**MCODER** is the first true compiler for use with the **ZX81.**

It has been carefully written to occupy as litte space as possible (just under 4K) in order to leave maximum space for your programs.

Although **MCODER** is a very sophisticated and versatile tool, do not expect miracles, you will have to become familar with its method of operation if you are to use it to its best advantage.

**How do I load MCODER?**

**MCODER** must always be present in the **ZX81** before you try and enter any code. It is loaded from tape using **LOAD" "**. **MCODER** contains its own self checking routine to ensure that it loads correctly, if for any reason the program corrupts, an error message will be displayed on screen and you should reload.

Once loaded you must remember that **MCODER** uses line numbers **1-3** inclusive and cannot be deleted.

**How do I use MCODER?**
Using **MCODER** is really a very simple operation. First of all, enter the BASIC program you wish to compile. It is essential to run and check the program thoroughly before it is compiled and there is no break facility in the compiled code. Before writing a program it is strongly advised that you carefully read through the list of commands which **MCODER** will compile and ensure that your BASIC program consists only of these commands.

Once you are satisfied with the **BASIC** program you can compile by using the command **LET L = USR 17300**. The **BASIC** statements will then scroll up the screen as they are compiled.

If **MCODER** finds a command or statement it cannot compile then it will return to **BASIC** with an inverse **S** at or near the offending command.

**Possible errors include:**
1. Using illegal variable names.
2. **GOTO** 'variable' or **GOSUB** 'variable' which are not allowed
3. Illegal statement such as **SAVE** or **DIM A$(10)**

If an error is reported then you should correct it and recompile.

At the end of a successful compilation you will be shown 3 pieces of information :

I.) the location of the end **MCODER** code,
II.) the command necessary to run your compiled code,
III.) a query as to whether Yw wish to run the compiled code immediately (**RUN**? - reply **Y** or **N** to this prompt).

**Can I compile basic programs I already have on tape?**
**YES**- simply load **MCODER** first and then load the program you wish to compile. Using **LET L = USR 32462** after loading your own program will automatically push **MCODER** into position. You can now continue as shown above.

It is important to stress that you will probably have to make considerable changes in your **BASIC** program as it is unlikely to have been written to suit **MCODER.**

**Which Commands will MCODER compile?**
Note: the variable names may contain only **A-Z** and **0-9**. Arithmetic is restricted to the four standard operations + - * /.

| | |
|---|---|
| **AND** | Boolean **AND**. Allowed only in an **IF** statement |
| **ABS** | As BASIC. |
| **CHR$** | As BASIC. |
| **CLS** | As BASIC. |
| **CLEAR** | As BASIC. |
| **CODE** | As BASIC. |
| **COPY** | As BASIC. |
| **DIM A(V)** | Only one dimensional arrays are availabe in **MCODER** and there must be at least 2***V** bytes spare space at run-time. No runtime array bound checking is done so make sure it works under normal **BASIC**. If you redefine an array a new version of it is made but the old one is not deleted. This means that repeated allocation can eventually fill the machine and give an **error 4** (out of memory) either when allocating an array or a string. Array and string space stretches from **STKEND** to 256 bytes below **RAMTOP**. All arrays, strings and variables are erased when you reenter an **MCODER** program and all the space is available again. There are no string arrays. |
| **FAST** | As **BASIC** except that **MCODER** does not return to **SLOW** mode when the screen is full during **INPUT** or during **PAUSE** |
| **FOR X = V TO U** | - X increments in steps of 1 from **V** to **U**. **NEXT** X Ends the loop. Note that (**U-V**) must be less than **32767**. |
| **GOSUB N** | Calls line **N** as a subroutine. If line **N** does not exist then it goes to the next line after **N**. Note: that **N** must be a positive integer constant. |
| **GOTO N** | Jumps unconditionally to line **N** .Otherwise as for **GOSUB**. |
| **IF V op U THEN** | where op is any of **AND**, **OR**, <>, =, =<, >= or >,<. Note that V and U must not differ by more than **32767** For string operations **AND** and **OR** are not applicable. |
| **INKEY$** | As BASIC. |
| **INPUT A or A$** | For numbers a leading negative sign is allowed. For strings the maximum length is 31 characters.<br>No graphics mode. |
| **INT** | Included to facilitate test under **BASIC**. |
| **LEN A$** | As **BASIC** exept that A$ cannot be sliced. |
| **LET** | As BASIC. |
| **LPRINT** | As BASIC. |
| **NEW** | As BASIC. |
| **NEXT** | See **FOR - NEXT** |
| **OR** | Boolean **OR** only available in **IF** statements |
| **PAUSE V** | Causes the program to wait for V/50 secs if in **SLOW** mode. It does not go from **FAST** to **SLOW** mode as in Sinclair Basic. There is no flicker as **PAUSE** starts and finishes. V must be positive and less than **32768**. Pressing **SHIFT+EDIT** returns to **BASIC**, any other key causes the next statement to be executed. |
| **PEEK** | As BASIC. |
| **PLOT** | As BASIC. |
| **POKE** | As BASIC. |
| **PRINT** | As BASIC. |
| **RAND** | As BASIC. |
| **REM** | Ignored as usual except that "**REM**?" looks to see if the **SHIFT+EDIT** keys are pressed and if so returns to BASIC with an **error D**. This is useful to escape |

from infinite loops.

**RETURN** Returns from a subroutine started by a **GOSUB**. Make sure that your **GOSUB**S and **RETURN**S match as no check is made.

**RND** Returns a random integer between 0 and 32767. (**NOT** the same as **BASIC**).
To obtain the same effect under **BASIC** use **USR 16550**.

**SCROLL** As BASIC.

**SGN** As BASIC.

**SLOW** As BASIC.

**SQR** Interger square root

**STOP** If **MCODER** finds a **STOP** statement compilation ceases there. If you wish there to be a **STOP** in the middle of your program then the command **LET L=USR 3292** will give an **error 9** and stop.

**Strings** By default strings hide a maximum length of 32 characters. If you exceed the maximum string length then you will write into whatever follows (either another string or an array). However we "Further Features". String slicing may not take the form **A$( TO M)** or **A$(M TO )**. There are no string arrays.

**UNPLOT** As BASIC.

**USR** As BASIC.

**Improved Scrolling of output.**
Under Sinclair **BASIC** when the output reaches the bottom of the screen an error 5 is reported. Under **MCODER** when the output reaches this point an inverse **?** is shown at the bottom left of the screen. Pressing **CONT** causes a **CLS** and the program continues (holding your finger on **CONT** causes this process to be repeated indefinitely). Pressing **SHIFT+EDIT** stops the program,

key **D** (SLOW) causes the screen to scroll at a maximum of 1 line per second, pressing **Z** (COPY) causes a hard copy to be produced on the **ZX printer**. Any other key than these causes the screen to scroll upwards while the key is depressed and will stop if no key is pressed. The choice may be varied each time that **?** is displayed.

**POINTS TO REMEMBER**
When running a compiled program, **MCODER** must always be present as it contains the run time system and arithmetic routines.

You must not use **SINCLAIR**'s scroll command if you intend to mix **BASIC** and compiled code as this destroys the regularity of the display file which **MCODER** needs.

**Further features of MCODER**
You may choose where to put the code generated by **MCODER** if you enter by **RAND USR 17287**. **MCODER** will prompt "**CODE**" to which you should reply with the positive integer number address to which the code should be compiled.
To find out how long the code will be compile to 0.
There are several other useful locations. In you program **POKE 16417, N** causes the screen to scroll without asking "?" **N** times. At the end of this time location **16417** will again contain zero.

**Locations (16507 and 16508)** contain the present default string length (usually 32) which may be **POKE**d in your program to change the default. This value is reset as your program starts.
**Locations (16536 and 16537)** are used to prevent lines at the top of the screen from scrolling. The value should be **n***33 where **n** is the number of lines to be left alone.

**Location 15535** is the location which causes the compiler to auto scroll during compilation. If this is poked with zero then the compiler will wait as described in the section on scrolling above. Pressing **D** causes the generated code to be displayed.

You may with to have werel sections of compiled code.
This is achieved by the following procedure:

1. Compile the first section. The entry point will be given (X). Now **POKE X-6,1** to convert line 2 into line 1.
2. Compile section 2. A new entry point will be given (Y) **RAND USR X** still works to access the first action and **RAND USR Y** accesses the second. This may be repeated as often as space allows. (*NOTE* to users with machines greater than 16k. On no account must you allow any section of code generated by **MCODER** to cross the 32k word boundary).
3. If you wish to have your own machine code put it in a **REM** at line 1. The first byte available will be at location **20498**. Your compiled code will not now start at **20500** but you will be given the correct entry address.

The **BASIC** code may be quickly deleted by using **RAND USR 17281.**

**As an example of MCODERS speed.**

Enter the following short program and compile It.

```
100 FOR A=1 TO 50
200 FOR B=1 TO 30
300 PLOT A,B
400 NEXT B
500 NEXT A
600 STOP
```

Once it has compiled compare the running speed of the compiled code with that of the **BASIC** and you will see the kind of improvement **MCODER** brings.



# MCODER II

## ZX81



**452, Stoney Stanton Road,**

**Coventry**

**CV6 5DG.**

**Telephone: (0203) 667556**