

## Matériel utilisé :

KIT Explorer 16 avec PIC24FJ128GA010 et Graphics PICtail Plus Board V2

Les spécifications de l'afficheur se trouvent ici :

<http://www.trulydisplays.com/tft/specs/2.8in%2040x320%20TFT-G240320UTSW-92W-TP-E%20Spec.pdf>



MPLAB 8.20 avec C30 (version étudiant) et ICD2 ou ICD3

Télécharger et installer la bibliothèque. Connecter les cartes et l'ICD.

**IMPORTANT** : le fichier `C:\Microchip Solutions\Microchip\Help\ Graphics Library Help` est la seule mais complète documentation fournie par Microchip, c'est une base documentaire qu'il est indispensable de consulter.

La progression s'effectue à partir de : `C:\Microchip Solutions\Graphics Primitive Layer Demo`

Ouvrir le projet «Primitive Layer Demo PIC24 », retirer du projet le fichier `maindemo.c`

Créer un fichier `demoperso.c` et l'inclure dans le projet

## demoperso.c, afficher un point

```
#include "MainDemo.h"
//config pour PIC24
_CONFIG2(FNOSC_PRIPLL & POSCMOD_XT) // Primary XT OSC with PLL
_CONFIG1(JTAGEN_OFF & FWDTEN_OFF) // JTAG off, watchdog timer off
int main(void){
    InitGraph();
    while(1){
        SetColor(WHITE);
        PutPixel(20,20);
        DelayMs(500);
        SetColor(BLACK);
        PutPixel(20,20);
        DelayMs(500);
    }
}
```

La fonction **InitGraph()** permet l'initialisation des périphériques du microcontrôleur .

Le matériel est défini dans le fichier **hardwareprofile.h** (type de processeur, connexions entre l'afficheur et le microcontrôleur etc ...)

Le fichier **GrafigConfig.h** définit le type d'afficheur utilisé et les paramètres qui lui sont propres (par exemple sa taille en pixels)

La fonction **SetColor** permet de choisir la couleur utilisée, pour connaître les couleurs possibles, il suffit de rechercher « color » dans le fichier d'aide (ci-dessus)

**PutPixel** active un pixel sur les coordonnées x,y fournies avec la couleur active.

Après compilation, téléchargement et lancement, un pixel devrait clignoter dans la quart haut-gauche de l'afficheur.

Pour faire clignoter ce pixel au milieu de l'afficheur il faut connaître les dimensions en pixels de l'afficheur du KIT (en recherchant dans le data sheet du G2400320UTSW qui équipe le KIT on trouve 320x240 pixels), cette information est définie dans **GrafigConfig.h** et est accessible par deux macros, GetMaxX et GetMaxY.

Le clignotement au milieu s'obtient comme ceci :

```
int main(void){
SHORT x,y;
x=GetMaxX()/2;
y=GetMaxY()/2;
    InitGraph();
    while(1){
        SetColor(WHITE);
        PutPixel(x,y);
        DelayMs(500);
        SetColor(BLACK);
        PutPixel(x,y);
        DelayMs(500);
    }
}
```

Etant capable d'allumer un pixel et de choisir sa couleur, on peut tracer des droites, des cercles, des rectangles (pleins ou vides) etc... La bibliothèque propose de nombreuses fonctions graphiques permettant de tracer des formes géométriques.

### Formes géométriques :

```
int main(void){
InitGraph();
SetColor(WHITE);
Line(20,20,100,100); // x haut, y haut, x bas, y bas
SetColor(BRIGHTBLUE);
Circle(250,80,40); // centre x,y rayon
SetColor(YELLOW);
Rectangle(150,150,200,200); // x haut, y haut, x bas, y bas
FillCircle(270,100,20); // centre x,y rayon
SetColor(BRIGHTRED);
Bar(250,150,280,170); // x haut, y haut, x bas, y bas
while(1);
}
```

**Line** trace une ligne du point en haut à gauche x,y au point en bas à droite x,y

**Circle** trace un cercle de centre x,y, et de rayon r

**Rectangle** trace un rectangle du point en haut à gauche x,y au point en bas à droite x,y

**FillCircle** , comme **Circle** mais plein

**Bar**, comme **Rectangle** mais plein.



La fonction **ClearDevice()**; efface l'afficheur avec la couleur courante (permet de mettre une couleur de fond) .

## Effacer tout :

```
int main(void){
  InitGraph();
  while(1) {
    SetColor(BRIGHTMAGENTA);
    ClearDevice();
    DelayMs(1000);
    SetColor(GRAY4);
    ClearDevice();
    DelayMs(1000);
    SetColor(BROWN);
    ClearDevice();
    DelayMs(1000);
    SetColor(LIGHTBLUE);
    ClearDevice();
    DelayMs(1000);
  }
}
```

## Le texte

```
#include "MainDemo.h"
//config pour PIC24
_CONFIG2(FNOSC_PRIPLL & POSCMOD_XT) // Primary XT OSC with PLL
_CONFIG1(JTAGEN_OFF & FWDTEN_OFF) // JTAG off, watchdog timer off
extern const FONT_FLASH Font35;
XCHAR * mon_texte="ici dans une chaine";
int main(void){
  InitGraph();
  SetFont((void*)&Font35);
  SetColor(WHITE);
  OutTextXY( 80,50,"Un bien beau texte");
  OutTextXY( 40,100,mon_texte);
  while(1) ;
}
```

**extern const FONT\_FLASH Font35;** déclare le fichier de la police de caractères (deux sont proposées dans la démo)

**SetFont((void\*)&Font35);** selectionne la police

**OutTextXY( 80,50,"Un bien beau texte");** écrit le texte en x,y

Rq : une chaîne de caractères peut être déclarée comme variable mais avec le type prédéfini XCHAR



## Afficher une image (notre ami Gaston Lagaffe):

```
#include "MainDemo.h"
//config pour PIC24
_CONFIG2(FNOSC_PRIPLL & POSCMOD_XT) // Primary XT OSC with PLL
_CONFIG1(JTAGEN_OFF & FWDTEN_OFF) // JTAG off, watchdog timer off
extern const BITMAP_FLASH gaston;
int main(void){
  InitGraph();
  PutImage(10,10,(void*)&gaston,2);
  while(1) ;
}
```

extern const BITMAP\_FLASH gaston; déclare gaston comme un pointeur sur une constante dans un fichier externe.

PutImage(10,10,(void\*)&gaston,2); Affiche l'image gaston en x,y



Le projet doit contenir le fichier image (ici gaston.c) généré par l'utilitaire :

C:\Microchip Solutions\Microchip\Graphics\Utilities\Font and Bitmap Converter\Bitmap and Font converter.exe

L'image de départ doit être au format bitmap Windows (bmp). Elle est convertie en un fichier C contenant les valeurs interprétées par la librairie. Le fichier obtenu est inclus dans le code et donc dans la mémoire flash du PIC, l'image ne doit pas avoir une taille supérieure à celle de l'afficheur. Il est préférable de supprimer le fichier « Pictures PIC24.C » afin de gagner de la place.

