M030G Rapid Pulse Shaping Output Demo

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M030G to achieve rapid pulse shaping output.
BSP Version	M029G_M030G_M031G_Series_BSP_CMSIS_V3.03.000
Hardware	NuMaker-M030GTD V1.1

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 and microprocessor 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. Overview

In some applications, there is a need to quickly change the irregular input pulse to a fixed pulse. If the I/O port of MCU is used to detect the input pulse, then output a fixed pulse, the output pulse cannot be generated quickly, and if the input pulse is more frequent, it will occupy a large amount of CPU resources.

1.1 Principle

This example code sets Timer0 to the event count function and enables time-out to trigger BPWM and PDMA. When Timer0 detects pulse input, it will trigger BPWM and PDMA at the same time. Because BPWM responds quickly, it will first output high level, and then PDMA will pull it to low to generate pulse with fixed width. The general principle is shown in Figure 1-1.

In this way, during operation, the MCU will automatically detect the input pulse and output the pulse with a fixed width, with all hardware completion, without the intervention of the CPU.

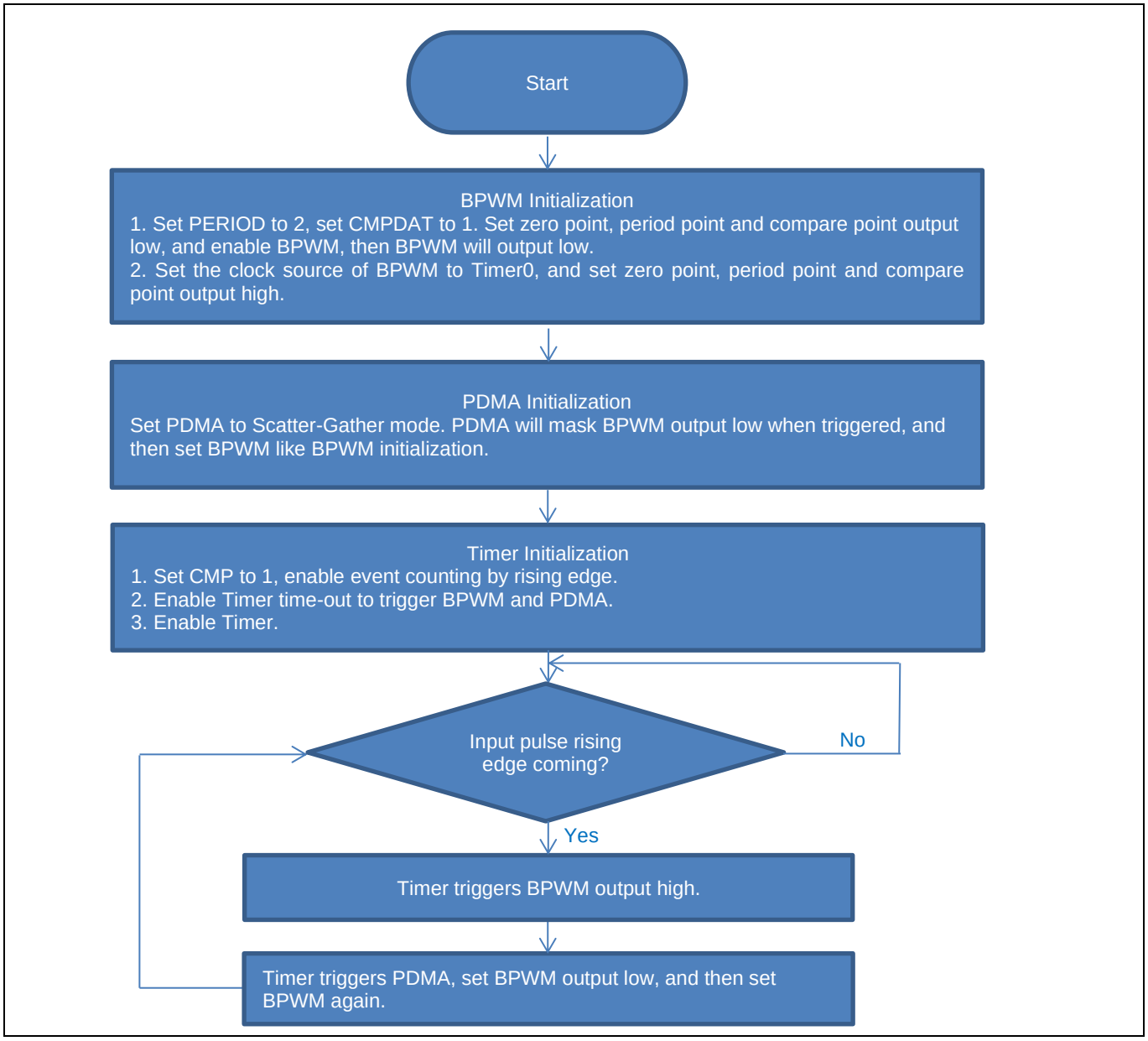

Figure 1-1 Rapid Pulse Shaping Output Flow Diagram

1.2 Demo Result

This example code uses PB4 output pulse to simulate the pulse that needs shaping. PB5 (TM0) input the pulse before shaping, and PF3 (BPWM1_CH0) output the pulse after shaping. For testing, PB4 and PB5 (TM0) need to be connected together. The operation results are shown in Figure 1-2. (Channel 1: input pulse; Channel 2: Output pulse).

As can be seen from Figure 1-2, the time between the edge of the input pulse to the edge of the output pulse is about 104 ns, and the width of the output pulse is about 420 ns.

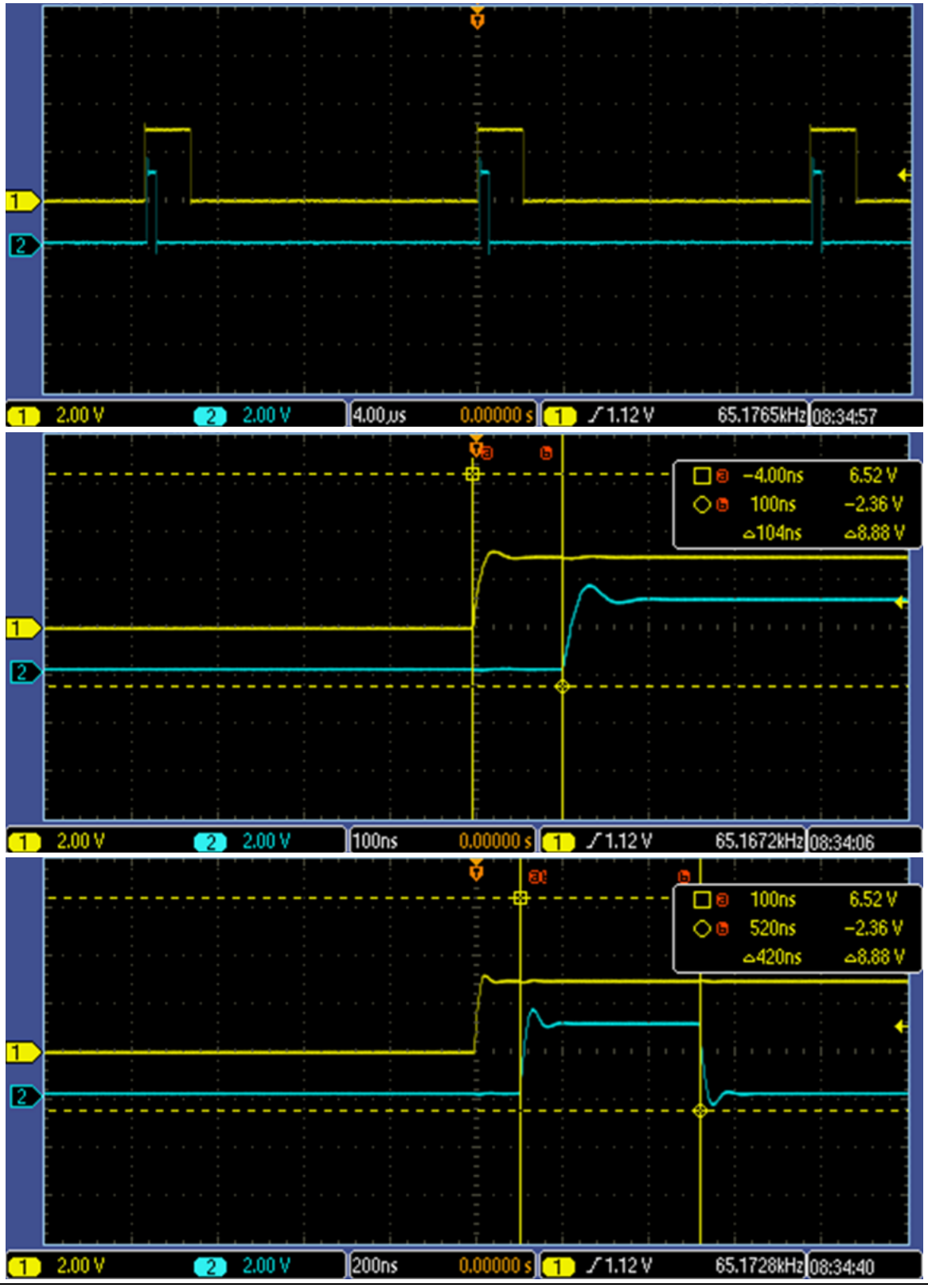

Figure 1-2 Operation Result

2. Code Description

Enable Timer0 event counting function and trigger BPWM and PDMA by time-out.

```
void TIMER0_Init()
{
    /* Set Timer0 to operate in periodic mode */
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 1);

    /* Set Timer0 prescale value */
    TIMER_SET_PRESCALE_VALUE(TIMER0, 0);

    /* Set Timer0 compared value */
    TIMER_SET_CMP_VALUE(TIMER0, 1);

    /* Counter increase on rising edge */
    TIMER_EnableEventCounter(TIMER0, TIMER_COUNTER_RISING_EDGE);

    /* Set trigger source is from Timer0 time-out interrupt signal */
    TIMER_SetTriggerSource(TIMER0, TIMER_TRGSRC_TIMEOUT_EVENT);

    /* Enalbe Timer0 trigger BPWM and PDMA */
    TIMER_SetTriggerTarget(TIMER0, TIMER_TRG_TO_BPWM | TIMER_TRG_TO_PDMA);
}
```

Set BPWM1 zero point, period point, compare up point and down point output low, set the clock source of BPWM1 to BPWM_CLK, and BPWM1 will output low when BPWM1 runs. Then set the clock source to Timer0, set BPWM1 zero point, period point, and compare up point and down point output high.

```
void BPWM_Init(void)
{
    /* Set BPWM1 period */
    BPWM_SET_CNR(BPWM1, 0, 2);

    /* Set BPWM1 duty */
    BPWM_SET_CMR(BPWM1, 0, 1);

    /* Enable output of BPWM1 channel 0 */
    BPWM_EnableOutput(BPWM1, BIT0);

    /* Start BPWM1 counter */
    BPWM_Start(BPWM1, BIT0);

    /* Set BPWM1 zero point, period point, compare up point and compare down point output
    low */
    BPWM_SET_OUTPUT_LEVEL(BPWM1, BIT0, BPWM_OUTPUT_LOW, BPWM_OUTPUT_LOW, BPWM_OUTPUT_LOW,
    BPWM_OUTPUT_LOW);

    /* Set BPWM1 clock source to BPWM_CLK */
    BPWM_SetClockSource(BPWM1, 0, BPWM_CLKSRC_BPWM_CLK);

    /* Set BPWM1 clock source to TIMER0 */
    BPWM_SetClockSource(BPWM1, 0, BPWM_CLKSRC_TIMER0);
```

```
    /* Set BPWM1 zero point, period point, compare up point and compare down point output
    high */
    BPWM_SET_OUTPUT_LEVEL(BPWM1, BIT0, BPWM_OUTPUT_HIGH, BPWM_OUTPUT_HIGH,
    BPWM_OUTPUT_HIGH, BPWM_OUTPUT_HIGH);
}
```

Set PDMA to Scatter-Gather mode, trigger by Timer0. It will enable the mask function of BPWM1 and mask BPWM1 output low when triggered, and then set BPWM1 like BPWM1 initialization.

```
void PDMA_Init(void)
{
    uint8_t i = 0, j = 1;

    /* Open PDMA */
    PDMA_Open(PDMA, 1 << BPWM_PDMA_CHANNEL);

    /* Set PDMA transfer mode */
    PDMA_SetTransferMode(PDMA, BPWM_PDMA_CHANNEL, PDMA_TMR0, TRUE,
    (uint32_t)&DMA_DESC[0]);

    /* Mask BPWM1 to output low */
    i=0;
    j=1;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_BURST | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32Mask_Enable;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->MSKEN;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);

    /* Set BPWM1 zero point and period point output low */
    i++;
    j++;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_BURST | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32WGCTL0_Low;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->WGCTL0;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);

    /* Set BPWM1 compare up point and compare down point output low */
    i++;
    j++;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_BURST | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32WGCTL1_Low;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->WGCTL1;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);

    /* Set BPWM1 clock source to BPWM_CLK */
    i++;
    j++;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_BURST | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32CLKSRC_CLK;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->CLKSRC;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);
```

```
    /* Set BPWM1 clock source to TIMER0 */
    i++;
    j++;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_BURST | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32CLKSRC_Timer0;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->CLKSRC;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);

    /* Set BPWM1 zero point and period point output high */
    i++;
    j++;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_BURST | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32WGCTL0_High;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->WGCTL0;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);

    /* Set BPWM1 compare up point and compare down point output high */
    i++;
    j++;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_BURST | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32WGCTL1_High;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->WGCTL1;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);

    /* Disable mask */
    i++;
    j = 0;
    DMA_DESC[i].ctl = ((1 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32 | PDMA_SAR_FIX
    | PDMA_DAR_FIX | PDMA_REQ_SINGLE | PDMA_OP_SCATTER;
    DMA_DESC[i].src = (uint32_t)&u32Mask_Disable;
    DMA_DESC[i].dest = (uint32_t)&BPWM1->MSKEN;
    DMA_DESC[i].offset = (uint32_t)&DMA_DESC[j] - (PDMA->SCATBA);
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M029G_M030G_M031G_Series_BSP_CMSIS_V3.03.000
- IDE version
  - Keil uVersion 4.74

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M030GTD V1.1
- Pin Connect

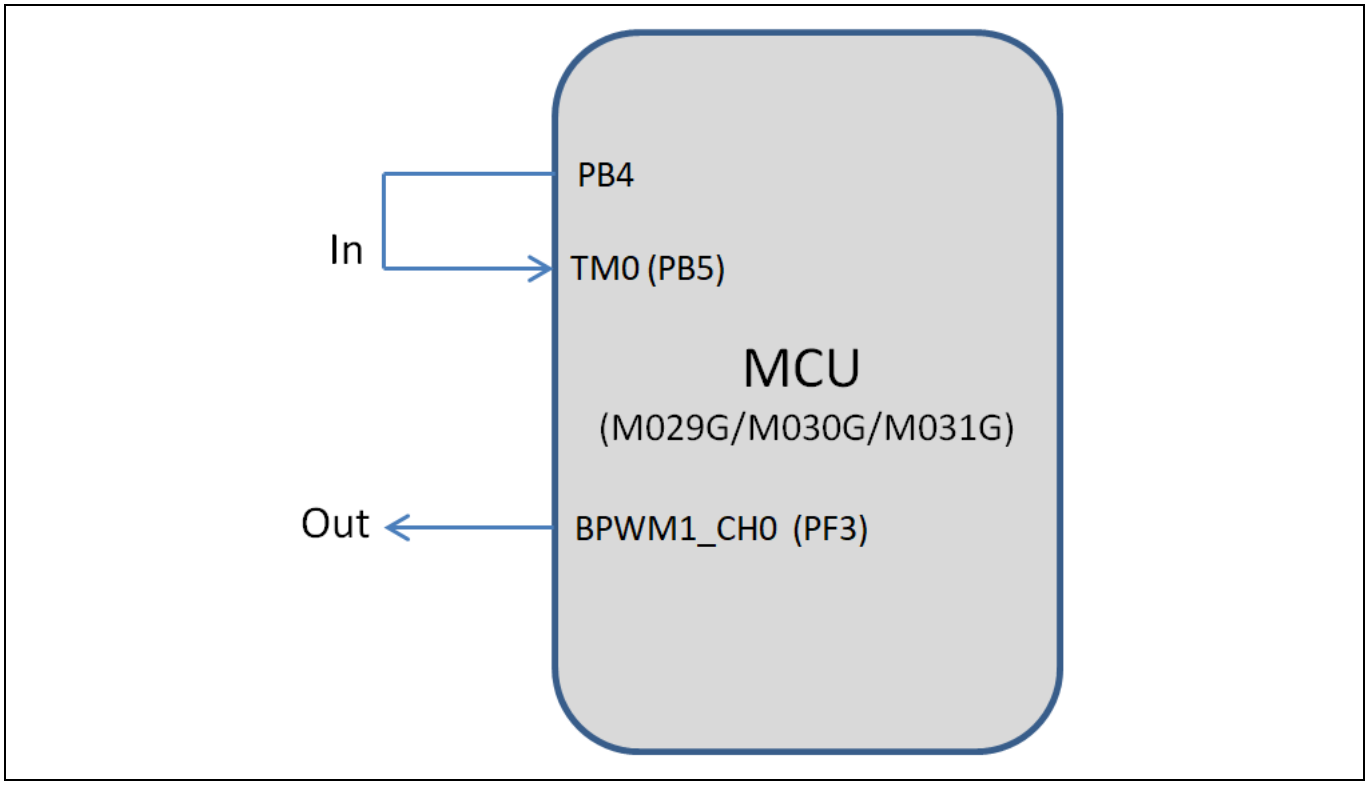

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_M030G_Rapid_Pulse_Shaping_Output_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	
Project	Source file of example code

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *Rapid_Pulse_Shaping_Output.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2024.04.08	1.00	Initial version.

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.