M031/M032 Five Motors Speed Sensing

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code shows how to read operation speed of five motors via four timer captures and one PWM capture.
BSP Version	M031 BSP CMSIS V3.03.000
Hardware	NuMaker-M032KI Ver1.0

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 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

This sample code shows how to read and calculate the speed of five motors via four timer captures and one PWM captures, and print out the calculation result.

Users can connect the fan, PFM1412DEB7V, to the NuMaker-M032KI board. The illustration of the PFM1412DEB7V is shown below.

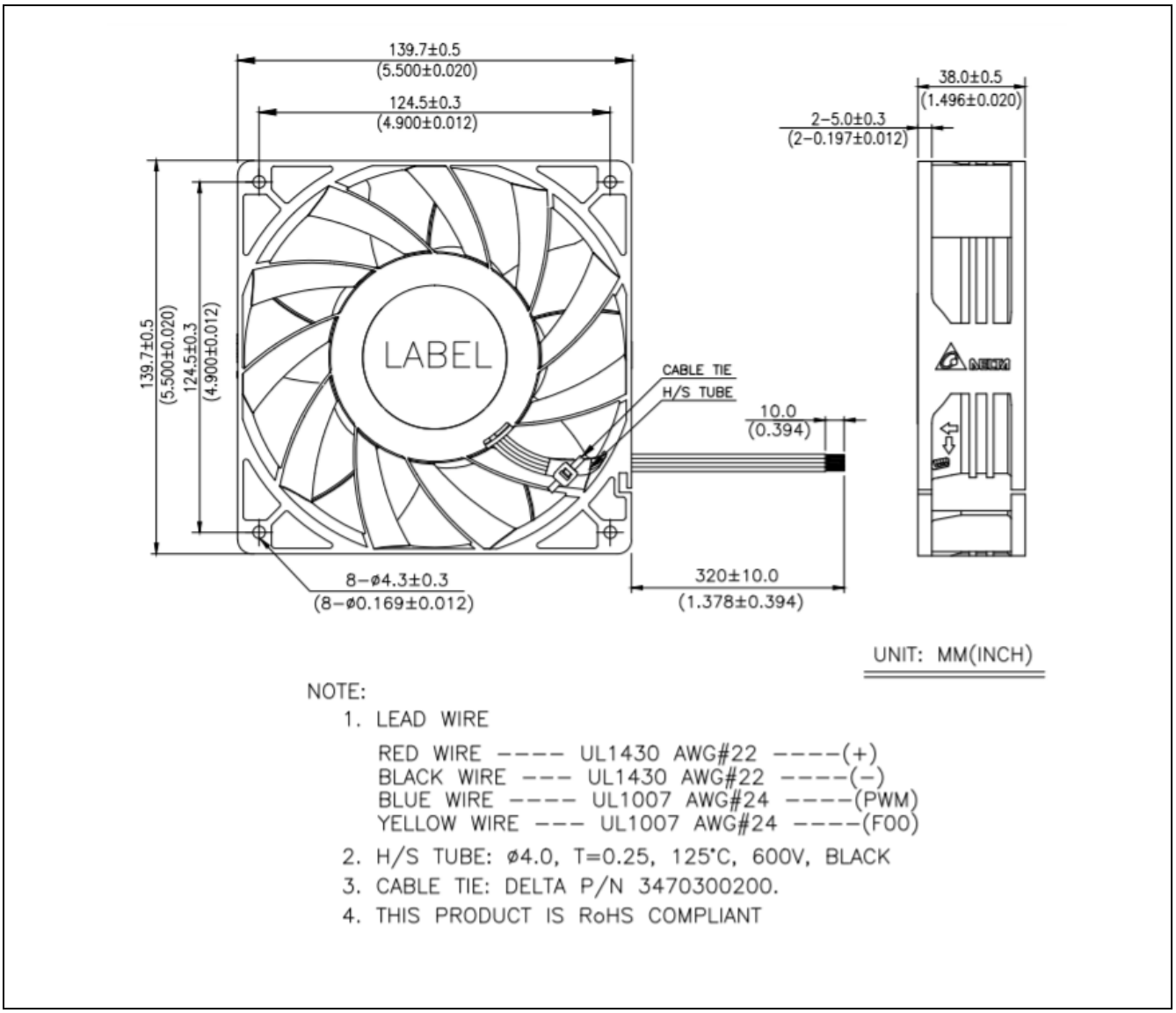

Figure 1-1 PFM1412DEB7V Overview

The motor of PFM1412DEB7V will generate two cycles of square wave in every turn. Refer to Figure 1-2 below. Users can also find more details on the link: https://www.delta-fan.com/Download/Spec/PFM1412DEB7V.pdf.

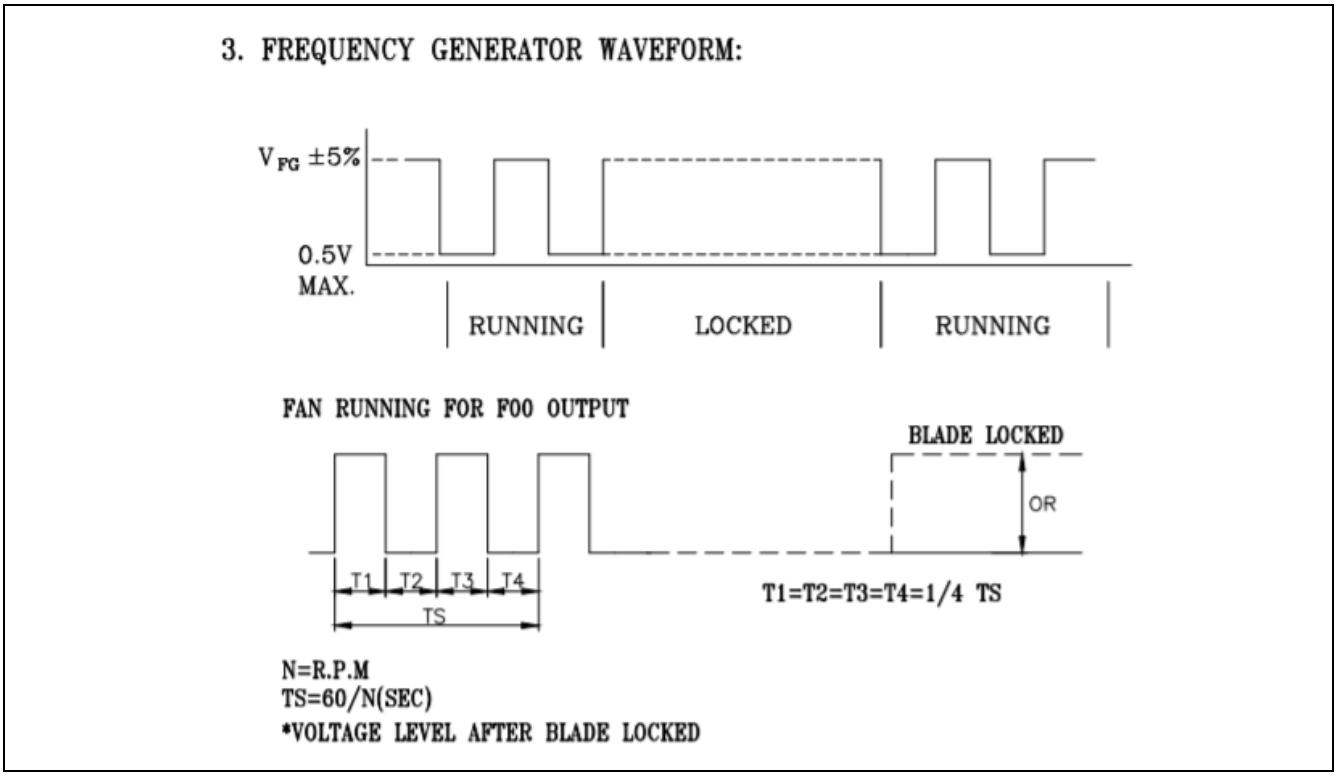

Figure 1-3 Frequency Generator Waveform

1.1 Principle

This sample code uses the M031/M032 series microcontroller to capture and calculate motor speed. The PWM capture has a 16-bit up/down counter, which must set prescale if the signal period is so long that counter overflows.

User can get prescare value by calculating the motor speed, and the calculation method is as follows.

DUTY CYCLE (%)	SPEED R.P.M. (REF)	CURRENT (A) TYP.
100	6500±10%	3.9
50	3460±10%	0.8
10	1000±250	0.12

Table 1-1 PFM1412DEB7V Speed Range

The motor feedback frequency and period in full speed is 216.667Hz and 4.615 ms; the lowest feedback frequency and period is 33.3 Hz and 30 ms.

The worst case is in the lowest speed, and the calculating accuracy set by this sample code is 0.1%, so the capture step of the counter is 30us. (30(ms)/1000).

Finally, the prescale value will be calculated, and prescale value is 1439, and the PWM counting value will be 48000000/1439 =33356 per second.

1.2 Demo Result

The demo result is shown below.

```
motor 1 operation speed is : 16 turns per second.

motor 2 operation speed is : 16 turns per second.

motor 3 operation speed is : 16 turns per second.

motor 4 operation speed is : 16 turns per second.

motor 5 operation speed is : 16 turns per second.
--------------------------------------
```

Figure 1-4 Demo Result

2 Code Description

First, initialize the M031/M032 in main.c, and initialize PWM, UART, and Timer. Then, read and calculate the motor speed.

```
int32_t main(void)
{
    /* Init System, IP clock and multi-function I/O
       In the end of SYS_Init() will issue SYS_LockReg()
       to lock protected register. If user want to write
       protected register, please issue SYS_UnlockReg()
       to unlock protected register if necessary */
    /* Unlock protected registers */
    SYS_UnlockReg();

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

    /* Lock protected registers */
    SYS_LockReg();

    /* Init UART to 115200-8n1 for print message */
    UART0_Init();
    /* Enable Timer1-4 and PWM1 NVIC */
    NVIC_EnableIRQ(TMR0_IRQn);
    NVIC_EnableIRQ(TMR1_IRQn);
    NVIC_EnableIRQ(TMR2_IRQn);
    NVIC_EnableIRQ(TMR3_IRQn);
    NVIC_EnableIRQ(PWM1_IRQn);

    /* set PWM1 channel 0 capture configuration */
    PWM_ConfigCaptureChannel(PWM1, 0, 29980, 0);
    /* Enable Timer for PWM1 channel 0 */
    PWM_Start(PWM1, PWM_CH_0_MASK);
    /* Enable Capture Function for PWM1 channel 0 */
    PWM_EnableCapture(PWM1, PWM_CH_0_MASK);
    PWM_EnableCaptureInt(PWM1, 0, PWM_CAPTURE_INT_FALLING_LATCH);
    /* Enable falling capture reload */
    PWM1->CAPCTL |= PWM_CAPCTL_FCRLDEN0_Msk;

    /* Enable Timer0 event counter input and external capture function */
    /*Open Timer0 with CONTINUOUS_MODE and Frequency*/
    TIMER_Open(TIMER0, TIMER_CONTINUOUS_MODE, 1);
    /*Set Timer0 Prescale = 0, clock will be divided by (prescale + 1)*/
    TIMER_SET_PRESCALE_VALUE(TIMER0, 0);
    /*Set Timer0 Compared Value*/
    TIMER_SET_CMP_VALUE(TIMER0, TIMER_CMP_MAX_VALUE);
    /*Enable Timer0 Capture and set output mode, trigger edge*/
    TIMER_EnableCapture(TIMER0, TIMER_CAPTURE_COUNTER_RESET_MODE,
TIMER_CAPTURE_RISING_EDGE);
    /*Enable Capture Trigger Interrupt*/
    TIMER_EnableCaptureInt(TIMER0);
    /*Timer0 start counting*/
    TIMER_Start(TIMER0);

    /* Enable Timer1 event counter input and external capture function */
    /*Open Timer1 with CONTINUOUS_MODE and Frequency*/
    TIMER_Open(TIMER1, TIMER_CONTINUOUS_MODE, 1);
    /*Set Timer1 Prescale = 0, clock will be divided by (prescale + 1)*/
    TIMER_SET_PRESCALE_VALUE(TIMER1, 0);
    /*Set Timer1 Compared Value*/
    TIMER_SET_CMP_VALUE(TIMER1, TIMER_CMP_MAX_VALUE);
    /*Enable Timer1 Capture and set output mode, trigger edge*/
```

```
    TIMER_EnableCapture(TIMER1, TIMER_CAPTURE_COUNTER_RESET_MODE,
TIMER_CAPTURE_RISING_EDGE);
    /*Enable Capture Trigger Interrupt*/
    TIMER_EnableCaptureInt(TIMER1);
    /*Timer1 start counting*/
    TIMER_Start(TIMER1);

    /* Enable Timer2 event counter input and external capture function */
    /*Open Timer2 with CONTINUOUS_MODE and Frequency*/
    TIMER_Open(TIMER2, TIMER_CONTINUOUS_MODE, 1);
    /*Set Timer2 Prescale = 0, clock will be divided by (prescale + 1)*/
    TIMER_SET_PRESCALE_VALUE(TIMER2, 0);
    /*Set Timer2 Compared Value*/
    TIMER_SET_CMP_VALUE(TIMER2, TIMER_CMP_MAX_VALUE);
    /*Enable Timer2 Capture and set output mode, trigger edge*/
    TIMER_EnableCapture(TIMER2, TIMER_CAPTURE_COUNTER_RESET_MODE,
TIMER_CAPTURE_RISING_EDGE);
    /*Enable Capture Trigger Interrupt*/
    TIMER_EnableCaptureInt(TIMER2);
    /*Timer2 start counting*/
    TIMER_Start(TIMER2);

    /* Enable Timer3 event counter input and external capture function */
    /*Open Timer3 with CONTINUOUS_MODE and Frequency*/
    TIMER_Open(TIMER3, TIMER_CONTINUOUS_MODE, 1);
    /*Set Timer3 Prescale = 0, clock will be divided by (prescale + 1)*/
    TIMER_SET_PRESCALE_VALUE(TIMER3, 0);
    /*Set Timer3 Compared Value*/
    TIMER_SET_CMP_VALUE(TIMER3, TIMER_CMP_MAX_VALUE);
    /*Enable Timer3 Capture and set output mode, trigger edge*/
    TIMER_EnableCapture(TIMER3, TIMER_CAPTURE_COUNTER_RESET_MODE,
TIMER_CAPTURE_RISING_EDGE);
    /*Enable Capture Trigger Interrupt*/
    TIMER_EnableCaptureInt(TIMER3);
    /*Timer3 start counting*/
    TIMER_Start(TIMER3);

    M1_pulse_cnt = M2_pulse_cnt = M3_pulse_cnt = M4_pulse_cnt = M5_pulse_cnt =0;
    M1_int_flag=0;
    cnt=0;
    while(1)
    {
        /*motor 1 will trigger capture interrupt two times per cycle*/
        /*setting constant M1_cnt_per_cycle as 2, this constant can be modify for
different motor.*/
        if (M1_int_flag == M1_cnt_per_cycle)
        {
            /* clear motor 1 capture complete flag */
            M1_int_flag = 0;
            /*reset M1_pulse_cnt when program is in abnormal*/
            if (M1_pulse_cnt >= M1_cnt_per_cycle)
                M1_pulse_cnt = 0;
            /* calculate motor 1 rps */
            /* get capture data */
            M1_period = M1_pulse;
            /* calculate motor 1 speed */
            M1_rps = 48000000 / (M1_period);
            /* clear Motor 1 period */
            M1_pulse=0;
            printf("----------------------------------------\n\n");
            printf("\nmotor 1 operation speed is : %d turns per second.\n\n", M1_rps);

        }

        /*motor 2 will trigger capture interrupt two times per cycle*/
```

```
        /*setting constant M2_cnt_per_cycle as 2, this constant can be modify for
different motor.*/
        if (M2_int_flag == M2_cnt_per_cycle)
        {
            /* clear motor 2 capture complete flag */
            M2_int_flag = 0;
            /*reset M2_pulse_cnt when program is in abnormal*/
            if (M2_pulse_cnt >= M2_cnt_per_cycle)
                M2_pulse_cnt = 0;
            /* calculate motor 2 rps */
            /* get capture data */
            M2_period = M2_pulse;
            /* calculate motor 2 speed */
            M2_rps = 48000000 / (M2_period);
            /* clear Motor 2 period */
            M2_pulse=0;
            printf("\nmotor 2 operation speed is : %d turns per second.\n\n", M2_rps);

        }

        /*motor 3 will trigger capture interrupt two times per cycle*/
        /*setting constant M3_cnt_per_cycle as 2, this constant can be modify for
different motor.*/
        if (M3_int_flag == M3_cnt_per_cycle)
        {
            /* clear motor 3 capture complete flag */
            M3_int_flag = 0;
            /*reset M3_pulse_cnt when program is in abnormal*/
            if (M3_pulse_cnt >= M3_cnt_per_cycle)
                M3_pulse_cnt = 0;
            /* calculate motor 3 rps */
            /* get capture data */
            M3_period = M3_pulse;
            /* calculate motor 3 speed */
            M3_rps = 48000000 / (M3_period);
            /* clear Motor 3 period */
            M3_pulse=0;
            printf("\nmotor 3 operation speed is : %d turns per second.\n\n", M3_rps);

        }

        /*motor 4 will trigger capture interrupt two times per cycle*/
        /*setting constant M4_cnt_per_cycle as 2, this constant can be modify for
different motor.*/
        if (M4_int_flag == M4_cnt_per_cycle)
        {
            /* clear motor 4 capture complete flag */
            M4_int_flag = 0;
            /*reset M4_pulse_cnt when program is in abnormal*/
            if (M4_pulse_cnt >= M4_cnt_per_cycle)
                M4_pulse_cnt = 0;
            /* calculate motor 4 rps */
            /* get capture data */
            M4_period = M4_pulse;
            /* calculate motor 4 speed */
            M4_rps = 48000000 / (M4_period);
            /* clear Motor 4 period */
            M4_pulse=0;
            printf("\nmotor 4 operation speed is : %d turns per second.\n\n", M4_rps);

        }

        /*motor 5 will trigger capture interrupt two times per cycle*/
        /*setting constant M5_cnt_per_cycle as 2, this constant can be modify for
different motor.*/
        if (M5_int_flag == M5_cnt_per_cycle)
```

```
        {
            /* clear motor 5 capture complete flag */
            M5_int_flag = 0;
            /*reset M5_pulse_cnt when program is in abnormal*/
            if (M5_pulse_cnt >= M5_cnt_per_cycle)
                M5_pulse_cnt = 0;
            /* calculate motor 5 rps */
            /* get capture data */
            M5_period = M5_pulse;
            /* PWM1 prescare is 59F(1439) and clock counter is 48000000/1439 =33356 per
second*/
            /* frequency = 1/period so 33356/M5_pulse is motor rps. */

            M5_rps = 33356 / M5_pulse;
            /* clear Motor 5 period*/
            M5_pulse=0;
            printf("\nmotor 5 operation speed is : %d turns per second.\n\n", M5_rps);

        }

        /* delay 500ms to slow down motor speed reporting times every second*/

        CLK_SysTickDelay(250000);
        CLK_SysTickDelay(250000);

        /*reset M1_int_flag when program is in abnormal*/
        if(M1_int_flag > M1_cnt_per_cycle)
        {
            M1_int_flag = 0;
        }
        /*reset M2_int_flag when program is in abnormal*/
        if(M2_int_flag > M2_cnt_per_cycle)
        {
            M2_int_flag = 0;
        }
        /*reset M3_int_flag when program is in abnormal*/
        if(M3_int_flag > M3_cnt_per_cycle)
        {
            M3_int_flag = 0;
        }
        /*reset M4_int_flag when program is in abnormal*/
        if(M4_int_flag > M4_cnt_per_cycle)
        {
            M4_int_flag = 0;
        }
        /*reset M5_int_flag when program is in abnormal*/
        if(M5_int_flag > M5_cnt_per_cycle)
        {
            M5_int_flag = 0;
        }
    }
}
```

2.1 Software Requirements

- BSP version
  - M031 Series BSP CMSIS v3.03.000
- IDE version
  - Keil uVersion 5.26.2

2.2 Hardware Requirements

- Circuit components
  - NuMaker-M032KI Ver1.0
  - PFM1412DEB7V
- Pin connect

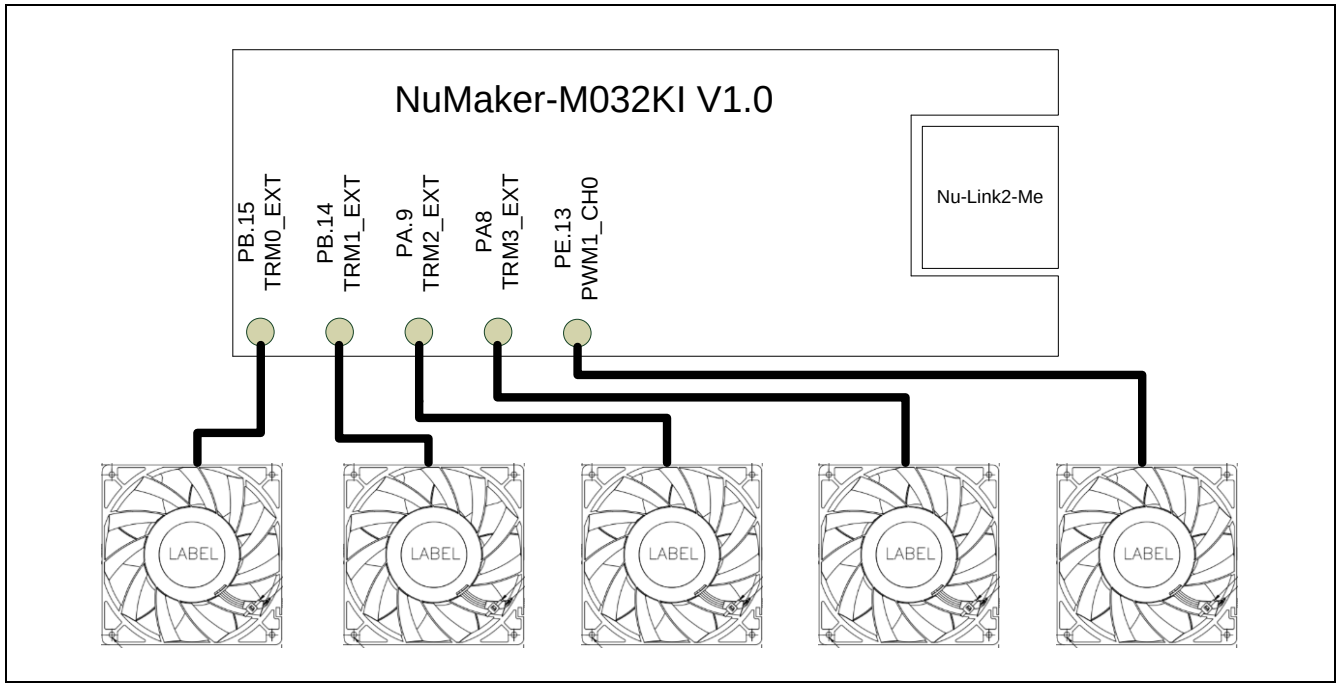

Figure 3-1 Pin Connect

3 Directory Information

The directory structure is shown below.

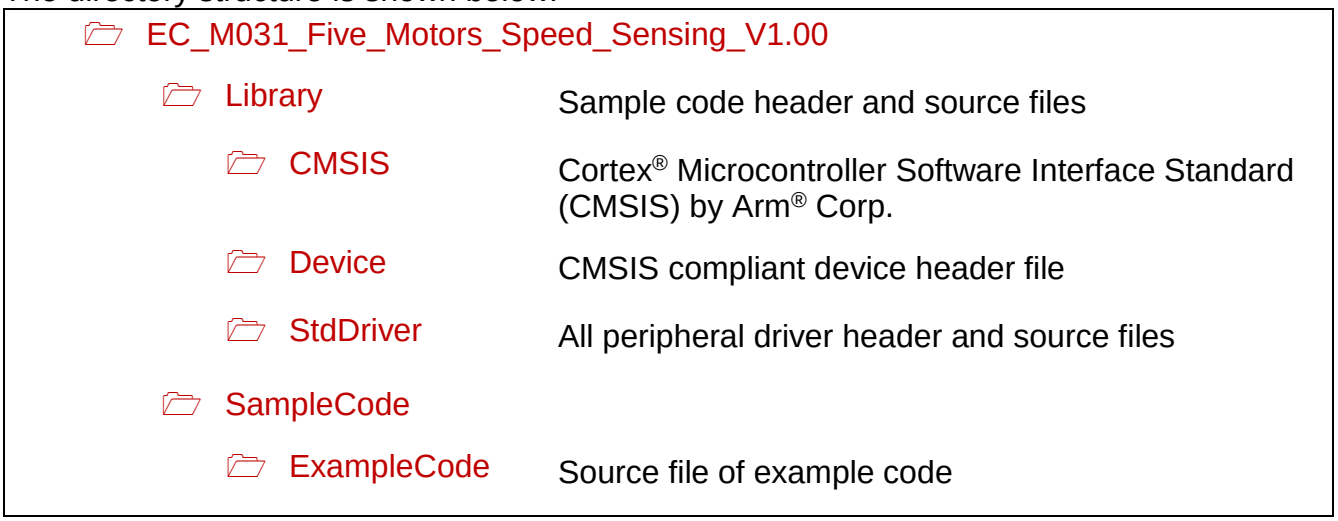

Figure 4-1 Directory Structure

4 Example Code Execution

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

5 Revision History

Date	Revision	Description
2021.12.16	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.