emWin

How To Migrate from emWin V4 to emWin V5

Document revision 0

Date: April 23, 2010

SEGGER

A product of SEGGER Microcontroller GmbH & Co. KG

www.segger.com

Disclaimer

Copyright notice

Trademarks

Contact address

SEGGER Microcontroller GmbH & Co. KG

In den Weiden 11
D-40721 Hilden

Germany

Tel.+49 2103-2878-0
Fax.+49 2103-2878-28
E-mail: support@segger.com
Internet: http://www.segger.com

Guide versions

This guide describes the simplest way to migrate from emWin V4 to emWin V5. If any error occurs, please inform us and we will try to assist you as soon as possible.

Guide version	Date	By	Explanation
Rev. 0	100311	JE	Initial version

Software versions

Refer to Release.html for information about the changes of the software versions.

About this document

Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (assembler, linker, C compiler)
- The C programming language
- The target processor
- DOS command line

If you feel that your knowledge of C is not sufficient, we recommend The C Programming Language by Kernighan and Richie (ISBN 0-13-1103628), which describes the standard in C-programming and, in newer editions, also covers the ANSI C standard.

How to use this manual

This manual explains all the functions and macros that the product offers. It assumes you have a working knowledge of the C language. Knowledge of assembly programming is not required.

Typographic conventions for syntax

This manual uses the following typographic conventions:

Style	Used for
Body	Body text.
`Keyword`	Text that you enter at the command-prompt or that appears on the display (that is system functions, file- or pathnames).
`Parameter`	Parameters in API functions.
`Sample`	Sample code in program examples.
Reference	Reference to chapters, sections, tables and figures or other documents.
GUIElement	Buttons, dialog boxes, menu names, menu commands.
Emphasis	Very important sections

Table 1.1: Typographic conventions

Table of Contents

Chapter 1

Introduction

This file describes the differences between emWin version 4.xx and 5.xx and what need to be considered when migrating to the new version. It should be a guideline for the migration. This file should mention the differences and refer to detailed explanations in the documentation. It does not contain a detailed instruction list for the process of migration.

1.1 General Differences

The main difference between the old and the new version is that the driver interface has been changed. Older drivers, for V4.18 or earlier, can no longer be used. We changed the interface to be able to support run-time configuration for new display drivers. This was required because many of our customers work with precompiled libraries which should not be changed when using a different display. We also changed the internal interaction of display drivers, memory devices, sprites and other items and the memory configuration.

Chapter 2

Configuration Files

This chapter gives a short description on how to handle the emWin configuration files in V5.

2.1 GUITouchConf.h, obsolete

No longer used. Only runtime configuration is supported.

2.2 GUIConf.h

Same purpose as in older versions with. Please note compile time configuration of available memory is no longer supported.

2.3 LCDConf.h

Display driver configuration has been changed from compile time configuration (*.h) to run time configuration (*.c). So this file can remain empty when using a newer driver, unless the display driver description in the emWin documentation contains different demands.

2.4 GUIConf.c, new in version 5

This file contains the memory configuration routine called during the initialization process. In earlier versions the available memory for emWin was configured with the compile time configuration macro `GUI_ALLOC_SIZE`. The new version requires the routine `GUI_X_Config()` which is located in this file per default.

2.5 LCDConf.c

This file contains the display driver configuration routine `LCD_X_Config()` and the display driver callback function `LCD_X_DisplayDriver()`.
`LCD_X_Config()` is called during the initialization process of emWin. The task of the routine is to create and configure a display driver device.
The callback function `LCD_X_DisplayDriver()` is called later by the driver for several purposes, among others for initializing the display controller.

Chapter 3

Configuration

This chapter describes the configurable parts, which have changed in emWin V5.

3.1 Features & Defaults

Not all configuration macros of the older versions are available in the new version. For the currently available configuration macros please refer to the documentation chapter '28.5 Compile time configuration'. It contains a table with the currently available configuration macros except the available widget configuration macros. The widget configuration has not been changed.

3.2 Memory

As mentioned above the memory configuration is done in the routine `GUI_X_Config()`. It replaces the use of the macros `GUI_ALLOC_SIZE` and `GUI_MAXBLOCKS`. The routine is located by default in the file `GUIConf.c` but can be stored anywhere else if required. A detailed description of `GUI_X_Config()` can be found in chapter '28.4 Run-time configuration' of your emWin documentation.

3.3 Display Driver

In older versions the configuration of the driver(s) to be used was done in `LCDConf.h` and in `LCDConf.c`. The file `LCDConf.h` contained the macro `LCD_CONTROLLER` which determined the driver to be used. The new version does not 'know' anything of the driver to be used. Instead of using a compile time macro for determining the driver to be used it calls the routine `LCD_X_Config()` which has to create and configure a 'display driver device' for each layer. It replaces the use of several configuration macros like `LCD_CONTROLLER`, `LCD_BITSPERPIXEL`, `LCD_XSIZE`, `LCD_YSIZE` and others. For a detailed description of the routine `LCD_X_Config()` please refer to the documentation chapter '28.4.2 Customizing LCDConf.c'.

The display driver initialization of older versions has been done with the macro `LCD_INIT_CONTROLLER`, which called the function `LCD_X_InitController()` per default. Instead of this the display drivers of the new version use the callback routine `LCD_X_DisplayDriver()` for initializing the controller. The driver passes a command to this routine which has to be processed by the customer. For initialization purpose the command `LCD_X_INITCONTROLLER` is passed to the routine.

3.4 Touch Screen Driver

Compile time configuration is no longer supported. Instead of this the functions `GUI_TOUCH_Calibrate()` and `GUI_TOUCH_SetOrientation()` need to be used. For a detailed description of these functions please refer to the documentation chapter '19.4.2.4 Driver API for analog touch-screens'.

Chapter 4

Migration Steps

This chapter describes the migration steps, which have to be done to have your application completely set up with emWin V5.

4.1 Adapting GUIConf.h

The purpose of this file is the same as in older versions. It mainly contains the configuration of the available features. Please refer to the documentation chapter '28.5 Compile time configuration' for the available configuration options.

4.2 Adapting LCDConf.h

This file can remain empty in most cases, because the driver configuration and initialization is done in the routines `LCD_X_Config()` and `LCD_X_DisplayDriver()`. For details please refer to the display driver description in the documentation.

4.3 Adapting GUI_X_Config() in GUIConf.c

The one and only purpose of this routine is to spend dynamically memory to emWin. Lets take a look to the routine:

```
#define GUI_NUMBYTES  XXXX
#define GUI_BLOCKSIZE 32
void GUI_X_Config(void) {
  static U32 aMemory[GUI_NUMBYTES / 4];
  GUI_ALLOC_AssignMemory(aMemory, GUI_NUMBYTES);
  GUI_ALLOC_SetAvBlockSize(GUI_BLOCKSIZE);
}
```

GUI_NUMBYTES

The right value can be taken from GUIConf.h of the old configuration.

GUI_BLOCKSIZE

The default value is 32 bytes but can be changed on demand. For details please refer to the documentation.

4.4 Adapting LCD_X_Config() in LCDConf.c

As mentioned above the purpose of this routine is to setup a display driver device. Lets take a look to the routine:

```
void LCD_X_Config(void) {
  GUI_DEVICE * pDevice;
  pDevice = GUI_DEVICE_CreateAndLink(DISPLAY_DRIVER, COLOR_CONVERSION, 0, 0);
  if (LCD_GetSwapXYEx(0)) {
    LCD_SetSizeEx (0, YSIZE_PHYS,  XSIZE_PHYS);
    LCD_SetVSizeEx(0, VYSIZE_PHYS, VXSIZE_PHYS);
  } else {
    LCD_SetSizeEx (0, XSIZE_PHYS,  YSIZE_PHYS);
    LCD_SetVSizeEx(0, VXSIZE_PHYS, VYSIZE_PHYS);
  }
}
```

DISPLAY_DRIVER

The right define for the macro `DISPLAY_DRIVER` can be found in the documentation or in the configuration sample file of the according driver.

COLOR_CONVERSION

The right define for the color conversion macro `COLOR_CONVERSION` should be one of the table 'Fixed palette modes' in the documentation in the chapter '11 Colors'. It can be derivatized from the old configuration as follows:

```
GUICC_[_MIRROR_][_CONVERSION_]
```

Variable	Description
`_MIRROR_`	Should be the letter 'M' if `LCD_SWAP_RB` was set to 1 in the old configuration file `LCDConf.h`
`_CONVERSION_`	Should be the same as the macro `LCD_FIXEDPALETTE` of the old configuration file `LCDConf.h` or the default fixed palette mode of the used color depth

4.5 Adapting LCD_X_DisplayDriver() in LCDConf.c

This routine is called by the display driver for several jobs, among others to initialize the display driver. Lets take a look to the routine:

```
int LCD_X_DisplayDriver(unsigned LayerIndex, unsigned Cmd, void * pData) {
  int r;
  switch (Cmd) {
  case LCD_X_INITCONTROLLER:
    //
    // This comment need to be replaced by a function call
    // to the customers display driver initialization routine
    //
    return 0;
  default:
    r = -1;
  }
  return r;
}
```

The only thing which need to be done is calling the display controller initialization routine by replacing the comment.