

# **NUC122 Board Supporting Package Directory Introduction**

Rev.3.00.005

## Directory Information

|                   |   |
|-------------------|---|
| <b>Document</b>   | Driver reference manual and revision history. |
| <b>Library</b>    | Driver header and source files.               |
| <b>SampleCode</b> | Driver sample code.                           |

## Document Information

|                               |  |
|-------------------------------|--|
| <b>BSP Revision History</b>   | Show all the revision history about specific BSP.      |
| <b>Driver Reference Guide</b> | Describe the definition, input and output of each API. |

## Library Information

|                  |  |
|------------------|--|
| <b>CMSIS</b>     | CMSIS definitions by ARM® Corp.                |
| <b>Device</b>    | CMSIS compliant device header file.            |
| <b>StdDriver</b> | All peripheral driver header and source files. |

## Sample Code Information

|                                      |   |
|--------------------------------------|---|
| <b>\SampleCode\Hard_Fault_Sample</b> | Show hard fault information when hard fault happened.     |
| <b>\SampleCode\ISP</b>               | Sample codes for In-System-Programming.                   |
| <b>\SampleCode\Template</b>          | Software Development Template.                            |
| <b>\SampleCode\Semihost</b>          | Show how to debug with semi-host message print.           |
| <b>\SampleCode\RegBased</b>          | The sample codes which access control registers directly. |
| <b>\SampleCode\StdDriver</b>         | NUC122 Driver Samples                                     |

## \SampleCode\ISP

|           |  |
|-----------|--|
| 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 I2C 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.  |

## \SampleCode\RegBased

|                             |   |
|-----------------------------|---|
| <b>FMC_RW</b>               | Show how to read/program embedded flash by ISP function.  |
| <b>GPIO_EINTAndDebounce</b> | Show the usage of GPIO external interrupt function and debounce 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/output control.  |
| <b>GPIO_PowerDown</b>       | Show how to wake up system from Power-down mode by GPIO interrupt.  |
| <b>GPIO_SwDebounce</b>      | Demonstrate how to implement software debounce with GPIO interrupt and timer.   |
| <b>I2C_EEPROM</b>           | Show how to use I <sup>2</sup> C interface to access EEPROM.  |
| <b>I2C_GCMode_Master</b>    | Show how a Master uses I <sup>2</sup> C address 0x0 to write data to Slave. This sample code needs to work with I2C_GCMode_Slave. |
| <b>I2C_GCMode_Slave</b>     | Show a Slave how to receive data from Master in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.    |
| <b>I2C_Master</b>           | Show a Master how to access Slave. This sample code needs to work with I2C_Slave.   |
| <b>I2C_Slave</b>            | Show how to set I <sup>2</sup> C in Slave mode and receive the data from Master. This sample code needs to work with I2C_Master.  |
| <b>PS2</b>                  | Show how to control PS/2 mouse movement on the screen.  |
| <b>PWM_Capture</b>          | Capture the PWMA Channel 1 waveform by PWMA Channel 0.  |
| <b>PWM_DeadZone</b>         | Demonstrate how to use PWM Dead Zone function.  |
| <b>PWM_DoubleBuffer</b>     | Change duty cycle and period of output waveform by PWM Double Buffer function.  |

|                             |   |
|-----------------------------|---|
| <b>RTC_PowerDown</b>        | Use RTC alarm interrupt event to wake-up system.  |
| <b>RTC_TimeAndTick</b>      | Get the current RTC data/time per tick.   |
| <b>SPI_Loopback</b>         | 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. |
| <b>SPI_MasterMode</b>       | 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.      |
| <b>SPI_SlaveMode</b>        | 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.     |
| <b>SYS</b>                  | Change system clock to different PLL frequency.   |
| <b>TIMER_Counter</b>        | Implement timer1 event counter function to count the external input event.  |
| <b>TIMER_PeriodicINT</b>    | Implement timer counting in periodic mode.  |
| <b>UART_Autoflow_Master</b> | Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Slave.  |
| <b>UART_Autoflow_Slave</b>  | Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Master.   |
| <b>UART_IrDA_Master</b>     | Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Slave.   |
| <b>UART_IrDA_Slave</b>      | Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Master.  |
| <b>UART_RS485_Master</b>    | Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Slave.   |
| <b>UART_RS485_Slave</b>     | Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Master.  |
| <b>UART_TxRx_Function</b>   | Transmit and receive data from PC terminal through RS232 interface.   |

|                         |   |
|-------------------------|---|
| <b>UART_Wakeup</b>      | Show how to wake up system form Power-down mode by UART interrupt.                              |
| <b>WDT_PowerDown</b>    | Use WDT time-out interrupt event to wake-up system.   |
| <b>WDT_TimeoutINT</b>   | Implement periodic WDT time-out interrupt event.  |
| <b>WDT_TimeoutReset</b> | Show how to generate time-out reset system event while WDT time-out reset delay period expired. |

## \SampleCode\StdDriver

|                             |   |
|-----------------------------|---|
| <b>FMC_RW</b>               | Show how to read/program embedded flash by ISP function.  |
| <b>GPIO_EINTAndDebounce</b> | Show the usage of GPIO external interrupt function and debounce 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/output control.  |
| <b>GPIO_PowerDown</b>       | Show how to wake up system from Power-down mode by GPIO interrupt.  |
| <b>GPIO_SwDebounce</b>      | Demonstrate how to implement software debounce with GPIO interrupt and timer.   |
| <b>I2C_EEPROM</b>           | Show how to use I <sup>2</sup> C interface to access EEPROM.  |
| <b>I2C_GCMode_Master</b>    | Show how a Master uses I <sup>2</sup> C address 0x0 to write data to Slave. This sample code needs to work with I2C_GCMode_Slave. |
| <b>I2C_GCMode_Slave</b>     | Show a Slave how to receive data from Master in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.    |
| <b>I2C_Master</b>           | Show a Master how to access Slave. This sample code needs to work with I2C_Slave.   |
| <b>I2C_Slave</b>            | Show how to set I <sup>2</sup> C in Slave mode and receive the data from Master. This sample code needs to work with I2C_Master.  |
| <b>PS2</b>                  | Show how to control PS/2 mouse movement on the screen.  |
| <b>PWM_Capture</b>          | Capture the PWMA Channel 1 waveform by PWMA Channel 0.  |
| <b>PWM_DeadZone</b>         | Demonstrate how to use PWM Dead Zone function.  |
| <b>PWM_DoubleBuffer</b>     | Change duty cycle and period of output waveform by PWM Double Buffer function.  |



|                             |   |
|-----------------------------|---|
| <b>RTC_PowerDown</b>        | Use RTC alarm interrupt event to wake-up system.  |
| <b>RTC_TimeAndTick</b>      | Get the current RTC data/time per tick.   |
| <b>SPI_Loopback</b>         | 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. |
| <b>SPI_MasterMode</b>       | 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.      |
| <b>SPI_SlaveMode</b>        | 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.     |
| <b>SYS</b>                  | Change system clock to different PLL frequency.   |
| <b>TIMER_Counter</b>        | Implement timer1 event counter function to count the external input event.  |
| <b>TIMER_PeriodicINT</b>    | Implement timer counting in periodic mode.  |
| <b>UART_Autoflow_Master</b> | Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Slave.  |
| <b>UART_Autoflow_Slave</b>  | Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Master.   |
| <b>UART_IrDA_Master</b>     | Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Slave.   |
| <b>UART_IrDA_Slave</b>      | Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Master.  |
| <b>UART_RS485_Master</b>    | Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Slave.   |
| <b>UART_RS485_Slave</b>     | Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Master.  |
| <b>UART_TxRx_Function</b>   | Transmit and receive data from PC terminal through RS232 interface.   |

|                          |   |
|--------------------------|---|
| <b>UART_Wakeup</b>       | Show how to wake up system form Power-down mode by UART interrupt.  |
| <b>USBD_HID_Keyboard</b> | Show how to implement a USB keyboard device. This sample code supports to use GPIO to simulate key input.   |
| <b>USBD_HID_Mouse</b>    | Show how to implement a USB mouse device. The mouse cursor will move automatically when this mouse device connecting to PC by USB.                          |
| <b>USBD_HID_Mouse2</b>   | 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. |
| <b>USBD_VCOM</b>         | Implement a USB virtual COM port device. It supports one virtual COM port.  |
| <b>USBD_HID_Transfer</b> | 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.          |
| <b>USBD_Billboard</b>    | A a sample code to show the implementation of USB Billboard Class.  |
| <b>WDT_PowerDown</b>     | Use WDT time-out interrupt event to wake-up system.   |
| <b>WDT_TimeoutINT</b>    | Implement periodic WDT time-out interrupt event.  |
| <b>WDT_TimeoutReset</b>  | 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.

---

*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.*