kMouse Guide
David Hunter 1/3/17

Introduction
kMouse is an interpreted language running on a PIC18 microcontroller. It is a simplified variant (to fit in 1kB) of the programming language *Mouse* created by Peter Grogono in the 1970s. One way to think about it is that *kMouse* is to *Mouse* as *Tiny BASIC* is to *BASIC*.

The goal of *Mouse* was to have a language designed to fit in small memory systems. kMouse was created in that same spirit. More information about *Mouse* can be found at :
http://mouse.davidgsimpson.com/

kMouse uses a serial port for user I/O. The serial port is configured for 57600 bps 8N1. A terminal emulator program such as TeraTerm can be used on a PC to communicate with the PIC18. For testing, a PIC18LF2620 was used. The interpreter is self contained and only needs a program to communicate on a serial port and send a file. Practically any computer with a serial port (or USB with adapter e.g. FTDI cable) can be used.

Command Interface
kMouse has a command line interface which provides two functions: load a program, run a program. The user prompt is a period (.). Invalid commands and errors are indicated by '!' sent to the terminal.
To load a program:

At the prompt type 'L' or 'l' (lower case is allowed).
A response of ':' will appear to indicate the loader is ready. A text file containing the source can then be sent. (in TeraTerm `File->Send File...`)
The loader will accept the source file and compress the program removing comments, blank space, etc. It also fills out a label table with jump positions. (Because of the loader compression, a kMouse program can be written with comments and white space for readability without suffering a performance penalty.)
While loading, a series of asterisks (*) will be displayed to show progress.
When the loader is complete, the prompt (.) will return.

To run a program:

At the prompt, type 'G' or 'g'. (for “go”)
A new line will be sent and the program will run. If a program has not been loaded, a '!' is printed and the prompt returned. If a label is undefined, a '!' will also be printed.
When the program reaches the end command (see below), it will return to the command prompt.
To stop a program while it is running send a control-C (^C) and the command prompt will return.

The interpreter only operates on a single program. When a program is loaded, it overwrites the previous program.

Syntax
The syntax of kMouse uses single characters for commands and hexadecimal numbers. The syntax is explained below.

kMouse is a stack based language (like Forth) so all operations are post fix. This is shown below in the syntax and examples. The stack can hold up to 16 words (16 bit values)

There are 26 variables which are lower case letters (a – z)
There are 26 labels which are upper case letters (A -Z)
All numbers are in hexadecimal and **must** be preceded with an '&'. (The interpreter can handle different length values. i.e. there can be 1 to 4 characters in a number)

The program has a defined format. There is a separation between the main program and the macro (subroutine) definitions. The '%' symbol indicates the program stop. A set of two '$'s indicates the end of the source code. The format is:

< Main program >
%
< Macro (subroutine) Definitions >
$$

Operation	**Syntax**	**Example**	**Top of Stack Value**
Assign a value to a variable	:	&1234 a :	< no change >
Write a byte to a register	|	&55 &0F80 |	< no change >
Put the contents of a variable on the stack	.	a .	<contents of variable 'a'>
Read a byte, place 16 bit value on stack	,	&0F80 ,	<contents of register &0F80
Add the top two values , leave result on stack	+	&12 &34 +	&0046
Subtract the top two values, leave result on stack	-	&46 &34 -	&0012
Put the value of an ASCII character on the stack	'	'c	&0043
Conditional Equal	=	&5 &6 =	0 if False, 1 if True
Conditional Less Than	<	&5 &6 <	0 if False, 1 if True
Conditional Greater Than	>	&5 &6 >	0 if False, 1 if True
If / Endif if value on stack is > 0, execute between [and] if value on stack is ≤ 0, skip to after]	[...]	&1 [&1 +]	< no change >
Label	$<UC>	$A	< no change >
Branch (go to) a label	}	}A	< no change >
Call a Macro	#	#A	< no change >
Return from a Macro	@	@	< no change >
Stop the program and return to the command line	%	%	< no change >
End of program	$$	$$	< no change >
Comment (ignore until end of line)	~	~ comment	< no change >
Print string between double quotes. If a '!' is present, execute a new line (CR/LF)	" ... "	" a string !and another"	< no change >
Input a word from the terminal and place on stack	?	?	< hexadecimal value >
Input a character from the terminal and place ASCII value on stack	?'	?'	< ASCII value (16 bits) >
Print the top of the stack in Hexadecimal	!	&1234 !	< no change >
Print the character that corresponds to the ASCII value on the stack	!'	&55 !'	< no change >

Example Program

This is an example program as well as a pseudo-Tiny BASIC equivalent to show the language features.

kMouse

```
~ kMouse Demo Program
"Hackaday 1kB Challenge"
"!Enter Hex Value for A: " ? a :
"!Enter Hex Value for B: " ? b :
"!A + B = " #A
"!A - B = " #B
"!A = B = " #C
"!A < B = " #D
"!A > B = " #E
"!Enter a character: " ?' c : #F

"!LED Test!"
&FE &0F92 |  ~ initialize A.0 as output

&0B k:   ~ number of loops + 1
&500 t:   ~ delay time
~ LED blink test
$K
t. i:   ~ counter = i
$L
i. &1 - i:
i. &0 > [}L]        ~ delay before on
i. &0 = [#G " on!"]

t. i:
$M
i. &1 - i:
i. &0 > [}M]        ~ delay before off
i. &0 = [#H " off!"]

k. &1- k:
k. &0 > [}K]        ~ outer loop

"Port B = " &0F81 , !

%      ~ stop program
~ macro definitions are always after program
$A a. b. + ! @
$B a. b. - ! @
$C a. b. = ["T"] a. b. = &0 = ["F"] @
$D a. b. < ["T"] a. b. < &0 = ["F"] @
$E a. b. > ["T"] a. b. > &0 = ["F"] @
$F "!The ASCII value of " c. !' " is " c. ! @
$G &1 &0F80 | @  ~ LED on  (PORTA.0 = 1)
$H &0 &0F80 | @  ~ LED off (PORTA.0 = 0)
$$
```

Tiny BASIC

```
1 REM kMouse Demo Program
2 PRINT "Hackaday 1kB Challenge"
3 INPUT "Enter Hex Value for A: ", A
4 INPUT "Enter Hex Value for B: ", B
5 PRINT "A + B = " : GOSUB 100
6 PRINT "A - B = " : GOSUB 110
7 PRINT "A = B = " : GOSUB 120
8 PRINT "A < B = " : GOSUB 130
9 PRINT "A > B = " : GOSUB 140
10 INPUT "Enter a character: ", L$ :
11 LET C = CHR$(L): GOSUB 150
12 PRINT "LED Test"
13 REM initialize A.0 as output
14 POKE(&0F92,&FE)
15 REM number of loops + 1
16 LET K = 11
17 REM delay time
18 LET T = &500
19 REM LED blink test
20 REM counter = I
21 LET I = T
22 LET I = I - 1
23 IF I > 0 GOTO 22
24 REM delay before on
25 IF I = 0 GOSUB 160 : PRINT " on"
26 LET I = T
27 LET I = I - 1
28 IF I > 0 GOTO 27
29 REM delay before off
30 IF I = 0 GOSUB 170 : PRINT " off"
31 LET K = K - 1
33 REM outer loop
33 IF K > 0 GOTO 20
34 PRINT "Port B = ",PEEK(&0F81)
35 REM stop program
36 STOP
100 PRINT A + B : RETURN
110 PRINT A - B : RETURN
120 IF A = B PRINT "T" : RETURN
121 IF A <> B PRINT "F" : RETURN
130 IF A < B PRINT "T" : RETURN
131 IF A >= B PRINT "F" : RETURN
140 IF A > B PRINT "T" : RETURN
141 IF A <= B PRINT "F" : RETURN
150 PRINT "The ASCII value of ",ASC$(C),
151 PRINT " is ",C
159 REM LED on  (PORTA.0 = 1)
160 POKE(&0F80,1)
169 REM LED off (PORTA.0 = 0)
170 POKE(&0F80,0)
```