

## M2351 Series CMSIS BSP Guide

Directory Introduction for 32-bit NuMicro™ Family

### Directory Information

Please extract the “M2351Series\_BSP\_CMSIS\_V3.00.001.zip” file firstly, and then put the “M2351Series\_BSP\_CMSIS\_V3.00.001” folder into the working folder (e.g. .\Nuvoton\BSP Library).

To experience the powerful features of M2351 in few minutes, please refer to NuMaker-PFM-M2351 Board Quick Start Guide. You can select the sample code of your interest to download and execute on the M2351 board. For example, you can try the TrustZone® technology in M2351 that provides system-wide hardware isolation for trusted software with the sample code in the folder BSP\SampleCode\TrustZone. This folder includes the sample code of TrustZone® for collaborative secure software development, Hard Fault handling and a TrustZone® template. You can open the project files to build them with Keil® MDK, IAR or Eclipse, and then download and trace them on the M2351 board to see how the TrustZone® works.

This BSP folder contents:

<b>Document\</b>	Device driver reference manual and reversion history.
<b>Library\</b>	Device driver header and source files.
<b>SampleCode\</b>	Device driver sample code.
<b>ThirdParty\</b>	Libraries of 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 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)

## 1 .\Document\

<b>CMSIS.html</b>	<p>Introduction of CMSIS version 5.0. CMSIS components included CMSIS-CORE, CMSIS-Driver, CMSIS-DSP, etc.</p> <ul style="list-style-type: none"> <li>● CMSIS-CORE: API for the Cortex-M0 processor core and peripherals.</li> <li>● CMSIS-Driver: Defines generic peripheral driver interfaces for middleware making it reusable across supported devices.</li> <li>● CMSIS-DSP: DSP Library Collection with over 60 Functions for various data types: fix-point (fractional q7, q15, q31) and single precision floating-point (32-bit).</li> </ul>
<b>NuMicro M2351 Series CMSIS BSP Revision History.pdf</b>	<p>The revision history of M2351 Series BSP.</p>
<b>NuMicro M2351 Series Driver Reference Guide.chm</b>	<p>The usage of drivers in M2351 Series BSP.</p>

## 2 .\Library\

<b>CMSIS\</b>	Cortex <sup>®</sup> Microcontroller Software Interface Standard (CMSIS) V5.0 definitions by ARM <sup>®</sup> Corp.
<b>Device\</b>	CMSIS compliant device header file.
<b>NuMaker\</b>	Specific libraries for M2351 NuMaker board.
<b>SmartcardLib\</b>	Library for accessing a smartcard.
<b>StdDriver\</b>	All peripheral driver header and source files.
<b>UsbHostLib\</b>	USB host library source code.

### 3 .\Sample Code\

<b>AttackCountermeasures\</b>	Sample codes for non-invasive countermeasures.
<b>CardReader\</b>	USB CCID Smartcard Reader sample code
<b>Hard_Fault_Sample\</b>	<p>Show hard fault information when hard fault happened.</p> <p>The hard fault handler show some information included program counter, which is the address where the processor was executing when the hard fault occur. The listing file (or map file) can show what function and instruction that was.</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>
<b>MKROM\</b>	Demonstrate the usage of M2351 MKROM libraries, and show how to generate a secure boot image for Secure Boot Verification.
<b>PowerManagement\</b>	Power management sample code.
<b>Semihost\</b>	Show how to print and get character through IDE console window.
<b>StdDriver\</b>	Demonstrate the usage of M2351 series MCU peripheral driver APIs.
<b>TrustZone\</b>	Includes the demo of secure codes and non-secure codes.
<b>XOM\</b>	Demonstrate how to create XOM library and use it.

## 4 .\ThirdParty\

Alibaba_TEE_Air\	Security TEE Air of Alibaba.
Crypto\	An open source library for elliptic curve cryptosystem (ECC).
emWin\	Segger embedded graphic library – emWin.
FatFs\	An open source FAT/exFAT file system library.

## 5 .\SampleCode\AttackCountermeasures

<b>CLK_FrequencyDetection</b>	Show the usage of clock fail detector and clock frequency monitor function.
<b>EADC_TemperatureDetection</b>	Show how to measure temperature by EADC.
<b>EADC_VoltageDetection</b>	Show how to measure AVDD voltage by EADC.
<b>RTC_FrequencyDetection</b>	Show the usage of LXT clock frequency monitor function.
<b>RTC_TamperDetection</b>	Show the usage of RTC static and dynamic tamper function.
<b>SYS_ResetDetection</b>	Demonstrate the methods of detecting reset anomalies.

## 6 .\SampleCode\MKROM

<b>CRYPTOLibDemo</b>	Demonstrate the usage of M2351 MKROM Crypto libraries on AES, ECC, SHA, TDES and TRNG applications.
<b>NSCLibDemo</b>	Show the usage of MKROM Non-secure callable libraries on Secure project and Non-secure project.
<b>SecureBootDemo</b>	Demonstrate how to generate a secure boot image(NuBL2), which can pass the Secure Boot verification in Secure Bootloader(NuBL1).
<b>SecureISPDemo</b>	Demonstrate how to initial SecureISP mode.
<b>SecureOTADemo</b>	Demonstrate to update NuBL32 and NuBL33 firmware securely over the air (OTA) by NuBL2 with MKROM library.

## 7 .\SampleCode\NuMaker

<b>Audio</b>	Demonstrate how to play/record audio. NAU88L25 is used in this sample code to play and record the audio data.
<b>Blinky</b>	A simple LED toggle sample code. It could be used as the startup of M2351 NuMaker board.
<b>DebugUART</b>	A simple debug message print demo.
<b>Key</b>	A simple key demo for NuMaker board.
<b>emWin_GUIDemo</b>	GUI demo for embedded graphics library – emWin.
<b>emWin_SimpleDemo</b>	Simple demo for embedded graphics library – emWin.
<b>WiFi_bypass</b>	By pass ESP8266 to UART port.



## 8 .\SampleCode\PowerManagement

<b>SYS_DPDMode_Wakeup</b>	Show how to wake up system form DPD Power-down mode by Wake-up pin(PC.0) or Wake-up Timer or RTC Tick or RTC Alarm or RTC Tamper 0.
<b>SYS_PowerDownMode</b>	Show how to enter to different Power-down mode and wake-up by RTC.
<b>SYS_PowerMode</b>	Show how to set different core voltage and main voltage regulator type.
<b>SYS_SPDMode_Wakeup</b>	Show how to wake up system form SPD Power-down mode by Wake-up pin(PC.0) or Wake-up Timer or Wake-up ACMP or RTC Tick or RTC Alarm and RTC Tamper 0.
<b>SYS_SPDMode_WakeupAndReturn</b>	Show how to continue executing code after wake-up form SPD Power-down mode by SRAM data retention function.
<b>SYS_SRAMPowerMode</b>	Show how to select SRAM power mode in system Power-down mode.

## 9 .\SampleCode\StdDriver

<b>ACMP</b>	Demonstrate how Analog Comparator (ACMP) works with internal band-gap voltage.
<b>ACMP_Wakeup</b>	Show how to wake up MCU from Power-down mode by ACMP wake-up function.
<b>BPWM_Capture</b>	Use BPWM0 Channel 0 to capture the BPWM1 Channel 0 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.
<b>BPWM_SyncStart</b>	Demonstrate how to use BPWM counter synchronous start function.
<b>CAN_BasicMode_Rx</b>	Demonstrate CAN bus receive a message with basic mode.
<b>CAN_BasicMode_Tx</b>	Demonstrate CAN bus transmit a message with basic mode.
<b>CAN_BasicMode_Tx_Rx</b>	Demonstrate CAN bus transmit and receive a message with basic mode by connecting CAN0 and CAN1 to the same CAN bus.
<b>CAN_NormalMode_Rx</b>	Demonstrate CAN bus receive a message with normal mode.
<b>CAN_NormalMode_Tx</b>	Demonstrate CAN bus transmit a message with normal mode.
<b>CAN_NormalMode_Tx_Rx</b>	Demonstrate CAN bus transmit and receive a message with normal mode by connecting CAN 0 and CAN1 to the same CAN bus.
<b>CLK_ClockDetector</b>	Show the usage of clock fail detector and clock frequency monitor function.
<b>CRC_CCITT</b>	Implement CRC in CRC-CCITT mode and get the CRC checksum result.
<b>CRC_CRC8</b>	Implement CRC in CRC-8 mode and get the CRC checksum

	result.
<b>CRC_CRC32</b>	Implement CRC in CRC-32 mode with PDMA transfer.
<b>CRYPTO_AES</b>	Show Crypto IP AES-128 ECB mode encrypt/decrypt function.
<b>CRYPTO_ECC_Demo</b>	ECDSA signature and verification demo
<b>CRYPTO_ECC_ECDH</b>	ECDH demonstrate how to calculate share key by A private key and B private key.
<b>CRYPTO_ECC_GenerateSecretZ</b>	ECDH demo to establish a shared secret key
<b>CRYPTO_ECC_KeyGeneration</b>	Show Crypto IP ECC P-192 key generation function.
<b>CRYPTO_ECC_SignatureGeneration</b>	Show Crypto IP ECC P-192 ECDSA signature generation function.
<b>CRYPTO_ECC_SignatureVerification</b>	Show Crypto IP ECC P-192 ECDSA signature verification function.
<b>CRYPTO_HMAC</b>	Show Crypto IP HMAC function.
<b>CRYPTO_PRNG</b>	Generate random numbers using Crypto IP PRNG.
<b>CRYPTO_SHA</b>	Use Crypto IP SHA engine to run through known answer SHA1 test vectors.
<b>CRYPTO_SHA256_Flash</b>	Show how to calculate SHA-256 of the data in Flash.
<b>CRYPTO_TDES</b>	Show Crypto IP Triple DES CBC mode encrypt/decrypt function.
<b>CRYPTO_TRNG</b>	Demonstrate how to generate random number.
<b>DAC_ExtPinTrigger</b>	Demonstrate how to trigger DAC conversion by external pin.
<b>DAC_GroupMode</b>	Demonstrate DAC0 and DAC1 work in group mode
<b>DAC_PDMA_EPWMTrigger</b>	Demonstrate how to use PDMA and trigger DAC0 by EPWM.
<b>DAC_PDMA_TimerTrigger</b>	Demonstrate how to PDMA and trigger DAC by Timer.
<b>DAC_EPWMTrigger</b>	Demonstrate how to trigger DAC by EPWM.

<b>DAC_SoftwareTrigger</b>	Demonstrate how to trigger DAC conversion by software.
<b>DAC_TimerTrigger</b>	Demonstrate how to trigger DAC by timer.
<b>EADC_BandGap</b>	Convert Band-gap (Sample module 16) and print conversion result.
<b>EADC_ADINT_Trigger</b>	Use ADINT interrupt to do the EADC continuous scan conversion.
<b>EADC_PDMA_EPWMTrigger</b>	Demonstrate how to trigger EADC by EPWM and transfer conversion data by PDMA.
<b>EADC_EPWMTrigger</b>	Demonstrate how to trigger EADC by EPWM.
<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_TimerTrigger</b>	Show how to trigger EADC by timer.
<b>EADC_VBat</b>	Convert VBAT/4 (Sample module 18) and print conversion result.
<b>EBI_NOR</b>	Configure EBI interface to access MX29LV320T (NOR Flash) on EBI interface.
<b>EBI_SRAM</b>	Configure EBI interface to access BS616LV4017 (SRAM) on EBI interface.
<b>ECAP_GetInputFreq</b>	Show how to use ECAP interface to get input frequency
<b>ECAP_GetQEIFreq</b>	Show how to use ECAP interface to get QEI frequency.
<b>FMC_CRC32</b>	Demonstrate how to use FMC CRC32 ISP Command to calculate the CRC32 checksum of APROM and LDROM.
<b>FMC_Dual_Bank</b>	Demonstrate how dual processes work in dual bank flash architecture.

<b>FMC_DualBankFwUpdate</b>	Implement a firmware update mechanism based on dual bank flash architecture.
<b>FMC_ExecInSRAM</b>	Implement a code and execute in SRAM to program embedded Flash (support KEIL MDK only).
<b>FMC_IAP</b>	Show how to call LDROM function from APROM.
<b>FMC_MultiBoot</b>	Implement a multi-boot system to boot from different applications in APROM. A LDROM code and 4 APROM code are implemented in this sample code.
<b>FMC_OTP</b>	Demonstrate how to program, read and lock OTP.
<b>FMC_ReadAllOne</b>	Demonstrate how to use FMC Read-All-One ISP command to verify APROM/LDROM pages are all 0xFFFFFFFF or not.
<b>FMC_RW</b>	Demonstrate how to read/program embedded Flash by ISP function.
<b>FMC_SecureKey</b>	This sample code shows how to setup the secure key and how to perform secure key comparison.
<b>FMC_XOM</b>	This sample code shows how to configure and setup an XOM region the perform XOM function.
<b>GPIO_EINTAndDebounce</b>	Show the usage of GPIO external interrupt function and debounce 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/output control.
<b>GPIO_PowerDown</b>	Show how to wake up system from Power-down mode by GPIO interrupt.
<b>I2C_EEPROM</b>	Demonstrate how to access EEPROM through a I2C interface
<b>I2C_GCMode_Master</b>	Demonstrate how a Master uses I2C address 0x0 to write data to I2C Slave. This sample code needs to work with I2C_GCMode_Slave.
<b>I2C_GCMode_Slave</b>	Demonstrate how to receive Master data in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.

<b>I2C_Loopback</b>	Demonstrate how a Master accesses Slave.
<b>I2C_Master</b>	Demonstrate how a Master accesses Slave. This sample code needs to work with I2C_Slave.
<b>I2C_Master_PDMA</b>	Demonstrate how a Master accesses Slave using PDMA TX mode and PDMA RX mode.
<b>I2C_MultiBytes_Master</b>	Demonstrate how to use multi-bytes API to access slave. This sample code needs to work with I2C_Slave.
<b>I2C_SingleByte_Master</b>	Demonstrate how to use single byte API to access slave. This sample code needs to work with I2C_Slave.
<b>I2C_Slave</b>	Demonstrate how to set I2C in slave mode to receive the data from a Master. This sample code needs to work with I2C_Master.
<b>I2C_Slave_PDMA</b>	Demonstrate how a Slave uses PDMA Rx mode receive data from a Master.
<b>I2C_Wakeup_Slave</b>	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.
<b>I2S_Codec</b>	This is an I2S demo using NAU8822/88L25 audio codec, and used to play back the input from line-in.
<b>I2S_Codec_PDMA</b>	This is an I2S demo with PDMA function connected with codec.
<b>I2S_WAVPLAYER</b>	This is a WAV file player which plays back WAV file stored in SD memory card.
<b>OTG_Dual_Role_UMAS</b>	Demonstrate how USB works as a dual role device. If it works as USB Host, it can access a mass storage device. If it works as USB Device, it acts as a mass storage device.
<b>PDMA_BasicMode</b>	Use PDMA0 Channel 2 to transfer data from memory to memory.
<b>PDMA_ScatterGather</b>	Use PDMA0 channel 4 to transfer data from memory to memory by scatter-gather mode.
<b>PDMA_ScatterGather_PingPongBuffer</b>	Use PDMA0 to implement Ping-Pong buffer by scatter-gather mode(memory to memory).

<b>EPWM_AccumulatorINT_Tri ggerPDMA</b>	Demonstrate EPWM accumulator interrupt trigger PDMA.
<b>EPWM_Brake</b>	Demonstrate how to use EPWM brake function.
<b>EPWM_Capture</b>	Capture the EPWM1 Channel 0 waveform by EPWM1 Channel 2.
<b>EPWM_DeadZone</b>	Demonstrate how to use EPWM Dead Zone function.
<b>EPWM_DoubleBuffer</b>	Change duty cycle and period of output waveform by EPWM Double Buffer function (Period loading mode).
<b>EPWM_OutputWaveform</b>	Demonstrate how to use EPWM output waveform.
<b>EPWM_PDMA_Capture</b>	Capture the EPWM1 Channel 0 waveform by EPWM1 Channel 2, and use PDMA to transfer captured data.
<b>EPWM_SwitchDuty</b>	Change duty cycle of output waveform by configured period.
<b>EPWM_SyncStart</b>	Demonstrate how to use EPWM counter synchronous start function.
<b>QEI_CompareMatch</b>	Show the usage of QEI compare function.
<b>QSPI_DualMode_Flash</b>	Access SPI flash using QSPI dual mode.
<b>QSPI_QuadMode_Flash</b>	Access SPI flash using QSPI quad mode.
<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_Dynamic_Tamper</b>	Demonstrate the RTC dynamic tamper function.
<b>RTC_Spare_Access</b>	Demonstrate the RTC spare register read/write function and displays test result to the UART console.
<b>RTC_Static_Tamper</b>	Demonstrate the RTC static tamper function.
<b>RTC_Time_Display</b>	Demonstrate the RTC function and displays current time to the UART console.
<b>SC_EmulateCard</b>	Emulate a smartcard and send ATR to card reader.
<b>SC_ReadATR</b>	Read the smartcard ATR from smartcard 0 interface.



<b>SC_ReadSIM_PhoneBook</b>	Demonstrate how to read phone book information in the SIM card.
<b>SC_Timer</b>	Demonstrate how to use SC embedded timer.
<b>SCUART_TxRx</b>	Demonstrate smartcard UART mode by connecting PB.4 and PB.5 pins.
<b>SDH_FATFS</b>	Access a SD card formatted in FAT file system
<b>SPI_Flash</b>	Access SPI flash through SPI interface.
<b>SPI_Loopback</b>	Implement SPI Master loop back transfer. This sample code needs to connect MISO pin and MOSI pin together. It will compare the received data with transmitted data.
<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 needs to work with SPI_SlaveFIFOmode sample code.
<b>SPI_PDMA_LoopTest</b>	Demonstrate SPI data transfer with PDMA. QSPI0 will be configured as Master mode and SPI1 will be configured as Slave 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 SPI_MasterFIFOmode sample code.
<b>SPII2S_Master</b>	Configure SPI0 as I2S Master mode and demonstrate how I2S works in Master mode. This sample code needs to work with SPII2S_Slave.
<b>SPII2S_PDMA_Codec</b>	This is an I2S demo with PDMA function connected with audio codec.
<b>SPII2S_PDMA_Play</b>	This is an I2S demo for playing data and demonstrate how I2S works with PDMA.
<b>SPII2S_PDMA_PlayRecord</b>	This is an I2S demo for playing and recording data with PDMA function.
<b>SPII2S_PDMA_Record</b>	This is an I2S demo for recording data and demonstrate how I2S works with PDMA.



<b>SPII2S_Slave</b>	Configure SPI0 as I2S Slave mode and demonstrate how I2S works in Slave mode. This sample code needs to work with SPII2S_Master.
<b>SYS_BODWakeup</b>	Show how to wake up system form Power-down mode by brown-out detector interrupt.
<b>SYS_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLK0 pin.
<b>SYS_TrimIRC</b>	Demonstrate how to use LXT to trim HIRC.
<b>TIMER_ACMPTrigger</b>	Use ACMP to trigger Timer0 counter reset mode.
<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>	Demonstrates the timer event counter function.
<b>TIMER_FreeCountingMode</b>	Use the timer0 pin PA.11 to demonstrate timer free counting mode function. And displays the measured input frequency to UART console.
<b>TIMER_InterTimerTriggerMode</b>	Use the timer pin PB.5 to demonstrate inter timer trigger mode function. Also 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_PWM_Brake</b>	Demonstrate how to use Timer0 PWM brake function..
<b>TIMER_PWM_ChangeDuty</b>	Change duty cycle and period of output waveform in PWM down count type.
<b>TIMER_PWM_DeadTime</b>	Demonstrate Timer PWM Complementary mode and Dead-Time function.
<b>TIMER_PWM_OuputWaveform</b>	Demonstrate output different duty waveform in Timer0~Timer3 PWM.
<b>TIMER_TimeoutWakeup</b>	Use timer to wake up system from Power-down mode

	periodically
<b>TIMER_ToggleOut</b>	Demonstrate the timer0 toggle out function on pin PB.5.
<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 data in UART IrDA mode.
<b>UART_LIN</b>	Transmit LIN frame including header and response in UART LIN mode.
<b>UART_PDMA</b>	Transmit and receive UART data with PDMA.
<b>UART_RS485</b>	Transmit and receive data in UART RS485 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.
<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. It supports to use GPIO to simulate key input.
<b>USBD_HID_Mouse</b>	Show how to implement a USB mouse device. The mouse cursor will move automatically when this mouse device connecting to PC by USB.
<b>USBD_HID_MouseKeyboard</b>	Simulate an USB HID mouse and HID keyboard. Mouse draws circle on the screen and Keyboard use GPIO to simulate key input.
<b>USBD_HID_RemoteWakeup</b>	Demonstrate how to implement a USB mouse device. It use PA0 ~ PA5 to control mouse direction and mouse key. It also supports USB suspend and remote wakeup.
<b>USBD_HID_Touch</b>	Demonstrate how to implement a USB touch digitizer device. Two lines demo in Paint.
<b>USBD_HID_Transfer</b>	Demonstrate how to 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_K</b>	Demonstrate how to implement a composite device (HID

<b>Keyboard</b>	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 (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_SRAM</b>	Use internal SRAM as back end storage media to simulate a 44 KB 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 (USB micro printer device and HID Transfer). 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_VCOM_And_HID_Keyboard</b>	Demonstrate how to implement a composite device (VCOM and HID Keyboard).
<b>USBD_VCOM_And_HID_Transfer</b>	Demonstrate how to implement a composite device (VCOM and HID Transfer). It supports one virtual COM port and transfers 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_VCOM_DualPort</b>	Demonstrate how to implement a USB dual virtual COM port device.
<b>USBD_VCOM_SerialEmulator</b>	Demonstrate how to implement a USB virtual COM port device.
<b>USBH_AudioClass</b>	Demonstrate how to use USBH Audio Class driver. It shows the mute, volume, auto-gain, channel and sampling rate control.

<b>USBH_DEV_CONN</b>	Use connect/disconnect callback functions to handle of device connect and disconnect events.
<b>USBH_HID</b>	Use USB Host core driver and HID driver. This sample demonstrates how to submit HID class request and how to read data from interrupt pipe. This sample supports dynamic device plug/un-plug and multiple HID devices.
<b>USBH_HID_Keybaord</b>	Demonstrate reading key inputs from USB keyboards. This sample includes an USB keyboard driver which is based on the HID driver.
<b>USBH_MassStorage</b>	This sample uses a command-shell-like interface to demonstrate how to use USBH mass storage driver and make it working as a disk driver under FATFS file system.
<b>USBH_UAC_HID</b>	This sample shows how to use USBH Audio Class driver and HID driver at the same time. The target device is a Game Audio (UAC+HID composite device).
<b>USBH_UAC_LoopBack</b>	Receives audio data from UAC device, and immediately send back to that UAC device.
<b>USBH_VCOM</b>	Demonstrates how to use the USB Host core driver and CDC driver to connect a CDC class VCOM device.
<b>USCI_I2C_EEPROM</b>	Demonstrate how to access EEPROM through a USCI_I2C interface.
<b>USCI_I2C_Loopback</b>	Demonstrate how a Master accesses Slave.
<b>USCI_I2C_Loopback_10bit</b>	Demonstrate how a Master uses 10-bit addressing access Slave.
<b>USCI_I2C_Master</b>	Demonstrate how a Master access Slave. This sample code needs to work with I2C_Slave.
<b>USCI_I2C_Master_10bit</b>	Demonstrate how a Master use 10-bit addressing access Slave. This sample code needs to work with I2C_Slave.
<b>USCI_I2C_MultiBytes_Master</b>	Demonstrate how to use multi-bytes API to access slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_SingleByte_Master</b>	Demonstrate how to use single byte API to access slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_Slave</b>	Demonstrate how to set I2C in slave mode to receive the data

	from a Master. This sample code needs to work with I2C_Master.
<b>USCI_I2C_Slave_10bit</b>	Demonstrate how to set I2C in 10-bit addressing slave mode to receive the data from a Master. This sample code needs to work with I2C_Master.
<b>USCI_I2C_Wakeup_Slave</b>	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.
<b>USCI_SPI_Loopback</b>	Implement USCI_SPI1 Master loop back transfer. This sample code needs to connect USCI_SPI1_MISO pin and USCI_SPI1_MOSI pin together. It will compare the received data with transmitted data.
<b>USCI_SPI_MasterMode</b>	Configure USCI_SPI1 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device. This sample code needs to work with USCI_SPI_SlaveMode sample code.
<b>USCI_SPI_PDMA_LoopTest</b>	Demonstrate USCI_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_SPI1 as Slave mode and demonstrate how to communicate with an off-chip SPI Master device. This sample code needs to work with USCI_SPI_MasterMode sample code.
<b>USCI_UART_AutoBaudRate</b>	Show how to use auto baud rate detection function.
<b>USCI_UART_Autoflow</b>	Transmit and receive data using auto flow control.
<b>USCI_UART_PDMA</b>	Transmit and receive UART data with PDMA.
<b>USCI_UART_RS485</b>	Transmit and receive data in RS485 mode.
<b>USCI_UART_TxRxFunction</b>	Transmit and receive data from PC terminal through a RS232 interface.
<b>USCI_UART_Wakeup</b>	Show how to wake up system from Power-down mode by USCI interrupt in UART mode.
<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.
WWDT_CompareINT	Show how to reload the WWDT counter value.

## 10 .\SampleCode\TrustZone

<b>CSSD_LED</b>	Demonstrate the how to implement code for Collaborative Secure Software Development in both secure and non-secure code.
<b>HardFault</b>	Show the hard fault usages in both secure and non-secure code.
<b>Template</b>	Demonstrate the how to implement code for secure and non-secure.





## 11 .\SampleCode\XOM

XOMLib	Demonstrate how to create XOM library.
XOMLibDemo	Demonstrate how to use XOMLib.

**12 REVISION HISTORY**

Date	Revision	Description
2018.09.06	3.00.001	Initially issued.
2018.10.30	3.00.002	Add TRNG sample code

### 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.*