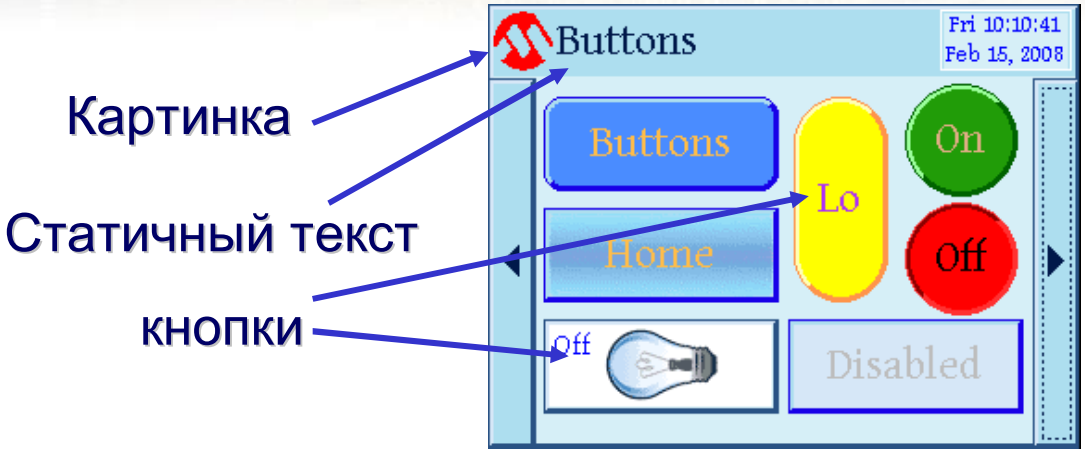
Обзор Графической библиотеки Microchip Graphics Library

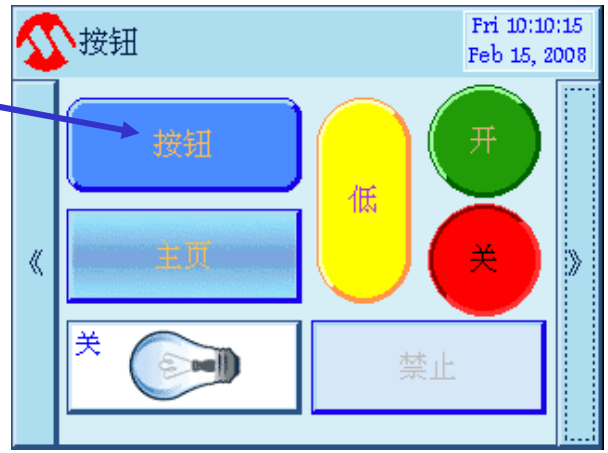
Терминология

- **Application Programming Interface (API):** **Набор функций, которые могут быть вызваны из Вашей программы для получения доступа к другой программе или библиотеки.**
- **Графические примитивы:** **графический элемент, такой как точка, линия, окружность и т.п. из которых можно создать более сложные Графические Объекты**
- **Графические объекты:** **любые из различных графических форм, которые можно использовать в качестве элементов управления** (кнопки, графики, слайдеры и т.п.). Эти объекты можно использовать для получения сообщения от пользователя системы.
- **Виджет:** **другое имя Графического Объекта.**
- **Глиф, символ:** **графическое представление в том или ином шрифте всех отдельных символов письма** (букв, иероглифов, цифр, знаков пунктуации и т.п.)

Графическая библиотека



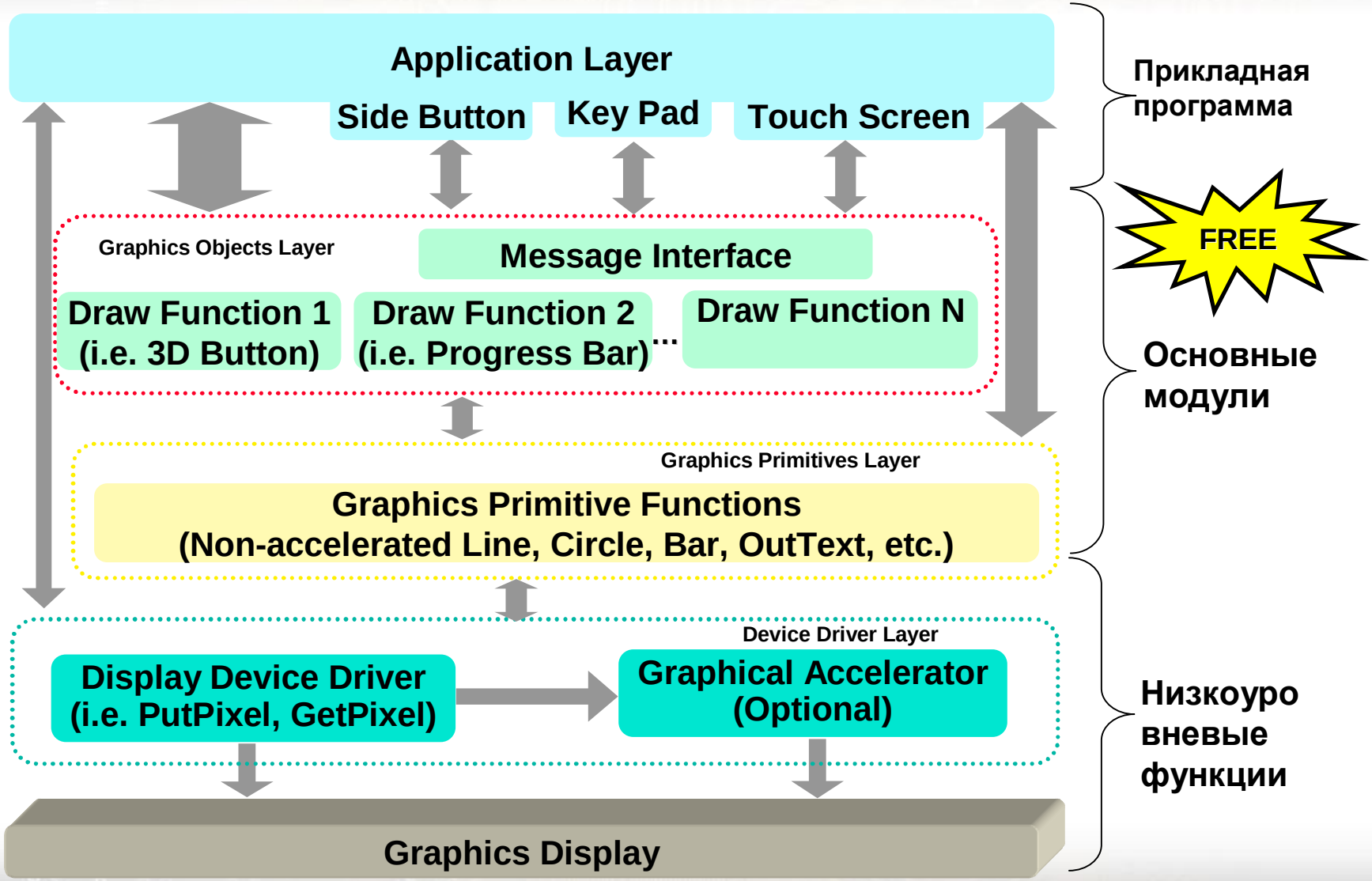
Разноязычные шрифты



Ключевые моменты:

- Работает с 16- и 32-х битными контроллерами Microchip
- Модульная структура – Подключайте только то, что Вам нужно
- Не зависит от размера экрана и разрешения
- Низкие требования по памяти и производительности
- Средства разработки и отладки

Структура библиотеки



Настройка библиотеки
`#defines` в `GraphicsConfig.h`

- Выбрать методы управления
  - Сенсорный дисплей или клавиатура
- Выберете используемые заготовки
  - кнопки, checkbox, слайдер, и т.п.
- Вкл/Выкл фокусировку объектов
- Графический режим
  - монохромный
  - Ориентация экрана
  - Поддержка UNICODE
- Выберете место расположения шрифтов и картинок
  - Внутренняя Flash, внешняя память или оба

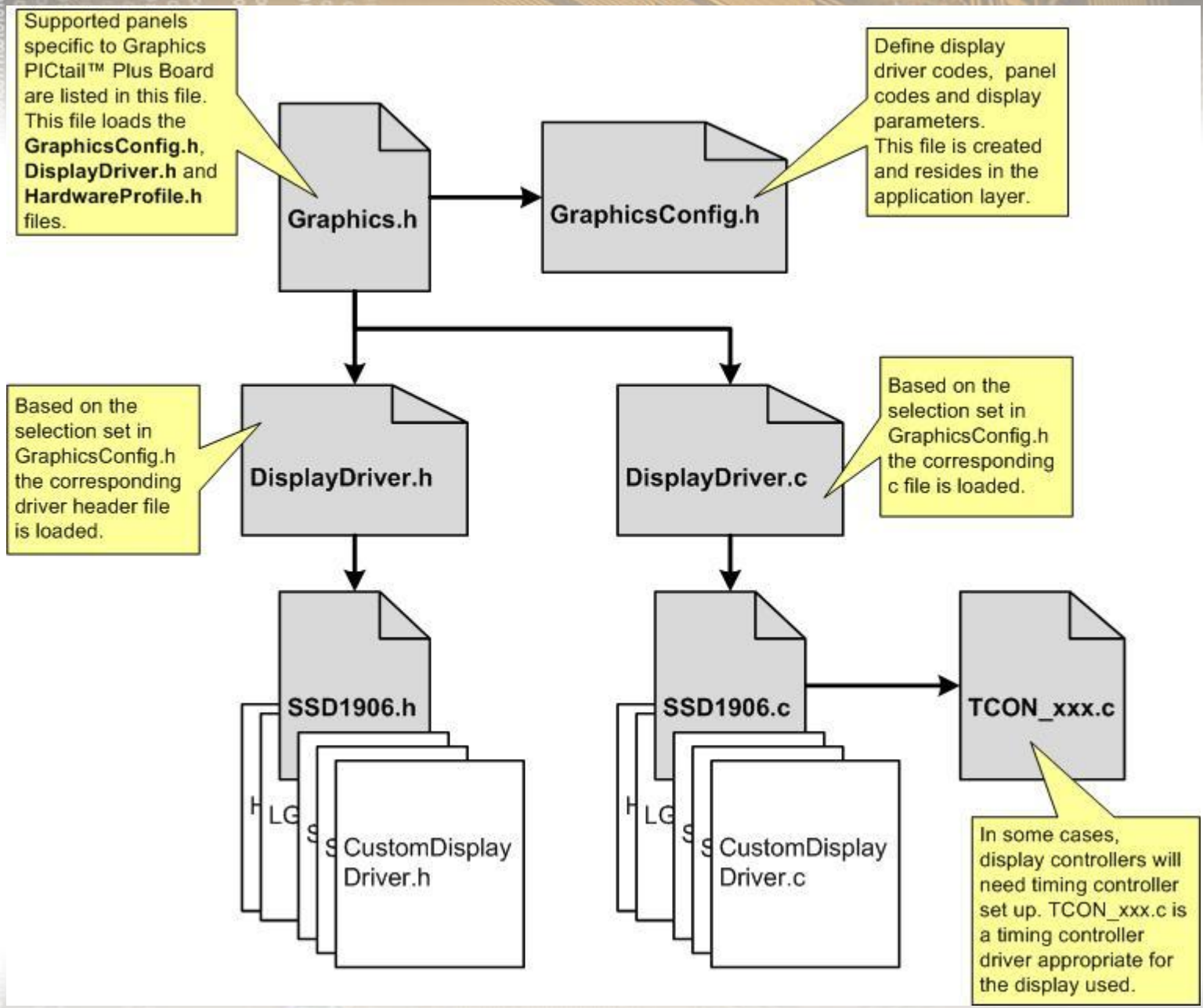
Создание Драйвера для дисплея

Основные задачи драйвера дисплея

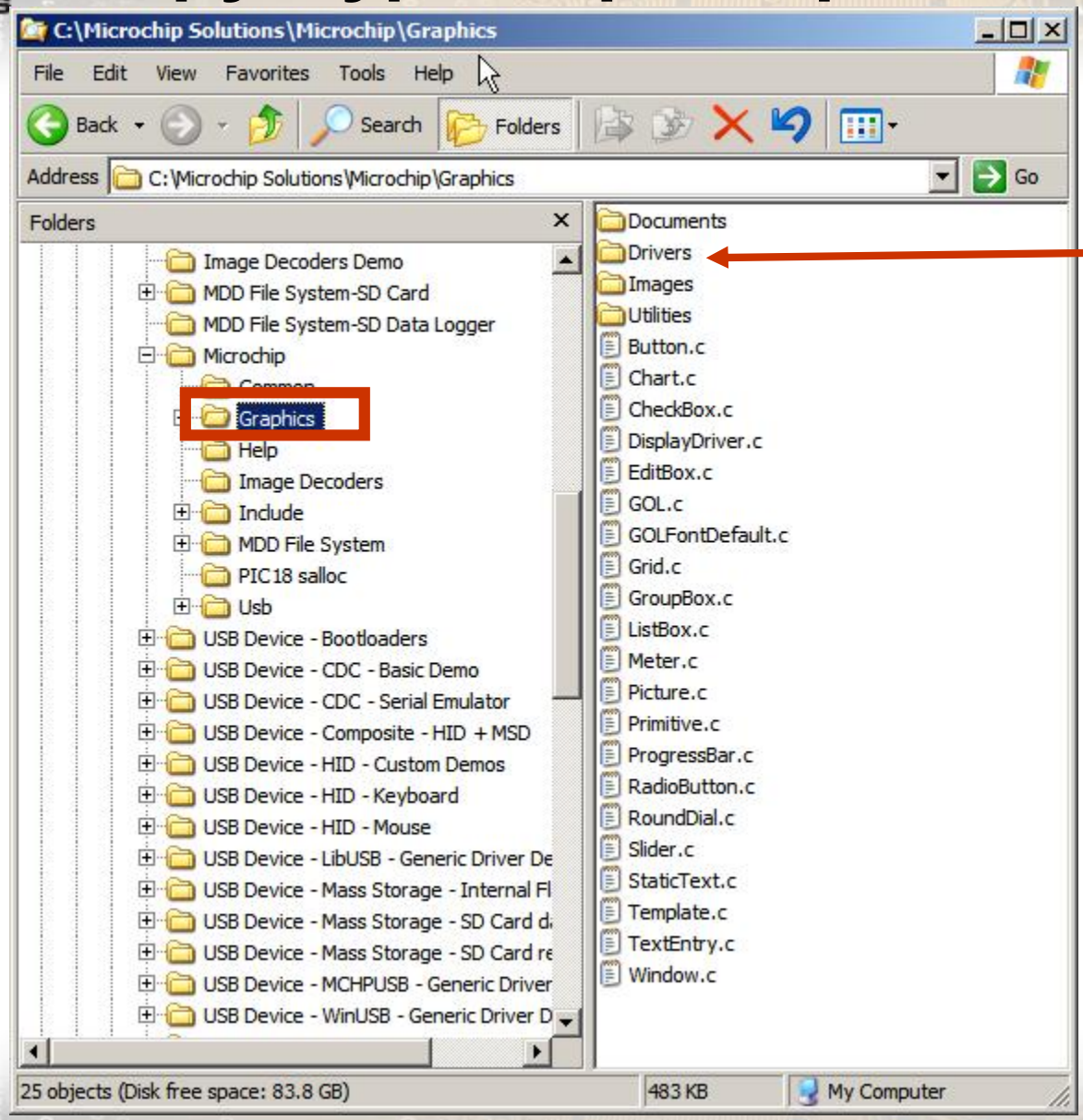
- Инициализация дисплея (ResetDevice())
- Перевод координат x,y в адрес буфера фрейма
- Запись информации о цвете (яркости) по этому адресу (PutPixel(…))
- Инициализация и управления доп. функциями дисплея
  - Timing controller
  - Аппаратного акселератора

Низкоуровневый драйвер

- **Нам нужно знать**
  - **Интерфейс с МК**
    - **если RGB шина, нужно использовать микросхему графического контроллера**
    - **Интерфейс типа I80 или M68 – используйте Параллельный Мастер Порт (Parallel Master Port -- PMP)**
    - **Так же встречается SPI**
  - **Получите код инициализации от производителя индикатора**
    - **Лучше качество изображения**

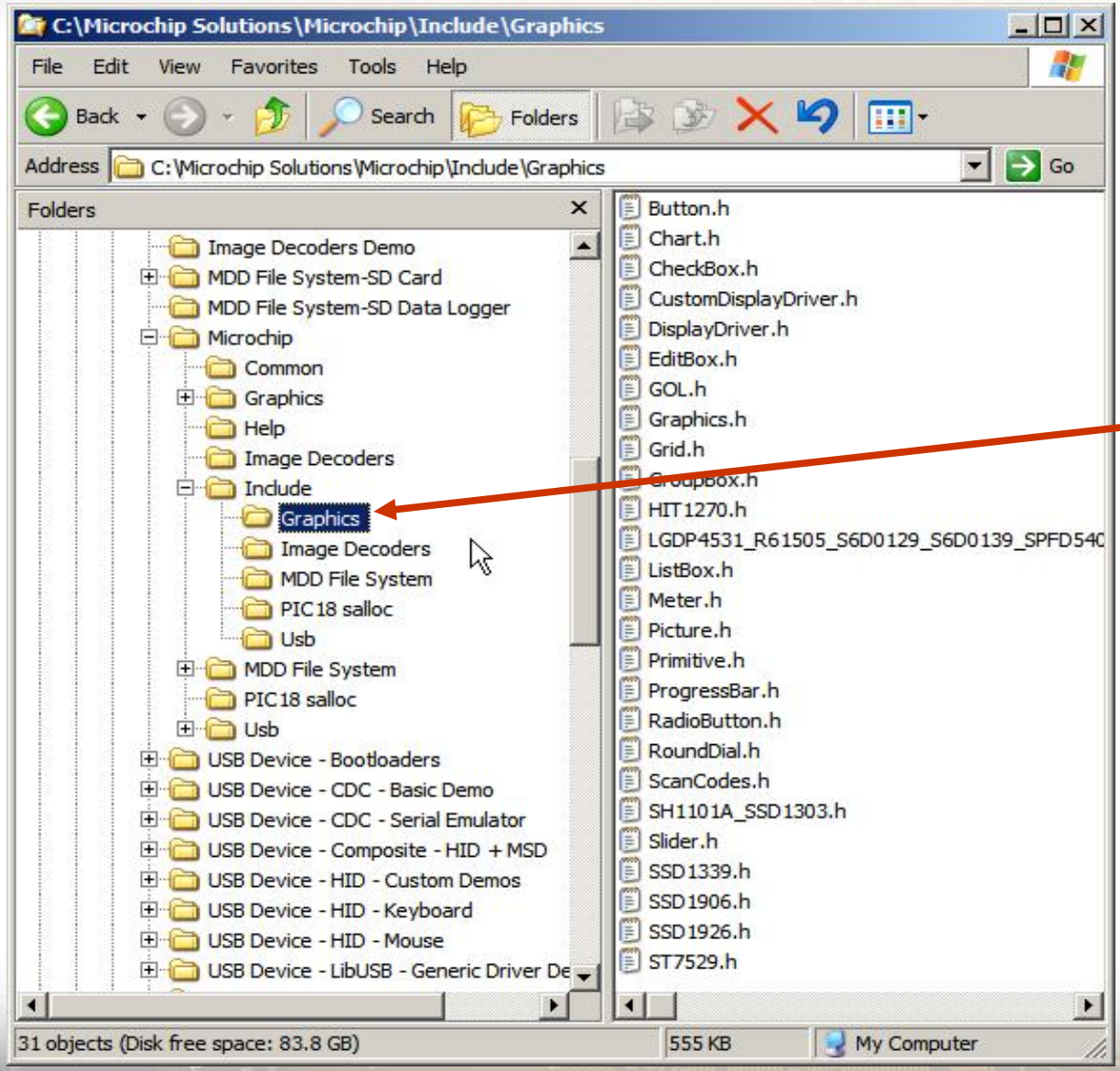


Структура директории библиотеки



[MyDriver].c
Поместите тут

Структура директории библиотеки



[MyDriver].h
Поместите тут

Уровень примитивов Графической библиотеки Microchip

Уровень примитивов

- Общается напрямую с драйвером
- Compile time options in GraphicsConfig.h:
  - Выбор драйвера
  - Расположение шрифтов (внутр. память, внешняя или обе)
  - Расположение картинок (внутр. память, внешняя или обе)
  - Поддержка Юникода (AN1182)
- Должны быть включены в проект:
  - `primitive.c`
  - `primitive.h`

Функции уровня примитивов

- **Функции установки:**
  - **`InitGraph()` – инициализация дисплея**
  - **`ClearDevice()`**
    - **Очищает экран установленным цветом**
    - **Устанавливает курсор в позицию (0,0)**

Функции уровня примитивов Свойства рисования

- **Влияют на свойства примитивов до следующего *Set***
  - **`SetColor(COLOR)`**
    - **макрос `COLOR` расположен в `driver.h`**
  - **`SetFont(&fontimage)`**
    - **`&fontimage` – указатель на структуру шрифта**
  - **`SetLineType(key)`**
    - ***`SOLID_LINE`***
    - ***`DOTTED_LINE`***
    - ***`DASHED_LINE`***
  - **`SetLineThickness(key)`**
    - ***`NORMAL_LINE`***
    - ***`THICK_LINE`***

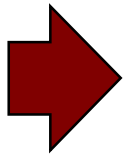
SetColor()

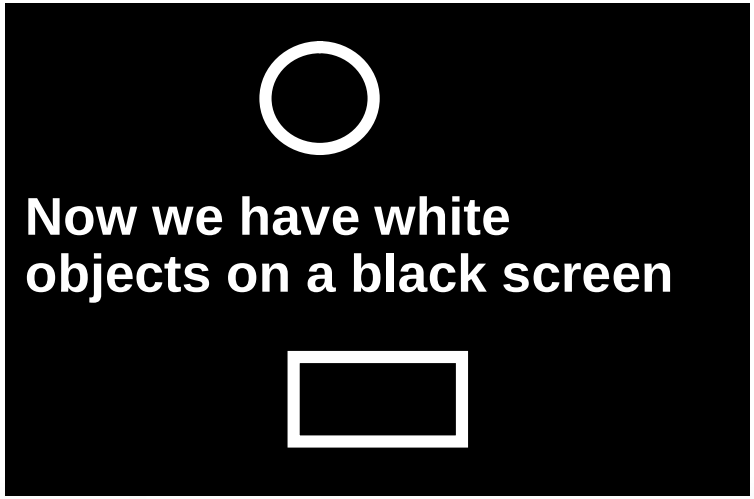
Создание нового цвета

- Используйте макрос `RGB565Convert(R,G,B)`
- Бесплатная утилита на www.colorschemer.com

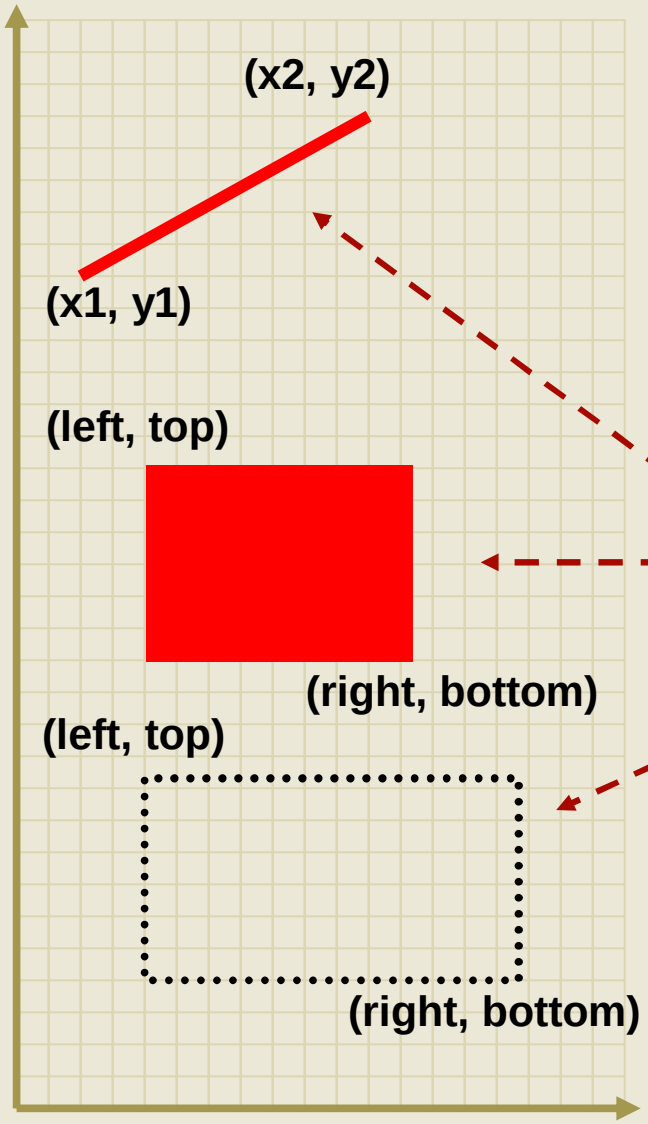
- **Пример:**
  - Установите цвет фона BLACK
  - Установите цвет точки WHITE

```
int main(void)
{
…
SetColor(BLACK);
ClearDisplay();
SetColor(WHITE);
…
}
```



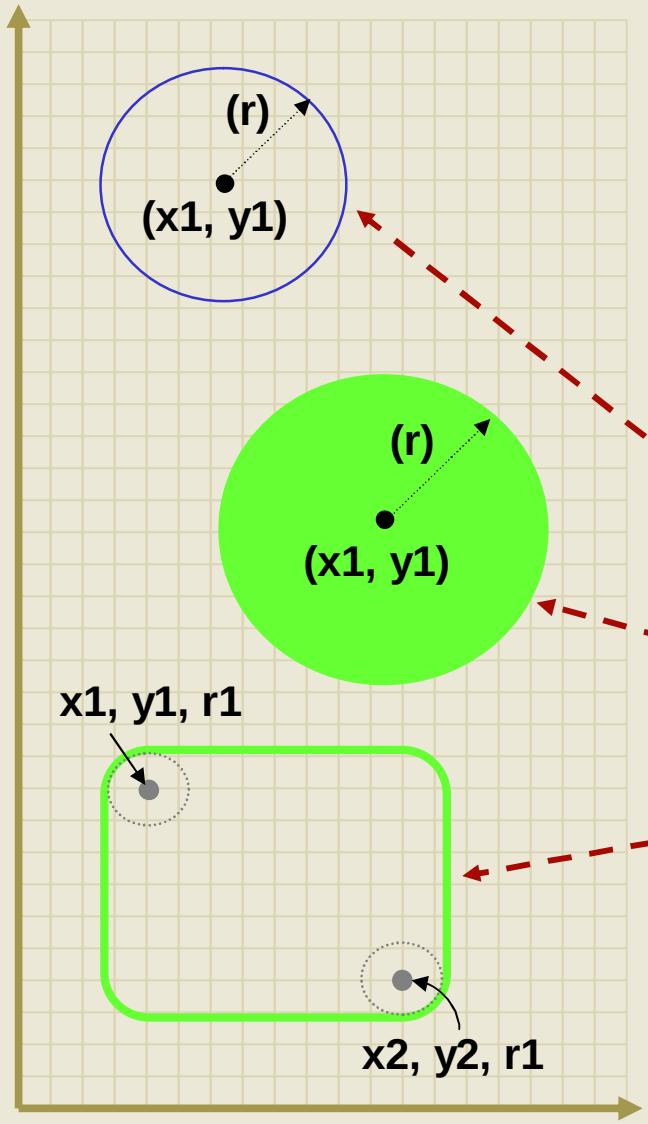


Функции уровня примитивов



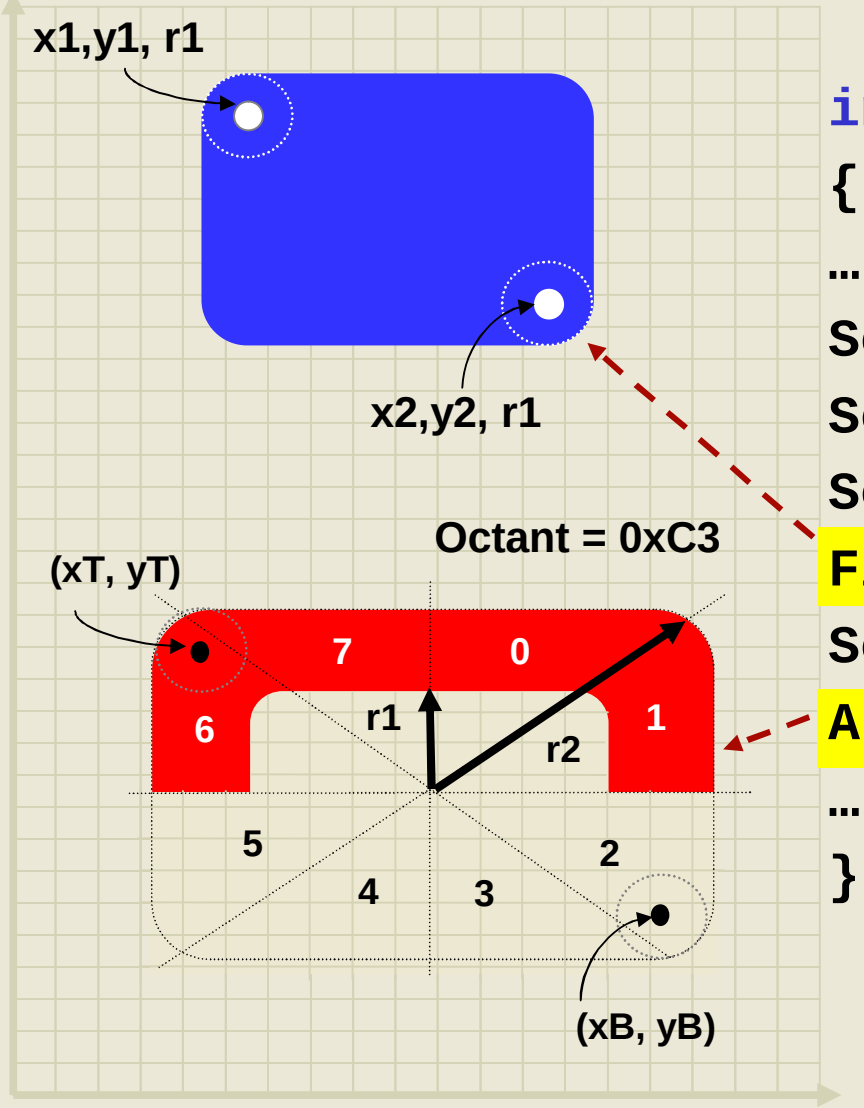

```
int main(void)
{
….
SetColor(BRIGHTRED);
SetLineType(SOLID_LINE);
SetLineThickness(THICK_LINE);
Line(x1, y1, x2, y2);
Bar(left, top, right,bottom);
SetColor(BLACK);
SetLineType(DOTTED_LINE);
Rectangle(left, top, right,bottom);
…
}
```

Функции уровня примитивов




```
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);
….
}
```

Функции уровня примитивов




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

Уровень примитивов
Шрифты

Шрифты

Определение:

Шрифт это файл, содержащий набор символов, букв, цифр, и .т.п. Шрифты создаются в редакторе шрифтов и часто связаны с работой дизайнеров. Готовые шрифты доступны из многих источников, но могут требовать лицензирования.

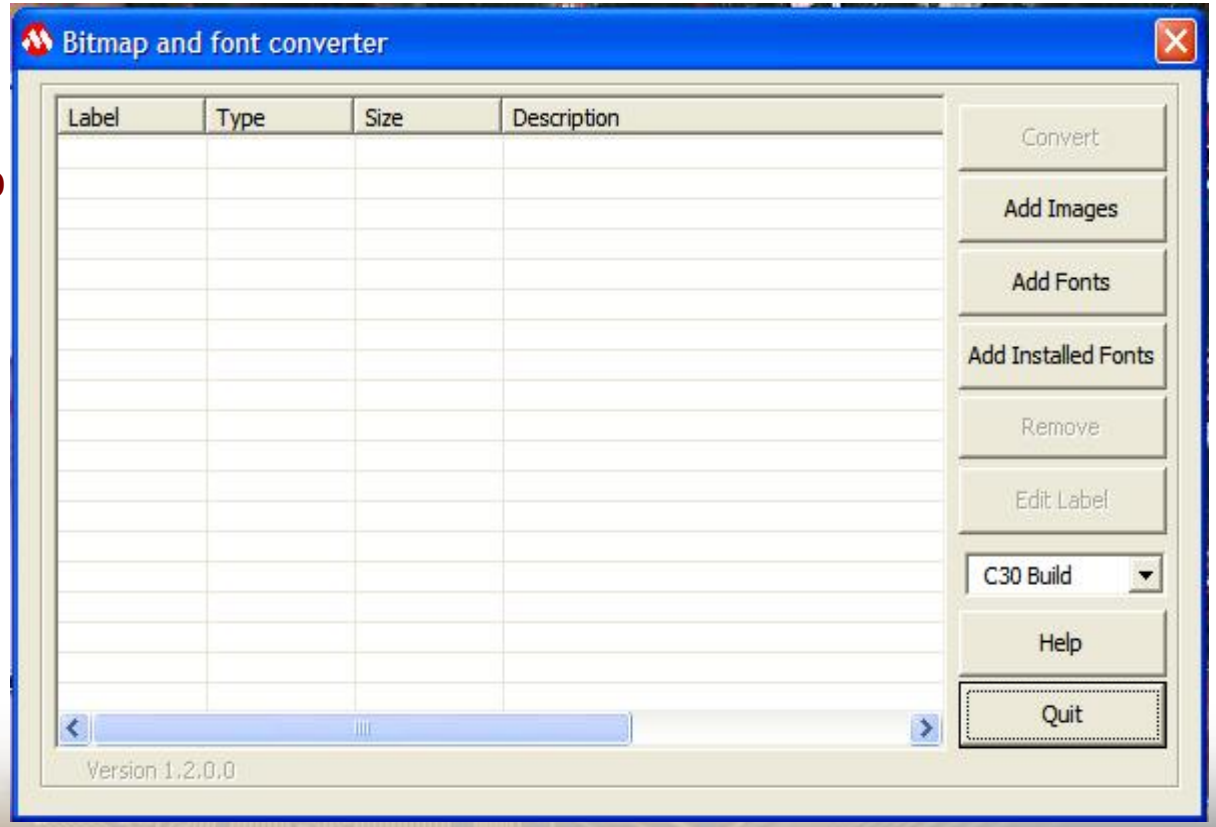
- **Библиотека поддерживает…**
  - **True Type и Open Type шрифты**
  - **Растровые шрифты**
- **Изображения шрифтов …**
  - **Могут храниться в Памяти программ или внешней памяти**
  - **Доступно создание «фильтрованных» шрифтов**
- **Поддержка юникода**
  - **AN1182 -- Fonts in the Microchip Graphics Library**

Использование шрифтов

- Преобразователь шрифтов и картинок
  - Бесплатная утилита, содержится в библиотеке
  - Преобразует в формат, понятный библиотеке

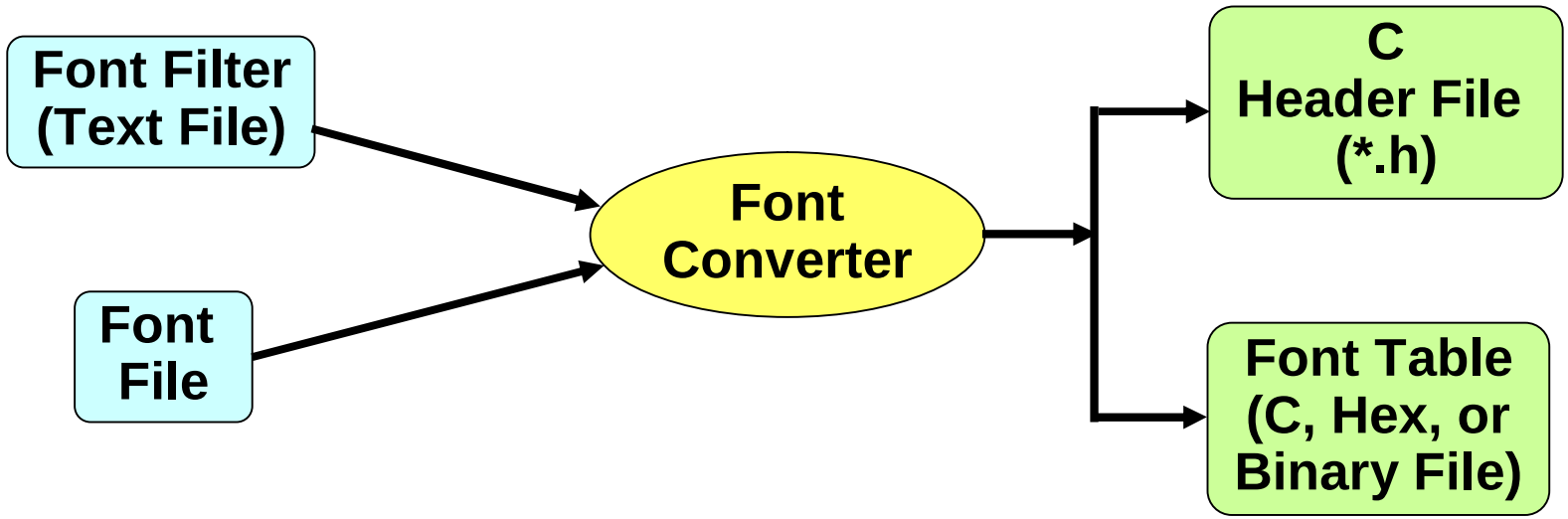
Как запустить:

Start ► Programs ► Microchip ► Graphics Library v1.75 ►Bitmap and Font converter



Фильтр для Шрифта

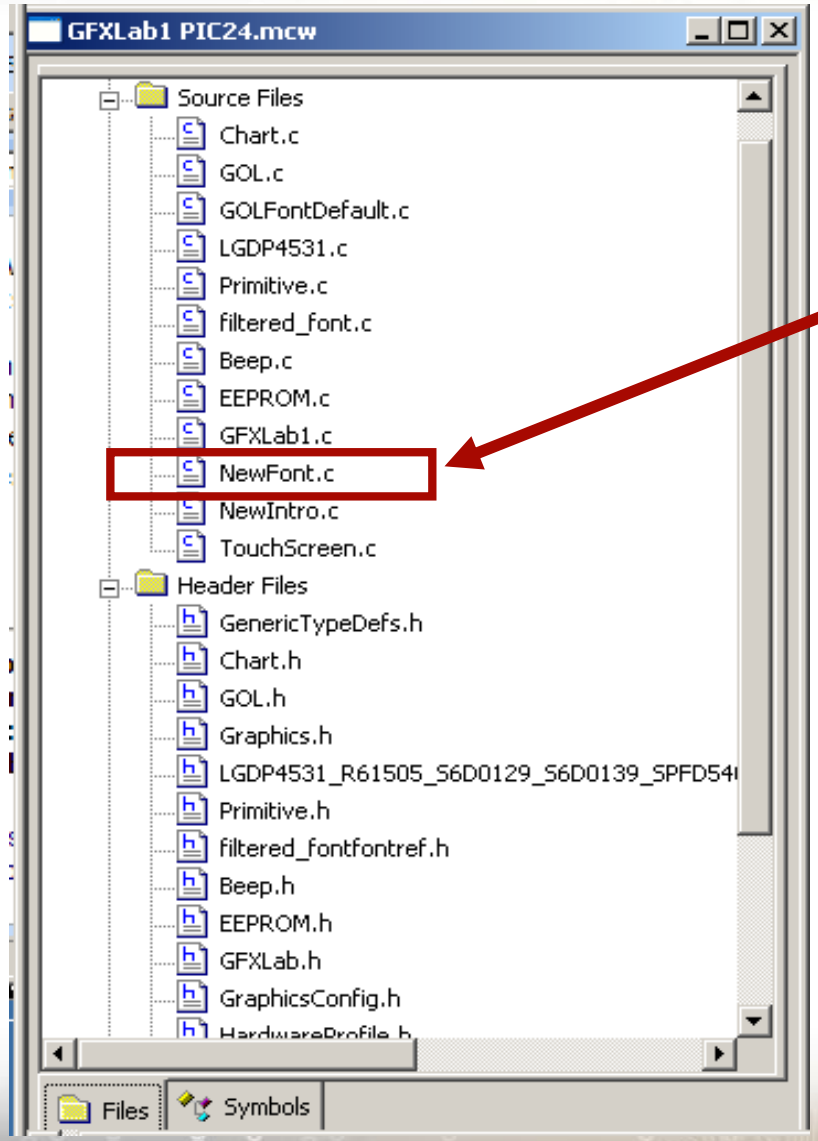
- Используется чтобы уменьшить память занимаемую шрифтом
  - Глифам назначаются новые коды
  - Требуется создать текстовый файл фильтра (font filter file)
  - Все сгенерированные файлы должны быть добавлены в проект



Формат Файла Фильтра

- Редактор должен поддерживать16-bit unicode формат
- Каждая линия должна иметь 3 секции:
  - `<String Label>:<String>//<comments>`
    - “//” разделитель необходим
    - Комментарии необязательны
- После редактирования строк шрифт должен быть сгенерирован заново
  - Теряется возможность адресовать каждую букву используя UNICODE или ASCII.
- App Note 1182 посвящена этой теме

Использование шрифтов внутренняя память



Добавьте файл в проект

Использование Шрифта

- Main.c – объявите изображения шрифтов
  - Имя должно совпадать с использованным при его создании

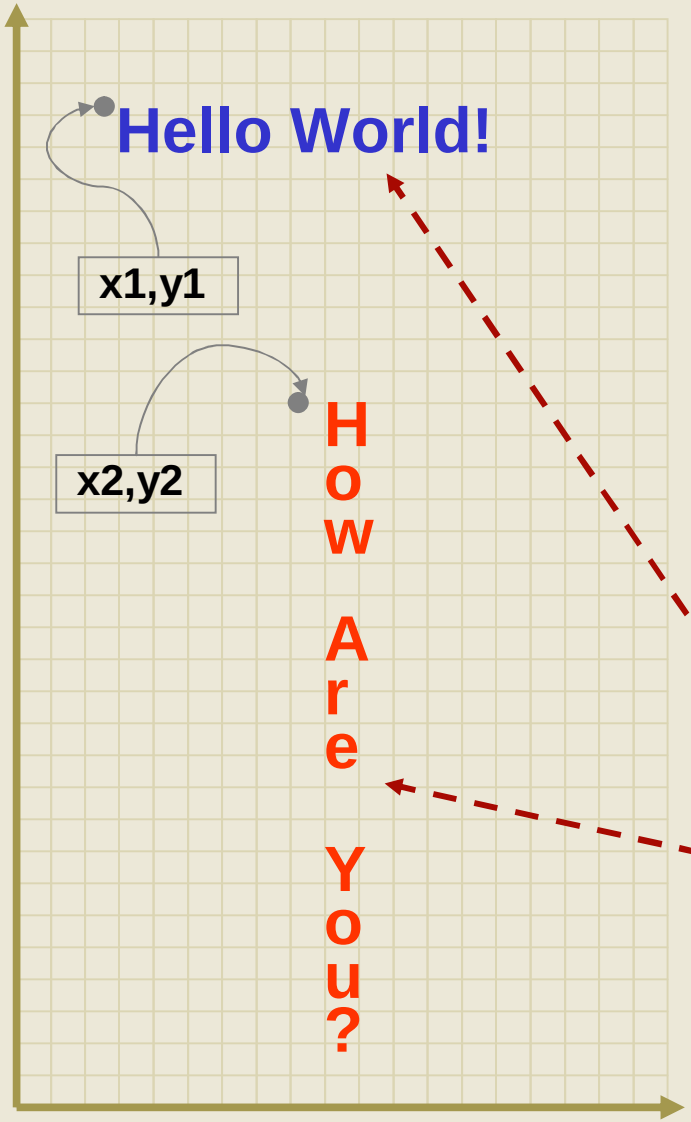
```
////////////////////// FONTS AND BITMAPS ///////////////////////
// This font is located in internal flash
extern const FONT_FLASH MyNewFont;

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

ИЛИ

- NewFont.c – измените имя в файле шрифта

```
extern const char L11298[] __attribute__((aligned(2)));
//NAME CAN BE CHANGED HERE.
const struct{short mem; const char* ptr;} MyNewFont =
{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 Шрифта




```
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);
….
}
```

Microchip Graphics
Уровень примитивов – Изображения и Иконки

Изображения

Определение:

Битмар это точечное представление графического изображения в памяти. Значение каждой точки храниться в одном или нескольких битах как определено глубиной цвета или bpp (bits per pixel).

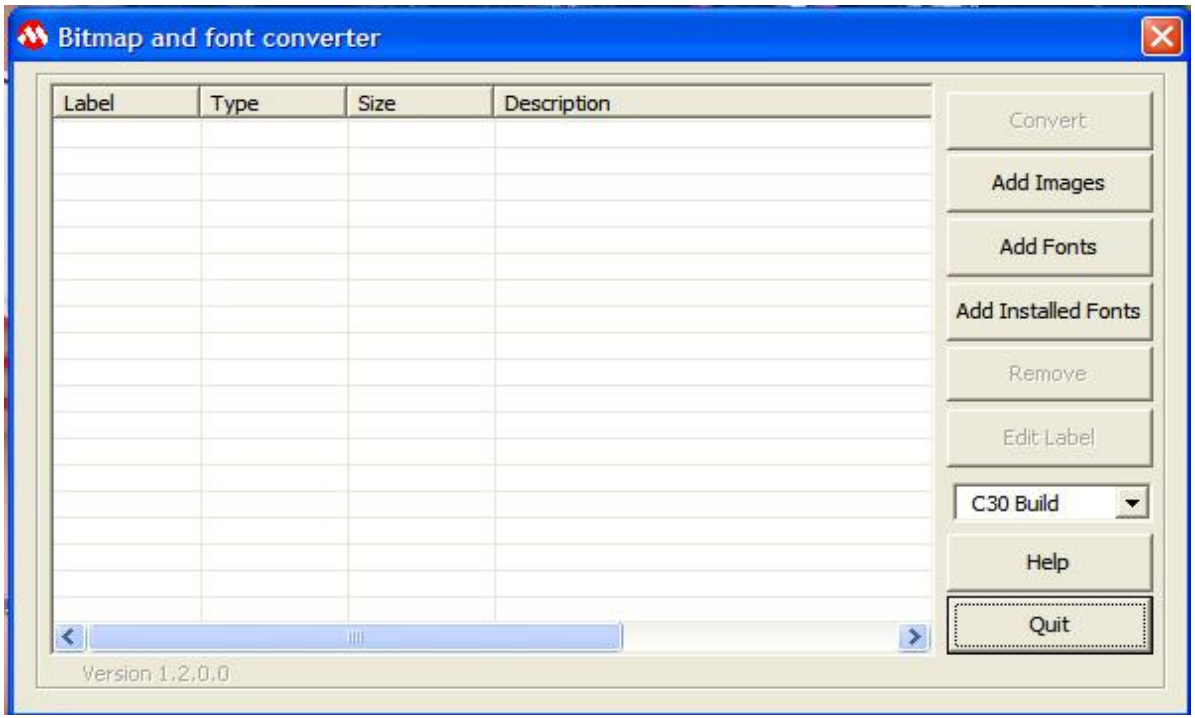
1 bpp

4 bpp

8 bpp

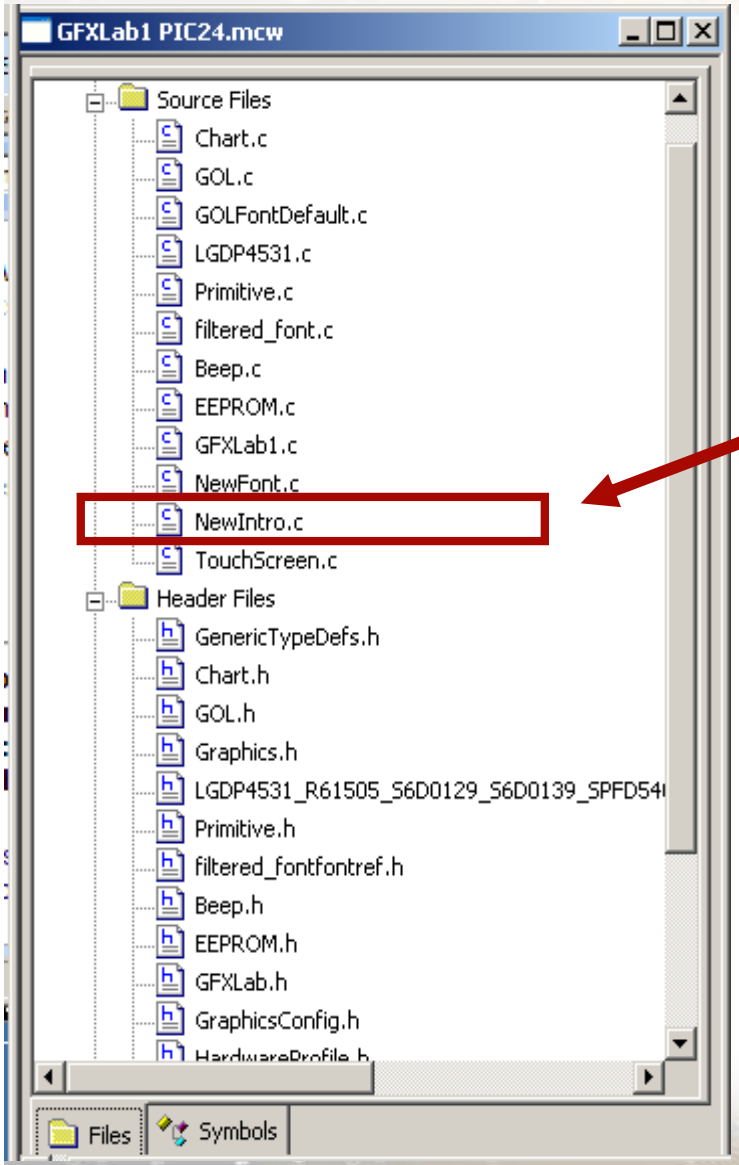
16 bpp

Преобразование изображений



1) Выберите файл изображения “Add Images” (только .bmp формат)
2) И нажмите “Convert”
3) Для хранения во внешней памяти, сохраните в .hex файле
4) Для внутренней памяти, сохраните в .c файле

Использование изображений (внутренняя память)



Добавьте полученные файлы к проекту

Использование Изображений

- Main.c – объявите ссылки на изображение
  - Имя должно совпадать с использованным при создании

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

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

ИЛИ

- NewIntro.c – измените имя в файле изображения

```
extern const char L11298[] __attribute__((aligned(2)));
//NAME CAN BE CHANGED HERE.
const struct{short mem; const char* ptr;} mchpLogo =
{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 изображения


```
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), &image1, stretch);
….
}
```

Предопределены следующие коэффициенты масштаба:
IMAGE_NORMAL
IMAGE_X2

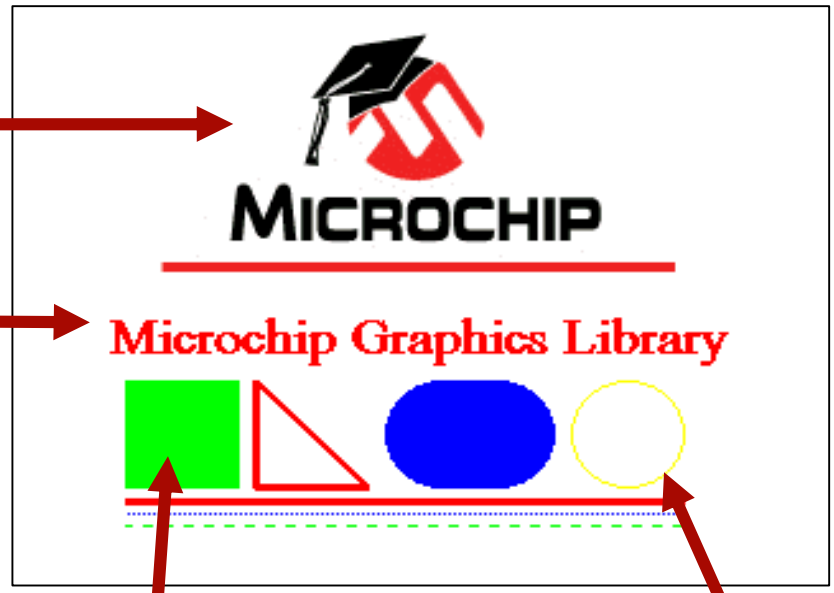
В Лабораторной Работе № 1 вы научитесь

Создавать Экран-Заставку

(применять картинки, шрифты, базовые фигуры)

Лабораторная работа 1
Создание Экрана-Заставки

Вы научитесь создавать вот такую заставку



mchpLogo.bmp

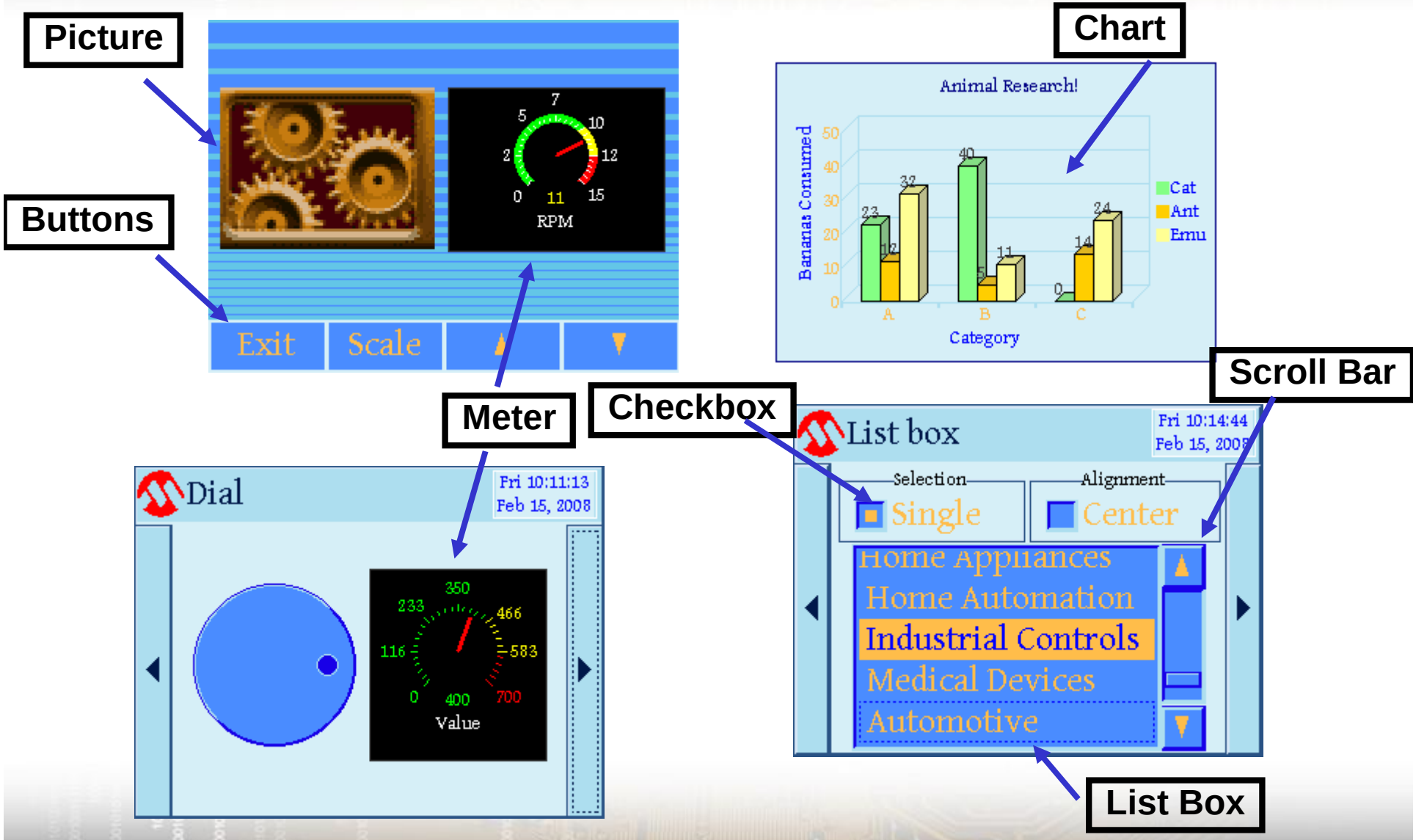
Текст с выбранным шрифтом

Прямоугольники

Окружность

Microchip Graphics
Создание виджетов (элементов управления)

Примеры библиотечных элементов управления



Создание виджетов

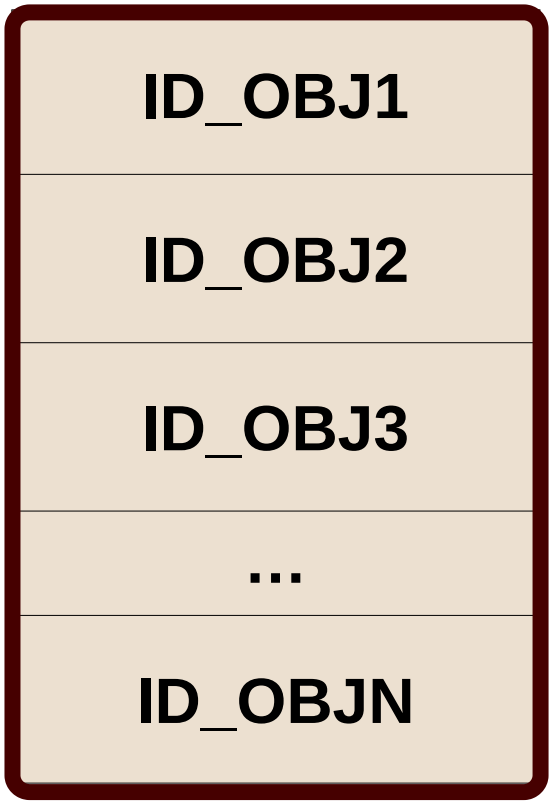
Определение:

`ObjCreate(,,)` это функция используемая библиотекой чтобы создать виджет с определенными параметрами. Эта функция автоматически заполняет структуру для него и добавляет виджет в глобальный связанный список используемых виджетов. Возвращает указатель на созданный обект.

- Каждый виджет имеет свою Create функцию
- `OBJ` = сокращенное имя виджета
  - Описано в файле помощи
  - Например: `BtnCreate(,,,)` создает кнопку
  - Например: `PbCreate(,,,)` создает progress bar
- Можно создать несколько объектов одного типа

Создание виджетов

Связанный список



- `ObjCreate(,,,)`
  - Заполняет структуру объекта
  - Добавляет к хвосту списка
  - Требуется Heap

В коде приложения …

- Пример:

```
void CreateButtons(void)
{
…

#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 Виджетов
Общие параметры

- *ID*
  - Уникальный номер используемый для ссылок на объект
- *Положение*
  - Top, left, bottom, right задают положение
- *Биты состояния*
  - Используются чтобы управлять виджетом
- *Стиль*
  - Определяет вид объекта

Microchip Graphics
Стили для виджетов

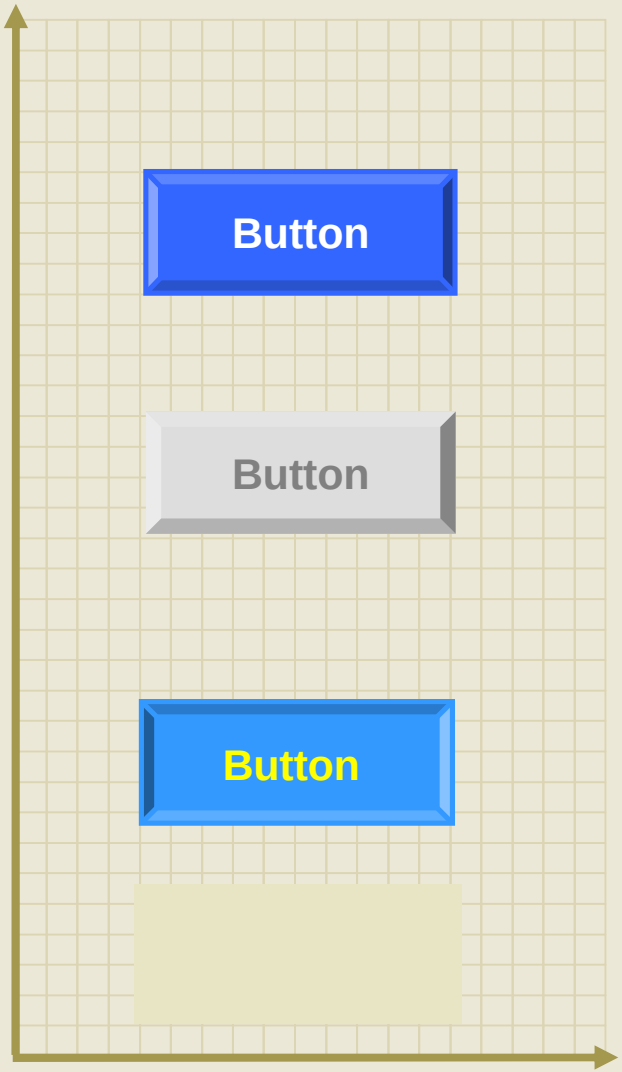
Стили для виджетов

Определение:

Стиль это структура используемая библиотекой чтобы задать вид объекта назначив цвета и шрифт.

- Несколько стилей могут быть созданы одновременно
- Стиль по умолчанию задан в `GOL.h`
- `GOLCreateScheme()`
  - Создает новый стиль
  - Возвращает указатель на стиль
- Структура содержит 10 членов
  - Значения по умолчанию назначаются при создании
  - Могут быть изменены напрямую
  - Каждый виджет использует стиль по своему

Структура стиля
Пример для кнопки




```
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;
…
}
```

Microchip Graphics
Рисование виджетов

Рисование виджетов

Связанный список

ID_OBJ1 -> биты состояния
ID_OBJ2 -> биты состояния
ID_OBJ3 -> биты сосояния
…
ID_OBJN -> биты состояния

- `GOLDraw()`
  - Нет входных параметров
  - Просматривает связанный список
  - Проверяет биты состояния
  - Если бит прорисовки установлен то выводит на экран
  - Возвращает TRUE когда все виджеты прорисованны

state общие поля

- Биты состояния для рисования:
  - 6 старших бита показывают что объект…
    - Невидим
    - Должен быть перерисован частично
    - Полностью перерисован
- Используются всеми виджетами:
  - OBJ_HIDE
    - Закрашивает область виджета CommonBkGnd цветом
  - OBJ_DRAW
    - Перерисовывает виджет целиком
    - Очищается автоматически менеджером рисования GOLDraw()
  - OBJ_DRAW_FOCUS:
    - Перерисовывает только фокус

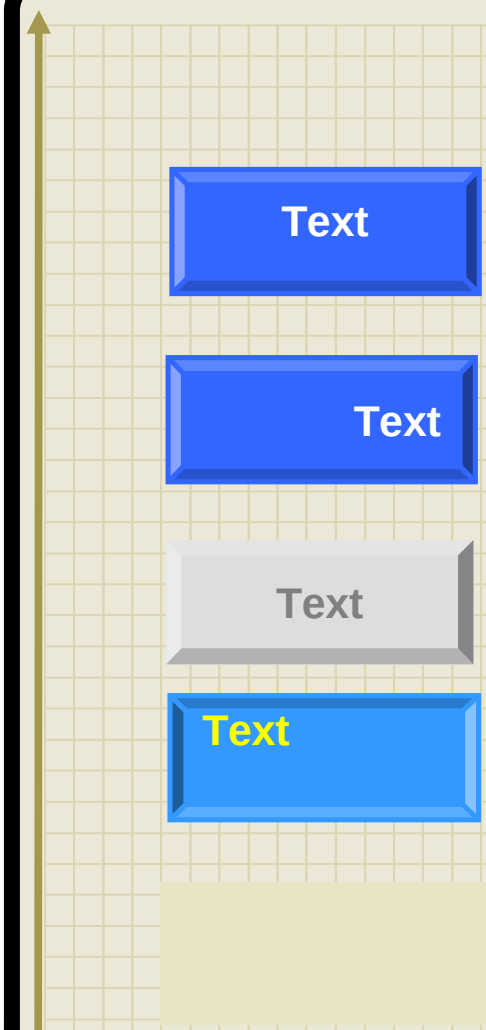
`state` общие биты

- Биты состояния определяющие свойства:
  - 10 младших битов
  - Определяют реакцию на сообщения и вид
  - Не изменяются автоматически `GOLDraw()`
- Используется некоторыми виджетами:
  - `OBJ_FOCUSED`: виджет в фокусе
- Используется всеми виджетами:
  - `OBJ_DISABLED`: виджет выключен
    - Все сообщения будут игнорироваться

Макросы для управления битами состояния

- `SetState(pObj, state)`
- `ClrState(pObj, state)`
- `GetState(pObj, state)`
- `GOLFindObject(ID)`
- **Where…**
  - `pObj` = указатель на виджет
  - `state` = ИЛИ комбинация битов

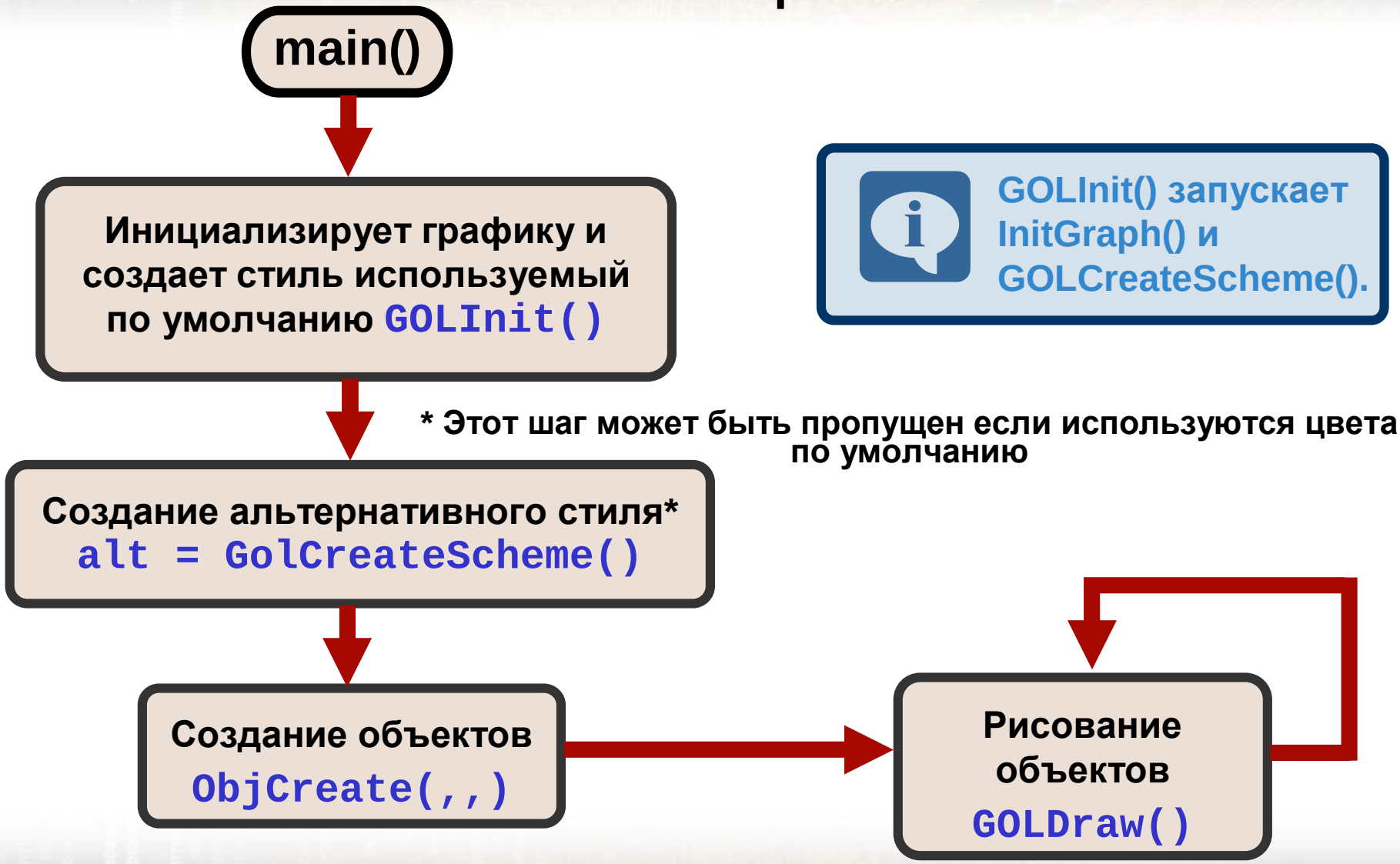
Биты состояния Пример



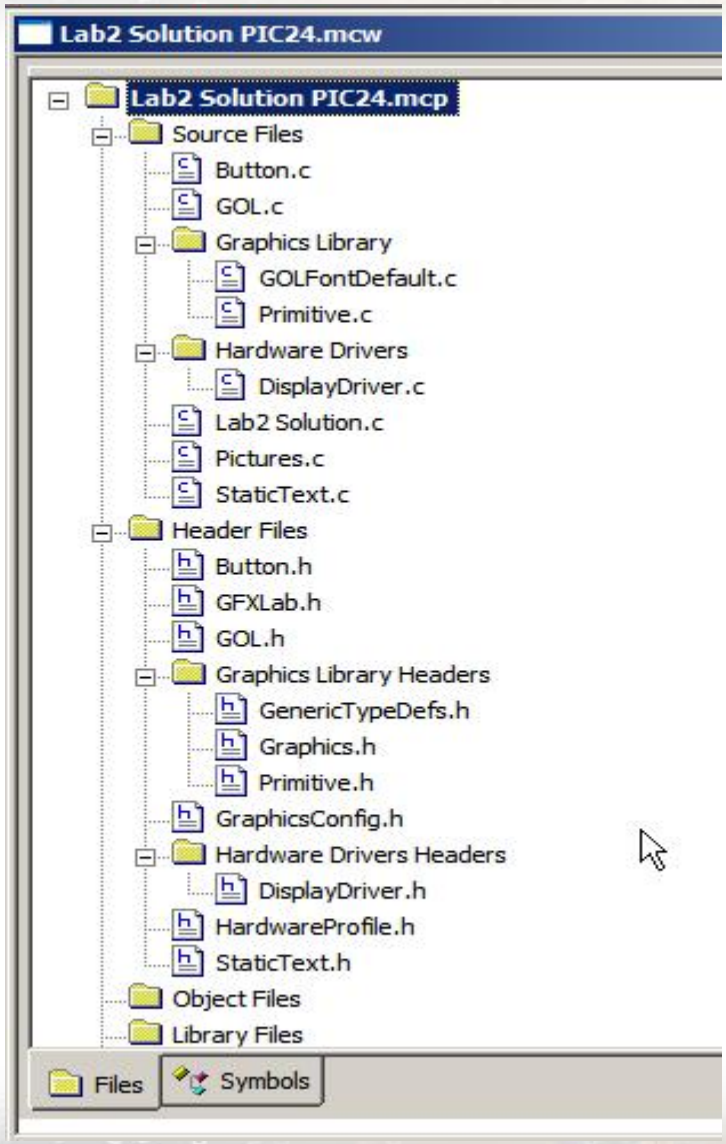

```
if (GOL_DRAW())
{
// Example Button text alignments
SetState(pBtn, BTN_TEXTTOP);
SetState(pBtn, BTN_TEXTRIGHT);

// Example Button Focus
SetState(pBtn, BTN_FOCUSED);

// Example Button Action
SetState(pBtn, BTN_DISABLED);
state = BTN_PRESSED|BTN_TEXTTOP|BTN_TEXTLEFT;
SetState(pBtn, state);

// Example Hiding Button
SetState(pBtn, BTN_HIDE)
}
```

Пример рисования

Типичное приложение



Использование виджетов



- **Добавьте необходимые файлы в проект:**
  - GOL.c
  - Primitive.c
  - *widget*.c
  - DisplayDriver.c
  - GOL.h
  - Graphics.h
  - GenericTypeDefs.h
  - Primitives.h
  - *widget*.h
  - DisplayDriver.h
  - GraphicsConfig.h
  - Plus Font and Bitmap files

Использование виджетов

- Разрешите нужные виджеты в `GraphicsConfig.h`

```
#define USE_GOL
#define USE_BUTTON              // Enable Button Object.
//#define USE_WINDOW            // Enable Window Object.
//#define USE_CHECKBOX          // Enable Checkbox Object.
//#define USE_RADIOBUTTON       // Enable Radio Button Object.
//#define USE_EDITBOX           // Enable Edit Box Object.
//#define USE_LISTBOX           // Enable List Box Object.
//#define USE_SLIDER            // Enable Slider or Scroll Bar Object.
//#define USE_PROGRESSBAR       // Enable Progress Bar Object.
#define USE_STATICTEXT          // Enable Static Text Object.
//#define USE_PICTURE           // Enable Picture Object.
//#define USE_GROUPBOX          // Enable Group Box Object.
//#define USE_ROUNDDIAL         // Enable Dial Object.
//#define USE_METER             // Enable Meter Object.
//#define USE_GRID              // Enable Grid Object.
//#define USE_CHART             // Enable Chart Object
//#define USE_CUSTOM            // Enable Custom Control Object
```

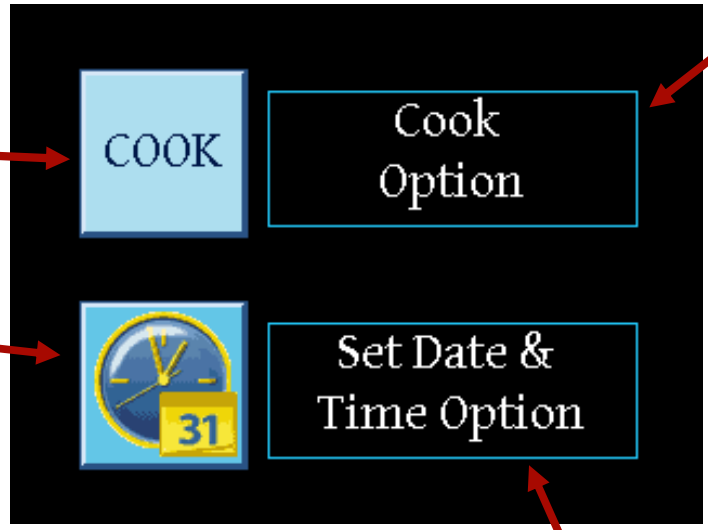
В Лабораторной Работе № 2 вы научитесь:

Создавать простое меню для приложения

Лабораторная работа 2

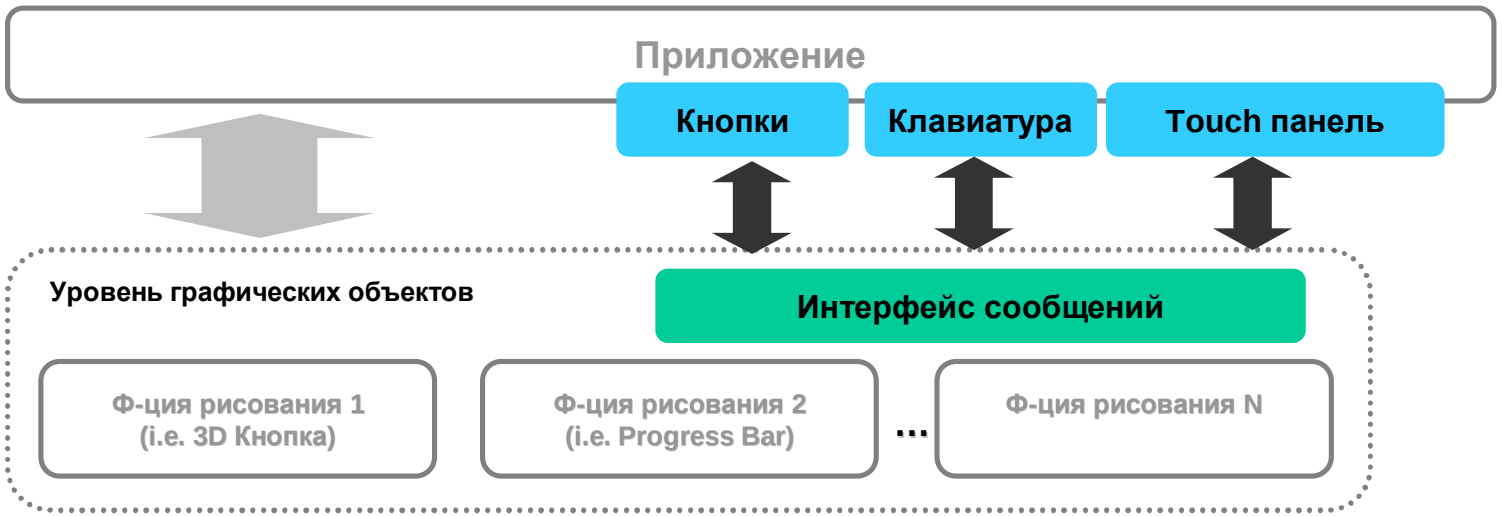
Создание меню

Вы научитесь создавать вот такое простое меню



Интерфейс с пользователем
Интерфейс сообщений

Интерфейс сообщений



- Упрощает интеграцию устройств ввода
- Позволяет эффективно управлять виджетами
- В будущем будет добавлена поддержка других устройств ввода (e.g., mouse)

Интерфейс с устройством ввода

- Получить событие от пользователя
  - Пример драйвера для touch панели включен в библиотеку
- Заполнить структуру сообщения
- Вызвать `GOLMsg(&msg)`
  - Где `&msg` адрес структуры сообщения
  - `GOLMsg(&msg)` интерпретирует сообщение вызывает `GOLMsgCallback(…)` для обработки пользователем и обрабатывает сообщения по умолчанию

Структура сообщения

- Структура сообщения

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

- `type` = *TYPE_KEYBOARD* или *TYPE_TOUCHSCREEN*
- `param1` и `param2` зависят от типа
  - Для touch панели:
    - `param1`: x координата
    - `param2`: y координата
  - Для клавиатуры:
    - `param1`: ID виджета получающего сообщение
    - `param2`: зависит от события

Структура Сообщения

- `uiEvent` определяет тип события
  - Для touch панели
    - `EVENT_PRESS`
    - `EVENT_RELEASE`
    - `EVENT_MOVE`
    - `EVENT_INVALID`
      - Не нажата
  - Для клавиатуры и кнопок
    - `EVENT_KEYSCAN`
      - Обычно скан код
    - `EVENT_CHARCODE` (только edit box)
      - `param2` = код символа
    - `EVENT_INVALID`
      - Не нажата

AT скан коды описаны в файле помощи.

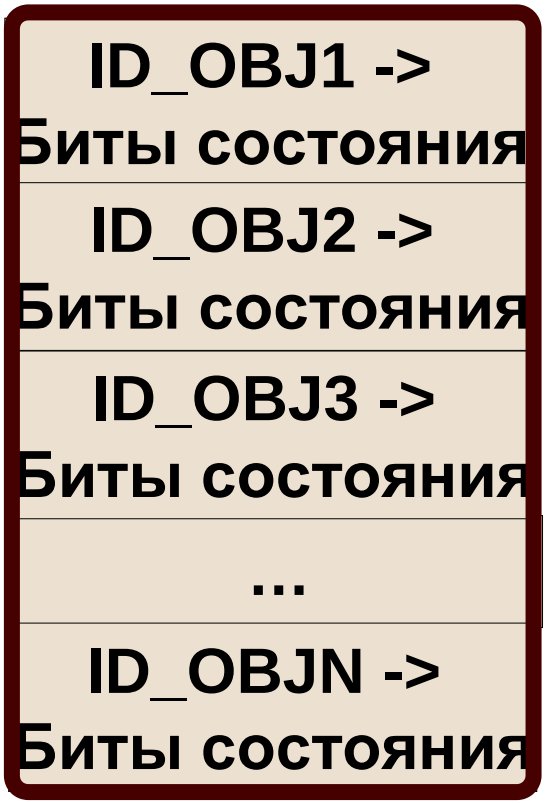
Пример
заполнения сообщения для кнопки

```
if(S5){//button is pressed
        msg->type    = TYPE_KEYBOARD;
        msg->uiEvent = EVENT_KEYSCAN;
        msg->param1  = ID;
        msg->param2  = SCAN_CR_PRESSED;
}else{//button is released
        msg->type    = TYPE_KEYBOARD;
        msg->uiEvent = EVENT_KEYSCAN;
        msg->param1  = ID;
        msg->param2  = SCAN_CR_RELEASED;
}return;
```

Интерфейс с пользователем
Обработка сообщений

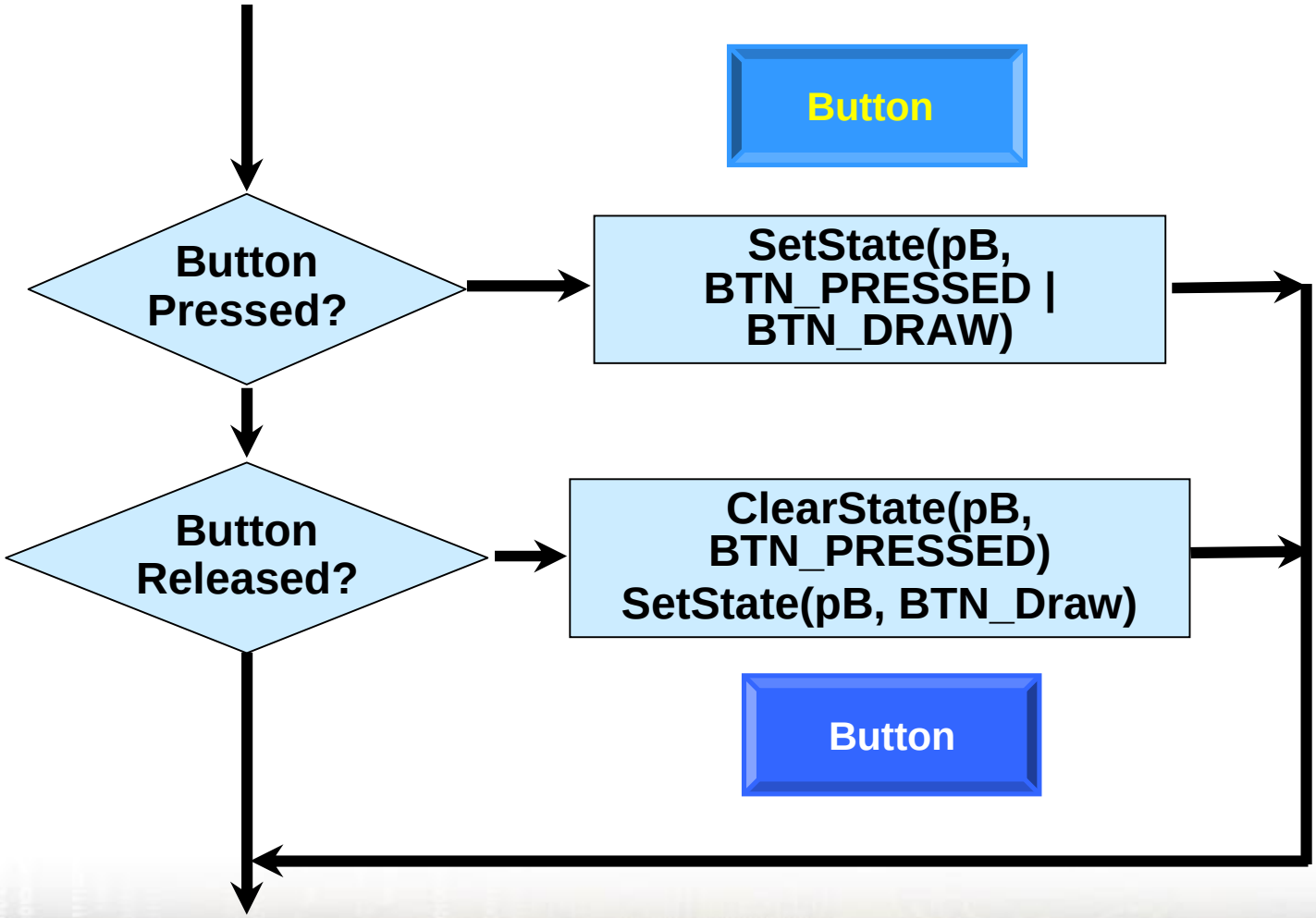
Менеджер сообщений

Связанный список



- `GOLMsg(&msg)`
  - Находит виджет который должен принять сообщение
  - Транслирует сообщение
  - Вызывает функцию `GOLMsgCallback(…)` для обработки сообщения пользователем
  - Выполняет обработку сообщения по умолчанию

Обработка сообщения по умолчанию для кнопки



Стандартное приложение

* Можно пропустить этот шаг

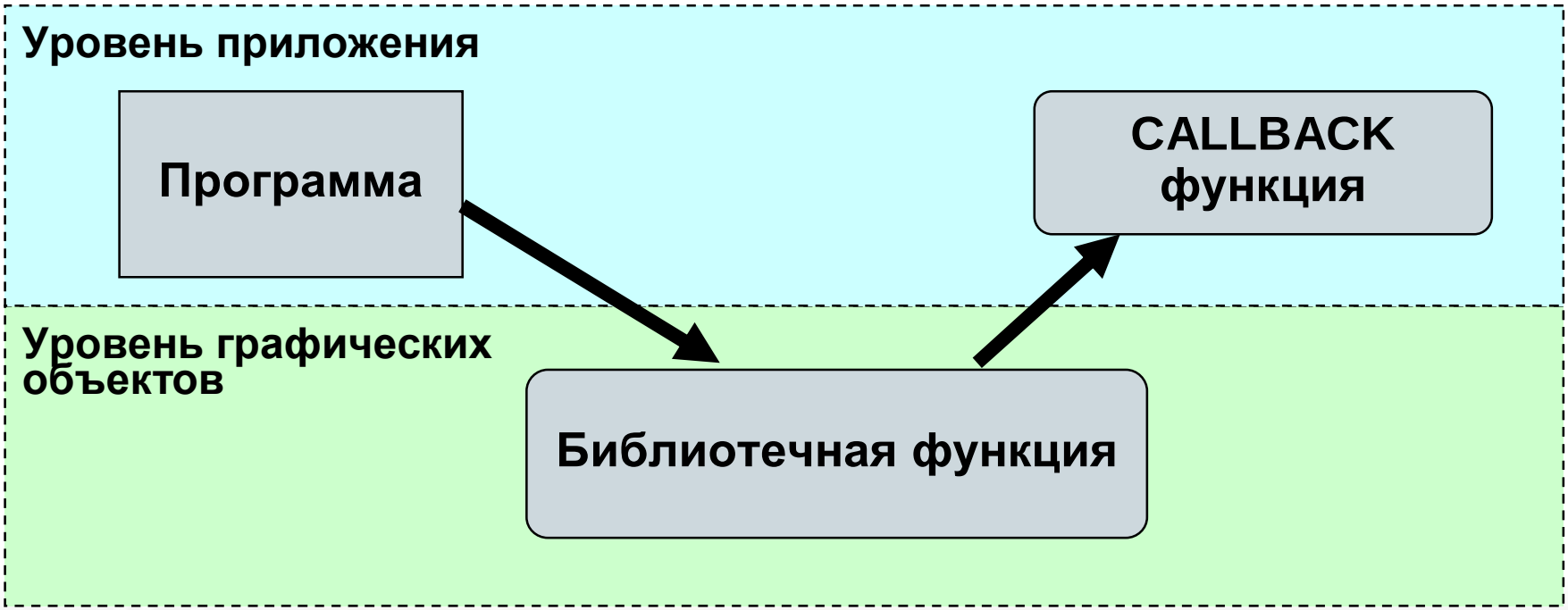
** Должна быть определена пользователем

Что такое Callback?

Определение:

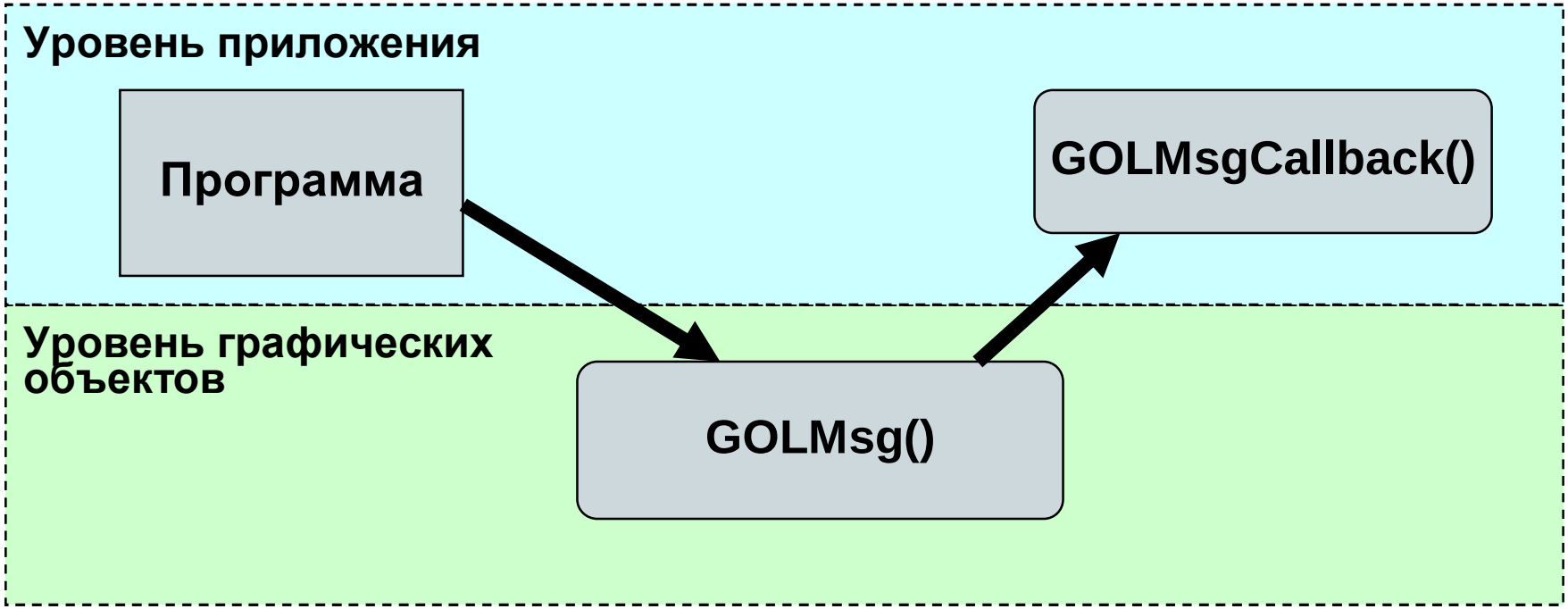
Callback функция позволяет нижнему уровню программы вызвать функцию определенную на верхнем уровне.

- **Используется чтобы написать код обрабатывающий событие для виджета**



GOLMsgCallback()

- Вызывается менеджером GOLMsg()
- Используется чтобы написать обработку сообщений



GOLMsgCallback()

- Вызывается менеджером `GOLMsg(&msg)`
- ***ДОЛЖНА*** быть определена в программе
- В ней пользователь выполняет необходимые действия:
  - Пример: Сменить картинку на кнопке когда она нажата
  - Пример: включить LED когда кнопка нажата
- Входные параметры:
  - `objMsg`: транслированное сообщение
  - `pObj`: указатель на виджет
  - `pMsg`: указатель на структуру сообщения
- Выход:
  - `TRUE`: Менеджер запустит обработку по умолчанию
  - `'0'` : Обработка по умолчанию будет пропущена

Функции для работы с виджетом

- **Виджет имеющий текст:**
  - `ObjSetText(*pObj, *pText)`
  - `ObjGetText(*pObj)`
- **Виджет с изображением:**
  - `ObjSetBitmap(*pObj, *pBitmap)`
  - `ObjGetBitmap(*pObj)`
- **Специальные функции:**
  - **Примеры:**
  - `PbSetPos(*pObj)` **– увеличивает и уменьшает позицию для progress bar**
  - `MtrIncVal(*pObj), MtrDecVal(*pObj)` **– увеличивает и уменьшает позицию для стрелочного индикатора**

GOLMsgCallback()

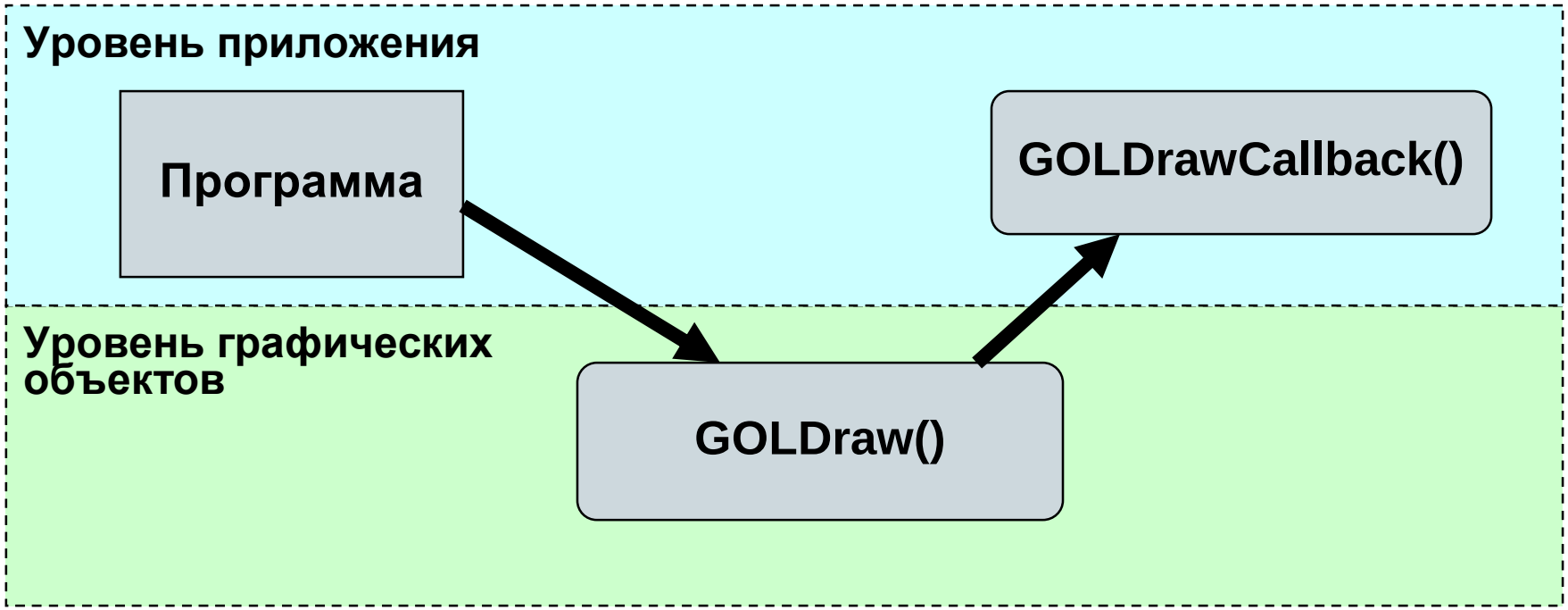
Пример


```
WORD GOLMsgCallback(objMsg,pObj,pMsg){
…
objID = GetObjID(pObj);
pSldObj = (*SLIDER)GOLFindObj(SLD1);
  if(objID == BTN1){
    if (objMsg == BTN_MSG_PRESSED){
      BtnSetBitmap(pObj, &blueDOWNarrow);
      SetState(pBTN1, BTN_TEXTBOTTOM);
      SldDecPos(pSldObj);
      SetState(pSldObj, SLD_DRAW_THUMB);
      SlowMotor();}
    else{
      BtnSetBitmap(pObj, NULL);
      ClrState(pObj, BTN_TEXTBOTTOM);}}
…
}
```

Интерфейс с пользователем GOLDrawCallback()

GOLDrawCallback

- Вызывается менеджером рисования GOLDraw()



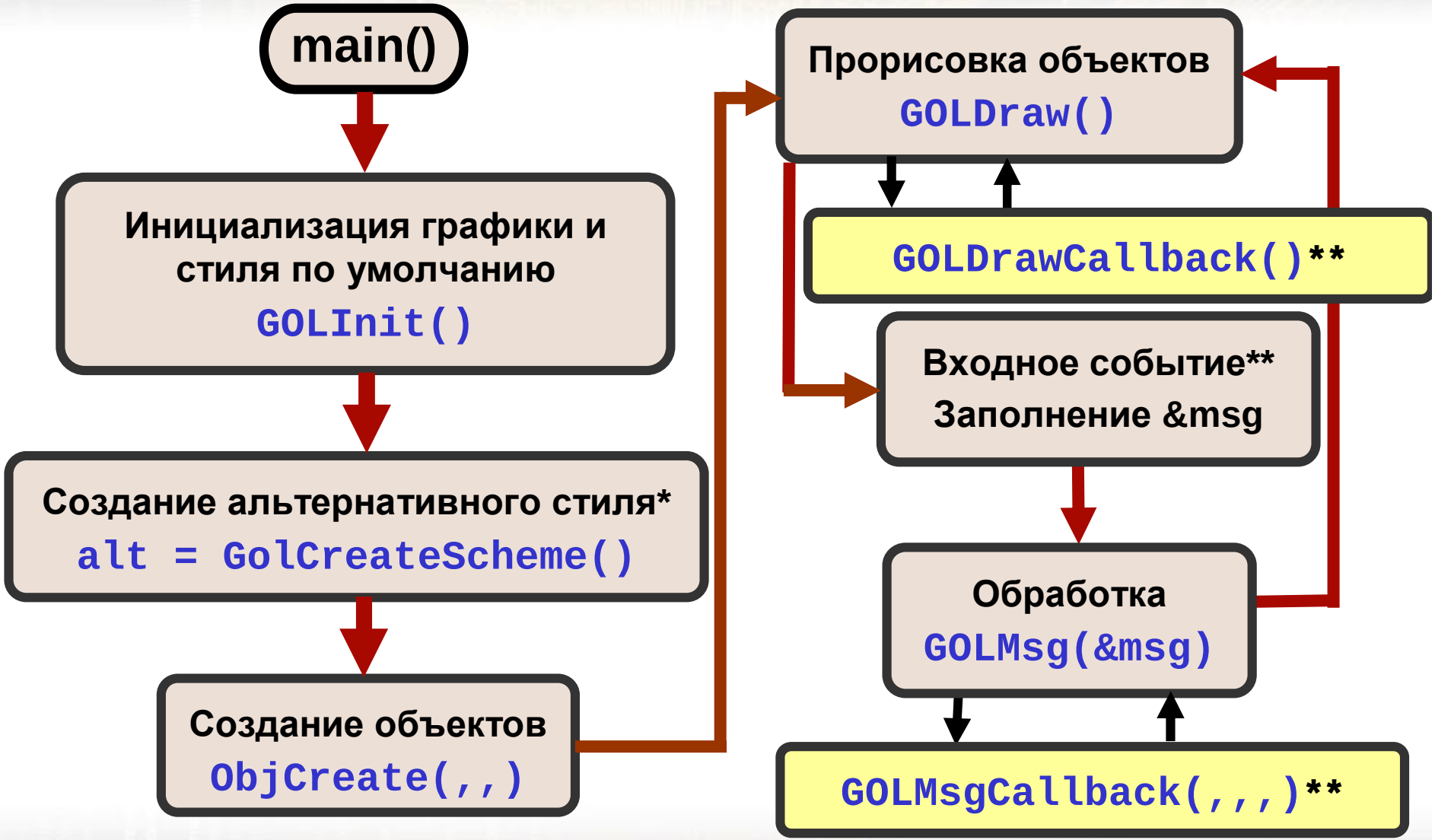
Customized Drawing
GOLDrawCallback()

- Вызывается менеджером GOLDraw()
  - Прорисовка всех виджетов закончена
- *ДОЛЖНА* быть определена пользователем
  - Если возвращает TRUE то управление переходит обратно к менеджеру GOLDraw()
  - Если возвращает ‘0’ пользователь не закончил рисование
- Здесь пользователь может рисовать дополнительные элементы
- Единственное безопасное место чтобы:
  - Изменить параметры рисования
  - Изменить или создать новый связанный список виджетов

Создание экрана

- Используйте `GOLFree()` чтобы удалить текущий связанный список
  - Удаляет все виджеты из heap памяти
- Используйте `ObjCreate(,,,)` чтобы сформировать новый список
- Установите размер heap для экрана с самым большим количеством виджетов

Стандартное приложение



* Этот шаг может быть пропущен

** Долны быть определены пользователем

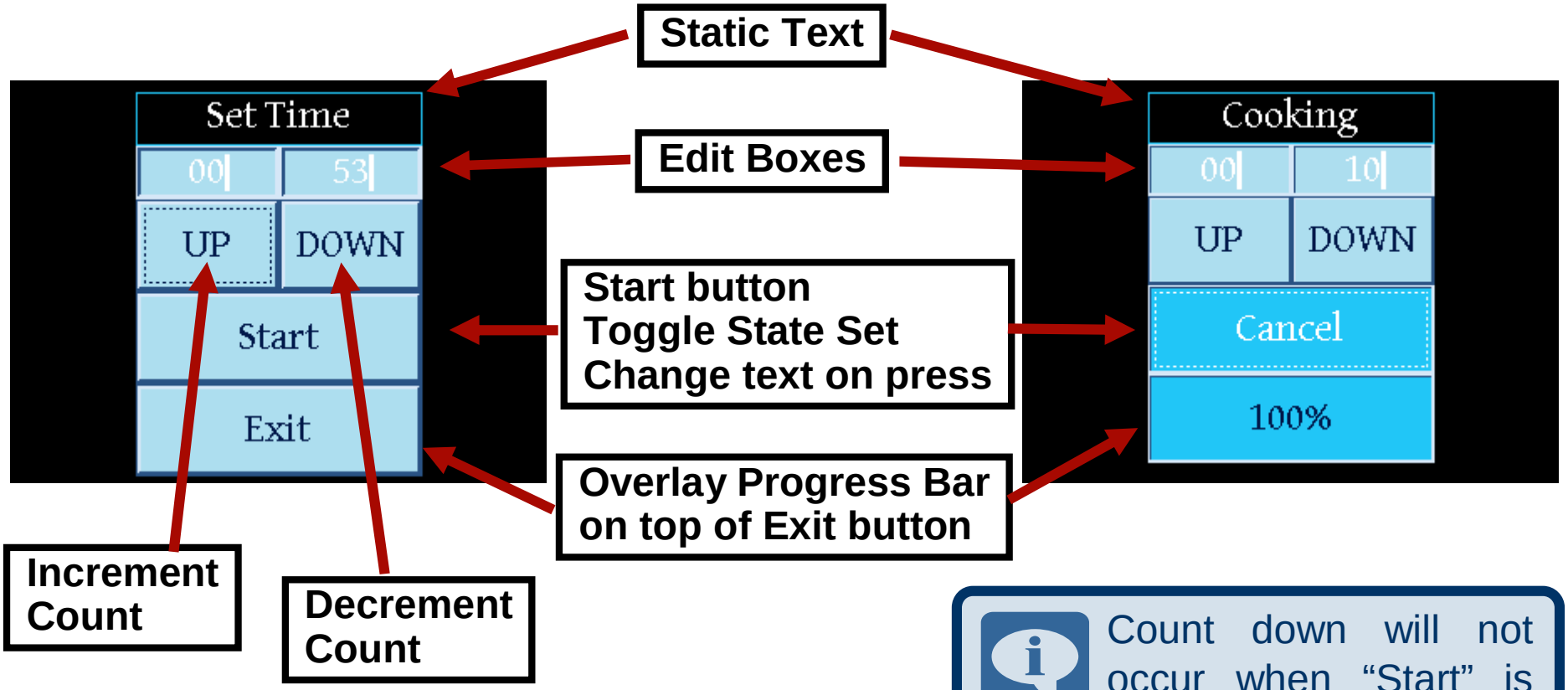
В Лабораторной Работе № 3 вы научитесь:

Обмену сообщениями между объектами библиотеки

Лабораторная работа 3

Управление печкой

- ***Вы научитесь обмениваться сообщениями между графическими объектами***



Count down will not occur when “Start” is pushed. We implement this functionality later.