

# **NUC131 Board Supporting Package Directory Introduction**

Rev.3.00.002

## 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\Template</b>	Software Development Template.
<b>\SampleCode\Semihost</b>	Show how to debug with semi-host message print.
<b>\SampleCode\RegBased</b>	The sample code able to access control registers directly.
<b>\SampleCode\StdDriver</b>	Driver Samples

## \SampleCode\RegBased

<b>ADC_ContinuousScanMode</b>	Perform A/D Conversion with ADC continuous scan mode.
<b>ADC_PwmTrigger</b>	Demonstrate how to trigger ADC by PWM.
<b>ADC_ResultMonitor</b>	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>BPWM_Capture</b>	Capture the BPWM1 Channel 0 waveform by BPWM0 Channel 0.
<b>BPWM_DoubleBuffer</b>	Change duty cycle and period of output waveform by BPWM Double Buffer function.
<b>CAN_Set_MaskFilter</b>	Use MaskFilter to receive message in Normal mode. This sample code needs to work with CAN_Test_MaskFilter.
<b>CAN_Test_MaskFilter</b>	Use message object No.1 to send message objects (ID=0x700~0x70F). This sample code needs to work with CAN_Set_MaskFilter.
<b>FMC_IAP</b>	Show how to call LDROM functions from APROM. The code in APROM will look up the table at 0x100E00 to get the address of function of LDROM and call the function.
<b>FMC_MultiBoot</b>	Implement a multi-boot system to boot from different applications in APROM. A LDROM code and 4 APROM code are implemented in this sample code.
<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>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>I2C_Wakeup_Master</b>	Show how to wake up MCU from Power-down. This sample code needs to work with I2C_Wakeup_Slave.
<b>I2C_Wakeup_Slave</b>	Show how to wake up MCU from Power-down mode through I <sup>2</sup> C interface. This sample code needs to work with I2C_Wakeup_Master.
<b>PWM_Capture</b>	Capture the PWM1 Channel 0 waveform by PWM0 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>SPI_Loopback</b>	Implement SPI Master loop back transfer. This sample code needs to connect SPI0_MISO0 pin and SPI0_MOSI0 pin together. It will compare the received data with transmitted data.
<b>SPI_MasterFifoMode</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_SlaveFifoMode sample code.
<b>SPI_SlaveFifoMode</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_MasterFifoMode sample code.
<b>SYS_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLK0 pin.
<b>TIMER_Capture</b>	Show how to use the timer0 capture function to capture timer0 counter value
<b>TIMER_Counter</b>	Implement timer0 event counter function to count the external input event.
<b>TIMER_PeriodicINT</b>	Implement timer counting in periodic mode.
<b>TIMER_PowerDown</b>	Use timer0 periodic time-out interrupt event to wake up system.
<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_LIN</b>	Transmit LIN header and response.
<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.
<b>WWDT_CompareINT</b>	Show how to reload the WWDT counter value.

### **\SampleCode\StdDriver**

<b>ADC_ContinuousScanMode</b>	Perform A/D Conversion with ADC continuous scan mode.
<b>ADC_PwmTrigger</b>	Demonstrate how to trigger ADC by PWM.
<b>ADC_ResultMonitor</b>	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>BPWM_Capture</b>	Capture the BPWM1 Channel 0 waveform by BPWM0 Channel 0.
<b>BPWM_DoubleBuffer</b>	Change duty cycle and period of output waveform by BPWM Double Buffer function.
<b>CAN_BasicMode_Receive</b>	Implement receive message in Basic mode. This sample code needs to work with CAN_BasicMode_Transmit.
<b>CAN_BasicMode_Transmit</b>	Implement transmit message in Basic mode. This sample code needs to work with CAN_BasicMode_Receive.
<b>CAN_NormalMode_Receive</b>	Implement receive message in Normal mode. This sample

	code needs to work with CAN_NormalMode_Transmit.
<b>CAN_NormalMode_Transmit</b>	Implement transmit message in Normal mode. This sample code needs to work with CAN_NormalMode_Receive.
<b>CAN_Wakeup</b>	Show how to wake up system form Power-down mode by detecting a transition.
<b>DAC_PWMTrigger</b>	Demonstrate how to trigger DAC by PWM.
<b>DAC_SoftwareTrigger</b>	Demonstrate how to trigger DAC conversion by software method.
<b>DAC_TimerTrigger</b>	Demonstrate how to trigger DAC by timer.
<b>FMC_IAP</b>	Show how to reboot to LDROM functions from APROM. This sample code set VECMAP to LDROM and reset to re-boot to LDROM.
<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>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>I2C_Wakeup_Master</b>	Show how to wake up MCU from Power-down. This sample code needs to work with I2C_Wakeup_Slave.
<b>I2C_Wakeup_Slave</b>	Show how to wake up MCU from Power-down mode through I <sup>2</sup> C interface. This sample code needs to work with I2C_Wakeup_Master.
<b>PWM_Capture</b>	Capture the PWM1 Channel 0 waveform by PWM0 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>SPI_Loopback</b>	Implement SPI Master loop back transfer. This sample code needs to connect SPI0_MISO0 pin and SPI0_MOSI0 pin together. It will compare the received data with transmitted data.
<b>SPI_MasterFifoMode</b>	Configure SPI0 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device. Needs to work with SPI_SlaveFifoMode sample code.
<b>SPI_SlaveFifoMode</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_MasterFifoMode sample code.
<b>SYS_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLK0 pin.
<b>TIMER_Capture</b>	Show how to use the timer0 capture function to capture timer0 counter value.
<b>TIMER_Counter</b>	Implement timer0 event counter function to count the external input event

<b>TIMER_Delay</b>	Show how to use timer0 to create various delay time.
<b>TIMER_PeriodicINT</b>	Implement timer counting in periodic mode.
<b>TIMER_PowerDown</b>	Use timer0 periodic time-out interrupt event to wake up system.
<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_LIN</b>	Transmit LIN header and response.
<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.
<b>WWDT_CompareINT</b>	Show how to reload the WWDT counter value.

### 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.*