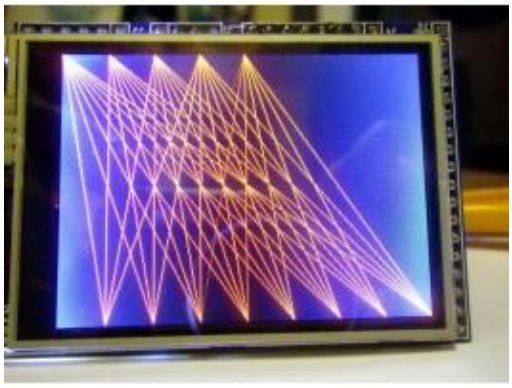
Module d'écran tactile TFT 240 x 320 pixels couleur +SD

Utilisation du module d'écran tactile TFT 240 x 320 pixels couleur avec lecteur SD



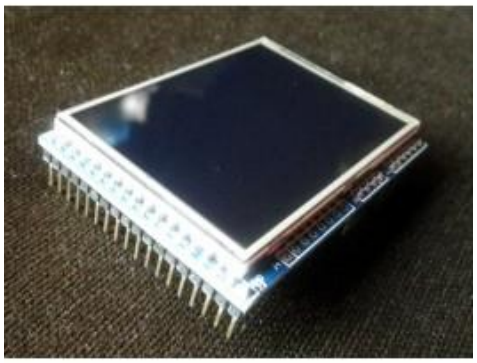
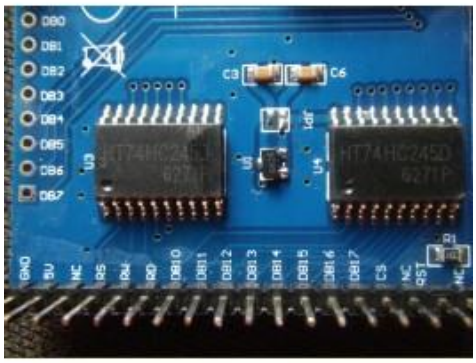
.

Matériel utilisé

- 1 carte Arduino UNO ou
- 1 carte Arduino Mega2560 (la librairie prend de la place en mémoire, éviter la carte Uno)
- 1 écran tactile TFT 240 x 320 pixels couleur
- Quinzaine de fils Dupont de branchement mâle / femelle
- 1 Alimentation 9V

Principe de fonctionnement
Ce module d'écran tactile 240x320 pixels couleurs est piloté par un driver ILI9325 IC.
Il possède trois fonctions :

- Affichage graphique
- Ecran tactile
- Lecteur enregistreur de données sur carte mémoire SD.

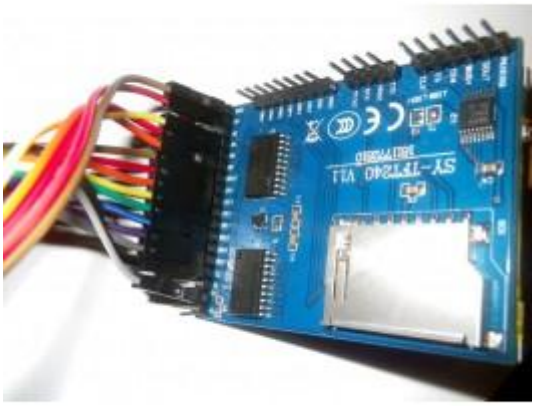
.

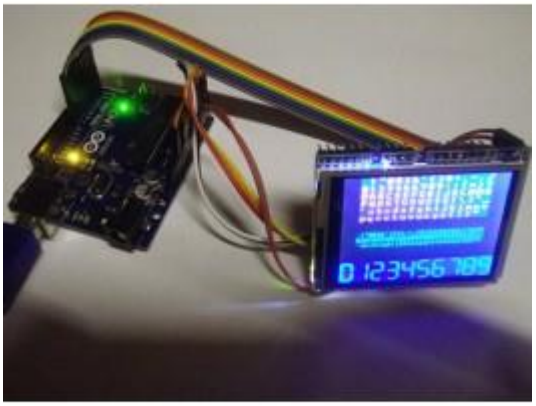
Caractéristiques du module écran tactile couleur TFT

- Ecran de 2.4 pouces
- Résolution de 240 x 320 pixels couleur
- Driver ILI9325 IC
- Pour micro contrôleurs AVR, 8051, PIC, cartes Arduino...
- Lecteur / enregistreur de carte mémoire SD intégré, en mode SPI.
- Alimentation 3.3V ou 5V, le TFT fonctionne sous 3.3V avec un régulateur que l'on peut shunter (J2).
- Alimentation de 5V par défaut.
- Taille du circuit 63 x 46 x 15 mm.
- Poids 35g

Brochage sur le module

- Alimentation : GND, 5V, NC
- Entrées 16 bits : RS, RW, RD,R0,BD0-DB7,CS,RTS, BD10-DB17
- Port SPI avec les 4 pins CS, MOSI, SCK, MISO pour carte SD
- Pen_Q, Dout, Busy, Din, CS, Dclk.



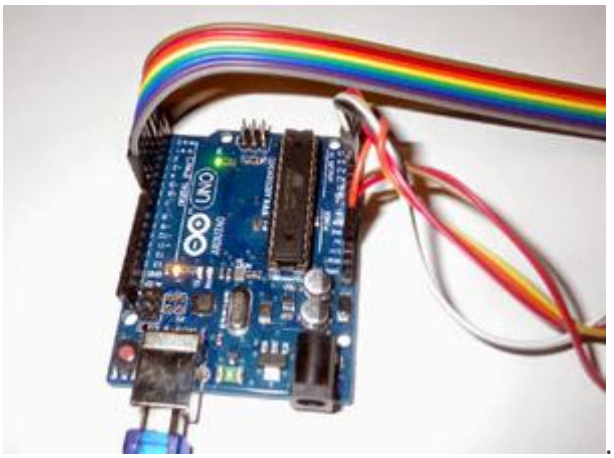


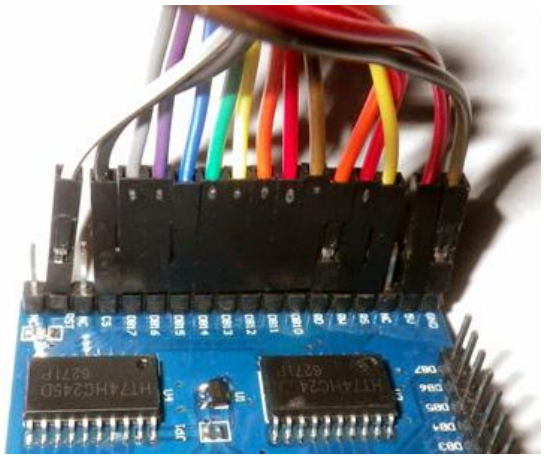
.

Câblage pour faire fonctionner l'écran seul

Il n'est pas nécessaire de souder une barrette de 8 pins sur le module d'écran sur les broches DB0-DB7.

En 8-bits on utilise les pins D10 à D17 du module.





Module écran --> Arduino

- Rétro éclairage 5V > 5V
- Rétro éclairage GND > GND
- DB 10 à DB 17 > Arduino pins digitales D0 - D7 respectivement (8 bits)
- RD > 3.3 V
- RSET > A2
- CS > A3
- RW > A4
- RS > A5

Librairie
On utilise la librairie Arduino *UTFT* de Henning Karlsen
Elle fonctionne sur les cartes Arduino Uno, Mega256, Due... mais on évitera la carte Uno pour avoir encore assez de mémoire flash pour son propre code.

Lancer Arduino.exe, IDE version 1.0.5, dans Tools/Boards choisir la carte Arduino MEGA 2560 (ou Arduino Uno selon le cas).
Vérifier que la librairie est installée avec sketch/Import Librairie

Si on ne trouve pas UTFT, il faut l'installer depuis ici
http://henningkarlsen.com/electronics/library.php?id=51
télécharger lefichier UTFT.rar, le décompresser et copier le dossier complet UTFT dans votre répertoire *Arduino/librairies*
Redémarrer l'IDE Arduino pour valider l'installation.

Exemples de programmation

Attention, ne pas utiliser Serial.print (broches Tx / Rx) en même temps que l'écran, qui utilise déjà les pins d0 et d1.
Avec une carte UNO, un message d'erreur est parfois normal (car pas assez de mémoire

flash).
On optimise la compilation en rajoutant pour UNO :
CODE:

```
#include <memorysaver.h>
```

L'écran est déclaré dans le code dans le void_setup()
CODE:

```
#include "UTFT.h"
UTFT myGLCD(ILI9325C,19,18,17,16); // Pour Arduino Uno
myGLCD.InitLCD(LANDSCAPE);   //Choix entre LANDSCAPE et PORTRAIT
myGLCD.clrScr();   //Vide l'écran
```

On indique le choix et le câblage de l'écran avec la syntaxe suivante
UTFT (model,RS,WR,CS,RST)

UTFT myGLCD(ILI9325C,19,18,17,16); // Pour Arduino Uno
en accord avec le branchement

- RS Register Select >A5=pin 19 (jaune)
- WR Write>A4=pin 18 (rouge)
- CS ChipSelect >A3=pin 17 (noir)
- RST Reset>A2=pin 16 (blanc)

Afficher un texte :
On dispose de 3 polices de caractères, à charger avec
CODE:

```
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];
```

L'affichage de texte nécessite d'indiquer les couleurs de texte et de fond
CODE:

```
myGLCD.setColor(red, green, blue);
myGLCD.setBackColor(red, green, blue);   //R G B entre 0 -255
```

Le choix de la fonte se fait à partir de
CODE:

```
myGLCD.setFont(SmallFont); // 20 lignes de 40 petits caractères
myGLCD.setFont(BigFont); // 15 lignes de 20 gros caractères
myGLCD.setFont(SevenSegNumFont); // chiffres 7 segments 0 à 9 sur 4 lignes
```

.

Exemple de code complet(1) : texte et chiffres

CODE:

```
//*************************************************
// Ecran graphique couleur tactile TFT
// tiptopboards.com modifié  03 11 2014
// affichage de texte etchiffres en 3 tailles
//
//*************************************************
#include <memorysaver.h>  //ajouté pour compiler sur la mémoire de UNO

// UTFT_ViewFont (C)2012 Henning Karlsen
// web: http://www.henningkarlsen.com/electronics
// modified by John Boxall, April 2013

#include <UTFT.h>

// Declaration des 3 fontes
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];

UTFT myGLCD(ILI9325C,19,18,17,16);    //Uno pins A2-A5,écran avec driver ILI9325C

void setup()
{
  myGLCD.InitLCD();
  myGLCD.clrScr();
}
```

```
void loop()
{  // couleurs de texte :
   //255,0,0 rouge   255,255,0 jaune  255,0,255 violet
   //0,255,0 vert   0,255,255 cyan  0,0,255  bleu  0,0,0 blanc
   // couleur de fond 0,0,0 noir  200,200,200 gris, etc... 255,255,255 blanc

  //myGLCD.setColor(0, 0,0);            //ecriture en noir
  //myGLCD.setBackColor(255, 255, 255);   //sur fond blanc

  //myGLCD.setColor(255, 255,255);   //ecriture en blanc
  myGLCD.setColor(255, 255,0);    //ecriture en jaune
  myGLCD.setBackColor(0, 0, 0);   //sur fond noir

  myGLCD.setFont(BigFont);    //En gros
  myGLCD.print(" !\"#$%&'()*+,-./", CENTER, 0);
  myGLCD.print("0123456789:;<=>?", CENTER, 16);
  myGLCD.print("@ABCDEFGHIJKLMNO", CENTER, 32);
  myGLCD.print("PQRSTUVWXYZ[\\]^_", CENTER, 48);
  myGLCD.print("`abcdefghijklmno", CENTER, 64);
  myGLCD.print("pqrstuvwxyz{|}~ ", CENTER, 80);

  myGLCD.setColor(0, 255, 0);  //vert
  myGLCD.setFont(SmallFont);  //en petit
  myGLCD.print(" !\"#$%&'()*+,-./0123456789:;<=>?", CENTER, 120);
  myGLCD.print("@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_", CENTER, 132);
  myGLCD.print("`abcdefghijklmnopqrstuvwxyz{|}~ ", CENTER, 144);

  myGLCD.setColor(0, 0, 255);  //bleu

  myGLCD.setFont(SevenSegNumFont);        //chiffres seulement en 7-segments
  myGLCD.print("0123456789", CENTER, 190);  //0-9 sans point decimal
  while(1) {}; // rien
}
```

Le texte se positionne avec

CODE:

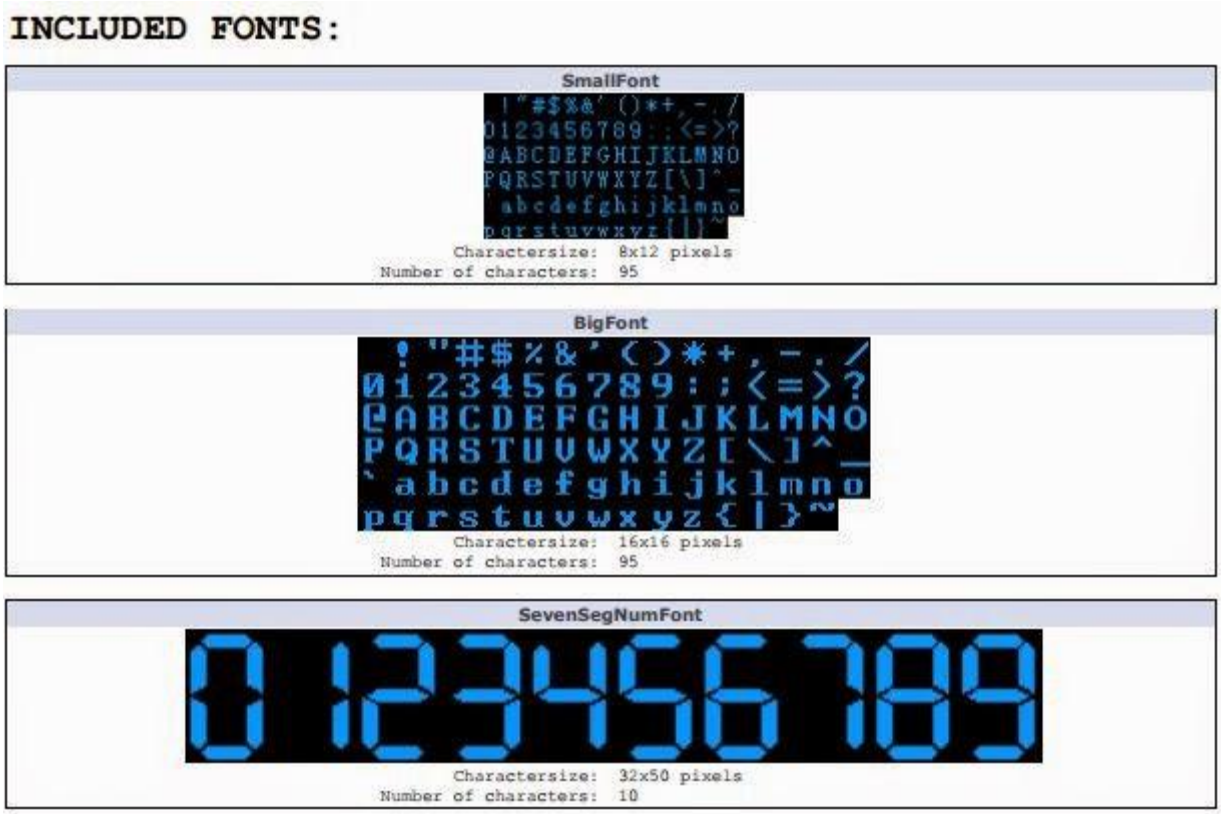
```
myGLCD.print("mon texte ",x, y);
```

On peut spécifier un angle d'inclinaison de texte

CODE:

```
myGLCD.print("mon texte penche", 20, 20, angle);
```

Afficher des chiffres
On peut afficher des chiffres comme du texte ordinaire, ou sous forme de 7 segments, ou avec 2 fonctions dédiées pour ça.



CODE:

```
a=random(32000);
myGLCD.printNumI(a,LEFT, 16);
c= 3.141592654;
myGLCD.printNumF(c,5,LEFT, 48);
```

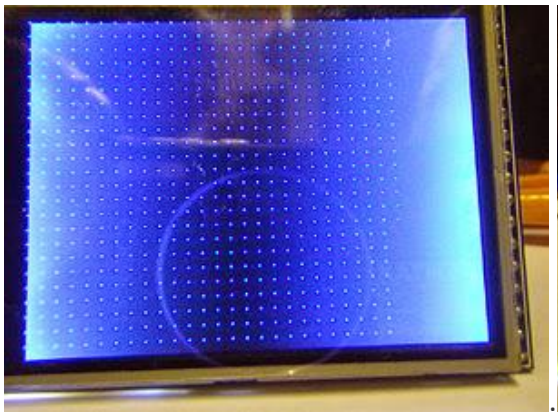
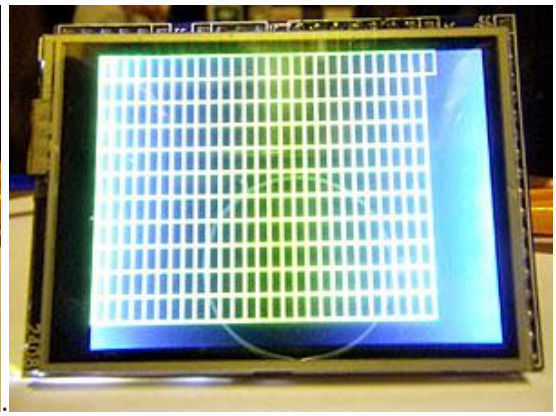
Affichage graphique géométrique

Remplir l'écran entier d'une couleur , afficher un seul pixel, une ligne, un rectangle, un rectangle arrondi aux coins

CODE:

```
myGLCD.fillScr(red, green, blue);
myGLCD.drawPixel(x,y);   //En haut à gauche = position 0,0
myGLCD.drawLine(x1,0,x2,239);
myGLCD.drawRect(x1,y2,x2,y2); // rectangle ouvert  x1, y1 en haut à gauche, x2, y2 en bas à droite
```

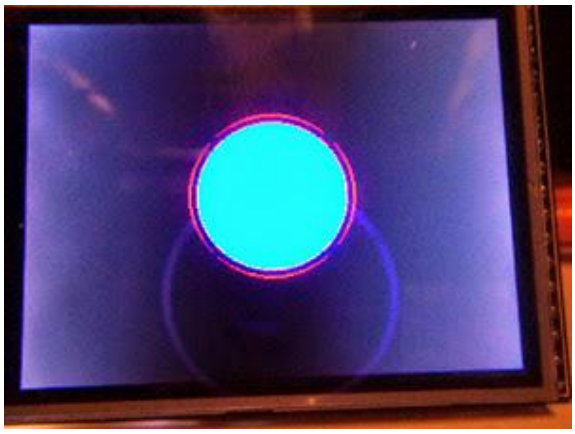
```
myGLCD.fillRect(x1,y2,x2,y2); //  rectangle rempli
myGLCD.drawRoundRect(x1,y2,x2,y2); // rectangle arrondi
myGLCD.fillRoundRect(x1,y2,x2,y2); // rectangle rempli et arrondi
```

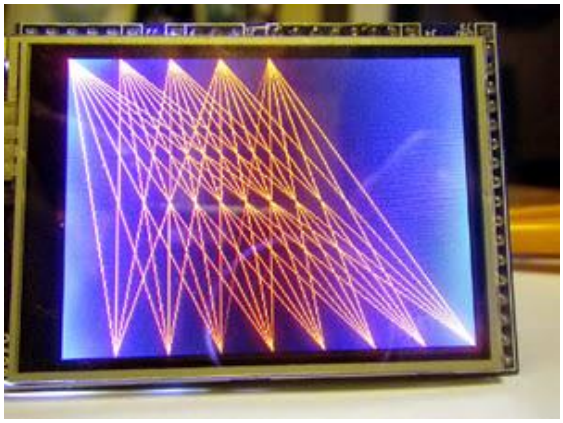
Dessiner des cercles et des disques

CODE: TOUT SÉLECTIONNER

```
myGLCD.drawCircle(x,y,r); // cercle ouvert
myGLCD.fillCircle(x,y,r); // Dessine un disque rempli
```



Exemple (2) Affichage de points, lignes, rectangles, cercles, fonds.



CODE:

```
//*****************************************************
// Ecran graphique 320x240 pixels couleur
// affichages géométriques
// (fonds, points, lignes, cercles, rectangles)
// modifié tiptopboards.com  03 11 2014
// demo_utft_2
// selon  http ://tronixstuff.com/
//
//*****************************************************
#include <memorysaver.h>  //Ajouté pour optimiser la mémoire Uno
#include "UTFT.h"

UTFT myGLCD(ILI9325C,19,18,17,16);    //uno

void setup()
{
  myGLCD.InitLCD(LANDSCAPE);
  myGLCD.clrScr();
  randomSeed(analogRead(0));
}

void loop()
{
  // demonstrate myGLCD.fillScr()  rouge, puis vert, puisbleu
  myGLCD.fillScr(255,0,0);
  myGLCD.fillScr(0,255,0);
  myGLCD.fillScr(0,255,255);

  // demonstrate myGLCD.drawPixel()  damier de points
  myGLCD.clrScr();
  myGLCD.setColor(0, 0, 255);
  for (int x=0; x<320; x+=10)
  {
    for (int y=0; y<240; y+=10)
    {
      myGLCD.drawPixel(x,y);
      delay(10);
    }
  }

  // demonstrates myGLCD.drawLine()  triangles rouges
```

```
myGLCD.clrScr();
myGLCD.setColor(255, 0, 0);  rouge
for (int x1=0; x1<320; x1+=40)
{
  for (int x2=319; x2>=0; x2-=40)
  {
    myGLCD.drawLine(x1,0,x2,239);
  }
}

// demonstrates myGLCD.drawRect()  grillage
myGLCD.clrScr();
myGLCD.setColor(255, 255, 0);
for (int x1=0; x1<320; x1+=10)
{
  for (int y2=0; y2<240; y2+=20)
  {
    myGLCD.drawRect(x1,0,0,y2);
    delay(20);
  }
}

// demonstrates myGLCD.fillRect()  remplissage
myGLCD.clrScr();
myGLCD.setColor(255, 255, 255);
for (int x1=0; x1<320; x1+=10)
{
  for (int y2=0; y2<240; y2+=20)
  {
    myGLCD.fillRect(x1,0,0,y2);
  }
}

// demonstrates myGLCD.drawCircle() and .fillCircle
myGLCD.clrScr();
for (int r=5; r<100; r+=5)
{
  myGLCD.setColor(255, 0, 0);  //cercles bleus et rouges
  myGLCD.drawCircle(160,120,r);
  delay(100);
  myGLCD.setColor(0, 0, 255);
  myGLCD.fillCircle(160,120,r);
```

```
    delay(50);
  }

}
```

Dessins en bitmap

Les images en .gif, .jpg, .png de moins de 300k peuvent s'afficher sur l'écran, en les convertissant en tableau avec un imageconverteur comme celui ci
http://www.henningkarlsen.com/electroni ... ter565.php
exporter en format .c pour arduino pour le copier /coller dans son code

On appellera le bitmap à la position x,y voulue avec
CODE:

```
myGLCD.drawBitmap(x,y,width,height, name, scale);
```

scale permet de zoomer l'image par interpolation.
on peut pivoter l'image avec
CODE:

```
myGLCD.drawBitmap(x,y,width,height, name, scale);
```

Exemple (3) Affichage d'un bitmap avec zoom et rotation

CODE:

```
//*********************************************************
// Ecran graphique en mode bitmap
// demo_utft_3
//
// modifié tiptopboards 03 11 2014
// affiche une image de chien avec zoom et rotation
// selon http: //tronixstuff.com/2013/04/26/tutorial-arduino-and-ili9325-colour-tft-
```

lcd-modules/

```
//************************************************************
#include "UTFT.h"

UTFT myGLCD(ILI9325C,19,18,17,16);

void setup()
{
  myGLCD.InitLCD(LANDSCAPE);
  myGLCD.clrScr();
}


#include <avr/pgmspace.h>

prog_uint16_t dog[1110] PROGMEM={
0xFFFF, 0xFFFF, 0xFFFE, 0xFFFF, 0xFFFF, 0xF7FF, 0xFFFF, 0xFFFD, 0xFFDF, 0xFFFF,
0xFFFE, 0xFFFF, 0xFFDF, 0xFFDF, 0xFFFF, 0xFFFF,   // 0x0010 (16) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0020 (32) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFE, 0xFFFF, 0xF7FF, 0xF7FF,   // 0x0030 (48) pixels
0xF7FF, 0xFFFF, 0xFFFF, 0xF7FF, 0xF7FF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xF7DF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7FF,   // 0x0040 (64) pixels
0xF7FF, 0xF7DF, 0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFDF, 0xFFDF, 0xFFFF, 0xFFFF,   // 0x0050 (80) pixels
0xF7FF, 0xF7FF, 0xF7FF, 0xFFFF, 0xFFFF, 0xEF7D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xF7BE, 0xFFDE, 0xFFBE, 0xFF9E, 0xFFFF,   // 0x0060 (96) pixels
0xFFFF, 0xFFFF, 0xFFBF, 0xFFFF, 0xF7DF, 0xFFFF, 0xFFFF, 0xF7BF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF,   // 0x0070 (112) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFF5A, 0xC570,
0xAC8E, 0x9C0E, 0xF6DA, 0xDE15, 0xBCCE, 0xC572,   // 0x0080 (128) pixels
0xBD10, 0xB46E, 0xAC4D, 0xCD10, 0xBCCF, 0xC510, 0xE655, 0xCD32, 0xC532,
0xEEB8, 0xFF9C, 0xFFFE, 0xFFFF, 0xF7DE, 0xF7FF, 0xFFFF,   // 0x0090 (144) pixels
0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFDF, 0xFFFF, 0xF7FF, 0xFFFF, 0xFFBF,
0xFFBE, 0xF75C, 0xBCF1, 0x8B28, 0x8AC4, 0x8284,   // 0x00A0 (160) pixels
0x5982, 0x934A, 0xB40B, 0xBBE8, 0xC4AD, 0xBC6C, 0xC46C, 0xD4EE, 0xED90,
0xCC6B, 0xD48B, 0xCC6A, 0xABE6, 0x9305, 0x9B68, 0xCCF0,   // 0x00B0 (176) pixels
0xE615, 0xF6FA, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFDE,
0xFFDE, 0xF7FF, 0xF7FF, 0xFFFF, 0xFF5C, 0xD5B3,   // 0x00C0 (192) pixels
0x8B68, 0x8328, 0x7A85, 0x82A6, 0x7A64, 0x7A64, 0xB42A, 0xBC0A, 0xBC2A,
0xE655, 0xF6D8, 0xDE36, 0xD5F5, 0xFEF8, 0xD551, 0xCCCD,   // 0x00D0 (208) pixels
```

0xCC6C, 0xABE7, 0x8262, 0x8AA5, 0xB3CA, 0xBC0A, 0xBC4C, 0xB48D, 0xCDD3,
0xFF7B, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDE, 0xFFFF, 0xFFFF, // 0x00E0 (224) pixels
0xFFFF, 0xFFFF, 0xCD94, 0x9BAB, 0x9327, 0xA387, 0x7285, 0x6A24, 0x7AC7, 0x5961,
0x9B67, 0xB3E9, 0xB40A, 0xC46C, 0xE6B9, 0xFFBE, // 0x00F0 (240) pixels
0xF7BE, 0xEF9E, 0xF79D, 0xD5F5, 0xCD0F, 0xCC8C, 0xCCAD, 0x9B28, 0x6161, 0x9B07,
0xBC09, 0xB3E7, 0xB3E8, 0xC46B, 0xD5D4, 0xFFFE, // 0x0100 (256) pixels
0xFFFF, 0xFFFF, 0xFFDF, 0xFFDF, 0xFFFF, 0xFFDF, 0xDE77, 0x9BA9, 0xA387, 0xA367,
0x9B08, 0x7224, 0x6A44, 0x9388, 0x61E2, 0x8AC5, // 0x0110 (272) pixels
0xAB88, 0xBC2A, 0xBC6B, 0xC532, 0xEED9, 0xF77C, 0xF79D, 0xF75B, 0xD592,
0xCC2A, 0xEC8A, 0xE4ED, 0xD4AD, 0x7A23, 0x8AE6, 0xABC8, // 0x0120 (288) pixels
0xB3E9, 0xB3A9, 0xC3CB, 0xAC4D, 0xFFFD, 0xFFFF, 0xF7FF, 0xFFFF, 0xFFFF, 0xFFDF,
0xFFFF, 0xD5F5, 0x9327, 0xB3A8, 0x9327, 0x72C7, // 0x0130 (304) pixels
0x6265, 0x7AA5, 0x9347, 0x7285, 0x8AE7, 0xAB67, 0xCC29, 0xB3C9, 0xC3EA,
0xBC4D, 0xEE55, 0xE79E, 0xF6D7, 0xCC4A, 0xD429, 0xDC4B, // 0x0140 (320) pixels
0xD4AC, 0xCC8C, 0xABA8, 0x8243, 0xB3C9, 0xB40A, 0xAC0A, 0xB42B, 0xAC0C,
0xFFFF, 0xF7FF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // 0x0150 (336) pixels
0xFF5B, 0x9349, 0x92E5, 0x8B07, 0x6A66, 0x6224, 0x7224, 0x9327, 0x6A44, 0x8B07,
0xA388, 0xB3E9, 0xBC4B, 0xBC07, 0xB3A7, 0xC409, // 0x0160 (352) pixels
0xE6DA, 0xFE55, 0xC46B, 0xCC6A, 0xD4CB, 0xD48B, 0xCC8B, 0xBC0A, 0x8AC5,
0xB3E9, 0xB3E9, 0xBC2A, 0xBC2A, 0xAC2C, 0xFFFF, 0xF7FF, // 0x0170 (368) pixels
0xFFFF, 0xFFFF, 0xFFFE, 0xFFFF, 0xF7FF, 0xFFFE, 0xB4D0, 0x9328, 0x82A6, 0x7287,
0x7266, 0x8285, 0x7A64, 0x61C2, 0x7A85, 0x92E5, // 0x0180 (384) pixels
0xB3E8, 0xBC6A, 0xD4CA, 0xC428, 0xBB24, 0xDE17, 0xED51, 0xCCCD, 0xBC08,
0xB407, 0xE4ED, 0xDCCC, 0xCC4A, 0xAB67, 0xA368, 0xB3C9, // 0x0190 (400) pixels
0xB3E9, 0xAB88, 0xB46E, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xF7BE, 0xE677, 0x936A, 0x6A04, 0x7A86, 0x59E3, // 0x01A0 (416) pixels
0x9348, 0x7202, 0x69E2, 0xB40B, 0xBC4B, 0x9AE5, 0x92C4, 0xC46B, 0xBBE9, 0xCBC6,
0xDE97, 0xCC6B, 0xC42A, 0xB386, 0xCC4A, 0xC46C, // 0x01B0 (432) pixels
0xD4EE, 0xE4EC, 0xC408, 0x8AE5, 0xB42B, 0xBC4B, 0x9369, 0xBD10, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xF7BF, 0xFFFF, 0xFFFF, // 0x01C0 (448) pixels
0xFFBD, 0x9BEC, 0x7AC6, 0x7A85, 0x6A44, 0x8B27, 0x5981, 0x8AE6, 0x7AE7, 0x72A8,
0x8329, 0x7245, 0xAB68, 0xABA9, 0xCC6A, 0xE6D8, // 0x01D0 (464) pixels
0xC44A, 0xBC09, 0xAB68, 0xA3AB, 0x28E2, 0x3922, 0xABA8, 0xCC09, 0xB40B,
0xA3EA, 0xB40B, 0x9B8A, 0xCDB3, 0xFFFF, 0xFFFF, 0xFFFF, // 0x01E0 (480) pixels
0xFFFF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFE, 0xFF7B, 0x7AE8, 0x7285, 0x6A23,
0x8327, 0x59C2, 0x4982, 0x20A1, 0x0862, 0xC619, // 0x01F0 (496) pixels
0x3965, 0x9B06, 0xB3EA, 0xC48C, 0xEED9, 0xD4EE, 0xBC0A, 0x7AE9, 0x3124,
0x52ED, 0x2104, 0x9B69, 0xB3A7, 0xE613, 0xB46D, 0xC42A, // 0x0200 (512) pixels
0xBC0B, 0xDE57, 0xFFFF, 0xFFFF, 0xFFFE, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDE, 0xFFFF,
0xFFFF, 0xF7BD, 0xE698, 0x6A87, 0x72C7, 0x8328, // 0x0210 (528) pixels
0x6A45, 0x7AC7, 0x28E2, 0x0001, 0x10C3, 0x49E7, 0x9326, 0xBBE9, 0xC48D, 0xFF3B,
0xDE37, 0xBC4C, 0x6AC9, 0x41A4, 0x1861, 0x3144, // 0x0220 (544) pixels

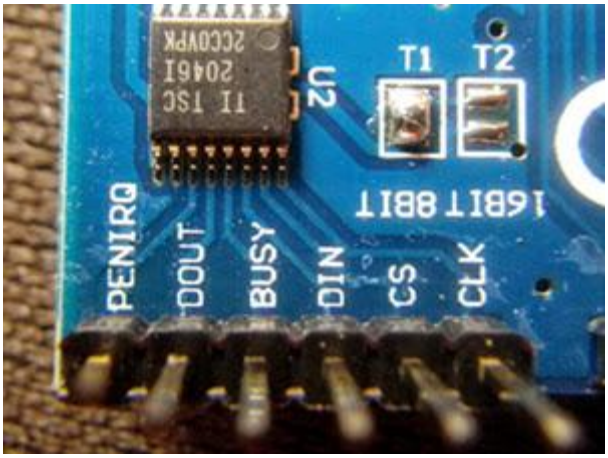
0xBC8D, 0xBC2A, 0xF719, 0xD5D4, 0xBC0A, 0xBBEA, 0xEEF9, 0xFFFF, 0xFFFF, 0xFFFE,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7FF, 0xFFFF, // 0x0230 (560) pixels
0xFFFF, 0xFFFF, 0xD617, 0x832A, 0xFF19, 0x7AA7, 0x7A85, 0x6225, 0x2923, 0x3101,
0x7245, 0xA3CA, 0xC46B, 0xCD31, 0xF75B, 0xE73D, // 0x0240 (576) pixels
0xF613, 0xAC6F, 0xABCA, 0x92E7, 0x9BAB, 0xCC6B, 0xBC4B, 0xF7BD, 0xFFFE, 0xB44C,
0xBC4C, 0xFF5B, 0xFFFF, 0xFFFF, 0xFFFE, 0xFFFF, // 0x0250 (592) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF77C,
0x7B09, 0x7AC6, 0x82C5, 0x9327, 0xA3A8, 0x92E6, // 0x0260 (608) pixels
0x9328, 0xBCAF, 0xCDD4, 0xEF7C, 0xEFBE, 0xF77D, 0xEE97, 0xCD10, 0xC46C, 0xC44A,
0xCC6B, 0xB44C, 0xFFDD, 0xFFFF, 0xBD31, 0xBC8D, // 0x0270 (624) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x942E, // 0x0280 (640) pixels
0x8B28, 0x82A5, 0x9B47, 0x8B06, 0x9328, 0xC553, 0xEED9, 0xF71A, 0xF77C, 0xF7BE,
0xF7BE, 0xFF9C, 0xEE97, 0xCCCD, 0xDCCB, 0xDCCC, // 0x0290 (656) pixels
0xBCF0, 0xFFBE, 0xFFFF, 0xFFFF, 0xFFBD, 0xFFFF, 0xFFDF, 0xF7DF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // 0x02A0 (672) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xF79D, 0xDE98, 0x938A, 0x8AE6, 0xA368, 0x8B48, 0xA46E,
0xDF1B, 0xE6FB, 0xF73C, 0xEF1B, 0xEF1B, 0xF7BE, // 0x02B0 (688) pixels
0xEF9D, 0xF7BE, 0xEE96, 0xD4CD, 0xCCAD, 0xEED9, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7FF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // 0x02C0 (704) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDE,
0xACF1, 0x8B28, 0x9348, 0x7AE8, 0xB594, 0xCEDA, // 0x02D0 (720) pixels
0xE73C, 0xEF3C, 0xF75C, 0xE6DB, 0xEF1C, 0xEF7D, 0xEF7D, 0xF79D, 0xE656, 0xDE15,
0xFFBD, 0xF7DF, 0xFFFF, 0xFFDE, 0xF7DF, 0xFFDF, // 0x02E0 (736) pixels
0xFFFF, 0xFFFF, 0xFFDF, 0xFFDF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xF7FF, 0xE6DA, // 0x02F0 (752) pixels
0xA42D, 0x8329, 0xA4B0, 0xBE18, 0xDE99, 0xC5F6, 0xB554, 0x7B8D, 0x83CF, 0xCE17,
0xEF3B, 0xF79C, 0xE75D, 0xDF1B, 0xD678, 0xE71B, // 0x0300 (768) pixels
0xEF7C, 0xE73C, 0xEF3B, 0xD699, 0xE6FB, 0xFFFF, 0xFFFF, 0xFFFF, 0xF79D, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, // 0x0310 (784) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xF7FF, 0xE73C, 0xBD74, 0x838C, 0x9CB1, 0xAD96, 0xD615,
0x4A06, 0x20E3, 0x39A6, 0x4A28, 0x5A68, 0xD657, // 0x0320 (800) pixels
0xFF7C, 0xEF7C, 0xE73B, 0xDEFA, 0xE73B, 0xE71B, 0xD6BA, 0xD6DA, 0xEF9D,
0xE71B, 0xB554, 0x8C0F, 0x9CB1, 0xB595, 0xFFFF, 0xFFFF, // 0x0330 (816) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDF, 0xFFFF, 0xF79E,
0xB575, 0xB574, 0xAD13, 0xBD95, 0xBD73, 0x3964, // 0x0340 (832) pixels
0x20E3, 0x3165, 0x2923, 0x41C6, 0xD637, 0xEF5C, 0xF75B, 0xE6D9, 0xE6FA, 0xDF1B,
0xDF1B, 0xEF3B, 0xC617, 0xA575, 0xE71B, 0xFFBD, // 0x0350 (848) pixels
0xEF3B, 0xCE37, 0xE6FA, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xDEFB, 0xAD75, // 0x0360 (864) pixels
0xB575, 0xA4B1, 0xACD1, 0xBD75, 0x6B2C, 0x18A2, 0x20E2, 0x3164, 0x62A9,
0xD679, 0xDF1C, 0xEF7D, 0xE6FA, 0xE6FB, 0xD71D, 0xEF7D, // 0x0370 (880) pixels

```
0xFF1A, 0xFF3A, 0xA514, 0x8C30, 0xEF3C, 0xF79D, 0xEF5C, 0xEF5C, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0380 (896) pixels
0xFFDF, 0xFFFF, 0xFFDE, 0xBDF7, 0xB595, 0xB575, 0xB575, 0x9C91, 0xA4F3, 0x8BEF,
0x41C6, 0x41E7, 0x49E7, 0x8C30, 0xD679, 0xE6FB,   // 0x0390 (912) pixels
0xEF1B, 0xDEDA, 0xDEDA, 0xE73B, 0xEF7C, 0xF79D, 0xEF5C, 0xE71B, 0x6AA9,
0xACF2, 0xEF5B, 0xEF5C, 0xF79E, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x03A0 (928) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFDE, 0xB595, 0xAD54,
0xB595, 0xBDB6, 0xB554, 0xA4B2, 0x83EF, 0x732C,   // 0x03B0 (944) pixels
0x5228, 0x6AEB, 0x7B8E, 0xCE38, 0xDE99, 0xE6FB, 0xE6FB, 0xE71B, 0xEF3C, 0xEF5C,
0xEF7C, 0xEF7C, 0xEF5C, 0xC574, 0x5226, 0xCE16,   // 0x03C0 (960) pixels
0xEF5C, 0xF77D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFDF, 0xFFDF, 0xB595, 0xAD54, 0xB595,   // 0x03D0 (976) pixels
0xBD75, 0xBD95, 0xC5D7, 0xB555, 0x83AE, 0x8C10, 0x8BEF, 0xC5B6, 0xD658,
0xDEDA, 0xE6FB, 0xE71B, 0xEF3C, 0xEF5C, 0xEF5C, 0xEF7C,   // 0x03E0 (992) pixels
0xF77D, 0xEF7D, 0xF73B, 0x6B0A, 0x9C90, 0xEF5C, 0xEF3C, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x03F0 (1008) pixels
0xFFDF, 0xEF5D, 0xB596, 0xAD55, 0xB595, 0xB554, 0xB555, 0xB534, 0xBDB6,
0xCDF7, 0xC5D7, 0xCE17, 0xD638, 0xD679, 0xDEDA, 0xE71B,   // 0x0400 (1024) pixels
0xE71B, 0xEF5C, 0xEF7C, 0xEF7D, 0xF79D, 0xF79D, 0xEF7D, 0xEF1A, 0xBDB5, 0x62A9,
0xD678, 0xEF7C, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0410 (1040) pixels
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xCE39, 0xAD55, 0xAD34, 0xB595,
0xB575, 0xBD96, 0xBDB6, 0xC5B6, 0xBDB6, 0xBDB6,   // 0x0420 (1056) pixels
0xCE17, 0xDE99, 0xD699, 0xDEDA, 0xE71B, 0xE71B, 0xEF5C, 0xEF7C, 0xF77D, 0xF7BD,
0xF7BD, 0xEF7D, 0xF75B, 0xDE99, 0x6B0A, 0xC5F6,   // 0x0430 (1072) pixels
0xF79D, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
0xFFFF, 0xAD54, 0xA4F3, 0xA513, 0xB595, 0xBDB6,   // 0x0440 (1088) pixels
0xBDB6, 0xBDB6, 0xC5F7, 0xC5D7, 0xC5D7, 0xCE17, 0xD679, 0xD679, 0xE6FB,
0xDEFA, 0xE71B, 0xEF5C, 0xEF7C, 0xEF7C, 0xF79D, 0xF7BD,   // 0x0450 (1104) pixels
};

void loop()
{
  myGLCD.clrScr();
  // Affiche un bitmap à plusieurs tailles
  myGLCD.drawBitmap(20,20,37,30, dog);  //Image du chien
  delay(2000);
  myGLCD.clrScr();
  myGLCD.drawBitmap(20,20,37,30, dog,2);//Zoom x2
  delay(2000);
  myGLCD.clrScr();
  myGLCD.drawBitmap(20,20,37,30, dog,3);  //Zoom x3
  delay(2000);
```

```
  myGLCD.clrScr();
  myGLCD.drawBitmap(20,20,37,30, dog,4);//Zoom x4
  delay(2000);
  myGLCD.clrScr();
  myGLCD.drawBitmap(20,20,37,30, dog,5);//Zoom x5
  delay(2000);
  myGLCD.clrScr();

  // Rotation du bitmap
  myGLCD.drawBitmap(20,20,37,30, dog, 45, 19,15);
  delay(5000);
 }
```

Autres fonctions

Ecran tactile



Lecteur SD