M5 SYSTEM—AN INTERPRETER FOR THE NASCOM 1 by Raymond Anderson

NASCOM Implementation

The M5 interpreter was designed for implementation on small 8-bit microcomputers and the NASCOM 1 standard system was an ideal choice because of its popularity and use of a fairly powerful processor (the Z80).

With only about 940 bytes available to the user, the language had to be compact enough to write decent programs in a small space, and also have a small interpreter to leave the maximum amount of spare memory. A simple editor was almost essential if programs of over about 50 bytes were to be written and debugged easily, and this required about 100 bytes.

The editor, interpreter and command mode are closely linked—for example, program variables are maintained over edits and resets, and the editor sets its cursor to show the user where an error occurred.

A compact M5 program can be difficult to follow initially, so error routines which give the exact location and type of a run-time error are included, despite the penalty in RAM usage. (Execution speed is unaffected by error checking).

M5 is a very fast interpreter, although loops are not as fast as in machine code because each loop involves a small search. A well-written M5 program will carry out general calculations at about 1/3-1/5 of the speed of machine code. (M5 programs are usually much faster to write and debug of course!)

The user may write programs of about 230 bytes in length—quite large in M5. Overlarge programs may cause trouble when entered, but the most likely indication of an overflow is a lot of garbage appearing on the end of the program when it is listed.

Introduction

This document describes version 5a which implements some small tweaks and bug-fixes. m5a_nassys.nas runs under NAS-SYS 1/3. It loads and starts at C80. m5a_nasbug runs under NASBUG T2/T4 or BBUG. It loads and starts at C60. Both versions are exactly the same size. When started, the prompt “M5:” appears at the start of a line, indicating that the system is in command mode. The following commands are available:

I	Input a new program and delete any previous one. System responds with “Input” followed by a newline. Back-space can be used to correct typing errors. A semi-colon terminates program input and returns to the command prompt.
L	List the program and return to the command prompt.
R	Run the program starting at the first symbol, after printing a newline.

E Edit the program, inserting the character pointer at the place the last instruction was executed, or where an error was found. (See “The Editor”, below.)

RS Reset the NASCOM. This causes a return to the monitor. However, M5 can be restarted at the same address and the program is retained. Reset must also be used to stop a looping program.

Initialisation

When entering M5 for the first time after loading, it is best to initialise the user work area by entering a null program. This can be done as follows: (Underlined characters are typed by the system.)

```
M5:Input          (type I)
;                 (terminate input after entering nothing)
M5:               (system is now initialised.)
```

Other commands

M5 ignores any unknown command and responds with a new prompt.

Special Characters

In Input mode, the backspace key deletes the last character.

In Edit mode, the backspace key is ignored. Any typing errors must be fixed later using a combination of the Delete and Insert commands.

Semicolons are illegal characters in a program; they cannot even be used in strings.

An open bracket “(“ is illegal in a string; it will be mistaken for label marker.

Backspaces cannot be entered into strings in a program.

Apart from that, any character can be used in a string, including newline.

BASIC M5 LANGUAGE PRINCIPLES

M5 Arithmetic

Arithmetic in M5 uses 16-bit unsigned integers so numbers are in the range 0 - 65535 (decimal) and are modulo 65536, so that 65536 seems the same as zero to the language.

The following operators are supported:

* (multiply) / (divide) + (add) - (subtract) £ (-1) &(+1)

the last two are included for faster execution and for compact programming of loop control.

The Stack

M5 uses stack based (Reverse polish) expression analysis. This system can be used to express arbitrary expressions without the use of parentheses.

The Current Value

On a pocket calculator, the Idea of a current value is easy to understand as it appears on the display and is often called "x". In M5 there Is also a current value (also called "x") and it is only altered in the following circumstances:

1. If a number appears in the program (not in a string) x takes its value.
2. On encountering an identifier A-Z x takes the value stored there.
3. On encountering a ? (not after =) x takes its value from the keyboard.
4. After a diadic operator (/ - + *) x becomes the result.
5. If x is incremented or decremented (using & or £).

Variables

M5 has variables named A-Z and a special one @. One of these variables becomes current by simply quoting it in the program (point 2 above).

x may be stored in a variable by simply using =*k* where *k* is a variable name.

If =? is used, the current value (x) is displayed as a decimal number on the screen. (This is how numbers are output in M5).

EXAMPLES (These are all legal M5 programs- Try if unsure!)

(i) A What is in location A is now also in x (the current value).

(ii) ABC x takes on the values in A then B then C and keeps the value C.

(iii) 23 x becomes 23.

(iv) 23A x becomes 23, then x becomes A (i.e., the number in A).

(v) 23 456 x becomes 23 and then x becomes 456.

(vi) A=B x becomes A, then this value is stored in B.

(vii) A=B=C=D x becomes A . then this value is put into B, C and D.

(viii) A=?D=? x becomes A and this is displayed, then x becomes B and is displayed.

(ix) =?=A what is in x (left from last program) is displayed and put in A.

If you want to check what is going on, you can print x at any point in your program by inserting the characters =? – or, for neatness and readability, use =? “ " to print number followed by a space. For example, 23=?” “ 11111 =? in a program will produce the output 00023 11111.

Calculating

When a comma is encountered in an M5 program, the value of x is put on the top of the stack - pushing down all other members. The top element o the stack is called y. We can represent the

stack diagramatically to show what happens. Consider a short example and follow it step by step:

```
Program: A,33,,BA      Where initially A=l and B=2
step:    abcdefgh      (could have run l=A2=3 before)
```

STEP	SYMBOL	MEANS	x	y	Rest of stack top-->bottom
a	A	Load A	1	-	- - -
b	,	Push x	1	1	- - -
c-d	33	Load 33	33	1	- - -
e	,	Push x	33	33	1 - -
f	,	Push x	33	33	33 1 -
g	B	Load B	2	33	33 1 -
h	A	Load A	1	33	33 1 -

Items are removed from the top of the slack using the operators + / * -

These operators work on x and y and put the result in x, removing y from the stack and moving the other elements up the stack. The operators £ and % have no effect on the stack. In summary:

Operator	Function	Remarks
£	x := x - 1	Equivalent to ,1- (but faster)
%	x := x + 1	Equivalent to ,1+ (but faster)
+	x := x + y	y is lost. Overflow not detected
-	x := x - y	y is lost. Underflow not detected
*	x := x * y	y is lost. Overflowing bits put in @
/	x := x / y	y is lost. Remainder is put in @

The second example displays the result of A+B. When run, 00003 is displayed.

```
Program: A,B+=?       Initially A=1 B=2
   step: abedef
```

STEP	SYMBOL	MEANS	x	y	Rest of stack
a	A	Load A	1	?	- - -
b	,	Push x	1	1	- - -
c	B	Load B	2	1	- - -
d	+	x:=x+y	3	-	- - -

e-f	=?	Display	3	-	-	-	-

The third example evaluates (2*3) + (7-2) and displays the result.

```
Program: 2,3*,7,2-+=?          Add result of 2,3* to 7,2-
   step: abcdefghijkl          and display the result
```

STEP	SYMBOL	MEANS	x	y	Rest of stack		
a	2	Load 2	2	?	-	-	-
b	,	Push x	2	2	-	-	-
c	3	Load 3	3	2	-	-	-
d	*	x:=x*y	6	-	-	-	-
e	,	Push x	6	6	-	-	-
f	7	Load 7	7	6	-	-	-
g	,	Push x	7	7	6	-	-
h	2	Load 2	2	7	6	-	-
i	-	x:=y-x	5	6	-	-	-
j	+	x:=x+y	11	-	-	-	-
kl	=?	Display	11	-	-	-	-

Here are some further examples, comparing expressions in BASIC with their equivalents in M5:

BASIC	M5
`Z = M*N*A`	`N,M*,A*=Z  or  N,M,A,**=Z`
`Z = (N+M)*A`	`N,M+,A*=Z`
`Z = (N+M)*(A-M)`	`N,M+,A,M-*=Z`
`Z = N*N`	`N,*=Z`
`Z = N*N*N*N`	`N,*,*=Z  or N,,,***=Z  (M5 only needs to get N once)`

Getting Data In

Whereas =? is used to print a number (the value of x), ? by itself (without “=”) is used to request a number from the keyboard.

A number is terminated by any non-numeric character. Usually the user will type a space or a newline after entering number and the program will continue.

EXAMPLE `?,?*=?` prompt for a number, then another and print the product.

String print

Any string of characters surrounded by quotes “" is printed to the display exactly as written- including newlines etc.

```
EXAMPLE   “Input the number"

or        “NEW
           LINE”
```

A jump will find labels in a string so do not use (in a string.

A nicer version of the program above Is:

```
“NUMBER" ?, "TIMES BY” ? *" IS “ =?
```

A newline is produced by a newline between quotes.

Loops and jumps

M5 implements program flow control using labels and conditional jumps.

A label is represented by (*n* where *n* is any symbol which an be entered at the keyboard. Examples are: `(A  (!  (1  (.`

A jump is represented by)*kn* where *n* is a symbol which matches a label, and *k* is a condition code indicating what condition involving x or x and y must be true for the jump to occur. Valid condition codes are as follows:

Character	Jump occurs if	Comments
U	Unconditional	U stands for Unconditional
Z	x == 0	Z stands for Zero
N	x != 0	N stands for Non-zero
E	x == y	E stands for Equal
X	x != y	X looks like notequal sign
L	x <= y	L stands for less-than-or-equal
G	x >= y	G stands for greater-than-or-equal
M	Unconditional	M is Monitor; return to M5 prompt

EXAMPLES of valid jump symbols are: `)UA  )NI  )X$  )G(  (Z.`

When a jump symbol is reached, the condition indicated by *k* is tested and if it is found to be true, a jump is made to the first occurrence of a label with the matching identifier symbol.

EXAMPLES

(i) 2000 (A “HELLO” £)NA Prints out “HELLO” 2000 times.

(ii)	0 (A =? “ “ &)NA	Prints out numbers from 0 to 65535 separated by spaces (Thinks 65536=0).
(iii)	(A)UA	Loops until reset is pressed.
(iv)	0=N (A N=? &=N , 5555)GA	Prints out numbers from 0 to 5555..

WRITING PROGRAMS

M5 is a powerful language when all its features are properly understood, but it can be a little confusing for the beginner. There is fortunately an easy way of generating programs which can be used until familiarity with M5 is achieved. The method is to write the program in a more standard language and then translate into M5. While this method does not exploit the valuable “current variable” feature of M5. It will yield workable programs which are easier to follow in many ways. The program can then be optimised if required.

EXAMPLE A Program to print a table of squares from 1 to 20

BASIC	M5	M5 Optimised
10 PRINT “TABLE OF SQUARES"	“TABLE OF SQUARES	“TABLE OF SQUARES
20 N=0	“ 0=N	“ 0=N
30 N=N+1	(B N,1+=N	(B N&=
40 PRINT N, N*N	N=? “ “ N,N*=? “	N=? “ “,*=? “
50 IF N<>20 GOTO 30	“ N,20)XB	“N,20)XB
60 END	)M	

In M5 programs, newlines in the output must be included between quotes. Numeric output in M5 is not spaced, hence the space in the line equivalent to line 40.

The M5 program will run at about the same speed as the tiny Basic program.

If the M5 is optimised, keeping N in x as much as possible and using the free layout and the & operator, the speed will be considerably faster, perhaps 4-5 times faster than a fast tiny basic.

THE EDITOR

Introduction

The Editor is entered by typing E when in the command mode. The edit prompt of E: will appear. The editor will show the point where the last instruction was executed when it is entered by positioning a cursor at this location. The cursor is a shaded square which is denoted here by a _ (underline). The cursor indicates the current position of the character pointer, and the character pointed at by the cursor appears at the top right of the screen. All manipulation of text is done relative to this cursor because there are no line numbers in M5.

The character indicating end of file in M5 is a null character which appears as a box when it is

pointed at. A hazard in the M5 interpreter is that the cursor can be moved into the actual M5 Interpreter. A Rule must therefore be: DO NOT use any Delete or Insert commands unless you can see where the cursor is positioned.

Commands

To manipulate the text of a program, move the cursor to the required position and then operate on the text. The editor supports these command:

> Move cursor forward one place[1].

< Move cursor backward one place.

R Rewind – move cursor to the start of the file.

N Next – move the cursor to the start of the next line (stop at the end of the program)

D Remove (delete) the character at the cursor. The cursor now points to the next character.

Innnn; Insert the string nnnn before the cursor. Semi-colon terminates the inserted text. The cursor points to the same character as it did before the insert.

W Leave the editor and return to command mode.

Edit commands (except W) can be repeated on a line. Pressing newline results in a printout of the text with any edits done and the cursor in its new position. The backspace key is ignored (does not echo and has no effect) within the editor.

EXAMPLE: You have typed in a program as follows:

```
(A "HELLO THERE " N=? " IS N
WHAT ZUMBER DO YOU WANT”... etc
```

You want to move the cursor to the spelling error. Use: RN>>>>> i.e. move to start, move down a line, move in 5 characters. Press newline to print out the edited text and return to the edit prompt.

EXAMPLE: Edit ABCDERTYIJKLMNOP to replace RTY by FGH

```
ABCDERTYIJKLMNOP
E:R>>>>>              Move pointer to start then right 5 (to R)
ABCDE_TYIJKLMNOP      Character R appears at top R.H. side of screen.
E:D                   Delete current character.
ABCDE_YIJKLMNOP       T appears at top right.
E:DD                  Delete two more.
ABCDE_JKLMNOP         I appears at top right.
E:IFGH;               Insert correct characters.
```

1 > is shift-N, < is shift-M on the NASCOM under NASBUG T2/T4.

```
ABCDEFGH_JKLMNOP        String now correct; I is still current character.
```

When editing is complete, use W to return to command mode.

ERROR MESSAGES

If M5 detects an error at run-time, one of the following messages appears:

SYM ERR x	The symbol x is not allowed in M5 (except in a string).
ID ERR x	The symbol x is not a valid identifier, and an attempt was made to copy a value into it. (e.g. =x occurred.)
JID ERR x	The label x was not found when a jump occurred to it.
JC ERR x	The symbol x occurred in a jump condition position and is not a valid code (not one of U Z N E X L G M).
ERR x	The symbol x caused an error to occur. (Not one of above.)

In addition to giving the error type, the edit cursor is set up to point at the faulty symbol, so when the editor is entered from the monitor to correct the error, the cursor is in the correct position for amendments.

SAMPLE PROGRAMS IN M5

Number summing program

```
(A"INPUT A NUMBER"?" THANKS
  NOW INPUT 2 MORE NUMBERS"?,"AND"?"GOOD!
  THEIR SUM IS"++=? "
  ")NA "THEIR SUM WAS ZERO – TYPE 0 FOR MORE FUN OR
  1 to END "?)ZA "GOODBYE!")M
```

Factorial of a number

```
1=N ? )ZB (A =M , N* =N M£ )NA (B N=?
```

M5 24 hour clock:
(N.b. remove all spaces for good timekeeping)

```
)US (D N£=N )ND
H=?" HRS "M=?" MINS "S=?" SECS
" L=N S&=S , T  )XD
  0=S M&=M , T  )XD
  0=M H&=H , 24 )XD
  0=H )UD
(S 1750=L 60=T
"SET HRS"?=H"SET MINS"?=M"SECS"?=S"
" )UD
```

(Start put at end)

Note that the main timing loop is at the beginning for higher speed. 1750 is the timekeeping constant, make smaller to speed up clock.

Square root of a number:

```
256=M ?=N (1 n,m/ , M )LS +,2/=M )U1
```

```
                        (S " "M=?" "

Method used is very fast but a little hard to follow.

Prime numbers:             1=T
                        (N T&&=T
                           1=G
                        (A G&&=G
                           T,G/,G )GP
                           @ )NA )UN
                        (P T=? "
                           " )UN

This can be compacted to only one line of course, (a bit baffling though ):

1=T (NT&&=T1=G(AG&&=GT,G/,G)GP@)NA)UN(PT=?" ")UN
```

Rebuilding from source

All sources are here: https://github.com/nealcrook/nascom/tree/master/m5

The source code is in “m5a.asm” and the script “build” shows how to assemble it and generate a .NAS file.

It would be simple to port the code to another platform; the origin, location for program store and location for variables store are all simple to change. The only area that might require thought is the memory-mapped location used to display the character from the current cursor position.

The code could be placed in ROM.

Change History

This document began as a copy of m5.odt, which attempted to reproduce the Liverpool Software Gazette article describing the original version of M5. This version of the document has been edited to make it somewhat more concise and to include these coding changes:

- Support for NAS-SYS monitors
- New code entry point
- backspace key wreaks havoc in the Edit command so the code was changed so that this key is ignored.

foofoobedoo@gmail.com April 2020.