M2U51 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.

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

TABLE OF CONTENTS

1 Document

CMSIS.html	Document of CMSIS version 6.1.0.
NuMicro M2U51 Driver Reference Guide.chm	This document describes the usage of drivers in M2U51 BSP.
NuMicro M2U51 Series CMSIS BSP Revision History.pdf	This document shows the revision history of M2U51 BSP.

2 Library

CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) V6.1.0 definitions by Arm® Corp.
Device	CMSIS compliant device header file.
StdDriver	All peripheral driver header and source files.

3 SampleCode

Hard_Fault_Sample	Show hard fault information when hard fault happened. The hard fault handler shows 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. 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.
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 M2U51 series MCU peripheral driver APIs.
Template	A project template for M2U51 series MCU.

4 SampleCode\ISP

ISP_I2C	In-System-Programming Sample code through I^{2}C 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\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_WakeupVTOR	Show how to continue executing code after wake-up from SPD Power-down mode by VTOR function.

6 SampleCode\StdDriver

System Manager (SYS)

SYS_BODWakeup	Demonstrate how to wake up system from Power-down mode by brown-out detector interrupt.
SYS_MIRCClockOutput	Change system clock to different MIRC 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_TrimMIRC	Demonstrate how to use LXT to trim MIRC.

Flash Memory Controller (FMC)

FMC_CRC32	Demonstrate how to use FMC CRC32 ISP command to calculate the CRC32 checksum of APROM and LDROM.
FMC_ExeInSRAM	Implement a code and execute it in SRAM to program embedded Flash.
FMC_IAP	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.
FMC_MultiBoot	Implement a multi-boot system to boot from different applications in APROM or LDROM by VECMAP.
FMC_MultiWordProgram	Show how to use FMC multi-word (word line) program flow to program APROM and LDROM.
FMC_ReadAllOne	Demonstrate how to use FMC Read-All-One ISP command to verify APROM or LDROM pages are all 0xFFFFFFFF or not.
FMC_RW	Show FMC read Flash IDs, erase, read, and write functions.

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 2 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 0 get/clear timeout flag with UART1.

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_InterTimerMode	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_Time_Display	Demonstrate the RTC function and displays current time to the UART console.

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.

Basic Pulse Width Modulation Controller (BPWM)

BPWM_Capture	Capture the BPWM1 Channel 0 waveform by BPWM1 Channel 2.
BPWM_DoubleBuffer	Change duty cycle and period of output waveform by

	BPWM Double Buffer function.
BPWM_OutputWaveform	Demonstrate how to use BPWM counter output waveform.
BPWM_SwitchDuty	Change duty cycle of output waveform by configured period.
BPWM_SyncStart	Demonstrate how to use BPWM 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_AutoOperation	Demonstrate how to enable UART Auto-operation mode to transmit and receive data when chip enters Power-down mode.
UART_IrDA	Transmit and receive UART data in UART IrDA 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_AutoOperation	Demonstrate how to do SPI loopback test in Auto-operation mode when chip enters power-down mode.
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. 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.

I^{2}C Serial Interface Controller (I^{2}C)

I2C_AutoOperation	Demonstrate I^{2}C Auto-operation mode when chip enters Power-down mode. This sample code needs to work with I2C_Slave.
I2C_Double_Buffer_Slave	Demonstrate how to set I^{2}C 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 I^{2}C interface to access EEPROM.
I2C_Loopback	Demonstrate how to set I^{2}C 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 I^{2}C Multi bytes API Read and Write data to Slave. This sample code needs to work with I2C_Slave.
I2C_PDMA_TRX	Demonstrate I^{2}C PDMA mode and need to connect I2C0 (master) and I2C1 (slave).
I2C_SingleByte_Master	Show how to use I^{2}C 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. 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_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.

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.

Cryptographic Accelerator (CRYPTO)

CRYPTO_AES	Show Crypto IP AES-128 ECB mode encrypt/decrypt function.

Analog-to-Digital Converter (ADC)

ADC_ADINT_Trigger	Use ADINT interrupt to do the ADC continuous scan conversion.
ADC_BandGap	Convert Band-gap and print conversion result.
ADC_BandGapCalculateAVDD	Demonstrate how to calculate battery voltage(AVdd) by using band-gap.
ADC_BPWM_Trigger	Demonstrate how to trigger ADC by BPWM.
ADC_BurstMode	Perform A/D Conversion with ADC burst mode.
ADC_ContinuousScanMode	Perform A/D Conversion with ADC continuous scan mode.
ADC_MaxSPS	Demonstrate how to use HIRC as ADC clock source to achieve maximum SPS ADC conversion rate.
ADC_PDMA_BPWM_Trigger	Demonstrate how to trigger ADC by BPWM and transfer conversion data by PDMA.
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.
ADC_SingleCycleScanMode	Perform A/D Conversion with ADC single cycle scan mode.
ADC_SingleCycleScanMode2	Perform A/D Conversion with ADC single mode to simulate single cycle scan mode.
ADC_SingleMode	Perform A/D Conversion with ADC single mode.
ADC_STADC_Trigger	Show how to trigger ADC by STADC pin.
ADC_SwTrg_Trigger	Trigger ADC by writing ADC software trigger register.
ADC_TempSensor	Convert temperature sensor and print conversion result.
ADC_Timer_Trigger	Show how to trigger ADC by Timer.

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.

Liquid Crystal Display (LCD)

LCD_Blinking	The LCD displays the text with a blinking effect.
LCD_Pixel_OnOff	Configure the specified pixels to be either ON or OFF.
LCD_Print_Text	Display the specified text on the LCD screen.

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.