Mr. M. Bell

Micro system–mini language

M5 is being offered by MICRODIGITAL as a high level language of 0.6K to run on a standard NASCOM 1 with a T2 monitor. At first glance the facilities offered are impressive for its size and the documentation is full of optimism. It must be accepted from the outset that if 2K and 4K BASIC interpreters have limitations then the 'M5' must have definite handicaps, but if it is successful in providing programming assistance for what is essentially a machine code micro–system then it fulfils its function and its limitations must be accepted.

Language Facilities

M5 is mainly intended for positive integer (ie whole number) arithmetic within the range 0 to 65,536. Its use of Reverse Polish Notation (RPN) in preference to arithmetic notation allys it with the early four function calculators, and indeed those still produced by Hewlett Packard. There are 26 possible register labelled A to Z for storing data and these can be entered, recalled or operated on at any time. Facilities have been provided to print 'strings', to input data from the keyboard and to output data onto the CRT.
One of the essentials of any high level language is its ability to make 'decisions', and here we have a choice of eight possibilities. It does however lack a 'Jump If Negative' instruction which would be very useful. The inclusion of a 'GOSUB' instruction would also be worth its sacrifice in RAM. All jumps are made to flags which can be any character available on the keyboard.
Once a user program has been entered it can be modified by using the 'EDIT' command which lists the program and positions a cursor. The cursor is in the form of a rub–out character positioned over the required entry. For reference the blotted out entry is printed at the top of the screen. Further commands will reposition, select next line or forward/backward space the cursor. Changes are then made by deletion or insertion commands.

The Hardware

The package was received containing one cassette tape and one read only floppy. The 'floppy' was dated 23rd March, 1979 and marked 'Provisional'. It is very well written and illustrated using functional programs. Each command and operator is well explained and a list bringing these together is shown in Table 1. A hexadecimal listing has been provided and this is correct in every detail. The cassette tape has three copies of the program in NASBUG T2 format, is of good quality and loaded first time.

The Program

The program is extremely well written and even an experienced programmer would benefit from analysing it. Briefly, and without giving the game away, the program uses three NASBUG routines, CHIN, SCROLL and CAT. These are included within the three main sections to facilitate input and output control. The first section, INPUT, simply enters the keyboard character into the user RAM and echoes the character on the CRT. The EDIT and RUN sections consist mainly of decision boxes leading to short action branches, to carry out each function. Extensive use is made of the stack when carrying out multiple arithmetic routines and this can encroach severly into the user RAM allocations which begin at location 0EFE.
As stated earlier there are bound to be disadvantages. The CRT is based on the 'write and scroll' techniques and this means that the top line of display cannot be used. Because of this the top line has been allocated for register storage and this results in a messy CRT appearance. What is more to the point however is that if 'shifted backspace' is ever used in a user program all the register contents will be destroyed.
When running a program errors are detected and displayed as shown in Table 2.

Operational Experience

The interpreter is accessed by keying 0C60 whereby a prompt 'M5' is displayed. The commands can then be entered as described. We found that it worked well, or almost. Th cursor control during edit requires the use of the "<" and ">" keys which do not exist on a NASCOM with T2 monitor, however changing locations 0E83 to 29H and 0E88 to 28H allowed the "(" and ")" brackets to be used instead. Another difficulty was encountered when trying to correct an entry during EDIT, INPUT facility. Not only does the erroneous character get entered but the backspace also. It must be remembered that although a backspace character appears to work on the CRT it does not function when the EDITOR is in use.
It is possible to make the user program area larger by including the 10H bytes of RAM thst are not usefully employed at the rear end of the program. These modifications to the original program are documented at the end of the article in Table 3.

Conclusions

A high level language can never be a bad thing for an inexperienced programmer, and this one works well, although most peoples needs would soon outgrow it. M5 is no real substitute for the sofhistication that can be obtained using machine code, nor can it compete against BASIC in larger systems. I does however have the advantage that it is small and can be used within a 1K RAM. The 230 odd bytes available to the user is very small but programs can be successfully written within this space. The "Lunar Lander" program detailed at the end of this article, Example 1., can be run on the standard NASCOM provided that the program improvements in Table 3 are incorporated.
There is mention of an 'M6' interpreter, and if this is can enhance the M5 to include negative integers, GOSUB and Go If Neg and is perhaps packaged in a 1K ROM, the advantages of this language may become more apparent.

Value For Money ?

The M5 is being given away free by MICRODIGITAL if you purchase your NASCOM from them, if not it will cost you £10 for the complete package. It is the authors opinion that this is overpriced and the money could possibly be better spent on a T4 or B–BUG monitor. If a high level language is desired then one of the 2K BASIC/Monitor packages that are coming on the market might be a better investment.

Example 1.

LUNAR LANDER

Note:– The modifications in Table 3 must be carried out.

```
(W720=F520=H65=V
(X" HT="H=?" VEL="V=?" FUEL="F=?)ZS
" BURN="?=B,F)GY0=F=B
(YF,B–=FB,2*=N256=A(RN,A/,A)LT+,2/=A)UR(S"
"(TA,2/,V+=SH,S–=H)ZPV,A–,5+=V,999)LZ
H,999)LM)UX
(M" CRASHED! AT"V=?"FT/SEC")M
(P9,V)GM" GOOD LANDING")M
(Z" TAKEN OFF!!")M
```

Another note:– The program must be entered exactly as shown to avoid errors. Spaces are important!

Table 1.

This table lists the available features of M5.

Commands:– There are four that can be used.

I	INPUT	Enters a new program from the start of the user RAM. It will over-write any existing program. A new program is terminated by a semi-colon ';'.
L	LIST	This will list the current program onto the CRT.
R	RUN	Runs the program.
E	EDIT	A very useful command which allows the current program to be easily modified.

Editor commands:–

R =Reposition the cursor at the start.
N =Next line.
(=Backspace cursor.
) =Forward space cursor.
D =Delete character.
I =Insert the following character until ;.
W =Return to the command mode.

Decisions

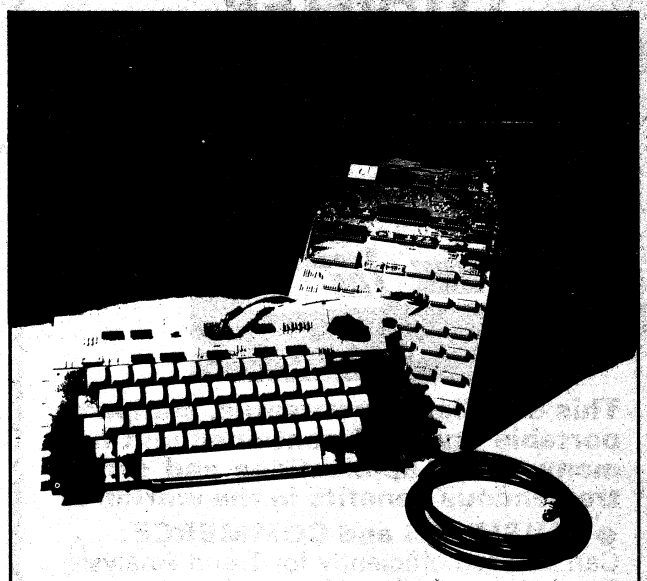
U	Unconditional jump
Z	Jump if zero
N	Jump if not zero
E	Jump if x = y
X	Jump if x ≠ y
L	Jump if x ⩽ y
G	Jump if x > y
M	Jump to command mode

Mathematical Operators

+	Add
–	Subtract
*	Multiply
/	Divide
£	Decrement 'x'
&	Increment 'x'

Other Operators

" " String
n=? Print the contents of register 'n'
? Input from the keyboard



The NASCOM 1 which the M5 runs on in its standard form.

Table 2.

Error Messages:–

SYM	ERR x	The symbol x is not allowed except in a string.
ID	ERR x	The symbol x is not a valid identifier, and an attempt was made to copy a value to it. Eg = x occurred.
JID	ERR x	The label was not found when a jump occurred.
JC	ERR x	The symbol x occurred in a jump condition position and is not a valid code.
	ERR x	The symbol x caused an error that is not one of the above.

Table 3.

The author has suggested the following changes to the M5 interpreter to rectify errors, clean up the display and increase the available user RAM by 10H bytes (i.e. run from 0EEE).

Location	Old	New	
0E83	3E	29	") "
0E88	3C	28	" ("
0E56	1F	1E	Clear CRT on RUN
0E5A	FD	ED	These changes start the
0EDB	FD	ED	user RAM at 0EEE
0E2F	FD	ED	instead of at 0FEE
0DE1	FE	EE	
0EB3	FF	EF	