**TTGO Display ESP32, notes pour programmation de l’écran**

**L'écran TFT est: IPS ST7789V de 1,14 pouces 135x240 avec interface SPI, soit 32400 octets (voir fichier bmp.h qui affiche le logo TTGO)**

**Il n’y a pas de défilement automatique, donc quand l’écran est rempli, avant de pouvoir afficher plus il faut le rafraîchir (ex: println(xxx) au max 14 lignes sur l’écran dans sa plus grande dimension)**

**Variables pour positionnement sur l’écran :**

TL\_DATUM 0 = Top left (default)

TC\_DATUM 1 = Top centre

TR\_DATUM 2 = Top right

ML\_DATUM = Middle left

CL\_DATUM = Centre left, same as above

MC\_DATUM = Middle centre

CC\_DATUM = Centre centre, same as above

MR\_DATUM = Middle right

CR\_DATUM = Centre right, same as above

BL\_DATUM = Bottom left

BC\_DATUM = Bottom centre

BR\_DATUM = Bottom right

L\_BASELINE = Left character baseline (Line the 'A' character would sit on)

C\_BASELINE = Centre character baseline

R\_BASELINE = Right character baseline

**Variables de couleurs :**

TFT\_BLACK 0x0000 /\* 0, 0, 0 \*/

TFT\_NAVY 0x000F /\* 0, 0, 128 \*/

TFT\_DARKGREEN 0x03E0 /\* 0, 128, 0 \*/

TFT\_DARKCYAN 0x03EF /\* 0, 128, 128 \*/

TFT\_MAROON 0x7800 /\* 128, 0, 0 \*/

TFT\_PURPLE 0x780F /\* 128, 0, 128 \*/

TFT\_OLIVE 0x7BE0 /\* 128, 128, 0 \*/

TFT\_LIGHTGREY 0xC618 /\* 192, 192, 192 \*/

TFT\_DARKGREY 0x7BEF /\* 128, 128, 128 \*/

TFT\_BLUE 0x001F /\* 0, 0, 255 \*/

TFT\_GREEN 0x07E0 /\* 0, 255, 0 \*/

TFT\_CYAN 0x07FF /\* 0, 255, 255 \*/

TFT\_RED 0xF800 /\* 255, 0, 0 \*/

TFT\_MAGENTA 0xF81F /\* 255, 0, 255 \*/

TFT\_YELLOW 0xFFE0 /\* 255, 255, 0 \*/

TFT\_WHITE 0xFFFF /\* 255, 255, 255 \*/

TFT\_ORANGE 0xFDA0 /\* 255, 180, 0 \*/

TFT\_GREENYELLOW 0xB7E0 /\* 180, 255, 0 \*/

TFT\_PINK 0xFC9F

**Classe**

TFT\_eSPI (largeur, hauteur) classe associée paramètres largeur, hauteur de l’écran

**Fonctions (méthodes de la classe TFT\_eSPI)**

// These are virtual so the TFT\_eSprite class can override them with sprite specific functions

drawPixel(int32\_t x, int32\_t y, uint32\_t color),

drawChar(int32\_t x, int32\_t y, uint16\_t c, uint32\_t color, uint32\_t bg, uint8\_t size),

drawLine(int32\_t x0, int32\_t y0, int32\_t x1, int32\_t y1, uint32\_t color),

drawFastVLine(int32\_t x, int32\_t y, int32\_t h, uint32\_t color),

drawFastHLine(int32\_t x, int32\_t y, int32\_t w, uint32\_t color),

fillRect(int32\_t x, int32\_t y, int32\_t w, int32\_t h, uint32\_t color);

virtual int16\_t drawChar(uint16\_t uniCode, int32\_t x, int32\_t y, uint8\_t font),

drawChar(uint16\_t uniCode, int32\_t x, int32\_t y),

height(void),

width(void);

// The TFT\_eSprite class inherits the following functions

void setWindow(int32\_t xs, int32\_t ys, int32\_t xe, int32\_t ye),

pushColor(uint16\_t color),

pushColor(uint16\_t color, uint32\_t len),

pushColors(uint16\_t \*data, uint32\_t len, bool swap = true), // With byte swap option

pushColors(uint8\_t \*data, uint32\_t len),

fillScreen(uint32\_t color);

void drawRect(int32\_t x, int32\_t y, int32\_t w, int32\_t h, uint32\_t color),

drawRoundRect(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, int32\_t radius, uint32\_t color),

fillRoundRect(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, int32\_t radius, uint32\_t color),

setRotation(uint8\_t r) : r= 0 affichage sur la largeur (petite valeur), r=1 affichage sur la longueur

invertDisplay(boolean i),

drawCircle(int32\_t x0, int32\_t y0, int32\_t r, uint32\_t color),

drawCircleHelper(int32\_t x0, int32\_t y0, int32\_t r, uint8\_t cornername, uint32\_t color),

fillCircle(int32\_t x0, int32\_t y0, int32\_t r, uint32\_t color),

fillCircleHelper(int32\_t x0, int32\_t y0, int32\_t r, uint8\_t cornername, int32\_t delta, uint32\_t color),

drawEllipse(int16\_t x0, int16\_t y0, int32\_t rx, int32\_t ry, uint16\_t color),

fillEllipse(int16\_t x0, int16\_t y0, int32\_t rx, int32\_t ry, uint16\_t color),

drawTriangle(int32\_t x0, int32\_t y0, int32\_t x1, int32\_t y1, int32\_t x2, int32\_t y2, uint32\_t color),

fillTriangle(int32\_t x0, int32\_t y0, int32\_t x1, int32\_t y1, int32\_t x2, int32\_t y2, uint32\_t color),

drawBitmap(int16\_t x, int16\_t y, const uint8\_t \*bitmap, int16\_t w, int16\_t h, uint16\_t color),

drawXBitmap(int16\_t x, int16\_t y, const uint8\_t \*bitmap, int16\_t w, int16\_t h, uint16\_t color),

drawXBitmap(int16\_t x, int16\_t y, const uint8\_t \*bitmap, int16\_t w, int16\_t h, uint16\_t fgcolor, uint16\_t bgcolor),

setBitmapColor(uint16\_t fgcolor, uint16\_t bgcolor), // For 1bpp sprites

setPivot(int16\_t x, int16\_t y),

setCursor(int16\_t x, int16\_t y),

setCursor(int16\_t x, int16\_t y, uint8\_t font),

setTextColor(uint16\_t color),

setTextColor(uint16\_t fgcolor, uint16\_t bgcolor) : couleurs du texte (fg) et du fond (bg)

setTextSize(uint8\_t size),

setTextWrap(boolean wrapX, boolean wrapY = false),

setTextDatum(uint8\_t datum) : positionnement du texte (voir variable de positionnement)

setTextPadding(uint16\_t x\_width),

#ifdef LOAD\_GFXFF

setFreeFont(const GFXfont \*f = NULL),

setTextFont(uint8\_t font),

#else

setFreeFont(uint8\_t font),

setTextFont(uint8\_t font),

#endif

spiwrite(uint8\_t),

writecommand(uint8\_t c),

writedata(uint8\_t d),

commandList(const uint8\_t \*addr);

uint8\_t readcommand8(uint8\_t cmd\_function, uint8\_t index = 0);

uint16\_t readcommand16(uint8\_t cmd\_function, uint8\_t index = 0);

uint32\_t readcommand32(uint8\_t cmd\_function, uint8\_t index = 0);

// Read the colour of a pixel at x,y and return value in 565 format

uint16\_t readPixel(int32\_t x0, int32\_t y0);

// The next functions can be used as a pair to copy screen blocks (or horizontal/vertical lines) to another look

// Read a block of pixels to a data buffer, buffer is 16 bit and the array size must be at least w \* h

void readRect(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, uint16\_t \*data);

// Write a block of pixels to the screen

void pushRect(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, uint16\_t \*data);

// These are used to render images or sprites stored in RAM arrays

void pushImage(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, uint16\_t \*data);

void pushImage(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, uint16\_t \*data, uint16\_t transparent);

// These are used to render images stored in FLASH (PROGMEM)

void pushImage(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, const uint16\_t \*data, uint16\_t transparent);

void pushImage(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, const uint16\_t \*data);

// These are used by pushSprite for 1 and 8 bit colours

void pushImage(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, uint8\_t \*data, bool bpp8 = true);

void pushImage(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, uint8\_t \*data, uint8\_t transparent, bool bpp8 = true);

// Swap the byte order for pushImage() - corrects endianness

void setSwapBytes(bool swap);

bool getSwapBytes(void);

// This next function has been used successfully to dump the TFT screen to a PC for documentation purposes

// It reads a screen area and returns the RGB 8 bit colour values of each pixel

// Set w and h to 1 to read 1 pixel's colour. The data buffer must be at least w \* h \* 3 bytes

void readRectRGB(int32\_t x0, int32\_t y0, int32\_t w, int32\_t h, uint8\_t \*data);

uint8\_t getRotation(void),

getTextDatum(void),

color16to8(uint16\_t color565); // Convert 16 bit colour to 8 bits

int16\_t getCursorX(void),

getCursorY(void);

int16\_t getPivotX(void),

getPivotY(void);

uint16\_t fontsLoaded(void),

color565(uint8\_t red, uint8\_t green, uint8\_t blue), // Convert 8 bit red, green and blue to 16 bits

color8to16(uint8\_t color332); // Convert 8 bit colour to 16 bits

int16\_t drawNumber(long long\_num, int32\_t poX, int32\_t poY, uint8\_t font),

drawNumber(long long\_num, int32\_t poX, int32\_t poY),

drawFloat(float floatNumber, uint8\_t decimal, int32\_t poX, int32\_t poY, uint8\_t font),

drawFloat(float floatNumber, uint8\_t decimal, int32\_t poX, int32\_t poY),

// Handle char arrays

drawString(const char \*string, int32\_t poX, int32\_t poY, uint8\_t font),

drawString(const char \*string, int32\_t poX, int32\_t poY),

drawCentreString(const char \*string, int32\_t dX, int32\_t poY, uint8\_t font), // Deprecated, use setTextDatum() and drawString()

drawRightString(const char \*string, int32\_t dX, int32\_t poY, uint8\_t font), // Deprecated, use setTextDatum() and drawString()

// Handle String type

drawString(const String& string, int32\_t poX, int32\_t poY, uint8\_t font),

drawString(const String& string, int32\_t poX, int32\_t poY),

drawCentreString(const String& string, int32\_t dX, int32\_t poY, uint8\_t font), // Deprecated, use setTextDatum() and drawString()

drawRightString(const String& string, int32\_t dX, int32\_t poY, uint8\_t font); // Deprecated, use setTextDatum() and drawString()

int16\_t textWidth(const char \*string, uint8\_t font),

textWidth(const char \*string),

textWidth(const String& string, uint8\_t font),

textWidth(const String& string),

fontHeight(int16\_t font),

fontHeight(void);

void setAddrWindow(int32\_t xs, int32\_t ys, int32\_t w, int32\_t h);

// Compatibility additions

void startWrite(void); // Begin SPI transaction

void writeColor(uint16\_t color, uint32\_t len); // Write colours without transaction overhead

void endWrite(void); // End SPI transaction

uint16\_t decodeUTF8(uint8\_t \*buf, uint16\_t \*index, uint16\_t remaining);

uint16\_t decodeUTF8(uint8\_t c);

size\_t write(uint8\_t);