以 emWin 设计示波器 GUI 专案

NuMicro® 32 位系列微控制器范例代码介绍

文件资讯

应用简述	本范例代码使用 SEGGER emWin GUIBuilder 建立图形用户界面(GUI)，结合 M467 系列开发平台，展示示波器图形化接口。
BSP 版本	M460_Series_BSP_CMSIS_V3.00.001
开发平台	NuMaker-HMI-M467

1. 概述

本范例程序将展示使用 SEGGER emWin 于 NuMaker-HMI-M467 开发平台，逐步完成示波器专案的 GUI(Graphical User Interface)界面设计，互动上设计 Run/Stop 按钮及 Trigger、Reset、Zoom in、Zoom out 等相应按钮点击功能。使用中断 EADC 撷取模拟信号并于画面上显示，EADC 的采样频率为 1kHz，最高支持 3.3v 信号源，同步启动 DAC 输出 30Hz 的弦波验证。

1.1 UI 设计

本范例使用 emWin GUIBuilder V6.30 做画面设计，GUIBuilder IDE 开发上可区分为 Function window、Editor window、Hierarchic Tree、Properties，如图 1-1 所示。用户只需透过 Function window 上的组件拖曳至 Editor window 就能设计 UI 画面。

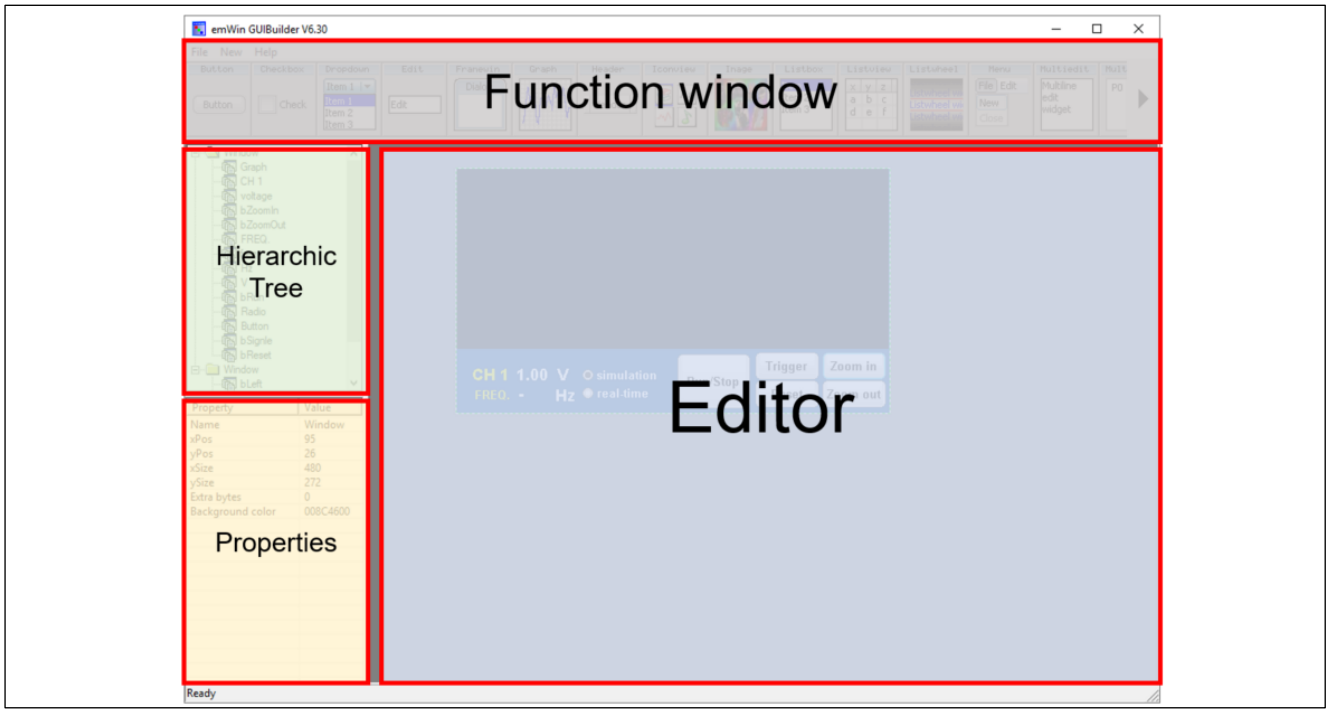

图 1-1 emWin GUIBuilder IDE

NuMaker-HMI-M467 开发板采用 NuMaker-THT-LCD43，window 大小为 480x272 pixels。设置示波器项目会需要 Graph 组件绘制波形，新增 Text 字段显示波形频率信息，Radio 单选当前画面显示的模式，分为 simulation 及 real-time 模式，其中 simulation 模式会仿真一个弦波输出，real-time 模式会真实采样 EADC 信息，按钮则绘制 Run/Stop 按钮及 Trigger、Reset、Zoom

in、Zoom out 等相应按钮点击功能。Run/Stop 按钮可以设置示波器画面开启卷动或停止卷动功能；Trigger 按钮可以撷取第一段信号源，与画面的大小配置相同；Reset 按钮会清除当前页面的数据及重置功能；Zoom in/out 会放大或缩小波形的振幅，设计至多 3 个 level。

组件的颜色大小配置都可以透过 **Properties** 中配置，GUI 画面设计及属性可参考图 1-3 至图 1-5。

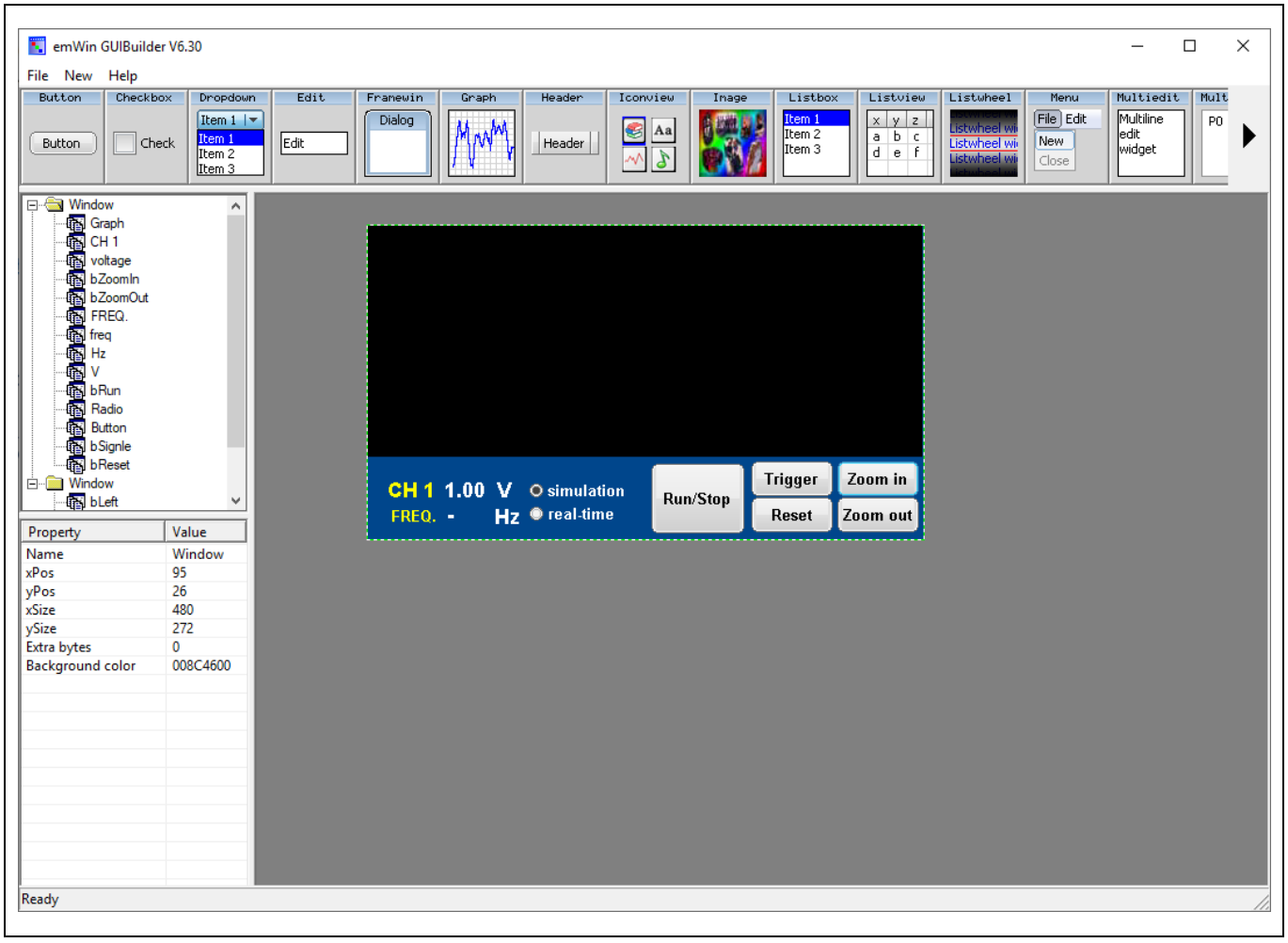

图 1-2 示波器项目格局配置图

存档后会更新 *FramewinDLG.c* 档案，需要将变更后的档案覆盖到示波器项目，此小节即完成初步的 UI 设计。于此范例程序中，*FramewinDLG.c* 存放于 Application 文件夹中，已设计好 UI 的框架及按钮的配置等设定，用户可以自行设计自己的 GUI。

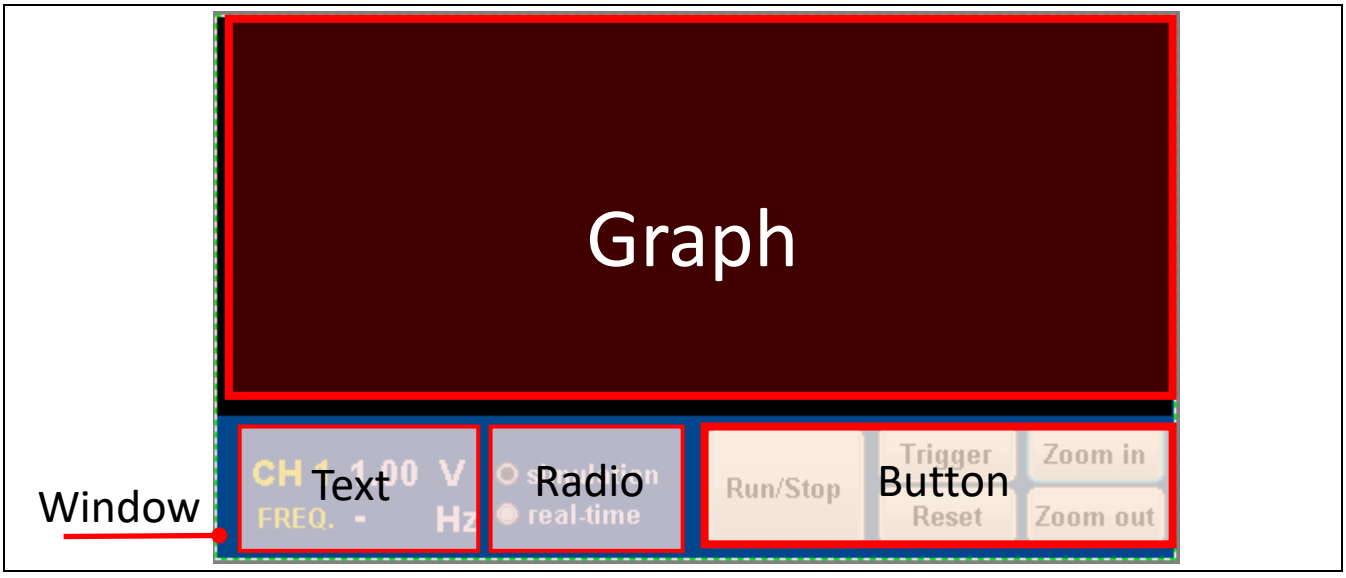

图 1-3 示波器组件规划配置图

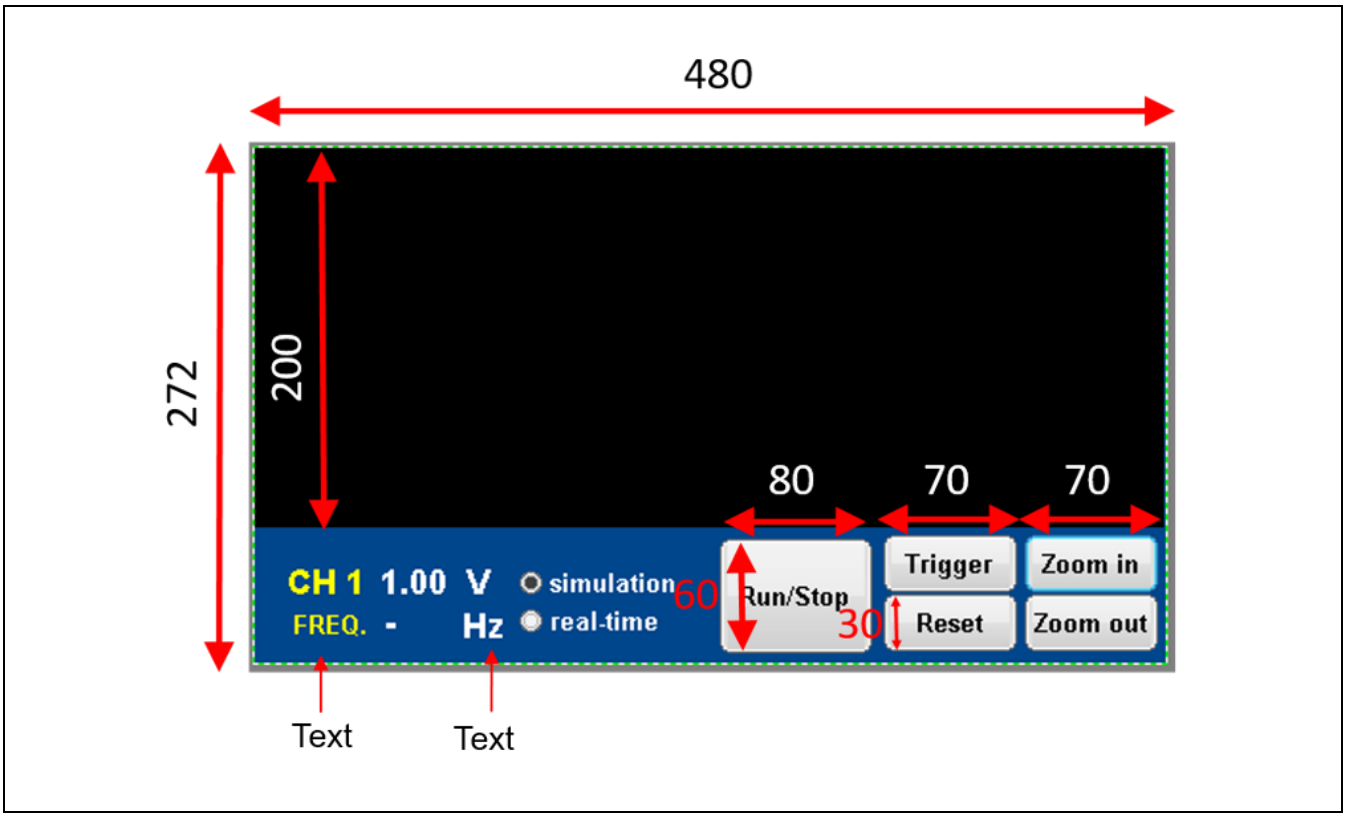

图 1-4 示波器组件大小图

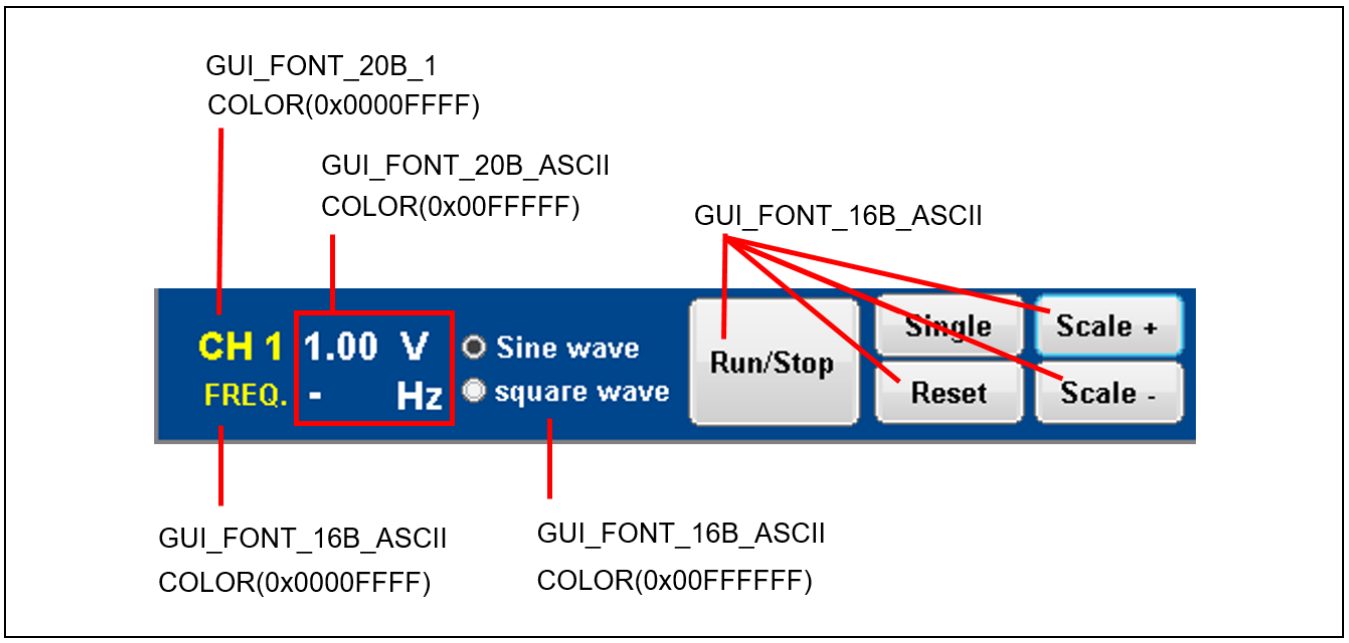

图 1-5 示波器组件字型色彩图

1.2 UI 代码原理

系统的框架定义，如图 1-6 所示。

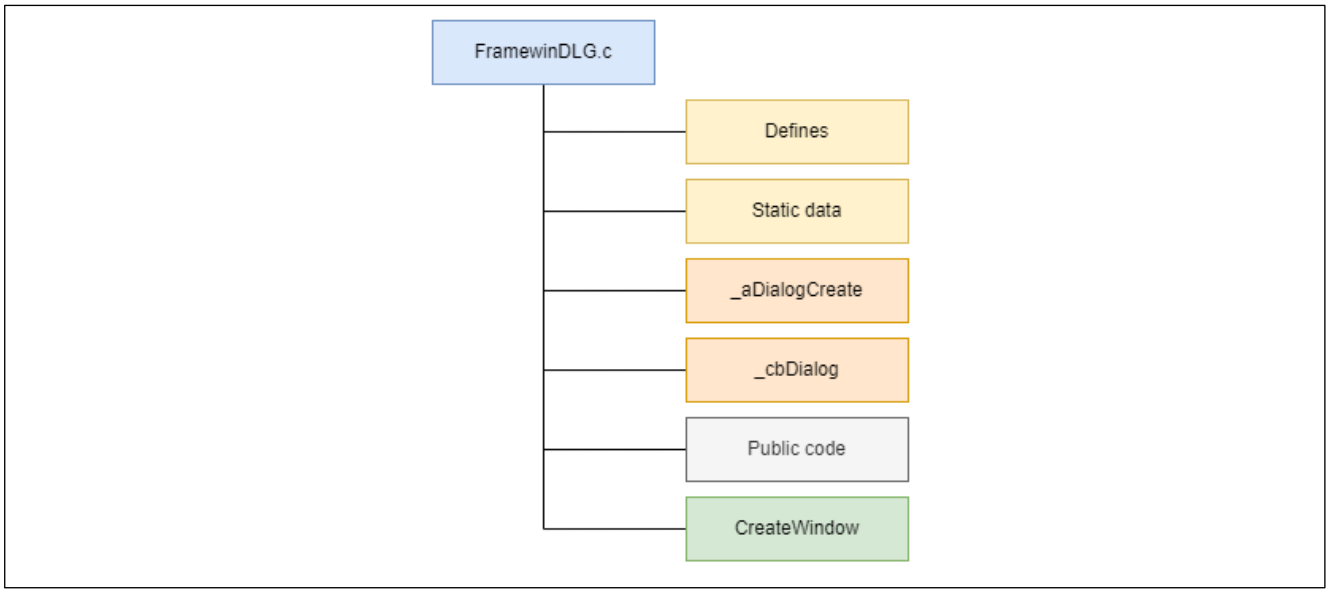

图 1-6 UX 框架

FramewinDLG.c 中以 API 方式来设定组件的外观，可以在**_aDialogCreate** 中发现到 GUIBuilder 于 Editor 中的位置等配置:

```
/*********************************************************************
*
*    _aDialogCreate
*/

static const GUI_WIDGET_CREATE_INFO _aDialogCreate[] =
{
    { WINDOW_CreateIndirect, "Window",  ID_WINDOW_0,  0,   0, 480, 272, 0, 0x0, 0 },
    { GRAPH_CreateIndirect,  "Graph",   ID_GRAPH_0,   0,   0, 480, 200, 0, 0x0, 0 },
    … …
    { BUTTON_CreateIndirect, "Button",  ID_BUTTON_3, 318, 314, 80, 30, 0, 0x0, 0 },
    { BUTTON_CreateIndirect, "bSingle", ID_BUTTON_4, 331, 204, 70, 30, 0, 0x0, 0 },
    { BUTTON_CreateIndirect, "bReset",  ID_BUTTON_5, 331, 235, 70, 30, 0, 0x0, 0 },
};
```

FramewinDLG.c 中**_cbDialog** 处理 GUI 生成及对应事件，而其中 GRAPH widget 设定网格网格线的配置:

```
/*********************************************************************
*
*    _cbDialog
*/

static void _cbDialog(WM_MESSAGE * pMsg)
{
… …
    switch (pMsg->MsgId)
    {

        case WM_INIT_DIALOG:
            // Initialization of 'Window'
            hItem = pMsg->hWin;
            WINDOW_SetBkColor(hItem, GUI_MAKE_COLOR(0x008C4600));

            // Enable multi-buffering
            WM_MULTIBUF_Enable(1);

            … …

            // Initialize the GRAPH widget
            hItem = WM_GetDialogItem(pMsg->hWin, ID_GRAPH_0);

            // Set the appearance and grid properties of the graph
            GRAPH_SetBorder(hItem, 1,1,1,1);
            GRAPH_SetColor(hItem, GUI_BLACK, GRAPH_CI_BK);
            GRAPH_SetColor(hItem, GUI_GRAY_E7, GRAPH_CI_BORDER);
            GRAPH_SetColor(hItem, GUI_GRAY_E7, GRAPH_CI_FRAME);
            GRAPH_SetColor(hItem, GUI_WHITE, GRAPH_CI_GRID);
            GRAPH_SetGridVis(hItem, 1);
            GRAPH_SetGridDistX(hItem, 50);
            GRAPH_SetGridDistY(hItem, 25);
            GRAPH_SetLineStyleH(hItem, GUI_LS_DOT);
            GRAPH_SetLineStyleV(hItem, GUI_LS_DOT);

            // Create and configure a vertical scale, then attach it to the graph
            hScaleHandle = GRAPH_SCALE_Create(15, GUI_TA_HCENTER | GUI_TA_VCENTER,
GRAPH_SCALE_CF_VERTICAL, 1);
            GRAPH_SCALE_SetFont(hScaleHandle, &GUI_Font8x8);
            GRAPH_SCALE_SetTextColor(hScaleHandle, GUI_BLACK);
```

```
            GRAPH_SCALE_SetNumDecs(hScaleHandle, 0);
            GRAPH_SCALE_SetTickDist(hScaleHandle, 20);
            GRAPH_SCALE_SetOff(hScaleHandle, 100);
            GRAPH_AttachScale(hItem,hScaleHandle);

            // Introduce a delay of 500 milliseconds
            GUI_Delay(500);

            // Create and configure graph data, then attach it to the graph
            hGraphData = GRAPH_DATA_YT_Create(GUI_YELLOW,LCD_GetXSize(),DrawData,0);
            GRAPH_DATA_YT_SetAlign(hGraphData, GRAPH_ALIGN_RIGHT);
            GRAPH_DATA_YT_SetOffY(hGraphData, 100);
            GRAPH_AttachData(hItem, hGraphData);


            // Set a custom callback, here we want to draw a cursor
            // WM_SetCallback(hItem, _cbGraph);
            break;


        //
        // event
        //
        case WM_NOTIFY_PARENT:
            // Retrieve the ID and notification code of the event sender
            Id    = WM_GetId(pMsg->hWinSrc);
            NCode = pMsg->Data.v;

            switch(Id)
            {
                case ID_BUTTON_0: // Notifications sent by 'bZoomIn'
                    switch(NCode)
                    {
                        case WM_NOTIFICATION_CLICKED:
                            // Clear existing graph data
                            GRAPH_DATA_YT_Clear(hGraphData);

                            // Increment the 'multiple' factor for zooming in
                            multiple++;
                            if(multiple > 2) multiple=2;

                            // Update graph data with adjusted values
                            for(i=0; i<480; i++)
                            {
                                GRAPH_DATA_YT_AddValue(hGraphData,  DrawData[i] *
multipleArray[multiple]);
                            }
                            break;
                        case WM_NOTIFICATION_RELEASED:
                            break;
                    }
                    break;

                case ID_BUTTON_1: // Notifications sent by 'bZoomOut'
                    switch(NCode)
                    {
                        case WM_NOTIFICATION_CLICKED:
                            // Clear existing graph data
                            GRAPH_DATA_YT_Clear(hGraphData);

                            if(multiple == 0)
                                multiple=0;
                            else
                                multiple--;

                            for(i=0; i<480; i++)
```

```
                                {
                                    GRAPH_DATA_YT_AddValue(hGraphData,  DrawData[i] *
multipleArray[multiple]);
                                }
                                break;

                            case WM_NOTIFICATION_RELEASED:
                                // USER START (Optionally insert code for reacting on
notification message)
                                break;
                        }
                        break;
             … …

             }
             break;

… …
}
```

CreateFramewin()来执行 GUI 的创建，并设置 multi-buffering。

```
/*********************************************************************
*
*    CreateWindow
*/
WM_HWIN CreateFramewin(void)
{

    WM_HWIN hWin;

    // Enable multi-buffering for the window
    WM_MULTIBUF_Enable(1);
    // Create a dialog box (FrameWin) using the specified dialog creation parameters
    hWin = GUI_CreateDialogBox(_aDialogCreate, GUI_COUNTOF(_aDialogCreate), _cbDialog,
WM_HBKWIN, 0, 0);

    return hWin;
}
```

2. 代码介绍

main.c 中先对系统初始化，设定触控屏幕后执行 MainTask()以启动 emWin 显示 GUI。

于 **MainTask()** 中，**Oscilloscope_init()**会启动示波器取得 EADC 的数据，采样率约 1kHz，同步输出一个弦波，并更新 GUI 上的数据资料。

```
/*---------------------------------------------------------------------------------------------------------*/
/*  Main Function                                                                                          */
/*---------------------------------------------------------------------------------------------------------*/

int main(void)
{
    /* init */
    SYS_Init();

    // Initialize the touch functionality.
    g_enable_Touch = 0;
    tmr_init();
    st1633i_low_level_init();
    st1633i_startup_sequence();
    g_enable_Touch = 1;

    // Print a message indicating the CPU frequency.
    printf("[M]Oscilloscope Demo - CPU @ %d Hz \r\n", SystemCoreClock);

    /* Main Loop  */
    MainTask();


    // This loop will keep the program running indefinitely.
    while(1) {}

    return 0;
}
```

```
void MainTask(void)
{
    // Initialize variables
    int i=0;
    int32_t bSingleCount = 0;
    int32_t bSingleFlag = 0;

    // Create a window handle
    WM_HWIN hWin;
    printf("Main Task -> \n");

    // Enable memory device for GUI rendering
    WM_SetCreateFlags(WM_CF_MEMDEV);

    // Initialize the GUI
    GUI_Init();
    WM_MULTIBUF_Enable(1);   // Enable multi-buffering for better performance
    hWin = CreateFramewin(); // Create a window using CreateFramewin function

    // Initialize the Oscilloscope
    Oscilloscope_init();
```

```
    // Enter the main loop of the task
    while (1)
    {
        GUI_Delay(1);
         …
    }
}
```

Oscilloscope.c 中初始化就设置 EADC(量测引脚为 PB0)、Timer1、Timer2、EPWM(输出脚位为 PC12)和 DAC(输出脚位为 PB12)。屏幕更新率同为 100Hz 刷新一次。

```
/*****************************************************************************************/
/* Oscilloscope init                                                                     */
/*   Use EADC  -> TMR1 1kHz                                                              */
/*       DAC   -> TMR2 2kHz                                                              */
/*       EPWM1 -> 30 Hz                                                                  */
/*****************************************************************************************/
void Oscilloscope_init(void)
{
     SYS_UnlockReg();

    /* Enable module clock */
    CLK_EnableModuleClock(EADC0_MODULE);
    CLK_EnableModuleClock(TMR1_MODULE);
    CLK_EnableModuleClock(TMR2_MODULE);
    CLK_EnableModuleClock(EPWM1_MODULE);
    CLK_EnableModuleClock(DAC_MODULE);

    /* Set module clock */
    CLK_SetModuleClock(EADC0_MODULE, CLK_CLKSEL0_EADC0SEL_PLL_DIV2,
CLK_CLKDIV0_EADC0(12));
    CLK_SetModuleClock(EPWM1_MODULE, CLK_CLKSEL2_EPWM1SEL_PCLK1, 0);
    CLK_SetModuleClock(TMR1_MODULE, CLK_CLKSEL1_TMR1SEL_HIRC, NULL);
    CLK_SetModuleClock(TMR2_MODULE, CLK_CLKSEL1_TMR2SEL_HIRC, 0);

    /* EADC0 channels. */
    SET_EADC0_CH0_PB0();
    SET_DAC0_OUT_PB12();

    /* Disable digital input path of EADC analog pin to prevent leakage */
    GPIO_DISABLE_DIGITAL_PATH(PB, BIT0);

    /* EPWM channels. */
    SET_EPWM1_CH0_PC12();
    EPWM_ConfigOutputChannel(EPWM1, 0, 30, 50);
    EPWM_EnableOutput(EPWM1, 0x3F);
    EPWM_Start(EPWM1, 0x3F);

    /* module init */
    TMR1_Init();
    EADC_Init();
    DAC_Init();
    SYS_LockReg();
}
```

若要变更 EADC 采样时间可以变更 Timer1 中设置，目前设置的采样频率是 1kHz 并触发 EADC 中断测量。

```
/*---------------------------------------------------------------------------------------------------------*/
/* TIMER1 init */
/*---------------------------------------------------------------------------------------------------------*/

void TMR1_Init(void)
{
    TIMER_Open(TIMER1, TIMER_PERIODIC_MODE, 1000);
    TIMER_SetTriggerTarget(TIMER1, TIMER_TRG_TO_EADC);

    // Start Timer 1
    TIMER_Start(TIMER1);
}
```

```
/*---------------------------------------------------------------------------------------------------------*/
/* EADC  init */
/*---------------------------------------------------------------------------------------------------------*/
void EADC_Init(void)
{
    int32_t i32Err;

    i32Err = EADC_Open(EADC0, EADC_CTL_DIFFEN_SINGLE_END);

    … …

    /* Configure the sample module 0 for analog input channel 0 and enable Timer1 trigger
source */
    EADC_ConfigSampleModule(EADC0, 0, EADC_TIMER1_TRIGGER, 0);

    /* Set sample module external sampling time to 0 */
    EADC_SetExtendSampleTime(EADC0, 0, 0);

    /* Enable Accumulate feature */
    EADC_ENABLE_ACU(EADC0, 0, EADC_MCTL1_ACU_32);

    /* Enable Average feature */
    EADC_ENABLE_AVG(EADC0, 0);

    /* Clear the A/D ADINT0 interrupt flag for safe */
    EADC_CLR_INT_FLAG(EADC0, EADC_STATUS2_ADIF0_Msk);

    /* Enable the sample module interrupt.  */
    EADC_ENABLE_INT(EADC0, BIT0);
    EADC_ENABLE_SAMPLE_MODULE_INT(EADC0, 0, BIT0);
    NVIC_EnableIRQ(EADC00_IRQn);

    /* Reset the ADC interrupt indicator and trigger sample module to start A/D
conversion */
    EADC_START_CONV(EADC0, BIT0);

}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M460_Series_BSP_CMSIS_V3.00.001
- IDE 版本
  - Keil uVersion 5.36

3.2 硬件需求

- 电路组件
  - NuMaker-HMI-M467HJ V1.1
- 线路示意图
  - 将 NuMaker-HMI-M467HJ 透过 USB 线材与 PC 相接。
  - 输入脚位: EADC (PB0)，请使用杜邦线接线连接至 PB12 或 PC12。
  - 输出脚位: DAC (PB12)、EPWM (PC12)

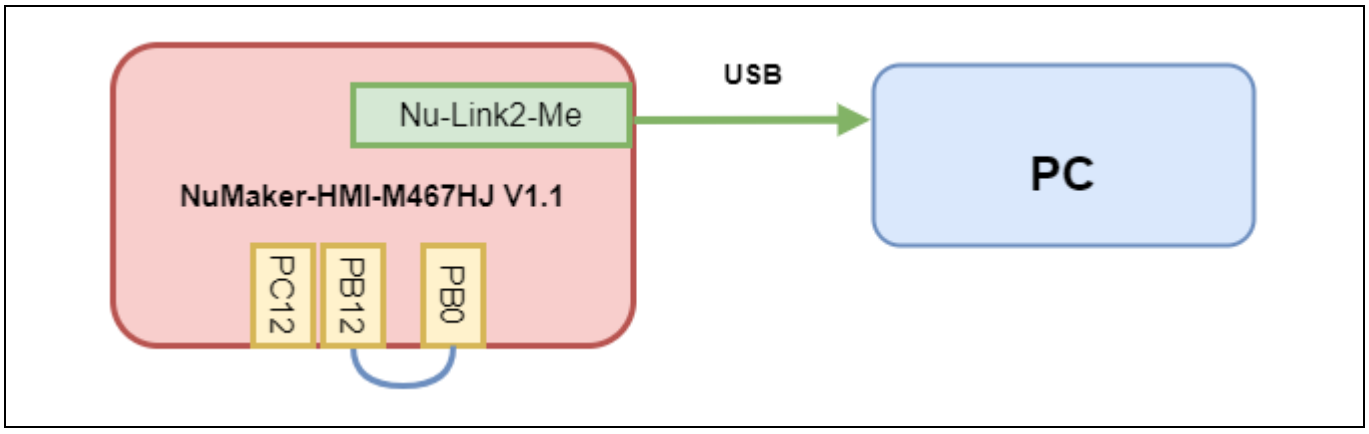

图 3-1 NuMaker-HMI-M467HJ 和 PC 连接示意图

4. 目录信息

EC_M467_emWin_Oscilloscope_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	
emWin_Oscilloscope	Source file of example code
ThirdParty	Header files for emWin project
Tool	Tool : GUIBuilder.exe

图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *emWin_Oscilloscope.uvprojx*。
2. 进入编译模式接口
   - 编译
   - 下载代码至内存
   - 进入 / 离开除错模式
3. 进入除错模式接口
   - 执行代码
   - 执行结果

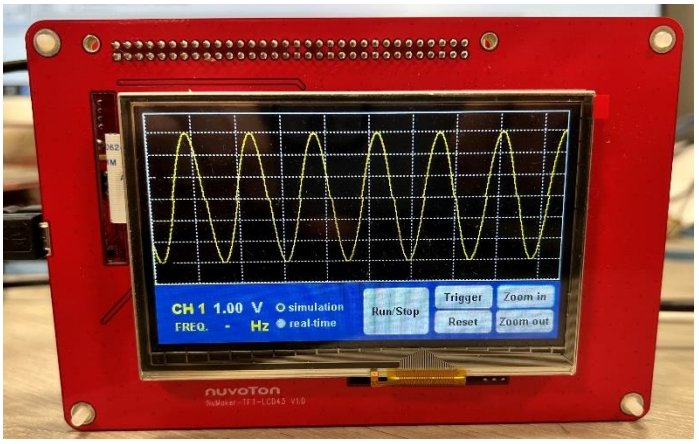

图 5-1 NuMaker-HMI-M467HJ 执行结果图

6. 修订纪录

Date	Revision	Description
2023.08.13	1.00	初始发布。

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.