M480 PLC Pulse Ramp

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	本範例代碼使用 M480 EPWM 與 PDMA Scatter-gather 模式來自動產生 PLC pulse ramp 梯形圖的波形圖
BSP Version	M480 Series BSP CMSIS V3.03.000
Hardware	NuMaker-PFM-M487 V3.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

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

www.nuvoton.com

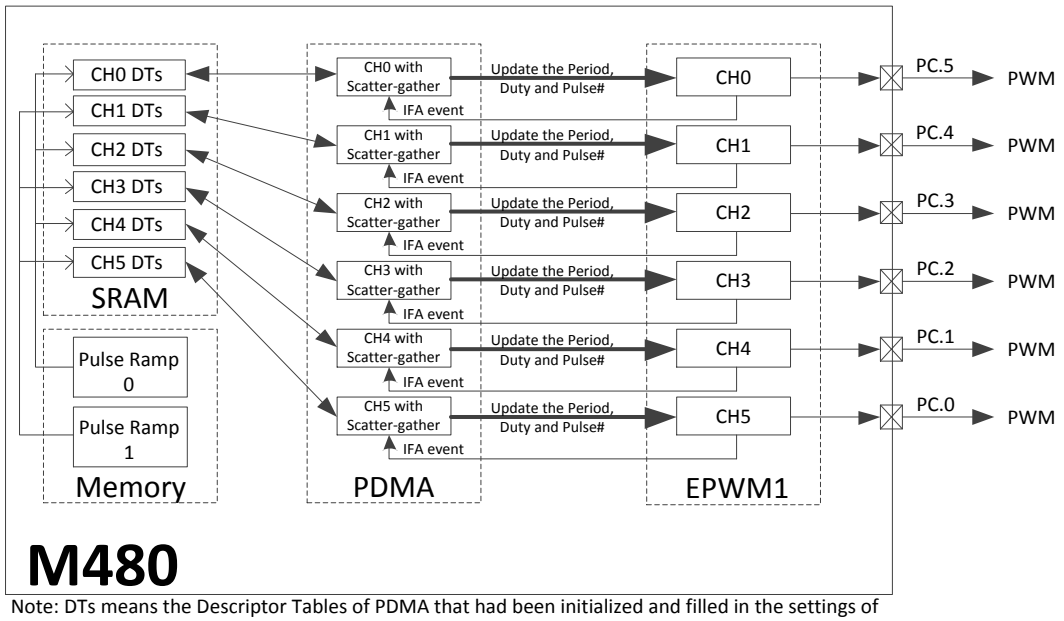
1 功能描述

1.1 介紹

本範例代碼使用 M480 EPWM 模塊與 PDMA 的 Scatter-gather 模式來產生 PLC 的 pulse ramp 梯形圖，並以示波器來觀測 PWM 的輸出波形。

1.2 原理

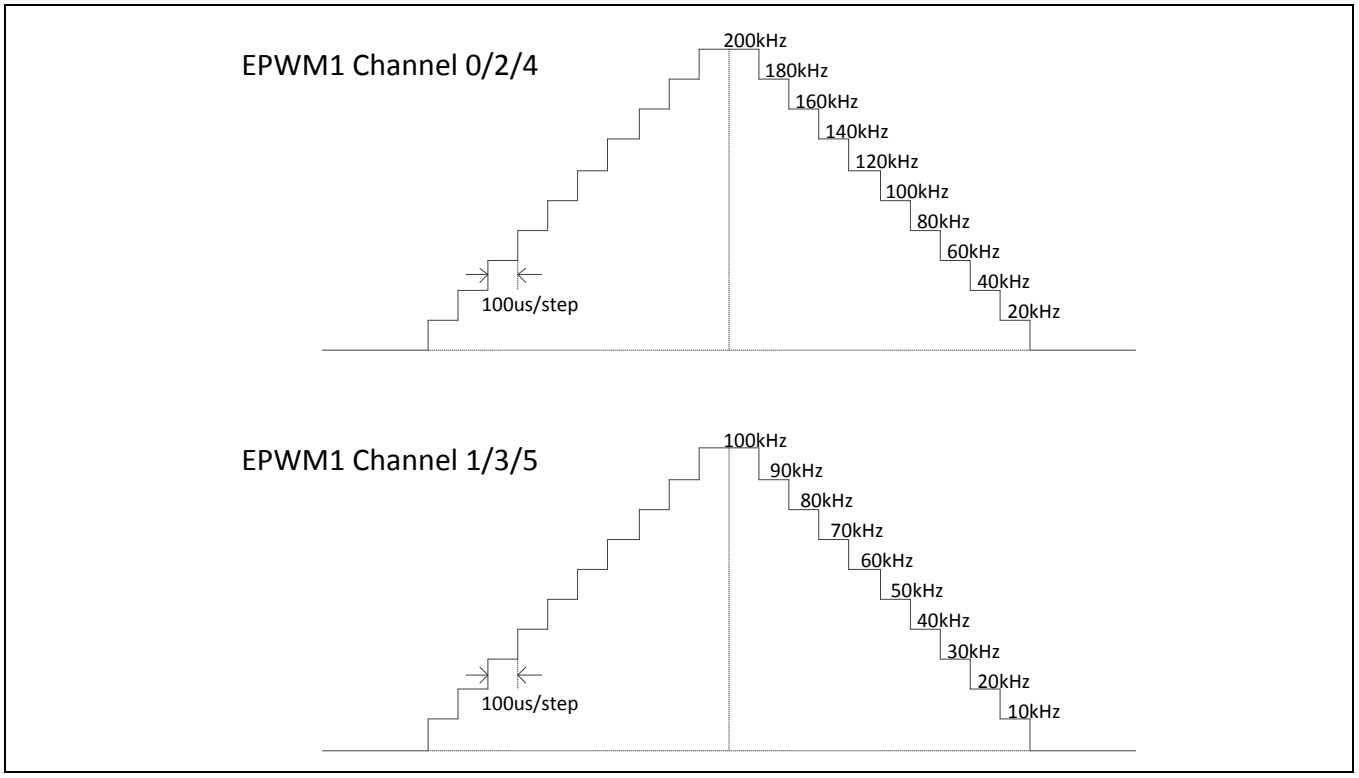
範例代碼中使用了 EPWM1 模塊的 6 個通道，分別為通道 0~5 (PC.5~PC.0)，並利用 EPWM 模塊內的 Interrupt Flag Accumulator (IFA) 中斷旗標計數的事件分別來觸發 PDMA 模塊的通道 0~5，而 PDMA 與其對應的每個通道都已事先規劃成 Scatter-gather 模式並已初始化它的描述表 (Descriptor Tables)，每個 PDMA 通道再利用這些描述表來自動更新下一個步驟 (step) 與其對應的 PWM 通道所要輸出的 period (週期 = 頻率的倒數)、duty (佔空比) 與 pulse 數 (脈衝數) 來達到自動產生 PLC 應用常見的 pulse ramp 梯形圖的 PWM 波形，M480 MCU 的 PDMA 與 EPWM1 功能方塊示意圖如下圖所示。



Note: DTs means the Descriptor Tables of PDMA that had been initialized and filled in the settings of Pulse Ramp 0 or Pulse Ramp 1 from memory by CPU at the initialization of sample code.

M480 PDMA 與 EPWM1 功能方塊示意圖

在此範例代碼中，規劃了兩種 pulse ramp 的梯形圖分別輸出在 EPWM1 的通道 0/2/4 與通道 1/3/5，每個梯形圖上升與下降分別分割成 10 個步驟 (step)，每個步驟 (step) 時間預設值為 100us，頻率差值為最高頻率的十分之一，如下圖所示。



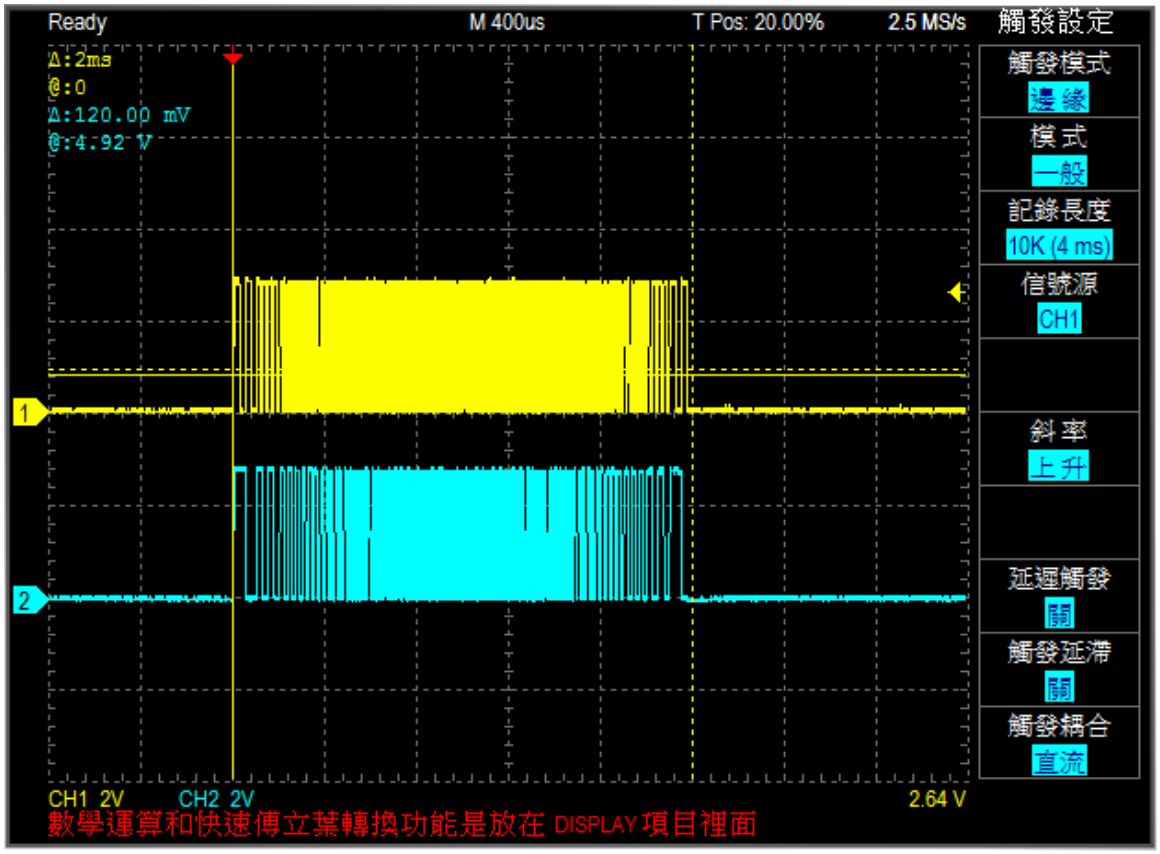
PLC Pulse Ramp 梯形圖

1.3 執行結果

執行代碼後，示波器顯示 EPWM 輸出 pulse ramp 梯形圖的波形圖。

CH1 (黃色) : EPWM1 通道 0 (or 2 or 4)

CH2 (藍色) : EPWM1 通道 1 (or 3 or 5)


Ready
M 400us
T Pos: 20.00%
2.5 MS/s
觸發設定
Δ:2ms
@:0
Δ:120.00 mV
@:4.92 V
觸發模式
邊緣
模式
一般
記錄長度
10K (4 ms)
信號源
CH1
斜率
上升
延遲觸發
關
觸發延滯
關
觸發耦合
直流
CH1 2V
CH2 2V
2.64 V
數學運算和快速傅立葉轉換功能是放在 DISPLAY 項目裡面

示波器顯示的波形圖

2 代碼介紹

Init_PDMA_CHx() 函式主要是對 PDMA 的通道 (0~5) 致能 Scatter-gather 模式與其對應的 Descriptor Tables 做初始化的設定。每一次 EPWM 模塊內的 Interrupt Flag Accumulator (IFA) 中斷旗標計數的事件發生時，就會觸發 PDMA 的通道做下一個步驟 (step) 對應的 EPWM 通道輸出的 period (頻率的倒數)、duty (佔空比) 與 pulse 數 (脈衝數) 做一次性的更新，直到最後的一個步驟 (step) 的 Descriptor Table 設定條件發生時，即產生 PDMA Transfer Done 事件觸發 PDMA 中斷，當進入 PDMA 中斷服務程式後，立即關閉 PWM 的輸出來結束此通道的 pulse ramp 梯形圖。

```
/**
 * @brief       Initial PDMA Channels
 *
 * @param       DMA_CHx: PDMA channel number (0~5)
 *              EPWMx: EPWM0 or EPWM1 module
 *              DescTable: Descriptor Table struct
 *              PeriodTable: Period table
 *              DutyTable: Duty table
 *              PulseTable: Pulse number table
 *
 * @return      None
 *
 * @details     To initial the PDMA channel 0~5 with Scatter-gather mode and their
descriptor tables
 */
void Init_PDMA_CHx(uint32_t DMA_CHx, EPWM_T *EPWMx, DMA_DESC_T *DescTable, uint32_t
*PeriodTable, uint32_t *DutyTable, uint32_t *PulseTable)
{
    uint8_t i;


    /* Enable Scatter Gather mode, assign the first scatter-gather descriptor table is
DescTable 0, and set transfer mode as memory to peripheral */
    if(EPWMx == EPWM0){
        if(DMA_CHx == 0)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM0_CH0_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 1)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM0_CH1_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 2)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM0_CH2_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 3)
```

```
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM0_CH3_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 4)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM0_CH4_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 5)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM0_CH5_TX, 1, (uint32_t)&DescTable[0]);
    }else{
         if(DMA_CHx == 0)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM1_CH0_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 1)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM1_CH1_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 2)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM1_CH2_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 3)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM1_CH3_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 4)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM1_CH4_TX, 1, (uint32_t)&DescTable[0]);
        else if(DMA_CHx == 5)
            PDMA_SetTransferMode(PDMA, DMA_CHx, EPWM1_CH5_TX, 1, (uint32_t)&DescTable[0]);
    }

    /* Inital the Descriptor Tables */
    for(i=0; i<=18; i++){
        /*--- Descriptor Table 0 ---*/
        DescTable[i*3].ctl =
            ((PDMA_TEST_LENGTH - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is
PDMA_TEST_LENGTH */ \
            PDMA_WIDTH_32 |   /* Transfer width is 32 bits(one word) */ \
            PDMA_SAR_FIX |    /* Source increment size is 32 bits(one word) */ \
            PDMA_DAR_FIX |    /* Destination increment size is 32 bits(one word) */ \
            PDMA_REQ_BURST |  /* Transfer type is single transfer type */ \
            PDMA_BURST_1 |    /* Burst size is 1. No effect in single transfer type */ \
            PDMA_TBINTDIS_DISABLE |/* Disable transfer done and table empty interrupt */ \
            PDMA_OP_SCATTER;  /* Operation mode is scatter-gather mode */

        /* Configure source address */
        DescTable[i*3].src = (uint32_t)&(PeriodTable[i]);
        /* Configure destination address */
        DescTable[i*3].dest = (uint32_t)&(EPWMx->PERIOD[DMA_CHx]);
        /* Configure next descriptor table address */
        DescTable[i*3].offset = (uint32_t)&DescTable[i*3+1] - (PDMA->SCATBA); /* next
descriptor table is table 1 */
```

```
        /*--- Descriptor Table 1 ---*/
        DescTable[i*3+1].ctl =
            ((PDMA_TEST_LENGTH - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is
PDMA_TEST_LENGTH */ \
            PDMA_WIDTH_32 |   /* Transfer width is 32 bits(one word) */ \
            PDMA_SAR_FIX |    /* Source increment size is 32 bits(one word) */ \
            PDMA_DAR_FIX |    /* Destination increment size is 32 bits(one word) */ \
            PDMA_REQ_BURST |  /* Transfer type is single transfer type */ \
            PDMA_BURST_1 |    /* Burst size is 1. No effect in single transfer type */ \
            PDMA_TBINTDIS_DISABLE |/* Disable transfer done and table empty interrupt */ \
            PDMA_OP_SCATTER;  /* Operation mode is scatter-gather mode */

        /* Configure source address */
        DescTable[i*3+1].src = (uint32_t)&(DutyTable[i]);
        /* Configure destination address */
        DescTable[i*3+1].dest = (uint32_t)&(EPWMx->CMPDAT[DMA_CHx]);
        /* Configure next descriptor table address */
        DescTable[i*3+1].offset = (uint32_t)&DescTable[i*3+2] - (PDMA->SCATBA); /* next
descriptor table is table 2 */

        /*--- Descriptor Table 2 ---*/
        DescTable[i*3+2].ctl =
            ((PDMA_TEST_LENGTH - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is
PDMA_TEST_LENGTH */ \
            PDMA_WIDTH_32 |   /* Transfer width is 32 bits(one word) */ \
            PDMA_SAR_FIX |    /* Source increment size is 32 bits(one word) */ \
            PDMA_DAR_FIX |    /* Destination increment size is 32 bits(one word) */ \
            PDMA_REQ_SINGLE | /* Transfer type is single transfer type */ \
            PDMA_BURST_1 |    /* Burst size is 1. No effect in single transfer type */ \
            PDMA_TBINTDIS_DISABLE |/* Disable transfer done and table empty interrupt */ \
            PDMA_OP_SCATTER;  /* Operation mode is scatter-gather mode */

        /* Configure source address */
        DescTable[i*3+2].src = (uint32_t)&(PulseTable[i]);
        /* Configure destination address */
        DescTable[i*3+2].dest = (uint32_t)&(EPWMx->IFA[DMA_CHx]);
        /* Configure next descriptor table address */
        DescTable[i*3+2].offset = (uint32_t)&DescTable[i*3+3] - (PDMA->SCATBA); /* next
descriptor table is table 3 */
    }
```

```
    /*--- The last Descriptor Table ---*/
    DescTable[i*3].ctl =
        ((PDMA_TEST_LENGTH - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is
PDMA_TEST_LENGTH */ \
        PDMA_WIDTH_32 |   /* Transfer width is 32 bits(one word) */ \
        PDMA_SAR_FIX |    /* Source increment size is 32 bits(one word) */ \
        PDMA_DAR_FIX |    /* Destination increment size is 32 bits(one word) */ \
        PDMA_REQ_SINGLE | /* Transfer type is single transfer type */ \
        PDMA_BURST_1 |    /* Burst size is 1. No effect in single transfer type */ \
        PDMA_OP_BASIC;    /* Operation mode is Basic mode */

    /* Configure source address */
    DescTable[i*3].src = (uint32_t)&(PeriodTable[i]);
    /* Configure destination address */
    DescTable[i*3].dest = (uint32_t)&(EPWMx->PERIOD[DMA_CHx]);
    /* Configure next descriptor table address */
    DescTable[i*3].offset = 0;

    /* Enable PDMA Transfer Done interrupt */
    PDMA_EnableInt(PDMA, DMA_CHx, PDMA_INT_TRANS_DONE);

    printf("Initial PDMA Channel %d done.\n", DMA_CHx);

}
```

Init_EPWM_CHx() 函式主要是對 EPWM 模塊內的通道 (0~5) 做初始化的設定。此初始化須先給定每個 EPWM 通道在 pulse ramp 梯形圖中第一個步驟 (step) 所要輸出的 period (頻率的倒數)、duty (佔空比) 與 pulse 數 (脈衝數)，並致能 EPWM 模塊內的 Interrupt Flag Accumulator (IFA) 中斷旗標計數的功能來觸發 PDMA 對應通道的 Scatter-gather 模式下的 Descriptor Tables 所要執行的相對動作。

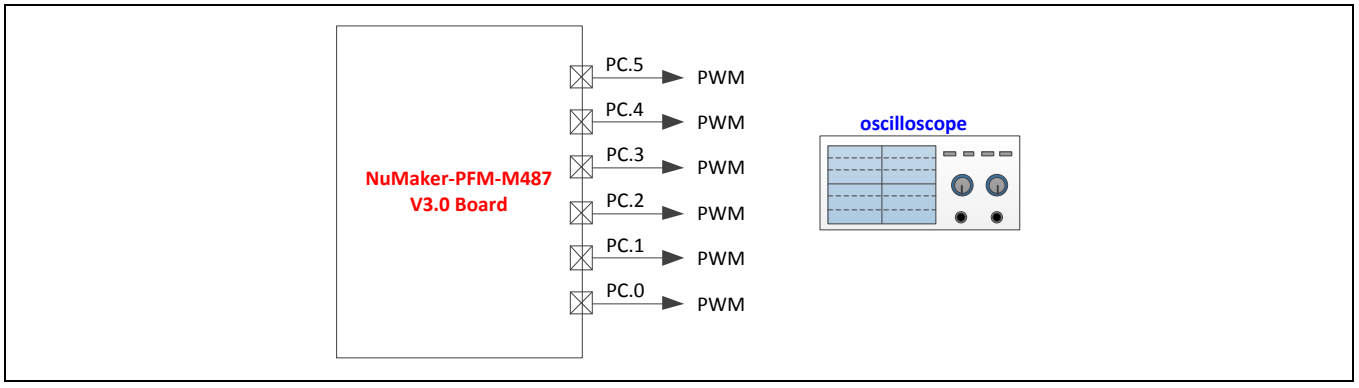
```
/**
 * @brief       Initial EPWM Channels
 *
 * @param       EPWMx: EPWM0 or EPWM1 module
 *              PWM_CHx: PWM channel number (0~5)
 *              Freq_1st: The frequency at the first step
 *              Duty_1st: The duty at the first step (In generally, it is 50%.)
 *              Pulse_1st: The pulse number at the first step
```

```
 *
 * @return      None
 *
 * @details     To initial frequency, duty and pulse number of EPWM channel 0~5 for the first step
 */
void Init_EPWM_CHx(EPWM_T *EPWMx, uint32_t PWM_CHx, uint32_t Freq_1st, uint32_t Duty_1st, uint32_t Pulse_1st)
{
    /* Set EPWMx channel x output configuration */
    EPWM_ConfigOutputChannel(EPWMx, PWM_CHx, Freq_1st, Duty_1st);

    /* Enable EPWM Output path for EPWMx channel x */
    EPWM_EnableOutput(EPWMx, 1 << PWM_CHx);

    /* Enable EPWMx channel x accumulator, interrupt count 1, accumulator source select to period point */
    EPWM_EnableAcc(EPWMx, PWM_CHx, Pulse_1st, EPWM_IFA_ZERO_POINT);

    /* Enable EPWMx channel x accumulator interrupt */
    EPWM_EnableAccInt(EPWMx, PWM_CHx);

    /* Enable EPWMx channel x accumulator interrupt trigger PDMA */
    EPWM_EnableAccPDMA(EPWMx, PWM_CHx);

#ifdef EN_EPWM_INDEP
    /* Enable Timer for EPWMx channel x */
    EPWM_Start(EPWMx, 1 << PWM_CHx);
#endif

    printf("Initial EPWM%d Channel %d done.\n", (EPWMx == EPWM1 ? 1 : 0), PWM_CHx);

}
```

3 軟體和硬體環境

- **軟體環境**
  - BSP 版本
    - M480 Series BSP CMSIS V3.03.000
  - IDE 版本
    - Keil uVersion 4.74
- **硬體環境**
  - 電路元件
    - NuMaker-PFM-M487 V3.0 開發板
  - 示意圖

M480 EPWM1 的通道 0~5 (亦即 PC.5~PC.0) 輸出 PLC pulse ramp 梯形圖的 PWM 波形，利用示波器來檢測 PWM 的波形。



硬體示意圖

4 目錄資訊

- EC_M480_PLC_PulseRamp_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 如何執行範例程式

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 M480_PLC_PulseRamp.uvproj
2. 進入編譯模式介面
   a. 編譯
   b. 下載代碼至記憶體
   c. 進入 / 離開除錯模式
3. 進入除錯模式介面
   a. 執行代碼

6 修訂紀錄

Date	Revision	Description
June 28, 2019	1.00	1. 初始發布.

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.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.