

## NUC121/NUC125 Series BSP Directory

Directory Introduction for 32-bit NuMicro™ Family

### Directory Information

<b>Document</b>	Driver reference manual and reversion history.
<b>Library</b>	Driver header and source files.
<b>SampleCode</b>	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 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)

## 1 Document Information

<b>CMSIS.html</b>	Document of CMSIS version 5.1.1.
<b>NuMicro NUC121_125 Series CMSIS BSP Driver Reference.chm</b>	This document shows the usage of drivers in NUC121/NUC125 BSP.
<b>NuMicro NUC121_125 Series CMSIS BSP Revision History.pdf</b>	This document shows the revision history of NUC121/NUC125 BSP.

## 2 Library Information

<b>CMSIS</b>	Cortex® Microcontroller Software Interface Standard (CMSIS) V5.1.1 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 Information

<b>Hard_Fault_Sample</b>	<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>
<b>ISP</b>	Sample code for Nuvoton NuMicro ISP Programming Tool.
<b>NuTiny-SDK-NUC121_125</b>	Sample code for NUC121/NUC125 Tiny Board.
<b>RegBased</b>	Sample codes which access control registers directly.
<b>Semihost</b>	Show how to print and get character through IDE console window.
<b>StdDriver</b>	Demonstrate the usage of NUC121/NUC125 series MCU peripheral driver APIs.
<b>Template</b>	A project template for NUC121/NUC125 series MCU.

## 4 \SampleCode\ISP

<b>ISP_DFU</b>	Sample ISP firmware communicated with DFU (Device Firmware Upgrade) tool through a USB DFU interface.
<b>ISP_HID</b>	Sample ISP firmware communicated with ISP tool through a USB HID interface.
<b>ISP_I2C</b>	Sample ISP firmware communicated with ISP tool through an I <sup>2</sup> C interface.
<b>ISP_MSC</b>	Sample ISP firmware communicated with ISP tool through a USB MSC interface.
<b>ISP_RS485</b>	Sample ISP firmware communicated with ISP tool through a RS485 interface.
<b>ISP_SPI</b>	Sample ISP firmware communicated with ISP tool through a SPI interface.
<b>ISP_UART</b>	Sample ISP firmware communicated with ISP tool through a UART interface.

## 5 \SampleCode\NuTiny-SDK-NUC121\_125

**LED\_Toggle**

Toggle PB.4 to turn on / off the board LED.

## 6 \SampleCode\RegBased

<b>ADC_BurstMode</b>	Use burst mode to finish more than 16 ADC conversions for the specified channel.
<b>ADC_ContinuousScanMode</b>	Use continuous scan mode to finish two cycles of ADC conversion for the specified channels.
<b>ADC_MeasureAVDD</b>	Use band-gap voltage to calculate AVDD voltage.
<b>ADC_PDMA_SingleCycleScanMode</b>	Use single-cycle scan mode to finish one cycle of ADC conversion for 4 specified channels and transfer conversion results by PDMA.
<b>ADC_PWMTrigger</b>	Trigger single mode ADC by PWM0 channel0.
<b>ADC_ResultMonitor</b>	Use digital compare function to monitor the ADC conversion result of channel2.
<b>ADC_SingleCycleScanMode</b>	Use single-cycle scan mode to finish one cycle of ADC conversion for 4 specified channels.
<b>ADC_SingleMode</b>	Use single mode to finish the ADC conversion of the specified channel.
<b>BPWM_Capture</b>	Use BPWM1 channel2 to capture BPWM0 channel0 waveform.
<b>BPWM_DoubleBuffer_PeriodLoadingMode</b>	Use BPWM0 channel0~5 and double buffering function to output waveforms. This sample code changes duty cycle of each channel and the period shared by 6 channels.
<b>BPWM_DutySwitch</b>	Change duty cycle of BPWM output waveform by configured period.
<b>BPWM_OutputWaveform</b>	Use BPWM counter to output waveform.
<b>BPWM_SyncStart</b>	Use BPWM counter synchronous start function.

<b>CLK_ClockDetector</b>	Show how to use clock fail detector for HXT / LXT and clock frequency monitor for HXT.
<b>CLK_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLK pin.
<b>FMC_ExecInSRAM</b>	Run code in SRAM by using scatter loading description file.
<b>FMC_IAP</b>	Switch both codes between LDROM and APROM.
<b>FMC_RW</b>	Read and program embedded flash by ISP function.
<b>FMC_SPROM</b>	Show how to change SPROM to secured mode to protect SPROM contents.
<b>GPIO_EINTAndDebounce</b>	Use GPIO external interrupt function.
<b>GPIO_INT</b>	Use GPIO interrupt function.
<b>GPIO_OutputInput</b>	Set GPIO pin mode and use pin data input / output control.
<b>GPIO_PowerDown</b>	Wake up form Power-down mode by GPIO interrupt.
<b>HIRC_Trim<sup>1</sup></b>	Show how to trim HIRC by LXT.
<b>I2C_EEPROM</b>	Access EEPROM by an I <sup>2</sup> C interface.
<b>I2C_GCMode_Master</b>	I <sup>2</sup> C master uses I <sup>2</sup> C address 0x00 to write data to slave. Need to work with the I2C_GCMode_Slave sample code.
<b>I2C_GCMode_Slave</b>	I <sup>2</sup> C slave receives master data in General Call mode. Need to work with the I2C_GCMode_Master sample code.

<sup>1</sup> This sample requires LXT on board.



<b>I2C_Master</b>	Demonstrate how an I <sup>2</sup> C master accesses slave. Need to work with the I2C_Slave sample code.
<b>I2C_MultiBytes_Master</b>	Use I <sup>2</sup> C multiple-byte function to read and write data to slave. Need to work with the I2C_Slave 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>	Use I <sup>2</sup> C single-byte function to read and write data to slave. Need to work with the I2C_Slave sample code.
<b>I2C_Slave</b>	Set I <sup>2</sup> C in slave mode to receive the data of master. Need to work with the I2C_Master, I2C_MultiBytes_Master or I2C_SingleByte_Master sample codes.
<b>I2C_Wakeup_Master</b>	Show how to wake up I <sup>2</sup> C slave from Power-down mode. Need to work with the I2C_Wakeup_Slave sample code.
<b>I2C_Wakeup_Slave</b>	Show how to wake up system in Power-down mode through an I <sup>2</sup> C interface. Need to work with the I2C_Wakeup_Master sample code.
<b>I2S_Master</b>	Demonstrate how I <sup>2</sup> S works in master mode. Need to work with the I2S_Slave sample code.
<b>I2S_PDMA_NAU8822</b>	I <sup>2</sup> S playback and record with NAU8822 codec by PDMA with scatter-gather mode
<b>I2S_PDMA_Play</b>	I <sup>2</sup> S playback by PDMA with scatter-gather mode
<b>I2S_PDMA_PlayRecord</b>	I <sup>2</sup> S playback and record by PDMA with scatter-gather mode
<b>I2S_PDMA_Record</b>	I <sup>2</sup> S record by PDMA with scatter-gather mode

<b>I2S_Slave</b>	Demonstrate how I <sup>2</sup> S works in slave mode. Need to work with the I2S_Master sample code.
<b>PDMA</b>	Use PDMA channel0 to transfer data from memory to memory.
<b>PDMA_Scatter_Gather</b>	Use PDMA channel4 to transfer data from memory to memory with scatter-gather mode.
<b>PDMA_ScatterGather_PingPongBuffer</b>	Use PDMA to implement ping-pong buffer from memory to memory with scatter-gather mode.
<b>PWM_Capture</b>	Use PWM1 channel2 to capture PWM1 channel0 waveform.
<b>PWM_DeadZone</b>	Use PWM dead zone function.
<b>PWM_DoubleBuffer_PeriodLoadingMode</b>	Use PWM0 channel0~1 and double buffering function to output waveforms. This sample code changes duty cycle of each channel and the period shared by 2 channels.
<b>PWM_DutySwitch</b>	Change duty cycle of PWM output waveform by configured period.
<b>PWM_OutputWaveform</b>	Use PWM counter to output waveform.
<b>PWM_PDMA_Capture</b>	Capture PWM1 channel0 waveform by PWM1 channel2, and use PDMA to transfer captured data.
<b>PWM_SyncStart</b>	Use PWM counter synchronous start function.
<b>SPI_MasterFIFOmode</b>	Communicate with an off-chip SPI slave device with FIFO mode. Need to work with the SPI_SlaveFIFOmode sample code.
<b>SPI_PDMA_LoopTest</b>	SPI loopback test with PDMA.
<b>SPI_SlaveFIFOmode</b>	Communicate with an off-chip SPI master device with FIFO mode. Need to work with the SPI_MasterFIFOmode sample code.

<b>SYS_BODWakeup</b>	Wake up system from BOD event.
<b>TIMER_CaptureCounter</b>	Timer2 uses Timer3 toggle output to capture an event counter value from Timer0 toggle output.
<b>TIMER_Delay</b>	Use timer to implement delay cycle.
<b>TIMER_EventCounter</b>	Use Timer1 counter input function to count the input event.
<b>TIMER_PeriodicINT</b>	Perform timer counting in periodic mode.
<b>TIMER_TimeoutWakeup</b>	Use Timer0 toggle-output interrupt event to wake up system.
<b>UART_AutoBaudRate_Master</b>	Show how to use auto baud rate detection function. Need to work with the UART_AutoBaudRate_Slave sample code.
<b>UART_AutoBaudRate_Slave</b>	Show how to use auto baud rate detection function. Need to work with the UART_AutoBaudRate_Master sample code.
<b>UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. Need to work with the UART_Autoflow_Slave sample code.
<b>UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. Need to work with the UART_Autoflow_Master sample code.
<b>UART_IrDA_Master</b>	Transmit and receive data in UART IrDA mode. Need to work with the UART_IrDA_Slave sample code.
<b>UART_IrDA_Slave</b>	Transmit and receive data in UART IrDA mode. Need to work with the UART_IrDA_Master sample code.
<b>UART_LIN</b>	Transmit LIN frame including header and response in LIN function mode.
<b>UART_PDMA</b>	Transmit and receive UART data with PDMA.

<b>UART_RS485_Master</b>	Transmit and receive data in RS485 function mode. Need to work with the UART_RS485_Slave sample code.
<b>UART_RS485_Slave</b>	Transmit and receive data in RS485 function mode. Need to work with the UART_RS485_Master sample code.
<b>UART_TxRx_Function</b>	Demonstrate how UART transmits and receives data from PC terminal through a RS232 interface.
<b>UART_Wakeup</b>	Wake up system form Power-down mode by nCTS / data interrupt.
<b>UART_Wakeup_LXT</b>	Wake up system form Power-down mode by Rx threshold and time-out / RS485 interrupt. This sample requires LXT on board.
<b>USCI_I2C_EEPROM</b>	Access EEPROM by an I <sup>2</sup> C interface.
<b>USCI_I2C_GCMode_Master</b>	I <sup>2</sup> C master uses I <sup>2</sup> C address 0x00 to write data to I <sup>2</sup> C slave. Need to work with the USCI_I2C_GCMode_Slave sample code.
<b>USCI_I2C_GCMode_Master</b>	I <sup>2</sup> C slave receives master data in General Call mode. Need to work with the USCI_I2C_GCMode_Master sample code.
<b>USCI_I2C_Master</b>	Demonstrate how an I <sup>2</sup> C master accesses slave. Need to work with the USCI_I2C_Slave sample code.
<b>USCI_I2C_Master_10bit</b>	Demonstrate how an I <sup>2</sup> C master accesses 10-bit address slave. Need to work with the USCI_I2C_Slave_10bit sample code.
<b>USCI_I2C_Monitor</b>	Use UI2C to monitor and log I2C bus traffic.
<b>USCI_I2C_MultiBytes_Master</b>	Use UI2C multiple-byte function to read and write data to slave. Need to work with the USCI_I2C_Slave sample code.

<b>USCI_I2C_SingleByte_Master</b>	Use I2C single-byte function to read and write data to slave. Need to work with the USCI_I2C_Slave sample code.
<b>USCI_I2C_Slave</b>	Set USCI-I2C in slave mode to receive the data of master. Need to work with the USCI_I2C_Master, USCI_I2C_MultiBytes_Master or USCI_I2C_SingleByte_Master sample codes.
<b>USCI_I2C_Slave_10bit</b>	Show a 10-bit address slave how to receive data from master. Need to work with the USCI_I2C_Master_10bit sample code.
<b>USCI_I2C_Wakeup_Master</b>	Show how to wake up I2C slave from Power-down mode. Need to work with the USCI_I2C_Wakeup_Slave sample code.
<b>USCI_I2C_Wakeup_Slave</b>	Show how to wake up system in Power-down mode through an I2C interface. Need to work with the USCI_I2C_Wakeup_Master sample code.
<b>USCI_SPI_MasterMode</b>	Communicate with an off-chip SPI slave device. Need to work with the USCI_SPI_SlaveMode sample code.
<b>USCI_SPI_PDMA_LoopTest</b>	USCI-SPI loop back test with PDMA.
<b>USCI_SPI_SlaveMode</b>	Communicate with an off-chip SPI master device. Need to work with the USCI_SPI_MasterMode sample code.
<b>USCI_UART_AutoBaudRate_Master</b>	Show how to use auto baud rate detection function. Need to work with the USCI_UART_AutoBaudRate_Slave sample code.
<b>USCI_UART_AutoBaudRate_Slave</b>	Show how to use auto baud rate detection function. Need to work with the UACI_UART_AutoBaudRate_Master sample code.
<b>USCI_UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. Need to work with the

	USCI_UART_Autoflow_Slave sample code.
<b>USCI_UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. Need to work with the USCI_UART_Autoflow_Master sample code.
<b>USCI_UART_PDMA</b>	Transmit and receive UART data with the PDMA.
<b>USCI_UART_RS485_Master</b>	Transmit and receive data in RS485 function mode. Need to work with the USCI_UART_RS485_Slave sample code.
<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 function mode. Need to work with the USCI_UART_RS485_Master sample code.
<b>USCI_UART_TxRx_Function</b>	Demonstrate how USCI-UART transmits and receives data from PC terminal through a RS232 interface.
<b>USCI_UART_Wakeup</b>	Wake up system form Power-down mode by nCTS / data interrupt.
<b>WDT_PowerDown</b>	Use WDT time-out interrupt event to wake up system.
<b>WDT_TimeoutINT</b>	Select one WDT time-out interval period time to generate time-out interrupt event.
<b>WDT_TimeoutReset</b>	Cause WDT time-out reset system event while WDT time-out reset delay period expired.
<b>WWDT_CompareINT</b>	Select one WWDT window compare value to generate window compare match interrupt event.

## 7 \SampleCode\StdDriver

<b>ADC_BurstMode</b>	Use burst mode to finish more than 16 ADC conversions for the specified channel.
<b>ADC_ContinuousScanMode</b>	Use continuous scan mode to finish two cycles of ADC conversion for the specified channels.
<b>ADC_MeasureAVDD</b>	Use band-gap voltage to calculate AVDD voltage.
<b>ADC_PDMA_SingleCycleScanMode</b>	Use single-cycle scan mode to finish one cycle of ADC conversion for 4 specified channels and transfer conversion result by PDMA.
<b>ADC_PWMTrigger</b>	Trigger single mode ADC by PWM0 channel0.
<b>ADC_ResultMonitor</b>	Use digital compare function to monitor the ADC conversion result of channel2.
<b>ADC_SingleCycleScanMode</b>	Use single-cycle scan mode to finish one cycle of ADC conversion for 4 specified channels.
<b>ADC_SingleMode</b>	Use single mode to finish the ADC conversion of the specified channel.
<b>BPWM_Capture</b>	Use BPWM1 channel2 to capture BPWM0 channel0 waveform.
<b>BPWM_DoubleBuffer_PeriodLoadingMode</b>	Use BPWM0 channel0~5 and double buffering function to output waveforms. This sample code changes duty cycle of each channel and the period shared by 6 channels.
<b>BPWM_DutySwitch</b>	Change duty cycle of BPWM output waveform by configured period.
<b>BPWM_OutputWaveform</b>	Use BPWM counter to output waveform.
<b>BPWM_SyncStart</b>	Use BPWM counter synchronous start function.

<b>CLK_ClockDetector</b>	Show how to use clock fail detector for HXT / LXT and clock frequency monitor for HXT.
<b>CLK_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLK pin.
<b>FMC_ExecInSRAM</b>	Run code in SRAM by using scatter loading description file.
<b>FMC_IAP</b>	Switch both codes between LDROM and APROM.
<b>FMC_RW</b>	Read and program embedded flash by ISP function.
<b>FMC_SPROM</b>	Show how to change SPROM to secured mode to protect SPROM contents.
<b>GPIO_EINTAndDebounce</b>	Use GPIO external interrupt function.
<b>GPIO_INT</b>	Use GPIO interrupt function.
<b>GPIO_OutputInput</b>	Set GPIO pin mode and use pin data input / output control.
<b>GPIO_PowerDown</b>	Wake up form Power-down mode by GPIO interrupt.
<b>I2C_EEPROM</b>	Access EEPROM by an I <sup>2</sup> C interface.
<b>I2C_GCMode_Master</b>	I <sup>2</sup> C master uses I <sup>2</sup> C address 0x00 to write data to slave. Need to work with I2C_GCMode_Slave sample code.
<b>I2C_GCMode_Slave</b>	I <sup>2</sup> C slave receives master data in General Call mode. Need to work with I2C_GCMode_Master sample code.
<b>I2C_Master</b>	Demonstrate how an I <sup>2</sup> C master accesses slave. Need to work with the I2C_Slave sample code.
<b>I2C_MultiBytes_Master</b>	Use I <sup>2</sup> C multiple-byte API to read and write data to slave. Need to work with the I2C_Slave 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>	Use I <sup>2</sup> C single-byte API to read and write data to slave. Need to work with the I2C_Slave sample code.
<b>I2C_Slave</b>	Set I <sup>2</sup> C in slave mode to receive the data of master. Need to work with the I2C_Master, I2C_MultiBytes_Master or I2C_SingleByte_Master sample codes.
<b>I2C_Wakeup_Master</b>	Show how to wake up I <sup>2</sup> C slave from Power-down mode. Need to work with the I2C_Wakeup_Slave sample code.
<b>I2C_Wakeup_Slave</b>	Show how to wake up system in Power-down mode through an I <sup>2</sup> C interface. Need to work with the I2C_Wakeup_Master sample code.
<b>I2S_Master</b>	Demonstrate how I <sup>2</sup> S works in master mode. Need to work with the I2S_Slave sample code.
<b>I2S_PDMA_NAU8822</b>	I <sup>2</sup> S playback and record with NAU8822 codec by PDMA with scatter-gather mode
<b>I2S_PDMA_Play</b>	I <sup>2</sup> S playback by PDMA with scatter-gather mode
<b>I2S_PDMA_PlayRecord</b>	I <sup>2</sup> S playback and record by PDMA with scatter-gather mode.
<b>I2S_PDMA_Record</b>	I <sup>2</sup> S record by PDMA with scatter-gather mode.
<b>I2S_Slave</b>	Demonstrate how I <sup>2</sup> S works in slave mode. Need to work with the I2S_Master sample code.
<b>PDMA</b>	Use PDMA channel0 to transfer data from memory to memory.
<b>PDMA_Scatter_Gather</b>	Use PDMA channel4 to transfer data from memory to memory with scatter-gather mode.

<b>PDMA_ScatterGather_PingPongBuffer</b>	Use PDMA to implement ping-pong buffer from memory to memory with scatter-gather mode.
<b>PWM_Capture</b>	Use PWM1 channel2 to capture PWM1 channel0 waveform.
<b>PWM_DeadZone</b>	Use PWM dead zone function.
<b>PWM_DoubleBuffer_PeriodLoadingMode</b>	Use PWM0 channel0~1 and double buffering function to output waveforms. This sample code changes duty cycle of each channel and the period shared by 2 channels.
<b>PWM_DutySwitch</b>	Change duty cycle of PWM output waveform by configured period.
<b>PWM_OutputWaveform</b>	Use PWM counter to output waveform.
<b>PWM_PDMA_Capture</b>	Capture PWM1 channel0 waveform by PWM1 channel2, and use PDMA to transfer captured data.
<b>PWM_SyncStart</b>	Use PWM counter synchronous start function.
<b>SPI_MasterFIFOmode</b>	Communicate with an off-chip SPI slave device with FIFO mode. Need to work with the SPI_SlaveFIFOmode sample code.
<b>SPI_PDMA_LoopTest</b>	SPI loopback test with PDMA.
<b>SPI_SlaveFIFOmode</b>	Communicate with an off-chip SPI master device with FIFO mode. Need to work with the SPI_MasterFIFOmode sample code.
<b>SYS_BODWakeup</b>	Wake up system from BOD event.
<b>TIMER_CaptureCounter</b>	Timer2 uses Timer3 toggle output to capture an event counter value from Timer0 toggle output.
<b>TIMER_Delay</b>	Use timer to implement delay cycle.
<b>TIMER_EventCounter</b>	Use Timer1 counter input function to count the input event.

<b>TIMER_PeriodicINT</b>	Perform timer counting in periodic mode.
<b>TIMER_TimeoutWakeup</b>	Use Timer0 toggle-output interrupt event to wake up system.
<b>UART_AutoBaudRate_Master</b>	Show how to use auto baud rate detection function. Need to work with the UART_AutoBaudRate_Slave sample code.
<b>UART_AutoBaudRate_Slave</b>	Show how to use auto baud rate detection function. Need to work with the UART_AutoBaudRate_Master sample code.
<b>UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. Need to work with the UART_Autoflow_Slave sample code.
<b>UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. Need to work with the UART_Autoflow_Master sample code.
<b>UART_IrDA_Master</b>	Transmit and receive data in UART IrDA mode. Need to work with the UART_IrDA_Slave sample code.
<b>UART_IrDA_Slave</b>	Transmit and receive data in UART IrDA mode. Need to work with the UART_IrDA_Master sample code.
<b>UART_LIN</b>	Transmit LIN frame including header and response in LIN function mode.
<b>UART_PDMA</b>	Transmit and receive UART data with PDMA.
<b>UART_RS485_Master</b>	Transmit and receive data in RS485 function mode. Need to work with the UART_RS485_Slave sample code.
<b>UART_RS485_Slave</b>	Transmit and receive data in RS485 function mode. Need to work with the UART_RS485_Master sample code.
<b>UART_TxRx_Function</b>	Demonstrate how UART transmits and receives data from PC terminal through a RS232 interface.

<b>UART_Wakeup</b>	Wake up system form Power-down mode by nCTS / data interrupt.
<b>UART_Wakeup_LXT</b>	Wake up system form Power-down mode by Rx threshold and time-out / RS485 interrupt. This sample requires LXT on board.
<b>USBD_Audio_HID_NAU8822</b>	USB audio class device with HID key. 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_Audio_HID_Transfer</b>	USB composite device with USB audio class and HID data transfer function.
<b>USBD_Audio_NAU8822</b>	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>	USB keyboard device. It supports to use GPIO to simulate key input.
<b>USBD_HID_Mouse</b>	USB mouse device. The mouse cursor will move automatically when this mouse device connecting to PC by USB.
<b>USBD_HID_Mouse2</b>	USB mouse device. It uses PC.0~5 to control mouse direction and mouse key. It also supports USB suspend and remote wakeup.
<b>USBD_HID_MouseKeyboard</b>	A USB mouse 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 pin to simulate key input.
<b>USBD_HID_Transfer</b>	Transfer data between USB device and PC through HID interface. A windows tool is also included in this sample code to connect with USB device.
<b>USBD_HID_Transfer_and_Keyboard</b>	USB composite device with HID Transfer and keyboard. Transfer data between USB

	device and PC through HID interface. A windows tool is also included in this sample code to connect with USB device.
<b>USBD_HID_Transfer_and_MSC</b>	USB composite device with HID Transfer and USB Mass-storage. Transfer data between USB device and PC through HID interface. A windows tool is also included in this sample code to connect with a USB device.
<b>USBD_MassStorage_CDROM</b>	Simulate a USB CD-ROM device.
<b>USBD_MassStorage_DataFlash</b>	Use internal flash as backend storage media to simulate a USB pen drive. GCC project does not support small size flash due to larger code size.
<b>USBD_Micro_Printer</b>	USB micro printer device.
<b>USBD_Printer_and_HID_Transfer</b>	USB composite device with USB micro printer device and HID Transfer. Transfer data between USB device and PC through 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>	USB composite device with virtual COM port and keyboard.
<b>USBD_VCOM_and_HID_Transfer</b>	USB composite device with virtual COM port and HID Transfer. It supports one virtual COM port and transfers 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_MassStorage</b>	USB composite device with virtual COM port and USB Mass-Storage.
<b>USBD_VCOM_DualPort</b>	USB virtual COM port device. It supports dual virtual COM ports using UART and UART.
<b>USBD_VCOM_SinglePort</b>	USB virtual COM port device. It supports single virtual COM port.

<b>USCI_I2C_EEPROM</b>	Access EEPROM by an I <sup>2</sup> C interface.
<b>USCI_I2C_GCMode_Master</b>	I <sup>2</sup> C master uses I <sup>2</sup> C address 0x00 to write data to I <sup>2</sup> C slave. Need to work with the USCI_I2C_GCMode_Slave sample code.
<b>USCI_I2C_GCMode_Master</b>	I <sup>2</sup> C slave receives master data in General Call mode. Need to work with the USCI_I2C_GCMode_Master sample code.
<b>USCI_I2C_Master</b>	Demonstrate how an I <sup>2</sup> C master accesses slave. Need to work with the USCI_I2C_Slave sample code.
<b>USCI_I2C_Master_10bit</b>	Demonstrate how an I <sup>2</sup> C master accesses 10-bit address slave. Need to work with the USCI_I2C_Slave_10bit sample code.
<b>USCI_I2C_Monitor</b>	Use UI2C to monitor and log I2C bus traffic.
<b>USCI_I2C_MultiBytes_Master</b>	Use UI2C multiple-byte API to read and write data to slave. Need to work with the USCI_I2C_Slave sample code.
<b>USCI_I2C_SingleByte_Master</b>	Use UI2C single-byte API to read and write data to slave. Need to work with the USCI_I2C_Slave sample code.
<b>USCI_I2C_Slave</b>	Set USCI-I <sup>2</sup> C in slave mode to receive the data of master. Need to work with the USCI_I2C_Master, USCI_I2C_MultiBytes_Master or USCI_I2C_SingleByte_Master sample codes.
<b>USCI_I2C_Slave_10bit</b>	Show a 10-bit address slave how to receive data from master. Need to work with the USCI_I2C_Master_10bit sample code.
<b>USCI_I2C_Wakeup_Master</b>	Show how to wake up I <sup>2</sup> C slave from Power-down mode. Need to work with the USCI_I2C_Wakeup_Slave sample code.
<b>USCI_I2C_Wakeup_Slave</b>	Show how to wake up system in Power-down mode through an I <sup>2</sup> C interface. Need

	to work with the USCI_I2C_Wakeup_Master sample code.
<b>USCI_SPI_MasterMode</b>	Communicate with an off-chip SPI slave device. Need to work with the USCI_SPI_SlaveMode sample code.
<b>USCI_SPI_PDMA_LoopTest</b>	USCI-SPI loop back test with PDMA.
<b>USCI_SPI_SlaveMode</b>	Communicate with an off-chip SPI master device. Need to work with the USCI_SPI_MasterMode sample code.
<b>USCI_SPI_SlaveModeINT</b>	Communicate with an off-chip SPI master device, transmit and receive data in the interrupt handler. This sample code needs to work with USCI_SPI_MasterMode sample code.
<b>USCI_UART_AutoBaudRate_Master</b>	Show how to use auto baud rate detection function. Need to work with the USCI_UART_AutoBaudRate_Slave sample code.
<b>USCI_UART_AutoBaudRate_Slave</b>	Show how to use auto baud rate detection function. Need to work with the UACI_UART_AutoBaudRate_Master sample code.
<b>USCI_UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. Need to work with the USCI_UART_Autoflow_Slave sample code.
<b>USCI_UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. Need to work with the USCI_UART_Autoflow_Master sample code.
<b>USCI_UART_PDMA</b>	Transmit and receive UART data with PDMA.
<b>USCI_UART_RS485_Master</b>	Transmit and receive data in RS485 function mode. Need to work with the USCI_UART_RS485_Slave sample code.

<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 function mode. Need to work with the USCI_UART_RS485_Master sample code.
<b>USCI_UART_TxRx_Function</b>	Demonstrate how USCI-UART transmits and receives data from PC terminal through a RS232 interface.
<b>USCI_UART_Wakeup</b>	Wake up system form Power-down mode by nCTS / data interrupt.
<b>WDT_PowerDown</b>	Use WDT time-out interrupt event to wake up system.
<b>WDT_TimeoutINT</b>	Select one WDT time-out interval period time to generate time-out interrupt event.
<b>WDT_TimeoutReset</b>	Cause WDT time-out reset system event while WDT time-out reset delay period expired.
<b>WWDT_CompareINT</b>	Select one WWDT window compare value to generate window compare match interrupt event.



### **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.*