# MS51 Series BSP User Guide

Based on

Keil uVision5 and PK51 Development Kit V9.52

IAR Embedded Workbench 8051 V10.10.1

NuEclipse_Windows (For NuMicro 8051) V1.02.022

For NuMicro® 8051 Family

## Directory Information

Please extract the “MS51_Series_BSP_V2.00.000.zip” file firstly and confirm the following content of this BSP folder.

| | |
|---|---|
| **MS51DA9AE_MS51BA9AE_ MS51IA9AE** | BSP for MS51DA9AE and MS51BA9AE.<br>8KB Flash APROM share with 4KB LDROM<br>256 Byte RAM, 1024 Byte XRAM<br>In TSSOP14 , MSOP10 and SOP8 package |
| **MS51FB9AE_MS51XB9AE_ MS51XB9BE** | BSP for MS51FB9AE, MS51XB9AE and MS51XB9BE.<br>16KB Flash APROM share with 4KB LDROM<br>256 Byte RAM, 1024 Byte XRAM<br>In TSSOP20 and QFN20 package |
| **MS51FC0AE_MS51XC0BE_ MS51EB0AE_MS51EC0AE_ MS51TC0AE_MS51PC0AE** | BSP for MS51FC0AE, MS51XC0BE, MS51EC0AE, MS51TC0AE, MS51PC0AE and MS51EC0AE.<br>32KB Flash APROM share with 4KB LDROM<br>256 Byte RAM, 2048 Byte XRAM<br>In QFN33, LQFP32, TSSOP28, TSSOP20 and QFN20 package |
| **Document** | Driver reference manual and revision history. |
| Each folder listed above with following content folders | |
| **Library** | Device driver header and source files. |
| **SampleCode** | Driver sample code. |

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

www.nuvoton.com

# 1 .\Document\

| | |
|---|---|
| **Nuvoton_MS51_Series_BSP_Revision_History.pdf** | This document shows the revision history of MS51 BSP for KEIL/IAR/NuEclipse. |
| **NuEclipse SDCC quick start_EN_V1.02.pdf** | The English version in NuEclipse tool, how to create a project, how to debug, and a quick guide to frequently used features in the debugging environment. |
| **NuEclipse SDCC quick start_SC_V1.02.pdf** | The Chinese version in NuEclipse tool, how to create a project, how to debug, and a quick guide to frequently used features in the debugging environment. |
| **Nuvoton_8051_BSP_QuickStart.pdf** | Settings and instructions for creating a project in Keil, IAR, and NuEclipse, respectively. |
| **Nuvoton_8051_WDT_reset_EN.pdf** | Common questions and suggestions for enabling and disabling the Watchdog timer. |

# 2 .\Library\

| | |
|---|---|
| **Device** | MS51 Device header file |
| **Startup** | A51 startup file and executable file |
| **StdDriver** | All peripheral driver header and source files. |

# 3 .\SampleCode\

| | |
|---|---|
| **ISP** | Standard ISP bootloader source code for ISP programmer. |
| **PowerManagement** | The power consumption setting samples. |
| **RegBased** | Demonstrate the usage of MS51 series MCU peripheral driver. |
| **Template** | A project template for MS51 series MCU. |

# 4 .\SampleCode\ISP

| | |
|---|---|
| **ISP_I2C** | In alignment with the Nuvoton Standard ISP Programmer Tool protocol, ISP standard example executed through the I2C communication port, primarily for writing to LDROM. |
| **ISP_UART0** | In alignment with the Nuvoton Standard ISP Programmer Tool protocol, ISP standard example executed through the UART0 communication port, primarily for writing to LDROM. |
| **ISP_UART1** | In alignment with the Nuvoton Standard ISP Programmer Tool protocol, ISP standard example executed through the UART1 communication port, primarily for writing to LDROM. |

# 5 .\SampleCode\PowerManagement

| | |
|---|---|
| **Idle_Current** | The MS51 series features a sample program designed to run in idle mode, allowing for the measurement of power consumption in this state. |
| **PowerDown_MinCurrent** | A sample program for the MS51 series is configured to operate in Power Down mode, facilitating the measurement of power consumption in this specific state. It's worth noting that the power consumption value attained in Power Down mode represents the lowest achievable power consumption state for the MS51 series. |

# 6 .\SampleCode\RegBased

| | |
|---|---|
| **ADC_Bandgap_VDD** | Calculate the real $V_{DD}$ value of the device system based on the difference between the pre-stored ADC conversion result values when $V_{DD}$ is 3.072V and the system converted band-gap value. |
| **ADC_Compare_Mode0** | Example of ADC compare mode: When the CONFIG is setting to comparison mode 0, the conditions for generating comparison interrupts mirror those of the N76E003 series product feature. |
| **ADC_Compare_Mode1** | Example of ADC compare mode: When the CONFIG is setting to comparison mode 0, the conditions for generating comparison interrupts mirror those of the MS51 series product feature. This mode is also the default mode. |
| **ADC_GPIO _Trig** | Demonstrate how to use GPIO to start ADC initial setting and show the conversion result in ADCRH and ADCRL register. |
| **ADC_Multi_channel** | Demonstrate how to regularly sample from different ADC input channel. |
| **ADC_PWM_Trig** | Demonstrate how to use each of PWM timer period timeout to trigger ADC conversion. |
| **ADC_Simple** | Start ADC conversion by triggering ADCS bit [ADCCON0.6] and check the flag register ADCF bit [ADCCON0.7] to confirm if a conversion is finished. |
| **Fsys_ModifyHIRC** | Call the library fie “sys.c” to modify system setting as 16 MHz or 24 MHz and check clock out pin to confirm if Fsys is modified. The MS51 HIRC can be selected within 16 MHz or 24 MHz. |
| **Fsys_Select_ECLK** | Change the MS51 system clock from HIRC to external clock initial setting. The MS51 external clock input ranges from 4 MHz to 24 MHz. |
| **Fsys_Select_LIRC** | Change the MS51 system clock from HIRC to LIRC initial setting. The MS51 internal LIRC value is 10 khz. Please note that all simulation runs in this mode will become very slow. |
| **GPIO_ClockOut** | Show the MS51 system clock and output from CLKO pin. |

| | |
|---|---|
| **GPIO_Input_Output** | Toggle each MS51 GPIO pin output from high to low after 200ms delay. |
| **GPIO_Pin_Interrupt** | GPIO pin interrupt example: Utilize the PIT interrupt vector for response. Demonstrate how to wake up MS51 from Power-down mode through external interrupt input by enabling MS51 pin interrupt function. |
| **I2C_EEPROM_Master_Interrupt** | Show how to use MS51 as master to read external connect EEPROM by I2C bus. As master, use interrupt to read. |
| **I2C_EEPROM_Master_Polling** | Show how to use MS51 as master to read external connect EEPROM by I2C bus. The master uses segmented polling to complete the entire process. |
| **I2C_Master** | Combine it with the I2C_Slave project to create a comprehensive I2C transmission master/slave protocol integration. |
| **I2C_Slave** | Combine it with the I2C_Master project to create a comprehensive I2C transmission master/slave protocol integration. |
| **IAP_AP_program_AP** | Demonstrate how MS51 APROM is used as Data Flash to implement erase / program / read verify function. All APROM memory can be used as Data Flash. |
| **IAP_AP_program_LD** | Demonstrate how MS51 IAP runs in APROM to program LDROM and implement erase / program / read verify function. User first needs to confirm if the LDROM is enabled through CONFIG setting. |
| **IAP_Dataflash_EEPROM** | Simulate Data Flash (APROM) as EEPROM mode by calling the library file “eeprom.c”. This process includes copy one page of Data Flash values in RAM, modify data, erase Data Flash, then copy new values from RAM to Data Flash. |
| **IAP_Dataflash_EEPROM_SPROM** | Simulate SPROM as EEPROM mode by calling the library file “eeprom_sprom.c”. This process includes copy one page of Data Flash values in RAM, modify data, erase Data Flash, then copy new values from RAM to Data Flash. |
| **IAP_LD_Program_AP** | Demonstrate how MS51 IAP runs in LDROM to program APROM and implement erase / program / read verify function. |
| **IAP_program_CONFIG** | Demonstrate using MS51 IAP command to modify CONFIG bytes. |

| | |
|---|---|
| **IAP_Read_UCID** | Demonstrate using MS51 IAP command to read the unique customer ID (UCID). Only for customer special order MS51 MCU. One UCID is only for one customer. |
| **IAP_Read_UID** | Demonstrate using MS51 IAP command to read the Unique code of MS51. The UID value of each MS51 is different. |
| **INT0_Ext_Interrupt** | Perform MS51 external interrupt pin INT0 enabled initial setting. |
| **INT1_Ext_Interrupt** | Perform MS51 external interrupt pin INT1 enabled initial setting. |
| **Interrupt_ISR_all** | List all interrupt sector in the library file “isr.c”. Call this file to implement interrupt subroutine. |
| **PWM0_DeadTime** | Configure PWM as Complementary mode. Control 3 pairs output, set each 2 channel PWM output as same duty and insert dead time. |
| **PWM0_Independent_Reload** | Configure PWM as independent mode. Each PWM channel outputs independently and each PWM channels output with different duty and interrupt enabled. |
| **PWM0_Synchronous** | Configure PWM as Synchronous mode. Each PWM0 channel 0/2/4 outputs different duty and PWM0 channel 1/3/5 duty following 0/2/4 setting. |
| **SPI_Flash_Read_Write** | Connect MS51 with W25Q16BV SPI Flash and set it as master to read and write data sample code. |
| **SPI_Master** | Combine it with the SPI_Slave_Interrupt or SPI_Slave_Polling project to create a comprehensive SPI transmission master/slave protocol integration. |
| **SPI_Slave_Interrupt** | Combine it with the SPI_master project to create a comprehensive SPI transmission master/slave protocol integration. SPI slave use interrupt subroutine to return value to master |
| **SPI_Slave_Polling** | Combine it with the SPI_master project to create a comprehensive SPI transmission master/slave protocol integration. SPI slave use polling wait flag to return value to master |
| **Timer0_Interrupt** | Call the library file "timer.c" for timer 0, reloading the delay settings with interrupt enablement according to the initial configuration. |

| | |
|---|---|
| **Timer1_ Interrupt** | Call the library file "timer.c" for timer 1, reloading the delay settings with interrupt enablement according to the initial configuration. |
| **Timer2_AutoReload_Capture** | Configure Timer 2 as one channel input capture with interrupt enabled initial setting. Timer 2 capture interrupt vector is different to the Timer 2 overflow interrupt. |
| **Timer2_Interrupt** | Configure Timer 2 as auto reload delay setting with interrupt enabled initial setting. |
| **Timer3_Interrupt** | Configure Timer 3 as auto reload mode initial setting and interrupt enabled. |
| **UART0_Interrupt_RW** | Configure UART0 transfer including transmit and receive with interrupt enabled. |
| **UART0_Printf** | Loop transmit from UART0 TXD pin initial setting with printf function API. |
| **UART1_Interrupt_RW** | Configure UART1 transmit and receive initial setting and enable interrupt subroutine. |
| **UART0_Printf** | Special include putchar.c or sdcc_stdio.c for printf function redirection to UART1 TXD pin. initial setting with printf function API. |
| **WakeupTimer_Interrupt** | Enable wake-up timer with interrupt function. Main loop enters Power-down mode after initial setting, and once WKT timeout, MS51 will wake up and then jump into interrupt subroutine to toggle GPIO output. |
| **Watchdog_Interrupt** | Demonstrate Watchdog Timer (WDT) initial setting with interrupt enabled and Watchdog Timer reset function disabled. The WDT counter overflow will jump into WDT interrupt subroutine. |
| **Watchdog_Reset** | Demonstrate Watchdog Timer (WDT) initial setting with reset function enabled in CONFIG. |

Based on the features of the different products these projects is not necessarily included in folder ..\SampleCode\RegBased .

# 1 REVISION HISTORY

| Date | Revision | Description |
|---|---|---|
| 2023.05.23 | 1.00 | Initial release. |
| 2025.03.10 | 2.00 | Collaborate with the MS51 BSP to consolidate KEIL/IAR/NuEclipse projects into a unified package for publication. |

## 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.**