M480 PLC Pulse Ramp

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This example code uses the M480 EPWM module that combined with the PDMA Scatter-gather mode to generate the PWM output waveforms of PLC pulse ramp automatically.
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

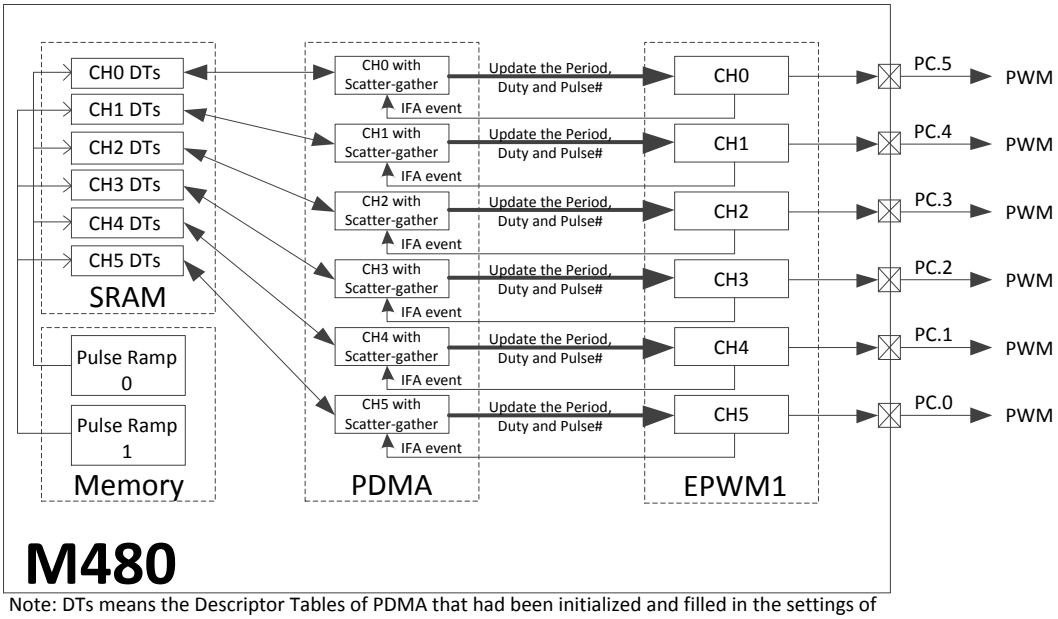
1 Function Description

1.1 Introduction

This example code utilizes the M480 EPWM module that combined with the PDMA Scatter-gather mode to generate the PWM output waveforms of PLC pulse ramp automatically and uses the oscilloscope to monitor these PWM output waveforms.

1.2 Principle

This example code total uses 6 PWM output channels of EPWM1 module, the channel 0 to channel 5 that are mapping to the PC.5~PC.0 pins on M480 chip, and also uses the Interrupt Flag Accumulator (IFA) mechanism of EPWM1 module to count the interrupt events to trigger the channel 0 to channel 5 of PDMA module individually. In the beginning of this example source code, the codes do the initialization to configure each channel of these 6 PDMA channels as Scatter-gather mode and its related Descriptor Tables. Each of PDMA channels utilizes these Descriptor Tables to update the period (1/frequency), duty and pulse numbers of the next step and to generate the PWM output waveforms of pulse ramp automatically that often seen in general PLC application field. The function block diagram of EPWM1 that combined with PDMA scatter-gather mode is shown below.

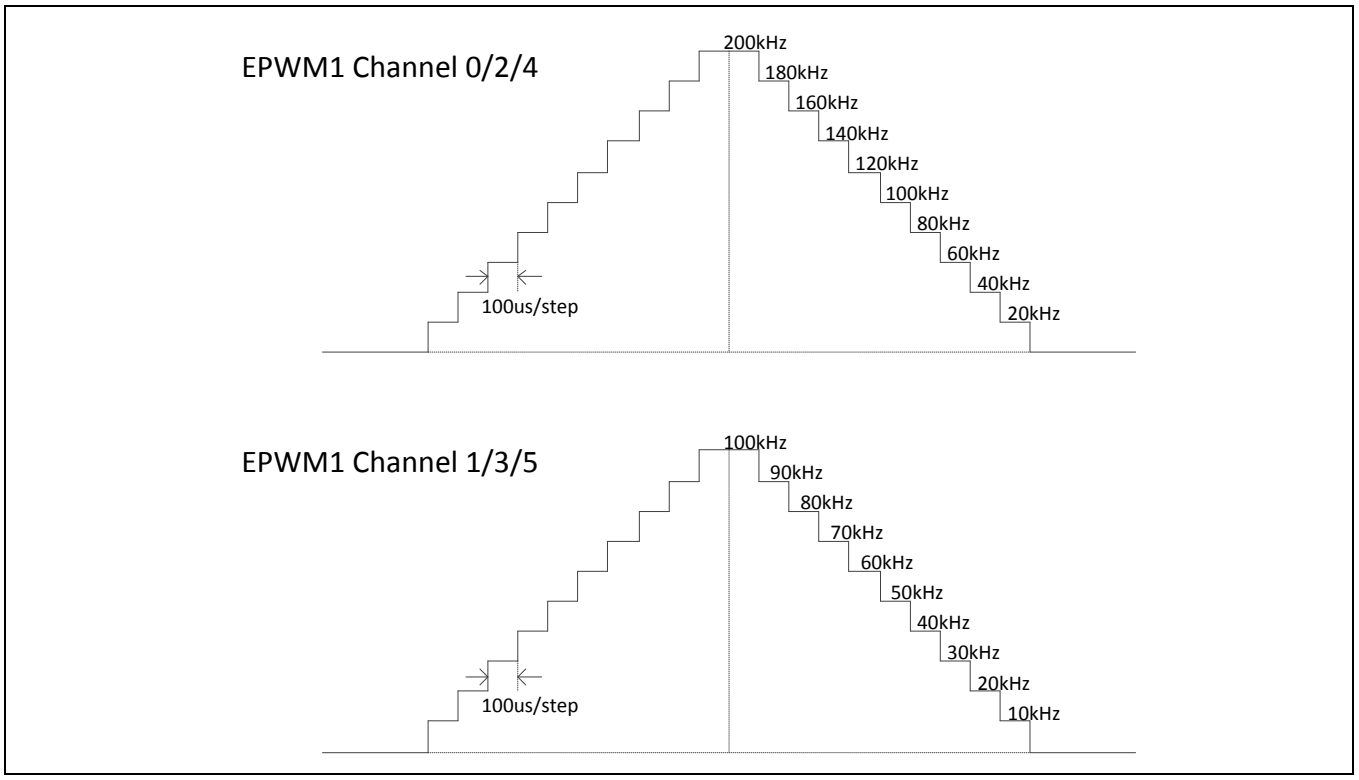


Note: DTs means the Descriptor Tables of PDMA that had been initialized and filled in the settings of Pulse Ramp 0 or Pulse Ramp 1 from memory by CPU at the initialization of sample code.

The Function Block Diagram of EPWM1 that Combined with PDMA Scatter-gather Mode

In this sample code, the ramp diagrams of two pulse ramps are planned to be output on

channels 0/2/4 and channels 1/3/5 of EPWM1, and each ramp rise and fall is divided into 10 steps. The time preset value of each step is 100us and the frequency difference is one tenth of the highest frequency, as shown in the following figure.



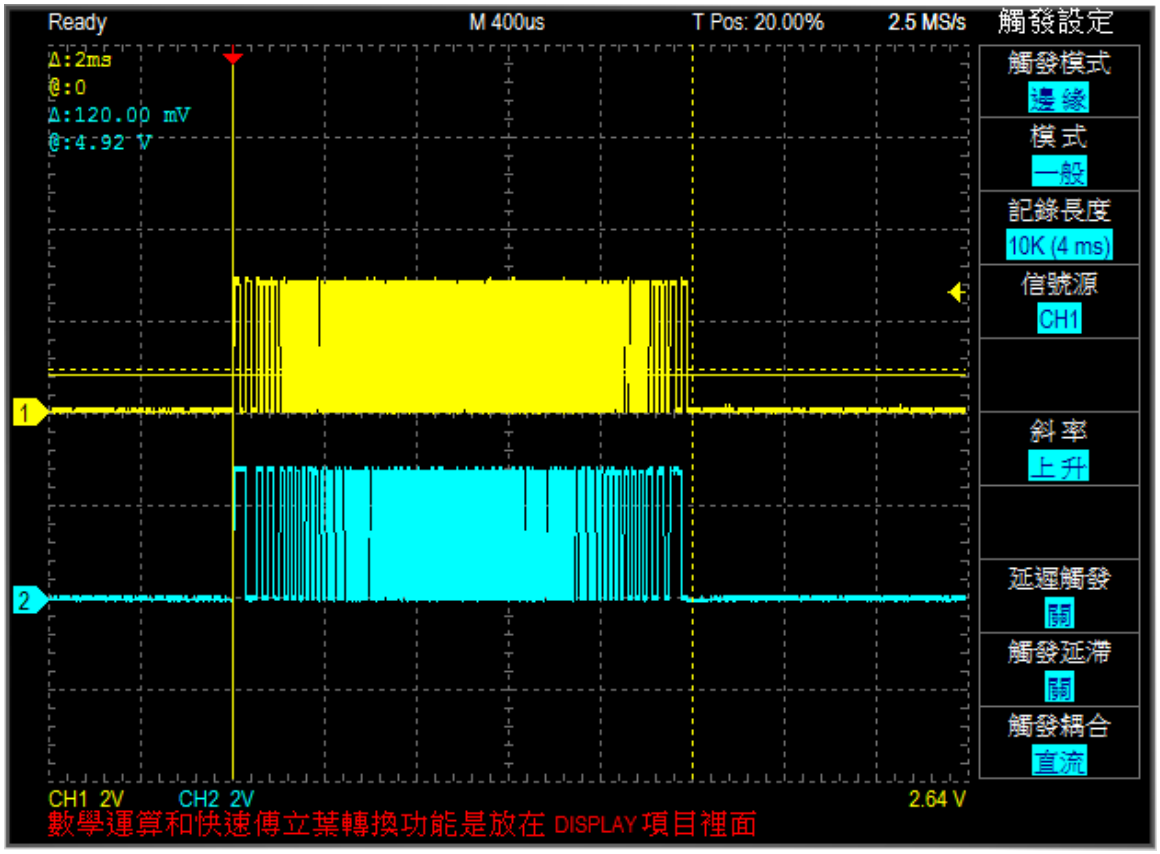
PLC Pulse Ramp

1.3 Demo Result

After executed this example code, the PWM output waveforms of pulse ramp are shown on the oscilloscope.

CH1 (Yellow) : EPWM1 channel 0 (or 2 or 4)

CH2 (Blue) : EPWM1 channel 1 (or 3 or 5)



The PWM Output Waveforms on Oscilloscope

2 Code Description

The Init_PDMA_CHx() function is mainly used to initialize the PDMA channels (0~5) to enable the Scatter-gather mode and its corresponding Descriptor Tables. Each time the interrupt flag count IFA event of EPWM channel occurs, then the corresponding PDMA channel is triggered to update the period, duty and pulse number of the EPWM output channel for the next step. When the setting condition of Descriptor Table of the last step occurs, the PDMA Transfer Done event is generated to trigger the PDMA interrupt. When entering the PDMA interrupt service routine, the source code immediately turns off the PWM output to end the pulse ramp for this channel.

```
/**
 * @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);

}
```

The following Init_EPWM_CHx() function is used to initialize the channel (0~5) in the EPWM module. This initialization must first configure the period, duty and pulse number of PWM waveform to be output in the first step of each EPWM channel in the pulse ramp, and enable the interrupt flag count IFA function in the PWM channel of EPWM module. Finally, the IFA event will trigger the relative action to be performed that described by the Descriptor Tables of the PDMA channel under the Scatter-gather mode.

```
/**
 * @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 Software and Hardware Environment

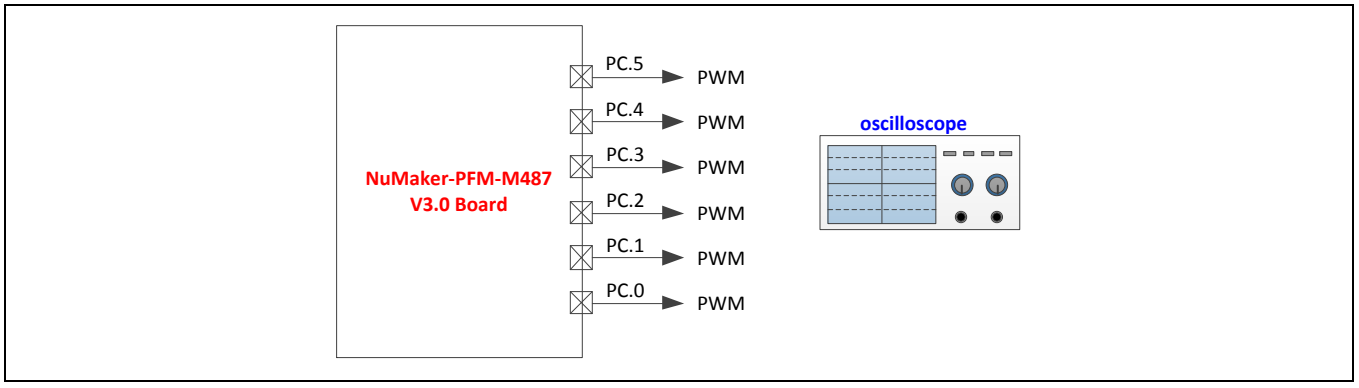
Software Environment

- BSP version
  - M480 Series BSP CMSIS V3.03.000
- IDE version
  - Keil uVersion 4.74

Hardware Environment

- Circuit components
  - NuMaker-PFM-M487 V3.0 board
- Diagram

The channel 0~5 (PC.5~PC.0 pins) of EPWM1 on M480 MCU will output the PWM waveforms of PLC pulse ramp and users can monitor these PWM waveforms on oscilloscope.



Hardware Diagram

4 Directory Information

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 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_PLC_PulseRamp.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
June 28, 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.

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.