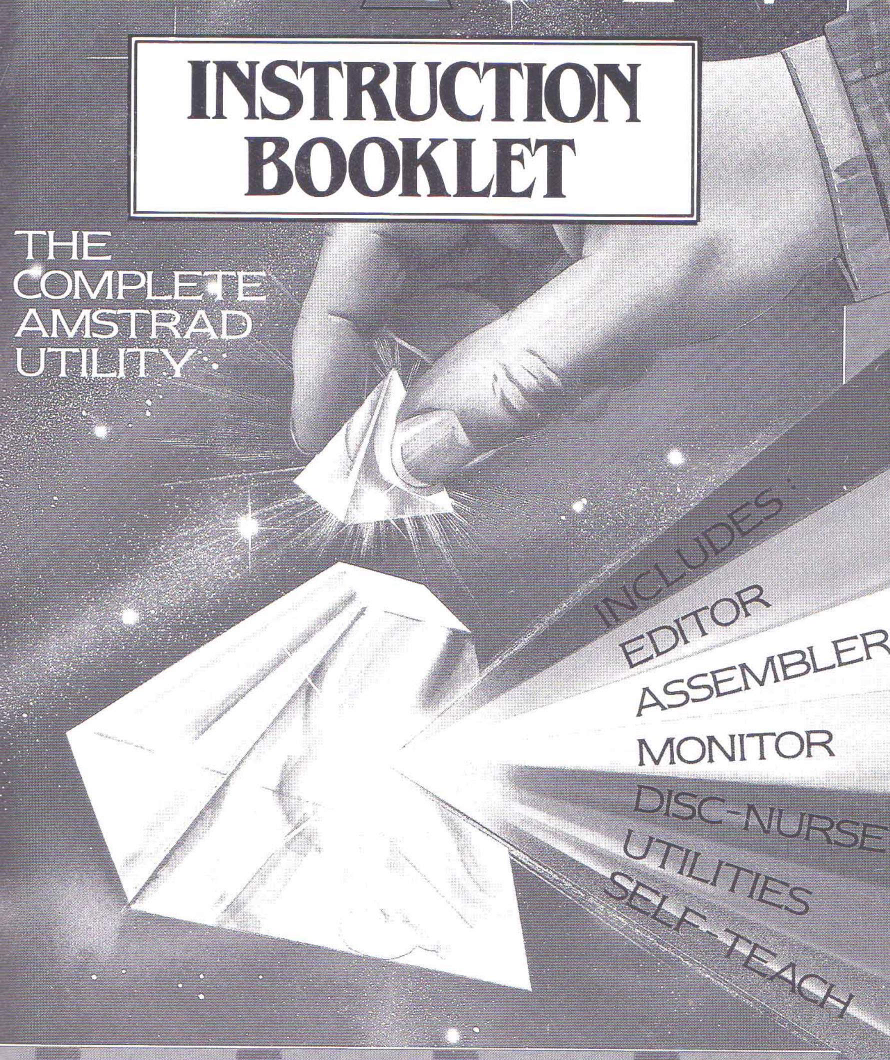
P-Y-R-A-D-E-V

INSTRUCTION BOOKLET

THE COMPLETE AMSTRAD UTILITY:



INCLUDES:
EDITOR
ASSEMBLER
MONITOR
DISC-NURSE
UTILITIES
SELF-TEACH

AMSTRAD CPC 464/664/6128

WHEN YOU HAVE MADE THE
DISCOVERY
NOTHING ELSE COMES CLOSE

USER MANUAL **TABLE OF CONTENTS**

1.0 INTRODUCTION.

PYRADEV is a set of five programs which collectively provide a secure and comfortable environment for the development of AMSDOS and CPM software on the AMSTRAD 464, 664 and 6128 micro computers using the DD1 disc system. The system is AMSDOS based and does not require extra ROM or RAM. It will support large programming projects on single or double drive systems and consists of the following programs:–

- A very fast full screen SOURCE FILE EDITOR.
- A multi-file Z80 MACRO ASSEMBLER/Linker.
- A powerful debug MONITOR and DIS-ASSEMBLER.
- A friendly and easy to use DISC-NURSE.
- A comprehensive FILE MANAGEMENT utility.

The EDITOR uses its own special disc read and write routines enabling large files (up to 32,300 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 (if two drives are used) allowing upto 992K of source code (about 5 discs) to be processed into a single binary file. Video or Print output can be toggled on/off as required. Symbol and Cross-Reference maps may also be produced. The Assembler is very fast, processing up to 40,000 source code characters a minute on 64K systems. On larger systems, the additional RAM is fully utilised for increased performance.

The self-relocating MONITOR will load, trap, single step, double step, modify and write code back to disc. In addition, ROMS can be selected and studied. During debug mode registers can be modified, and screen prints can be taken at any time. Code dis-assembly can be output to a printer or ASCII files.

The DISC-NURSE handles system, data and IBM style discs. File sectors can be searched, viewed, printed, modified and re-written. The search can be extended to the whole disc if required. Whenever a sector is read the file-owner is displayed. Previously deleted files can be re-claimed if subsequent 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 UTILITY (file-manager) program provides directory display, erase, rename, copy to/from discs and tape 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 AMSDOS files.

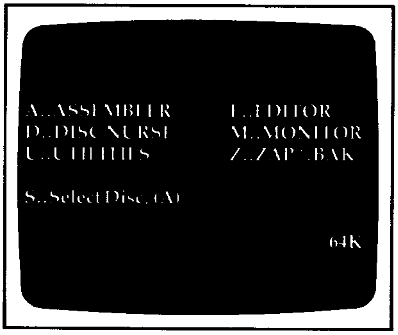
1.1 SETTING UP.

PYRADEV is supplied on a MASTER DISC which must never be written to. It should only be used to make WORKING DISC copies with the CPM utility DISCCOPY or DISCKIT3. Refer to the appropriate manual and use the PYRADEV MASTER DISC as the Source Disc to create a WORKING (destination) Disc.

After the copy is complete, remove the PYRADEV MASTER DISC and store it away somewhere safe.

1.2 GETTING STARTED.

Place the WORKING DISC in drive A and clear the machine by pressing CTRL-SHIFT-ESCAPE. Enter RUN"PYRADEV and press the large enter key. The SYSTEM-MENU will appear. The amount of memory present is displayed on the bottom right of the screen.



```
A..ASSEMBLER        E..EDITOR
D..DISC NURSE       M..MONITOR
U..UTILITIES        Z..ZAP *.BAK

S..Select Disc (A)

                                 64K
```

Figure 1.0 System-Menu

The system programs are selected by pressing the appropriate single key «A, E, D, M, U». They all return to the above menu upon 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) or (B). The PYRADEV disc must always be in drive (A) as it expects to find its programs there, regardless of the disc default. The «S» option should NOT be used on single drive systems.

1.3 NEW USERS.

When the SYSTEM MENU appears, press «E» to select the Editor. Enter the name PROGRAM.001 as your input file and press the space bar twice. The file will be read into memory. Browse through the file with the UP and DOWN arrow keys. The file is a small program with lots of comments and examples about using the Editor and Assembler.

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 «D»elete command can be used to remove files. This list describes the system files.

PYRADEV.BIN	SYSTEM-MENU, required on ALL Working Discs.
PYRAMED.BIN	Full Screen Editor for Source Program Editing.
PYRAMAS.BIN	Z80 Multi-File Macro Assembler.
PYRAMON.BIN	Test Monitor and Dis-Assembler.

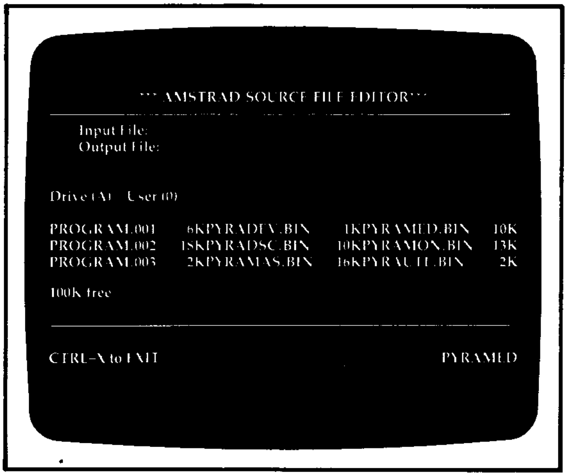
PYRADSC.BIN	Disc Nurse for Modifying Sectors.
PYRAUTL.BIN	File Utilities for General Copying.
PROGRAM.001	Self-Teach Demonstration Source Program.
PROGRAM.002	Assembler Code Examples and Test Files.
PROGRAM.003	Simple ASCII file copier program.

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

SYSTEM RESTRICTIONS.
PYRADEV uses locations 40 hex through to 4F hex to pass variables amongst its programs. Please avoid using these areas during program development and testing.
When started, PYRADEV will initialise all background ROMS by calling the KL.ROM.WALK routine. It then sets its own HIMEM value based on the returned HL value. Ideally, only the DISC.ROM should be connected, however, other ADD-ONS should not cause problems, unless they use large areas of RAM. The Editor work area is normally about 32,300 bytes.
When using the MONITOR, be aware that all background ROMS connected to the AMSTRAD have been initialised. Initialising a ROM twice may cause un-desirable effects.

2.0 SOURCE FILE EDITOR.

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 display the default disc directory, and then wait for Input and Output file names to be entered. The Editor always returns to the Idle Screen after the edit session is complete.



```
              *** AMSTRAD SOURCE FILE EDITOR***
-------------------------------------------------------------
     Input File:
     Output File:


Drive (A)   User (0)

PROGRAM.001     6KPYRADEV.BIN     1KPYRAMED.BIN    10K
PROGRAM.002    18KPYRADSC.BIN    10KPYRAMON.BIN    13K
PROGRAM.003     2KPYRAMAS.BIN    16KPYRAUTL.BIN     2K

100K free

-------------------------------------------------------------

CTRL-X to EXIT                                  PYRAMED
```

Figure 2.00 Idle Screen

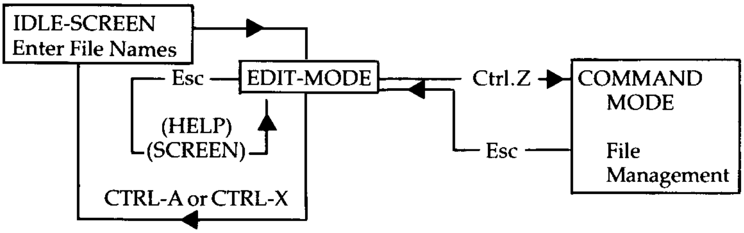
○ To VIEW a file;	Just enter an INPUT-FILE name.
○ To CREATE a file;	Just enter an OUTPUT-FILE name.
○ To MODIFY a file;	Enter INPUT and OUTPUT file names.
○ To DISPLAY directory;	Press SPACE SPACE (ie no names).

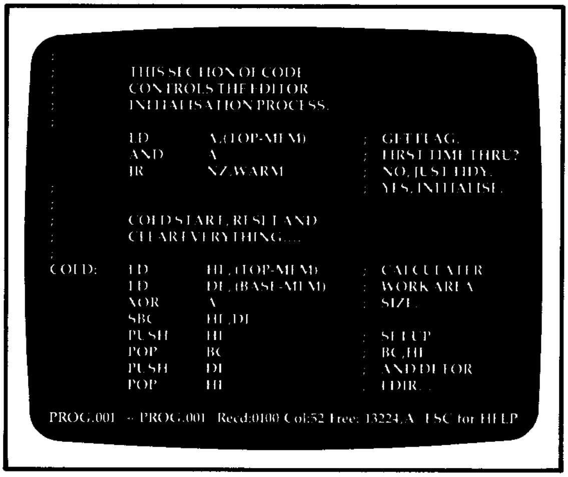
The Input file name can be prefixed with A: or B: to set a new disc drive default.

2.1 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 ESCAPE key or CONTROL-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)      ;  CALCULATER
          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: 13224,A  ESC 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.

ESCAPE KEY.
When used in Edit-Mode, the escape 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 escape key a second time returns the screen to Edit-Mode.

ABORT & EXIT:
The (CTRL-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 and displays the Disc Directory.

SAVE & EXIT:
The (CTRL-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 Editor returns to the Idle Screen.

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

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

CURSOR-SPEED.
The cursor speed can be toggled between fast and slow by using the (CTRL-Y) key. Each time it is changed, the prompt (Flash?) will appear. Answer (Y) if you require a flashing cursor, or (N) for a non-flashing cursor.

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 CTRL-DOWN-ARROW and CTRL-UP-ARROW will scroll forwards and backwards 24 lines at a time. These four functions will not scroll past the beginning or end of the file.

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 (CLR) and (DEL) keys also provide character delete and reverse delete capabilities (as they do in BASIC).

LINE-INSERT-DELETE:
The CTRL-RIGHT-ARROW and CTRL-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 ENTER key is pressed. Line Insert Mode is cancelled by the use of CTRL-LEFT-ARROW (to delete a newly created blank line) or by scrolling down past row 24.

OPEN SPACE.
The (CTRL-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.

BLOCK COPY:
To copy a block of code, move the cursor to the first line of the code and press (CTRL-B) to set the BEGIN marker. Move the cursor to the last line of the code and press (CTRL-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.

BLOCK-DELETE:
Mark the Block with the (CTRL-B) and (CTRL-E) keys as for the Block Copy function. When the 'Block Saved' message appears, press the (CTRL-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.

BLOCK MOVE:
This is the BLOCK-DELETE function followed by use of the (COPY) key as described above.

TAB STOPS:
Pressing (CTRL-T) will display the current tab settings as small triangles on line 24. Tabs are set or cleared by positioning the cursor and pressing the (TAB) key until the triangle symbol appears or disappears. Pressing «CTRL-TAB» will clear all tab stops. When the new tab stops are ready, press the large Enter key to record them. Tab marks can be saved for future use, see Command Mode.

MARK-FIND-LINE:
A single LINE can be marked by using the CTRL-L key. The line can be brought back to the screen with the CTRL-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.

VIEW-BEGIN-END:
Press CTRL-V to select 'View' then press (B) to see the beginning of the file (record-1) or (E) to see the end of file (last record).

GO TO RECORD:
Pressing CTRL-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 CTRL-G function is helpful in locating program code from the Assembler Listing.

NEW-INPUT:
Use of the CTRL-N function allows another INPUT file to be appended to the current memory workspace 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 the 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.

SEARCH SYSTEM:
CTRL-S selects the search system and two types of operation are provided.

Search and Stop.
Enter a search string in single or double quotes then press the enter 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 preceeding and following record in its own small 'window'. The following options can then be used:–

○ Enter Key	Ignore this match and continue searching.
○ Escape Key	Terminate, and return to original text.
○ (G) Key	Goto the record where the match was found.
○ 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 in single or double quotes. Then enter a replace string in single or double quotes. 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 preceeding 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 CTRL-V B 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 CTRL-L CTRL-V B. This marks the current line and goes to the beginning of the file. After using the search function CTRL-F will return to the marked line.

Any search can be cancelled with the ESCAPE key.

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

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 ESCAPE key to return to Edit-Mode.

Record:
Press the SMALL-ENTER key then select a memory using the island keys 1 thru 9. Once selected the message 'Keystroke Recording ON' will be displayed. 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 numeric island key 1 thru 9 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 ESCAPE key.

Erase:
Press CTRL-SMALL-ENTER and select a memory with the numeric island keys 1 thru 9. 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. See Command Mode.

PRINTING:
Pressing the decimal point key on the numeric island 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 ESCAPE key once pauses the print which can be continued with any key. Pressing the ESCAPE key twice cancels the print activity.

Special print codes may be embedded in the text being printed in order to control the printer as follows:–

During printing, the up-arrow symbol (on the pound sign key) indicates that the following character is a control symbol. The value 33 is subtracted from 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,
⌂ represents code 15, ⌂/ represents code 14.
⌂∅ represents code 15.

COMMAND MODE.
Command mode is selected with the CTRL-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 Escape 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: Select Disk	C: Compression
D,R: Delete, Rename	Z: Zap BAKups
L,S: Load Options	S: Save Options
J,K: Paper Pen Inks	

A,B: Select Disc A or B.
Pressing (A) or (B) 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 BASIC files, before using CTRL-X to write the file back to the disc you should turn compression OFF. The file will be expanded to normal and can then be used by the BASIC system.

D: Delete File.
This is similar to the BASIC/CPM 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.

J: Jump PAPER Ink.
Each time the (J) key is pressed the Paper Ink will be advanced to its next value. Pressing (CTRL-J) does the opposite.

K: Jump PEN Ink.
Each time the (K) key is pressed the Pen Ink will be advanced to its next value. Pressing (CTRL-K) does the opposite.

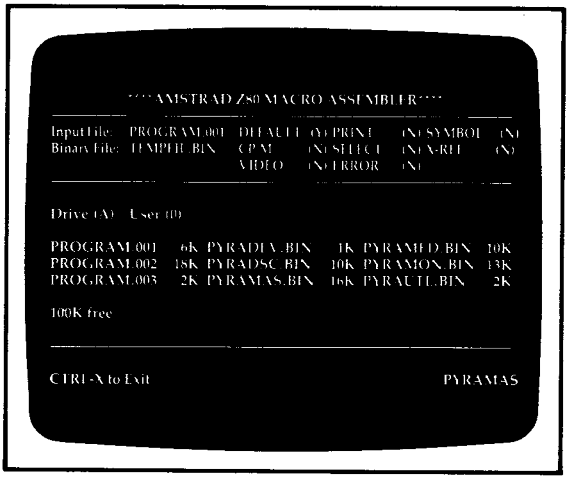
S: Save User Options.
Pressing (S) causes the current Tabs, Paper and Pen Ink Values, and Keystroke Memories to be written to a disc file for future use. A file-name must be entered and can be any name. Suggested file extension (suffix) is -OPT. Several different Option Files can be used to define different '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, Inks, 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.

3.0 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 a binary load file, directly on disc. It is selected from the SYSTEM MENU by pressing (A). Once loaded it displays the default disc directory and waits as the various RUN OPTIONS are entered....



```
               ****AMSTRAD Z80 MACRO ASSEMBLER****
Input File:  PROGRAM.001  DEFAULT  (Y) PRINT    (N) SYMBOL   (N)
Binary File: TEMPFIL.BIN  CP/M     (N) SELECT   (N) X-REF    (N)
                          VIDEO    (N) ERROR    (N)

Drive (A)  User (0)

PROGRAM.001   6K  PYRADEV.BIN   1K  PYRAMED.BIN  10K
PROGRAM.002  18K  PYRADSC.BIN  10K  PYRAMON.BIN  13K
PROGRAM.003   2K  PYRAMAS.BIN  16K  PYRAUTL.BIN   2K

100K free

CTRL-X to Exit                                    PYRAMAS
```

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 .BIN 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 catalogued on disc. It is built from the input file name (by adding a .BIN suffix) and may be overkeyed. Pressing the ENTER key leaves it as it is. If no binary file is required, this field should be set to blanks.

DEFAULTS.
If the Enter 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.

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	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

Figure 3.1 Print Options

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 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 preceeded by a (U), (D) or (N) denoting Undefined, Doubly-Defined or Not-Referenced.

CROSS-REFERENCE.
If set to (Y) a 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.

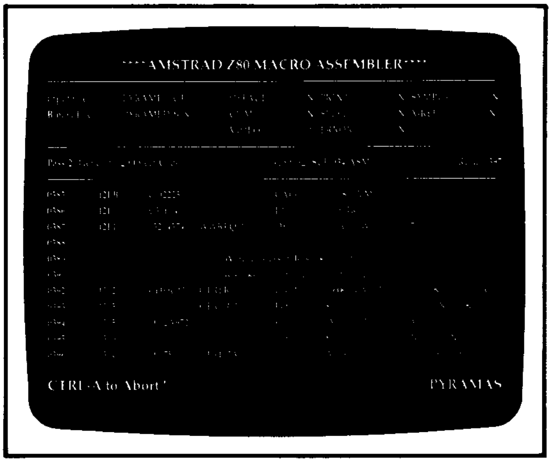
ASSEMBLER RUN TIME KEYS.
Once started, the Assembly process can be controlled by 'toggling' the RUN OPTIONs described above. This is done by pressing single keys. The screen run options display will change as the keys are used.

The following keys may be used:–

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

In addition the Escape key may be used to PAUSE the Assembly. This is useful in order to study object code being displayed on the screen. Press any key to resume the assembly process.

Pressing the CTRL-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****

[illegible]

CTRL-A to Abort'                                    PYRAMAS
```

Figure 3.1 Run Time (VIDEO-ON)

ASSEMBLY 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.

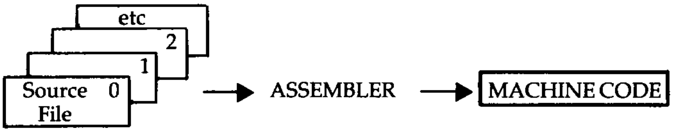
b) If the VIDEO option is set to (Y)es, the error code and the description will be displayed. The assembly process will pause, and can be resumed by pressing any key.

c) If the PRINT, ERROR and VIDEO options are all set to (N)o, the assembly process will be running at its 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 resume the assembly process.

Error Codes.

B	– Branch/Jump Error	N	– Numeric Expression
C	– Conditional Error	OE	– Operand(s) Error
D	– Doubly Defined	OM	– Operand Missing
F	– File Error	SF	– Too many SELECTs
G	– Org Usage Error	U	– Undefined Label
L	– Label (syntax)	X	– Extra Opcode Invalid.
MP	– Macro Parameter	FW	– Forward Ref. Invalid.

ASSEMBLER 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 values
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 preceeded 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 a 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.

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, 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	#5C	(must start numeric)
				(ie OFFH for 255D)

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.

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. Its 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   ; This coding example shows
; area is reserved at the front of   ; how endbin is used at the
; of a program.                      ; end of a program.
;                                    ;
             ORG      50H                       POP  BC
             ENDBIN                             POP  DE
;                                               RET
BUFF-1:      DEFS     2048           ;
BUFF-2:      DEFS     2048                      ENDBIN
;                                    ;
TRUE-START:  ORG      $              BUFF-1:  DS  2048
             LD       (SAVE),SP      BUFF2:  DS  2048
             CALL     INIT-PGM       ;
             JP       GO-GAME        ;
;                                                END
             ETC.......
```

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

EXEC
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 page 14.

FREE
Enables the Assembler to process 'Free Format' expressions. Allows mixing of different types of storage expressions. See page 15.

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. Its 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 its 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. The first occurence 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 programmer's 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. Its 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' (assumes output drive A).

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, L-Feeds.
PRINTR W132,D66,P	Width 132 chars. Depth 66 lines, Page-Pause On, LF's
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 statements may 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   ; ;
```

On large programming projects, the following information may be useful, assuming you are using twin disc drives.

	DRIVE-A		DRIVE-B
Disc 1 Side A	PYRADEV SYSTEM + Binary Files + Source Files.	Disc 2 Side A	SOURCE CODE – 178K, accessed by Editor and Assembly System
Disc 1 Side B	Spare, suggest you use PYRA-WORD to document project	Disc 2 Side B	SOURCE CODE –178K, accessed by Editor and Assembly System
	Disc 1 stays fixed, during development		Disc 2 is reversed as / when required to edit / assemble source files.

TITLE
Change to the Top of Form Assembly Listing Header Message, example:–

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

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 show 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                 FLAG:       EQU     0
;                                     ;
            IF      FLAG                          IF      FLAG
THEN-CODE:  LD      A,(VALUE-1)       THEN-CODE:  LD      A,(VALUE-1)
            ELSE                                  ELSE
ELSE-CODE:  LD      A,(VALUE-2)       ELSE-CODE:  LD      A,(VALUE-2)
            ENDIF                                 ENDIF
;                                     ;
            END                                   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.

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.

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          JL,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.

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 (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.

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 (!) preceeds 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 control 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 !.

MEMORY MANAGEMENT.

During the assembly process, memory allocation for the symbol table, source code storage, cross reference table and macro definitions varies according to available memory.

On the Amstrad 464 and 664 machines which both have 64K of memory, the symbol table is stored from the bottom of the memory upwards, whilst the optional cross-reference table is stored from the top downwards. In addition a 2K buffer is used to read source code. If you assemble very large programs, at some stage it will not be possible to obtain a cross-reference list as the 64K system will be fully utilised.

On the Amstrad 6128 (or a 464/664 using a RAM addon kit*) the following memory allocation scheme is used:–

- The source code is stored in the extra banks in pass 1, enabling pass 2 to read source from memory, speeding up the assembly process.

- The optional cross-reference table is held in 32K in the second bank of RAM if selected, allowing upto 6,500 entries.
  This permits very large assemblies.

* The PYRADEV system has been extensively tested using a CPC464 with a DD1 disc drive and CPC6128. Memory sizes from 64K upto a total of 384K have been used by using the DK 'Tronics 256K Ram Expansion Kit.

Additional RAM above 128K only provides a marginal improvement (about 10 – 15%) in overall assembly time.

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
```

SUMMARY OF ASSEMBLER DIRECTIVES

COND	«exp»	Conditional assembly.
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.
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»	Equates a label.
EXEC	«exp»	Define EXEC address.
EXTRA		Enable extra opcodes.
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.
ORG	«exp»	Define object code address.
PRINTR	«parms»	Define printer options.
PAUSE	message	Pause Assembly.
SELECT	«filename»	Append another file.
TITLE	message	Change Listing Header.

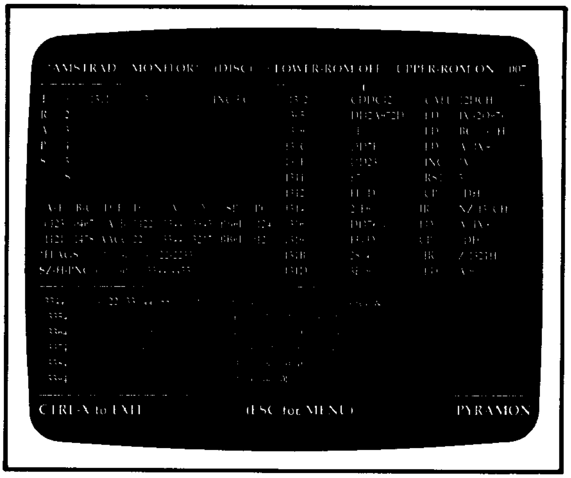
DB	same as DEFB
DL	same as DEFL
DM	same as DEFM
DS	same as DEFS
DW	same as DEFW

4.0 MONITOR.

The PYRADEV Monitor is a powerful debug monitor which provides all the features necessary to drive sub-programs and routines to perfection before adding them into a major system program. 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 ROMS, writing the dis-assembled source code to an ASCII file if required.

WARNING: If you have not used a Monitor before, please practice using it on very simple routines (such as PROGRAM.001). Do not start using it immediately on code which modifies the screen area for example as it will be difficult to understand.



```
*AMSTRAD MONITOR* (DISC)  LOWER-ROM OFF  UPPER-ROM ON  [illegible]
[illegible]
CTRL-X to EXIT              (ESC for MENU)                PYRAMON
```

Figure 4.0   Debug Monitor

NOTE: Your test code should not initialise background ROMS. These are already active (via KL.ROM.WALK) which PYRADEV calls at startup time.

SCREEN DISPLAY.
The Monitor Screen is split into five 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 very top right of the screen shows the status of the lower and upper ROM sections.

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

RELOCATION.
The Monitor is fully relocatable, and will load itself initially into a high memory address (which it displays). It will then ask if relocation is required. If you reply (Y)es, you may then select the (S)tandard Monitor or (M)ini Monitor.

The Mini-Monitor is sub-set of the Standard-Monitor which can be used when memory space is restricted. The following functions are excluded from the Mini-Monitor: Write-File-to-Disc, Copy-Memory, Help-Menu, Ink-Changes, ROM-Display Screen-Printing and Dis-Assembly to Disc or Printer.

The Monitor will then ask whether you wish to specify a Low Boundary (start address) or High Boundary (end address).

Specifying a Low boundary instructs the Monitor to relocate itself such that its lowest address does not go below the Low-Boundary.

Specifying a High boundary instructs the Monitor to relocate itself such that its highest address does not exceed the High-Boundary.

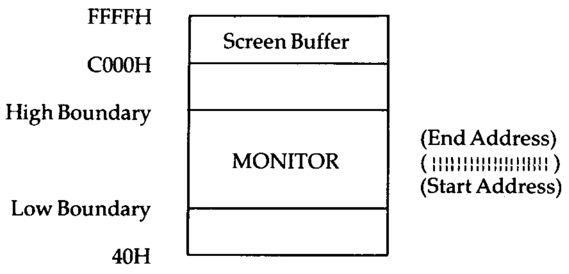


Figure 4.1

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-ADDR L...LOAD     Q...QUERY J   J⌂..PAPER     C⌂...CAT
Z.CLR-TRAP     G⌂..STEP-1ST  W..WRITE    N.NEW-SCN     KK⌂....PEN    T⌂A...TAPE
M.A⌂MEMORY S..SSTEP(f1)      O..OTHER    X....ROMS     D⌂..DIS-ASM   «.»..PRINT
R.....REGS     S⌂.DSTEP(f2)  U.UPDATE    Y....COPY     X⌂.....EXIT
```

Figure 4.2 Monitor Menu

DEVICE 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. Press (CTRL-C) to see the device CATALOGUE DISPLAY.

LOADING TEST CODE.

Press (L) to load a program file and enter the file-name. The file header will be examined in order to display the load-point and execution address will be displayed. You can alter the load point before pressing the enter key to read the file into memory if you wish.

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.

CHANGING CODE and MEMORY.

Use the ESC key until the memory display appears on the bottom of the screen. Use the (O) key until the Right-Arrow symbol appears 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 (CTRL-A) to modify memory using ASCII. Press the ESC key after all changes have been made. If you have made code changes, press (O) to move the cursor back to the top right display, enter the address where the changes were made and the new code will be dis-assembled onto the screen.

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 control states as required.

SETTING TRAPS.
Use the top right display to study code, then press (T) to set a trap. 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 – Start single-step code execution from the trap address.
G – Goto address, the default one continues execution.

When a trap occurs it automatically clears itself. If you want the trap to occur again, you should re-install the trap before continuing.

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

STEPPING THROUGH CODE.
You can step through code after a trap occurs, or by setting the first trap point with (CTRL-G).

Press (S) or (f1) to SINGLE-STEP through code. This form of stepping will follow the logical address path (control flow) as the code executes. If a CALL to ROM code occurs, the step operation will discontinue since the step mechanism cannot over-write ROM hardware in order to set the necessary traps.

Press (CTRL-S) for (f2) to DOUBLE-STEP. This form of stepping will set traps in ascending memory locations. This means that when single stepping, if a CALL occurs to a ROM routine, use a double step. This will set a trap on the next memory instruction so that stepping (single or double) can be resumed after the CALL operation is complete and control returns from the ROM routine. AMSTRAD programs normally use many ROM calls to effect system operation.

The MODE (0,1,2) your program uses can be altered using shift 0,1, or 2. When set to 2, it is the same as the MONITOR and so the display screen will NOT be cleared each time a trap or step is executed.

CHANGING REGISTER CONTENTS.
Press (R) to change register contents, then carefully overtype the existing values with the new ones. Press the ESCAPE 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.

NEW and UPDATE.
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.

QUERY, SEARCH.
Pressing (Q) selects query mode and allows 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.

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.

The display areas will change if they are displaying memory areas below 16K or above 48K as the lower and upper ROMS are switched.

DIS-ASSEMBLE TO DISC, VIDEO OR PRINTER.
Press CTRL-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 the 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 30K of source code has been generated so that subsequent file names can be entered before the process resumes.

WRITE.
To write a section of memory as a binary file press (W). The load point, execution address and length (saved when the file was loaded) are re-displayed and can be overtyped if required. After entering a file-name the relevant section of memory is written to disc as a binary file.

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 inital block area.

MEMORY BANK-BLOCK SELECTION.
Press (CTRL-B) to select an alternate block of memory to be mapped into the address range 4000H to 8000H. (128K and larger systems only).

SCREEN-PRINT.
Press the (.) key on the numeric (function) island. The contents of the screen will be decoded and printed. Press the ESC key twice to cancel printing.

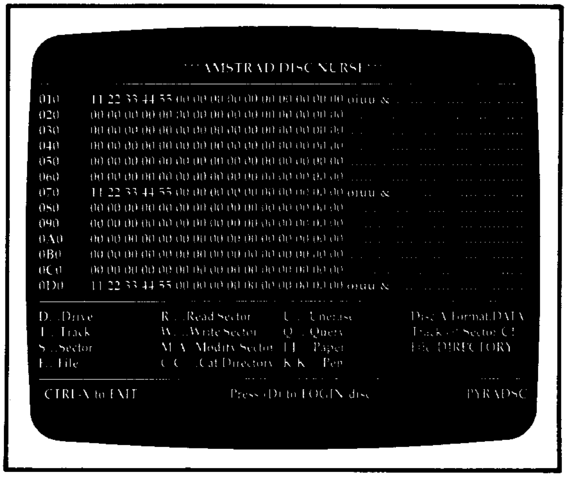
PAPER and PEN COLOURS.
Use of the (J) and (K) keys will step the paper and pen colours through their next ink values. Experimenting with the inks will improve screen readability on both mono-chrome and colour monitors according to background lighting conditions. CTRL-J and CTRL-K can be used to step through the ink colours in a reverse fashion.

STACK ALLOCATION.
If the Monitor is loaded below 4100H, it allocates a stack area using the current HIMEM value. If the Monitor is loaded above 4100H the stack is allocated just below the load-point used.

The catalogue (CTRL-C) function uses a 2K buffer located 2048 + 200 bytes below the stack.

EXTERNAL ACCESS.
The first instruction in the monitor is JP nnnn. This can be used to jump into the monitor from your own program if required.

5.0 THE DISC-NURSE.
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.



```
                          *** AMSTRAD DISC NURSE ***
010     11 22 33 44 55 00 00 00 00 00 00 00 00 00 00 oiuu & ......................
020     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
030     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
040     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
050     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
060     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
070     11 22 33 44 55 00 00 00 00 00 00 00 00 00 00 oiuu & ......................
080     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
090     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
0A0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
0B0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
0C0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ............................
0D0     11 22 33 44 55 00 00 00 00 00 00 00 00 00 00 oiuu & ......................
--------------------------------------------------------------------------------
D. .Drive          R. .Read Sector       U. . Unerase      Disc A Format DATA
T. .Track          W. .Write Sector      Q. . Query        Track 0 Sector C1
S. .Sector         M A  Modify Sector    J J  . Paper      File DIRECTORY
F. .File           C C  .Cat Directory   K K  . Pen
--------------------------------------------------------------------------------
CTRL-A to EXIT              Press (D) to LOGIN disc                    PYRADSC
```

Figure 5.0 Disc-Nurse

WARNING: You should not modify disc sectors directly 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!

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

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

TRACK and SECTOR.
Press (T) to enter a track address, press (S) to enter a sector address.

READ.
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 or backwards at sector level.

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

Figure 5.1

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. When the sector changes are complete, press the ESC key. The changed sector can be written to disc with the (W) command...

WRITE.
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.

UN-ERASE.
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.

FILE ACCESS.
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.

QUERY.
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 a 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 unlikely 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 ESC key.

CATALOGUE and EXTENDED DIRECTORY.
Pressing (C) will display the disc catalogue using the standard display format. For extended directory information and hard-copy facilities press (CTRL-C). 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.

SCREEN-PRINT.
Press the (.) key on the numeric (function) island. The contents of the screen will be decoded and printed. Press the ESC key twice to cancel printing.

PAPER and PEN COLOURS.
Use of the (J) and (K) keys will step the paper and pen colours through their next ink values. Experimenting with the inks will improve screen readability on both mono-chrome and colour monitors according to background lighting conditions. CTRL-J and CTRL-K can be used to step through the ink colours in a reverse fashion.

6.0 UTILITIES.

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

- Directory Display of Drives (A) and (B).
- File Renaming, using a link to the |REN command.
- File Erasing, using a link to the |ERA command.
- File Transfer; any AMSDOS file type, Tape and Disc.

INITIAL PROMPT.
The following prompt is displayed....

Drive (A), (B), (D)elete, (R)ename, (C)opy or (X) to Exit.

Selection of (A) or (B) displays the appropriate directory. Selection of (D) or (R) provides file DELETE and RENAME functions and need no further explanation. (Wild cards may be used).

COPYING FILES.
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 reply 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. The options and valid responses are described on the next page.

NOTE:– The Destination Tape or Disc may be changed BEFORE the reply to 'PROTECTION ?' is entered.

COPY OPTION RESTRICTIONS.

	Disc to Disc	Disc to Tape	Tape to Disc
ASCII	Compression? Reply Yes or No. 36K limit ***		
BINARY	36K limit ***	Note-1	Note-1
BASIC	36K limit ***		
CPM	Compression? Reply SPACE-BAR 36K limit ***		
Protection can be INSERTED or REMOVED in all cases.			

Figure 6.0   Copy Options.

NOTE-1: A BINARY file written to tape via the PYRADEV Utility System MUST be restored to disc with the same Utility in order to restore the correct LOAD point address in the File Header. Such files may NOT be RUN until this has been done. They can however be LOADED to a specific memory location from tape and CALLed. (Game Writers BEWARE).

7.0 TRAINING.

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 LOADed and CALLed from BASIC 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 CTRL-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 ALL 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.

Please ensure you understand how the system operates before using it on live project code, and make sure you always keep backups of development source code.

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

8.0 CONTROL KEY SUMMARY.

EDITOR (edit mode)

CTRL-A: Abort and Exit	CTRL-G: Goto Line nnn	CTRL-U: Undo Changes
CTRL-B: Begin Pointer	CTRL-L: Set Mark	CTRL-V: View Begin/End
CTRL-D: Delete Block	CTRL-O: Open Up Text	CTRL-X: Save and Exit
CTRL-E: End Pointer	CTRL-S: Search System	CTRL-Y: Vary Cursor
CTRL-F: Find Marker	CTRL-T: Tab Settings	CTRL-Z: Command Mode
Island(.): Print	Island(0): KS memories	Island(1-9): Replay KS
ESCAPE: Help-Display	Small-Enter: Toggle KS Recording On and Off	

EDITOR (command mode)

A: Set Drive A	D: Delete File	L: Load Options	J,CTRL-J: Paper
B: Set Drive B	R: Rename File	S: Save Options	K,CTRL-K: Pen Ink
C: Compression	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	S: Exec Single Step	Q: Query / Search
Z: Clear Trap Point	CTRL-S: Double Step	X: Examine all ROMs
R: Change Registers	L: Load Test File	Y: Copy Memory Area
M: Modify using HEX	W: Write Disc File	CTRL-D: Dis-Assembly
CTRL-A: Modify ASCII	O: Other Display	CTRL-X: Exit Monitor
G: Goto an Address	U: Update Screen	J,CTRL-J: Paper Ink
CTRL-G: Set Step Addr.	N: Refresh Screen	K,CTRL-K: Pen Ink
CTRL-T: TAPE/DISC	CTRL-C: Catalogue	Island(.): Screen Print
SHIFT 0,1,2: Set mode	CTRL-B: Bank Select	

DISC-NURSE (main menu)

D: Select Disc Drive	R: Read Sector	C: Catalogue Display
T: Set Track Value	W: Write Sector	CTRL-C: Extended Dir.
S: Set Sector Value	Q: Query / Search	M: Modify using HEX
F: Select File	J,CTRL-J: Paper	A: Modify using ASCII
U: Un-Erase File	K,CTRL-K: Pen Ink	(CTRL-H: Hddr Checksum)
Island(.): Screen Print		

UTILITIES (main menu)

A: Display Drive A	D: Delete File	X: Exit Program
B: Display Drive B	R: Rename File	C: Copy AMSDOS File

UTILITIES (copy menu)

1: Disc to Disc	(Optional ASCII Compression)
2: Disc to Tape	(Protection Insert or Remove)
3: Tape to Disc	(Always displays header information)