# Nano103 CMSIS BSP Directory

Directory Introduction for 32-bit NuMicro™ Family

## Directory Information

| | |
|---|---|
| **Document** | Driver reference manual and revision history. |
| **Library** | Driver header and source files. |
| **SampleCode** | Driver sample code. |



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

www.nuvoton.com

# 1 Document Information

| | |
|---|---|
| **CMSIS.html** | Document of CMSIS version 4.5.0 |
| **NuMicro Nano103 CMSIS BSP Revision History.pdf** | This document shows the revision history of Nano103 BSP. |
| **NuMicro Nano103 Driver Reference Guide.chm** | This document describes the usage of drivers in Nano103 BSP. |

# 2 Library Information

| | |
|---|---|
| **CMSIS** | Cortex® Microcontroller Software Interface Standard (CMSIS) V4.5.0 definitions by ARM® Corp. |
| **Device** | CMSIS compliant device header file. |
| **SmartcardLib** | Library for accessing a smartcard. |
| **StdDriver** | All peripheral driver header and source files. |

# 3 Sample Code Information

| | |
|---|---|
| **Hard_Fault_Sample** | Show hard fault information when hard fault happened. |
| **NuTiny-EVB-NANO103** | Sample code for Nano103 Tiny Board |
| **PowerDown_Chk** | Sample code which implements a function to test system state before entering power-down mode. If a system consumes more power than expected in power-down mode, this function can be used to check if there is any system setting that may cause power leakage. |
| **Semihost** | Show how to print and get character with IDE console window. |
| **StdDriver** | Demonstrate the usage of Nano103 MCU peripheral driver APIs. |
| **Template** | A project template for Nano103 MCU. |

# 4 \SampleCode\NuTiny-EVB-NANO103

| | |
|---|---|
| **LED** | Toggle PB.14 to turn on / off the board LED. |
| **RTC_PowerDown** | Demonstrate how to wake up system periodically with RTC interrupt. |
| **SYS_OperatingCurrent_HIRC** | Demonstrate how to minimize operating current while HCLK is from HIRC. |
| **SYS_OperatingCurrent_MIRC** | Demonstrate how to minimize operating current while HCLK is from MIRC. |

# 5 \SampleCode\StdDriver

| | |
|---|---|
| **ACMP** | Demonstrate Analog comparator (ACMP) comparison by comparing CMP0_P with Band-gap voltage and shows the result on UART console. |
| **ADC_Compare** | Demonstrate ADC conversion and comparison function by monitoring the conversion result of channel 0. |
| **ADC_ContinuousScan** | Convert ADC channel 0, 1, 2 in Continuous Scan mode and print conversion results. |
| **ADC_PDMA** | Use PDMA channel 1 to move ADC channel 0, 1, 2 converted data to SRAM |
| **ADC_PWMTrigger** | Configure PWM0 channel 0 to trigger ADC. |
| **ADC_Single** | Convert ADC channel 0 in Single mode and print conversion results. |
| **ADC_Single_BandGap** | Convert the band-gap voltage using an internal ADC channel. |
| **ADC_Single_TempSensor** | Convert temperature sensor voltage using an internal ADC channel. |
| **ADC_Single_VBat** | Convert the $V_{BAT}$ voltage using an internal ADC channel. |
| **ADC_SingleCycleScan** | Convert ADC channel 0, 1, 2 in Single Cycle Scan mode and print conversion results. |
| **ADC_TimerTrigger** | Configure Timer0 to trigger ADC and move converted data to SRAM using PDMA. |
| **CRC_CCITT** | Calculate the CRC-CCITT checksum value by CRC DMA mode. |
| **FMC_CRC32** | Show FMC CRC32 calculating capability. |
| **FMC_IAP** | Demonstrate IAP (In-Application Programming) function. To run this sample, the boot mode must be “Boot from APROM with IAP”. |
| **FMC_ReadAllOne** | Show FMC flash Read-All-One function. |

| | |
|---|---|
| **FMC_RW** | Show FMC read Flash IDs, erase, read, and write function. |
| **FMC_SecurityKey** | Show FMC security key function. |
| **GPIO_IOTest** | Use GPIO driver to control the GPIO pin direction and the high/low state, and show how to use GPIO interrupts. |
| **GPIO_PowerDown** | Demonstrate how to wake up system form Power-down mode by GPIO interrupt. |
| **I2C_EEPROM** | Read/write EEPROM via an I²C interface. |
| **I2C_Loopback** | An I²C master/slave demo by connecting I²C0 and I²C1 interface. |
| **I2C_Wakeup** | Demonstrate how to wake up system form Power-down mode by I²C interrupt. |
| **PDMA_Memory** | Use PDMA channel 2 to demonstrate memory to memory transfer. |
| **PWM_Capture** | Demonstrate PWM Capture function by using PWM0 channel 2 to capture the output of PWM0 channel 0. |
| **PWM_DeadZone** | Demonstrate the dead-zone feature with PWM0. |
| **RTC_Alarm_Test** | Demonstrate the RTC alarm function which sets an alarm 10 seconds after execution. |
| **RTC_Snoop_Detection** | Show how to use RTC snoop detect function. |
| **RTC_Time_Display** | Demonstrate the RTC function and display the current time to the UART console. |
| **SC_ReadATR** | Read the smartcard ATR from smartcard 0 interface. |
| **SC_ReadSimPhoneBook** | Demonstrate how to read phone book information in the SIM card. |
| **SCUART_TxRx** | Demonstrate smartcard UART mode by connecting PC.4 and PC.6 pins. |
| **SPI_FIFO_Flash** | Access SPI Flash using FIFO mode. |
| **SPI_LoopBack** | Demonstrate SPI loop back transfer |

| | |
|---|---|
| **SPI_TxRxLoopback_PDMA** | Demonstrate SPI loop back transfer with PDMA. |
| **SYS_CLKO** | Demonstrate how to output different clocks one after another to the same CLKO (PB.2) pin. |
| **SYS_Control** | Demonstrate how to change different PLL settings for the system clock source, and output system clock to CLKO (PB.2) pin with the system clock / 4 frequency. |
| **SYS_PLLClockOutput** | Change system clock to different PLL frequency and output system clock from CLKO pin. |
| **SYS_ScalableLDO** | Demonstrate maximum system operating frequency with different LDO settings. |
| **SYS_TrimIRC** | Demonstrate how to use LXT to trim HIRC. |
| **Timer_Delay** | Demonstrate the usage of TIMER_Delay() API to generate a 1 second delay. |
| **Timer_EventCounter** | Use the pin PB.8 to demonstrate timer event counter function. |
| **Timer_FreeCountingMode** | Use the timer pin PD.11 to demonstrate timer free counting mode function. Also display the measured input frequency to UART console. |
| **Timer_InterTimerTriggerMode** | Use the timer pin PB.8 to demonstrate inter timer trigger mode function. Also display the measured input frequency to UART console. |
| **Timer_Periodic** | Use the timer periodic mode to generate timer interrupt every 1 second. |
| **Timer_ToggleOut** | Demonstrate the timer 0 toggle out function on pin PB.8. |
| **Timer_TriggerCountingMode** | Use the timer pin PD.11 to demonstrate timer trigger counting mode function. And displays the measured input frequency to UART console. |
| **Timer_Wakeup** | Use timer to wake up system from Power-down mode periodically. |
| **UART_AutoBaudRate** | Demonstrate how to use auto baud rate detection function. |
| **UART_FlowCtrl** | Transmit and receive data using auto flow control. |

| | |
|---|---|
| **UART_IrDA** | Show how to transmit and receive UART data in UART IrDA mode. |
| **UART_LIN** | Demonstrate how to transmit LIN header and response. |
| **UART_PDMA** | Demonstrate UART transmit and receive function with PDMA. |
| **UART_RS485_Receive** | Demonstrate how to receive data in UART RS485 mode. |
| **UART_RS485_Transmit** | Demonstrate how to transmit data in UART RS485 mode. |
| **UART_Rx_Wakeup** | Demonstrate how to wake up system form Power-down mode by UART interrupt. |
| **UART_TxRx_Function** | Transmit and receive data from PC terminal through RS232 interface. |
| **WDT_Polling** | Use Polling mode to check WDT time-out state and reset WDT after time-out occurs. |
| **WDT_Wakeup** | Use WDT to wake up system from Power-down mode periodically. |
| **WWDT_Reload** | Demonstrate the WWDT counter reload function. |

## 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.**