A.D.A.M

INTRODUCTION

ADAM is a powerful machine language development system for the Amstrad 464, 664 and 6128 computers. Unlike some systems, all three modules (assembler, disassembler and monitor) are resident in memory at the same time. ADAM is written in relocatable machine code, allowing you to position it almost anywhere you like in memory so as to avoid conflict with your own programs.

GETTING STARTED

ADAM may be located at any address from 1000 decimal to 28000 decimal. To load the system, type:

RUN "ADAM"

Normally ADAM loads at 16384 (4000 hex), but if you wish to specify a different address, press the space bar whilst the first part of the program is loading.

Loading the system like this has the disadvantage that any BASIC program will be overwritten. To load ADAM and keep any BASIC program intact, decide where you want to locate the program - let's say the address is 6000 - and type:

```
MEMORY 6000-1
LOAD "ADAM.BIN",6000
CALL 6000
```

When the system first runs, you are in the MONITOR. Most ADAM commands are entered from the monitor, and consist of a single letter, often followed by a single digit number (which specifies a particular option), and sometimes by an additional parameter. For example, the command to save (or PUT) a source file on tape or disk is **P***n*,*filename*. Obviously you must insert your own values for the parameters *n* and *filename*.

An example session using ADAM is shown in Appendix D to help you get started.

ADAM occupies 10485 bytes. It contains its own display routines and stack to help keep it immune from the effects of programs that are being debugged. ADAM uses as few ROM routines as possible, but when it does it always uses the standard vectors.

There are two important addresses that ADAM uses: HRAM and HMEM. HRAM is the highest address in RAM used by the ADAM system, normally the top of user RAM - but you can change it if you wish. Memory from the start of ADAM up to HMEM is protected by ADAM from being overwritten. Normally the symbol table begins at HMEM. You can alter HMEM if you wish.

ADAM uses the # symbol to identify hexadecimal numbers, eg. **#4000**.

THE ADAM ASSEMBLER

The heart of ADAM is a powerful two pass assembler. Using the assembler you can assemble a source file created by the EDITOR into a machine code object file which can be executed directly by the Z80 processor.

Each line is divided into several zones or fields - labels, opcodes, operands and comments separated by at least one space. For example:

```
LOOP   ADD   A,(HL)   ;this is a comment
label  opcode operand        comment
```

A label must always start at the beginning of a line, begin with an alphabetical character and be between one and eight characters long. Operands can be up to 26 characters long.

To assemble a source file which is currently in memory, use the monitor A command. The file will be assembled in two passes. If an error is found, assembly will be aborted and an error message displayed (see Appendix A). The only exception to this is error 11, LABEL ABSENT, which is flagged but does not stop assembly.

If during assembly ADAM realises that the symbol table will overwrite the object code or extend beyond HRAM, it will automatically copy the table into display memory before assembly continues. You can force ADAM to use the display memory for the symbol table if you select assembly options 1 or 3. The assembly option you choose also determines where the object code is positioned:

COMMAND	SYMBOL TABLE	OBJECT CODE
A	HMEM	ORG address
A1	Display RAM	ORG address
A2	HMEM	After symbol table
A3	Display RAM	HMEM

After assembly the symbol table can be displayed (or printed) using the monitor command **X1**.

LINKING TEXT FILES TOGETHER

When a text file exceeds about 20k it becomes necessary to split it into smaller blocks in order to assemble it, otherwise there is insufficient memory to contain both the object code and the symbol table. Each link in the chain except the last should end with the special directive ***F,***filename* where *filename* is the name of the next block. ***F** must be entered in the label field and must be the only command on the line.

To assemble a chain of linked blocks you should use a command of the form:

A1,*filename* or
F2,*filename*

The **A** version of the command will start the assembly with the file in memory (which should be the same as the one named in the filename). Make sure that you have saved it away first!

The **F** version of the command must have option 2 selected, and will load the named file before starting the assembly.

PSEUDO-OPS & DIRECTIVES

In addition to the standard Z80 opcodes, ADAM accepts several pseudo-ops and assembler directives. They should all be entered in the opcode field.

ORG *address*
Defines the origin or start address at which the assembled machine code will execute.

EQU *value label*
Assigns the value specified (which must be a constant or a previously defined label) to the label. Labels can be up to 8 characters long.

DEFB *byte*(,*byte*...)
Writes a series of 8-bit values into memory.

DEFW *word*(,*word*...)
Writes a series of 16-bit values into pairs of bytes in memory (low byte followed by high byte).

DEFS *bytes*(,*value*)
Reserves a block of memory and (optionally) fills the block with the specified value.

DEFM *string*
Writes an ASCII string into memory.

ENT *address*
Defines an entry address for use in conjunction with the monitor J command (see debugger section).

IF *expression*(...**ELSE**)...**END**
Allows conditional assembly of sections of source code. If the expression is TRUE (not equal to 0) when it is evaluated, then the code following the expression is assembled. If the result is FALSE (0) and an optional ELSE pseudo-op is found, then the code following the ELSE will be assembled instead. Assembly returns to normal after an END is encountered.

ARITHMETIC EXPRESSIONS

ADAM allows the use of the following operands and operators:

OPERANDS

12	Decimal numerical constant
#20	Hexadecimal numerical constant
%10110	Binary numerical constant
"a"	ASCII code constant
LABEL	label (which has a constant value)
$	current location counter value
:x	contents of address x (like PEEK(x))
::x	contents of x,x+1 (like DEEK(x))

OPERATORS

+	addition
-	subtraction
*	multiplication
/	division
&	logical AND
@	logical OR
!	logical XOR
?	modulus (a?b=a-b*INT(a/b))

THE ADAM EDITOR

ADAM has a powerful full screen editor which allows you to prepare and edit source files. Type **L** to enter the editor from the monitor. The editor really has two modes of operation - EDIT mode allows you to examine and edit lines of text already typed, as well as perform more complex functions such as moving and copying blocks of text; INSERT mode allows you to type in new lines. When you first enter the editor using the **L** command, you are in EDIT mode.

INSERT MODE

To enter INSERT mode, press CURSOR LEFT when the cursor is already at the start of the line. You can now type your source code into the computer (it will be inserted immediately following the indicated line).

When you enter each line it will be checked and any unneeded spaces removed. At this stage the opcode is also 'tokenised' to save space and speed up assembly time. The line will be reformatted and redisplayed on the screen. If ADAM detects a mistake in the line, then the cursor will be positioned on the line at the point that the editor detected the error. Opcodes can be typed in upper or lower case - ADAM will always convert them to upper case. Labels can also be typed in upper or lower case, but ADAM will treat **START** as being different from **start**.

To exit INSERT mode and return to EDIT mode, simply enter a blank line.

EDIT MODE

When you are in EDIT mode, the line you are currently working on is indicated by the > indicator. Use CURSOR UP and CURSOR DOWN to move the indicator up and down the text. To make changes to the current line, press CURSOR RIGHT. You can then move about the line using CURSOR LEFT and CURSOR RIGHT and delete characters to the left of the cursor using the delete key. To delete an entire line, simply position the > indicator and press DEL.

To return to the monitor, press RETURN when the > indicator is displayed. When you subsequently enter the editor this line will be remembered as the current line (unless an assembly error has occurred). Alternatively, you can enter the editor at a chosen place in the text by typing **L***label*, where *label* has already been defined.

SAVING AND LOADING TEXT

The P (Put) command in the monitor allows you to save source files:

P*n*,*filename*

There are three options:

0 - Save the entire text file
1 - Save the text file starting from current line
2 - Save object code (valid only after assembly)

The G (Get) command will reload a text file previously saved:

G*n,filename*

If there is already text in memory, the new text will be appended to it rather than loaded over it. The option is a dummy argument and its value has no effect.

DELETING A TEXT FILE

To delete the text in memory, enter N in the monitor. If you execute this command by accident, you may recover the text by entering 59 into the TEXT address given by the X command (see page 9).

ADVANCED EDITING

Search and Replace
This facility uses strings set up in the monitor using the S command - two strings can be set up, *string1* (the search string) and *string2* (the replacement string). The command takes the form:

S*n,text*

Where *n* is 1 or 2 then the text is assigned to the appropriate string and both values confirmed. If *n* is 0, ADAM will display the current values of *string1* and *string2* without changing them.

To activate a search for *string1* (from the current line onwards), press the COPY key. If an occurrence is found, the page of the text where it is located will be displayed and the line will be indicated by a dollar symbol ($). To replace *string1* by *string2* press S - any other key will abort the function. If the replacement is valid, the search will continue for the next occurrence of *string1*. If the replacement would cause an error in the format of the line it must be corrected before the search can continue.

Block Editing Commands
These commands require that @ symbols are used to mark the start and end of the block. The @ symbol must be the first and only character on a line. These markers are ignored by the assembler.

Delete block - press CTRL-D. This erases all the text between (and including) the first pair of block markers found.

Copy Block - press CTRL-C. This will create a copy of the text between the first pair of block markers encountered and insert it immediately after the current line.

Kill Markers - press CTRL-K. This will erase all block markers in the text.

THE ADAM MONITOR

Apart from allowing access to the editor,assembler and debugger, the monitor has several other commands which allow you to modify system parameters or directly examine or change registers and memory:

SYSTEM COMMANDS

Upper/Lower Case Toggle
ADAM accepts both upper and lower case. Toggle between the two using the CAPS LOCK key.

Change Number Base
ADAM normally displays numbers in decimal. Press CTRL-B if you prefer numbers displayed in hex. To switch back to decimal press CTRL-B again. The number base being used is shown on the top line of the screen.

Printer Hard Copy
If you want to get a hard copy of the output sent to the screen, press CTRL-P. To switch this feature off, press CTRL-P again.

Memory Banking
ADAM allows you to switch between having RAM or ROM between #0000-#3FFF and between #C000-#FFFF. CTRL-R toggles between RAM or ROM in the lower block (observe the LRAM/ROM indicator on the top line of the screen - LRAM indicates that there is RAM from #0000-#3FFF, ROM indicates that there is ROM). CTRL-E toggles between RAM or ROM in the upper block (URAM or EXT on the top line). When EXT is indicated the ROM can be selected using a command of the form:

E*rom number*

The ROM number must be between 0 and 251. For example, **E0** selects the BASIC ROM, whilst **E7** would select the disk drive ROM.

Display System Addresses
There are three useful system addresses which can be displayed using the **X** command:

TEXT - the address of the first byte of the text file
END - the address of the last byte of the text file
HMEM - the address of the last byte occupied by the ADAM system

If you type **X1**, the symbol table is also displayed.

Varying the Size of ADAM's Workspace
The entire area of memory between the start of ADAM and HMEM is reserved exclusively for use by ADAM. Any command which attempts to modify this memory will be aborted with error 12, BAD LOCATION. The lower HMEM, the smaller the space reserved for the text file and vice-versa. You may change HMEM using the command:

M*address*

The address of HMEM must be in the central 32k of RAM (ie. between #4000-#BFFF). If HMEM exceeds HRAM, error 8, BAD MEMORY will occur. If it is too low then error 12 will occur. Using the **M** command destroys an existing symbol table.

Return to BASIC
To exit ADAM and return to BASIC, enter **B**. To return to ADAM you simply need to call it at its start address.

MEMORY/REGISTER COMMANDS

Examine Memory
Use **K***address* to examine memory from the chosen address. To halt the listing, press **Q** or CTRL-C. Note that bit 7 of the characters in the listing is always reset to 0 so that they are readable.

Modify Memory
You can modify bytes or pairs of bytes using the **Q** command and the **W** command:

Q*address,byte*
W*address,word*

Q stores a single byte in a given address (similar to POKE in BASIC). **W** stores a word (2 bytes) into two consecutive addresses with the low byte first (similar to DOKE).

Disassemble Memory
To disassemble memory enter the command **D***address*. If the printer is selected, ADAM will ask for an end address. The **Q** key or CTRL-C will halt the disassembly - any other key will continue the disassembly an instruction at a time.

Disassemble to Source
You can create a source text file from object code if you wish - use the command **H***address*. The disassembly is effected in two passes and the temporary symbol table generated is stored in the display memory.

Because machine code programs are rarely made up entirely of executable code but usually blocks of code and blocks of data, after entering the **H** command, you will be prompted with TEXT? You should enter an address. All bytes from the initial address to this address will be disassembled as assembly code.

Bytes following the text address will be disassembled as individual bytes and included in the source file using the pseudo-op DEFB. If there is no block of text bytes in the code you are disassembling you should enter nothing in response to the prompt.

If you enter an address, ADAM will prompt you with CODE? This allows you to enter an address after which bytes will be disassembled as assembly code again. You can alternate between blocks of code and data tables if you continue to enter addresses in response to the prompts. When you have entered the last transition, you should enter nothing in response to the next prompt. You will then be asked for the end address of the final block.

Labels are automatically generated according to the following criteria: a label beginning with R represents an address at which a branch terminates. If the label occupies a line on its own, it is because several branches terminate at that address. An address referenced by a call or a jump generates an empty comment line and a label beginning with W.

Modify Registers
The contents of registers can be modified by use of the full stop (.), for example:

.BC,2	- loads BC with the value 2
..BC,2	- loads B with the value 2
...BC,2	- loads C with the value 2
.BC',#ABCD	- loads BC' with #ABCD

MISCELLANEOUS

Evaluate Expression
O*expression* will evaluate the given expression and display the result. The terms and operators described on page 5 may all be used. If a symbol table exists, labels may also appear in the expression. For example, **Ostart** will display the value of the label **start**; **O%101+#13-9** will give the result 15 (or #0F if hexadecimal mode is selected).

THE ADAM DEBUGGER

As well as the monitor commands for examining and modifying registers and memory, ADAM includes a versatile run-time TRACE mechanism. This allows a program to be executed instruction by instruction and the contents of the processor registers to be displayed at each stage.

SINGLE-STEP TRACING

To enter single-step trace mode enter **T** in the monitor. Press SPACE to single-step through the program starting at the address contained in the Program Counter. At each stage the contents of the registers and the first 16 bytes of the stack are displayed, then the instruction is disassembled, executed and the register display updated. Instructions which modify the RAM/ROM configuration will also modify the top of page indicators.

Each instruction is checked before execution to make sure that it will not corrupt ADAM. Any attempt to modify the memory are between ADAM's starting address and HMEM will cause error 12, BAD LOCATION.

You can return to the monitor at any time by pressing **Q** (Quit) or CTRL-C. To start tracing again, simply press **T**. Take care about the program counter contents! To start tracing at a selected address use the command T*address*. The Program counter will be loaded with *address* before the trace begins.

QUICK TRACING A SUBROUTINE

To execute a subroutine starting at the address in the PC use the **R** command. The instruction disassembly and register contents are not displayed, though they are checked before execution. ADAM stores the contents of SP and enters single-step mode when the SP has the stored value plus 2, ie. a RET, POP etc. has been encountered. If the **R** command does not return (for example, an endless loop), CTRL-C will allow it to be stopped and returned to single-step mode.

You can quick-trace from a chosen address if you specify the address, ie. R*address*

PROGRAM EXECUTION USING ADAM

To execute a program, use the **J** command. This command has three forms - if you use **J** by itself, it will jump to an address previously set by the ENT pseudo-op in an assembled piece of code. If you follow the command by an address, then execution will commence in RAM at that point (regardless of the LRAM and URAM settings), with the BC' register taking the value #7F8E and F' reset to 0. Finally, **J$** will execute a routine starting at the address (in ROM or RAM as indicated at the top of the screen) in PC, and with BC' and AF' unchanged.

To execute a chosen address without affecting BC' and AF' you can use a command of the form **J$-$+***address*.

Before executing a **J** command, ensure that the contents of SP is within the central 32k of RAM.

In order to return properly to the ADAM monitor, routines being executed using **J** should be terminated with a JP to the start of ADAM rather than a RET.

APPENDIX A - ERROR MESSAGES

When an error occurs, an appropriate message will be displayed on the screen and there will be a return to the monitor. Below is a list of error messages and their probable causes.

0. SYNTAX ERROR	- command not recognised or syntax incorrect - a source code line is incorrect*
1. REDEFINED LABEL	- an attempt is made to define a label more than once
2. RELATIVE NUMBER	- a relative reference is out of range*
3. OUT OF RANGE	- a number >255 is encountered when a byte is expected - a number >65535 is found when a word is expected
4. ILLEGAL CHARACTER	- an unrecognised character is found on a source line*
5. ILLEGAL REFERENCE	- an undefined label is referenced* - a *F command is used on an unlinked file*
6. BAD ORG	- the ORG address is occupied by ADAM - the resultant object code would overwrite ADAM*
7. I/O ERROR	- printer off-line/not connected - tape/Disk operation failed
8. BAD MEMORY	- loading too large a file for current configuration - not enough room to add a line in the editor - object code exceeded upper limit of RAM (HRAM)* - the argument to the M command is too high
9. NO TABLE SPACE	- symbol table exceeds 16k during option 1 assembly*
10. BREAK	- you have interrupted an operation
11. LABEL ABSENT	- you have used an undefined label in an expression
12. BAD LOCATION	- attempt to modify a byte within ADAM using Q or W - traced routine has tried to modify a byte in ADAM - the argument for the M command is too low

*Errors which occur during assembly

APPENDIX B - TEXT FILE COMPATIBILITY

In order to assemble a text file created by another editor using ADAM, it is necessary to convert the file to ADAM format. ADAM text files are basically ASCII files with the exception that opcodes are tokenised. Lines are terminated by an ASCII Carriage Return (13) and the file is terminated by an ASCII Null (0). Comments are signified by ASCII 255 followed by a chain of alphanumeric characters.

The opcodes are tokenised in accordance with a table that starts at the address TEXT-242. Bit 7 of the last character of each mnemonic is set to 1 in order to signal its end.

You may find it instructive to edit a few lines of text and examine them using the K command.

APPENDIX C - USEFUL ADAM ADDRESSES

Since ADAM is relocatable the addresses are given relative to its start address.

ADAM+37 PTEXT (2 bytes)
Stores the address of the beginning of the line indicated by the > indicator.

ADAM+42 FLG1
Bit 1 of this variable is at 1 if an assembly has been initiated using option 2. The expression IF :FLG1&2 in the text file will validate the assembly of the following lines if option 2 is selected. The expression IF :FLG1&2!2 will invalidate the assembly if option 2 is selected. Bit 5 at 1 indicates that the assembly is carried out by blocks.

ADAM+80 HRAM (2 bytes)
Stores the last address at which ADAM may deposit the object code or may establish the value of HMEM (M command).

ADAM+43 FLG2
Bit 1 is the decimal/hexadecimal flag. Bit 7 is the printer on flag.

ADAM+10243
This is the beginning of the table address which contains the mnemonics.

ADAM+2354
This is the point of entry to the monitor.

APPENDIX D - EXAMPLE SESSION

You have just loaded ADAM, the text file is empty, and the monitor is awaiting a command. Press L, then CURSOR LEFT, and enter the following program:

```
        ORG 40000      ;the program will be located at 40000
start
        ENT $
        LD A,"A"
        LD B,26
loop
        CALL #BB5A     ;displays the character in A
        INC A
        DJNZ loop      ;loop 26 times
        RET
finish
```

Now, to see what ADAM is capable of, press <ENTER> twice, then:

Assemble it:	A <ENTER>
Execute it:	J <ENTER>
Disassemble it:	D start <ENTER>
Trace it:	T start <ENTER>

Press SPACE until the PC displays #BB5A. At this point, if you have the time, continue to press SPACE and you will trace the ROM routine which displays a character on the screen.

You could press Q then R <ENTER> to quick trace the routine. After 26 iterations of the loop, single step tracing will resume. At this point, the contents of the registers will be displayed - B will be zero and the PC will have the value which was at the top of the stack before tracing the routine. Finally, let's try out the disassembler - enter Q to return to the monitor then type:

Hstart <ENTER>

When ADAM displays 'Text?' press <ENTER>; in response to the 'END' prompt type 'finish' which is the label at the end of the program. Now when you enter the editor you will find that the machine code has been disassembled into source code.

Manufactured and distributed in the UK by
Audiogenic Software Ltd, PO Box 88, Reading, Berkshire