M480 Maximum EADC Sampling Frequency

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses multiple EADC sampling modules for sampling one EADC channel to achieve the maximum sampling frequency.
BSP Version	M480_BSP_CMSIS_V3.05.004
Hardware	NuMaker-PFM-M487 V3.0

1. Overview

This example code uses multiple EADC sampling modules for sampling one EADC channel to achieve the maximum sampling frequency.

1.1 Principle

There are two kinds of analog input channels of M480 series which are fast and slow channels. CH10~15 is the fast channel and CH0~9 is the slow channel. The maximum sampling rate of fast channel is 5.14 MSPS and slow channel is 2.14 MSPS. Exceeding the limitation of sampling frequency will cause incorrect conversion results.

To achieve the maximum sampling frequency, this example code uses 8 EADC sampling modules for sampling one EADC fast channel, moving the sample data to the specified buffer by PDMA, and triggers the modules which have been executed for the next round when module 4 has been executed and subsequent modules are in the process.

For more information about EADC, please refer to the *M480 Series Technical Reference Manual*.

1.2 Demo Result

To measure the actual sampling frequency, this example code will set PB5 to low at the beginning of sampling, and set PB5 to high at the end of the sampling. The actual sampling frequency can be obtained by the oscilloscope to capture the low level time of the PB5.

This example code has been sampled 1024 times and took 199us as shown in Figure 1-1.

Sampling frequency = 1024/199us = 5.14 MSPS.

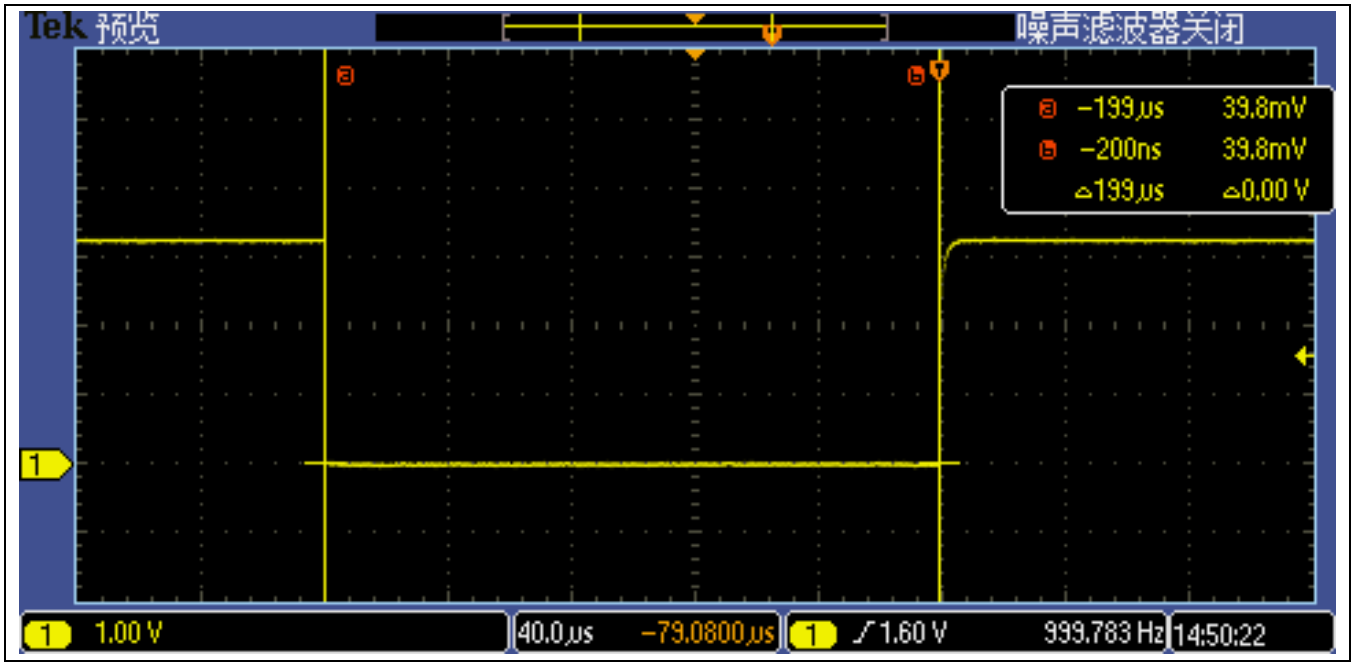

Figure 1-1 Sampling Time

To view the sampling data of EADC, you can load the array g_u16conversiondata into the Memory1 window in debug mode, as shown in Figure 1-2.

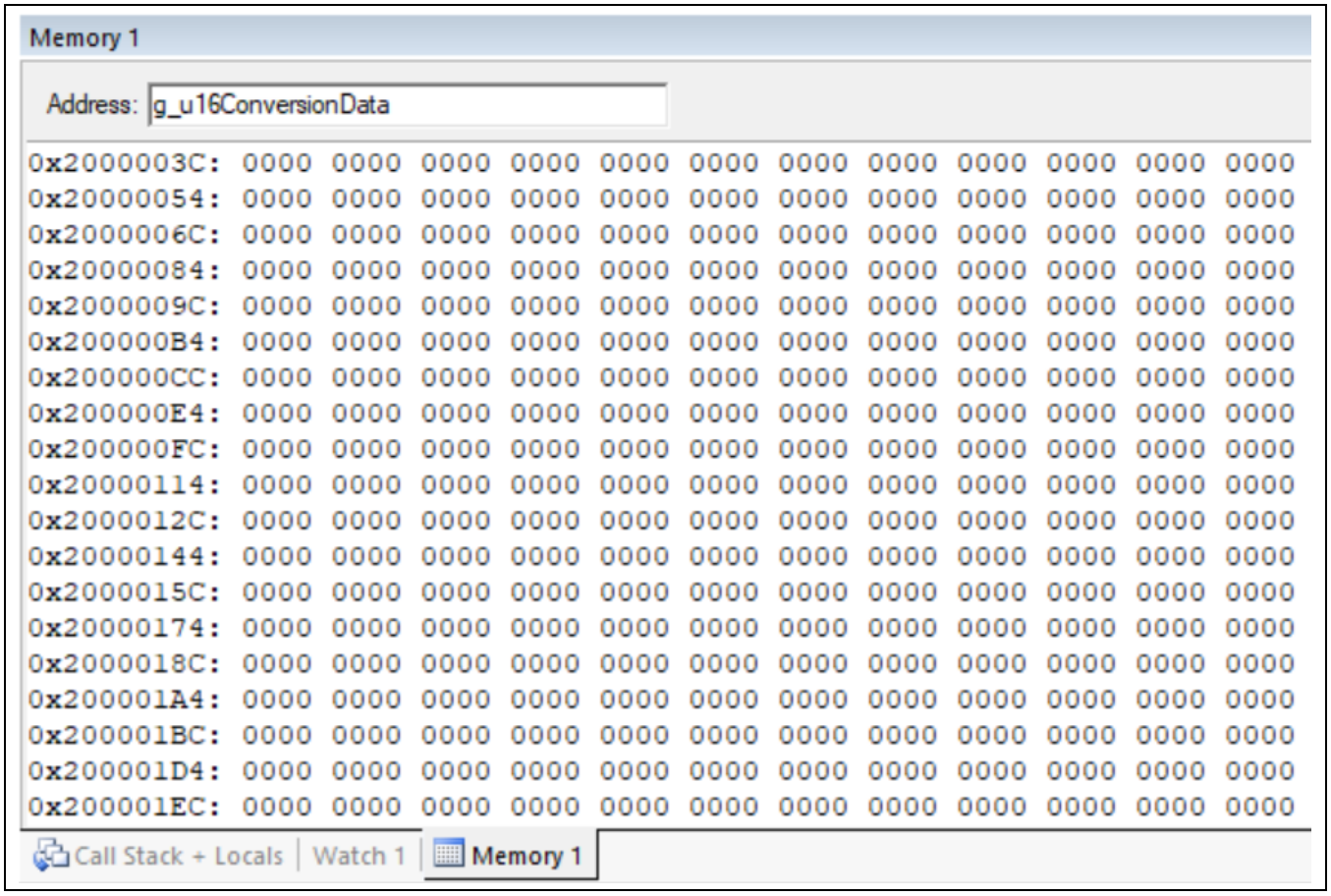

Figure 1-2 EADC Sampling Data

2. Code Description

Use module 0~7 for sampling EADC channel 10, and trigger the modules which have been executed for the next round when module 4 has been executed and subsequent modules are in the process.

```
void EADC_Init()
{
    /* Set input mode as single-end and enable the A/D converter */
    EADC_Open(EADC, EADC_CTL_DIFFEN_SINGLE_END);

    /* Configure the sample 0 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 0, EADC_ADINT0_TRIGGER, 10);
    /* Configure the sample 1 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 1, EADC_ADINT0_TRIGGER, 10);
    /* Configure the sample 2 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 2, EADC_ADINT0_TRIGGER, 10);
    /* Configure the sample 3 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 3, EADC_ADINT0_TRIGGER, 10);

    /* Configure the sample 4 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 4, EADC_ADINT0_TRIGGER, 10);
    /* Configure the sample 5 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 5, EADC_ADINT0_TRIGGER, 10);
    /* Configure the sample 6 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 6, EADC_ADINT0_TRIGGER, 10);
    /* Configure the sample 7 module for analog input channel 10 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 7, EADC_ADINT0_TRIGGER, 10);

    /* Clear the A/D ADINT0 interrupt flag for safe */
    EADC_CLR_INT_FLAG(EADC, EADC_STATUS2_ADIF0_Msk);

    /* Enable the sample module 4 interrupt */
    EADC_ENABLE_SAMPLE_MODULE_INT(EADC, 0, BIT4);
}
```

Move the sample data to g_u16ConversionData by PDMA.

```
void PDMA_Init()
{
    /* Configure PDMA peripheral mode form EADC to memory */
    /* Open Channel 2 */
    PDMA_Open(PDMA, BIT2);

    /* Transfer width is half word(16 bit) and transfer count is EADC_SAMPLE_COUNT */
    PDMA_SetTransferCnt(PDMA, 2, PDMA_WIDTH_16, EADC_SAMPLE_COUNT);

    /* Set source address as EADC data register(no increment) and destination address as
g_u16ConversionData array(increment) */
    PDMA_SetTransferAddr(PDMA, 2, (uint32_t)&EADC->CURDAT, PDMA_SAR_FIX,
(uint32_t)g_u16ConversionData, PDMA_DAR_INC);

    /* Select PDMA request source as ADC RX */
    PDMA_SetTransferMode(PDMA, 2, PDMA_EADC0_RX, FALSE, 0);

    /* Set PDMA as single request type for EADC */
    PDMA_SetBurstType(PDMA, 2, PDMA_REQ_SINGLE, PDMA_BURST_4);

    /* Enable EADC PDMA transfer */
    EADC_ENABLE_PDMA(EADC);

    /* Enable PDMA Interrupt */
    PDMA_EnableInt(PDMA, 2, PDMA_INT_TRANS_DONE);
    NVIC_EnableIRQ(PDMA_IRQn);
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480_BSP_CMSIS_V3.05.004
- IDE version
  - Keil uVersion 4.74

3.2 Hardware Requirements

- Circuit components
  - NuMaker-PFM-M487 V3.0
- Pin Connect

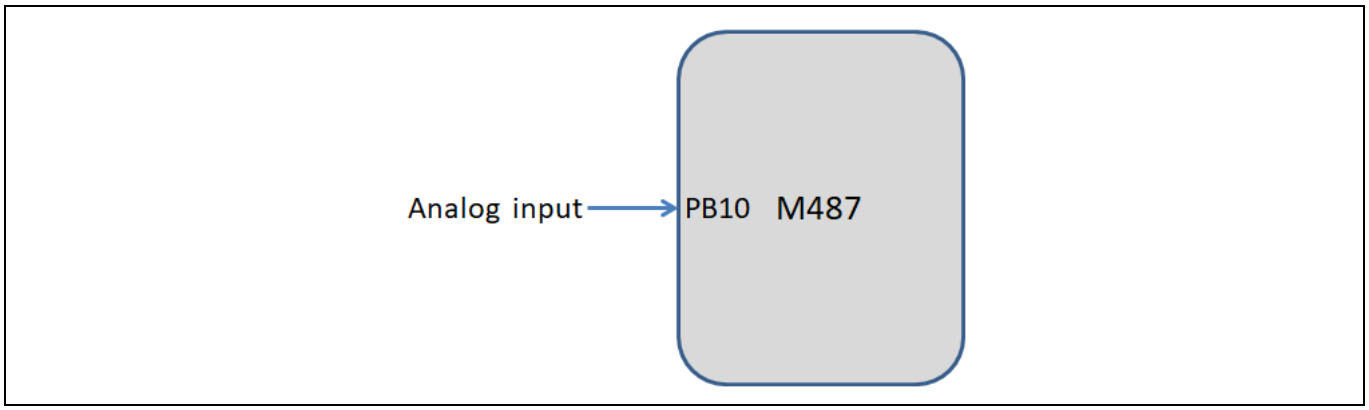

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

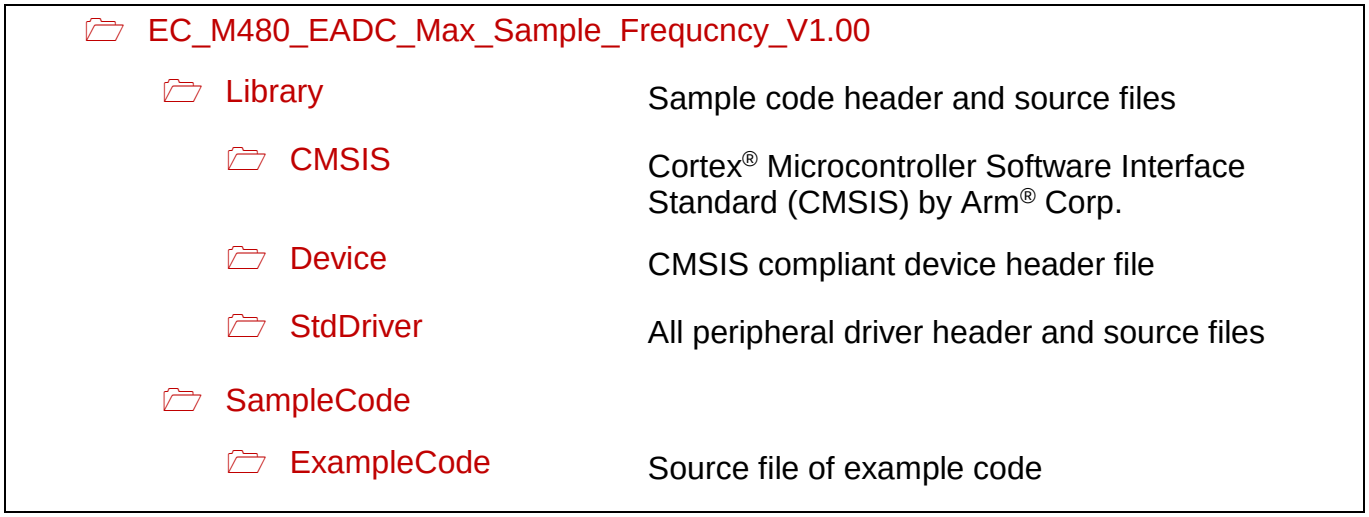

Figure 4-1 Directory Structure

5. Example Code Execution

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

6. Revision History

Date	Revision	Description
2023.06.05	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.