



# **PY32MD420 series**

## **32-bit ARM® Cortex®-M0+ microcontroller**

### **HAL Library Sample Manual**



# 1 ADC

## 1.1 ADC\_AnalogWatchdog

此样例演示了 ADC 的模拟看门狗功能，当开启看门狗的通道的电压值不在设定的上下限中，会进入看门狗中断。

This example demonstrates the analog watchdog function of ADC. When the voltage value of the channel that opens the watchdog is not within the set upper or lower limits, Will enter watchdog interrupt.

## 1.2 ADC\_MultiChannels\_TriggerTimer\_DMA

此样例演示了 ADC 的多通道 DMA 传输的功能。

This example demonstrates the functionality of multi-channel DMA transmission in ADC.

## 1.3 ADC\_SingleConversion\_TriggerSW\_IT

此样例演示了 ADC 的中断功能，每隔 1s，软件触发 ADC 采样，在中断中通过串口打印通道 4 的 DR 值。

This example demonstrates the interrupt function of ADC. Every 1 second, the software triggers ADC sampling and prints the DR value of channel 4 through the serial port during the interrupt.

## 1.4 ADC\_SingleConversion\_TriggerSW\_Polling

此样例演示了 ADC 模块的软件触发和轮询功能。

This example demonstrates the software triggering and polling functions of the ADC module.

## 1.5 ADC\_TempSensor

此样例演示了 ADC 模块的 Tempsensor 功能，并通过串口打印出温度值。

This example demonstrates the Tempsensor function of the ADC module and prints the temperature value through the serial port.

## 1.6 ADC\_Vrefbuf

此样例演示了 ADC 模块的 Vrefbuf 功能，利用 vrefbuf 作为基准去采样通道的值，并转换成电压通过串口打印出来。

This example demonstrates the Vrefbuf function of the ADC module, which uses Vrefbuf as the reference to sample channel values and convert them into voltage. Print it out through the serial port.

## 1.7 ADC\_Vrefint

此样例演示了 ADC 模块的 VREFINT 采样功能，通过采样 VREFINT 的值，计算得出 VCC 的值，并通过串口打印出来。

This example demonstrates the VREFINT sampling function of the ADC module. By sampling the value of VREFINT, the VCC value is calculated and printed through the serial port.

## 2 COMP

### 2.1 COMP\_CompareGpioVs32\_64VCC\_IT

此样例演示了 COMP 比较器中断功能，PA1 作为比较器正端输入，32/64VCC 作为比较器负端输入，当 PA1 的电压大于 32/64VCC 电压时，LED 灯亮，小于 32/64VCC 电压时，LED 灯灭。

This example demonstrates the interrupt function of the COMP comparator, with PA1 as the positive input and VREFINT as the negative input. When the voltage of PA1 is greater than 32/64 voltage, the LED lights up, and when it is less than 32/64VCC voltage, the LED lights up.

### 2.2 COMP\_CompareGpioVs32\_64VCC\_WakeUpFromStop

此样例演示了 COMP 比较器唤醒功能，PA1 作为比较器正端输入，32/64VCC 作为比较器负端输入，进入 stop 模式后，通过调整 PA1 上的输入电压，产生中断唤醒 stop 模式。

This example demonstrates the COMP comparator wake-up function, with PA1 as the positive input and 32/64VCC as the negative input of the comparator. After entering stop mode, the interrupt wake-up stop mode is generated by adjusting the input voltage on PA1.

### 2.3 COMP\_CompareGpioVsVrefbuf\_Polling

此样例演示了 COMP 比较器轮询功能，PA1 作为比较器正端输入，Vrefbuf2.5V 作为比较器负端输入，当 PA1 的电压大于 Vrefbuf 电压时，LED 灯亮，小于 Vrefbuf 电压时，LED 灯灭。

This example demonstrates the COMP comparator polling function, with PA1 as the positive input and Vrefbuf2.5V as the negative input. When the voltage of PA1 is greater than Vrefbuf voltage, the LED lights up, and when it is less than Vrefbuf voltage, the LED light off.

## 3 CORDIC

### 3.1 CORDIC\_CalculateArctanMod\_IT

此样例演示了通过中断方式计算 arctan、mod 的值。

This example demonstrates calculating the values of arctan and mod through interruption.

### 3.2 CORDIC\_CalculateSinCos

此样例演示了通过轮询方式计算 sin、cos 的值。

This example demonstrates calculating the values of sin and cos through polling.

### 3.3 CORDIC\_CalculateSqrt\_IT

此样例演示了通过中断方式计算 sqrt 的值。

This example demonstrates calculating the value of sqrt through interruption.

## 4 CRC

### 4.1 CRC\_CalculateCheckValue

此样例演示了 CRC 校验功能，通过对一个数组里的数据进行校验，得到的校验值与理论校验值进行比较，相等则 LED 灯亮，否则 LED 灯熄灭。

This example demonstrates the CRC verification function. By verifying the data in an array, the obtained verification value is compared with the theoretical verification value. If it is equal, the LED light will be on, otherwise the LED light will be off.

## 5 DMA

### 5.1 DMA\_SramToSram

此样例演示了 DMA 从 SRAM 到 SRAM 传输数据的功能（SRAM 和外设之间传输的样例请参考相关外设样例工程）。

This example demonstrates the function of DMA transferring data from SRAM to SRAM (please refer to the relevant peripheral sample project for the example of transfer between SRAM and peripherals).



## 6 EXTI

### 6.1 EXTI\_Toggleled\_IT

此样例演示了 GPIO 外部中断功能, 按键 (PB0) 引脚上的每一个下降沿都会产生中断, 中断函数中 LED 灯会翻转一次。

This example demonstrates the GPIO external interrupt function, where each falling edge on the key (PB0) pin generates an interrupt, and the LED light in the interrupt function flips once.

### 6.2 EXTI\_WakeUp\_Event

此样例演示了通过 PA6 引脚唤醒 MCU 的功能。下载程序并运行后, LED 灯处于常亮状态; 按下用户按键后, LED 灯处于常暗状态, 且 MCU 进入 STOP 模式; 拉低 PA6 引脚后, MCU 唤醒, LED 灯处于闪烁状态。

This example demonstrates the function of waking up an MCU through the PA6 pin. After downloading the program and running it, the LED light is constantly on; After pressing the user button, the LED light is in a constant dark state and the MCU enters STOP mode; After pulling down the PA6 pin, the MCU wakes up and the LED light is in a flashing state.

## 7 FLASH

### 7.1 FLASH\_OptionByteWrite\_RST

此样例演示了通过软件方式将 RESET 引脚改为普通 GPIO。

This example demonstrates changing the RESET pin to regular GPIO through software.

### 7.2 FLASH\_PageEraseAndWrite

此样例演示了 flash page 擦除和 page 写功能。

This example demonstrates the flash page erase and page write functions.

### 7.3 FLASH\_SectorEraseAndWrite

此样例演示了 flash sector 擦除和 Page 写功能。

This example demonstrates the flash sector erase and page write functions.

### 7.4 FLASH\_UserDataEraseAndWrite

此样例演示了 UserData 擦除和写功能。

This example demonstrates the UserData erase and write functions.

## 8 GPIO

### 8.1 GPIO\_FastIO

本样例主要展示 GPIO 的 FAST IO 输出功能。FAST IO 速度可以达到单周期翻转速度。

This sample demonstrates the FAST IO output functionality of GPIO. FAST IO speed can achieve single-cycle toggling speed.

### 8.2 GPIO\_Toggle

此样例演示了 GPIO 输出模式，配置 LED 引脚为数字输出模式，并且每隔 250ms 翻转一次 LED 引脚电平，运行程序，可以看到 LED 灯以 2Hz 的频率闪烁。

This sample demonstrates GPIO output mode. It configures the LED pin as a digital output and toggles the LED pin level every 250ms. When the program runs, you can observe the LED blinking at a frequency of 2Hz.

## 9 HDIV

### 9.1 Division\_signed

此样例演示了硬件除法器计算有符号除法。

This example demonstrates how a hardware divider calculates signed division.

### 9.2 Division\_unsigned

此样例演示了硬件除法器计算无符号除法。

This example demonstrates how a hardware divider calculates unsigned division.

## 10 I2C

### 10.1 I2C\_TwoBoard\_CommunicationMaster\_DMA

此样例演示了 I2C 通过 DMA 方式进行通讯，主机先向从机发送 15byte 数据，然后再接收从机发送的 15byte 数据,主机、从机接收数据成功后，主机和从机板上的小灯处于“常亮”状态。

This sample demonstrates communication between I2C devices using DMA. The master device sends 15 bytes of data to the slave device and then receives 15 bytes of data from the slave. After successful data transmission and reception between the master and slave, the LEDs on both boards remain constantly lit.

### 10.2 I2C\_TwoBoard\_CommunicationMaster\_DMA\_MEM

此样例演示了主机 I2C 通过 DMA 方式进行通讯，从机使用 EEPROM 外设芯片 P24C32，按下 user 按键，主机先向从机写 15bytes 数据为 0x1~0xf，然后再从 EEPROM 中将写入的数据读出，读取成功后，主机板上的小灯处于“常亮”状态。

This sample demonstrates communication between the master device using I2C and the slave device using the EEPROM peripheral chip P24C32. When the user button on the master device is pressed, the master device first writes 15 bytes of data to the slave device, ranging from 0x1 to 0xF. Then it reads the written data from the EEPROM. Once the data is successfully read, the LED on the master board will remain constantly lit.

### 10.3 I2C\_TwoBoard\_CommunicationMaster\_IT

此样例演示了 I2C 通过中断方式进行通讯，主机先向从机发送 15byte 数据，然后再接收从机发送的 15byte 数据,主机、从机接收数据成功后，主机和从机板上的小灯处于“常亮”状态。

This sample demonstrates communication between I2C devices using interrupts. The master device sends 15 bytes of data to the slave device and then receives 15 bytes of data from the slave. After successful data transmission and reception between the master and slave, the LEDs on both boards remain constantly lit.

## 10.4 I2C\_TwoBoard\_CommunicationMaster\_Polling

此样例演示了 I2C 通过轮询方式进行通讯，主机先向从机发送 15byte 数据，然后再接收从机发送的 15byte 数据,主机、从机接收数据成功后，主机和从机板上的小灯处于“常亮”状态。

This sample demonstrates communication between I2C devices using polling. The master device sends 15 bytes of data to the slave device and then receives 15 bytes of data from the slave. After successful data transmission and reception between the master and slave, the LEDs on both boards remain constantly lit.

## 10.5 I2C\_TwoBoard\_CommunicationSlave\_DMA

此样例演示了 I2C 通过 DMA 方式进行通讯，主机先向从机发送 15byte 数据，然后再接收从机发送的 15byte 数据,主机、从机接收数据成功后，主机和从机板上的小灯处于“常亮”状态。

This sample demonstrates communication between I2C devices using DMA. The master device sends 15 bytes of data to the slave device and then receives 15 bytes of data from the slave. After successful data transmission and reception between the master and slave, the LEDs on both boards remain constantly lit.

## 10.6 I2C\_TwoBoard\_CommunicationSlave\_IT

此样例演示了 I2C 通过中断方式进行通讯，主机先向从机发送 15byte 数据，然后再接收从机发送的 15byte 数据,主机、从机接收数据成功后，主机和从机板上的小灯处于“常亮”状态。

This sample demonstrates communication between I2C devices using interrupts. The master device sends 15 bytes of data to the slave device and then receives 15 bytes of data from the slave. After successful data transmission and reception between the master and slave, the LEDs on both boards remain constantly lit.

## 10.7 I2C\_TwoBoard\_MasterTxIndefiniteLengthData\_IT

此样例演示了通过中断方式，主机发送不定长数据，从机接收不定长数据。主机向从机发送 10 字节的数据（0~9），然后从机接收数据（0~9）并通过串口打印；主机向从机发送 100 字节数据（1~100），然

后从机接收数据（1~100）并通过串口打印；主机向从机发送 10 字节的数据（0~9），然后从机接收数据（0~9）并通过串口打印。

This example demonstrates how the host sends variable length data and the slave receives variable length data through interrupt mode. The host sends 10 bytes of data (0-9) to the slave, and then the slave receives the data (0-9) and prints it through the serial port; The host sends 100 bytes of data (1-100) to the slave, and then the slave receives the data (1-100) and prints it through the serial port; The host sends 10 bytes of data (0-9) to the slave, and then the slave receives the data (0-9) and prints it through the serial port.

## 10.8 I2C\_TwoBoard\_SlaveRxIndefiniteLengthData\_IT

此样例演示了通过中断方式，主机发送不定长数据，从机接收不定长数据。主机向从机发送 10 字节的数据（0~9），然后从机接收数据（0~9）并通过串口打印；主机向从机发送 100 字节数据（1~100），然后从机接收数据（1~100）并通过串口打印；主机向从机发送 10 字节的数据（0~9），然后从机接收数据（0~9）并通过串口打印。

This example demonstrates how the host sends variable length data and the slave receives variable length data through interrupt mode. The host sends 10 bytes of data (0-9) to the slave, and then the slave receives the data (0-9) and prints it through the serial port; The host sends 100 bytes of data (1-100) to the slave, and then the slave receives the data (1-100) and prints it through the serial port; The host sends 10 bytes of data (0-9) to the slave, and then the slave receives the data (0-9) and prints it through the serial port.

## 11 I2S

### 11.1 I2S\_TwoBoard\_CommunicationMaster\_DMA

此样例是对 I2S 主机与 I2S 从机以 DMA 方式进行通信的演示, I2S 主机先向 I2S 从机发送数据 0x1~0x10, I2S 从机接收到数据后, 再向 I2S 主机回发数据 0x1~0x10, 当 I2S 主机和 I2S 从机成功接收数据时, 小灯处于常亮状态; 否则小灯处于闪烁状态。

This sample demonstrates communication between the I2S master and I2S slave using DMA. The I2S master sends data 0x1 to 0x10 to the I2S slave. The I2S slave receives the data and sends back data 0x1 to 0x10 to the I2S master. When both the I2S master and I2S slave successfully receive the data, the LED will be constantly on. Otherwise, the LED will be blinking.

### 11.2 I2S\_TwoBoard\_CommunicationMaster\_IT

此样例是对 I2S 主机与 I2S 从机以中断方式进行通信的演示, I2S 主机先向 I2S 从机发送数据 0x1~0x10, I2S 从机接收到数据后, 再向 I2S 主机回发数据 0x1~0x10, 当 I2S 主机和 I2S 从机成功接收数据时, 小灯处于常亮状态; 否则小灯处于闪烁状态。

This sample demonstrates communication between the I2S master and I2S slave using interrupts. The I2S master sends data 0x1 to 0x10 to the I2S slave. The I2S slave receives the data and sends back data 0x1 to 0x10 to the I2S master. When both the I2S master and I2S slave successfully receive the data, the LED will be constantly on. Otherwise, the LED will be blinking.

### 11.3 I2S\_TwoBoard\_CommunicationMaster\_Polling

此样例是对 I2S 主机与 I2S 从机以轮询方式进行通信的演示, I2S 主机先向 I2S 从机发送数据 0x1~0x10, I2S 从机接收到数据后, 再向 I2S 主机回发数据 0x1~0x10, 当 I2S 主机和 I2S 从机成功接收数据时, 小灯处于常亮状态; 否则小灯处于闪烁状态。

This sample demonstrates communication between the I2S master and I2S slave using polling. The I2S master sends data 0x1 to 0x10 to the I2S slave. The I2S slave receives the data and sends back data 0x1 to 0x10 to the I2S master. When both the I2S master and I2S slave successfully receive the data, the LED will be constantly on. Otherwise, the LED will be



blinking.

## 11.4 I2S\_TwoBoard\_CommunicationSlave\_DMA

此样例是对 I2S 主机与 I2S 从机以 DMA 方式进行通信的演示, I2S 主机先向 I2S 从机发送数据 0x1~0x10, I2S 从机接收到数据后, 再向 I2S 主机回发数据 0x1~0x10, 当 I2S 主机和 I2S 从机成功接收数据时, 小灯处于常亮状态; 否则小灯处于闪烁状态。

This sample demonstrates communication between the I2S master and I2S slave using DMA. The I2S master sends data 0x1 to 0x10 to the I2S slave. The I2S slave receives the data and sends back data 0x1 to 0x10 to the I2S master. When both the I2S master and I2S slave successfully receive the data, the LED will be constantly on. Otherwise, the LED will be blinking.

## 11.5 I2S\_TwoBoard\_CommunicationSlave\_IT

此样例是对 I2S 主机与 I2S 从机以中断方式进行通信的演示, I2S 主机先向 I2S 从机发送数据 0x1~0x10, I2S 从机接收到数据后, 再向 I2S 主机回发数据 0x1~0x10, 当 I2S 主机和 I2S 从机成功接收数据时, 小灯处于常亮状态; 否则小灯处于闪烁状态。

This sample demonstrates communication between the I2S master and I2S slave using interrupts. The I2S master sends data 0x1 to 0x10 to the I2S slave. The I2S slave receives the data and sends back data 0x1 to 0x10 to the I2S master. When both the I2S master and I2S slave successfully receive the data, the LED will be constantly on. Otherwise, the LED will be blinking.

## 11.6 I2S\_TwoBoard\_CommunicationSlave\_Polling

此样例是对 I2S 主机与 I2S 从机以轮询方式进行通信的演示, I2S 主机先向 I2S 从机发送数据 0x1~0x10, I2S 从机接收到数据后, 再向 I2S 主机回发数据 0x1~0x10, 当 I2S 主机和 I2S 从机成功接收数据时, 小灯处于常亮状态; 否则小灯处于闪烁状态。

This sample demonstrates communication between the I2S master and I2S slave using polling. The I2S master sends data 0x1 to 0x10 to the I2S slave. The I2S slave receives the data and sends back data 0x1 to 0x10 to the I2S master. When both the I2S master and I2S

slave successfully receive the data, the LED will be constantly on. Otherwise, the LED will be blinking.

## 12 IWDG

### 12.1 IWDG\_Reset

此样例演示了 IWDG 看门狗功能。配置看门狗的重载计数值为 1 秒，当计数达到 1 秒后，系统会被复位。通过调整每次喂狗的时间（main 函数 while 循环中的代码），可以观察到以下情况：如果每次喂狗时间小于 1 秒，程序能够正常运行（LED 灯闪烁）；如果喂狗时间超过 1 秒，程序会一直被复位（LED 灯熄灭）。

This sample demonstrates the IWDG (Independent Watchdog) functionality. The watchdog is configured with a reload value of 1 second. Once the watchdog timer reaches 1 second, the system will be reset. By adjusting the time for feeding the watchdog (code in the main loop), the following observations can be made: If the feeding time is less than 1 second, the program can run normally (LED blinks); If the feeding time exceeds 1 second, the program will be continuously reset (LED turns off).

## 13 LPTIM

### 13.1 LPTIM\_OnceModeWakeup\_WFE

此样例演示了 LPTIM 单次模式事件唤醒 STOP 模式。

This example demonstrates the LPTIM once mode event wake-up STOP mode.

### 13.2 LPTIM\_OnceModeWakeup\_WFI

此样例演示了 LPTIM 单次模式中断唤醒 STOP 模式。

This example demonstrates the LPTIM once mode interrupt wake-up STOP mode.

### 13.3 LPTIM\_Wakeup\_WFE

此样例演示了 LPTIM 连续模式事件唤醒 STOP 模式。

This example demonstrates the LPTIM continuous mode event wake-up STOP mode.

### 13.4 LPTIM\_Wakeup\_WFI

此样例演示了 LPTIM 连续模式中断唤醒 STOP 模式。

This example demonstrates the LPTIM continuous mode interrupt wake-up STOP mode.

## 14 OPA

### 14.1 OPA\_VoltageFollow

此样例演示了 OPA 的电压跟随功能，PB3 为正端输入，PA15 为负端输入，PB4 为输出，PB4 会输出和 PB3 相同的电压值。

This sample demonstrates the voltage follower functionality of the OPA. PB3 is the positive input, PA15 is the negative input, and PB4 is the output. PB4 will output the same voltage as PB3.

## 15 PWR

### 15.1 PVD

此样例演示了 PVD 电压检测功能。样例中配置 PB07 引脚的电压与 VREF (1.2V) 进行比较。当 PB07 引脚的电压高于 VREF 时，LED 灯灭；当低于 VREF 时，LED 灯亮。

This sample demonstrates the PVD (Programmable Voltage Detector) voltage monitoring functionality. In this example, PB07 pin is configured to compare its voltage with VREF (1.2V). When the voltage at PB07 is higher than VREF, the LED light turns off. When it is lower than VREF, the LED light turns on.

### 15.2 PWR\_SLEEP\_WFE

此样例演示了在 sleep 模式下，使用 GPIO 事件唤醒。

This example demonstrates using GPIO event wake-up in sleep mode.

### 15.3 PWR\_SLEEP\_WFI

此样例演示了在 sleep 模式下，使用 GPIO 中断唤醒。

This example demonstrates using GPIO interrupt wake-up in sleep mode.

### 15.4 PWR\_STOP\_WFE

此样例演示了在 stop 模式下，使用 GPIO 事件唤醒。

This example demonstrates using GPIO event wake-up in stop mode.

## 15.5 PWR\_STOP\_WFI

此样例演示了在 stop 模式下，使用 GPIO 中断唤醒。

This example demonstrates using GPIO interrupt wake-up in stop mode.

## 16 RCC

### 16.1 RCC\_HSEOutput

此样例配置系统时钟为 HSE，并通过 MCO (PA08) 引脚输出。

This sample configures the system clock as HSE and outputs it through the MCO (PA08) pin.

### 16.2 RCC\_HSIOutput

此样例配置系统时钟为 HSI，并通过 MCO (PA08) 引脚输出。

This sample configures the system clock as HSI and outputs it through the MCO (PA08) pin.

### 16.3 RCC\_LSEOutput

此样例使能 LSE，并通过 MCO (PA08) 引脚输出。

This sample enables the LSE and is output via the MCO (PA08) pin.

### 16.4 RCC\_LSIOutput

此样例使能 LSI，并通过 MCO (PA08) 引脚输出。

This sample enables the LSI and is output via the MCO (PA08) pin.

### 16.5 RCC\_PLLOutput



此样例配置系统时钟为 PLL，并通过 MCO（PA08）引脚输出。PLL 的输入时钟源选择 HSI。

This sample configures the system clock as PLL and outputs it through the MCO (PA08) pin. The PLL input clock source is set to HSI.

## 16.6 RCC\_SysclockSwitch

此样例演示系统时钟切换功能。样例中配置系统时钟从 HSI 切换到 HSE，并通过 MCO（PA08）引脚输出系统时钟。

This sample demonstrates the system clock switching functionality. The sample configures the system clock to switch from HSI to HSE and outputs the system clock through the MCO (PA08) pin.

## 17 RTC

### 17.1 RTC\_AlarmSecond\_IT

此样例演示 RTC 的秒中断和闹钟中断功能。每次秒中断，在中断函数中会打印字符“RTC\_IT\_SEC”并输出实时时间。

This sample demonstrates the RTC's second interrupt and alarm interrupt functionality. Each time the second interrupt occurs, the interrupt function prints the string "RTC\_IT\_SEC" and outputs the current RTC count time.

### 17.2 RTC\_WakeUpAlarm

此样例演示通过 RTC 闹钟中断每隔 1 秒将 MCU 从 STOP 模式下唤醒，并且每次唤醒会翻转 LED，LED 翻转间隔为 1 秒。

This sample demonstrates waking up the MCU from STOP mode every 1 second using RTC alarm interrupt. Each time the MCU wakes up, the LED will toggle its state. The LED toggling interval is 1 second.

### 17.3 RTC\_WakeUpSecond

此样例演示了通过 RTC 的秒中断唤醒 MCU 的功能。下载程序并运行后，LED 灯处于常亮状态；按下用户按键后，LED 灯处于常暗状态，且 MCU 进入 STOP 模式；RTC 秒中断唤醒 MCU 后，LED 灯处于闪烁状态。

This sample demonstrates waking up the MCU using RTC second interrupt. After downloading and running the program, the LED is continuously on. Pressing the user button turns off the LED and puts the MCU into STOP mode. When the RTC second interrupt wakes up the MCU, the LED starts blinking.

## 18 SPI

### 18.1 SPI\_TwoBoards\_FullDuplexMaster\_DMA

此样例是利用 DMA 对串口外设接口 (SPI) 与外部设备以全双工串行方式进行通信的演示，主设备提供通信时钟 SCK，通过 MOSI/MISO 引脚发送/接收数据。从设备通过 MOSI/MISO 引脚接收/发送数据。数据以主机提供的 SCK 沿同步被移位，完成全双工通信。

This sample is a demonstration of using DMA to communicate with a serial peripheral interface (SPI) and an external device in full-duplex serial mode. The master device provides the communication clock SCK and sends/receives data through the MOSI/MISO pin. The slave device receives/transmits data through the MOSI/MISO pins. The data is shifted synchronously along the SCK provided by the master to complete full-duplex communication.

### 18.2 SPI\_TwoBoards\_FullDuplexMaster\_IT

此样例是利用中断对串口外设接口 (SPI) 与外部设备以全双工串行方式进行通信的演示，主设备提供通信时钟 SCK，通过 MOSI/MISO 引脚发送/接收数据。从设备通过 MOSI/MISO 引脚接收/发送数据。数据以主机提供的 SCK 沿同步被移位，完成全双工通信。

This sample is a demonstration of using interrupts to communicate with a serial peripheral interface (SPI) and an external device in full-duplex serial mode. The master device provides the communication clock SCK and sends/receives data through the MOSI/MISO pin. The slave device receives/transmits data through the MOSI/MISO pins. The data is shifted synchronously along the SCK provided by the master to complete full-duplex communication.

### 18.3 SPI\_TwoBoards\_FullDuplexMaster\_Polling

此样例是通过轮询方式对串口外设接口 (SPI) 与外部设备以全双工串行方式进行通信的演示。主设备提供通信时钟 SCK，通过 MOSI/MISO 引脚发送/接收数据。从设备通过 MOSI/MISO 引脚接收/发送数据。数据以主机提供的 SCK 沿同步被移位，完成全双工通信。

This sample is a demonstration of the Serial Peripheral Interface (SPI) communicating with an external device in full-duplex serial mode by polling. The master device provides the

communication clock SCK and sends/receives data via the MOSI/MISO pin. The slave device receives/transmits data through the MOSI/MISO pins. The data is shifted synchronously along the SCK provided by the master to complete full-duplex communication.

## 18.4 SPI\_TwoBoards\_FullDuplexSlave\_DMA

此样例是利用 DMA 对串口外设接口 (SPI) 与外部设备以全双工串行方式进行通信的演示，主设备提供通信时钟 SCK，通过 MOSI/MISO 引脚发送/接收数据。从设备通过 MOSI/MISO 引脚接收/发送数据。数据以主机提供的 SCK 沿同步被移位，完成全双工通信。

This sample is a demonstration of using DMA to communicate with a serial peripheral interface (SPI) and an external device in full-duplex serial mode. The master device provides the communication clock SCK and sends/receives data through the MOSI/MISO pin. The slave device receives/transmits data through the MOSI/MISO pins. The data is shifted synchronously along the SCK provided by the master to complete full-duplex communication.

## 18.5 SPI\_TwoBoards\_FullDuplexSlave\_IT

此样例是利用中断对串口外设接口 (SPI) 与外部设备以全双工串行方式进行通信的演示，主设备提供通信时钟 SCK，通过 MOSI/MISO 引脚发送/接收数据。从设备通过 MOSI/MISO 引脚接收/发送数据。数据以主机提供的 SCK 沿同步被移位，完成全双工通信。

This sample is a demonstration of using interrupts to communicate with a serial peripheral interface (SPI) and an external device in full-duplex serial mode. The master device provides the communication clock SCK and sends/receives data through the MOSI/MISO pin. The slave device receives/transmits data through the MOSI/MISO pins. The data is shifted synchronously along the SCK provided by the master to complete full-duplex communication.

## 18.6 SPI\_TwoBoards\_FullDuplexSlave\_Polling

此样例是通过轮询方式对串口外设接口 (SPI) 与外部设备以全双工串行方式进行通信的演示。主设备提供通信时钟 SCK，通过 MOSI/MISO 引脚发送/接收数据。从设备通过 MOSI/MISO 引脚接收/发送数据。数据以主机提供的 SCK 沿同步被移位，完成全双工通信。

This sample is a demonstration of the Serial Peripheral Interface (SPI) communicating with an external device in full-duplex serial mode by polling. The master device provides the communication clock SCK and sends/receives data via the MOSI/MISO pin. The slave device receives/transmits data through the MOSI/MISO pins. The data is shifted synchronously along the SCK provided by the master to complete full-duplex communication.

## 19 TIM

### 19.1 TIM14\_LSI Calibrate

此样例配置 LSI 时钟从 MCO (PA8) 输出并将 TIM14 的通道 1 连接到 MCO。将系统时钟配置为 32MHz (HSI 16MHz, PLL 配置 2 倍频), TIM14 时钟为 320KHz, 设置重载值为 10001, 使能 TIM14 的输入捕获功能, 调整 LSI trimming 值, LED 从闪烁变成常亮表示校准完成。

This example configures the LSI clock to output from MCO (PA8) and connects channel 1 of TIM14 to MCO. Set the system clock to 32MHz (HSI 16MHz, PLL frequency doubling), set the clock of TIM14 to 320KHz, set the overload value to 10001, enable the input capture function of TIM14, adjust the value of LSI trimming, When the LED changes from blinking to steady on, the calibration is complete

### 19.2 TIM1\_6Step

此样例是对高级定时器功能“六步 PWM 的产生”的演示, 通过 systick 中断作为 COM commutation 事件的触发源, 实现 (无刷电机的) 换向下表是换向步骤, 比如第一步中的 CH1 和 CH3N 为 1, 即设置打开这两个通道的 PWM 输出。

This sample demonstrates advanced timer function 'six-step PWM generation', systick interrupt as COM commutation event trigger source to achieve commutation (brushless motor). The following table shows the commutating steps. For example, CH1 and CH3N in the first step are set to 1, that mean the PWM output of these two channels is set to start

### 19.3 TIM1\_AutoReloadPreload

此样例实现了定时器的基本计数功能, 以及演示了 ARR 自动重载功能, 样例在定时器重载中断中翻转 LED 灯 修改 main.c 中的配置 `TimHandle.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE`;使能自动重载功能, 新的 ARR 值在第四次进中断时生效, 配置 `TimHandle.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE`;禁止自动重载功能, 新的 ARR 值在第三次进中断时生效,生效后, LED 灯以 2.5HZ 的频率翻转

This sample demonstrates base count function of the timer, and show ARR register autoreload function. Example toggle LED in timer update interrupt. Modify in main.c. Set

TimHandle.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE to enable autoreload, and new ARR value will take effect on the fourth interrupt generate. Set TimHandle.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE to disable autoreload, and new ARR value will take effect on the third interrupt generate. After taking effect, the LED lights blinked at a frequency of 2.5HZ.

## 19.4 TIM1\_ComplementarySignals

此样例实现了定时器的互补输出功能，三组互补共六路 pwm 输出，此样例没有实现死区功能 CH1 -> PA8CH1N -> PA7CH2 -> PA9CH2N -> PA1CH3 -> PA10CH3N -> PB1

This sample demonstrates complementary output function of the timer, Three sets of complementary outputs total six pwm outputs, this example does not implement the dead zone function

## 19.5 TIM1\_ComplementarySignals\_break

此样例实现了定时器的刹车功能，CH1 和 CH1N 互补 pwm 输出，接收到外部 IO 口的刹车信号（低电平）后，PWM 信号关闭，由于 BDTR.AOE 置位，所以刹车信号取消（高电平）后，继续 pwm 输出，此样例实现了死区功能。CH1 -> PA8CH1N -> PA7 刹车输入 -> PA6 通过调整 OCxE, CCxP, OISx, CCxNE, CCxNP, OISxN 的配置，可实现刹车功能的各种应用

This sample demonstrates brake function of the timer, the CH1 and CH1N complementary pwm outputs. After receiving the brake signal (low level) from the external IO port, the PWM signal is turned off. Because BDTR.AOE is set, the pwm output continues after the brake signal is cancelled (high level). This example realizes the dead zone function CH1 -> PA8 CH1N -> PA7 Brake input -> PA6 By adjusting the OCxE, CCxP, OISx, CCxNE, CCxNP, OISxN configuration, which can realize the brake function of a variety of applications

## 19.6 TIM1\_ComplementarySignals\_break\_it

此样例实现了定时器的刹车功能，CH1 和 CH1N 互补 pwm 输出，接收到外部 IO 口的刹车信号（低电平）后，PWM 信号关闭，由于 BDTR.AOE 置位，所以刹车信号取消（高电平）后，继续 pwm 输出，此样例实现了死区功能。本样例开启了刹车中断，并在刹车中断里翻转 LED 灯通过调整 OCxE, CCxP, OISx, CCxNE, CCxNP, OISxN 的配置，可实现刹车功能的各种应用

This sample demonstrates brake function of the timer,the CH1 and CH1N complementary pwm outputs.After receiving the brake signal (low level)from the external IO port, the PWM signal is turned off. Because BDTR.AOE is set, the pwm output continues after the brake signal is cancelled (high level). This example realizes the dead zone function. This example turns on the brake interrupt and toggle the LED light in the brake interrupt. By adjusting the OCxE, CCxP, OISx, CCxNE, CCxNP, OISxN configuration, which can realize the brake function of a variety of applications

## 19.7 TIM1\_ComplementarySignals\_DeadTime

此样例实现了定时器的刹车功能，CH1 和 CH1N 互补 pwm 输出，接收到外部 IO 口的刹车信号（低电平）后，PWM 信号关闭，由于 BDTR.AOE 置位，所以刹车信号取消（高电平）后，继续 pwm 输出，此样例实现了死区功能。通过调整 OCxE,CCxP,OISx,CCxNE,CCxNP,OISxN 的配置，可实现刹车功能的各种应用

This sample demonstrates brake function of the timer,the CH1 and CH1N complementary pwm outputs.After receiving the brake signal (low level)from the external IO port, the PWM signal is turned off. Because BDTR.AOE is set, the pwm output continues after the brake signal is cancelled (high level). This example realizes the dead zone function. By adjusting the OCxE, CCxP, OISx, CCxNE, CCxNP, OISxN configuration, which can realize the brake function of a variety of applications

## 19.8 TIM1\_DmaBurst\_twice

此样例演示了在 TIM1 中使用 DMA 连续两次 burst 传输数据的功能,burst 每传输一次更新三个寄存器,PSC,ARR,RCR, 在更新事件中断中, PA0 会进行翻转, 通过逻辑分析仪监测, 可看到 PA0 的翻转间隔会从第一次的 400ms, 第二次 400ms, 第三次 20ms,第四次及后续变为 200us, 此时两次 burst 传输完成, 并且 PCS,ARR,RCR 均更新完毕。

This sample demonstrates the function to transfer data in TIM1 using DMA in two consecutive bursts.burst updates three registers(PSC,ARR,RCR) per transfer.In the interruption of update event, PA0 will be flipped. Through the monitoring of logic analyzer, it can be seen that the flipping interval of PA0 will change from 400ms for the first time, 400ms for the second time, 20ms for the third time, and 200us for the fourth and subsequent times. At this time, the two burst transmission is completed, and PCS,ARR and RCR are all updated.



## 19.9 TIM1\_EncoderTI2AndTI1

此样例实现了 TIM1 中的编码器计数功能，TI1(PA8)和 TI2(PA9)作为编码器输入引脚，通过 CNT 寄存器可观察到计数器变化，通过 uwDirection 变量可观察到计数器的计数方向，通过打印数据也可观察计数方向和 CNT 寄存器计数值，打印数据 Direction = 0 为向上计数，Direction = 1 为向下计数。

This sample demonstrates encoder count function of the TIM1, TI1(PA8) and TI2(PA9) configured as encoder input pins. The change of the counter can be observed through the CNT register, and the counting direction of the counter can be observed through the uwDirection variable. The counting Direction and CNT register can also be observed by printing data. The printed data Direction = 0 indicates CounterMode:Up, and direction = 1 indicates CounterMode:down.

## 19.10 TIM1\_ExternalClockMode1

此样例演示了 TIM1 的外部时钟模式 1 功能，选择 ETR(PA12)引脚作为外部时钟输入源，并使能更新中断，在中断中翻转 LED 灯

This sample demonstrates external clock mode 1 function of the TIM1. Select the ETR(PA12) pin as the external clock input source and enable the update interrupt to flip the LED light in the interrupt.

## 19.11 TIM1\_ExternalClockMode1\_TI1F

此样例演示了 TIM1 的外部时钟模式 1 功能，选择 TI1FD(PA8)引脚作为外部时钟输入源，并使能更新中断，在中断中翻转 LED 灯

This sample demonstrates the external clock mode 1 function of TIM1, selects the TI1FD(PA8) pin as the external clock input source, and enables the update interrupt and toggle the LED light in the interrupt

## 19.12 TIM1\_ExternalClockMode2

此样例演示了 TIM1 的外部时钟模式 2 功能，选择 ETR(PA12)引脚作为外部时钟输入源，并使能更新中

断，在中断中翻转 LED 灯。

This sample demonstrates the external clock mode 2 function of TIM1, selects the ETR(PA12) pin as the external clock input source, and enables the update interrupt and toggle the LED light in the interrupt

### 19.13 TIM1\_InputCapture\_TI1FP1

此样例演示了在 TIM1(PA8)输入捕获功能，PA8 输入时钟信号，TIM1 捕获成功后，会进入捕获中断，每进一次中断，翻转一次 LED

This sample demonstrates the input capture function of TIM1(PA8), PA8 input clock signal, when TIM1 capture success, will enter the capture interrupt, and toggle the LED in the interrupt

### 19.14 TIM1\_InputCapture\_XORCh1Ch2Ch3

此样例演示了在 TIM1 输入捕获功能，PA8 或 PA9 或 PA10 输入时钟信号，TIM1 捕获成功后，会进入捕获中断，每进一次中断，翻转一次 LED

This sample demonstrates the input capture function of TIM1, input clock signal into PA8 or PA9 or PA10 will generate the capture interrupt after TIM1 capture successfully. Toggle the LED once per interruption

### 19.15 TIM1\_OCToggle

此样例演示了 TIM1 比较模式下的 OC 翻转输出功能，使能 CH1(PA08),CH2(PA09),CH3(PA10),CH4(PA11)四个通道的输出功能，并且当计数器 TIMx\_CNT 与 TIMx\_CCRx 匹配时输出信号翻转，频率为 100KHz

This sample demonstrates the OC toggle output function in TIM1 comparison mode, enabling CH1(PA08),CH2(PA09),CH3(PA10),CH4(PA11) four channel output function, then the output signal toggle when the counter TIMx\_CNT matches TIMx\_CCRx. The frequency is 100KHz

## 19.16 TIM1\_OnePulseOutput

此样例演示了 TIM1 的单脉冲模式，CH2(PA09)引脚上的上升沿，触发计数器开始计数，当计数值与 CCR1 匹配时，CH1(PA08)输出高电平，直到计数器溢出，CH1 再次输出低电平，计数器溢出后，定时器停止工作，本例程脉冲宽度计算  $(TIM1\_ARR - TIM1\_CCR1) / CLK = (65535 - 16383) / 32000000 = 1.536ms$

This sample demonstrates the one pulse mode of TIM1. The rising edge on the CH2(PA09) pin triggers the counter to start counting. When the count value matches CCR1, CH1(PA08) outputs a high level. When the counter overflows, CH1 outputs the low level again. After the counter overflows, the timer stops working. This example pulse width calculation  $(TIM1\_ARR - TIM1\_CCR1) / CLK = (65,535 - 16383) / 32,000,000 = 1.536ms$

## 19.17 TIM1\_PWM

本例程输出 4 路 PWM，通道 1 的占空比为 20%，通道 2 为 40%，通道 3 为 60%，通道 4 为 80%，本例程周期为  $8000000 / (50 + 1) / 800 = 196Hz$

This sample outputs 4 channels PWM, the duty cycle of channel 1 is 20%, channel 2 is 40%, channel 3 is 60%, channel 4 is 80%. The period is  $8000000 / (50 + 1) / 800 = 196Hz$

## 19.18 TIM1\_SynchronizationEnable

定时器 1 的使能由定时器 2 控制，当定时器 2 计数时，LED 会常亮，当定时器 2 发生更新事件时，更新事件会触发定时器 1，定时器 1 开始计数后，LED 会以 5Hz 的频率进行翻转

The enable of TIM1 is controlled by TIM2. When TIM2 counts, the LED will be steady on. The update event generated by TIM2 will triggers TIM1, and when TIM1 starts counting, the LED is toggled at a frequency of 5Hz

## 19.19 TIM1\_TIM2\_Cascade

此样例实现了 TIM1 和 TIM2 级联成 48 位计数器，TIM2 做主机，TIM2 的计数溢出信号作为 TIM1 的输入时钟，通过配置 TIM1 和 TIM2 的重载寄存器值，（在 TIM1 中断回调函数中）实现 LED 灯以 0.5Hz

频率闪烁。

This example realizes the cascade of TIM1 and TIM2 into a 48-bit counter, with TIM2 as the host. The count overflow signal of TIM2 acts as the input clock of TIM1. By configuring the reloaded register values of TIM1 and TIM2, the LED is toggled at 0.5Hz (in the TIM1 interrupt callback function).

## 19.20 TIM1\_Update\_DMA

此样例演示了在 TIM1 中使用 DMA 传输数据的功能，通过 DMA 从 SRAM 中搬运数据到 ARR 寄存器，实现 TIM1 周期变化，在 TIM1 第一次溢出后，PA0 会翻转，此时翻转间隔为 400ms，DMA 开始搬运数据到 TIM1\_ARR，第一次 PA0 翻转间隔为 400ms，第二次翻转间隔为 100ms，第三次翻转间隔为 200ms，第四次翻转间隔为 300ms，此时 DMA 搬运结束，后续翻转间隔均为 300ms

This sample demonstrates the function of using DMA to transfer data in TIM1, carrying data from SRAM to ARR register by DMA to achieve TIM1 cycle change. After the first overflow of TIM1, PA0 will toggle, at this time the toggle interval is 400ms. DMA starts to carry data to TIM1\_ARR, the first PA0 toggle interval is 400ms, the second toggle interval is 100ms, the third toggle interval is 200ms, the fourth toggle interval is 300ms, at this time the DMA carrying ends, the subsequent toggle interval are 300ms

## 19.21 TIM1\_Update\_IT

此样例演示了在 TIM1 中基本计数功能，并使能了更新中断，每次重装 ARR 值时会产生一次更新中断，并在中断中翻转 LED 灯，LED 灯会以 5Hz 的频率进行翻转。

This sample demonstrates basic count function of the TIM1 and enable update interrupt. Each time an update interrupt is generated, the ARR value is reloaded and the LED light is toggled in the interrupt. The LED light is toggled at a frequency of 5Hz.

## 20 USART

### 20.1 USART\_HyperTerminal\_AutoBaud\_IT

此样例演示了 USART 的自动波特率检测功能，调试助手发送一个字符 0x7F，MCU 反馈字符串：Auto BaudRate Test。

This sample demonstrates automatic baud rate detection function of the USART. The serial assistant sends a character 0x7F, the MCU feedback character string: Auto BaudRate Test.

### 20.2 USART\_HyperTerminal\_DMA

此样例演示了 USART 的 DMA 方式发送和接收数据，USART 配置为 115200，数据位 8，停止位 1，校验位 None，下载并运行程序后，打印提示信息，然后通过上位机下发 12 个数据，例如 0x1~0xC，则 MCU 会把接收到的数据再次发送到上位机，然后打印结束信息。

This example demonstrates how to use USART to send an amount of data in DMA mode. USART configuration is 115200 baud rate, data bit 8, stop bit 1, check bit None. Download and run the program, print the prompt message, and then send 12 data through the upper computer, such as 0x1~0xC, the MCU will send the received data to the upper computer again, then print the end message.

### 20.3 USART\_HyperTerminal\_IndefiniteLengthData\_IT

此样例演示了 USART 的中断方式发送和接收不定长数据，USART 配置为 115200，数据位 8，停止位 1，校验位 None，下载并运行程序后，然后通过上位机下发任意长度个数据（不超过 128bytes），例如 0x1~0xC，则 MCU 会把接收到的数据再次发送到上位机。

This example demonstrates the interrupt method of USART to send and receive variable length data. USART is configured as 115200, with data bit 8, stop bit 1, and check bit None. After downloading and running the program, the MCU will send any length of data (not exceeding 128bytes) through the upper computer, such as 0x1~0xC. The MCU will send the received data to the upper computer again.

## 20.4 USART\_HyperTerminal\_IT

此样例演示了 USART 的中断方式发送和接收数据，USART 配置为 115200，数据位 8，停止位 1，校验位 None，下载并运行程序后，打印提示信息，然后通过上位机下发 12 个数据，例如 0x1~0xC，则 MCU 会把接收到的数据再次发送到上位机，然后打印结束信息。

This example demonstrates how to use USART to send an amount of data in interrupt mode. USART configuration is 115200 baud rate, data bit 8, stop bit 1, check bit None. After download and run the program, print the prompt message, and then send 12 data through the upper computer, such as 0x1~0xC, the MCU will send the received data to the upper computer again. Then print the end message.

## 20.5 USART\_HyperTerminal\_Polling

此样例演示了 USART 的 POLLING 方式发送和接收数据，USART 配置为 115200，数据位 8，停止位 1，校验位 None，下载并运行程序后，打印提示信息，然后通过上位机下发 12 个数据，例如 0x1~0xC，则 MCU 会把接收到的数据再次发送到上位机，然后打印结束信息。

This example demonstrates how to use USART to send an amount of data in polling mode. USART configuration is 115200 baud rate, data bit 8, stop bit 1, check bit None. After download and run the program, print the prompt message, and then send 12 data through the upper computer, such as 0x1~0xC, the MCU will send the received data to the upper computer again. Then print the end message.

## 21 WWDG

### 21.1 WWDG\_IT

此样例演示了 WWDG 的提前唤醒中断功能，看门狗计数器向下计数到 0x40 时产生中断，中断中喂狗，可以确保看门狗不会复位。

This example demonstrates early wake up interrupt function of the WWDG. When the watchdog counter counts down to 0x40 will generate an interrupt. Refresh the WWDG in interrupt to ensure that the WWDG does not reset.

### 21.2 WWDG\_Window

此样例演示了 WWDG 的窗口看门狗功能，配置 WWDG 的窗口上限（下限固定是 0x3F），程序中通过 delay 延时函数，确保程序是在 WWDG 计数窗口内进行喂狗动作，通过 LED 灯闪烁，可以判断窗口内喂狗并未产生复位。

This example demonstrates the window watchdog function of WWDG. Set the upper limit of the window of WWDG (the lower limit is fixed at 0x3F). The program ensures that the WWDG is refreshed in the WWDG counting window through the delay function, and can judge that the WWDG is refreshed in the window without resetting through the LED light blinking.