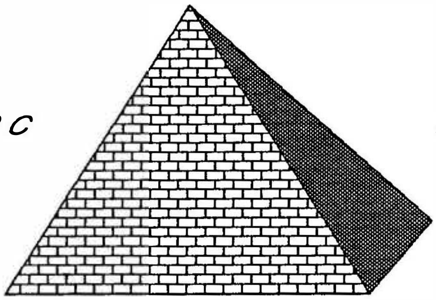
CPC



PCW

PYRADEV+

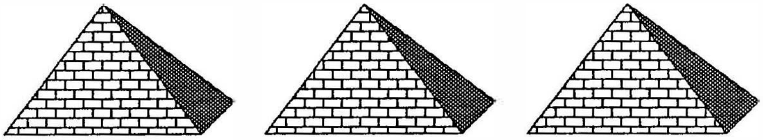
Everything you need

for your

ASSEMBLER programming

on all AMSTRADs with CP/M+

- *Program Editor*
- *Macro Assembler*
- *Monitor (CP/M)*
- *Monitor (AMSDOS)*
- *Disc Nurse*
- *Disc Clone*
- *Utilities*
- *Tutorial*



USER MANUAL

TABLE OF CONTENTS

SECTION 1: PYRADEV

1.1 INTRODUCTION

PYRADEV is a set of six programs which provide all the tools you need for the development of CP/M and AMSDOS software on the AMSTRAD PCW8256, PCW8512, PCW9512 and CPC6128 microcomputers. It also runs on the CPC464/664 machines if they are fitted with at least 128 kb of memory, and running under the CP/M+ operating system.

It will support large programming projects on single or double drive systems and consists of the following programs:-

- o A very fast full screen EDITOR.
- o A Z80 MACRO ASSEMBLER that produces COM, HEX, REL or BIN files.
- o A powerful symbolic MONITOR and DIS-ASSEMBLER.
- o A friendly and easy to use DISC-NURSE.
- o A very effective DISC-CLONE.
- o A comprehensive FILE MANAGEMENT utility.

The EDITOR enables large files (up to 48,000 bytes) to be quickly accessed and edited. Code can be searched, replaced, entered, deleted, moved, copied, merged (across files). In addition, files can be erased, renamed, deleted and so on. Space compression is used throughout to make the best use of the AMSTRAD memory and disc capacity.

The ASSEMBLER allows up to 32 files to be selected in a single assembly. They can be on different discs allowing virtually unlimited sized source code to be processed. The output can be a CP/M COM file, a relative file, a HEX file, or and AMSDOS binary file (for 6128 only). Video or Print output can be toggled on/off as required. Symbol and Cross-Reference maps may also be produced. The Symbol table can be stored on disc and later used by the monitor. The Assembler is very fast, processing up to 40,000 source code characters a minute.

The symbolic MONITOR will load, trap, single step, double step, modify and write code back to disc. Labels, produced by the assembler, are displayed with the dis-assembled code and makes debugging very easy without the need for a printed listing. Memory can be searched for Ascii or Hex string, modified and printed. Code dis-assembly can be output to a printer or ASCII files.

The DISC-NURSE handles all types of discs, even discs that changes format between the tracks. File sectors can be searched, viewed, printed, modified and re-written. The search can be extended to the whole disc if required. Sectors, or part of them, can be moved to memory and written to a disc file for later dis-assembly. Whenever a sector is read the file-owner is displayed. Previously deleted files can be re-claimed if sub-sequent sector allocation has not corrupted data. An extended Directory feature enables the disc directory to be printed with a header message, in a detailed format suitable for disc housekeeping lists.

The DISC-CLONE lets you make backup copies of all those discs with strange formats. You can copy between two different disc drives or use the same drive for source and destination.

The UTILITY (file-manager) program provides directory display, erase, rename, copy to/from discs and tape (6128) all in a single and easy to use program. Input file headers are always displayed, and protection can be inserted or removed in all copy operations. This is the easy way to manage ALL standard files.

All programs support all disc drives; A,B,C,D...M..P and User 0 - 15.

1.2 SETTING UP

PYRADEV is supplied on a MASTER DISC which must never be written to. Start by making a copy of your MASTER DISC using the CP/M utility DISCKIT. (DISCKIT3 on the CPC6128) Copy side A for PCW and side B if you are using a CPC-machine.

Store away the MASTER DISC and use the copy for the rest of the procedures. Open the Write Protect Holes on your copy, so that you cannot accidentally write to it.

The easiest way of using PYRADEV is to make a Turnkey disc, i.e. a disc that automatically starts PYRADEV when you turn your computer on (PCW) or start CP/M (CPC).

The following procedure is explained i detail in the Tutorial, 8.2.

To produce a Turnkey disc, use DISCKIT to copy the PYRADEV disc to a new disc. Use PIP to copy PIP.COM, SETKEYS.COM and *.EMS from your System discs to your new disc. (*.EMS will copy the CP/M file itself)

To complete your new Turnkey disc, type REN PROFILE.SUB=PROFILE.PCW and REN KEYS.DEV=KEYS.PCW if you are using a PCW or REN PROFILE.SUB=PROFILE.CPC and REN KEYS.DEV=KEYS.CPC for the CPC machines. PROFILE.SUB contains commands that will be executed automatically when the machine starts up. This file makes the disc a Turnkey disc. (See the Tutorial section for further details)

KEYS.DEV contains redefinition of keys to make it easier to use PYRADEV. You can later on change the way PYRADEV uses the keys, by changing the file KEYS.DEV.

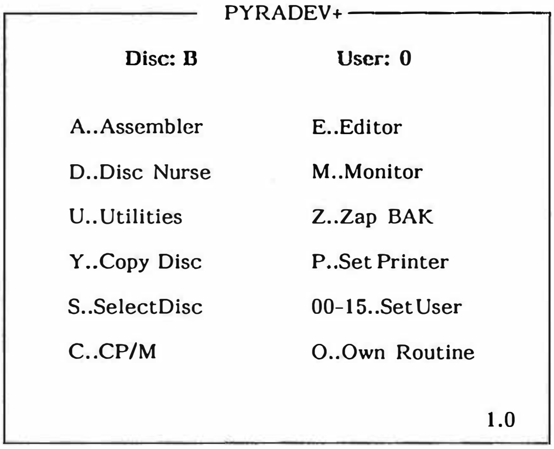
When you now start CP/M with the new Turnkey disc, it starts by redefining the keys. On the PCW, all PYRADEV files are then copied to drive M for faster access.

Finally, PYRADEV is started and the Main Menu is displayed.

The best way to get to know PYRADEV is to go trough the Tutorial in section 8. This will quickly teach you most of the features in PYRADEV.

1.3 MAIN MENU

PYRADEV can be started either by restarting the computer with your new Turnkey disc, or simply by typing PYRADEV if you already are in CP/M. The SYSTEM-MENU will now appear.



```
                    PYRADEV+

        Disc: B                 User: 0

     A..Assembler            E..Editor

     D..Disc Nurse           M..Monitor

     U..Utilities            Z..Zap BAK

     Y..Copy Disc            P..Set Printer

     S..SelectDisc           00-15..SetUser

     C..CP/M                 O..Own Routine

                                           1.0
```

Figure 1.0 System-Menu.

The system programs are selected by pressing the appropriate single key <A, E, D, M, O, U, Y>. They all return to the above menu on Exit.

The <Z> option will delete all backup files (*.BAK) on the current Default Disc. Use it with care.

The <S> option allows you to change the default disc drive setting for Data/Source files to (A), (B) ... (M). The default User number is set by typing in a two digit number, 00, 01 up to 15. This is used to divide your disc into different sections. It becomes more useful the larger the discs are.

The <O> option is used to execute your own program from the PYRADEV menu. The file PYRAOWN.ASM shows you how to make the program return to the PYRADEV menu once it is finished. The program must be named PYRAOWN.COM

<P> is used to change the left margin or the page length for print-outs. It is also used for the "Typewriter" mode, or to print a simple file.

Pressing P will display:

> Change Left Margin 00 (M) or Page Length 70 (L)
> "Typewriter"(T), Print File(F), Screen Colour (K)

Press **M** or **L** to change one of these values, **T** to send lines straight to your printer or P to print a simple file. On the PCW, **K** toggles between light or dark background, while on the CPC, **J** and **K** lets you select any paper and pen colour. Any other key will return to the System Menu.

Once set, these parameters are used by all PYRADEV programs.

PCW: If you are using contiguous paper, type "PAPER c F70" before you start PYRADEV to avoid that the printer software interferes with PYRADEV. This can also be achieved by printing the file "SETUP.PRT" as described in the TUTORIAL section.

The "Typewriter" mode can be used for sending control codes to the printer. It allows you to type one line at a time. As soon as you press <C/R> the line is sent to the printer. To make it possible to send "esc-sequences" to the printer without the final <C/R>, the program does not print the <C/R> if the line starts with a non alphanumeric character, such as <ESC> (<STOP> on PCW).

ALT-C exits the Type-Writer mode.

The "Print-File" mode is selected by pressing P from the Printer Menu. You can store different files with printer commands on the disc and then easily send them to the printer with this command.

A prompt: File-name: is displayed. Type the name of the file, followed by <C/R> and that file will be printed. See the "Printer parameters"paragraph in the "MAIN MENU" section of the TUTORIAL for a detailed description of how to set up a control file.

1.4 NEW DISCS

When a WORKING-DISC is created there may be system files on it which are not required. Press U to select the Utilities program, then press A or B to see the disc directory. The Delete command can be used to remove files. This list describes the system files.

PYRADEV.COM	SYSTEM-MENU
PYRAMED.COM	Full Screen Editor for Source Program Editing.
PYRAMAS.COM	Z80 Multi-File Macro Assembler.
PYRAMON.COM	Monitor (Start up and loader)
PYRAMON.PRL	Monitor (Main part)
PYRADSC.COM	Disc Nurse for Modifying Sectors.
PYRACOP.COM	Disc Clone for Copying Discs.
PYRAUTL.COM	File Utilities for General Copying.
DSPKEY	Self-Teach Source Program.
DSPKEY.COM	Self-Teach Load Module.
PROGRAM.001	Self-Teach Demonstration Source Program.
PROGRAM.002	Assembler Code Examples and Test File.
PROGRAM.003	Simple file copier program.

After deleting non-essential files, the new Working Disc is Ready for use. File deletion is also possible withln the Editor.

1.5 GENERAL FUNCTIONS

The STOP key (ESC on CPC) is used by all programs to abort a function that you have started.

ALT-X (CTRL-X on CPC) exits the current program and returns to the main menu.

When displaying a directory, pressing STOP (ESC) will stop the display and pressing any other key will pause the display until a key is pressed a second time.

SECTION 2: SOURCE FILE EDITOR

2.1 FILE SELECTION

The Editor is designed for very fast program development and editing. It is selected from the System Menu by pressing (E). Once loaded it will wait for Input and Output file names to be entered. The Editor always returns to the Idle Screen after the edit session is complete.

Pressing STOP (PCW) or ESC (CPC) will display the default directory.

```
                    *** AMSTRAD SOURCE EDITOR ***

   INPUT..File: ______________
   OUTPUT.File: ______________

 Drive A    User 00    17k free

 DATE     COM 02k   DIR      COM  15k   J14CPM3 EMS  40k
 KEYS     DEV 02k   PAPER    COM  02k   PIP     COM  09k
 PROFILE  SUB 01k   PYRACOP  COM  03k   PYRADEV.COM  01k
 PYRADSC  COM 15k   PYRAMAS  COM  21k   PYRAMED COM  14k
 PYRAMON  COM 02k   PYRAMON  PRL  16k   PYRAUTL.COM  07k
 SETKEYS.COM 02k

 ALT -X to EXIT  STOP for Dir   ALT-Z to ZAP BAKups   PYRAMED+ 1.0
```

Figure 2.0 Idle Screen.

- o To VIEW a file; Just enter an INPUT-FILE name.
- o To CREATE a file; Just enter an OUTPUT-FILE name.
- o To MODIFY a file: Enter INPUT and OUTPUT file names.
- o To DISPLAY directory; Press STOP (ESC)

The Input file name can be prefixed with A: or B: to set a new disc drive default.

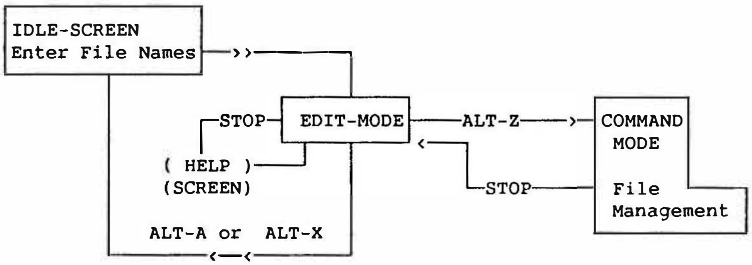
To set a new User number, prefix the file name with the new User number followed by a colon, or by a drive letter followed by a colon.

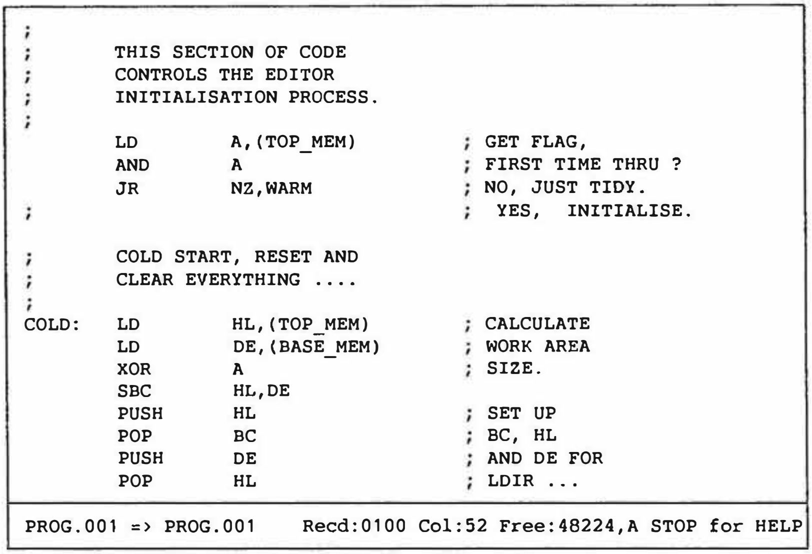
Example:	FILE.ASM	(edit FILE.ASM on the current drive and User number)
	B:FILE.ASM	(edit FILE.ASM on drive B, which will be the new default drive)
	2:FILE.ASM	(edit FILE.ASM on the current drive, User 2)
	2B:FILE.ASM	(edit FILE.ASM on drive B, User 2)

2.2 EDIT-MODE

After the file names have been entered, the INPUT file will be read and the first 24 records will be displayed. If CREATE mode is selected the screen will be blank.

Use of the **STOP** (ESC) key or **ALT-Z** key will alternate the Edit Session between Edit-Mode, the Help-Page and Command Mode.





```
;
;       THIS SECTION OF CODE
;       CONTROLS THE EDITOR
;       INITIALISATION PROCESS.
;
        LD      A,(TOP_MEM)         ; GET FLAG,
        AND     A                   ; FIRST TIME THRU ?
        JR      NZ,WARM             ; NO, JUST TIDY.
;                                   ;  YES,  INITIALISE.

;       COLD START, RESET AND
;       CLEAR EVERYTHING ....
;
COLD:   LD      HL,(TOP_MEM)        ; CALCULATE
        LD      DE,(BASE_MEM)       ; WORK AREA
        XOR     A                   ; SIZE.
        SBC     HL,DE
        PUSH    HL                  ; SET UP
        POP     BC                  ; BC, HL
        PUSH    DE                  ; AND DE FOR
        POP     HL                  ; LDIR ...

PROG.001 => PROG.001    Recd:0100 Col:52 Free:48224,A STOP for HELP
```

Line 25 of the Edit Screen always displays status information:- The current Input and (probable) Output file names, current record number and cursor column, Free-Bytes and default Disc Drive are shown.

2.3 STOP (ESC) KEY

When used in Edit-Mode,the STOP (ESC) key will cause the Help-Page screen to be displayed. This is just an 'aide memoire' and describes very briefly the functions below. Pressing the STOP (ESC) key a second time returns the screen to Edit-Mode.

2.4 ABORT & EXIT: (ALT-A)

The ALT-A key may be used to ABORT the Edit-Session. When used all files are abandoned and no disc changes take place. The character Y must be entered to confirm the Abort Request. The Editor returns to the Idle Screen.

2.5 SAVE & EXIT: (ALT-X)

The ALT-X key is the normal exit. The output file name will be displayed and can be overtyped as required. The file will be written to disc and control will return to the Idle Screen. If the File-Write fails, the message ** FILE SAVE FAILED ** will be shown and Command Mode will be selected so you 'action' the disc (ie delete non-required files) or use another disc. After a successful save the the Editor returns to the Idle Screen.

The Output file name can be prefixed with A: or B: to set a new disc drive default and/or with a new User number.

2.6 CURSOR-MOVEMENT

This is controlled by using the UP, DOWN, LEFT and RIGHT ARROW keys or by entering text. Forward and Reverse scrolling is automatic as row-1 or row-24 are reached. Text is either overkeyed at the cursor position or pushed in front of the new text (insert mode). Records are added to the front or back of the file if the cursor is taken past the first or last records records using the up and down arrow keys. The RETURN key always goes to the next line, column one. The TAB and SHIFT-TAB keys move the cursor to the next tab or previous tab stop respectively.

2.7 SCREEN-SCROLLING

The SHIFT-DOWN-ARROW and SHIFT-UP-ARROW will scroll the screen one line up or down respectively whilst maintaining the cursor position. The ALT-DOWN-ARROWand ALT-UP-ARROWwill scrollforwards and backwards 24 lines at a time. These four functions will not scroll past the beginning or end of the file.

2.8 CHARACTER-INSERT-DELETE

The SHIFT-RIGHT-ARROW and SHIFT-LEFT-ARROW provide these functions. Note that the insert function places a single blank in the text which can then be overtyped. The Insert and Delete actions both occur at the cursor position. The two DEL keys (CLR and DEL on CPC) also provide character delete and reverse delete capabilities.

2.9 LINE-INSERT-DELETE

The ALT-RIGHT-ARROW and ALT-LEFT-ARROW provide these functions, and they both operate on the current line, indicated by the cursor. The line insert action places a blank line into the text which can be overtyped. As text is typed of the end of the line another blank line is inserted. The same occurs if the RETURN key is pressed. Line Insert Mode is cancelled by the use of ALT-LEFT-ARROW (to delete a newly created blank line) or by scrolling down past row 24.

2.10 OPEN SPACE: (ALT-O)

The ALT-O key can be used to open or split a line at the cursor position into two new lines. The first will contain text up to the cursor position. The second line will contain text from the cursor position. This is useful when inserted new code at a line which already has a label.

2.11 BLOCK-COPY: (COPY)

To copy a block of code, move the cursor to the first line of the code and press ALT-B to set the BEGIN marker. Move the cursor to the last line of the code and press ALT-E to set the END marker. The message Block Saved will be displayed. Position the cursor where the block is to be copied to and press the COPY key. The block will be COPIED-INSERTED at the following line. The saved block will stay in memory until another Block is marked. It can be copied repeatedly anywhere in the file, or to another file or to another disc.

2.12 BLOCK-DELETE: (ALT-D)

Mark the Block with the ALT-B and ALT-E keys as for the Block Copy function. When the 'Block-Saved' message appears, press the ALT-D key. This will display the marked block in reverse INKS. Press Y to permit the delete operation. Note after the delete, the block is still saved and can be copied back if required with the COPY key.

2.13 BLOCK-MOVE: (ALT-COPY)

This is identical to the BLOCK-COPY function, but ALT-COPY should be pressed to move the block from the original position to the new destination. The original block will be deleted.

2.14 SETTING TABS and MARGINS: (ALT-TAB)

Pressing ALT-TAB will display the current tab settings as small "t"s on line 24. Tabs are set or cleared by positioning the cursor and pressing the TAB key until the "t" appears or dis-appears. Pressing ALT-TAB will clear all tab stops. When the new tab stops are ready, press the RETURN key to record them. The leftmost and rightmost tabs are used as the left and right margin. Tab marks can be saved for future use, see Command Mode.

2.15 MARK-FIND-LINE: (ALT-L, ALT-F)

A single LINE can be marked by using the ALT-L key. The line can be brought back to the screen with the ALT-F key (FIND). These two operations are useful in that they allow a rapid return to code being entered after studying code elsewhere in the file.

2.16 VIEW-TOP-OF-FILE: (ALT-T)

Press ALT-T to display the top of the file.

2.17 VIEW-END-OF-FILE: (ALT-V)

Press ALT-V to display the end of the file.

2.18 GO TO RECORD: (ALT-G)

Pressing ALT-G allows a record number to be entered. The Editor will go to the record immediately. Use of zero or numbers greater than the last record will cause the Beginning or End of File to be displayed respectively. The ALT-G function is helpful in locating program code from the Assembler Listing.

2.19 MERGE-FILE: (ALT-M)

Use of the ALT-M function allows another INPUT file to be appended to the current file. In this way it is possible to build up new programs or documents from previous code or text. Once the new file has been added into memory, the Block Delete and Move functions can quickly bring code or text to where it is required.

This function may only be used if at least 1028 bytes are still free.

2.20 SEARCH-SYSTEM: (ALT-S)

ALT-S selects the search system and two types of operation are provided.

Search and Stop

Enter a search string and press the RETURN key twice, leaving the replace string empty. The search will start. Each time the string is found, the record containing it will be displayed together with the preceding and following record in its own small 'window'. The following options can then be used:-

o	Return Key .	Ignore this match and continue searching.
o	STOP	Terminate, and return to original text.
o	G Key.	Goto the record where the match was found.
o	Up/Down. . . .	Scroll the records where the match was found.

Search and stop enables rapid positioning to program code using source code labels which are (usually) unique.

Search and Replace

Enter a search string followed by RETURN. Then enter a replace string. Be sure to use upper or lower case as required for the replace string. After the replace string is entered, answer the question to select Automatic or Conditional replacement and the search will start.

If the Auto-Replace option is chosen, the process will run to completion without further action. If the Conditional option is used, each time the string is found it will be shown with the preceding and following records. Press R to do the REPLACE, or press C to CONTINUE with no action.

All search functions operate from the current line to end-of-file. Use **ALT-T** to select record 1 if you intend to search the whole file. A good keystroke memory sequence to use prior to using the search function is **ALT-L ALT-T.** This marks the current line and goes to the beginning of the file. After using the search function **ALT-F** will return to the marked line.

Any search can be cancelled with the STOP (ESC) key.

Note: Truncation will occur if a replace string is longer than the source string and characters are forced past column 80.

2.21 KEYSTROKE-MEMORIES

The editor contains nine keystroke memories which can each record 32 keystrokes. Each of them may be used to record text or control keys. These can then be 'replayed' as required.

Display

To view the contents of the keystroke memories press the numeric island key 0 (f0). The memories will be displayed. Press the STOP (ESC) key to return to Edit-Mode.

Record

Press the ENTER key then select a memory using the f1 through f9. (f9 is SHIFT-0 (RELAY) on the PCW). Once selected the message **Keystroke** Recording ON will be displayed. Type the text and/or control keys that you want to save. To terminate the recording press the small enter key again. If a recording exceeds the 32 keys limit, the next memory will be attached and used. If the end of keystroke-memory nine is reached, recording will be turned off, and the KS-sequence-saved message will be displayed.

Replay

To replay any recorded sequence press the f1 through f9 which was used to make the recording. The keys will be taken and used as though they had just been entered at the keyboard. The replay can be stopped by pressing the STOP (ESC) key.

Erase

Press ALT-ENTER and select a memory with f1 through f9. The memory will be cleared ready for use once again.

Note.1: Memory functions may only be recorded or initiated for playback in normal Edit-Mode. Recorded sequences may for example select the Search system and start a search for a character string then revert to Edit-Mode. However it is not possible to start a recording or keystroke memory playback if the Search system has already been selected manually.

Note.2: The keystroke memories can be saved on disc. See Command Mode.

2.22 PRINTING

Pressing the decimal point key on the numeric island (EXTRA-point on PCW) will cause printing to start from the current line. Pressing the key twice will stop line-feeds being sent with carriage-returns. Printing will continue until End Of File. Pressing the STOP (ESC) key once pauses the print which can be continued with any key. Pressing the STOP (ESC) key twice cancels the print activity.

Special print codes may be embedded in the text being printed in order to control the printer as foilows...

During printing, the up-arrow symbol (CPC: on the pound sign key) (PCW: EXTRA-U) indicates that the following character is a control symbol. The value 33 is subtracted from the ASCII value of the character. If it is minus, the character is ignored, else it is sent to the printer. An up-arrow symbol at the end of a line causes the invisible CR character to be skipped, allowing two screen lines to make one long line of printing.

Examples

```
↑<E represents ESCAPE-E,  ↑<S represents ESCAPE-S,
↑0  represents code 15,   ↑/  represents code 14.
```

See section 8.3 for a complete table of print codes.

2.23 COMMAND MODE: (ALT-Z)

Command mode is selected with the **ALT-Z** key whilst in Edit Mode. The Edit Session is temporarily suspended while the File Management and Option routines become available. Exit from Command Mode with the **STOP** (ESC) Key.

A small menu is displayed at the bottom of the screen and Functions can be selected by pressing the appropriate key as follows:-

```
A,B,M: Select Disk        C: Compression
D,R: Delete, Rename       Z: Zap BAKups
L,S: Load,Save Options    K: Colour
```

A,B,M: Select Disc A, B or **M**

Pressing (A), (B) or (M) changes the default disc drive and displays the new disc directory, file-names and free space.

C: Compression ON/OFF

If you are Editing and Assembling code within the PYRADEV system, you should leave this option ON. The Editor and Assembler will use data compression to keep your files as small as possible. If you use the Editor to edit other files, before using ALT-X to write the file back to disc you should turn compression OFF. The file will be expanded to normal.

D: Delete File

This is similar to the CP/M directive 'ERA'. It can be used to delete a single file or group of files. A specific disc drive can be selected by using an A: or B: prefix.

Examples:	TEMPFILE.001	-	delete the file.
	TEMP.*	-	delete all TEMP files.
	TEMP????.*	-	delete file names starting with TEMP.
	B:TEMP.FIL	-	delete TEMP.FIL on drive B.

R: Rename File

Any existing OLD file can be renamed to a NEW file if the new file does not already exist.

K: Colour

Pressing K, reverses the background and foreground colours.

S: Save User Options

Pressing S causes the current Tabs and Keystroke Memories to be written to a disc file for future use. A file-name must be entered and can be any name. Suggested file extension (suffix) is -OPT. Several different Option Files can be used to define different 'environments'.

L: Load User Options

Use of the L key allows an 'option file' created with the (S)ave command to be re-instated for the current Edit Session. After the file is read, the new Tabs and Keystroke Memories will be in operation.

Z: Zap *.BAK files

Removes all backup files from the default disc directory. It is the same as using the (D)elete function with the file name *.BAK.

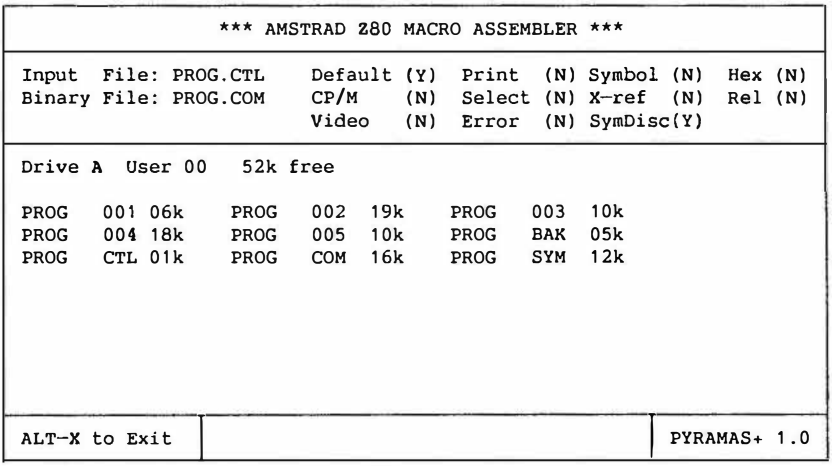
SECTION 3: MACRO ASSEMBLER

The PYRADEV Macro Assembler supports the ZILOG Z80 programming language (with a few alterations and extensions) and will process one or more source program files to create an output file, directly on disc. This output file can be a COM file, a HEX file, a relocatable REL file used by CP/M or a BIN file to be run under AMSDOS.

It is selected from the SYSTEM MENU by pressing **A.** Once loaded it waits as the various RUN OPTIONS are entered...

3.1 START RUN OPTIONS

Press RETURN without any Input File name to display the current directory.



```
                   *** AMSTRAD Z80 MACRO ASSEMBLER ***

 Input  File: PROG.CTL    Default (Y)  Print  (N) Symbol (N)   Hex (N)
 Binary File: PROG.COM    CP/M    (N)  Select (N) X-ref  (N)   Rel (N)
                          Video   (N)  Error  (N) SymDisc(Y)

 Drive A  User 00   52k free

 PROG     001 06k    PROG     002   19k    PROG     003   10k
 PROG     004 18k    PROG     005   10k    PROG     BAK   05k
 PROG     CTL 01k    PROG     COM   16k    PROG     SYM   12k

 ALT-X to Exit                                          PYRAMAS+ 1.0
```

Figure 3.0   Assembler Run Options

INPUT-FILE

This names the first source file for the input stream. It may be a complete program or the first of many which are to be assembled together to make up the binary load program. SELECT statements embedded in the source code files can be used to append other files to the input stream. One way of using the SELECT system is for the first file to contain nothing but SELECT statements. A good file suffix for these files is .CTL as it represents a CONTROL file. A default binary file name with a .COM suffix is created from the input file name. An END statement in the input stream is optional, but if present will end the input stream immediately, even if there are more files.

BINARY-FILE

This is the name which is to be used when the binary code file is stored on disc. It is built from the input file name (by adding a it is. If no binary file is required, this field should be set to blanks.

Pressing TAB positions the cursor on the "C" in .COM . This is useful if you are creating non-COM files and want to use another suffix on the output file, i.e. REL , HEX or BIN .

DEFAULTS

If the Return key is pressed or (Y) is pressed the Assembler will use the displayed defaults and start the assembly process. If any of the run options are to be altered, press (N) for this first option.

CP/M-OUTPUT

If set to (Y), the binary file is written to disc without a file header, suitable for use as a CP/M transient program. No other checks are made to ensure that the program conforms to CP/M standards.

If set to (N), an AMSDOS binary file will be created. These file must be run under AMSDOS, not CP/M. (Relevant on CPC only).

VIDEO-DISPLAY

If set to (Y), the source record being processed will be displayed in pass 1. During pass 2, the object code and source record will be displayed.

PRINT-OUTPUT

Setting the PRINT option to (Y)es instructs the Assembler to produce an assembly listing during pass 2 object code generation. This option works together with the SELECT option ...

SELECTIVE PRINT

The SELECT option can be set to (Y)es or (N)o and instructs the Assembler to honour (YES) or ignore (NO) the LIST and NOLIST directives contained in the source code.

LIST ERRORS

When set to (Y)es, this option forces ERROR messages to be listed on the printer for subsequent use when correcting coding mistakes with the EDITOR. It works independently of the PRINT and SELECT options described above, however errors are always listed if the PRINT option is (Y)es.

PRINT OPTIONS SUMMARY

PRINT	SELECTIVE	ERRORS	Printed Code	Printed Errors
NO	NO	NO	NONE.	NONE.
NO	NO	YES	NONE.	ALL.
NO	YES	NO	NONE.	NONE.
NO	YES	YES	NONE.	ALL.
YES	NO	NO	ALL.	ALL.
YES	NO	YES	ALL.	ALL.
YES	YES	NO	SELECTIVE.	ALL.
YES	YES	YES	SELECTIVE.	ALL.

WARNING: The use of the VIDEO or PRINT options will SIGNIFICANTLY slow the Assembly Process down. For fastest results reply (Y) to defaults. An exception is to list ERRORS which you will find very useful during the first few Assemblies of large programs.

SYMBOL-TABLE

If set to (Y), a sorted list of the symbols (labels) used in the source program(s) will be output with their values. Output will be to VIDEO or PRINTER depending on the options chosen above. Each symbol may be preceded by a (U), (D) or (N) denoting Undefined, Doubly-Defined or Not-Referenced.

CROSS-REFERENCE

If set to (Y) a sorted cross reference map will be output to the Video Display or Printer after the assembly process is complete. The listing will display all source files read, numbering them as files 1,2,3 and so on. The symbol table is printed, together with all file numbers and line numbers of statements that reference them.

Please note that if they are used, the SYMBOL or X-REF options will wait for the VIDEO or PRINT options to be set to ON before the assembly process can output the information requested.

SYMBOL-TABLE TO DISC

If set to (Y), a sorted list of the symbols (labels) used in the source program(s) will be output to a disc file. This file is named yyyyy.SYM where yyyyy is the selected Binary file name.

The file can later be used by the Monitor to simplify debugging.

HEX-OUTPUT

If set to (Y), the output file is written to disc as a .HEX file.

REL-OUTPUT

If set to (Y), the output file is written to disc as a relocatable libraries of relocatable object modules. LINK can then be used to produce a .COM file from the REL files and the libraries.

3.2 ASSEMBLER RUN TIME KEYS

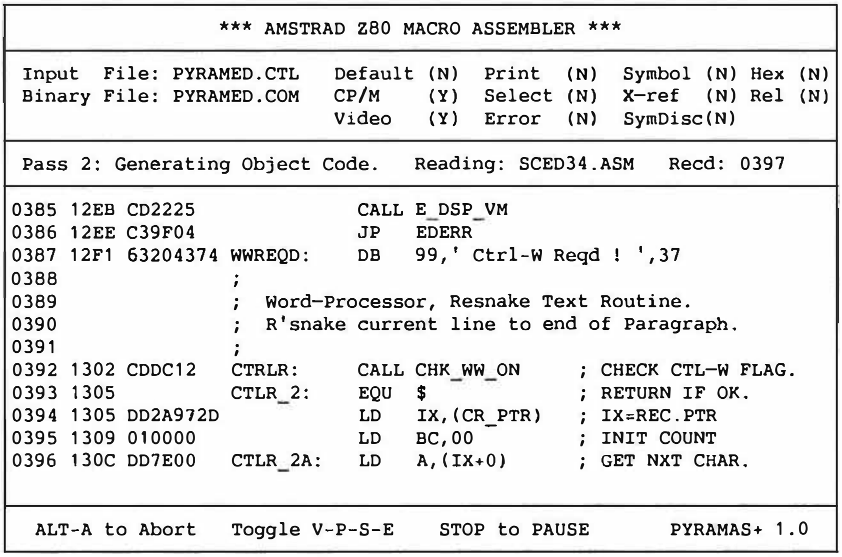
Once started, the Assembly process can be controlled by 'toggling' the RUN OPTIONs described above. This is done by pressing single keys. The screen run options display will change as the keys are used.

The following keys may be used:-

(P) Reverse the PRINT option.
(S) Reverse the SELECT option.
(E) Reverse the ERRORS option.
(V) Reverse the VIDEO option.

In addition the STOP (ESC) Key may be used to PAUSE the Assembly. This is useful in order to study object code belng displayed on the screen. Press any key to resume the assembly process.

Pressing the ALT-A key will abort the assembly process. All files will be abandoned and no disc changes will take place. The Assembler will ask for any key to be pressed so that it can return control to the System Menu....



```
                    *** AMSTRAD Z80 MACRO ASSEMBLER ***

 Input  File: PYRAMED.CTL   Default (N)  Print  (N)  Symbol (N) Hex (N)
 Binary File: PYRAMED.COM   CP/M    (Y)  Select (N)  X-ref  (N) Rel (N)
                            Video   (Y)  Error  (N)  SymDisc(N)

 Pass 2: Generating Object Code.   Reading: SCED34.ASM   Recd: 0397

0385 12EB CD2225                 CALL E_DSP_VM
0386 12EE C39F04                 JP   EDERR
0387 12F1 63204374 WWREQD:       DB   99,' Ctrl-W Reqd ! ',37
0388               ;
0389               ;  Word-Processor, Resnake Text Routine.
0390               ;  R'snake current line to end of Paragraph.
0391               ;
0392 1302 CDDC12   CTRLR:        CALL CHK_WW_ON     ; CHECK CTL-W FLAG.
0393 1305          CTLR_2:       EQU  $             ; RETURN IF OK.
0394 1305 DD2A972D               LD   IX,(CR_PTR)   ; IX=REC.PTR
0395 1309 010000                 LD   BC,00         ; INIT COUNT
0396 130C DD7E00   CTLR_2A:      LD   A,(IX+0)      ; GET NXT CHAR.

  ALT-A to Abort   Toggle V-P-S-E   STOP to PAUSE      PYRAMAS+ 1.0
```

Figure 3.1 Run Time (VIDEO-ON)

3.3 ERROR HANDLING

The detection of errors by the assembly process will be handled in one of three ways according to the output options set above.

a) If any of the PRINT options or ERROR options are set to (Y)es, the Assembler will assume a printer device is attached. The error code and description will be printed. The assembly process will NOT pause.

b) If the VIDEO option is set to (Y)es, the error code and the description will be displayed. The assembly process will pause, and can be resumed by pressing any key.

c) If the PRINT, ERROR and VIDEO options are all set to (N)o, the assembly process will be running at it's fastest setting. Errors will be sent to the Video Screen, but the assembly process will only pause after ten error messages have been displayed. If this occurs, press any key to continue the assembly process.

Error Codes

B - Branch/Jump Error
Too long relative jump. Replace JR with JP (Long jump)

C - Condition error
ELSE is found without a matching IF

D - Double defined label
The same label appears twice.

F - Error when closing file
The disc is probably full

FW - Illegal Forward ref
The value in a DS instruction must be defined

G - Org value error
The value that follows an ORG statement must be defined

L - Label error

ML - Loop in MACRO call
A Macro is calling itself

MP - Error in Macro Parameter
An error is detected in a Macro definition.

MS - Macro stack overflow
Macro calls are too deeply nested, i.e. one macro calls another which calls another ...

N - Numeric error
An error is detected when evaluating an expression

OC - Invalid opcode

OE - Operand error

OG - ORG missing
Every program must contain at least one ORG statement.
ORG 100H (For a COM program)

OM - Operand Missing

R - Allowed in REL-file only
The directives ASEG, DSEG, CSEG, NAME, EXTRN and PUBLIC are only allowed in a REL file

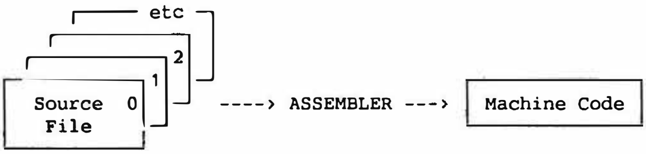
SF - Too many SELECT Files
A maximum of 32 files can be included in an assembly

U - Undefined Label
A label (symbol) is referenced but never defined

X - EXTRA Z80 not allowed
The "undocumented" instructions are only allowed if an EXTRA instruction has been entered

3.4 LANGUAGE DEFINITION

The Assembler Language is the Z80 language and the programs are written as one or more Source Files which are assembled to machine code by the Assembler.



Each source file consists of source statements, one per line (screen row). The PYRADEV EDITOR is the ideal program to create and edit these source files. Each source file statement consists of an optional label, an opcode, optional operands and optional comments.

```
LABEL          OPCODE    OPERANDS        COMMENTS
--------       ------    ---------       --------
LABEL_1:       LD        HL,(VALUE)      ; load HL
               LD        (OLD_VALUE),HL  ; save HL
               JP        LABEL_X         ; jump ...
;
VALUE:         DEFW      00              ; current value
OLD_VALUE:     DEFW      00              ; old value
```

LABELS: These must start in column one and may be any length although it is usual to keep them down to less than 10 characters. The use of a colon after the label is optional.

OPCODES: These may be anywhere on a line and must be preceded by at least one space.

OPERANDS: An opcode must be followed by at least one space before the operand can be entered. The operand field must not contain embedded blanks.

COMMENTS: Operands must be followed by at least one space before a comment can be entered. The use of a semi-colon before the comment is optional except on RET statements. A semi-colon must be used if the comment starts at column one.

3.5 EXPRESSIONS

An expression is an OPERAND which consists of one or more variables, labels and constants which the Assembler must evaluate into a 16 bit integer value. An expression is evaluated from LEFT to RIGHT and parenthesis may not be used. The following operators may be used..

+ - * / .MOD. .SHR. .SHL. .AND. .OR. .XOR. .EQ. .GT. .LT. .UGT. .ULT.

They represent plus, minus, multiply, modulo, divide, shift-right, shift-left, AND, OR, exclusive-OR, Equates, greater-than, less-than, unsigned-greater, unsigned-less-than.

The dollar ($) symbol may be used to represent the value of the program counter during assembly. For example JP $+3 would generate a branch to the next instruction (a JP is 3-bytes).

The Assembler will accept numeric notation for binary, octal, decimal and hex-decimal expressions in the following formats..

```
Binary:  1011100B    or   %1011100
Octal:   134Q        or   @134         (or  134O)
Decimal: 134D        or   134
Hex:     5CH         or   #5C          (must start numeric)
                                       (ie 0FFH for 255D  )
```

3.6 ASSEMBLER DIRECTIVES

These are written like instructions (opcodes and operands) but are commands to the Assembler and are 'executed' at assembly time. They control assembly listing options, code generation, and the construction of the binary (machine code) file.

DEFINING VARIABLES and STORAGE

Bytes, Words and Character Strings may be defined using the directives shown in the following example.

```
          DBXON                             ; DB Expansion
LABEL1:   DB        "This is a string"      ; These are 16 byte strings.
LABEL2:   DEFB      "This is a string"      ;
BYTE:     DB        "B"                     ; Single byte
MIXTURE:  DB        1,"A",2,"b"             ; Mixed string
SSTRING:  DB        "SPECIAL"+80H           ; L has bit 7 on.
;
LABEL3:   DW        256*2                   ; A word can hold a
LABEL4:   DEFW      512                     ; 16 bit integer.
;
LABEL5:   DB        255                     ; Max value single byte
LABEL6:   DEFB      0FFH                    ; is 255 decimal.
;
VALUE7:   EQU       1000                    ; VALUE7 equals 1000
LABEL8:   DW        VALUE7                  ; LABEL8 contains 1000
VALUE7:   DEFL      1001                    ; VALUE7 redefined to 1001
LABEL9:   DEFW      VALUE7                  ; LABEL9 contains 1001
          DBXOFF
```

The DBXON and DBXOFF cause the DB strings to be listed in long form or short form (1st four bytes) respectively.

The Source Files PROGRAM.001, PROGRAM.002 and PROGRAM.003 on the MASTER-DISC contain more examples of the assembler directives. The files can be viewed and/or printed using the PYRADEV Editor.

EJECT

This instructs the assembly print process to feed to next top of form. Normally a page width of 80 characters and form depth of 70 lines are assumed. This is standard A4 size. These parameters may be altered via the PRINTR directive, shown below, or from the PYRADEV main menu.

END

This statement signals the END OF INPUT. The Assembler treats this as a hard end of file, even if source code follows the statement. Use of an END statement is optional.

ENDBIN

Ends binary code generation. Generally used at the end of a program before the DEFS or DS statements to keep a file size small, but can be used anywhere. It's opposite is the ORG statement which resumes code generation or the LOAD statement which 'pads' and resumes code generation.

```
;
; This coding example shows how an
; area is reserved at the front of
; of a program.
;
             ORG        100H
             ENDBIN
;
BUFF_1:      DEFS       2048
BUFF_2:      DEFS       2048
;
TRUE_START:  ORG        $
             LD         (SAVE),SP
             CALL       INIT_PGM
             JP         GO_GAME
;
             ETC........
```

```
;
; This coding example shows
; how endbin is used at the
; end of a program.
;
        POP  BC
        POP  DE
        RET
;
        ENDBIN
;
BUFF1:  DS 2048
BUFF2:  DS 2048
;
;
        END
```

EQU

This is the EQUATES directive. It equates a LABEL to an expression, for instance, MINUS1 EQU -1

EXEC

(Only relevant on BIN files)

This directive defines the address to be used in the binary file header. When the assembled program is RUN, it will be loaded according to the LOAD (or ORG) statement. AMSDOS will then pass control to the EXEC address. If this directive is not supplied, the EXEC address is set to the first (true) ORG or LOAD address.

EXTRA

Enables the Assembler to process the additional NON-STANDARD Zilog Instructions. See EXTRA INSTRUCTIONS. 3.9

FREE

Enables the Assembler to process 'Free Format' expressions. Allows mixing of different types of storage expressions. See FREE FORMAT.

LIST and NOLIST

Turn Selective Printing On and Off.

LOAD

This statement tells the Assembler to generate binary zeros code until the Program Counter reaches the LOAD value expression. Normal code generation is then resumed. It's purpose is to force sections of code to their proper positions in the binary file so that when the file is loaded, the code is at it's correct address.

If it is not supplied, the LOAD address defaults to the first true ORG address. One of the two must be supplied (it is usually ORG) before any code generation can occur. For BIN-files, the first occurrence of an ORG or LOAD directive is used in the binary file header (unless it is immediately followed by an ENDBIN statement). Subsequent ones are only used to control code generation and binary file structure.

ORG

This statement tells the Assembler to SET the Program Counter to the operand expression. Unlike the Load statement no filler code is generated. Use of the ORG statement allows sections of code which will be widely apart in memory to be squeezed together in the disc file being generated. It is the programmers responsibility to ensure that such sections of code are moved to their correct locations before being executed. It is NORMAL practice for the first statement in a set of programs being assembled to be an ORG directive.

```
     ORG  100H       . This Section locates
     code            . correctly in memory
     code            . from location 100 hex
     LOAD 200H       . upwards.
     code            .
     code            .
     code            .
;
     ORG  $+100         . This block will need
     code               . to be moved to its true
     code               . address as no filler code
;                       . is generated.
     END
```

PAUSE

This statement causes the assembly process to 'Pause and Display' the message. This occurs when the END of the current input file is reached. It's use is to allow multiple discs to be used when assembling multiple source files. It should only be used on TWIN drive systems as the BINARY file must be written continuously to one disc. After changing the disc, press the space bar to continue.

```
  PAUSE   'MOUNT NEXT SOURCE DISC IN DRIVE B'
```

PRINTR

This statement can be used to define Paper-Width, Lines per Page (form-depth), Page-Pause and Line-Feed suppress. The Page-Pause option causes printing to pause at each top of form to allow paper adjustment. This is a REAL requirement on some friction feed printers which 'slew' the paper and make Assembler Listings difficult to produce. Line-Feed suppress stops line feeds being sent with each carriage return as some printers do this automatically.

```
PRINTR  W80,D70      Width 80 chars, Depth 70 lines,  P-Pause  Off,
                     Line-feeds.
PRINTR  W132,D66,P   Width 132 chars, Depth 66  lines,  Page-Pause  On,
                     Line-feeds.
PRINTR N             Use the defaults (W80,D70) and suppress line-feeds.
PRINTR P             Use the defaults (W80,D70) and use Page-Pause.
```

SELECT

This is a very powerful directive. When used, the named Source File is ADDED (not INCLUDED) to the END of the current input stream. Upto 32 files can be CHAINED in this manner, and the SELECT statementsmay appear anywhere. When assembling large programs, it is possible to start the assembly process with a small control file which does nothing more than SELECT files for the INPUT stream...

```
ORG       100H          ; Program Origin
SELECT    PROG1.ASM     ;
SELECT    PROG2.ASM     ; Drive A Files
SELECT    PROG3.ASM     ;
SELECT    B:PROG1.ASM   ;;
SELECT    B:PROG1.ASM   ;;Drive B Files
SELECT    B:PROG1.ASM   ;;
```

TITLE

Changes the Top of Form Assembly Listing Header Message, example:-

```
            TITLE       MegaGame, Section 6.
```

DIRECTIVES FOR REL FILES

Relative files, type .REL, may contain the following directives:

ASEG - Absolute segment
Code will be placed in the absolute segment until the next CSEG or DSEG.

CSEG - Code segment
Code will be placed in the code segment until the next ASEG or DSEG.

DSEG - Data segment
Code will be placed in the data segment until the next ASEG or CSEG.

EXTRN - Declare foreign label
Labels defined in other REL files, must be declared in an EXTRN statement before they are referenced.

NAME - Names the REL file
This name is used for libraries. If not supplied, the file name will be used.

PUBLIC - Define a public label
The label, which must exist in the module, may be referenced in other REL files where it is declared with an EXTRN statement.

3.7 CONDITIONAL ASSEMBLY DIRECTIVES

The Assembler is able to include or exclude certain blocks of code during the assembly process through the use of flags and conditional directives. Combined with the SELECT system described above, the system becomes very flexible, as the first files may define flags which control the assembly of code in subsequent files.

The mechanism of conditional assembly is the classic IF *something* THEN *do this* ELSE *do that*

The *something* is an Arithmetic Expression. If the expression is TRUE (non-zero) the first path (THEN) is taken. Otherwise the second path (ELSE) is taken. The second path is optional and the final directive must be an ENDIF statement.

The following examples shows how the process can be used. The code on the left has the FLAG set to 1 (TRUE) so the THEN_CODE is assembled. The code on the right has the FLAG set to 0 (FALSE) so the ELSE_CODE is assembled.

```
;
FLAG:       EQU     1
;
            IF      FLAG
THEN_CODE:  LD      A,(VALUE_1)
            ELSE
ELSE_CODE:  LD      A,(VALUE_2)
            ENDIF
;
            END
```

```
;
FLAG:       EQU     0
;
            IF      FLAG
THEN_CODE:  LD      A,(VALUE_1)
            ELSE
ELSE_CODE:  LD      A,(VALUE_2)
            ENDIF
;
            END
```

Note the usage is always IF ... ELSE ... ENDIF. The THEN statement is implicit as the first branch after the IF statement. The ELSE section is optional. An IF directive must always have a corresponding ENDIF statement.

An alternate to using the IF..ELSE..ENDIF directives are the COND..ELSE..ENDC directives. They are both valid.

3.8 MACRO DEFINITIONS and USAGE

A MACRO is a short piece of code, defined in a file at the beginning of the assembly process. When its name is used, the previously defined piece of code is generated again.

A MACRO statement defines the start of the definition and it must have a label which is used as the Macro-Name. The name must be ALL alphabetic and may be up to SIX characters long. Imagine the Macro as a new Instruction for the Assembler. The two examples here show a macro without Parameter Usage on the left, and with Parameter Usage on the right.

```
SBCX:   MACRO                         SWAP:   MACRO #P1,#P2
        XOR     A                             PUSH  AF
        SBC     HL,DE                         LD    A,#P2        (LD A,C)
        ENDM                                  LD    #P2,#P1      (LD C,D)
;                                             LD    #P1,A        (LD D,A)
        LD      HL,(VALUE1)                   POP   AF
        LD      DE,(VALUE2)                   ENDM
        SBCX                          ;
        LD      (VALUE3),HL                   SWAP  D,C
;
```

The example on the left shows how a small macro can be used as an additional instruction. In this case the SBCX macro is assembled as two instructions. The first clears the carry flag before executing the second SUBTRACT with CARRY instruction.

The example on the right shows parameter substitution. During Assembly usage of the SWAP macro causes a five byte routine to be generated which will cause the contents of the C and D registers to be exchanged. The SWAP macro can be used to exchange any two registers except for register A.

A macro may include another already defined macro. (Recursion)

Macro Parameter Usage

As shown in the above example, parameter usage is positional and works by substitution. If parameters contain commas or quotes, they must be enclosed within single or double quotes as shown here:-

```
STRING:   MACRO    #P1,#P2
          DB       #P1            (DB 5)
          DB       #P2            (DB 21,22,23,24,'X')
          ENDM
;
          STRING   5,"21,22,23,24,'X'"
;
```

Macro Symbol Generator

If a macro definition contains labels, DUPLICATE LABEL errors will occur during the assembly process if the Macro is used more than once. In such cases the #SYM suffix must be added to the label. Each time the Macro is used, a 4-digit suffix is incremented and attached to the label.

The following example is a macro which tests HL and substitutes the hex-decimal constant 0FFFFH if HL is zero. Each time it is used, the JR TEST_#SYM instruction and TEST_#SYM labels are expanded with the next value. The first time thru the JR will be to the TEST_0001 label, then it will be TEST_0002 and so on ...

```
;
;            MACRO tests HL, if ZERO replace with 0FFFFH.
;
TEST:        MACRO
             XOR       A               ; Clear Carry.
             LD        DE,0            ; Set DE to zero.
             SBC       HL,DE           ; Subtract / TEST
             JR        NZ,TEST_#SYM    ; if NZ JMP to TEST_000n
             LD        HL,0FFFFH       ; else set HL to 0FFFFH
TEST_#SYM:   EQU       $               ; continue ... TEST_000n
;
```

Macro Listings

Normally only Macro Definitions are listed. To see the expanded code you must use the MLIST directive. To turn this facility OFF use the MNLIST directive.

3.9 EXTRA INSTRUCTIONS

There are a number of Z80 instructions which are not normally shown in Z80 programming manuals because they do not always work! If you are writing software for other Z80 users, DO NOT USE THEM. If you must use them, the PYRADEV Assembler will accept them, but the Directive 'EXTRA' must be given first.

The first group of op-codes allow the general purpose 16 bit IX and IY registers to be used as four 8 bit registers by classifying them as LOW or HIGH order registers. We use operands LX, HX, LY, and HY to represent the Low and High bytes of IX and IY respectively. The alternate form XH, XL, YH and YL may also be used.

```
LD  HX,A   LD  HX,B   LD  HX,C   LD  HX,D   LD  HX,E   LD  HX,n
LD  LX,A   LD  LX,B   LD  LX,C   LD  LX,D   LD  LX,E   LD  LX,n

LD  HY,A   LD  HY,B   LD  HY,C   LD  HY,D   LD  HY,E   LD  HY,n
LD  LY,A   LD  LY,B   LD  LY,C   LD  LY,D   LD  LY,E   LD  LY,n

LD  A,HX   LD  B,HX   LD  C,HX   LD  D,HX   LD  E,HX
LD  A,LX   LD  B,LX   LD  C,LX   LD  D,LX   LD  E,LX

LD  A,HY   LD  B,HY   LD  C,HY   LD  D,HY   LD  E,HY
LD  A,LY   LD  B,LY   LD  C,LY   LD  D,LY   LD  E,LY

LD  HX,LX  LD  LX,HX  LD  HY,LY  LD  LY,HY

INC HX     INC LX     INC HY     INC LY
DEC HX     DEC LX     DEC HY     DEC LY

ADD A,HX   ADD A,LX   ADD A,HY   ADD A,LY
ADC A,HX   ADC A,LX   ADC A,HY   ADC A,LY
SBC A,HX   SBC A,LX   SBC A,HY   SBC A,LY
SUB HX     SUB LX     SUB HY     SUB LY

CP  HX     CP  LX     CP  HY     CP  LY
AND HX     AND LX     AND HY     AND LY
OR  HX     OR  LX     OR  HY     OR  LY
XOR HX     XOR LX     XOR HY     XOR LY
```

A second group of codes provide additional SHIFT-LEFT-LOGICAL opcodes complementing the existing SRL instruction. They are:-

```
SLL A  SLL B  SLL C  SLL D  SLL E  SLL H  SLL L  SLL (HL)
```

The operations is the same as the SLA instruction, however a (1) bit is placed into bit position 0 instead of a (0).

Please note: The above op-codes are not standard (since there is no formal definition for them) however they do correspond with mnemonics used by a number of publications concerning Z80 programming. Where possible we have used common definitions.

3.10 FREE FORMAT

The use of the FREE directive enables the Assembler to process a type of Z80 expression useful to games writers. It is a mixed data and value expression system suitable for defining tables. Byte generation is always assumed unless a single exclamation symbol (!) precedes an expression in which case a word-value is generated. The use of DBXON is ideal when you first use free format to check that tables are being set up correctly. The following examples show how FREE format can be used ...

```
;
           FREE                   ; FREE FORMAT.
           DBXON                  ; DB EXPANSION.
;
FLAG1:     EQU       0
FLAG2:     EQU       1
VALUE1:    EQU       2
;
ROUTINE1:  CALL      GET_CURSOR
           INC       H
           CALL      SET_CURSOR
           RET
;
;          CONTROL   TABLE
;
   1,2,3,!100,!200,!300,FLAG0,FLAG1,"STRINGA","ABC1","111",!ROUTINE1
   4,5,6,!200,!400,!600,FLAG1,FLAG2,"STRINGB","ABC2","222",!ROUTINE2
   7,8,9,!300,!600,!900,FLAG2,FLAG2,"STRINGC","ABC3","333",!ROUTINE3
;
;          END OF CONTROL TABLE
;
           DBXOFF                 ; DB EXPANSION OFF
;
```

Each of the three lines in the controi table above will be assembled in a similar manner. Here we describe just the FIRST line and what binary code is generated ...

Three bytes containing the values 1, 2, and 3.

Three words containing the values 100, 200 and 300. Notice the !.

A byte containing the flag-0 value.

A byte containing the flag-1 value.

A string of bytes containing 'STRINGA'.

A string of bytes containing 'ABC1'.

A string of bytes containing '111'.

A single word with the address of 'ROUTINE1'. Notice the !.

3.11 ASSEMBLER STATISTICS

At the end of pass 2 (object code generation) the following statistics are displayed in order that you can see how close you are to the Assembler's processing limits. The Free Symbol Memory is the critical value, and it must always be 'well above' zero !

```
Number of Errors.....nnnn
Number of Symbols....nnnn
Symbol Table from....nnnn to nnnn
Macro List from......nnnn to nnnn
Number of X-Refs.....nnnn
X-Ref table from.....nnnn to nnnn
Free Symbol Memory...nnnn
File Start: nnnn  End: nnnn  Length: nnnn
```

3.12 SUMMARY OF ASSEMBLER DIRECTIVES

ASEG		Use the Absolute location counter.
COND	<exp>	Conditional assembly.
CSEG		Use the Code location counter.
DBXOFF		DB expansion Off.
DBXON		DB expansion On.
DEFB	<exp>,<exp>	Define bytes.
DEFL	<exp>	Redefine a label.
DEFM	<exp>	Define memory.
DEFS	<exp>	Define storage.
DEFW	<exp>	Define single word value.
DSEG		Use the Data location counter.
EJECT		Form Feed.
ELSE		Part of conditional assembly.
END		End of source input stream.
ENDBIN		End binary code generation.
ENDC		End of conditional assembly.
ENDIF		End of conditional (IF) assembly.
ENDM		End of MACRO.
EQU	<exp>	Equate a label.
EXEC	<exp>	Define EXEC address.
EXTRA		Enable extra opcodes.
EXTRN	<label>	Declare an External label
FREE		Enable FREE format.
IF	<exp>	Conditional (IF) assembly.
LIST		Turn on selective printing.
LOAD	<exp>	Generate zero bytes until <exp>
MACRO	<parm list>	Start macro definition.
MLIST		Enable macro listing.
MNLIST		Disable macro listing.
NOLIST		Disable selective printing.
NAME	<name>	Name a RELative object module.
ORG	<exp>	Define object code address.
PAUSE	message	Pause Assembly.
PRINTR	<parms>	Define printer options.
PUBLIC	<label>	Define a Public label
SELECT	<filename>	Append another file.
TITLE	message	Change Listing Header.

DB	same as DEFB
DL	same as DEFL
DM	same as DEFM
DS	same as DEFS
DW	same as DEFW

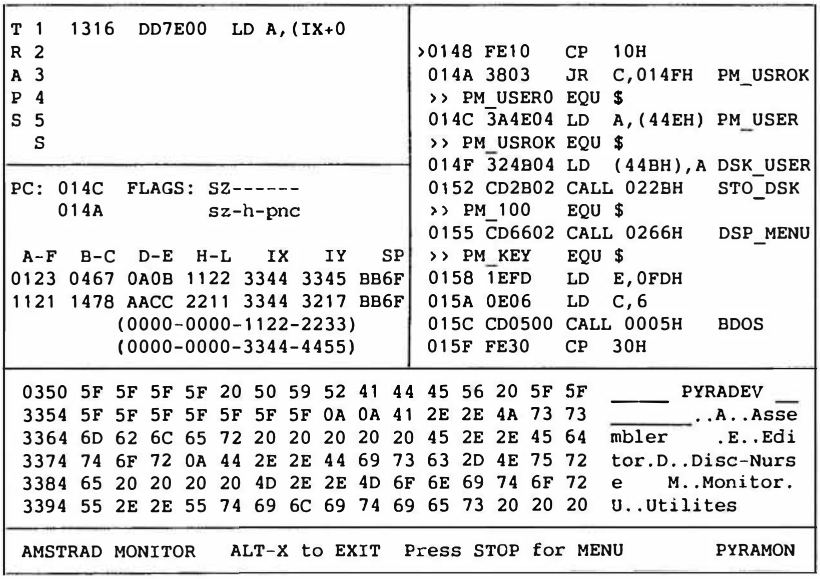
SECTION 4: MONITOR

The PYRADEV symbolic Monitor is a powerful debug monitor which provides all the features necessary to test and find the problems in your programs. It is selected from the SYSTEM-MENU by pressing (M).

It allows you to load test programs, set traps, start code execution, single step, change code, write code to disc and so on. In addition you can dis-assemble memory contents and write the dis-assembled source code to an ASCII file if required.

Memory contents can be displayed and printed in hex- and ascii format.

If used with a symbol file produced by the assembler, the monitor displays labels and symbols instead of just hexadecimal addresses and values. This very useful feature makes debugging much easier and reduces the need for a printed listing of the source code.



```
T 1  1316  DD7E00  LD A,(IX+0      |
R 2                                |>0148 FE10    CP   10H
A 3                                | 014A 3803    JR   C,014FH   PM_USROK
P 4                                | >> PM_USER0  EQU  $
S 5                                | 014C 3A4E04  LD   A,(44EH)  PM_USER
  S                                | >> PM_USROK  EQU  $
-----------------------------------| 014F 324B04  LD   (44BH),A  DSK_USER
PC: 014C  FLAGS: SZ------          | 0152 CD2B02  CALL 022BH     STO_DSK
    014A         sz-h-pnc          | >> PM_100    EQU  $
                                   | 0155 CD6602  CALL 0266H     DSP_MENU
 A-F  B-C  D-E  H-L   IX   IY   SP | >> PM_KEY    EQU  $
0123 0467 0A0B 1122 3344 3345 BB6F | 0158 1EFD    LD   E,0FDH
1121 1478 AACC 2211 3344 3217 BB6F | 015A 0E06    LD   C,6
          (0000-0000-1122-2233)    | 015C CD0500  CALL 0005H     BDOS
          (0000-0000-3344-4455)    | 015F FE30    CP   30H
--------------------------------------------------------------------------
 0350 5F 5F 5F 5F 20 50 59 52 41 44 45 56 20 5F 5F   ____ PYRADEV __
 3354 5F 5F 5F 5F 5F 5F 5F 0A 0A 41 2E 2E 4A 73 73   _______..A..Asse
 3364 6D 62 6C 65 72 20 20 20 20 20 45 2E 2E 45 64   mbler     .E..Edi
 3374 74 6F 72 0A 44 2E 2E 44 69 73 63 2D 4E 75 72   tor.D..Disc-Nurs
 3384 65 20 20 20 20 4D 2E 2E 4D 6F 6E 69 74 6F 72   e    M..Monitor.
 3394 55 2E 2E 55 74 69 6C 69 74 69 65 73 20 20 20   U..Utilites
--------------------------------------------------------------------------
 AMSTRAD MONITOR   ALT-X to EXIT  Press STOP for MENU        PYRAMON
```

Figure 4.0   Debug Monitor

4.1 SCREEN DISPLAY

The Monitor Screen is split into four different sections. The top left section contains trap information.

The middle-left section contains register contents (current and previous), cpu flag settings and the four bytes of memory pointed at by each of the address pair registers DE, HL, IX and IY.

The bottom section is a memory display. It alternates with a menu display.

The right hand section of the display is the main dis-assembly display.

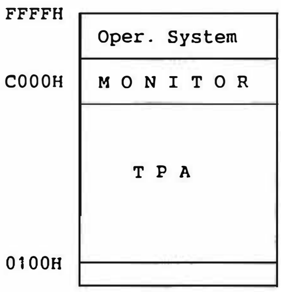
4.2 RELOCATION

The Monitor is by default loaded to the top of memory (TPA) but can be relocated to any part of memory. This is useful if you want to debug modules that themselves relocate to the top of memory.

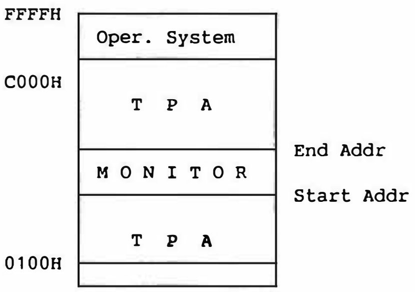
If you select to relocate the monitor it will ask whether you wish to specify a start address or end address.

Specifying a Start Address instructs the Monitor to relocate itself such that it's lowest address does not go below this start address.

Specifying a End Address instructs the Monitor to relocate itself such that it's highest address does not exceed the End Address.



Monitor loaded at top of Transient Program Area (default)



Monitor relocated to the middle of the TPA

4.3 MENU DISPLAY

When ready, the Monitor Menu can be displayed by pressing the ESC key. The menu alternates with the memory display section at the bottom of the screen.

```
T.....Trap  G........Goto  L...Load Q...Query  D^..Dis-Asm  C^.Cat
Z.Clr-Trap  G^...Step-1st  W..Write N.New-Scn  K.Load Symb  M..Mem
T^Dsp Cond  S....SStep(f1) O..Other R^..Reset  P^.Prnt mem  A^.MemA
R.....Regs  S^...DStep(f2) U.Update Y....CopY  B.Bank swap  X^.Exit
```

4.4 (L)..LOADING TEST CODE

Press (L) to load a program file and enter the file-name. The load-point will be displayed. You can alter the load point before pressing the enter key to read the file into memory.

4.5 (K)..LOADING SYMBOL FILE

Press (K) to load the matching Symbol File. This file is produced by the Assembler when you select SymDisc.

The name of the Symbol file is automatically displayed as filename.SYM where filename is the name of the .COM file that you loaded with "L".

The load address and size of the Symbol file is displayed. If you just press <Return>, the file will be loaded to the current top of memory and set a new top of memory. You can change the load point by entering a new load address. In this case, the top of memory information (in address 6-7) will not be changed. This is useful when you are debugging programs that need part of the common memory (above C000). Make sure that you do not overload part of the monitor with the symbol table file.

(ALT-K)..TOGGLE SYMBOLS ON/OFF

You can switch on/off the display of symbols by pressing ALT-K. This is only possible if you have loaded a Symbol file.

4.6 DIS-ASSEMBLY DISPLAY

A Right-Arrow symbol indicates where the address cursor is. Press the (O) key until the Right-Arrow symbol is at the top right screen display and enter a dis-assembly address. The display can be scrolled with the arrow and shift arrow keys.

If you have loaded a symbol table (K), you may enter a symbol name instead of an absolute address. Press (.). A highlighted field will appear above the disassembly fields. Type the symbol name and press return. The monitor will now search for the first symbol name that matches the entered name and display the corresponding address.

If the current Program Counter location is among the displayed instructions, that address is highlighted. This is automatically updated as you are single stepping through the code, and makes it very easy to follow the Program counter.

(P)..DIS-ASSEMBLE FROM CURRENT PROGRAM COUNTER

When you are debugging a module, you often want to enter the current Program Counter location as the first address in the dis-assembly. Hitting a "P" does this automatically.

4.7 (M) and (ALT-A)..CHANGING CODE and MEMORY

Use the STOP key until the memory display appears on the bottom of the screen. Press the (O) key to make the Right-Arrow symbol appear at the memory display address line. Enter the address to be changed. Press (M) to select MODIFY-MEMORY. Move the cursor and over-type memory as required by using two-digit hex-codes per byte. Instead of (M) you may use (ALT-A) to modify memory using ASCII. Press the ESC key after all changes have been made.

The display can be scrolled up and down by 1 byte, 16 bytes or 128 bytes by using the up and down arrow keys in normal, shifted and alt states as required.

4.8 (T)..SETTING TRAPS

Use the top right display to study code, then press (T) to set a trap. The monitor will now display: "CONDITIONAL ? (Y/N)". Press (N) if you want the program to stop at that address as soon as it is about to be executed, regardless of any conditions. Enter the first byte address of the instruction you wish to trap. One of the five trap lines will be used to show which instruction the trap has replaced. Use the (G)oto function to start executing code. When the trap occurs, the relevant trap line will be displayed in reverse inks. There are two ways to continue from a trap: S - Single-step code execution from the trap address. G - Goto address, continues execution.

When a trap occurs it automatically clears itself.

If you want the trap to be conditional, (example: only stop the program if the contents of address 1220 is greater than register HL or stop the 20th time the instruction is executed) you answer (Y) to the "CONDITIONAL ?" prompt. You will then be prompted for the condition that you want to put on that trap.

Conditions are entered in the following form:

```
        Operand  Relational-operator  Operand

        where the operand can be any of:

            any single register:    A, B, C, D, E, H or L
            any register pair:      BC, DE, HL, IX, IY, SP
            any hexadecimal number: 0AB12, 2, 1123 (start with digit)
            any symbol name:        .TABLE1   (start with a .)
            a counter:              CO
```

If you precede the operand with an asterisk (*) the contents of the address pointed at by the operand will be used instead of the operand itself.

The relational operator can be any of:

=	(equal to)
>	(greater than)
<	(less than)
>=	(greater than or equal to)
<=	(less than or equal to)
!	(not equal to)

Examples:

```
      HL = 0AB23      (stop if register HL contains 0AB23)
      B  > 8          (stop if register B is greater than 8)
      CO = 3          (stop the third time the instruction is executed)
      *HL >= .VAL1    (stop if the contents of the address pointed at
                       by HL is equal or less than the symbol VAL 1)
```

Note: The operands and operators must be separated with a space.

The current trap conditions are displayed if you press CTRL/T.

4.9 (G)..GOTO ADDRESS

After traps are set, or have occurred press (G) to start or resume code execution. A resume address will be displayed and can be accepted by pressing the Return key. To go to a specific execution point, overtype the address shown with one of your own choice.

4.10 (S)..SINGLE STEPPING

Press (S) or (f1) to SINGLE-STEP through code. This will execute the instruction at the current program counter address and follow the logical address path (control flow) as the code executes.

Use (ALT-G) to start single stepping from an address other than the current program counter. It will ask for the address and then execute the instruction at that address.

4.11 (ALT-S)..DOUBLE STEPPING

Press (ALT-S) or (f2) to DOUBLE-STEP. This form of stepping does not follow the logical program flow, but always sets a trap on the next instruction. Use this to avoid single stepping through an already tested subroutine (CALL ROUTINE) or to stop execution at the end of a loop (DJNZ ADDR or JR NZ,ADDR).

4.12 (R)..CHANGING REGISTER CONTENTS

Press (R) to change register contents, then carefully overtype the existing values with the new ones. Press the STOP (ESC) key to exit from the change function. The new values will be used when code execution is resumed either by using (G)oto or the single (S)tep function.

Note that the current Flag settings are changed by changing bits in the F-register.

4.13 (N) (U) (ALT-R)..NEW, UPDATE and RESET

The current memory display or dis-assembly display can be updated by pressing (U). This will be required when monitoring memory which is changing. When code execution is over-painting the screen (making it difficult to read), the (N)ew command can be used to update the entire screen. (ALT-R) is even more powerful than (U) in that it resets any windows before it repaints the screen. This is useful when you are debugging code that sets windows that are smaller than the full screen.

4.14 (Q)..QUERY, SEARCH

Pressing (Q) selects query mode and allows a HEX or ASCII of upto 30 characters to be searched for. Wild cards (?) may be used in both ASCII and HEX formats. As each match is found, it is displayed on line two of the lower-screen memory display. The search can be confined between a low and high memory address and can be stopped or continued at each match.

4.15 (ALT-D)..DIS-ASSEMBLE TO DISC OR PRINTER

Press ALT-D and enter a START and END address to identify the area of memory to be dis-assembled. Enter a Workspace address where the symbol table can be built. The default address supplied defines a 1000 byte area and may be used in most cases. If the space is exhausted by the dis-assembly process (about 500 labels) an error message will be displayed and the dis-assembly will have to be re-run with an alternate and larger work-space.

Data areas may then be defined by pairs of Start-End addresses. Use a zero-pair to terminate the selections. The output of the dis-assembly process may be selected with (V), (D) or (P) denoting Video, Disc or Printer. Note: References to non-existent labels may occur if embedded data-areas are not defined before the dis-assembly.

If (D) is selected enter the file-name to be used. Dis-Assembly to Video or Print can be paused with the Escape key and resumed with any other key. If the (D) option is used, a single Escape key aborts the process.

Dis-assembly to disc file(s) will pause after each 25K of source code has been generated so that subsequent file names can be entered before the process resumes.

4.16 (W)..WRITE BINARY TO DISC

To write a section of memory as a binary file press (W). You will be asked to enter the Start- and End Address of the section. After entering a file-name the relevant section of memory is written to disc as a binary file.

4.17 (Y)..COPY MEMORY

A section of memory can be copied by pressing (Y). Define the block to be moved with a BEGIN and END address. Then specify the destination address. The block will be copied 'intelligently'. This means the destination may be anywhere, and may overlap the initial block area.

4.18 (ALT-P)..PRINT MEMORY

To print part of the memory in Hex and Ascii format, press ALT-P. You will be asked for the BEGIN and END address of the block that you want to print.

4.19 (ALT-B)..MEMORY BANK- SELECTION

The memory in your computer is divided in blocks (banks) of 64k. The operating system (BDOS and BIOS) resides in bank 0. The program that you are testing is loaded into the Transient Program Area (TPA) which is in bank 1. Bank 2 is used for the Console Command Processor and data buffers (4000 - 7FFF hex). On the PCW, the rest of the memory banks are used for the RAM disc. The top 16k of the 64k (C000 - FFFF hex) is common to banks 0 and 1.

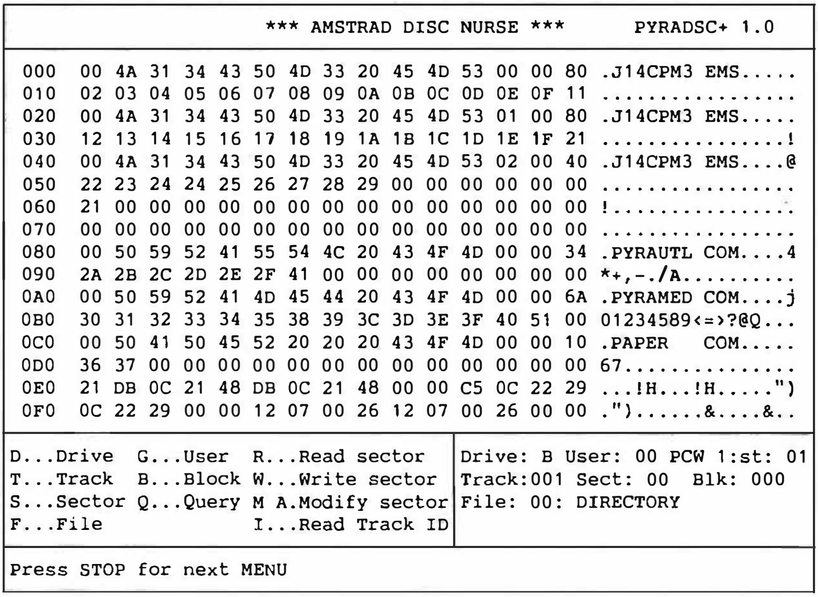
Press (ALT-B) to select a new bank of memory. This bank will now be selected when you display or change in memory.

SECTION 5: DISC-NURSE

5.1 DISC LOGIN

The DISC-NURSE is selected from the System Menu by pressing (D). It then waits for a disc to be loaded. After loading a disc the (D) key must be pressed again before the Disc-Nurse can be used. A number of functions are provided in a friendly and easy to use menu system. These allow you to explore your disc(s), and if necessary make changes.

The DISC-NURSE is capable of reading sectors of a disc even if the format of the tracks vary from track to track.



```
                      *** AMSTRAD DISC NURSE ***      PYRADSC+ 1.0

000  00 4A 31 34 43 50 4D 33 20 45 4D 53 00 00 80 .J14CPM3 EMS.....
010  02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 11 ................
020  00 4A 31 34 43 50 4D 33 20 45 4D 53 01 00 80 .J14CPM3 EMS.....
030  12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 21 ................!
040  00 4A 31 34 43 50 4D 33 20 45 4D 53 02 00 40 .J14CPM3 EMS....@
050  22 23 24 24 25 26 27 28 29 00 00 00 00 00 00 ................
060  21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 !...............
070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
080  00 50 59 52 41 55 54 4C 20 43 4F 4D 00 00 34 .PYRAUTL COM....4
090  2A 2B 2C 2D 2E 2F 41 00 00 00 00 00 00 00 00 *+,-./A.........
0A0  00 50 59 52 41 4D 45 44 20 43 4F 4D 00 00 6A .PYRAMED COM....j
0B0  30 31 32 33 34 35 38 39 3C 3D 3E 3F 40 51 00 01234589<=>?@Q...
0C0  00 50 41 50 45 52 20 20 20 43 4F 4D 00 00 10 .PAPER   COM.....
0D0  36 37 00 00 00 00 00 00 00 00 00 00 00 00 00 67..............
0E0  21 DB 0C 21 48 DB 0C 21 48 00 00 C5 0C 22 29 ...!H...!H.....")
0F0  0C 22 29 00 00 12 07 00 26 12 07 00 26 00 00 .")......&....&..

D...Drive  G...User  R...Read sector    Drive: B User: 00 PCW 1:st: 01
T...Track  B...Block W...Write sector   Track:001 Sect: 00  Blk: 000
S...Sector Q...Query M A.Modify sector  File: 00: DIRECTORY
F...File             I...Read Track ID

Press STOP for next MENU
```

Figure 5.0 Disc-Nurse

WARNING: You should not modify disc directory sectors without first making a backup of the disc. You can easily lose your favourite game, weeks of source code development or the entire disc contents. You must be aware of the AMSTRAD disc structures and file header constructions before changing anything. Please note this warning!

5.2 (D)..DRIVE SELECT

Press (D) and the prompt 'Select Drive' will be displayed. Press (A) ,(B) or (M) to select required drive and the relevant directory will be displayed.

You MUST do this if you change the disc(s) being examined !

5.3 (T) and (S)..TRACK and SECTOR

Press (T) to enter a track address, press (S) to enter a sector address. The number of the block that contains the sector is automatically displayed when the TRACK or SECTOR is changed. The name of the file that "owns" the sector is also displayed.

5.4 (B)..BLOCK NUMBER

Instead of specifying the sector and track, you can enter the corresponding block number. This is done by pressing (B). The block number is entered in hexadecimal.

5.5 (R)..READ SECTOR

Press (R) to read the selected sector. After a sector is read, the following keys can be used. They provide the ability to follow file chains or read forwards and backwards at sector level.

KEY:	Normal	Shift	Alt
LEFT	Chain back 1/2 sec	Chain back 1 sec.	Locate Beginning
RIGHT	Chain frwd 1/2 sec	Chain frwd 1 sec.	Locate End
UP	Previous 1/2 sector	Previous sector	Not used.
DOWN	Next 1/2 sector	Next sector	Not used.

Figure 5.1

5.6 (M) and (A)..MODIFY SECTOR

Press (M) to modify using HEX or (A) to modify using ASCII. Use the arrow keys to move the cursor to the required bytes and over-key the values as required. If you modify File-Header bytes, after making the changes press (CTRL-H) to re-calculate the checksum byte (This applies only to AMSDOS BIN files, not CP/M files). When the sector changes are complete, press the STOP (ESC) key. The changed sector can be written to disc with the (W) command ...

5.7 (W)..WRITE SECTOR

Press (W) to write the displayed/modified sector back to disc. The write request must be confirmed with the (Y) key. Any other key aborts the Write Request.

WARNING: The sector will be written to the Track and Sector shown on the bottom right of the screen.

5.8 (E)..DISPLAY ERASED FILES

Pressing (E) will display the names of all erased files on the selected disc. These files can be un-erased with the (U) command (see below).

5.9 (U)..UN-ERASE FILE

This can be used to re-claim a file which has been accidentally deleted. Press (U) to select the un-erase function and enter a file-name. The DISC-NURSE will check the sector-allocation tables. If the file sectors have not been used, the file will be restored for normal use and will re-appear on subsequent directory displays.

5.10 (F)..FILE SELECTION

To access the sectors belonging to a specific file, press (F) and enter a file-name. The first sector of the file will be read. See the (R)ead function above for a description of the scroll/browse key functions.

5.11 (Q)..QUERY/SEARCH MODE

Pressing (Q) selects the query / search function. The search can be limited to a (S)ector, (F)ile or (D)isc and may be for an ASCII string or HEX string. Wild Cards (?) are permitted.

The hex string may be entered as a continuous or broken string of hex numbers, ie NNNNNN or NN NN NN.

The search starts from the current track/sector position and proceeds to the last sector. Searching for an un-likely ASCII string is a good way of checking a disc.

When a match is found, the sector address, word offset and sector contents are displayed on the screen. The search can be continued by pressing the SPACE bar, or terminated with the STOP (ESC) key.

5.12 (C)..CATALOGUE

Pressing (C) will display the a sorted disc catalogue with the size of each file.

5.13 (X)..EXTENDED DIRECTORY

For extended directory information and hard-copy facilities press (X). If the output is to be printed, answer (Y) to the print question. A title line may be entered which will appear at the top of the listing.

This feature is particularly useful for AMSDOS files, as it reads the header information for each file and displays it (or prints it) as Filename, Filetype, Protection, Size, Start address, End address, Logical length and Execution address.

5.14 (P)..SECTOR PRINT

The current sector will be printed in HEX and ASCII format when you press the (P) key.

5.15 (Y)..COPY SECTOR DATA

The two last features of the DISC-NURSE can be used to copy information from sectors into a file on any disc. This file can then for instance be disassembled by the MONITOR.

Once a sector has been read with the (R) command, the whole sector or part of it can be copied into memory by pressing (Y).

A prompt: "Copy 200H bytes from 000H to 6D00H" is displayed.

All numbers and addresses are entered in hexadecimal. Change the first field if you want to copy less than a full sector (512 = 200 hex bytes). The second field indicates where in the sector the copy should start from. (000 for a full sector copy). The last field displays the next free address in memory where you can store your copied sectors. Note, this address will automatically be updated for each copy you do.

5.16 (V)..WRITE MEMORY TO DISC

The sectors that you have copied to memory can now be written to a disc file by pressing (V). The DISCNURSE will prompt you for the start address and the end address of the memory that will be written to disc. The addresses used by the copy command (Y) are inserted in the prompts and will be used if you just press return. Before writing to disc, you will be asked for the name of the file that will be created.

The STOP- (ESC)- key can be used in this command (as in most other command in PYRADEV) to abort the command and return to the menu.

5.17 (I)..READ TRACK ID

Each track on the disc contains a number of sectors. Each sector is preceded by a four byte Sector ID block containing information about the sector. This information is displayed for all sectors on the current track when you press (I).

5.18 DISCS WITHOUT A DIRECTORY

Some discs, do not contain a proper file directory. When this is detected by the Disc Nurse, a message *** NO DIRECTORY *** is displayed. All file dependent commands are then disabled until you log in a disc with a file directory.

SECTION 6: UTILITIES

The Utilities Program provides general file management and copy facilities in a single and easy to use package. The following features are provided:-

- o Directory Display of Drives (A),(B) or (M)
- o File Renaming
- o File Erasing
- o File Transfer; any AMSDOS file type, Tape and Disc.
- o Tape <--> Disc copying. (6128 only)

6.1 INITIAL PROMPT

The following prompt is displayed ....

```
A> A,B,M:Disc C:Copy D:Delete R:Rename Z:ZapBAK 00-15:User X:EXIT
```

Selection of (A),(B) or (M) displays the appropriate directory.

Entering a two digit number between 00 and 15 will set the User number. Once set, the prompt will show the selected User number and Disc. Example: 2B> indicates that User 2 on drive B will be used when accessing files unless another User or Disc is specified in the file name.

6.2 DELETE/RENAME FILES

Selection of (D) or (R) provides file DELETE and RENAME functions and need no further explanation. (Wild cards, * and ?, may be used).

6.3 COPYING FILES

On the PCW, only Disc - Disc copying is possible for obvious reasons. Pressing (C) will prompt you for the name of the file that you want to copy. The file-name can be preceded by a User number and/or a Drive character e.g. 1B:prog1.asm will copy the file "prog1.asm" from User 1 on drive B.

The file is now read from the disc into memory, and file-information is displayed.

o Copy on the CPC6128

The Copy Function is a general purpose copy routine which will copy ANY standard AMSDOS file to and from DISC or TAPE. When selected the following prompt appears:-

1: Disc-Disc   2: Disc-Tape   3: Tape-Disc

A valid replay must be given or control returns to the initial prompt. After a selection of 1,2 or 3 an INPUT file name must be entered. A file-name is not required for option 3.

After the INPUT file is opened, the header information is displayed. The copy can then be continued by responding (Y)es to the COPY-? prompt.

Depending on the option chosen and file-type detected, the copy operation will go through a number of prompts.

If the file is protected, you can make it unprotected by answering "Y" to the prompt: "Remove Protection ?"

NOTE:- The Destination Tape or Disc may be changed BEFORE the reply to the second "File-Name:" is entered.

6.4 DISC-CLONE

The DISC-CLONE is selected by pressing Y from the main PYRADEV menu. It is a very powerful program which lets you copy one 40-track disc onto another. It works with one as well as two drives. It copies the discs track by track, examining and duplicating the format of each track. The DISC-CLONE should only be used to make back-up copies of your own discs

Once selected, the following menu is displayed:

```
Copy from     A
Copy to       A
From track    00
To track      39
Reading/Writing
Print info:   NO

Press Y to start
   or X to exit
```

Use the down/up arrow keys to move to any option that you want to change. Then use the left/right arrow keys to change the option.

Start the copying by pressing Y.

If you selected to copy from B to A, a message:

Insert the SOURCE disc in drive B and the DESTINATION disc in drive A

Hit 'Y' to continue or 'E' to exit

is displayed. Press Y and the first tracks will be read from the source disc (drive B). If you have two drives, these tracks will be formatted and copied onto the destination disc. If you are using a single drive system, the program will ask you to switch to the destination disc before copying the first tracks. The program then reads the next tracks etc until all requested tracks are copied.

For each track read, the sector numbers and sector IDs for that track are displayed. The contents of three status registers from the disc controller chip is also displayed. Any error information in these registers is translated into a 2 character code: DE Data error in ID block, ND No Data (Sector ID block without a sector), CM Control Mark (The sector is marked a deleted data), DD Data error in Data, WC Wrong Cylinder (The track information in the ID filed does not match the current track), MD Missing address mark in Data field.

By selecting Reading only you can examine a disc without copying it.

SECTION 7: SELF-TEACH

7.1 DEMONSTRATION PROGRAMS

As you may have realised, the PYRADEV system is small but powerful. There are many features and functions to explore and new commands to learn. For this reason the file 'PROGRAM.001' is supplied on the MASTER DISC for you to Assemble, Edit and generally play with.

The program contains a single routine called 'DEBUG'. It can be ran from CP/M and will display register contents on row 25 of the screen. The program is very heavily commented with two types of text.

The (UPPER CASE TEXT) enclosed in brackets is all about the program code. Hopefully you will quickly see how the routine works, and perhaps adapt it as an additional debug routine for code that you will be writing.

The ;* Normal Text *; enclosed in semi-colons contains information about using the Source File Editor. You should read these comments and practice the functions. When you have finished 'playing' in the Editor, press ALT-A to abort, then (Y) to confirm and exit.

The file PROGRAM.001 can be Assembled as it is. You should do this at an early stage with the Macro Assembler, and use the various OUTPUT options to see what happens. You are advised to use the default options to start with.

The file PROGRAM.002 is also supplied on disc. This is a complete example of the Z80 instructions. If you have any problems with your code, check that you are using correct syntax, and code mnemonics by looking in this program. It also contains examples of the Assembler Directives described in section 3. It is a useful piece of reference code and can be Assembled and Listed as a reference chart.

The file PROGRAM.003 is a simple FILE-COPY program. It can be modified and used to transfer other ASCII file formats into the PYRADEV system by adding custom code to effect special changes during the transfer/copy operation.

SECTION 8: TUTORIAL

8.1 GENERAL INFORMATION

In addition to the three programs mentioned above, a fourth one, **DSPKEY**, has been supplied with PYRADEV. This program will, once assembled and run, display the code in hexadecimal for any key you press.

That program will be used in the following sections, to take you through most parts of PYRADEV.

The tutorial is written for the PCW keyboard. If you are using a CPC machine, the **CONTROL** and **ESC** keys on your keyboard correspond to the **ALT** and **STOP** keys described in this tutorial.

If you haven't made a copy of your Master PYRADEV disc by now, please do so before you start the exercises.

<name> below means press the named key, ie <STOP> press the STOP key, and <C/R> the RETURN key.

When you are asked to enter "ABC" just press the keys A B C, not the quotation marks.

Note that with all commands in PYRADEV, it doesn't matter if you enter them in Capital letters or in lower case. If it says "Press <P>" you can either type a <p> or a <P>.

The only case dependent feature in PYRADEV is when you specify a symbol in the Monitor. This is because "AAA" and "aaa" are treated as two different labels in the Assembler.

8.2 SETTING UP YOUR WORK-DISCS

Before we start the exercise, and indeed before you start any project, you have to create the disc where you are going to keep your files.

CREATING YOUR PYRADEV-DISC (PCW)

The most efficient way of using PYRADEV, is to keep all PYRADEV files in the memory drive, M:, and your project files on a disc in drive A: or B:. To create a PYRADEV disc that automatically copies all necessary files to drive M: and starts PYRADEV do as follows:

Your action	Result / Comment
RESET the computer with the CP/M PLUS SYSTEM/UTILITIES disc in drive A. (Press SHIFT and EXTRA. Without releasing them, press EXIT)	CP/M is restarted. The memory disc (M:) is cleared.
	CP/M Plus Amstrad ....
	A>
PIP <C/R>	**CP/M 3 PIP VERSION 3.0**
	*****
	(Start the program that will copy files)
M:=PIP.COM <C/R>	(Copy PIP.COM to drive M:)
	*
M:=SUBMIT.COM <C/R>	*
M:=SETKEYS.COM <C/R>	*
M:=*.EMS <C/R>	COPYING-
	J14CPM3.EMS (can vary)
	(the CP/M operating system is copied)
	*
<C/R>	(End of copying)
	A>
Change disc to your copy of the PYRADEV disc (side A, PCW)	
M:PIP A:=M:*.*	(Copy all files on drive M: to the PYRADEV disk in A:)
	COPYING-
	PIP.COM
	SUBMIT.COM
	J14CPM3.EMS
	A>
REN KEYS.DEV=KEYS.PCW	(Rename the file KEYS.PCW)
REN PROFILE.SUB=PROFILE.PCW	(Rename the file PROFILE.PCW)

You have now created your Turnkey PYRADEV disc.

CREATING YOUR PYRADEV-DISC (CPC)

Side B of the PYRADEV disc is formatted as System Format, but does not contain the operating system, i.e. the 'EMS-file'. You could therefore use your copy of the disc as it is and boot CP/M from another System disc.

It is however recommended that you use PYRADEV directly from a system disc. You can then boot the disc, and PYRADEV will start automatically.

To create a PYRADEV disc on a System disc do as follows:

Your action	Result / Comment
Copy side B of the PYRADEV to a new disc by using DISCKIT3 under CP/M.	A>
Put the CP/M PLUS SYSTEM/UTILITIES disc in drive A.	
(If you have two drives, put your new PYRADEV disc in drive B, if not, CP/M will tell you when to switch discs.)	
PIP <C/R>	CP/M 3 PIP VERSION 3.0 * (Start the program that will copy files)

B:=SUBMIT.COM <C/R>
B:=SETKEYS.COM <C/R>
B:=*.EMS <C/R>
	*
	*
	*
	COPYING -
	J14CPM3.EMS (can vary)
	(the CP/M operating system is copied)
	*

<C/R> (End of copying)	A>
(Put your copy of the PYRADEV disc, in drive A)	
REN KEYS.DEV=KEYS.CPC	(Rename the file KEYS.CPC)
REN PROFILE.SUB=PROFILE.CPC	(Rename the file PROFILE.CPC)

You have now created your Turnkey PYRADEV disc.

CREATING YOUR PROJECT-DISC

You should now create a disc where you will keep your working files. All you have to do at this stage is to format a disc. We will use PYRADEV to copy the files you need later on.

Use DISCKIT3 to format a new disc. (Use Data format on the CPC)

You have now created the Project disc where you will store your programs.

Starting PYRADEV from your Turnkey (PYRADEV) disc

On the PCW

Put your PYRADEV disc into drive A: Reset the computer by pressing SHIFT EXTRA AND EXIT.

CP/M will now start by executing all commands in the file PROFILE.SUB These include redefining the keyboard to fit PYRADEV and copying all PYRADEV programs to drive M. Finally PYRADEV starts and its main menu will be displayed.

On the CPC

Put your PYRADEV disc into drive A: Reset the computer by pressing SHIFT CONTROL AND ESC. Type |CPM <C/R>

CP/M will now start by executing all commands in the file PROFILE.SUB These include redefining the keyboard to fit PYRADEV. Finally PYRADEV starts and its main menu will be displayed.

8.3 MAIN MENU

Printer parameters

Press <P>. The current Left Margin and Page Length parameters are displayed. Press <M>. Enter <05> to change the Left Margin. Press <P> again. Press <L>. Enter the form length of the paper that you are using. <66> if you are using 11-inch paper or <70> for A4 format paper.

Typewriter mode: This mode is mainly for sending control codes to the printer, but you can of course also use it to print any lines on your printer. Note that the line you type is not sent to the printer until you press <C/R>. If you start a line with a non alphanumeric character, e.g. <ESC>, the <C/R> will not be sent to the printer.

Press <P>. Select typewriter mode by pressing <T>. Type your name and <C/R>. Press <C/R> a few times. Press <STOP> <4> <C/R>. Type your name again followed by a few returns <C/R>. Note that the printer now has changed to the *italic character set* Type <STOP> <5> <C/R> to change back to the normal character set.

Press <ALT-C> to exit the typewriter mode.

Print File: This mode is useful when you want to send a file with control characters to the printer. Your PYRADEV disc contains a file, SETUP.PRT, which when sent to the printer sets it to:

Printer function	Control code	Code in file
continuous mode	ESC c	↑<c↑
override paper end	ESC 8	↑<8↑
pagelength 70 lines	ESC CF	↑<CF↑
condensed typestyle	ESC <SI>	↑<↑0↑

Note that an up-arrow <↑> in a document is treated in a special way when the document is printed. The <↑> itself is ignored, but 33 is subtracted from the next character before it is sent to the printer. ↑< will therefore be sent as 50-33 = 27 which is the code for ESC. The ↑ at the end of each line makes the program ignore the invisible <C/R> which follows the ↑.

The following list shows the characters you should use after a ↑ to send the wanted code to the printer.

Code	+33	Char	Code	+33	Char
0	33	!	20	53	5
1	34	"	21	54	6
2	35	#	22	55	7
3	36	$	23	56	8
4	37	%	24	57	9
5	38	&	25	58	:
6	39	'	26	59	;
7	30	(	27	50	<
8	41	)	28	61	=
9	42	*	29	62	>
10	43	+	30	63	?
11	44	,	31	64	@
12	45	-	32	65	A
13	46	.	33	66	B
14	47	/	34	67	C
15	48	0	35	68	D
16	49	1	36	69	E
17	50	2	37	60	F
18	51	3	38	71	G
19	52	4	39	72	H

Example: To set left margin to 10, the sequence ESC l ‹10› should be sent to the printer. As ‹10› represents the character LF (line feed) it has to be represented as a ‹↑› and the character for 10+33=43 which is a plus sign ‹+›. As the ESC (27) is represented by a <, the final sequence in the document would be: ↑<↑+

Now, to initialise the printer as described above, press <P> <F>. Type SETUP.PRT <C/R> in response to the prompt "File name:"

Set User

You can divide your discs in up to 16 different sections (also called users or groups). You can set the User from the main menu or from any of the PYRADEV programs.

Just for the practice, set User 2 and then back to 0:

Enter <02>. Note how the last line changes to show the new User.
Enter <00>. You are now back to the original User again.

Select Disc

All PYRADEV programs, Editor, Assembler etc are always loaded from the same disc as PYRADEV was started from (Drive M on the PCW). Your Project files however, can be loaded from any disc drive. You can always specify a drive by preceding the file-name with a drive character. Example: B:FILE.ASM will load the file FILE.ASM from drive B. If you don't specify the drive, ie just FILE.ASM , the program will try to load FILE.ASM from the currently selected disc drive.

You can change the current drive either by preceding a file name with a drive character or by Selecting Disc from the main menu.

As you will use Drive A for your programs, change the current disc drive to A:

Press <S> until you see Disc: A on the last line.

8.4 UTILITIES

Press <U> to select Utilities. Press <A>. The names on all files on your PYRADEV disc are now sorted and displayed with size information.

If no files are displayed, you are either looking at the wrong User (enter "00" to change to User 0) or you haven't got the PYRADEV disc in drive A)

Copy DSPKEY from the PYRADEV disc to your Project disc.

```
 Your action                                Result / Comment
Press <C>
                                            (Copy function starts)
                                           File-Name:
(Note that you can Abort this, and any
 other function by pressing <STOP>)

"DSPKEY" <C/R>

                                            (The file DSPKEY is searched
                                            and examined:)
                                           HEADER: Type: ASC
                                           Copy ? y
<Y>
                                           Compression ? n
<Y>
                                            (The file is read from disc
(Textfiles can be compressed to save         into memory)
 space on disc)
                                           File-Name: DSPKEY
(Change Disc to your Project Disc !)
<C/R>
                                            (The file is written from
                                             memory to disc)
                                           COPY DONE
<C/R>
```

A> A,B,M:Disc ......

Copy your new file and name the copy TEMP (press space twice to blank out the last two characters in DSPKEY)

Press <A> to make sure that both files are on your disc.

<u>Rename file</u>

Press <R>

File-Name:

"temp" <C/R>

NEW-NAME:

"test" <C/R>

A> A,B,M:Disc ......

<u>Delete file</u>

Press <D>

File-Name:

"test" <C/R>

(TEST will be deleted)
The directory is displayed.

Press <X> to exit utilities.

The PYRADEV Main Menu is displayed.

8.4 ASSEMBLER

The program that you have copied from the PYRADEV disc to your Project disc contains the source code (program statements) to a program. You have to create a binary file, load file, to be able to run it. You use the Assembler to translate all those program statement into a load file.

Press <A>. The Assembler menu will be displayed.

Enter DSPKEY <C/R> as the Input file. Note how the assembler automatlcally generates the file name DSPKEY.COM as the Binary file name. As this is a suitable name for a CP/M load file, just press <C/R> to accept that name.

If you just wanted a standard assembly, with no printing or Symbol Table on disc, you would just press <C/R> again (and thereby accepting the defaults) and the program would be assembled.

To make it easier to debug the program with the monitor later on, you should create a Symbol Table on disc. If you have a printer connected you might also want a listing of the program. Press <n> <C/R> <C/R> <y> <C/R> <C/R> <y> <y> <y> <C/R> <C/R>.

This will create a loadfile called DSPKEY.COM. The program will be printed with both a symbol-table and a cross reference table. The X-ref table is particularly useful for programs consisting of several source files. It lists all symbols with information of all files and lines which reference the symbols.

While the file still is assembling, press <V>. The instructions will now scroll on the screen as they are examined by the assembler. Press <STOP> to pause the assembler. Press any key to continue and then <V> again to stop the assembler from displaying.

Once the message Assembly complete. Any key to continue is displayed, press any key and the PYRADEV Main Menu is displayed again.

The assembler has now in addition to the .COM file also generated a symbol table file called DSPKEY.SYM on disc. This file will later be used when you use the Monitor.

8.5 MONITOR

Press <M> to activate the monitor. A message:

```
Monitor will be loaded between BA00 and ED1D hex
Do you want to relocate ? y
```

will be displayed.

Normally, you just press <C/R> to accept to default loading of the monitor. The only case when you would want the monitor to be relocated is when your program has to use the common memory area starting at C000 hex.

In this case it doesn't matter, so let's relocate the monitor.

Press <y>.

```
Do you want to specify Start Address ? Y
```

is now displayed. Press <C/R> for default Yes.

In response to the next prompt:

```
Enter Start Address between 0100 and BA00: BA00
```

enter "7000"

```
Monitor will load between 7000 and AB1D hex
Any key to continue or <STOP> to change.
```

Press <C/R>.

The monitor screen will now be displayed. Press <STOP>. A help menu describing all functions is displayed instead of the memory display. The up-arrow <^> in this display represent the <ALT> key. Example: <ALT-C> displays the disc directory.

Press <STOP> again.

You should now load the program that you want to test:
Press <L>. Type "DSPKEY.COM" <C/R>.

The load address 0100 and length 0180 is displayed.
Press <C/R> to accept the load address

The 15 first instructions of the program are now displayed in the top right corner of the screen. The right-arrow head, ">" which is located just to the left of those instructions, indicates the current address, i.e. if you would enter a 4-digit address now, or use the up- or down arrows, it would change the address of the first disassembled instruction. Press the down-arrow a couple of times. The display will move one instruction every time. Press <shift-down-arrow>. The display moves 15 instructions ahead. Press up-arrow. The display now moves one byte backwards, and tries to disassemble from that new byte. As one instruction could consist of between one and four bytes, you might have to repeat the up-arrow a couple of times until the disassembler is in "the right phase".

Enter 0100 to return the display to the starting address of DSPKEY.COM. Note that the address 0100 is display in inverse video. This indicates that this is the current program counter, i.e. the next executed instruction is located at address 0100.

Press <O>. The right-arrow is now pointing at the memory display in the bottom part of the screen. Press <ALT-down-arrow>. The next 128 memory cells are displayed. Press <shift-down-arrow> to move the memory display another 16 cells.

Changing memory contents

Press <M>. The cursor now located on memory cell 0190. Move down to 01F1 by pressing the down-arrow 6 times and the right-arrow twice. Type "41". Note how the ASCII display on the right hand side changes from **CTRL/C** to **CARL/C** as you changed the cell. Press <STOP> to exit the change mode. Instead of changing the hex value of a cell, you can change it directly in ASCII: Press <ALT-A>. The cursor is now in the ASCII area of the display. Move down to the first <C> in CARL/C (Down-arrow 6 times). Type <space> <A> <L> <T>. Not how the hex-value changes at the same time. Press <STOP> to exit the change mode.

Writing to disc

You should now save the changed program to disc: Press <W>.

```
    Write file to Disc
       From Address (HEX): 0100
```

is displayed. Press <C/R> to accept the starting address, and again when To Address (HEX) 027F is displayed.

Type "TEMP.COM" in response to the File name: prompt.

Press <ALT-C> to confirm that the file "TEMP.COM" now has been created on disc. Press any key to get back to the monitor display.

Disassemble to printer

Disassembling, converting object code back to source instructions, is essential when you are trying to understand a program but you don't have the original source code. You can disassemble any part of the memory to either the screen, the printer or into a disc-file.

Press <ALT-D> and you are prompted for the starting address. Enter "0112". Type 0180 in response to the **To Address** prompt. The suggested workarea should be sufficient for this disassembly, so just press <C/R>. Next, you are prompted for start- and end addresses of data areas within the entered range of instructions. As the ascii representation of all bytes are displayed/printed as comments to the disassembled instructions, it is not that essential to define all the data areas correctly. Just press <C/R> twice in response to **Data** Start: 0000 and Data End: 0000. (D)isc, (P)rinter **or** (V)ideo ? is now displayed. If you have a printer connected, press <P>, else <V> to display the instructions to the screen. You can pause the display by pressing <STOP> and the press any key to continue.

Loading the Symbol Table

Look at the first two instruction on the top right hand side of the screen:

```
0100  21 A2 01        LD    HL,01B5H
0103  CD 6F 01        CALL 016FH
```

Unless you have a recent printout from the assembly, it could be difficult to remember what all the labels are and which call was which. Now for a bit of magic... Press <K>. The prompt **Symbol Tab: DSPKEY.SYM** is displayed. Type <C/R> to accept the suggested file name. This is the Symbol Table file that you created by answering <Y> to "Sym Tab" when you assembled DSPKEY. That symbol table is now loaded into memory.

An extra column of information has now been added to the disassembled instructions in the top right hand part of the screen. The first two instructions now read:

```
0100  21 A2 01        LD    HL,01B5H   MESS01
0103  CD 6F 01        CALL 016FH       DSPMSG
```

It is now easy to see that you load HL with the address of the first message MESS01 and the make a CALL to DSPMSG to display that message. The program is now much easier to follow.

A bit further down you can now see: >>> LOOP EQU $ This is how a label is displayed when a Symbol Table is loaded.

Let's have a quick look at the subroutine called BINHEX. So where is it located ? Press full stop <.>. The blinking cursor moved to an empty field above the instructions. Type "BINHEX" in capital characters followed by <C/R>. The monitor found the address to BINHEX and displayed those instructions.

Type 0100 to redisplay the first instructions of DSPKEY.

Stepping through the program

The main use of a monitor, it to find bugs in the program that you are developing. To make that easier, you have to be able to stop the execution of the program, examine registers and memory, and then continue to the next stop. There are three ways of setting these stops, or traps:

a) You can set a trap at any address that you specify. These traps can be conditional, i.e. the program stops only if a specific condition is met. Up to five different traps can be set in this way.

b) You often want to execute just one instruction , check the result and then execute the next instruction and so on. This procedure, which is called single stepping, can of course be achieved by using method a) and always specifying the next instruction address as the next trap address.

c) This would take a very long time. Instead you can just press <S>. The program then automatically sets a trap on the next instruction address, and continues execution from the current address. The next <S> continues with the new instruction and so on. The contents of all registers are automatically updated on the screen.

While single stepping, it is often desirable to execute a subroutine without single stepping through it. You could do this by setting a trap on the instruction after the call. A faster way is to press <ALT-S>.

This procedure is called double stepping. It also sets a trap on the next instruction, but does not follow the program flow. It is also useful for setting a trap at the end of a conditional loop. A normal single step would follow the loop until it ends, while a double step sets the trap after the loop, and catches the program when the loop ends.

Enough of theory. Press <S>. Note how the next address, 0103, now is displayed in inversed video. The first instruction, **LD HL,01B5H** has just been executed. Look at the contents of register HL in the Register section. It has changed to 01B5. You can see the old contents just below the current one.

Press <S> again. The inverse address disappeared from the disassembled instructions. Look at the contents of the Program Counter (PC) in the Register section. It has changed from 0103 to 016F. The line above the Registers and FLAGS, contains the next instruction to be executed: **016F CALL 0194H CURPOS**

As it is much easier to follow the program if you can see the current instruction among the disassembled ones, you want to change the address of the first instruction to be the current address of the program counter, 016F. Instead of typing 016F, you can just press <P>. Do that. The first two lines now read:

```
>>> DSPMSG       EQU  $
 016F CD 94 01  CALL 0194H        CURPOS
```

DSPMSG is the routine that displays a message on the screen. HL contains the address to the message, and the end of the message is indicated by a zero. It would be interesting to see what the message looks like in memory, before it is displayed on the screen. The address of the message is 01B5, the contents of HL, so this is the first memory address you want to display.

Type 01B5. The message is now displayed both in hex and in ascii. The first two bytes, characters, of the message are used to position the cursor before the rest of the message is displayed. This first message consists only of blank characters, hex 20.

The subroutine DSPMSG starts by calling another subroutine CURPOS. CURPOS uses the first two characters pointed out by HL to send an ESCape sequence to the monitor. This sequence will make the cursor move to the wanted row and column. Let's not single step through the routine CURPOS, but just set a trap where it returns: The fast way of doing that is to press <ALT/S>. Do that.

The next instruction is: LD A,(HL). Press <S>. Register "A" is now loaded with the first byte pointed out by register "HL". Press <S> again. The instruction OR A is used to set the FLAGS depending of the contents of register A. One of these FLAGS is the ZERO flag, which is set if a zero condition is met, i.e. if register A contains a zero when the instruction OR A is executed.

The next instruction: JR Z,017CH DM_RET jumps to address 017C only if the zero flag is set. Look at the FLAG display to the right of the PC register display. The position above the lower case "z" is a "-". This indicates that the ZERO-FLAG is not set, and the program will not jump to 017C. Press <S> to prove it.

The next instruction is a CALL to CHAR_OUT (017A). Let's not single step through that routine but just set a trap when the program returns from CHAR_OUT. Press <ALT-S> followed by <S> and <S>. The subroutine will now continue to display characters until a zero is found. It will then jump to address 017C. As this is the next instruction, you are currently on 017A, <ALT-S> will set a trap on the wanted instruction (it ignores the logical flow of the program and just goes for the next instruction). Press <ALT-S>. Note that register "A" now contains a zero, and that part of the monitor display, the fourth row of the ASCII display, has been blanked out by your blank message. Press <S> to follow the RET instruction back to the calling routine. Type <P> to move the display to the current Program Counter address.

Type <S> <S> <P> <ALT-S>. You should now be at **0172 LD A,(HL)**. The message "Displaying Key Codes" is about to be displayed. Let's trap the program just before it is about to display the "K" in "Key". Type <T>. A prompt "Trap address 016F" is displayed. Type "0173" to trap the program after the **LD A,(HL)** instruction. Press "Y" in response to the Conditional **(Y/N)** ? prompt, as you want to put a condition on the trap. Enter condition: is now displayed.

Type: "a = 4b" <C/R> (Note the spaces to separate the "a" from the "=" and between the "=" and the "4b"). You have now told the monitor to stop at 0173 only if register A equals 4B which is the ASCII code for a "K". Press <ALT-T> to display the trap conditions. Type space to continue.

Press <G> followed by <C/R> to continue execution from the current address. When the program stops, **Displaying** is displayed in the lower right hand part of screen. The first trap-line is displayed in inverse video to indicate that this trap has been executed. Register A now contains "4B" which was the condition you set on the trap.

Change Register Contents

You should now change the contents of register A from "K" to "L" i.e. from "4B" to "4C". Press <R>. The cursor moves to the contents of the Program Counter, PC. Type <C/R> to move to the next register pair A-F. (F -s the flag register). Type "4C" and press the right-arrow twice. Press <STOP> to exit the Register Change mode.

Query/Search

You can search for bytes in any part of the memory. These bytes can be defined either in Hex or Ascii.

Press <Q> to select the search function. Type <A> in response to the **SEARCH for** A(scii) or H(ex) ? prompt. Let's search for character strings that start with a <P> then any character followed by a <R>. "Any character" is represented by a <?> so type "P?R" <C/R>. You are now asked to enter the start and stop address for the search. Type <C/R> twice to search the whole memory. After a second, the first match is found and

Match found at 0207
Space to continue, or STOP to Exit

is displayed in inverse video. The memory display is also updated. It displays memory from the found address - 10 hex. In this case from 01F7. The Ascii part of the display, shows the hit: PYRAMID

Press the spacebar to search for the next match. Press <STOP> to stop the search.

Memory Bank Switch

Finally, let's try the "Bank Switch".

Type <ALT-B>. Enter "00" in response to the prompt **Memory block:**. Note how the contents of the memory display changes. You are now displaying memory used by the CP/M operating system, so be careful not to make any changes as it could change the behaviour of the system.

If you have a PCW type <Q> to enter the search function and search for the Ascii string "RESET". When found, press <STOP> to exit the search. Press <ALT-A> to change memory in Ascii. Move to RESET on the second line. Change it to "HELLO" and press <STOP>. What is it you just changed ? Press <PTR> for the answer. Press <EXIT> to get back.

Type <N> to redisplay the screen before you press <ALT-X> to exit the monitor.

8.7 DISC NURSE

Before you select the DISC NURSE, press <S> from the main menu until Disc A is selected. Type <U> to select Utilities. Press <D> to delete a file. Type "DSPKEY.COM" <C/R>. Yes, go ahead, we'll get it back! Press <X> to get back to the PYRADEV Main Menu.

Enter <D> to select the DISC NURSE. Press <D> again to "log in" the disc. A sorted disc directory is displayed.

Un-erasing a file

You will start by getting the file that you just deleted back. Press <E>. Another disc directory is displayed. This one however, consists of all deleted files on the disc. Press <U> to select a file to un-erase. Type "dspkey.com" <C/R>. The monitor now checks if it is possible to un-erase the file. If successful, **File successfully UNERASED** is displayed. The displayed disc directory proves that you really got the file back.

Displaying a disc sector

The main purpose of the disc nurse is to display and change sectors on the disc. There are many ways of finding the sector that you want to examine:

a) Enter the track and sector number
b) Enter the block number
c) Enter the file name
d) Search for a unique character string

a) Press <T> to enter a track number. When you logged in the disc, the monitor examined it, and it now displays the possible track numbers that you can enter: **Track** number (0-039) Type "010". Press <S> to enter a sector number on track 10. All sector numbers are relative to the first sector number on the track. The first sector number is displayed in the field 1:st. For a PCW disc it is 01, for a CPC system disc 41. Enter "05".

You have now specified one specific sector on the disc, Track 10 Sector 5. The contents of this sector is not read into the monitor until you press <R>. It is therefore possible to read a sector, change the sector number, and write the sector to the new sector number. You can even copy sectors between discs.

Now, press <R> to read and display sector 05 on track 10. The first 256 bytes of the sector are now displayed. Press <DOWN-ARROW> to display the next 256 bytes in the sector.

b) Note how the block number automatically is updated when you enter a track or a sector number. The size of a block varies with the type of disc. CPC discs and PCW discs for drive A have blocksizes of 1k bytes, i.e. 2 sectors, while the blocksize for PCW drive B discs is 4 sectors or 2k bytes. The block numbers are entered as hexadecimal numbers. Press <B> and type "002", block number 2. With a PCW disc, this will translate into track 001 and sector 04. Block number 000 is always the first directory sector, which on a PCW disc start on track 001 sector 000. Type <B> "000" <R>. The first 8 directory entries are now displayed. Each entry consists of 32 bytes:

```
byte       contents
----       ---------

 0         User number    (E5 if  deleted)
                          (21 if time stamp entry)
1-8        File name      (Padded  with  blanks)
9-11       Extension name (One extent per 16k of the file)
12         Extent number
15         Record count
16-31      Block numbers
```

Be very careful not to change the information in the directory sectors, unless you have to "repair" files and you are confident that you know what you are doing.

c) Press <F> followed by "DSPKEY" <C/R>. The first sector of the source file DSPKEY is displayed. To follow the chain of sectors in a file, you should use the <RIGHT-ARROW> key rather than the <DOWN-ARROW> key. Press <SHIFT RIGHT-ARROW>. The first half of the next sector is displayed. Press <ALT RIGHT-ARROW>. The last half of the last sector of the file is displayed. Note that this part of the last block could be, and in this case is, outside the actual text file. The end of the file is indicated by a "1A" byte which you will find in the first half of the first sector of the last block. Press <LEFT ARROW> three times to get there.

d) The last way of finding a sector on disc is to search for a string of bytes contained in the sector.

Searching always starts from the current sector, so type <B> "000" to start from the beginning of the disc. Press <Q>.

```
QUERY/SEARCH (Use ?  as wild card)
Search (F)ile, (S)ector or (D)isc ?
(A)SCII  or (H)EX String ?
String:
```

Is now displayed. Enter <D> <A> "ALT" <C/R> to search the whole disc for the ASCII string "ALT". The monitor now reads sector by sector looking for the specified character string. Once it is found, that particular sector is displayed with the message:

```
Match found at byte 0A9 above
Space to continue, or STOP to Exit
```

The file we are interested in is "TEMP.COM". If the found sector not is part of TEMP.COM press <SPACE> until the correct sector is displayed. Press <STOP>

Changing a sector and writing to disc

You change bytes in a sector in the same way as you changed memory bytes in the monitor. Press <A> to change in ASCII. Move the cursor to the "P" in "Press" (Use <UP-ARROW> to wrap around to the last line). Type "Type ". Press <STOP> to exit the change mode. Press <W>. Enter <Y> in response to the prompt: Write to disc (Y/N)? N. The changed sector is now written to disc.

Changing the header of an Amsdos file (CPC only)

The header of an Amsdos file consists of the first 128 bytes of the file. Bytes 43 and 44 hex contains the checksum of bytes 00 to 42. If you change any of the first bytes in a header, the checksum bytes must be changed as well. PYRADEV can calculate the checksum for you. Press <ALT/H>.

```
Checksum of bytes 0 - 42 hex is 18D2 hex
Insert checksum (Y/N) ? N
```

As you are not going to write this sector to disc, there is no harm in changing its contents in memory. So, while you keep an eye on cells 043 and 044, press <Y>. The checksum has now replaced the old contents of these cells. If this had been the first sector of an Amsdos file. you would now have written the sector to disc, but not in this case.

Copy sector data to a disc file

It can sometimes be useful to copy parts of one or many sectors into a disc file for later disassembly by the monitor.

Press <Y>. You are now prompted for:

a) how many bytes from the sector you what to copy (default 200 hex = 512 = the whole sector)
b) what sector address the first byte is at (default 000)
c) Where in memory these bytes are to be copied to. (default the next available address. 6F00 hex)

Press <C/R> three times to accept the default values, i.e. to copy all 512 bytes of the sector to the next free address in memory. Press <C/R> again in response to the OK to copy (Y/N) ? Y prompt.

Press <SHIFT RIGHT-ARROW> to display the next sector. Type <Y>. Note how the memory address now is updated with the number of bytes you copied. Press <STOP> to abort this next copy.

Write Copied Memory to Disc

The disc sectors, or part of them, that you now have copied into memory can be written to a disc file. Press <V>.

```
Write MEMORY to DISC
From address: 6F00H to 70FFH
```

is displayed. These addresses have been adjusted depending on how much you have copied with the <Y> command. Press <C/R> twice to accept the defaults. You are now prompted for a file name. Type "TEMP2.TMP" <C/R>. The file is now created on disc.

Printing a sector

Type <B> "000" <R> to display the first directory sector. If you have a printer connected, press <P>. The whole sector will now be printed in hex and ASCII.

Extended directory

This feature is very useful when you want a printed catalogue of all files on your discs.

Press <X>. The screen clears and EXPANDED DIRECTORY is displayed together with the prompt Print the Directory ? N. Press <Y>. The next prompt: (N)ew page, (R)eset line count or (C)ontinue ? C lets you control the printer. Type <R> to reset the internal line count.

Now, type the title that you want to appear on top of the printed directory. Example: Disc 1A PYRADEV Test disc 29 Feb 1988

Press <C/R>. A sorted directory will now be displayed and printed. If the disc contains Amsdos files, additional information like file type, Start and End address, Length and Execution address will be printed.

If you want to print directories for more discs, you should type <Y> in response to the prompt Another Directory (Y/N) ? but for now, just type a <C/R> to get back to the Disc Nurse main screen.

As you have seen, the Disc Nurse is a very powerful tool, so please be very careful and always double check that you are doing your changes to the correct disc, the correct sector etc.

8.8 EDITOR

The final lesson will take you through a session with the editor. Press <E> to select the Editor. You are now prompted for the name of the file you want to edit. Press <STOP>. A sorted directory is now displayed. You should see DSPKEY among the files. Type "DSPKEY". Note how the name is copied to the output file name as you type. Press <C/R> twice. The first 24 lines of the file are displayed. Line 25 reads:

```
DSPKEY => DSPKEY          Recd:0001 Col:01 Free:40647,A  STOP for HELP
```

Help menu

Press <STOP>. The screen clears and a help menu is displayed. All special functions in the editor are reached with control keys (ALT keys). The help menu helps you to remember what control key to press to achieve what function. Example, to mark a line, press <ALT> and without releasing it press <L>. Press <STOP> to get back to the document.

Moving around the document

Move the blinking cursor around the screen, using the four cursor keys. Note how the Column and Record (Line) counter changes as you move the cursor. Press <RETURN>. The cursor moves to column 1 on the next line. Press <SHIFT DOWN-ARROW> five times. The cursor stays in the same position on the screen, while the document scrolls up. Hold the <DOWN ARROW> down for a few seconds. As the cursor reaches the bottom of the screen, the document starts to scroll up the screen, and you move ahead on the document. Press <ALT DOWN-ARROW>. The next 24 lines of the document are displayed. Press <ALT UP-ARROW>. The previous 24 lines are displayed.

In the following session you will "mess up" the document quite a bit, but don't worry. You are only doing these changes in memory. As long as you don't save the changed document to disc, with the <ALT/X> command, no harm is done.

GOTO the end of the document

Press <ALT-V> (View end of document, sorry). The last 24 lines of the document are displayed.

GOTO the Top of the document

Press <ALT-T>. The first 24 lines of the document are displayed.

GOTO a record

The assembler reports any found errors with a line number. It is therefore very useful to be able to go straight to a line (record) number. Press <ALT-G>."GOTO Record: **0** " is displayed on the last line. Type "57" <C/R>. 24 new lines are now displayed. The first line is the wanted one, line 57.

COPY a block of text

When you do block operations, copy, move or delete a block of text, you always start by pointing at the first and the last line of the block. Once this is done, that block is saved in memory outside the actual document. It is kept in memory until you either define the start of a new block or exit the editor to the PYRADEV main menu.

You are now going to copy the 4-line block surrounding the comment "
; START OF THE MAIN PROGRAM "

Press <DOWN-ARROW> three times. The cursor should now be on line 60. Press <ALT-B> to indicate the (B)eginning of the block. This is confirmed by the message **"BEG mrkr SET"**. Move the cursor to line 63 by pressing the <DOWN-ARROW> three times. Press <ALT-E> to mark the (E)nd of the block. A message **"END mrkr, Block Saved"** confirms the operation.

You are now going to make five copies of this block after line 68. Press <DOWN-ARROW> 5 times to move to line 68. Press <COPY> once. The 4-line block is now inserted between the line with the cursor and the next line. Press <COPY> another four times. Another four copies of the block are created. At this stage you could save your document, change to another disc, open a document on the new disc, press <COPY> and a copy of the original text block would be copied into the new document.

As you have noticed, it is very easy to copy blocks of text inside a file or between files.

MOVE a block of text

The difference between copying and moving a block of text is that when moving it, it is removed from its original position.

Goto line 50 by pressing <ALT-G> "50" <C/R>. Move the cursor to line 54. You will now move the first two lines, starting with "ESC:" and "LF:" a bit further down. Press <ALT-B> <DOWN-ARROW> <ALT-E>. The block that will be moved is now defined. Move to line 58. Press <ALT-COPY>. The block has moved to the new position.

DELETE a block of text

The final block operation is to delete a specified block of text. Let's delete all copies that you created in the COPY-block-session. Move the cursor to line 69. Press <ALT-B>. Move the cursor to the last line you want to delete, line 88. Press <ALT-E>. Press <ALT-D> to indicate that you want to delete the marked block. The block is now displayed in inverse video. You could now stop the deletion by pressing <STOP>, but you want to go ahead, so press <y> to confirm. The block is now deleted. Note however that as the block is still in memory it could be copied back, or copied anywhere, simply by pressing <COPY>.

Insert lines of text

Move the cursor to line 70. Let's insert two lines just after line 70. Press <ALT RIGHT-ARROW>. An empty line is created below line 70. Type "; THIS LINE IS INSERTED" <C/R>. Note that when you press the <C/R> another blank line is created. This is repeated until you either move the cursor to the last line and then press <DOWN-ARROW> once more, or until you delete the blank line. Type "; AND THIS ONE TOO" <C/R>. Stop the line-insert mode by pressing <ALT LEFT-ARROW>. The blank line is deleted and you are back to the normal edit mode. Press <C/R> to prove it. The cursor just moves to the next line without inserting a new one.

Deleting lines

Delete the next four lines by pressing <ALT LEFT-ARROW> four times.

Inserting characters

Move the cursor to the "M" in ";   DISPLAY A MESSAGE" Press <SHIFT RIGHT-ARROW> four times. You have now opened up some space. Type "NEW" <C/R>.

Deleting characters

Move the cursor to the "D" in ";   DISPLAY A MESSAGE " on the next line. Press <SHIFT LEFT-ARROW> three times. The characters "DIS" have been deleted.

Undelete changes

Sometimes, when you have started to make changes on a line, you change your mind and want to get the original line back again. Easy ! Without moving the cursor away from the line where you just deleted three characters, press <ALT-U> for (U)ndelete. The original line is back again. You can always get the original line back as long as the cursor has not left that particular line.

More on deleting characters

<SHIFT LEFT-ARROW> deletes characters to the right of the cursor. The same effect is achieved by pressing the left <DEL> key (CLR on the CPC). To delete characters to the left of the cursor, press the right <DEL> (DEL on the CPC).

Splitting a line

The main use of this feature is when you have a label on an instruction and want to insert another instruction at this label. Move to the "L" in "LD" on line 83. Press <ALT-O>. The rest of the line moves down one line. Type "EQU" <TAB> <TAB> "$".

Setting TABs

The TABs are normally set at every fifth column. It is easier to write source code files, like "DSPKEY" if you set the TABs at the beginning of each field, and no TABs in between.

Press <ALT-TAB>. The current TAB-setting is displayed on line 24. A TAB is represented by a lower case "t". To change the TABs, you move the cursor to the wanted column and press TAB. If that column already had a TAB, it is removed, else a new TAB is set.

Move the cursor to column 5. Press TAB to remove the existing TAB. Repeat for column 10, 20, 30, 35. Press <C/R> to accept the changes. Press TAB three times. Note how the cursor now moves straight to the operand field, the operator field and the comment field.

Mark and Find a Line

Press <ALT-L> to mark the current (L)ine. Press <ALT DOWN-ARROW> four times to get further down in the file. Press <ALT-F> to (F)ind the marked line. The line that you just marked is now displayed on top of the screen.

Insert/Overwrite mode

The editor is normally in Overwrite mode. This means that if you move the cursor over old text, that old text will be overwritten as you type new text. Move the cursor to the "M" in "MAIN LOOP" on line 78. Type "HEAD". "MAIN" has been replaced by "HEAD". Move the cursor to the space after the "A" in "A KEY STROKE" on line 81. Press <DEL> to remove the "A". Press <ALT-I> to switch to Insert mode. Note the "I" that appears after the disc drive indicator on the last line. Type "THE NEXT ". The rest of the line is pushed forward as you type. Press <ALT-I> to return to the Overwrite mode.

Search function

Let's look at the subroutine "DSPMSG". Press <ALT-S>.

Search For ......
Replace With ....

is displayed in the bottom half of the screen. Type "dspmsg" <C/R>. (The search function ignores the case of the characters). As you don't want to replace "DSPMSG" with another string, but just find it, press <C/R> again.

The editor finds the first occurrence of "DPSMSG" and the prompt

"RETURN, STOP, (G)oto or <Up or Down arrow> "

is displayed on the last line. This gives you the following choices:

<RETURN>	Ignore the found string and search for the next one
<STOP>	Abort the SEARCH function. Return to the original line.
<G>	Goto the found line. Stop the search.
<Up/Down Arrow>	Scroll the text around the found line.

Press <C/R> to find the next occurrence of "dspmsg". Once found press <C/R> again until the label "DSPMSG:" is found. Press <G>. Hint: Even if the assembler does not require that a label ends with a colon, always use it. You can then search for "LABEL:" and find it faster.

Let's change all occurrences of "DSPMSG" to "DSPMES". As the search function starts from the current line, and you want to change the whole document, press <ALT-T> to go to the top of the document. Press <ALT-S>. As the previous search string still is there, press <C/R> to get to the "Replace prompt". Type "DSPMES". (The case is important here) A new prompt Auto or Cond ? is displayed. If you would answer with a "C" for conditional, the editor would stop every time the string is found and ask if you want to replace it or continue. Let's do all of them in one go, so type "A". A few seconds later, End - File is displayed. Press <STOP> to get back to the document.

Press <ALT-DOWN ARROW> twice. You can now see that the old "DSPMSG" have been changed.

Keystroke Memory

The Keystroke Memories are used to save keystrokes and replay them just by pressing a single key.

You can save both control-keys and ordinary text.

In assembler code, a label is often followed by "EQU $" Let's save this sequence under function key "f5". Press <ALT RIGHT-ARROW> to open up a new line. Type "LABEL1:" Press <ENTER>. **"Select f1 thru f9"** is displayed. Press <f5>. Type "<TAB>EQU<TAB>$<TAB>" <ENTER>. The keystroke sequence is saved.

Press <C/R> to get to the next empty line. Type "LABEL2:" <f5>. Note how the saved keystrokes are replayed and displayed.

A big advantage with keystroke memories is that they can call themselves to create a loop. Let's move the comment field on a lot of lines another 5 columns to the right. Move the cursor to the ";" in a comment field, (column 40). Press <ENTER> <f1> <SHIFT RIGHT ARROW> <SHIFT RIGHT ARROW> <SHIFT RIGHT ARROW> <SHIFT RIGHT ARROW> <SHIFT RIGHT ARROW> <SHIFT DOWN ARROW> <f1> <ENTER>.

To start the replay, press <f1>. Press <STOP> when you want to stop the replay.

COMMAND mode

The COMMAND mode lets you display disc directories, delete and rename files without leaving the Editor. You can also load and save option files, set or reset the file compression mode and finally change foreground and background colours.

Press <ALT-Z>. A new menu is displayed at the bottom of the screen:

```
A,B,M: Select Drive         C: Compression
D,R  : Delete, Rename       Z: ZAP..BAKups
L,S  : Load, Save Options   K: Colour
```

Press "A". The disc directory for drive A is displayed. Press "C". A message "OFF" is displayed. This indicates that if the file now is saved, it would be saved as a standard ASCII file. Press "C" again. Compression is on again. As long as you use the files with the editor and the assembler, use the compression, as it makes the files take up much less space on disc.

Press "K". The foreground colour and the background colour switches.
Press "K" again to switch back.

The "Z" is useful if you have tried to save a file, but it failed because the disc was full. Pressing "Z" will delete all files ending with " .BAK", and free up space on the disc.

The TAB setting and the Keystroke Memories that you have created during an edit session, can be saved on disc. This option-file can then be loaded the next time you use the editor, and your TAB settings and keystroke memories will then come into effect without you having to retype them.

Press <S>. Type "OPT1" <C/R> in response to the "File-name" prompt. The option file is now saved on disc. To load it back in, you would just press <L> and type "OPT1" <C/R>.

Exit the Command mode by pressing <STOP>.

You would normally save your changed document by pressing <ALT-X> <C/R>, but in this case, abort from the document by pressing <ALT-A>. Confirm that you want to exit without saving your changes by pressing 'Y' in response to the prompt '(Y) TO ABORT'

This concludes the tutorial. You should now have a pretty good idea of most of the features in PYRADEV, and should be able to continue on your own.

Thank you for buying PYRADEV. We hope that you will enjoy using it.

GOOD LUCK

SECTION 9: CONTROL KEY SUMMARY

EDITOR (edit mode)

(On the CPC, ALT is the CTRL key)

ALT-A: Abort and Exit	ALT-I: Insert Mode	ALT-T: Top Of Document
ALT-B: Begin Pointer	ALT-L: Line Marker	ALT-U: Undo Changes
ALT-D: Delete Block	ALT-M: Merge File	ALT-V: View End of Doc
ALT-E: End Pointer	ALT-O: Open Up Text	ALT-X: Save and Exit
ALT-F: Find Marker	ALT-P: Print the Doc	ALT-Z: Command Mode
ALT-G: Goto Line nnn	ALT-S: Search System	ALT-TAB: Tab Settings

COPY: Copy Block ALT-COPY: Move Block

<f0>:	Display Keystroke Memories
<f1> - <f9>:	Replay Keystroke Memories
<ENTER>:	Start/Stop Keystroke Recording
<ALT/ENTER>:	Erase Keystroke Memories

On the PCW, <f0> is the <RELAY 0> key
<f9> is the <SHIFT/RELAY> key

ESCAPE: Help-Display

EDITOR (command mode) (ALT/Z)

A: Set Drive A	D: Delete File	L: Load Options	K: Colour change
B: Set Drive B	R: Rename File	S: Save Options	
M: Set Drive M			

C: Compression OFF/ON Z: ZAP Backups

ASSEMBLER (run time toggle keys)

V: Video Output P: Print Listing S: Selective Print E: Error Printing

MONITOR (main menu)

T: Set Trap Point	L: Load Test File	P: Display from PC
Z: Clear Trap Point	K: Load Symbol Table	ALT-P: Print Memory
R: Change Registers	W: Write Disc File	ALT-B: Bank Select
M: Modify using HEX	ALT-K: Symbols on/off	Q: Query / Search
ALT-A: Modify ASCII	O: Other Display	ALT-D: Dis-Assembly
G: Goto an Address	U: Update Screen	ALT-X: Exit Monitor
ALT-G: Set Step Addr.	N: Refresh Screen	
S: Exec Single Step	ALT-R: Reset Screen	
ALT-S: Double Step	ALT-C: Catalogue	

DISC NURSE (main menu)

D: Select Disc Drive	R: Read Sector	C: Catalogue Display
T: Set Track Value	W: Write Sector	E: Display Erased Files
S: Set Sector Value	P: Print Sector	X: Extended Directory
G: Set User (Group)	Q: Query / Search	U: Unerase File
B: Set Block	M: Modify in HEX	Y: Copy Sector Data
F: Select File	A: Modify in ASCII	V: Write to File
H: Header Checksum	I: Displ Track ID	ALT-X: Exit Disc Nurse

UTILITIES (main menu)

A: Display Drive A	00-15: User	X: Exit Program
B: Display Drive B	D: Delete File	C: Copy File
M: Display Drive M	R: Rename File	

UTILITIES (copy menu)

1: Disc to Disc	(Optional ASCII Compression.)
2: Disc to Tape (CPC only)	(Protection Remove.)
3: Tape to Disc (CPC only)	(Always displays header information.)

SECTION 10: AMSDOS MONITOR

The AMSDOS Monitor is only relevant for the CPC6128.

The standard Monitor is designed for testing and debugging CP/M programs. The AMSDOS Monitor makes it possible to debug your programs that run under the AMStrad Disc Operating System, AMSDOS.

The AMSDOS Monitor does not handle symbolic labels or conditional breakpoints, but is otherwise almost identical to the CP/M monitor (Section 4), and its functions will therefore not be described here.

The Monitor, PYRAMON.BIN, is started from AMSDOS by typing "RUN PYRAMON <C/R>".

10.1 RELOCATION

The Monitor is fully relocatable, and will load itself into a high memory address (which it displays). It will then ask if relocation is required:

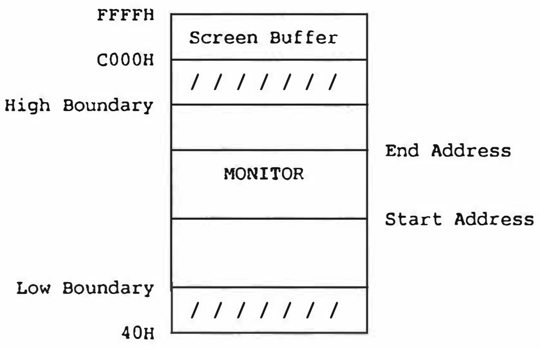
```
Monitor loaded between 6000 and EFAA hex
      Do you want to relocate ?
```

Answering "y" gives you a choice between the standard Monitor, or a smaller 'Mini Monitor'. The smaller one could be useful if you are short of space in memory. Once this choice is made, a new prompt is displayed:

```
Do you want to specify Low (Start) boundary ?
```

Specifying a Start Address instructs the Monitor to relocate itself such that it's lowest address does not go below this start address.

Specifying a End Address (by answering "N" to the prompt) instructs the Monitor to relocate itself such that it's highest address does not exceed the End Address.



The following functions are not included in the CP/M monitor:

10.2 (X)..EXAMINE ROMS

Both the dis-assembly and memory displays read memory according to the LOWER and UPPER ROM selections. Pressing the (X) key allows the status of the ROM selections to be altered, ie the monitor reads either from the ROM or from the RAM.

This function also let you change the UPPER rom.

10.2 (J,K) PAPER and PEN COLOURS

Use of the (J) and (K) keys will step the paper and pen colours through to their next ink values. CTRL-J and CTRL-K can be used to step through the ink colours in a reverse fashion.

10.3 (CTRL-T)..TAPE/DISC SELECTION

Press (CTRL-T) to toggle TAPE or DISC operation. The selected device type is shown at the top of the screen and will be used to READ and WRITE files.