

# NUC1263 Series CMSIS BSP Guide

Directory Introduction for 32-bit NuMicro™ Family

## Directory Information

Please extract the “NUC1263Series\_BSP\_CMSIS\_V3.00.004.zip” file firstly, and then put the “NUC1263Series\_BSP\_CMSIS\_V3.00.004” 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\

<p><b>CMSIS.html</b></p>	<p>Introduction of CMSIS version 5. 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>
<p><b>Revision History.pdf</b></p>	<p>The revision history of NUC1263 Series BSP.</p>
<p><b>NuMicro NUC1263 Series Driver Reference Guide.chm</b></p>	<p>The usage of drivers in NUC1263 Series BSP.</p>

## 2 .\Library\

<b>CMSIS\</b>	Cortex® Microcontroller Software Interface Standard (CMSIS) V5 definitions by ARM® Corp.
<b>Device\</b>	CMSIS compliant device header file.
<b>SpdhLib\</b>	Library of SPD5 Hub.
<b>StdDriver\</b>	All peripheral driver header and source files.

### 3 .\Sample Code\

<p><b>Hard_Fault_Sample\</b></p>	<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>
<p><b>ISP\</b></p>	<p>Sample codes for In-System-Programming.</p>
<p><b>Semihost\</b></p>	<p>Show how to print and get character through IDE console window.</p>
<p><b>RegBased\</b></p>	<p>The sample codes which access control registers directly.</p>
<p><b>StdDriver\</b></p>	<p>Demonstrate the usage of NUC1263 series MCU peripheral driver APIs.</p>
<p><b>Template\</b></p>	<p>A project template for NUC1263 series MCU.</p>

## 4 .\SampleCode\ISP

<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 I2C interface.
<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.

## 5 .\SampleCode\RegBased

<b>ACMP</b>	Demonstrate how 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>ADC_BandGap</b>	Convert Band-gap (channel 29) and print conversion result.
<b>ADC_ContinuousScanMode</b>	Perform A/D Conversion with ADC continuous scan mode.
<b>ADC_PDMA_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan mode and transfer result by PDMA.
<b>ADC_PwmTrigger</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>BPWM_Capture</b>	Use BPWM0 channel 0 to capture the BPWM1 channel 0 waveform.
<b>BPWM_DoubleBuffer_PeriodLoadingMode</b>	Change duty cycle and period of output waveform by BPWM Double Buffer function(Period loading mode).
<b>BPWM_DutySwitch</b>	Change duty cycle of output waveform by configured period.
<b>BPWM_OutputWaveform</b>	Demonstrate how to use BPWM output waveform.
<b>BPWM_SyncStart</b>	Demonstrate how to use BPWM counter synchronous start function.
<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 CRC checksum results.
<b>CRC_CRC8</b>	Implement CRC in CRC-8 mode and get CRC checksum results.

<b>CRC_CRC32</b>	Implement CRC in CRC-32 mode with PDMA transfer.
<b>DAC_ExtPinTrigger</b>	Demonstrate how to trigger DAC conversion by external pin.
<b>DAC_PDMA_TimerTrigger</b>	Demonstrate how to PDMA and trigger DAC by Timer.
<b>DAC_SoftwareTrigger</b>	Write a data to DAC_DAT register to trigger DAC conversion.
<b>DAC_TimerTrigger</b>	Use a Timer to trigger DAC Conversion.
<b>FMC_ExecInSRAM</b>	Implement a code and execute the code in SRAM to program embedded Flash (support KEIL MDK only).
<b>FMC_IAP</b>	Show how to call LDROM functions from APROM. The code in APROM will look up the table at 0x100E00 to get the address of function of LDROM and call the function.
<b>FMC_MultiBoot</b>	Implement a multi-boot system to boot from different applications in APROM. A LDROM code and four APROM code are implemented in this sample code.
<b>FMC_RW</b>	Demonstrate how to read/program embedded Flash by ISP function.
<b>FMC_SPROM</b>	Demonstrate how to set SPROM in security mode.
<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 access a Slave.
<b>I2C_Loopback_10bit</b>	Demonstrate how a Master uses 10-bit addressing to access a Slave.
<b>I2C_Master</b>	Demonstrate how a Master accesses a 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_Slave</b>	Demonstrate how to set I2C in slave mode to receive 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.
<b>I2C_SMBUS</b>	Demonstrate how I2C SMBUS works.
<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_Master</b>	Configure SPI1 as I2S Master mode and demonstrate how I2S works in Master mode. This sample code needs to work with I2S_Slave.
<b>I2S_PDMA_NAU8822</b>	This is an I2S demo with PDMA function connected with NAU8822 codec.
<b>I2S_PDMA_Play</b>	This is an I2S demo for playing data and demonstrating how I2S works with PDMA.
<b>I2S_PDMA_PlayRecord</b>	This is an I2S demo for playing and recording data with PDMA function.
<b>I2S_PDMA_Record</b>	This is an I2S demo for recording data and demonstrating how I2S works with PDMA.
<b>I2S_Slave</b>	Configure SPI1 as I2S Slave mode and demonstrate how I2S works in Slave mode. This sample code needs to work with I2S_Master.
<b>LLSI_Marquee</b>	This is a LLSI demo for marquee display in software mode. It needs to be used with WS2812 LED strip.

<b>LLSI_PDMA_Marquee</b>	This is a LLSI demo for marquee display in PDMA mode. It needs to be used with WS2812 LED strip.
<b>PDMA</b>	Use PDMA Channel 2 to transfer data from memory to memory.
<b>PDMA_ScatterGather_PingPongBuffer</b>	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).
<b>PDMA_Scatter_Gather</b>	Use PDMA Channel 4 to transfer data from memory to memory by scatter-gather mode.
<b>SPI_Loopback</b>	Implement SPI Master loop back transfer. This sample code needs to connect SPI0_MISO0 pin and SPI0_MOSI0 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.
<b>SPI_Mux</b>	Configure SPI1 as SPI Master1 and demonstrate how to access SPI flash through SPI_MUX interface. This sample code needs to work with SPI_Flash_Master2 sample code.
<b>SPI_PDMA_LoopTest</b>	Demonstrate SPI data transfer with PDMA. SPI0 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.
<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_PowerDown_MinCurrent</b>	Demonstrate how to minimize power consumption when entering power down mode.
<b>SYS_TrimHIRC</b>	Demonstrate how to use LXT to trim HIRC.

<b>SYS_VoltageDetector</b>	Show how to use voltage detector to detect pin input voltage.
<b>TIMER_CaptureCounter</b>	Show how to use the timer2 capture function to capture timer2 counter value.
<b>TIMER_EventCounter</b>	Show how to use the timer2 capture function to capture timer2 counter value.
<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>TS_TemperatureMeasure</b>	Measure the current temperature by Temperature Sensor.
<b>UART_AutoBaudRate</b>	Show how to use auto baud rate detection function.
<b>UART_Autoflow</b>	Transmit and receive data with auto flow control.
<b>UART_IrDA</b>	Transmit and receive data in UART IrDA 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>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_CompareINT</b>	Show how to reload the WWDT counter value.

## 6 .\SampleCode\StdDriver

<b>ACMP_CompareDAC</b>	Demonstrate how ACMP compare DAC output with ACMP1_P1 value.
<b>ACMP_CompareVBG</b>	Demonstrate how ACMP compare VBG output with ACMP1_P1 value.
<b>ACMP_Wakeup</b>	Show how to wake up MCU from Power-down mode by ACMP wake-up function.
<b>ACMP_WindowCompare</b>	Demonstrate the usage of ACMP window compare function.
<b>ADC_BandGap</b>	Convert Band-gap (channel 29) and print conversion result.
<b>ADC_ContinuousScanMode</b>	Perform A/D Conversion with ADC continuous scan mode.
<b>ADC_PDMA_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan mode and transfer result by PDMA.
<b>ADC_PwmTrigger</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>BPWM_Capture</b>	Use BPWM0 channel 0 to capture the BPWM1 channel 0 waveform.
<b>BPWM_DoubleBuffer_PeriodLoadingMode</b>	Change duty cycle and period of output waveform by BPWM Double Buffer function.(Period loading mode).
<b>BPWM_DutySwitch</b>	Change duty cycle of output waveform by configured period.
<b>BPWM_OutputWaveform</b>	Demonstrate how to use BPWM counter output waveform.
<b>BPWM_SyncStart</b>	Demonstrate how to use BPWM counter synchronous start function.
<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_CRC32</b>	Implement CRC in CRC-32 mode with PDMA transfer.
<b>CRC_CRC8</b>	Implement CRC in CRC-8 mode and get the CRC checksum result.
<b>DAC_ExtPinTrigger</b>	Demonstrate how to trigger DAC conversion by external pin.
<b>DAC_PDMA_TimerTrigger</b>	Demonstrate how to PDMA and trigger DAC by Timer.
<b>DAC_SoftwareTrigger</b>	Write a data to DAC_DAT register to trigger DAC conversion.
<b>DAC_TimerTrigger</b>	Use a Timer to trigger DAC Conversion.
<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 set VECMAP to LDROM and reboot to LDROM from APROM.
<b>FMC_RW</b>	Demonstrate how to read/program embedded Flash by ISP function.
<b>FMC_SPROM</b>	Demonstrate how to set SPROM in security mode.
<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</b>	Show how a Master uses I2C address 0x0 to write data to Slave. This sample code needs to work with I2C_GCMode_Slave.
<b>I2C_Loopback</b>	Demonstrate how a Master access Slave.
<b>I2C_Loopback_10bit</b>	Demonstrate how a Master uses 10-bit addressing to access

	a 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>	Show how to set I2C use Multi bytes API Read and Write data to Slave. Needs to work with I2C_Slave sample code
<b>I2C_SingleByte_Master</b>	Show how to use I2C Single byte API Read and Write data to Slave. Needs to work with I2C_Slave sample code.
<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_SMBUS</b>	Demonstrate how I2C SMBUS works.
<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_Master</b>	Configure SPI1 as I2S Master mode and demonstrate how I2S works in Master mode. This sample code needs to work with I2S_Slave.
<b>I2S_PDMA_NAU8822</b>	This is an I2S demo with PDMA function connected with NAU8822 codec.
<b>I2S_PDMA_Play</b>	This is an I2S demo for playing data and demonstrating how I2S works with PDMA.
<b>I2S_PDMA_PlayRecord</b>	This is an I2S demo for playing and recording data with PDMA function.
<b>I2S_PDMA_Record</b>	This is an I2S demo for recording data and demonstrating how I2S works with PDMA.
<b>I2S_Slave</b>	Configure SPI1 as I2S Slave mode and demonstrate how I2S works in Slave mode. This sample code needs to work with I2S_Master.

<b>I3CS_Slave_HotJoin</b>	Demonstrate how to initiate a Hot-Join request to an I3C Master.
<b>I3CS_Slave_IBI</b>	Demonstrate how to initiate an In-Band Interrupt request to an I3C Master.
<b>I3CS_Slave_Wakeup</b>	Wake up the MCU from Power-down mode after receiving read/write request from the Master.
<b>I3CS_SlaveRW</b>	Demonstrate how to use I3C0 Slave to receive and transmit the data from a Master.
<b>I3CS_SlaveRW_PDMA</b>	Demonstrate how to use I3C0 Slave to receive and transmit the data through PDMA from a Master.
<b>LLSI_Marquee</b>	This is a LLSI demo for marquee display in software mode. It needs to be used with WS2812 LED strip.
<b>LLSI_PDMA_Marquee</b>	This is a LLSI demo for marquee display in PDMA mode. It needs to be used with WS2812 LED strip.
<b>PDMA</b>	Use PDMA Channel 2 to transfer data from memory to memory.
<b>PDMA_ScatterGather_PingPongBuffer</b>	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).
<b>PDMA_Scatter_Gather</b>	Use PDMA Channel 4 to transfer data from memory to memory by scatter-gather mode.
<b>SPI_Loopback</b>	Implement SPI Master loop back transfer. This sample code needs to connect MISO_0 pin and MOSI_0 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.
<b>SPI_Mux</b>	Configure SPI1 as SPI Master1 and demonstrate how to access SPI flash through SPI_MUX interface. This sample code needs to work with SPI_Flash_Master2 sample code.
<b>SPI_PDMA_LoopTest</b>	Demonstrate SPI data transfer with PDMA. SPI0 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.
<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_TrimHIRC</b>	Demonstrate how to use LXT to trim HIRC.
<b>SYS_VoltageDetector</b>	Show how to use voltage detector to detect pin input voltage.
<b>TIMER_CaptureCounter</b>	Show how to use the timer2 capture function to capture timer2 counter value.
<b>TIMER_Delay</b>	Show how to use timer0 to create various delay time.
<b>TIMER_EventCounter</b>	Implement timer1 event counter function to count the external input event.
<b>TIMER_InterTimerTriggerMode</b>	Demonstrate how to use Inter-Timer trigger function.
<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>	Implement timer counting in toggle-output mode.
<b>TS_TemperatureMeasure</b>	Measure the current temperature by Temperature Sensor.
<b>UART_AutoBaudRate</b>	Show how to use auto baud rate detection function.
<b>UART_Autoflow</b>	Transmit and receive data with auto flow control.
<b>UART_IrDA</b>	Transmit and receive data in UART IrDA 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_NAU8822</b>	This sample code demonstrates how to implement a USB audio class device. NAU8822 is used in this sample code to play the audio data from Host. It also supports to record data from NAU8822 to Host.
<b>USBD_HID_Keyboard</b>	Show how to implement a USB keyboard device. This sample code 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_Mouse_BC12</b>	Demonstrate how to implement a USB mouse device with BC1.2 (Battery Charging). which shows different type of charging port after connected USB port. The mouse cursor will move automatically when this mouse device connecting to PC by USB.
<b>USBD_HID_Mouse2</b>	Demonstrate how to implement a USB mouse device. It uses PC0 ~ PC5 to control mouse directions and mouse keys. It also supports USB suspend and remote wakeup.
<b>USBD_HID_MouseKeyboard</b>	Demonstrate how to implement a USB mouse function and a USB keyboard on the same USB device. The mouse cursor will move automatically when this mouse device connecting to PC. This sample code uses a GPIO to simulate key input.
<b>USBD_HID_Transfer</b>	Transfer data between USB device and PC through USB HID interface. A windows tool is also included in this sample code to connect with 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 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_MassStorage_CDRO</b>	Demonstrate how to simulate a USB CD-ROM device.

<b>M</b>	
<b>USB_D_MassStorage_DataFlash</b>	Demonstrate how to implement a USB mass storage. It uses embedded Data Flash as storage.
<b>USB_D_MassStorage_SDCard</b>	Use SD card as storage to implement a USB mass storage device.
<b>USB_D_Micro_Printer</b>	Show how to implement a USB micro printer device.
<b>USB_D_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>USB_D_VCOM_and_HID_Keyboard</b>	Implement a USB composite device with virtual COM port and keyboard functions.
<b>USB_D_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>USB_D_VCOM_and_MassStorage</b>	Implement a USB composite device. It supports one virtual COM port and one USB mass storage device.
<b>USB_D_VCOM_DualPort</b>	Demonstrate how to implement a USB dual virtual COM port device.
<b>USB_D_VCOM_MultiPort</b>	Demonstrate how to implement a USB multi virtual COM port device.
<b>USB_D_VCOM_SinglePort</b>	Implement a USB virtual COM port device. It supports one virtual COM port.
<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_CompareINT</b>	Show how to reload the WWDT counter value.

## 7 REVISION HISTORY

Date	Revision	Description
2024.04.29	3.00.004	<ol style="list-style-type: none"> <li>1. Add USBD_VCOM_MultiPort sample code.</li> <li>2. Add FMC_SPROM sample code.</li> <li>3. Fix ISP_RS485 and ISP_UART sample code GetApromSize is 64KB.</li> <li>4. Add slew rate control for SPI sample codes.</li> <li>5. Remove USBD LPM function.</li> <li>6. Revise retarget.c to fix IAR9.x printf issue.</li> <li>7. Fix LLSI_Open uses PCLK as clock source in LLSI driver.</li> <li>8. Update SYS_PowerDown_MinCurrent sample code, GPIO and UART driver.</li> <li>9. Update USBD HID keyboard sample code to support LED status.</li> <li>10. Modify enable/disable interrupt function to be mask type.</li> </ol>
2023.07.17	3.00.003	<ol style="list-style-type: none"> <li>1. Revise ADC driver for sampling HSA.</li> <li>2. Add spdh GCC Lib.</li> <li>3. Remove SPDH sample code.</li> </ol>
2023.06.15	3.00.002	<ol style="list-style-type: none"> <li>1. Support block write mode in SPDH_LLSIDevice.</li> <li>2. Add spdh_device library for HUB Device application.</li> <li>3. Modify I2C_Master sample code to support 10 bit mode by compiler option.</li> <li>4. Change the verification conditions for read and write operations in I2C_Master to check each address group.</li> <li>5. Add MR register settings file for LLSI device.</li> <li>6. Modify VendorName of Nu_DFU.inf.</li> <li>7. Sync code with internal Hub with lighting device sample code.</li> <li>8. Add Double Flashing Mode on SPDH_LLSIDevice sample.</li> <li>9. Add MR55 register to adjust LED speed on SPDH_LLSIDevice sample.</li> <li>10. Revise I3CS driver.</li> <li>11. Modify UART_Read() to return received data count when time-out.</li> <li>12. Update SPI_Loopback sample code.</li> <li>13. Update SYS_PowerDown_MinCurrent sample code.</li> <li>14. Remove the PHY type setting of USB driver in Library/StdDriver/src/usbd.c</li> <li>15. Modify SYS_UnLockReg() time-out handler in Library/StdDriver/inc/sys.h</li> </ol>

---

2022.09.26	3.00.001	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.*