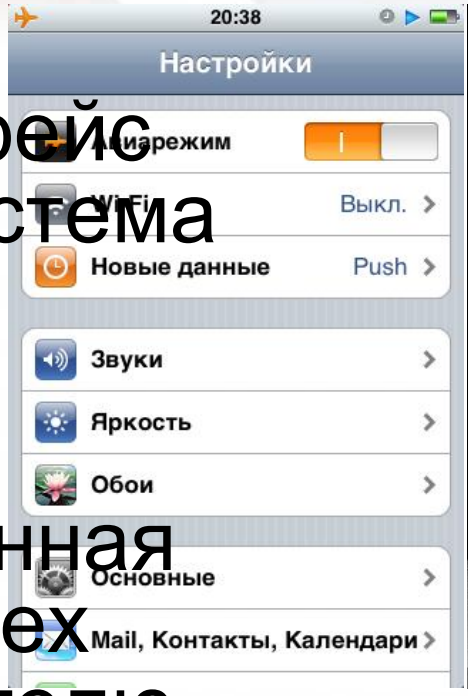
Графический интерфейс пользователя

Разработка на базе графической библиотеки Microchip

Что такое GUI?

- Графический интерфейс пользователя это система средств для взаимодействия пользователя с устройством, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана.



О чем данный класс

Чему учит данный класс:

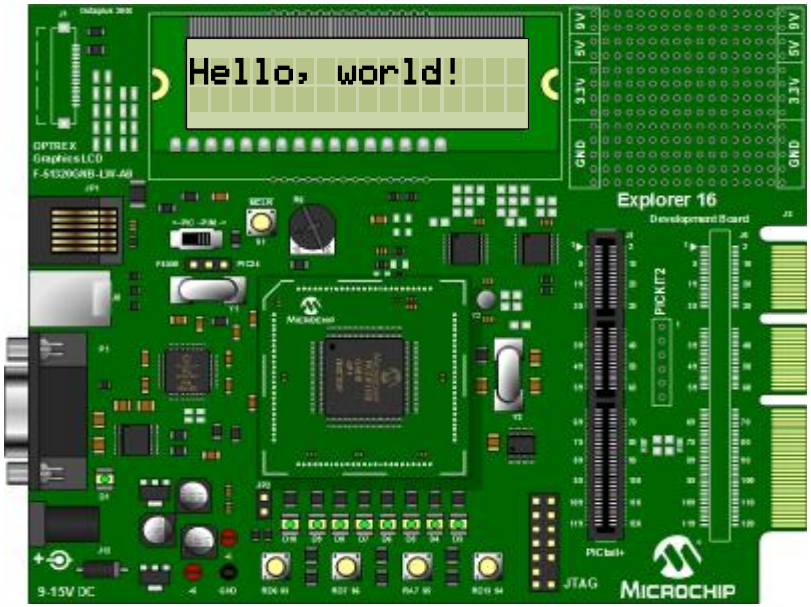
- Написанию низкоуровневых драйверов для графической библиотеки
- Созданию программ для отображения картинок, шрифтов и примитивов на ЖК-панель
- Созданию графического интерфейса с использованием всех возможностей графической библиотеки Microchip

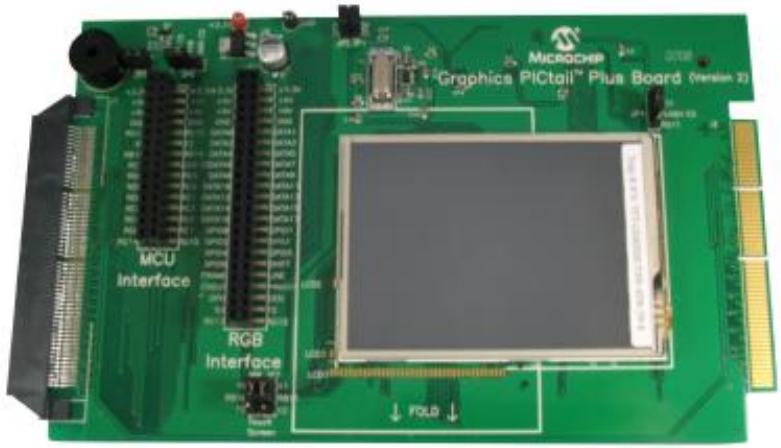
Краткое описание графической библиотеки

Особенности графической библиотеки Microchip

- **Работает с 16-и и 32-битными PIC® микроконтроллерами**
- **Бесплатная библиотека**
  - Открытый исходный код
  - Поддержка различных ЖК-панелей
- **Поддержка многоканальных методов ввода**
- **Независимость от размера экрана и разрешения**
- **Дешевые полнофункциональные отладочные средства:**
  - Explorer 16
  - Дочерняя плата Graphics PICtail™ Plus
  - Бесплатная утилита для создания шрифтов и картинок
- **Скачайте с www.microchip.com/graphics**

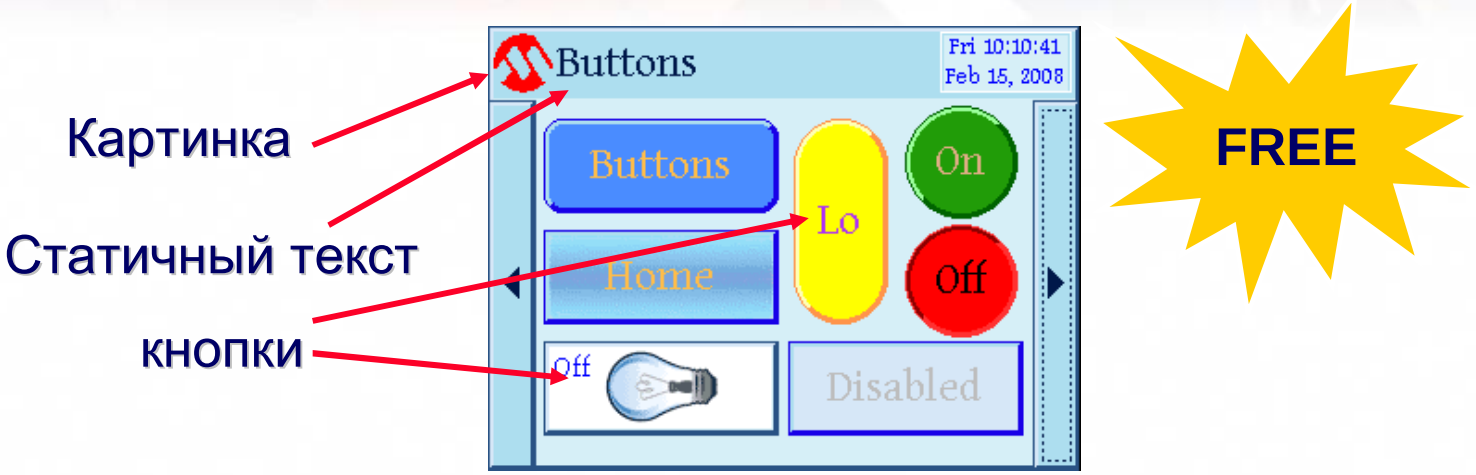
Демонстрационная плата Graphics PICtail™ Plus



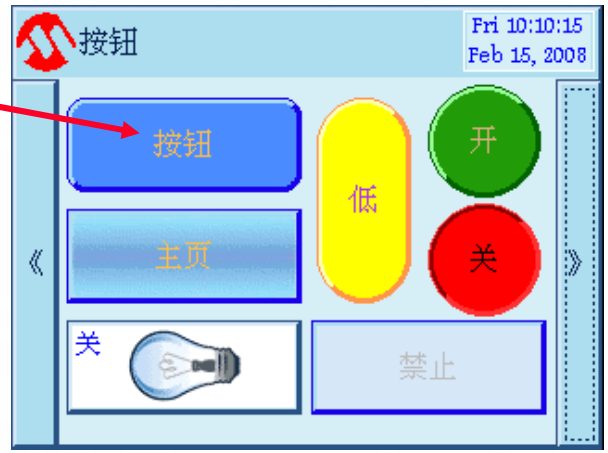


- **Работает с 16-и и 32-битными микроконтроллерами**
- **Конфигурирование перемычками:**
  - ЖК-панель с интегрированным контроллером (LGDP4531) или
  - Разъем для подключения внешнего ЖК и контроллер SSD1906

Графическая библиотека



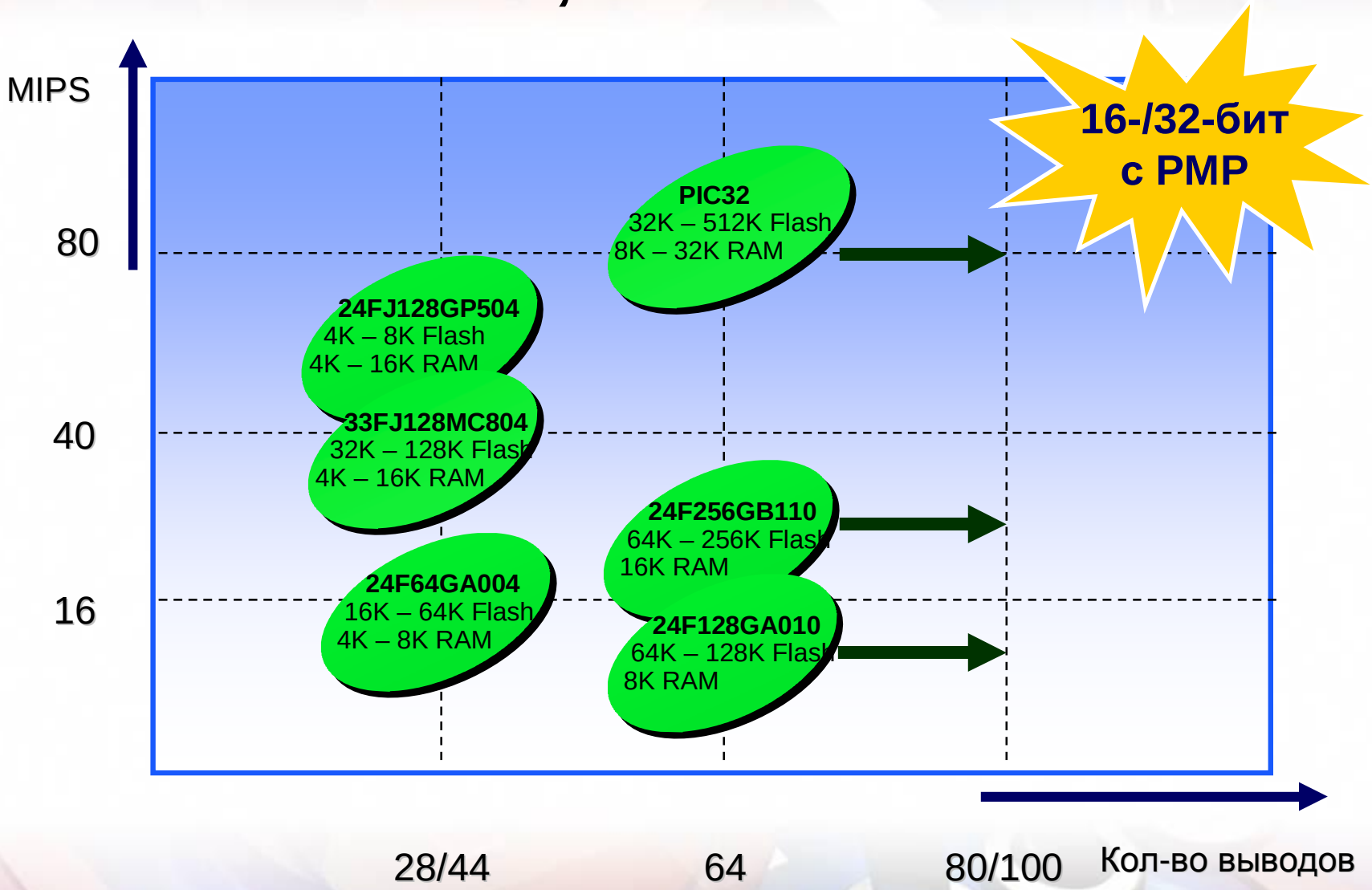
Разноязычные шрифты



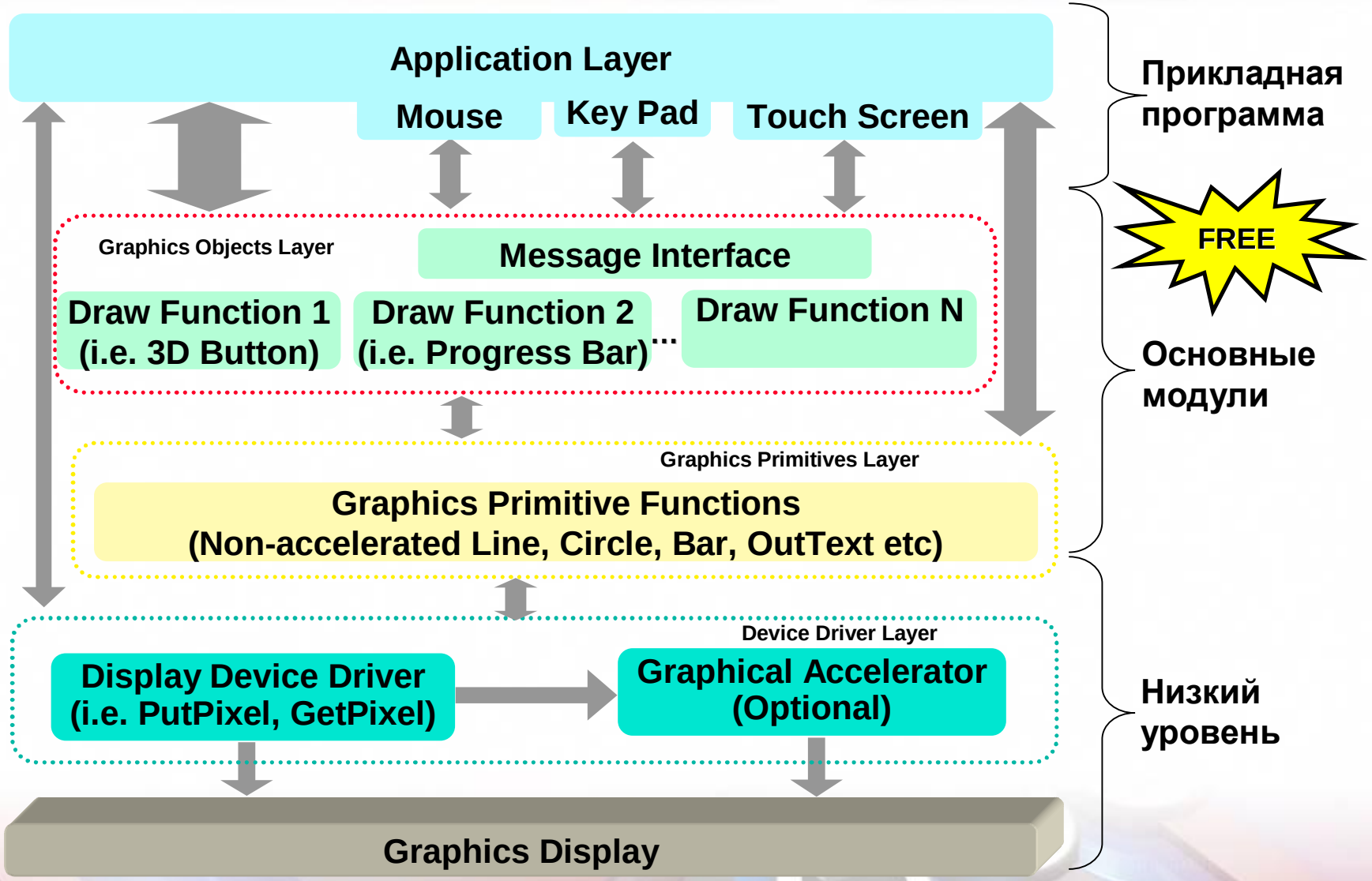
Ключевые моменты:

- Бесплатно
- Модульная структура
- Низкие требования по памяти и производительности
- Поддержка и тренинги Microchip

Поддержка как 16-и, так и 32-битными PIC®

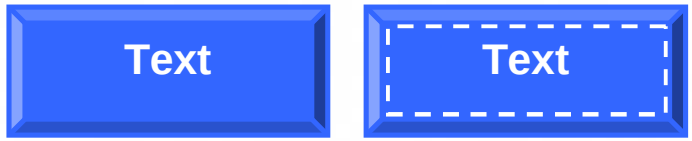


Структура библиотеки



Настройка библиотеки #defines в GraphicsConfig.h

- #include для header-файлов
- **Выбрать методы управления**
  - Сенсорный дисплей или клавиатура
  - *Другие методы в ближайшем будущем…*
- **Выберете используемые заготовки**
  - кнопки, checkbox, слайдер, и т.п.
- **Вкл/Выкл фокусировку объектов**
- **Графический режим**
  - монохромный
  - ориентация
  - Поддержка UNICODE



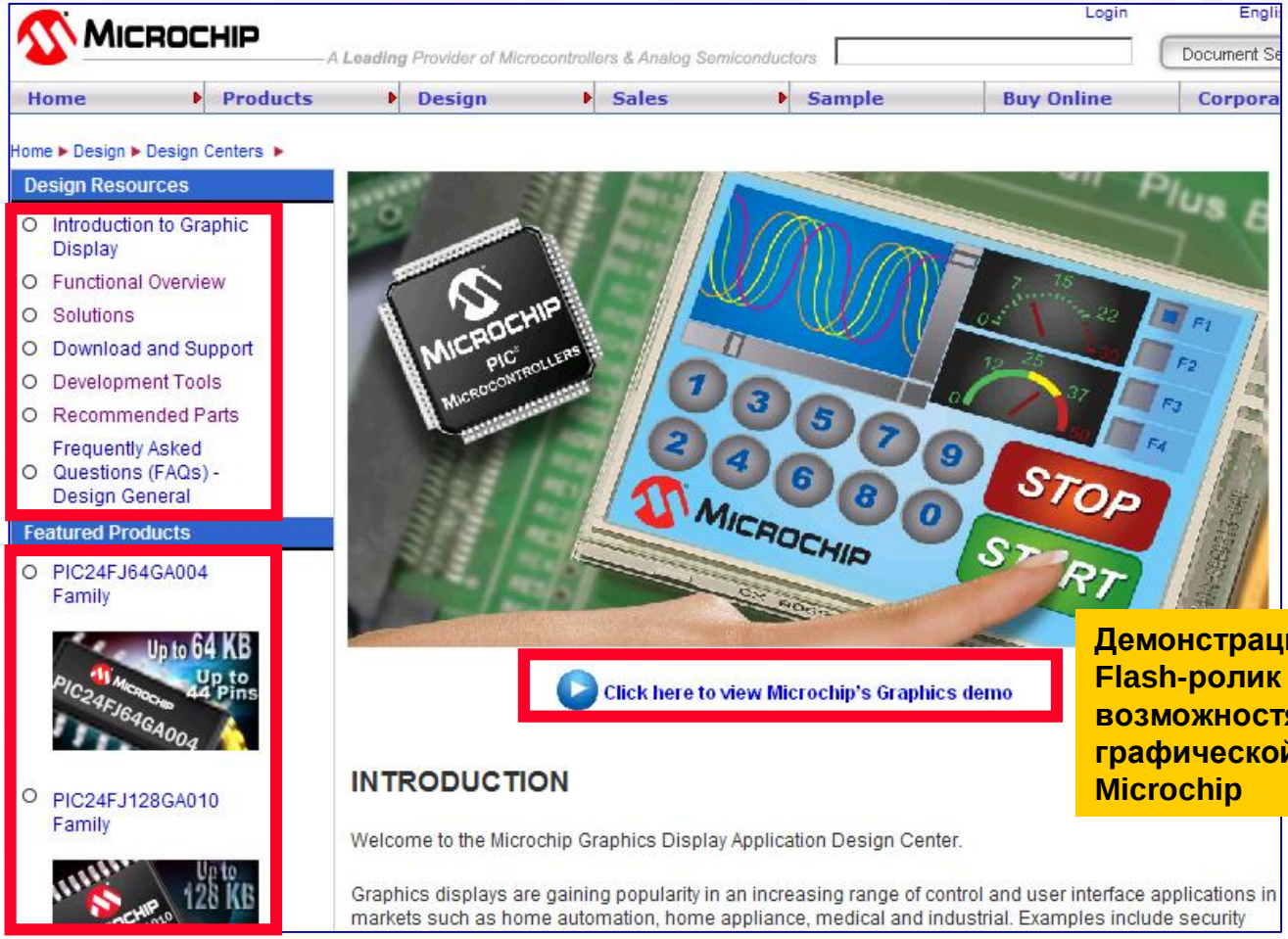
Настройка библиотеки #defines в GraphicsConfig.h

- **Расположение картинок**
  - Внутренняя Flash, внешняя память, или и там, и там
- **Неблокирующая конфигурация**
  - Прорисовка объектов на основе состояния объектов
  - Эффективное использование ресурсов процессора
  - `#define USE_NONBLOCKING_CONFIG`
- **Блокирующая конфигурация**
  - Линейное выполнение программы
  - Прямое управление обновлением экрана
  - `// #define USE_NONBLOCKING_CONFIG`

Дизайн-центр по графическим приложениям

Навигация сайта

Перечень продукции



Демонстрационный Flash-ролик о возможностях графической библиотеки Microchip

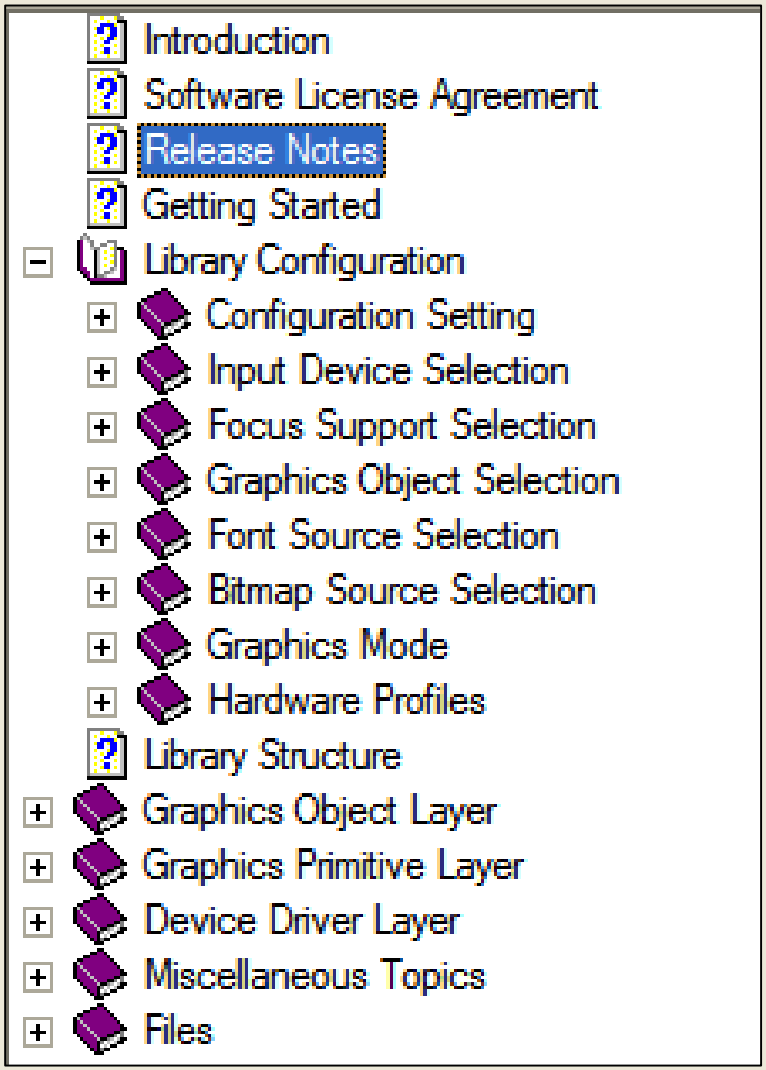
http://www.microchip.com/graphics

Графический дизайн-центр

- Графическая библиотека
  - Исходный код, схемы, примеры, документация, утилиты
- Дополнительная информация
  - AN1227 – Использование клавиатуры с библиотекой Microchip
  - AN1136 – Как использовать элементы библиотеки Microchip
  - AN1182 – Шрифты в библиотеке Microchip
- Web-семинары
  - Знакомство с графическими решениями Microchip
  - Как работает графическая ЖК-панель?
  - Интерфейс PIC24 и графического ЖК-дисплея
  - Архитектура графической библиотеки Microchip
- Microchip RTC (часы реального времени)
  - Использование в графической библиотеке Microchip (HIF 2131)

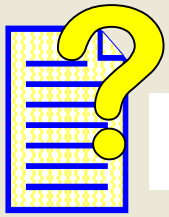
http://www.microchip.com/graphics

Удобный Help к библиотеке



Help files are included as part of the MPLAB® Graphics Library installation and are located in the following directory:

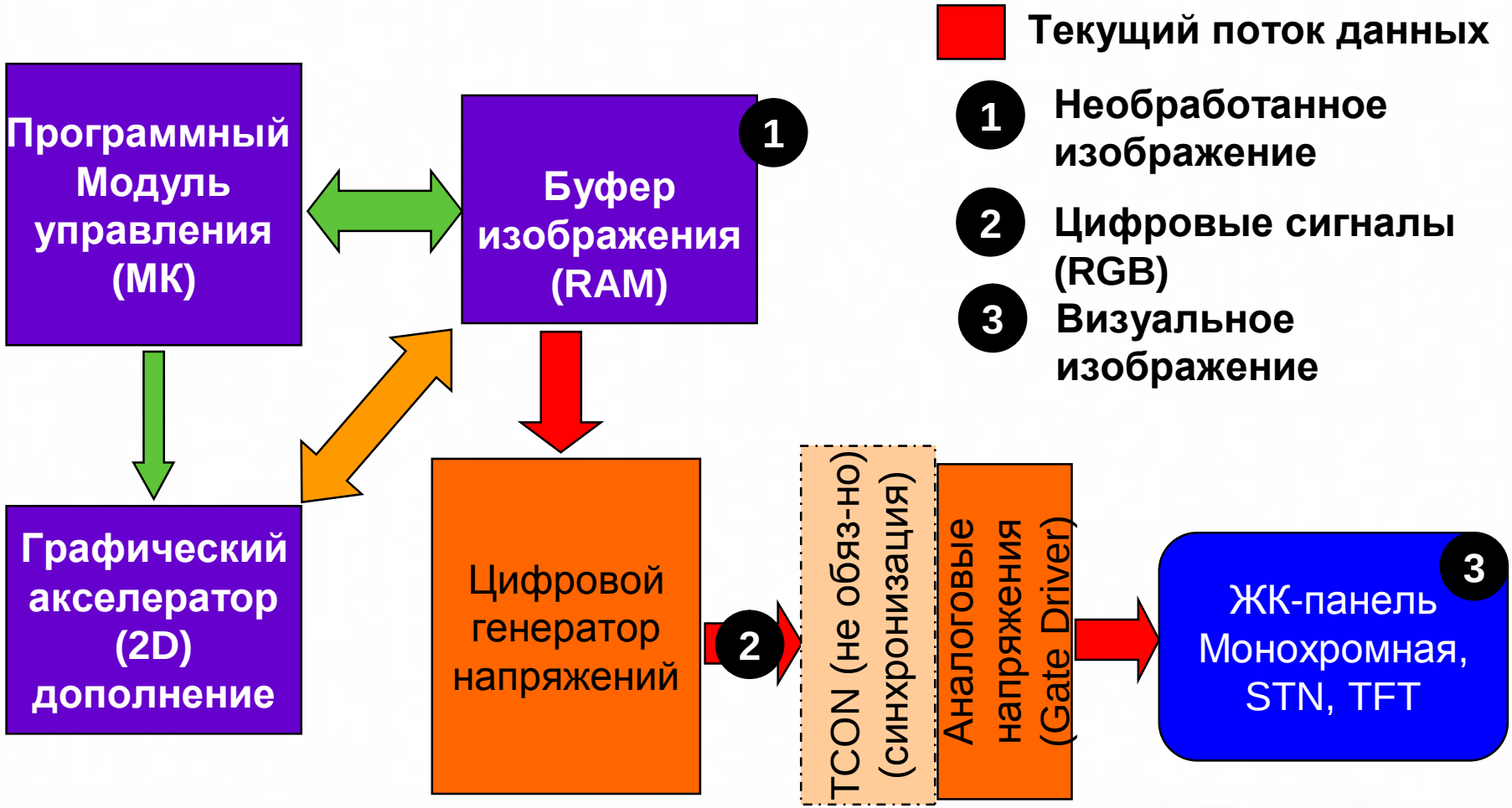
C:\Microchip Solutions\Microchip\Help



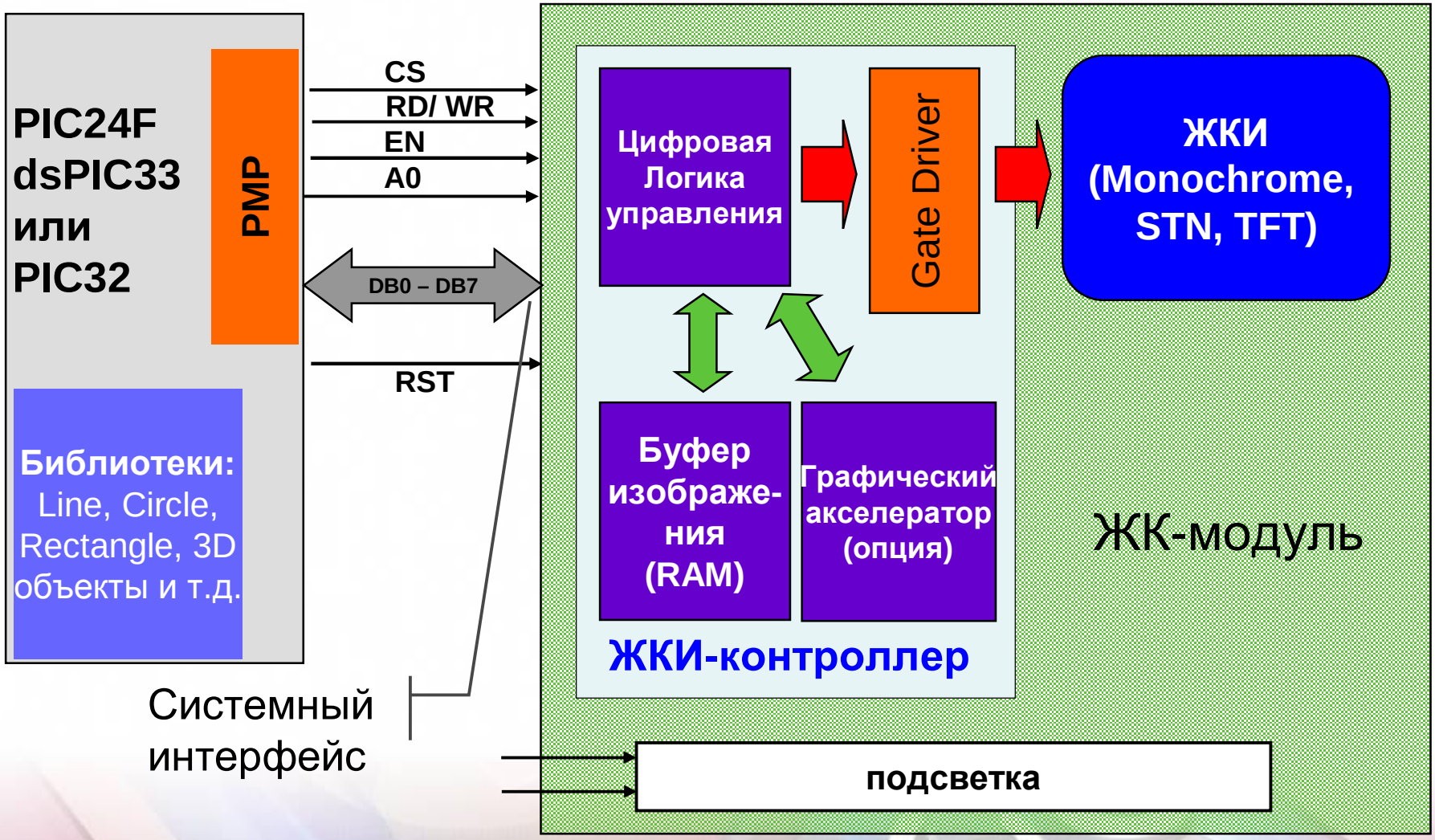
Graphics Library Help.chm

Основы работы с ЖК-дисплеями

Система с графической ЖК-панелью

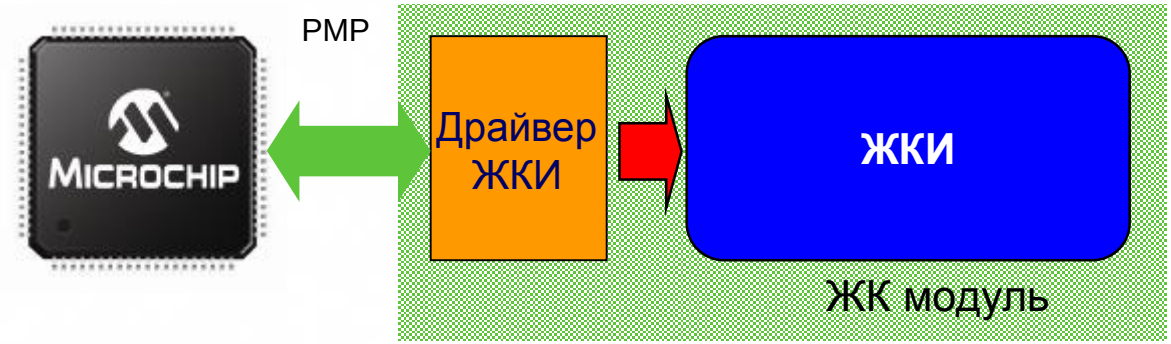


Параллельный интерфейс для управления ЖК-дисплеем



Графические решения

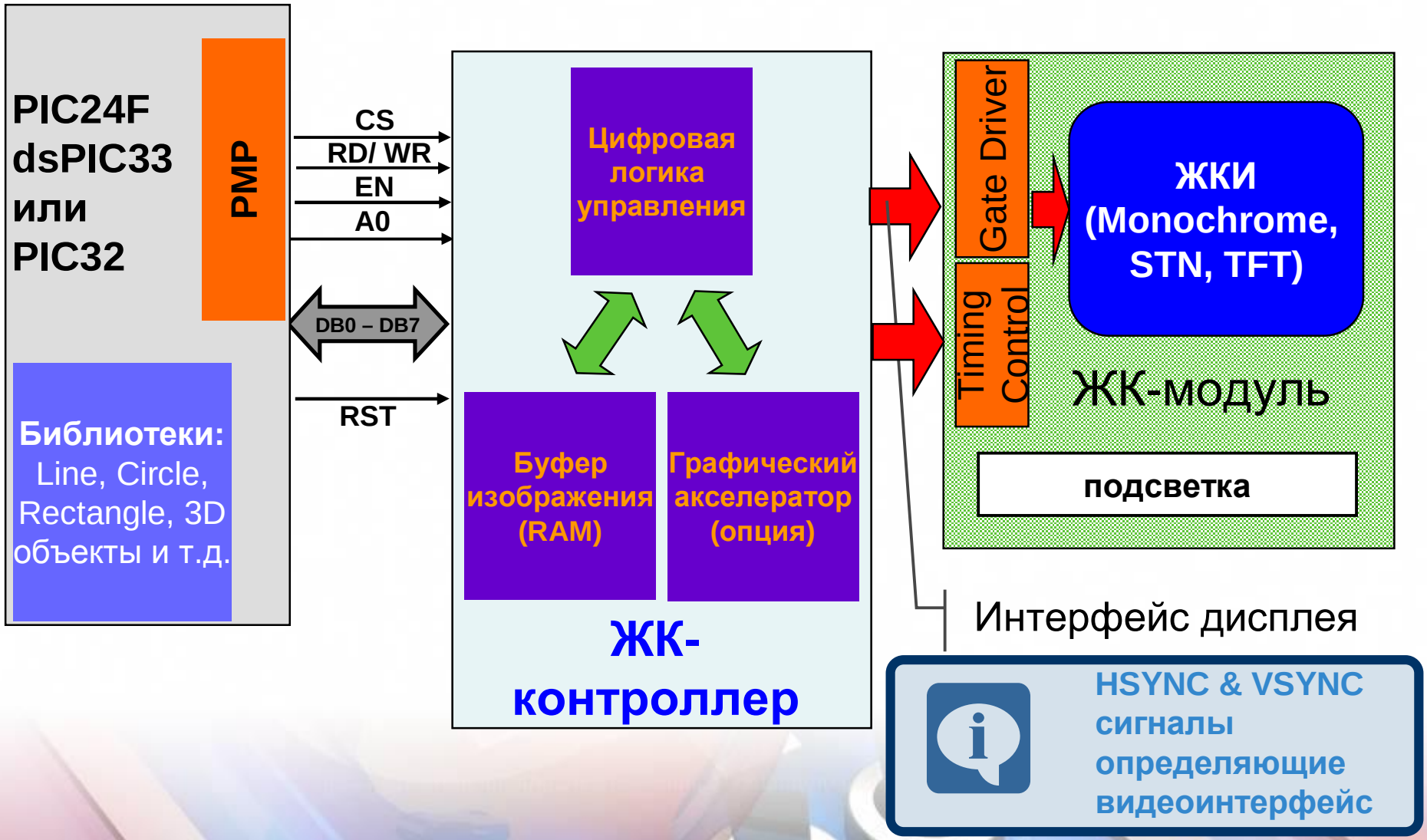
Решение #1 на базе параллельного порта (PMP)



- Параллельный интерфейс PMP для связи с ЖК-модулем
- Решение для ЖКИ<2.8”
- Библиотека Microchip может работает с любыми драйверами и ЖКИ
- Готовые драйвера для некоторых моделей

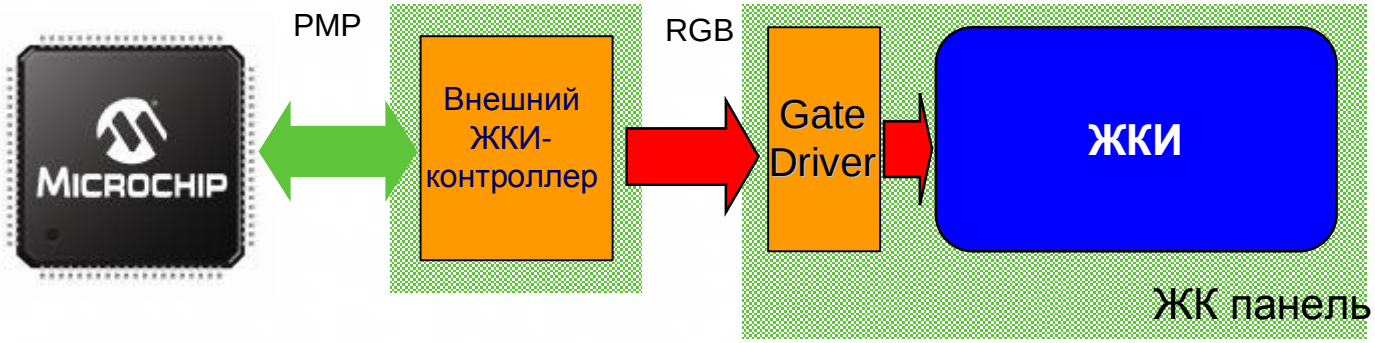
Поддержка ЖКИ-контроллеров	
производитель	**наименование**
HiTech	HIT1270
LG	LGDP4531
Renesas	R61505U
Orise Tech	SPFD5408A
Samsung	S6D0129/0139
Solomon Systech	SSD1339

Видеоинтерфейс ЖКИ (например, RGB)



Графические решения

Решение #2 на безе PMP



- Решение для ЖКИ>2.8”
- ЖКИ-контроллер включает RAM-буфер изображения и графический акселератор
- Библиотека Microchip может работать с любым ЖКИ с RGB интерфейсом

Поддержка ЖКИ-контроллеров	
производитель	**наименование**
HiMax	HX8306A
Ampire	FSA506
Solomon Systech	SSD1906/1905

Основы работы с ЖК-дисплеями

советы и уловки по написанию низкоуровневых драйверов

Низкоуровневые драйвера

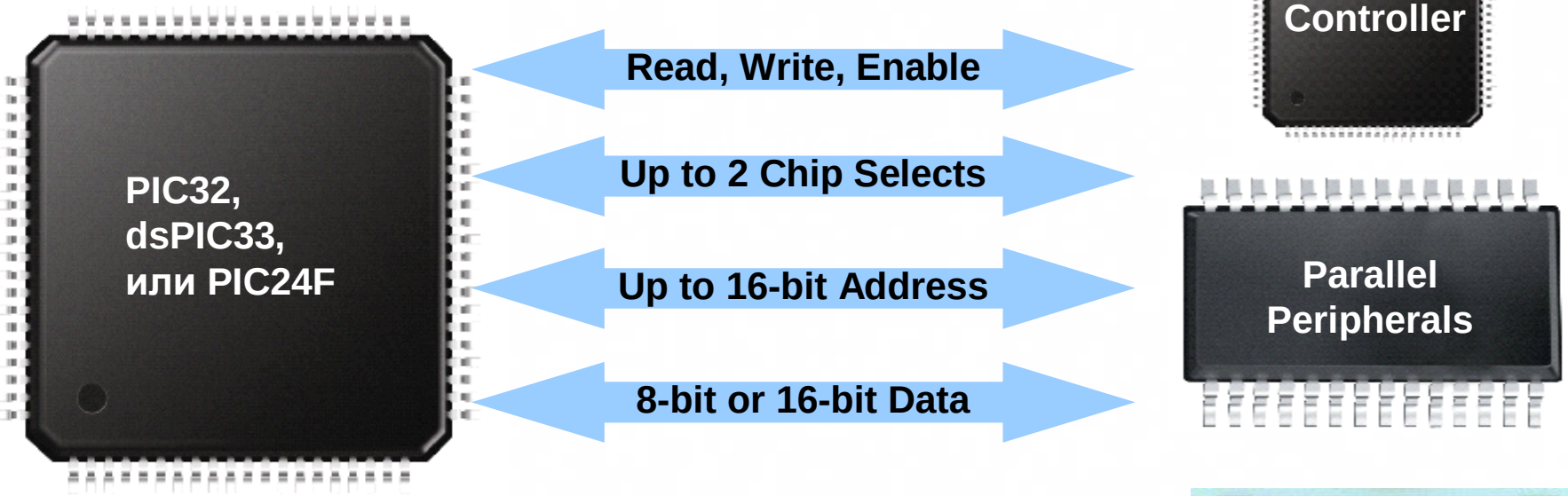
- **Интегрированные в ЖКИ контроллеры:**
  - HiTech: HIT1270
  - LG: LGDP4531
  - Renesas: R61505U
  - Orise Tech: SPFD5408A ***(new)***
  - Samsung: S6D0129, S6D0139
  - Solomon Systech: SSD1339
  - Ampire: FSA506 ***(new)***
  - HiMax: HX8306A
- **Внешние ЖКИ-контроллеры:**
  - Solomon Systech: SSD1905, SSD1906 ***(new)***
- *Перечень постоянно расширяется…*

Низкоуровневые драйвера советы и уловки

- **Убедитесь, что ЖКИ поддерживает интерфейс I80 или M68**
  - если RGB-интерфейс, используйте графический контроллер
- **Скопируйте файлы `[driver].c` и `[driver].h` соответствующие Вашему контроллеру**
  - Тип дисплея
  - Глубина цвета
  - Разрешение
- **Проверьте адреса и доступ к регистрам**
  - Индексные регистры (как в LGDP4531)
    - **Адресные линии PMP не используются**
  - Распределение в памяти (как в SSD1906)
    - **Адресные линии PMP задействованы**
- **Если ЖК-модуль/контроллер имеет акселератор, дополните соответствующими функциями `[driver].c`**

Написание драйверов для параллельного порта – PMP

- `ResetDevice()` инициализация PMP
- PMP поддерживает интерфейсы I80 и M68



Для увеличения производительности используйте DMA в PIC32 и dsPIC

Низкоуровневые драйверы
Советы

- **Исправьте файл `[Driver].h`**
  - Установите разрешение …

```
…
// Defines the horizontal screen size.
#define SCREEN_HOR_SIZE          240
// Defines the vertical screen size.
#define SCREEN_VER_SIZE          320
```

  - задайте основные цвета …

```
…
#define BLACK                RGB565CONVERT(0,0,0)
#define BRIGHTBLUE           RGB565CONVERT(0,0,255)
#define BRIGHTGREEN          RGB565CONVERT(0,255,0)
#define BRIGHTCYAN           RGB565CONVERT(0,255,255)
…
```

Низкоуровневые драйверы Советы

- **Определение основных примитивов драйвера …**
  - Необх. функции определяются на уровне примитивов

```
//#define USE_DRV_FONT
//#define USE_DRV_LINE
//#define USE_DRV_CIRCLE
//#define USE_DRV_FILLCIRCLE
#define USE_DRV_BAR
#define USE_DRV_CLEARDEVICE
#define USE_DRV_PUTIMAGE
```

- **Установите память строки (буфер изображения) …**
  - Необходимо для определения адреса координаты пикселя

```
…
// Memory pitch for line
#define LINE_MEM_PITCH                256
…
```

240(RGB) x 320 LINE_MEM_PITCH

- Распределение памяти, согласно даташиту на драйвер LGDP4531

AD[16:0]	GRAM Data Map
17’h00000 … 17’h000EF	1st Line GRAM Data
17’h00100 … 17’h001EF	2nd Line GRAM Data
……	…..
17’h13D00 … 17’h13DEF	318th Line GRAM Data
17’h13E00 … 17’h13EEF	319th Line GRAM Data
17’h13F00 … 17’h13FEF	320th Line GRAM Data

- 256 байт на линию

Низкоуровневые драйверы Советы

- **Возьмите код инициализации у производителя**
- **Модифицируйте API в копиях файлов:**
  - `ResetDevice()`– инициализация
  - `SetIndex()` – используется для доступа к индексным регистрам
- **Другие API, которые могут не требовать изменений:**
  - `PutPixel()`
  - `GetPixel()`
  - `WriteData()`
  - `SetAddress()`
- **Допишите следующие API при необходимости:**
  - `SetActivePage()`
  - `SetVisualPage()`
  - `SetPalette()`

Библиотека Microchip уровень примитивов

Уровень примитивов

- **Обращается исключительно к низкоуровневому драйверу**
- **Конфигурация компиляции: `GraphicsConfig.h`**
  - Выберете нужный драйвер
  - Источник шрифтов (внутренняя или внешняя)
  - Поддержка UNICODE (AN1182)
  - Источник картинок (внутренняя или внешняя память)

Основные примитивы API

- **Начальные установки**
  - **`InitGraph()`** – инициализация дисплея
    - **Примитивы в Primitive.c**
  - **`ClearDevice()`** –
    - **Очистка дисплея нужным цветом**
    - **Установка курсора в (0,0)**

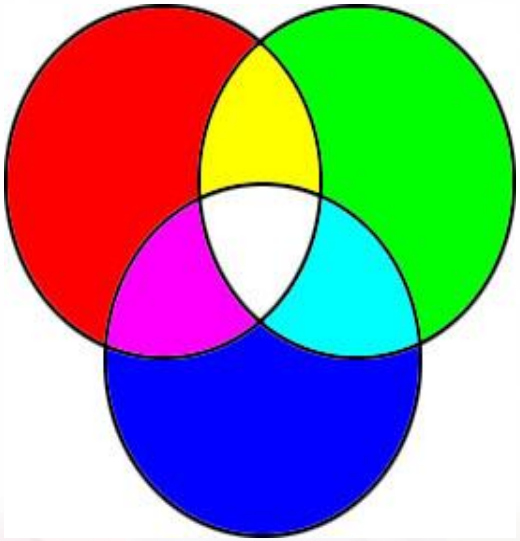
Примитивы APIs

- **Универсальные функции**
  - Функции действующие до следующей установки *Set*
    - **`SetColor(COLOR)` – выбор цвета**
      - Цвета описаны в файле **`driver.h`**
    - **`SetFont(&fontimage)` – выбор шрифта**
    - **`SetLineType(key)`**
      - ***`SOLID_LINE`***
      - ***`DOTTED_LINE`***
      - ***`DASHED_LINE`***
    - **`SetLineThickness(key)`**
      - ***`NORMAL_LINE`***
      - ***`THICK_LINE`***

Модель RGB

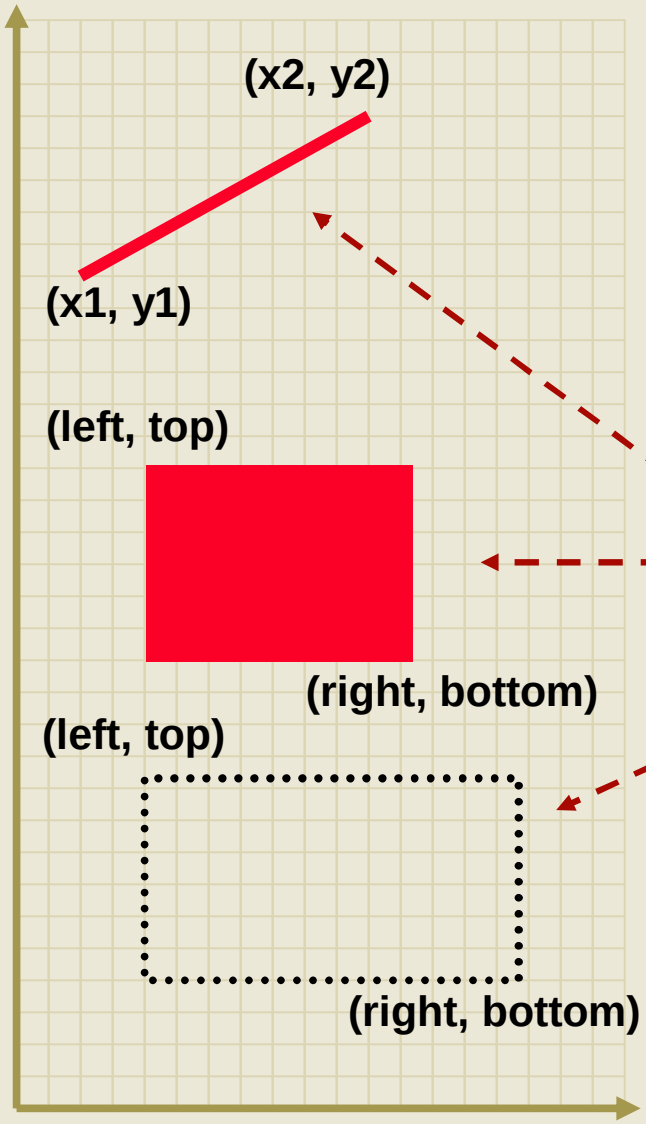
Определение:

RGB модель это аддитивная цветовая модель, основанная на синтезе цвета за счет добавления к черному красного, зеленого и синего цветов в нужных пропорциях.



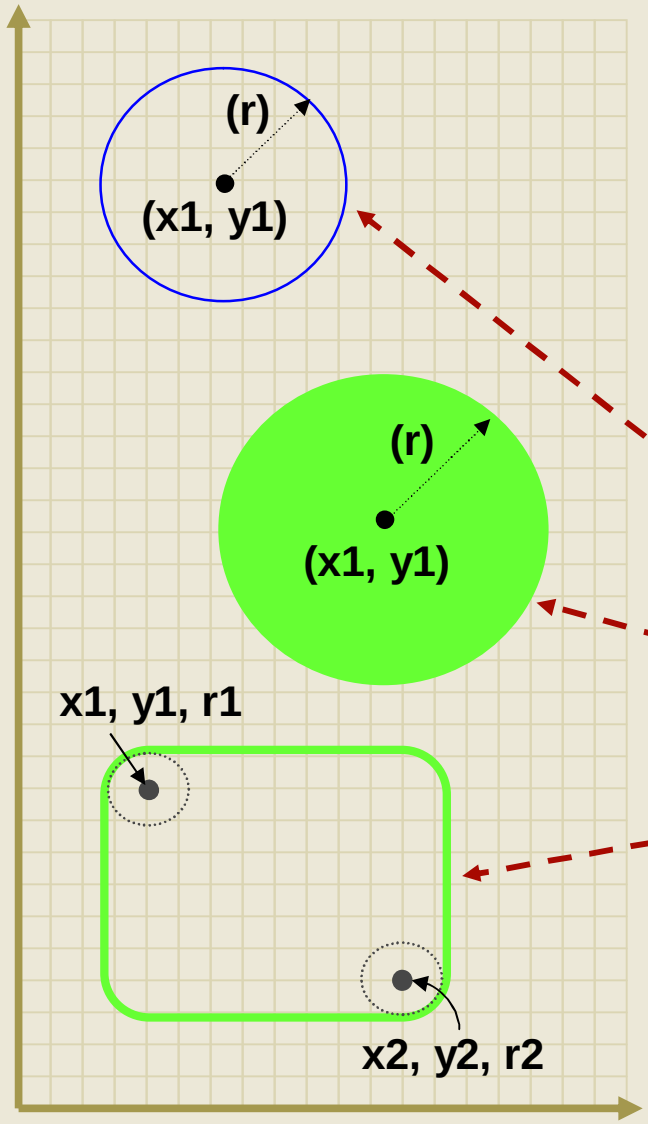
- **16-bit (65,536 цветов)**
  - Red: 5 bits
  - Green: 6 bits
  - Blue: 5 bits
- **18-bit (262,144 цветов)**
  - Red: 6 bits
  - Green: 6 bits
  - Blue: 6 bits
- **Составляющие определяют интенсивность**

Графические примитивы



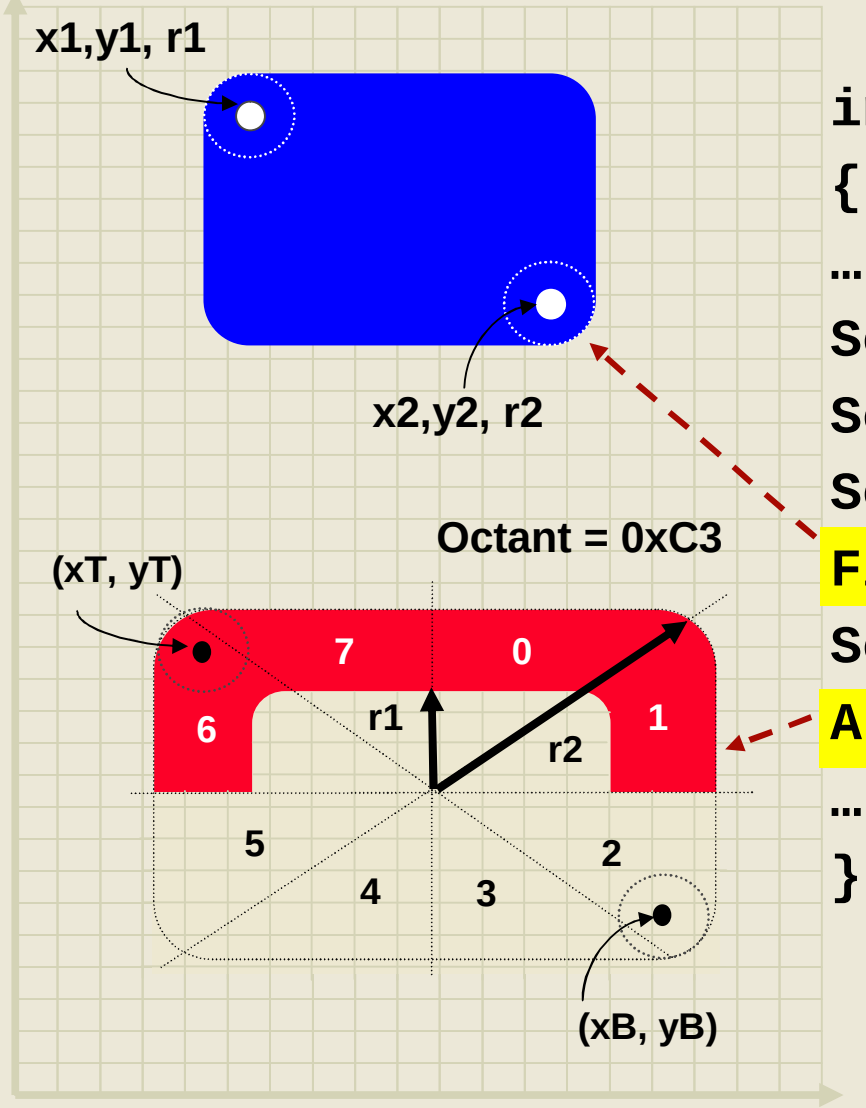

```
int main(void)
{
….
SetColor(BRIGHTRED);
SetLineType(SOLID_LINE);
SetLineThickness(THICK_LINE);
Line(x1, y1, x2, y2);
Bar(left, top, right,bottom);
SetColor(BLACK);
SetLineType(DOTTED_LINE);
Rectangle(left, top, right,bottom);
…
}
```

Графические примитивы




```
int main(void)
{
….
SetColor(BRIGHTBLUE);
SetLineType(SOLID_LINE);
SetLineThickness(NORMAL_LINE);
Circle(x1, y1, r);
SetColor(BRIGHTGREEN);
FillCircle(x1, y1, r);
SetLineThickness(THICK_LINE);
Bevel(x1, y1, x2, y2, r1, r2);
….
}
```

Графические примитивы




```
int main(void)
{
….
SetColor(BRIGHTBLUE);
SetLineType(SOLID_LINE);
SetLineThickness(NORMAL_LINE);
FillBevel(x1,y1,x2,y2,r1,r2);
SetColor(BRIGHTRED);
Arc(xT,yT,xB,yB,r1,r2,Octant);
….
}
```

Уровень примитивов библиотеки Microchip (шрифты и картинки)

Шрифты

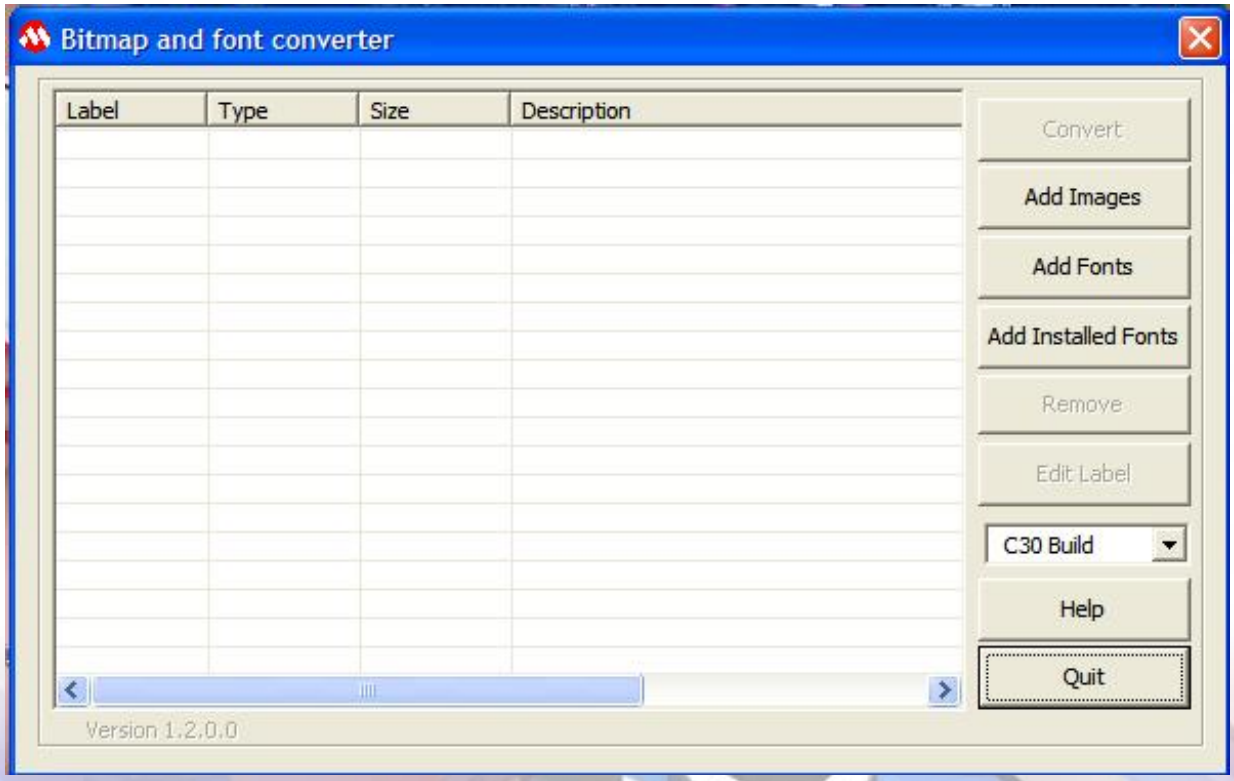
Определение:

Шрифт – графический рисунок начертаний букв и знаков, составляющих единую стилистическую и композиционную систему. Существуют специализированные утилиты для создания шрифтов. Обычно используются готовые шрифты.

- **Библиотека поддерживает конвертацию …**
  - Шрифты True Type и Open Type
  - Растровые шрифты
- **Картинки шрифтов …**
  - Хранение во внешней или внутренней памяти
  - Возможно частичное использование шрифта
- **Поддержка уникода**
  - AN1182 – Fonts in the Microchip Graphics Library

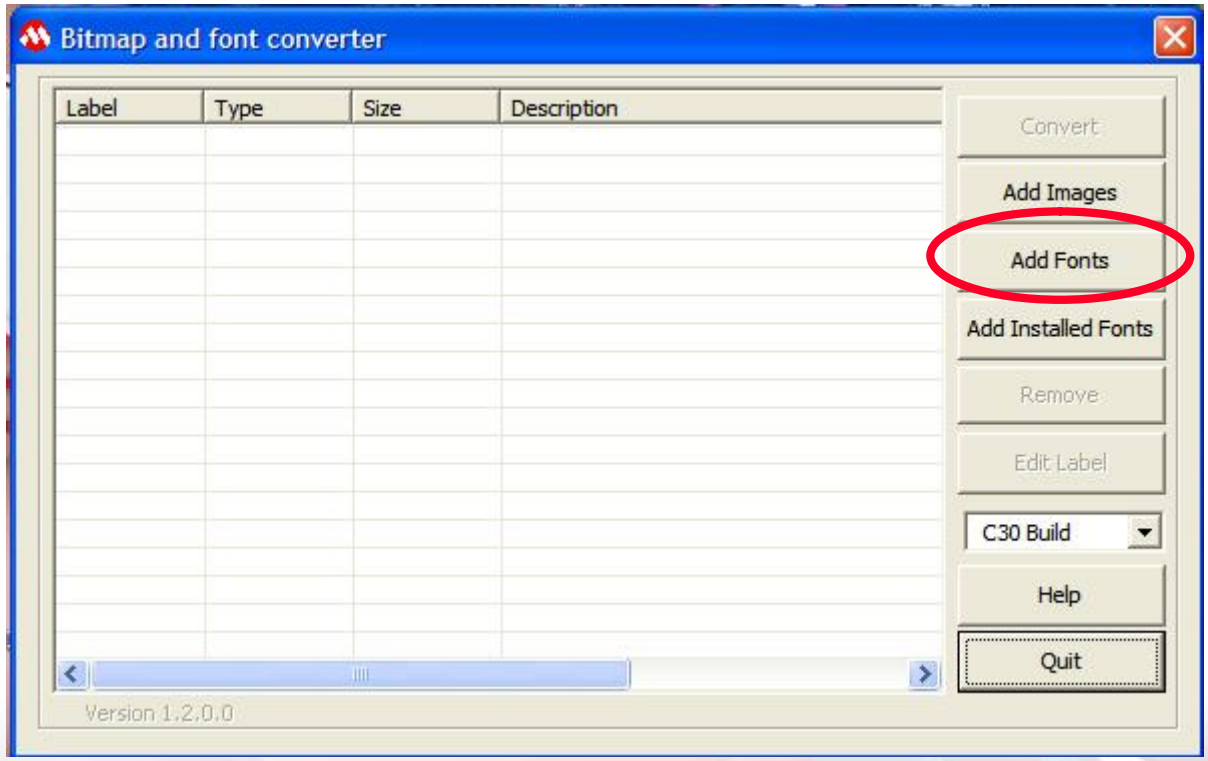
Использование шрифтов

- **Конвертер шрифтов и растровых изображений**
  - Бесплатная утилита в составе библиотеки
  - Конвертирование во внутренний формат



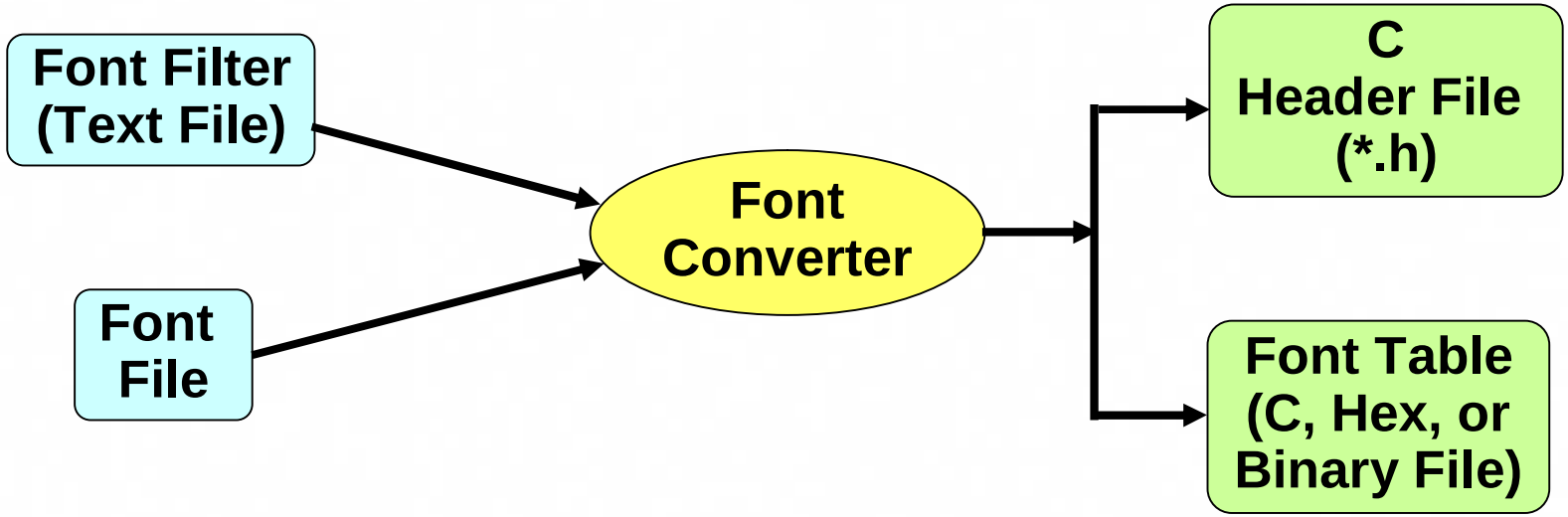
Конвертация шрифтов

- **Конвертированные шрифты …**
  - Могут храниться во внутренней памяти (c-файл)
  - Или во внешней (c или hex-файл)



Фильтрация шрифтов

- **Используется для экономии памяти**
  - Глифам присваивается новый идентификатор ID
  - Требуется «отфильтровать» символы шрифта (в текстовом файле шрифта)
  - Подключить сгенерированный файл к проекту



Фильтрация текстового файла шрифта

- Создать в уникоде(например в Wordpad)
- Сохранить в 16-битном формате
- Каждая строка должна иметь следующее:
  - `<String Label>:<String>//<комментарии>`
    - “//” - обязательно
    - Комментарии необязательно
- Необходимо пересоздать образ шрифта, чтобы отредактировать строки
- Подробнее в документе AN1182 (Fonts in the Microchip Graphics Library)

Пример фильтрования шрифта

- **filter.txt …**
  - Создано и сохранено в Microsoft Wordpad

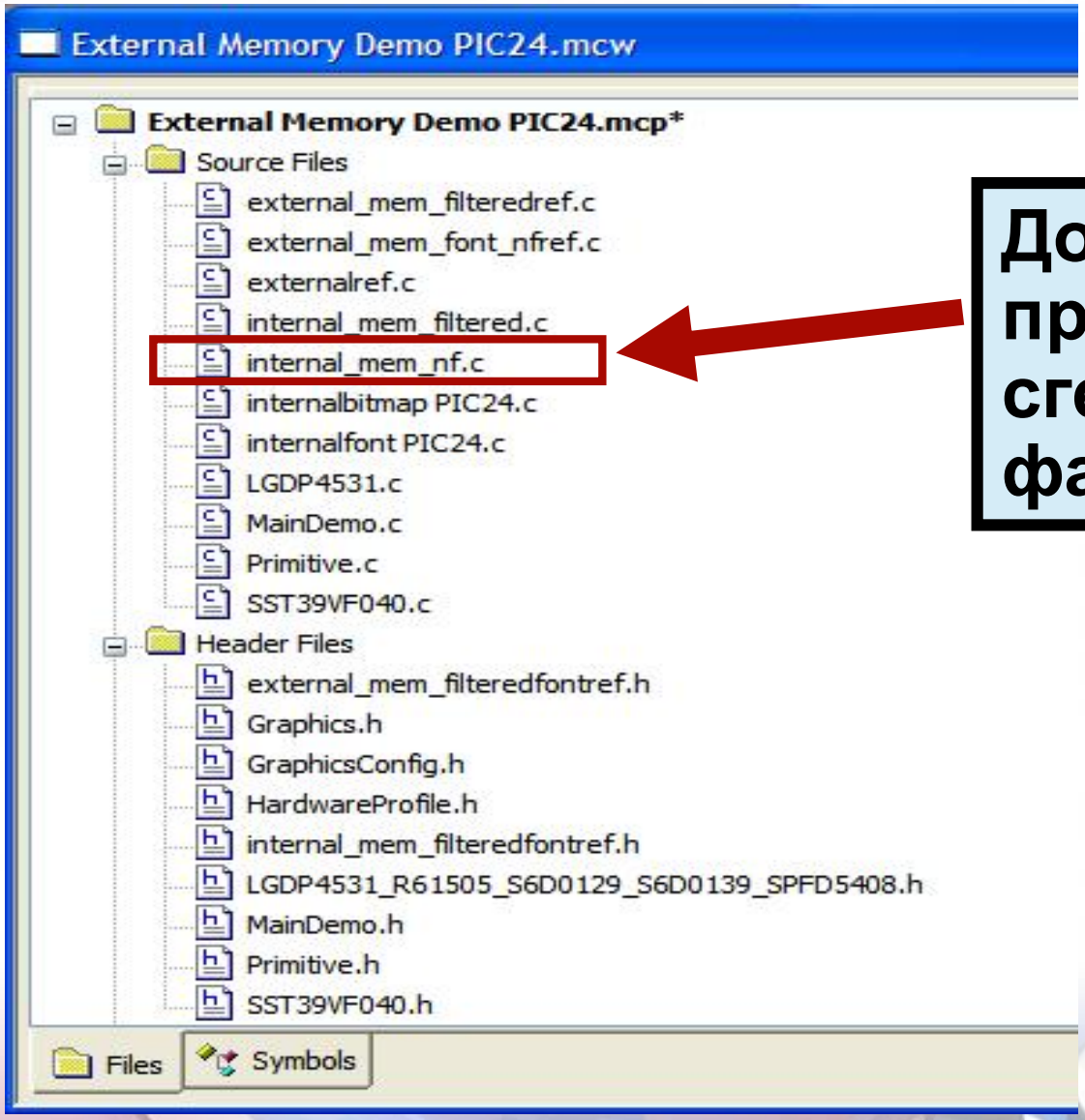
```
HelloWorldStr1:    Hello World!       // first string
HelloworldStr2:    How are you?       //second string
```

- **filtered_font.h**
  - обязательно прописать #include

Преобразовываем в…

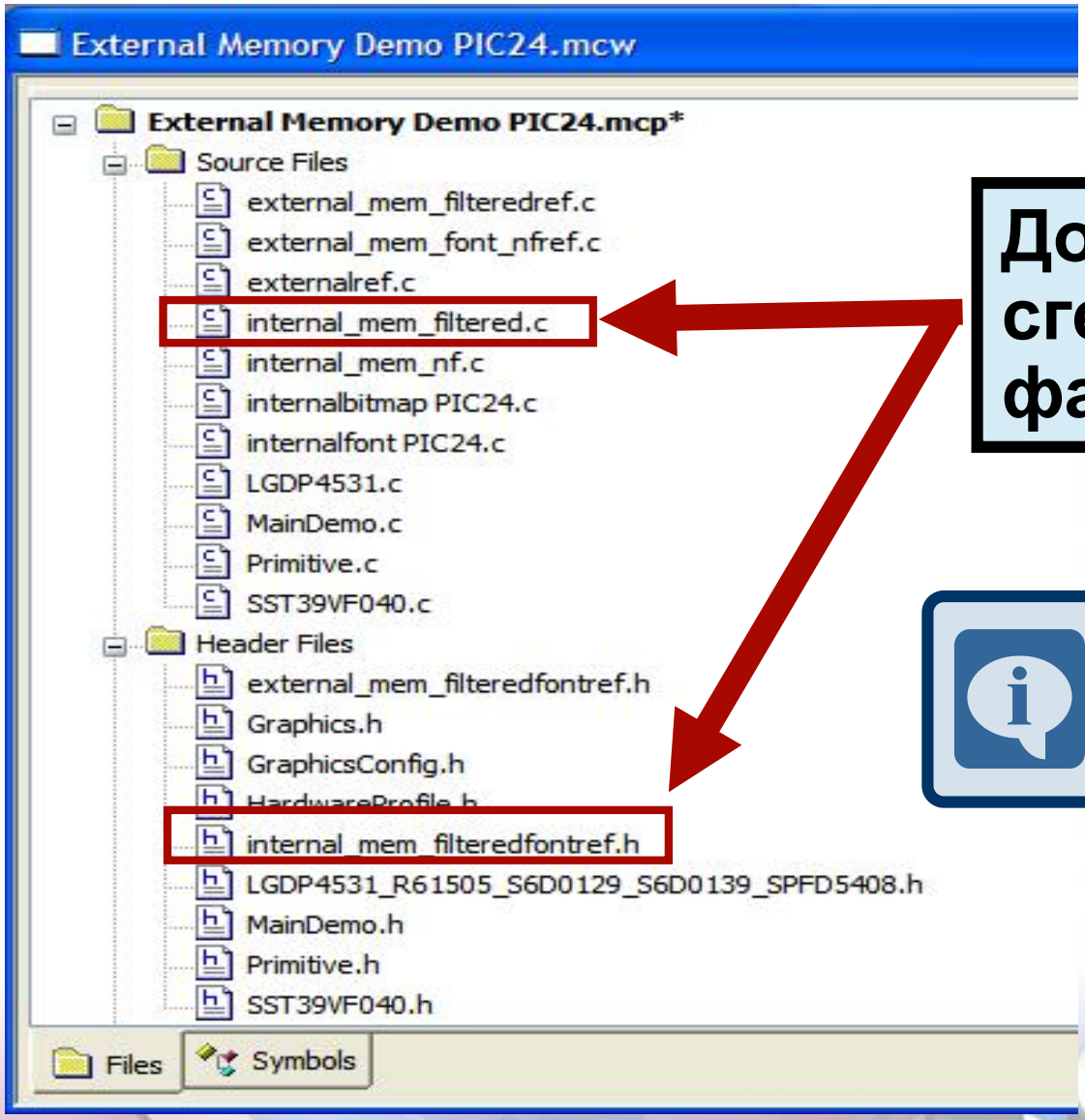
```
#include “Graphics\Graphics.h"

// Automatically generated reference arrays for the filter.txt  file.

XCHAR HelloWorldStr1[] = {0x0020, 0x0020, 0x0020, 0x0020, 0x0022,
0x0029, 0x0025, 0x002A, 0x002B, 0x002D, 0x0020, 0x0023, 0x0024,
0x002D, 0x0027, 0x0028, 0x002C, 0x0029, 0x0020, 0x0020, 0x0020,
0x0020, 0x0020, 0x0020, 0x0000};    // first string

XCHAR HelloWorldStr2[] = {0x0020, 0x0020, 0x0020, 0x0021, 0x0024,
0x0026, 0x0026, 0x0028, 0x0020, 0x0022, 0x0029, 0x0025, 0x002A,
0x002B, 0x002D, 0x0000};            // second string
```

Использование шрифтов



Добавить в проект сгенерированный файл

Using Filtered Fonts



Добавить сгенерированные файлы

Только фильтрованные шрифты требуют h-файл. Не забудьте его включить.

Использование шрифтов

- **Main.c** – объявить образы шрифтов

```
//////////////////// FONTS AND BITMAPS ///////////////////////
// This font is located in internal flash
extern const FONT_FLASH internalfont

// This font must be stored in external flash memory installed on
// Graphics PICTail Plus board
extern FONT_EXTERNAL externalfont;
```

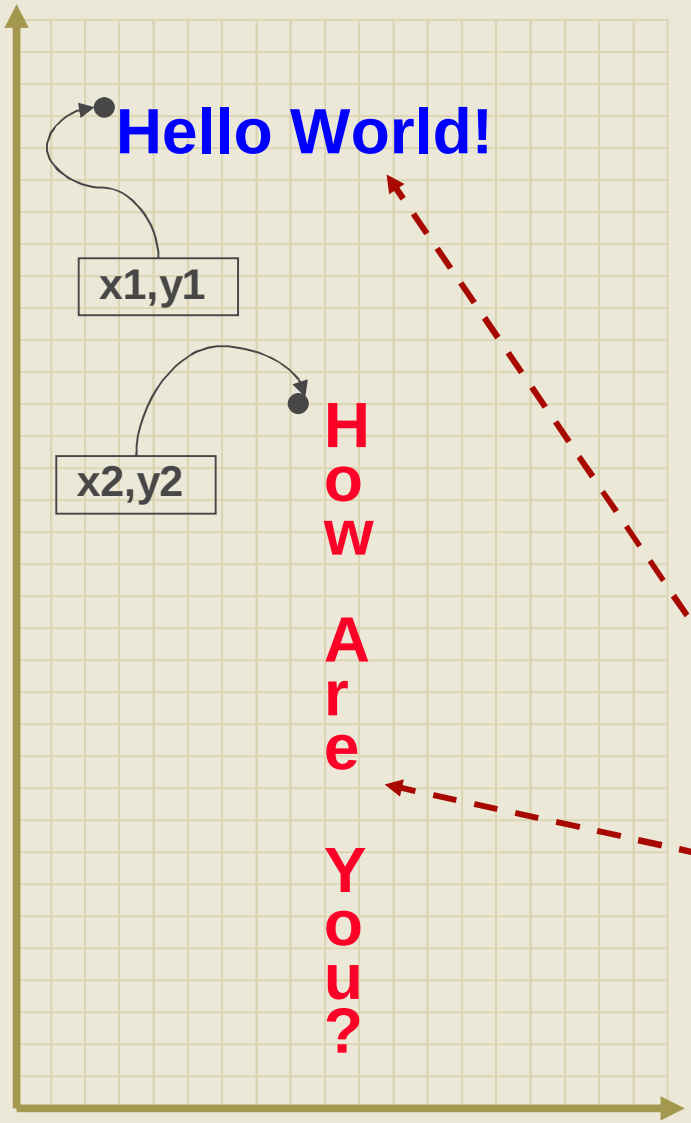
- **internalfont.c** – сгенерировано соотв. утилитой

```
extern const char L11298[] __attribute__((aligned(2)));
//NAME CAN BE CHANGED HERE.
const struct{short mem; const char* ptr;} internalfont =
{0,L11298};
const char L11298[] __attribute__((aligned(2)))
={0x00,0x00,0x20,0x00,0x7F,0x00,0x00,0x23,0x00,0x06,0x88,0x01,0x0
0,0x08,0xAB,0x01,0x00,0x0C,0xCE,0x01,0x00,0x0E,0x14,0x02,0x00,…
```

- **externalfont.c** – сгенерировано соотв. утилитой

```
#include "Graphics\Graphics.h"
FONT_EXTERNAL externalfont = {0x0001,0x0000,0x00000000};
```

Использование шрифтов




```
int main(void)
{
extern const FONT_FLASH myFont;
XCHAR myString1[]="Hello World!";
XCHAR myString2[]="How are you?";
….
SetFont((void*)&myFont);
SetColor(BLUE);
MoveTo(x1,y1); //move cursor
OutText(myString1);
SetColor(BRIGHTRED);
SetFontOrientation(ORIENT_VER);
OutTextXY(x2, y2, myString2);
….
}
```

Картинки

Определение:

Побитовое изображение (bitmap) это поточечное отображение картинки в памяти. Значение каждой точки занимает несколько бит, что зависит от глубины цвета (бит на пиксель - bpp).

1 bpp

4 bpp

8 bpp

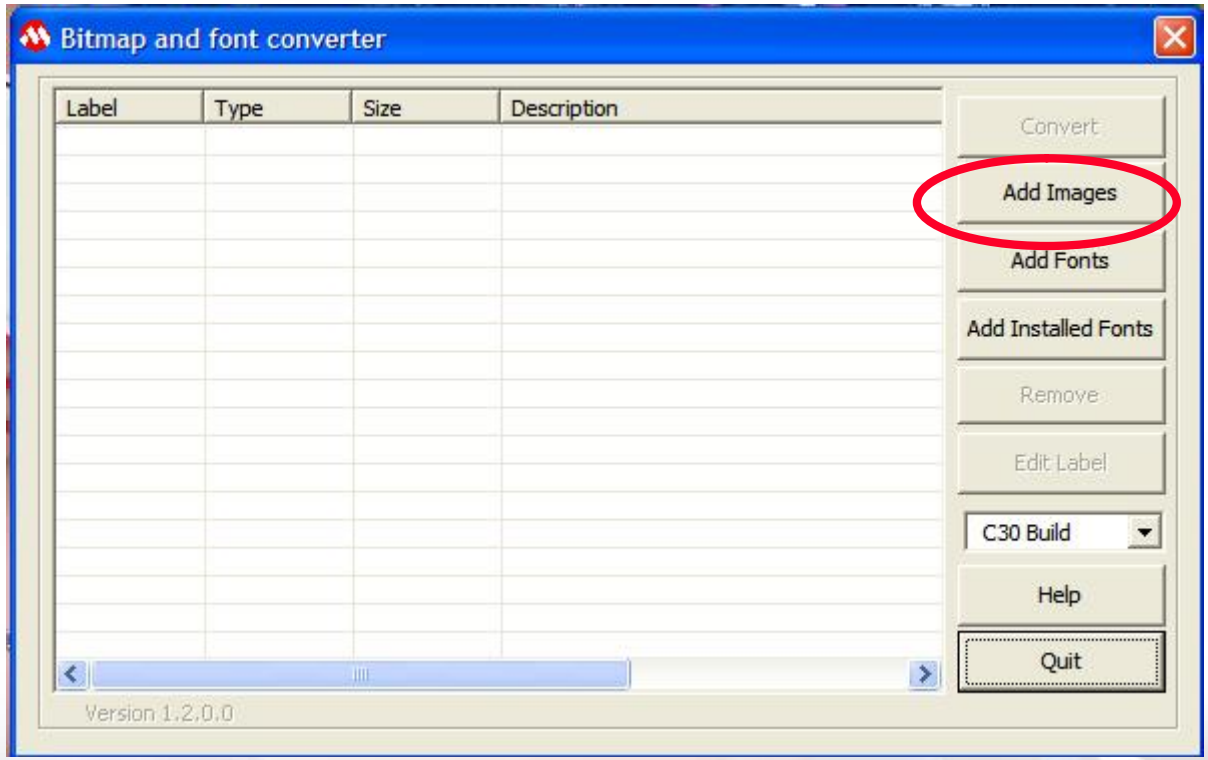
16 bpp

Разрешение и буфер изображения

	X Resolution	Y Resolution	Pixels	Color Depth (bpp)	Colors	Bytes Required
Monochrome						
1/16 VGA	160	120	19,200	1	2	2,400
1/8 VGA	240	160	38,400	1	2	4,800
QVGA	320	240	76,800	1	2	9,600
Color STN						
1/16 VGA	160	120	19,200	8	256	19,200
1/8 VGA	240	160	38,400	8	256	38,400
QVGA	320	240	76,800	8	256	76,800
Color TFT						
1/16 VGA	160	120	19,200	16	65,536	38,400
1/8 VGA	240	160	38,400	16	65,536	76,800
QVGA	320	240	76,800	16	65,536	153,600
1/16 VGA	160	120	19,200	18	262,144	43,200
1/8 VGA	240	160	38,400	18	262,144	86,400
QVGA	320	240	76,800	18	262,144	172,800

Конвертация картинок

- **Конвертированные картинки …**
  - Во внутренней памяти – генерируется c-файл
  - Во внешней – hex-файл



Использование картинок

- **Main.c** – объявление образов картинок

```
//////////////////// FONTS AND BITMAPS ///////////////////////
// This font is located in internal flash
extern const BITMAP_FLASH internal_bitmap;

// This font must be stored in external flash memory installed on
// Graphics PICTail Plus board
extern BITMAP_EXTERNAL  external_bitmap;
```

- **internalbitmap.c** – сгенерировано утилитой

```
extern const char L11298[] __attribute__((aligned(2)));
//NAME CAN BE CHANGED HERE.
const struct{short mem; const char* ptr;} internal_bitmap =
{0,L11298};
const char L11298[] __attribute__((aligned(2)))
={0x00,0x00,0x20,0x00,0x7F,0x00,0x00,0x23,0x00,0x06,0x88,0x01,0x0
0,0x08,0xAB,0x01,0x00,0x0C,0xCE,0x01,0x00,0x0E,0x14,0x02,0x00,…
```

Использование картинок

IMAGE x2

NORMAL IMAGE

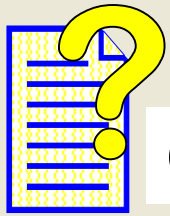
```
int main(void)
{
extern const BITMAP_FLASH image1;
BYTE stretch = IMAGE_NORMAL;
….
X = GetMaxX()-GetImageWidth((void*)&image1);
Y = GetMaxY()-GetImageHeight((void*)&image1);

//put bitmap in the center
PutImage((X >>1),(Y >> 1), stretch);
….
}
```

Опция растяжения stretch:
IMAGE_NORMAL
IMAGE_X2

Помощь по уровню примитивов

- Graphics Primitive Layer
  - Set Up Functions
    - ClearDevice Function
    - InitGraph Function
  - Text Functions
    - SetFont Function
    - OutChar Function
    - OutText Function
    - OutTextXY Function
    - GetTextHeight Function
    - GetTextWidth Function
    - XCHAR Macro
    - FONT_HEADER Structure
    - FONT_FLASH Structure
    - FONT_EXTERNAL Macro
  - Line Functions
  - Rectangle Functions
  - Circle Functions
  - Graphic Cursor
  - Bitmap Functions
    - GetImageHeight Function
    - GetImageWidth Function
    - PutImage Function
    - BITMAP_HEADER Structure
    - Bitmap Settings
    - Bitmap Source
  - External Memory

The Primitive Layer section of the help file describes the various APIs to help you with drawing primitives, using fonts, and using images.

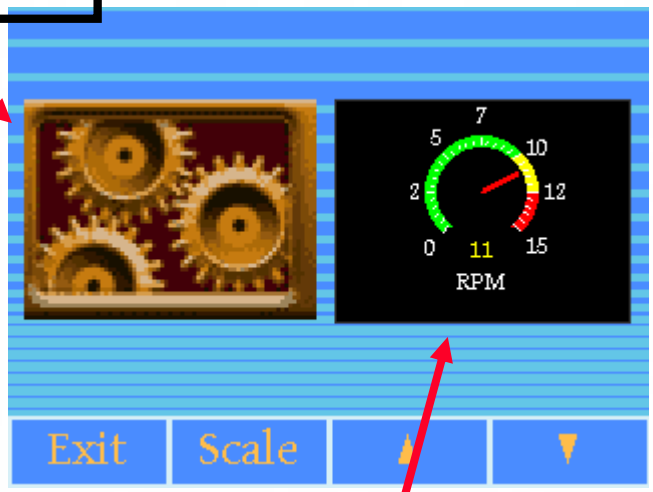


Graphics Library Help

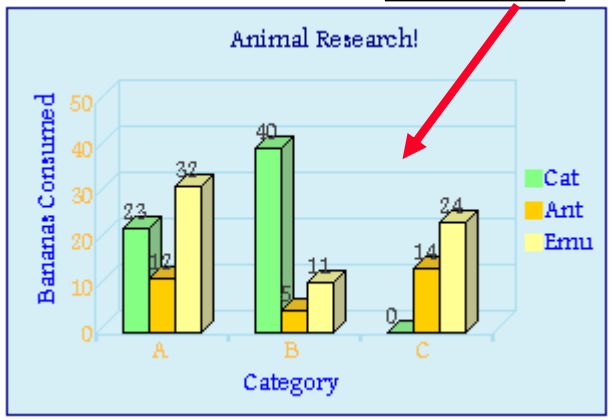
Элементы графической библиотеки Microchip

Элементы библиотеки

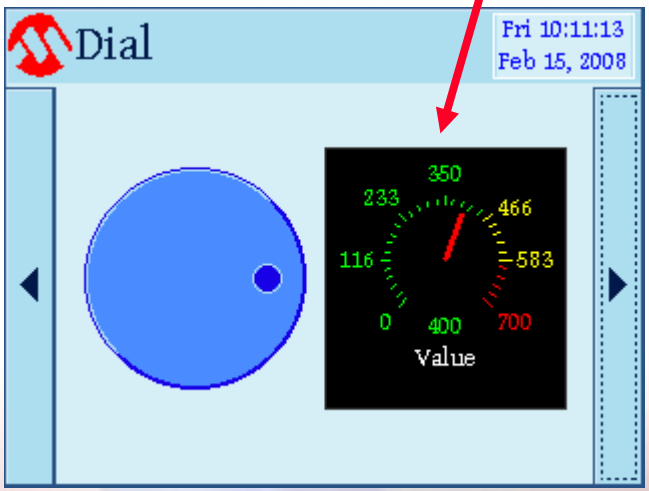
Picture

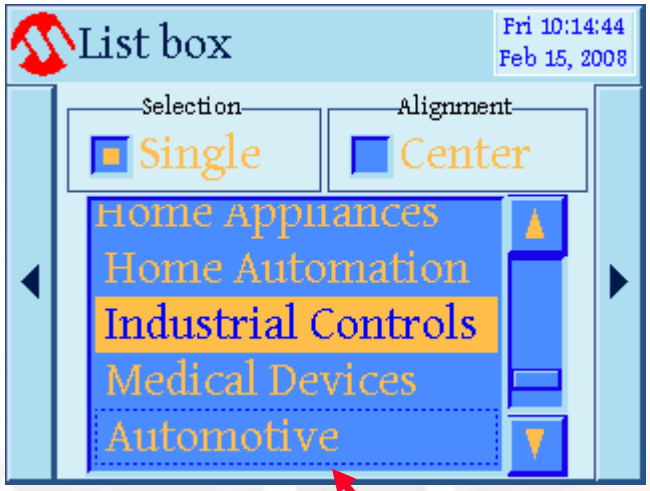


Chart



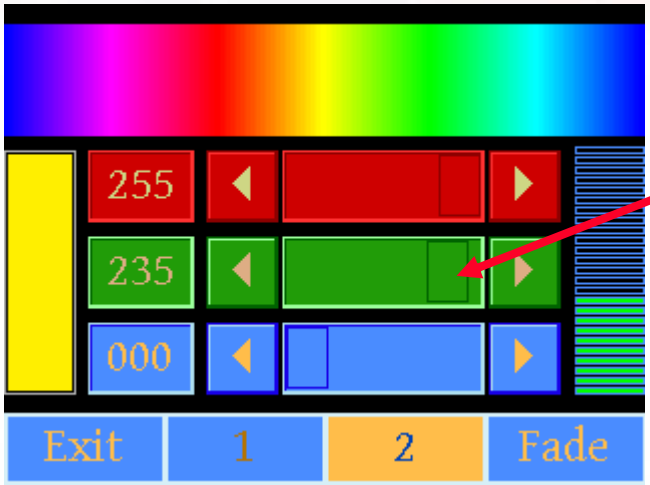
Meter





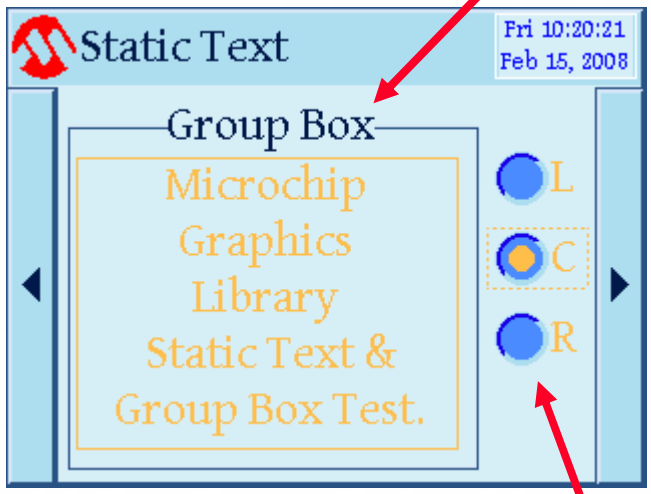
List Box

Элементы библиотеки

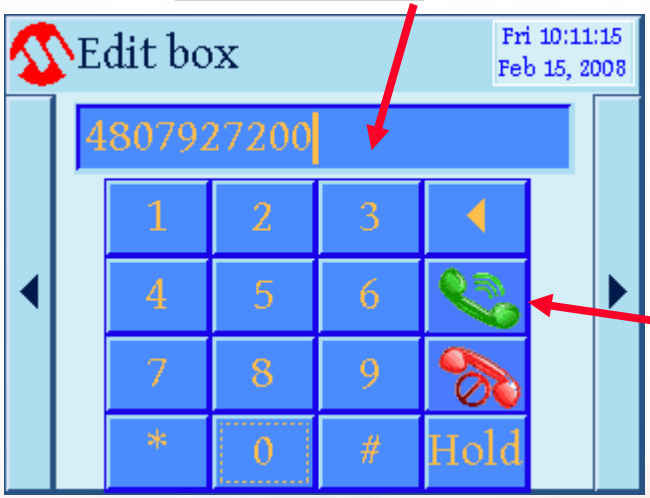


Slider

Group Box



Edit Box



Buttons w/ Image

Radio Buttons

Создание объектов

Определение:

`ObjCreate(,,)` - функция создания графических элементов со специфическими параметрами. Она автоматически заполняет структуру элемента, создает глобальный связный список и возвращает указатель на созданный элемент.

- **`Obj` = название объекта**
  - Определено в help'е
  - пример:
    - **`BtnCreate(,,,)` создание кнопок**
  - пример:
    - **`PbCreate(,,,)` создание индикатора выполнения (progress bar)**
- **Несколько элементов одного класса**

Общие параметры элементов

ID

- Уникальный идентификатор

Location (местоположение)

- left, top, bottom, right
  - **расположение и размеры**

State (состояние)

- Для управления элементом

Style Scheme (стиль)

- Определяет вид элемента

Уникальные параметры элементов

- **`BtnCreate(,,)` – кнопка**
  - ***`Radius`***: закругление углов
  - ***`*pBitmap`***: картинка на кнопке
  - ***`*pText`***: текст на кнопке
- **`StCreate(,,)` – статичный текст**
  - ***`*pText`***: текст
- **`PbCreate(,,)` – индикатор выполнения (progress bar)**
  - ***`pos`***: начальное значение
  - ***`range`***: максимальное значение

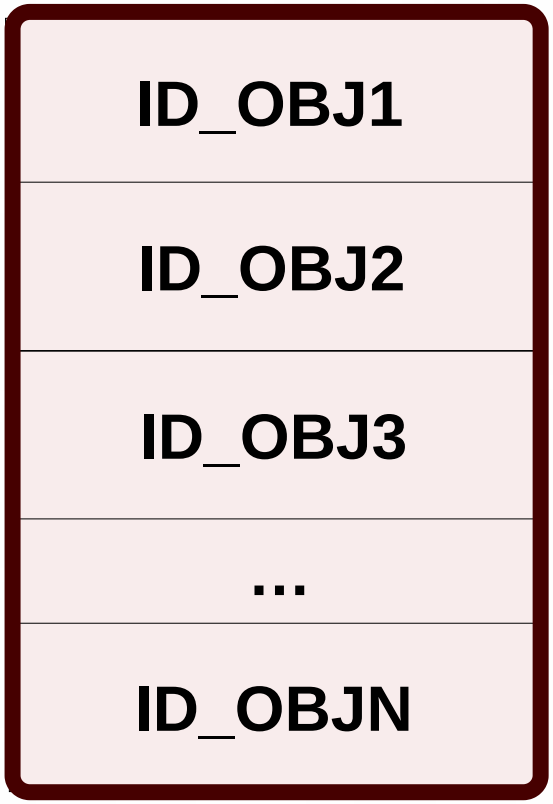
В коде программы …

- пример:

```
void CreateButtons(void)
{
…
BUTTON          *pBtn
#define          ID_BTN2      16
…
…
BtnCreate(    ID_BTN2,       // 2nd Button ID
              x3, y3,        // left, top
              x4, y4,        // right, bottom
              Radius,        // Rounded edges
              BTN_DRAW,      // Display button
              &arrow,        // use this bitmap
              NULL,          // no text
              altScheme);    // style scheme
…
}
```

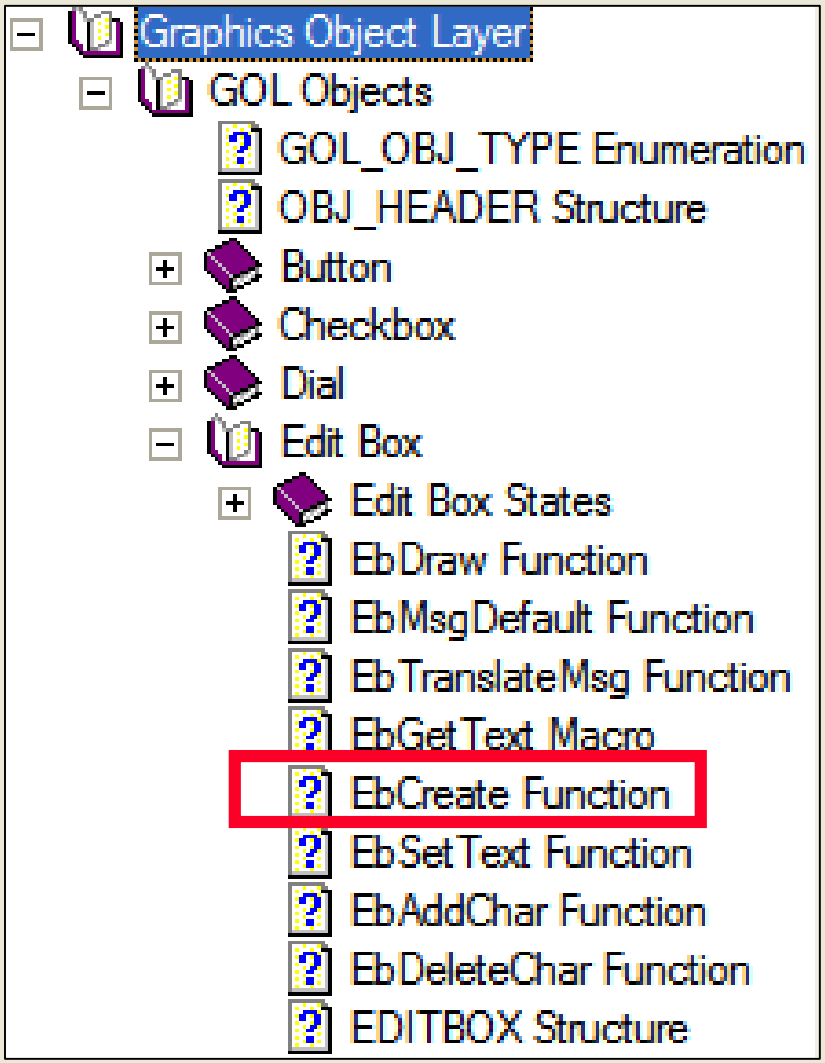
Создание элементов

Связный список

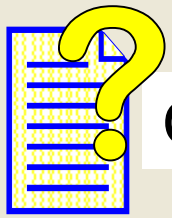


- `ObjCreate(,,,)`
  - заполните структуру элемента
  - Поместите объект в конец связного списка
  - Динамически выделяемая память
  - Подробности далее …

Помощь по созданию элементов



To find the ObjCreate APIs, expand the desired widget and select the appropriate create function.



Graphics Library Help

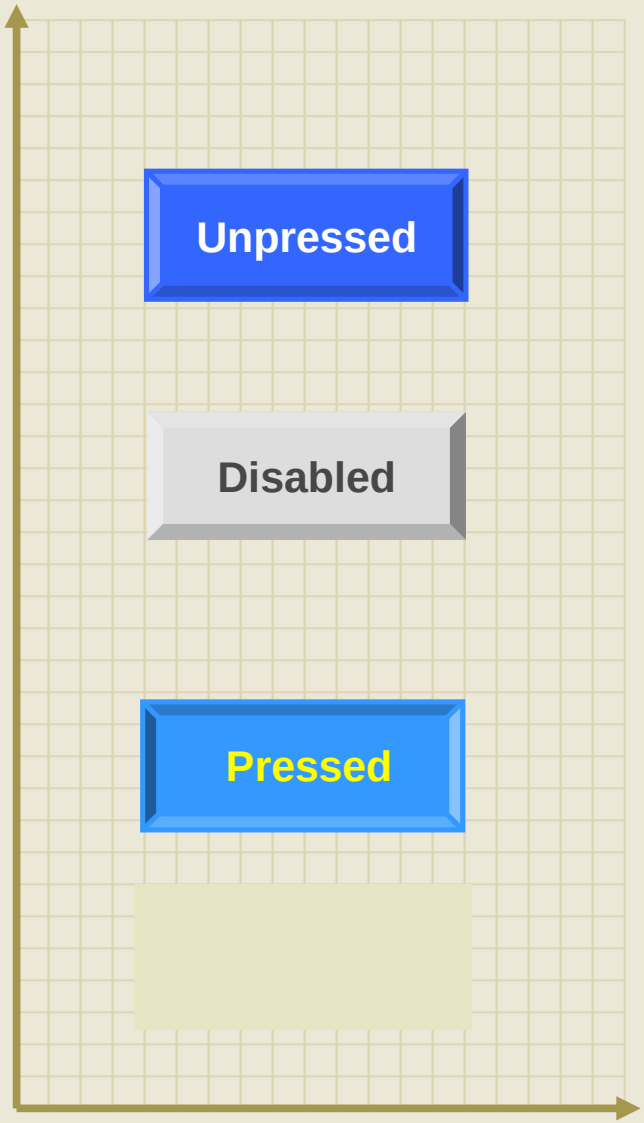
Стили элементов

Определение:

Стиль – это структура, определяющая цвета и шрифты, используемые при создании элемента.

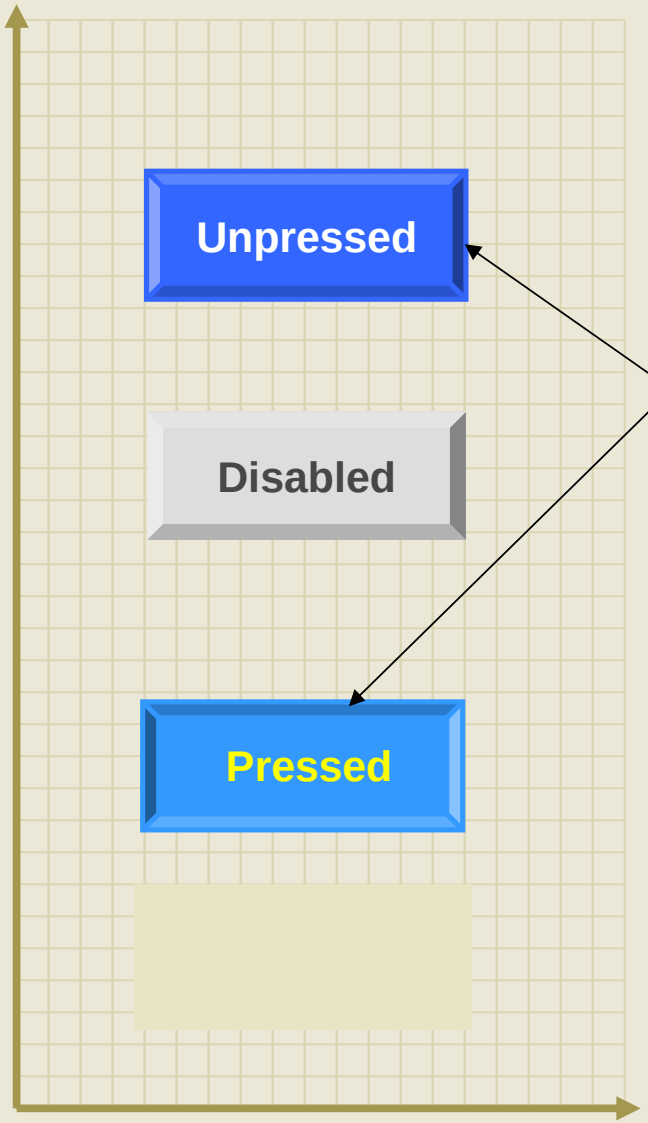
- **Многократное применение стилей**
- **Стандартные прописаны в файле `GOL.h`**
- **`GOLCreateScheme()`**
  - Создает структуру стиля
  - Возвращает указатель на стиль
- **10 составляющих стиля**
  - Стили по-разному выглядят на разных элементах
  - Изменение цвета по ходу программы меняет вид элемента

Структура стиля



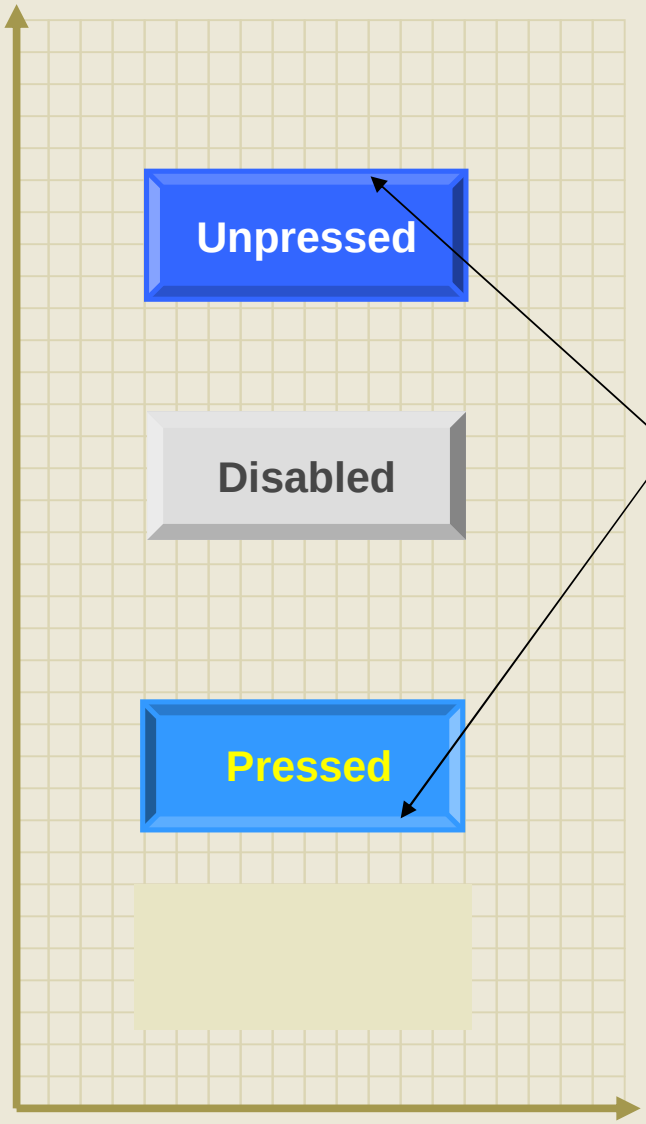

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = GREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTTAN;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Структура стиля



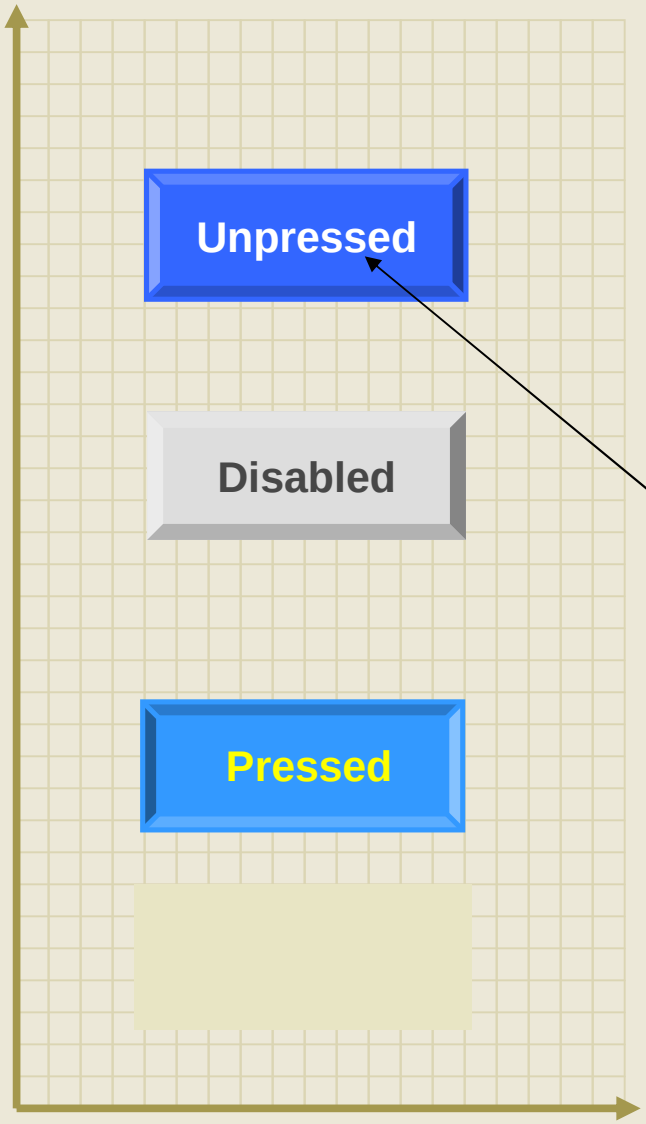

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = GREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTTAN;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



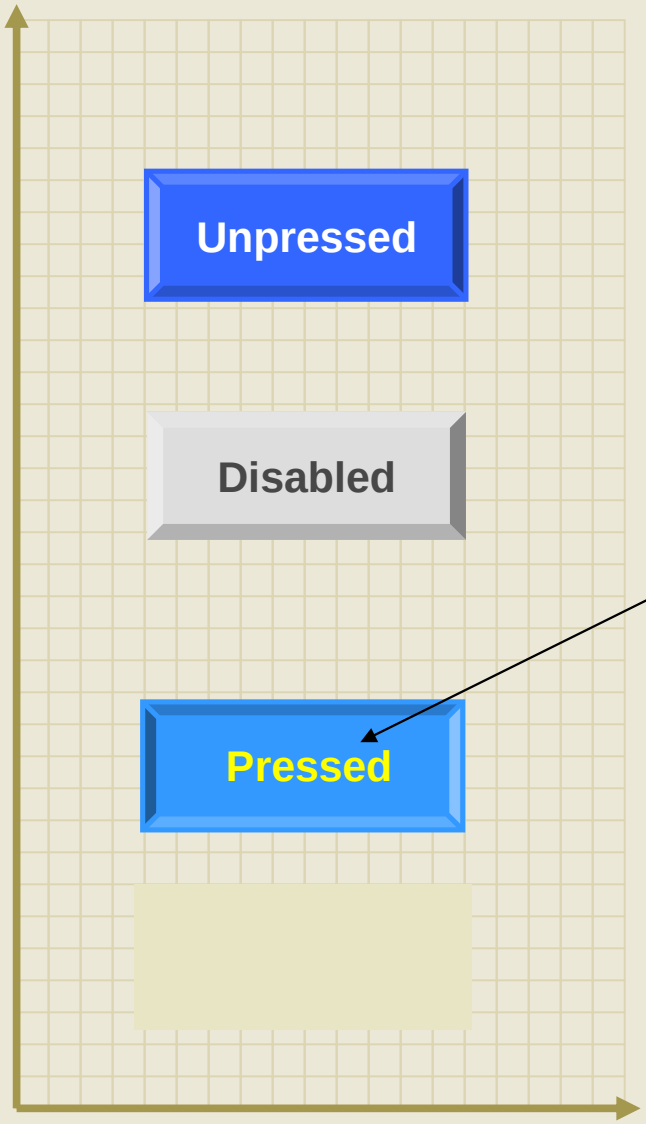

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = GREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTTAN;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



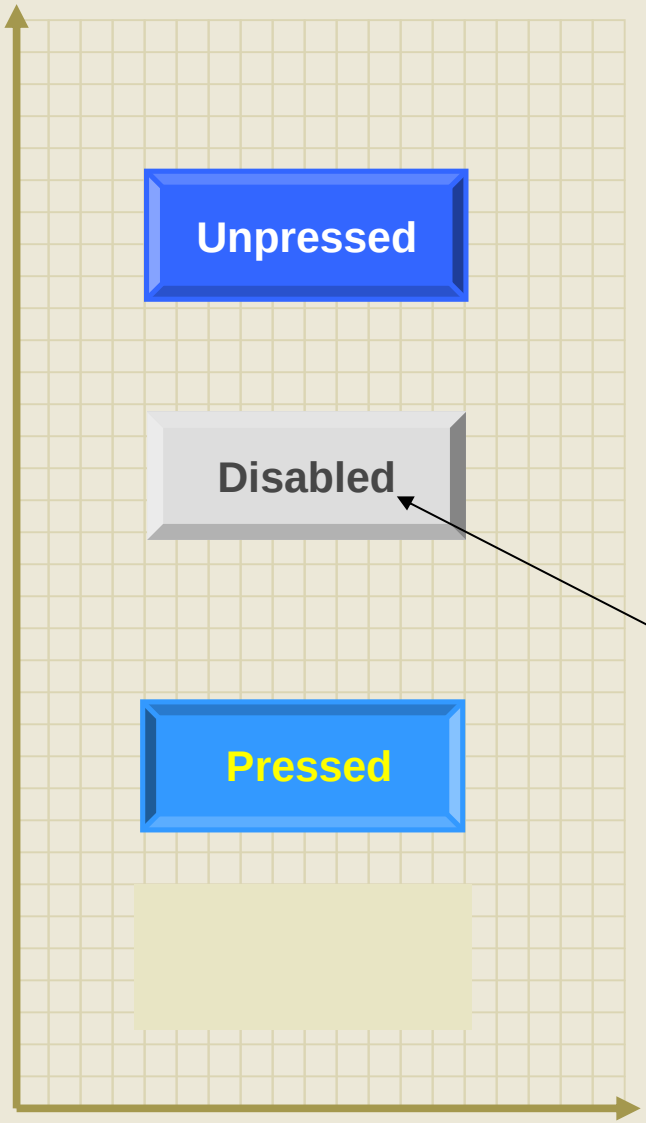

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = GREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTTAN;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



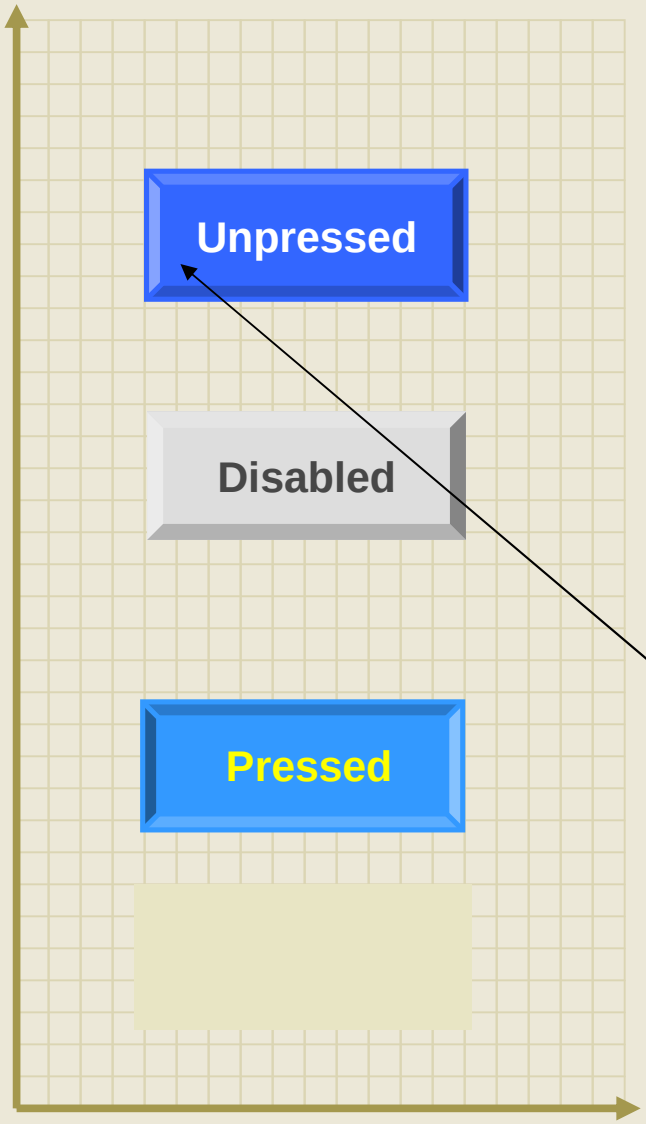

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = GREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTGREY;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



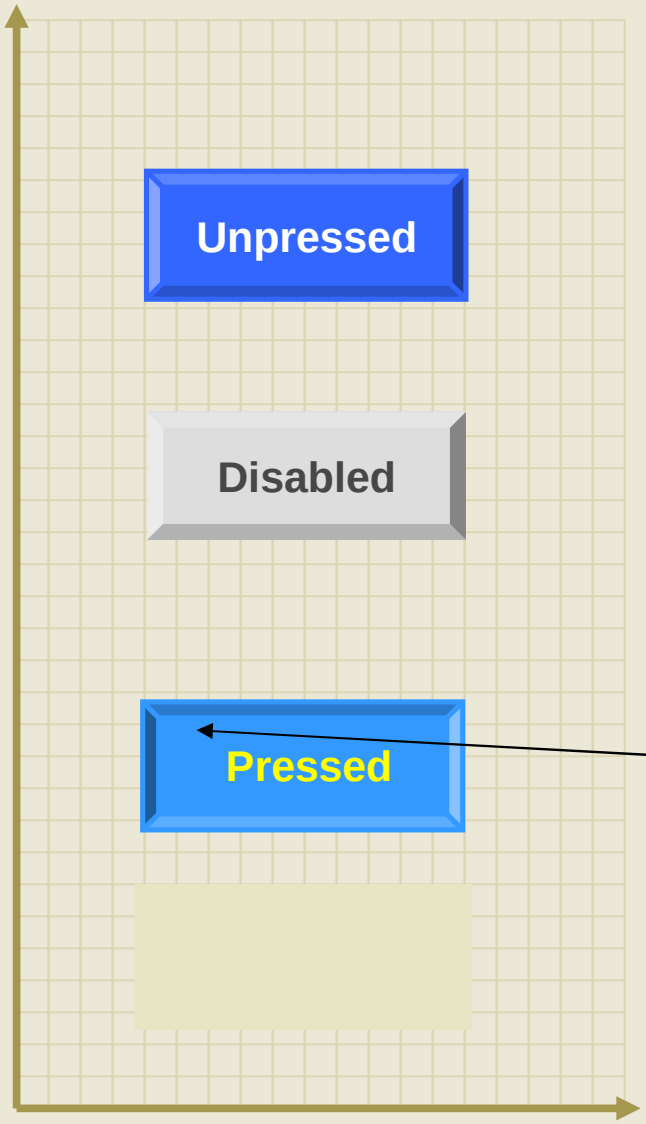

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = DKGREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTGREY;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



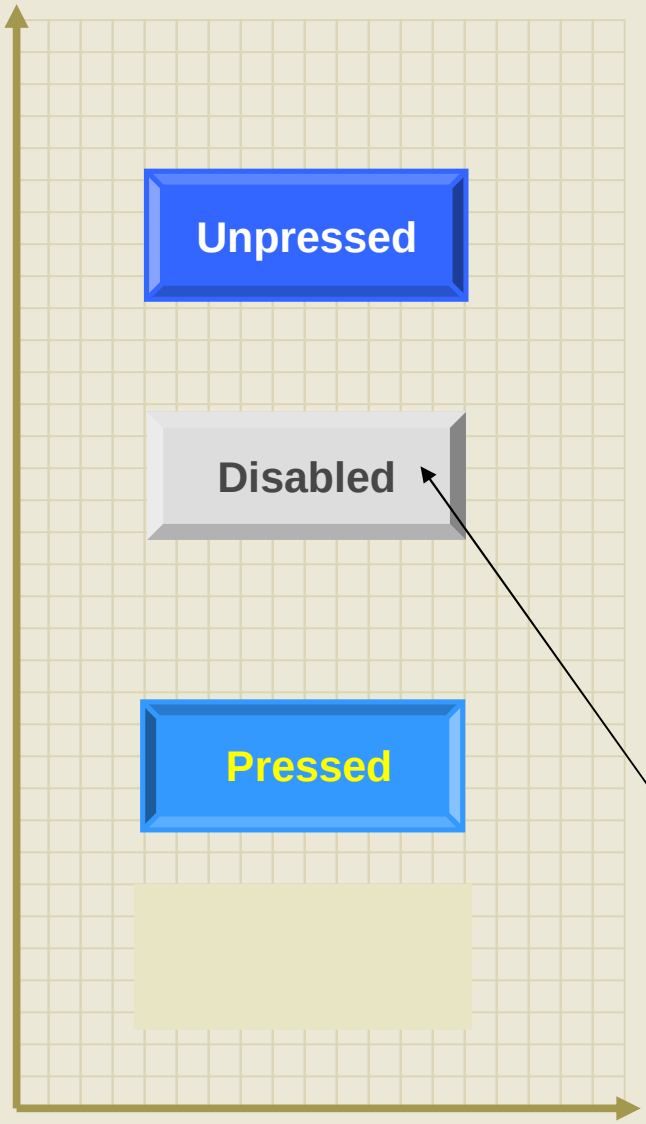

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = DKGREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTGREY;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



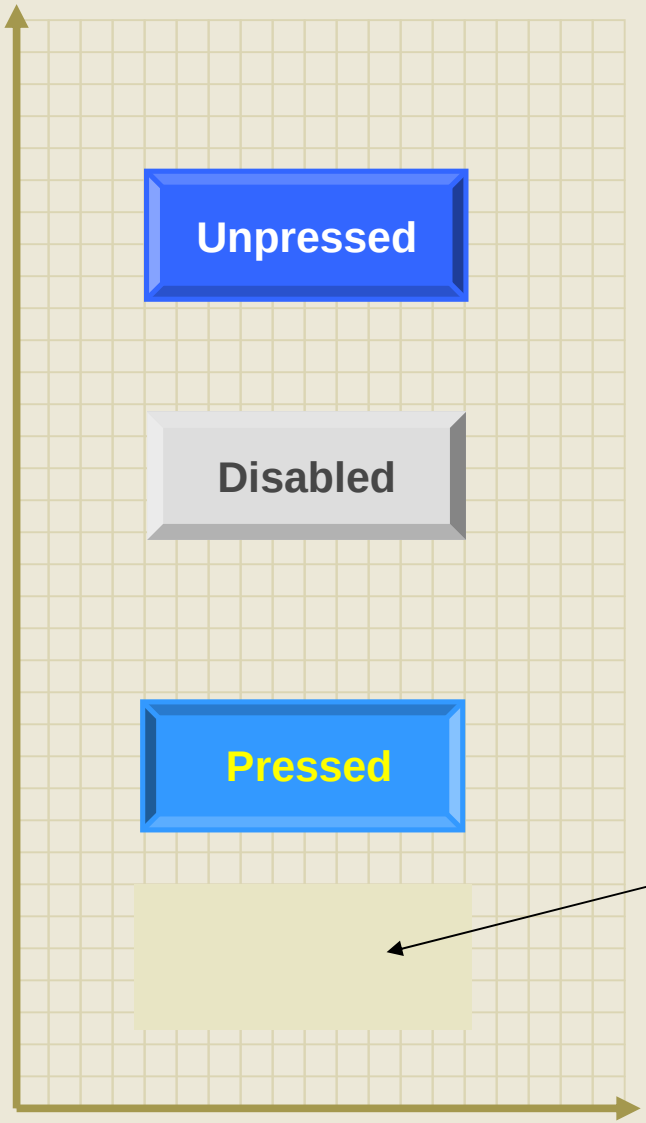

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = DKGREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTGREY;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



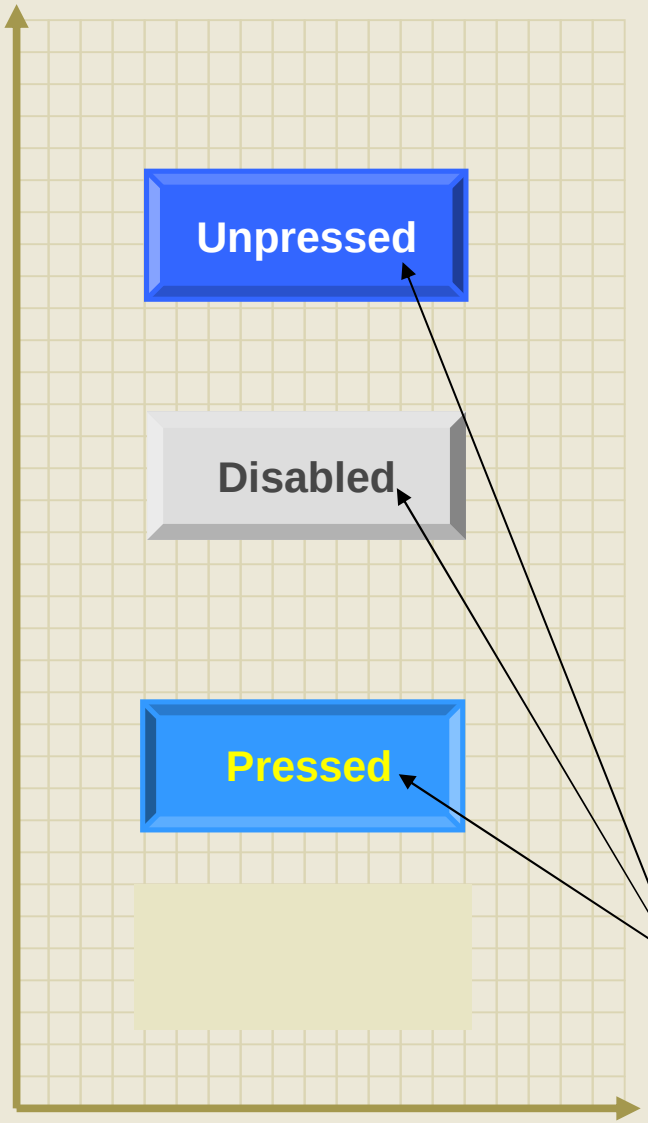

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = DKGREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTGREY;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей



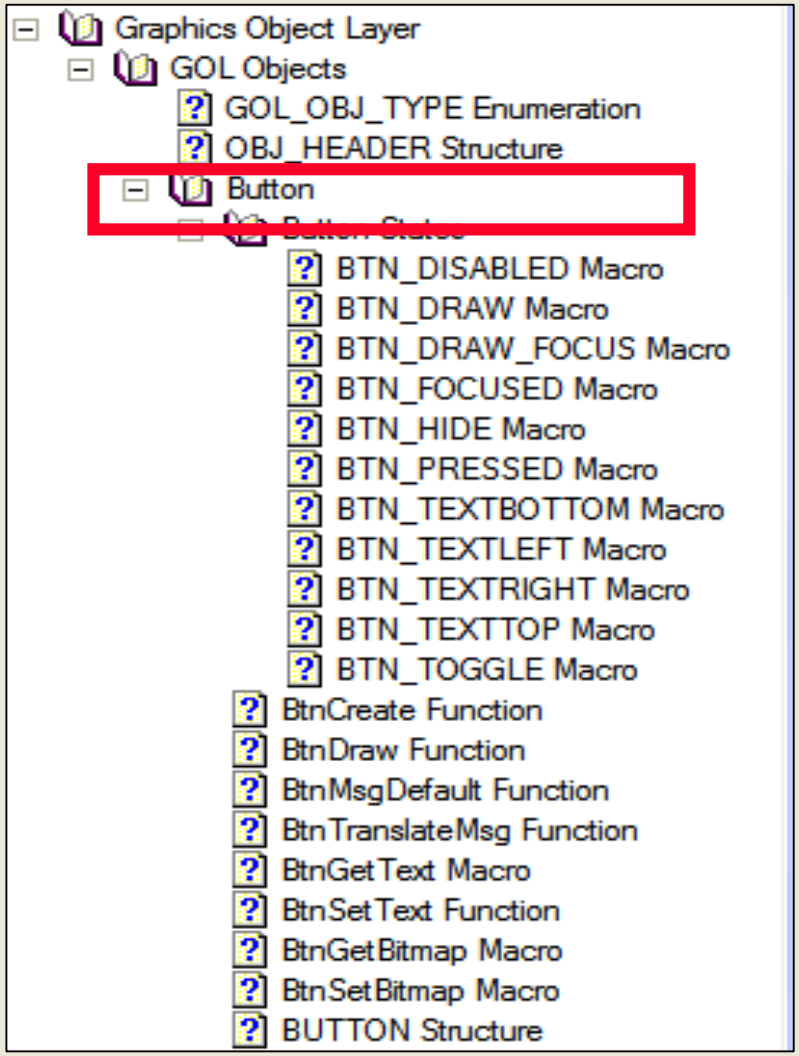

```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = DKGREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTGREY;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

Применение стилей

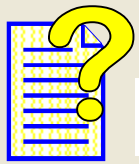



```
int main(void)
{
…
GOL_SCHEME     *altScheme;
altScheme = GOLCreateScheme();
…
altScheme -> EmbossDkColor = DKBLUE;
altScheme -> EmbossLtColor = BLUE;
altScheme -> TextColor0 = WHITE;
altScheme -> TextColor1 = YELLOW;
altScheme -> TextColorDisabled = DKGREY;
altScheme -> Color0 = BRIGHTBLUE;
altScheme -> Color1 = LTBLUE;
altScheme -> ColorDisabled = LTGREY;
altScheme -> CommonBkColor = TAN;
altScheme -> pFont = GOLFontDefault;
…
}
```

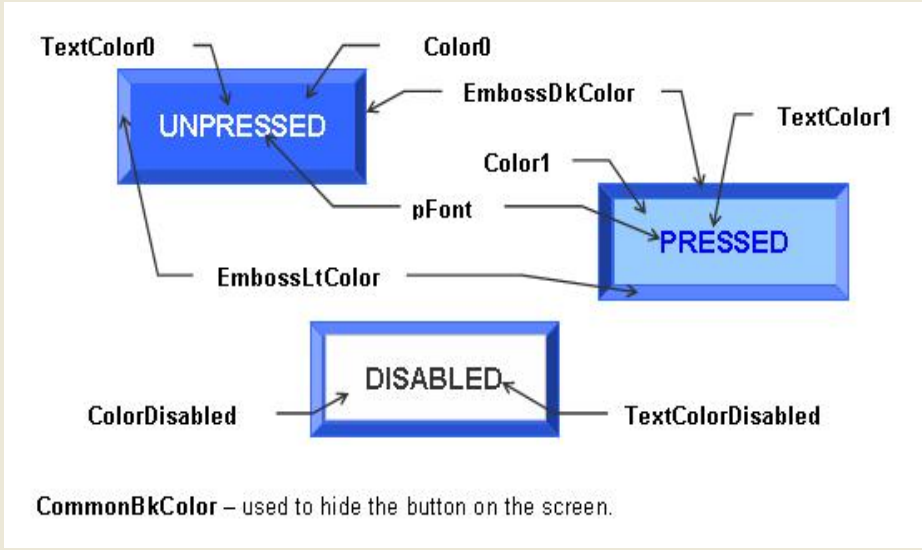
Помощь по стилям элементов



The top level of each widget gives a diagram showing how the style scheme fields are applied.



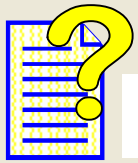
Graphics Library Help



API стилей

- Style Scheme
  - GOLCreateScheme Function
  - GOLSetScheme Macro
  - GOLGetScheme Macro
  - GOLGetSchemeDefault Macro
  - GOL_SCHEME Structure
  - Default Style Scheme Settings
    - COLOR0DEFAULT Macro
    - COLOR1DEFAULT Macro
    - COLORDISABLEDDEFAULT Macro
    - COMMONBACKGROUNDCOLORDEFAULT Macro
    - TEXTCOLOR0DEFAULT Macro
    - TEXTCOLOR1DEFAULT Macro
    - EMBOSSDKCOLORDEFAULT Macro
    - EMBOSSLTCOLORDEFAULT Macro
    - FONTDEFAULT Macro
    - TEXTCOLORDISABLEDDEFAULT Macro
  - GOLFontDefault Variable
  - GOL_EMBOSS_SIZE Macro
  - RGB565CONVERT Macro

Descriptions for other APIs that affect the style scheme are found in the Graphics Library help file.



Graphics Library Help

To change the style scheme of a widget after it is created, use …

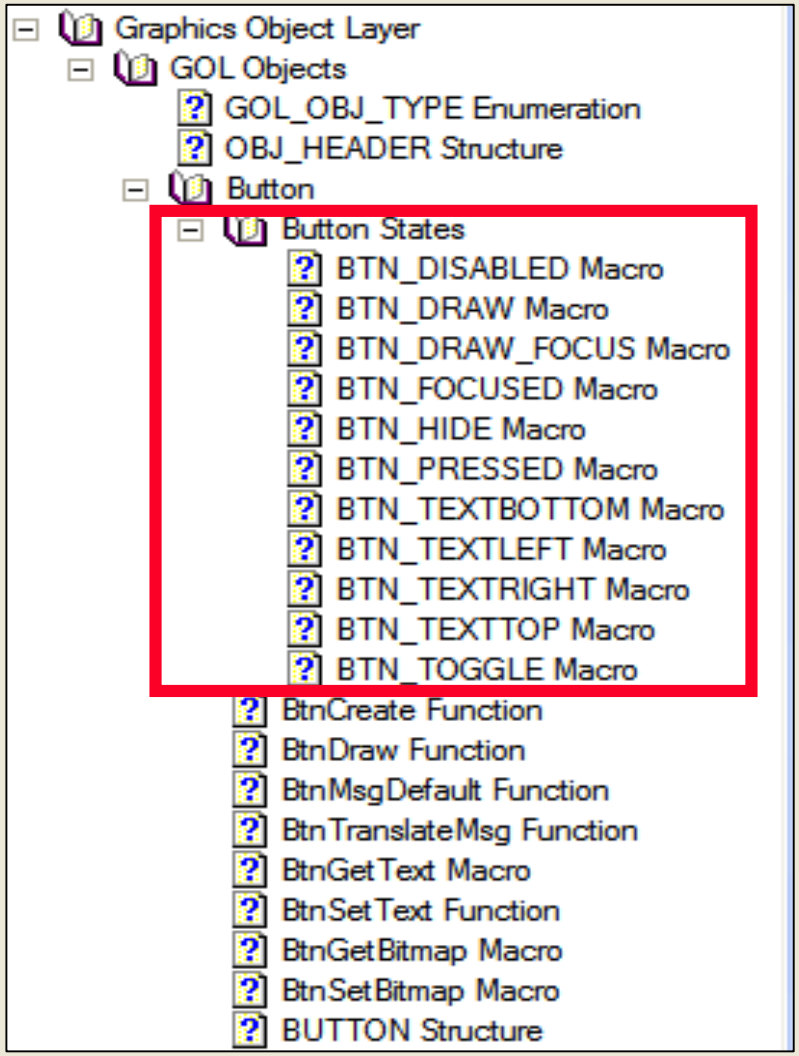
```
GOLSetScheme(*pObj, *pScheme)
```

Работа с графическими элементами на базе библиотеки Microchip

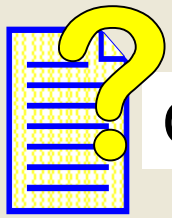
Поле `state` биты прорисовки

- **Часть структуры элемента**
- **Управляет состояниями элемента…**
  - **невидимый**
  - **Частичная перерисовка**
  - **Полная перерисовка**
- **Используется всеми элементами**
  - **`OBJ_HIDE`**: закрасить фоновым цветом CommonBkColor
  - **`OBJ_DRAW`**: перерисовать элемент
    - **сбрасывается функцией `GOLDraw()`**
  - **`OBJ_DRAW_FOCUS`**: перерисовать только активный элемент (в фокусе)
    - **Устанавливается функцией `GOLDraw()`**

помощь по полю state



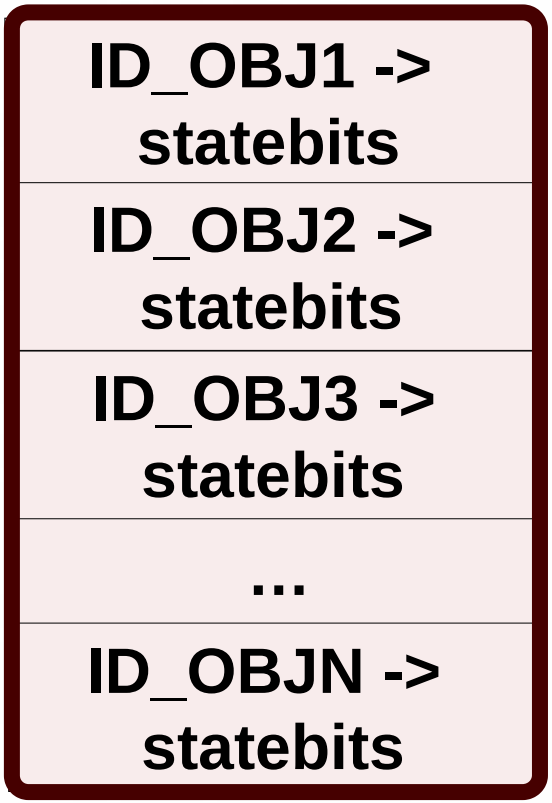
Every widget also has unique statebits. These can be found in the library help file as shown …



Graphics Library Help

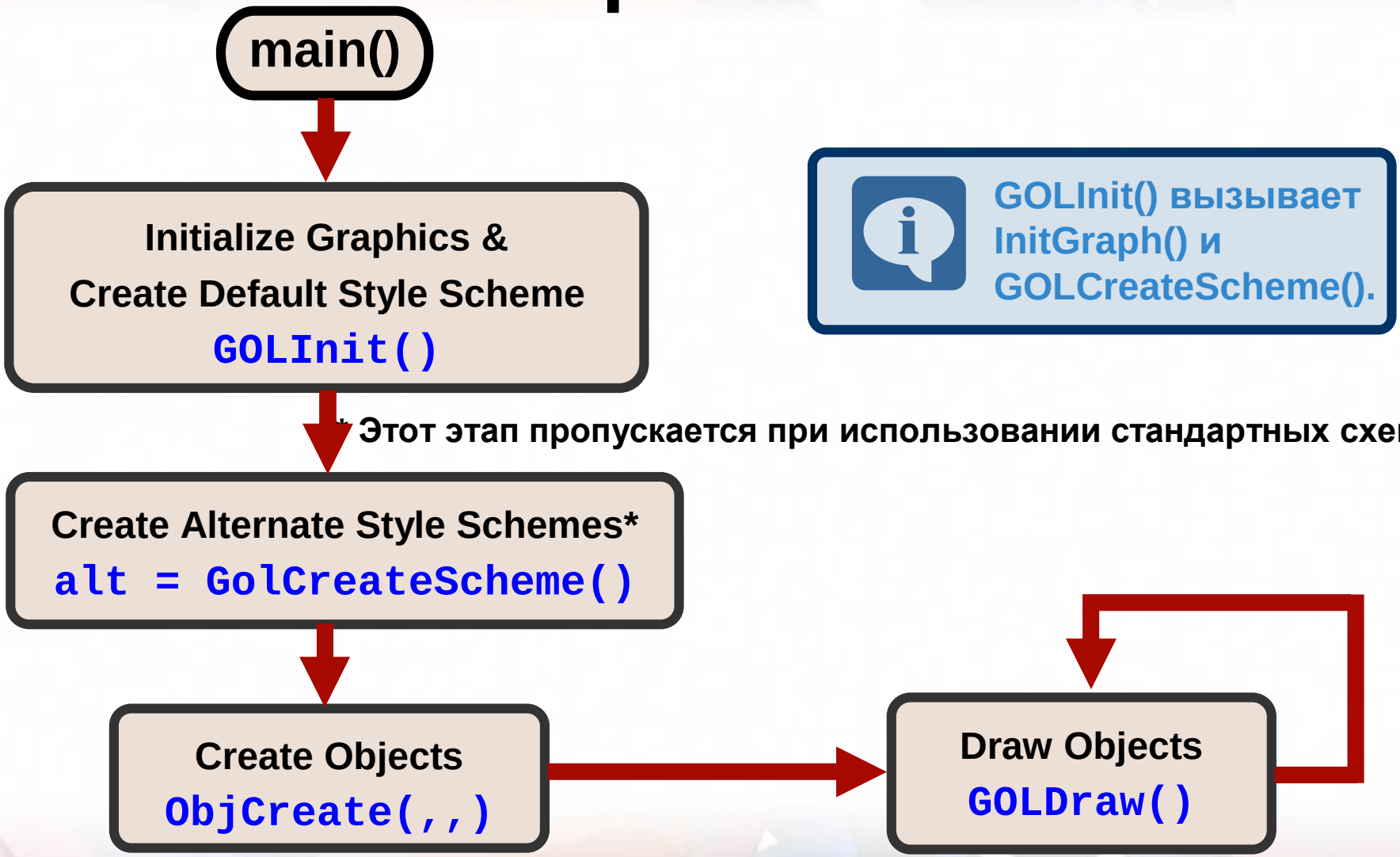
Работа с элементами

Связный список



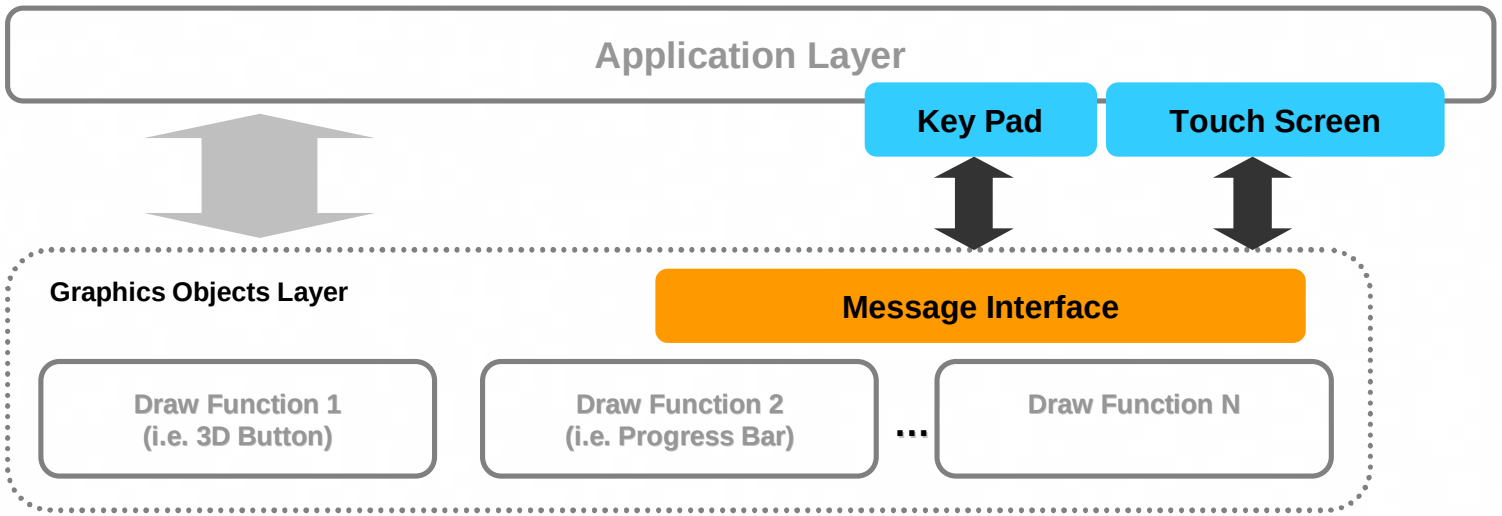
- `GOLDraw()`
  - Анализирует связный список
  - Проверка состояний каждого элемента
  - Если установлен бит прорисовки, то элемент будет перерисован
  - возврат TRUE по окончанию

Пример типичного алгоритма работы



Интерфейс передачи команд

Интерфейс передачи сообщений



- Упрощает интеграцию пользовательского управления
- Позволяет более эффективно работать с элементами
- Обеспечивает большую надежность и гибкость
- Поддержка других устройств управления в будущем (например, мышка)

Интерфейс передачи команд требования к программе

- **Обнаружение команд**
- **Полная структура сообщения**
- **Call `GOLMsg(&msg)`**
  - `&msg` – адрес структуры сообщения
- **Обеспечение внешнего вызова(обязательно)**
  - **`GOLMsgCallback(,,,)`**
    - Действие системы/элемента по единичному событию
    - Пример: нажатие клавиши включает светодиод
    - Вызов из **`GOLMsg()`**
  - **`GOLDrawCallback()`**
    - Действие системы/элемента по продолжительному событию
    - пример: нажатая кнопка разрешает изменение громкости
    - Вызов из **`GOLDraw()`**

Структура команды

- **Структура команды**

```
typedef struct {
    BYTE                type;
    BYTE                uiEvent;
    SHORT               param1;
    SHORT               param2;
    } GOL_MSG;
```

- **`type` = *TYPE_KEYBOARD* или *TYPE_TOUCHSCREEN***
- **`param1 and param2` зависят от события и типа команды**
  - Для сенсорного управления:
    - **`param1`: значение x**
    - **`param2`: значение y**
  - Для клавиатуры:
    - **`param1`: ID выбранного элемента**
    - **`param2`: зависит от элемента и события**

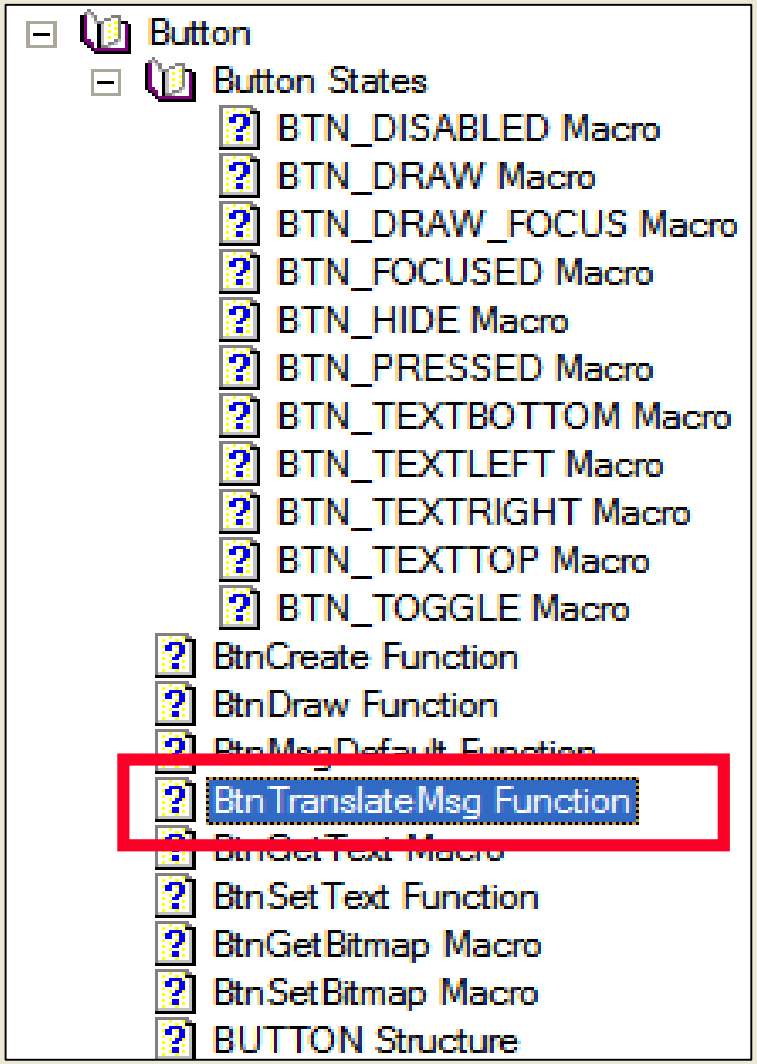
Структура команды

- **`uiEvent` определяет событие**
  - `uiEvent` значения при тач-управлении
    - **`EVENT_PRESS`**
    - **`EVENT_RELEASE`**
    - **`EVENT_MOVE`**
    - **`EVENT_INVALID`**
      - Касание не влияет на элемент
  - `uiEvent` значения при командах с клавиш
    - **`EVENT_KEYSCAN`**
      - **`param2`** значение, зависящее от элемента
      - Обычно скан-код
    - **`EVENT_CHARCODE` (только для «edit box»)**
      - **`param2`** = новый вводимый символ
    - **`EVENT_INVALID`**

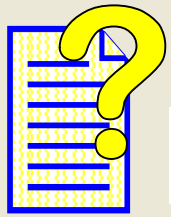
Пример
создания команды от клавиши

```
if(S5){
        msg->type    = TYPE_KEYBOARD;
        msg->uiEvent = EVENT_KEYSCAN;
        msg->param1  = obj->ID;
        msg->param2  = SCAN_CR_PRESSED;
}else{
        msg->type    = TYPE_KEYBOARD;
        msg->uiEvent = EVENT_KEYSCAN;
        msg->param1  = obj->ID;
        msg->param2  = SCAN_CR_RELEASED;
}return;
```

Помощь по командам элементов управления



A table describing the valid input sources, events, and default behaviors can be found in the `ObjTranslateMsg` function description for each widget



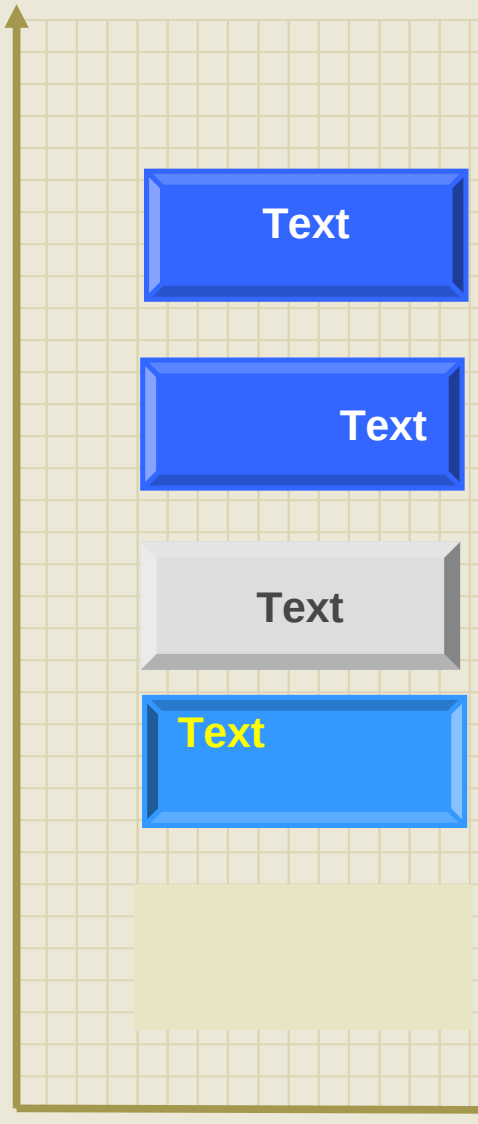
Graphics Library Help

Обработка команд элементов управления

Поле `state`

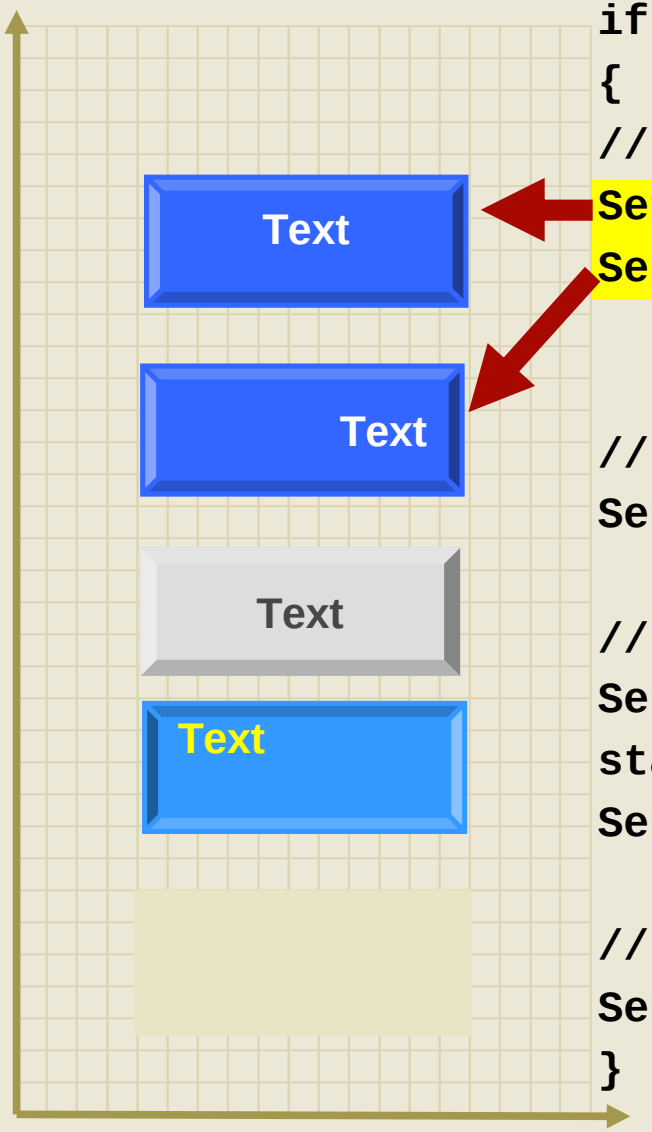
- **Определите действие и внешний вид**
  - Нет непосредственного влияния от `GOLDraw()`
- **НЕКОТОРЫЕ элементы:**
  - `OBJ_FOCUSED`: элемент в фокусе
- **ВСЕ элементы:**
  - `OBJ_DISABLED`: элемент не активен
    - **Все команды будут игнорированы**

Состояния кнопок



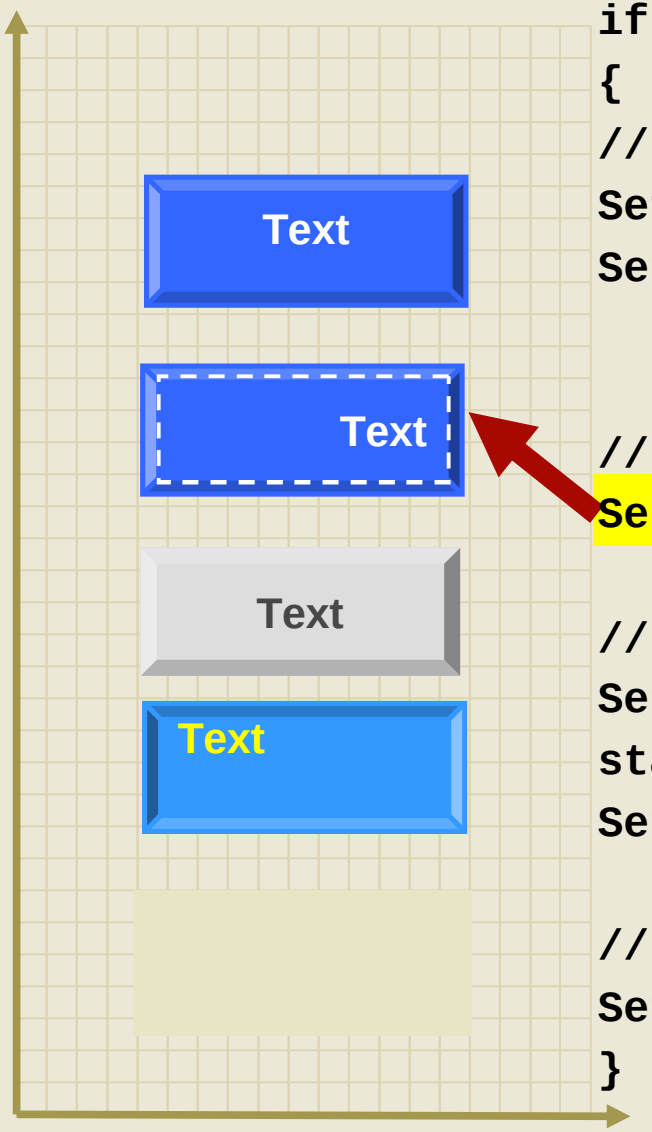

```
if (GOL_DRAW())
{
// примеры выравнивания текста в кнопках
SetState(pBtn, BTN_TEXTTOP);
SetState(pBtn, BTN_TEXTRIGHT);


// пример фокусировки на кнопке
SetState(pBtn, BTN_FOCUSED);

// пример действия кнопки
SetState(pBtn, BTN_DISABLED);
state = BTN_PRESSED|BTN_TEXTTOP|BTN_TEXTRIGHT;
SetState(pBtn, state);

// пример скрытия кнопки
SetState(pBtn, BTN_HIDE)
}
```

Состояния кнопок



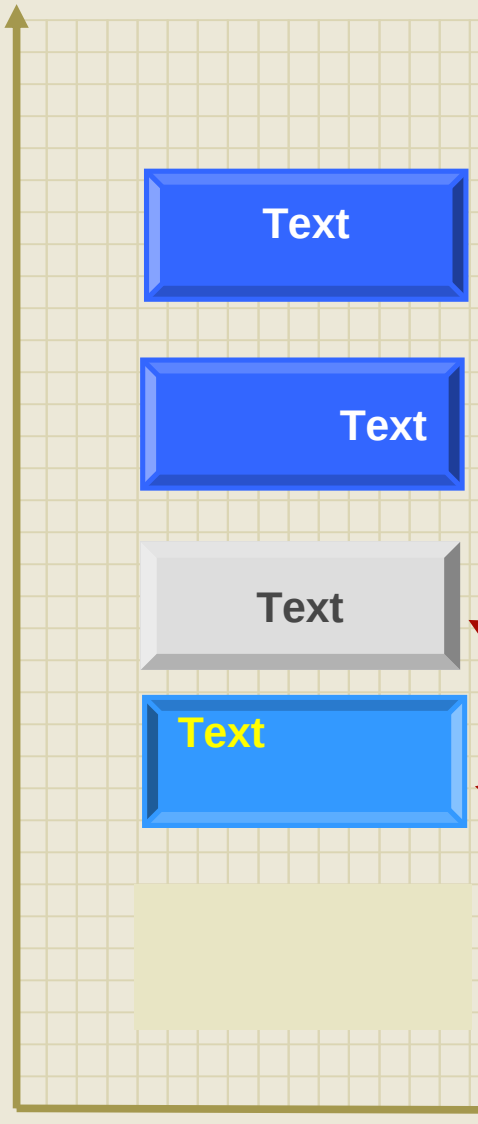

```
if (GOL_DRAW())
{
// примеры выравнивания текста в кнопках
SetState(pBtn, BTN_TEXTTOP);
SetState(pBtn, BTN_TEXTRIGHT);


// пример фокусировки на кнопке
SetState(pBtn, BTN_FOCUSED);

// пример действия кнопки
SetState(pBtn, BTN_DISABLED);
state = BTN_PRESSED|BTN_TEXTTOP|BTN_TEXTRIGHT;
SetState(pBtn, state);

// пример скрытия кнопки
SetState(pBtn, BTN_HIDE)
}
```

Состояния кнопок



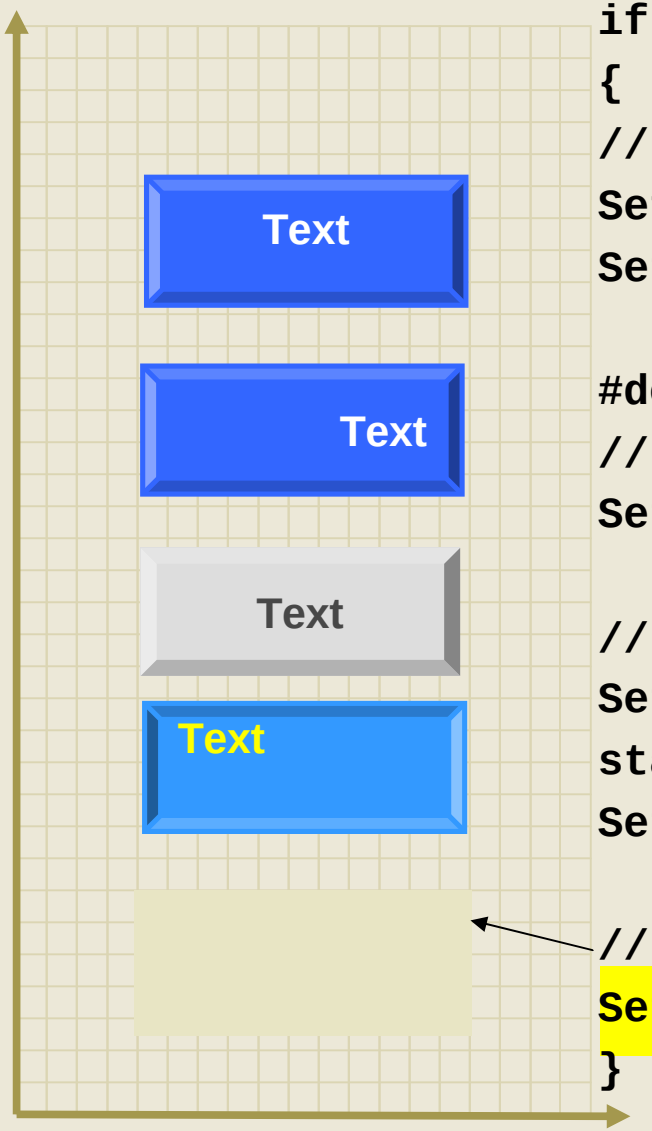

```
if (GOL_DRAW())
{
// примеры выравнивания текста в кнопках
SetState(pBtn, BTN_TEXTTOP);
SetState(pBtn, BTN_TEXTRIGHT);


// пример фокусировки на кнопке
SetState(pBtn, BTN_FOCUSED);

// пример действия кнопки
SetState(pBtn, BTN_DISABLED);
state = BTN_PRESSED|BTN_TEXTTOP|BTN_TEXTRIGHT;
SetState(pBtn, state);

// пример скрытия кнопки
SetState(pBtn, BTN_HIDE)
}
```

Button States



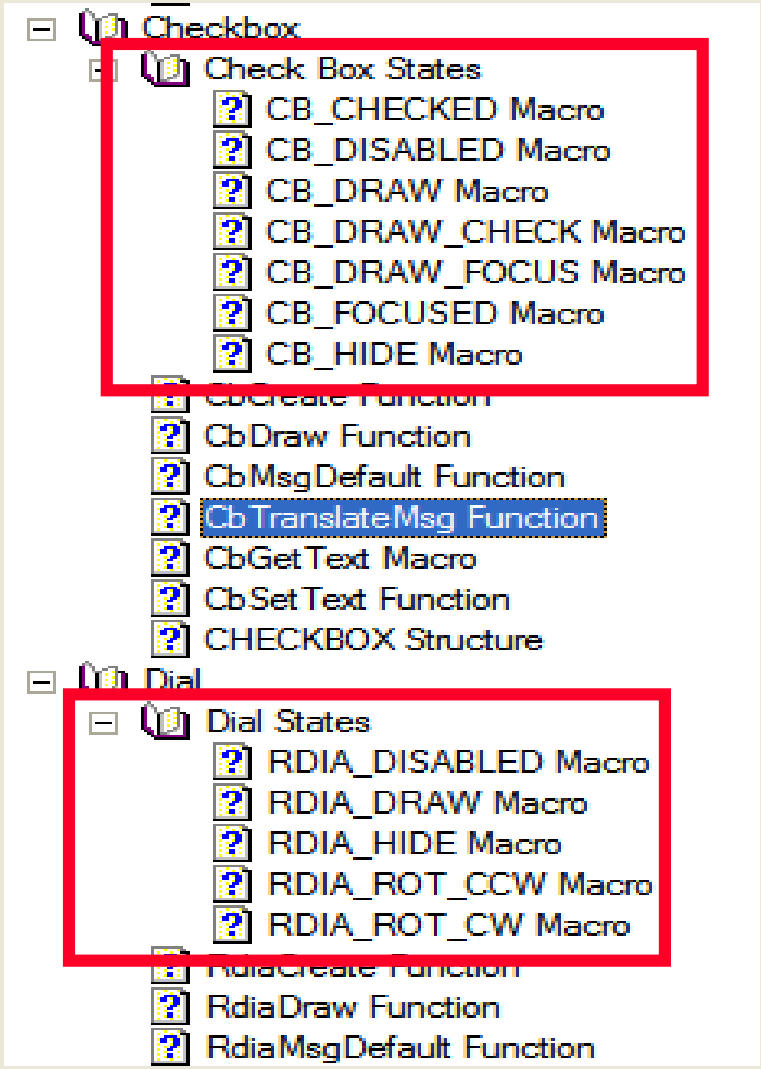

```
if (GOL_DRAW())
{
// примеры выравнивания текста в кнопках
SetState(pBtn, BTN_TEXTTOP);
SetState(pBtn, BTN_TEXTRIGHT);


// пример фокусировки на кнопке
SetState(pBtn, BTN_FOCUSED);

// пример действия кнопки
SetState(pBtn, BTN_DISABLED);
state = BTN_PRESSED|BTN_TEXTTOP|BTN_TEXTRIGHT;
SetState(pBtn, state);

// пример скрытия кнопки
SetState(pBtn, BTN_HIDE)
}
```

Button States

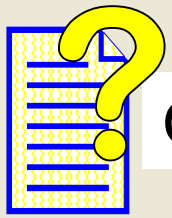



```
if (GOL_DRAW())
{
// примеры выравнивания текста в кнопках
SetState(pBtn, BTN_TEXTTOP);
SetState(pBtn, BTN_TEXTRIGHT);

#define USE_FOCUS
// пример фокусировки на кнопке
SetState(pBtn, BTN_FOCUSED);

// пример действия кнопки
SetState(pBtn, BTN_DISABLED);
state = BTN_PRESSED|BTN_TEXTTOP|BTN_TEXTRIGHT;
SetState(pBtn, state);

// пример скрытия кнопки
SetState(pBtn, BTN_HIDE);
}
```

ПОМОЩЬ С СОСТОЯНИЯМИ



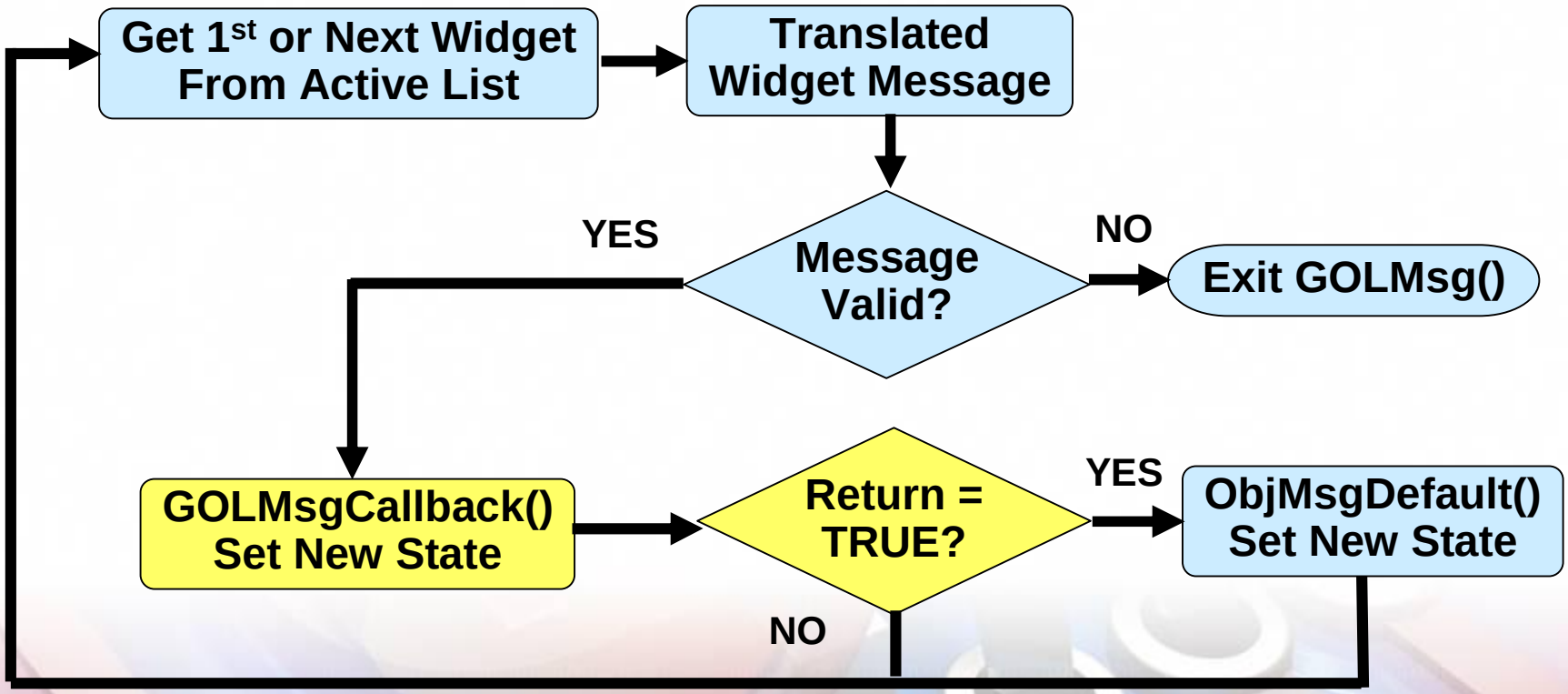
Statebit descriptions for all the widgets are located in the Graphics library help file.



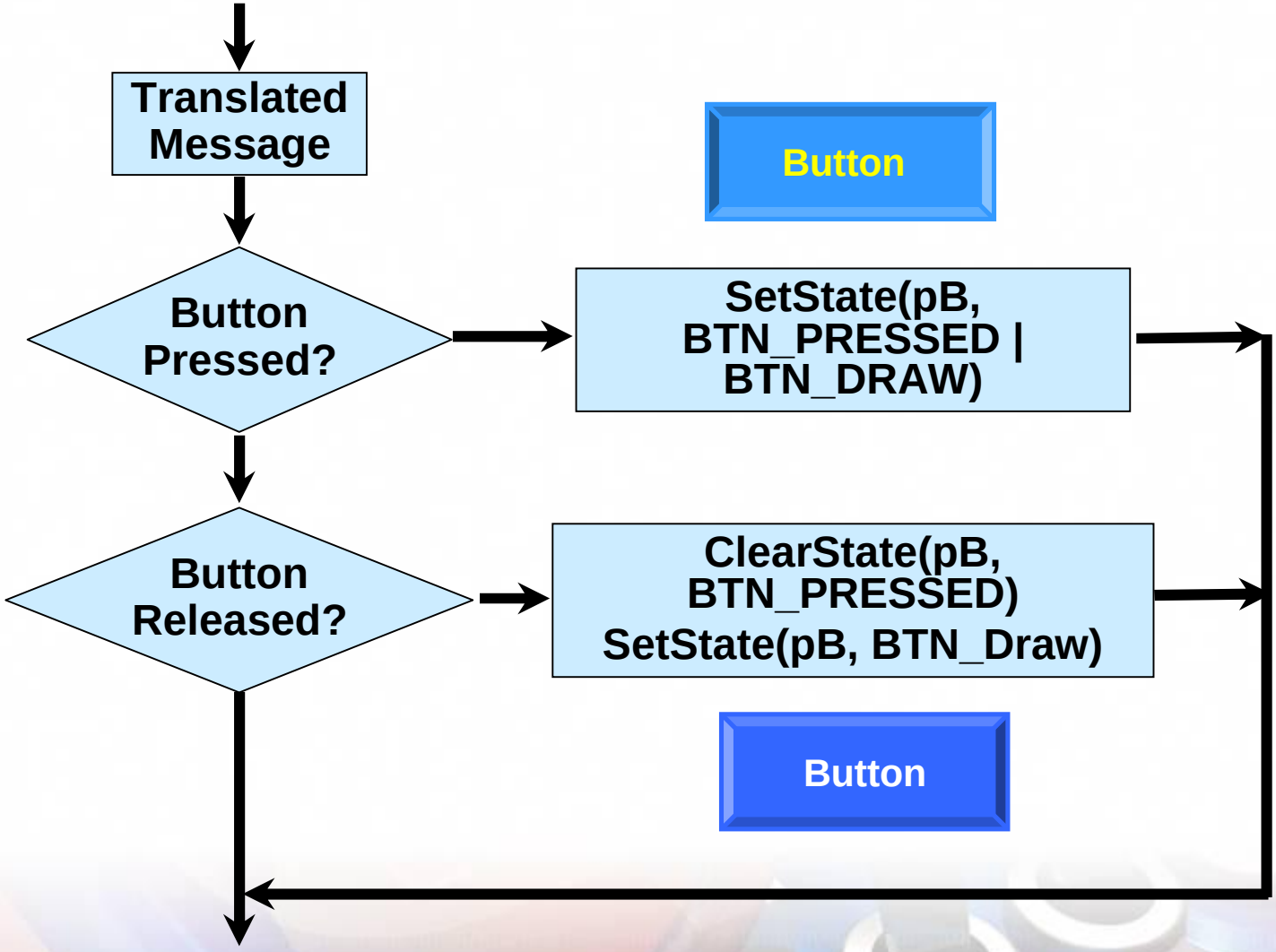
Graphics Library Help

GOLMsg(&msg)

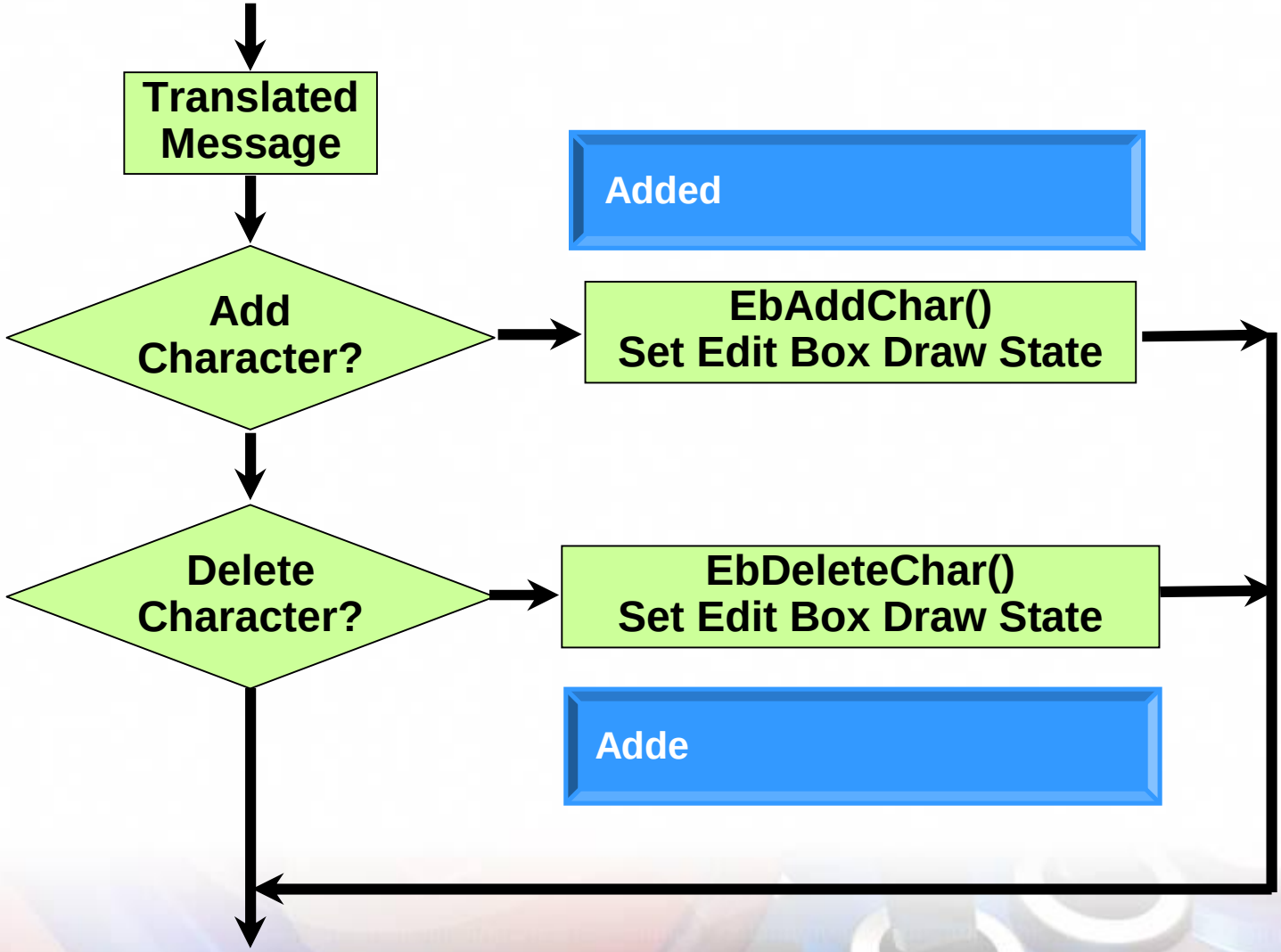
- Вызов в основном цикле программы
- НЕ вызывать, пока `GOLDraw()` не выполнено
  - `if (GOLDraw()) GOLMsg(&msg);`



типичная обработка кнопок



Типичная обработка «Edit Box»

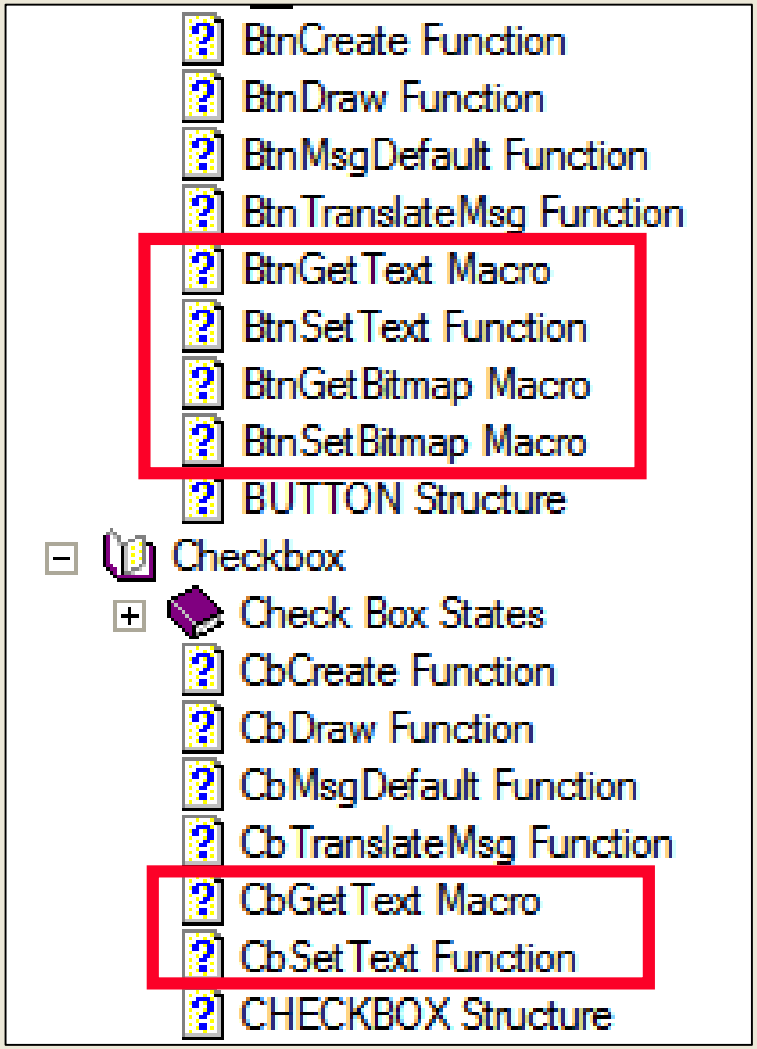


TIP

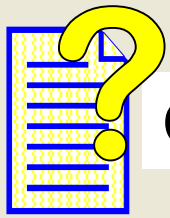
- Ищем значения по умолчанию `ObjMsgDefault()` в GOL.c

```
void RbMsgDefault(WORD translatedMsg, RADIOBUTTON* pRb, GOL_MSG* pMsg)
{
…
    if(translatedMsg == RB_MSG_CHECKED){

        // снять выделение со всех кнопок с зависимой фиксацией
        pointer = (RADIOBUTTON*) pRb->pHead;

        while(pointer != NULL){
            if(GetState(pointer,RB_CHECKED)){
               ClrState(pointer, RB_CHECKED);     // снять выделение
               SetState(pointer, RB_DRAW_CHECK);  // перерисовать
            }
            pointer = (RADIOBUTTON*)pointer->pNext;
        }
       //установить выделение и перерисовать
        SetState(pRb, RB_CHECKED|RB_DRAW_CHECK);
    }
}
```

Помощь по API элементов

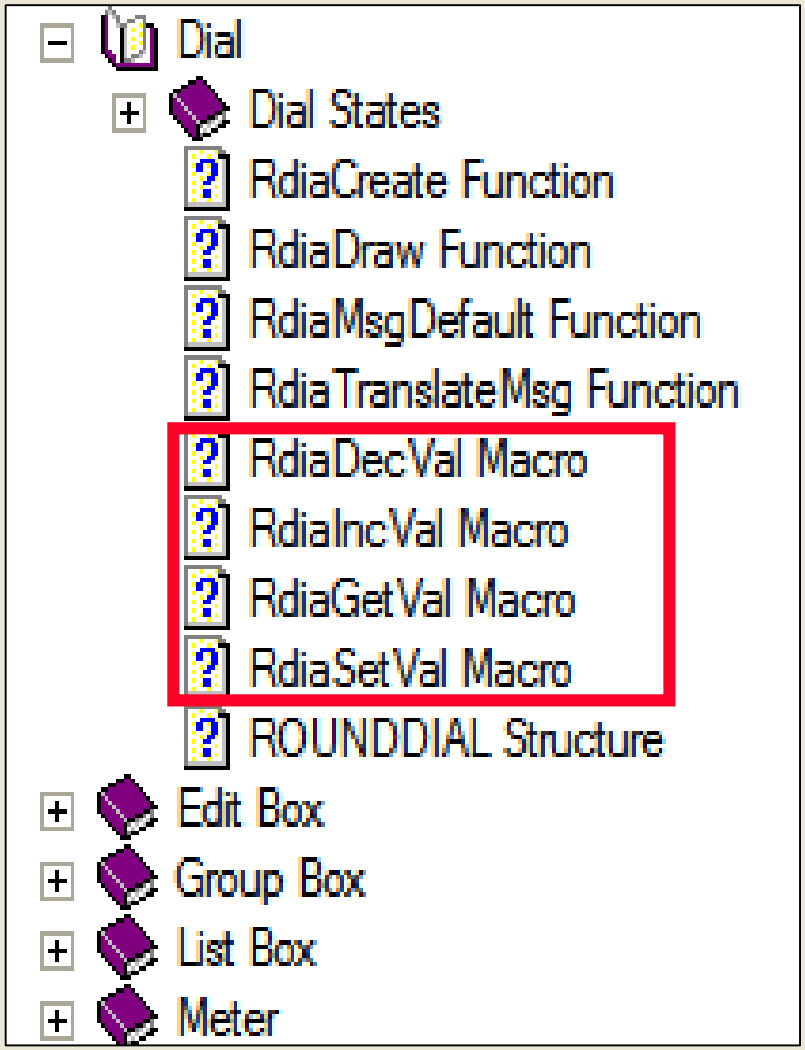


The widget APIs are found under the individual widgets in the Graphics Object Layer section of the help file.

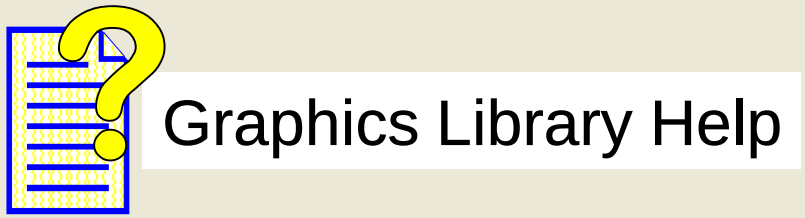


Graphics Library Help

Помощь по API элементов



The widget APIs are found under the individual widgets in the Graphics Object Layer section of the help file.

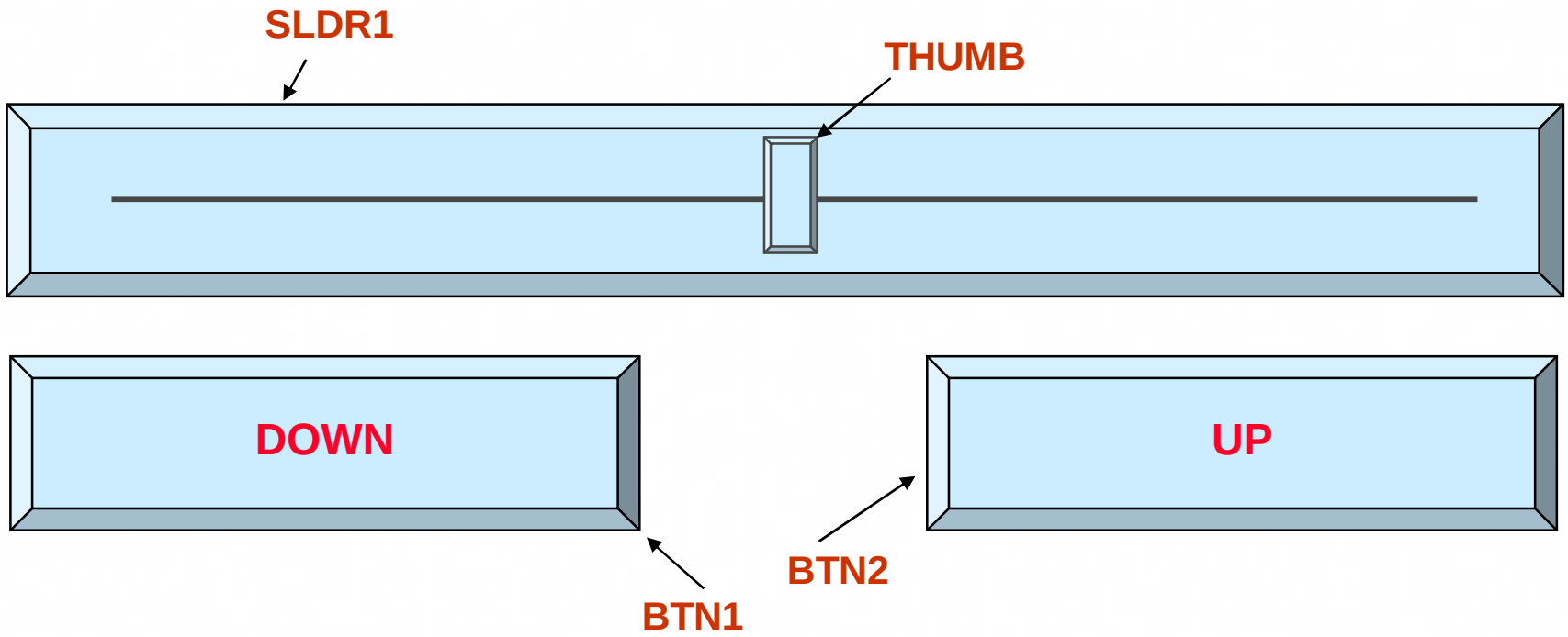


GOLMsgCallback()

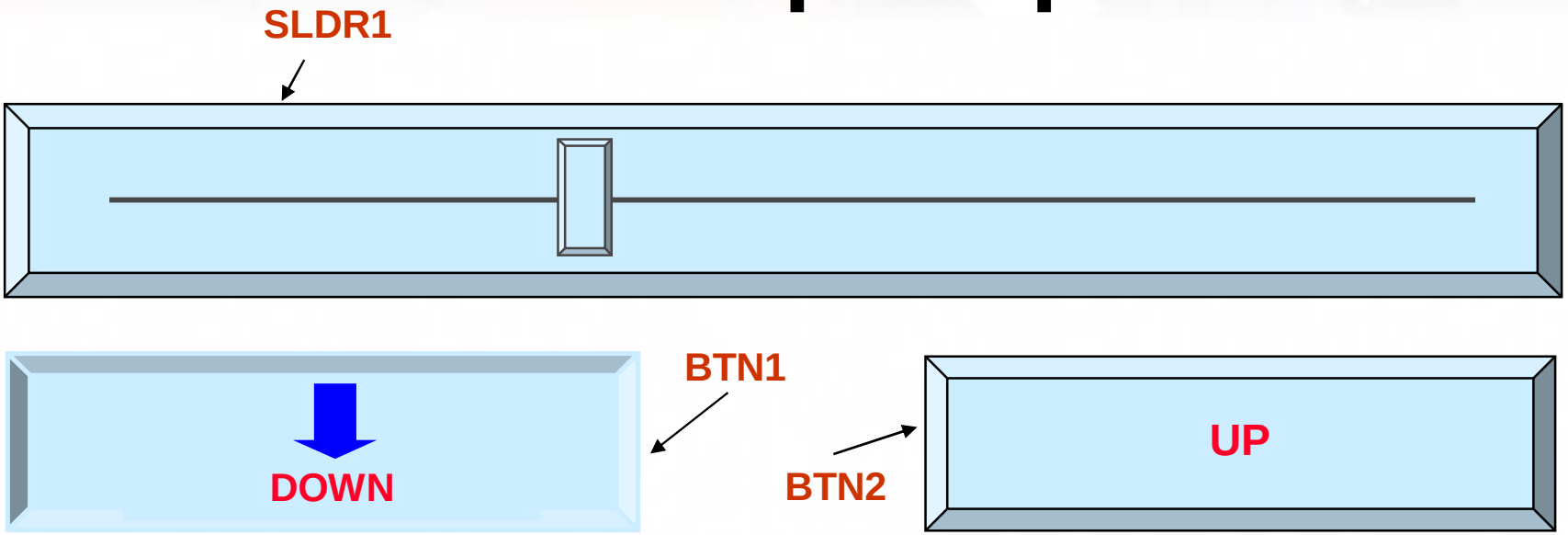
GOLMsgCallback()

- **ДОЛЖНА находиться в коде приложения**
- **Выполняет различные действия элементов**
  - пример: сменить картинку при нажатой кнопке
- **Взаимодействие с системой**
  - пример: зажечь светодиод при нажатой кнопке
- **Входные параметры:**
  - **objMsg**: команда, переданная элементом
  - **pObj**: указатель на элемент
  - **pMsg**: указатель на структуру команды
- **выходные:**
  - **TRUE**: выполнено
  - **'0'**: пропущено

GOLMsgCallback() пример

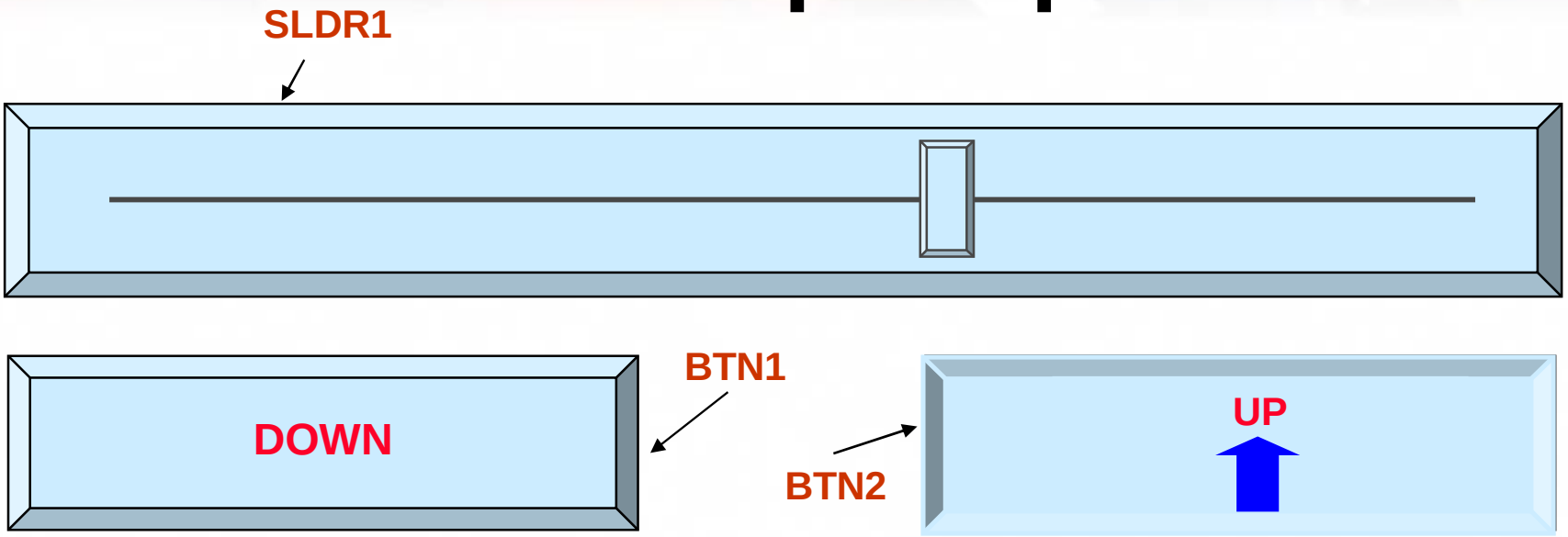


GOLMsgCallback() пример



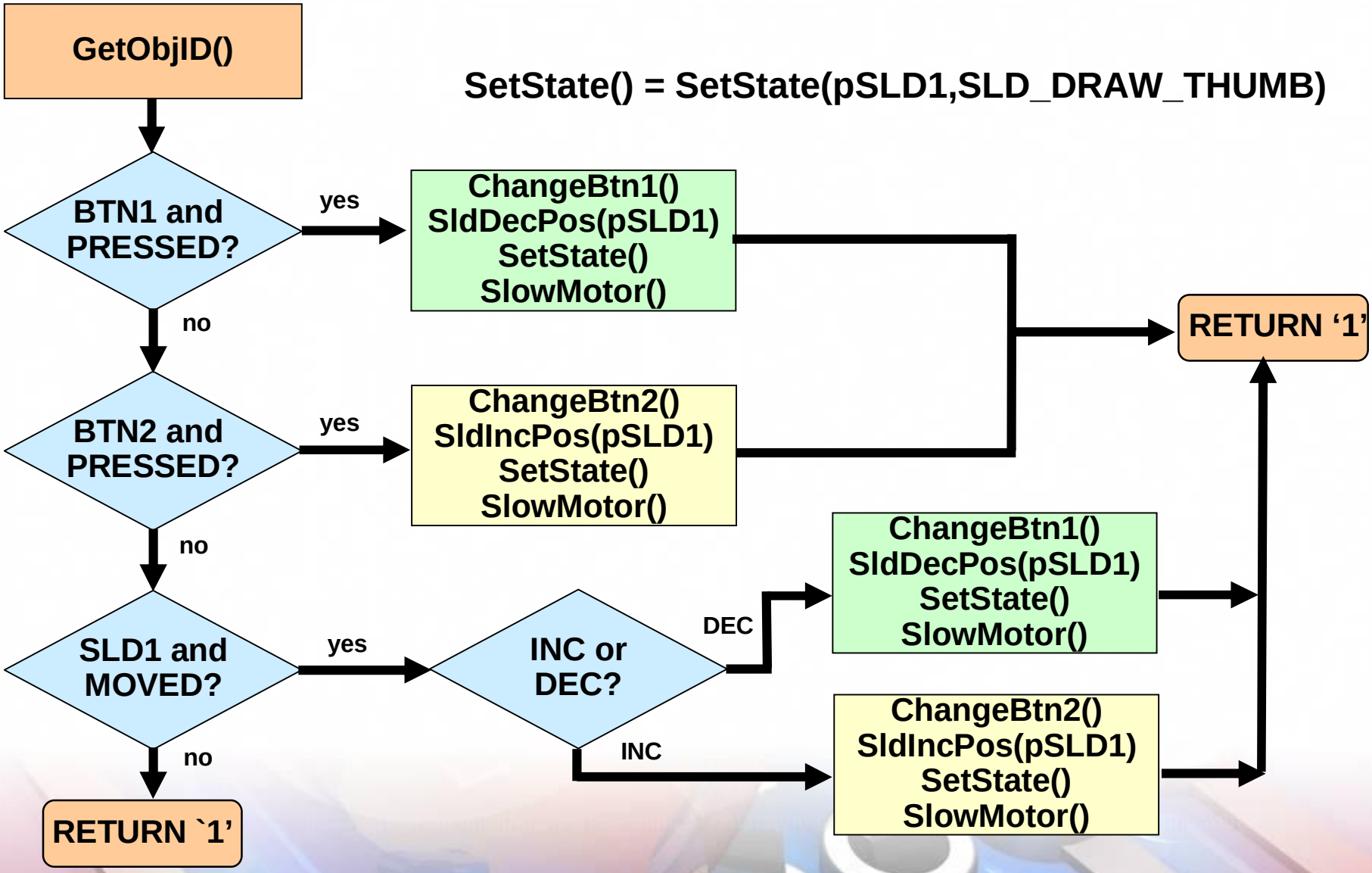
- действия элементов:
  - движение бегунка слайдера налево
  - смещение текста в кнопке BTN1
  - дополнительная стрелка в кнопке BTN1
- действие системы:
  - уменьшение скорости мотора

GOLMsgCallback() пример



- действия элементов:
  - движение бегунка слайдера вправо
  - смещение текста в кнопке BTN2
  - дополнительная стрелка в BTN2
- действие системы:
  - увеличение скорости двигателя

GOLMsgCallback() пример

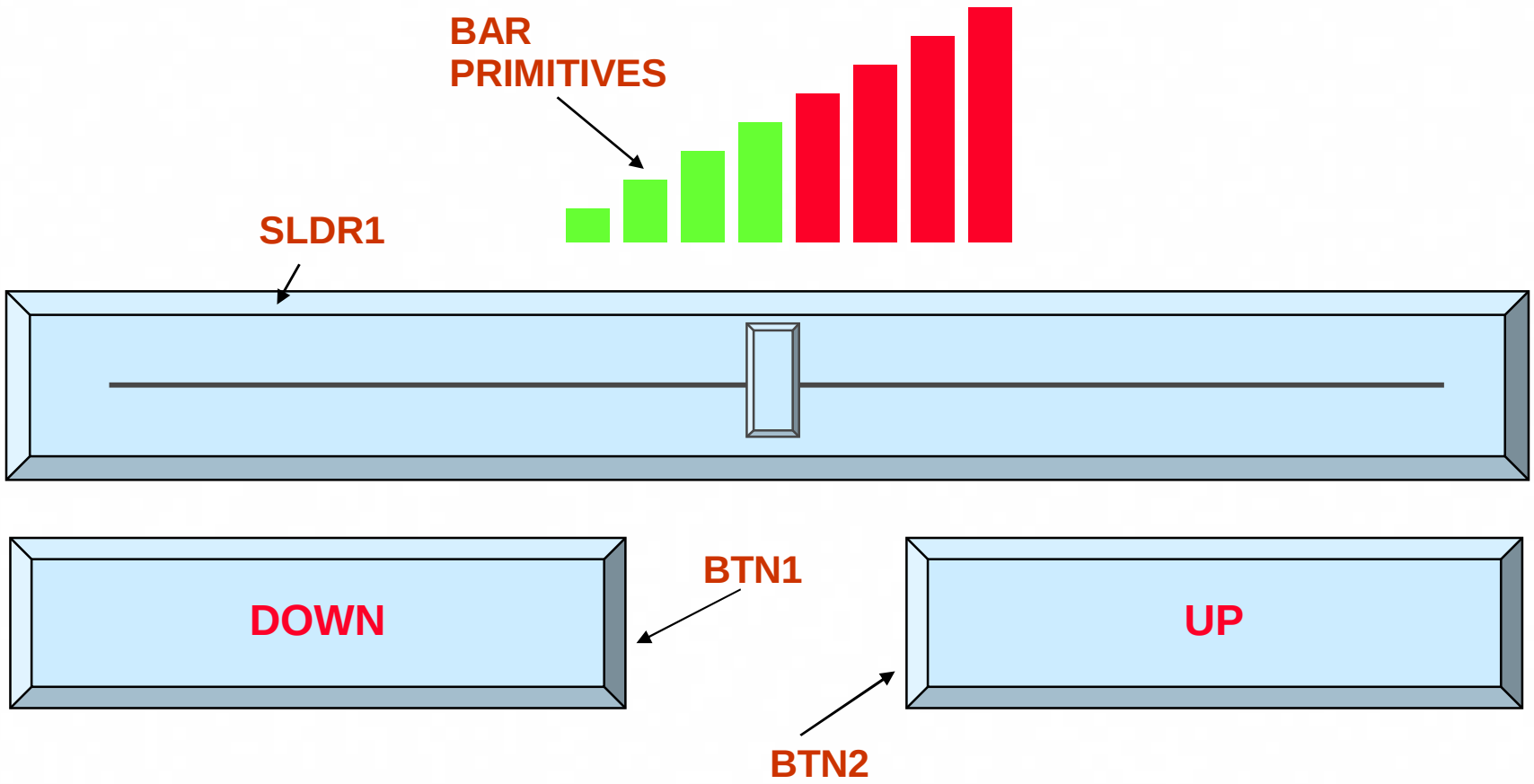


GOLDrawCallback()

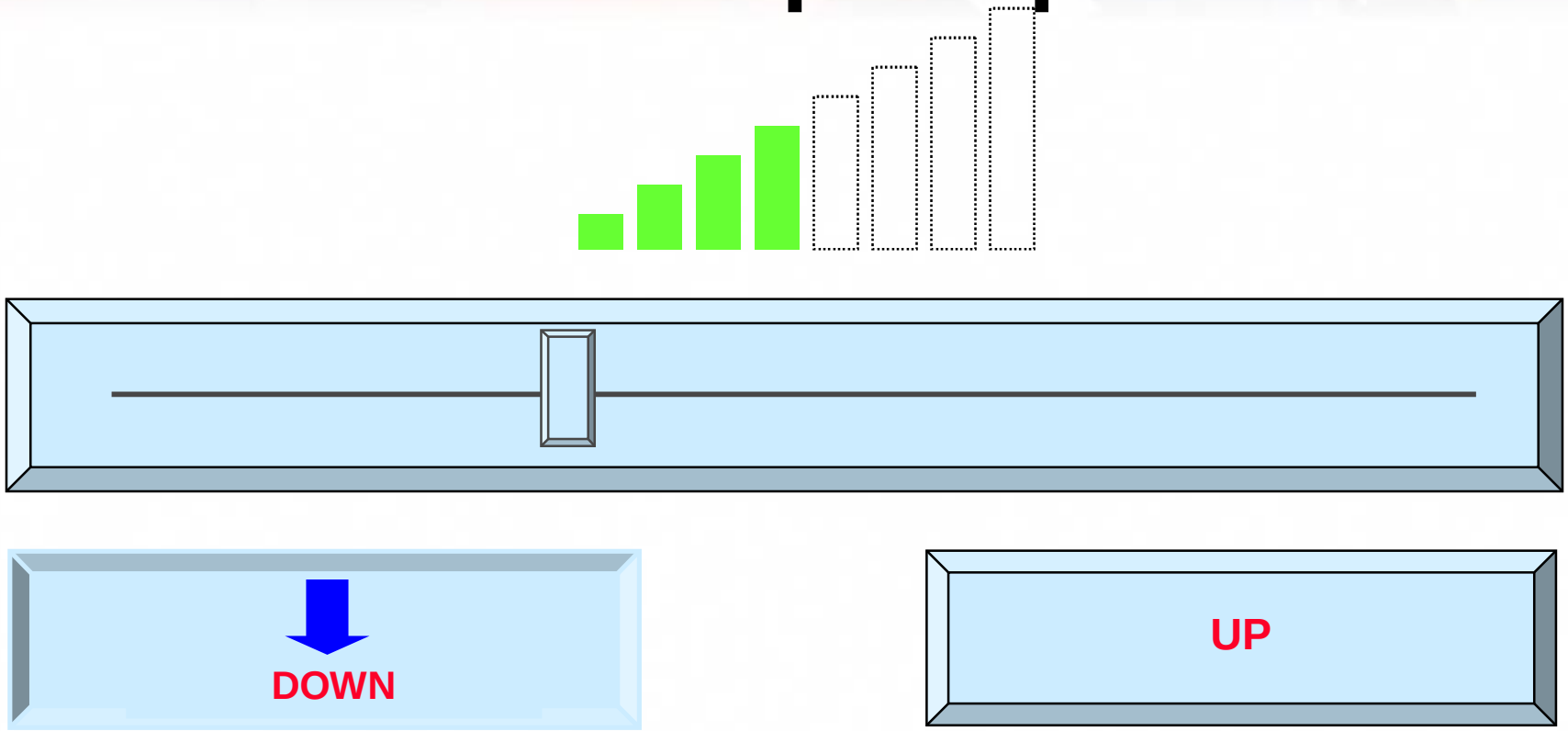
Продвинутое управление GOLDrawCallback()

- **вызывается из `GOLDraw()` по окончании прорисовки**
- **ДОЛЖНА находиться в коде приложения**
  - `TRUE` при передаче управления `GOLDraw()`
  - `'0'` при удержании управления
- **Реализация расширенных возможностей**
  - пример: индикатор уровня сигнала
- **Мониторинг и управление непрерывными процессами**
  - пример: удержание кнопки для снижения громкости
- **Only safe place to:**
  - Change drawing parameters
  - Modify active linked list

GOLDrawCallback()
пример

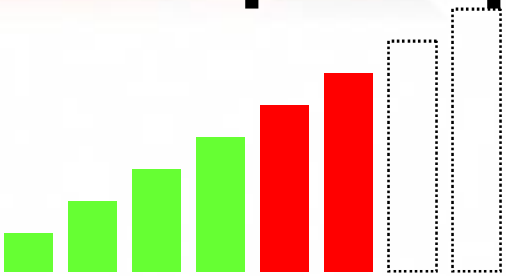


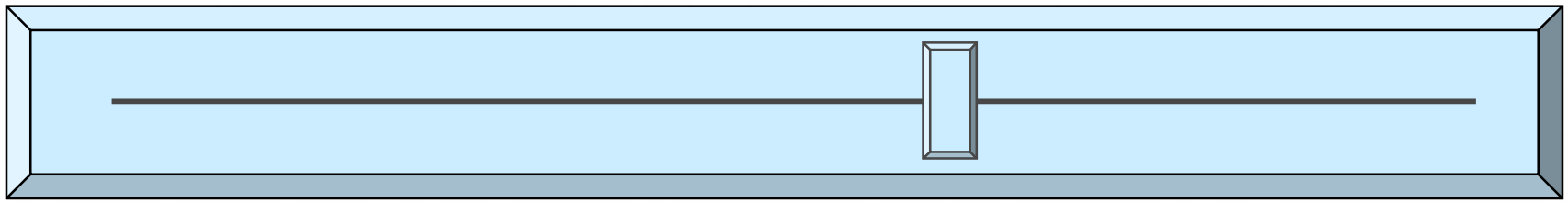
GOLDrawCallback() пример

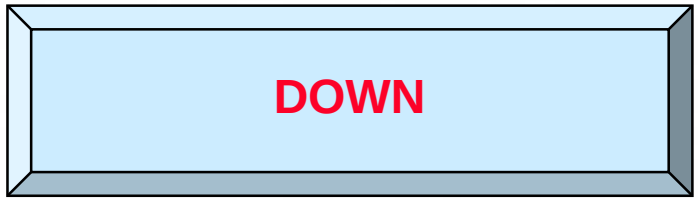


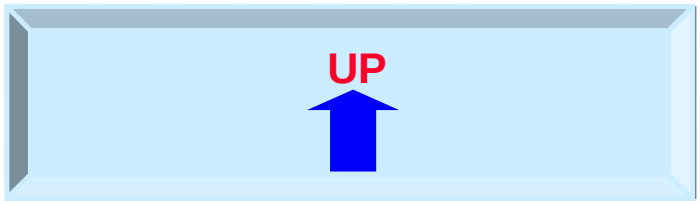
- действия элементов:
  - движение бегунка слайдера налево
  - смещение текста и дополнение картинкой в кнопке BTN1
  - снятие окраски прямоугольников
- действие системы:
  - уменьшение громкости

GOLDrawCallback()
пример



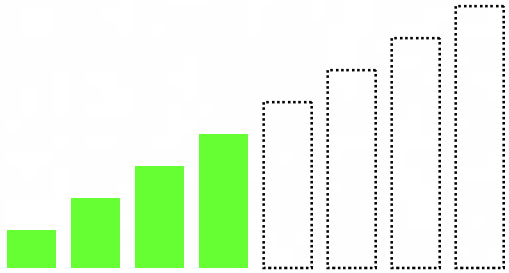






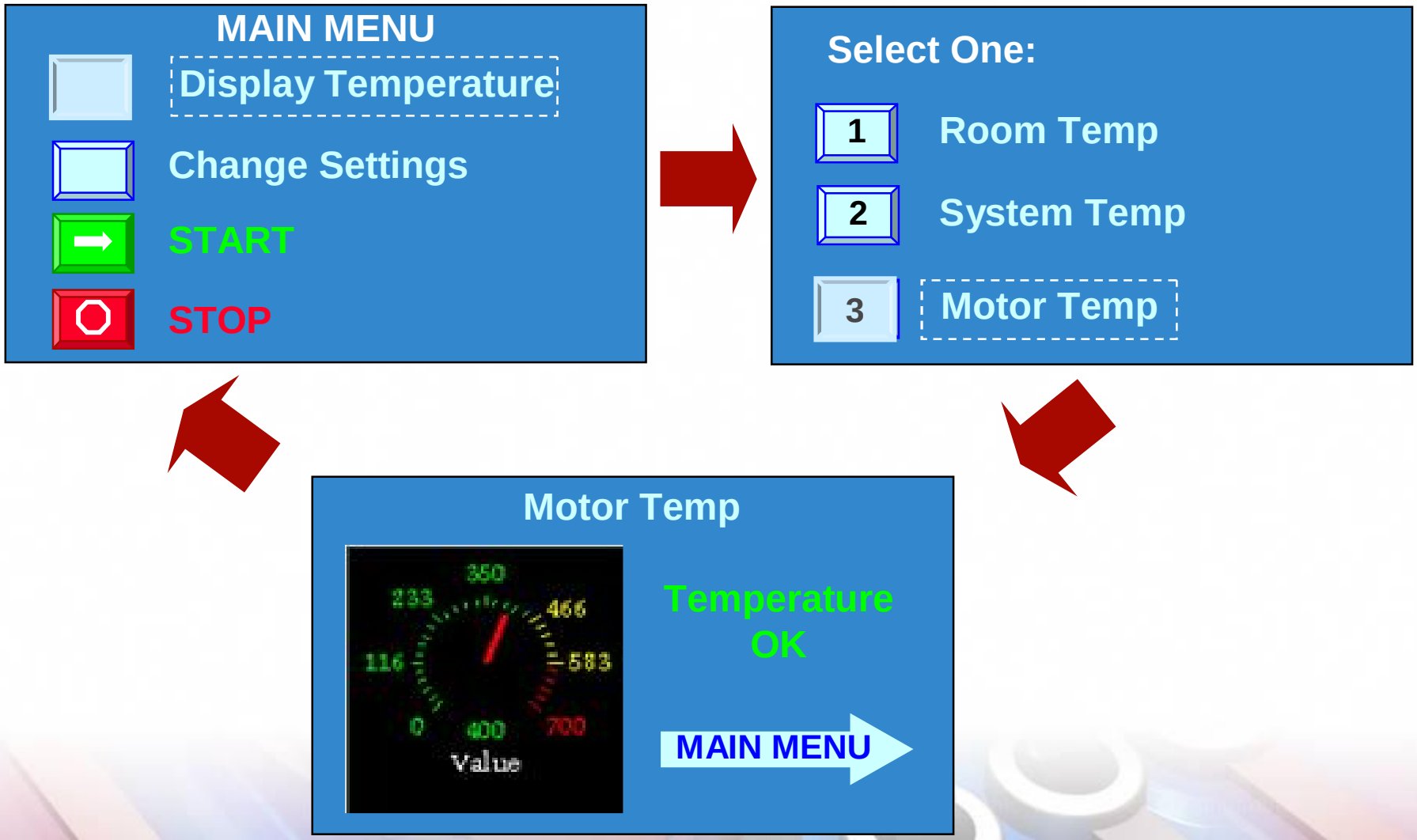
- действия элементов:
  - движение бегунка слайдера налево
  - смещение текста и дополнение картинкой в кнопке BTN1
  - закраска прямоугольников
- действие системы:
  - увеличение громкости

GOLDrawCallback()
пример – блок-схема

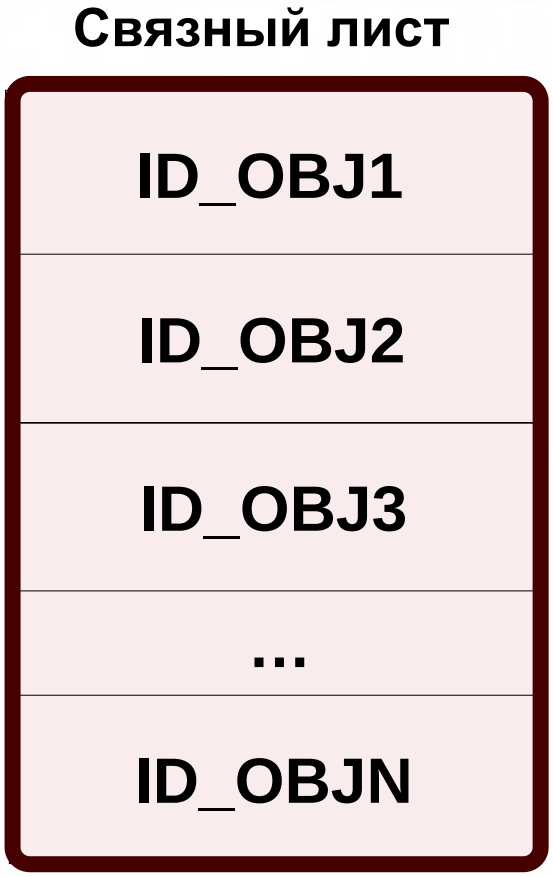


Использование в комплексе

Примеры GUI



Создание элементов



цикл…

- **`ObjCreate(,,,)`**
  - Запонлить структуру элемента
  - Добавить элемент в конец связного листа
- **`GOLDraw()`**
  - Анализ связного листа
  - Прорисовка согласно битам статуса состояния

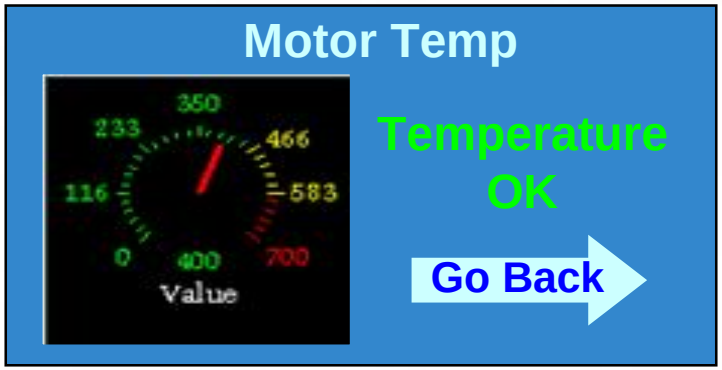
Настройки управления экраном один список

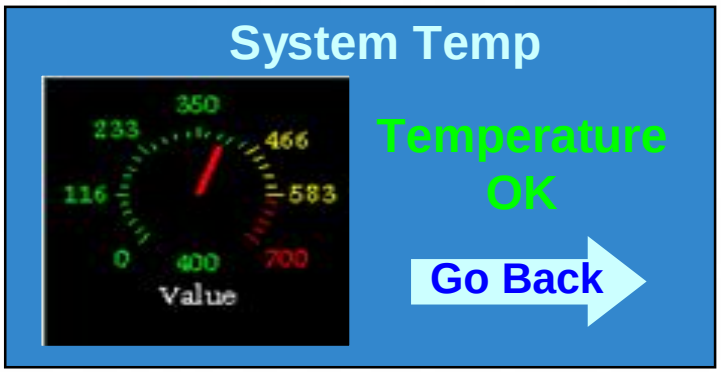
- **Скрыть старые элементы и нарисовать новые**
  - Может увеличить время прорисовки
- **Рисовать поверх старых элементов**
  - Прорисовка только части изображения
  - Меньше время прорисовки
  - Необходимо отключать «нижние» элементы

Настройки управления экраном много списков

- **Меньше время прорисовки**
- **Один список на экране**
  - используются GOL управление API
    - **`GOLGetList()` – сохранить активный список**
    - **`GOLNewList()` – установить указатель на пустой список**
    - **`GOLSetList(pList)` – выбрать активный список**
  - **`ObjCreate(,,,)`** для создания новых списков
  - Используется heap-область
    - **Пример: GUI …**

Один список на экран требования к heap





MotorTemp

1 Buttons	(1 * 28 = 28 bytes)
2 Static Text	(2 * 22 = 44 bytes)
1 Meter	(1 * 40 = 40 bytes)
	Total 112 bytes

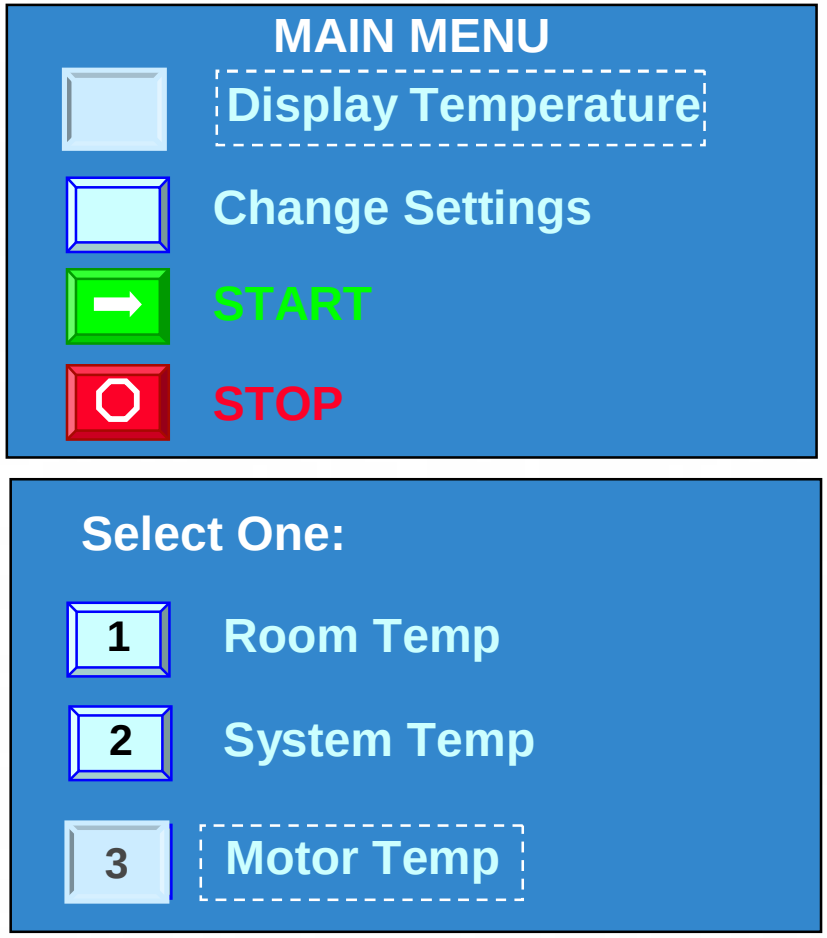
SystemTemp

1 Buttons	(1 * 28 = 28 bytes)
2 Static Text	(2 * 22 = 44 bytes)
1 Meter	(1 * 40 = 40 bytes)
	Total 112 bytes

RoomTemp

1 Buttons	(1 * 28 = 28 bytes)
2 Static Text	(2 * 22 = 44 bytes)
1 Meter	(1 * 40 = 40 bytes)
	Total 112 bytes

Один список на экран требования к heap



MainMenu

4 Buttons	(4 * 28 = 112 bytes)
5 Static Text	(5 * 22 = 110 bytes)
Total	222 bytes

TempSelect

3 Buttons	(3 * 28 = 84 bytes)
4 Static Text	(4 * 22 = 88 bytes)
Total	172 bytes

Total Heap Required…

5 screens	730 bytes
3 style schemes	60 bytes
Total	790 bytes

Настройки управления экраном
много списков

- **Создание списков «на лету»**
  - функция **`GOLFree()`** для выделения heap-области
  - **`ObjCreate(,,,)`** для создания новых списков
  - Распределяйте heap-область для длинных списков
    - **Пример GUI: 222 байта**
  - Наиболее эффективный с точки зрения использования памяти метод


Microchip Technology
MASTERS 08
CONFERENCE
YOU + MICROCHIP
ENGINEERING THE FUTURE TOGETHER

Выводы

Выводы

По итогам данной презентации Вы освоили:

- **Основные особенности создания низкоуровневых драйверов для графической библиотеки**
- **Написание программ для отображения картинок, шрифтов и графических примитивов на ЖКИ-панели**
- **Написание программ для создания и управления основными объектами (элементами) GUI**
- **Создание приложения на базе графической библиотеки Microchip**

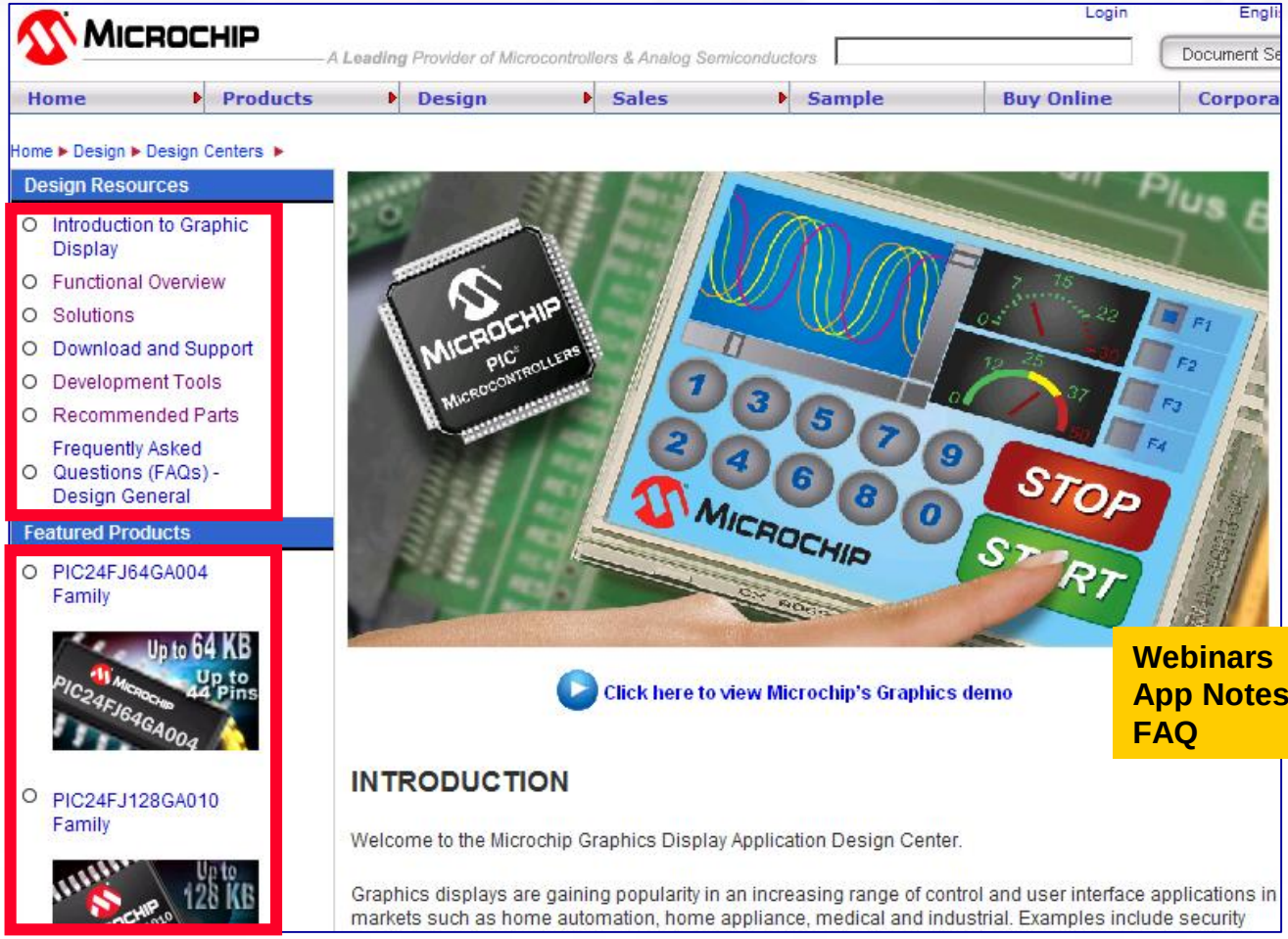
Полезные советы

- **Избегайте дробления heap-области**
  - Используйте GOLFree()
  - НЕ рекомендуется …
    - **Вручную удалять элементы связного списка**
    - **Изменять список до окончания `GOLDraw()`**
- **Во избежание нежелательных эффектов**
  - НЕ рекомендуется …
    - **Изменяйте свойства прорисовки в ISR**
    - **Изменяйте биты состояния элементов и стили до окончания `GOLDraw()`**

Поддержка на сайте

Site navigator

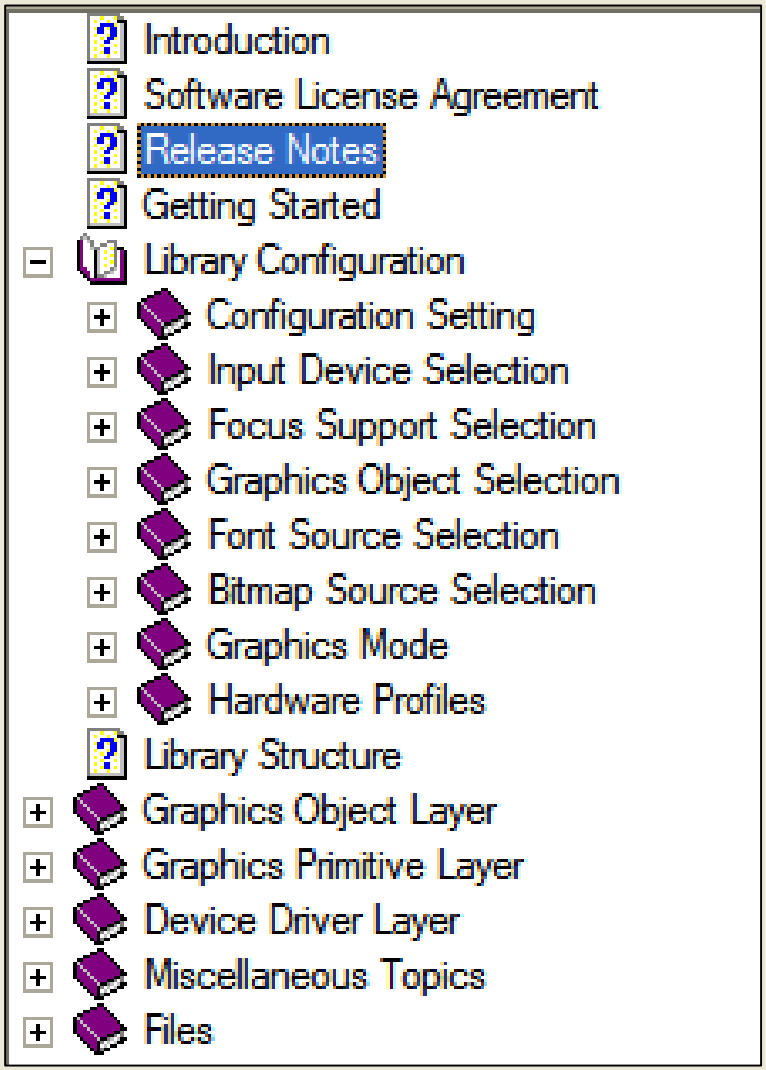
Product page



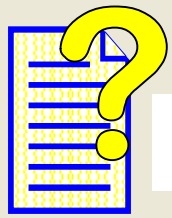
Webinars
App Notes
FAQ

http://www.microchip.com/graphics

Поддержка Library Help



Help files are included as part of the MPLAB® Graphics Library installation and are located in the following directory:



Graphics Library Help

Tools

- **Microchip Graphics Library v1.52**
  - http://www.microchip.com/graphics
- **MPLAB® IDE (v8.14)**
  - http://www.microchip.com/mplab
- **MPLAB C18 / C30 / C32 compilers**
  - http://www.microchip.com/mplabc
- **Цветовые схемы**
  - http://www.colorschemer.com

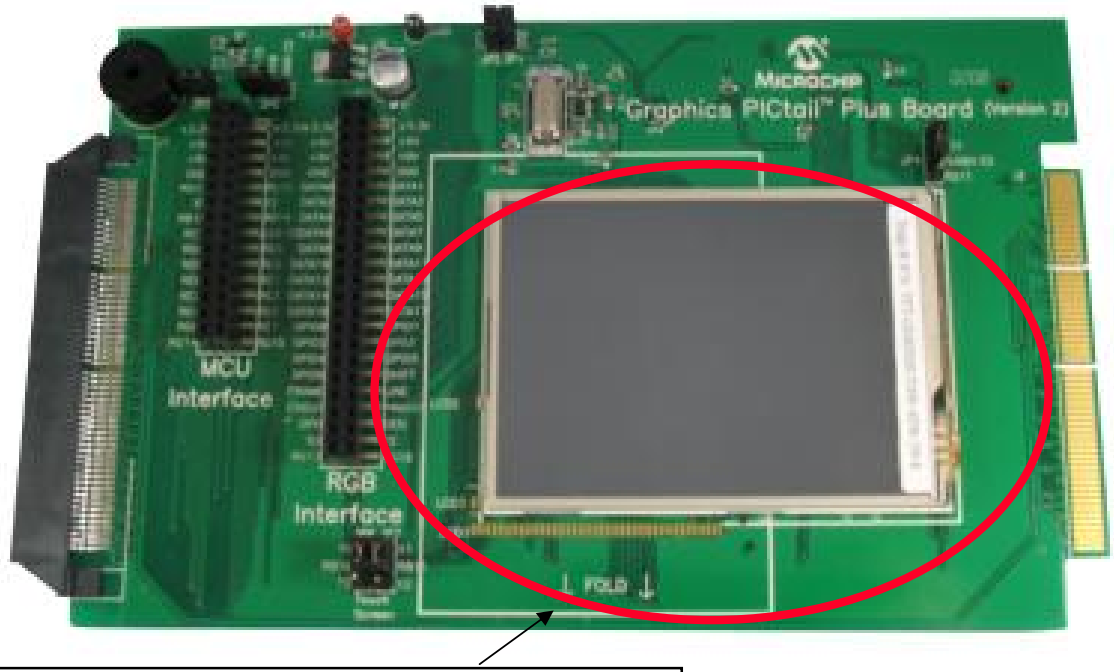
Средства отладки

Graphics PICtail™ Plus Demo Board

- **Работа с любыми 16- и 32-битными МК на PIM**
- **Выбор перемычкой между:**
  - ЖКИ со встроенным контроллером (LGDP4531)

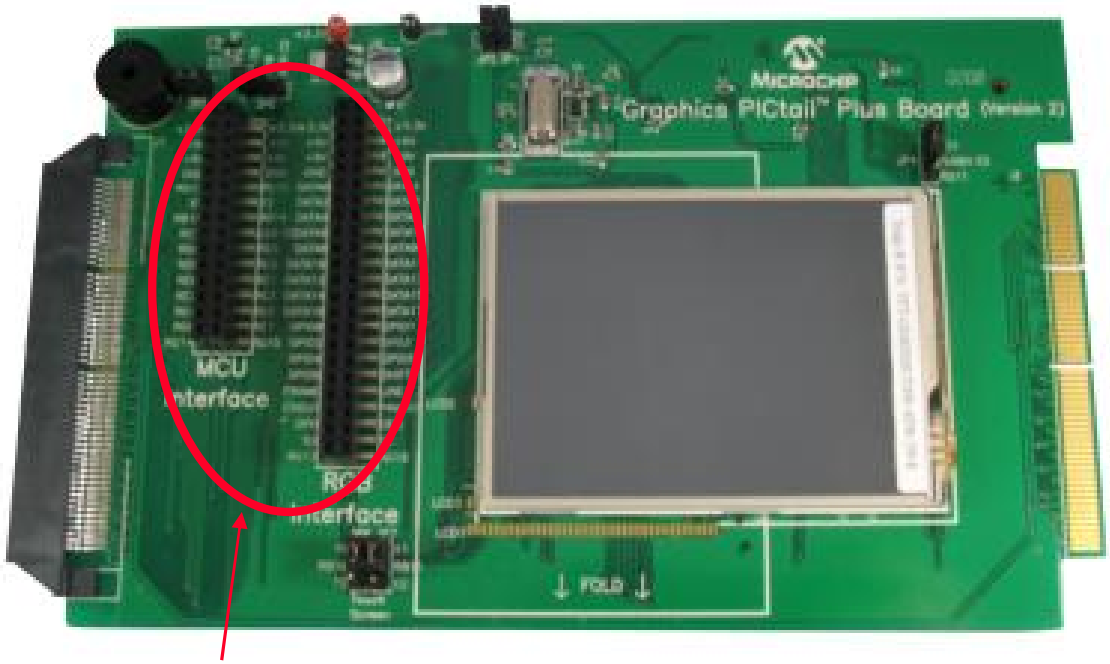
  ИЛИ
  - RGB разъем и SSD1906 контроллер

Graphics PICtail™ 2 Plus Board Features



- 2.8” QVGA
- 65K цветов
- Портретная/ландшафтная ориентация
- Резистивная тач-панель

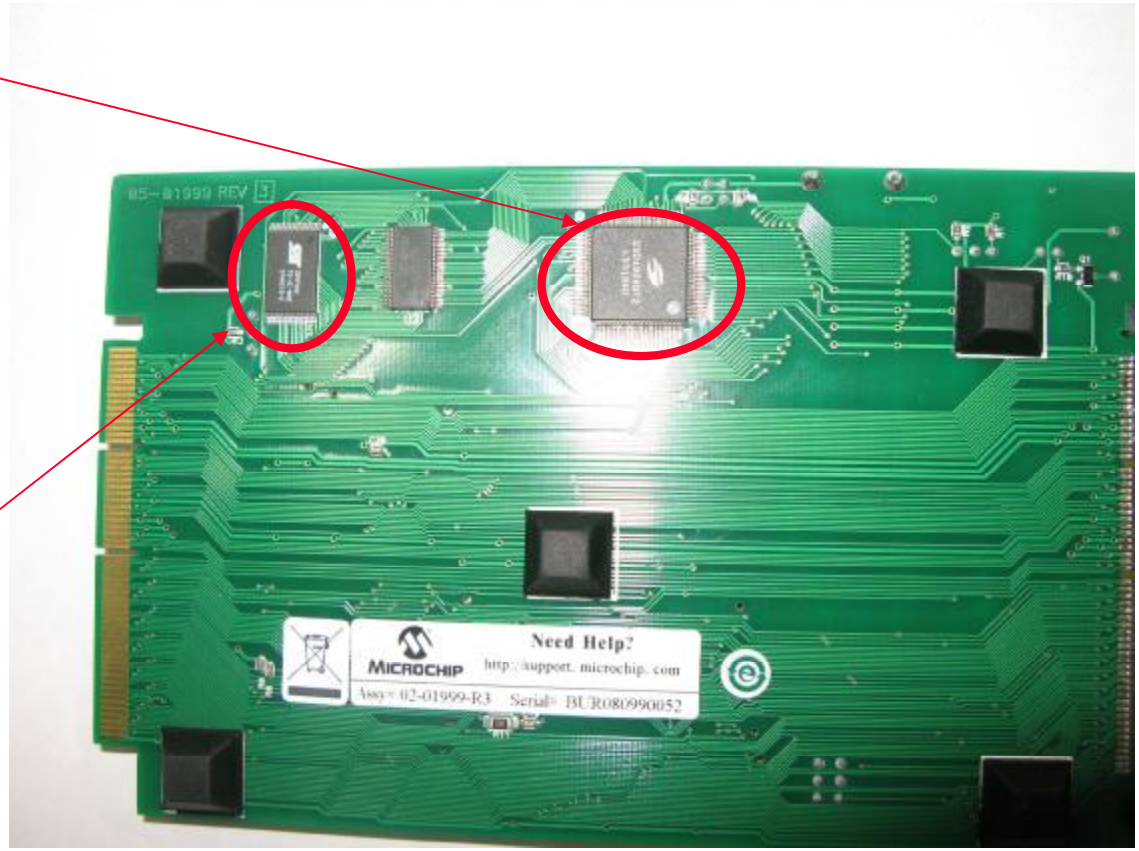
Graphics PICtail™ 2 Plus Board Features



- Разъем для автономного графического контроллера (Soloman Systech SSD1906)
- 4/8-бит STN или CSTN
- 18-бит HR-TFT
- 9/12/18-бит TFT

Graphics PICtail™ 2 Plus Board Features

- SSD1906 автономный графический контроллер
- 4M (512K x 8) flash-память



YOU + MICROCHIP ENGINEERING THE FUTURE TOGETHER

СПАСИБО ЗА ВНИМАНИЕ!!

Trademarks