

## M0A21 Series CMSIS BSP Guide

Directory Introduction for 32-bit NuMicro® Family

### Directory Information

<b>Document</b>	Driver reference guide and revision 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 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)

## TABLE OF CONTENTS

<b>1</b>	<b>DOCUMENT.....</b>	<b>3</b>
<b>2</b>	<b>LIBRARY .....</b>	<b>4</b>
<b>3</b>	<b>SAMPLECODE .....</b>	<b>5</b>
<b>4</b>	<b>SAMPLECODE\ISP .....</b>	<b>6</b>
<b>5</b>	<b>SAMPLECODE\STDDRIVER.....</b>	<b>7</b>
	System Manager (SYS) .....	7
	Clock Controller (CLK).....	7
	Flash Memory Controller (FMC).....	7
	General Purpose I/O (GPIO).....	7
	PDMA Controller (PDMA) .....	8
	Timer Controller (TIMER).....	8
	Watchdog Timer (WDT) .....	9
	Window Watchdog Timer (WWDT) .....	9
	PWM Generator and Capture Timer (PWM).....	9
	UART Interface Controller (UART).....	10
	Universal Serial Control Interface Controller - I <sup>2</sup> C Mode (USCI-I2C) .....	10
	Universal Serial Control Interface Controller - SPI Mode (USCI-SPI).....	11
	Universal Serial Control Interface Controller - UART Mode (USCI-UART) .....	12
	Controller Area Network Converter (CAN) .....	12
	CRC Controller (CRC) .....	13
	Hardware Divider (HDIV).....	13
	Analog-to-Digital Converter (ADC) .....	13
	Digital-to-Analog Converter (DAC) .....	14
	Analog Comparator Controller (ACMP) .....	14

## 1 Document

<b>CMSIS.html</b>	Document of CMSIS version 5.1.1.
<b>NuMicro M0A21 Series CMSIS BSP Driver Reference Guide.chm</b>	This document describes the usage of drivers in M0A21 BSP.
<b>NuMicro M0A21 Series CMSIS BSP Revision History.pdf</b>	This document shows the revision history of M0A21 BSP.

## 2 Library

<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 SampleCode

<b>Hard_Fault_Sample</b>	<p>Show hard fault information when hard fault happened.</p> <p>The hard fault handler show some information included program counter, which is the address where the processor was executing when the hard fault occurs. The listing file (or map file) can show what function and instruction that was.</p> <p>It also shows the Link Register (LR), which contains the return address of the last function call. It can show the status where CPU comes from to get to this point.</p>
<b>ISP</b>	Sample codes for In-System-Programming.
<b>Semihost</b>	Show how to print and get character through IDE console window.
<b>StdDriver</b>	Sample code to demonstrate the usage of M0A21 series MCU peripheral driver APIs.
<b>Template</b>	A project template for M0A21 series MCU.

## 4 SampleCode\ISP

ISP_RS485	In-System-Programming Sample code through RS485 interface.
ISP_UART	In-System-Programming Sample code through UART interface.
ISP_USCI_I2C	In-System-Programming Sample code through USCI I2C interface.

## 5 SampleCode\StdDriver

### System Manager (SYS)

<b>SYS_BODWakeup</b>	Demonstrate how to wake up system from Power-down mode by brown-out detector interrupt.
<b>SYS_PowerDown_MinCurrent</b>	Demonstrate how to minimize power consumption when entering power down mode.
<b>SYS_TrimHIRC</b>	Demonstrate how to use LXT to trim HIRC.

### Clock Controller (CLK)

<b>CLK_ClockDetector</b>	Demonstrate the usage of clock fail detector and clock frequency range detector function.
--------------------------	---

### Flash Memory Controller (FMC)

<b>FMC_CRC32</b>	Demonstrate how to use FMC CRC32 ISP command to calculate the CRC32 checksum of APROM and LDROM.
<b>FMC_IAP</b>	Demonstrate FMC 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.
<b>FMC_RW</b>	Show FMC read Flash IDs, erase, read, and write functions.

### General Purpose I/O (GPIO)

<b>GPIO_EINTAndDebounce</b>	Show the usage of GPIO external interrupt function and de-bounce function.
<b>GPIO_INT</b>	Show the usage of GPIO interrupt function.
<b>GPIO_OutputInput</b>	Show how to set GPIO pin mode and use pin data input and output control.

<b>GPIO_PowerDown</b>	Show how to wake up system from Power-down mode by GPIO interrupt.
-----------------------	--

## PDMA Controller (PDMA)

<b>PDMA_BasicMode</b>	Use PDMA channel 1 to transfer data from memory to memory.
<b>PDMA_ScatterGather</b>	Use PDMA channel 1 to transfer data from memory to memory by scatter-gather mode.
<b>PDMA_ScatterGather_PingPongBuffer</b>	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).

## Timer Controller (TIMER)

<b>TIMER_ACMPTrigger</b>	Use ACMP to trigger Timer0 counter reset mode.
<b>TIMER_CaptureCounter</b>	Show how to use the Timer capture function to capture Timer counter value.
<b>TIMER_Delay</b>	Demonstrate the usage of TIMER_Delay API to generate a 1 second delay.
<b>TIMER_EventCounter</b>	Use TM0 pin to demonstrate Timer event counter function.
<b>TIMER_FreeCountingMode</b>	Use the timer TM0_EXT pin to demonstrate timer free counting mode function, and display the measured input frequency to UART console.
<b>TIMER_InterTimerTriggerMode</b>	Use the timer TM0 pin to demonstrate inter timer trigger mode function, and display the measured input frequency to UART console.
<b>TIMER_Periodic</b>	Use the Timer periodic mode to generate Timer interrupt every 1 second.
<b>TIMER_PeriodicINT</b>	Implement Timer counting in periodic mode.
<b>TIMER_TimeoutWakeup</b>	Use timer to wake up system from Power-down mode periodically.



<b>TIMER_ToggleOut</b>	Demonstrate the Timer0 toggle out function on TM0 pin.
------------------------	--

## Watchdog Timer (WDT)

<b>WDT_TimeoutWakeupAndReset</b>	Implement WDT time-out interrupt event to wake up system and generate time-out reset system event while WDT time-out reset delay period expired.
----------------------------------	--

## Window Watchdog Timer (WWDT)

<b>WWDT_ReloadCounter</b>	Show how to reload the WWDT counter value.
---------------------------	--

## PWM Generator and Capture Timer (PWM)

<b>PWM_240KHz_DutySwitch</b>	Demonstrate how to set PWM0 channel 0 output 240 kHz waveform and switch duty in each 0.5%.
<b>PWM_Capture</b>	Capture the PWM Channel 2 waveform by PWM Channel 0.
<b>PWM_DeadZone</b>	Demonstrate how to use PWM Dead Time function.
<b>PWM_DoubleBuffer_PeriodLoadingMode</b>	Change duty cycle and period of output waveform by PWM Double Buffer function.
<b>PWM_DutySwitch</b>	Change duty cycle of output waveform by configured period.
<b>PWM_OutputWaveform</b>	Demonstrate how to use PWM counter output waveform.
<b>PWM_PDMA_Capture</b>	Capture the PWM0 Channel 0 waveform by PWM0 Channel 2, and use PDMA to transfer captured data.
<b>PWM_PDMA_Capture_1MHzSignal</b>	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.
<b>PWM_SyncStart</b>	Demonstrate how to use PWM counter synchronous start function.

## UART Interface Controller (UART)

<b>UART_AutoBaudRate</b>	Show how to use auto baud rate detection function.
<b>UART_AutoFlow</b>	Transmit and receive data using auto flow control.
<b>UART_IrDA</b>	Transmit and receive UART data in UART IrDA mode.
<b>UART_LIN</b>	Demonstrate how to send data to LIN bus.
<b>UART_LIN_Wakeup</b>	Demonstrate how to wake up a device by LIN Wake-up Signal on LIN bus.
<b>UART_PDMA</b>	Demonstrate UART transmit and receive function with PDMA.
<b>UART_RS485</b>	Transmit and receive data in UART RS485 mode.
<b>UART_SingleWire</b>	Transmit and receive data in UART single-wire mode.
<b>UART_TxRxFunction</b>	Transmit and receive data from PC terminal through RS232 interface.
<b>UART_Wakeup</b>	Show how to wake up system from Power-down mode by UART interrupt.

## Universal Serial Control Interface Controller - I<sup>2</sup>C Mode (USCI-I2C)

<b>USCI_I2C_EEPROM</b>	Show how to use USCI_I2C interface to access EEPROM.
<b>USCI_I2C_Loopback</b>	Show a Master how to access 7-bit address Slave (loopback).
<b>USCI_I2C_Loopback_10bit</b>	Show a Master how to access 10-bit address Slave (loopback).
<b>USCI_I2C_Master</b>	Show how a master accesses a slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_Master_10bit</b>	Show how a master accesses a 10-bit address slave. This sample code need works with USCI_I2C_Slave_10bit.

<b>USCI_I2C_MultiBytes_Master</b>	Show how to set USCI_I2C Multi bytes API Read and Write data to Slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_SingleByte_Master</b>	Show how to use USCI_I2C Single byte API Read and Write data to Slave. This sample code needs to work with USCI_I2C_Slave.
<b>USCI_I2C_Slave</b>	Show how a slave receives data from a master. This sample code needs to work with USCI_I2C_Master.
<b>USCI_I2C_Slave_10bit</b>	Show how a 10-bit address slave receives data from a master. This sample code need works with USCI_I2C_Master_10bit.
<b>USCI_I2C_Wakeup_Slave</b>	Show how to wake up USCI_I2C from Deep Sleep mode. This sample code needs to work with USCI_I2C_Master.

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

<b>USCI_SPI_Loopback</b>	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.
<b>USCI_SPI_MasterMode</b>	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.
<b>USCI_SPI_PDMA_LoopTest</b>	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.
<b>USCI_SPI_SlaveMode</b>	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.

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

<b>USCI_UART_AutoBaudRate</b>	Show how to use auto baud rate detection function.
<b>USCI_UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. This sample code needs to work with USCI_UART_Autoflow_Slave.
<b>USCI_UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. This sample code needs to work with USCI_UART_Autoflow_Master.
<b>USCI_UART_PDMA</b>	This is a USCI_UART PDMA demo and needs to connect USCI_UART TX and RX.
<b>USCI_UART_RS485_Master</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Slave.
<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Master.
<b>USCI_UART_TxRxFunction</b>	Transmit and receive data from PC terminal via the RS232 interface.
<b>USCI_UART_Wakeup</b>	Show how to wake up system from Power-down mode by USCI interrupt in UART mode.

## Controller Area Network Converter (CAN)

<b>CAN_BasicMode_Rx</b>	Demonstrate CAN bus receive a message with basic mode.
<b>CAN_BasicMode_Tx</b>	Demonstrate CAN bus transmit a message with basic mode.
<b>CAN_NormalMode_RX</b>	Demonstrate CAN bus receive a message with normal mode.
<b>CAN_NormalMode_TX</b>	Demonstrate CAN bus transmit a message with normal mode.

## 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.

## Hardware Divider (HDIV)

HDIV	Demonstrate how to divide two signed integers by HDIV engine.
------	---

## Analog-to-Digital Converter (ADC)

ADC_1411ksps_ContinuousScan Mode	Demonstrate how to use HIRC as ADC clock source to achieve 1411 ksps ADC conversion rate.
ADC_ADINT_Trigger	Use ADINT interrupt to do the ADC Single-cycle scan conversion.
ADC_BandGap	Convert Band-gap (channel 29) and print conversion result.
ADC_BandGapCalculateAVDD	Demonstrate how to calculate battery voltage (AVdd) by using band-gap.
ADC_BurstMode	Perform A/D Conversion with ADC burst mode.
ADC_ContinuousScanMode	Perform A/D Conversion with ADC continuous scan mode.
ADC_PDMA_PWM_Trigger	Demonstrate how to trigger ADC by PWM and transfer conversion data by PDMA.
ADC_PWM_Trigger	Demonstrate how to trigger ADC by PWM.
ADC_ResultMonitor	Monitor the conversion result of channel 2 by the digital compare function.

<b>ADC_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan mode.
<b>ADC_SingleMode</b>	Perform A/D Conversion with ADC single mode.
<b>ADC_STADC_Trigger</b>	Show how to trigger ADC by STADC pin.
<b>ADC_SwTrg_Trigger</b>	Trigger ADC by writing ADC software trigger register.
<b>ADC_TempSensor</b>	Convert temperature sensor (channel 30) and print conversion result.
<b>ADC_Timer_Trigger</b>	Show how to trigger ADC by Timer.

## Digital-to-Analog Converter (DAC)

<b>DAC_PDMA_TimerTrigger</b>	Show how Timer triggers DAC to fetch data with PDMA and convert sine wave outputs.
<b>DAC_SoftwareTrigger</b>	Demonstrate how software triggers DAC to convert sine wave outputs.
<b>DAC_TimerTrigger</b>	Demonstrate how Timer triggers DAC to convert sine wave outputs.

## Analog Comparator Controller (ACMP)

<b>ACMP_ComapreDAC</b>	Demonstrate how ACMP compare DAC output with ACMP1_N1 value.
<b>ACMP_Wakeup</b>	Use ACMP to wake up system from Power-down mode while comparator output changes.
<b>ACMP_WindowCompare</b>	Show how to monitor ACMP input with window compare function.
<b>ACMP_WindowLatch</b>	Demonstrate how to use ACMP window latch mode.

### **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.*