Contents

1) MMBasic

MMBasic is a powerfull BASIC-Interpreter written by Geoff Graham
for a PIC32 Microcontroller from Microchip.

Link to MMBasic : http://mmbasic.com/
Link to Geoff's Maximite-Project : http://geoffg.net/maximite.html

2) F746-Port

The „MMBasic-STM32F746“ is a port from the original Version to
a STM32F746 Microcontroller from ST.

The starting point of the Firmware is MMBasic for the Maximite in Version 4.5.
The Hardware base is a STM32F746-Discovery Board from ST.

The authors are : Uwe Becker and Fabrice Muller
Link to the German webblog : http://mikrocontroller.bplaced.net/wordpress/?page_id=5487
(here you can find the binary from the newest version)

Link to an English forum : http://www.thebackshed.com/Forum/forum_posts.asp?TID=7969&PN=1

Post release patches (Version 1.xx) made by : Peter

3) STM32F746-Discovery

The Discovery Boards includes all necessary parts for a stand alone system.

INPUT : USB-Connector (to plug in a standard USB-Keyboard)
OUTPUT : 4,3 inch Display (480x272 Pixel @ 65k Color)
DRIVES : uSD-Slot (Drive b:) and USB-Connector for a USB-Drive (Drive c:)
COM : UART connection (@ USB and VCP) to a terminal program
GPIO : 22 User defined IO-Signals (IO, UART, SPI, I2C etc)

and has additional parts like :

RAM : 8Mbyte (to store Graphics, Sounds etc)
TOUCH : 4,3 inch Multitouch (to realize a GUI)
FLASH : 16MByte (to store user settings, programs, files)
SOUND : Stereo Audio-DAC (for Music)
ETHERNET : for LAN connection

4) Differences Maximite <> F746-Port

There are some differences in the implementation of MMBasic between
the Maximite and the F746 Version.

4.1) Display and Colors

A big difference between Maximite and F746-Port is the Video system.

Maximite : 480 x 432 Pixel (monochrome)
Color-Maximite : 480 x 432 Pixel (8 colors)
F746-Port : 480 x 272 Pixel (65535 colors)

All Graphic commands are changed to handle 65535 colors.
There are no need for different color modes so the „MODE“ command is deleted.
Also the „SCANLINE“ command is deleted.

8 default colors are :
"BLACK","WHITE","RED","GREEN","BLUE","CYAN","PURPLE","YELLOW"

8 new default colors are defined :
"ORANGE","BROWN","LRED","DGREY","GREY","LGREY","LGREEN","LBLUE"

and you can use **any** RGB656 color code between 0x0000 and 0xFFFF

You can disable/enable the output for the "PRINT" command in a basic program with :

OPTION VIDEO OFF (disable output)
OPTION VIDEO ON (enable output)

You can store the default color for background and text in Flash with :

CONFIG BGCOLOR #color
CONFIG FGCOLOR #color

4.2) Fonts

At the Moment 6 different Fonts are installed in Flash.

#1 to #3 are the same as in the Maximite Version :

#1 = 6x12 Pixel (standard Font for Editor)
#2 = 13x20 Pixel
#3 = 24x23 Pixel (only Chars : '+,-.',Space,'0...9')

#4 to #6 are generated from UB :

#4 = 8x8 Pixel (C64-Font)
#5 = 8x13 Pixel
#6 = 10x15 Pixel

All Fonts can be used in Basic Programs with the "FONT #nr" command.
All Fonts (except #3) can be stored in Flash as default Font with :

CONFIG FONT #nr

4.3) Touch

The Multi-Touch can handle up to 5 touch positions at a time.

The standard BASIC command „TOUCHED()“ works like the TFT-Maximite Version
and readout only the first touch position.

With the new „MTOUCHED()“ command all 5 positions can be used without loosing speed.

4.4) Keyboard

You can connect a standard USB Keyboard at CN13 (USB_FS).

3 Keyboard Layouts are implemented, use command :

CONFIG KEYBOARD US (for US QWERTY Layout)
CONFIG KEYBOARD GR (for German QWERTZ Layout)
CONFIG KEYBOARD FR (for France AZERTY Layout)

There are two Keyboard shortcuts with :

ALT+F11 = switch Keyboard-Layout (QWERTZ, QWERTY, AZERTY)

ALT+F12 = send display screenshot (as BMP-File) to console output

You can setup the function keys F1 to F12 with a short string like :

example : OPTION F1 "Run" + CHR$(13)

4.5) Console connection

The USB/ST-Link-Port (CN14) is used as UART connection to a terminal
(TeraTerm recommended). Setting is : 115200 Baud, 8N1

You can disable/enable the output in a basic program with :

OPTION USB OFF (disable output)
OPTION USB ON (enable output)

To change baudrate use :

CONFIG BAUDRATE #baudrate

4.6) External Drives

You can connect a Micro-SD card at CN3 and a USB-Drive at CN12 (USB_HS).
Both drives must be formatted with FAT filesystem.

LFN : long filenames are available (up to 32 characters)

To change the active drive :

DRIVE A: (a = internal Flash, not supported)
DRIVE B: (b = uSD)
DRIVE C: (c = USB)

To list the directory of the active drive :

FILES (list all files and directorys)
FILES "*.bas" (list all BAS-Files)
FILES "H*.*" (list all files with first char "H")
FILES "?H*.*" (list all files with second char "H")

To change the directory :

CHDIR <name> (change into <name>)
CHDIR .. (change in parent directory)

Other commands :

KILL <filename> (delete file <filename>)

MKDIR <name> (create directory <name>)

RMDIR <name> (delete direcotry <name>)

To setup the default drive use :

CONFIG DRIVE [A/B/C]

4.7) Sprites / Maps

Because of the better graphic capabilities of the F746-Port there are many differences in the sprite and maps handling.

So many BASIC commands are different in function and/or syntax and the files are not compatible with the Maximite Version.

A spritefile can now handle different sprite sizes up to 32x32 pixel and supports ARGB1555 color format.

Maps for Background also supports the ARGB1555 Format and can have a size up to 1.000.000 Pixel.
The CPU fit the map in every window size like 480x272 or 300x150 or whatever.

4.8) 3D-Objects

These Graphic-Objects are a new feature and made by Fabrice.

Please read the command description for details.

4.9) UART

COM1 and COM2 can be used with these Baudrates :
115200, 57600, 38400, 19200, 9600, 4800 [9600 = default]

Stopbits one or two
FlowControl only works at COM2
BufferSize is fixed at 256 Bytes
RS485 is not supported
OpenCollector Output is not supported
Interrupt functions not supported

Com1 or COM2 can used as console in/out.

GPIO-Pins automatically configured by the basic-command "open"
so there is no need to set input,output with the "setpin" command.

Before the basic program is started, all com-ports are closed.
(Except the port is defined as console)

4.10) SPI

SPI1 and SPI2 can be used as "Master" with these settings :

Frame : 8bit
Speed : 0...7 (default=3)
Mode : 0,1,2,3 (default=3)
Bitorder : MSB, LSB
SlaveSelect-Pin : can be automatically handelt

GPIO-Pins automatically configured by the basic-command "spi open"
so there is no need to set input,output with the "setpin" command.

Before the basic program is started, all spi-ports are closed.

Speed	0	1	2	3	4	5	6	7
SPI-1	195kHz	390kHz	781kHz	1,56MHz	3,12MHz	6,25MHz	12,5MHz	25MHz
SPI-2	390kHz	781kHz	1,56MHz	3,12MHz	6,25MHz	12,5MHz	25MHz	50MHz

4.11) I2C

I2C1 can be used as "Master" with these settings :

Slave-Adress : 8bit
Speed : 10...400 (kHz)
Timeout : 100...n (ms)
Option : 0

GPIO-Pins automatically configured by the basic-command "i2c open"
so there is no need to set input,output with the "setpin" command.

Before the basic program is started, i2c-port is closed.

I2C-Frq : 10,20,30,40,50,60,70,80,90,100,200,300,400 [kHz]

4.12) Sound

At the moment you can load WAV-Files from SD- or USB-Drive and play them in background.
Output is the Headphone Mini-Jack (CN10).

There is space to hold 20 Files (or max 1MByte).

WAVE-Format must be : PCM, 16bit, Mono or Stereo, 8kHz - 48kHz

To load a wave use : "WAVE LOAD"

There are two commands to play the sound "WAVE PLAY" and "WAVE LOOP"
and one command to stop the sound "WAVE STOP"

You can play up to 4 WAV-Files simultaneously (wav formates must be equal).

To unload all WAV-Files use "WAVE CLEAR" (only if no wave is playing).

4.13) Flash

Some settings are stored in the external Flash

If the Flash is not initialized (at the very first start) the default settings are stored.
After that, every time the basic-command "config" is used to store new settings.
The Flash can be erased up to 100.000 times.

4.14) Autorun

After PowerOn the default drive is checked for a file with the name "Autorun.bas".
If this file is found then it will be automatically loaded and started.

To configure this option :

CONFIG AUTORUN OFF (disable autorun)
CONFIG AUTORUN ON (enable autorun)

4.15) GPIO-Pins

Modes : OFF, INTL, INTH, INTB, DIN, DOU, AIN, FIN, PIN, CIN are working.

Before the basic program is started, all GPIO-Pins are set to "OFF"
(except the LED-Pin and the Console-Pins if opened)

Digital-Inputs (DIN) can have 3 internal resistor Modes : "NONE", "UP", "DOWN"
(default is "NONE") e.g. "setpin 1,DIN,DOWN"

Interrupt-Inputs (INTL,INTH,INTB) can have 3 internal resistor Modes : "NONE", "UP", "DOWN"
(default is "NONE") e.g. "setpin 1,INTL,UP,1000"

Pins for CIN, PIN and FIN are 6,8,9,1

4.16) External Devices

You can connect some external Devices to the STM32F7 IO-Ports.
We have build in a few Basic-Commands to have easy access to these devices :

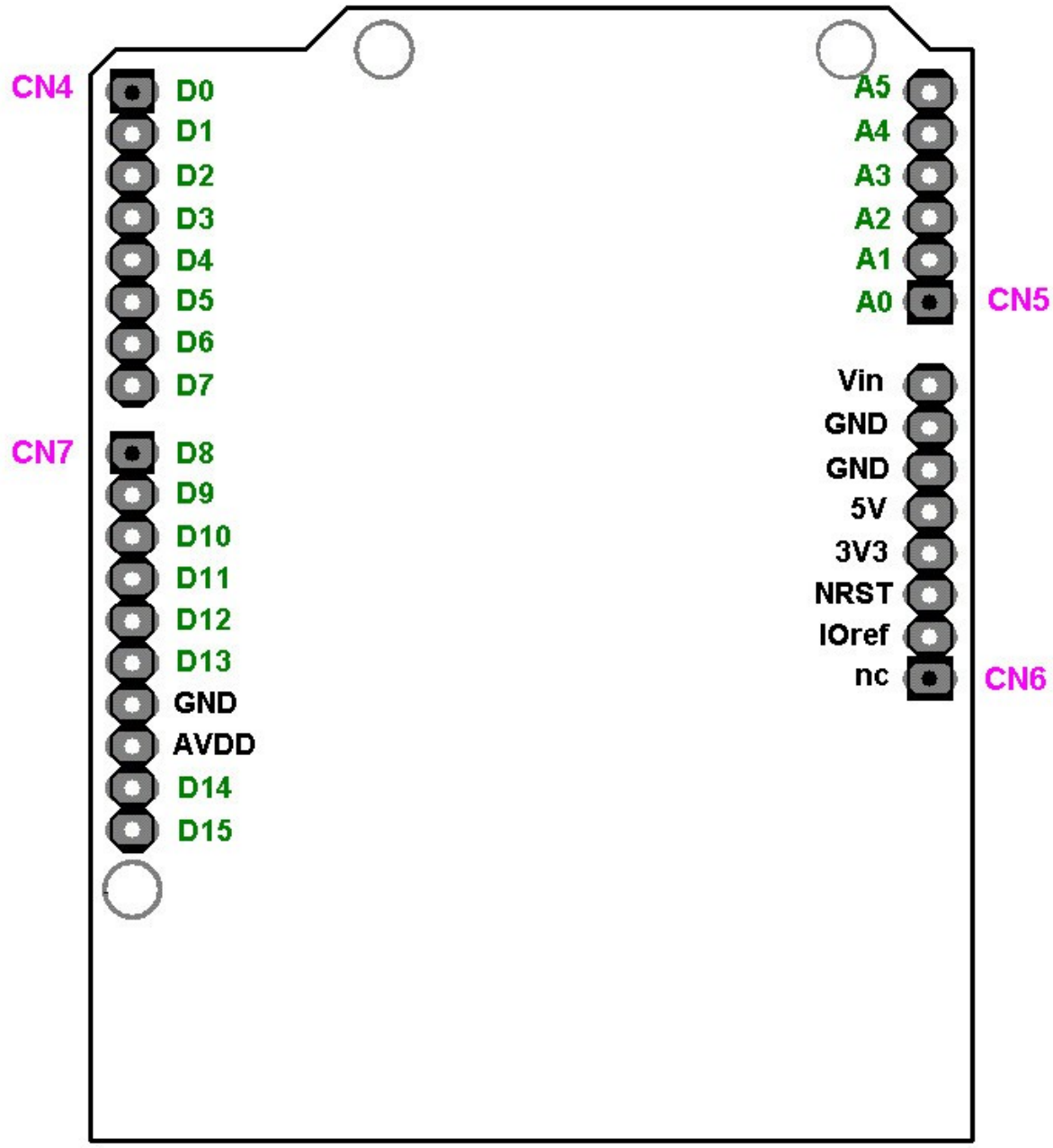
Device	**Function**	**IO-Connection**	**Basic-Command**
PCF8563 or PCF8586	RealTimeClock	I2C-Bus	RTC
MPU6050	3 axis Gyro and Acceleration-Sensor	I2C-Bus	MPU6050
Joystick	4 direction Joystick with 2 Firebuttons	6x Digital-IO	JoyInit JoyGet
DS18B20	Temperature sensor	1x Digital-IO	DS18B20
HC-SR04	Distance sensor	2x Digital-IO	DISTANCE
DHT-22	Humidity & Temperature sensor	1x Digital-IO	DHT22
IR	Infrared Sensor/Transmitter	1x Digital-IO	IR

5) GPIO-Ports

Pin-Nr / Name	CPU	Pinhead	Function-1	Function-2	Supported Modes
0	nc	nc	Not used		nc
1 / D0	PC7	CN4/1	COM1_RX		STD
2 / D1	PC6	CN4/2	COM1_TX		STD
3 / D2	PG6	CN4/3			STD
4 / D3	PB4	CN4/4			STD, IR
5 / D4	PG7	CN4/5			STD
6 / D5	PI0	CN4/6			STD, CIN, PIN, FIN
7 / D6	PH6	CN4/7			STD
8 / D7	PI3	CN4/8			STD, CIN, PIN, FIN
9 / D8	PI2	CN7/1			STD, CIN, PIN, FIN
10 / D9	PA15	CN7/2			STD
11 / D10	PA8	CN7/3	SPI1_SS		STD
12 / D11	PB15	CN7/4	SPI1_MOSI		STD
13 / D12	PB14	CN7/5	SPI1_MISO		STD
14 / D13	PI1	CN7/6	SPI1_SCK	User-LED	STD, CIN, PIN, FIN
15 / D14	PB9	CN7/9	I2C_SDA		STD
16 / D15	PB8	CN7/10	I2C_SCL		STD
17 / A0	PA0	CN5/1			STD,AIN
18 / A1	PF10	CN5/2			STD,AIN
19 / A2	PF9	CN5/3	COM2_CTS	SPI2_MOSI	STD,AIN
20 / A3	PF8	CN5/4	COM2_RTS	SPI2_MISO	STD,AIN
21 / A4	PF7	CN5/5	COM2_TX	SPI2_SCK	STD,AIN
22 / A5	PF6	CN5/6	COM2_RX	SPI2_SS	STD,AIN
23	PI11	nc		User-Button	DIN, INTL,INTH,INTB

STD = DIN, DOUT, INTL, INTH, INTB,

5.1) Pinout


CN4
D0
D1
D2
D3
D4
D5
D6
D7
CN7
D8
D9
D10
D11
D12
D13
GND
AVDD
D14
D15
A5
A4
A3
A2
A1
A0
CN5
Vin
GND
GND
5V
3V3
NRST
IOref
nc
CN6

6) Actual software state

In the following table i tried to comment all implemented features.

Should work :	the command should work like the original one
Not implemented yet :	the command is not implemented in the current version
Changed syntax :	the command has a different syntax to the original one
Not necessary :	the command is not used in F746-Version

Please use the MMBasic Language Manual from Geoff for a detailed description
of all the functions, commands and the syntax.

If you found errors in the list or bugs in the software,
please write a comment and send it back to me so i can correct it in the next version.

7) Operators

Here the arithmetic and logic operators

^	exponentiation	a = b ^ c
*	multiplication	a = b * c
/	division	a = b / c
\	integer division	a = b \ c
Mod	modulus	a = b MOD c
+	addition	a = b + c ; a$ = b$ + c$
-	subtraction	a = b - c
NOT	logical NOT	a = NOT b
AND	logical AND	a = b AND c
OR	logical OR	a = b OR c
XOR	logical XOR	a = b XOR c
<<	logical shift left	a = b << c
>>	logical shift right	a = b >> c
"=", "<", ">", "<=", ">=", "< >"	comparison	if(a=3) then print "a=3" if(a>=3) then print "a>=3"
"&b" "&h"	binary format hex format	a = &b10100011 a = &hA3

8) Predefined read only variables

MM.HRES	Should work
MM.VRES	Should work
MM.HPOS	Should work
MM.VPOS	Should work
MM.VER	Should work
MM.DEVICE$	Should work
MM.DRIVE$	Should work
MM.FNAME$	Should work
MM.CMDLINE$	?
MM.ERRNO	?
BLACK	Changed syntax : RGB(BLACK)
BLUE	Changed syntax : RGB(BLUE)
GREEN	Changed syntax : RGB(GREEN)
CYAN	Changed syntax : RGB(CYAN)
RED	Changed syntax : RGB(RED)
PURPLE	Changed syntax : RGB(PURPLE)
YELLOW	Changed syntax : RGB(YELLOW)
WHITE	Changed syntax : RGB(WHITE)
MM.I2C	Should work
MM.ONEWIRE	Not implemented yet

9) Commands

program command-line	Not implemented yet
' (single quotation mark)	Should work
? (question mark)	Should work
AUTO	Should work
BLIT	Changed syntax : BLIT x,y,w,h,xd,yd,roll x,y = source position w,h = size xd,yd = destination offset roll = 0:no, 1=yes
CHAIN	Not implemented yet
CHDIR	Should work
old : CIRCLE (x,y),r [,c [,aspect [,F]]]	Changed syntax : CIRCLE (x,y),r [,c1 [,c2 [,F]]] c1=outcolor, c2=incolor if aspect <> 1 use the new command ELLIPSE
CLEAR	Should work
CLOSE #nbr	Should work (Files not implemented yet)
CLOSE CONSOLE	Should work
CLS	Should work
COLOR	Use COLOUR
CONFIG	Should work Implemented settings : VIDEO [ON / OFF] FONT [1...6] CASE [LOWER / UPPER / TITLE] KEYBOARD [US / FR / GR] TAB [2 / 4 / 8] DRIVE [A / B / C] BGCOLOR [0...65535] FGCOLOR [0...65535] AUTORUN [ON / OFF] BAUDRATE [2400...115200] --> restart
CONTINUE	Should work
COPY	Not implemented yet
COPYRIGHT	Should work

DATA	Should work
DATE$	Should work
DELETE	Should work
DIM	Should work
DO / LOOP	Should work
DRIVE	Should work a: = Flash (not implemented yet) b: = uSD c: = USB-Drive
DS18B20	Should work
EDIT	Should work
ELSE / ELSEIF	Should work
END	Should work
END FUNCTION / END SUB	Should work
ERASE	Should work
ERROROK	Should work
EXIT	Should work
FILES	Should work (wildcards '*' or '?')
FONT	Should work
FONT LOAD / FONT UNLOAD	Not implemented yet
FOR	Should work
FUNCTION	Should work
GOSUB	Should work
GOTO	Should work
IF / THEN / ELSE	Should work
INPUT	Should work
INPUT #nbr	Should work (Files not implemented yet)
IR	Should work
IRETURN	Should work
KILL	Should work
KEYPAD	Not implemented yet
LET	Should work
LCD	Not implemented yet
LIBRARY	Not implemented yet

LINE	Should work
LINE INPUT	Should work
LINE INPUT #nbr	Not implemented yet
LIST	Should work
LOAD	Should work
LOADBMP "File.bmp" [,x[,y]]	Changed syntax : LOADBMP "File.bmp" [,x[,y[,layer]]]
LOCAL	Should work
LOOP	Should work
MEMORY	Should work
MERGE	Not implemented yet
MKDIR	Should work
MODE	Not necessary in F746-Version
NAME	Not implemented yet
NEW	Should work
NEXT	Should work
ON	Should work
ON KEY	Should work
OPEN „fnam“	Not implemented yet
OPEN „comspec“ as #nbr or OPEN „comspec“ as console comspec : COMn:bd,buf,int,intlevel,FC,DE,OC,S2 e.g. : open "COM1:" as 1	Opens a seriell port „COM1“ or „COM2“ bd = 115200, 57600, 38400, 19200, **9600**, 4800 fixed buf size interrupt not implemented yet FC only at COM2 DE not implemented yet OC not implemented yet S2 = two stopbits
OPTION	Should work Implemented settings : BASE, ERROR, PROMT, USB, BREAK, VIDEO, Fnn
PAUSE	Should work
PIN	Should work
PIXEL	Should work
PLAYMOD	Not implemented yet
POKE	Not implemented yet
PORT	Should work

PRINT	Should work
PRINT @	Should work
PRINT #nbr	Should work (Files not implemented yet)
PULSE	Should work
PWM	Not implemented yet
QUIT	Not necessary in F746-Version
RANDOMIZE	Should work
READ	Should work
REM	Should work
RESTORE	Should work
RETURN	Should work
RMDIR	Should work
RTC RTC GETTTIME [typ] RTC SETTIME y,m,d,h,m,s [,typ]	Changed syntax : [typ] can be : PCF8563 (default) PCF8583 (in this case year must be 2015-2018)
RUN	Should work
SAVE	Should work
SAVEBMP	Should work
SCANLINE	Not necessary in F746-Version
SEEK #fnbr	Not implemented yet
SETPIN pin,mode [,resistor]	Should work Implemented Modes : OFF, DIN, DOUT,AIN, FIN, PIN, CIN (DIN can have a 3rd parameter for resistor) NONE,UP,DOWN (default is "none")
SETPIN pin,mode [,resistor], target	Should work Implemented Modes : OFF, INTL, INTH, INTB can have a 3rd parameter for resistor NONE,UP,DOWN (default is "none")
SETTICK	Should work
SPRITE	Should work
SUB	Should work
SYSTEM	Not necessary in F746-Version
TIME$	Should work
TIMER	Should work

TONE	Not implemented yet
TROFF	Should work
TRON	Should work
WATCHDOG	Not implemented yet
XMODEM	Should work

10) Functions

ABS	Should work
ASC	Should work
ATN	Should work
BIN$	Should work
CHR$	Should work
CINT	Should work
CLR$	Should work
COLLISION	Not implemented yet
COS	Should work
CWD$	Should work
DATE$	Should work
DEG	Should work
DIR$	Not implemented yet
DISTANCE	Should work
DS18B20	Should work
EOF(#nbr)	Should work (Files not implemented yet)
EXP	Should work
FIX	Should work
FORMAT$	Should work
HEX$	Should work
INKEY$	Should work
INPUT$(nbr, [#]fnbr)	Should work (Files not implemented yet)
INSTR	Should work
INT	Should work
KEYDOWN	Should work
LEFT$	Should work
LEN	Should work
LOC([#]fnbr)	Should work (Files not implemented yet)
LOF([#]fnbr)	Should work (Files not implemented yet)
LOG	Should work

LCASE$	Should work
MID$	Should work
OCT$	Should work
PEEK	Not implemented yet
PI	Should work
PIN	Should work
PORT	Should work
POS	Should work
PIXEL	Should work
RAD	Should work
RIGHT$	Should work
RND	Should work
SGN	Should work
SIN	Should work
SPACE$	Should work
old : SPI(rx,tx,clk[,dat[,speed]])	Changed syntax : SPI(port_nr,dat) port_nr is 1 or 2 for SPI-1 or SPI-2 dat is the data byte to be send
SQR	Should work
STR$	Should work
STRING$	Should work
TAB	Should work
TAN	Should work
TIME$	Should work
TIMER	Should work
UCASE$	Should work
VAL	Should work

11) Other MMBasic commands

CAN	Not implemented yet
I2C	Should work Only in Master-Mode Speed = 10kHz to 400 kHz Option must be 0
ONEWIRE	Should work
TOUCHVAL	Should work
TOUCH	Should work with TouchObj nr [0...31] Implemented Modes : CREATE (B,P,S,C,R,L,H,V) SIZE (default = 80 x 25) DISABLE ENABLE REMOVE hint : RadioButtons (R) with the same color are in the same group hint : Push-Buttons (B) must be cleared by user with TouchVal(nr)=0 hint : Obj S,H,V don't use the Text-Field hint : size of Vertical-Sliders (V) must be defined first e.g. with "Touch Size 25,80"

TOUCHED(#S)
or
TOUCHED(#X)
or
TOUCHED(#Y)
or
TOUCHED(nr)
	Should work
	Implemented Modes :
	#S = reads the status (0=released, 1=pressed)
	#X = reads the x-position
	#Y = reads the y-position
	nr = check the touch status of TouchObj nr (0=untouched, 1=touched)

DHT22	Should work (see Micromite manual)
PULSEIN	Should work (see Micromite manual)

12) New commands for the F746-MMBasic

MTOUCHED(#S) or MTOUCHED(#X,nbr) or MTOUCHED(#Y,nbr)	#S: must be done before #X and #Y reads the number of touch contacts (0 to 5) #X: reads the x-position from contact (nbr=1 to 5) #Y: reads the y-position from contact (nbr=1 to 5)
MM.FONTHEIGHT MM.FONTWIDTH	current font height in pixel current font width in pixel
ACOS, ASIN, CALCCOS, CALCSIN	trigonometric functions
<<, >>	logic functions (shift left, shift right)
INTRND	random integer (0...65535)
ORANGE, BROWN, LRED, DGREY, GREY, LGREY, LGREEN, LBLUE	default colors Use as RGB(required_colour)
RGB(colour) or RGB(red,green,blue)	See Micromite manual
SPI OPEN port_nr [,speed [,mode [,first [,SS]]] or SPI CLOSE port_nr	opens or close a SPI-Port port_nr is 1 or 2 for SPI-1 or SPI-2 speed is optional and can be 0 to 7 mode is optional and can be 0 to 3 first is optional : 'L' or 'M' for LSB/MSB 'SS' is optional and activates the SlaveSelect-Pin
TRIANGLE ELLIPSE QUAD RECT x,y,width,height,c1[,c2,F] ROUNDRECT x,y,width,height,r,c1[,c2,F]	draws a triangle draws a ellipse draws a rectangle (with a given angle) draws a rectangle (F=filled) draws a rounded recrangle (r=radius, F=filled)

SETLAYER #nr
a = GETLAYER
SHOWLAYER #nr
COPYLAYER #source,#destination
	set the active layer to draw into
	get the active layer number
	show the layer at the lcd
	copy one layer to another
	(#nr = layer number 0..2)

MAP	draws a map
LOADJPG "File.jpg" [,x[,y[,layer]]]	loads a JPG and draw at x,y on a layer
WAVE LOAD "File.wav",ID WAVE PLAY ID,Volume WAVE LOOP ID,Volume WAVE STOP ID WAVE CLEAR	load one WAVE File (ID = 0 to 20) play one WAVE File (once) play one WAVE File (in a loop) Stop playing Clear all WAVE Files in RAM

OBJ3D	draws a 3d object
POLY POINT POLY CENTER POLY POLY MOVE POLY ROTATE POLY LOAD	defines a polygon point defines a center point from a polygon draws a polygon moves a polygon rotates a polygon load a polygon (use Polygon.exe to create)
MPU6050 INIT MPU6050(#S) or MPU6050(#GyroX) MPU6050(#GyroY) MPU6050(#GyroZ) or MPU6050(#AccX) MPU6050(#AccY) MPU6050(#AccZ) or MPU6050(#Temp)	Inits a MPU6050 on I2C-Port #S must be done before #Gyro and #Acc reads all Data from the MPU (-1=err, 0=ok) #Gyro reads the Gyro-Data #Acc reads the Accelerometer-Data #Temp reads the Temperature
JoyInit JoyGet	Inits a Joystick at Digital-IOs Read one Joystick direction

13) Syntax of the new commands in F746-MMBasic

13.1) aCos

aCos(x)

Give back the inverse Cosine of x

13.2) aSin

aSin(x)

Give back the inverse Sine of x

13.3) CalcCos

CalcCos(angle)

Give back the pre computed Cosine from Angle , the angle is rounded to integer.

13.4) CalcSin

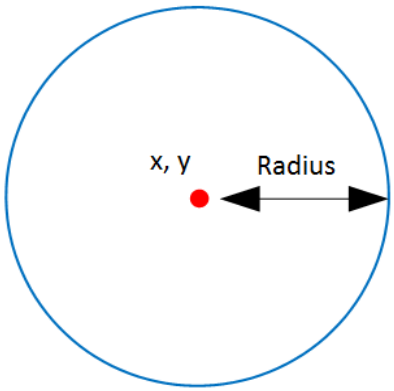
CalcSin(angle)

Give back the pre computed Sine from Angle , the angle is rounded to integer.

13.5) Circle

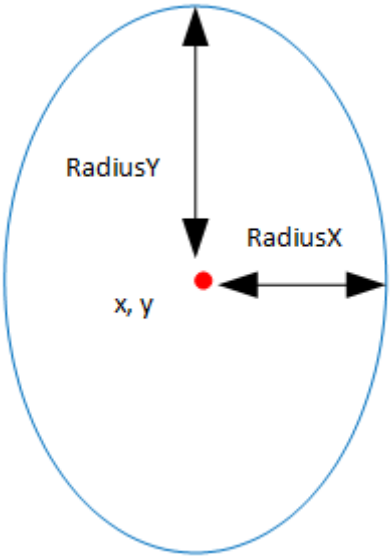
Circle(x,y),radius,outcolor[,incolor[,F]]

x,y	: Coordinate from the Circle center
radius	: Radius from the Circle
outcolor	: Circle border color
incolor	: Circle Fill color
F	: Circle is filled



13.6) Ellipse

Ellispse(x, y), radiusx, radiusy, outcolor, incolor, F

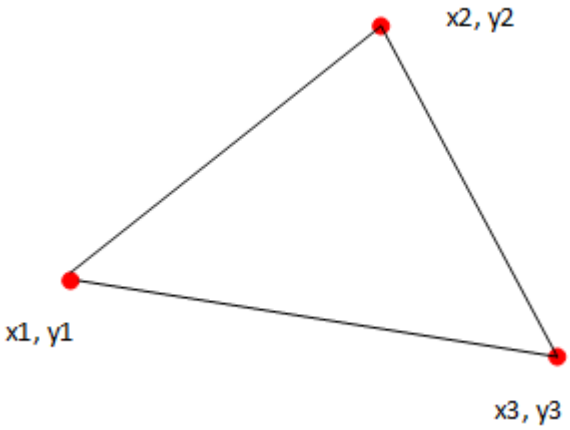


x, y	: Coordinate from the Ellipse center
radiusx	: Radius in the x direction (left/right)
radiusy	: Radius in the y direction (up/down)
outcolor	: Border color
incolor	: Fill color
F	: Ellipse is filled

13.7) Triangle

Triangle(x1, y1, x2, y2, x3, y3), outcolor, incolor, F

x1, y1 to x3, y3	: The 3 coordinate that represent your triangle
outcolor	: The border color
incolor	: The fill color
F	: The triangle is filled



13.8) Sprites

Sprite are no more compatible with the PIC32 Maximite version.
The Sprite Editor can Load the 2 colors and 8 colors Sprite from Maximite and convert them to the new Binary format.

You can load up to 500 Sprites, each of them can be up to 32 x 32 pixels.

Syntax:

Sprite Load "spritefile.bin"

The command will Load all sprites from the file and put them in memory.
The first Sprite loaded will be the Sprite 0.

Sprite Set source,destination

Information from the Sprite Source will be copied into the destination array to be able to use this Sprite. For example, if you want to animate Sprites you just have to copy another source sprite number into the same destination one.

Sprite ON Sprite_number, x, y

The Sprite will be shown at coordinate x, y on screen.
Note that the Sprite number have to be set with Sprite Set command in first case.
The x, y coordinate represent the upper left corner from the sprite.

13.9) Maps

With map you can load Games map for use with your game, you can also use it to store another picture. This map is saved into the SDRAM and can be up to 1Million pixels.

The Height and Width from the map is not limited so long the (Height * Width) is less than 1 Million Pixels. For be able to load a Map file you have to convert your picture with the Map converter.

Syntax:

Map Load "mapfile.map"

The command will Load the Map from the file and put it in Map memory.

Map ON startx, starty, width, height, destx, desty, transparent, transparent color.

The command will get a part from the map graphics and show it on screen.
If you get a part of the Map that is bigger than the screen (480*272), the result will be clipped on screen.

13.10) Pixel

The pixel function is still compatible with the original Maximite, but, I added the ability to get the pixel color from Any of the Layer.

Syntax:

Color = Pixel(x, y)

The function return the actual color from the pixel at x,y from the current used graphics layer.

Color = Pixel(x, y, layer)

This one return the pixel color at x, y from the defined layer.
Layer number can be 0, 1, 2 or 10 for Map buffer. (See Layer command)

13.11) Layer

The STM32F7 can use 2 layer for build picture on screen.
You can set a layer, draw on this layer, then show the layer.
This method prevent flickering on screen
I added one working layer that can be copied to one of the other layer.
It's good for put in GUI graphics that never change or other graphics element we want often copy to other layer.

Syntax :

SetLayer number

Set the layer where we want draw
Number can be 0, 1, 2

ShowLayer number

Show the layer set with number
Number can be 0 or 1

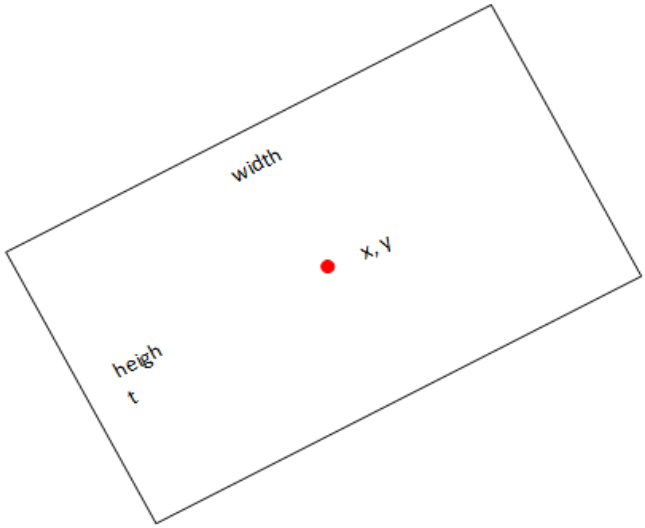
CopyLayer source, destination

Copy from one layer to another one
Source and destination can be 0, 1, or 2
You cannot copy one layer to itself!

13.12) Quad

Quad(x, y, width, height, angle), outcolor, incolor, F

The Quad is a rectangle like polygon that can be rotated
x, y : Middle coordinate from the Quad
width, height : Size of the Quad
angle : Rotation angle from the Quad
outcolor : border color
incolor : Fill color
F : The Quad is filled



13.13) 3D-Objects

You have the ability to show 3D Object on this version of MMBasic.

The actual limitation are:

You can load a maximum of 25 object (0 to 24)
Each object can have a maximum of 600 Vertex and 600 Faces

The 3D Object converter take 3D Studio ASC file format and save it in a form that can be read with MMBasic. (make sure to deactivate "UV")

For convert OBJ, DAE etc. … to 3DS ASC, I use AcuTrans3D.
The not registered version can save in ASC and its easy to use!

Syntax:

Obj3D Load "obj.b3d", Objnumber, incolor, outcolor

Here we Load the file "obj.b3d"
Objnumber can be 0 to 24
Incolor is the border color from vertices
Outcolor is the fill color

Obj3D Set Objnumber, x, y, z, ax, ay, az, zoom, active

We give parameter to a loaded 3D Object for specify where we want it to draw, the angle and the size.

Objnumber	: Number from the object from 0 to 24
x, y, z	: Coordinate from the object in 3D Space
ax, ay, az	: Angle to apply on each axis from the object
Zoom	: For resize the object, zoom of 1 will show the original size
Active	: 1 will set this object active, 0 will set the object inactive and it will not be shown

Object3D ON filled

We show all loaded and active object to screen.

Filled	: If filled is set to 1, all 3D Objects will be filled

13.14) Polygons

The easy way to draw little more complex form is to use the Polygon command (or **Polygon.exe**).
A polygon is defined from a list of points.
Actually a polygon can have up to 100 points (0 to 99)
You can define 100 different Polygons, Polygon number must be between 0 and 99
They are several easy to use commands for set , move, rotate Polygons (and load from file).

Syntax:

Poly Point Poly_number, Point_number, x, y

We have to use this command to set all points from a polygon.
Poly_number : Number from the Polygon (0 to 99)
Point number : Number from the point we set the coordinate (0 to 99)
x, y : Coordinate from this point

Poly Center Poly_number, x,y

We set the center point from the Polygon
This point will be used if you want rotate the Polygon
This point can be set everywhere, you don't have to place it into the Polygon itself.
Poly_number : Number from the Polygon (0 to 99)
x, y : Coordinate from the Polygon center point

Poly Poly_number, Point_count, outcolor [, incolor[, F]]

Draw a Polygon on screen.
Poly_number : Number from the Polygon (0 to 99)
Point_count : Number of point defining this polygon, if you have defined point 0 to 23 then this polygon have 24 points.
Outcolor : border color
Incolor : fill color
F : The polygon is filled 'optional'

Poly Move Poly_number, x,y

Use this command for move the Polygon somewhere on the screen.
It is like you move the center point and the rest of the Polygon follow it.
Poly_number : Number from the Polygon (0 to 99)
x, y : New Coordinate from the Polygon center point (All other points follow)

Poly Rotate Poly_number, Point_count, angle, outcolor [, incolor[, F]]

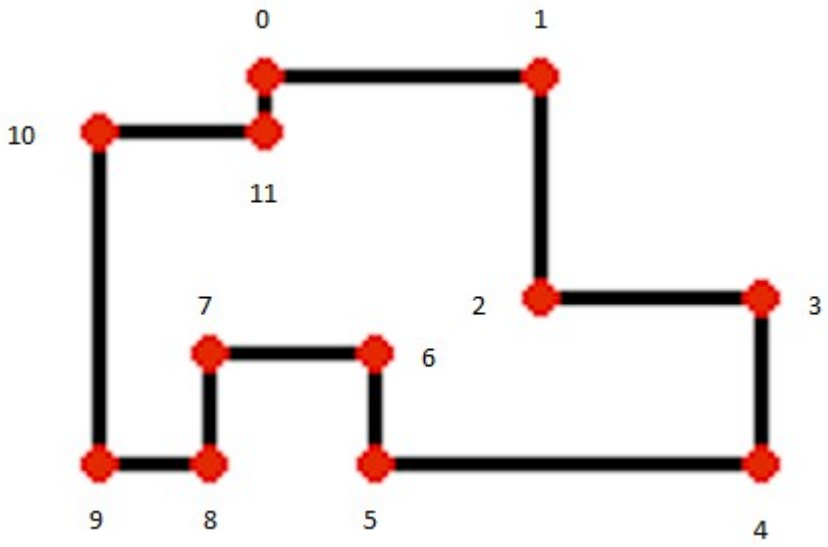
Draw a Polygon rotated to the given angle.
The Polygon will rotate around his Center point defined with the PolyCenter command.
Poly_number : Number from the Polygon (0 to 99)
Point_count : Number of point defining this polygon, if you have defined point 0 to 23 then this polygon have 24 points.
Angle : angle we rotate the polygon
Outcolor : border color
Incolor : fill color
F : The polygon is filled 'optional'

PolyPoint example :

Point 0 to 11 define this Polygon , it's a 12 points Polygon

Poly Point 0,0,100,20 set the point 0 from polygon 0 to coordinate 100 , 20
Poly Point 0,1,200,20 set the point 1 from polygon 0 to coordinate 200 , 20
……
Poly Point 0,11,100,30 set the point 11 from polygon 0 to coordinate 100 , 30

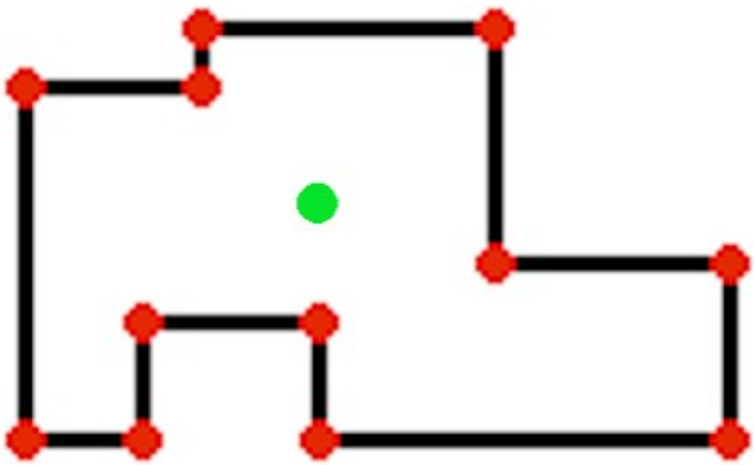
You will get a polygon with this form :



PolyCenter example :

Now we are going to set the Center from the Polygon ,
or if you prefer the coordinate from where the Polygon can be rotated

Poly Center 0,140,60 set the Center point to coordinate 140 , 60



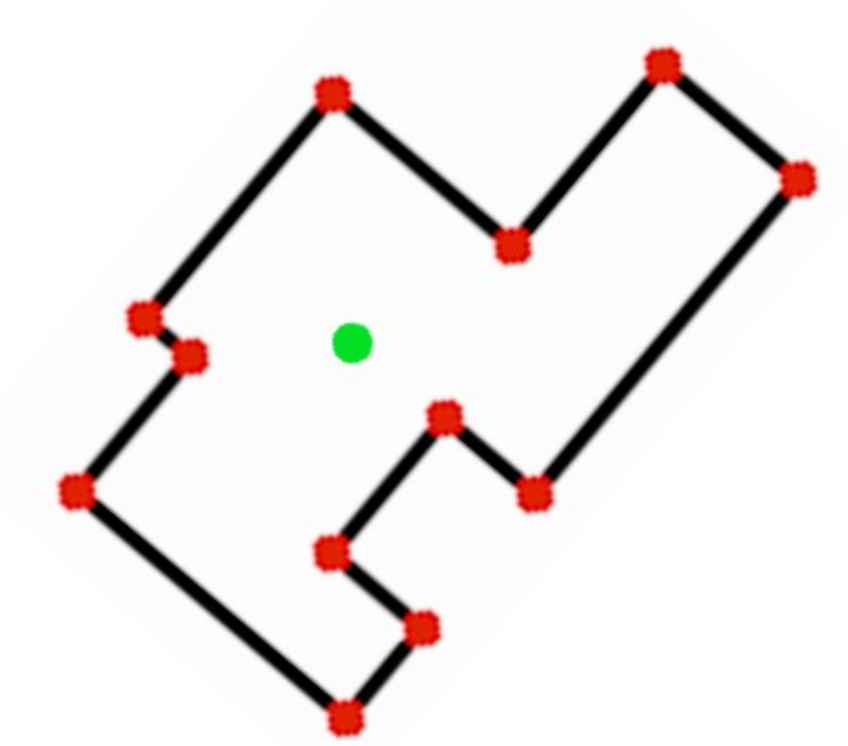
Polygon example :

The polygon is now complete and can be drawn on screen with :
Poly 0,12,Red draw the 12 points polygon 0 with Red outer line
Poly 0,12,Red,Blue,F draw the 12 points polygon 0 with Red outer line and Filled with Blue color.

PolyRotate example :

You can also Rotate the polygon like this :

Poly Rotate 0,12,45,Red It will show the polygon rotated to 45 degrees around the center point.

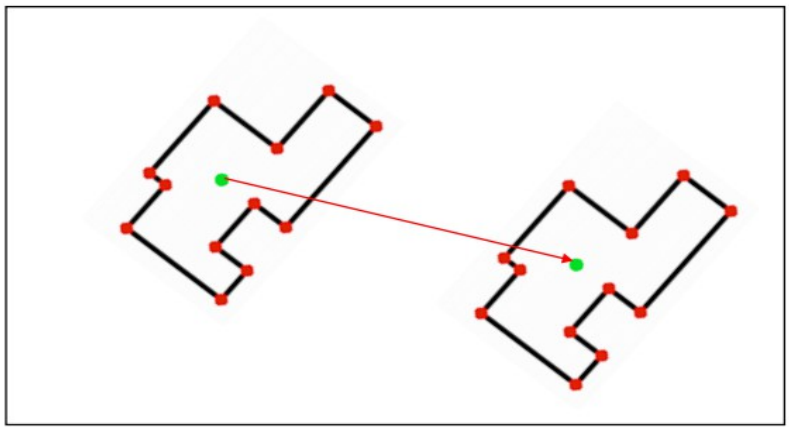


Note that the center point is invisible and just drawn here for information.

PolyMove example :

Now we are going to move the Polygon somewhere else on the screen.

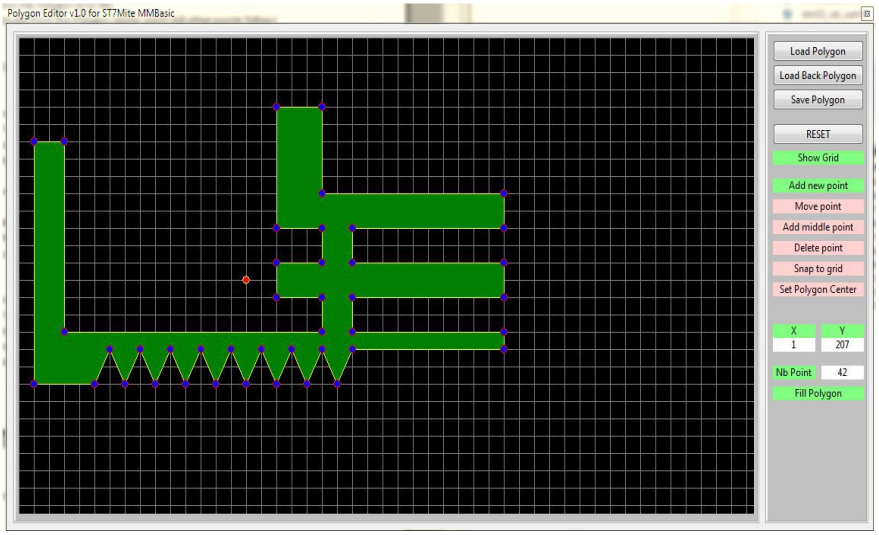
Poly Move 0,340,150 Move the polygon from the original 140,60 coordinate to 340,150 coordinate.
This will move the Center point with the Polygon
For move the Center point alone use the PolyCenter command.



PolyLoad example :

Instead of creating a polygon at runtime with the "polypoint" command you can also use the external program "Polygon.exe". Here you can create polygons and save it to a file.

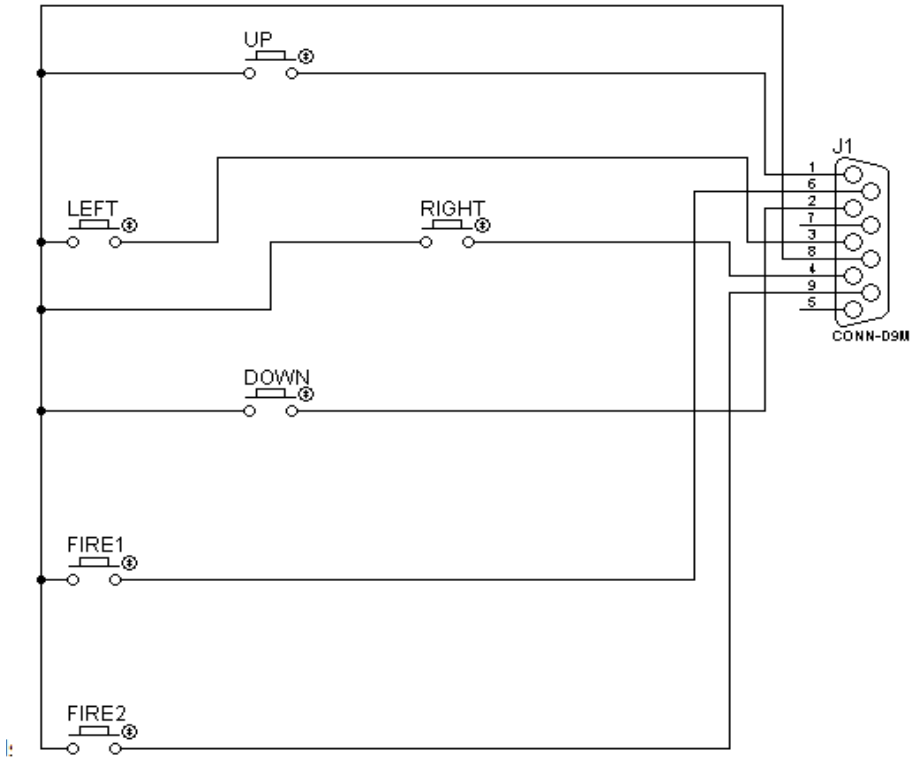
To load a polygon you have to know the number of points (see "Nb Point")



Syntax : **Poly Load "Filename.pol" , Poly_number** Loads a polygon from file

13.15) Joystick

You can connect a Joystick with 4 directions and two firebuttons at the GPIO-Pins :



JoyInit(up, down, left, right, button1, button2)

Use this command to init a Joystick at the GPIO-Pins.
Each of the parameter correspond to a pin number on the Arduino connector.

For example , if you use : JoyInit(3,4,5,6,7,8)
Pin 3 is UP
Pin 4 is DOWN
Pin 5 is LEFT
Pin 6 is RIGHT
Pin 7 is Fire button 1
Pin 8 is Fire button 2

The JoyInit command setup the needed pin and activate the internal pull-up resistor.
It means that the status from a non-used direction or button will be 1.

JoyGet(#UP) **:** Give back the UP direction
JoyGet(#DOWN) **:** Give back the DOWN direction
JoyGet(#LEFT) **:** Give back the LEFT direction
JoyGet(#RIGHT) **:** Give back the RIGHT direction
JoyGet(#B1) **:** Give back the Button1 status
JoyGet(#B2) **:** Give back the Button2 status