Micromite MMBasic Version 5.4 Quick Reference

Communications

Serial:

```
OPEN comspec$ AS #fnbr
Conspec$:  "COMn: baud, buf, int, intlevel, DE,
        9BIT, INV, OC, S2"
CLOSE [#]nbr [,[#]nbr] …
EOF( [#]nbr )
INPUT ["prompt string$";] list of variables
LINE INPUT #nbr, string variable$
PRINT #nbr,  expression1 [, expression2, …]
INKEY$
INPUT$(nbr,  [#]fnbr)
LOC( [#]fnbr )
LOF( [#]fnbr )
```

I2C:

```
I2C OPEN speed, timeout [, PU]
I2C WRITE addr, option, sendlen, senddata
        [,sendata ....]
I2C READ addr, option, rcvlen, rcvbuf
I2C CLOSE
I2C SLAVE OPEN addr, mask, opt, i_send, i_rcv
I2C SLAVE WRITE len, data [, data ....]
I2C SLAVE READ len, buf, rcvd
I2C SLAVE CLOSE
```

1-Wire:

```
ONEWIRE RESET pin
ONEWIRE WRITE pin, flag, len, data, …
ONEWIRE READ pin, flag, len, data, …
```

SPI:

```
SPI OPEN speed, mode, bits
received_data = SPI(data_to_send)

SPI WRITE nbr, data1, data2, data3, … etc
or
SPI WRITE nbr, string$
or
SPI WRITE nbr, array()
SPI READ nbr, array()
SPI CLOSE
```

Input/Output

```
IR dev, key , int
IR CLOSE
KEYPAD  var, int, r1, r2, r3, r4, c1, c2, c3 , c4
KEYPAD  CLOSE
LCD INIT d4, d5, d6, d7, rs, en
LCD line, pos, text$
LCD CLEAR | CLOSE
LCD CMD d1 [, d2 [, etc]]
LCD DATA d1 [, d2 [, etc]]
PIN( pin ) = value
PORT(start, nbr [,start, nbr]…) = value
PULSE pin, width
PWM channel, freq, pwm1 [, pwm2 [, pwm3]]
PWM channel, STOP
RTC GETTIME
RTC SETTIME yr, mth, day, hr, min, sec
RTC SETREG reg, value
RTC GETREG reg, var
SERVO channel [, freq], out1 [, out2 [, out3]]
SERVO channel, STOP

SETPIN pin, cfg [, option]
cfg = OFF | AIN | DIN | FIN | PIN | CIN | DOUT
option = PULLUP | PULLDOWN | OC | nbr

SETPIN pin, cfg, target [, option]
cfg = OFF | INTH | INTL | INTB
option = PULLUP | PULLDOWN

TEMPR START pin [, precision]
```

Functions:

```
value = PIN(pin)
Value = PORT(start, nbr [,start, nbr]…)
Temperature = TEMPR( pin )
Pulsewidth = PULSIN( pin, polarity [, t1 [, t2]])
```

System

MM.VER	MM.DEVICE$
MM.ERRNO	MM.ERRMSG$
MM.HRES	MM.VRES
MM.FONTHEIGHT	MM.FONTWIDTH
MM.WATCHDOG	
MM.I2C	
MM.ONEWIRE	

Program Management

```
CONTINUE
CPU speed
CPU SLEEP [ sec [, abortpin]]
CPU RESTART
EDIT
CSUB name(type [, type]) rtype
  END CSUB
CFUNCTION name type [,type] [AS type]
  END CFUNCTION
LIBRARY SAVE | DELETE | LIST
LIST  [ALL]
MEMORY
NEW
POKE BYTE | WORD |VAR | VARTBL, addr, dat
RUN
TIME$ = "HH:MM:SS" | "HH:MM" | "HH"
TIMER = msec
TRACE ON | OFF | LIST nn
VAR SAVE var [, var]…
VAR RESTORE | CLEAR
WATCHDOG timeout | OFF
XMODEM SEND | RECEIVE

nbr = PEEK(BYTE | WORD | VARADDR |
        CFUNADDR | VAR | VARTBL |
        PROGMEM,  args)
```

Variables

```
FLOAT:  var!
INTEGER:  var%
STRING:  var$
Maximun 32 chars.
Must start with A to Z or _ (underscore)
Name can include A to Z, 0 to 9, period or "_"
Constants prefix: nbr (decimal), &H (hex) . &O
(octal) or &B (binary)
```

BASIC Language

```
‘ (single quotation mark) - comment
? (question mark) – shorthand for PRINT
CLEAR
CONST id1 = expression [,  id2 = expression, …]
CONTINUE –or- CONTINUE DO | FOR
DATA constant[,constant]...
DATE$ = "DD-MM-YY" | "DD/MM/YY"
DEFINEFONT #Nbr - END DEFINEFONT
DIM [type] var [, var, ...]
DIM var AS type [, var AS type , ...]
DO - LOOP
DO [WHILE <test>]  < stmts> LOOP
DO  < stmts>  LOOP UNTIL <test>
END –or- END FUNCTION | SUB
ERASE variable [,variable]...
ERROR [error_msg$]
EXIT DO | FOR | FUNCTION | SUB
FOR var = start TO finish [STEP increment]
   NEXT [var1 [, var2, …]
FUNCTION name (arg1 [,arg2, …]) [AS <type>]
   END FUNCTION
GOSUB target   -and-  RETURN
GOTO target
IF <test> THEN <stmt> ELSE <stmts>
IF <test> THEN | ELSEIF | ELSE | ENDIF
INPUT #nbr, var1 [, var2, …]
LET variable = expression (LET is optional)
LOCAL [type] decl [, decl, ...]
ON ERROR ABORT | IGNORE | SKIP [nn]
ON ERROR CLEAR
ON nbr GOTO | GOSUB t1 [, t3, ...]
ON KEY subroutine
PAUSE ms
PRINT expression1 [, expression2, …]
RANDOMIZE nbr
READ var1[, var2, ...]  -and-  RESTORE [line]
REM comment
SELECT CASE
     CASE [ELSE]
  END SELECT
SETTICK period, target [, nbr]
SUB name arg1 [, arg2, …]
   END SUB
```

Operators

NOT	Logical inverse
^	Exponentiation
*	Multiply
/ \	Division (float and integer)
MOD	Modulus (remainder)
+ -	Addition and subtraction
x << y	Shift bits up
x << y	Shift bits down
<>	Not equals
< >	Less/greater than
<= >=	Less/greater than or equals
AND OR	Logical and/or
XOR	Exclusive or

BASIC Functions

ACOS(number)	ABS(number)
ASC(string$)	ASIN(number)
ATN(number)	BIN$(number [, chars])
CHR$(number)	CINT(number)
COS(number)	DATE$
DEG(radians)	EXP(number)
FIELD$(str$, field, delim$)	
FIX(number)	
HEX$(number [, chars])	
INSTR([start,] str$, pattern$)	
INT(number)	LEFT$(string$, nbr)
LEN(string$)	LOG(number)
LCASE$(string$)	MAX(arg1 [, arg2 [, ..]])
MIN(arg1 [, arg2 [, …]])	
MID$(string$, start [, nbr])	
OCT$(number [, chars])	
PI	
POS	
RAD(degrees)	
RIGHT$(string$, number-of-chars)	
RND(number)	SGN(number)
SIN(number)	SPACE$(number)
SQR(number)	
STR$(number [, m [, n [, c$]]])	
STRING$(nbr, ascii \| string$)	
TAB(number)	TAN(number)
TIME$	TIMER
UCASE$(string$)	VAL(string$)

Options

```
OPTION AUTORUN OFF | ON
OPTION BASE 0 |  1
OPTION BAUDRATE nbr
OPTION BREAK nn
OPTION CASE UPPER | LOWER | TITLE
OPTION CLOCKTRIM ±n
OPTION COLOURCODE ON | OFF
OPTION CONSOLE ECHO | NOECHO
OPTION CONSOLE INVERT | NOINVERT
OPTION CONSOLE AUTO
OPTION DEFAULT FLOAT | INTEGER |
                              STRING | NONE
OPTION DISPLAY lines [,chars]
OPTION EXPLICIT
OPTION LIST
OPTION PIN nbr
OPTION RESET          OPTION TAB 2 | 4 | 8
```

LCD Display Panel

```
OPTION LCDPANEL ILI9341, orientation, D/C
      pin, reset pin [,CS pin]
OPTION LCDPANEL DISABLE
OPTION TOUCH T_CS pin, T_IRQ pin
OPTION TOUCH DISABLE

BOX  x1,  y1,  w,  h  [, lw]  [,c] [,fill]
CIRCLE  x,  y,  r  [,lw]  [, a]  [, c]  [, fill]
CLS [colour]
COLOUR fore [, back]
COLOR fore [, back]
FONT [#]font-number, scaling
GUI BITMAP  x,  y,  b  [, w]  [, h]  [, s]  [, c]  [, bc]
GUI CALIBRATE
GUI RESET LCDPANEL
GUI TEST LCDPANEL | TEST TOUCH
LINE  x1, y1, x2, y2 [, LW [, C]]
PIXEL  x,  y  [,c]
RBOX  x1,  y1,  w,  h  [, r]  [,c] [,fill]
TEXT  x,  y,  str$  [,just$]  [, fnt]  [, scale]  [,
      colour]  [, bc]

Colour = RGB(red, green, blue)
Colour = RGB(shortcut)
XCord = TOUCH(X)
YCord = TOUCH(Y)
```