

## M031 Series BSP Directory

Directory Introduction for 32-bit NuMicro® Family

### Directory Information

Please extract the “M031\_Series BSP\_CMSIS\_V3.00.000.zip” file firstly, and then put the “M031\_Series BSP\_CMSIS\_V3.00.000” folder into the working folder (e.g. .\Nuvoton\BSP Library).

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.

*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\

CMSIS.html	<p>Introduction of CMSIS version 4.5.0. 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>
NuMicro M031 Series CMSIS BSP Revision History.pdf	The revision history of M031 BSP.
NuMicro M031 Series Driver Reference Guide.chm	The usage of drivers in M031 BSP.

## 2 .\Library\

<b>CMSIS\</b>	Cortex® Microcontroller Software Interface Standard (CMSIS) V4.5.0 definitions by ARM® Corp.
<b>Device\</b>	CMSIS compliant device header file.
<b>StdDriver\</b>	All peripheral driver header and source files.

### 3 .\Sample Code\

<b>Hard_Fault_Sample\</b>	<p>Show hard fault information when hard fault happened.</p> <p>The hard fault handler shows some information included program counter, which is the address where the processor was executing when a hard fault occurred. 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>Semihost\</b>	<p>Show how to print and get character through IDE console window.</p>
<b>RegBased\</b>	<p>The sample codes that access control registers directly.</p>
<b>StdDriver\</b>	<p>Demonstrate the usage of M031 series MCU peripheral driver APIs.</p>
<b>Template\</b>	<p>A project template for M031 series MCU.</p>

## 4 .\SampleCode\RegBased

<b>ACMP_ComapreVBG</b>	Demonstrate analog comparator (ACMP) comparison by comparing ACMP1_P1 input and VBG voltage and shows the result on UART console.
<b>ACMP_Wakeup</b>	Use ACMP to wake up system from Power-down mode while comparator output changes.
<b>ACMP_Window Compare</b>	Show how to monitor ACMP input with window compare function.
<b>ACMP_WindowLatch</b>	Demonstrate how to use ACMP window latch mode.
<b>ADC_2Msps_ContinuousScanMode</b>	Demonstrate how to use the iPLL as ADC clock source to achieve 2 Msps ADC conversion rate.
<b>ADC_1411ksps_ContinuousScanMode</b>	Demonstrate how to use HIRC as ADC clock source to achieve 1411 ksps ADC conversion rate.
<b>ADC_ADINT_Trigger</b>	Use ADINT interrupt to do the ADC Single-cycle scan conversion.
<b>ADC_BandGap</b>	Convert Band-gap (channel 29) and print conversion result.
<b>ADC_BandGap CalculateAVDD</b>	Demonstrate how to calculate precise battery voltage (AV <sub>DD</sub> ) by using internal voltage reference (band-gap).
<b>ADC_BurstMode</b>	Perform A/D Conversion with ADC burst mode.
<b>ADC_Continuous ScanMode</b>	Perform A/D Conversion with ADC continuous scan mode.
<b>ADC_PDMA_PWM_Trigger</b>	Demonstrate how to trigger ADC by PWM and transfer conversion data by PDMA.
<b>ADC_PWM_Trigger</b>	Demonstrate how to trigger ADC by PWM.
<b>ADC_ResultMonitor</b>	Monitor the conversion result of channel 2 by the digital compare function.
<b>ADC_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan

	mode.
<b>ADC_SingleMode</b>	Perform A/D Conversion with ADC single mode.
<b>ADC_STADC_Trigger</b>	Show how to trigger ADC by STADC pin.
<b>ADC_SwTrg_Trigger</b>	Trigger ADC by writing ADC software trigger register.
<b>ADC_Timer_Trigger</b>	Show how to trigger ADC by timer.
<b>CLK_ClockDetector</b>	Demonstrate the usage of clock fail detector and clock frequency range detector 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_PDMA</b>	Implement CRC in CRC-32 mode and get the CRC checksum result.
<b>EBI_NOR</b>	Configure EBI interface to access NOR Flash connected to EBI interface.
<b>EBI_SRAM</b>	Configure EBI interface to access SRAM connected to EBI interface.
<b>FMC_IAP</b>	Demonstrate FMC IAP boot mode and show how to use vector remap function. LDROM image was embedded in APROM image and be programmed to LDROM Flash at run-time. This sample also shows how to branch between APROM and LDROM.
<b>FMC_RW</b>	Show FMC read Flash IDs, erase, read, and write functions.
<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/output control.
<b>GPIO_PowerDown</b>	Show how to wake up system from Power-down mode by GPIO interrupt.

<b>HDIV</b>	Demonstrate how to divide two signed integers by HDIV engine.
<b>I2C_Double_Buffer_Slave</b>	Demonstrate how to set I <sup>2</sup> C double buffer in slave mode to receive data from a master continuously.
<b>I2C_EEPROM</b>	Show how to use I <sup>2</sup> C interface to access EEPROM.
<b>I2C_GCMode_Master</b>	Show how a master uses I <sup>2</sup> C address 0x0 to write data to a slave. This sample code needs to work with I2C_GCMode_Slave.
<b>I2C_GCMode_Slave</b>	Show how a Slave receives data from a master in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.
<b>I2C_Loopback</b>	Demonstrate how to set I <sup>2</sup> C Master mode and Slave Mode, and show how a master accesses a slave on a chip.
<b>I2C_Master</b>	Show how a master accesses a slave. This sample code needs to work with I2C_Slave.
<b>I2C_Master_PDMA</b>	Show how a master accesses a slave using PDMA TX and PDMA RX mode (Loopback).
<b>I2C_MultiBytes_Master</b>	Show how to set I <sup>2</sup> C Multi bytes API Read and Write data to Slave. This sample code needs to work with I2C_Slave.
<b>I2C_PDMA</b>	Demonstrate I <sup>2</sup> C PDMA mode and need to connect I2C0 (master) and I2C1 (slave).
<b>I2C_SingleByte_Master</b>	Show how to use I <sup>2</sup> C Single byte API Read and Write data to Slave. This sample code needs to work with I2C_Slave.
<b>I2C_Slave</b>	Demonstrate how to set I <sup>2</sup> C in Slave mode to receive 256 bytes 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 to receive data from a master (Loopback).
<b>I2C_Wakeup_Slave</b>	Show how to wake up MCU from Power-down mode via the I <sup>2</sup> C interface. This sample code needs to work with I2C_Master.

<b>I2S_Master</b>	Configure SPI in I <sup>2</sup> S Master mode and demonstrate how I <sup>2</sup> S works in Master mode.
<b>I2S_PDMA_NAU8822</b>	An I <sup>2</sup> S demo with PDMA function connected to audio codec NAU8822.
<b>I2S_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>I2S_PDMA_PlayRecrod</b>	An I <sup>2</sup> S demo for playing and recording data with PDMA function.
<b>I2S_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>I2S_Slave</b>	Configure SPI 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 I2S_Master.
<b>PDMA_ADC_1882ksps_ContinuousScanMode</b>	Demonstrate how to transfer 1.822 Msps conversion data from ADC to SRAM by PDMA when an ADC clock source is assigned to HXT that is connected to 32 MHz external crystal.
<b>PDMA_BasicMode</b>	Use PDMA channel 1 to transfer data from memory to memory.
<b>PDMA_ScatterGather</b>	Use PDMA channel 1 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).
<b>PDMA_TimerCapture_CalculatePlusePeriod</b>	Demonstrate timer capture and PDMA transfer function to measure the external signal frequency.
<b>PDMA_TimerTrigger_SRAM2GPIO</b>	Demonstrate how to use PDMA channel 1 to transfer SRAM data to GPIO for generating specified output waveform according to the SRAM data.
<b>PWM_240kHz_DutySwitch</b>	Demonstrate how to set PWM0 channel 0 outputs 240 kHz waveform and switch duty in each 0.5%.
<b>PWM_Capture</b>	Capture the PWM0 Channel 0 waveform by PWM0 Channel 2.



<b>PWM_DeadZone</b>	Demonstrate how to use PWM Dead Time function.
<b>PWM_DoubleBuffer_PeriodLoadingMode</b>	Change duty cycle and period of output waveform by PWM Double Buffer function.
<b>PWM_DutySwitch</b>	Change duty cycle of output waveform by configured period.
<b>PWM_OutputWaveForm</b>	Demonstrate how to use PWM counter output waveform.
<b>PWM_PDMA_Capture</b>	Capture the PWM0 Channel 0 waveform by PWM0 Channel 2, and use PDMA to transfer captured data.
<b>PWM_PDMA_Capture_1MHz Signal</b>	Demonstrate how to use PWM Channel 2 capture to capture PWM0 Channel 0 waveform, and use PDMA to transfer captured data to SRAM. The frequency of PWM Channel 0 is 1 MHz, which is used to test the maximum input frequency for PWM Capture function.
<b>PWM_SyncStart</b>	Demonstrate how to use PWM0 counter synchronous start function.
<b>SPI_LoopBack</b>	SPI read/write demo connecting SPI MISO and MOSI pins.
<b>SPI_MasterFIFOmode</b>	Configure SPI 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.
<b>SPI_PDMA_LoopTest</b>	SPI read/write demo in PDMA mode. Connecting SPI MISO and MOSI pins. Both TX PDMA function and RX PDMA function will be enabled.
<b>SPI_SlaveFIFOmode</b>	Configure SPI 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.
<b>SYS_BODWakeup</b>	Demonstrate how to wake up system from Power-down mode by brown-out detector interrupt.
<b>SYS_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLKO pin.
<b>SYS_PowerDown_</b>	Demonstrate how to set the related GPIO and analog

<b>MinCurrent</b>	macro setting for getting the minimize power consumption when the chip enters Power-down mode.
<b>SYS_TrimHIRC</b>	Demonstrate how to use LXT to trim HIRC.
<b>TIMER_ACMPTrigger</b>	Use ACMP to trigger timer reset mode.
<b>TIMER_CaptureCounter</b>	Show how to use the Timer capture function to capture Timer counter value.
<b>TIMER_Delay</b>	Demonstrate the usage of TIMER_Delay() API to generate a 1 second delay.
<b>TIMER_EventCounter</b>	Use TM0 pin to demonstrate timer event counter function.
<b>TIMER_FreeCountingMode</b>	Use the timer TM0_EXT pin to demonstrate timer free counting mode function, and display the measured input frequency to UART console.
<b>TIMER_InterTimer TriggerMode</b>	Use the timer TM0 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_SW_RTC</b>	Show how to use software to simulate RTC in Power-down mode. Use the Timer to wake up the MCU and display RTC time per second.
<b>TIMER_TimeOutWakeup</b>	Use timer to wake up system from Power-down mode periodically.
<b>TIMER_ToggleOut</b>	Demonstrate the timer 0 toggle out function on TM0 pin.
<b>UART_6Mbps_SingleWire</b>	Demonstrate how to transfer 6 Mbps UART data through UART single-wire interface.
<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 via the RS232 interface.
<b>UART_Wakeup</b>	Show how to wake up system from Power-down mode by UART interrupt.
<b>USCI_I2C_EEPROM</b>	Show how to use USCI_I2C interface to access EEPROM.
<b>USCI_I2C_Master</b>	Show how a master accesses a slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_Master_10bit</b>	Show how a master accesses a 10-bit address slave. This sample code need works with USCI_I2C_Slave_10bit.
<b>USCI_I2C_Monitor</b>	Demonstrate USCI_I2C Monitor mode.
<b>USCI_I2C_MultiBytes_Master</b>	Show how to set USCI_I2C Multi bytes API Read and Write data to Slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_SingleByte_Master</b>	Show how to use USCI_I2C Single byte API Read and Write data to Slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_Slave</b>	Show how a slave receives data from a master. This sample code needs to work with USCI_I2C_Master.
<b>USCI_I2C_Slave_10bit</b>	Show how a 10-bit address slave receives data from a master. This sample code need works with USCI_I2C_Master_10bit.
<b>USCI_I2C_Wakeup_Slave</b>	Show how to wake up USCI_I2C from Deep Sleep mode. This sample code needs to work with USCI_I2C_Master.
<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 USCI_SPI_SlaveMode.
<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 USCI_SPI_MasterMode.
<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 USCI_UART_Autoflow_Slave.
<b>USCI_UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. This sample code needs to work with USCI_UART_Autoflow_Master.
<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 USCI_UART_RS485_Slave.
<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Master.
<b>USCI_UART_TxRxFunction</b>	Transmit and receive data from PC terminal via the 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.
<b>WWDT_ReloadCounter</b>	Demonstrate how to reload the WWDT counter value without resetting the MCU.

## 5 .\SampleCode\StdDriver

<b>ACMP_ComapreVBG</b>	Demonstrate analog comparator (ACMP) comparison by comparing ACMP1_P1 input and VBG voltage and shows the result on UART console.
<b>ACMP_Wakeup</b>	Use ACMP to wake up system from Power-down mode while comparator output changes.
<b>ACMP_WindowCompare</b>	Show how to monitor ACMP input with window compare function.
<b>ACMP_WindowLatch</b>	Demonstrate how to use ACMP window latch mode.
<b>ADC_2Msps_ContinuousScanMode</b>	Demonstrate how to use the iPLL as ADC clock source to achieve 2 Msps ADC conversion rate.
<b>ADC_1411ksps_ContinuousScanMode</b>	Demonstrate how to use HIRC as ADC clock source to achieve 1411 ksps ADC conversion rate.
<b>ADC_ADINT_Trigger</b>	Use ADINT interrupt to do the ADC Single-cycle scan conversion.
<b>ADC_BandGap</b>	Convert Band-gap (channel 29) and print conversion result.
<b>ADC_BandGapCalculateAVDD</b>	Demonstrate how to calculate precise battery voltage (AV <sub>DD</sub> ) by using internal voltage reference (band-gap).
<b>ADC_BurstMode</b>	Perform A/D Conversion with ADC burst mode.
<b>ADC_ContinuousScanMode</b>	Perform A/D Conversion with ADC continuous scan mode.
<b>ADC_PDMA_PWM_Trigger</b>	Demonstrate how to trigger ADC by PWM and transfer conversion data by PDMA.
<b>ADC_PWM_Trigger</b>	Demonstrate how to trigger ADC by PWM.
<b>ADC_ResultMonitor</b>	Monitor the conversion result of channel 2 by the digital compare function.
<b>ADC_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan

	mode.
<b>ADC_SingleMode</b>	Perform A/D Conversion with ADC single mode.
<b>ADC_STADC_Trigger</b>	Show how to trigger ADC by STADC pin.
<b>ADC_SwTrg_Trigger</b>	Trigger ADC by writing ADC software trigger register.
<b>ADC_Timer_Trigger</b>	Show how to trigger ADC by timer.
<b>CLK_ClockDetector</b>	Demonstrate the usage of clock fail detector and clock frequency range detector 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_PDMA</b>	Implement CRC in CRC-32 mode and get the CRC checksum result.
<b>EBI_NOR</b>	Configure EBI interface to access NOR Flash connected to EBI interface.
<b>EBI_SRAM</b>	Configure EBI interface to access SRAM connected to EBI interface.
<b>FMC_IAP</b>	Demonstrate FMC IAP boot mode and show how to use vector remap function. LDROM image was embedded in APROM image and be programmed to LDROM Flash at run-time. This sample also shows how to branch between APROM and LDROM.
<b>FMC_RW</b>	Show FMC read Flash IDs, erase, read, and write functions.
<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/output control.
<b>GPIO_PowerDown</b>	Show how to wake up system from Power-down mode by GPIO interrupt.

<b>HDIV</b>	Demonstrate how to divide two signed integers by HDIV engine.
<b>I2C_Double_Buffer_Slave</b>	Demonstrate how to set I <sup>2</sup> C double buffer in slave mode to receive data from a master continuously.
<b>I2C_EEPROM</b>	Show how to use I <sup>2</sup> C interface to access EEPROM.
<b>I2C_GCMode_Master</b>	Show how a master uses I <sup>2</sup> C address 0x0 to write data to a slave. This sample code needs to work with I2C_GCMode_Slave.
<b>I2C_GCMode_Slave</b>	Show how a slave receives data from a master in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.
<b>I2C_Loopback</b>	Demonstrate how to set I <sup>2</sup> C Master mode and Slave Mode, and show how a master accesses a slave on a chip.
<b>I2C_Master</b>	Show how a master accesses a slave. This sample code needs to work with I2C_Slave.
<b>I2C_Master_PDMA</b>	Show how a master accesses a slave using PDMA TX and PDMA RX mode (Loopback).
<b>I2C_MultiBytes_Master</b>	Show how to set I <sup>2</sup> C Multi bytes API Read and Write data to Slave. This sample code needs to work with I2C_Slave.
<b>I2C_PDMA</b>	Demonstrate I <sup>2</sup> C PDMA mode and need to connect I2C0 (master) and I2C1 (slave).
<b>I2C_SingleByte_Master</b>	Show how to use I <sup>2</sup> C Single byte API Read and Write data to Slave. This sample code needs to work with I2C_Slave.
<b>I2C_Slave</b>	Demonstrate how to set I <sup>2</sup> C in Slave mode to receive 256 bytes 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 to receive data from a master (Loopback).
<b>I2C_Wakeup_Slave</b>	Show how to wake up MCU from Power-down mode via the I <sup>2</sup> C interface. This sample code needs to work with I2C_Master.



<b>I2S_Master</b>	Configure SPI in I <sup>2</sup> S Master mode and demonstrate how I <sup>2</sup> S works in Master mode.
<b>I2S_PDMA_NAU8822</b>	An I <sup>2</sup> S demo with PDMA function connected to audio codec NAU8822.
<b>I2S_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>I2S_PDMA_PlayRecrod</b>	An I <sup>2</sup> S demo for playing and recording data with PDMA function.
<b>I2S_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>I2S_Slave</b>	Configure SPI 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 I2S_Master.
<b>PDMA_ADC_1882ksps_ContinuousScanMode</b>	Demonstrate how to transfer 1.822 Msps conversion data from ADC to SRAM by PDMA when an ADC clock source is assigned to HXT that is connected to 32 MHz external crystal.
<b>PDMA_BasicMode</b>	Use PDMA channel 1 to transfer data from memory to memory.
<b>PDMA_ScatterGather</b>	Use PDMA channel 1 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).
<b>PDMA_TimerCapture_CalculatePlusePeriod</b>	Demonstrate timer capture and PDMA transfer function to measure the external signal frequency
<b>PDMA_TimerTrigger_SRAM2GPIO</b>	Demonstrate how to use PDMA channel 1 to transfer SRAM data to GPIO for generating specified output waveform according to the SRAM data
<b>PWM_240kHz_DutySwitch</b>	Demonstrate the 24 Mbsp SPI master function for implementing SPI Master loop back transfer. This sample code needs to connect MISO_0(PA.0) and MOSI_0(PA.1) together. It will compare the received data with transmitted data.



<b>PWM_Capture</b>	Capture the PWM0 Channel 0 waveform by PWM0 Channel 2.
<b>PWM_DeadZone</b>	Demonstrate how to use PWM Dead Time function.
<b>PWM_DoubleBuffer_PeriodLoadingMode</b>	Change duty cycle and period of output waveform by PWM Double Buffer function.
<b>PWM_DutySwitch</b>	Change duty cycle of output waveform by configured period.
<b>PWM_OutputWaveForm</b>	Demonstrate how to use PWM counter output waveform.
<b>PWM_PDMA_Capture</b>	Capture the PWM0 Channel 0 waveform by PWM0 Channel 2, and use PDMA to transfer captured data.
<b>PWM_PDMA_Capture_1MHz Signal</b>	Demonstrate how to use PWM Channel 2 capture to capture PWM0 Channel 0 waveform, and use PDMA to transfer captured data to SRAM. The frequency of PWM Channel 0 is 1 MHz, which is used to test the maximum input frequency for PWM Capture function.
<b>PWM_SyncStart</b>	Demonstrate how to use PWM0 counter synchronous start function.
<b>SPI_LoopBack</b>	SPI read/write demo connecting SPI MISO and MOSI pins.
<b>SPI_MasterFIFOmode</b>	Configure SPI 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.
<b>SPI_PDMA_LoopTest</b>	SPI read/write demo in PDMA mode. Connecting SPI MISO and MOSI pins. Both TX PDMA function and RX PDMA function will be enabled.
<b>SPI_SlaveFIFOmode</b>	Configure SPI 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.
<b>SYS_BODWakeup</b>	Demonstrate how to wake up system from 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_PowerDown_MinCurrent</b>	Demonstrate how to set the related GPIO and analog macro setting for getting the minimize power consumption when the chip enters Power-down mode.
<b>SYS_TrimHIRC</b>	Demonstrate how to use LXT to trim HIRC.
<b>TIMER_ACMPTrigger</b>	Use ACMP to trigger timer reset mode.
<b>TIMER_CaptureCounter</b>	Show how to use the Timer capture function to capture Timer counter value.
<b>TIMER_Delay</b>	Demonstrate the usage of TIMER_Delay() API to generate a 1 second delay.
<b>TIMER_EventCounter</b>	Use TM0 pin to demonstrate timer event counter function.
<b>TIMER_FreeCountingMode</b>	Use the timer TM0_EXT pin to demonstrate timer free counting mode function, and display the measured input frequency to UART console.
<b>TIMER_InterTimerTriggerMode</b>	Use the timer TM0 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_SW_RTC</b>	Show how to use software to simulate RTC in Power-down mode. Use the Timer to wake up the MCU and display RTC time per second.
<b>TIMER_TimeOutWakeup</b>	Use timer to wake up system from Power-down mode periodically.
<b>TIMER_ToggleOut</b>	Demonstrate the timer 0 toggle out function on TM0 pin.
<b>UART_6Mbps_SingleWire</b>	Demonstrate how to transfer 6 Mbps UART data through UART single-wire interface.
<b>UART_115200bps_SingleWire_ISP</b>	Demonstrate the ISP protocol for using 115200 bps UART single wire interface to update the APROM.
<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 via the RS232 interface.
<b>UART_Wakeup</b>	Show how to wake up system from Power-down mode by UART interrupt.
<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 draw circles on the screen.
<b>USBD_HID_MouseKeyboard</b>	Simulate a USB HID mouse and HID keyboard. Mouse draws circles on the screen and Keyboard uses GPIO to simulate key input.
<b>USBD_HID_RemoteWakeup</b>	Simulate how a HID mouse 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 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 (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 (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 back end 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 (USB micro printer device and HID Transfer) to transfer data between a 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 (VCOM and HID keyboard).
<b>USBD_VCOM_And_HID_Transfer</b>	Demonstrate how to implement a composite device (VCOM and HID Transfer) to transfer data between a 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 (Virtual COM port and Mass storage 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>USCI_I2C_EEPROM</b>	Show how to use USCI_I2C interface to access EEPROM.
<b>USCI_I2C_Master</b>	Show how a master accesses a slave. This sample code needs to work with USCI_I2C_Slave.

<b>USCI_I2C_Master_10bit</b>	Show how a master accesses a 10-bit address slave. This sample code need works with USCI_I2C_Slave_10bit.
<b>USCI_I2C_Monitor</b>	Demonstrate USCI_I2C Monitor mode.
<b>USCI_I2C_MultiBytes_Master</b>	Show how to set USCI_I2C use Multi bytes API Read and Write data to Slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_SingleByte_Master</b>	Show how to use USCI_I2C Single byte API Read and Write data to Slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_Slave</b>	Show how a slave receives data from a master. This sample code needs to work with USCI_I2C_Master.
<b>USCI_I2C_Slave_10bit</b>	Show how a 10-bit address slave receives data from a master. This sample code needs to work with USCI_I2C_Master_10bit.
<b>USCI_I2C_Wakeup_Slave</b>	Show how to wake up USCI_I2C from Deep Sleep mode. This sample code needs to work with USCI_I2C_Master.
<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 USCI_SPI_SlaveMode.
<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 USCI_SPI_MasterMode.
<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 USCI_UART_Autoflow_Slave.
<b>USCI_UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. This sample code needs to work with

	USCI_UART_Autoflow_Master.
<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 USCI_UART_RS485_Slave.
<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Master.
<b>USCI_UART_TxRxFunction</b>	Transmit and receive data from PC terminal via the 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.
<b>WWDT_ReloadCounter</b>	Demonstrate how to reload the WWDT counter value without resetting the MCU.

## 6 REVISION HISTORY

Date	Revision	Description
2018.08.22	3.00.000	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.*