MMBasic Version 2.7
Code Help Snippets R1

by

Bryan1

(reformatted by DuinoMiteMegaAndy)

3/12/2011

MMBasic command code snippets in alphabetical sort

AUTO
CHDIR
CIRCLE
CLEAR
CLOSE
CLS
CLS_CON
CONTINUE
CY_RIGHT
DATA
DATE
DELETE
DIM
DO
EDIT
'ELSE
ELSEIF
END
ENDIF
ERASE
ERROR
EXIT
FILES
FONT
FOR
GOSUB
GOTO
HELP
IF
INPUT
IRETURN
KILL
LET
LINE
LINE_INP
LIST
LOAD
LOCATE
LOOP
MEMORY
MERGE
MKDIR
NAME
NEW
NEXT
ON

OPEN
OPEN_COM
OPTION
PAUSE
PIN
PIXEL
POKE
PRINT
RANDOM
READ
REM
RENUMBER
RESTORE
RETURN
RUN
SAVE
SAVEBMP
SETPIN
SETTICK
SOUND
TIME
TIMER
TROFF
TRON
XMODEM

Maximite Quick Reference – v2.7

Literals / Variables

Literals

Strings are contained in double quotes, e.g. "Maximite".
Numbers may be decimal or represented as:

&Hnn	Hex Literal, e.g. &H3C (60)
&Bnn...	Binary Literal, e.g. &B00100011 (35)
n.nE+n	Scientific, e.g. 1.6E+4 (16000)

System Variables

MM.HRES	Horizontal Screen Resolution
MM.VRES	Vertical Screen Resolution
MM.VER	Firmware Version
MM.DRIVE$	Current Default Drive
MM.FNAME$	Current Default File
MM.ERRNO	Last Error Code
MM.I2C	Last I2C Result Code

User Variables

Variable names start with an alpha character or underscore and can contain any alpha or numeric character, period (.) and underscore (_); maximum length is 32 characters. String variable names are terminated with a $ symbol. Number variable names are not terminated with a $ symbol.

Operators

Arithmetic

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

Logical

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

Codes

Error

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	File does not exist
7	Directory does not exist
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 / Hardware error

Format String

% [flags] [width] [.prec] type

flags:	- Left justify
	0 Use 0 for the pad char, not space.
	+ A plus sign is shown for positive values.
	space Space as sign, unless negative.
width:	min. chars to output, less than this causes padding, more than this causes expansion.
prec:	no. of fraction digits for e, or f type, or the max. no of significant digits for g type. Must be preceded by a dot (.) if used.
type:	**g** format for the best presentation
	f format with decimal point and digits
	e Format in exponential format
	G exponential output with uppercase E
	F exponential output with uppercase E.

If the format specification is not specified "%g" is assumed.

I2C

0	No error
1	Received NACK response
2	Command timed out
4	Received general call addr. (slave mode)

File Open Mode

INPUT Read Only
OUTPUT Write (Overwrite if exists)
APPEND Write (Append if exists)

Commands / Statements

Assignment

CLEAR
DATA
DIM variable(elements...)
ERASE variable
LET variable =
READ variable[, variable]...
RESTORE

Editor

AUTO [start] [, increment]
DELETE line
DELETE -lastline
DELETE firstline [- lastline]
EDIT [line-number] *(extended editing mode)*
LIST [line]
LIST -lastline
LIST firstline [- lastline]
RENUMBER [first] [,incr] [,start]

External Pins

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

File System

CHDIR dir$
CLOSE [#]nbr [,[#]nbr]
CLOSE CONSOLE
DRIVE drive$
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
? or PRINT #nbr, expression [[,;]expression]...
RMDIR dir$
SAVE [file$]
SAVEBMP file$
~~WRITE [#nbr,] expression [,expression]~~

Flow Control

CONTINUE
DO <statements> LOOP
DO WHILE expression <statements> LOOP
DO <statements> LOOP UNTIL expression
ELSE
ELSEIF expression THEN
ENDIF

crackerjack – 10/2011

Maximite Quick Reference – v2.7

Literals / Variables

Literals

Strings are contained in double quotes, e.g. "Maximite".
Numbers may be decimal or represented as:

&Hnn	Hex Literal, e.g. &H3C (60)
&Bnn...	Binary Literal, e.g. &B00100011 (35)
n.nE+n	Scientific, e.g. 1.6E+4 (16000)

System Variables

MM.HRES	Horizontal Screen Resolution
MM.VRES	Vertical Screen Resolution
MM.VER	Firmware Version
MM.DRIVE$	Current Default Drive
MM.FNAME$	Current Default File
MM.ERRNO	Last Error Code
MM.I2C	Last I2C Result Code

User Variables

Variable names start with an alpha character or underscore and can contain any alpha or numeric character, period (.) and underscore (_); maximum length is 32 characters. String variable names are terminated with a $ symbol. Number variable names are not terminated with a $ symbol.

Operators

Arithmetic

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

Logical

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

Codes

Error

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	File does not exist
7	Directory does not exist
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 / Hardware error

Format String

% [flags] [width] [.prec] type

flags:	- Left justify
	0 Use 0 for the pad char, not space.
	+ A plus sign is shown for positive values.
	space Space as sign, unless negative.
width:	min. chars to output, less than this causes padding, more than this causes expansion.
prec:	no. of fraction digits for **e**, or **f** type, or the max. no of significant digits for **g** type. Must be preceded by a dot (.) if used.
type:	**g** format for the best presentation
	f format with decimal point and digits
	e Format in exponential format
	G exponential output with uppercase E
	F exponential output with uppercase E.

If the format specification is not specified "%g" is assumed.

I2C

0	No error
1	Received NACK response
2	Command timed out
4	Received general call addr. (slave mode)

File Open Mode

INPUT	Read Only
OUTPUT	Write (Overwrite if exists)
APPEND	Write (Append if exists)

Commands / Statements

Assignment

CLEAR
DATA
DIM variable(elements...)
ERASE variable
LET variable =
READ variable[, variable]...
RESTORE

Editor

AUTO [start] [, increment]
DELETE line
DELETE -lastline
DELETE firstline [- lastline]
EDIT [line-number] *(extended editing mode)*
LIST [line]
LIST -lastline
LIST firstline [- lastline]
RENUMBER [first] [,incr] [,start]

External Pins

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

File System

CHDIR dir$
CLOSE [#]nbr [,[#]nbr]
CLOSE CONSOLE
DRIVE drive$
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
? *or* PRINT #nbr, expression [[,;]expression]...
RMDIR dir$
SAVE [file$]
SAVEBMP file$
~~WRITE [#nbr,] expression [,expression]~~

Flow Control

CONTINUE
DO <statements> LOOP
DO WHILE expression <statements> LOOP
DO <statements> LOOP UNTIL expression
ELSE
ELSEIF expression THEN
ENDIF

crackerjack – 10/2011

```
--------
Auto
--------
4000 CLS
4001 FONT#2,1
4002 PRINT TAB(10); "The Auto Command "
4003 PRINT
4004 FONT#1,1
4005 PRINT TAB(5); " There are three ways to use the 'auto' command. "
4006 PRINT
4007 PRINT TAB(5); "AUTO or AUTO start or AUTO start,increment"
4008 PRINT
4009 PRINT TAB(5); "Enter automatic line entry mode. 'start' is the line number to start"
4010 PRINT TAB(5); "numbering at and 'increment' is the increment for each new line."
4011 PRINT TAB(5); "Both are optional and default to 10 if the command has not"
4012 PRINT TAB(5); "previously been used or the state of the last used AUTO command"
4013 PRINT TAB(5); "when it was terminated"
4014 PRINT
4015 PRINT TAB(5); "When in automatic line entry mode the next line number will be"
4016 PRINT TAB(5); "automatically generated and the cursor placed after it ready for"
4017 PRINT TAB(5); "a program line to be entered. When ENTER is pressed the line will"
4018 PRINT TAB(5); "be saved to memory and the next number generated."
4019 PRINT
4020 PRINT TAB(5); "If the line number is the same as an existing line in memory"
4021 PRINT TAB(5); "it will be preceded by an asterisk(*). In this case pressing"
4022 PRINT TAB(5); "ENTER without entering any text will preserve the line in"
4023 PRINT TAB(5); "memory and generate the next number."
4024 PRINT
4025 PRINT TAB(5); " To terminate AUTO Mode use CONTROL-C."
4026 PRINT: PRINT: PRINT: PRINT: PRINT:PRINT
4045 PRINT "Press `ESC' to return to the search";
4046 a$=INKEY$:IF a$="" THEN GOTO 4046
4047 IF a$=CHR$(27) THEN RUN"help"
4049 '-----------------------------------------------------
```

```
---------
CHDIR
---------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The CHDIR command "
40 PRINT
50 FONT#1,1
60 PRINT TAB(5); " The CHDIR dir$ command changes the working directory on"
70 PRINT TAB(5); "SD card to 'dir$'. The special entry '..' represents the"
80 PRINT TAB(5); "parent of the current directory and '.' represents the current directory"
90 PRINT
100 FONT#2,1
110 PRINT TAB(14); " EXAMPLE "
120 FONT#1,1
130 print:print
140 print tab(8); "\> chdir'help'"
150 print tab(8); "\HELP>"
160 PRINT: PRINT: PRINT: PRINT: PRINT: PRINT
170 PRINT "Press `ESC' to return to the search";
180 a$=INKEY$:IF a$="" THEN GOTO 180
190 IF a$=CHR$(27) THEN RUN"help"
```

```
----------
CIRCLE
----------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The CIRCLE Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The CIRCLE command has the following prefix CIRCLE(x and y),r[,c[,F]]"
70 PRINT
80 PRINT TAB(5); "The circle is drawn on the Video output centered at x and y"
90 PRINT TAB(5); "with a radius of r. If c is zero the pixels are turned off"
100 PRINT TAB(5); "if it is non zero or not specified the pixels are turned on"
110 PRINT TAB(5); "(IE:- the circle is drawn). The F option will cause the circle"
120 PRINT TAB(5); "to be filled according to the c parameter."
130 PRINT
140 PRINT TAB(5); " NOTE:- That because the pixels are not exactly square"
150 PRINT TAB(5); "the circle will be oval to some degree."
160 PRINT
170 FONT#2,1
180 FONT#1,1
190 PRINT: PRINT:PRINT: PRINT: PRINT: PRINT
200 PRINT "Press `ESC' to return to the search";
210 a$=INKEY$:IF a$="" THEN GOTO 210
220 IF a$=CHR$(27) THEN RUN"help"
```

CLS

```
4700 CLS
4701 FONT#2,1
4702 PRINT TAB(10); " The CLEAR Command"
4703 FONT#1,1
4704 PRINT
4705 PRINT TAB(5); " The CLEAR command will delete all variables and recover"
4706 PRINT TAB(5); "the memory used by them."
4707 PRINT
4708 PRINT TAB(5); " Also see the ERASE command for deleting specific array variables"
4709 PRINT: PRINT: PRINT
4710 PRINT "Press `ESC' to return to the search";
4711 a$=INKEY$:IF a$="" THEN GOTO 4711
4718 IF a$=CHR$(27) THEN RUN "help"
```

CLOSE

```
4709 CLS
4710 FONT#2,1
4711 PRINT TAB(10); "The CLOSE Command "
4712 FONT#1,1
4713 PRINT
4714 PRINT TAB(5); " The CLOSE command has the variable CLOSE[*][,[*]nbr]"
4715 PRINT
4716 PRINT TAB(5); " The close command will close the file(s) or serial port(s)"
4717 PRINT TAB(5); "previously opened with the file number 'nbr'. The [*] is optional"
4718 PRINT
4719 PRINT TAB(5); " Also see the OPEN command for further reference"
4720 PRINT: PRINT: PRINT
4721 PRINT "Press `ESC' to return to the search";
4722 a$=INKEY$:IF a$="" THEN GOTO 4722
4723 IF a$=CHR$(27) THEN RUN "help"
```

CLS

```
4207 CLS
4208 FONT#2,1
4209 PRINT TAB(7); " The CLS Command "
4210 FONT#1,1
4211 PRINT
4212 PRINT TAB(5); "The CLS command will clear the Video display and place the curosr"
4213 PRINT TAB(5); "in the top left hand corner of the screen."
4214 PRINT: PRINT: PRINT
4215 PRINT "Press `ESC' to return to the search";
4216 a$=INKEY$:IF a$="" THEN GOTO 4216
4217 IF a$=CHR$(27) THEN RUN"help"
```

```
--------------
CLR_CON
--------------
4200 CLS
4201 FONT#2,1
4202 PRINT TAB(7); " The CLOSE CONSOLE Command"
4203 FONT#1,1
4204 PRINT
4205 PRINT TAB(5); " The CLOSE CONSOLE command will close a serial port that"
4206 PRINT TAB(5); "been opened as the console."
4207 PRINT: PRINT:PRINT
4208 PRINT "Press `ESC' to return to the search";
4209 a$=INKEY$:IF a$="" THEN GOTO 4209
4210 IF a$=CHR$(27) THEN RUN"help"
```

```
---------------
CONTINUE
---------------
4214 CLS
4215 FONT#2,1
4216 PRINT TAB(7); " The CONTINUE Command"
4217 FONT#1,1
4218 PRINT
4219 PRINT TAB(5); " The CONTINUE command will resume a program that has been stopped"
4220 PRINT TAB(5); "by the END statement, an error, or CTRL-C. The program will"
4221 PRINT TAB(5); "restart with the next statement following the previous stopping point"
4222 PRINT: PRINT: PRINT
4223 PRINT "Press `ESC' to return to the search";
4224 a$=INKEY$:IF a$="" THEN GOTO 4224
4225 IF a$=CHR$(27) THEN RUN"help"
```

```
--------------
CY_RIGHT
--------------
4222 CLS
4223 FONT#2,1
4224 PRINT TAB(7); " The COPYRIGHT command"
4225 FONT#1,1
4226 PRINT
4227 PRINT TAB(5); " The COPYRIGHT command will list all the contributors to the"
4228 PRINT TAB(5); " MMBasic and summarize the open source copyright statement"
4229 PRINT: PRINT
4246 PRINT "Press `ESC' to return to the search";
4247 a$=INKEY$:IF a$="" THEN GOTO 4247
4248 IF a$=CHR$(27) THEN RUN"help"
```

```
--------
DATA
--------
4250 CLS
4251 FONT#2,1
4252 PRINT TAB(10); " The DATA Command"
4253 FONT#1,1
4254 PRINT
4255 PRINT TAB(5); "The DATA command stores numerical and string constants to "
4257 PRINT TAB(5); " be accessed by READ. String constants do not need to be"
4258 PRINT TAB(5); "quoted unless they contain significant spaces, the comma"
4259 PRINT TAB(5); "or a keyword (such as THEN,WHILE,etc). Numerical constants"
4260 PRINT TAB(5); "can also be expressions such as 5*60."
4261 PRINT: PRINT: PRINT
4262 PRINT "Press `ESC' to return to the search";
4263 a$=INKEY$:IF a$="" THEN GOTO 4263
4264 IF a$=CHR$(27) THEN RUN"help"
```

```
-------
DATE
-------
4350 CLS
4351 FONT#2,1
4352 PRINT TAB(6); " The TIME and DATE Commands"
4353 PRINT
4354 FONT#1,1
4356 PRINT TAB(6); "In order to set the DATES and TIMES the following commands"
4357 PRINT TAB(6); "are used. The DATES can be set by:-"
4358 PRINT TAB(6); "DATE$='DD-MM-YY' or 'DD/MM/YY' On power up the date will'"
4359 PRINT TAB(6); "to 1-1-2000. "
4360 PRINT
4361 PRINT TAB(6); " The TIME is set in a similar way like TIME$='HH-MM-SS'"
4362 PRINT TAB(6); "The MM and SS are optional and will default to zero if"
4363 PRINT TAB(6); 'not specified. IE:- TIME$='14.30' will set the clock to"
4364 PRINT TAB(6); "14:30 with zero seconds. Time is set to zero on power-up"
4365 PRINT
4366 PRINT TAB(6); " To avoid confusion ' was used instead of the double as"
4367 PRINT TAB(6); "MM basic uses the doubles for programming. Ensure you"
4368 PRINT TAB(6); "the double quotes in your programs."
4369 PRINT: PRINT: PRINT
4396 PRINT "Press `ESC' to return to the search";
4397 a$=INKEY$:IF a$="" THEN GOTO 4397
4398 IF a$=CHR$(27) THEN RUN "help"
```

```
-----------
DELETE
-----------
4400 CLS
4401 FONT#2,1
4402 PRINT TAB(10); " The DELETE Command"
4403 FONT#1,1
4404 PRINT
4405 PRINT TAB(5); " The DELETE command deletes a program line or a range"
4406 PRINT TAB(5); "of lines."
4407 PRINT
4408 PRINT TAB(5); "DELETE line"
4409 PRINT TAB(5); "DELETE -lastline"
4410 PRINT TAB(5); "DELETE firstline-"
4411 PRINT TAB(5); "DELETE firstline-lastline"
4412 PRINT
4413 PRINT TAB(5); "If -lastline is used it will start with the first line"
4415 PRINT TAB(5); "to the end of the program"
4416 PRINT: PRINT: PRINT
4417 PRINT "Press `ESC' to return to the search";
4418 a$=INKEY$:IF a$="" THEN GOTO 4418
4419 IF a$=CHR$(27) THEN RUN "help"
```

```
-----
DIM
-----
4524 CLS
4550 FONT#2,1
4551 PRINT TAB(10); " The DIM Command"
4552 FONT#1,1
4553 PRINT
4554 PRINT TAB(5); " DIM variable(elements....)[variable(elements....)... Specifies variables that have
more than"
4555 PRINT TAB(5); "one element in a single dimension.  ie:- arrayed variables."
4589 PRINT: PRINT: PRINT
4596 PRINT "Press `ESC' to return to the search";
4597 a$=INKEY$:IF a$="" THEN GOTO 4597
4598 IF a$=CHR$(27) THEN RUN "help"
```

DO

```
4500 CLS
4501 FONT#2,1
4502 PRINT TAB(10); " The DO Command"
4503 FONT#1,1
4504 PRINT
4505 PRINT TAB(5); " The DO LOOP command - This structure will loop forever"
4506 PRINT TAB(5); "the EXIT command can be used to terminate the loop or"
4507 PRINT TAB(5); "control must be explicitly transferred outside of the"
4508 PRINT TAB(5); "loop by commands like GOTO and RETURN. "
4509 PRINT
4510 FONT#2,1
4511 PRINT TAB(10); " The DO WHILE LOOP Command"
4512 FONT#1,1
4513 PRINT
4514 PRINT TAB(5); " The DO WHILE LOOP command loops while 'expressions'"
4515 PRINT TAB(5); "is true(this is equivalent to the older WHILE-WEND"
4516 PRINT TAB(5); 'loop, also implemented in MM Basic."
4517 PRINT
4518 FONT#2,1
4519 PRINT TAB(10); " The DO WHILE UNITL Command"
4520 FONT#1,1
4521 PRINT
4522 PRINT TAB(5); " The DO WHILE UNTIL COMMAND will loop until the"
4523 PRINT TAB(5); "expression following the UNTIL is true."
4524 PRINT: PRINT: PRINT
4525 PRINT "Press `ESC' to return to the search";
4526 a$=INKEY$:IF a$="" THEN GOTO 4526
4527 IF a$=CHR$(27) THEN RUN "help"
```

```
------
EDIT
------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The EDIT Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); "EDIT[Line number] EDIT the line number"
70 PRINT
80 PRINT TAB(5); " If no line number is used this command will edit the previous"
90 PRINT TAB(5); "entry typed at the command prompt. If a running running program"
100 PRINT TAB(5); "has just terminated with an error this"will automatically edit"
110 PRINT tab(5); "the line that caused the error."
115 print
120 PRINT TAB(3); "The Editing keys are:-"
130 PRINT TAB(4); "Left/right Arrows      Moves the cursor within the line"
140 PRINT TAB(4); "Home/End               Moves the cursor to the start or the end of the line"
150 PRINT TAB(4); "Delete                 Delete the character over the cursor"
160 PRINT TAB(4); "Backspace              Delete the character before the cursor"
170 PRINT TAB(4); "Insert                 Will switch between insert and overtype mode"
180 PRINT
190 PRINT TAB(5); " Use Enter(or Return) to finish editing(even in insert mode). "
200 PRINT TAB(5); "The line is added to the program just as if it had been typed "
210 PRINT TAB(5); "at the command prompt. If the line number had been changed a "
215 print tab(5); "new(edited) copy of the line will be added to memory, if it is"
220 PRINT TAB(5); "unchanged the line will replace 'line-number'. When editing a "
230 PRINT TAB(5); "program line the up arrow will switch to the previous line and "
240 PRINT TAB(5); "down arrow to the next line number. When doing this any changes"
245 print tab(5); "will be automatically saved (the same as using Enter) before "
246 print tab(5); "moving to the next line."
250 PRINT TAB(3); "Press 'ENTER' to continue"
260 a$=INKEY$:IF a$="" THEN GOTO 260
270 IF a$=CHR$(&H0D) THEN GOTO 280
280 PRINT
290 PRINT TAB(5); "MMBasic is always in edit mode when entering data at the command"
300 PRINT TAB(5); "prompt or for the INPUT and LINE INPUT commands. In these cases"
310 PRINT TAB(5); "the arrow keys can be used to move within the line to correct errors."
315 print tab(5); "If the Up arrow key is pressed at the command prompt it will act"
320 PRINT TAB(5); "the same as EDIT with no line number (edit the last error line or "
330 PRINT TAB(5); "entered command). Subsequent up/down arrow presses will move through "
340 PRINT TAB(5); "the list of recent entries. All the editing keys work with Tera term "
350 PRINT TAB(5); "and Putty(in vt100 mode) so editing can be accomplished over a USB "
360 PRINT TAB(5); "or serial link using these terminal emulators or any other vt100 "
370 PRINT TAB(5); "compatible terminal emulator. The maximum line length that can be "
375 print tab(5); "edited is 79"chars in VGA mode and 49 chars in composite mode. If "
380 PRINT TAB(5); "more than this number of characters are entered in overtype mode "
```

(continued)

```
390 PRINT TAB(5); "MMBasic will automatically enter normal text entry mode without "
395 print tab(5); "the editing functions. If insert mode any extra characters will be rejected"
400 PRINT: PRINT: PRINT
410 PRINT "Press `ESC' to return to the search";
420 a$=INKEY$:IF a$="" THEN GOTO 420
430 IF a$=CHR$(27) THEN RUN "help"
```

```
-------
ELSE
-------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The IF Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The IF expr THEN statement or IF expr THEN statement ELSE statement,
evaluates the"
70 PRINT TAB(5); 'expr' and performs the THEN statement if it is true or skips to the next line if
false."
80 PRINT TAB(5); "The option ELSE statement is the reverse of the THEN test. The 'statement' can be
just"
90 PRINT TAB(5); "a line number and in that case a GOTO is assumed. For Microsoft compatibility
the"
100 PRINT TAB(5); "'THEN statement' construct can be also replaced with a 'GOTO line number."
110 PRINT TAB(5); "This type of IF statement is all on one line."
120 PRINT
130 PRINT TAB(5); " The IF expression THEN"
140 PRINT TAB(5); "<statement>"
150 PRINT TAB(5); "[ELSE"
160 PRINT TAB(5); "<statements>]"
170 PRINT TAB(5); "[ELSEIF expression THEN"
180 PRINT TAB(5); "statements>]"
190 PRINT TAB(5); "ENDIF"
200 PRINT
210 PRINT TAB(5); " Multiline IF statement with optional ELSE and ELSEIF cases and ending with
ENDIF."
220 PRINT TAB(5); "Each component is on a separate line."
230 PRINT TAB(5); " The IF evaluates 'expression' and performs the statement(s) following THEN if
the"
240 PRINT TAB(5); "expression is true or optionally the statement(s) following the ELSE statements"
250 PRINT TAB(5); " as required."
260 print tab(3); " Press 'ENTER' to continue"
270 a$=INKEY$:IF a$="" THEN GOTO 270
280 IF a$=CHR$(&H0D) THEN goto 290
290 PRINT
300 FONT#2,1
310 PRINT TAB(10); " The ELSE Command"
320 FONT#1,1
330 PRINT
340 PRINT TAB(5); " The ELSE statement introduces a default condition in a multiline IF statement."
350 PRINT
360 FONT#2,1
370 PRINT TAB(10); " The ELSEIF expression THEN Command"
380 FONT#1,1
```

(continued)

```
390 PRINT
400 PRINT TAB(5); " Introduces a secondary condition in a multiline IF statement."
410 PRINT
420 FONT#2,1
430 PRINT TAB(10); " THE ENDIF Command"
440 FONT#1,1
450 PRINT
460 PRINT TAB(5); "Terminates a multiline IF statement."
470 PRINT: PRINT: PRINT
480 PRINT "Press `ESC' to return to the search";
490 a$=INKEY$:IF a$="" THEN GOTO 4896
500 IF a$=CHR$(27) THEN RUN "help"
```

```
---------
ELSEIF
---------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The IF Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The IF expr THEN statement or IF expr THEN statement ELSE statement, evaluates the"
70 PRINT TAB(5); "'expr' and performs the THEN statement if it is true or skips to the next line if false."
80 PRINT TAB(5); "The option ELSE statement is the reverse of the THEN test. The 'statement' can be just"
90 PRINT TAB(5); "a line number and in that case a GOTO is assumed. For Microsoft compatibility the"
100 PRINT TAB(5); "'THEN statement' construct can be also replaced with a 'GOTO line number."
110 PRINT TAB(5); "This type of IF statement is all on one line."
120 PRINT
130 PRINT TAB(5); " The IF expression THEN"
140 PRINT TAB(5); "<statement>"
150 PRINT TAB(5); "[ELSE"
160 PRINT TAB(5); "<statements>]"
170 PRINT TAB(5); "[ELSEIF expression THEN"
180 PRINT TAB(5); "statements>]"
190 PRINT TAB(5); "ENDIF"
200 PRINT
210 PRINT TAB(5); " Multiline IF statement with optional ELSE and ELSEIF cases and ending with ENDIF."
220 PRINT TAB(5); "Each component is on a separate line."
230 PRINT TAB(5); " The IF evaluates 'expression' and performs the statement(s) following THEN if the"
240 PRINT TAB(5); "expression is true or optionally the statement(s) following the ELSE statements"
250 PRINT TAB(5); " as required."
260 print tab(3); " Press 'ENTER' to continue"
270 a$=INKEY$:IF a$="" THEN GOTO 270
280 IF a$=CHR$(&H0D) THEN goto 290
290 PRINT
300 FONT#2,1
310 PRINT TAB(10); " The ELSE Command"
320 FONT#1,1
330 PRINT
340 PRINT TAB(5); " The ELSE statement introduces a default condition in a multiline IF statement."
350 PRINT
360 FONT#2,1
370 PRINT TAB(10); " The ELSEIF expression THEN Command"
380 FONT#1,1
```

(continued)

```
390 PRINT
400 PRINT TAB(5); " Introduces a secondary condition in a multiline IF statement."
410 PRINT
420 FONT#2,1
430 PRINT TAB(10); " THE ENDIF Command"
440 FONT#1,1
450 PRINT
460 PRINT TAB(5); "Terminates a multiline IF statement."
470 PRINT: PRINT: PRINT
480 PRINT "Press `ESC' to return to the search";
490 a$=INKEY$:IF a$="" THEN GOTO 4896
500 IF a$=CHR$(27) THEN RUN "help"
```

END

```
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The END Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); "The END command will end the running program and return "
70 print tab(5); "to the command prompt"
80 PRINT: PRINT: PRINT
90 PRINT "Press `ESC' to return to the search";
100 a$=INKEY$:IF a$="" THEN GOTO 100
110 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
ENDIF
--------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The IF Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The IF expr THEN statement or IF expr THEN statement ELSE statement, evaluates the"
70 PRINT TAB(5); "'expr' and performs the THEN statement if it is true or skips to the next line if false."
80 PRINT TAB(5); "The option ELSE statement is the reverse of the THEN test. The 'statement' can be just"
90 PRINT TAB(5); "a line number and in that case a GOTO is assumed. For Microsoft compatibility the"
100 PRINT TAB(5); "'THEN statement' construct can be also replaced with a 'GOTO linenumber."
110 PRINT TAB(5); "This type of IF statement is all on one line."
120 PRINT
130 PRINT TAB(5); " The IF expression THEN"
140 PRINT TAB(5); "<statement>"
150 PRINT TAB(5); "[ELSE"
160 PRINT TAB(5); "<statements>]"
170 PRINT TAB(5); "[ELSEIF expression THEN"
180 PRINT TAB(5); "statements>]"
190 PRINT TAB(5); "ENDIF"
200 PRINT
210 PRINT TAB(5); " Multiline IF statement with optional ELSE and ELSEIF cases and ending with ENDIF."
220 PRINT TAB(5); "Each component is on a separate line."
230 PRINT TAB(5); " The IF evaluates 'expression' and performs the statement(s) following THEN if the"
240 PRINT TAB(5); "expression is true or optionally the statement(s) following the ELSE statements"
250 PRINT TAB(5); " as required."
260 print tab(3); " Press 'ENTER' to continue"
270 a$=INKEY$:IF a$="" THEN GOTO 270
280 IF a$=CHR$(&H0D) THEN goto 290
290 PRINT
300 FONT#2,1
310 PRINT TAB(10); " The ELSE Command"
320 FONT#1,1
330 PRINT
340 PRINT TAB(5); " The ELSE statement introduces a default condition in a multiline IF statement."
350 PRINT
360 FONT#2,1
370 PRINT TAB(10); " The ELSEIF expression THEN Command"
380 FONT#1,1
```

(CONTINUE)

```
390 PRINT
400 PRINT TAB(5); " Introduces a secondary condition in a multiline IF statement."
410 PRINT
420 FONT#2,1
430 PRINT TAB(10); " THE ENDIF Command"
440 FONT#1,1
450 PRINT
460 PRINT TAB(5); "Terminates a multiline IF statement."
470 PRINT: PRINT: PRINT
480 PRINT "Press `ESC' to return to the search";
490 a$=INKEY$:IF a$="" THEN GOTO 4896
500 IF a$=CHR$(27) THEN RUN "help"
```

ERASE

```
4726 CLS
4727 FONT#2,1
4728 PRINT TAB(10); " The ERASE Command"
4729 FONT#1,1 : PRINT
4730 PRINT TAB(5); "The ERASE variable[,variable]... command will delete arrayed variables and "
4731 PRINT TAB(5); "frees up the memory. Use CLEAR to delete all variables including all arrayed
variables."
4794 PRINT: PRINT: PRINT: PRINT: PRINT: PRINT
4795 PRINT "Press `ESC' to return to the search";
4796 a$=INKEY$:IF a$="" THEN GOTO 4796
4797 IF a$=CHR$(27) THEN RUN "help"
```

```
---------
ERROR
---------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The ERROR Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The ERROR[error_msg$] will force an error and terminate the program."
70 print tab(5); "This is normally"
80 PRINT TAB(5); "used in debugging or to trap events that’s should not occur."
90 PRINT: PRINT: PRINT
100 PRINT "Press `ESC' to return to the search";
110 a$=INKEY$:IF a$="" THEN GOTO 110
120 IF a$=CHR$(27) THEN RUN "help"
```

EXIT

```
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The EXIT Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The EXIT command will exit by itself will terminate a DO.....LOOP."
70 PRINT
80 PRINT TAB(5); " The EXIT FOR command will terminate a FOR.....NEXT loop."
90 PRINT: PRINT: PRINT
100 PRINT "Press `ESC' to return to the search";
110 a$=INKEY$:IF a$="" THEN GOTO 110
120 IF a$=CHR$(27) THEN RUN "help"
```

```
-------
FILES
-------
4496 IF a$=CHR$(27) THEN RUN "help"
4914 CLS
4915 FONT#2,1
4916 PRINT TAB(10); " The FILES Command"
4917 FONT#1,1
4918 PRINT
4919 PRINT TAB(5); " The FILES[search_pattern$] command will list files in the current directory on the SD card."
4920 PRINT TAB(5); "The SD card(drive:B) may use an optional 'search_pattern$'. Question marks(?) may"
4921 PRINT TAB(5); "match any character and an asterisk(*) as the first character of the file name or extension"
4922 PRINT TAB(5); "will match any file or any extension. If omitted all files will be listed."
4923 PRINT: PRINT: PRINT
4924 PRINT "Press `ESC' to return to the search";
4925 a$=INKEY$:IF a$="" THEN GOTO 4925
4926 IF a$=CHR$(27) THEN RUN "help"
```

```
-------
FONT
-------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The FONT Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The FONT#nbr or FONT#nbr,scale or FONT#nbr,scale,reverse"
70 PRINT TAB(5); "commands will select a font for the video output.'nbr' is "
80 PRINT TAB(5); "the font number in the range a 1 to 10, the # symbol is"
85 print tab(5); "optional. 'scale' is the multiply factor in the range of "
90 PRINT TAB(5); "1 to 8 (eg, a scale of 2 will double the size of the pixel"
100 PRINT TAB(5); "in both vertical and horizontal). Default is 1.If 'reverse'"
105 print tab(5); "is a number other than zero the font will be displayed in "
110 PRINT TAB(5); "reverse video. Default is no reverse."
120 PRINT
130 PRINT TAB(5); " There are three fonts built in to MMBasic."
140 PRINT TAB(2); "#1 is the standard font of 10x5 pixels containing the full ASCII set."
150 PRINT TAB(2); "#2 is a larger font of 16x11 pixels also with a full ASCII set."
160 PRINT TAB(2); "#3 is a jumbo font of 30x22 pixels consisting of the numbers zero to "
170 PRINT TAB(2); "   nine and the characters plus,minus,comma and fullstop."
180 PRINT
190 PRINT TAB(6); "EXAMPLES:- 10 FONT#3,2,1     'double scale and reverse video "
200 PRINT TAB(6); "           10 FONT#2         ' just selects font #2"
210 print tab(5); " Press 'ENTER' to continue"
220 a$=INKEY$:IF a$="" THEN GOTO 220
230 IF a$=CHR$(&H0D) THEN goto 240
240 PRINT
250 FONT#2,1
260 PRINT TAB(10); " The FONT LOAD Command"
270 FONT#1,1
280 PRINT
290 PRINT TAB(5); " The FONT LOAD file$ AS #nbr will load the font contained in"
300 PRINT TAB(5); "'files' and install it as font 'nbr' which can be any number"
305 print tab(5); "between 3 and 10, the # symbol is optional."
310 PRINT
320 FONT#2,1
330 PRINT TAB(10); " THE FONT UNLOAD Command"
340 FONT#1,1
350 PRINT
360 PRINT TAB(5); " The FONT UNLOAD#nbr will remove font 'nbr' and free the memory used, "
370 PRINT TAB(5); "the # symbol is optional. You cannot unload the built in fonts."
380 PRINT: PRINT: print
390 PRINT "Press `ESC' to return to the search";
400 a$=INKEY$:IF a$="" THEN GOTO 400
410 IF a$=CHR$(27) THEN RUN "help"
```

```
-----
FOR
-----
4923 CLS
4924 FONT#2,1
4925 PRINT TAB(10); " The FOR Command"
4926 FONT#1,1
4927 PRINT
4928 PRINT TAB(5); " The FOR counter = start TO finish[STEP increment] command will iniate a
FOR-Next loop"
4929 PRINT TAB(5); "with the 'counter' initially set to 'start' and incrementing in 'increment' steps"
4930 PRINT TAB(5); "(default is 1) until 'counter' equals 'finish'. The 'increment' must be an integer"
4931 PRINT TAB(5); "but may be negative."
4932 PRINT
4933 FONT#2,1
4934 PRINT TAB(10); " The NEXT Command"
4935 FONT#1,1
4936 PRINT
4937 PRINT TAB(5); " The NEXT[counter_variable][,counter-variable],etc command comes at the end
of the"
4938 PRINT TAB(5); "FOR-NEXT loop. 'The counter-variable' specifies exactly which loop is being"
4939 PRINT TAB(5); "operated on. If no 'counter-variable' is specified the NEXT will default to the'"
4940 PRINT TAB(5); "innermost loop. It is also possible to specify multiple counter-variables as in"
4941 PRINT
4942 PRINT TAB(5); " NEXT x,y,z "
4994 PRINT: PRINT: PRINT
4995 PRINT "Press `ESC' to return to the search";
4996 a$=INKEY$:IF a$="" THEN GOTO 4996
4997 IF a$=CHR$(27) THEN RUN "help"
```

GOSUB

```
5100 CLS
5101 FONT#2,1
5102 PRINT TAB(10); " The GOSUB Command"
5103 FONT#1,1
5104 PRINT
5105 PRINT TAB(5); " The GOSUB line command initiates a subroutine call to the line specified.
The"
5106 PRINT TAB(5); "sub routine must end with a RETURN."
5107 PRINT: PRINT: PRINT
5108 PRINT "Press `ESC' to return to the search";
5109 a$=INKEY$:IF a$="" THEN GOTO 5109
5110 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
GOTO
--------
5113 CLS
5114 FONT#2,1
5115 PRINT TAB(10); " The GOTO Command"
5116 FONT#1,1
5117 PRINT
5118 PRINT TAB(5); " The GOTO line command branches program execution to the specified line."
5119 PRINT: PRINT: PRINT
5120 PRINT "Press `ESC' to return to the search";
5121 a$=INKEY$:IF a$="" THEN GOTO 5121
5122 IF a$=CHR$(27) THEN RUN "help"
```

```
-------
HELP
-------
10 REM      DunioMite Help File Version 0.01
20 REM      First BAS file by Bryan1
30 REM      Open source so feel free to add
40 REM      Just put your username in 20 if
50 REM      if contribute.
60 '-----------------------------------------------------
199 CLS
200 FONT#1,2
210 PRINT TAB(9); "Maximite/ DuinoMite Help File"
220 PRINT
230 FONT#1,1
240 PRINT TAB(9);"This help file is a work in progress and in time will be the"
250 PRINT TAB(8);"main help file that users can store on the SD card and have"
260 PRINT TAB(8); "a precise online help file with code examples to ensure a"
270 PRINT TAB(8); "handy reference will always be on hand when needed."
280 PRINT
285 FONT#2,1
290 PRINT TAB(11); "Commands and Codes"
292 FONT#1,1
295 PRINT
300 PRINT TAB(13); " AUTO - CHDIR - CIRCLE - CLEAR - CLOSE-
CLOSE_CONSOLE[cls_con]"
310 PRINT TAB(13); " CLS - CONTINUE - COPYRIGHT[cy_right] - DATA - DATE "
320 PRINT TAB(13); " DELETE - DIM - DO - EDIT - ELSE - ELSEIF - ENDIF - END "
330 PRINT TAB(13); " ERASE - ERROR - EXIT - FILES - FONT - FOR - GOSUB - GOTO"
340 PRINT TAB(13); " IF - INPUT - IRETURN - KILL - LET - LINE - LINE_INPUT[line_inp]"
350 PRINT TAB(13); " LIST - LOAD - LOCATE - LOOP - MEMORY - MERGE - MKDIR - NAME"
360 PRINT TAB(13); " NEW - NEXT - ON - OPEN - OPEN comspec$[open_com] - REM"
370 PRINT TAB(13); " OPTION - PAUSE - PIN - PIXEL - POKE - PRINT - RANDOMIZE[random]"
380 PRINT TAB(13); " READ - RENUMBER - RESTORE - RETURN - RUN - SAVE - SAVEBMP"
390 PRINT TAB(13); "SETPIN - SETTICK - SOUND - TIME - TIMER - TROFF - TRON -
XMODEM"
400 PRINT : PRINT
410 PRINT TAB(5); " Due to 8 characters being the limit for file names longer"
420 PRINT TAB(5); "names have the codes in [  ] tags."
590 PRINT: PRINT: PRINT: PRINT: PRINT
600 PRINT; " Type commands in Lower Case"
610 PRINT
620 INPUT " Type in the command you want to see";query$
960 CLS
980 ' PAUSE 1000
1000 RUN query$
```

```
---
IF
---
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The IF Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The IF expr THEN statement or IF expr THEN statement ELSE statement,
evaluates the"
70 PRINT TAB(5); "'expr' and performs the THEN statement if it is true or skips to the next line if
false."
80 PRINT TAB(5); "The option ELSE statement is the reverse of the THEN test. The 'statement' can be
just"
90 PRINT TAB(5); "a line number and in that case a GOTO is assumed. For Microsoft compatibility
the"
100 PRINT TAB(5); "'THEN statement' construct can be also replaced with a 'GOTO line number."
110 PRINT TAB(5); "This type of IF statement is all on one line."
120 PRINT
130 PRINT TAB(5); " The IF expression THEN"
140 PRINT TAB(5); "<statement>"
150 PRINT TAB(5); "[ELSE"
160 PRINT TAB(5); "<statements>]"
170 PRINT TAB(5); "[ELSEIF expression THEN"
180 PRINT TAB(5); "statements>]"
190 PRINT TAB(5); "ENDIF"
200 PRINT
210 PRINT TAB(5); " Multiline IF statement with optional ELSE and ELSEIF cases and ending with
ENDIF."
220 PRINT TAB(5); "Each component is on a separate line."
230 PRINT TAB(5); " The IF evaluates 'expression' and performs the statement(s) following THEN if
the"
240 PRINT TAB(5); "expression is true or optionally the statement(s) following the ELSE statements"
250 PRINT TAB(5); " as required."
260 PRINT TAB(3); " Press 'ENTER' to continue"
270 a$=INKEY$:IF a$="" THEN GOTO 270
280 IF a$=CHR$(&H0D) THEN GOTO 290
290 PRINT
300 FONT#2,1
310 PRINT TAB(10); " The ELSE Command"
320 FONT#1,1
330 PRINT
340 PRINT TAB(5); " The ELSE statement introduces a default condition in a multiline IF statement."
350 PRINT
360 FONT#2,1
370 PRINT TAB(6); " The ELSEIF expression THEN Command"
380 FONT#1,1
```

(continue)

```
390 PRINT
400 PRINT TAB(5); " Introduces a secondary condition in a multiline IF statement."
410 PRINT
420 FONT#2,1
430 PRINT TAB(10); " THE ENDIF Command"
440 FONT#1,1
450 PRINT
460 PRINT TAB(5); "Terminates a multiline IF statement."
470 PRINT: PRINT: PRINT
480 PRINT "Press `ESC' to return to the search";
490 a$=INKEY$:IF a$="" THEN GOTO 490
500 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
INPUT
--------
5500 CLS
5501 FONT#2,1
5502 PRINT TAB(10); " The INPUT Command"
5503 FONT#1,1
5504 PRINT
5505 PRINT TAB(5); " The INPUT[prompt string$] list of variables command allows input from the"
5506 PRINT TAB(5); "keyboard to a list of variables. The input command will prompt with a question
mark(?)"
5507 PRINT TAB(5); "The input must contain commas to separate each data item if there is more than
one"
5508 PRINT TAB(5); "variable.  For Example, if the command is INPUT a,b,c "
5509 PRINT TAB(5); "And the following is typed on the keyboard 23,87,66"
5510 PRINT TAB(5); "Then a=23 and b=87 and c=66"
5511 PRINT TAB(5); "If the 'prompt string$' is specified it will be printed before the question mark."
5512 PRINT TAB(5); "If the prompt string is terminated with a comma (,) rather than the semicolon(;)"
5513 PRINT TAB(5); "the question mark will be suppressed."
5514 PRINT TAB(5); " The IMPUT#nbr,list of variables command is the same as above except that the
input"
5515 PRINT TAB(5); "is read from a file previously opened for INPUT as 'nbr'. See the OPEN
command."
5516 PRINT: PRINT: PRINT
5517 PRINT "Press `ESC' to return to the search";
5518 a$=INKEY$:IF a$="" THEN GOTO 5518
5519 IF a$=CHR$(27) THEN RUN "help"
```

```
-----------
RETURN
-----------
5200 CLS
5201 FONT#2,1
5202 PRINT TAB(10); "The IRETURN Command"
5203 FONT#1,1
5204 PRINT
5205 PRINT TAB(5); " The IRETURN command will return from an interrupt. The next statement to
be executed"
5206 PRINT TAB(5); "will be the one that was about to be executed when the interrupt was detected."
5207 PRINT: PRINT: PRINT
5208 PRINT "Press `ESC' to return to the search";
5209 a$=INKEY$:IF a$="" THEN GOTO 5209
5210 IF a$=CHR$(27) THEN RUN "help"
```

KILL

```
5207 CLS
5208 FONT#2,1
5209 PRINT TAB(10); " The KILL Command"
5210 FONT#1,1
5211 PRINT
5212 PRINT TAB(5); " The KILL file$ command will delete the file specified by file$ from the SD
card."
5213 PRINT TAB(5); "Quote marks are required around a string constant and the extension, if there is"
5214 PRINT TAB(5); "one, must be specified. Example:- KILL " "SAMPLE.DAT"
5215 PRINT: PRINT: PRINT
5216 PRINT "Press `ESC' to return to the search";
5217 a$=INKEY$:IF a$="" THEN GOTO 5217
5218 IF a$=CHR$(27) THEN RUN "help"
```

```
-----
LET
-----
5215 CLS
5216 FONT#2,1
5217 PRINT TAB(10); " The LET Command"
5218 FONT#1,1
5219 PRINT
5220 PRINT TAB(5); " The LET variable= expression command assigns the value of 'expression' to the
variable."
5221 PRINT TAB(5); "LET is automatically assumed if a statement does not start with a command."
5222 PRINT: PRINT: PRINT
5223 PRINT "Press `ESC' to return to the search";
5224 a$=INKEY$:IF a$="" THEN GOTO 5224
5225 IF a$=CHR$(27) THEN RUN "help"
```

```
------
LINE
------
5300 CLS
5301 FONT#2,1
5302 PRINT TAB(10); " The LINE Command"
5303 FONT#1,1
5304 PRINT
5305 PRINT TAB(5); " The LINE[(x1,y1)] - (x2,y2)[,c[,B[F]]] command draws a line or box on the
video screen."
5306 PRINT TAB(5); "x1,y1 and x2,y2 specify the beginning and end points of the line. c specifies
the"
5307 PRINT TAB(5); "displayed pixel (0=off, non zero=on) and defaults to on if not specified."
5308 PRINT TAB(5); "(x1,y1) is optional and if omitted the last drawing point will be used. The
optional"
5309 PRINT TAB(5); "B will draw a box with the points (x1,y1) and (x2,y2) at opposite corners. The
optional"
5310 PRINT TAB(5); "BF will draw a box (as,B) and will fill the interior."
5323 PRINT: PRINT: PRINT
5324 PRINT "Press `ESC' to return to the search";
5325 a$=INKEY$:IF a$="" THEN GOTO 5325
5326 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
LINE_INP
------------
5311 CLS
5312 FONT#2,1
5313 PRINT TAB(10); " The LINE INPUT Command"
5314 FONT#1,1
5315 PRINT
5316 PRINT TAB(5); " The LINE INPUT [prompt$] string-variable$ command reads the entire line
from the keyboard"
5317 PRINT TAB(5); "into 'string-variable$'. If specified the 'prompt$' will be printed first. Unlike
INPUT,"
5318 PRINT TAB(5); "LINE INPUT will read a whole line, not stopping for comma delimited data
items."
5319 PRINT TAB(5); "A question mark is not printed unless it is part of 'prompt$'."
5320 PRINT
5321 PRINT TAB(5); " The LINE INPUT #nbr,string-variable$ command is the same as above execpt
the the input"
5322 PRINT TAB(5); "is read from a file previously opened for INPUT as 'nbr'. Also see the OPEN
command."
5323 PRINT: PRINT: PRINT
5324 PRINT "Press `ESC' to return to the search";
5325 a$=INKEY$:IF a$="" THEN GOTO 5325
5326 IF a$=CHR$(27) THEN RUN "help"
```

```
------
LIST
------
5400 CLS
5401 FONT#2,1
5402 PRINT TAB(10); " The LIST Command"
5403 FONT#1,1
5404 PRINT
5405 PRINT TAB(5); " The LIST"
5406 PRINT TAB(5); "     LIST line"
5407 PRINT TAB(5); "     LIST -lastline"
5408 PRINT TAB(5); "     LIST firstline-"
5409 PRINT TAB(5); "     LIST firstline - lastline"
5410 PRINT TAB(5); " Lists all lines in a program or a range of lines."
5411 PRINT TAB(5); "If -lastline is used it will start with with the first line in the program."
5412 PRINT TAB(5); "Ifs startline - is used it will list to the end line of the program."
5413 PRINT: PRINT: PRINT
5414 PRINT "Press `ESC' to return to the search";
5415 a$=INKEY$:IF a$="" THEN GOTO 5415
5416 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
LOAD
--------
5413 CLS
5414 FONT#2,1
5415 PRINT TAB(10); " The Load Command"
5416 FONT#1,1
5417 PRINT
5418 PRINT TAB(5); " The LOAD file$ command loads a program called file$ from the SD card into
working memory."
5419 PRINT TAB(5); "Quote marks are required around a string constant."
5420 PRINT
5421 PRINT TAB(5); "Example:- LOAD" "TEST.BAS"
5422 PRINT
5423 PRINT TAB(5); "If an extension is not specified '.BAS' will be added to the file name."
5424 PRINT: PRINT: PRINT
5425 PRINT "Press `ESC' to return to the search";
5426 a$=INKEY$:IF a$="" THEN GOTO 5426
5427 IF a$=CHR$(27) THEN RUN "help"
```

```
----------
LOCATE
----------
5323 CLS
5324 FONT#2,1
5325 PRINT TAB(10); " The Locate Command"
5326 FONT#1,1
5327 PRINT
5328 PRINT TAB(5); " The LOCATE x,y command positions the cursor to a location in pixels and the
next"
5329 PRINT TAB(5); "PRINT command will place its output at this location. This only affects the
video output."
5393 PRINT: PRINT: PRINT: PRINT: PRINT: PRINT: PRINT
5394 PRINT "Press `ESC' to return to the search";
5395 a$=INKEY$:IF a$="" THEN GOTO 5395
5396 IF a$=CHR$(27) THEN RUN "help"
```

```
-------
LOOP
-------
4500 CLS
4501 FONT#2,1
4502 PRINT TAB(10); " The DO Command"
4503 FONT#1,1
4504 PRINT
4505 PRINT TAB(5); " The DO LOOP command - This structure will loop forever"
4506 PRINT TAB(5); "the EXIT command can be used to terminate the loop or"
4507 PRINT TAB(5); "control must be explicitly transferred outside of the"
4508 PRINT TAB(5); "loop by commands like GOTO and RETURN. "
4509 PRINT
4510 FONT#2,1
4511 PRINT TAB(10); " The DO WHILE LOOP Command"
4512 FONT#1,1
4513 PRINT
4514 PRINT TAB(5); " The DO WHILE LOOP command loops while 'expressions'"
4515 PRINT TAB(5); "is true(this is equivalent to the older WHILE-WEND"
4516 PRINT TAB(5); "loop, also implemented in MM Basic."
4517 PRINT
4518 FONT#2,1
4519 PRINT TAB(10); " The DO WHILE UNITL Command"
4520 FONT#1,1
4521 PRINT
4522 PRINT TAB(5); " The DO WHILE UNTIL COMMAND will loop until the"
4523 PRINT TAB(5); "expression following the UNTIL is true."
4524 PRINT: PRINT: PRINT
4525 PRINT "Press `ESC' to return to the search";
4526 a$=INKEY$:IF a$="" THEN GOTO 4526
4527 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
MEMORY
------------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The MEMORY Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The MEMORY command will list the amount of memory is currently in use."
70 print
80 PRINT TAB(5); " For Example  5kB (17%) Program Memory used."
90 PRINT TAB(5); "              3kB (16%) Variable Memory used."
100 PRINT TAB(5); "             12kB (30%) Array and String memory used."
110 PRINT
120 PRINT TAB(5); " Program memory is cleared by the NEW command. Variable, array and "
130 print tab(5); "string memory spaces are cleared by many commands(ie:- NEW,RUN,LOAD,etc)"
140 PRINT TAB(5); "as well as the and ERASE. "
150 PRINT: PRINT: PRINT
160 PRINT "Press `ESC' to return to the search";
170 a$=INKEY$:IF a$="" THEN GOTO 170
180 IF a$=CHR$(27) THEN RUN "help"
```

MERGE

```
5228 CLS
5229 FONT#2,1
5230 PRINT TAB(10); " The MERGE Command"
5231 FONT#1,1
5232 PRINT
5233 PRINT TAB(5); " The MERGE file$ command adds program lines from file$ to the program in memory. Unlike"
5234 PRINT TAB(5); "LOAD, it does not clear the program currently in memory."
5294 PRINT: PRINT: PRINT: PRINT: PRINT
5295 PRINT "Press `ESC' to return to the search";
5296 a$=INKEY$:IF a$="" THEN GOTO 5296
5297 IF a$=CHR$(27) THEN RUN "help"
```

MKDIR

```
5222 CLS
5223 FONT#2,1
5224 PRINT TAB(10); " The MKDIR Command "
5225 FONT#1,1
5226 PRINT
5227 PRINT TAB(5); " The MKDIR dir$ command will make, or create, the directory on the SD card
from 'old$' to 'new$'"
5228 PRINT: PRINT: PRINT
5229 PRINT "Press `ESC' to return to the search";
5230 a$=INKEY$:IF a$="" THEN GOTO 5230
5231 IF a$=CHR$(27) THEN RUN "help"
```

```
---------
NAME
---------
4423 CLS
4424 FONT#2,1
4425 PRINT TAB(10); " The NAME Command"
4426 FONT#1,1
4427 PRINT
4428 PRINT TAB(5); " The NAME old$ AS new$ command will rename a file or"
4429 PRINT TAB(5); "a directory on the SD card from old$ to new$"
4490 PRINT: PRINT: PRINT
4495 PRINT "Press `ESC' to return to the search";
4496 a$=INKEY$:IF a$="" THEN GOTO 4496
4497  IF a$=CHR$(27) THEN RUN "help"
```

```
------
NEW
------
4415 CLS
4416 PRINT
4417 FONT#2,1
4418 PRINT TAB(10); "The NEW Command"
4419 FONT#1,1
4420 PRINT
4421 PRINT TAB(5); " The NEW command will delete the program in memory"
4422 PRINT TAB(5); "and clears all variable."
4423 PRINT: PRINT: PRINT
4424 PRINT "Press `ESC' to return to the search";
4425 a$=INKEY$:IF a$="" THEN GOTO 4425
4426 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
NEXT
--------
4923 CLS
4924 FONT#2,1
4925 PRINT TAB(10); " The FOR Command"
4926 FONT#1,1
4927 PRINT
4928 PRINT TAB(5); " The FOR counter = start TO finish[STEP increment] command will iniate a
FOR-Next loop"
4929 PRINT TAB(5); "with the 'counter' initially set to 'start' and incrementing in 'increment' steps"
4930 PRINT TAB(5); "(default is 1) until 'counter' equals 'finish'. The 'increment' must be an integer"
4931 PRINT TAB(5); "but may be negative."
4932 PRINT
4933 FONT#2,1
4934 PRINT TAB(10); " The NEXT Command"
4935 FONT#1,1
4936 PRINT
4937 PRINT TAB(5); " The NEXT[counter_variable][,counter-variable],etc command comes at the end
of the"
4938 PRINT TAB(5); "FOR-NEXT loop. 'The counter-variable' specifies exactly which loop is being"
4939 PRINT TAB(5); "operated on. If no 'counter-variable' is specified the NEXT will default to the'"
4940 PRINT TAB(5); "innermost loop. It is also possible to specify multiple counter-variables as in"
4941 PRINT
4942 PRINT TAB(5); " NEXT x,y,z "
4994 PRINT: PRINT: PRINT
4995 PRINT "Press `ESC' to return to the search";
4996 a$=INKEY$:IF a$="" THEN GOTO 4996
4997 IF a$=CHR$(27) THEN RUN "help"
```

```
----
ON
----
5119 CLS
5120 FONT#2,1
5121 PRINT TAB(10); " The ON Command"
5122 FONT#1,1
5123 PRINT
5124 PRINT TAB(5); " The ON variable"
5125 PRINT TAB(5); "     GOTO|GOSUB"
5126 PRINT TAB(5); "     line[,line,line..]"
5127 PRINT TAB(5); " ON either branches (GOTO) or calls a subroutine (GOSUB) based on the"
5128 PRINT TAB(5); "rounded value of variable: if it is 1, the first line is called, if 2"
5129 PRINT TAB(5); "the second line is called etc..."
5135 PRINT: PRINT: PRINT
5194 PRINT "Press `ESC' to return to the search";
5195 a$=INKEY$:IF a$="" THEN GOTO 5195
5196 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
OPEN
--------
10 CLS
20 FONT#2,1
30 PRINT TAB(10); " The OPEN Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The OPEN fname$ FOR mode AS [*]fnbr command will open a file "
70 print tab(5); "on the SD card for reading and writing. 'fname' is the filename"
70 PRINT TAB(5); "(8 chars max) with an optional extension (3 chars max) separated"
80 PRINT TAB(5); "by a dot(.). 'mode' is INPUT or OUTPUT or APPEND. INPUT will open"
90 PRINT TAB(5); "the file for reading and throw an error if the file doesn't exist."
100 PRINT TAB(5); "OUTPUT will open the file for writing and will automatically"
110 PRINT TAB(5); "overwrite any existing file with the same name. APPEND will "
120 PRINT TAB(5); "for writing but if will not overwrite an existing file, instead "
130 PRINT TAB(5); "any writes also open the file will be appended to the end of the"
140 print tab(5); "file. If there is no existing file the APPEND option will act the"
150 PRINT TAB(5); "same as the OUPUT mode (ie:- the file is created then opened"
160 PRINT TAB(5); "for writing). 'fnbr' is the file number (1 to 10). The # is"
170 PRINT TAB(5); "optional. Up to 10 files va be open simultaneously. "
180 print tab(5); "The INPUT,LINE INPUT,PRINT,WRITE and CLOSE commands as well as the"
190 PRINT TAB(5); "EOF() and INPUT$() functions all use 'fnbr' to identify"
200 PRINT TAB(5); "the file being operated on."
210 PRINT: PRINT: PRINT
220 PRINT "Press `ESC' to return to the search";
230 a$=INKEY$:IF a$="" THEN GOTO 230
240 IF a$=CHR$(27) THEN RUN "help"
```

```
---------------
OPEN_COM
---------------
4500 CLS
4501 FONT#2,1
4502 PRINT TAB(7); " The OPEN [comspec] Command"
4503 FONT#1,1
4504 PRINT
4505 PRINT TAB(5); "OPEN comspec$ AS[#]fnbr"
4506 PRINT TAB(5); "or"
4507 PRINT TAB(5); "OPEN comspec$ AS console"
4508 PRINT
4509 PRINT TAB(5); "Will open a serial port for reading and writing. Two ports are"
4510 PRINT TAB(5); "data received will be treated the same as keystrokes received"
4511 PRINT TAB(5); "from the keyboard and any characters sent to the video output"
4512 PRINT TAB(5); "will also be transmitted via the serial port. This enables the"
4513 PRINT TAB(5); "remote control of MMBasic via a serial interface."
4514 PRINT: PRINT: PRINT
4515 PRINT TAB(5); "COM2: uses pin 19 for receive data and pin 20 for transmit data."
4516 PRINT TAB(5); "If the port is opened using 'fnbr' the port can be written to"""
4517 PRINT TAB(5); "and read from using any commands or functions that use a file number."
4518 PRINT
4519 PRINT TAB(5); " A serial port can be opened with 'AS CONSOLE'. In this case any"
4520 PRINT: PRINT: PRINT
4525 PRINT "Press `ESC' to return to the search";
4526 a$=INKEY$:IF a$="" THEN GOTO 4526
4527 IF a$=CHR$(27) THEN RUN "help"
```

```
-----------
OPTION
-----------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The OPTION Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); "OPTION BASE 0"
70 PRINT TAB(5); "or"
80 PRINT TAB(5); "OPTION BASE 1"
90 PRINT
100 PRINT TAB(5); "Sets the lowest value for array subscripts to either 0 or 1. The"
110 PRINT TAB(5); " default is 0. This must be used before any arrays are declared."
120 PRINT: PRINT
130 PRINT TAB(5); "OPTION ERROR"
140 PRINT TAB(5); "CONTINUE"
150 PRINT TAB(5); "or"
160 PRINT TAB(5); "OPTION ERROR ABORT"
170 PRINT
180 PRINT TAB(5); "Set the treatment for errors in file input/output. The option"
190 PRINT TAB(5); "CONTINUE will cause MMBasic to ignore file related errors. The"
200 PRINT TAB(5); "program must check the variable MM.ERRNO to determine if and"
210 PRINT TAB(5); "what error has occurred. The option ABORT sets the normal behavior"
220 PRINT TAB(5); "(ie:- stop the program and print an error message. The default"
230 PRINT TAB(5); "is ABORT. Note that this option only relates to errors reading"
240 PRINT TAB(5); "or writing from the SD card, it does not affect the handling"
250 PRINT TAB(5); "of syntax and other program errors."
260 PRINT
270 PRINT TAB(5); "OPTION PROMT string$"
280 PRINT
290 PRINT TAB(5); "Set the command prompt to the contents of 'string$' (which"
300 PRINT TAB(5); "can also be an expression which will be evaluated when the prompt"
310 PRINT TAB(5); "is printed.)"
320 PRINT TAB(5); " Press 'ENTER' to continue"
330 a$=INKEY$:IF a$="" THEN GOTO 330
340 IF a$=CHR$(&H0D) THEN GOTO 350
350 PRINT TAB(5); "For Example"
360 PRINT TAB(6); "   OPTION PROMPT 'OK'"
370 PRINT TAB(5); "or OPTION PROMPT TIME$ + ':'"
380 PRINT TAB(5); "or OPTION PROMPT CWD$ +':'"
390 PRINT
400 PRINT TAB(5); "Maximum length of the prompt string is 48 characters. The prompt"
410 PRINT TAB(5); "is reset to the default ('>') on power up but you can automatically"
420 PRINT TAB(5); "set it by saving the following example program as 'autorun.bas' on"
430 PRINT TAB(5); "the internal flash driveA:"
440 PRINT TAB(5); "                    10 OTION PROMPT 'My prompt:'"
```

(continued)

```
450 PRINT TAB(5); "                    20 NEW"
460 PRINT
470 PRINT TAB(5); " OPTION Fnn string$"
480 PRINT
490 PRINT TAB(5); " Sets the programmable function keys 'Fnn' to the contents of'string$'"
500 PRINT TAB(5); "'Fnn' is the function key F1 to F12. Maximum string length is 12"
510 PRINT TAB(5); "characters. 'string$' can also be an expression which will be evaluated"
520 PRINT TAB(5); "at the time of running the OPTION command. This is most often used"
530 PRINT TAB(5); "to append the ENTER key (chr$(13)), or double quotes (chr$(34))."
540 PRINT TAB(5); "  For Example:-"
550 PRINT TAB(5); "          OPTION F1 'RUN' + CHR$(13)"
560 PRINT TAB(5); "          OPTION F6 'SAVE' + CHR$(34)"
570 PRINT TAB(5); "          OPTION F10 'ENDIF'"
580 PRINT TAB(5); " Normally these commands are included in an AUTORUN.BAS file (see"
590 PRINT TAB(5); "OPTION PROMPT for an Example."
600 PRINT: PRINT: PRINT
610 PRINT "Press `ESC' to return to the search";
620 a$=INKEY$:IF a$="" THEN GOTO 620
630 IF a$=CHR$(27) THEN RUN "help"
```

PAUSE

```
10 CLS
20 FONT#2,1
30 print tab(8); " The PAUSE Command"
40 FONT#1,1
50 print
60 print tab(5); " The PAUSE nbr command will halt execution of the running program"
70 print tab(5); "for 'nbr' milliseconds. The maximum value of 'nbr' is 4294967295"
80 print tab(5); "( about 49 days)."
90 print: print: print
100 PRINT "Press `ESC' to return to the search";
110 a$=INKEY$:IF a$="" THEN GOTO 110
120 IF a$=CHR$(27) THEN RUN "help"
```

```
-----
PIN
-----
10 CLS
20 FONT#2,1
30 Print tab(8); " The PIN Command"
40 FONT#1,1
50 print
60 print tab(5); " The PIN (pin)= value command. for a 'pin' configured as digital"
70 print tab(5); "output this will set the output to low ('value' is zero) or high"
80 print tab(5); "('value' non zero). You can set an output high or low before it"
90 print tab(5); "is configured as an output and that setting will be the default"
100 print tab(5); "output when the SETPIN command takes effect. 'pin' zero is a "
110 print tab(5); "special case and will always control the LED on the front panel."
120 print tab(5); "A 'value' of non zero will turn the LED on, or zero for off."
130 print
140 print tab(5); "See the function PIN() for reading from a pin and the command"
150 print tab(5); "SETPIN for configuring it."
160 print: print: print
170 PRINT "Press `ESC' to return to the search";
180 a$=INKEY$:IF a$="" THEN GOTO 180
190 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
PIXEL
--------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The PIXEL Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The PIXEL(x,y) = value command will set a pixel on the VGA or"
70 PRINT TAB(5); "composite screen off (if the value is zero) or on (if the value"
80 PRINT TAB(5); "is non zero) See the Graphic's section for a definition of the"
90 PRINT TAB(5); "graphic coordinates. See the function PIXEL(x,y) for obtaining"
100 PRINT TAB(5); "the value of a pixel."
110 PRINT: PRINT: PRINT
120 PRINT "Press `ESC' to return to the search";
130 a$=INKEY$:IF a$="" THEN GOTO 130
140 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
POKE
--------
10 CLS
20 FONT#2,1
30 print tab(8); " The POKE Command"
40 FONT#1,1
50 print
60 print tab(5); " The POKE hi word, low word, val command will set a byte within the PIC32"
70 print tab(5); "virtual memory space to 'val'. 'hi word' is the top 16 bits of the address"
80 print tab(5); "while 'low word' is the bottom 16 bits."
90 print
100 print tab(5); " THIS COMMAND IS FOR EXPERT USERS ONLY"
110 print
120 print tab(5); "The PIC32 maps all control registers, flash (program) memory and volatile"
130 print tab(5); "(RAM) memory into a single address space so there is no need for INP or"
140 print tab(5); "OUT commands. The PIC32MX5XX/6XX/7XX Family Data Sheet lists the details"
150 print tab(5); 'of this address space while the source code will provide the symbolic"
160 print tab(5); "names used in the firmware and Maximite.map file (produced after a"
170 print tab(5); "successful compile) will list address's of these symbols. These address's"
180 print tab(5); "will change with each version of the firmware so programs should use"
190 print tab(5); "the predefined variable MM.VER to determine the currently running version."
200 print
210 print tab(5); " WARNING: If you use this facility to access an invalid memory address"
220 print tab(5); "the MIPS CPU will throw an exception which causes the processor to"
230 print tab(5); "reset and clear all memory. to see this effect try POKE 0,0,0
240 print: print: print
250 PRINT "Press `ESC' to return to the search";
260 a$=INKEY$:IF a$="" THEN GOTO 260
270 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
PRINT
--------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The PRINT Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The PRINT expression"
70 PRINT TAB(5); "[[,;]expression]...etc"
80 PRINT
90 PRINT TAB(5); " Command outputs text to the screen. Multiple expressions can be used"
100 PRINT TAB(5); "and must be separated by  either:-"
110 PRINT TAB(5); "    Comma (,) which will output the tab character"
120 PRINT TAB(5); '   Semicolon (;) which will not output anything ( it is just used to"
130 PRINT TAB(5); "separate expressions)."
140 PRINT TAB(5); "   Nothing or a space which will act the same as a semicolon."
150 PRINT
160 PRINT TAB(5); "A semicolon(;) at the end of the expression list will suppress the"
170 PRINT TAB(5); "automatic output of a carriage return/ newline at the end of a print"
180 PRINT TAB(5); "statement. When printed, a number is preceded with a space if positive"
190 PRINT TAB(5); "or a minus(-) if negative but is not followed by a space. Integers"
200 PRINT TAB(5); "(whole numbers) are printed without a decimal point while fractions"
210 PRINT TAB(5); "are printed with the decimal point and the significant decimal digits"
220 PRINT TAB(5); "Large numbers( greater than six digits) are printed in scientific"
230 PRINT TAB(5); "format/ The function FORMAT$() can be used to format numbers. The"
240 PRINT TAB(5); "function TAB() can be used to space to a certain column and the"
250 PRINT TAB(5); "string functions can be used to justify or otherwise format strings."
260 PRINT
270 PRINT TAB(5); ' A single question mark (?) can be used as a shortcut for the PRINT"
280 PRINT TAB(5); "keyword."
290 PRINT
300 PRINT TAB(5); " The PRINT #nbr, expression"
310 PRINT TAB(5); " [[,;] expression]...etc"
320 PRINT
330 PRINT TAB(5); " Is the same as the above command except that the output is directed'"
340 PRINT TAB(5); "to a file previously opened for OUTPUT or APPEND as 'nbr'."
350 PRINT: PRINT: PRINT
360 PRINT "Press `ESC' to return to the search";
370 a$=INKEY$:IF a$="" THEN GOTO 370
380 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
RANDOM
------------
10 CLS
20 FONT#2,1
30 print tab(8); " The RANDOMIZE Command"
40 FONT#1,1
50 print
60 print tab(5); " The RANDOMIZE nbr command seeds the random number generator with 'nbr'."
70 print tab(5); "To generate a different random sequence each time you must use a"
80 print tab(5); "different value for 'nbr'. One good way to do this is use the TIMER"
90 print tab(5); "function. For Example:- 100 RANDOMIZE TIMER"
100 print: print: print
110 PRINT "Press `ESC' to return to the search";
120 a$=INKEY$:IF a$="" THEN GOTO 120
130 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
READ
--------
4261 CLS
4262 FONT#2,1
4263 PRINT TAB(10); " The READ Command"
4264 FONT#1,1
4265 PRINT
4266 PRINT TAB(5); "The READ command has the variable READ variable['varible]"
4277 PRINT
4278 PRINT TAB(5); " The READ command reads values from the DATA statements and"
4279 PRINT TAB(5); "assigns these values to the named variables. Variable types"
4280 PRINT TAB(5); "in a READ statement must match the data types in DATA"
4281 PRINT TAB(5); "statements as they are read."
4282 PRINT: PRINT: PRINT
4283 PRINT "Press `ESC' to return to the search";
4284 a$=INKEY$:IF a$="" THEN GOTO 4284
4285 IF a$=CHR$(27) THEN RUN"help"
```

```
------
REM
------
10 CLS
20 FONT#2,1
30 print tab(5); " The REM Command"
40 FONT#1,1
50 print
60 print tab(5); " The REM string command allows remarks to be included in a program."
70 print tab(5); "Note the Microsoft style use of the single quotation mark to denote"
80 print tab(5); "remarks are also supported and is preferred."
90 print: print: print
100 PRINT "Press `ESC' to return to the search";
110 a$=INKEY$:IF a$="" THEN GOTO 110
120 IF a$=CHR$(27) THEN RUN "help"
```

```
---------------
RENUMBER
---------------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The RENUMBER Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " RENUMBER"
70 PRINT TAB(5); "or"
80 PRINT TAB(5); "RENUMBER first"
90 PRINT TAB(5); "or"
100 PRINT TAB(5); "RENUMBER first,incr"
110 PRINT TAB(5); "or"
120 PRINT TAB(5); "RENUMBER first,incr,start"
130 PRINT
140 PRINT TAB(5); "The RENUMBER command will renumber the program currently held in"
150 PRINT TAB(5); "memory including all references to line numbers in commands such"
160 PRINT TAB(5); "as GOTO, GOSUB, ON etc. 'first' is the number to be used in the"
170 PRINT TAB(5); "new sequence. Default is 10. 'start' is the line number in the old"
180 PRINT TAB(5); "program where renumbering should commence from. The default is the"
190 PRINT TAB(5); "first line of the program. This command will first check for errors"
200 PRINT TAB(5); "that my disrupt the renumbering process and it will only change"
210 PRINT TAB(5); "the program in memory if no errors are found. However, it is "
220 PRINT TAB(5); "prudent to save the program before running this command in case"
230 PRINT TAB(5); "there are some errors that are not caught."
240 PRINT: PRINT: PRINT
250 PRINT "Press `ESC' to return to the search";
260 a$=INKEY$:IF a$="" THEN GOTO 260
270 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
RESTORE
------------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The RESTORE Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The RESTORE command resets the line and position counters"
70 PRINT TAB(5); "for DATA and READ statements to the top of the program file."
80 PRINT: PRINT: PRINT
90 PRINT "Press `ESC' to return to the search";
100 a$=INKEY$:IF a$="" THEN GOTO 100
110 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
RETURN
------------
5107 CLS
5108 FONT#2,1
5109 PRINT TAB(10); " The RETURN Command"
5110 FONT#1,1
5111 PRINT
5112 PRINT TAB(5); " The RETURN command concludes a subroutine called by a GOSUB."
5113 PRINT: PRINT: PRINT
5114 PRINT "Press `ESC' to return to the search";
5115 a$=INKEY$:IF a$="" THEN GOTO 5115
5116 IF a$=CHR$(27) THEN RUN "help"
```

```
------
RUN
------
10 CLS
20 FONT#2,1
30 print tab(8); " The RUN Command"
40 FONT#1,1
50 print
60 print tab(5); " The RUN [line][file$] command will execute the program in memory."
70 print tab(5); "If a line number is supplied, then execution begins at that line."
80 print tab(5); "Or, if a file name (file$) is supplied, the current program will"
90 print tab(5); "be erased and that program will be loaded from the SD card and"
100 print tab(5); "executed. This enables one program to load and run another."
110 print tab(5); "For Example:- RUN'TEST.BAS' "
120 print tab(5); "If an extension is not specified '.bas' will be added to the file name."
130 print: print: print
140 PRINT "Press `ESC' to return to the search";
150 a$=INKEY$:IF a$="" THEN GOTO 150
160 IF a$=CHR$(27) THEN RUN "help"
```

```
-------
SAVE
-------
10 CLS
20 FONT#2,1
30 print tab(8); " The SAVE Command"
40 FONT#1,1
50 print
60 print tab(5); " The SAVE[file$] command saves the program in the current working directory"
70 print tab(5); "on the SD card as 'file$'. The file name is optional and if omitted the"
80 print tab(5); "last file name used in SAVE, LOAD or RUN will be automatically used."
90 print tab(5); "For Example:- SAVE 'TEST.BAS' "
100 print
110 print tab(5); " An extension is not specified '.bas' will be added to the file name.
120 print: print: print
130 PRINT "Press `ESC' to return to the search";
140 a$=INKEY$:IF a$="" THEN GOTO 140
150 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
SAVEBMP
------------
10 CLS
20 FONT#2,1
30 print tab(8); " The SAVEBMP Command"
40 FONT#1,1
50 print
60 print tab(5); " The SAVEBMP file$ command saves the current VGA or composite screen"
70 print tab(5); "as a BMP file file in the current working directory on the SD card."
80 print tab(5); "For Example :- SAVEBMP'IMAGE.BMP'"
90 print
100 print tab(5); "If an extension is not specified'.BMP' will be added to the file name."
110 print
120 print tab(5); "Note that windows 7 Paint has trouble displaying the image. This"
130 print tab(5); "appears to be a bug in Paint as all the software tested (including"
140 print tab(5); "windows XP Paint) can display the image without fault."
150 print: print: print
160 PRINT "Press `ESC' to return to the search";
170 a$=INKEY$:IF a$="" THEN GOTO 170
180 IF a$=CHR$(27) THEN RUN "help"
```

```
----------
SETPIN
----------
10 CLS
20 FONT#2,1
30 print tab(8); " The SETPIN Command"
40 FONT#1,1
50 print
60 print tab(5); " The SETPIN pin,cfg command will configure the external I/P pin"
70 print tab(5); "according to 'cfg'."
80 print
90 print tab(5); "   0 Not configured or inactive"
100 print tab(5); "   1 Analog Input                                (pins 1 to 10)"
110 print tab(5); "   2 Digital Input    (all pins and 5V tolerant on pins 11 to 20)"
120 print tab(5); "   3 Frequency Input                                  (pins 11 to 14)"
130 print tab(5); "   4 Period Input                                (pins 11 to 14)"
140 print tab(5); "   5 Counting input                              (pins 11 to 14)"
150 print tab(5); "   6 Interrupt on low to high change                  (all pins)"
160 print tab(5); "   7 Interrupt on high to low change                  (all pins)"
170 print tab(5); "   8 Digital Output                              (all pins)"
180 print tab(5); "   9 Open collector digital output to 5v    (pins 11 to 20)
190 print
200 print tab(5); "See the function PIN() for reading inputs and the statement PIN()="
210 print tab(5); "for outputs. See the command below if an interrupt is configured."
220 print
230 print tab(5); " SETPIN pin, cfg, line will configure 'pin to generate an interrupt"
240 print tab(5); "according to 'cfg'."
250 print
260 print tab(5); "   0 Not configured or inactive"
270 print tab(5); "   6 Interrupt on low to high input change   (all pins)"
280 print tab(5); "   7 Interrupt on high to low input change   (all pins)"
290 print
300 print tab(5); "The starting line number of the interrupt routine is specified in"
310 print tab(5); "the third parameter 'line'"
320 print tab(5); "This mode also configures the pin as a digital input so the value of"
330 print tab(5); "the pin can always be retrieved using the function PIN()."
340 print tab(5); " See also IRETURN to return from the interrupt."
350 print: print: print
360 PRINT "Press `ESC' to return to the search";
370 a$=INKEY$:IF a$="" THEN GOTO 370
380 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
SETTICK
------------
10 CLS
20 FONT#2,1
30 print tab(8); " The SETTICK C command"
40 FONT#1,1
50 print
60 print tab(5); "The SETTICK period,line command will setup a periodic interrupt(or tick)"
70 print tab(5); "The time between interrupts is 'period' milliseconds and 'line' is the"
80 print tab(5); "line number of the interrupt routine. "
90 print tab(5); "The period can range from 1 to 4294967295 milliseconds (about 49 days)."
100 print tab(5); " This interrupt can be disabled by setting 'line to zero(IE: SETTICK 0,0)"
110 print: print: print
120 PRINT "Press `ESC' to return to the search";
130 a$=INKEY$:IF a$="" THEN goto 130
140 IF a$=CHR$(27) THEN RUN "help"
```

```
----------
SOUND
----------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The SOUND Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " The SOUND freq,dur"
70 PRINT TAB(5); "     or"
80 PRINT TAB(5); "     SOUND freq,dur,duty"
90 PRINT
100 PRINT TAB(5); " The SOUND command will generate a single tone of 'freq'(between 20Hz"
110 PRINT TAB(5); "to 1MHz) for 'dur' milliseconds. The sound is played in the background"
120 PRINT TAB(5); "and does not stop program execution. If 'dur' ia zero any active"
130 PRINT TAB(5); "SOUND statement is turned off. If no SOUND statement is running, a 'dur'"
140 PRINT TAB(5); "of zero has no effect. 'duty' is the optional duty cycle of the"
150 PRINT TAB(5); "waveform in percent. If it is close to zero the output will be a narrow"
160 PRINT TAB(5); "positive pulse. If it is not specified the duty cycle will default to"
170 PRINT TAB(5); "50%. Setting the duty cycle allows the sound output to be used as a "
180 PRINT TAB(5); "Pulse Width Modulation(PWM) output for driving analogue circuits."
190 PRINT TAB(5); "The signal will be available at the sound connector and the voltage"
200 PRINT TAB(5); "divider on this output should be removed so the full signal level"
210 PRINT TAB(5); "is available. The frequency of the output is locked to the PIC32"
220 PRINT TAB(5); "crystal and is very accurate and for frequencies below 100KHz the"
230 PRINT TAB(5); "duty cycle will be accurate to 0.1%."
240 PRINT: PRINT: PRINT
250 PRINT "Press `ESC' to return to the search";
260 a$=INKEY$:IF a$="" THEN GOTO 260
270 IF a$=CHR$(27) THEN RUN "help"
```

```
-------
TIME
-------
4350 CLS
4351 FONT#2,1
4352 PRINT TAB(6); " The TIME and DATE Commands"
4353 PRINT
4354 FONT#1,1
4356 PRINT TAB(6); "In order to set the DATES and TIMES the following commands"
4357 PRINT TAB(6); "are used. The DATES can be set by:-"
4358 PRINT TAB(6); "DATE$='DD-MM-YY' or 'DD/MM/YY' On power up the date will'"
4359 PRINT TAB(6); "to 1-1-2000. "
4360 PRINT
4361 PRINT TAB(6); " The TIME is set in a similar way like TIME$='HH-MM-SS'"
4362 PRINT TAB(6); "The MM and SS are optional and will default to zero if"
4363 PRINT TAB(6); 'not specified. ie:- TIME$='14.30' will set the clock to"
4364 PRINT TAB(6); "14:30 with zero seconds. Time is set to zero on power up"
4365 PRINT
4366 PRINT TAB(6); " To avoid confusion ' was used instead of the double as"
4367 PRINT TAB(6); "MM basic uses the doubles for programming. Ensure you"
4368 PRINT TAB(6); "the double quotes in your programs."
4369 PRINT: PRINT: PRINT
4396 PRINT "Press `ESC' to return to the search";
4397 a$=INKEY$:IF a$="" THEN GOTO 4397
4398 IF a$=CHR$(27) THEN RUN "help"
```

```
---------
TIMER
---------
10 CLS
20 FONT#2,1
30 print tab(8); " The TIMER Command"
40 FONT#1,1
50 print
60 print tab(5); " The TIMER=msec command rests the timer to a number of milliseconds."
70 print tab(5); "Normally this is just used to reset the timer to zero but you can"
80 print tab(5); "set it to any positive integer. See the TIMER function for more details."
90 print: print: print
100 PRINT "Press `ESC' to return to the search";
110 a$=INKEY$:IF a$="" THEN GOTO 110
120 IF a$=CHR$(27) THEN RUN "help"
```

```
--------
TROFF
--------
10 CLS
20 FONT#2,1
30 print tab(8); " The TROFF Command"
40 FONT#1,1
50 print
60 print tab(5); " The TROFF turns off the trace facility. this facility will print each"
70 print tab(5); "line number in square brackets as the program is executed. This is "
80 print tab(5); "useful in debugging programs."
90 print
100 print tab(5); " The TRON command will turn off the trace facility."
110 print: print: print
120 PRINT "Press `ESC' to return to the search";
130 a$=INKEY$:IF a$="" THEN GOTO 130
140 IF a$=CHR$(27) THEN RUN "help"
```

TRON

```
10 CLS
20 FONT#2,1
30 print tab(8); " The TRON Command"
40 FONT#1,1
50 print
60 print tab(5); " The TRON turns on the trace facility. this facility will print each"
70 print tab(5); "line number in square brackets as the program is executed. This is "
80 print tab(5); "useful in debugging programs."
90 print
100 print tab(5); " The TROFF command will turn off the trace facility."
110 print: print: print
120 PRINT "Press `ESC' to return to the search";
130 a$=INKEY$:IF a$="" THEN GOTO 130
140 IF a$=CHR$(27) THEN RUN "help"
```

```
------------
XMODEM
------------
10 CLS
20 FONT#2,1
30 PRINT TAB(8); " The XMODEM Command"
40 FONT#1,1
50 PRINT
60 PRINT TAB(5); " XMODEM SED file$"
70 PRINT TAB(5); " or"
80 PRINT TAB(5); " XMODEM RECIEVE file$"
90 PRINT
100 PRINT TAB(5); " The XMODEM command transfers a file to or from a remote computer using"
110 PRINT TAB(5); "the XModem protocol. The transfer is done over the USB connection or,"
120 PRINT TAB(5); "if a serial port is opened as console, over that serial port. 'file$' is"
130 PRINT TAB(5); "the file on the SD card or internal flash to be sent or received."
140 PRINT TAB(5); "The XModem protocol requires a cooperating software program running"
150 PRINT TAB(5); "on the remote computer and connected to its serial port. It has been"
160 PRINT TAB(5); "tested on Tera Term running on windows and it is recommended that this'"
170 PRINT TAB(5); "be used. After running the XMODEM command in MMBasic select:"
180 PRINT TAB(5); "  FILE->Transfer->XMODEM->Receive/Send"
190 PRINT TAB(5); "from the Tera Term menu to start the transfer." The transfer can take""
200 PRINT TAB(5); "up to 15 seconds to start if the XMODEM command fails to establish"
210 PRINT TAB(5); "communications it will return to the MMBasic prompt after 30 seconds."
220 PRINT: PRINT: PRINT
230 PRINT "Press `ESC' to return to the search";
240 a$=INKEY$:IF a$="" THEN GOTO 240
250 IF a$=CHR$(27) THEN RUN "help"
```