NUC1263 Series CMSIS BSP Guide

Directory Introduction for 32-bit NuMicro™ Family

Directory Information

Please extract the “NUC1263Series_BSP_CMSIS_V3.00.002.zip” file firstly, and then put the “NUC1263Series_BSP_CMSIS_V3.00.002” folder into the working folder (e.g. .\Nuvoton\BSP Library\).

This BSP folder contents:

Document	Device driver reference manual and reversion history.
Library	Device driver header and source files.
SampleCode	Device driver sample code.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 .\Document\

CMSIS.html	Introduction of CMSIS version 5. CMSIS components included CMSIS-CORE, CMSIS-Driver, CMSIS-DSP, etc. ● CMSIS-CORE: API for the Cortex-M0 processor core and peripherals. ● CMSIS-Driver: Defines generic peripheral driver interfaces for middleware making it reusable across supported devices. ● 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).
Revision History.pdf	The revision history of NUC1263 Series BSP.
NuMicro NUC1263 Series Driver Reference Guide.chm	The usage of drivers in NUC1263 Series BSP.

2 .\Library\

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

3 .\Sample Code\

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

4 .\SampleCode\ISP

ISP_DFU	In-System-Programming Sample code through USB interface and following Device Firmware Upgrade Class Specification.
ISP_HID	In-System-Programming Sample code through USB HID interface.
ISP_I2C	In-System-Programming Sample code through I2C interface.
ISP_RS485	In-System-Programming Sample code through RS485 interface.
ISP_SPI	In-System-Programming Sample code through SPI interface.
ISP_UART	In-System-Programming Sample code through UART interface.

5 .\SampleCode\RegBased

ACMP	Demonstrate how ACMP works with internal band-gap voltage.
ACMP_Wakeup	Show how to wake up MCU from Power-down mode by ACMP wake-up function.
ADC_BandGap	Convert Band-gap (channel 29) and print conversion result.
ADC_ContinuousScanMode	Perform A/D Conversion with ADC continuous scan mode.
ADC_PDMA_SingleCycleScanMode	Perform A/D Conversion with ADC single cycle scan mode and transfer result by PDMA.
ADC_PwmTrigger	Demonstrate how to trigger ADC by PWM.
ADC_ResultMonitor	Monitor the conversion result of Channel 2 by the digital compare function.
ADC_SingleCycleScanMode	Perform A/D Conversion with ADC single cycle scan mode.
ADC_SingleMode	Perform A/D Conversion with ADC single mode.
BPWM_Capture	Use BPWM0 channel 0 to capture the BPWM1 channel 0 waveform.
BPWM_DoubleBuffer_PeriodLoadingMode	Change duty cycle and period of output waveform by BPWM Double Buffer function(Period loading mode).
BPWM_DutySwitch	Change duty cycle of output waveform by configured period.
BPWM_OutputWaveform	Demonstrate how to use BPWM output waveform.
BPWM_SyncStart	Demonstrate how to use BPWM counter synchronous start function.
CLK_ClockDetector	Show the usage of clock fail detector and clock frequency monitor function.
CRC_CCITT	Implement CRC in CRC-CCITT mode and get CRC checksum results.
CRC_CRC8	Implement CRC in CRC-8 mode and get CRC checksum results.

CRC_CRC32	Implement CRC in CRC-32 mode with PDMA transfer.
DAC_ExtPinTrigger	Demonstrate how to trigger DAC conversion by external pin.
DAC_PDMA_TimerTrigger	Demonstrate how to PDMA and trigger DAC by Timer.
DAC_SoftwareTrigger	Write a data to DAC_DAT register to trigger DAC conversion.
DAC_TimerTrigger	Use a Timer to trigger DAC Conversion.
FMC_ExeInSRAM	Implement a code and execute the code in SRAM to program embedded Flash (support KEIL MDK only).
FMC_IAP	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.
FMC_MultiBoot	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.
FMC_RW	Demonstrate how to read/program embedded Flash by ISP function.
GPIO_EINTAndDebounce	Show the usage of GPIO external interrupt function and de-bounce function.
GPIO_INT	Show the usage of GPIO interrupt function.
GPIO_OutputInput	Show how to set GPIO pin mode and use pin data input/output control.
GPIO_PowerDown	Show how to wake up system from Power-down mode by GPIO interrupt.
I2C_EEPROM	Demonstrate how to access EEPROM through a I2C interface
I2C_GCMode_Master	Demonstrate how a Master uses I2C address 0x0 to write data to I2C Slave. This sample code needs to work with I2C_GCMode_Slave.
I2C_GCMode_Slave	Demonstrate how to receive Master data in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.
I2C_Loopback	Demonstrate how a Master access a Slave.

I2C_Loopback_10bit	Demonstrate how a Master uses 10-bit addressing to access a Slave.
I2C_Master	Demonstrate how a Master accesses a Slave. This sample code needs to work with I2C_Slave.
I2C_Master_PDMA	Demonstrate how a Master accesses Slave using PDMA TX mode and PDMA RX mode.
I2C_Slave	Demonstrate how to set I2C in slave mode to receive data from a Master. This sample code needs to work with I2C_Master.
I2C_Slave_PDMA	Demonstrate how a Slave uses PDMA Rx mode to receive data from a Master.
I2C_SMBUS	Demonstrate how I2C SMBUS works.
I2C_Wakeup_Slave	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.
I2S_Master	Configure SPI1 as I2S Master mode and demonstrate how I2S works in Master mode. This sample code needs to work with I2S_Slave.
I2S_PDMA_NAU8822	This is an I2S demo with PDMA function connected with NAU8822 codec.
I2S_PDMA_Play	This is an I2S demo for playing data and demonstrating how I2S works with PDMA.
I2S_PDMA_PlayRecord	This is an I2S demo for playing and recording data with PDMA function.
I2S_PDMA_Record	This is an I2S demo for recording data and demonstrating how I2S works with PDMA.
I2S_Slave	Configure SPI1 as I2S Slave mode and demonstrate how I2S works in Slave mode. This sample code needs to work with I2S_Master.
LLSI_Marquee	This is a LLSI demo for marquee display in software mode. It needs to be used with WS2812 LED strip.
LLSI_PDMA_Marquee	This is a LLSI demo for marquee display in PDMA mode. It needs to be used with WS2812 LED strip.

PDMA	Use PDMA Channel 2 to transfer data from memory to memory.
PDMA_ScatterGather_PingPongBuffer	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).
PDMA_Scatter_Gather	Use PDMA Channel 4 to transfer data from memory to memory by scatter-gather mode.
SPI_Loopback	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.
SPI_MasterFifoMode	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.
SPI_Mux	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.
SPI_PDMA_LoopTest	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.
SPI_SlaveFifoMode	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.
SYS_BODWakeup	Show how to wake up system form Power-down mode by brown-out detector interrupt.
SYS_PLLClockOutput	Change system clock to different PLL frequency and output system clock from CLKO pin.
SYS_PowerDown_MinCurrent	Demonstrate how to minimize power consumption when entering power down mode.
SYS_TrimHIRC	Demonstrate how to use LXT to trim HIRC.
SYS_VoltageDetector	Show how to use voltage detector to detect pin input voltage.
TIMER_CaptureCounter	Show how to use the timer2 capture function to capture

	timer2 counter value.
TIMER_EventCounter	Show how to use the timer2 capture function to capture timer2 counter value.
TIMER_PeriodicINT	Implement timer counting in periodic mode.
TIMER_TimeoutWakeup	Use timer0 periodic time-out interrupt event to wake up system.
TS_TemperatureMeasure	Measure the current temperature by Temperature Sensor.
UART_AutoBaudRate	Show how to use auto baud rate detection function.
UART_Autoflow	Transmit and receive data with auto flow control.
UART_IrDA	Transmit and receive data in UART IrDA mode.
UART_PDMA	Transmit and receive UART data with PDMA.
UART_RS485	Transmit and receive data in UART RS485 mode.
UART_TxRxFunction	Transmit and receive data from PC terminal through RS232 interface.
UART_Wakeup	Show how to wake up system from Power-down mode by UART interrupt.
WDT_TimeoutWakeupAndReset	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.

6 .\SampleCode\StdDriver

ACMP_CompareDAC	Demonstrate how ACMP compare DAC output with ACMP1_P1 value.
ACMP_CompareVBG	Demonstrate how ACMP compare VBG output with ACMP1_P1 value.
ACMP_Wakeup	Show how to wake up MCU from Power-down mode by ACMP wake-up function.
ACMP_WindowCompare	Demonstrate the usage of ACMP window compare function.
ADC_BandGap	Convert Band-gap (channel 29) and print conversion result.
ADC_ContinuousScanMode	Perform A/D Conversion with ADC continuous scan mode.
ADC_PDMA_SingleCycleScanMode	Perform A/D Conversion with ADC single cycle scan mode and transfer result by PDMA.
ADC_PwmTrigger	Demonstrate how to trigger ADC by PWM.
ADC_ResultMonitor	Monitor the conversion result of Channel 2 by the digital compare function.
ADC_SingleCycleScanMode	Perform A/D Conversion with ADC single cycle scan mode.
ADC_SingleMode	Perform A/D Conversion with ADC single mode.
BPWM_Capture	Use BPWM0 channel 0 to capture the BPWM1 channel 0 waveform.
BPWM_DoubleBuffer_PeriodLoadingMode	Change duty cycle and period of output waveform by BPWM Double Buffer function.(Period loading mode).
BPWM_DutySwitch	Change duty cycle of output waveform by configured period.
BPWM_OutputWaveform	Demonstrate how to use BPWM counter output waveform.
BPWM_SyncStart	Demonstrate how to use BPWM counter synchronous start function.
CLK_ClockDetector	Show the usage of clock fail detector and clock frequency monitor function.
CRC_CCITT	Implement CRC in CRC-CCITT mode and get the CRC

	checksum result.
CRC_CRC32	Implement CRC in CRC-32 mode with PDMA transfer.
CRC_CRC8	Implement CRC in CRC-8 mode and get the CRC checksum result.
DAC_ExtPinTrigger	Demonstrate how to trigger DAC conversion by external pin.
DAC_PDMA_TimerTrigger	Demonstrate how to PDMA and trigger DAC by Timer.
DAC_SoftwareTrigger	Write a data to DAC_DAT register to trigger DAC conversion.
DAC_TimerTrigger	Use a Timer to trigger DAC Conversion.
FMC_ExeInSRAM	Implement a code and execute in SRAM to program embedded Flash (support KEIL MDK only).
FMC_IAP	Show how to set VECMAP to LDROM and reboot to LDROM from APROM.
FMC_RW	Demonstrate how to read/program embedded Flash by ISP function.
GPIO_EINTAndDebounce	Show the usage of GPIO external interrupt function and de-bounce function.
GPIO_INT	Show the usage of GPIO interrupt function.
GPIO_OutputInput	Show how to set GPIO pin mode and use pin data input/output control.
GPIO_PowerDown	Show how to wake up system from Power-down mode by GPIO interrupt.
I2C_EEPROM	Demonstrate how to access EEPROM through a I2C interface
I2C_GCMode	Show how a Master uses I2C address 0x0 to write data to Slave.This sample code needs to work with I2C_GCMode_Slave.
I2C_Loopback	Demonstrate how a Master access Slave.
I2C_Loopback_10bit	Demonstrate how a Master uses 10-bit addressing to access a Slave.

I2C_Master	Demonstrate how a Master accesses Slave. This sample code needs to work with I2C_Slave.
I2C_Master_PDMA	Demonstrate how a Master accesses Slave using PDMA TX mode and PDMA RX mode.
I2C_MultiBytes_Master	Show how to set I2C use Multi bytes API Read and Write data to Slave. Needs to work with I2C_Slave sample code
I2C_SingleByte_Master	Show how to use I2C Signle byte API Read and Write data to Slave. Needs to work with I2C_Slave sample code.
I2C_Slave	Demonstrate how to set I2C in slave mode to receive the data from a Master. This sample code needs to work with I2C_Master.
I2C_Slave_PDMA	Demonstrate how a Slave uses PDMA Rx mode receive data from a Master.
I2C_SMBUS	Demonstrate how I2C SMBUS works.
I2C_Wakeup_Slave	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.
I2S_Master	Configure SPI1 as I2S Master mode and demonstrate how I2S works in Master mode. This sample code needs to work with I2S_Slave.
I2S_PDMA_NAU8822	This is an I2S demo with PDMA function connected with NAU8822 codec.
I2S_PDMA_Play	This is an I2S demo for playing data and demonstrating how I2S works with PDMA.
I2S_PDMA_PlayRecord	This is an I2S demo for playing and recording data with PDMA function.
I2S_PDMA_Record	This is an I2S demo for recording data and demonstrating how I2S works with PDMA.
I2S_Slave	Configure SPI1 as I2S Slave mode and demonstrate how I2S works in Slave mode. This sample code needs to work with I2S_Master.
I3CS_Slave_HotJoin	Demonstrate how to initiate a Hot-Join request to an I3C Master.

I3CS_Slave_IBI	Demonstrate how to initiate an In-Band Interrupt request to an I3C Master.
I3CS_Slave_Wakeup	Wake up the MCU from Power-down mode after receiving read/write request from the Master.
I3CS_SlaveRW	Demonstrate how to use I3C0 Slave to reveive and transmit the data from a Master.
I3CS_SlaveRW_PDMA	Demonstrate how to use I3C0 Slave to reveive and transmit the data through PDMA from a Master.
LLSI_Marquee	This is a LLSI demo for marquee display in software mode. It needs to be used with WS2812 LED strip.
LLSI_PDMA_Marquee	This is a LLSI demo for marquee display in PDMA mode. It needs to be used with WS2812 LED strip.
PDMA	Use PDMA Channel 2 to transfer data from memory to memory.
PDMA_ScatterGather_PingPongBuffer	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).
PDMA_Scatter_Gather	Use PDMA Channel 4 to transfer data from memory to memory by scatter-gather mode.
SPDH_LLSIDevice	LLSI device firmware.
SPI_Loopback	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.
SPI_MasterFifoMode	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.
SPI_Mux	Configure SPI1 as SPI Master1 and demonstrate how to access SPI flash through SPI_MUX interface. This sample code needs to work ith SPI_Flash_Master2 sample code.
SPI_PDMA_LoopTest	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.

SPI_SlaveFifoMode	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.
SYS_BODWakeup	Show how to wake up system form Power-down mode by brown-out detector interrupt.
SYS_PLLClockOutput	Change system clock to different PLL frequency and output system clock from CLKO pin.
SYS_TrimHIRC	Demonstrate how to use LXT to trim HIRC.
SYS_VoltageDetector	Show how to use voltage detector to detect pin input voltage.
TIMER_CaptureCounter	Show how to use the timer2 capture function to capture timer2 counter value.
TIMER_Delay	Show how to use timer0 to create various delay time.
TIMER_EventCounter	Implement timer1 event counter function to count the external input event.
TIMER_InterTimerTriggerMode	Demonstrate how to use Inter-Timer trigger function.
TIMER_PeriodicINT	Implement timer counting in periodic mode.
TIMER_TimeoutWakeup	Use timer0 periodic time-out interrupt event to wake up system.
TIMER_ToggleOut	Implement timer counting in toggle-output mode.
TS_TemperatureMeasure	Measure the current temperature by Temperature Sensor.
UART_AutoBaudRate	Show how to use auto baud rate detection function.
UART_Autoflow	Transmit and receive data with auto flow control.
UART_IrDA	Transmit and receive data in UART IrDA mode.
UART_PDMA	Transmit and receive UART data with PDMA.
UART_RS485	Transmit and receive data in UART RS485 mode.
UART_TxRxFunction	Transmit and receive data from PC terminal through RS232 interface.

UART_Wakeup	Show how to wake up system from Power-down mode by UART interrupt.
USBD_Audio_NAU8822	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.
USBD_HID_Keyboard	Show how to implement a USB keyboard device. This sample code supports to use GPIO to simulate key input.
USBD_HID_Mouse	Show how to implement a USB mouse device. The mouse cursor will move automatically when this mouse device connecting to PC by USB.
USBD_HID_Mouse_BC12	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.
USBD_HID_Mouse2	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.
USBD_HID_MouseKeyboard	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.
USBD_HID_Transfer	Transfer data between USB device and PC through USB HID interface. A windows tool is also included in this sample code to connect with USB device.
USBD_HID_Transfer_and_Keyboard	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.
USBD_HID_Transfer_and_MSC	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.
USBD_MassStorage_CDRO	Demonstrate how to simulate a USB CD-ROM device.

M	
USBD_MassStorage_DataFlash	Demonstrate how to implement a USB mass storage. It uses embedded Data Flash as storage.
USBD_MassStorage_SDCard	Use SD card as storage to implement a USB mass storage device.
USBD_Micro_Printer	Show how to implement a USB micro printer device.
USBD_Printer_and_HID_Transfer	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.
USBD_VCOM_and_HID_Keyboard	Implement a USB composite device with virtual COM port and keyboard functions.
USBD_VCOM_and_HID_Transfer	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.
USBD_VCOM_and_MassStorage	Implement a USB composite device. It supports one virtual COM port and one USB mass storage device.
USBD_VCOM_DualPort	Demonstrate how to implement a USB dual virtual COM port device.
USBD_VCOM_SinglePort	Implement a USB virtual COM port device. It supports one virtual COM port.
WDT_TimeoutWakeupAndReset	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.

7 REVISION HISTORY

Date	Revision	Description
2023.06.15	3.00.002	1. Support block write mode in SPDH_LLSIDevice. 2. Add spdh_device library for HUB Device application. 3. Modify I2C_Master sample code to support 10 bit mode by compiler option. 4. Change the verification conditions for read and write operations in I2C_Master to check each address group. 5. Add MR register settings file for LLSI device. 6. Modify VendorName of Nu_DFU.inf. 7. Sync code with internal Hub with lighting device sample code. 8. Add Double Flashing Mode on SPDH_LLSIDevice sample. 9. Add MR55 register to adjust LED speed on SPDH_LLSIDevice sample. 10. Revise I3CS driver. 11. Modify UART_Read() to return received data count when time-out. 12. Update SPI_Loopback sample code. 13. Update SYS_PowerDown_MinCurrent sample code. 14. Remove the PHY type setting of USBD driver in Library/StdDriver/src/usbd.c 15. Modify SYS_UnLockReg() time-out handler in Library/StdDriver/inc/sys.h
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.