

## CM2053 Series CMSIS BSP Guide

Directory Introduction for 32-bit NuMicro® Family

Please extract the “CM2053\_Series\_BSP\_CMSIS\_V3.00.009.zip” file firstly and then put the “CM2053\_Series\_BSP\_CMSIS\_V1.00” folder into the working folder.

To experience the powerful features of CM2053 series in few minutes, please select the sample code to download and execute on the NuMaker board. Open the project files to build them with Keil® MDK, IAR, NuEclipse or VS Code, and then download and trace them on the NuMaker board to see how it works.

### Directory Information

<b>Document</b>	Driver reference guide and revision history.
<b>Library</b>	Driver header and source files.
<b>SampleCode</b>	Driver sample code.
<b>ThirdParty</b>	Libraries from third parties.

*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](http://www.nuvoton.com)

## TABLE OF CONTENTS

<b>1</b>	<b>DOCUMENT.....</b>	<b>4</b>
<b>2</b>	<b>LIBRARY .....</b>	<b>5</b>
<b>3</b>	<b>SAMPLECODE .....</b>	<b>6</b>
<b>4</b>	<b>THIRDPARTY.....</b>	<b>7</b>
<b>5</b>	<b>SAMPLECODE\FREERTOS .....</b>	<b>8</b>
<b>6</b>	<b>SAMPLECODE\ISP .....</b>	<b>9</b>
<b>7</b>	<b>SAMPLECODE\POWERMANAGEMENT.....</b>	<b>10</b>
<b>8</b>	<b>SAMPLECODE\STDDRIVER.....</b>	<b>11</b>
	System Manager (SYS) .....	11
	Clock Controller (CLK).....	11
	Flash Memory Controller (FMC) .....	11
	General Purpose I/O (GPIO).....	12
	PDMA Controller (PDMA) .....	12
	Timer Controller (TIMER).....	12
	Watchdog Timer (WDT) .....	13
	Window Watchdog Timer (WWDT) .....	13
	Real Timer Clock (RTC) .....	13
	Basic PWM Generator and Capture Timer (BPWM).....	14
	UART Interface Controller (UART).....	14
	Serial Peripheral Interface (SPI) .....	14
	I <sup>2</sup> C Serial Interface Controller (I <sup>2</sup> C) .....	16
	Universal Serial Control Interface Controller - UART Mode (USCI-UART) .....	16
	Universal Serial Control Interface Controller - SPI Mode (USCI-SPI) .....	17
	Universal Serial Control Interface Controller - I <sup>2</sup> C Mode (USCI-I2C) .....	18
	USB 2.0 Full-Speed Device Controller (USBBD) .....	19
	Controller Area Network with Flexible Data-Rate (CAN FD) .....	20

---

Enhanced 12-bit Analog-to-Digital Converter (EADC) .....	21
CRC Controller (CRC) .....	22
<b>9 SAMPLECODEXOM.....</b>	<b>23</b>

## 1 Document

<b>CMSIS.html</b>	Document of CMSIS version 6.1.0.
<b>NuMicro CM2053 Series CMSIS BSP Driver Reference Guide.chm</b>	This document describes the usage of drivers in CM2053 BSP.
<b>NuMicro CM2053 Series CMSIS BSP Revision History.pdf</b>	This document shows the revision history of CM2053 BSP.
<b>VS Code Quick Start Guide.pdf</b>	This document guides to install, configure and use VS Code.

## 2 Library

<b>CMSIS</b>	Cortex® Microcontroller Software Interface Standard (CMSIS) V6.1.0 definitions by Arm® Corp.
<b>Device</b>	CMSIS compliant device header file.
<b>StdDriver</b>	All peripheral driver header and source files.

### 3 SampleCode

FreeRTOS	Simple FreeRTOS™ demo code.
Hard_Fault_Sample	<p>Show hard fault information when hard fault happened.</p> <p>The hard fault handler shows some information including program counter, which is the address where the processor is executed when the hard fault occurs. The listing file (or map file) can show what function and instruction that is.</p> <p>It also shows the Link Register (LR), which contains the return address of the last function call. It can show the status where CPU comes from to get to this point.</p>
ISP	Sample codes for In-System-Programming.
PowerManagement	<p>Power management sample code.</p> <p>For more information about CM2053 series power management, please refer to the <a href="#">application note</a>.</p>
Semihost	Show how to print and get character through IDE console window.
StdDriver	Sample code to demonstrate the usage of CM2053 series MCU peripheral driver APIs.
Template	A project template for CM2053 series MCU.
XOM	<p>Demonstrate how to create XOM library and use it.</p> <p>For more information about CM2053 series XOM, please refer to the <a href="#">application note</a>.</p>

## 4 ThirdParty

### FreeRTOS

A real time operating system available for free download. Its official website is: <http://www.freertos.org/>.

## 5 SampleCode\FreeRTOS

---

### Blinky

This project provides two demo applications. A simple blinky style project, and a more comprehensive test and demo application.

---



## 6 SampleCode\ISP

<b>ISP_CAN</b>	In-System-Programming Sample code through CAN interface.
<b>ISP_DFU</b>	In-System-Programming Sample code through USB interface and following Device Firmware Upgrade Class Specification.
<b>ISP_HID</b>	In-System-Programming Sample code through USB HID interface.
<b>ISP_I2C</b>	In-System-Programming Sample code through I <sup>2</sup> C interface.
<b>ISP_MSC</b>	In-System-Programming Sample code through USB interface and following Mass Storage Class Specification.
<b>ISP_RS485</b>	In-System-Programming Sample code through RS485 interface.
<b>ISP_SPI</b>	In-System-Programming Sample code through SPI interface.
<b>ISP_UART</b>	In-System-Programming Sample code through UART interface.

## 7 SampleCode\PowerManagement

The CM2053 series MCU provides some power modes with different power consumption level and wake-up time. For more information, please refer to the [application note](#).

<b>SYS_PowerDownMode</b>	Show how to enter a different Power-down mode and wake up by RTC.
<b>SYS_PowerDown_MinCurrent</b>	Demonstrate how to minimize power consumption when entering Power-down mode.
<b>SYS_PowerMode</b>	Show how to set different core voltage and main voltage regulator type.

## 8 SampleCode\StdDriver

### System Manager (SYS)

<b>SYS_BODWakeup</b>	Demonstrate how to wake up system from Power-down mode by brown-out detector interrupt.
<b>SYS_TrimHIRC</b>	Demonstrate how to use LXT to trim HIRC.
<b>SYS_TrimMIRC</b>	Demonstrate how to use Timer to trim MIRC.

### Clock Controller (CLK)

<b>CLK_ClockDetector</b>	Demonstrate the usage of clock fail detector and clock frequency range detector function.
--------------------------	---

### Flash Memory Controller (FMC)

<b>FMC_CRC32</b>	Demonstrate how to use FMC CRC32 ISP command to calculate the CRC32 checksum of APROM and LDROM.
<b>FMC_ExecInSRAM</b>	Implement a code and execute it in SRAM to program embedded Flash.
<b>FMC_IAP</b>	Demonstrate FMC IAP boot mode and show how to use vector remap function. LDROM image has been embedded in APROM image and programmed to LDROM Flash at run-time. This sample also shows how to branch between APROM and LDROM.
<b>FMC_MultiBoot</b>	Implement a multi-boot system to boot from different applications in APROM or LDROM by VECMAP.
<b>FMC_MultiWordProgram</b>	Show how to use FMC multi-word program ISP command to program APROM 0x18000~0x20000 area.
<b>FMC_ReadAllOne</b>	Demonstrate how to use FMC Read-All-One ISP command to verify APROM or LDROM pages are all 0xFFFFFFFF or not.
<b>FMC_RW</b>	Show FMC read Flash IDs, erase, read, and write functions.

<b>FMC_XOM</b>	<p>Show how to configure and set up an XOM region then perform XOM function.</p> <p>For more information, please refer to the <a href="#">application note</a>.</p>
----------------	---

## General Purpose I/O (GPIO)

<b>GPIO_EINTAndDebounce</b>	Show the usage of GPIO external interrupt function and de-bounce function.
<b>GPIO_INT</b>	Show the usage of GPIO interrupt function.
<b>GPIO_OutputInput</b>	Show how to set GPIO pin mode and use pin data input and output control.
<b>GPIO_PowerDown</b>	Show how to wake up system from Power-down mode by GPIO interrupt.
<b>GPIO_SingleCycleIO</b>	Show GPIO single cycle I/O bus performance.

## PDMA Controller (PDMA)

<b>PDMA_BasicMode</b>	Use PDMA channel 2 to transfer data from memory to memory.
<b>PDMA_ScatterGather</b>	Use PDMA channel 4 to transfer data from memory to memory by scatter-gather mode.
<b>PDMA_ScatterGather_PingPongBuffer</b>	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).

## Timer Controller (TIMER)

<b>TIMER_CaptureCounter</b>	Show how to use the Timer2 capture function to capture Timer2 counter value.
<b>TIMER_Delay</b>	Demonstrate the usage of TIMER_Delay API to generate a 1 second delay.

<b>TIMER_EventCounter</b>	Use pin PB.4 to demonstrate Timer event counter function.
<b>TIMER_FreeCountingMode</b>	Use the Timer pin to demonstrate Timer free counting mode function, and display the measured input frequency to UART console.
<b>TIMER_InterTimerTriggerMode</b>	Use the Timer pin to demonstrate inter-timer trigger mode function, and display the measured input frequency to UART console.
<b>TIMER_Periodic</b>	Use the Timer periodic mode to generate Timer interrupt every 1 second.
<b>TIMER_PeriodicINT</b>	Implement Timer counting in periodic mode.
<b>TIMER_TimeoutWakeup</b>	Use Timer0 periodic time-out interrupt event to wake up system.
<b>TIMER_ToggleOut</b>	Demonstrate the Timer0 toggle out function on pin PB.5.

### Watchdog Timer (WDT)

<b>WDT_TimeoutWakeupAndReset</b>	Implement WDT time-out interrupt event to wake up system and generate time-out reset system event while WDT time-out reset delay period expired.
----------------------------------	--

### Window Watchdog Timer (WWDT)

<b>WWDT_CompareINT</b>	Show how to reload the WWDT counter value.
------------------------	--

### Real Timer Clock (RTC)

<b>RTC_Alarm_Test</b>	Demonstrate the RTC alarm function. It sets an alarm 10 seconds after execution.
<b>RTC_Alarm_Wakeup</b>	Use RTC alarm interrupt event to wake up system.
<b>RTC_Time_Display</b>	Demonstrate the RTC function and display current time to the UART console.

## Basic PWM Generator and Capture Timer (BPWM)

<b>BPWM_Capture</b>	Use BPWM0 channel 0 to capture the Timer0 waveform.
<b>BPWM_DoubleBuffer</b>	Change duty cycle and period of output waveform by BPWM double buffer function.
<b>BPWM_OutputWaveform</b>	Demonstrate how to use BPWM counter output waveform.
<b>BPWM_SwitchDuty</b>	Change duty cycle of output waveform by configured period.

## UART Interface Controller (UART)

<b>UART_AutoBaudRate</b>	Show how to use auto baud rate detection function.
<b>UART_AutoFlow</b>	Transmit and receive data using auto flow control.
<b>UART_IrDA</b>	Transmit and receive UART data in UART IrDA mode.
<b>UART_PDMA</b>	Demonstrate UART transmit and receive function with PDMA.
<b>UART_RS485</b>	Transmit and receive data in UART RS485 mode.
<b>UART_SingleWire</b>	Transmit and receive data in UART single-wire mode.
<b>UART_TxRxFunction</b>	Transmit and receive data from PC terminal through RS232 interface.
<b>UART_Wakeup</b>	Show how to wake up system from Power-down mode by UART interrupt.

## Serial Peripheral Interface (SPI)

<b>SPI_Flash</b>	Access SPI Flash through SPI interface.
<b>SPI_HalfDuplex</b>	Demonstrate SPI half-duplex mode. Configure USPIO as master mode and SPI0 as slave mode. Both USPIO and SPI0 are half-duplex mode.

<b>SPI_Loopback</b>	A SPI read/write demo connecting SPI0 MISO and MOSI pins.
<b>SPI_MasterFIFOmode</b>	Configure SPI0 as master mode and demonstrate how to communicate with an off-chip SPI slave device with FIFO mode. This sample code could work with <a href="#">SPI_SlaveFIFOmode</a> sample code.
<b>SPI_PDMA_LoopTest</b>	Demonstrate SPI data transfer with PDMA. SPI0 will be configured as slave mode and USPI0 will be configured as master mode. Both Tx PDMA function and Rx PDMA function will be enabled.
<b>SPI_SlaveFIFOmode</b>	Configure SPI0 as slave mode and demonstrate how to communicate with an off-chip SPI master device with FIFO mode. This sample code needs to work with <a href="#">SPI_MasterFIFOmode</a> sample code.
<b>SPI_SlaveFIFOmodeINT</b>	Configure SPI0 as slave mode and demonstrate how to use FIFO mode to communicate with an off-chip SPI master device, transmit and receive data in the interrupt handler. This sample code needs to work with <a href="#">SPI_MasterFIFOmode</a> sample code.
<b>SPII2S_Master</b>	Configure SPI0 as I <sup>2</sup> S master mode and demonstrate how I <sup>2</sup> S works in master mode. This sample code needs to work with <a href="#">SPII2S_Slave</a> sample code.
<b>SPII2S_PDMA_Codec</b>	An I <sup>2</sup> S demo with PDMA function connected with audio codec.
<b>SPII2S_PDMA_Play</b>	An I <sup>2</sup> S demo for playing data and demonstrating how I <sup>2</sup> S works with PDMA.
<b>SPII2S_PDMA_PlayRecord</b>	An I <sup>2</sup> S demo for playing and recording data with PDMA function.
<b>SPII2S_PDMA_Record</b>	An I <sup>2</sup> S demo for recording data and demonstrating how I <sup>2</sup> S works with PDMA.
<b>SPII2S_Slave</b>	Configure SPI0 as I <sup>2</sup> S slave mode and demonstrate how I <sup>2</sup> S works in slave mode. This sample code needs to work with <a href="#">SPII2S_Master</a> sample code.

## I<sup>2</sup>C Serial Interface Controller (I<sup>2</sup>C)

<b>I2C_EEPROM</b>	Read and write EEPROM via I <sup>2</sup> C interface.
<b>I2C_GCMode_Master</b>	Demonstrate how a master uses I <sup>2</sup> C address 0x0 to write data to I <sup>2</sup> C slave. This sample code needs to work with <a href="#">I2C_GCMode_Slave</a> sample code.
<b>I2C_GCMode_Slave</b>	Demonstrate how to receive master data in GC (General Call) mode. This sample code needs to work with <a href="#">I2C_GCMode_Master</a> sample code.
<b>I2C_Loopback</b>	Show how a I <sup>2</sup> C master accesses a slave via loopback of 2 I <sup>2</sup> C ports.
<b>I2C_Master</b>	Show how a I <sup>2</sup> C master accesses a slave. This sample code needs to work with <a href="#">I2C_Slave</a> sample code.
<b>I2C_MultiBytes_Master</b>	Demonstrate how to use multi-bytes API to access slave. This sample code needs to work with <a href="#">I2C_Slave</a> sample code.
<b>I2C_PDMA_TRX</b>	Demonstrate I <sup>2</sup> C PDMA mode, which need to connect I <sup>2</sup> C0 (master) and I <sup>2</sup> C1 (slave).
<b>I2C_SingleByte_Master</b>	Demonstrate how to use single byte API to access slave. This sample code needs to work with <a href="#">I2C_Slave</a> sample code.
<b>I2C_Slave</b>	Show how a I <sup>2</sup> C slave receives data from a master. This sample code needs to work with <a href="#">I2C_Master</a> sample code.
<b>I2C_Wakeup_Slave</b>	Demonstrate how to set I <sup>2</sup> C to wake up MCU from Power-down mode. This sample code could work with <a href="#">I2C_Master</a> sample code.

## Universal Serial Control Interface Controller - UART Mode (USCI-UART)

<b>USCI_UART_AutoBaudRate</b>	Show how to use auto baud rate detection function.
<b>USCI_UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. This sample code needs to work with



	<a href="#">USCI_UART_AutoFlow_Slave</a> sample code.
<b>USCI_UART_AutoFlow_Slave</b>	Transmit and receive data with auto flow control. This sample code needs to work with <a href="#">USCI_UART_AutoFlow_Master</a> sample code.
<b>USCI_UART_PDMA</b>	This is a USCI_UART PDMA demo and needs to connect USCI_UART Tx and Rx.
<b>USCI_UART_RS485_Master</b>	Transmit and receive data in RS485 mode. This sample code needs to work with <a href="#">USCI_UART_RS485_Slave</a> sample code.
<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 mode. This sample code needs to work with <a href="#">USCI_UART_RS485_Master</a> sample code.
<b>USCI_UART_TxRxFunction</b>	Transmit and receive data from PC terminal through RS232 interface.
<b>USCI_UART_Wakeup</b>	Show how to wake up system from Power-down mode by USCI interrupt in UART mode.

## Universal Serial Control Interface Controller - SPI Mode (USCI-SPI)

<b>USCI_SPI_Loopback</b>	Implement USCI_SPI0 master loop back transfer. This sample code needs to connect USCI_SPI0_MISO pin and USCI_SPI0_MOSI pin together. It will compare the received data with transmitted data.
<b>USCI_SPI_MasterMode</b>	Configure USCI_SPI0 as master mode and demonstrate how to communicate with an off-chip SPI Slave device. This sample code needs to work with <a href="#">USCI_SPI_SlaveMode</a> sample code.
<b>USCI_SPI_PDMA_LoopTest</b>	Demonstrate SPI data transfer with PDMA. USCI_SPI0 will be configured as master mode and USCI_SPI1 will be configured as slave mode. Both Tx PDMA function and Rx PDMA function will be enabled.
<b>USCI_SPI_SlaveMode</b>	Configure USCI_SPI0 as slave mode and demonstrate how to communicate with an off-chip SPI master device. This sample code needs to work with

	<a href="#">USCI_SPI_MasterMode</a> sample code.
<b>USCI_SPI_SlaveModeINT</b>	Configure USCI_SPI0 as slave mode and demonstrate how to communicate with an off-chip SPI master device, transmit and receive data in the interrupt handler. This sample code needs to work with <a href="#">USCI_SPI_MasterMode</a> sample code.

## Universal Serial Control Interface Controller - I<sup>2</sup>C Mode (USCI-I2C)

<b>USCI_I2C_EEPROM</b>	Show how to use USCI_I2C interface to access EEPROM.
<b>USCI_I2C_Master</b>	Show how an I <sup>2</sup> C master accesses 7-bit address slave. This sample code needs to work with <a href="#">USCI_I2C_Slave</a> sample code.
<b>USCI_I2C_Master_10bit</b>	Show how an I <sup>2</sup> C master accesses 10-bit address slave. This sample code needs to work with <a href="#">USCI_I2C_Slave_10bit</a> sample code.
<b>USCI_I2C_Monitor</b>	Use USCI_I2C to monitor and log I2C bus traffic.
<b>USCI_I2C_MultiBytes_Master</b>	Use UI2C multiple-byte functions to read and write data to slave. Need to work with the <a href="#">USCI_I2C_Slave</a> sample code.
<b>USCI_I2C_SingleByte_Master</b>	Use UI2C single-byte functions to read and write data to slave. Need to work with the <a href="#">USCI_I2C_Slave</a> sample code.
<b>USCI_I2C_Slave</b>	Show how an I <sup>2</sup> C 7-bit address slave receives data from master.
<b>USCI_I2C_Slave_10bit</b>	Show how an I <sup>2</sup> C 10-bit address slave receives data from master. This sample code needs to work with <a href="#">USCI_I2C_Master_10bit</a> sample code.
<b>USCI_I2C_Wakeup_Slave</b>	Demonstrate how to set I <sup>2</sup> C to wake up MCU from Power-down mode. This sample code needs to work with <a href="#">USCI_I2C_Master</a> sample code.

## USB 2.0 Full-Speed Device Controller (USBD)

<b>USBD_Audio_Codec</b>	Demonstrate how to implement a USB audio class device.
<b>USBD_HID_Keyboard</b>	Demonstrate how to implement a USB keyboard device. This sample code supports to use GPIO to simulate key input.
<b>USBD_HID_Mouse</b>	Simulate a USB mouse and draws circle on the screen.
<b>USBD_HID_MouseKeyboard</b>	Simulate an USB HID mouse and HID keyboard. Mouse draws circle on the screen and Keyboard uses GPIO to simulate key input.
<b>USBD_HID_RemoteWakeup</b>	Simulate a HID mouse supporting USB suspend and remote wakeup.
<b>USBD_HID_Transfer</b>	Demonstrate how to transfer data between a USB device and PC through a USB HID interface. A windows tool is also included in this sample code to connect with a USB device.
<b>USBD_HID_Transfer_And_Keyboard</b>	Demonstrate how to implement a composite device of HID transfer and keyboard. Transfer data between USB device and PC through USB HID interface. A windows tool is also included in this sample code to connect with a USB device.
<b>USBD_HID_Transfer_And_MSC</b>	Demonstrate how to implement a composite device of HID transfer and mass storage. Transfer data between USB device and PC through USB HID interface. A windows tool is also included in this sample code to connect with a USB device.
<b>USBD_HID_Transfer_CTRL</b>	Use USB host core driver and HID driver. It shows how to submit HID class request and how to read data from control pipe. A windows tool is also included in this sample code to connect with a USB device.
<b>USBD_Mass_Storage_CDROM</b>	Demonstrate the emulation of USB mass storage device, CD-ROM.

<b>USBD_Mass_Storage_Flash</b>	Use internal Flash as backend storage media to simulate a USB pen drive.
<b>USBD_Micro_Printer</b>	Demonstrate how to implement a USB micro printer device.
<b>USBD_Printer_And_HID_Transfer</b>	Demonstrate how to implement a composite device of USB micro printer and HID transfer. Transfer data between USB device and PC through USB HID interface. A windows tool is also included in this sample code to connect with a USB device.
<b>USBD_VCOM_And_HID_Keyboard</b>	Demonstrate how to implement a composite device of VCOM and HID keyboard.
<b>USBD_VCOM_And_HID_Transfer</b>	Demonstrate how to implement a composite device of VCOM and HID transfer. Transfer data between USB device and PC through USB HID interface. A windows tool is also included in this sample code to connect with a USB device.
<b>USBD_VCOM_And_Mass_Storage</b>	Demonstrate how to implement a composite device of VCOM and mass storage.
<b>USBD_VCOM_DualPort</b>	Demonstrate how to implement a USB dual virtual COM port device.
<b>USBD_VCOM_MultiPort</b>	Demonstrate how to implement a USB multi virtual COM ports device.
<b>USBD_VCOM_MultiPort_CMD</b>	Demonstrate how to implement a USB multi virtual COM ports device with a terminal echo port.
<b>USBD_VCOM_SerialEmulator</b>	Demonstrate how to implement a USB virtual COM port device.

## Controller Area Network with Flexible Data-Rate (CAN FD)

<b>CANFD_CAN_Loopback</b>	Use CAN mode function to do internal loopback test.
<b>CANFD_CAN_MonitorMode</b>	Use CAN Monitor mode to monitor the CAN bus communication test.

<b>CANFD_CAN_TxRx</b>	Transmit and receive CAN messages through CAN interface.
<b>CANFD_CAN_TxRxINT</b>	An example of interrupt control using CAN bus communication.
<b>CANFD_CANFD_Loopback</b>	Use CAN FD mode function to do internal loopback test.
<b>CANFD_CANFD_MonitorMode</b>	Use CAN FD Monitor mode to monitor the CAN bus communication test.
<b>CANFD_CANFD_TxRx</b>	Transmit and receive CAN FD messages through CAN interface.
<b>CANFD_CANFD_TxRxINT</b>	An example of interrupt control using CAN FD bus communication.

### Enhanced 12-bit Analog-to-Digital Converter (EADC)

<b>EADC_Accumulate</b>	Demonstrate how to get accumulated conversion result.
<b>EADC_ADINT_Trigger</b>	Use ADINT interrupt to trigger the EADC conversion.
<b>EADC_Average</b>	Demonstrate how to get average conversion result.
<b>EADC_AverageCMP</b>	Demonstrate how to compare average conversion result.
<b>EADC_BandGap</b>	Convert band-gap (Sample module 16) and print conversion result.
<b>EADC_BandGapCalculateAVDD</b>	Demonstrate how to calculate battery voltage (AVDD) by using band-gap.
<b>EADC_BPWM_Trigger</b>	Demonstrate how to trigger EADC by BPWM.
<b>EADC_PDMA_BPWM_Trigger</b>	Demonstrate how to trigger EADC by BPWM and transfer conversion data by PDMA.
<b>EADC_Pending_Priority</b>	Demonstrate how to trigger multiple sample modules and got conversion results in order of priority.
<b>EADC_ResultMonitor</b>	Monitor the conversion result of channel 2 by the digital compare function.

<b>EADC_SWTRG_Trigger</b>	Trigger EADC by writing EADC_SWTRG register.
<b>EADC_TempSensor</b>	Convert temperature sensor (Sample module 17) and print conversion result.
<b>EADC_Timer_Trigger</b>	Show how to trigger EADC by Timer.

## **CRC Controller (CRC)**

<b>CRC_CCITT</b>	Implement CRC in CRC-CCITT mode and get the CRC checksum result.
<b>CRC_CRC32</b>	Implement CRC in CRC-32 mode and get the CRC checksum result.
<b>CRC_CRC8</b>	Implement CRC in CRC-8 mode and get the CRC checksum result.

## 9 SampleCode\XOM

In the CM2053 series MCU, XOM (Execute-Only Memory) is a secure ROM region which forbids any data access. However, the code stored in the XOM region could still be executed by CPU since it is accessed by instruction fetch. For more information, please refer to the [application note](#).

XOMLib	Demonstrate how to create XOM library.
XOMLibDemo	Demonstrate how to use <a href="#">XOMLib</a> .

### **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.*