# M2L31 Series CMSIS BSP Guide

Directory Introduction for 32-bit NuMicro® Family

## Directory Information

| | |
|---|---|
| **Document** | Driver reference guide and revision history. |
| **Library** | Driver header and source files. |
| **SampleCode** | Driver sample code. |
| **ThirdParty** | Libraries from third parties. |

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

www.nuvoton.com

# TABLE OF CONTENTS

# 1 Document

| | |
|---|---|
| **CMSIS.html** | Document of CMSIS version 5.1.1. |
| **NuMicro M2L31 Driver Reference Guide.chm** | This document describes the usage of drivers in M2L31 BSP. |
| **NuMicro M2L31 Series CMSIS BSP Revision History.pdf** | This document shows the revision history of M2L31 BSP. |

## 2 Library

| | |
|---|---|
| **CMSIS** | Cortex® Microcontroller Software Interface Standard (CMSIS) V5.1.1 definitions by Arm® Corp. |
| **Device** | CMSIS compliant device header file. |
| **PowerDeliveryLib** | Power Delivery Libraries and header files for dual, source and sink role. |
| **StdDriver** | All peripheral driver header and source files. |
| **TKLib** | Touch Key Libraries and header files for calibration and free-run. |
| **UsbHostLib** | USB host library source code. |

# 3 SampleCode

| | |
|---|---|
| **FreeRTOS** | Simple FreeRTOS™ demo codes. |
| **Hard_Fault_Sample** | Show hard fault information when hard fault happened.<br>The hard fault handler show some information included program counter, which is the address where the processor was executing when the hard fault occurs. The listing file (or map file) can show what function and instruction that was.<br>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. |
| **PowerDelivery** | Demonstrate Dual, Source, and Sink role sample code that is based on MAD025_UTCPD_V1 board. |
| **PowerManagement** | Sample codes for power management. |
| **Semihost** | Show how to print and get character through IDE console window. |
| **StdDriver** | Sample code to demonstrate the usage of M2L31 series MCU peripheral driver APIs. |
| **Template** | A project template for M2L31 series MCU. |
| **TouchKey** | Demonstrate how to calibrate Touch Key through NuSenAdj Tool. After calibration stage, show how to free-run the Touch Key based on the calibration data. |

# 4 ThirdParty

| | |
|---|---|
| **FatFs** | An open-source FAT/exFAT file system library. A generic FAT file system module for small embedded systems.<br>Its official website is http://elm-chan.org/fsw/ff/00index_e.html. |
| **FreeRTOS** | Real-time operating system for microcontrollers.<br>Its official website is http://www.freertos.org/. |

# 5 SampleCode\FreeRTOS

| | |
|---|---|
| **Blinky** | A simple LED toggle sample code. It could be used as the startup of M2L31 NuMaker board. |
| **TicklessIdle** | Show how to enable FreeRTOS™ tickless idle mode, and enable RTC for long duration task idle. |

# 6 SampleCode\ISP

| | |
|---|---|
| **ISP_CAN** | In-System-Programming Sample code through CANFD interface. |
| **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 $I^2C$ interface. |
| **ISP_MSC** | In-System-Programming Sample code through USB interface and following Mass Storage Class Specification. |
| **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. |

# 7 SampleCode\PowerDelivery

| | |
|---|---|
| **UTCPD_DualRole** | Demonstrate Dual Role Power Device. |
| **UTCPD_SinkRole** | Demonstrate Sink Power Device. |
| **UTCPD_SourceRole** | Demonstrate Source Power Device. |

# 8 SampleCode\PowerManagement

| | |
|---|---|
| **SYS_DPDMode_Wakeup** | Show how to wake up system from DPD Power-down mode by Wake-up pin 0 (PC.0) or Wake-up Timer or RTC Tick or RTC Alarm or RTC Tamper 0. |
| **SYS_PowerDownMode** | Show how to enter different Power-down mode and wake up by RTC. |
| **SYS_PowerMode** | Show how to set different core voltage. |
| **SYS_SPDMode_Wakeup** | Show how to wake up system from SPD Power-down mode by Wake-up pin (PC.0) or Wake-up Timer or Wake-up ACMP or RTC Tick or RTC Alarm or RTC Tamper 0 or BOD or LVR. |
| **SYS_SPDMode_WakeupAndReturn** | Show how to continue executing code after wake-up from SPD Power-down mode by SRAM data retention function. |
| **SYS_SPDMode_WakeupVTOR** | Show how to continue executing code after wake-up from SPD Power-down mode by VTOR function. |
| **SYS_SRAMPowerMode** | Show how to select SRAM power mode in system Power-down mode. |

# 9 SampleCode\StdDriver

## System Manager (SYS)

| | |
|---|---|
| **SYS_BODWakeup** | Demonstrate how to wake up system from 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. |

## RRAM Memory Controller (RMC)

| | |
|---|---|
| **RMC_CRC32** | Demonstrate how to use RMC CRC32 ISP command to calculate the CRC32 checksum of APROM and LDROM. |
| **RMC_DualBank** | Demonstrate how dual processes work in dual bank Flash architecture. |
| **RMC_ExeInSRAM** | Implement a code and execute it in SRAM to program embedded Flash. |
| **RMC_IAP** | Demonstrate RMC 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. |
| **RMC_MultiBoot** | Implement a multi-boot system to boot from different applications in APROM or LDROM by VECMAP. |
| **RMC_MultiWordProgram** | Show how to use RMC multi-word (word line) program flow to program APROM and LDROM. |
| **RMC_OTP** | Demonstrate how to program, read and lock OTP. |
| **RMC_ReadAllOne** | Demonstrate how to use RMC Read-All-One ISP command to verify APROM or LDROM pages are all 0xFFFFFFFF or not. |
| **RMC_RW** | Show RMC read Flash IDs, erase, read, and write |

| | |
|---|---|
| | functions. |
| **RMC_XOM** | Show how to configure and set up an XOM region then perform XOM function. |

## General Purpose I/O (GPIO)

| | |
|---|---|
| **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 and output control. |
| **GPIO_PowerDown** | Show how to wake up system from Power-down mode by GPIO interrupt. |

## PDMA Controller (PDMA)

| | |
|---|---|
| **PDMA_BasicMode** | Use PDMA0 channel 2 to transfer data from memory to memory. |
| **PDMA_ScatterGather** | Use PDMA0 channel 4 to transfer data from memory to memory by scatter-gather mode. |
| **PDMA_ScatterGather_ PingPongBuffer** | Use PDMA0 to implement Ping-Pong buffer by scatter-gather mode (memory to memory). |
| **PDMA_TimeOut** | Demonstrate PDMA0 channel 1 get/clear timeout flag with UART0. |

## Timer Controller (TIMER)

| | |
|---|---|
| **TIMER_ACMPTrigger** | Use ACMP to trigger Timer0 counter reset mode. |
| **TIMER_CaptureCounter** | Show how to use the Timer capture function to capture Timer counter value. |
| **TIMER_Delay** | Demonstrate the usage of TIMER_Delay API to generate a 1 second delay. |

| | |
|---|---|
| **TIMER_EventCounter** | Use TM0 pin to demonstrate Timer event counter function. |
| **TIMER_FreeCountingMode** | Use the timer TM0_EXT pin to demonstrate timer free counting mode function. And displays the measured input frequency to UART console. |
| **TIMER_InterTimerTriggerMode** | Use the timer TM0 pin 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_PeriodicINT** | Implement Timer counting in periodic mode. |
| **TIMER_PWM_AccumulatorINTStopMode** | Demonstrate TIMER PWM accumulator interrupt to stop counting. |
| **TIMER_PWM_AccumulatorINTTriggerPDMA** | Demonstrate TIMER PWM accumulator interrupt to trigger PDMA transfer. |
| **TIMER_PWM_ChangeDuty** | Change duty cycle and period of output waveform in PWM up count type. |
| **TIMER_PWM_OutputWaveform** | Demonstrate output different duty waveform in Timer0~3 PWM. |
| **TIMER_TimeoutWakeup** | Use timer to wake up system from Power-down mode periodically. |
| **TIMER_ToggleOut** | Demonstrate the Timer0 toggle out function on TM0 pin. |

## Watchdog Timer (WDT)

| | |
|---|---|
| **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. |

## Window Watchdog Timer (WWDT)

| | |
|---|---|
| **WWDT_ReloadCounter** | Show how to reload the WWDT counter value. |

## Real Timer Clock (RTC)

| | |
|---|---|
| **RTC_Alarm_Test** | Demonstrate the RTC alarm function. It sets an alarm 10 seconds after execution. |
| **RTC_Alarm_Wakeup** | Use RTC alarm interrupt event to wake up system. |
| **RTC_SpareRegisterRW** | Demonstrate the RTC spare register read/write function and displays test result to the UART console. |
| **RTC_Tamper** | Demonstrate the RTC tamper function. |
| **RTC_Time_Display** | Demonstrate the RTC function and displays current time to the UART console. |
| **RTC_TimeAndTick** | Get the current RTC data/time per tick. |

## Pulse Width Modulation Controller (PWM)

| | |
|---|---|
| **PWM_240KHz_SwitchDuty** | Demonstrate how to set PWM0 channel 0 output 240 kHz waveform and switch duty in each 0.5%. |
| **PWM_Brake** | Demonstrate how to use PWM brake function. |
| **PWM_Capture** | Capture the PWM Channel 2 waveform by PWM Channel 0. |
| **PWM_DeadTime** | Demonstrate how to use PWM Dead Time function. |
| **PWM_DoubleBuffer** | Change duty cycle and period of output waveform by PWM double buffer function. |
| **PWM_OutputWaveform** | Demonstrate how to use PWM counter output waveform. |
| **PWM_PDMA_Capture** | Capture the PWM0 Channel 0 waveform by PWM0 Channel 2, and use PDMA to transfer captured data. |
| **PWM_PDMA_Capture_1MHzSingal** | Capture the PWM0 Channel 0 waveform by PWM0 Channel 2, and use PDMA to transfer captured data. The frequency of PWM Channel 0 is 1 MHz, which is used to test the maximum input frequency for PWM Capture function. |
| **PWM_SwitchDuty** | Change duty cycle of output waveform by configured |

| | |
|---|---|
| | period. |
| **PWM_SyncStart** | Demonstrate how to use PWM counter synchronous start function. |

## Enhanced Pulse Width Modulation Controller (EPWM)

| | |
|---|---|
| **EPWM_AccumulatorINT_TriggerPDMA** | Demonstrate how to use EPWM accumulator interrupt trigger PDMA. |
| **EPWM_Brake** | Demonstrate how to use EPWM brake function. |
| **EPWM_Capture** | Capture the EPWM1 Channel 0 waveform by EPWM1 Channel 2. |
| **EPWM_DeadTime** | Demonstrate how to use EPWM Dead Time function. |
| **EPWM_DoubleBuffer** | Change duty cycle and period of output waveform by EPWM Double Buffer function. |
| **EPWM_OutputWaveform** | Demonstrate how to use EPWM counter output waveform. |
| **EPWM_PDMA_Capture** | Capture the EPWM1 Channel 0 waveform by EPWM1 Channel 2, and use PDMA to transfer captured data. |
| **EPWM_SwitchDuty** | Change duty cycle of output waveform by configured period. |
| **EPWM_SyncStart** | Demonstrate how to use EPWM counter synchronous start function. |

## UART Interface Controller (UART)

| | |
|---|---|
| **UART_AutoBaudRate** | Show how to use auto baud rate detection function. |
| **UART_AutoFlow** | Transmit and receive data using auto flow control. |
| **UART_IrDA** | Transmit and receive UART data in UART IrDA mode. |
| **UART_LIN** | Transmit LIN frame including header and response in UART LIN mode. |

| | |
|---|---|
| **UART_PDMA** | Demonstrate UART transmit and receive function with PDMA. |
| **UART_RS485** | Transmit and receive data in UART RS485 mode. |
| **UART_SingleWire** | Transmit and receive data in UART single-wire 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. |

## Serial Peripheral Interface (SPI)

| | |
|---|---|
| **SPI_Loopback** | SPI read/write demo connecting SPI MISO and MOSI pins. |
| **SPI_MasterFIFOMode** | 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. |
| **SPI_PDMA_LoopTest** | SPI read/write demo in PDMA mode.<br>Connecting SPI MISO and MOSI pins. Both TX PDMA function and RX PDMA function will be enabled. |
| **SPI_SlaveFIFOMode** | 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. |
| **SPII2S_Master** | Configure SPI in I$^2$S Master mode and demonstrate how I$^2$S works in Master mode. |
| **SPII2S_PDMA_NAU8822** | An I$^2$S demo with PDMA function connected to audio codec NAU8822. |
| **SPII2S_PDMA_Play** | An I$^2$S demo for playing data and demonstrating how I2S works with PDMA. |
| **SPII2S_PDMA_PlayRecord** | An I$^2$S demo for playing and recording data with PDMA function. |

| | |
|---|---|
| **SPII2S_PDMA_Record** | An I²S demo for recording data and demonstrating how I²S works with PDMA. |
| **SPII2S_Slave** | Configure SPI as I²S Slave mode and demonstrate how I²S works in Slave mode. This sample code needs to work with SPII2S_Master. |

## Quad Serial Peripheral Interface (QSPI)

| | |
|---|---|
| **QSPI_DualMode_Flash** | Access SPI Flash using QSPI dual mode. |
| **QSPI_QuadMode_Flash** | Access SPI Flash using QSPI quad mode. |
| **QSPI_Slave3Wire** | Configure QSPI0 as Slave 3 wire 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 sample code. |

## I²C Serial Interface Controller (I²C)

| | |
|---|---|
| **I2C_Double_Buffer_Slave** | Demonstrate how to set I²C two-level buffer in Slave mode to receive 256 bytes data from a master. This sample code needs to work with I2C_MultiBytes_Master. |
| **I2C_EEPROM** | Show how to use I²C interface to access EEPROM. |
| **I2C_Loopback** | Demonstrate how to set I²C Master mode and Slave Mode, and show how a master accesses a slave on a chip. |
| **I2C_Master** | Show how a master accesses a slave. This sample code needs to work with I2C_Slave. |
| **I2C_MultiBytes_Master** | Show how to set I²C Multi bytes API Read and Write data to Slave. This sample code needs to work with I2C_Slave. |
| **I2C_PDMA_TRX** | Demonstrate I²C PDMA mode and need to connect I2C0 (master) and I2C1 (slave). |
| **I2C_SingleByte_Master** | Show how to use I²C Single byte API Read and Write data to Slave. This sample code needs to work with |

| | |
|---|---|
| | I2C_Slave. |
| **I2C_Slave** | Demonstrate how to set I$^2$C in Slave mode to receive 256 bytes data from a master. This sample code needs to work with I2C_Master. |
| **I2C_SMBus** | Demonstrate how to control SMBus interface and use SMBus protocol between Host and Slave. |
| **I2C_Wakeup_Slave** | Show how to wake up MCU from Power-down mode via the I$^2$C interface. This sample code needs to work with I2C_Master. |

## Universal Serial Control Interface Controller – UART Mode (USCI-UART)

| | |
|---|---|
| **USCI_UART_AutoBaudRate** | Show how to use auto baud rate detection function. |
| **USCI_UART_Autoflow** | Transmit and receive data using auto flow control. |
| **USCI_UART_PDMA** | Transmit and receive UART data with PDMA. |
| **USCI_UART_RS485** | Transmit and receive data in RS485 mode. |
| **USCI_UART_TxRxFunction** | Transmit and receive data from PC terminal through a RS232 interface. |
| **USCI_UART_Wakeup** | Show how to wake up system from Power-down mode by USCI interrupt in UART mode. |

## Universal Serial Control Interface Controller – SPI Mode (USCI-SPI)

| | |
|---|---|
| **USCI_SPI_Loopback** | 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. |
| **USCI_SPI_MasterMode** | 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 sample code. |
| **USCI_SPI_PDMA_LoopTest** | Demonstrate USCI_SPI data transfer with PDMA. USCI_SPI0 will be configured as Master mode and USCI_SPI1 will be configured as Slave mode. Both TX |

| | |
|---|---|
| | PDMA function and RX PDMA function will be enabled. |
| **USCI_SPI_SlaveMode** | 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 sample code. |

## Universal Serial Control Interface Controller – I$^2$C Mode (USCI-I2C)

| | |
|---|---|
| **USCI_I2C_EEPROM** | Demonstrate how to access EEPROM through a USCI_I2C interface. |
| **USCI_I2C_Loopback** | Demonstrate how a Master accesses 7-bit address Slave (loopback) |
| **USCI_I2C_Loopback_10bit** | Demonstrate how a Master accesses 10-bit address Slave (loopback) |
| **USCI_I2C_Master** | Demonstrate how a Master accesses Slave. This sample code needs to work with USCI_I2C_Slave sample code. |
| **USCI_I2C_Master_10bit** | Demonstrate how a Master uses 10-bit addressing access Slave. This sample code needs to work with USCI_I2C_Slave_10bit sample code. |
| **USCI_I2C_Monitor** | Demonstrate how USCI_I2C monitors transmission between I2C Master and I2C Slave. |
| **USCI_I2C_MultiBytes_Master** | Demonstrate how to use multi-bytes API to access slave. This sample code needs to work with USCI_I2C_Slave sample code. |
| **USCI_I2C_SingleByte_Master** | Demonstrate how to use single byte API to access slave. This sample code needs to work with USCI_I2C_Slave sample code. |
| **USCI_I2C_Slave** | Demonstrate how to set USCI_I2C in slave mode to receive the data from a Master. This sample code needs to work with USCI_I2C_Master sample code. |
| **USCI_I2C_Slave_10bit** | Demonstrate how to set USCI_I2C in 10-bit addressing slave mode to receive the data from a Master. This sample code needs to work with |

| | |
|---|---|
| | USCI_I2C_Master_10bit sample code. |
| **USCI_I2C_Wakeup_Slave** | Demonstrate how to set USCI_I2C to wake up MCU from Power-down mode. This sample code needs to work with USCI_I2C_Master sample code. |

## External Bus Interface (EBI)

| | |
|---|---|
| **EBI_NOR** | Configure EBI interface to access NOR Flash connected on EBI interface. |
| **EBI_SRAM** | Configure EBI interface to access SRAM connected on EBI interface. |

## USB 1.1 Device Controller (USBD)

| | |
|---|---|
| **USBD_HID_Keyboard** | Demonstrate how to implement a USB keyboard device. It 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_MouseKeyboard** | Simulate an USB HID mouse and HID keyboard. Mouse draws circle on the screen and Keyboard use GPIO to simulate key input. |
| **USBD_HID_RemoteWakeup** | Demonstrate how to implement a USB mouse device. It uses PA0 ~ PA5 to control mouse direction and mouse key. It also supports USB suspend and remote wakeup. |
| **USBD_HID_Touch** | Demonstrate how to implement a USB touch digitizer device. Two lines demo in Paint. |
| **USBD_HID_Transfer** | Demonstrate how to 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_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 |

| | |
|---|---|
| | a 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_HID_Transfer_CTRL** | 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. |
| **USBD_Mass_Storage_CDROM** | Demonstrate the emulation of USB Mass Storage Device CD-ROM. |
| **USBD_Mass_Storage_Flash** | Use Flash as storage to implement a USB Mass-Storage device. |
| **USBD_Mass_Storage_SRAM** | Use internal SRAM as back end storage media to simulate an USB pen drive. |
| **USBD_Micro_Printer** | Demonstrate 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** | Demonstrate how to implement a composite device (VCOM and HID Keyboard). |
| **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** | Demonstrate how to implement a composite device of VCOM and mass storage. |

| | |
|---|---|
| **USBD_VCOM_DualPort** | Demonstrate how to implement a USB dual virtual COM port device. |
| **USBD_VCOM_SerialEmulator** | Demonstrate how to implement a USB virtual COM port device. |

## CRC Controller (CRC)

| | |
|---|---|
| **CRC_CCITT** | Implement CRC in CRC-CCITT mode and get the CRC checksum result. |
| **CRC_CRC32_PDMA** | Implement CRC in CRC-32 mode and get the CRC checksum result. |
| **CRC_CRC8** | Implement CRC in CRC-8 mode and get the CRC checksum result. |
| **CRC_POLYNOMIAL** | Demonstrate how to use polynomial mode and get the CRC checksum result. |

## Cryptographic Accelerator (CRYPTO)

| | |
|---|---|
| **CRYPTO_AES** | Show Crypto IP AES-128 ECB mode encrypt/decrypt function. |
| **CRYPTO_PRNG** | Generate random numbers using Crypto IP PRNG. |

## Enhanced Analog-to-Digital Converter (EADC)

| | |
|---|---|
| **EADC_Accumulate** | Demonstrate how to get accumulate conversion result. |
| **EADC_ADINT_Trigger** | Use ADINT interrupt to do the EADC continuous scan conversion. |
| **EADC_Average** | Demonstrate how to get average conversion result. |
| **EADC_BandGap** | Convert Band-gap and print conversion result. |
| **EADC_EPWM_Trigger** | Demonstrate how to trigger EADC by EPWM. |
| **EADC_PDMA_EPWM_Trigger** | Demonstrate how to trigger EADC by EPWM and transfer conversion data by PDMA. |

| | |
|---|---|
| **EADC_Pending_Priority** | Demonstrate how to trigger multiple sample modules and got conversion results in order of priority. |
| **EADC_ResultMonitor** | Monitor the conversion result of channel 2 by the digital compare function. |
| **EADC_SwTrg_Trigger** | Trigger EADC by writing EADC software trigger register. |
| **EADC_TempSensor** | Convert temperature sensor and print conversion result. |
| **EADC_Timer_Trigger** | Show how to trigger EADC by Timer. |

## Digital-to-Analog Converter (DAC)

| | |
|---|---|
| **DAC_PDMA_TimerTrigger** | Show how Timer triggers DAC to fetch data with PDMA and convert sine wave outputs. |
| **DAC_SoftwareTrigger** | Demonstrate how software triggers DAC to convert sine wave outputs. |
| **DAC_TimerTrigger** | Demonstrate how Timer triggers DAC to convert sine wave outputs. |

## Analog Comparator Converter (ACMP)

| | |
|---|---|
| **ACMP_CompareVBG** | Demonstrate analog comparator (ACMP) comparison by comparing ACMP1_P1 input and VBG voltage and shows the result on UART console. |
| **ACMP_Wakeup** | Use ACMP to wake up system from Power-down mode while comparator output changes. |
| **ACMP_WindowCompare** | Show how to monitor ACMP input with window compare function. |
| **ACMP_WindowLatch** | Demonstrate how to use ACMP window latch mode. |

## OP Amplifier (OPA)

| | |
|---|---|
| **OPA_NonInvertingAmp** | Demonstrate how to use OPA as a non-inverting amplifier. |

| | |
|---|---|
| **OPA_InvertingAmp** | Demonstrate how to use OPA as an inverting amplifier. |

## Controller Area Network with Flexible DataRate (CANFD)

| | |
|---|---|
| **CANFD_CAN_Loopback** | Use CAN mode function to do internal loopback test. |
| **CANFD_CAN_MonitorMode** | Use CAN Monitor mode to listen to CAN bus communication test. |
| **CANFD_CAN_TxRx** | Transmit and receive CAN message through CAN interface. |
| **CANFD_CAN_TxRxINT** | An example of interrupt control using CAN bus communication. |
| **CANFD_CANFD_Loopback** | Use CANFD mode function to do internal loopback test. |
| **CANFD_CANFD_MonitorMode** | Use CANFD Monitor mode to listen to CAN bus communication test. |
| **CANFD_CANFD_TxRx** | Transmit and receive CANFD message through CAN interface. |
| **CANFD_CANFD_TxRxINT** | An example of interrupt control using CANFD bus communication. |

## Enhanced Input Capture Timer (ECAP)

| | |
|---|---|
| **ECAP_GetInputFreq** | Show how to use ECAP interface to get input frequency. |
| **ECAP_GetQEIFreq** | Show how to use ECAP interface to get EQEI-A frequency. |

## Enhanced Quadrature Encoder Interface (EQEI)

| | |
|---|---|
| **EQEI_CompareMatch** | Show the usage of EQEI compare function. |
| **EQEI_PeriodicINT** | Show the usage of EQEI Unit Timer function. |

## USB Host Controller (USBH)

| | |
|---|---|
| **USBH_AudioClass** | Demonstrate how to use USBH Audio Class driver. It shows the mute, volume, auto-gain, channel and sampling rate control. |
| **USBH_DEV_CONN** | Use connect/disconnect callback functions to handle of device connect and disconnect events. |
| **USBH_HID** | Use USB Host core driver and HID driver. This sample demonstrates how to submit HID class request and how to read data from interrupt pipe. This sample supports dynamic device plug/un-plug and multiple HID devices. |
| **USBH_HID_KEYBOARD** | Demonstrate reading key inputs from USB keyboards. This sample includes an USB keyboard driver which is based on the HID driver. |
| **USBH_HID_Mouse_Keyboard** | Use USB Host core driver and HID driver. This sample demonstrates how to support mouse and keyboard input. |
| **USBH_MassStorage** | Use a command-shell-like interface to demonstrate how to use USBH mass storage driver and make it working as a disk driver under FATFS file system. |
| **USBH_UAC_HID** | Show how to use USBH Audio Class driver and HID driver at the same time. The target device is a Game Audio (UAC+HID composite device). |
| **USBH_VCOM** | Receive audio data from UAC device, and immediately send back to that UAC device. |

## Random Number Generator (RNG)

| | |
|---|---|
| **RNG_EntropyPoll** | Generate random numbers using TRNG. |
| **RNG_Random** | Generate random numbers using TRNG. |

## Tick Timer Controller (TTMR)

| | |
|---|---|
| **TTMR_Delay** | Demonstrate the usage of TTMR_Delay() API to |

| | |
|---|---|
| | generate a 1 second delay. |
| **TTMR_Periodic** | Use the TTMR periodic mode to generate timer interrupt every 1 second. |
| **TTMR_PeriodicINT** | Implement TTMR counting in periodic mode. |
| **TTMR_TimeoutWakeup** | Use TTMR to wake up system from Power-down mode periodically. |

## Low Power Analog to Digital Converter (LPADC)

| | |
|---|---|
| **LPADC_MaxSps_ContinuousScanMode** | Demonstrate how to use PCLK2 as LPADC clock source to achieve maximum ksps LPADC conversion rate. |
| **LPADC_ADINT_Trigger** | Use ADINT interrupt to do the LPADC Single-cycle scan conversion. |
| **LPADC_AutoOperation** | Demonstrate how to enable LPADC Auto-operation mode to convert when chip enters Power-down mode. |
| **LPADC_BandGap** | Convert Band-gap and print conversion result. |
| **LPADC_BandGapCalculateAVDD** | Demonstrate how to calculate AVdd voltage by using Band-gap. |
| **LPADC_BurstMode** | Perform A/D Conversion with LPADC burst mode. |
| **LPADC_ContinuousScanMode** | Perform A/D Conversion with LPADC continuous scan mode. |
| **LPADC_LPPDMA_PWM_Trigger** | Demonstrate how to trigger LPADC by PWM and transfer conversion data by LPPDMA. |
| **LPADC_PWM_Trigger** | Demonstrate how to trigger LPADC by PWM. |
| **LPADC_ResultMonitor** | Monitor the conversion result of channel 2 by the digital compare function. |
| **LPADC_SingleCycleScanMode** | Perform A/D Conversion with LPADC single cycle scan mode. |
| **LPADC_SingleMode** | Perform A/D Conversion with LPADC single mode. |

| | |
|---|---|
| **LPADC_STADC_Trigger** | Show how to trigger LPADC by STADC pin. |
| **LPADC_SwTrg_Trigger** | Trigger LPADC by writing LPADC software trigger register. |
| **LPADC_Timer_Trigger** | Show how to trigger LPADC by Timer. |

## Low Power General Purpose I/O Controller (LPGPIO)

| | |
|---|---|
| **LPGPIO_OutputInput** | Show how to set LPGPIO pin mode and use pin data input/output control. |

## Low Power I$^2$C Serial Interface Controller (LPI2C)

| | |
|---|---|
| **LPI2C_AutoOperation** | Demonstrate LPI2C Auto-operation mode when chip enters Power-down mode. This sample code needs to work with LPI2C_Slave. |
| **LPI2C_EEPROM** | Show how to use LPI2C interface to access EEPROM. |
| **LPI2C_LPPDMA_TRX** | Demonstrate LPI2C LPPDMA mode and need two boards to connect LPI2C0 (Master) and LPI2C0 (Slave). |
| **LPI2C_Master** | Show how a master accesses a slave. This sample code needs to work with LPI2C_Slave. |
| **LPI2C_MultiBytes_Master** | Show how to set LPI2C use Multi bytes API Read and Write data to Slave. This sample code needs to work with LPI2C_Slave sample code. |
| **LPI2C_SingleByte_Master** | Show how to use LPI2C Single byte API Read and Write data to Slave. This sample code needs to work with LPI2C_Slave sample code. |
| **LPI2C_Slave** | Show how to set LPI2C in Slave mode and receive the data from Master. This sample code needs to work with LPI2C_Master. |
| **LPI2C_Wakeup_Slave** | Show how to wake up MCU from Power-down mode through LPI2C interface. This sample code needs to work with LPI2C_Master. |

## Low Power PDMA Controller (LPPDMA)

| | |
|---|---|
| **LPPDMA_BasicMode** | Use LPPDMA0 channel 2 to transfer data from memory to memory. |
| **LPPDMA_ScatterGather** | Use LPPDMA0 channel 2 to transfer data from memory to memory by scatter-gather mode. |
| **LPPDMA_ScatterGather_PingPongBuffer** | Use LPPDMA0 to implement Ping-Pong buffer by scatter-gather mode (memory to memory). |

## Low Power Serial Peripheral Interface (LPSPI)

| | |
|---|---|
| **LPSPI_AutoOperation** | Demonstrate how to do LPSPI loopback test in Auto-operation mode when chip enters Power-down mode |
| **LPSPI_Loopback** | LPSPI read/write demo by connecting LPSPI MISO and MOSI pins. |
| **LPSPI_MasterFIFOMode** | Configure LPSPI 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 LPSPI_SlaveFIFOMode. |
| **LPSPI_LPPDMA_LoopTest** | LPSPI read/write demo in LPPDMA mode.<br>Connecting LPSPI MISO and MOSI pins. Both TX LPPDMA function and RX LPPDMA function will be enabled. |
| **LPSPI_SlaveFIFOMode** | Configure LPSPI 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 LPSPI_MasterFIFOMode. |

## Low Power Timer Controller (LPTMR)

| | |
|---|---|
| **LPTMR_ACMPTrigger** | Use ACMP to trigger LPTMR0 counter reset mode. |
| **LPTMR_CaptureCounter** | Show how to use the LPTMR capture function to capture LPTMR counter value. |
| **LPTMR_Delay** | Demonstrate the usage of LPTMR_Delay() API to |

| | |
|---|---|
| | generate a 1 second delay. |
| **LPTMR_EventCounter** | Use LPTM0 pin to demonstrates LPTMR event counter function. |
| **LPTMR_FreeCountingMode** | Use the LPTMR LPTM0_EXT pin to demonstrate timer free counting mode function. And displays the measured input frequency to UART console. |
| **LPTMR_Periodic** | Use the LPTMR periodic mode to generate timer interrupt every 1 second. |
| **LPTMR_PeriodicINT** | Implement LPTMR counting in periodic mode. |
| **LPTMR_PWM_AccumulatorINTStopMode** | Demonstrate LPTMR PWM accumulator interrupt to stop counting. |
| **LPTMR_PWM_AccumulatorINTTriggerLPPDMA** | Demonstrate LPTMR PWM accumulator interrupt to trigger LPPDMA transfer. |
| **LPTMR_PWM_ChangeDuty** | Change duty cycle and period of output waveform in PWM up count type. |
| **LPTMR_PWM_OutputWaveform** | Demonstrate output different duty waveform in LPTMR0~1 PWM. |
| **LPTMR_TimeoutWakeup** | Use LPTMR to wake up system from Power-down mode periodically. |
| **LPTMR_ToggleOut** | Demonstrate the LPTMR toggle out function on LPTM0 pin. |

## Low Power UART Interface Controller (LPUART)

| | |
|---|---|
| **LPUART_AutoBaudRate** | Show how to use auto baud rate detection function. |
| **LPUART_AutoFlow** | Transmit and receive data using auto flow control. |
| **LPUART_LPPDMA** | Demonstrate LPUART transmit and receive function with LPPDMA. |
| **LPUART_PD_LPTMR0_Trigger** | Demonstrate how to enable LPUART Auto-operation mode to transmit and receive data when chip enters |

| | |
|---|---|
| | Power-down mode. |
| **LPUART_RS485** | Transmit and receive data in LPUART RS485 mode. |
| **LPUART_SingleWire** | Transmit and receive data in LPUART single-wire mode. |
| **LPUART_TxRxFunction** | Transmit and receive data from PC terminal through RS232 interface. |
| **LPUART_Wakeup** | Show how to wake up system from Power-down mode by LPUART interrupt. |

# 10 Sample Code Compatibility List

| Category Type | Part Number | Note |
|---|---|---|
| Type-A | M2L31KIDAE, M2L31KGDAE, M2L31SIDAE, M2L31SGDAE, M2L31CGDAE, M2L31CIDAE, M2L31LIDAE, M2L31LGDAE, M2L31YIDAE, M2L31YGDAE | |
| Type-B | M2L31SG4AE, M2L31SE4AE, M2L31LG4AE, M2L31LE4AE, M2L31LD4AE, M2L31YG4AE, M2L31YE4AE, M2L31YD4AE, M2L31ZE4AE, M2L31ZD4AE, M2L31XD4AE | 4KB SRAM |

| Category Type / Sample Code | Type-A | Type-B |
|---|---|---|
| Blinky | √ | √ |
| TicklessIdle | √ | √ |
| Hard_Fault_Sample | √ | √ |
| ISP_CAN | √ | √ |
| ISP_DFU | √ | √ |
| ISP_HID | √ | √ |
| ISP_I2C | √ | √ |
| ISP_MSC | √ | √ |
| ISP_RS485 | √ | √ |
| ISP_SPI | √ | √ |
| ISP_UART | √ | √ |
| UTCPD_DualRole | √ | √ |
| UTCPD_SinkRole | √ | √ |
| UTCPD_SourceRole | √ | √ |
| SYS_DPDMode_Wakeup | √ | √ |
| SYS_PowerDownMode | √ | √ |
| SYS_PowerMode | √ | √ |
| SYS_SPDMode_Wakeup | √ | √ |
| SYS_SPDMode_WakeupAndReturn | √ | √ |

| | | |
|---|---|---|
| SYS_SPDMode_WakeupVTOR | √ | √ |
| SYS_SRAMPowerMode | √ | √ (#define __M2L31_40KB_SRM__) |
| Semihost | √ | √ |
| Template | √ | √ |
| AllInOne_TK | √ | - |
| ACMP_ComapreVBG | √ | √ |
| ACMP_Wakeup | √ | √ |
| ACMP_WindowCompare | √ | √ |
| ACMP_WindowLatch | √ | √ |
| CANFD_CANFD_Loopback | √ | √ |
| CANFD_CANFD_MonitorMode | √ | √ |
| CANFD_CANFD_TxRx | √ | √ |
| CANFD_CANFD_TxRxINT | √ | √ |
| CANFD_CAN_Loopback | √ | √ |
| CANFD_CAN_MonitorMode | √ | √ |
| CANFD_CAN_TxRx | √ | √ |
| CANFD_CAN_TxRxINT | √ | √ |
| CRC_CCITT | √ | √ |
| CRC_CRC32_PDMA | √ | √ |
| CRC_CRC8 | √ | √ |
| CRC_POLYNOMIAL | √ | √ |
| CRYPTO_AES | √ | √ |
| CRYPTO_PRNG | √ | √ |
| DAC_PDMA_TimerTrigger | √ | √ |
| DAC_SoftwareTrigger | √ | √ |
| DAC_TimerTrigger | √ | √ |
| EADC_Accumulate | √ | √ |
| EADC_ADINT_Trigger | √ | √ |
| EADC_Average | √ | √ |

| | | |
|---|---|---|
| **EADC_BandGap** | √ | √ |
| **EADC_BandGapCalculateAVDD** | √ | √ |
| **EADC_EPWM_Trigger** | √ | √ |
| **EADC_PDMA_EPWM_Trigger** | √ | √ |
| **EADC_Pending_Priority** | √ | √ |
| **EADC_ResultMonitor** | √ | √ |
| **EADC_SwTrg_Trigger** | √ | √ |
| **EADC_TempSensor** | √ | √ |
| **EADC_Timer_Trigger** | √ | √ |
| **EBI_NOR** | √ | - |
| **EBI_SRAM** | √ | - |
| **ECAP_GetInputFreq** | √ | √ |
| **ECAP_GetQEIFreq** | √ | √ |
| **EPWM_AccumulatorINT_TriggerPDMA** | √ | √ |
| **EPWM_Brake** | √ | √ |
| **EPWM_Capture** | √ | √ |
| **EPWM_DeadTime** | √ | √ |
| **EPWM_DoubleBuffer** | √ | √ |
| **EPWM_OutputWaveform** | √ | √ |
| **EPWM_PDMA_Capture** | √ | √ |
| **EPWM_SwitchDuty** | √ | √ |
| **EPWM_SyncStart** | √ | √ |
| **EQEI_CompareMatch** | √ | √ |
| **EQEI_PeriodicINT** | √ | √ |
| **GPIO_EINTAndDebounce** | √ | √ |
| **GPIO_INT** | √ | √ |
| **GPIO_OutputInput** | √ | √ |
| **GPIO_PowerDown** | √ | √ |
| **I2C_Double_Buffer_Slave** | √ | √ |
| **I2C_EEPROM** | √ | √ |

| | | |
|---|---|---|
| **I2C_Loopback** | √ | √ |
| **I2C_Master** | √ | √ |
| **I2C_MultiBytes_Master** | √ | √ |
| **I2C_PDMA_TRX** | √ | √ |
| **I2C_SingleByte_Master** | √ | √ |
| **I2C_Slave** | √ | √ |
| **I2C_SMBus** | √ | - |
| **I2C_Wakeup_Slave** | √ | √ |
| **LPADC_ADINT_Trigger** | √ | √ |
| **LPADC_AutoOperation** | √ | √ |
| **LPADC_BandGap** | √ | √ |
| **LPADC_BandGapCalculateAVDD** | √ | √ |
| **LPADC_BurstMode** | √ | √ |
| **LPADC_ContinuousScanMode** | √ | √ |
| **LPADC_LPPDMA_PWM_Trigger** | √ | - |
| **LPADC_MaxSps_ContinuousScanMode** | √ | √ |
| **LPADC_PWM_Trigger** | √ | - |
| **LPADC_ResultMonitor** | √ | √ |
| **LPADC_SingleCycleScanMode** | √ | √ |
| **LPADC_SingleMode** | √ | √ |
| **LPADC_STADC_Trigger** | √ | √ |
| **LPADC_SwTrg_Trigger** | √ | √ |
| **LPADC_Timer_Trigger** | √ | √ |
| **LPGPIO_OutputInput** | √ | √ |
| **LPI2C_AutoOperation** | √ | √ |
| **LPI2C_EEPROM** | √ | √ |
| **LPI2C_LPPDMA_TRX** | √ | √ |
| **LPI2C_Master** | √ | √ |
| **LPI2C_MultiBytes_Master** | √ | √ |
| **LPI2C_SingleByte_Master** | √ | √ |

| | | |
|---|---|---|
| **LPI2C_Slave** | √ | √ |
| **LPI2C_Wakeup_Slave** | √ | √ |
| **LPPDMA_BasicMode** | √ | √ |
| **LPPDMA_ScatterGather** | √ | √ |
| **LPPDMA_ScatterGather_PingPongBuffer** | √ | √ |
| **LPSPI_AutoOperation** | √ | √ |
| **LPSPI_Loopback** | √ | √ |
| **LPSPI_LPPDMA_LoopTest** | √ | √ |
| **LPSPI_MasterFIFOMode** | √ | √ |
| **LPSPI_SlaveFIFOMode** | √ | √ |
| **LPTMR_ACMPTrigger** | √ | √ |
| **LPTMR_CaptureCounter** | √ | √ |
| **LPTMR_Delay** | √ | √ |
| **LPTMR_EventCounter** | √ | √ |
| **LPTMR_FreeCountingMode** | √ | √ |
| **LPTMR_Periodic** | √ | √ |
| **LPTMR_PeriodicINT** | √ | √ |
| **LPTMR_PWM_AccumulatorINTStopMode** | √ | √ |
| **LPTMR_PWM_AccumulatorINTTriggerLPPDMA** | √ | √ |
| **LPTMR_PWM_ChangeDuty** | √ | √ |
| **LPTMR_PWM_OutputWaveform** | √ | √ |
| **LPTMR_TimeoutWakeup** | √ | √ |
| **LPTMR_ToggleOut** | √ | √ |
| **LPUART_AutoBaudRate** | √ | √ |
| **LPUART_AutoFlow** | √ | √ |
| **LPUART_LPPDMA** | √ | √ |
| **LPUART_PD_LPTMR0_Trigger** | √ | √ |
| **LPUART_RS485** | √ | √ |
| **LPUART_SingleWire** | √ | √ |
| **LPUART_TxRxFunction** | √ | √ |

| | | |
|---|---|---|
| LPUART_Wakeup | √ | √ |
| OPA_InvertingAmp | √ | √ |
| OPA_NonInvertingAmp | √ | √ |
| PDMA_BasicMode | √ | √ |
| PDMA_ScatterGather | √ | √ |
| PDMA_ScatterGather_PingPongBuffer | √ | √ |
| PDMA_TimeOut | √ | √ |
| PWM_240kHz_SwitchDuty | √ | - |
| PWM_Brake | √ | - |
| PWM_Capture | √ | - |
| PWM_DeadTime | √ | - |
| PWM_DoubleBuffer | √ | - |
| PWM_OutputWaveform | √ | - |
| PWM_PDMA_Capture | √ | - |
| PWM_PDMA_Capture_1MHzSignal | √ | - |
| PWM_SwitchDuty | √ | - |
| PWM_SyncStart | √ | - |
| QSPI_DualMode_Flash | √ | √ |
| QSPI_QuadMode_Flash | √ | √ |
| QSPI_Slave3Wire | √ | √ |
| RMC_CRC32 | √ | √ |
| RMC_DualBank | √ | √ |
| RMC_ExeInSRAM | √ | √ |
| RMC_IAP | √ | √ |
| RMC_MultiBoot | √ | √ |
| RMC_MultiWordProgram | √ | √ |
| RMC_OTP | √ | √ |
| RMC_ReadAllOne | √ | √ |
| RMC_RW | √ | √ |
| RMC_XOM | √ | √ |

| | | |
|---|---|---|
| RNG_EntropyPoll | √ | - |
| RNG_Random | √ | - |
| RTC_Alarm_Test | √ | √ |
| RTC_Alarm_Wakeup | √ | √ |
| RTC_SpareRegisterRW | √ | √ |
| RTC_Tamper | √ | √ |
| RTC_TimeAndTick | √ | √ |
| RTC_Time_Display | √ | √ |
| SPII2S_Master | √ | √ |
| SPII2S_PDMA_NAU8822 | √ | √ |
| SPII2S_PDMA_Play | √ | √ |
| SPII2S_PDMA_PlayRecord | √ | √ |
| SPII2S_PDMA_Record | √ | √ |
| SPII2S_Slave | √ | √ |
| SPI_Loopback | √ | √ |
| SPI_MasterFIFOMode | √ | √ |
| SPI_PDMA_LoopTest | √ | √ |
| SPI_SlaveFIFOMode | √ | √ |
| SYS_BODWakeup | √ | √ |
| SYS_PLLClockOutput | √ | √ |
| SYS_PowerDown_MinCurrent | √ | √ |
| TIMER_ACMPTrigger | √ | √ |
| TIMER_CaptureCounter | √ | √ |
| TIMER_Delay | √ | √ |
| TIMER_EventCounter | √ | √ |
| TIMER_FreeCountingMode | √ | √ |
| TIMER_InterTimerTriggerMode | √ | √ |
| TIMER_Periodic | √ | √ |
| TIMER_PeriodicINT | √ | √ |
| TIMER_PWM_AccumulatorINTStopMode | √ | √ |

| | | |
|---|---|---|
| TIMER_PWM_AccumulatorINTTriggerPDMA | √ | √ |
| TIMER_PWM_ChangeDuty | √ | √ |
| TIMER_PWM_OutputWaveform | √ | √ |
| TIMER_TimeoutWakeup | √ | √ |
| TIMER_ToggleOut | √ | √ |
| TTMR_Delay | √ | √ |
| TTMR_Periodic | √ | √ |
| TTMR_PeriodicINT | √ | √ |
| TTMR_TimeoutWakeup | √ | √ |
| UART_AutoBaudRate | √ | √ |
| UART_AutoFlow | √ | √ |
| UART_IrDA | √ | √ |
| UART_LIN | √ | √ |
| UART_PDMA | √ | √ |
| UART_RS485 | √ | √ |
| UART_SingleWire | √ | √ |
| UART_TxRxFunction | √ | √ |
| UART_Wakeup | √ | √ |
| USBD_HID_Keyboard | √ | √ |
| USBD_HID_Mouse | √ | √ |
| USBD_HID_MouseKeyboard | √ | √ |
| USBD_HID_RemoteWakeup | √ | √ |
| USBD_HID_Touch | √ | √ |
| USBD_HID_Transfer | √ | √ |
| USBD_HID_Transfer_And_Keyboard | √ | √ |
| USBD_HID_Transfer_And_MSC | √ | √ |
| USBD_HID_Transfer_CTRL | √ | √ |
| USBD_Mass_Storage_CDROM | √ | √ |
| USBD_Mass_Storage_Flash | √ | √ |
| USBD_Mass_Storage_SRAM | √ | √ |

| | | |
|---|---|---|
| USBD_Micro_Printer | √ | √ |
| USBD_Printer_And_HID_Transfer | √ | √ |
| USBD_VCOM_And_HID_Keyboard | √ | √ |
| USBD_VCOM_And_HID_Transfer | √ | √ |
| USBD_VCOM_and_MassStorage | √ | √ |
| USBD_VCOM_DualPort | √ | √ |
| USBD_VCOM_SerialEmulator | √ | √ |
| USBH_AudioClass | √ | √ |
| USBH_DEV_CONN | √ | √ |
| USBH_HID | √ | √ |
| USBH_HID_KEYBOARD | √ | √ |
| USBH_HID_Mouse_Keyboard | √ | √ |
| USBH_MassStorage | √ | √ |
| USBH_UAC_HID | √ | √ |
| USBH_VCOM | √ | √ |
| USCI_I2C_EEPROM | √ | √ |
| USCI_I2C_Loopback | √ | - |
| USCI_I2C_Loopback_10bit | √ | - |
| USCI_I2C_Master | √ | √ |
| USCI_I2C_Master_10bit | √ | √ |
| USCI_I2C_Monitor | √ | √ |
| USCI_I2C_MultiBytes_Master | √ | √ |
| USCI_I2C_SingleByte_Master | √ | √ |
| USCI_I2C_Slave | √ | √ |
| USCI_I2C_Slave_10bit | √ | √ |
| USCI_I2C_Wakeup_Slave | √ | √ |
| USCI_SPI_Loopback | √ | √ |
| USCI_SPI_MasterMode | √ | √ |
| USCI_SPI_PDMA_LoopTest | √ | - |
| USCI_SPI_SlaveMode | √ | √ |

| | | |
|---|---|---|
| **USCI_UART_AutoBaudRate** | √ | √ |
| **USCI_UART_Autoflow** | √ | √ |
| **USCI_UART_PDMA** | √ | √ |
| **USCI_UART_RS485** | √ | √ |
| **USCI_UART_TxRxFunction** | √ | √ |
| **USCI_UART_Wakeup** | √ | √ |
| **WDT_TimeoutWakeupAndReset** | √ | √ |
| **WWDT_ReloadCounter** | √ | √ |

## 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.**