M480 Intelligent Power Station

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This example code uses M480 GPIO pins to control the relay and MOSFET switches and the Enhance ADC (EADC) with peripheral hardware circuits to measure the AC power currents and USB DC currents that passing thru the enabled AC and USB sockets.
BSP Version	M480 Series BSP CMSIS V3.03.001
Hardware	The M487 Smart Power board but it needs to work with NuMaker-M487D board and DALI master and slave boards.

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

www.nuvoton.com

1 Function Description

1.1 Introduction

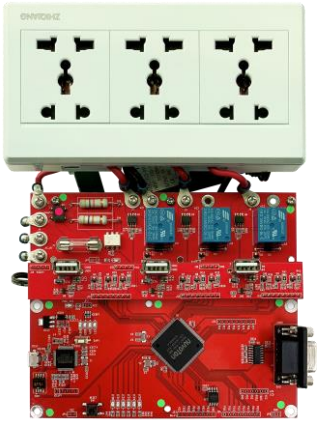
This example code utilizes the M487 Smart Power board (Ver. 1.1) as the following figure that based on the main controller M487 MCU to control the relay and MOSFET switches by some GPIO pins of MCU and measure the AC power currents and USB DC currents by the Enhance ADC (EADC) of MCU with peripheral hardware circuits.

The Intelligent Power Station hardware consists of three boards, M487 Smart Power board, NuMaker-M487D board and DALI master/slave boards. For more information on the Intelligent Power Station, please visit our Nuvoton website to research this reference design and download the User Manual UM_Smart-Power-M487_User_Manual_EN document and hardware information HW_M487_SMART_POWER_V1.1 file.

https://www.nuvoton.com/hq/applications/smart-home-appliances/intelligent-power-station/?__locale=en

http://www.nuvoton.com/hq/products/microcontrollers/arm-cortex-m4-mcus/User-Manual/?__locale=en&resourcePage=Y&category=&pageIndex=2

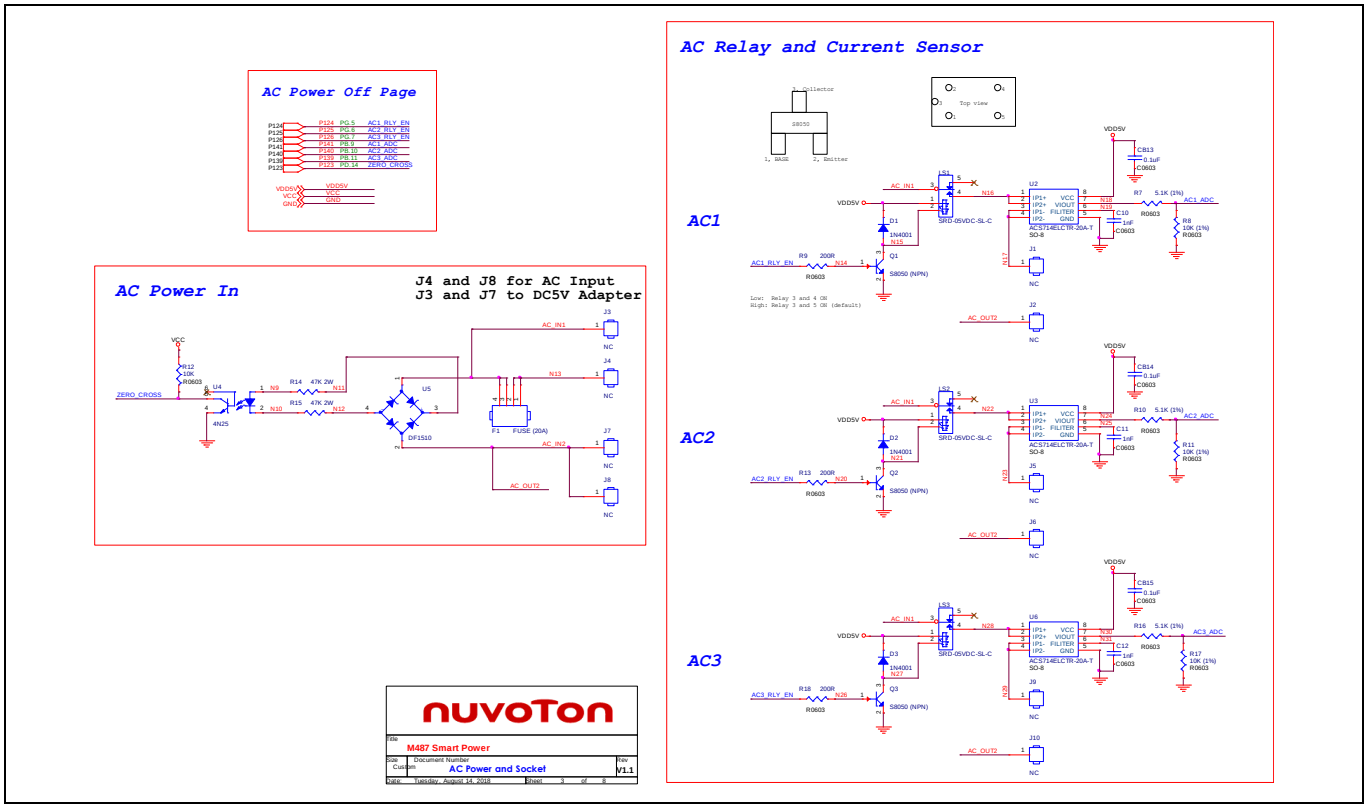
https://www.nuvoton.com/hq/products/microcontrollers/arm-cortex-m4-mcus/Development-Tool/?__locale=en&resourcePage=Y



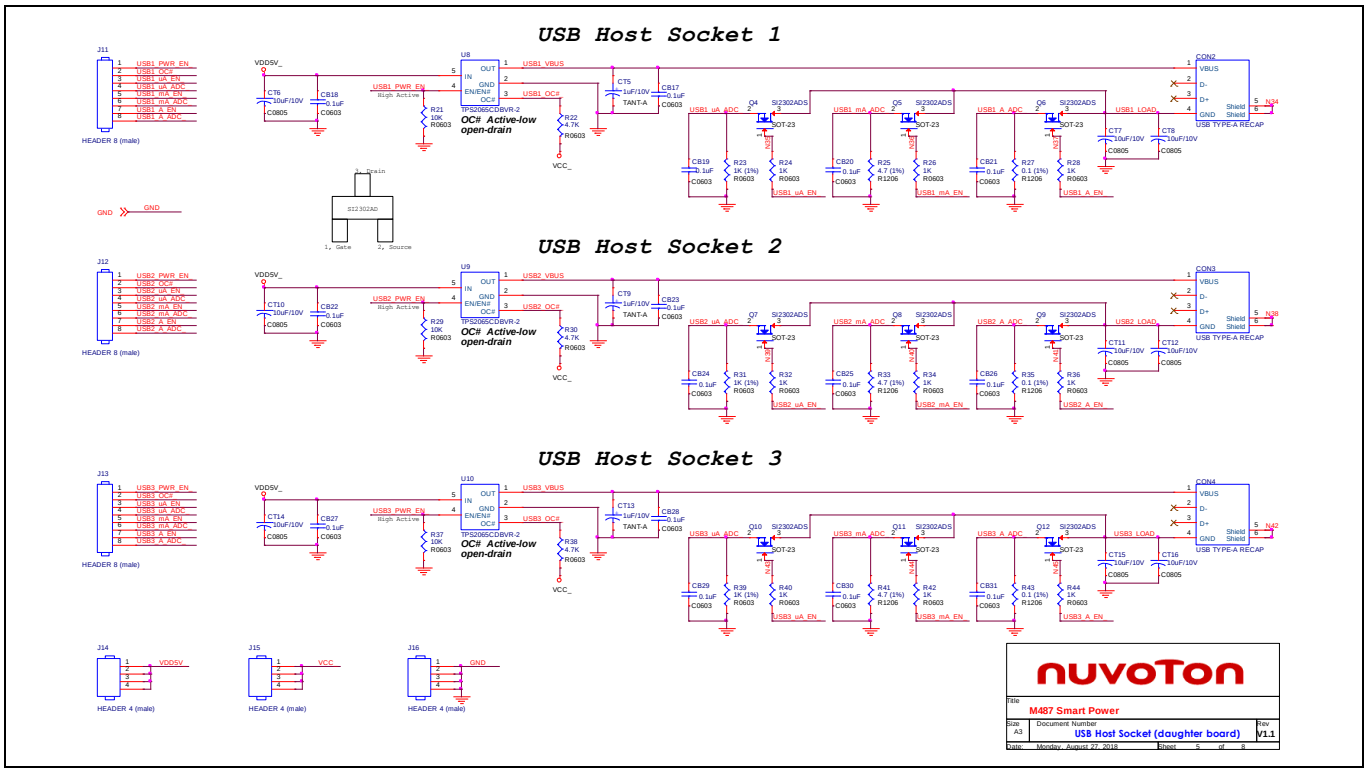
M487 Smart Power Board

1.2 Principle

This sample code utilizes some GPIO pins of M487 MCU to control the relay and MOSFET switches and EADC of MCU with peripheral hardware circuits to measure the AC power currents and USB DC currents that passing thru the enabled AC and USB sockets. The following figures are the hardware schematics for the AC and USB DC circuits.



AC Power Schematics



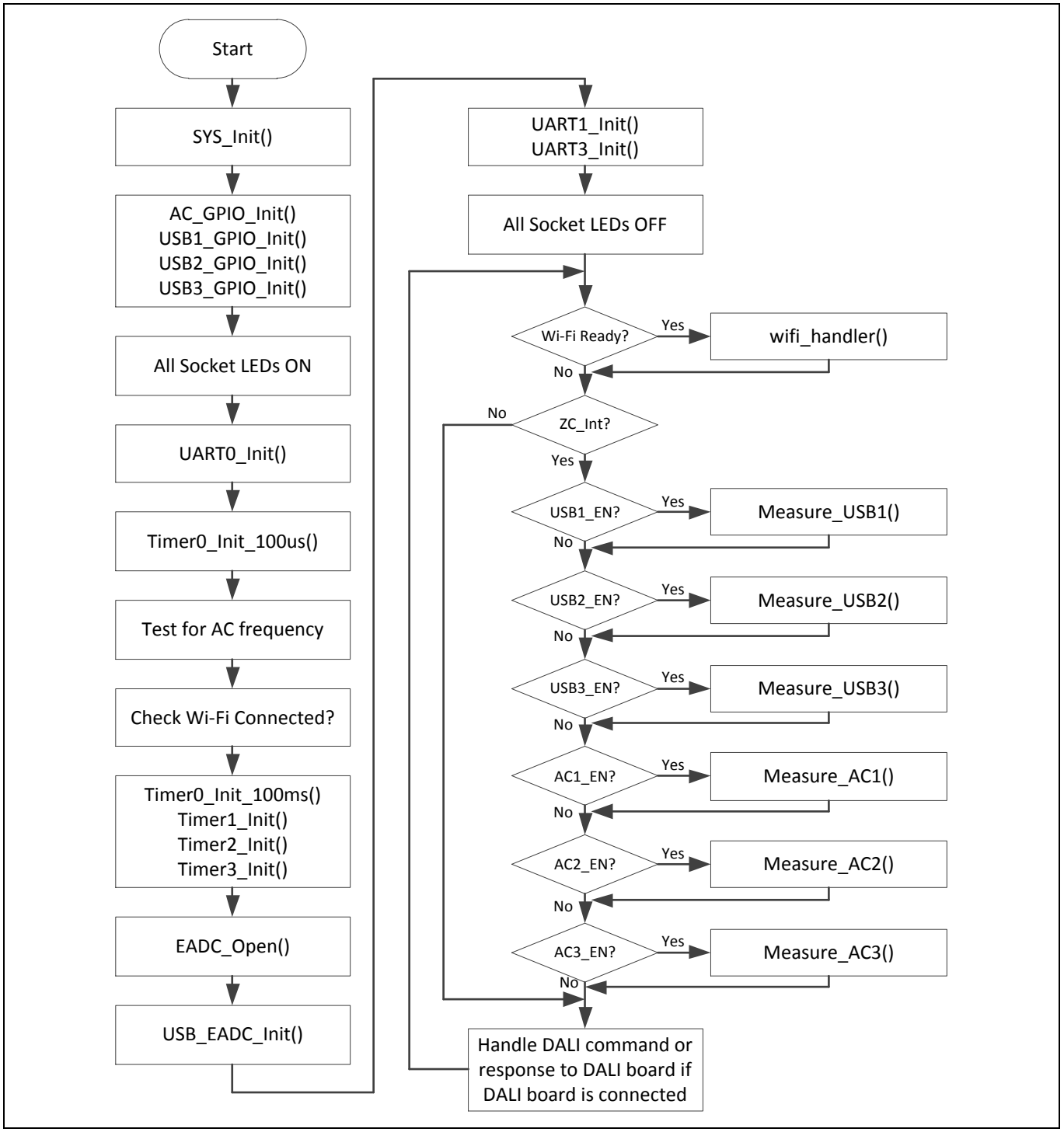
USB DC Schematics

1.3 Demo Result

After executed this example firmware code that had been programmed in the flash memory of M487 MCU on the M487 Smart Power board, user can measure the AC currents and USB DC currents that passing thru the enabled AC and USB sockets and these current values will be shown on the LCD panel of the NuMaker-M487D board.

2 Code Description

The main() function flow chart of the firmware on M487 Smart Power board is shown below.



Main() Flow Chart

```
int32_t main(void)
{
    uint8_t wifi_rdy = 0;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, IP clock and multi-function I/O */
    SYS_Init();

    /* Lock protected registers */
    SYS_LockReg();

    /* Set GPIO as output for emWin to show the Wi-Fi status */
    /* PD2: = 0, Wi-Fi connect checking, 1 = Wi-Fi check done */
    /* PD3: = 0, Wi-Fi not connected, 1 = connected */
    GPIO_SetMode(PD, BIT2|BIT3, GPIO_MODE_OUTPUT);
    PD2 = 0; // Wi-Fi connect checking
    PD3 = 0; // Wi-Fi not connected

    /* Set GPIO for AC socket */
    AC_GPIO_Init();

    /* Set GPIO for USB1 socket */
    USB1_GPIO_Init();
    /* Set GPIO for USB2 socket */
    USB2_GPIO_Init();
    /* Set GPIO for USB3 socket */
    USB3_GPIO_Init();

    /* Set GPIO as output for LEDs */
    GPIO_SetMode(PH, BIT0|BIT1|BIT2|BIT3|BIT4|BIT5, GPIO_MODE_OUTPUT);
    /* Turn on all LEDs */
    LED0_ON
    LED1_ON
    LED2_ON
    LED3_ON
    LED4_ON
    LED5_ON

    /* Init UART0 for printf */
```

```
    UART0_Init();
    printf("\n=========================================\n");
    printf("=                                       =\n");
    printf("=      M480 Intelligent Power Station      =\n");
    printf("=                                       =\n");
    printf("=========================================\n");
    printf("System clock rate: %d Hz\n", SystemCoreClock);

    /* Init Timer0 as 100us periodically for AC frequency measurement */
    Timer0_Init_100us();
    /* Test for AC frequency = 60Hz or 50 Hz */
    ZC_Int = 0;
    while(ZC_Int == 0);
    ZC_Int = 0;
    TIMER_Start(TIMER0);
    while(ZC_Int == 0);
    if((Timer0_Cnt == 82) || (Timer0_Cnt == 83))
        AC_60Hz = 1;
    else
        AC_60Hz = 0;

    if(AC_60Hz)
        printf("AC Frequency(%d): 60Hz\n", Timer0_Cnt);
    else
        printf("AC Frequency(%d): 50Hz\n", Timer0_Cnt);

    /* Wi-Fi connect checking */
    if(wifi_setup()) {
        PD3 = 1; // Wi-Fi connected
        printf("Setup Wifi success...\r\n");
        wifi_rdy = 1;
    }

    /* Init Timer0 as 100ms periodically for DALI time-out */
    Timer0_Init_100ms();
    /* Init Timer1 to trigger EADC for AC1 currnet measure */
    Timer1_Init();
    /* Init Timer2 to trigger EADC for AC2 currnet measure */
    Timer2_Init();
    /* Init Timer3 to trigger EADC for AC3 currnet measure */
    Timer3_Init();
```

```
/* Open ADC */
/* Set input mode as single-end and enable the A/D converter */
EADC_Open(EADC, EADC_CTL_DIFFEN_SINGLE_END);

/* Init EADC for USB */
USB_EADC_Init();

/* Init UART1 for DALI */
UART1_Init();

/* Init UART3 for LCD disply */
UART3_Init();

PD2 = 1; // Wi-Fi check and Initial done

/* Turn off all LEDs */
LED0_OFF
LED1_OFF
LED2_OFF
LED3_OFF
LED4_OFF
LED5_OFF

/*=== While loop ===*/
while(1) {
    /* Check Wi-Fi ready */
    if(wifi_rdy)
        wifi_handler();

    /* Parse Command from LCD */
    if(bUartDataReady_LCD) {
        bUartDataReady_LCD = 0;
        ParseCommand_LCD();
    }

    /*=== Waiting for Zero-cross interrupt ===*/
    if(ZC_Int) {
        ZC_Int = 0;

        /*=== USB1~3 current and power consumption measuring ===*/
```

```
            /* USB1 socket */
            if(USB1_En) {
                if(USB1_Init) {
                    /* To mesaure the USB1 current */
                    Measure_USB1();
                } else {
                    /* To do the initialization for USB1 socket */
                    USB1_PWR_EN = 1;        // Power-on USB1 socket
                    USB1_uA_EN = 1;         // Enable uA switch (1k ohm)
                    EADC_ConfigSampleModule(EADC, 0, EADC_SOFTWARE_TRIGGER, USB1_uA_CH);
// Configure EADC0_SM0 for CH0
                    Measure_USB1();         // Mesaure USB1 current
                    USB1_Init = 1;          // USB1 initial done
                    ZC_Cnt_USB1 = 0;        // Clear ZC_Cnt_USB1
                    USB1_Pwr_Unit = 0;      // Clear USB1_Pwr_Unit
                    USB1_Power = 0;         // Clear USB1_Power
                    USB1_Power_Acc = 0;     // Clear USB1_Power_Acc
                    LED3_ON
                }
            }
            /* USB2 socket */
            if(USB2_En) {
                if(USB2_Init) {
                    /* To mesaure the USB2 current */
                    Measure_USB2();
                } else {
                    /* To do the initialization for USB2 socket */
                    USB2_PWR_EN = 1;        // Power-on USB2 socket
                    USB2_uA_EN = 1;         // Enable uA switch (1k ohm)
                    EADC_ConfigSampleModule(EADC, 1, EADC_SOFTWARE_TRIGGER, USB2_uA_CH);
// Configure EADC0_SM1 for CH3
                    Measure_USB2();         // Mesaure USB2 current
                    USB2_Init = 1;          // USB2 initial done
                    ZC_Cnt_USB2 = 0;        // Clear ZC_Cnt_USB2
                    USB2_Pwr_Unit = 0;      // Clear USB2_Pwr_Unit
                    USB2_Power = 0;         // Clear USB2_Power
                    USB2_Power_Acc = 0;     // Clear USB2_Power_Acc
                    LED4_ON
                }
            }
            /* USB3 socket */
```

```
            if(USB3_En) {
                if(USB3_Init) {
                    /* To mesaure the USB3 current */
                    Measure_USB3();
                } else {
                    /* To do the initialization for USB3 socket */
                    USB3_PWR_EN = 1;        // Power-on USB3 socket
                    USB3_uA_EN = 1;         // Enable uA switch (1k ohm)
                    EADC_ConfigSampleModule(EADC, 2, EADC_SOFTWARE_TRIGGER, USB3_uA_CH);
// Configure EADC0_SM2 for CH6
                    Measure_USB3();         // Mesaure USB3 current
                    USB3_Init = 1;          // USB3 initial done
                    ZC_Cnt_USB3 = 0;        // Clear ZC_Cnt_USB3
                    USB3_Pwr_Unit = 0;      // Clear USB3_Pwr_Unit
                    USB3_Power = 0;         // Clear USB3_Power
                    USB3_Power_Acc = 0;     // Clear USB3_Power_Acc
                    LED5_ON
                }
            }

            /*=== AC1~3 current and power consumption measuring ===*/
            /* AC1 socket */
            if(AC1_En) {
                if(AC1_Init) {
                    /* Configure the sample module 3 for analog input channel 9 and enable
Timer1 trigger source */
                    EADC_ConfigSampleModule(EADC, 3, EADC_TIMER1_TRIGGER, 9);
                    /* To measure the AC1 current */
                    Measure_AC1();
                } else {
                    /* To do the initialization for AC1 */
                    AC1_EADC_Init();        // EADC1_SM3 intialization for AC1
                    AC1_RLY_ON              // Turn-on AC1 socket relay
                    TIMER_Start(TIMER1);    // Start Timer1
                    ZC_Cnt_AC1 = 0;         // Clear ZC_Cnt_AC1
                    AC1_Pwr_Unit = 0;       // Clear AC1_Pwr_Unit
                    AC1_Power = 0;          // Clear AC1_Power
                    AC1_Power_Acc = 0;      // Clear AC1_Power_Acc
                    LED0_ON
                }
            }
```

```
            /* AC2 socket */
            if(AC2_En) {
                if(AC2_Init) {
                    /* To measure the AC2 current */
                    Measure_AC2();
                } else {
                    /* To do the initialization for AC2 */
                    AC2_EADC_Init();        // EADC2_SM4 intialization for AC2
                    AC2_RLY_ON              // Turn-on AC2 socket relay
                    TIMER_Start(TIMER2);    // Start Timer2
                    ZC_Cnt_AC2 = 0;         // Clear ZC_Cnt_AC2
                    AC2_Pwr_Unit = 0;       // Clear AC2_Pwr_Unit
                    AC2_Power = 0;          // Clear AC2_Power
                    AC2_Power_Acc = 0;      // Clear AC2_Power_Acc
                    LED1_ON
                }
            }
            /* AC3 socket */
            if(AC3_En) {
                if(AC3_Init) {
                    /* To measure the AC2 current */
                    Measure_AC3();
                } else {
                    /* To do the initialization for AC3 */
                    AC3_EADC_Init();        // EADC3_SM5 intialization for AC3
                    AC3_RLY_ON              // Turn-on AC3 socket relay
                    TIMER_Start(TIMER3);    // Start Timer3
                    ZC_Cnt_AC3 = 0;         // Clear ZC_Cnt_AC3
                    AC3_Pwr_Unit = 0;       // Clear AC3_Pwr_Unit
                    AC3_Power = 0;          // Clear AC3_Power
                    AC3_Power_Acc = 0;      // Clear AC3_Power_Acc
                    LED2_ON
                }
            }
        }

        /*=== Waiting for DALI responsed ===*/
        if(DALI_SetOnOff_Cmd) {
            /* Response DALI 'SET_ON_OFF' command to LCD */
            if(Timer0_Cnt <= 2) {
                if(bUartDataReady_DALI) {
```

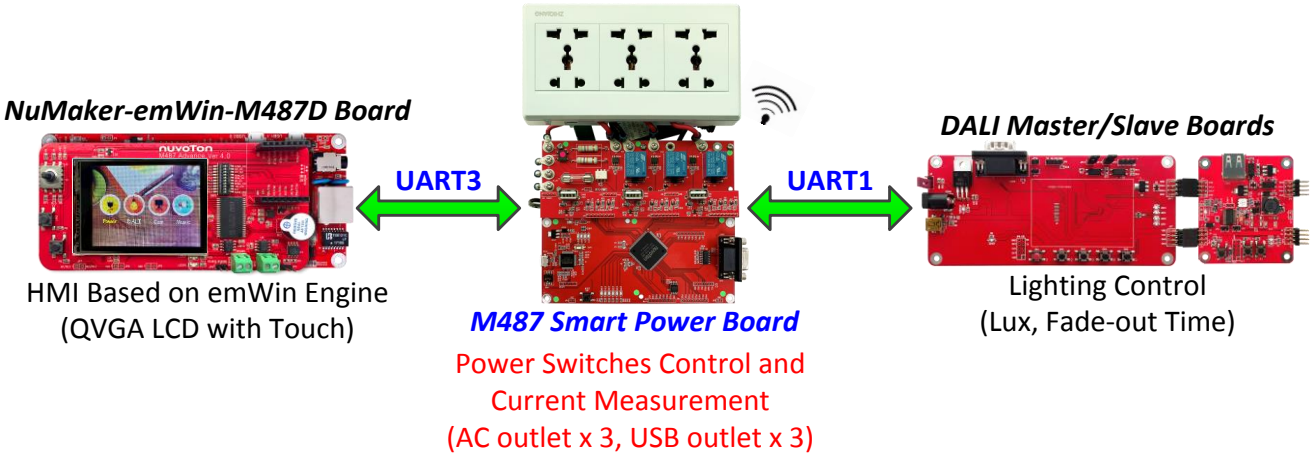
```
                    bUartDataReady_DALI = 0;
                    DALI_SetOnOff_Cmd = 0;
                    ParseCommand_DALI();
                    printf("DALI Connected.\n");
                } else if(Timer0_Cnt == 2) {
                    DALI_SetOnOff_Cmd = 0;
                    response_buff_LCD[1] = pass_buff_DALI[1];
                    response_buff_LCD[2] = pass_buff_DALI[2];
                    response_buff_LCD[3] = NC;
                    response_buff_LCD[4] = pass_buff_DALI[4];
                    response_buff_LCD[5] = pass_buff_DALI[5];
                    PutString_to_LCD();
                    printf("DALI Not Connected!!!\n");
                }
            }
        }
    }

}
```

3 Software and Hardware Environment

- **Software Environment**
  - BSP version
    - M480 Series BSP CMSIS V3.03.001
  - IDE version
    - Keil uVersion 4.74
- **Hardware Environment**
  - Circuit components
    - M487 Smart Power board
  - Diagram

The following figure shows the hardware block diagram of the M487 Intelligent Power Station reference design.



Hardware Block Diagram

4 Directory Information

- EC_M480_Intelligent_Power_Station_V1.00
  - Library — Sample code header and source code files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header files
    - StdDriver — All peripheral driver header and source code files
  - SampleCode
    - ExampleCode — Source code files of this example code

5 How to Execute Example Code

1. Browsing into the sample code folder by Directory Information (section 4) and double click this sample code project M480_Intelligent_Power_Station.uvproj.
2. Enter Keil compile mode.
   a. Build.
   b. Download.
   c. Start/Stop debug session.
3. Enter debug mode.
   a. Run.

6 Revision History

Date	Revision	Description
Aug. 19, 2019	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.