HP 13256A
FIRMWARE SUPPORT PACKAGE

Reference Manual

Part Number: 13256-90001
Printed: September 1977

Section I. INTRODUCTION

General Information

The HP 13256A is made up of a 9-track magnetic tape containing HP 2645A terminal source code, an HP 8080 Cross Assembler program, and various data files and utility programs. The assembler is designed to run on an HP RTE-II or RTE-III system. This allows you to use the RTE system file manager, editor and other library programs available on the RTE system. The system must contain an HP 7970B magnetic tape drive. The remaining descriptions in this manual assume that you are familiar with the HP RTE system, RTE file manager, and the RTE editor.

The HP 13256A tape files are described in table 1.

Table 1. HP 13256A Tape File Content

File Name	Description	Type	Length In Blocks
X13256	Transfer file containing bootstrap file loader	AS (4)	4
Terminal Code			
KY36CI	2645A Keyboard code	AS (3)	266
DC14FI	2645A Data Comm code	AS (3)	239
PT774I	2645A Main code	AS (3)	1466
IO260I	2645A I/O code	AS (3)	1029
MPTS2I	2645A Multipoint code	AS (3)	823
Cross Assembler Relocatable			
%ITL80	8080 Cross Assembler Relocatable Module	BR (5)	136
%RTERD	Assembly Utility Routines Relocatable Module	BR (5)	3
%. .MAP	Virtual Memory Package	BR (5)	2
%DCLVA		BR (5)	1
%VADD		BR (5)	5
XITL80	Transfer file to load 8080 assembler	AS (4)	2
&GNCRA	Programs to generate CRC and checksum characters	AS (4)	5
&GNCRB			90
&AMROM	Program to prepare AMD ROMs	AS (4)	43

AS = ASCII Source
BR = Binary Relocatable
block = 256 bytes

RTE System Software

File Manager

The RTE File Manager can be used to store, list, and dump source programs. However, it cannot be used to store or dump object programs. Object programs require special formatting for output. The utility programs described later in this manual are used to output the object code. If you have special output requirements, you can develop a simple application program based on the existing utilities.

Editor

The RTE Interactive Editor can be used to create and edit source programs. Refer to the RTE documentation for additional information on the editor. Note that the semicolon (";") normally used by the editor as a tab character is also used by the assembler as a delimiter in source programs. Before using the editor it is necessary to instruct the editor to use the colon (":") as the new tab character. This is done by typing "T:8:22") after the editor is called and before the first edit operation is begun.

Section II. THE ASSEMBLER

Relocatable Library: %ITL80, %RTERD, %..MAP, %DCLVA, %VADD

The assembler converts source code for the Intel 8080 processor into absolute object code. The assembler is designed to run on an HP RTE-II or RTE-III operating system. The peripheral equipment required is as follows:

- A source code input device: Disc, magnetic tape, or terminal cartridge tape.
- A listing output device: Line printer or terminal.
- An object code output device: Paper tape or terminal cartridge tape.

Assembler Differences

The HP cross assembler is very similar to the Intel assembler. Refer to the Intel Assembler Manual (Intel part numbers 98-301, 98-004C). There are minor differences between the HP cross assembler and the Intel 8080 assembler. These differences are in the following areas:

- Listing Format
- Pseudo Operators
- Number Bases
- Operators

Listing Format

The first statement in every program to be assembled must be a listing control statement. This statement causes assembled addresses and op codes to be listed in either octal or hexadecimal. The format statement is in the following form:

```
Op Code    Title/Comment
ASB,BIN    title
or
ASB,HEX    title
```

ASB,BIN causes the assembled address and op code fields to be listed in octal. ASB,HEX causes the fields to be listed in hexadecimal. The ASB is considered to be an operator and must be placed in the op code field otherwise an error will result. Normally the title field will contain the name and version of the program assembled and will be output to paper tape or cartridge tape with the assembled object code.

Pseudo Operators

The HP cross assembler does not provide the same set of pseudo operators used by the Intel assembler. Table 2 lists the pseudo ops that differ between the two assemblers. The Intel pseudo operators listed in the right half of the table are not used. In addition, the pseudo operators listed in the left half of the table are supported by HP but not Intel. The additional HP pseudo operators UNL, LIS, and SKP provide control of the assembly listing. The ORG $+exp pseudo operator allows you to create data storage blocks by continuing the assembly at the current program location ("$") plus some increment of bytes contained in the expression ("exp").

Table 2. HP and Intel Assembler Differences

HP Cross Assembler		Intel Assembler	
Name	**Description**	**Name**	**Description**
Assembly Listing Control			
UNL	Stop listing	—	—
LIS	Resume listing	—	—
SKP	Skip listing to top of form	—	—
Other			
ORG $+exp	Reset program counter	DS exp	Assign storage
—	—	TITLE	Create heading
—	—	SET	Change flag
—	—	IF,ELSE,ENDIF	Conditional branch
—	—	MACRO,ENDM	Define function

Number Bases

The Intel assembler allows you to specify number bases for data ("D" - decimal, "H" - hexadecimal, "B" - binary, "O" or "Q" - octal). In the HP assembler all data is assumed to be decimal unless specified as octal with the "Q". For example, 63 = 77Q.

Operators

The Intel assembler allows you to use the following operators:

MOD — Modulo
NOT — Not
AND — And
OR — Or
XOR — Exclusive Or
SHR — Shift Right
SHL — Shift Left

These operators are not supported by the HP cross assembler. The Intel assembler also allows you to use parentheses to indicate the order in which an expression is to be evaluated. The HP cross assembler ignores parentheses and always evaluates expressions from left to right. This means that operators (multiply, divide, etc.) do not determine the order in which an expression is evaluated.

Example: X = 1 + (3*4)

Intel:
```
X = 1  + (12)
  = 1  + 12
  = 13
```

HP:
```
X = 1  + 3 * 4
  = 4  * 4
  = 16
```

Example: Compute an index and add it to a base address.

```
BUFFER EQU 1000
         .
         .
         .
       LDA 10 * 12 - 5 / 5 + BUFFER
```

This would result in the contents of location 1023 being loaded into the accumulator.

Section III. SYSTEM GENERATION

The cross assembler and utility programs can be loaded into the system at generation time or on-line using the RTE Relocating Loader. Note that the system must have an HP 7970B magnetic tape drive. When the 13256A program library is loaded on-line a transfer file, X13256 is used. This file contains all of the commands required for the RTE File Manager to load the library. Once the entire library has been loaded, the assembler programs can be brought into the system work area. It may also be desirable to compile the utility programs at this time.

Loading the Program Library

The program library is transferred from magnetic tape to disc using commands stored on the X13256 file. The procedure is as follows:

```
:ST, <magnetic tape L.U.#>, X13256::<disc L.U. or CR>::-1
                                                    (Store the X13256 file)
:TR, X13256,<disc L.U. or CR>          (Store the tape files on the specified disc)
```

Note that if your RTE system uses a different logical unit number for the magnetic tape drive, the transfer file X13256 will have to be edited to correct the logical unit number. The required edit procedure is as follows:

```
:RU,EDITOR
/SOURCE FILE?
/X13256
/W4,6
/Z8/<your magnetic tape logical unit number>
/999
/EOF
/ER
```

A list of the contents of the X13256 file is given in figure 1.

```
:* PRINT TRANSFER FILE USED TO LOAD 13256A PACKAGE
:DU,X13256,6
:CN,6
:*
:* 2645A TERMINAL KEYBOARD CODE
:ST,8,&KY36C::1G::-1,AS
:* 2645A TERMINAL DATACOM CODE
:ST,8,&DC14F::1G::-1,AS
:* 2645A TERMINAL MAIN CODE
:ST,8,&PT774::1G::-1,AS
:* 2645A TERMINAL DEVICE I/O CODE
:ST,8,&IO260::1G::-1,AS
:* 2645A TERMINAL MULTIPOINT DATACOM CODE
:ST,8,&MPTS2::1G::-1,AS
:*
:* 8080 CROSS ASSEMBLER RELOCATABLE MODULE
:ST,8,%ITL80::1G::-1,BR
:* ASSEMBLER UTILITY ROUTINES RELOCATABLE MODULE
:ST,8,%RTERD::1G::-1,BR
:* VIRTUAL MEMORY PACKAGE
:ST,8,%..MAP::1G::-1,BR
:ST,8,%DCLVA::1G::-1,BR
:ST,8,%VAADD::1G::-1,BR
:* TRANSFER FILE TO LOAD 8080 ASSEMBLER
:ST,8,XITL80::1G::-1,AS
:*
:* PROGRAM THAT GENERATES CRC AND CHECKSUMS
:ST,8,&GNCRA::1G::-1,AS
:ST,8,&GNCRB::1G::-1,AS
:*
:* PROGRAM THAT PREPARES TAPES FOR AMD ROMS
:ST,8,&AMROM::1G::-1,AS
:*
:CN,8
::
```

Figure 1. X13256 File Contents

Loading the Assembler

The program library is loaded using the transfer command file as follows:

```
ON, FMGR    (Turn on the RTE File Manager)
::XITL80    (Execute commands in XITL80 file)
```

A list of the contents of the XITL80 file is given in figure 2.

```
:PU,%TEMP
:LG,20
:OF,ITL80,8
:OF,PASS1,8
:OF,PASS2,8
:OF,PASS3,8
:MR,%ITL80
:SA,LG,%TEMP
:ST,%TEMP,%T1,BR,1,1
:ST,%TEMP,%T2,BR,2,40
:LG,30
:MR,%T1
:MR,%..MAP
:MR,%RTERD
:MR,%DCLVA
:MR,%VADD
:MR,%T2
:PU,%TEMP
:PU,%T1
:PU,%T2
:RU,LOADR,99,6,0,15001
:PU,ITL80
:PU,PASS1I
:PU,PASS2I
:PU,PASS3I
:SP,ITL80
:SP,PASS1I
:SP,PASS2I
:SP,PASS3I
:LG
:CN,6
:CN,6
::
```

Note that %..MAP must be loaded even if a similar routine is already present in the RTE system.

Figure 2. XITL80 File Contents

Partition Size

The assembler is intended to execute in a background partition of the system. Performance of the assembler is dependent on the size of the partition. The larger the partition the faster the assembler can assemble a given program. The minimum partition size that should be used is 12K words. A partition size of 18K words will result in improved performance.

Preparing the Utilities

The source code for the utility programs is provided to allow you to modify the code to meet unique requirements. This means that the utility programs must be compiled before they can be used. The following procedures create relocatable code that can be stored in the library for later use.

Compile the CRC and Checksum Routines. GNCRA is a FORTRAN program and GNCRB is coded in assembly language. Use one of the following procedures depending on the FORTRAN compiler present in your system.

```
LG,5
MS,&GNCRA
LG,2
RU,FTN4,2,99
SA,LG,%GNCRA
```

or

```
LS,5
MS,&GNCRB
RU,ASMB,2,99
SA,LG,%GNCRB
RU,LOADR,99
```

```
RU,FTN4,&GNCRA,6,%GNCRA
```

```
LS,5
MS,&GNCRB
RU,ASMB,2,99
SA,LG,%GNCRB
```

```
CMMD1
RE,0/0GNCRA
RE,0/0GNCRB
```

At this point the program GNCRC could be run by entering "RU,GNCRC". If the program is not to be run at this time it is also stored in relocatable form on the disc as %GNCRA and %GNCRB.

Compile the AMROM Routine. AMROM is coded in FORTRAN.

```
LS,5
MS,&AMROM
LG,2
RU,FTN4,2,99
SA,LG,%AMROM
RU,LOADR,99
```

or

```
RU,FTN4,&AMROM,6,%AMROM
```

At this point the program AMROM could be run by entering "RU,AMROM". If the program is not to be run at this time it is also stored in relocatable form on the disc as %AMROM.

Section IV. USING THE ASSEMBLER

Programs are assembled using the following steps:

1. Mount the source code to be assembled (mag tape, tape cartridge, or disc file). If the source code is on a disc file you must enter MS,<disc file name> to load the file into the logical source area of the RTE system. Note that if the object tape is to be used with any of the utility programs described later in this manual, paper tape must be selected as the output.

2. Type RU,ITL80, <source L.U.>,<list L.U.>,<object L.U.>

   where: <source L.U.> is the logical unit number of the source device (2 if in the logical source area of the system disc).

   <list L.U.> is the logical unit number of the list device. The default device is the line printer (logical unit number 6).

   <object.L.U.> is the logical unit number of the device to output the assembled object code. The default device is the paper tape punch (logical unit number 4).

3. The code is assembled. Any errors detected are displayed on the system console as well as in the listing.

4. Make sure that there is at least 2 feet (61 cm) of trailer if paper tape is punched. This much trailer is required if the object tape is to be used as input to any of the utilities described later in this manual.

Object Code Format

If the object code is sent to a cartridge tape, the code is stored in the format used by the 13290A binary loader. (Refer to the 13290 Reference Manual, part no. 13290-90003, for additional information.) When the code is punched on a paper tape, the code is formatted in blocks of 256 bytes. This block format can be used as input to the special utilities to produce paper or magnetic tapes for making ROMs. (Refer to the discussion of utility programs later in this manual.)

Assembler Example

The following example shows the dialog produced when the assembler is used.

```
*ON,FMGR
:MS,TRMRTE                (Move the file to the logical source area of the system disc.)
FMGR 015
LS LU 2 TRACK 031         (System indicates the starting disc track of the logical source area to
                           be used.)
:RU,ITL80,2,6 (,4)        (Run the assembler using logical unit 2 as the source device, logical
                           unit 6 as the list device, and logical unit 4 as the default output
                           device. Normally the object code would be written to a cartridge tape
                           drive. Output is generated during the 3rd pass of the assembler.)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * (Printed at the start of the 1st pass)

  8080 INTEL ASSEMBLER. . . . .VERSION HP-2.0

  (A pause of up to several minutes depending on the size of the assembly)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * (Printed at the end of the 1st pass)
```

(If an error is detected it is displayed on the system console as well as appearing in the assembly listing.)

```
0 ERRORS FOUND IN ASSEMBLY CODE

:RT,ITL80 (Release the temporary disc tracks that were assigned to the assembler.)

$END FMGR
```

Linking Your Program To Existing Code

Your program can reference routines or variables in the terminal source code by using the absolute addresses of the desired code. The following example uses the Keyboard Code routine GETKEY to detect when a key is pressed. GETKEY is at location 44005Q in the version of code being used.

```
GETKEY EQU  44005Q ; DEFINE ADDRESS OF KEYBOARD ROUTINE
         .
         .
         .
; WAIT FOR KEYBOARD INPUT
SCAN   EQU  $
       CALL GETKEY ; MONITOR KEYBOARD
       JNZ  SCAN ; LOOP UNTIL KEY IS STRUCK
; PROCESS KEY
         .
         .
         .
```

Section V. UTILITIES

The utility programs allow you to generate object tapes that include 8-bit checksum and CRC 16 characters for each block of code. There is also a utility program that allows you to generate a tape for the preparation of ROM masks used by Advanced Micro Devices, Inc. The source code as well as the assembled reloctable binary forms of the utility programs are provided. The utilities are present on the same magnetic tape as the HP 2645A source code.

Checksum and CRC 16 Routine

Source: &GNCRA, &GNCRB

This program uses the output object tape produced by the assembler and produces a new object tape appending the checksum and CRC 16 characters to the end of each block of code. These three characters actually over write the last three characters of the assembled code. For this reason you should include three or more NOP instructions at the end of each ROM block in the program when check characters are required.

ROM Block	1st CRC 16 Character	2nd CRC 16 Character	Block Checksum	ROM Block	...

To use the checksum program proceed as follows:

1. Place the paper tape containing the assembled object code in the paper tape reader. Make sure there is at least two feet (61 cm) of trailer at the end.

2. If the program has been stored in relocatable form enter the following commands:

   ```
   LG,5
   MR,%GNCRA
   MR,%GNCRB
   RU,LOADR,99
   ```

   If the programs have not been previously compiled and stored, you must now do so. Use the steps given under "System Generation", and enter "RU,GNCRC".

3. The terminal will respond with "ENTER THE ROM BLOCK SIZE (K)". You should enter 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, or 24 depending on the type of ROM used. The ROM's normally used with the HP 2649A are 2K.

4. The program will then punch a new tape with the check characters. The check characters for each block will also be displayed at the terminal.

AMD ROM Control Tape Generator

Source: &AMROM

This program converts binary assembler code to hexadecimal code and generates a magnetic tape containing card images of the code. This tape can then be converted to punched cards on a system with a card punch using a simple file copy process. The cards can then be used by a ROM manufacturer (such as Advanced Micro Devices) to prepare code ROMs. If your RTE system has a card punch, the cards can be punched directly by entering the card punch logical unit number in the RU command for the hex output device. To use the AMROM utility proceed as follows:

1. Place the paper tape containing the binary object code (the output of the GNCRA utility) in the tape reader.

2. If the program has been stored in relocatable form, enter the following commands:

```
LG,5
MR,%AMROM
RU,LOADR,99
```

   If the program has not been previously compiled and stored, you must now do so. Use the steps given under "System Generation" and enter the following:

```
RU, AMROM, <binary input L.U.#>,<list L.U.#>,<Hex output L.U.#>,
           <verification input (cards) L.U.#>,<user terminal L.U.#>
```

Default device assignments (typical)	
device	L.U.#
paper tape reader	5
line printer	6
9-track mag tape	8
card reader	7
terminal (system console)	1

3. The program will print "(V)ERIFY OR OUTPUT TAPE (O)?"
   You should respond "V" or "O".

4. If the output function "(O)" is selected, the program will print the following:

   "SKIP NEXT 2K ON INPUT TAPE? YES(Y) OR NO(N)"

   Entering "Y" will cause a block of 2K bytes to be skipped on the input tape. The prompt will then be repeated.

   Entering "N" will cause the following prompt:

   "PART #?
    (TYPE 8 CHARACTERS, NO IMBEDDED SPACES)"

   When the part number has been entered, 2K bytes of code are read and the hexadecimal data output. After the 2K block is converted, the program prints the following:

   "END? YES(Y) or NO(N)?"

   Entering "N" will cause the program to return to the "SKIP 2K..." prompt.

5. If the verify function "(V)" was selected in step 3, blocks of 2K bytes on the input tape are compared with the verification input (normally a card reader). Bytes that do not compare properly will be printed on the list device (line printer). The program will then print the following at the terminal:

   "SKIP NEXT 2K ON INPUT TAPE? YES(Y) OR NO(N)"

   Entering "Y" will cause a block of 2K bytes to be skipped on the input tape. Entering "N" will cause a 2K block of bytes on the input device to be compared with data read from the verification input device. The following prompt is then printed:

   "END? YES(Y) OR NO(N)?"

   Entering "N" will cause the program to return to the "SKIP NEXT 2K..." prompt.