

CHAPITRE 3

LA BIBLIOTHEQUE GRAPHIQUE DE MICROCHIP

3.1 INTRODUCTION

La prolifération des **interfaces graphiques** dans des dispositifs ordinaires devient apparente.

De plus en plus nous allons le long de nos activités quotidiennes, rencontrer des produits pour avoir une certaine forme d'interface graphique.

Pendant que ce dispositif devient une norme de fait, la nécessité de fabriquer ces dispositifs **peu coûteux devient évidente**.

Les microcontrôleurs de PIC®, avec leur réputation pour le développement de produit à **faible risque, coût de système ,et délai d'arrivée au marché** plus rapide, rend ceci réalisable.

La bibliothèque graphiques de MICROCHIP le rend très facile d'intégrer les dispositifs graphiques dans une application.

3.2 VUE D'ENSEMBLE DE LA BIBLIOTHÈQUE GRAPHIQUES

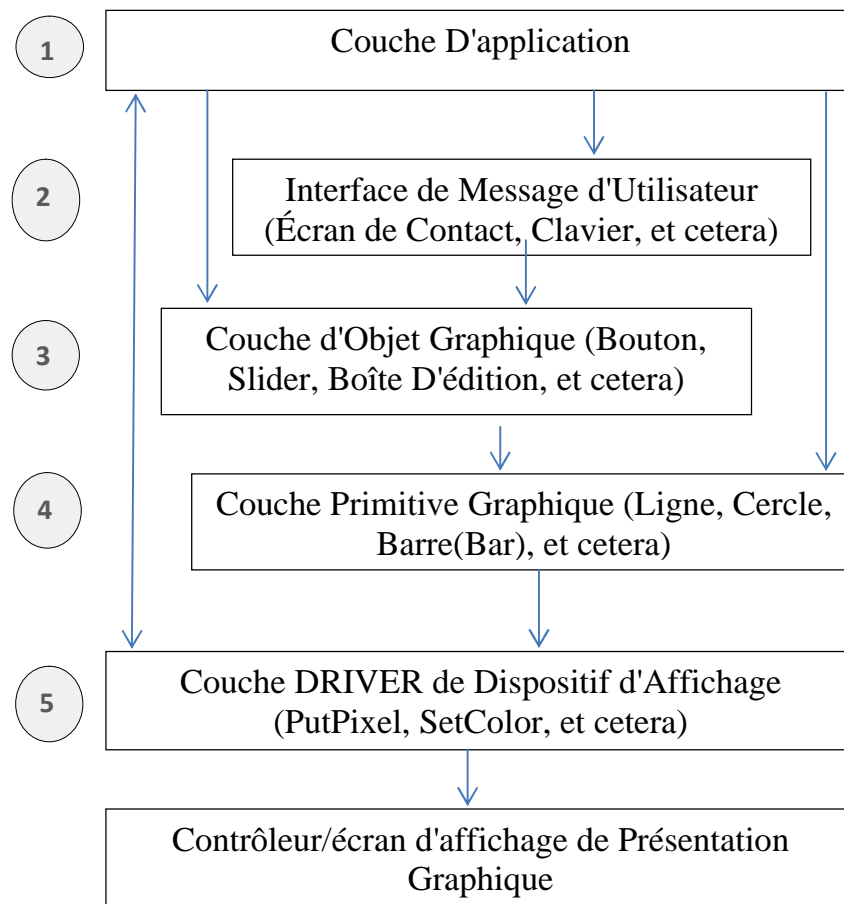
La bibliothèque graphique de Microchip a été créée pour couvrir une large **gamme des contrôleurs de dispositif d'affichage**. Visé pour l'usage avec les microcontrôleurs de PIC, elle offre une interface de programmation API pour rendre exécutable des objets primitifs aussi bien que des objets semblables au gadget avancés.

La bibliothèque facilite également l'intégration des dispositifs d'entrée par une interface de transmission de messages.

Les applications créées utilisant la bibliothèque trouveront également un processus simple et direct pour changer des dispositifs d'affichage si le besoin se fait sentir.

La structure de bibliothèque graphiques de MICROCHIP est montrée dans La figure 3.1 ci-dessous .

Figure 3.1 :SYSTÈME TYPE AVEC LA BIBLIOTHÈQUE GRAPHIQUES DE MICROCHIP



1 couche application :

La couche application (API) est un programme qui utilise la Bibliothèque graphique.

La couche d'objet de graphiques (GOL) :

2+3

couche d'objets graphiques (GOL) rend les gadgets, tels que le bouton, glisseur, fenêtre, etc. Dans tout ce document, des gadgets seront référés comme objets de GOL ou objets. Pour commander ces objets, la couche de GOL a une interface de message qui accepte messages de la couche application.

Cette interface soutient une série de dispositifs d'entrée, tels que des claviers, boutons, écrans tactiles, souris, etc.

4 couche graphique primitive :

La couche graphique primitive met en œuvre les fonctions de dessin primitives. Ces fonctions exécutent le rendu d'objet graphiques, comme la ligne, la barre, le cercle, etc.

Le Driver de dispositif d'affichage :

5 Driver de dispositif d'affichage est la couche dépendante de l'architecture. Cette couche parle directement au contrôleur de dispositif d'affichage. Pour chaque contrôleur d'affichage, a le Driver séparé devrait être mis en application.

Cette bibliothèque vient avec une liste de driver de contrôleur d'affichage déjà mis en application en tant qu'élément de la couche de Driver de dispositif d'affichage.

Si le contrôleur d'affichage choisi n'est pas dans la liste, la seule modification requise pour employer la bibliothèque sera la création ou modification du Driver de dispositif d'affichage. Cet arrangement permet à la bibliothèque d'être portable entre les afficheurs.

L'API étendue permet l'application à accéder à n'importe quelle couche de la bibliothèque.

le traitement de Schéma et message sont effectués intérieurement et peuvent être gardés transparent à l'application.

La bibliothèque fournit également deux configurations (Blocking et Non-Blocking), qui sont placés à un moment de la compilation :

- Pour la configuration Blocking , les fonctions d'aspiration retardent l'exécution de la programmation jusqu'au schéma soit terminer .

- Pour La configuration Non-Blocking, fonctions d'aspiration n'attendent pas pour l'accomplissement de dessin et donne la main au programme.

Ceci permet l'utilisation efficace du microcontrôleur, puisque le programme peut effectuer **d'autres tâches** à la place d'attendre les tâches de schéma à finir.

La configuration Non-Blocking donne des avantages dans les systèmes avec des accélérateurs graphiques et DMA. du point de vue d'application, bloquant et Non-Blocking du cadre de configuration est transparent.

3.3 États d'objet (GOL) :

Les Objets GOL suivent deux types d'états :

- **les États de Propriété**
- **les États de Schéma**

Les **états de Propriété** définissent l'action et l'apparition d'Objets.

Les états Schéma, d'autre part, indiquent si l'Objet a besoin d'être caché, redessiné partiellement ou redessiné entièrement dans l'affichage.

Quelques États de Propriété communs et Schéma Ont montré des états dans la Tableau 3.1. Chaque Objet a sa propre Propriété unique et États Schéma (voir documentation d'API)

TABLEAU 3.1 : ÉTATS D'OBJET COMMUNS

état	type	description
OBJ_FOCUSED	Schéma	L'objet est dans l'état focalisé. Ceci est habituellement employé pour montrer le choix de l'objet. tous les objets n'ont pas ce dispositif
OBJ_DISABLED	Schéma	L'objet est handicapé et ignorera tous les messages
OBJ_DRAW_FOCUS	Schéma	Le foyer pour l'objet sera redessiné.
OBJ_DRAW	propriété	L'objet sera refait complètement.
OBJ_HIDE	propriété	L'objet sera caché en remplissant secteur occupé par l'objet de couleur commune de fond. Ceci a la priorité la plus élevée au-dessus de tous les états de schéma. Quand un objet est placé pour être caché, tous autres états de schéma sont dépassés.

3.4 Arrangement de modèle

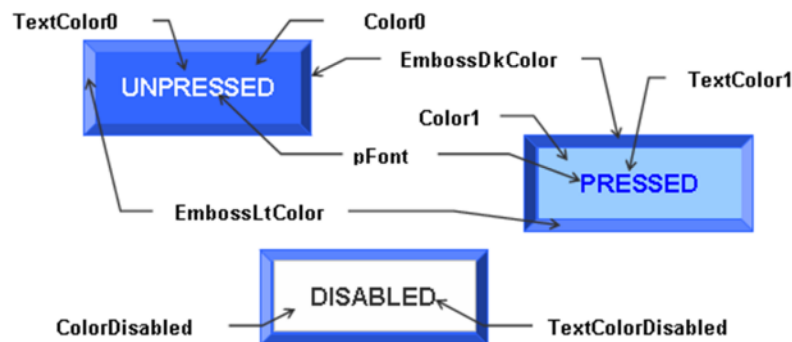
Tous les objets emploient un modèle d'une structure d'arrangement qui définit la police et les couleurs utilisées.

Sur la création de l'objet, un arrangement défini par l'utilisateur de modèle peut être assigné à l'Objet.

En l'absence de l'arrangement défini par l'utilisateur, l'arrangement de défaut est employé. Le tableau 3.2 récapitule les composants d'arrangement de modèle.

TABLEAU 3.2 : COMPOSANTS D'ARRANGEMENT DE MODÈLE

Composant de modèle	Description
EmbossDkColor	L'obscurité gravant la couleur en relief utilisée pour l'effet à 3D des objets.
EmbossLtColor	La lumière gravant la couleur en relief utilisée pour l'effet à 3D des objets.
TextColor0 TextColor1	Couleurs génériques des textes employées par les objets.
TextColorDisabled	L'utilisation peut varier d'un objet à l'autre. La Couleur des textes utilisée pour les objets qui sont désactiver .
Color0 Color1	Couleurs génériques employées pour rendre des objets.
ColorDisabled	L'utilisation peut varier d'un objet à l'autre. Couleur employée pour rendre les objets qui sont désactiver.
CommonBkColor	Une couleur commune de fond des objets.
pFont	Indicateur à la police employée par l'objet. les texte ne sont pas toutes utilisé par les objet.



TextColorDisabled et ColorDisabled sont employés quand l'objet est dans l'état désactiver .

Autrement, TextColor0, TextColor1, Color0 et Color1 sont employés Quand L'état de schéma d'objet est placé à l'état activer, le CommonBkColor est employé pour remplir secteur occupé par l'objet.

Un avantage a dérivé de l'utilisation de l'arrangement de modèle est que chaque objet peut être assigné un modèle unique.

Deux objets ou plus du **même type** peuvent avoir un arrangement unique qui leur sera appliqué.

Ceci donne flexibilité en adaptant le regard et la sensation des objets utilisés d'une application à l'autre.

3.5 Liste d'objet active :

La bibliothèque graphiques groupe les objets qui sont actuellement montré et recevant des messages avec listes lié. À un point quelconque, la transmission de messages de GOL et schéma les fonctions opèrent cette liste. Des objets créés sont automatiquement ajoutés à la liste courante. Seulement un objet de la liste liée peut être en activité à la fois ; c'est possible à maintenir les listes multiples d'objets.

Pour les listes multiples, les applications seront responsables dans la gestion de changement d'une liste à l'autre. Cet arrangement laisse les applications pour traiter chaque liste comme page d'affichage qui résultats dans une gestion facile des pages d'affichage. Seulement la liste active sera montrée dans l'écran.

3.6 Schéma

Pour rendre les objets, l'application devrait appeler un directeur d'aspiration, GOLDraw (). La fonction analyse la liste de lien actif et refait les objets avec les états de schéma réglés. Quand le rendu est accompli, des états de schéma des objets sont dégagés automatiquement. Le premier objet créé sera dessiné d'abord. Après que tous les objets dans la liste courante de lien soient dessinés, le GOLDraw () appelle la fonction de GOLDrawCallback ().

3.7Transmission de messages

La portabilité est un des dispositifs principaux de la bibliothèque. A la variété de dispositifs d'entrée est soutenue.

La bibliothèque fournit une interface pour accepter des messages des dispositifs d'entrée. N'importe quel événement de dispositif d'entrée est envoyé à la bibliothèque suivant la structure de message de GOL. La structure a la définition suivante :

```
typedefstruct {  
  BYTE type;  
  BYTE event;  
  int param1;  
  int param2;  
} GOL_MSG;
```

Le type de champ définit le type ID du dispositif d'entrée. L'événement de champ indique le type d'action. **Les champs, type et événement**, décideront comment **param1** et **param2** seront interprétés.

Pour quelques cas, seulement param1 est employé, alors que dans d'autres, les deux champs de paramètre seront exigés.

Pour illustrer l'utilisation du GOL_MSG, prenons comme exemple le module d'écran tactile. Le GOL_MSG des champs sont définis dans le tableau 3.3.

TABLEAU 3.3 : DEFINITION GOL_MSG DES CHAMP

champ	Description
type	TYPE_TOUCHSCREEN
Événement	Les IDs possibles d'événement sont les suivants : EVENT_INVALID EVENT_MOVE EVENT_PRESS EVENT_RELEASE
param1	X-coordonnez la position du contact
param2	Y-coordonnez la position du contact

Quand l'écran est touché, l'application doit peupler la structure de message et passer à la fonction du gestionnaire de messages de bibliothèque, GOLMsg (pMsg de GOL_MSG*).

L'objet qui inclut la position x, y changera son état basé sur son état actuel et l'événement. Les actions faites sur commande sur les événements de dispositif d'entrée peuvent être faites dans la fonction de GOLMsgCallback (). La fonction s'appelle chaque fois que un message valide pour un certain objet est reçu.

3.8UTILISATION DE BIBLIOTHÈQUE GRAPHIQUE

La bibliothèque est conçue pour permettre l'intégration sans couture d'une interface graphique dans une application. Utilisant les objets déjà définis exige le codage très minimal.

La bibliothèque fournit un API facilement pour créer, contrôler et détruire les objets.

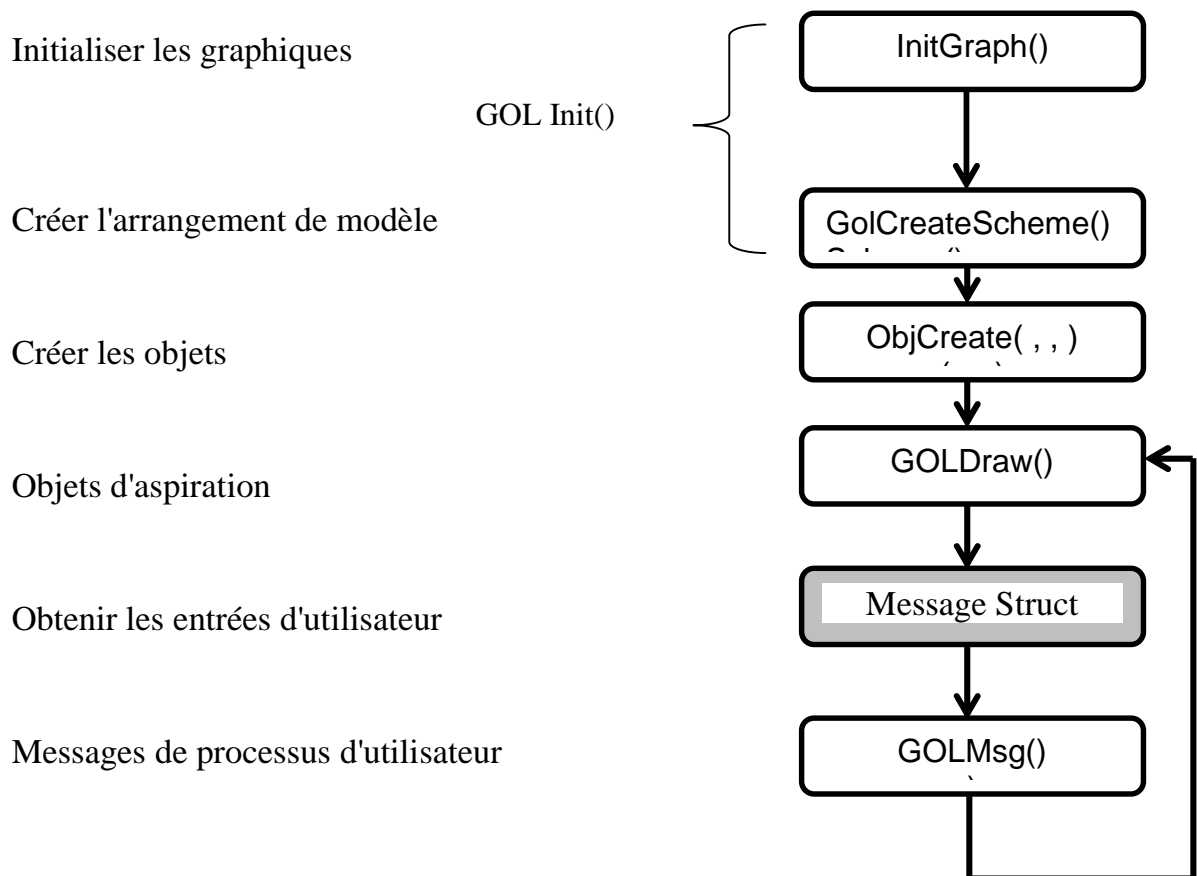
Normalement, le comportement d'objet est contrôlé par la bibliothèque. Ceci est facilité en employant l'arrangement de transmission de messages décrit plus tôt. Les messages reçus sont traités, et basés sur le contenu de message, l'état d'objet affecté est changé. La bibliothèque refait alors automatiquement l'objet pour montrer le changement de l'état.

La figure 3.2 montre un flux simple employant la Bibliothèque graphique. Supposant que les drivers de module et d'affichage d'interface utilisateurs sont choisis et le codage supplémentaire et minimal sera nécessaire d'abord :

- `InitGraph()` est appelée pour remettre à zéro le contrôleur d'affichage, pour déplacer la position de curseur à (0.0) et pour initialiser l'affichage à tout noir.
- Après, `GOLCreateScheme()` est appelée pour définir le modèle d'arrangement à employer pour les objets.

Si aucun changement à l'arrangement de modèle n'est spécifié, l'arrangement par défaut est employé. Dans ce cas-ci, les fonctions `InitGraph()` et `GOLCreateScheme()` peuvent être exécutées juste après un appel de la fonction à `GOL_Init()`.

Figure 3.2 : schéma utilisation de bibliothèque graphique



3.9 CONCLUSION

La bibliothèque graphiques de Microchip est une bibliothèque disponible pour le microcontrôleur PIC.

Elle fournit les objets prêts pour des applications qui exigent une commande de type gadget dans leurs interfaces. Son architecture rend la bibliothèque indépendante sur le matériel d'affichage utilisé et l'exige seulement une création ou une modification d'un programme pilote de périphérique. Ceci facilite une migration d'un dispositif d'affichage à des autres.