Keywords

THEN, ELSE, GOTO, GOSUB, TO, STEP, FOR, WHILE, UNTIL, MOD, NOT, AND, OR, XOR

System Variables

MM.HRES Horizontal Screen Resolution
MM.VRES Vertical Screen Resolution
MM.VER Firmware Version
MM.ERRNO Last Error Code (See Codes)
MM.I2C Last I2C Result Code (See Codes)

User Variables

Variable names can start with an alphabetic character or underscore and can contain any alphabetic or numeric character, period (.) and underscore (_); maximum 32 characters in length.
String variable names are terminated with a $ symbol
Number variable names are terminated with a $ symbol

Literals

Strings are contained in double quotes, e.g. "Maximite Basic"
Numbers may be decimal or represented as:
&Hnn Hexadecimal Literal, e.g. &H3C (60 decimal)
&Bnn... Binary Literal, e.g. &B00100011 (35 decimal)
n.nE+n Scientific notation, e.g. 1.6E+4 (16000 decimal)..

Arithmetic Operators

^ * / Exponentiation, Multiplication, Division
MOD \ Modulus (remainder), Integer Division
+ + - Addition, String Concatenation, Subtraction

Logical Operators

NOT Logical inverse
<> Inequality
< Less Than
> Greater Than
<= or =< Less Than Equal To
>= or => Greater Than Equal To
= Equality
AND , OR, XOR Conjunction, Disjunction, Exclusive OR

Error Codes

0 = No error
1 = No SD card found
2 = SD card is write protected
3 = Not enough space
4 = All root directory entries are taken
5 = Invalid filename
6 = Cannot find file
7 = Cannot find directory
8 = File is read only
9 = Cannot open file
10 = Error reading from file
11 = Error writing to file
12 = Not a file
13 = Not a directory
15 = Directory not empty
15 = Hardware error accessing the storage media

I2C Codes

0 = No error.
1 = Received a NACK response
2 = Command timed out
4 = Received a general call address (when in slave mode)

Pin Configuration Codes

0 Not configured or inactive
1 Analog input (pins 1 to 10)
2 Digital input (all pins and 5V tolerant on pins 11 to 20)
3 Frequency input (pins 11 to 14)
4 Period input (pins 11 to 14)
5 Counting input (pins 11 to 14)
6 Interrupt on low to high input change (all pins)
7 Interrupt on high to low input change (all pins)
8 Digital output (all pins)
9 Open collector digital output to 5V (pins 11 to 20)

Format String Codes

% [flags] [width] [.precision] type
Where 'flags' can be **-** Left justify; **0** Use 0 for the pad character instead of space; **+** Forces a + sign to be shown for positive values.
space Display space as sign, unless val is negative
'width' - min. chars to output, less than this causes padding, more than this the width will be expanded.
'precision' specifies the no. of fraction digits to generate with an e, or f type or the max. no of significant digits to generate with a g type.
If specified, precision must be preceded by a dot (.).
'type' can be one of: **g** format for the best presentation; f Format with the decimal point and following digits; e Format the no. in exponential format; **G** or **F** then the exponential output will use an uppercase E.
If the format specification is not specified "%g" is assumed.

DRAFT

Statements

Miscellaneous

' Comment
COPYRIGHT
DATE$
ERROR [error_msg$]
MEMORY
OPTION BASE 0 | 1
OPTION ERROR CONTINUE
OPTION ERROR ABORT
OPTION PROMPT str$
POKE hiword, loword, val
RANDOMIZE nbr
REM string
RUN [line] [file$]
NEW
SETTICK period, line
TIME$ = "HH:MM:SS"
TIMER = msec
TROFF
TRON
Assignment
CLEAR
DATA
DIM variable(elements...)
ERASE variable
LET variable =
READ variable[, variable]...
RESTORE

I/O - File System

CHDIR dir$
CLOSE [#]nbr [,[#]nbr]
FILES [search_pattern$]
INPUT #nbr, list of variables
KILL file$
LINE INPUT #nbr, string-variable$
LOAD file$
MERGE file$
MKDIR dir$
NAME old$ AS new$
OPEN fname$ FOR mode AS [#]fnbr
OPEN comspec AS [#]fnbr
PRINT #nbr, expression [[,;]expression]...
RMDIR dir$
SAVE file$
SAVEBMP file$
WRITE [#nbr,] expression [,expression] .

I/O - Keyboard

INPUT ["prompt string";]
LINE INPUT [prompt$,]

I/O - Screen

CLS
CIRCLE (x, y) ,r [,c [,F]]
LINE [(x1 , y1)] - (x2, y2) [,c [,B[F]]]
LOCATE x, y
PIXEL(x,y)
PRINT / ? expression [[,;]expression]...
PRESET (x, y)
PSET (x, y)

I/O - Serial

CLOSE CONSOLE
OPEN COMSPEC as CONSOLE
Sound/PWM
SOUND frequency, duration
SOUND frequency, duration, dutycycle

I/O - Pins

PIN(pin) = value
SETPIN pin, cfg
SETPIN pin, cfg, line

Flow Control

CONTINUE
DO <statements> LOOP
DO WHILE expression <statements> LOOP
DO <statements> LOOP UNTIL expression
ELSE
ELSEIF expression THEN
ENDIF
END
EXIT
EXIT FOR
FOR counter = start TO finish [STEP increment]
GOSUB
GOTO
IF expression THEN
IRETURN
NEXT [counter-variable] [, counter-variable]...
ON variable GOTO|GOSUB line[,line,line,...]
PAUSE nbr
RETURN
WHILE expression... <statements> WEND

Editor

DELETE line
DELETE -lastline
DELETE firstline -
DELETE firstline - lastline
EDIT [line-number]
LIST
LIST line
LIST -lastline
LIST firstline -
LIST firstline - lastline
RENUMBER
RENUMBER first
RENUMBER first, incr
RENUMBER first, incr, start

DRAFT

Functions

ABS(nbr) Absolute value of 'nbr'.
ASC(str$) ASCII code for the 1st char in 'str$'.
ATN(nbr) Arctangent of 'nbr' in radians.
CHR$(nbr) Character of the ASCII code 'nbr'.
CINT(nbr) Round up/down to the next integer.
COS(nbr) Cosine of 'nbr' in radians.
CWD$ Current working directory on the SD card.
DATE$ Current date as a string - "DD-MM-YYYY".
EOF([#]nbr) True if file 'nbr' is at the end of the file.
EXP(nbr) Exponential value of 'nbr'.
FIX(nbr) Truncate to a whole number.
FORMAT$(nbr[,fmt$]) 'nbr' as formatted string.
HEX$(nbr) Base 16 value of 'nbr' as string.
INKEY$ Single character of keyboard status (or "").
INPUT$(nbr,[#]fnbr) 'nbr' characters read from open file
INSTR([start,]srch$,ptn$) Position, from start, where ptn$ occurs in srch$.
INT(nbr) Next whole number <= 'nbr'.
LEFT$(str$,nbr) Substring of 'nbr' chars from start of str$.
LEN(str$) No. of characters in str$.
LOC([#]nbr) No. of bytes in serial port receive buffer.
LOF([#]nbr) Bytes left in serial port transmit buffer.
LOG(nbr) Natural logarithm of 'nbr'.
LCASE$(str$) 'str$' converted to lowercase characters.
MID$(str$, start[,nbr]) Returns a substring of 'str$' from 'start' for 'nbr' bytes (or end if 'nbr' omitted).
OCT$(nbr) Returns a string giving the octal (base 8) representation of 'nbr'.
PEEK(hiword,loword) Will return a byte within the PIC32 virtual memory space.
PIN(pin) Returns the value on the external I/O 'pin'.
POS Returns the current cursor position in the line.
PIXEL(x,y) Returns the value of a pixel on the VGA or composite screen. 0=off, 1=on.
RIGHT$(str$, nbrof- chars) Returns a substring of 'str$' with 'nbr-of-chars' from the right (end) of the string.
RND(nbr) Returns a pseudo random nbr in the range of 0 to 0.99999. The 'nbr' value is ignored if supplied.
SGN(nbr) Returns the sign of 'nbr', +1 for positive, 0 for 0, and -1 for negative.
SIN(nbr) Returns the sine of 'nbr' in radians.
SPACE$(nbr) Returns a string of blank spaces 'nbr' bytes long.
SPC(nbr) Returns a string of blank spaces 'nbr' bytes long.
SPI(rx,tx,clk[,data[,speed]]) Sends and receives a byte using the SPI protocol (Maximite is master)
SQR(nbr) Returns the square root of 'nbr'.
STR$(nbr) Returns a string in the decimal (base 10) representation of the argument 'nbr'.
STRING$(nbr, val|str$) Returns a string 'nbr' bytes long of either 1st char of str$ or ASCII char of val.
TAB(nbr) Outputs spaces until the column indicated by 'nbr' has been reached.
TAN(nbr) Returns the tangent of the argument 'nbr' in radians.
TIME$ Returns current time as a "HH:MM:SS" in 24 hour notation.
TIMER Returns elapsed time in milliseconds (eg, 1/1000 of a second) since reset.
UCASE$(str$) Returns 'str$' converted to uppercase characters.
VAL(str$) Returns the numerical value of the 'str$'.

DRAFT