NUC122 Board Supporting Package Directory Introduction

Rev.3.00.002

Directory Information

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

Document Information

BSP Revision History	Show all the revision history about specific BSP.
Driver Reference Guide	Describe the definition, input and output of each API.

Library Information

CMSIS	CMSIS definitions by ARM® Corp.
Device	CMSIS compliant device header file.
StdDriver	All peripheral driver header and source files.

Sample Code Information

\SampleCode\Hard_Fault_ Sample	Show hard fault information when hard fault happened.
\SampleCode\Template	Software Development Template.
\SampleCode\Semihost	Show how to debug with semi-host message print.
\SampleCode\RegBased	The sample codes which access control registers directly.
\SampleCode\StdDriver	NUC122 Driver Samples

\SampleCode\RegBased

FMC_RW	Show how to read/program embedded flash by ISP function.
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/output control.
GPIO_PowerDown	Show how to wake up system from Power-down mode by GPIO interrupt.
I2C_EEPROM	Show how to use I^2C interface to access EEPROM.
I2C_GCMode_Master	Show how a Master uses I^2C address 0x0 to write data to Slave. This sample code needs to work with I2C_GCMode_Slave.
I2C_GCMode_Slave	Show a Slave how to receive data from Master in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.
I2C_Master	Show a Master how to access Slave. This sample code needs to work with I2C_Slave.
I2C_Slave	Show how to set I^2C in Slave mode and receive the data from Master. This sample code needs to work with I2C_Master.
PS2	Show how to control PS/2 mouse movement on the screen.
PWM_Capture	Capture the PWMA Channel 1 waveform by PWMA Channel 0.
PWM_DeadZone	Demonstrate how to use PWM Dead Zone function.
PWM_DoubleBuffer	Change duty cycle and period of output waveform by PWM Double Buffer function.
RTC_PowerDown	Use RTC alarm interrupt event to wake-up system.
RTC_TimeAndTick	Get the current RTC data/time per tick.

SPI_Loopback	Implement SPI Master loop back transfer. This sample code needs to connect MISO00 pin and MOSI00 pin together. It will compare the received data with transmitted data.
SPI_MasterMode	Configure SPI0 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device. This sample code needs to work with SPI_SlaveMode sample code.
SPI_SlaveMode	Configure SPI0 as Slave mode and demonstrate how to communicate with an off-chip SPI Master device. This sample code needs to work with SPI_MasterMode sample code.
SYS	Change system clock to different PLL frequency.
TIMER_Counter	Implement timer1 event counter function to count the external input event.
TIMER_PeriodicINT	Implement timer counting in periodic mode.
UART_Autoflow_Master	Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Slave.
UART_Autoflow_Slave	Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Master.
UART_IrDA_Master	Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Slave.
UART_IrDA_Slave	Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Master.
UART_RS485_Master	Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Slave.
UART_RS485_Slave	Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Master.
UART_TxRx_Function	Transmit and receive data from PC terminal through RS232 interface.
UART_Wakeup	Show how to wake up system form Power-down mode by UART interrupt.

WDT_PowerDown	Use WDT time-out interrupt event to wake-up system.
WDT_TimeoutINT	Implement periodic WDT time-out interrupt event.
WDT_TimeoutReset	Show how to generate time-out reset system event while WDT time-out reset delay period expired.

\SampleCode\StdDriver

FMC_RW	Show how to read/program embedded flash by ISP function.
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/output control.
GPIO_PowerDown	Show how to wake up system from Power-down mode by GPIO interrupt.
I2C_EEPROM	Show how to use I^2C interface to access EEPROM.
I2C_GCMode_Master	Show how a Master uses I^2C address 0x0 to write data to Slave. This sample code needs to work with I2C_GCMode_Slave.
I2C_GCMode_Slave	Show a Slave how to receive data from Master in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.
I2C_Master	Show a Master how to access Slave. This sample code needs to work with I2C_Slave.

I2C_Slave	Show how to set I^2C in Slave mode and receive the data from Master. This sample code needs to work with I2C_Master.
PS2	Show how to control PS/2 mouse movement on the screen.
PWM_Capture	Capture the PWMA Channel 1 waveform by PWMA Channel 0.
PWM_DeadZone	Demonstrate how to use PWM Dead Zone function.
PWM_DoubleBuffer	Change duty cycle and period of output waveform by PWM Double Buffer function.
RTC_PowerDown	Use RTC alarm interrupt event to wake-up system.
RTC_TimeAndTick	Get the current RTC data/time per tick.
SPI_Loopback	Implement SPI Master loop back transfer. This sample code needs to connect MISO00 pin and MOSI00 pin together. It will compare the received data with transmitted data.
SPI_MasterMode	Configure SPI0 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device. This sample code needs to work with SPI_SlaveMode sample code.
SPI_SlaveMode	Configure SPI0 as Slave mode and demonstrate how to communicate with an off-chip SPI Master device. This sample code needs to work with SPI_MasterMode sample code.
SYS	Change system clock to different PLL frequency.
TIMER_Counter	Implement timer1 event counter function to count the external input event.
TIMER_PeriodicINT	Implement timer counting in periodic mode.
UART_Autoflow_Master	Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Slave.
UART_Autoflow_Slave	Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Master.
UART_IrDA_Master	Transmit and receive data in UART IrDA mode. This sample

	code needs to work with UART_IrDA_Slave.
UART_IrDA_Slave	Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Master.
UART_RS485_Master	Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Slave.
UART_RS485_Slave	Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Master.
UART_TxRx_Function	Transmit and receive data from PC terminal through RS232 interface.
UART_Wakeup	Show how to wake up system form Power-down mode by UART interrupt.
USBD_HID_Keyboard	Show how to implement a USB keyboard device. This sample code 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_Mouse2	Demonstrate how to implement a USB mouse device. It use PC0 ~ PC5 to control mouse direction and mouse key. It also supports USB suspend and remote wakeup.
USBD_VCOM	Implement a USB virtual COM port device. It supports one virtual COM port.
USBD_HID_Transfer	Transfer data between USB device and PC through USB HID interface. A windows tool is also included in this sample code to connect with USB device.
USBD_Billboard	A a sample code to show the implementation of USB Billboard Class.
WDT_PowerDown	Use WDT time-out interrupt event to wake-up system.
WDT_TimeoutINT	Implement periodic WDT time-out interrupt event.
WDT_TimeoutReset	Show how to generate time-out reset system event while

	WDT time-out reset delay period expired.

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.