LEAP

UNIVERAL PROGRAMMER/TESTER MODEL : LEAPER-10

User's Manual

TABLE OF CONTENTS

I. Introduction

The LEAPER-10 is a universal programmmer/tester characterized by its handiness, power-saving and exactness. With connection by the parallel interface of the printer, the LEAPER-10 makes it possible to promote the usage of the conventional programmer. Its portablity with battery and easy installation lend itself to working with the notebook PCs.

The concordance in the operation style is one feature of the LEAPER-10's software. Selective procedure control makes it easy to read, program, and verify the device. The project file is another feature provided to execute the batch instructions for the system. Parameters appended with the main program is another option to execute directly under DOS or the batch file.

The devices supported by the LEAPER-10 are as follows:
EPROM, EEPROM, Flash EPROM, BPROM, Nonvolatile RAM, Serial EEPROM, Micro Chip PLD, PAL, GAL,PEEL,EPLD...........

II. Standard Accessories

1. the LEAPER-10 host x 1

2. 25-pin printer connect cable x 1

3. DC 12.0v **2000 mA** power adaptor x 1

III.Software Installation

1. Run setup.exe in CD-ROM

2. The minimum of memory to run the LEAPER-10 is 300K bytes. If system memory is insufficient to run the LEAPER-10, please re-adjust the allocation of memory, e.g. release some TSR's or change some other setting to gain more memory.

3. If the system software is not able to run, first please check if the hardware installation is proper, then check from step 5. Supposing the system software is still not able to run, it is recommended to backup the data files to another directory. Make installation from step 2 once again with the backup copy if it is usable or install from step 1 if it is unuseble.

IV. Hardware Installation

1. Turn off the power of the PC and the LEAPER-10.

2. Connect the printer output of the PC with the LEAPER-10 by the 25-pin printer connect cable.

3. Plug the DC adaptor with AC power, then plug with the DC input of the LEAPER-10; or load two alkaline batteries into the battery case.

4. Turn on the power of the PC and the LEAPER-10.

5. Execute 'LP10.EXE'.

V. Hardware Initialization

It is necessary to initialize the LEAPER-10 when the system software is running without the LEAPER-10 connected or the power of the LEAPER-10 is turned off for power saving. Steps to initialize the LEAPER-10 are as follows:

1. Turn off the power of the LEAPER-10.

2. Connect the printer output of the PC with the LEAPER-10 by the 25-pin printer connect cable.

3. Turn on the power of the LEAPER-10.

4. Execute 'Initiate system' under the item 'Option' or press [F5] to initialize the hardware directly.

VI. Points to Be Taken Care of

1. Be sure to connect the printer connect cable properly to the parallel port, not to the RS-232 or other 25 pins interface.

2. When the cable is to be taken apart from the LEAPER-10 or the PC, never pull the cable without holding the connector to avoid bad contact.

3. Never use any DC adaptor which does not come with the LEAPER-10.

4. Keep watching the following cases to avoid any unexpected damage.

   a. Never move the device on the TEXTOOL or turn off the power of the LEAPER-10 and the PC or take apart the printer connector cable while the system is programming, verifying, or reading the device. (LED light to represent working)

   b. Never place the device reversely on the TEXTOOL. It must be placed with the gap upwards and its bottom coincided with the TEXTOOL's. Please refer to the following figure.

   c. Before processing the device, be sure that its manufacturer and the type are both correct, especially when the device is the programmable logic device, single chip, BPROM..., etc.

d. The alkaline battery is the only choice when it is necessary to work with the battery. The battery with insufficient power is not recommended to use lest it should do unexpected damage to the device or the system.

e. Never execute any control program which will take care of the interface of the printer while the power of the LEAPER-10 is turned on.

5. The device is not supposed to be put on the TEXTOOL with the following conditions:

   a. Turn on or off the power of the LEAPER-10.

   b. The power of the LEAPER-10 is turned on with that of the PC turned off.

   c. Self-testing is processing.

6. The LEAPER-10 will make more power exhausted with the following conditions:

   a. The power of the LEAPER-10 is turned on with that of the PC turned off. (absolutely forbidden)

   b. The device processed is bad or power exhausted, or it is placed on the TEXTOOL wrongly or reversely.

VII.Simple Troubleshooting:

1. With system running, if the LED is not light, or some other abnormal response arises, please do step by step as follows:

   a. Check if it is set at 'DEMO' mode in the 'Parallel port no' under 'Option' and reset this option with the correct port no.

   b. Check if the power switch of LEAPER-10 is set at 'ON'. Then make sure the DC adaptor is connected well with proper power supply.

   c. Check if the connection among the PC, printer cable and the LEAPER-10 is good, then reconnect them all.

   d. (1) Exit the LEAPER-10 system software.
      (2) Turn off the power of the LEAPER-10.
      (3) Turn off the power of the PC.
      (4) Re-turn on the power of the PC and the LEAPER-10.
      (5) Execute LP10.EXE
      (6) Execute 'self Test' of 'Option' to check if the hardware is OK.

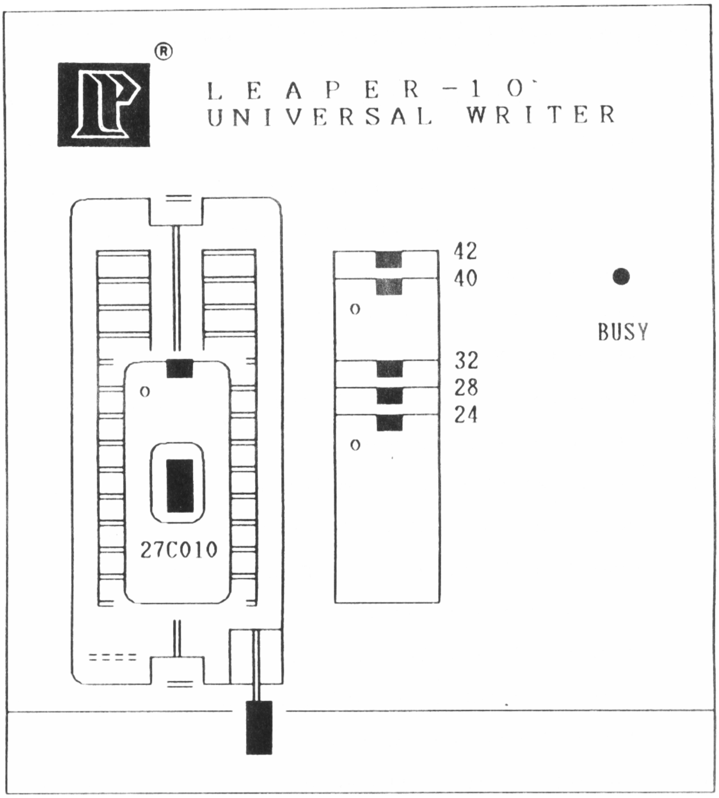
2. If the check sum is unsteady, please do as follows:

   a. Check if the manufacturer and type of the device is correct.

   b. Slow down the reading speed of the memory devices or the single chips.

   c. Check if the device connect with the TEXTOOL properly or the pins of the device had been oxidized. Then reconnect them all.

   d. Check if the TEXTOOL has aged because of overusage or has been oxidized. (affected by the pins of the device, especially used devices)


®
LEAPER-10
UNIVERSAL WRITER
42
40
32
28
24
BUSY
27C010

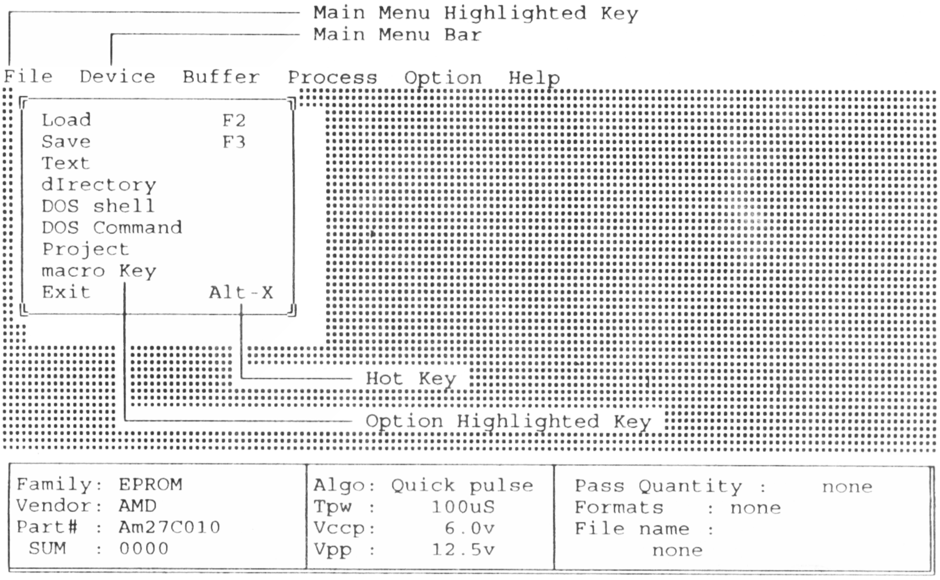
VIII. Specification of Executable Files in LEAPER-10

INSTALL.BAT	the installation file .... (executable)
LP10.EXE	the main program .... (executable)
LP10.SET	the file to store parameters .... (data)
L10E.MNU	the definition of the main menu ..... (system reference)
L10C.MNU	the same as the above
LP10.PRJ	the project file (sample) .... (text)
LP10.KEY	the definition of macros .... (text)
DEVICE.PIN	the pinout map of the devices .... (data)
DEVICEC.PIN	the same as the above
PRODUCT.LST	the brief of other associated products .... (text)
LP10.CHI	operation information in Chinese .... (text)
LP10.ENG	operation information in English .... (text)
*.PDR	system driver files .... (overlay)
*.DRV	device driver files .... (overlay)
*.LIB	device parameter libraries .... (system reference)
LIB*.LIB	device libraries for test .... (system reference)
MENU*.MNU	the definition of the menu .... (system reference)

[NOTES] All the library files and data files in the above are for system reference only. Any modification to those files will probably cause the system to be unstable, not executable, even damages to the device, etc. It is also not encouraged to access those files with some text editors, for those files had been compressed.

IX. Software Operation Guide :

Main Menu Operation Function



```
Family: EPROM        Algo: Quick pulse    Pass Quantity :    none
Vendor: AMD          Tpw  :     100uS     Formats    : none
Part# : Am27C010     Vccp:       6.0v     File name  :
 SUM  : 0000         Vpp  :     12.5v           none
```

```
[F1] Help   [<>^v] Move   [Enter] Enter select   [Esc] Escape
```

[COMMENTS]

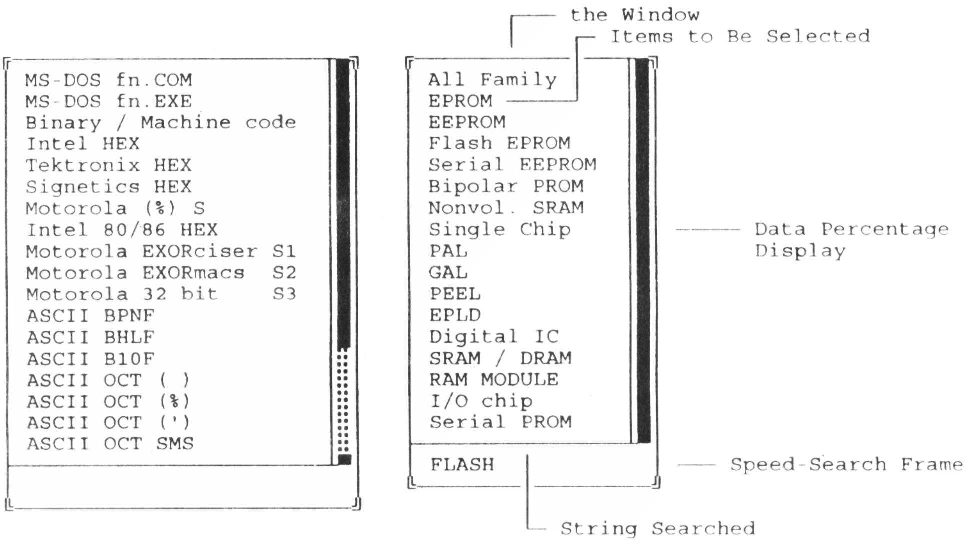
1. Function keys:

[^], [v]	Move the cursor UP or DOWN.
[<], [>]	Move the cursor LEFT or RIGHT.
[Enter]	Pull down the menu table, or execute the chosen item.
[Esc]	Close the window

[Examples] Two ways to execute FILE/LOAD function :

1. Press keys by sequence
   (1) press [F] key to pull down File function table.
   (2) press [L] key to execute Load function.
2. Using Hot key :
   press [F2] to execute FILE/LOAD function.

Option List Operation Window



[COMMENTS] Speed-search function is applied to all the option lists, such as device family, manufacturer, type, files format and mislaneous parameters setting... and so on. After typing a few synonym, you will find the settings you need.

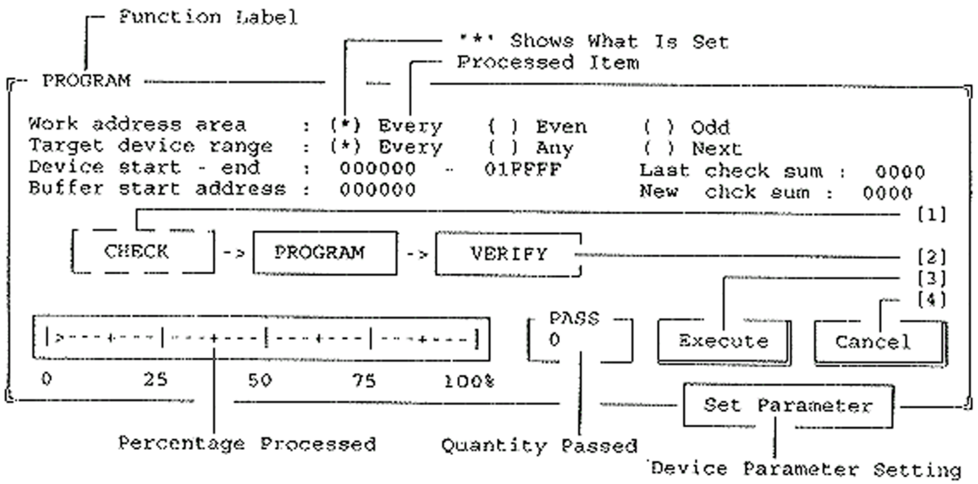
1. Function keys:

[^], [v]	Move the cursor UP or DOWN.
[Enter]	Select the item.
[Esc]	Exit.
[Home], [End]	Move the cursor to the first or to the last item.
[PgUp], [PgDn]	Move the cursor to the upper page or to the lower page.
[0] - [Z]	Input characters.
[?]	Magic character, any non-specific one.
[< Backspace]	Delete the last character in the string.

2. The speed-search function works in sequence is from left to right, then from up to down. Candidates found in the upper part supersede those in the lower part.

```
[Examples]  1. Supposed that 'GAL' is to be set,
               (1) Press [G], the cursor moves where ( Sin 'g'..) is.
               (2) Press [A], the cursor moves where ( 'GA' L ..) is.
               (3) Press [L], the cursor moves where ( 'GAL'     ) is.
               (4) Press [Enter], 'GAL'is set.

            2. Supposed that 'Intel 80/86 HEX' is to be set,
               (1) Press [I], the cursor moves where ( B 'i' n ..) is.
               (2) Press [N], the cursor moves where ( B 'in'  ..) is.
               (3) Press [T], the cursor moves where ( 'Int'el ..) is.
               (4) Press [Enter], 'Intel 80/86 HEX'is set.

            3. Using magic character [?] to set at 'Tektronix HEX,'
               (1) Press [T], the cursor moves where ( In 't' e ..) is.
               (2) Press [?], the cursor moves where ( In 'te' ...) is.
               (3) Press [K], the cursor moves where ( 'Tek' tr.. ) is.
               (4) Press [Enter], 'Tektronix HEX' is set.

            4. Press 'S1' in sequence, the cursor moves where
               'Motorola EXORciser S1' is.

            5. Pressing [< Backspace] will delete the last character
               in the searched string. And the cursor will move to
               the previous candicate.
               (1) Press [I], the cursor moves where ( B 'i' n ..) is.
               (2) Press [N], the cursor moves where ( B 'in'  ..) is.
               (3) Press [T], the cursor moves where ( 'Int'el ..) is.
               (4) Press [< Backspace], the cursor moves back where
                   ( B 'in'  ..) is.
```

Device Operation Window



```
PROGRAM

Work address area     : (*) Every     ( ) Even      ( ) Odd
Target device range   : (*) Every     ( ) Any       ( ) Next
Device start - end    :  000000  -  01FFFF      Last check sum :  0000
Buffer start address  :  000000                 New  chck sum :  0000

   CHECK  ->  PROGRAM  ->  VERIFY

 |>---+---|---+---|---+---|---+---|      PASS      Execute    Cancel
 0       25      50      75      100%    0

                                              Set Parameter
```

[1]	Procedure(s) not executed. (Dotted frame)
[2]	Procedure(s) to be executed. (Solid frame) <1> and <2> conditions are switched by [Space] or [Enter].
[3]	Start executing.
[4]	Exit.

[COMMENTS] Function keys:

[^], [v], [<], [>]	Move the cursor UP, DOWN, LEFT, and RIGHT.
[Space], [Enter]	Start or end the item.
[Esc]	Press once to move the cursor to 'Cancel'. Press twice to exit.
[Home], [End]	Move the cursor to the top or the bottom.

Device Parameter Setting Window

```
┌─ DEVICE PARAMETER ──────────────────────────────────────────────────────────┐
│ Read or verify speed.. ( ) Slow     ( ) Middle   (*) Fast                    │
│                                                                               │
│ Vccp voltage ......... ( ) 5.00v    (*) 6.00v    ( ) 6.25v    ( ) 6.50v      │
│                                                                               │
│ Vpp  voltage .........   12.5v   [ - ] [ + ]                                  │
│                                                                               │
│ Programming algorithm. < Quick pulse        >                                 │
│                                                                               │
│ Program pulse width .. < 100uS  >                                             │
│                                                                               │
│ Maximal tries ( n )...   25      [ - ] [ + ]                                  │
│                                                                               │
│ Over program pulse ... (*) none     ( ) 1n mS    ( ) 3n mS    ( ) 4n mS      │
└───────────────────────────────────────────────── │ Return option │ ──────────┘
                                                            │
                                                        return to
                                               "Device Operation Window'
```

[COMMENT] This window is invoked by executing 'Set Parameter'.

HEX Data Input Function (HEX EDIT)

```
                ┌─ Function Label
                │                 ┌─ HEX Data Input Frame
   ┌────────────┴────────────────────┐
   │  Copy block of buffer      │    │
   │  Source start : [ 00000 ]       │
   │           End : [ 0FFFF ]       │
   │  Target start : [ 00000 ]       │
   │ * Input range : 0 ~ 3FFFF       │
   └─────────────────────────────────┘
```

[COMMENTS] 1. This function is applicable to Copy, Change, Delete, Verify, etc.

2. Function keys:

[0] - [F]	HEX data input code.
[<], [>]	Move the cursor LEFT or RIGHT.
[< Backspace]	Delete the previous character.
[Enter]	Input data.
[Esc]	Exit.

<< 1 >> File Management Function Window

```
File
┌──────────────────────┐
│ Load           F2    │
│ Save           F3    │
│ Text                 │
│ dIrectory            │
│ DOS shell            │
│ DOS Command          │
│ Project              │
│ macro Key            │
│ Exit          Alt-X  │
└──────────────────────┘
```

		Memory	uP	PLD	TEST
Load	[F2]	✓	✓	✓	

[PURPOSE] Load files into the buffer which include data such as binary code, HEX code, fuse map, etç. To load files associated with memory device, one must first select File Format, then input the file name, and the buffer start address. Then one can decide if the buffer will be cleared in advance by Fill function, which will fill the buffer using 00(HEX) or FF(HEX). Supported file formats are as follows:

1. MS-DOS fn.COM	12. ASCII BPNF
2. MS-DOS fn.EXE	13. ASCII BHLF
3. Binary / Machine code	14. ASCII B10F
4. Intel HEX	15. ASCII OCT ( )
5. Tektronix HEX	16. ASCII OCT (%)
6. Signetics HEX	17. ASCII OCT (')
7. Motorola (%) S	18. ASCII OCT SMS
8. Intel 80/86 HEX	19. ASCII HEX ( )
9. Motorola EXORciser S1	20. ASCII HEX (%)
10. Motorola EXORmacs S2	21. ASCII HEX (')
11. Motorola 32 bit S3	22. ASCII HEX (,)
	23. ASCII HEX SMS

[NOTES]
1. The buffer range supported for files with binary code format is up to 8 Mega bits.
2. The size of files with HEX code format which can be loaded (including Intel, Motorola HEX...) is dependent on the buffer size setting, such as 64K, 128K, and 256K. Please refer to buffer size setting function.
3. If the size of the file to be loaded is larger than the buffer size or the address with adding the size of the file is over the range of the buffer, what over the range of the buffer will not be loaded.

Load PAL file	Memory	uP	PLD	TEST
			√	

[PURPOSE] Load the fuse map data of PAL into the buffer, then convert the data to those of GAL. (This is a feature only for GAL16V8 and GAL20V8)

[Example] Supposed GAL device number has been chosen,
(1) Select the device number to convert in the option list of PAL's.
(2) Input the file name.
(3) Edit the electronic signature (dependent on whether the device supports this feature).

[NOTES] If other device number (such as PEEL 18CV8, 22V10, etc) is expected to be converted, one can contact with the suppliers of PEEL to get the associated software.

load Vector	Memory	uP	PLD	TEST
			√	

[PURPOSE] Load the data file of the programmable logic device test vector into the buffer. When the fuse map file is loaded through the function of Load, the test vector is automatically loaded if it is included. The test vector may be built by the function of generating the vector provided by the complier, which can be referred to its manual.

[Example] Test vector data file

```
V00001 000000000N0HLLHHLHHN*
V00002 000010000N0HLLHHLHHN*
V00003 000001000N0HLHHHLHHN*
```

Vector Symbol 'V00001' line number

'1'	Output '1' to IC
'0'	Output '0' to IC
'H'	IC output 'H'
'L'	IC output 'L'
'C'	Output '0' > '1' > '0' to ICí@
'K'	Output '1' > '0' > '1' to ICí@
'N'	Power pin (Vcc or GND)
'*'	row end code

[NOTE] 1. No compiler is included in this product. As to such information, one can contact with the suppliers of all the device or associated software suppliers. (such as ABEL, PALASM, ORCAD, etc)
2. The length limitation of whole test vector data file is 64K bytes.

Save	[F3]	Memory	uP	PLD	TEST
		✓	✓	✓	

[PURPOSE] Save data in the buffer to the files. Data included is binary code, HEX code, fuse map, etc. To save data as files associated with memory device, the File Format must first be selected, then the file name, and the buffer start and end addresses where data are to be saved. Supported file formats are as follow:

1. Binary / Machine code	10. ASCII B10F
2. Motorola (%) S	11. ASCII HEX ( )
3. Motorola EXORciser S1	12. ASCII HEX (%)
4. Motorola EXORmacs S2	13. ASCII HEX (')
5. Intel Intellce 8/MDS	14. ASCII HEX (,)
6. Intel80/86 HEX (MCS86)	15. ASCII HEX SMS
7. Tektronix HEX	16. ASCII OCT ( )
8. ASCII BPNF	17. ASCII OCT (%)
9. ASCII BHLF	18. ASCII OCT (')
	19. ASCII OCT SMS

[NOTES]

1. The buffer range supported for files with binary code format is up to 8 Mega bits.
2. The size of files with HEX code format which can be saved (including Intel, Motorola HEX...) is dependent on the buffer size setting, such as 64K, 128K, and 256K. Please refer to buffer size setting function.
3. The system will ask if the old file is to be overwritten, if the input file name has already existed. If it is the case, the file to be overwritten will first be renamed to *.BAK, then the new file is saved.

Text	Memory	uP	PLD	TEST
	✓	✓	✓	

[PURPOSE] Process the designated file using the text mode.

.Edit	Use the specified text editor to load the designated file into the buffer.
.View	Use the specified text editor to list file names and view the designated file.
.Set Editor	Specify the file name of the text editor and its path.

[Examples]

1. Supposed C:\PE2\PE2 TEST.DOC is specified, the 'Edit' function will always use C:\PE2\PE2 in the current directory to load TEST.DOC.

2. Supposed C:\PE2\PE2 !.DOC is specified, and PRIMARY.ROM has been loaded, the 'Edit' function will use C: \PE2\PE2 in the current directory to load PRIMARY.DOC. That is, 'PRIMARY' in PRIMARY.DOC is derived from that of 'PRIMARY.ROM'; 'DOC' is derived from that of 'C:\PE2\PE2!.DOC'. Other setting will result in other combination.

!.ext Automatically the working file name is taken as main name; the extention name is replaced by that set from 'ext'. For example,

```
            Set Editor: C:\PE2\PE2!.ASM
         Working file : C:\WORK\TEST.HEX
     What 'Edit' does : C:\PE2\PE2 C:\WORK\TEST.ASM
```

& Automatically the working file name is taken as the whole name. For example,

```
           Set Editor : C:\PE2\PE2 &
         Working file : C:\WORK\TEST.HEX
     What 'Edit' does : C:\PE2\PE2 C:\WORK\TEST.HEX
```

3. View the text file.
Supposed TEST.ROM has been loaded, then in the current directory:

'*.*'	:	List all files.
'f.*'	:	List all files with TEST as main name.
'f.ASM'	:	List TEST.ASM
'f.DOC'	:	List TEST.DOC
'f.LST'	:	List TEST.LST
'f.PAL'	:	List TEST.PAL
'f.PSD'	:	List TEST.PSD
'f.XPT'	:	List TEST.XPT

The above is default setting. The file name also can be directly input until the file is to be viewed.

	Memory	uP	PLD	TEST
Directory	✓	✓	✓	✓

[PURPOSE] List all filenames and those assoicated data in the current directory.

	Memory	uP	PLD	TEST
DOS Shell	✓	✓	✓	✓

[PURPOSE] Return to DOS temporarily. One can type 'exit' to return to this system.

	Memory	uP	PLD	TEST
DOS Command	✓	✓	✓	✓

[PURPOSE] Execute DOS instructions or other programs without leaving this system.

	Memory	uP	PLD	TEST
Project	√	√	√	√

[COMMENTS] One can edit the project file using text mode to execute some batch instructions. Contents of the project file may inlcude device, manufacturer, type, fixed file name, and other fixed actions. When LEAPER-10 starts, LP10.PRJ is loaded as the project file. LP10.PRJ can be modified using normal text editor such DOS EDIT or PE2.
There are three kinds of function:

. Execute	Execute contents of the project file
. Load	Load the project file
. View	View contents of the project file

[Example] Definition of LP10.PRJ

```
#DEF_PROJECT_MENU {
                   ' Test Project'———————— procedure comments
                   ' PCB01 U25 ROM 27C010 Project'———————┘
                   }
#PROFUN0
      {
      'FC' 'All Family' [CR] 'ATMEL' [CR] 'AT2817A' [CR]
      }
#PROFUN1          /AUTOEXEC
      {
      'DC' 'EPROM'  [CR] 'AMD'  [CR] 'Am27C010' [CR]
      'FL' 'BINARY' [CR] 'C:\WORK\PCB-U1.ROM' [CR] '0' [CR] 'N'
      'PP'
      }
```

Procedure 'Test Project' .... executes actions defined by #PROFUN0.
Procedure ' PCB01 U25 ROM 27C010 Project'executes actions defined by #PROFUN0
The procedure included with '/AUTOEXEC' string will be automatically executed whenever the system starts.

[NOTES]
1. Length of the project file is limited to be below 8K Bytes.

2. The function of space compression is highly recommended for the text editor used to edit the project file.

3. Before writing the project file, it is necessary to execute step by step and write down those steps.

4. ' or " can be used to describe the text data such as device, manufacturer, type and file name, etc.
   For example,

```
      'intel' [CR]
      'C:\TEST\KEY-1.JED' [CR]
      '12340' /CR
      "'74244" .... If the first character in the string
                    is a digit, it must be specfied by
                    adding " under speed-search condition..
```

5. The control keys can be replaced by the following text :

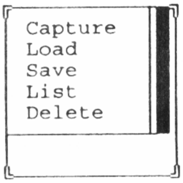
[ENTER]	Symbol:	[CR] , /CR , [ENTER] , /ENTER , or ','
[ESC]	Symbol:	[ESC] , /ESC
[TAB]	Symbol:	[TAB] , /TAB

Other key symbols :

[UP]	[DOWN]	[LEFT]	[RIGHT]
[PGUP]	[PGDN]	[HOME]	[END]
[INS]	[DEL]	[BACK]	
[F1]	[F2]	[F3]	[F4]
[F5]	[F6]	[F7]	[F8]
[F9]	[F10]		

macro Key

Memory	uP	PLD	TEST
√	√	√	√



[PURPOSE] One can record some fixed input actions as a control key, which will subsequently take place of those actions.

. Capture	Define macro key (those to be defined are [sF1] - [sF10]).
. Load	Load macro file. (extention name is .KEY).
. Save	Save defined macro key to a file.
. List	List defined macro key and purposes.
. Delete	Delete some macro key.

[NOTES] 1. Define macro keys: Select one from [sF1] to [sF10] at Capture: [ ], then type at most 8 characters as purposes. Upon this window disappears, the system records literally every key pressed until [Ctrl][M] is pressed. Please use HOT keys, don't use cursor control keys.

2. Save macro file: If 'LP10.KEY' is used, it will be loaded automatically when the system starts next time.

Exit	[Alt-X]	Memory	uP	PLD	TEST
		✓	✓	✓	✓

[PURPOSE] Quit the system, and return to DOS.
There are three options to select :

```
Press [N]o ........... Continue
Press [S]ave ......... Save opeation parameters to the
                       disk then exit.
Press [Y]es .......... Exit without saving parameters.
```

<< 2 >> Device Function Window

Device

```
┌──────────────────────┐
│ Category         F7  │
│ Manufacturer     F8  │
│ Type number      F9  │
│ History          F10 │
└──────────────────────┘
```

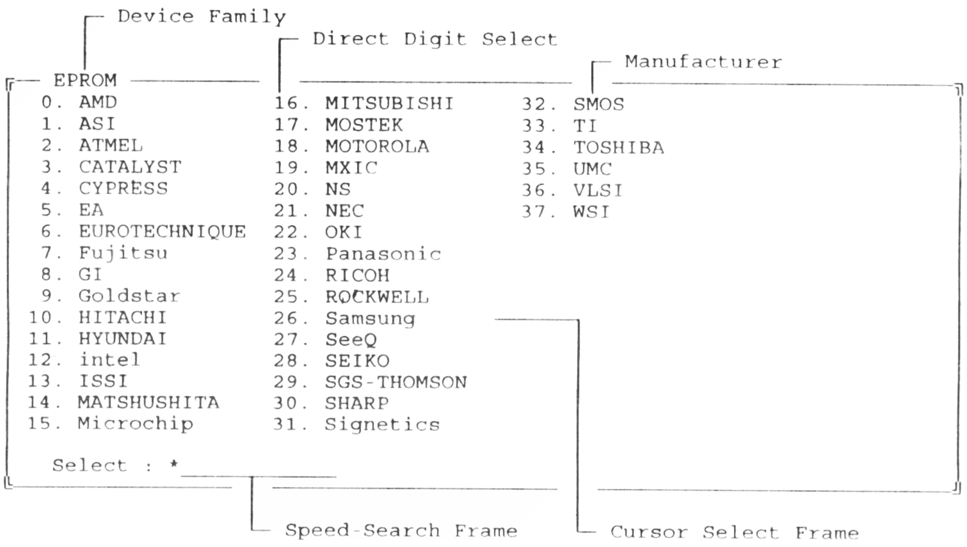
Category	[F7]	Memory	uP	PLD	TEST
		✓	✓	✓	✓

[PURPOSE] Sort out the devices by their family such as EEPROM, FLASH EPROM, Single Chip, PAL, etc. Then chooses the manufacturer and type of that device family.

[NOTES] Those not sorted by manufacturer such as TTL, COMS, ... can be selected here.

Manufacturer	[F8]	Memory	uP	PLD	TEST
		✓	✓	✓	✓

[PURPOSE] Select manufacturer and type.(based on the previous selected device family)



```
EPROM
  0. AMD               16. MITSUBISHI      32. SMOS
  1. ASI               17. MOSTEK          33. TI
  2. ATMEL             18. MOTOROLA        34. TOSHIBA
  3. CATALYST          19. MXIC            35. UMC
  4. CYPRESS           20. NS              36. VLSI
  5. EA                21. NEC             37. WSI
  6. EUROTECHNIQUE     22. OKI
  7. Fujitsu           23. Panasonic
  8. GI                24. RICOH
  9. Goldstar          25. ROCKWELL
 10. HITACHI           26. Samsung
 11. HYUNDAI           27. SeeQ
 12. intel             28. SEIKO
 13. ISSI              29. SGS-THOMSON
 14. MATSHUSHITA       30. SHARP
 15. Microchip         31. Signetics

   Select : *
```

[NOTES] 1a. One can input a string in speed-search frame.
For example : List all manufacturers with INT or ICS.
List all device with '27C or -7

1b. The first character in the input string must be an English letter, and it must be added with the symbole ['] if it is a digit.
For Example, '244 .... TTL 74244

1c. If only one type is matched with the input condition, it will be selected directly without listing.

1d. In 'Speed-Search Frame', one can input digits directly to select the device.
For example, 12 [Enter] (it is set as Intel here)

1e. Pressing [Tab] can switch to 'Cursor Select Frame'.

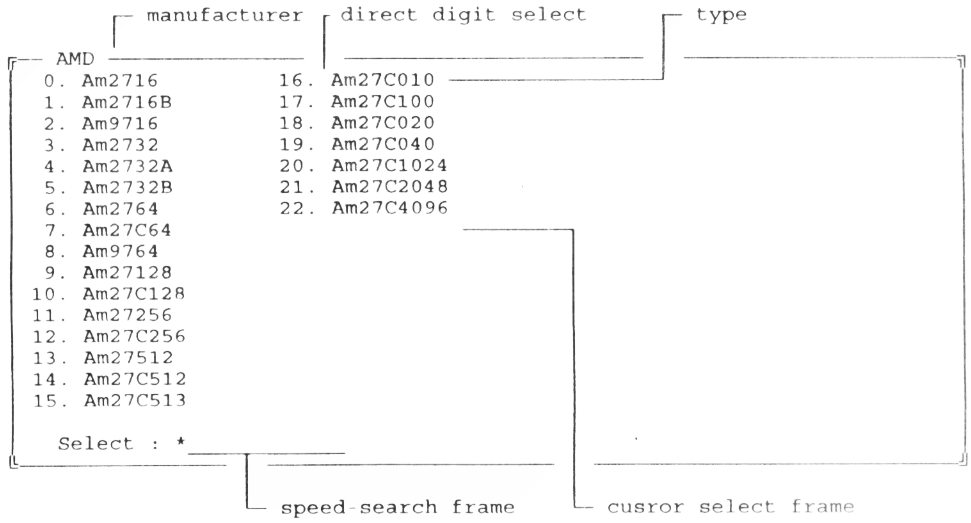
2a. In 'cursor select frame', supported keys are as follow :

[^], [v], [<], [>]	Move the cursor UP, DOWN, LEFT, or RIGHT
[PgUp], [PgDn]	Move to the previous page or to the next page
[Home], [End]	Move the cursor to the first item or to the last one.
[Tab]	switches between 'Speed-Search Frame' and 'Cursor Select Frame'.
[Enter]	Set the item.
[Esc]	Exit.

2b. The speed-search function works in sequence is from left to right, then from up to down. Candidates found in the upper part supersede those in the lower part. Magic character '?' is also supported here.

		Memory	uP	PLD	TEST
Type number	[F9]	✓	✓	✓	✓

[PURPOSE] Select the device type number (according to the previous selection).



```
AMD
  0. Am2716          16. Am27C010
  1. Am2716B         17. Am27C100
  2. Am9716          18. Am27C020
  3. Am2732          19. Am27C040
  4. Am2732A         20. Am27C1024
  5. Am2732B         21. Am27C2048
  6. Am2764          22. Am27C4096
  7. Am27C64
  8. Am9764
  9. Am27128
 10. Am27C128
 11. Am27256
 12. Am27C256
 13. Am27512
 14. Am27C512
 15. Am27C513

   Select : *
```

[NOTES] 1. The system will store data in the buffer to EMS/XMS or to the buffer file (U1.BUF) automatically, or vice versa whenever the kind of device is changed.

2. If there is no or not enough (< 1 MB) EMS / XMS, or the buffer file (U1.BUF) does not exist, the function in 1. will not work.

3. Please refer to examples of manufacturer.

		Memory	uP	PLD	TEST
History	[F10]	√	√	√	√

[PURPOSE] Display the last 8 sets of selection issued in the system for the next quick selection. Every set of selection is composed of the device, manufacturer, and its number.

[Example] Supposed there had been three sets of selection issued, i.e., EPROM AMD Am27C512, EEPROM intel i2817A, and GAL AMD AmPAL16L8-5, press [F10] will display a list as follows:

```
EPROM          AMD            Am27C512
EEPROM         intel          i2817A
GAL            AMD            AmPAL16L8-5
```

[NOTES]
1. Number of sets in the list is limited to 8 ones by first in, first out (FIFO).
2. Whatever set selected in the list will subsequently become the last one in the list.
3. If there will be more than 8 sets, the oldest set or the least accessed one will be deleted from the list.

<< 3 >> Buffer Management Function Window

Buffer

```
Edit            F4
Disassemble
Used map
eXtra buffer
Fill
dIvide
comBine
Copy
chAnge
```

```
                                                            Memory  uP   PLD   TEST
Edit                              [ F4 ]                       √     √
```

```
current active segment          HEX code                                      ASCII code
+20000 .0 .1 .2 .3 .4 .5 .6 .7 .8 .9 .A .B .C .D .E .F       ASCII Code
010200:EB 07 90 76 0F 44 52 56-CB 0E 1F BB 81 08 26 8A   k__v_DRV K__;__&_
010210:04 88 07 46 43 81 FB 8C-09 75 F3 E8 DB 00 80 3E   ___FC_{_ _ush[__>
010220:95 08 00 74 07 BB EC 09-B0 C3 88 07 80 3E 96 08   ___t_;l_ 0C___>__
010230:00 74 1E 80 3E 97 08 00-75 17 C6 06 96 08 01 BB   _t__>___ u_F____;
010240:31 84 BE F3 83 8A 07 88-04 46 43 81 FB 6F 84 75   1_>s____ _FC_{o_u
010250:F4 89 26 96 84 89 2E 98-84 A1 F6 6A A3 F8 6A 8A   t_&___._ _!vj#xj_
010260:1E 81 08 2A FF D1 E3 8B-87 8E 00 FF D0 8B 26 96   ___*_Qc_ ____P_&_
010270:84 8B 2E 98 84 8E 06 8D-08 BE 0B 09 8B 1E 92 08   __._____ _>______
010280:8A 04 26 88 07 43 46 81-41 8C 09 75 F3 CF F8 00   __&__CF_ A__usOx_
010290:C3 00 B8 00 D6 05 D8 00-C9 23 D2 00 CC 00 B8 00   C_8_V_X_ I#R_L_8_
0102A0:B8 00 B2 04 6E 04 02 05-4F 5A ED 00 32 04 B8 00   8_2_n___ OZm_2_8_
0102B0:B8 00 B8 00 B8 00 B8 00-C6 06 0B 09 01 C6 06 6D   8_8_8_8_ F____F_m
0102C0:09 05 C3 E8 41 0B E8 AE-09 E9 77 5E E8 4F 22 E9   __ChA_h. _iw^hO"i
0102D0:A5 09 E8 16 21 E9 9F 09-E8 A5 1C E8 99 09 A1 82   %_h_!i__ h%_h__!_
0102E0:85 A3 C3 83 A1 84 85 A3-C5 83 E9 56 5E C6 06 0B   _#C_!__# E_iV^F__
0102F0:09 00 E8 2A 5E E9 4B 5E-C3 C6 06 20 09 01 A0 F2   __h*^iK^ CF_ ___r
   offset address in row
```

[PURPOSE] Edit contents of memory.

[Example]

```
           current active segment = 20000H
            offset address in row = 102F0H
              correlative address = 302F0H
```

[COMMENTS] Entering HEX edit function , press [F10], one menu will be pulled down as follows :

```
Go to
Jump
Edit/Dump
Search
Search next
Used map
Copy
Change
Delete
Verify
Fill
Checksum
Swap
Invert
```

Go to [^D]

[PURPOSE] Move the cursor to any address within the current buffer and show contents of that address.

[Exmaples]

Goes to 01234H address.

```
┌──────────────────────────────┐
│ Start address for display    │
│                              │
│ Change dump   :  [ 01234 ]   │
│ * Input range : 0 ~ 3FFFF    │ ──── dependent on
└──────────────────────────────┘      the buffer size
```

[NOTES] The range of the buffer is dependent on the setting of the buffer size.

```
64 KByte = 0 - 0FFFF ( plus current active SEGMENT )
128 KByte = 0 - 1FFFF.
256 KByte = 0 - 3FFFF.
```

Jump [^G]

[Example] Jump to address 56789H.

```
┌──────────────────────────────┐
│ Start address for display    │
│                              │
│ Change dump   :  [ 56789 ]   │ ──── target address
│ * Input range : 0 ~ FFFFF    │
└──────────────────────────────┘
```

Edit/Dump [F4]

[PURPOSE] Modify the contents within the buffer, and shift between edit mode and view mode, switching to HEX or ASCII by [Tab].

Function keys:

[^], [v], [<], [>]	Move the cursor UP, DOWN, LEFT, or RIGHT
[Home], [End]	Move the cursor to the top or to the bottom.
[^PgUp], [^PgDn]	Move forward or backward by 1000H
[^W] or [Shift-PgUp]	Move to the previous segment
[^Z] or [Shift-PgDn]	Move to the next segment
[PgUp], [PgDn]	Move to the previous page or to the next page
[F7]	switch rotationally three data display modes, including 8 bits HEX, 12 bits HEX, and 16 bits HEX.
[Esc]	Exit.

[NOTES] 1. PgUp and PgDn of [Shift-PgUp] and [Shift-PgDn] is referred to those on the KEYPAD.

64K bytes	10000H
128K bytes	20000H
256K bytes	40000H

2. Buffer size / Memory size

Buffer size	Memory size
64K bytes	10000H
128K bytes	20000H
256K bytes	40000H

Search

[PURPOSE] Search data within memory blocks. Another menu will be pulled down as follows :

```
 Byte     <--- Search BYTE in the memory block.
 Word     <--- Search WORD in the memory block.
 ASCII    <--- Search ASCII in the memory block.
```

[Examples]

Byte :

```
 Search data
 Source start  :  [ 00000 ]
          End  :  [ 0FFFF ]
 Target   data :  [ 55 ]
* Input  range :  0 ~ 3FFFF
```

Word:

```
 Search data
 Source start  :  [ 00000 ]
          End  :  [ 0FFFF ]
 Target   data :  [ 55AA ]
* Input  range :  0 ~ 3FFFF
```

ASCII:

```
 Search data
 Source start  :  [ 00000 ]
 Target  data  :  [ LEAPER-10
* Input  range :  0 ~ 3FFFF
```

(Supposed buffer size is 256 KB)

[NOTES] The working range is dependent on the buffer size. If the target address is beyond the working range, please execute Jump (^G) to some other segment where the target address exists.

Search next

[PURPOSE] Search for the next candicate for the specified condition.

Used map

[PURPOSE] Show the used map of the buffer.

Function keys:

[^W] or [Shift-PgUp]	Move to the previous segment
[^Z] or [Shift-PgDn]	Move to the next segment
[PgUp], [PgDn]	Move forward or backward by 10000H in the working range.
[Esc]	Exit.

[NOTES] 1. PgUp and PgDn of [Shift-PgUp] and [Shift-PgDn] is referred to those on the KEYPAD.

2. Buffer size .... Memory size

Buffer size	Memory size
64K bytes	10000H
128K bytes	20000H
256K bytes	40000H

Copy

[PURPOSE] Copy data in the memory block to another block.

[Example] Copy data between address 00002 and 00005 in the buffer to the block starting from address 01000H.

```
 Copy block of buffer
 Source start  :  [ 00002 ]
          End  :  [ 00005 ]
 Target start  :  [ 01000 ]
* Input range  :  0 ~ 3FFFF
```

```
original  00000: 00 01 02 03 04 05 06 07 08 09 ...
                       └─────────┘
          01000: 31 32 33 34 35 36 37 38 39 3A ...
                          ↓
  copied  00000: 00 01 02 03 04 05 06 07 08 09 ...
                       └─────────┘
          01000: 02 03 04 05 35 36 37 38 39 3A ...
                 └─────────┘
```

(Supposed the buffer size is 256 KB)

[NOTES] The working range is dependent on the buffer size.

Change

[PURPOSE] Exchange data within one address range for those within another address range.

[Example] Exchange data within 00000-00005 for those
           within address starting from 01000H.

```
Change block
Source start  :  [ 00000 ]
         End  :  [ 00005 ]
Target start  :  [ 01000 ]
* Input range  :  0 ~ 3FFFF
```

```
 original   00000: 00 01 02 03 04 05 06 07 08 09 ...
            01000: 31 32 33 34 35 36 37 38 39 3A ...
                             |
exchanged   00000: 31 32 33 34 35 36 06 07 08 09 ...
                     |  |  |  |  |  |
            01000: 00 01 02 03 04 05 37 38 39 3A ...
                    ( Supposed the buffer size is 256 KB )
```

[NOTES] The working range is dependent on the buffer size.

Delete

[PURPOSE] Delete data within the buffer and move what came after the deleted ones toward where the deleted ones exists.

[Example] Delete data between address 00002 to 00005 in the buffer.

```
Delete block
Source start  :  [ 00002 ]
         End  :  [ 00005 ]

* Input range  :  0 ~ 3FFFF
```

```
 original   00000: 00 01 02 03 04 05 06 07 08 09 ...
                         └──────────┘ deleted
            3FFF6: ... 52 13 32 22 12 12 7A 22 12 12
                           |
  deleted   00000: 00 01 06 07 08 09 0A 0B 0C 0D ...
            3FFF6: ... 12 12 7A 22 12 12 FF FF FF FF
                    ( Supposed the buffer size is 256 KB )
```

[NOTES] The working range is dependent on the buffer size.

Verify

[PURPOSE] Distinguish data within two different addresses by listing their difference.

Function keys:

[S]	suspends listing; presses any key to resume.
[Esc]	quits this function.

[NOTES] The working range is dependent on the buffer size.

Fill

[PURPOSE] Fill data into the whole or partial buffer, and there are some options as follows :

. All bit 1	fill the whole buffer with 1 (FFh).
. All bit 0	fill the whole buffer with 0 (00h).
. User define	fill the partial buffer with bytes the user defined.
. Sequential word	fill the whole buffer with sequential words. For example, 00000: 00 00 02 00 04 00 .. FC FF FE FF
. Sequential byte	fill the whole buffer with sequential bytes. For example, 00000: 00 01 02 03 04 05 .. FC FD FE FF
. Random data	fill the whole buffer with random data.

Checksum

[PURPOSE] Calculate the checksum of data within the whole or partial buffer in the active segment.

. All	Calculate the checksum of data within the whole buffer.
. Portion	Calculate the checksum of data within the partial buffer.
. Make	Generate checksum integer. Calculate the checksum of data within the partial buffer. In the address the user defined, fill some proper data to make the checksum ends with 00h.

[Example] Generate the checksum integer of one 27256.

```
         source start address = 00000
                  end address = 07FFF
               target address = 07EFF ( no data here )

         original   00000: 00 01 02 03 04 05 06 07 08 09 ...
                    07EF6: 52 13 00 00 00 00 00 00 00 00 (7EFF)
                    07FF6: FF FF FF FF FF FF 7A 22 12 12
 original checksum = 79F0H

        processed   00000: 00 01 02 03 04 05 06 07 08 09 ...
                    07EF6: 52 13 00 00 00 00 00 00 00 10 <
                    07FF6: FF FF FF FF FF FF 7A 22 12 12
      new checksum = 7A00H
```

Swap

[PURPOSE] Swap data in the whole buffer.

. Even/Odd byte — Swap data within odd addresses and even address in the whole buffer. For example,

```
original 00000: 00 01 02 03 04 05 ...
                  X     X     X
 swapped 00000: 01 00 03 02 05 04 ...
```

. High/Low nibble — Swap high/low nibble of data in the whole buffer. For example,

```
original  00000: 00 01 02 03 04 05 ...
                 || || || || || ||
 swapped  00000: 00 10 20 30 40 50 ...
```

[NOTES] The working range is dependent on the buffer size.

Invert

[PURPOSE] Invert data in the partial buffer.
(Exchange 0 for 1, and vice versa)

[Example]

```
original   00000: 00 01 02 03 04 05 ...
                  || || || || || ||
inverted   00000: FF FE FD FC FB FA ...
```

[NOTES] The working range is dependent on the buffer size.

Disassemble

Memory	uP	PLD	TEST
	✓		

[PURPOSE] Disassemble data in the buffer for the single chips. Supported single chips are MCS-48 (8748) and MCS-51 (8751). Three options to start this function are as follows:

```
┌──────────────┐
│ Screen       │  <—— screen output only
│ Printer      │  <—— screen with printer output
│ File         │  <—— screen with file output
├──────────────┤
│              │
└──────────────┘
```

Commands supported by Disassemble :

```
U  {range} Dump disassembly.
D  {range} Dump last type memory data
DB {range} Dump memory byte.
DW {range} Dump memory word.
DD {range} Dump memory double word.
DA {range} Dump memory ASCII.
M {range}{address} move data block.
C {range}{address} Change data block.
V {range}{address} Verify two data block.
X {range}  Delete data block.
? or HELP  Display operating expression.
?   {data} Display HEX,BIN,DEC,ASCII.
={data}{data} Accumulator two operator.
H{data}{data}Logic acculator two word
VER   Display system version.
CLS   Clear screen and go home.
Q     Quit system return master menu.
```

[NOTES] 1. The working memory range is 0-0FFFF.
2. If one selects 'screen with printer output', the printer is forbidden to share the same port (LPT) with LEAPER-10.

Used map	Memory	uP	PLD	TEST
	√	√		

[PURPOSE] Show the used map of the buffer.

[NOTES] Please refer to page 25.

Extra Buffer (Encryption table)	Memory	uP	PLD	TEST
		√		

[COMMENTS] This is generally used for the data encryption in MCS-51. It is difficult to get the correct data from the device after encryption unless the encryption table is known.

Supported functions :

.Edit	Edit the encryption table
.Load	loads the encryption table into the buffer.
.Save	stores the encryption table in the buffer to the file .
.Lock/Unlock	with the encryption table, locks or unlocks data in the buffer.

FILL	Memory	uP	PLD	TEST
	√	√		

[PURPOSE] Fill data into the whole or partial buffer.

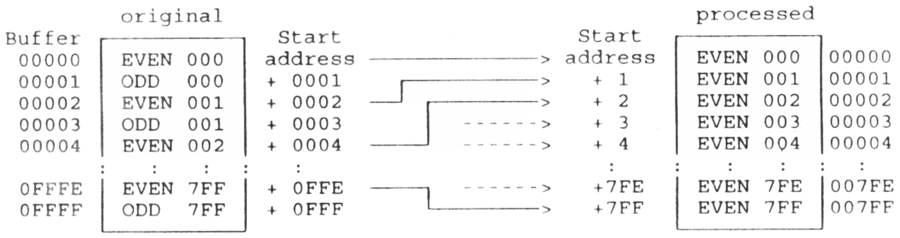
[NOTES] Please refer to page 27.

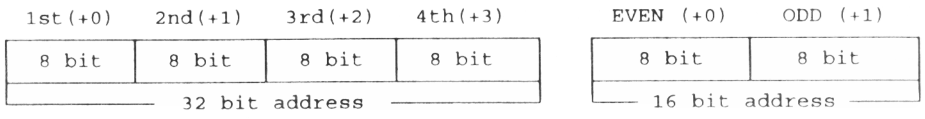
	Memory	uP	PLD	TEST
Divide	✓	✓		

[COMMENTS]

. 16 bits source	Collect only those in odd address or only those in even address from the 16-bit data in the buffer to form the 8-bit data.
. 32 bits source	Collect only those in odd address or only those in even address from the 32-bit data in the buffer to form the 8-bit data.
. 64 bits source	Collect only those in odd address or only those in even address from the 64-bit data in the buffer to form the 8-bit data.

[Example] Collect only those in even address from the 16-bit data in a 27512 to form the 8-bit data.





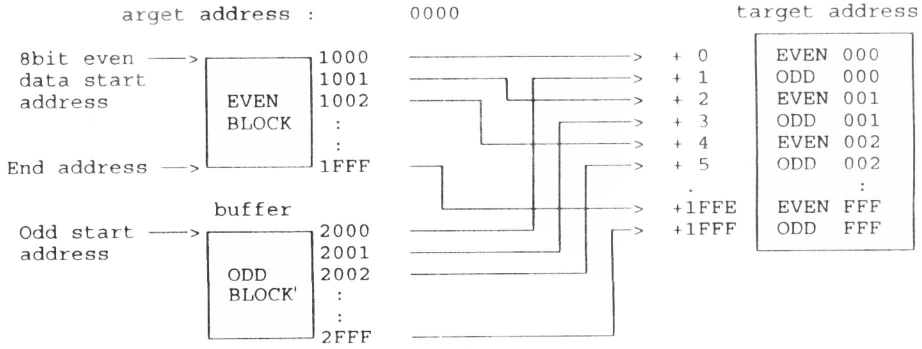
	Memory	uP	PLD	TEST
Combine	✓	✓		

[COMMENTS] . 8 to 16 bits Combines two 8-bit data into one 16-bit data.
. 8 to 32 bits Combines four 8-bit data into one 32-bit data

[Example]

Combine two 8-bit data into one 16-bit data.

Even data block	1000 - 1FFF
Odd data block	2000 - 2FFF
arget address :	0000



Copy	Memory	uP	PLD	TEST
	✓	✓		

[PURPOSE] Copy data in the memory block to another address.

[NOTES] Please refer to page 25.

Change	Memory	uP	PLD	TEST
	✓	✓		

[PURPOSE] Exchange data within one address range for those within another address range.

[NOTES] Please refer to page 25.

Buffer Function Window for programmable logic device

```
Edit JEDEC          F4
edit Signature
view Vector
Fill
```

Edit JEDEC [F4]	Memory	uP	PLD	TEST
			✓	

[PURPOSE] Edit the fuse map of the programmal logic device.

Function keys:

[^], [v], [<], [>]	Move the cursor UP, DOWN, LEFT, or RIGHT
[PgUp], [PgDn]	Move to the previous page or to the next page.
[space]	changes state
[^F]	fill all the fuse with intact data.
[^L]	fill one single fuse with intact data.
[^C]	fill all the fuse with blown data.
[Esc]	Exit.

edit Signature	Memory	uP	PLD	TEST
			✓	

[PURPOSE] Edit the electronic signature. (only GAL)

[NOTES] The electronic signature of GAL makes it easy to recognize the device with the encryption table.

	Memory	uP	PLD	TEST
View Vector			√	

[PURPOSE] List and view the vector data of the programmable logic device.

Function keys:

[PgUp], [PgDn]	Move to the previous page or to the next page.
[Esc]	Exit.

	Memory	uP	PLD	TEST
FILL			√	

[PURPOSE] Fill the whole fuse buffer with intact or blown data.

[NOTES] It is dependent on the PLD selected to fill with intact or blown data.

<< 4 >> Process Window

```
Process
┌────────────────────┐
│ Read            ^R │
│ blank Check     ^C │
│ Program         ^P │
│ Verify          ^V │
│ Security        ^B │
│ Erase           ^E │
│ Test            ^T │
└────────────────────┘
```

		Memory	uP	PLD	TEST
Read	[^R]	√	√	√	

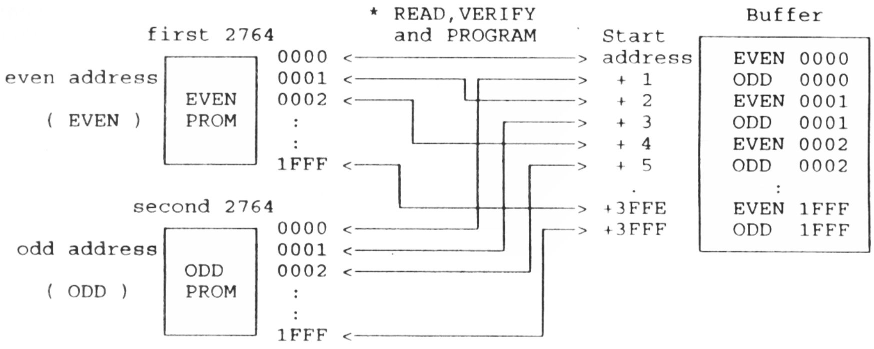
[PURPOSE] Read data stored in the device into the specified buffer range. If the frame of 'VERIFY' or 'LIST ERROR' is solid, the function in the frame will be executed with reading; otherwise, the function will be skipped. The new checksum will be displayed remarkably if it is different from the previous one.

1. Three kinds of address area to process source data :
   a. all addresses : data in the device is read into the buffer with address starting from 00.
      Work address area : (*) Every ( ) Even ( ) Odd

   b. even addresses : data in the device is read into the buffer all with even address starting from where specified.
      Work address area : ( ) Every (*) Even ( ) Odd

   c. odd addresses : data in the device is read into the buffer all with odd address starting from where specified.
      Work address area : ( ) Every ( ) Even (*) Odd

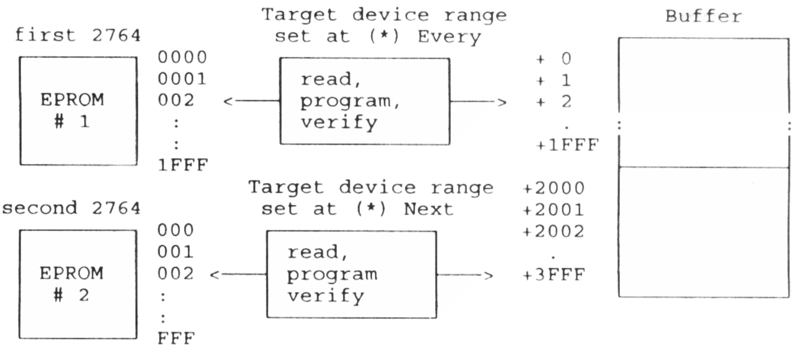
Illustration below will show the processing of even or odd addresses while reading, programming or verifying the device.

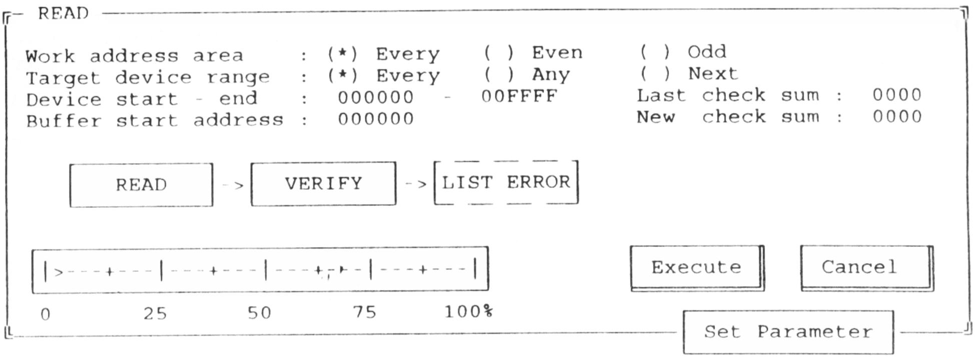


2. Three kinds of device range to process the device :
   a. It will be taken care automatically by the system if the selected device range is the whole one, (*) Every, or the range succeeds to the last one, (*) Next.
      Target device range : (*) Every ( ) Any ( ) Next
      Target device range : ( ) Every ( ) Any (*) Next

   c. One can input the start and end addresses if the selected device range is specific, (*) Any.
      Target device range : ( ) Every (*) Any ( ) Next

Illustration below will show the processing of reading, programming and verifying the device respectively with three kinds of device range.





```
READ

Work address area      :  (*) Every    ( ) Even      ( ) Odd
Target device range    :  (*) Every    ( ) Any       ( ) Next
Device start - end     :   000000   -  00FFFF        Last check sum :   0000
Buffer start address   :   000000                    New  check sum :   0000

     READ    ->   VERIFY   ->  LIST ERROR

 |>---+---|---+---|---+-+-|---+---|         Execute        Cancel
 0       25      50      75      100%
                                                 Set Parameter
```

The function in the frame with dotted lines will be not executed; the one with solid lines will be executed. The status of execution can be swapped with [Space] or [Enter].

		Memory	uP	PLD	TEST
Blank Check	[^C]	✓	✓	✓	

[PURPOSE] Checks if there were already data in the device before it is to be programmed.

[NOTES] 1. Most of the deivces with encryption will be checked to be blank. There is no identifying the device without other data (e.g., ID) included.
2. Some devices damaged will also be checked to be blank as those in 1.

		Memory	uP	PLD	TEST
Program	[^P]	✓	✓	✓	

[PURPOSE] Program the device with data in the marked range of the buffer or with data in the fuse map.

1. Normal procedure for programming the memory devices :
   Blank Checking -> Programming -> Verifying
2. Normal procedure for programming the single chips :
   Blank Checking -> Programming -> Verifying -> Encryption
3. Normal procedure for programming the programmable logic devices :
   Blank Checking -> Programming -> Verifying -> Encryption -> Logic testing

[NOTES] In the logic testing for the programmable logic device, there must be vector data included in the JEDEC files or the vector data is loaded separately into the buffer.

		Memory	uP	PLD	TEST
Verify	[^V]	✓	✓	✓	

[PURPOSE] Verify the data in the device with those in the marked range of the buffer, and list the data and their addresses when there is difference.

		Memory	uP	PLD	TEST
Security	[^B]		✓	✓	

[PURPOSE] Program the security fuse of the device.

		Memory	uP	PLD	TEST
Encryption			✓		

[PURPOSE] Program the single chip with the encryption table in the Extra Buffer.

[NOTES] This is a feature only for MCS-51.

		Memory	uP	PLD	TEST
Erase	[^E]	✓	✓	✓	✓

[PURPOSE] Erases the contents of the device to make it to be programmable.

[NOTES] This is a feature only for electronic erasable devices. (EEPROM, PEEL, GAL...)

		Memory	uP	PLD	TEST
Test	[^T]			✓	✓

[PURPOSE]
1. For the programmable logic device (PLD):
   tests the logic function of the PLD by vector data.
2. For IC , SRAM, DRAM, I/O CHIP :
   tests one time the function of the device.

[NOTES] Only the logic function of the device is tested here, no other test parameters included such as speed, Vcc voltage limit, input voltage level, fans out, etc.

		Memory	uP	PLD	TEST
Loop test					✓

[PURPOSE] Repeatedly tests the function of the device for a long period to observe its stability. This test can be interrupted by pressing [Esc] key.

[NOTES] The temperature of the device must be always monitored to avoid the damage to the device and subsequent damage to the system.

Search

Memory	uP	PLD	TEST
			√

[PURPOSE] Search for the number of the device wanted, and list all numbers of its compatible ones.

[NOTES] The logic function is the only feature used to search for the device, no other parameters included such as speed, Vcc voltage limit, input voltage level, fans out, etc.

<< 5 >> Option System Parameter Function Window

```
 Option
┌──────────────────────┐
│ Buffer size          │
│ Initiate system   F5 │
│ Parallel port no     │
│ Menu level           │
│ Country              │
│ Hardware test        │
│ self Test            │
│ clear Window     aF4 │
└──────────────────────┘
```

Buffer size

Memory	uP	PLD	TEST
√	√		

[PURPOSE] The maximum value of this parameter will be dectected whenever the system starts. As for the buffer editing and the file processing, this parameter restricts the working range of some functions such as Search, Copy, Change, Load HEX ..., etc.

```
                          range of data in the buffer
┌─────────────┐
│  64K bytes  │ ──── 00000 - 0FFFFH
│ 128K bytes  │ ──── 00000 - 1FFFFH
│ 256K bytes  │ ──── 00000 - 3FFFFH
│             │
└─────────────┘
```

[NOTES]

1. The buffer size must be at least 64K bytes to keep the system work well. If there is no enough memory in the PC, a message will be shown as follows:

   There isn't enough memory to execute LEAPER-10.

2. The minimum of memory to run the LEAPER-10 is 300K bytes. If system memory is insufficient to run the LEAPER-10, please re-adjust the allocation of memory, e.g, release some TSR's or change some other setting to gain more memory.

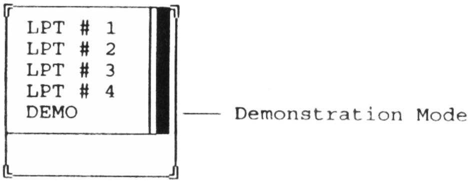
Initiate system	[F5]	Memory	uP	PLD	TEST
		√	√	√	√

[PURPOSE] Initiates LEAPER-10's hardware while running the system sofware. This function may be applied when LEAPER-10 is not yet connected or its power is switched off for the sake of power saving while the system has already run.

[NOTES] If initialization fails, the system software will request the use select the parallel port number to connect with LEAPER-10.

Parallel port no	Memory	uP	PLD	TEST
	√	√	√	√

[PURPOSE] Set the parallel port number to connect with LEAPER-10.



[NOTES]
1. The hardware decoding of LPT # 1 - 4 is based on the parallel port addresses recognized by BIOS upon PC system booting.
2. The default number is LPT#1.
3. Under demonstration mode, the system will simulate all functions to process the device.

Menu level	Memory	uP	PLD	TEST
	√	√	√	√

[COMMENTS] There are two kinds of menu, 'Easy' or 'Powerful' to select.

'Easy'	Support only basic functions. Every function is with one hot key for quick and easy operation.
'Powerful'	Support all functions, partial of which are with the hot key. This is the default setting.

[Example]

```
                Easy menu
        ┌──────────────────────────────────┐
        │  F7     Select device            │
        │  F10    Part history             │
        │  R      Read device into buffer  │
        │  C      Blank Check              │
        │  P      Program device           │
        │  V      Verify device            │
        │  E      Erase                    │
        │  F2     Load file form disk      │
        │  F3     Save file to disk        │
        │  F4     Edit buffer              │
        │  I      Parallel port no         │
        │  M      Menu level               │
        │  F5     Initiate system          │
        │  F1     Help                     │
        │ Alt-X   Exit                     │
        └──────────────────────────────────┘
```

[NOTES] When in 'Easy' function table, one can execute 'Menu level' to switch to 'Powerful'.

self Test

	Memory	uP	PLD	TEST
self Test	✓	✓	✓	✓

[PURPOSE] Proceed self testing of hardware respectively for items such as power, high voltage supply, impulse generation, logic signals, high voltage switches, ground switch,..., etc. One of three options shown below is to be selected before proceeding this test:

```
┌──────────────────┐
│ Screen           │  ──── Screen output only
│ Printer          │  ──── Screen with printer output
│ File             │  ──── Screen with file output
├──────────────────┤
│                  │
└──────────────────┘
```

A list below will be shown if everything is alright.

```
UNIVERSAL PROGRAMMER AND TESTER
Self test function  Version 1.00  1994 December

    [S] ..... Stop list, Any key to continue.
 [ESC]/[C] .. Interrupt test procedure.

Signal unit:   - OK -
Clock unit :   - OK -
I/O Signal :   - OK -
XTAL Drive :   - OK -
Power unit :   - OK -
Vcc Drive  :   - OK -
Vpp Drive  :   - OK -
Vhh Drive  :   - OK -
GND Drive  :   - OK -
```

[NOTES] 1. No device is permitted to put on the TEXTOOL while the self test is proceeding.
2. The self test will not proceed properly in demonstration mode.
3. If one selects 'screen with printer output', the printer is forbidden to share the same port (LPT) with LEAPER-10.

clear Window	[Alt-F4]	Memory	uP	PLD	TEST
		✓	✓	✓	✓

[PURPOSE] Close whatever opened windows in the screen.

<< 6 >> Help Function Window

```
    Help
┌──────────────────────────┐
│ Help                  F1 │
│ Information              │
│ Clock/calendar           │
└──────────────────────────┘
```

Help	[F1]	Memory	uP	PLD	TEST
		✓	✓	✓	✓

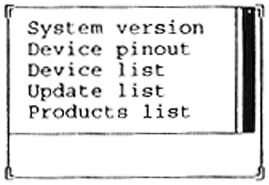
[PURPOSE] View the operation information of the system.

Function keys:

[^], [v], [<], [>]	Move the cursor UP, DOWN, LEFT, or RIGHT
[^Home], [^End]	Move the cursor to the first or to the last line of the text
[Home], [End]	Move the cursor to the very left or to the very right of the text
[PgUp], [PgDn]	Move to the previous page or to the next page.
[F5]	Zoom in or out the text. to the next page.
[^S]	Search for the string input by the user.
[^L]	Search for the next candicate string.
[^A]	Wrap or unwrap the text
[Tab]	Move the cursor to the next title.
[Esc]	Exit.

	Memory	uP	PLD	TEST
Information	✓	✓	✓	✓

[PURPOSE]   View the mislaneous information of the system.



—— LEAPER-10 system software version
—— partial device pinout maps
—— device list supported by the system
—— history of system software update
—— brief of other related products of LEAP

[NOTES]   Function keys are referred to those of 'Help'.

	Memory	uP	PLD	TEST
Clock/calendar	✓	✓	✓	✓

[PURPOSE]   Open digital clock or calendar window, pressing [Space] to swap.

[NOTES]   Both windows are shown in English only.

X. Direct or batch instructons under DOS

[COMMENTS] Some parameters are to be added together with 'LP10' to execute as batch instructons. Refer. to 'Project' for syntex of procedure.

[Examples] 1. Contents of PRO.BAT are as follows:

```
                                   ┌───── 'Process' menu 'Program' function
                                   │ ┌─── [Esc] key code
                                   │ │ ┌─ 'File' menu 'Exit' function, 'Y'es
LP10 'DC''EPROM','AMD','27C010','FL''BINARY','%1','0','N','PP',/ESC /ESC 'FXY'
      │     │      │              │     │       │    │    └ Not fill buffer
      │     │      │              │     │       │    └ File start address
      │     │      │              │     │       └ User define file name
      │     │      │              │     └ 'Binary / Machine code' format
      │     │      │              └ 'File' menu 'Load' function
      │     │      └ [Enter] key code
      │     └ 'EPROM' category
      └ 'Device' menu 'Category' function
```

```
C:\LP10>PRO TEST.ROM
```

Program AMD Am27C010 EPROM with data in the file 'TEST.ROM'.

2. Contents of GAL.BAT is as follows:

```
LP10 'DC''ALL','%1','%2','FL''%3',,'PP',/ESC /ESC 'FXY'
      │    │   │    │     │    │     └ 'Process' menu 'Program' function
      │    │   │    │     │    └ file name
      │    │   │    │     └ File' menu 'Load' function
      │    │   │    └ Type number  symbol
      │    │   └ Manufacturer symbol
      │    └ [Enter] key code
      └ 'All Family' category
        'Device' menu 'Category' function
```

```
C:\LP10>GAL LATTICE 16V8 TEST.JED
                   |      |    |
                   %1     %2   %3
```

Program Lattice GAL16V8 GAL with data in the file 'TEST.JED'.

[NOTES] 1. Length of words in parameters included with 'LP10 ' is limited to 128.
2. Project instruction will not work in this function.
3. The example above is not included in the LEAPER-10 package, it is left to the user to construct it.

XI. Critical Error Code Information

Error Code 1:

```
    - ATTENTION -
Now the system is in demo status,
If you reconnect or turn on LEAPER-10
power, Please press the key [F5] to
initiate the system.
```

Error Code 2 :

```
    - WARNING -
LEAPER-10 not connect or fail,
Please check Power, Switch, Cable.
```

Error Code 3:

```
    - WARNING -
Hardwave error ! Please check LP-10.
```

Error Code 4:

```
Device not ready ,
or ID incorrect or bad !
```

Error Code 5:

```
This device couldn't support
this Function.
```

Error Code 6:

```
    - ATTENTION -
Not found mega byte cache file !
Please exit L10 system, then :
1. Modify CONFIG.SYS to allocate
   at least 1MB XMS or EMS memory.
2. Execute SETBUF.EXE, if no
   (<1MB) XMS or EMS supported.
3. Enter L10 system again.
```