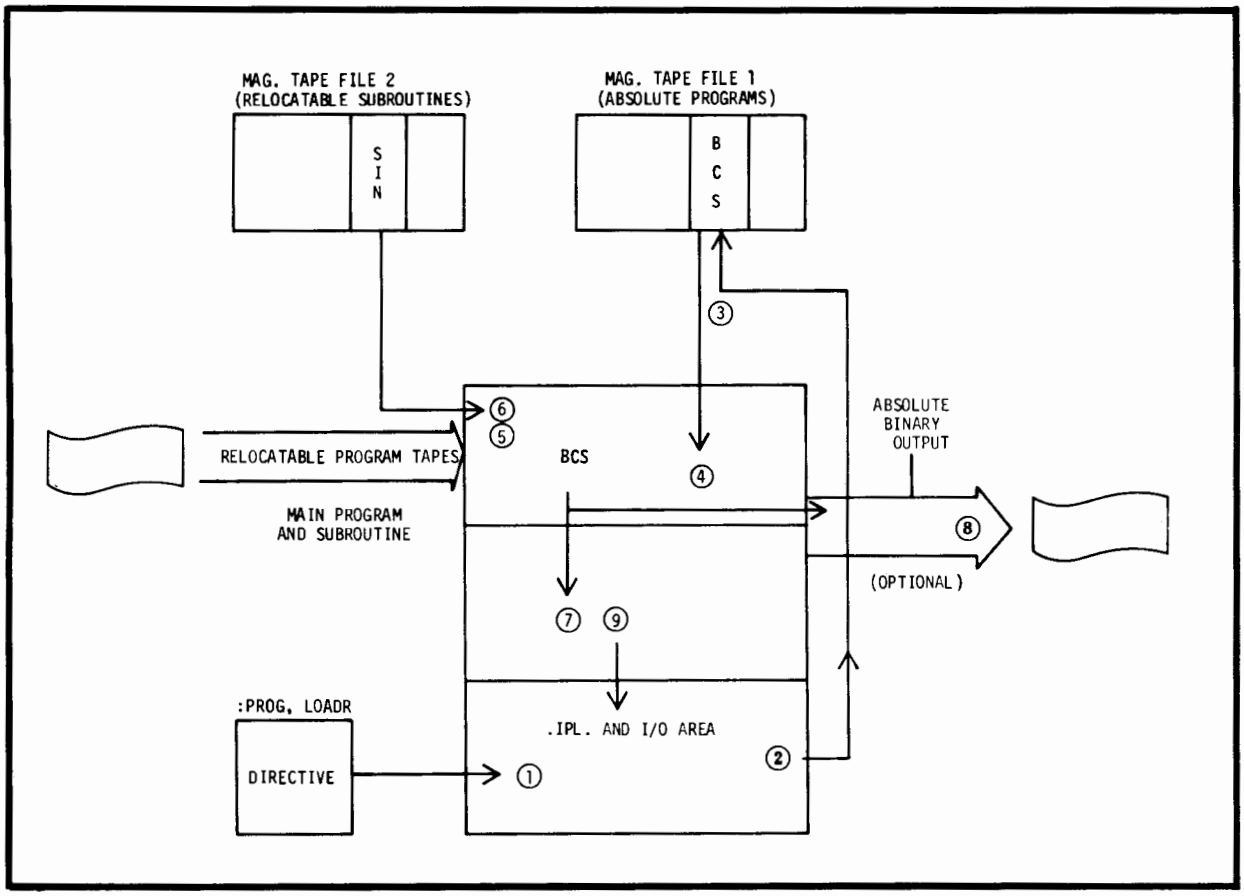

Figure 1-6. Execution of Relocatable Programs

USE OF FILE 2

File 1 may include one or more Basic Control Systems. Using a Basic Control System, relocatable object programs are relocated and run on-line or converted to absolute binary object programs suitable for inclusion in File 1.

File 2 contains relocatable subroutines, i.e., the Relocatable Program Library, which the Basic Control System integrates into programs that request them. Figure 1-6 shows the sequence of operation using the Basic Control System and File 2:

1. A directive specifies that a Basic Control System be transferred into core;

2-4. .IPL locates the Basic Control System by its identifier, loads it and transfers control to it;

5. The Basic Control System reads in the relocatable program tapes;
6. Then it merges in any subroutines from File 2;
7. The absolute code which is generated is loaded into core, or
8. Punched on paper tape;
9. If the program was loaded into core, the program runs, and when completed, makes a call to .STOP which returns to .IPL..

SYSTEM GENERATION

MTS is generated using the Prepare Tape System, an independent software program. The magnetic tape created by PTS is initiated using a Bootstrap paper tape program. Figure 1-7 shows a flow chart of the system generation procedure. The procedure is described in *PREPARE TAPE SYSTEM,* HP 02116-91751.

PTS generates File 1 of MTS from absolute programs and File 2 from relocatable programs input by the user. The user may desire several MTS tapes for different purposes: one system with only HP software programs for processing of programs, another system with a different Basic Control System for running user programs on-line, and a system with only File 1 for user absolute programs.

Each program stored in File 1 of the magnetic tape by PTS must be assigned an identifier by the user. Most identifiers are arbitrary; the user may select whichever names are useful. For consistency, this book uses the identifiers assigned in the sample PTS of Appendix A. Some programs, however, require specific identifiers: FTN2 for pass two of FORTRAN, X-REF for the Cross-Reference Symbol Table Generator, .IPL. for the inter-pass loader, and S.SIO for the standard SIO module. The user may not choose his own identifier for these programs. All identifiers must be unique.

Bootstrap

When the bootstrap tape, which includes the resident part of .IPL., is loaded using the Basic Binary Loader (BBL), it reads in the non-resident part of

.IPL and the standard SIO module from the magnetic tape. This operation initiates the Magnetic Tape System.

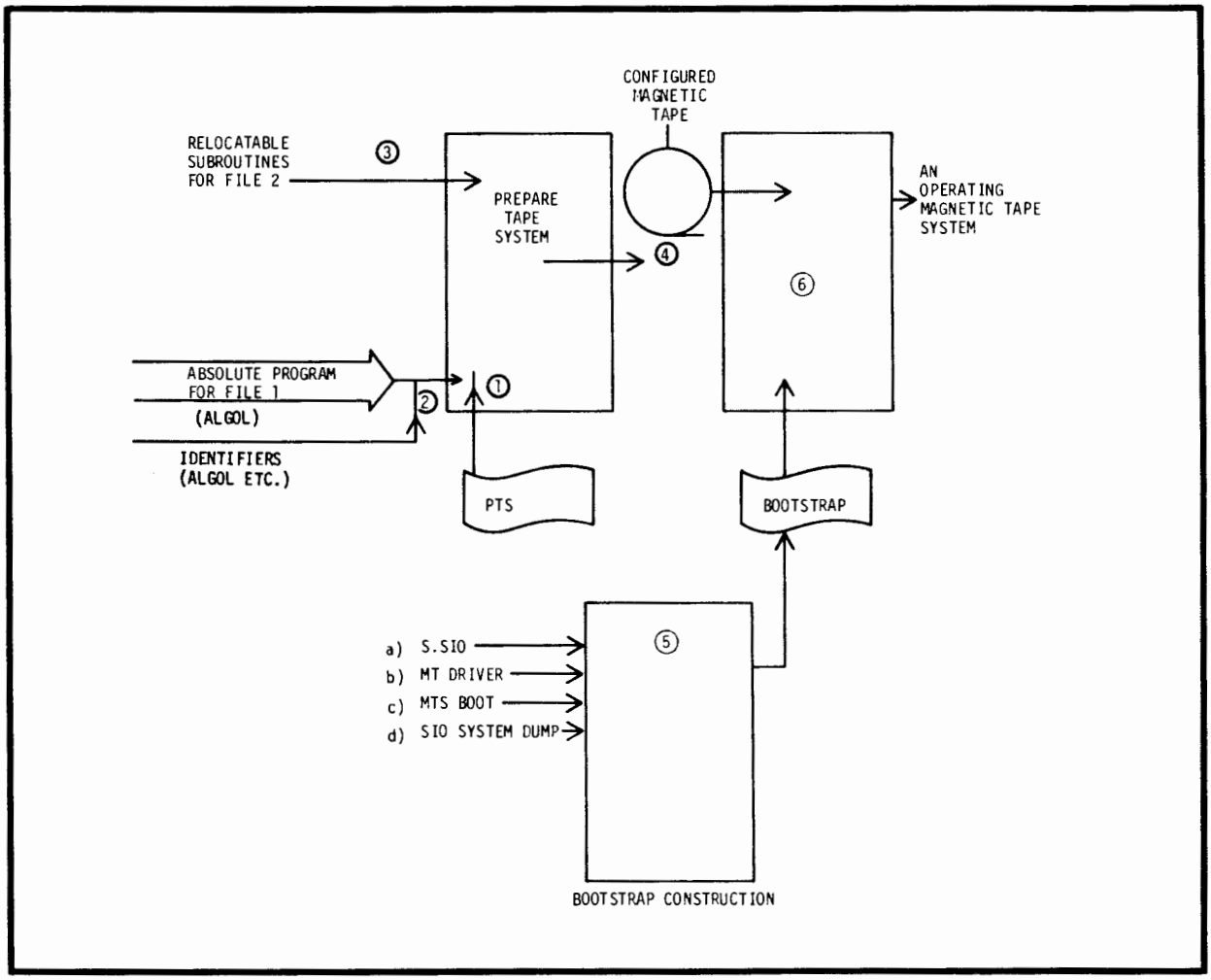

Figure 1-7. MTS Generation Procedure

SECTION II

OPERATING PROCEDURES

Section II covers the principles of system operating, bootstrapping, and restarting.

BOOTSTRAPPING MTS

A bootstrap tape initiates the operation of MTS from the PTS-created magnetic tape. Bootstrap configuration is described in the PTS manual.

To start MTS, the user must load the MTS magnetic tape and insure that a write enable ring is present; then:

	2116/2115		2114
a.	Place the bootstrap tape in the reader.	a.	Place the bootstrap tape in the reader.
b.	Set the Switch Register to $Ø377ØØ_8$; set the LOADER switch to ENABLED; press LOAD ADDRESS, PRESET and RUN.	b.	Press PRESET and LOAD simultaneously.
c.	When the computer halts with $1Ø2Ø77_8$ in the T-Register, set the LOADER switch to PROTECTED.	c.	Computer halts.
d.	Set the switch register to $1ØØ_8$ and press LOAD ADDRESS.	d.	Set the switch register to $1ØØ_8$ and press LOAD ADDRESS.
e.	Set the following switch register bits before pressing RUN:	e.	Set the following switch register bits before pressing RUN:

Bit 15 = 1 (on) to enable batch mode, Ø (off) to disable
(See *MTS OPERATING MODES*, below.)

Bit 1 = 1 (on) to suppress bootstrap messages.

MTS prints the bootstrap messages on the teleprinter (unless suppressed) and then prints *NEXT?, indicating that MTS is ready for directives. All switches

should be off because several programs use these switches (see *FORTRAN* and *ASSEMBLER* manuals).

For example:

```
        HP MAGNETIC TAPE SYSTEM
*BATCH OPTION ENABLED.  (or DISABLED.)  } suppressed by switch
*SET ALL SWITCH REGISTER BITS TO ZERO   } register bit 1.
*NEXT?
```

MTS OPERATING MODES

MTS has two modes of operation: keyboard and batch. In keyboard mode, directives from the user are entered through the system teleprinter. In batch mode, however, directives may be pre-punched and submitted for automatic reading through the batch input device. Operator intervention is not generally required.

Starting Addresses

For FORTRAN and the Assembler, the user may want to specify a starting address for keyboard mode. In keyboard mode, FORTRAN and the Assembler read their control statements from the teleprinter if they begin execution at 50_8 or 120_8 respectively. This can be accomplished by defining the starting address at PTS-time to 50_8 or 120_8. The PTS assigned address is used when in keyboard mode; 100_8 is used when in batch mode.*

For example, the Assembler's starting address is 100_8 and its implicit keyboard starting address is 120_8. If 120_8 was used as the PTS assigned starting address, then MTS would give control to the Assembler at location 100_8 when in batch mode, and at location 120_8 when in keyboard mode. If the user makes the PTS-assigned address equal to 100_8, then the HP software program begins execution at 100_8 in batch *and* keyboard modes.

*If the PTS-assigned starting address is equal to 2_8 or greater than 1777_8 (as in a BCS related program), the system always uses the PTS-assigned starting address, even in batch mode.

If a batch input device (card reader or paper tape reader) is available *and* is also included in the standard SIO module, then the operator can enable batch mode when bootstrapping MTS. The operator may switch from keyboard mode (which MTS always starts in) to batch mode by means of the :BATCH directive.

OPERATOR RESPONSIBILITY

When MTS is in keyboard mode, the programmer is responsible for all phases of the operation.

In batch mode, however, the programmer is responsible only for the MTS directives and for preparation of his job deck (directives plus data). If operator attention is required (e.g., to tear off a binary tape produced by FORTRAN and place it in the tape reader for BCS), the programmer must notify the operator verbally or by using :COMMENT directives within his job deck.

The operator is responsible for:

a. Insuring that all I/O devices are turned on, operable, and supplied with paper, etc.

b. Submitting jobs to the batch input device.

c. Handling output according to programmer specifications (e.g., what to do with binary tape output).

d. Setting switch register bits according to programmer specifications.

e. Shutting down MTS with the :PAUSE directive.

f. Continuing MTS whenever a halt occurs (see next page).

g. Monitoring the system teleprinter for messages from the user or the system.

Halt Conditions

If a halt occurs during MTS operation, a halt code appears in the top series of register lights on the computer panel. The possible codes and their meanings are:

Halt Code	Meaning
102000	Halt in response to a slash (/) in :PROG directive. Some operator action required before executing program by pressing RUN. Check teleprinter.
102001	Error in a directive during :BATCH mode. Message is printed: *CS ERR*. Correct and/or replace directive card. Press RUN.
102002	EOT (or two blank cards) in batch mode. Ready next source deck and press RUN.
102011	MT parity error. Message is printed on teleprinter: *TP ERR*. Press RUN. MTS attempts to reload itself and if successful, prints *NEXT?.
102044	Magnetic Tape error; unit in local; press AUTO button, then press RUN.
102077	Halt in response to :PAUSE directive. Check for comments on system teleprinter. Press RUN to continue.

HP software programs also contain halts which are documented in the appropriate manuals. In general, a halt code 102057 means that more source input is required. Also, a halt code 106055 means that programs execution has exceeded the program bounds or entered a data area when executing under BCS.

MTS RESTART PROCEDURE

After an operator abort or a system halt, the operator can restart MTS--make it ready for the next directive--by one of these methods:

1. During program execution:

2116/2115

a. Set Switch Register to 77_8.
b. Press LOAD ADDRESS.
c. Set all Switch Register bits to Ø.
d. Press LOAD A.
e. Press PRESET.
f. Press RUN.

2114

a. Press CLEAR REGISTER.
b. Press LOAD ADDRESS.
c. Press LOAD MEMORY.
d. Set Switch Register to 77_8.
e. Press LOAD ADDRESS.
f. Press CLEAR REGISTER.
g. Press PRESET.
h. Press RUN.

At this point MTS should type *NEXT? (keyboard mode) or read in the next directive (batch mode). If it does not, use method 3.

2. After a "/" halt (102000):

2116/2115

a. Set Switch Register to $1Ø6_8$.
b. Press LOAD ADDRESS.
c. Press DISPLAY MEMORY.
d. Set Switch Register to the value in the T-Register plus 1.
e. Press LOAD ADDRESS.
f. Set all Switch Register bits to Ø.
g. Press LOAD A.
h. Press PRESET.
i. Press RUN.

2114

a. Press CLEAR REGISTER.
b. Press LOAD ADDRESS.
c. Press LOAD MEMORY.
d. Set Switch Register to $1Ø6_8$.
e. Press LOAD ADDRESS.
f. Press DISPLAY MEMORY.
g. Set Switch Register to the value in the Memory Data Register plus 1.
h. Press LOAD ADDRESS.
i. Press CLEAR REGISTER.
j. Press PRESET.
k. Press RUN.

If MTS does not type *NEXT? (keyboard mode) or read in the next directive (batch mode), use method 3.

3. If these two methods do not restart MTS, the resident part of .IPL. has been destroyed. MTS must be re-initialized with the bootstrap procedure.

SECTION III

MTS DIRECTIVES

Directives are the user's line of communication with .IPL., the MTS control program. Directives consist of one line of ASCII characters, beginning with a colon (:) and a directive word. In the keyboard mode, .IPL. prints *NEXT? when it is ready for the operator to type in a directive. In batch mode, .IPL. automatically reads the directives from the batch input device without operator intervention.

Directive examples are:

```
:TYPE
:BATCH
:COMMENT,THIS IS A COMMENT
:PAUSE
:PROG,LOADR
```

Parameters within a :PROG directive are separated by a comma (,). A semicolon (;) may be used to separate program names in a :PROG directive. Directives are logically terminated by the first space; therefore, any following characters are treated as comments.

Each directive,whether correct or erroneous,is printed on the system teleprinter. Directives which are in error because of an illegal format or parameters cause .IPL. to print the following message:

```
*CS ERR*
```

The operator should re-enter the directive correctly.

:BATCH

Purpose

To switch from keyboard mode to batch mode.

Format

:BATCH

Comments

When MTS is initialized using the bootstrap tape, MTS is always in keyboard mode. (See Section II.) The operator may type :BATCH (if batch mode has been enabled by the bootstrap) on the system teleprinter at any time. This switches MTS to batch mode and allows directives to be submitted as part of the input deck.

To return to keyboard mode, the operator enters a :TYPE directive through the batch input device.

:TYPE

Purpose

To switch MTS from batch mode to keyboard mode.

Format

:TYPE

Comment

The :TYPE directive is legal only from the batch input device. Therefore, it must be prepunched on a card or paper tape. When MTS returns to keyboard mode, directives are again entered manually through the teleprinter.

:PAUSE

Purpose

To pause in the operations of MTS.

FORMAT

:PAUSE

Comment

A :PAUSE directive causes the computer to halt. Halts are often necessary for the operator to prepare I/O devices, set switch register bits, etc. before continuing operations. When the operator is ready to resume operations, he presses RUN. .IPL. returns to the mode it was in before the :PAUSE. If .IPL. is in keyboard mode, it prints *NEXT? and waits for the next directive. If in batch mode, .IPL. reads the next directive immediately from the batch input device.

:PAUSE should be used to suspend the operation of MTS when not in use.

:COMMENT

Purpose

To print a comment on the system teleprinter.

Format

:COMMENT, *character string*

where *character string* is a string of ASCII characters.

Comment

The :COMMENT directive may be used with the :PAUSE directive to relay instructions to the operator for preparation of I/O devices during batch processing. The comma must occur immediately following the "T" in :COMMENT.

:PROG

Purpose

To load one or more absolute programs from File 1. The loading order is determined by the order of specification.

Format

:PROG,*control-prog*[,*sub-prog*,...][/]

where *control-prog* is the identifier of a program in File 1. .IPL. transfers control to the starting address of this program.

sub-prog is the identifier of an optional program from File 1 to be loaded in addition to the *control-prog* (there may be more than one *sub-prog*).

/ causes .IPL. to halt before transferring control to the *control-prog*; this allows time to set switches or prepare data before pressing RUN. Items in [] are optional.

Comments

The order in which the programs are specified must be the order in which they appear on File 1. See Section IV for the use of :PROG to run HP software programs such as the FORTRAN compiler.

The "/" causes :PAUSE to be printed unless the PTS-assigned starting address of the *control-prog* is 2 or the overlay program BYLIST is used. When the computer halts, all programs specified in the directive are loaded, locations $1Ø6_8$ and $1Ø7_8$ of the link table are set, but the linkages back to .IPL. are *not* set.

SECTION IV

PROGRAMMING

To aid the user in preparing and running his programs, MTS provides four programming languages--ALGOL, FORTRAN, BASIC, and Assembly Language--and three types of programming--absolute, relocatable, and conversational.

Programs with absolute or fixed core memory assignments are written only in Assembly Language; they use the SIO drivers for input/output and are added to File 1 of the magnetic tape (using PTS) before being run with a :PROG directive. (See Samples 1,2, and 6 in Section V.)

Relocatable programs have no fixed memory assignment; they have memory addresses relative to a relocatable base address so they must be relocated into absolute locations by the Basic Control System before they can run. Relocatable programs use the BCS drivers for input/output with interrupt capability. FORTRAN and ALGOL generate only relocatable programs; assembly language can generate either relocatable or absolute programs. (See Samples 3,4, and 5 in Section V.)

The third type of programming, conversational, is done with the BASIC Interpreter. BASIC programs are developed by the user interacting with the computer through the teleprinter, modifying and checking his program until it is "debugged." BASIC programs are executed interpretively; therefore, no object code is generated and BASIC programs cannot be added to File 1.

Using the Symbolic Editor in MTS allows the programmer to easily edit source programs consisting of ASCII character statements. (See Samples 2 and 4 in Section V.)

ABSOLUTE PROGRAMMING

The Extended Assembler, an absolute program that may be included in File 1 of the magnetic tape, is the only software program that generates absolute object code directly from a source program. (The Basic Control System generates absolute code from relocatable object code.)

The Extended Assembler

As Appendix A shows, there are two types of Assemblers: EAU (Extended Arithmetic Unit) and non-EAU. The EAU Assembler generates special machine instructions to take advantage of EAU hardware. MTS should include the Assembler appropriate to the hardware configuration.

Also shown in Appendix A is another program identified as ASMB-CS. This program has the starting address $12Ø_8$ which is an alternative assembler starting address. When started at $12Ø_8$, the assembler accepts the control statement from the teleprinter keyboard instead of from the source program. This allows the user to assemble a program several times, using different control statement Options, without editing his program. The control statement may not be entered from the keyboard when in batch mode. An automatic Cross-Reference occurs after assembly if a "C" appears in the Control Statement. (See Samples 2 and 6 in Section V.)

ASMB-CS may be a non-executing dummy program, such as:

```
ASMB,A,B
      ORG 5ØB
      JMP 12ØB
      ORG 1ØØB
      JMP 12ØB
      ORG 12ØB
      CLA
      JSB 1Ø6B,I
      END
```

At PTS time, this program is added to File 1 and given the name ASMB-CS and a starting address of $12Ø_8$.

When ASMB-CS is a non-executing dummy program, it is run in conjunction with the Assembler:

```
:PROG,ASMB-CS,ASMB
```

.IPL. loads ASMB-CS and ASMB, then transfers control to the starting address of ASMB-CS since it was loaded first. Because ASMB-CS is a dummy program, control actually transfers to location $12\emptyset_8$ of the Assembler.

The Assembler accepts assembly language source programs as defined in the *ASSEMBLER* manual, HP 02116-9014. Absolute programs may be written to run within or without MTS; but if planned for execution within MTS, they must follow certain rules described in "Programming Conventions" below.

Operating Procedures

Assuming the Assembler is included in File 1, it is run by a :PROG directive. (Note that the items in brackets are optional.)

:PROG[,ASMB-CS],ASMB[,*sio*][,*overlays*][/]

where ASMB-CS if used, switches control statement input to the teleprinter.
ASMB (or ASMB-EAU) specifies the Assembler (or EAU-Assembler),
sio is any non-standard SIO module (S.SIO is used if none is given),
overlays are any of the three overlay programs (see "Overlay Programs" this section) and
/ causes .IPL. to halt before transferring control to ASMB so that the operator may set necessary switch register bits.

SWITCH REGISTER BITS

In addition to the standard switch register options described in the *ASSEMBLER* manual, switches 2 and 3 have special meaning in MTS:

Switch 2 - If on (1), read source program from magnetic tape File 3 (program must have been stored in File 3 by the Editor or a previous assembly).

Switch 3 - If on (1), line printer is list device; do not truncate list output to 72 characters.

Programming Conventions

ABSOLUTE PROGRAMMING

If an absolute program is to be added to File 1 as described in the *PREPARE TAPE SYSTEM* manual, it must follow these conventions:

a. No code may begin prior to location 6_8,

b. Location 77_8 and the starting address minus one must be a HLT 77B (.IPL. changes this to a JSB 1Ø6B,I),

c. Location $1ØØ_8$ (the starting address*) should be JMP $11Ø_8$ to avoid the system link table ($1Ø1$-$1Ø7_8$),

d. Locations $1Ø1$-$1Ø4_8$ must be a BSS 4,

e. Location $1Ø5_8$ must be set to the last word address plus one of the user's program,

f. Locations $1Ø6$-$1Ø7_8$ must be a BSS 2.

g. The program must make a call to .IPL. upon completion, either to:

Terminate: Set A-register to zero and jump to the starting address less one (this causes .IPL. to continue with the next user directive), or

Chain: Set A-register to -3 and JSB 1Ø6B,I with an identifier specified (this causes .IPL. to load the program identified from File 1). For example,

```
    LDA N3
    JSB 1Ø6B,I
    ASC 5, identifier
N3  DEC -3
```

where, *identifier* is the name of a program (10 characters) as defined at PTS time.

A program written according to these conventions, but not using the -3. .IPL. call, may also be run stand-alone (using BBL as described in the *ASSEMBLER* Manual). If any of these conventions are violated, the integrity of MTS cannot be guaranteed.

*The starting address may actually be between 6_8 and 15777_8 (8K) or 35777_8 (16K).

RELOCATABLE PROGRAMMING

Relocatable programs may be written in FORTRAN, ALGOL, or Assembly Language. The FORTRAN and ALGOL Compilers generate only relocatable code from source programs. Relocatable programs must be loaded by the Basic Control System before they can run.

The Extended Assembler

The Extended Assembler generates relocatable programs in the same manner as with absolute programs. Assembler operation has been described previously under "Absolute Programming." However, the programming conventions do not apply to relocatable programs. The relocatable code produced by the Assembler is equivalent to that produced by FORTRAN or ALGOL.

CHAINING (Also see Glossary p. G-4)

Relocatable assembly language programs must return to .IPL. by calling the library subroutine .STOP routine. In order to chain from relocatable assembly language programs, the programmer must use the following instructions:

```
        LDA M3
        JSB .1Ø6B,I
        ASC 5, identifier
M3      DEC -3
.1Ø6B   ABS 1ØØ1Ø6B
```

A relocatable program must not call in an SIO-environment program that does not include its own SIO module.

FORTRAN

The FORTRAN Compiler accepts source programs written in the FORTRAN Language, as defined in the *FORTRAN* manual, HP 02116-9015. Source programs may be read from paper tape or cards. File 3 is used to store intermediate code, and the final relocatable object program is punched on tape.

FORTRAN has an alternate starting address, $5Ø_8$, which is equivalent in function to the alternate starting address of the Assembler. A dummy program,

FTN-CS, should be included in File 1 as shown in Appendix A. FNT-CS may be a copy of ASMB-CS, but is assigned the starting address $5Ø_8$. When a :PROG directive includes FTN-CS and FTN, the FORTRAN Compiler reads the source program control statement from the teleprinter keyboard. The control statement may not be entered from the keyboard when in batch mode.

FORTRAN is run with a :PROG directive:

:PROG[,FTN-CS],FTN[,*sio*][,*overlays*][/]

where FTN-CS if used, switches control statement input to the teleprinter
FTN identifies the FORTRAN compiler in File 1,
sio is any non-standard SIO module (S.SIO is used if none is given),
overlays are any of the three overlay programs (see "Overlay Programs"), and
/ causes .IPL. to halt before transferring control to FTN so that the operator may set necessary switch register bits. (See the *FORTRAN* manual.)

ALGOL

The ALGOL Compiler identified in Appendix A as ALGOL, generates relocatable object code from source programs, as defined in the *ALGOL* manual, (HP 02116-9072).

Source program input may be from the photoreader or card reader, depending on the SIO module specified in the :PROG directive

:PROG,ALGOL[,*sio*][,*overlay*][/]

where *sio* is any non-standard SIO module (S.SIO if none is given),
overlay is one of the three overlay programs (see "Overlay Programs"), and
/ causes .IPL. to halt before transferring control to ALGOL so that the operator may set necessary switch register bits.

Loading Relocatable Programs

The relocatable programs and subroutines generated by FORTRAN, ALGOL, and Assembler must be relocated into absolute programs and linked with the appropriate BCS drivers and subroutines of File 2. This process is accomplished by the Basic Control System (BCS), its Relocating Loader and drivers.

Relocatable programs generated under MTS may be relocated with BCS in a stand-alone environment as described in the *BASIC CONTROL SYSTEM* manual, HP 02116-9017. Programs loaded in this way are not run in the MTS environment.

Alternatively, a BCS may be included in File 1 so that relocatable programs can be loaded in the MTS environment. BCS offers two methods of relocation (described in *BASIC CONTROL SYSTEM*):

a. Relocatable programs are relocated directly into core by BCS; in this case they are run immediately and there is no paper tape produced; or

b. Relocatable programs are relocated onto punched tape along with BCS drivers. This tape may be run stand-alone or may be added to File 1 of MTS.

A BCS to be used in MTS must be constructed in a certain way. See the sample PCS listing in Appendix A.

Operating Procedures

The operator indicates BCS with a :PROG directive:

```
:PROG,LOADR[/]
```

where LOADR is the identifier for the Basic Control System, and

/ allows the operator to place the first relocatable tape in the reader and set switch register bits.

Follow the operating procedures for BCS as described in the *BASIC CONTROL SYSTEM* manual starting with the switch register options. BCS searches File 2 of the magnetic tape for undefined external references. If an absolute tape is created, it may be added to File 1 using PTS. Programs added to File 1 may not use BCS DEBUG.

Cross-Reference Symbol Table Generator

The Cross-Reference Symbol Table Generator scans an Assembly Language source program on the magnetic tape and cross-reference each symbol. For each symbol, the line number where it is defined is cross-referenced to every line where the symbol is used. X-REF is the required identifier for the Cross-Reference Symbolic Table Generator. (See Samples 2, 6, and 7 in Section V.)

If the control statement of an assembly language source program contains a "C", the Assembler calls .IPL. to run X-REF when assembly is complete.

Alternatively, X-REF may be called following assembly by a separate :PROG directive:

:PROG,X-REF[,*sio*][/]

where *sio* should be the same SIO module used with the assembler, and
/ may be used to halt the system so that switch register bits 15 can be set for X-REF. (See the *ASSEMBLER* manual.)

Overlay Programs

Three overlay programs, identified in Appendix A as ONLINE, BYLIST, and BYPUNCH, provide additional options for ALGOL, FORTRAN, and the Assembler.

ONLINE causes the main program to be read from the teleprinter (regardless of the SIO module configuration.)

BYLIST causes the main program to bypass all listing including .IPL. messages until .IPL. is ready for the next directive (regardless of what is requested by the control statement.)

BYPUNCH causes the main program to bypass all punching (regardless of the control statement.)

The overlay programs exist in the above order on the MTS Utility Tape. They must be after the SIO modules on File 1. When calling for the main program (ALGOL, etc.), the overlay program is specified:

:PROG,*main-prog,overlay*,....

The overlays may be used simultaneously, but BYPUNCH and BYLIST cannot be used with the teleprinter only.

CONVERSATIONAL PROGRAMMING

The HP BASIC Interpreter may be included in File 1 of MTS. BASIC is initiated using the :PROG directive; for example,

```
:PROG,BASIC
```

BASIC includes its own I/O drivers so an SIO module must *not* be loaded over BASIC. The Interpreter accepts user commands and programs from the teleprinter keyboard and returns to MTS when the user types BYE. The structure of BASIC commands and statements is described in the *HP BASIC* manual, HP 02116-9077.

EDITING

Operating under MTS-control as an absolute program of File 1, the Symbolic Editor allows the programmer to edit symbolic files (e.g., source programs) by inserting, replacing, and deleting statements or characters. As it is edited, the symbolic file is transferred from an input device to an output device. The possible transfers in the MTS environment are:

Input	Output
Cards	Punched or Magnetic Tape
Punched Tape	Punched or Magnetic Tape
Magnetic Tape	Punched Tape

When not editing, the Editor may list symbolic files; also it can copy them directly onto the magnetic tape for processing by the Assembler.

There are two inputs to the Editor: an Edit File describing the edit operations, and a Symbolic File to be edited. The output consists of an updated Symbolic File or listing. (See Figure 4-1.)

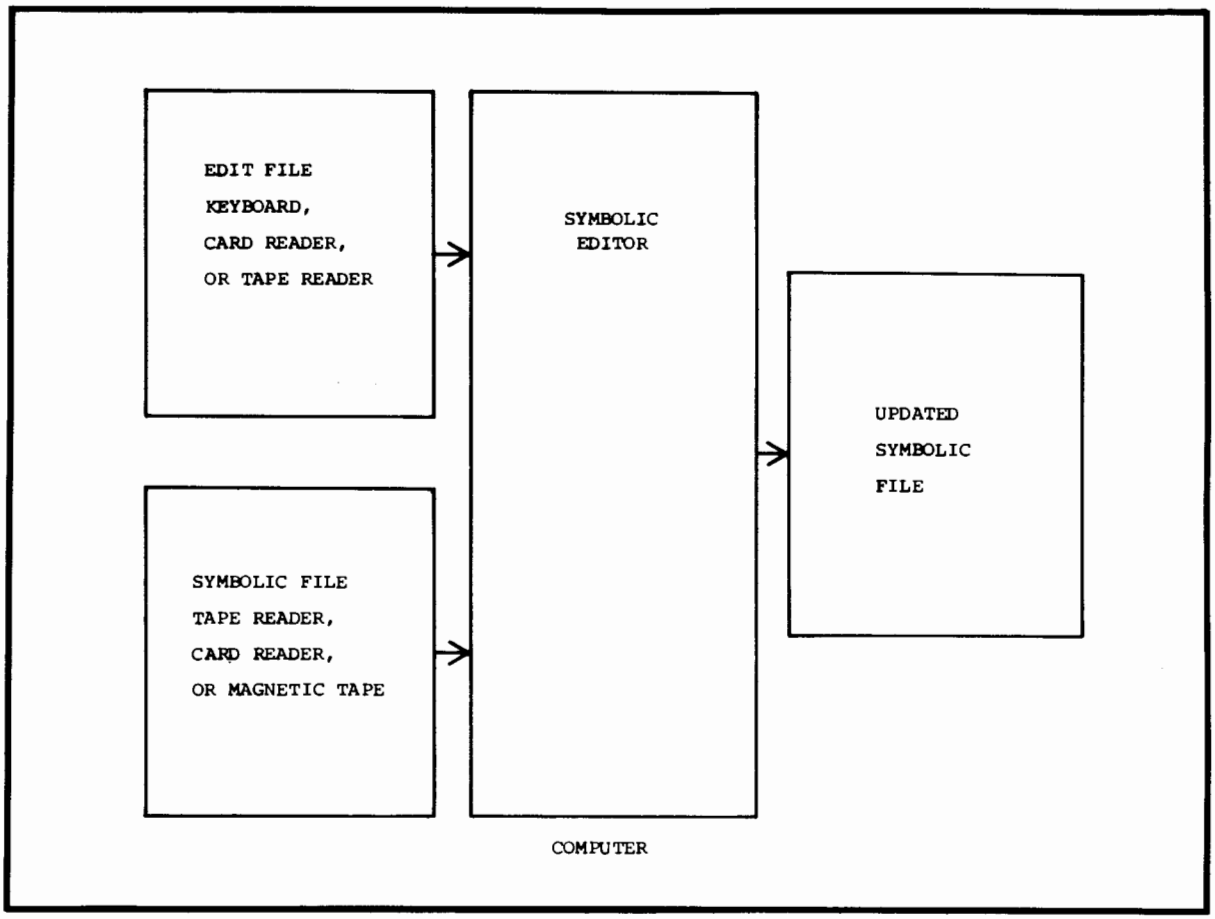

Figure 4-1. Overview of the Symbolic Editor

For further details on the Symbolic Editor, consult the *SYMBOLIC EDITOR* manual, HP 02116-9016.

SECTION V

MTS USAGE

Section V describes eight sample tasks run on MTS:

SAMPLE 1: Assemble an absolute program, add it to File 1, and run it.

SAMPLE 2: Edit, assemble, and cross-reference an absolute program.

SAMPLE 3: Compile, load into core, and run an ALGOL program.

SAMPLE 4: Edit, compile, relocate and punch, add to file 1, and run an ALGOL program.

SAMPLE 5: Compile a FORTRAN program, assemble a subroutine, link them with BCS, and run them in core.

SAMPLE 6: Assemble a program on-line and cross-reference it.

SAMPLE 7: Cross-reference only.

SAMPLE 8: Use of magnetic tape.

In all examples, certain typeface conventions are followed:

COMPUTER OUTPUT

COMPUTER INPUT

COMMENTS

TEXT

All tasks use S.SIO for input/output.

SAMPLE 1

Assemble an absolute program, add it to File 1, and run it with a :PROG directive.

MTS is initiated using a bootstrap tape.

```
                    HP MAGNETIC TAPE SYSTEM

*BATCH OPTION ENABLED.
*SET ALL SWITCH REGISTER BITS TO ZERO (Ø).
*NEXT?
:PROG,ASMB              (Directive to assemble the absolute program)


ØØØ1                    ASMB,A,L,B,T
LEN       ØØØ116
LWA       ØØØ1Ø5
MSG       ØØØ11Ø
RET       ØØØØ77
ADDR      ØØØ117
START     ØØØ12Ø
**     NO ERRORS*

ØØØ1                    ASMB,A,L,B,T
ØØØ2      ØØØ77                 ORG 77B
ØØØ3      ØØØ77 1Ø2Ø77  RET     HLT 77B          MTS CHANGES TO A JSB I
ØØØ4      ØØ1ØØ                 ORG 1ØØB         THIS IS THE ENTRY POINT
ØØØ5      ØØ1ØØ Ø2412Ø          JMP START         JUMP OVER THE TABLE OF SIO
ØØØ6      ØØ1Ø1 ØØØØØØ         BSS 4             DRIVER ADDRESSES
ØØØ7      ØØ1Ø5 ØØØ125  LWA     OCT 125          FIRST WORD AVAILABLE MEMORY
ØØØ8      ØØ1Ø6 ØØØØØØ         BSS 2             MAG TAPE LINKAGE
ØØØ9      ØØ11Ø Ø521Ø5  MSG     ASC 6,TEST  OUTPUT.
```

```
          ØØ111 Ø51524
          ØØ112 Ø2Ø117
          ØØ113 Ø52524
          ØØ114 Ø5Ø125
          ØØ115 Ø52Ø56
ØØ1Ø      ØØ116 ØØØØ14  LEN   OCT 14          THIS IS THE MESSAGE LENGTH
ØØ11      ØØ117 ØØØ11Ø  ADDR  DEF MFG MSG     ADDRESS OF THE MESSAGE
ØØ12      ØØ12Ø Ø6Ø116  START LDA LEN         LOAD BUFFER LENGTH
ØØ13      ØØ121 Ø64117        LDB ADDR        LOAD BUFFER ADDRESS
ØØ14      ØØ122 114Ø2         JSB 1Ø2B,I      GO TO SIO DRIVER
ØØ15      ØØ123 ØØ24ØØ        CLA             CLEAR THE A REGISTER
ØØ16      ØØ124 Ø24Ø77        JMP RET         RETURN
ØØ17                          END
** NO ERRORS*
```

Assembly is complete. Operator should save the absolute binary tape which is produced. This tape will be added to File 1.

```
*NEXT?
:PAUSE
```

MTS halts. Operator must now load the Prepare Tape System program using BBL. (Set the teleprinter select code in the switch register before starting at 100_8.)

PREPARE TAPE SYSTEM

```
PROGRAM INPUT DEVICE S.C.= ?
13
ABSOLUTE PROGRAMS, FILE#1.
LOAD THESE TWO(2) MODULES FIRST:
.IPL.
S.SIO
```

```
I.D. NAME:                  (Programs to be added to the end of File 1)
/A
I.D. NAME:
.ALP1.                      (Absolute program assembled above)
100
*LOAD                       (Place tape in tape reader, press RUN)
I.D. NAME:
/E
*EOF
RELOCATABLE LIBRARY, FILE#2.
*LOAD                       (File 2 must be rewritten)
*LOAD
*EOF
*END
```

MTS must be re-initiated using a bootstrap tape.

HP MAGNETIC TAPE SYSTEM

```
*BATCH OPTION ENABLED.  (Switch 15 set to 1)
*SET ALL SWITCH REGISTER BITS TO ZERO(0).  (All switches were not set to 0)
*NEXT?
:PROG,.ALP1.            (Directive to run the absolute program)
TEST OUTPUT.
*NEXT?
:PAUSE
```

SAMPLE 2

Edit, assemble, and cross-reference an absolute program.

```
*NEXT?                    (MTS is active and ready for a directive)
:PROG,EDIT                (Directive to call the Symbolic Editor)

                          HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T
*
/L                        (Just list the Symbolic File)
/E
SYMBOLIC FILE SOURCE DEVICE?
/P
ØØØ1      ASMB,A,L,B,T
ØØØ2            ORG 77B
ØØØ3      RET   HLT 77B          MTS CHANGES TO A JSB I
ØØØ4            ORG 1ØØB       THIS IS THE ENTRY POINT
ØØØ5            JMP START        JUMP OVER THE TABLE OF SIO ADDRESSES
ØØØ6            BSS 4            DRIVER ADDRESSES
ØØØ7      LWA   OCT 125          FIRST WORD AVAILABLE MEMORY
ØØØ8            BSS 2            MAG TAPE LINKAGE
ØØØ9      MSG   ASC 6,TEST OUTPUT.
ØØ1Ø      LEN   OCT 14           THIS IS THE MESSAGE LENGTH
ØØ11      ADDR  DEF MSG            ADDRESS OF THEMESSAGE
ØØ12      START LDA LEN          LOAD BUFFER LENGTH
ØØ13            LDB ADDR         LOAD BUFFER ADDRESS
ØØ14            JSB 1Ø2B,I       GO TO SIO DRIVER
ØØ15            CLA              CLEAR THE A REGISTER
ØØ16            JMP RET          RETURN
ØØ17            END
```

```
**END-OF TAPE
*
/E
*END
*NEXT?
:PROG,EDIT              (Call Editor again)

                            HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T
*
/R,1
ASMB,A,L,B,T,C
/E
SYMBOLIC FILE SOURCE DEVICE?
/P
SYMBOLIC FILE DESTINATION DEVICE?
/M                      (For Magnetic Tape)
**END-OF-TAPE
*
/E
*END                    (Edited program is now on File 3 of magnetic tape)
*NEXT?
:PROG,ASMB              (Operator should set switch register bit 2 on to
PAGE 0001                read source from magnetic tape)
0001                    ASMB,A,L,B,T,C
LEN    000116
LWA    000105
MSG    000110
RET    000077
ADDR   000117
START  000120
** NO ERRORS*
```

```
ØØØ1                    ASMB,A,L,B,T,C          (C specifies cross-reference)
ØØØ2    ØØØ77                   ORG 77B
ØØØ3    ØØØ77 1Ø2Ø77    RET     HLT 77B         MTS CHANGES TO A JSB I
ØØØ4    ØØ1ØØ                   ORG 1ØØB        THIS IS THE ENTRY POINT
ØØØ5    ØØ1ØØ Ø2412Ø            JMP START         JUMP OVER THE TABLE OF SIO
ØØØ6    ØØ1Ø1 ØØØØØØ           BSS 4             DRIVER ADDRESSES
ØØØ7    ØØ1Ø5 ØØØ125    LWA     OCT 125         FIRST WORD AVAILABLE MEMORY
ØØØ8    ØØ1Ø6 ØØØØØØ           BSS 2             MAG TAPE LINKAGE
ØØØ9    ØØ11Ø Ø521Ø5    MSG     ASC 6,TEST OUTPUT.
        ØØ111 Ø51524
        ØØ112 Ø2Ø117
        ØØ113 Ø52524
        ØØ114 Ø5Ø125
        ØØ115 Ø52Ø56
ØØ1Ø    ØØ116 ØØØØ14    LEN     OCT 14          THIS IS THE MESSAGE LENGTH
ØØ11    ØØ117 ØØØ11Ø    ADDR    DEF MSG           ADDRESS OF THE MESSAGE
ØØ12    ØØ12Ø Ø6Ø116    START   LDA LEN           LOAD BUFFER LENGTH
ØØ13    ØØ121 Ø64117            LDB ADDR          LOAD BUFFER ADDRESS
ØØ14    ØØ122 114102            JSB 102B,I        GO TO SIO DRIVER
ØØ15    ØØ123 ØØ24ØØ            CLA             CLEAR THE A REGISTER
ØØ16    ØØ124 Ø24Ø77            JMP RET           RETURN
ØØ17                            END
** NO ERRORS*
```

Assembly is complete. Now the program is cross-referenced automatically because of the C option.

CROSS-REFERENCE SYMBOL TABLE

```
ADDR    ØØ11  ØØ13
LEN     ØØ1Ø  ØØ12
LWA     ØØØ7
MSG     ØØØ9  ØØ11
RET     ØØØ3
START   ØØ12  ØØØ5
NEXT?
:PAUSE
```

SAMPLE 3

Compile, load into core, and run an ALGOL program.

```
*NEXT?                    (MTS is active and ready for a directive)
:PROG,ALGOL               (Directive to call the ALGOL compiler)
PAGE  ØØ1

ØØ1  Ø2ØØØ HPAL,L,B,"CRD"
ØØ2  Ø2ØØØ BEGIN COMMENT
ØØ3  Ø2ØØ3      THIS PROGRAM CONVERTS CARTESIAN TO POLAR COORDINATES:
ØØ4  Ø2ØØ3 LABEL EN, ST, OUT;
ØØ5  Ø2ØØ7 REAL X,Y,R,THETA,PI;
ØØ6  Ø2ØØ7 FORMAT  F1 ("ENTER VALUES FOR X AND Y");
ØØ7  Ø2Ø25 FORMAT  F2("R="F6.2,"     THETA+"F6.2"RADIANS");;
ØØ8  Ø2Ø5Ø WRITE(2,F1);
ØØ9  Ø2Ø56 ST: READ(1,*,X,Y);
ØlØ  Ø2Ø71 PI←3.1416;
Ø11  Ø2Ø75 R←SQRT(X*X+Y*Y);
Ø12  Ø2114 IF X=Ø THEN
Ø13  Ø212Ø        BEGIN
Ø14  Ø212Ø          IF Y>=Ø THEN THETA←.5*PI ELSE
Ø15  Ø2133          THETA←-.55*PI;
Ø16  Ø2141          GO TO EN
Ø17  Ø2142        END;
Ø18  Ø2142 THETA←ARCTAN(ABS(Y)/ABS(X));
Ø19  Ø2163 IF Y=Ø AND X>=Ø THEN
Ø2Ø  Ø2176         BEGIN
Ø21  Ø2176         THETA←Ø.
Ø22  Ø2176 ;       GO TO EN;
Ø23  Ø22Ø3         END
Ø24  Ø22Ø3 ELSE IF Y=Ø AND X<Ø THEN
Ø25  Ø2216         BEGIN
Ø26  Ø2216       THETA←PI;
Ø27  Ø2222         GO TO EN
Ø28  Ø223  ;       END
```

```
Ø29  Ø2223 ELSE
Ø Ø  Ø2224 IF Y>=Ø AND X>=Ø THEN GO TO EN ELSE IF Y>=Ø AND X<=Ø THEN
Ø31  Ø2253        BEGIN
Ø32  Ø2253           THETA←THETA+.5*PI;
Ø33  Ø2263           GO TO EN;
Ø34  Ø2264        END
Ø35  Ø2264        ELSE IF Y<Ø AND X<Ø THEN
Ø36  Ø2275            BEGIN
Ø37  Ø2275            THETA←THETA+PI;
Ø38  Ø23Ø3            GO TO EN;
Ø39  Ø23Ø4            END;
Ø4Ø  Ø23Ø4 THETA←THETA+1.5*PI;
Ø41  Ø2314 EN:  WRITE(2,F2,R,THETA);
Ø42  Ø233Ø IF Y=Ø AND X=Ø THEN GO TO OUT;
Ø43  Ø2345 GO TO ST;
Ø44  Ø2346 OUT: END$

PROGRAM= ØØØ347 BASE PAGE= ØØØØ34  ERRORS=ØØØ
```

Compilation is complete. Operator should save the relocatable binary tape for loading.

```
*NEXT?
:PROG,LOADR              (Directive to call the Basic Control System)
CRD                      (Operator places relocatable tape in reader)
 Ø2ØØØ Ø2346             (Switch Register bit 15 = Ø)
 ØØ334 ØØ367
*LOAD                    (Switch Register bit 2 = 1)
FRMTR                    (Library is loaded from File 2)
 Ø2347 Ø4423
 ØØ37Ø Ø1Ø71
```

```
ATAN
 Ø4424 Ø45Ø5
 Ø1Ø72 Ø1136
SQRT
 Ø45Ø6 Ø46Ø2
 Ø1137 Ø1154
CHEBY
 Ø46Ø3 Ø466Ø
 Ø1155 Ø1167
ABS
 Ø4661 Ø4665
..FCM
Ø4666 Ø4673
Ø1173 Ø1175
.STOP
Ø5647 Ø5661
Ø1176 Ø1177
.ERRR
 Ø5662 Ø5676
 Ø12ØØ Ø12Ø3
PWR2
 Ø5677 Ø5716
 Ø12Ø4 Ø12Ø7
.FLUN
 Ø5717 Ø5727
 Ø121Ø Ø1211
ENDIO
```

```
 05730 05736

*LST                    (Switch Register bit 15 = 0, produces loader
.IOC    11343            symbol table)
.SQT.   11304
.MEM.   11277
.BUFR   11511
HALT    11274
.DIO.   04025
.DTA.   04123
.IOR.   03675
.FMP    05155
.FAD    04705
SQRT    04506
ABS     04661
.FDV    05051
ARCTA   04424
.STOP   05647
.BIO.   04100
.IAR.   03761
.IOI.   03722
.RAR.   03735
.DLD    05557
.DST    05567
.FLUN   05717
.MPY    05241
.PACK   05357
FLOAT   05352
IFIX    05615
ATAN    04424
.CHEB   04603
.FSB    04710
.ERRR   05662
.PWR2   05677
..FCM   04666
..DLC   04674
```

```
.DIV   Ø5464
ENDIO  Ø573Ø
*LINKS
 Ø1725 Ø1777

*RUN                    (Program is in core, ready to run)
ENTER VALUES FOR X AND Y
7658  8463
R=11353.     THETA=   .84RADIANS
Ø Ø
R=   .ØØ     THETA=  1.57RADIANS
STOP                    (End of execution)
*NEXT?
:PAUSE
```

SAMPLE 4

Edit, compile, relocate and punch, add to File 1, and run an ALGOL program.

```
:NEXT?                    (MTS is active and ready to accept a directive)
:PROG,EDIT


                                HP SYMBOLIC EDITOR

EDIT FILE DEVICE?
/T
*
/L                        (Just lists the program)
/E
SYMBOLIC FILE SOURCE DEVICE?
/P
0001   HPAL,L,B,"CRD"
0002   BEGIN COMMENT
0003        THIS PROGRAM CONVERTS CARTESIAN TO POLAR COORDINATES;
0004   LABEL EN,ST,OUT;
0005   REAL X,Y,R,THETA,PI;
0006   FORMAT F1 ("ENTER VALUES FOR X AND Y");
0007   FORMAT F2("R="F6.2,"     THETA="F6.2"RADIANS");;
0008   WRITE(2,F1);
0009   ST: READ(1,*,X,Y);
0010   PI←3.1416;
0011   R←SQRT(X*X+Y*Y);
0012   IF X=0 THEN
0013          BEGIN
0014            IF Y>=0 THEN THETA←.5*PI ELSE
0015            THETA← -.5*PI;
0016            GO TO EN
0017          END;
```

```
0018   THETA←ARCTAN(ABS(Y)/ABS(X));
0019   IF Y=0 AND X>=0 THEN
0020           BEGIN
0021           THETA←0
0022   :       GO TO EN;
0023           END
0024   ELSE IF Y=0 AND X<0 THEN
0025           BEGIN
0026         THETA←PI;
0027           GO TO EN
0028   ;       END
0029   ELSE
0030   IF Y>0 AND X>=0 THEN GO TO EN ELSE IF Y>=0 AND X<=0 THEN
0031         BEGIN
0032            THETA←THETA+.5*PI;
0033            GO TO EN;
0034         END
0035         ELSE IF Y<0 AND X<0 THEN
0036             BEGIN
0037             THETA←THETA+PI;
0038             GO TO EN;
0039             END;
0040   THETA←THETA+1.5*PI;
0041   EN:  WRITE(2,F2,R,THETA);
0042   IF Y=0 AND X=0 THEN GO TO OUT;
0043   GO TO ST;
0044   OUT: END$
**END-OF-TAPE
*
/E
*END
*NEXT?
:PROG,EDIT
```

HP SYMBOLIC EDITOR

```
EDIT FILE DEVICE:
/T
*
/R,8
ST: WRITE(2,F1);
/R,9
READ(1,*,X,Y);
/E
SYMBOLIC FILE SOURCE DEVICE:
/P
SYMBOLIC FILE DESTINATION DEVICE?
/P

**END-OF-TAPE
*
/E
*END                    (Edit is completed.  Operator should save paper tape
*NEXT?                   for compilation.)
:PROG,ALGOL,PR-LP       (Directive to call ALGOL compiler)
PAGE  001

001  02000 HPAL,L,B,"CRD"
002  02000 BEGIN COMMENT
003  02003      THIS PROGRAM CONVERTS CARTESIAN TO POLAR COORDINATES;
004  02003 LABEL EN,ST,OUT;
005  02007 REAL X,Y,R,THETA,PI;
006  02007 FORMAT F1 ("ENTER VALUES FOR XAND Y");
007  02025 FORMAT F2("R="F6.2,"     THETA="F6.2"RADIANS");;
008  02050 ST: WRITE(2,F1);
009  02056 READ(1,*,X,Y);
010  02071 PI←3.1416;
011  02075 R←SQRT(X*X+Y*Y);
```

```
Ø12  Ø2114 IF X=Ø THEN
Ø13  Ø212Ø        BEGIN
Ø14  Ø212Ø          IF Y>=Ø THEN THETA←.5*PI ELSE
Ø15  Ø2133          THETA← -.5*PI;
Ø16  Ø2141          GO TO EN
Ø17  Ø2142        END;
Ø18  Ø2142 THETA←ARCTAN(ABS(Y)/ABS(X));
Ø19  Ø2163 IF Y=Ø AND X>=Ø THEN
Ø2Ø  Ø2176         BEGIN
Ø21  Ø2176         THETA←Ø.
Ø22  Ø2176 ;       GO TO EN;
Ø23  Ø22Ø3         END
Ø24  Ø22Ø3 ELSE IF Y=Ø AND X<Ø THEN
Ø25  Ø2216         BEGIN
Ø26  Ø2216       THETA←PI;
Ø27  Ø2222         GO TO EN
Ø28  Ø2223 ;       END
Ø29  Ø2223 ELSE
Ø3Ø  Ø2224 IF Y>=Ø AND X>=Ø THEN GO TO EN ELSE IF Y>=Ø AND X<=Ø THEN
Ø31  Ø2253           BEGIN
Ø32  Ø2253              THETA←THETA+.5*PI;
Ø33  Ø2263              GO TO EN;
Ø34  Ø2264           END
Ø35  Ø2264           ELSE IF Y<Ø AND X<Ø THEN
Ø36  Ø2275               BEGIN
Ø37  Ø2275               THETA←THETA+PI;
Ø38  Ø23Ø3               GO TO EN;
Ø39  Ø23Ø4               END;
Ø4Ø  Ø23Ø4 THETA←THETA+1.5*PI;
Ø41  Ø2314 EN:  WRITE(2,F2,R,THETA);
Ø42  Ø233Ø IF Y=Ø AND X=Ø THEN TO TO OUT;
Ø43  Ø2345 GO TO ST;
Ø44  Ø2346 OUT: END$

PROGRAM= ØØØ347 BASE PAGE= ØØØØ34 ERRORS=ØØØ

End of compilation.  Operator should save relocatable binary tape for loading.
```

```
*NEXT?
:PROG,LOADR/          (/ halts MTS so that switches can be set)
                      (Switch register bit 14 = 1)
CRD                   (Place relocatable tape in reader)

 Ø2ØØØ Ø2346
 ØØ334 ØØ367
*LOAD                 (Library is read from File 2)
FRMTR
 Ø2347 Ø4423
 ØØ37Ø Ø1Ø71
ATAN
 Ø4424 Ø45Ø5
 Ø1Ø72 Ø1136
SQRT
 Ø45Ø6 Ø46Ø2
 Ø1137 Ø1154
CHEBY
 Ø46Ø3 Ø466Ø
 Ø1155 Ø1167
ABS
 Ø4661 Ø4665
IFIX
 Ø5615 Ø5646
 Ø1173 Ø1175
.STOP
 Ø5647 Ø5661
 Ø1176 Ø1177
.ERRR
 Ø5662 Ø5676
 Ø12ØØ Ø12Ø3
PWR2
 Ø5677 Ø5716
 Ø12Ø4 Ø12Ø7
```

```
.FLUN
 Ø5717 Ø5727
 Ø121Ø Ø1211

ENDIO
 Ø573Ø Ø5736

*LST
.IOC.  11343
.SQT.  113Ø4
.MEM.  11277
.BUFR  11511
HLAT   11274
.DIO.  Ø4Ø25
.DTA.  Ø4123
.IOR.  Ø3675
.FMP   Ø5155
.FAD   Ø47Ø5
SQRT   Ø45Ø6
ABS    Ø4661
.FDV   Ø5Ø51
ARCTA  Ø4424
.STOP  Ø5647
.BIO.  Ø41ØØ
.IAR.  Ø3761
.IOI.  Ø3722
.RAR.  Ø3735
.DLD   Ø5557
.DST   Ø5567
.FLUN  Ø5717
.MPY   Ø5241
.PACK  Ø5357
FLOAT  Ø5352
IFIX   Ø5615
ATAN   Ø4424
.CHEB  Ø46Ø3
.FSB   Ø471Ø
```

```
.ERRR  Ø5662
.PWR2  Ø5677
..FCM  Ø4666
..DLC  Ø4674
.DIV   Ø5464
ENDIO  Ø573Ø
*LINKS
 Ø1725 Ø1777

*END                    (End of loading.  Operator should save absolute tape
*NEXT?                   for PTS.)
:PAUSE
```

PTS must be loaded into core. (Program may also be run stand-alone)

PREPARE TAPE SYSTEM

```
PROGRAM INPUT DEVICE S.C.= ?
13

ABSOLUTE PROGRAMS, FILE#1.
LOAD THESE TWO(2) MODULES FIRST:
.IPL.
S.SIO

I.D. NAME
/A
I.D. NAME:              (Relocated program is added to the end of File 1)
.ALGOL.
S.A.
2
*LOAD                   (Place tape in reader)
```

```
I.D. NAME:
/E
*EOF
RELOCATABLE LIBRARY, FILE#2.
*LOAD
*LOAD
*EOF
*END
```

MTS must be re-initiated with the bootstrap tape.

HP MAGNETIC TAPE SYSTEM

```
*BATCH OPTION ENABLED.    (All switches were down).

*NEXT?
:PROG,.ALGOL.             (Directive to call the program)
ENTER VALUES FOR X AND Y
4567  9876
R=1Ø881.      THETA=  1.14RADIANS
ENTER VALUES FOR XAND Y
654  8375
R=84ØØ.5      THETA=  1.49RADIANS
ENTER VALUES FOR X AND Y
Ø Ø
R=   .ØØ      THETA=  1.57RADIANS
STOP
*NEXT?
:PAUSE
```

SAMPLE 5

Compile a FORTRAN program, assemble a subroutine, link them with BCS, and run them in core

```
*NEXT?                    (MTS is active and ready for a directive)
:PROG,FTN                 (Directive to call the FORTRAN compiler)
FTN,L,B
      PROGRAM MST1
      CALL MST2 (J,K,L)
      I=(J+K)*L
      WRITE(2,1Ø)I
   1Ø FORMAT(///"THE ANSWER IS ",I12)
      END
      END$
*NEXT?                    (End of compilation.  Save tape for loading.)
:PROG,ASMB                (Directive to call assembler)
PAGE ØØØ1

ØØØ1                   ASMB,R,B,L,T
A     R ØØØØØØ
B     R ØØØØØ1
C     R ØØØØØ2
D     R ØØØØØ3
E     R ØØØØØ4
F     R ØØØØØ5
.ENTRX ØØØØØ1
MST2  R ØØØØØ6
** NO ERRORS*
```

```
PAGE  0002 #01

0001                ASMB,R,B,L,T
0002  00000                NAM MST2
0003                       ENT MST2
0004                       EXT .ENTR
0005  00000 000000  A       BSS 1
0006  00001 000000  B       BSS 1
0007  00002 000000  C       BSS 1
0008  00003 000005  D       DEC 5
0009  00004 000012  E       DEC 10
0010  00005 000004  F       DEC 4
0011  00006 000000  MST2    NOP
0012  00007 016001X         JSB .ENTR
0013  00010 000000R         DEF A
0014  00011 062003R         LDA D
0015  00012 172000R         STA A,I
0016  00013 062004R         LDA E
0017  00014 172001R         STA B,I
0018  00015 062005R         LDA F
0019  00016 172002R         STA C,I
0020  00017 126006R         JMP MST2,I
0021                       END
**  NO ERRORS*
```

Assembly is completed. Save the relocatable tape for loading.

```
*NEXT?
:PROG,LOADR              (Directive to call the Basic Control System)

MST1                     (Place program tape in reader)

 02000 02053
```

```
 *LOAD                    (Place subroutine tape in reader)
MST2
 Ø2Ø54 Ø2Ø73              (Load library subroutines)
*LOAD
FRMTR
 Ø2Ø74 Ø415Ø
 ØØ334 Ø1Ø35
MPY
 Ø4151 Ø4261
FLOAT
 Ø4262 Ø4266
.PACK
 Ø4267 Ø4373
.ENTR
 Ø4374 Ø4434
 Ø1Ø36 Ø1Ø42
DLDST
 Ø4435 Ø4472
IFIX
 Ø4473 Ø4524
 Ø1Ø43 Ø1Ø45
.STOP
 Ø4525 Ø4537
 Ø1Ø46 Ø1Ø47
.FLUN
 Ø454Ø Ø455Ø
 Ø1Ø5Ø Ø1Ø51
```

```
ENDIO
 04551 04557
CLRIO
 04560 04564

*LST                        (Loader symbol table)
.IOC.  11343
.MEM.  11277
.BUFR  11511
HALT   11274
MST1   02000
CLRIO  04560
MST2   02062
.MPY   04151
.DIO.  03552
.IOI.  03447
.DTA.  03650
.STOP  04525
.ENTR  04374
.BIO.  03625
.IAR.  03506
.IOR.  03422
.RAR.  03462
.DLD   04435
.DST   04445
.FLUN  04540
.PACK  04267
FLOAT  04262
IFIX   04473
ENDIO  04551
*LINKS
 01730 01777
```

```
*RUN                    (Program and Subroutine are loaded in core, ready to
                         run)
THE ANSWER IS           6Ø
STOP                    (Printed by .STOP subroutine)
*NEXT?
:PAUSE
```

SAMPLE 6

Assemble a program on-line and cross-reference it.

```
*NEXT?                  (MTS is active and ready for a directive)
:PROG,ASMB,ONLINE       (Directive to call the Assembler with the ONLINE option)
ASMB,A,L,B,T,C↵         (Program is entered through keyboard as soon as magnetic
 PAGE 0001               tape has stopped)

0001                    ASMB,A,L,B,T,C (C causes cross-reference at the end of
       ORG 77B                          assembly)
RET    HLT 77B
       ORG 100B
       JMP START
       BSS 4
LWA    OCT 125
       BSS 2
MSG    ASC 6,***ONLINE***
LEN    OCT 14
ADDR   DEF MSG
START  LDA LEN
       LDB ADDR
       JSB 102B,I
       CLA
       JMP RET
       END

LEN       000116
LWA       000105
MSG       000110
RET       000077
ADDR      000117
START     000120
**  NO ERRORS*
```

```
 PAGE  0002 #01

0001                    ASMB,A,L,B,T,C
0002   00077                    ORG 77B
0003   00077 102077  RET       HLT 77B
0004   00100                    ORG 100B
0005   00100 024120             JMP START
0006   00101 000000             BSS 4
0007   00105 000125  LWA        OCT 125
0008   00106 000000             BSS 2
0009   00110 025052  MSG        ASC 6,***ONLINE***
       00111 025117
       00112 047114
       00113 044516
       00114 042452
       00115 025052
0010   00116 000014  LEN        OCT 14
0011   00117 000110  ADDR       DEF MSG
0012   00120 060116  START      LDA LEN
0013   00121 064117             LDB ADDR
0014   00122 114102             JSB 102B,I
0015   00123 002400             CLA
0016   00124 024077             JMP RET
0017                            END
**  NO ERRORS*
```

Assembly is completed. Program is automatically cross-referenced because C appeared in the control statement.

CROSS-REFERENCE SYMBOL TABLE

ADDR	ØØ11	ØØ13
LEN	ØØ1Ø	ØØ12
LWA	ØØØ7	
MSG	ØØØ9	ØØ11
RET	ØØØ3	ØØ16
START	ØØ12	ØØØ5

*NEXT?

:PAUSE

SAMPLE 7

Cross-reference directly from external source input.

The Assembler is called with a control statement containing only C and R or A. The Assembler writes the source program on File 3; then it skips pass 2 because no output was requested, and chains directly to the Cross-Reference Symbol Table Generator.

```
:PROG,ASMB (Directive to call Assembler)
(Program contains:  ASMB,R,C)
PAGE ØØØ1
ØØØ1          ASMB,R,C
**  NO ERRORS*

                    CROSS-REFERENCE SYMBOL TABLE

ADDR   ØØ11   ØØ13
LEN    ØØ1Ø   ØØ12
LWA    ØØØ7
MSG    ØØØ9   ØØ11
RET    ØØØ3   ØØ16
START  ØØ12   ØØØ5
*NEXT?
```

SAMPLE 8

Magnetic tape programming in FORTRAN and Assembly Language.

This sample shows two programs that exercise the READ/WRITE functions of the magnetic tape unit according to the programming techniques discussed in Appendix B.

FORTRAN Program

```
FTN,L,B
      PROGRAM M2Ø2Ø
      DIMENSION I2(12)
C
C     TEST TO SEE IF TAPE DRIVE IS IN LOCAL
C
    5 IF (LOCAL(12B))1Ø,2Ø
   1Ø WRITE (2,15)
   15 FORMAT(///"YOUR TAPE UNIT IS IN LOCAL MODE"
     1       ///"PUSH AUTO AND PUSH RUN")
      PAUSE
C
C     REWIND THE TAPE
C
   2Ø REWIND 12B
C
C     WRITE OUT 5Ø RECORDS AND TEST FOR END OF TAPE
C     AFTER EACH WRITE OPERATION
C
      NUM=1
      DO 35 I=1,5Ø
      WRITE(12B,3Ø)NUM
   3Ø FORMAT(I2," MAGNETIC TAPE RECORD ")
      NUM=NUM+1
      IF (IEOT(12B))8Ø,35
   35 CONTINUE
C
C     WRITE AN END OF FILE
C
      ENDFILE 12B
C
C     CALL PTAPE AND BACKSPACE 25 RECORDS
C
      K1=12B
      K2=Ø
      K3=-25
      CALL PTAPE(K1,K2,K3)
```

```
C
C     READ THE REMAINING RECORDS ON MAG TAPE
C
   4Ø READ(12B,45)I2
   45 FORMAT(12A2)
C
C     TEST FOR AN END OF FILE
C
   5Ø IF(IEOF(12B))8Ø,55
C
C     TEST FOR READ ERROR
C
   55 IF(IERR(12B))6Ø,65
   6Ø WRITE(2,61)
   61 FORMAT("TAPE READ ERROR, RECORD NOT READ"/////)
      GO TO 4Ø
C
C     TEST FOR AND END OF TAPE
C
   65 IF(IEOT(12B))8Ø,7Ø
C
C     WRITE OUT THE RECORD JUST READ
C
   7Ø WRITE(2,75)I2
   75 FORMAT(12A2)
      GO TO 4Ø
C
   8Ø WRITE(2,85)
   85 FORMAT(///"YOU HAVE JUST WRITTEN THE LAST RECORD"///)
C
C     JOB COMPLETE REWIND THE TAPE
C
      CALL RWSTB(12B)
      PAUSE
      GO TO 5
      END
      END$
```


Program Output

```
YOUR TAPE UNIT IS IN LOCAL MODE

PUSH AUTO AND PUSH RUN
PAUSE
27 MAGNETIC TAPE RECORD
28 MAGNETIC TAPE RECORD
29 MAGNETIC TAPE RECORD
3Ø MAGNETIC TAPE RECORD
31 MAGNETIC TAPE RECORD
32 MAGNETIC TAPE RECORD
33 MAGNETIC TAPE RECORD
34 MAGNETIC TAPE RECORD
```

```
35 MAGNETIC TAPE RECORD
36 MAGNETIC TAPE RECORD
37 MAGNETIC TAPE RECORD
38 MAGNETIC TAPE RECORD
39 MAGNETIC TAPE RECORD
4Ø MAGNETIC TAPE RECORD
41 MAGNETIC TAPE RECORD
42 MAGNETIC TAPE RECORD
43 MAGNETIC TAPE RECORD
44 MAGNETIC TAPE RECORD
45 MAGNETIC TAPE RECORD
46 MAGNETIC TAPE RECORD
47 MAGNETIC TAPE RECORD
48 MAGNETIC TAPE RECORD
49 MAGNETIC TAPE RECORD
5Ø MAGNETIC TAPE RECORD

YOU HAVE JUST WRITTEN THE LAST RECORD

PAUSE
```

Assembly Language Program

```
 PAGE ØØØ1

ØØØ1                        ASMB,R,B,L,T
M1     R ØØØØØØ
M2     R ØØØØ31
M4     R ØØØØ54
.IOC.  X ØØØØØ1
CNTR   R ØØØ126
COUNT  R ØØØ132
FILCT  R ØØØ13Ø
LINE1  R ØØØØ76
LINE2  R ØØØ112
LOCAL  R ØØØ134
MASK1  R ØØØØ71
MASK2  R ØØØØ72
MASK3  R ØØØØ73
MASK4  R ØØØØ74
MSG1   R ØØØ236
MSG2   R ØØØ251
MSG4   R ØØØ267
PAGE   R ØØØ224
PTAPE  X ØØØØØ2
READT  R ØØØ2Ø1
RECCT  R ØØØ131
RWND   R ØØØ141
SAVEA  R ØØØØ75
START  R ØØØ133
UNIT   R ØØØ127
WRITE  R ØØØ151
**  NO ERRORS*
```

```
 PAGE ØØØ2 #Ø1

ØØØ1                    ASMB,R,B,L,T
ØØØ2  ØØØØØ                    NAM TAPE
ØØØ3                            ENT START
ØØØ4                            EXT .IOC.
ØØØ5                            EXT PTAPE
ØØØ6  ØØØØØ Ø54517  M1          ASC 19,YOUR TAPE UNIT IS IN LOCAL MODE PRESS
      ØØØØ1 Ø52522
      ØØØØ2 Ø2Ø124
      ØØØØ3 Ø4Ø52Ø
      ØØØØ4 Ø4244Ø
      ØØØØ5 Ø52516
      ØØØØ6 Ø44524
      ØØØØ7 Ø2Ø111
      ØØØ1Ø Ø5144Ø
      ØØØ11 Ø44516
      ØØØ12 Ø2Ø114
      ØØØ13 Ø475Ø3
      ØØØ14 Ø4Ø514
      ØØØ15 Ø2Ø115
      ØØØ16 Ø475Ø4
      ØØØ17 Ø4244Ø
      ØØØ2Ø Ø5Ø122
      ØØØ21 Ø4252Ø
      ØØØ22 Ø5144Ø
ØØØ7  ØØØ23 Ø4Ø525              ASC 6,AUTO AND RUN
      ØØØ24 Ø52117
      ØØØ25 Ø2Ø1Ø1
      ØØØ26 Ø471Ø4
      ØØØ27 Ø2Ø122
      ØØØ3Ø Ø52516
ØØØ8  ØØØ31 Ø54517  M2          ASC 19,YOU HAVE JUST WRITTEN THE LAST RECORD
      ØØØ32 Ø5244Ø
      ØØØ33 Ø441Ø1
      ØØØ34 Ø531Ø5
      ØØØ35 Ø2Ø112
      ØØØ36 Ø4252Ø
      ØØØ37 Ø52Ø4Ø
      ØØØ4Ø Ø53522
      ØØØ41 Ø44524
      ØØØ42 Ø521Ø5
      ØØØ43 Ø47Ø4Ø
      ØØØ44 Ø5211Ø
      ØØØ45 Ø4244Ø
      ØØØ46 Ø461Ø1
      ØØØ47 Ø51524
      ØØØ5Ø Ø2Ø122
      ØØØ51 Ø425Ø3
      ØØØ52 Ø47522
      ØØØ53 Ø42Ø4Ø
ØØØ9  ØØØ54 Ø521Ø1  M4          ASC 13,TAPE ERROR RECORD NOT READ
      ØØØ55 Ø5Ø1Ø5
```

```
       00056 020105
       00057 051122
       00060 047522
       00061 020122
       00062 042503
       00063 047522

 PAGE  0003  #01

       00064 042040
       00065 047117
       00066 052040
       00067 051105
       00070 040504
0010*
0011   00071 000001  MASK1 OCT 1
0012   00072 000040  MASK2 OCT 40
0013   00073 000200  MASK3 OCT 200
0014   00074 000002  MASK4 OCT 2
0015*
0016   00075 000000  SAVEA BSS 1
0017   00076 046501  LINE1 ASC 12,MAGNETIC TAPE SYSTEM....
       00077 043516
       00100 042524
       00101 044503
       00102 020124
       00103 040520
       00104 042440
       00105 051531
       00106 051524
       00107 042515
       00110 027056
       00111 027056
0018   00112 000000  LINE2 BSS 12
0019   00126 000000  CNTR  BSS 1
0020   00127 000012  UNIT  OCT 12
0021   00130 000000  FILCT DEC 0
0022   00131 177747  RECCT DEC -25
0023   00132 177716  COUNT DEC -50
0024*
0025*
0026   00133 000000  START NOP
0027   00134 016001X LOCAL JSB .IOC.    DYNAMIC STATUS CHECK
0028   00135 030012        OCT 30012
0029   00136 012071R       AND MASK1    NORMAL RETURN AND TO
0030*                                   TEST BIT ZERO
0031   00137 052071R       CPA MASK1    COMPARE TO MASK
0032   00140 026236R       JMP MSG1     EQUAL..UNIT IN LOCAL
0033*                                  GO TO MESSAGE ONE
0034*
0035   00141 016001X RWND  JSB .IOC.    START REWIND OF TAPE
0036   00142 030412        OCT 30412
0037   00143 026151R       JMP WRITE    REJECT POINT IF TAPE
0038*                                  ALREADY AT LOAD POINT
```

```
0039  00144 016001X        JSB .IOC.    NORMAL RETURN ISSUE
0040*                                  DYNAMIC STATUS TO SEE
0041*                                  IF UNIT STILL REWINDING
0042  00145 030012         OCT 30012
0043  00146 002021         SSA,RSS      NORMAL RETURN TEST
0044*                                  BIT 15 FOR A 1 WHICH
0045*                                  SAYS UNIT STILL REWINDING
0046  00147 026151R        JMP WRITE    BIT IS 0  NOT MOVING
0047  00150 026144R        JMP *-4      BIT IS 1  IS  MOVING
0048*
0049  00151 062132R WRITE LDA COUNT
0050  00152 072126R        STA CNTR     PUT -50 INTO LOOP COUNTER

PAGE  0004  #01

0051  00153 016001X        JSB .IOC.    CALL IOC TO WRITE A RECORD
0052  00154 020012         OCT 20012
0053  00155 026153R        JMP *-2      REJEC  ADDRESS
0054  00156 000076R        DEF LINE1
0055  00157 177750         DEC -24
0056*
0057  00160 016001X        JSB .IOC.    TEST FOR END OF TAPE
0058  00161 040012         OCT 40012
0059  00162 012072R        AND MASK2
0060  00163 052072R        CPA MASK2
0061  00164 026251R        JMP MSG2     AT END OF TAPE
0062*
0063  00165 036126R        ISZ CNTR     REDUCE VALUE IN COUNTER BY ONE
0064*                                  AND SKIP IF ZERO
0065  00166 026153R        JMP WRITE+2
0066*
0067  00167 016001X        JSB .IOC.     WRITE AND END OF FILE
0068  00170 030112         OCT 30112
0069  00171 026167R        JMP *-2
0070  00172 000000         NOP
0071  00173 000000         NOP
0072*
0073  00174 016002X        JSB PTAPE
0074  00175 000201R        DEF *+4      CALL PTAPE AND BACKSPACE
0075  00176 000127R        DEF UNIT     ON UNIT NUMBER
0076  00177 000130R        DEF FILCT    SO MANY FILES
0077  00200 000131R        DEF RECCT    SO MANY RECORDS
0078*
0079  00201 016001X READT  JSB .IOC.     READ FROM MAG TAPE
0080  00202 010012         OCT 10012
0081  00203 026201R        JMP *-2
0082  00204 000112R        DEF LINE2
0083  00205 177750         DEC -24
0084*
0085  00206 016001X        JSB .IOC.     TEST STATUS AFTER LAST READ
0086  00207 040012         OCT 40012
0087  00210 072075R        STA SAVEA     SAVE STATUS INFORMATION
0088  00211 012073R        AND MASK3     TEST BIT 7
0089  00212 052073R        CPA MASK3     TEST FOR END OF FILE
```

```
0090  00213 026251R       JMP MSG2      AT END OF FILE
0091*
0092  00214 062075R       LDA SAVEA     RESTORE A
0093  00215 012074R       AND MASK4     TEST BIT 1
0094  00216 052074R       CPA MASK4     TEST FOR PARITY ERROR
0095  00217 026267R       JMP MSG4     ERROR FOUND
0096*
0097  00220 062075R       LDA SAVEA     RESTORE A
0098  00221 012072R       AND MASK2     TEST BIT 5
0099  00222 052072R       CPA MASK2     TEST FOR END OF TAPE
0100  00223 026251R       JMP MSG2      END OF TAPE FOUND
0101*
0102  00224 016001X PAGE   JSB .IOC.     WRITE THE RECORD OUT
0103  00225 020002         OCT 20002     ON SYSTEM TELEPRINTER
0104  00226 026224R       JMP *-2
0105  00227 000112R       DEF LINE2
0106  00230 177750        DEC -24
0107  00231 016001X       JSB .IOC.

 PAGE 0005 #01

0108  00232 040002        OCT 40002
0109  00233 002020        SSA
0110  00234 026231R       JMP *-3
0111  00235 026201R       JMP READT     GO AND READ ANOTHER RECORD
0112*
0113  00236 016001X MSG1   JSB .IOC.     MESSAGE ONE
0114  00237 020002         OCT 20002
0115  00240 026236R       JMP *-2
0116  00241 000000R       DEF M1
0117  00242 177716        DEC -50
0118  00243 016001X       JSB .IOC.
0119  00244 040002        OCT 40002
0120  00245 002020        SSA
0121  00246 026243R       JMP *-3
0122  00247 102077        HLT 77B
0123  00250 026141R       JMP RWND      GO TO REWIND
0124*
0125  00251 016001X MSG2   JSB .IOC.    MESSAGE TWO
0126  00252 020002         OCT 20002
0127  00253 026251R       JMP *-2
0128  00254 000031R       DEF M2
0129  00255 177732        DEC -38
0130  00256 016001X       JSB .IOC.
0131  00257 040002        OCT 40002
0132  00260 002020        SSA
0133  00261 026256R       JMP *-3
0134  00262 016001X       JSB .IOC.
0135  00263 030512        OCT 30512     START REWIND AND STANDBY
0136  00264 026265R       JMP *+1
0137  00265 102077        HLT 77B
0138  00266 026134R       JMP LOCAL
0139*
```

```
0140  00267 016001X MSG4  JSB .IOC.     MESSAGE FOUR
0141  00270 020002        OCT 20002
0142  00271 026267R       JMP *-2
0143  00272 000054R       DEF M4
0144  00273 177746        DEC -26
0145  00274 016001X       JSB .IOC.
0146  00275 040002        OCT 40002
0147  00276 002020        SSA
0148  00277 026274R       JMP *-3
0149  00300 026201R       JMP READT
0150                      END START
**  NO ERRORS*
```

Program Output

```
YOUR TAPE UNIT IS IN LOCAL MODE PRESS AUTO AND RUN
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
MAGNETIC TAPE SYSTEM....
YOU HAVE JUST WRITTEN THE LAST RECORD
```

APPENDIX A

SAMPLES

SAMPLE LISTING OF PTS OPERATION

Prepare Tape System (PTS) is fully described in the *PREPARE TAPE SYSTEM* manual. What follows is a sample system generation that is used throughout this book for reference. The same typeface conventions are used as in Section V.

PREPARE TAPE SYSTEM

PROGRAM INPUT DEVICE S.C.= ?
13

ABSOLUTE PROGRAMS, FILE #1.
LOAD THESE TWO (2) MODULES FIRST:

.IPL.
.S.SIO

I.D. NAME: *(Inter-Pass Loader:* tape-resident segment)
.IPL. *(required name)*
S.A
77
* LOAD

I.D. NAME: *(Standard SIO module: TY-CR-PU)*
S.SIO *(required name)*
S.A.
77
* LOAD

I.D. NAME: *(Cross-Reference Symbol Table Generator)*
X-REF *(required name)*
S.A.
1ØØ
* LOAD

```
I.D. NAME:          (Option to enter Assembler control statement through
ASMB-CS              keyboard)
S.A.                (dummy program)
12Ø
* LOAD

I.D. NAME:          (Extended Assembler: non-EAU)
ASMB
S.A.
1ØØ
* LOAD

I.D. NAME:          (ALGOL compiler)
ALGOL
S.A.
1ØØ
* LOAD

I.D. NAME:          (Symbolic Editor: paper tape and magnetic tape)
EDIT
S.A.
1ØØ
* LOAD

I.D. NAME:          (Extended Assembler: EAU)
ASMB-EAU
S.A.
1ØØ
* LOAD

I.D. NAME:          (Option to enter FORTRAN Control Statement through
FTN-CS               keyboard)
S.A.                (dummy program)
5Ø
* LOAD
```

```
I.D. NAME:              (Fortran Compiler)
FTN
S.A.
100
* LOAD

I.D. NAME:              (Non-standard SIO module: LP-CR-PU)
CR-LP
S.A.
77
* LOAD

I.D. NAME:              (Non-standard SIO module: LP-PR-PU)
PR-LP
S.A.
77
* LOAD

I.D. NAME:              (Non-standard SIO module: TY-PR-PU)
PR-TY
S.A.
77
* LOAD

I.D. NAME:              (Option to input source program through keyboard)
ONLINE
S.A.
77
* LOAD

I.D. NAME               (Option to bypass list output)
BYLIST
S.A.
77
* LOAD

I.D. NAME               (Option to bypass punch output)
BYPUNCH
S.A
77
*LOAD
```

```
I.D. NAME:              (FORTRAN Compiler: pass 2)
FTN2                    (required name)
S.A.
100
* LOAD
I.D. NAME:              (Basic Control System (BCS), Relocating Loader)
LOADR
S.A.
2
* LOAD
I.D. NAME:              (BASIC I/O drivers)
BASIC
S.A.
100
*LOAD
I.D. NAME:              (BASIC Interpreter)
/C
*LOAD
I.D. NAME:
/E
*EOF
RELOCATABLE LIBRARY, FILE #2.
*LOAD
*LOAD
*EOF
*END
```

SAMPLE OF PREPARE CONTROL SYSTEM (PCS)

For a complete description of PCS, consult the *BASIC CONTROL SYSTEM* manual.

```
HS INP?
21
HS PUN?
22

FWA MEM?
11Ø                    (MTS required response)
LWA MEM?
35777                  (MTS response; 15777 for 8K)

* LOAD

D.22                   (Magnetic tape driver should be loaded first.
 34633 35777           Use non-buffered IOC with 2020 magnetic tape,
                       because D.21 turns off the interrupt system
* LOAD                 during data transfer.  D.22 may not be used
                       as an external driver to BCS because it uses
D.ØØ                   DMA)
 34Ø77 34632

* LOAD

D.Ø1
 3354Ø 34Ø76

* LOAD

D.Ø2
 3323Ø 33537

* LOAD

D.12
 32573 33227
```

```
* LOAD

D.15
 32Ø36 32572

* LOAD

IOC
 31621 32Ø35

* TABLE ENTRY

EQT?
12,D.15
21,D.Ø1
22,D.Ø2
15,D.ØØ
14,D.ØØ
16,D.12
26,D.22,D,U1        (Magnetic tape unit is protected with this EQT entry)
26,D.22,D           (Magnetic tape unit is unprotected with this EQT entry)
/E

SQ ?
-KYBD?
12
-TTY?
12
-LIB?               (Library is on the magnetic tape)
15
-PUNCH?
11
-INPUT?
1Ø
-LIST?
14
```

```
DMA?
6,7

* LOAD

LOADR
 27155 31551

 INTERRUPT LINKAGE ?

4,1Ø3ØØ4              (Power fail halt)
5,1Ø6ØØ5              (Parity error halt)
12,3Ø,I.15
14,31,I.ØØ
15,31,I.ØØ
16,32,I.12
21,33,I.Ø1
22,34,I.Ø2
26,35,I.22
27,36,C.22
77,1Ø2Ø77             (Abort halt; replaced by JSB 1Ø6B,I at run-time)
/E

.SQT.  31552
.EQT.  3156Ø
C.22   35611
D.22   34633
I.22   35611
.BUFR  3177Ø
DMAC1  32Ø34
DMAC2  32Ø35
D.ØØ   34Ø77
I.ØØ   34253
D.Ø1   3354Ø
I.Ø1   33655
D.Ø2   3323Ø
I.Ø2   33344
```

```
D.12   32573
I.12   32737
D.15   32Ø36
I.15   32242
.IOC.  31621
IOERR  32Ø13
XEQT   32Ø33
XSQT   32Ø32
HALT   3154Ø
LST    272Ø6
.LDR.  3Ø653
.MEM.  31545

*SYSTEM LINK
 ØØ11Ø ØØ34Ø

*BCS ABSOLUTE OUTPUT
*END
```

APPENDIX B
PROGRAMMING TECHNIQUES

Magnetic tape units cannot be considered direct programming substitutes for paper tape devices; that is, programs written for paper tape I/O devices must be modified to use magnetic tape. Magnetic tape units require special considerations in initialization, status checking, error conditions and recovery, data formats, blocking techniques, end-of-tape conditions, and hardware modes of I/O. Many of these features are demonstrated in Sample 8, Section V.

INITIALIZATION

A program performs two initializing operations before carrying out any data transfers on a given magnetic tape (MT) unit:

1. Make a dynamic status check of the MT unit to determine if it is ready for operation. It may be in LOCAL mode or busy. If the unit is in LOCAL mode, the program notifies the operator to place it in AUTO mode.

2. When the unit is in AUTO mode and not busy, the program issues a REWIND request to the unit to insure that various software flags, counters, and status of the MT driver are properly initialized and that the MT unit itself is at LOAD POINT and ready to read or write the first record.

READING MAGNETIC TAPE

If the magnetic tape has been previously written by a similar type of magnetic tape unit (7 track vs. 9 track) and the bit density has been properly selected (200, 556, or 800 bits per inch), then the MT unit should be ready to READ a record.

Because the MT hardware is a record-oriented device, a READ request passes over a complete physical record of the magnetic tape (an End-Of-File is always

considered to be a complete record) regardless of the actual record length or number of words requested by the program. The entire record or only a portion is transmitted to the memory buffer.

If the physical record length on the magnetic tape is unknown, then an extremely large I/O request (limited to 16383 words for HP3Ø3Ø) may be used to READ in the entire record. If only part of the record is to be transmitted to the memory buffer, then the READ request need only specify that part; but the remainder of the record is passed over by the magnetic tape unit regardless. After a READ request, the magnetic tape unit halts between records.

Status of Magnetic Tape after READ (non-SIO-environment)

After a magnetic tape record has been READ, the program must examine the status of the magnetic tape unit in the following order:

a. Check for End-Of-File (EOF): Determine whether or not the record read was an End-Of-File mark (a special record recognized by the MT hardware). A parity error cannot occur during the reading of an EOF. An EOF is considered a complete record for the purposes of positioning and reading.

b. Check for Parity Error: If the record was not an EOF, then the validity of the record is indicated by the parity and timing bits of the MT status word. If the bits are not set (equal to zero), then the record just read probably is correct. (It is possible but extremely rare to incorrectly read a record from MT, but get proper status.) Most MT drivers reread a record about three times on parity error before indicating failure to the program. The last read attempt is transmitted to the program buffer. The program may prefer to ignore the bad record rather than attempt to retry reading. To reread a record, the program must backspace over the record.

c. Check for End-Of-Tape (EOT: Determine if the last forward motion operation (positioning or reading) passed over the End-Of-Tape marker. The magnetic tape (hardware) unit does not halt automatically at End-Of-Tape. The magnetic tape driver, however,

does not perform forward movement functions after EOT. Upon determining that EOT has been reached, the program issues either a REWIND or a REWIND/STANDBY request.

POSITIONING THE MAGNETIC TAPE

After a backspace or forward-space function, only the EOF, SOT, and EOT status conditions are valid.

A READ request should never immediately follow a WRITE request without an intervening BACKSPACE or REWIND request. Once a WRITE or WRITE END-OF-FILE request is executed on a magnetic tape unit, all succeeding information on that magnetic tape is lost because the magnetic tape unit cannot reliably write a record on exactly the same area of the tape more than once.

WRITE AND WRITE END-OF-FILE

The program should check that a write enable ring is in the magnetic tape unit before initiating any WRITE or WRITE EOF operations.

After a WRITE request is complete, the program should check for EOT. If EOT has been reached, then an EOF mark should be written to "close" the magnetic tape and a REWIND or REWIND/STANDBY request issued. In general, an EOF should be written to "close" magnetic tapes before issuing a REWIND after writing. The MT drivers will always automatically retry writing bad write operations until successful or the EOT is reached.

The following procedure is used by MT drivers upon detecting an unsuccessful write operation:

1. Backspace over the bad record.
2. Erase three inches of tape (zeroes).
3. Rewrite the record.

If the EOT is reached during this procedure and parity or timing status bits are still set, then a hardware failure (either the magnetic tape unit or the magnetic tape itself) is highly probable.

SIO vs BCS, DOS, RTE RECORD FORMATS

SIO records contain an extra word containing the actual record data length in front of the data record; this extra word does not occur in records written by BCS, DOS, or RTE. When the SIO driver is reading a record, it strips this word off before returning to the program, but a BCS (DOS, RTE) driver transmits it into the user buffer as part of the record. When reading an SIO record, a program using a BCS, DOS, or RTE driver should ignore this word. When a program (BCS, DOS, or RTE) is writing a record to be read by an SIO program, the program should add this record length word to its data.

SIO drivers only READ and WRITE in binary mode on magnetic tape. For nine-track magnetic tape units, there is no physical difference between binary and ASCII record modes. However, on a seven-track magnetic tape unit, the hardware reverses the parity of the record depending on the mode: ASCII records are written in even parity and binary records are written in odd parity. For compatability, BCS, DOS, and RTE drivers recognize both types of I/O request. See the record diagrams in the *BASIC CONTROL SYSTEM* manual.

RECORD BLOCKING

MT units are capable of writing data in highly packed densities for efficient operation. For example, a "card image" record (a record containing a copy of a punched card of 80 columns) is highly condensed when compared to the original card. However, there is a 3/4-inch record gap of all zeroes between all records, no matter how large or small the records themselves. This means that a magnetic tape completely filled with "card image" records is actually about 90% record gap or empty. Thus, the longer the records, the more efficiently the MT is used. The higher the tape density (e.g.,556 vs. 800 b.p.i.), the longer the data records must be to achieve efficient use of tape space.

Logical vs. Physical Records

A physical record on a magnetic tape is a series of contiguous data items preceded and followed by a record gap. A logical record is a collection of related data items. Usually a single logical record is written on the magnetic tape as a single physical record. However, the Formatter Library routine which handles FORTRAN and ALGOL input/output has an internal buffer of 60 words, and can only write a maximum physical record of 60 words. Therefore, if the user program requests the Formatter to write 100 words (binary request) on the magnetic tape, the Formatter will break the 100 words into two physical records: one of 60 words and one of 40 words. The FORTRAN program assumes that the Formatter wrote a physical record of 100 words because a read request for 100 words causes the Formatter to read both physical records and return them as one logical record. Record blocking efficiency can be increased by treating several logical blocks as one physical record.

Positioning with PTAPE

The PTAPE routine allows a BCS program to backspace or forward space any number of files and/or records on a magnetic tape unit. The calling sequence is given in the *PROGRAM LIBRARY* manual (HP 02116-9032).

After using PTAPE, the programmer should call MAGTP to check the status of the MT.

Checking Status with MAGTP

The MAGTP routine allows a BCS program to check the status of a magnetic tape unit. Status should be checked for all READ/WRITE requests. The calling sequences are given in the *PROGRAM LIBRARY* manual.

BASIC CONTROL SYSTEM

When programming for the magnetic tape unit in a BCS environment, the programmer must consider the positioning of the magnetic tape with PTAPE (the file protect feature), and the checking of status with MAGTP. PTAPE and MAGTP are subroutines of the Relocatable Program Library.

File Protect Feature

The first two files of the magnetic tape in a MTS environment are File 1 and 2 that contain absolute and relocatable programs. The BCS MT drivers provide the file protect feature to relieve the programmer of the responsibility for skipping over these two files. When protected, the first two files may not be referenced by user programs. A READ or WRITE request to the magnetic tape will operate on the third file of the MT as if it were the first physical file. The driver allows the programmer to declare and use up to 255 files in file protect mode. Any attempt to READ or WRITE beyond 255 files causes an EOT, regardless of the actual physical amount of tape used.

The magnetic tape driver may be set to the file protect mode, the unprotected mode, or both modes, when the BCS is constructed by PCS. The EQT entry determines the mode; for example:

a. To use the file protect mode, enter 1Ø,D,21,U1

b. To use unprotected mode,enter 1Ø,D.21

c. To use both modes, enter 1Ø,D.21,U1
1Ø,D.21

HP2Ø2Ø Magnetic Tape must be in I/O channels No. 10,11 because it is a relatively high-speed synchronus device which does not use DMA.

SIO System Dump

When software programs are configured and dumped using the SIO System Dump, the number of physical records written on the MT by PTS is reduced. The savings involved with programs such as the FORTRAN compiler significantly increase the speed of MTS. The software should contain its "non-standard" SIO module in this case.

APPENDIX C
STAND ALONE ENVIRONMENT

All of the HP software included in MTS may also be executed stand-alone; i.e. loaded from paper tape with the Basic Binary Loader.

Programming Compatability

Absolute Assembly Language programs which are assembled in MTS may be run stand-alone or added to MTS. This is possible because the linkages to MTS (through 77_8 and 106_8) are established at run-time, not at assembly-time.

Absolute programs assembled in stand-alone may be added to MTS if they follow the programming conventions given in Section IV.

Relocatable programs, written in Assembly Language, FORTRAN, or ALGOL, and compiled stand-alone or in MTS must be relocated using the Basic Control System, either in MTS or stand-alone. Absolute tapes punched by a stand-alone BCS must be run stand-alone (they cannot be run under MTS). Absolute tapes punched by a BCS in the MTS environment may be run in either environment; stand-alone or MTS. However, programs added to the magnetic tape may not use the BCS DEBUG subroutine.

GLOSSARY

Underlined terms in definitions are defined elsewhere in the glossary.

ABSOLUTE PROGRAMS

An absolute program is an object program with fixed memory address assignments for its instructions and data (cf. RELOCATABLE PROGRAMS). There are two classes of absolute programs: those generated by the Assembler which use the SIO drivers for non-interrupt input/output, and those relocated from relocatable code by the Basic Control System which use the BCS drivers for interrupt input/output. (Section IV.)

BATCH MODE

Batch is one of the operating modes of MTS. In batch mode, user programs, data, and directives are read from a batch input device, such as a card reader, without requiring operator intervention (cf. KEYBOARD MODE).

BASIC CONTROL SYSTEM

The Basic Control System (BCS) allows the user to relocate programs written in Assembly Language, FORTRAN and ALGOL, and to link them to BCS drivers and library subroutines of File 2. Relocated programs may be run in core or punched in absolute binary format on tape. (Section IV.)

CONTROL STATEMENT

In addition to the :PROG directives (Section III, IV) which are required to initiate them, the HP software programs -- FORTRAN, ALGOL, and Assembler -- require control statements to establish certain options for their operation. The control statement is the first statement processed, and it may be entered separately,

or as a part of the program. Control statements are described in the appropriate software manual.

DIRECTIVES

Directives are the user's means of communication with MTS; using them he may switch operating modes, print comments, suspend MTS, or run programs. (Section III.)

FILE 1

The magnetic tape of MTS is divided into three files, created by PTS. The first file, called File 1, contains absolute programs selected by the user. The first two programs must be .IPL. and S.SIO. The File 1 programs may be run by entering a :PROG directive, calling for them by identifier.

FILE 2

File 2 of the magnetic tape contains relocatable subroutines such as those of the Relocatable Library. These programs are linked with user programs by the Basic Control System whenever the user programs make external reference to them.

FILE 3

File 3 of the magnetic tape is a scratch file which is used by FORTRAN, the Symbolic Editor, the Cross-Reference Symbol Table Generator, and the Assembler. User programs may also use it for temporary storage of data.

IDENTIFIERS

Each program in File 1 has a ten-character identifier which is assigned by the user at PTS-time. The user may pick any identifiers he finds meaningful; however, a few programs have required identifiers. (Section I and Appendix A.) Directives use these identifiers to call programs for execution.

INTER-PASS LOADER (.IPL.)

.IPL.--the control program for MTS--consists of a core-resident part located in MTS BOOT and a tape-resident part which is the first program of File 1. .IPL. is responsible for loading programs from File 1, running and terminating them, switching operating modes, printing comments to the operator from the user, suspending MTS, and program chaining.

KEYBOARD MODE

Keyboard is one of the operating modes of MTS. In keyboard mode, the user's directives are entered through the keyboard, rather than as an integral part of the user's program and data, as in batch mode. This requires that an operator be in attendance, but provides greater dynamic flexibility. (Section II.)

MAGNETIC TAPE SYSTEM (MTS)

The Magnetic Tape System consists of a control program --.IPL.--and an ordered set of programs stored in Files 1 and 2 of the magnetic tape. The basic function of MTS is to load and turn execution control over to these programs. MTS must be configured using the Prepare Tape System and then initiated using a bootstrap created from MTS BOOT.

MTS BOOT

MTS BOOT is a separate program that contains the core-resident part of .IPL.. When MTS BOOT is configured with S.SIO and an SIO magnetic tape driver, a bootstrap tape is produced. This tape is used to initiate the operation of MTS.

OBJECT PROGRAMS

Source programs written by programmers must be compiled into machine instructions before they can be run. Programs, such as the FORTRAN and ALGOL Compilers and the Assembler, accept source programs and produce object programs which

consist of machine instructions (in either absolute or relocatable format) that carry out the operations specified in the source program.

PREPARE CONTROL SYSTEM (PCS)

The Prepare Control System program configures a Basic Control System for the loading and execution of relocatable programs.

PREPARE TAPE SYSTEM (PTS)

The Prepare Tape System accepts the user's absolute and relocatable programs and creates the two files on the magnetic tape required for MTS.

PROGRAM CHAINING

When a program has completed its execution, it can pass execution to another tape-resident program. (A BCS program cannot call in an SIO-environment program that does not include its own SIO module.) The program makes a termination call to .IPL. with -3 in the A-register and a specified program identifier. .IPL. then finds the specified program in File 1, loads it, and runs it.

RELOCATABLE PROGRAMS

Relocatable programs are object programs with relative, not fixed, memory assignments for instructions and data. These programs, produced by the ALGOL and FORTRAN Compilers and the Assembler, must be loaded and relocated by the Basic Control System before they can execute.

SIO DRIVERS

SIO (Software Input/Output) drivers are absolute programs that control the input and output of information on a specific device. They operate without the interrupt system on; i.e., only one device may be active at a time. The software programs, such as the FORTRAN Compiler, but not

BCS, and absolute user programs written in Assembly Language use SIO drivers. In MTS, several SIO drivers are gathered together to produce an SIO module. All modules reside on the magnetic tape and may be called by directives, but only one module is designated as the standard SIO module. (Section I.)

SIO MODULE

SIO modules are collections of from one to four SIO drivers for different devices. Since many different SIO modules may reside on the magnetic tape, different programs may use different I/O devices.

SIO SYSTEM DUMP

The SIO System Dump is an absolute program that punches SIO modules and HP software programs. The separate programs are loaded and configured; then the SIO System Dump punches a single absolute tape that includes all of them.

SOURCE PROGRAMS

Source programs are programs written in FORTRAN, ALGOL, and Assembly Language by the programmer. They must be compiled into object programs and relocated, if relocatable, before they can run on the computer.

S.SIO
(STANDARD SIO MODULE)

S.SIO is the identifier that the user must assign to the SIO module that he wants to be the standard SIO module. S.SIO is loaded into core by .IPL. between the execution of every user program.

SYSTEM LINK TABLE

The system link table is located in locations 101_8 through 107_8 and contains links to .IPL., the SIO drivers, and the last location of the user program. This table is updated every time a user program or SIO module of File 1 is loaded.

INDEX

INDEX