M480 emWin Quick Start Guide

Document Information

Abstract	Introduce the steps to build and launch emWin for the M480 series microcontroller (MCU).
Apply to	NuMicro® M480 series

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 Introduction

emWin is a graphic library with graphical user interface (GUI) designed to provide an efficient, processor and display controller-independent GUI for any application that operates with a graphical display.

Nuvoton provides emWin GUI library for free with the M480 series microcontroller (MCU) supporting up to 320x240 (16 bpp) resolution. The emWin platform can be implemented on HMI for industrial, machines, appliances, etc.

2 emWin BSP Directory Structure

This chapter introduces emWin related files and directories in the M480 BSP.

2.1 Sample Codes (sampleCode)

emWin_GUIDemo	Utilize emWin library to demonstrate widgets feature.
emWin_SimpleDemo	Utilize emWin library to demonstrate interactive feature.

2.2 Configuration Files (ThirdParty\emWin\Config)

GUI_X.c	Configuration and system dependent code for GUI.
GUIConf.c	Display controller initialization source code.
GUIConf.h	A header file configures emWins features, fonts, etc.
LCDConf.c	Display controller configuration source code.
LCDConf.h	Display driver configuration header file.

2.3 Documents (ThirdParty\emWin\Doc)

AN03002_Custom_ Widget_Type.pdf	emWin custom widget type creation guide.
UM03001_emWin5.pdf	emWin user guide and reference manual.

2.4 Include Files (ThirdParty\emWin\Include)

This directory contains header files for emWin project.

2.5 Library (ThirdParty\emWin\Lib)

NUemWin_CM4_Keil.lib	emWin library for M480 series MCU.

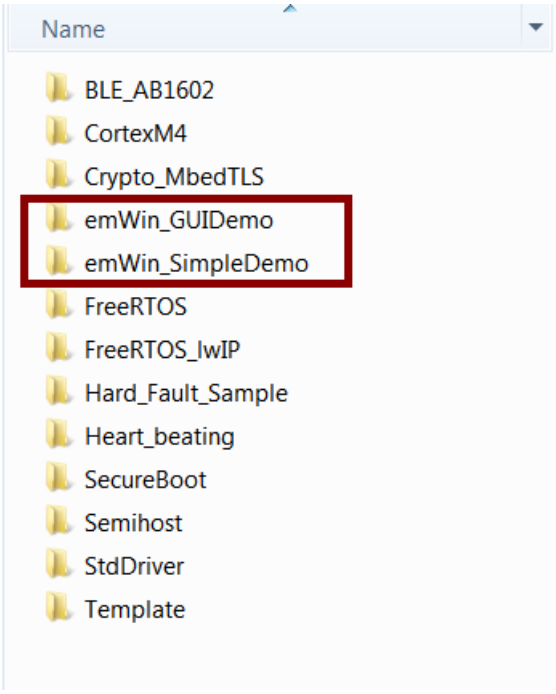
2.6 Tools (ThirdParty\emWin\Tool)

BmpCvtNuvoton.exe	The Bitmap Converter is designed for converting common image file formats like BMP, PNG or GIF into the desired emWin bitmap format.
emWinPlayer.exe	This tool can show the previously created emWin Movie File (EMF) on a Computer with a Windows operating system.
GUIBuilder.exe	A tool for creating dialogs by drag and drop operation.
JPEG2Movie.exe	A tool to convert JPEG files to an EMF file.

3 emWin Sample Code

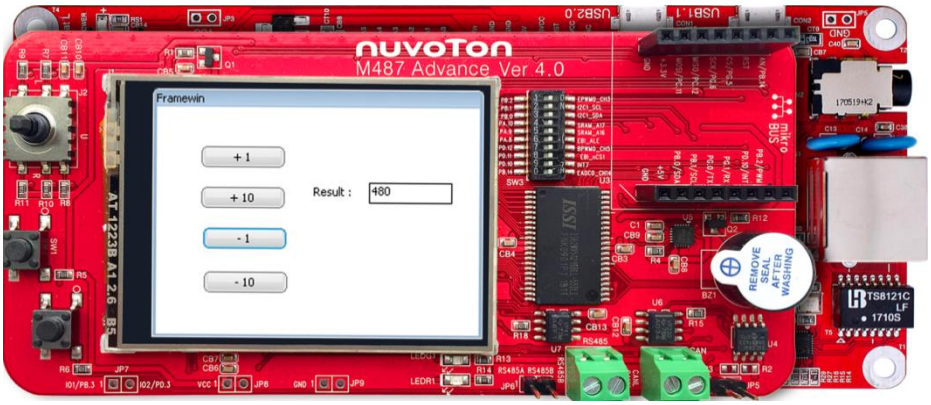
There are two emWin sample code in the M480 BSP SampleCode directory:

- **emWin_GUIDemo**: utilizes the emWin library to demonstrate widgets feature;
- **emWin_SimpleDemo**: utilizes the emWin library to demonstrate interactive feature.



3.1 Project Structure

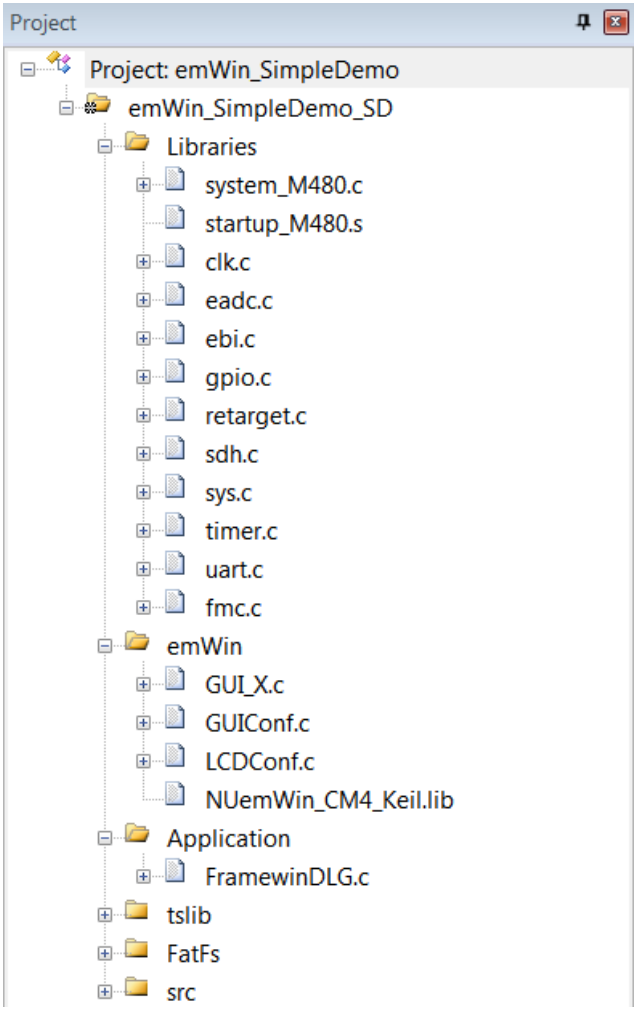
The following uses emWin_SimpleDemo as a sample to explain the emWin project structure in BSP. This sample contains a frame window, four buttons, a text and a text editor. User can update the number shown in the text field by clicking four buttons shown on the display panel.



The project structure is shown in the following figure. The project contains two targets:

- **emWin_SimpleDemo**: stores touch screen calibration parameters in APROM;
- **emWin_SimpleDemo_SD**: stores touch screen calibration parameters in a SD card.

The Libraries group contains low level driver and system startup code. The emWin group contains emWin library and panel configuration for the NuMicro® family. The Application group contains the C code generated by emWin GUIBuilder. The tslib group is the touch screen library. The FatFs group contains the file system library to access the SD card. The src group contains the main file.



3.2 System Initialization

The system initialization code is located in main function, including peripheral clock preparation, multi- function pin configuration, and UART debug port setting. Also, a 1000Hz timer is configured to keep track of time elapsed.

```
int main(void)
{
```

```
    // Init System, IP clock and multi-function I/O
    _SYS_Init();
    //
    // Init UART to 115200-8n1 for print message
    //
    UART_Open(UART0, 115200);

    // Enable Timer0 clock and select Timer0 clock source
    //
    CLK_EnableModuleClock(TMR0_MODULE);
    CLK_SetModuleClock(TMR0_MODULE, CLK_CLKSEL1_TMR0SEL_HXT, 0);
    //
    // Initial Timer0 to periodic mode with 1000Hz
    //
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 1000);
    //
    // Enable Timer0 interrupt
    //
    TIMER_EnableInt(TIMER0);
    NVIC_EnableIRQ(TMR0_IRQn);

    //
    // Start Timer0
    //
    TIMER_Start(TIMER0);

    printf("\n\nCPU @ %d Hz\n", SystemCoreClock);

   MainTask();
   while(1);
}
```

3.3 emWin Initialization

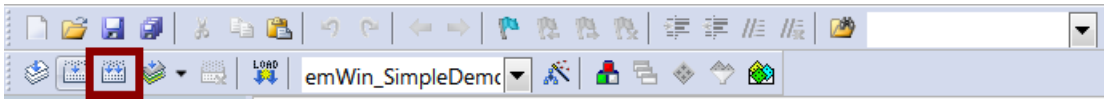
To initialize emWin GUI, the application needs to call GUI_Init() and CreatFramewin() function. The code is in MainTask() in main.c.

```
void MainTask(void)
{
    WM_HWIN hWin;
    Char acVersion[40] = "Framewin: Version of emWin: ";
```

```
    printf("Main Task -> \n");
    GUI_Init();
    strcat(acVersion, GUI_GetVersionString());
    hWin = CreateFramewin();
    FRAMEWIN_SetText(hWin, acVersion);
    while (1)
    {
        GUI_Delay(500);
    }
}
```

3.4 Build emWin Project

To build the emWin project in Keil MDK, click the rebuild icon as shown below or press F7 function key.

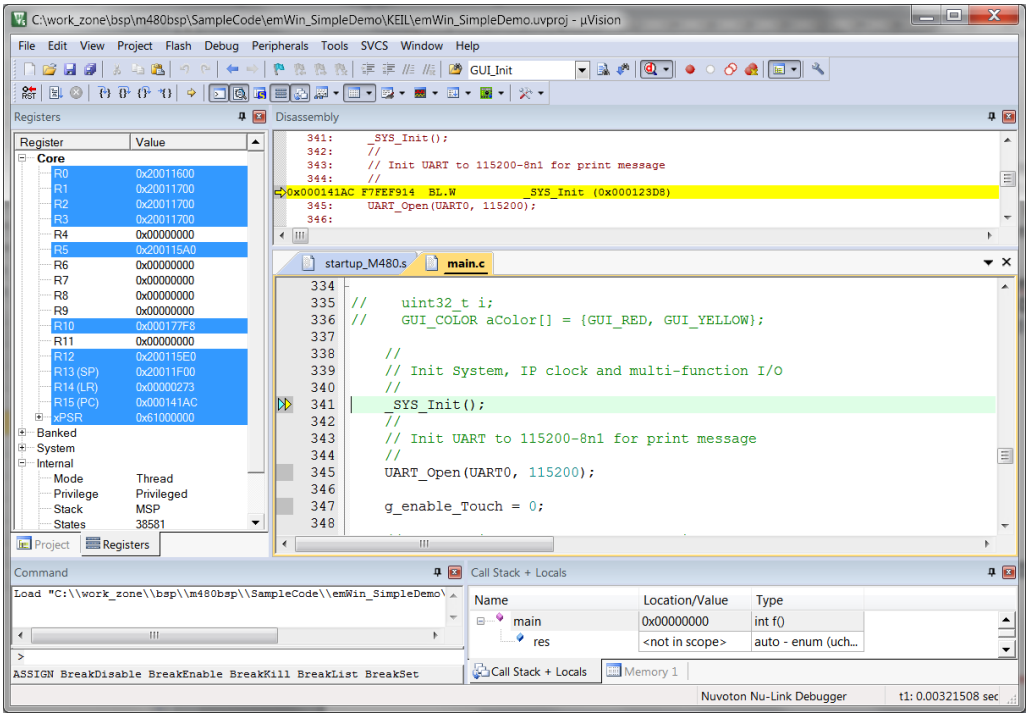


3.5 Download and Run

Press Ctrl + F5 to download the application and start a debug session or click start/stop debug session icon as shown below.

After entering debug session, press F5 to start code execution.

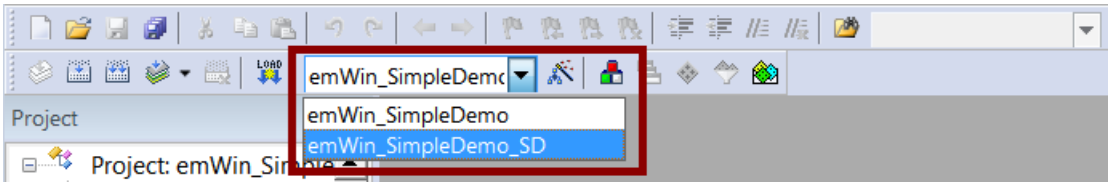
The following figure shows the application halts in main() function after starting a debug session.



3.6 Touch Screen

To support resistive touch screen, use ADC to convert the voltage of X axis and Y axis, and then use the open source tslib to map the ADC conversion result into the coordination. The conversion result can be affected by power noise, mechanical misalignment, etc. To overcome this issue, the tslib supports calibration function, and the calibration parameter is stored ether in APROM or a SD card.

As mentioned in section 3.1, there are two targets in this project: emWin_SimpleDemo uses the calibration parameter in the APROM offset 0x00040000, and emWin_SimpleDemo_SD uses the calibration parameter results in a SD card called ts_calib. User can switch between different targets using the pull down menu marked in the red rectangle shown below.



In the emWin_SimpleDemo_SD, a preprocessor symbol __USE_SD__ is defined to build the sample and use a SD card to store the calibration parameter as shown below.

The touch resolution and the APROM offset store calibration parameters in the M48XTouchPanel.h.

```
#ifndef __M48XTOUCHPANEL_H__
#define __M48XTOUCHPANEL_H__

#define __DEMO_TSFILE_ADDR__    0x00040000 /* APROM offset */

#define __DEMO_TS_WIDTH__       320
#define __DEMO_TS_HEIGHT__      240

#endif
```

If APROM is used to store the calibration parameter, main function will load the parameter from APROM. If the parameter doesn't exist, main function will call ts_calibrate() to generate a copy.

```
    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Enable FMC ISP function */
    FMC_Open();

    /* If calibration parameter exists, call ts_calibrate to generate a copy */
    if (FMC_Read(__DEMO_TSFILE_ADDR__ + 0x1C) != 0x55AAA55A)
    {
        FMC_ENABLE_AP_UPDATE();
        ts_calibrate(__DEMO_TS_WIDTH__, __DEMO_TS_HEIGHT__);
        // Erase page
        FMC_Erase(__DEMO_TSFILE_ADDR__);
        ts_writefile();
        FMC_DISABLE_AP_UPDATE();
    }
    else
    {
```

```
        ts_readfile();
    }

    /* Disable FMC ISP function */
    FMC_Close();

    /* Lock protected registers */
    SYS_LockReg();
```

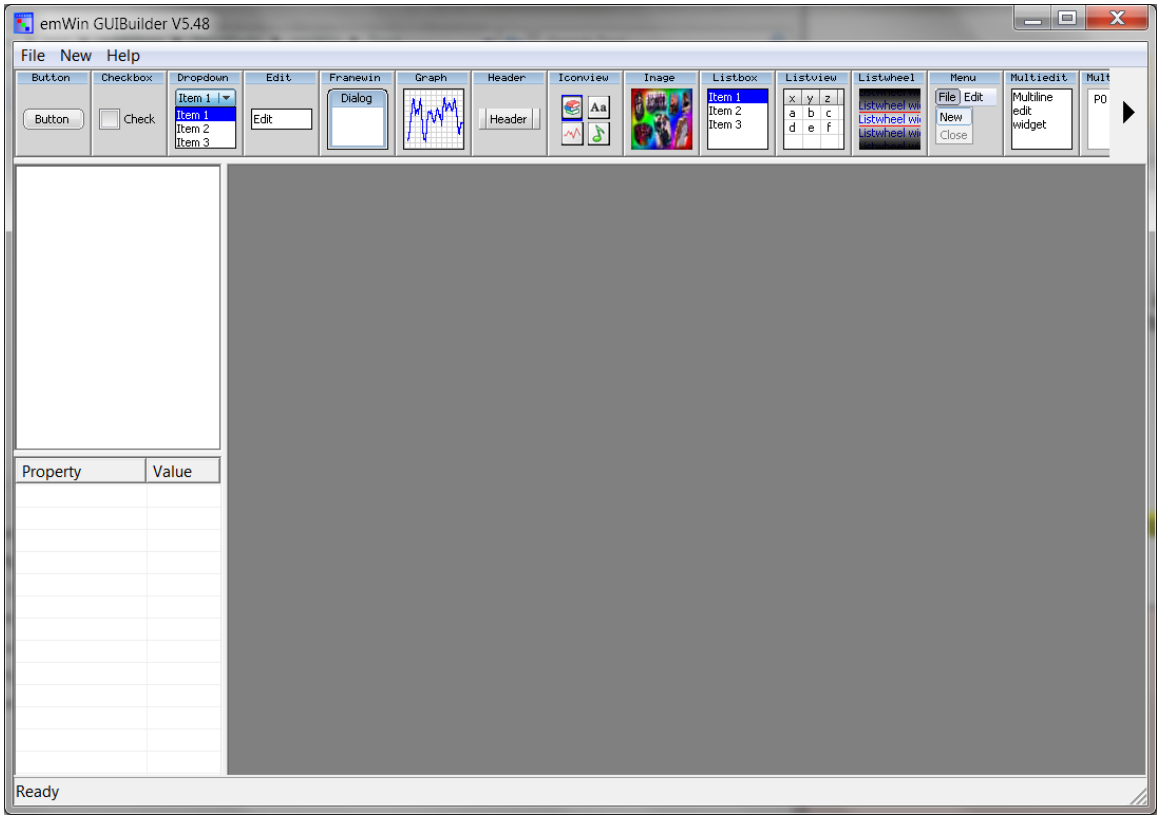
If a SD card is used to store calibration parameters, main function will load the parameter file ts_calib from the SD card root directory. If the parameter doesn't exist, main function will call ts_calibrate() to generate a copy. This sample uses FatFS to access FAT file system.

```
    SDH_Open_Disk(TEST_SDH, CardDetect_From_GPIO);
    printf("rc=%d\n", (WORD)disk_initialize(0));
    disk_read(0, Buff, 2, 1);

    GUI_Init();
    res = f_open(&hFile, "0:\\ts_calib", FA_OPEN_EXISTING | FA_READ);
    if (res)
    {
        // file does not exists, so do calibration
        res = f_open(&hFile, "0:\\ts_calib", FA_CREATE_ALWAYS | FA_WRITE);
        if ( res )
        {
            f_close(&hFile);
            GUI_DispStringAt("CANNOT create the calibration file.\nPlease insert a SD card
then reboot.", 0, 0);
            while(1);
        }

        ts_calibrate(__DEMO_TS_WIDTH__, __DEMO_TS_HEIGHT__);
        ts_writefile();
    }
    else
    {
        ts_readfile();
    }
    f_close(&hFile);
```

4 emWin GUIBuilder

4.1 Create Widget

Segger provides a Windows tool GUIBuilder to create application with drag and drop interface. The tool is located under the ThirdParty\emWin\Tool\ directory. This tool can generate a file named FramewinDLG.c for the widget of target application. Please refer to chapter 20 of UM03001_emWin5.pdf for the usage of GUIBuilder.



4.2 Handle Widget Event

FramewinDLG.c is only the framework of widget and programmers still need to add their desired widget event handler in this file after copying the FramewinDLG.c file into the project directory. Below is the event handling code of emWin_SimpleDemo.

```
......
switch (pMsg->MsgId)
{
case WM_INIT_DIALOG:
    //
    // Initialization of 'Edit'
    //
    value = 123;
```

```
    sprintf(sBuf,"%d    ", value);
    hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
    EDIT_SetText(hItem, sBuf);

    // USER START (Optionally insert additional code for further widget initialization)
    // USER END
    break;
case WM_NOTIFY_PARENT:
    Id    = WM_GetId(pMsg->hWinSrc);
    NCode = pMsg->Data.v;
    switch(Id)
    {
    case ID_BUTTON_0: // Notifications sent by '+ 1'
        switch(NCode)
        {
        case WM_NOTIFICATION_CLICKED:
            // USER START (Optionally insert code for reacting on notification message)
            // USER END
             value += 1;
             sprintf(sBuf,"%d    ", value);
             hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
             EDIT_SetText(hItem, sBuf);
             break;
        case WM_NOTIFICATION_RELEASED:
             // USER START (Optionally insert code for reacting on notification message)
             // USER END
             break;
             // USER START (Optionally insert additional code for further notification
handling)
             // USER END
            break;
......
```

5 Change Display Panel

5.1 emWin Display Configuration

emWin declares its display panel resolution in LCDConf.c under the ThirdParty\emWin\Config\ directory. The resolution is different from the touch panel resolution defined in the M48XTouchPanel.h. This is because the panel is a portrait display and data is swapped before output for a landscape view by LCD driver IC.

```
......
// Physical display size
#define XSIZE_PHYS 240
#define YSIZE_PHYS 320
```

In the LCDConf.c file, the panel orientation and control functions are also defined. These settings need to be modified according to the display panel attached to the system.

```
......
void LCD_WR_REG(uint8_t cmd)
{
    EBI0_WRITE_DATA16(0x00030000, cmd);
}
......
uint8_t LCD_RD_DATA(void)
{
    return EBI0_READ_DATA16(0x00030000);
}

void LCD_X_Config(void)
{
    ......
    // Orientation
    Config.Orientation = GUI_MIRROR_X | GUI_MIRROR_Y | GUI_SWAP_XY;
    GUIDRV_FlexColor_Config(pDevice, &Config);

    // Set controller and operation mode
    PortAPI.pfWrite16_A0  = LCD_WR_REG;
    PortAPI.pfWrite16_A1  = LCD_WR_DATA;
    PortAPI.pfWriteM16_A1 = LcdWriteDataMultiple;
    PortAPI.pfReadM16_A1  = LcdReadDataMultiple;
    GUIDRV_FlexColor_SetFunc(pDevice,
                             &PortAPI,
                             GUIDRV_FLEXCOLOR_F66709,
```

```
                                GUIDRV_FLEXCOLOR_M16C0B16);
......
```

5.2 Display Driver

The project file includes the ebi.c driver since demo system is connected to a MPU display using the EBI interface. For systems connecting display with the SPI or I^2C interface, spi.c or i2c.c needs to be added to the project.

Revision History

Date	Revision	Description
2018.10.04	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.