M480_M4_DSP_PID

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	The CMSIS-DSP library is a suite of common signal processing and mathematical functions that have been optimized for the NuMicro® M4 microcontroller. This document describes how to use the PID controller functions from the DSP library to help users get DSP high performance advantages.
BSP Version	M480 Series BSP CMSIS V3.03.000
Hardware	NuMaker-ETM-M487 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 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 sample demonstrates how to use the PID controller functions from the DSP library. There are two methods to implement PID controller in the example. One is using DSP library, and the other is using C functions. Comparing with C functions, DSP library has the faster execution speed.

1.2 Principle

The PID (proportional-integral-derivative) controller continuously calculates an error value as the difference between the measured output and the reference input. The controller attempts to minimize the error by adjusting the process control inputs. The proportional value determines the reaction to the current error, the integral value determines the reaction based on the sum of recent errors, and the derivative value determines the reaction based on the rate at which the error has been changing. The Figure 1 shows block diagram of a PID controller in closed-loop. The formula is defined as:

$y[n] = y[n-1] + A0 * x[n] + A1 * x[n-1] + A2 * x[n-2]$

$A0 = K_p + K_i + K_d$

$A1 = (-K_p) - (2 * K_d)$

$A2 = K_d$

where Kp is proportional constant, Ki is Integral constant and Kd is Derivative constant

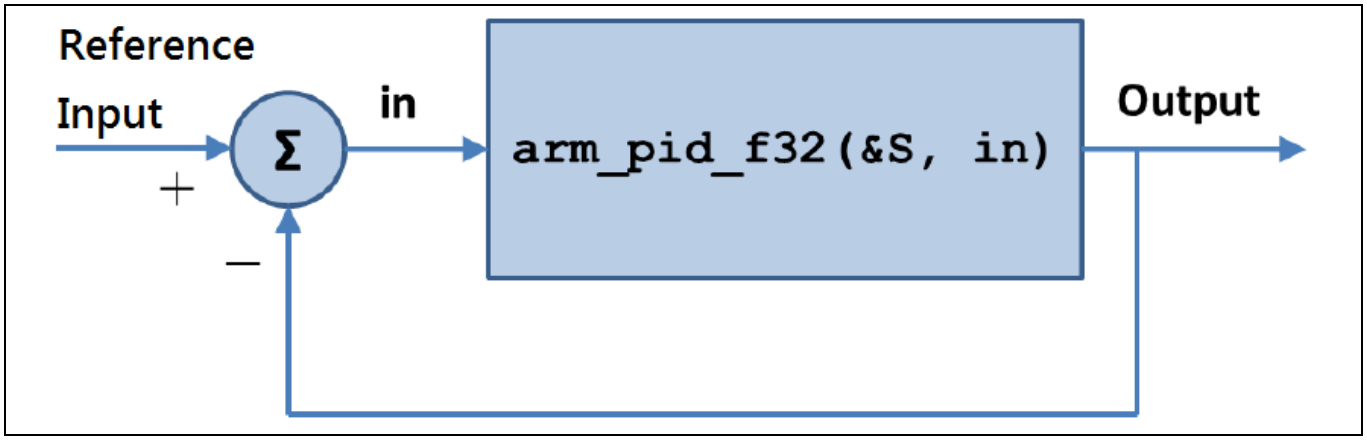


Figure 1 PID Controller Block Diagram

arm_pid_init_f32(arm_pid_instance_f32 *S, int32_t resetStateFlag)		
Parameters	*S	[in,out] points to an instance of the PID structure
	resetStateFlag	[in] value = 0: no change in state value = 1: reset state
Returns		none

Table 1 arm_pid_init_f32

arm_pid_f32(arm_pid_instance_f32 *S, float32_t in)		
Parameters	*S	[in,out] an instance of the floating-point PID Control structure
	in	[in] input sample to process
Returns		processed output sample

Table 2 arm_pid_f32

1.3 Demo Result

After the program is compile completed, user can download it to the NuMaker-ETM-M487 development board for verification. User can see the speed difference between the two implementations from the printed results, as shown in Figure 2 and Figure 3.
In this example, the target value is 500 with the PID controller, and the Kp, Ki, and Kd values are 0.4, 0.4, and 0, respectively. Finally, the result will be 500 after 100 times calculations, as shown in Figure 4.

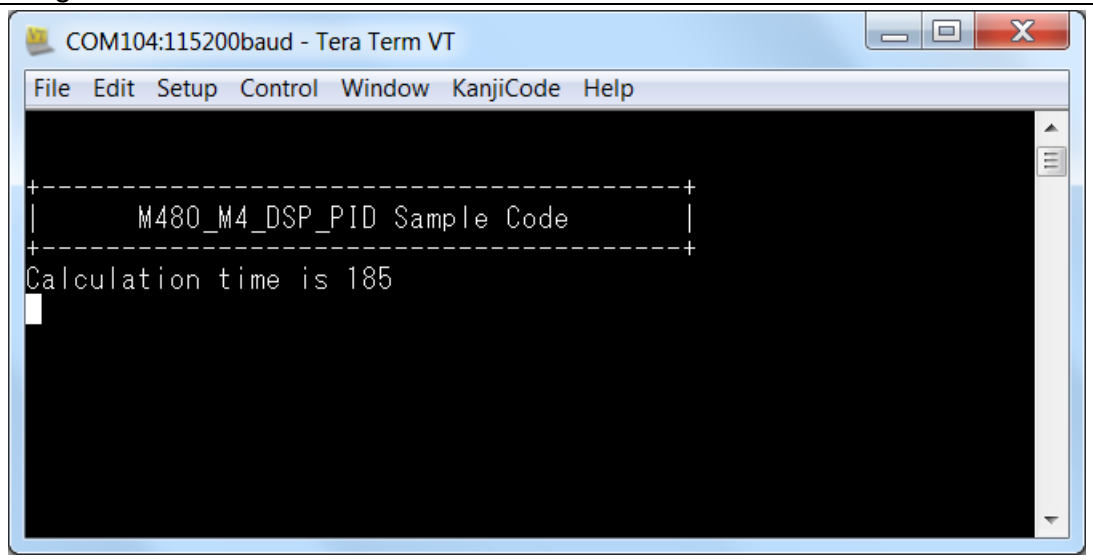


Figure 2 Use M4 DSP Math

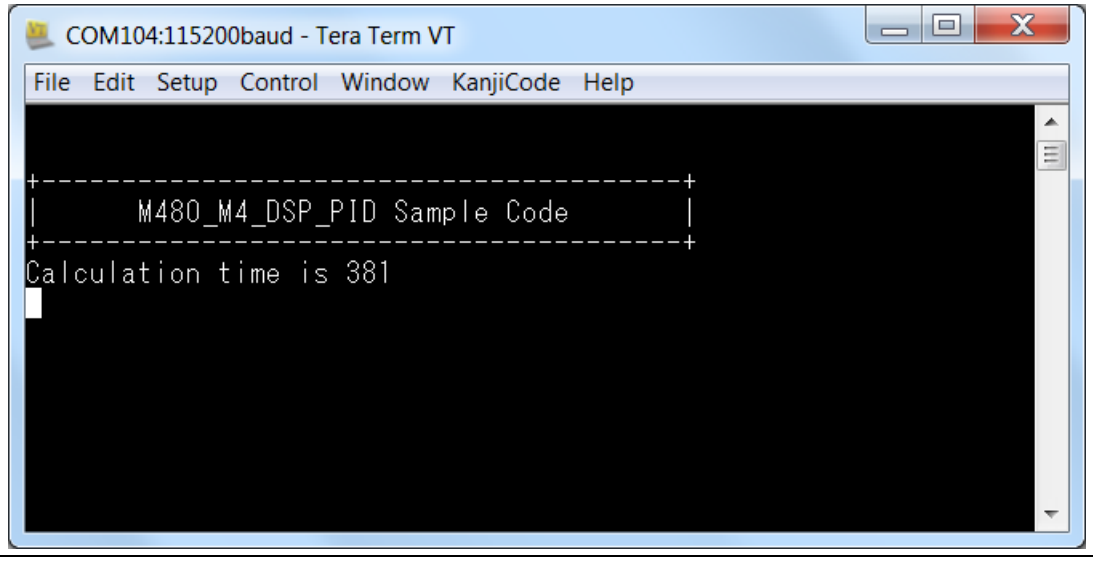


Figure 3 Not use M4 DSP Math

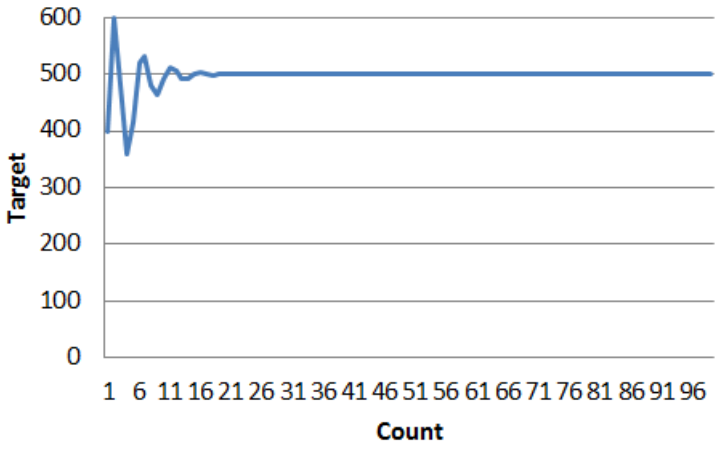


Figure 4 PID controller calculation result

2 Code Description

The example is using Keil uVersion 5.26. The following steps will introduce how to enable DSP with the Keil environment.

Step 1. After starting Keil, select Target Options, go to the C/C++ tab page, and input “ARM_MATH_CM4=1” and “__FPU_PRESENT=1UL” in the Define field of Preprocessor Symbols, as shown in Figure 5.

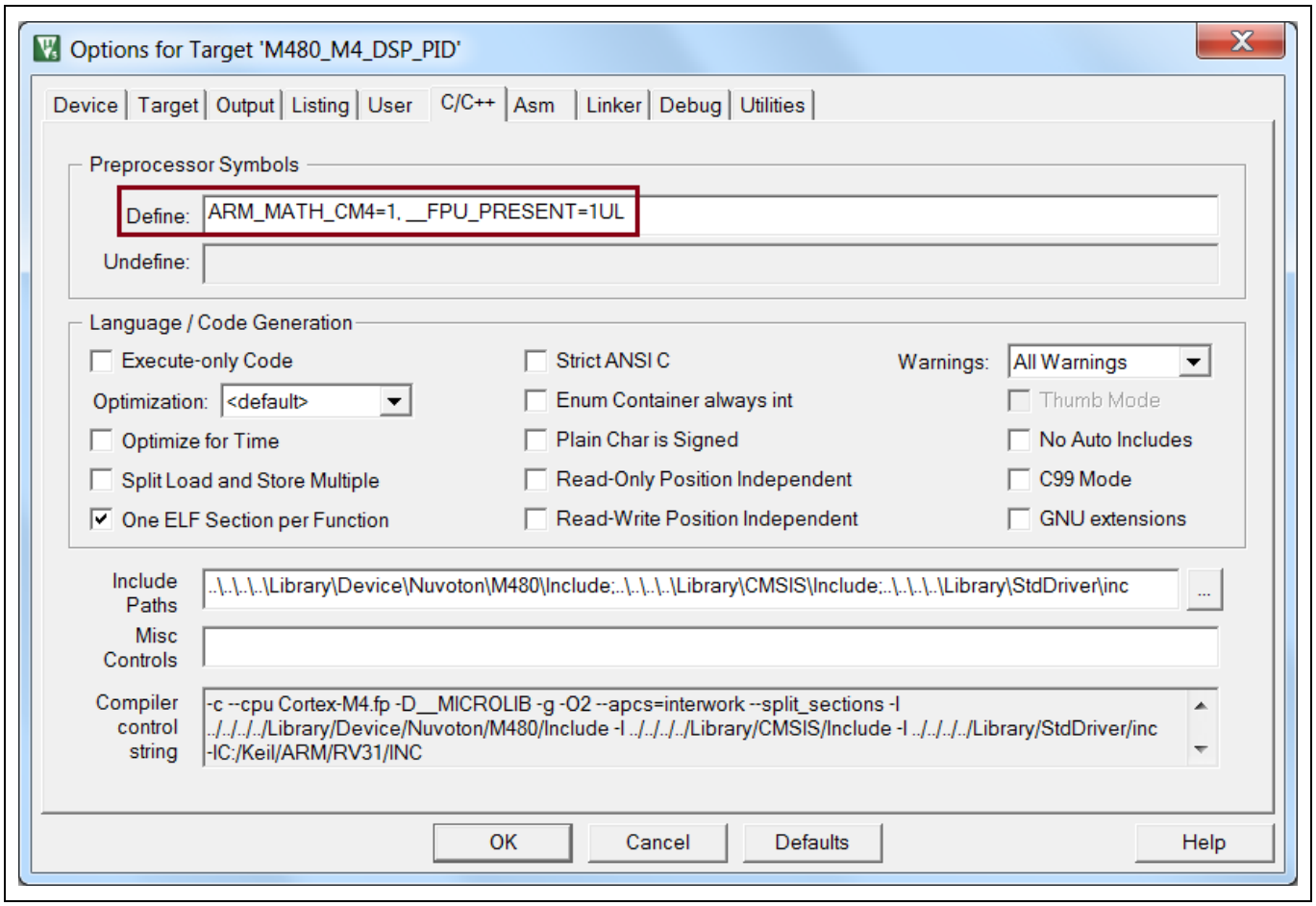


Figure 5 Add data to Define item

Step 2: Add “arm_cortexM4lf_math.lib” to the Library (\Library\CMSIS\Lib\ARM), as shown in Figure 6.

Step 3: Finally, include the arm_math.h file to the main.c(as shown in Figure 7). After above steps are ready, user can call the DSP library in the main.c.

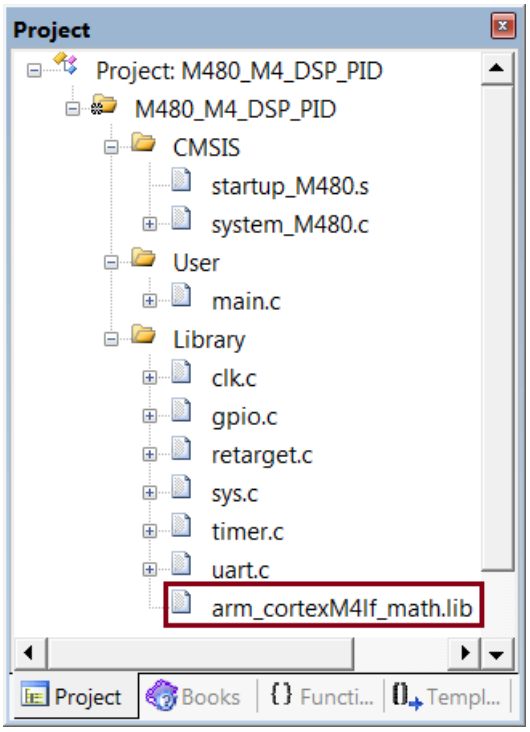


Figure 6 Add arm_cortexM4lf_math.lib

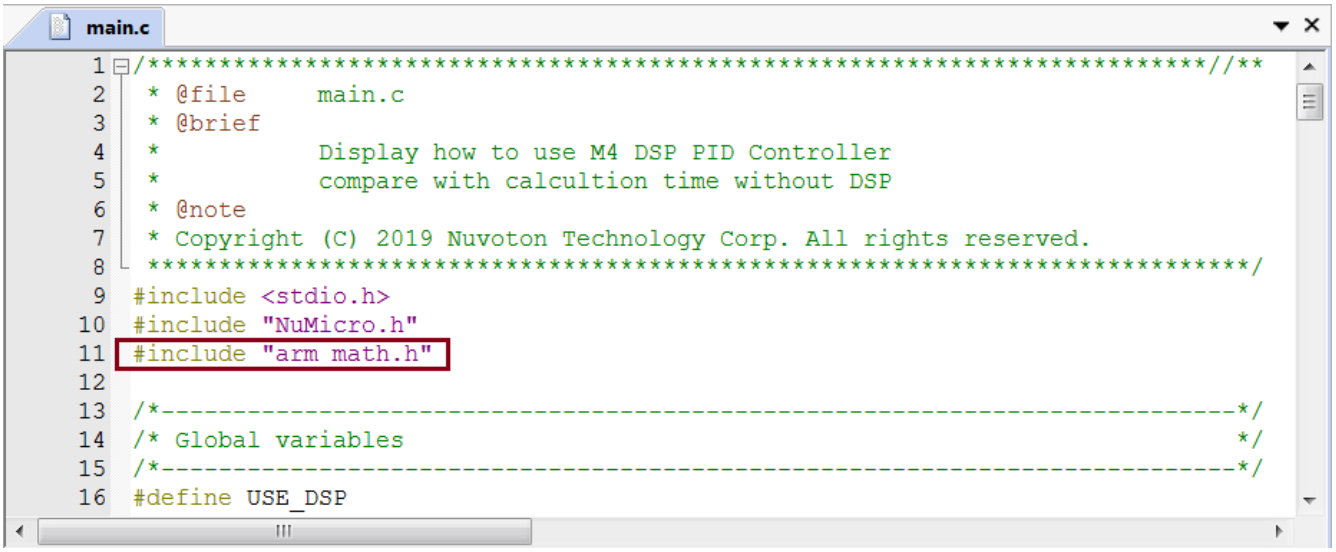


Figure 7 Include arm_math.h

In this example, users can use #define USE_DSP to decide whether to use the DSP library function to perform PID controller or use C functions. The timer is used to calculate the execution time. The data type of the variable is 32-bit floating-point values (f32).

```
/* ----------------------------------------------------------------
* Global variables
* ---------------------------------------------------------------- */
#define USE_DSP
```

```
uint32_t u32CalTime;
arm_pid_instance_f32 PIDS;
float32_t doutput[100], derr;
float fA0, fA1, fA2, afstate[3], fKp=0.4, fKi=0.4, fKd=0, ftarget, fival;

int main()
{
    …
    TIMER_Open(TIMER0, TIMER_CONTINUOUS_MODE, 1);
    TIMER_Start(TIMER0);

#ifdef USE_DSP
    /* PID by DSP library function */
    …
#else
    /* PID by C functions */
    …
#endif

    TIMER_Close(TIMER0);
    u32CalTime =TIMER_GetCounter(TIMER0);

    /* Display function calculation time */
    printf("Calculation time is %d \n", u32CalTime);

    while(1);
}
```

The following code lists PID controller function under DSP library.

```
    /* Initial DSP PID controller function*/
    arm_pid_init_f32(&PIDS,0);
    /* Calculate PID controller function 100 times*/
    for(i=1; i<100; i++)
    {
        doutput[i] = arm_pid_f32(&PIDS,derr);
        /* Update error */
        derr = ftarget - doutput[i-1];
    }
```

The following code lists implement PID algorithm by C functions.

```
float PID(float fin)
{
```

```
    float fout;


    fA0 = fKp + fKi;
    fA1 = -fKp -(2 * fKd);
    fA2 = fKd;

    /* y[n] = y[n-1] + A0 * x[n] + A1 * x[n-1] + A2 * x[n-2]  */
    fout = (fA0 * fin) + (fA1 * afstate[0]) + (fA2 * afstate[1]) + (afstate[2]);

    /* Update state */
    afstate[1] = afstate[0];
    afstate[0] = fin;
    afstate[2] = fout;

    /* return to application */
    return (fout);
}
```

3 Software and Hardware Environment

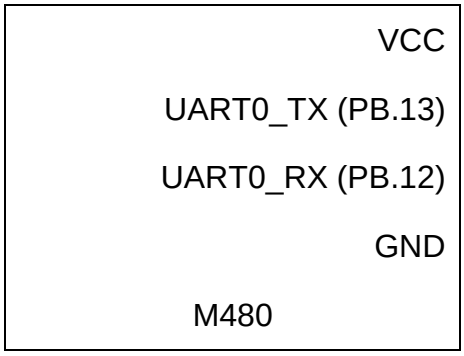
Software Environment

- BSP version
  - M480 Series BSP CMSIS V3.03.000
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuMaker-ETM-M487
- Diagram

Using printf function to output message and the UART0 TX RX pin are PB.13 and PB.12 respectively.



4 Directory Information

- EC_M480_M4_DSP_PID_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 — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click M480_M4_DSP_PID.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
Jun. 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.