摘 要

本文档描述了 CS32F103xB 开发板 V1.1 的硬件资源配置， Pack 包的安装，常用外设的例程介绍等， 帮助用户快速使用 CS32F103xB 开发板，评估 CS32F103xB 系列芯片的性能。

版 本

历史版本	修改内容	日期
V1.0	初版生成	2022-09-19

目 录

1 CS32F103xB 开发板 V1.1 硬件

开发板实物图和主要功能说明如下：

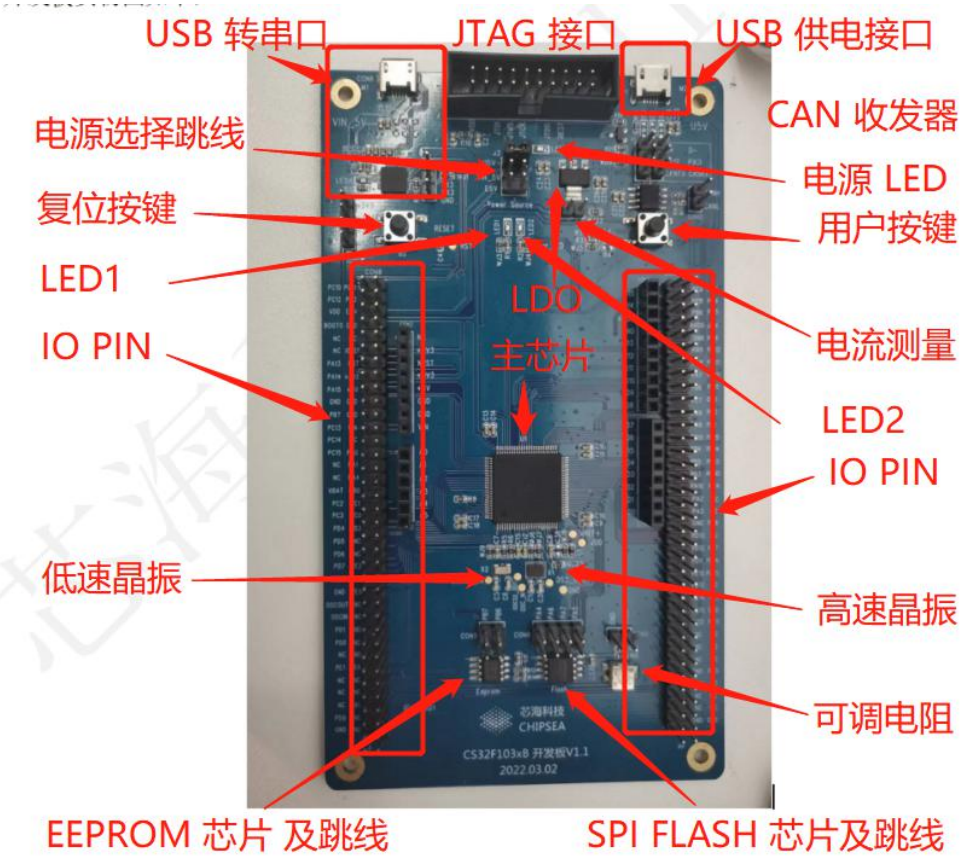


图 1 CS32F103xB 开发板 V1.1

开发上件芯片，连接器等信息如下：

表 1 芯片连接器说明表

芯片	U1	CS32F103VBT7	MCU
芯片	U3	AMS1117-3.3	LDO
芯片	U4	CP2102	USB to UART
芯片	U5	AT24C02C	EEPROM
芯片	U6	W25X16SSIG	SPI Flash
芯片	U7	TJA1050	CAN Transceiver
按键	B1	KEY	用户按键
按键	B2	KEY	复位按键
跳线/连接器	J1	3x1 单排排针	USB CAN 功能选择
跳线/连接器	J2	3x1 单排排针	UART3
跳线/连接器	J3	3X2 双排排针	电源跳线
跳线/连接器	J4	2x1 单排排针	电流测量端
跳线/连接器	J5	3x1 单排排针	USB CAN 功能选择
跳线/连接器	CON1	10X1 单排排母	Arduino 接口
跳线/连接器	CON2	8X1 单排排母	Arduino 接口
跳线/连接器	CON3	6X1 单排排母	Arduino 接口

跳线/连接器	CON4	8X1 单排排母	Arduino 接口
跳线/连接器	CON5	2X2 双排排针	EEPROM 芯片跳线
跳线/连接器	CON6	Micro-B 母座	Micro USB 转串口
跳线/连接器	CON7	2x10 牛角插座	J-Link 调试器接口
跳线/连接器	CON8	4X2 双排排针	SPI FLASH 芯片跳线
跳线/连接器	CON9	36X2 双排排针	
跳线/连接器	CON10	36X2 双排排针	
跳线/连接器	CON11	Micro-B 母座	Micro USB 供电
跳线/连接器	CON12	2x1 单排排针	电位器跳线
跳线/连接器	CON13	2x1 单排排针	CAN 总线
跳线/连接器	CON14	4x1 单排排针	3.3V
LED	LED1	GPIO 控制 LED	PB15
LED	LED2	GPIO 控制 LED	PB14
LED	LED3	电源 LED	5V，LDO 输入源

- 电源部分电路： 可以分别通过两个 USB 母座供电，也可以通过调试器 J-Link 给 开发板供电。

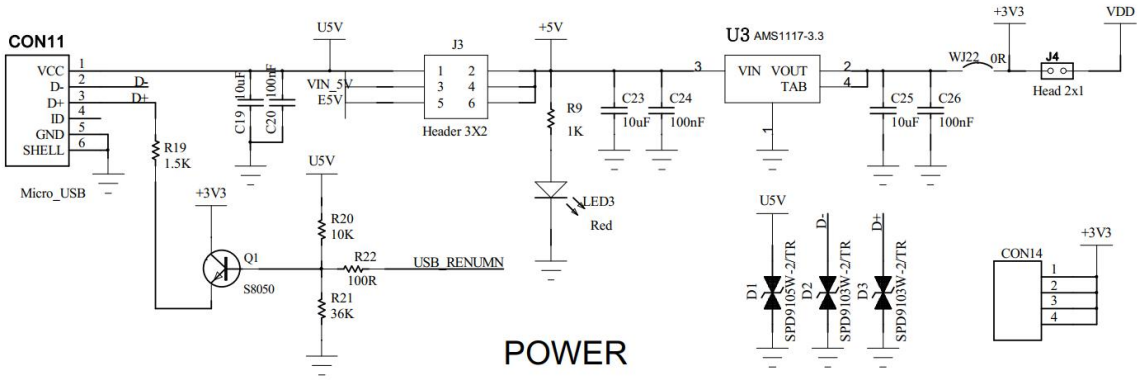


图 2 电源部分电路

- 调试器电路：支持 JTAG 和 SWD 模式， 调试器 VCC 可直接给 MCU 供电。

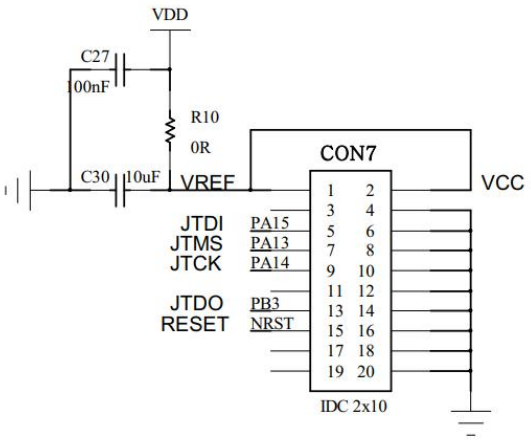


图 3 JTAG 电路

➢ USB 转串口电路： USB 通过 CP210 连接到 MCU USART3 的 PB10 PB11 。

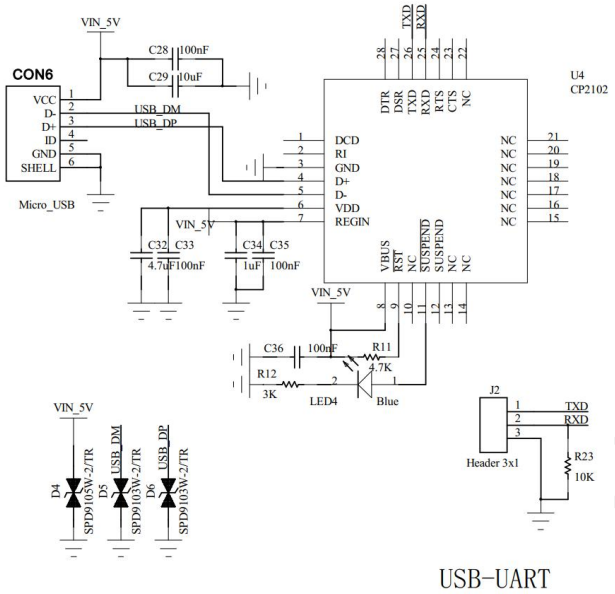


图 4 USB 转串口电路

➢ EEPROM 电路： I2C1 使用 PB6 和 PB7 两个 PIN 来连接问 EEPROM.

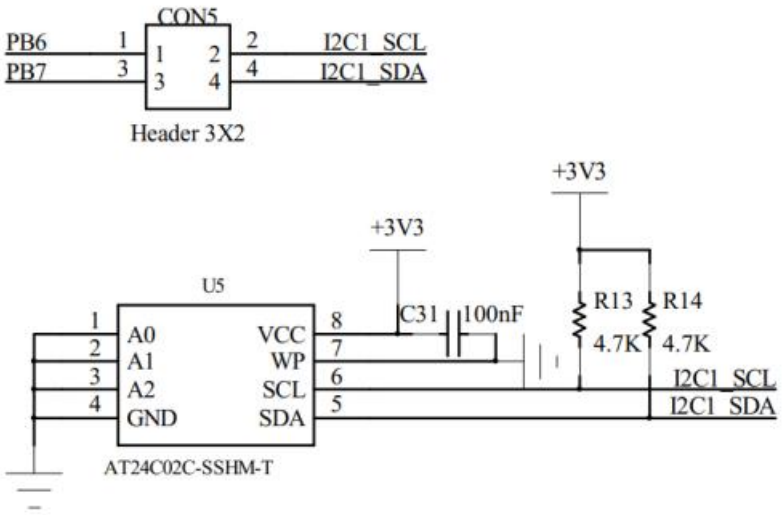


图 5 EEPROM 电路

- SPI FLASH 电路： SPI1 使用 PA4 PA5 A6 PA7 来连接 SPI FLASH

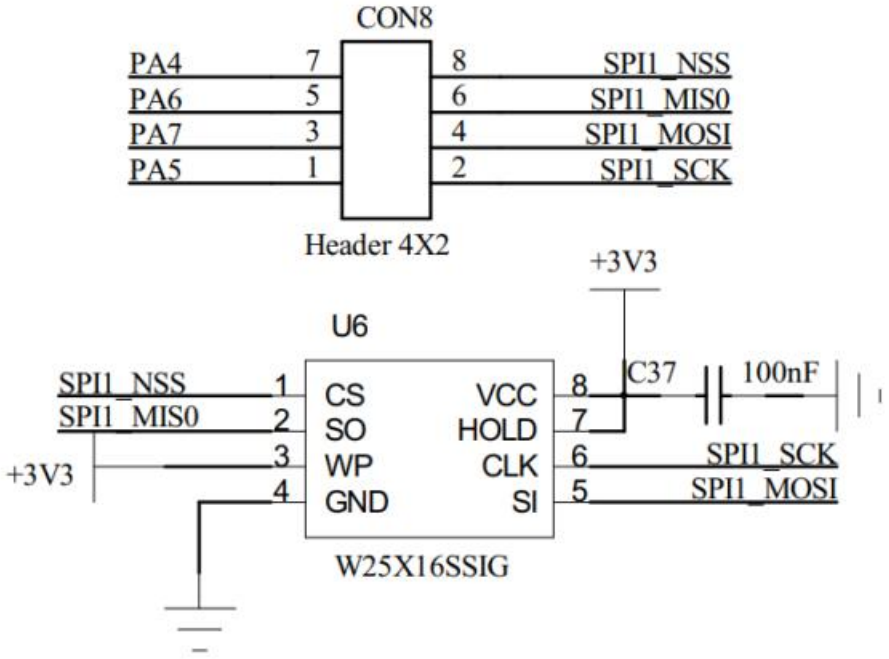


图 6　SPI FLASH 电路

- CAN 总线电路：MCU CAN 总线的 PA11 PA12 接到 CAN 收发器 TJA1050， 再通过 CON13 接到外部 CAN 总线上。

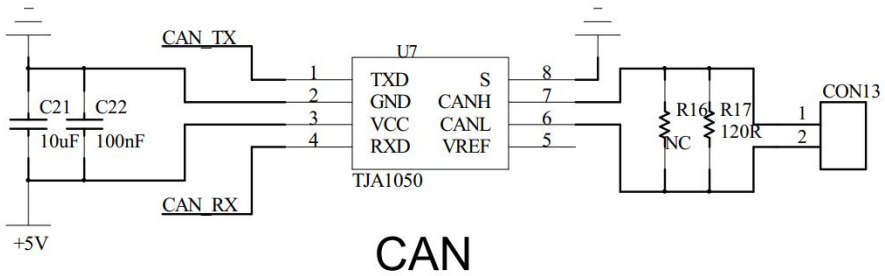


图 7　CAN 总线电路

- 电位器电路： 这个一个独立的可调分压电路，可把 CON2 的 PIN1 连接到 ADC 的输入，测量 ADC 采样值。

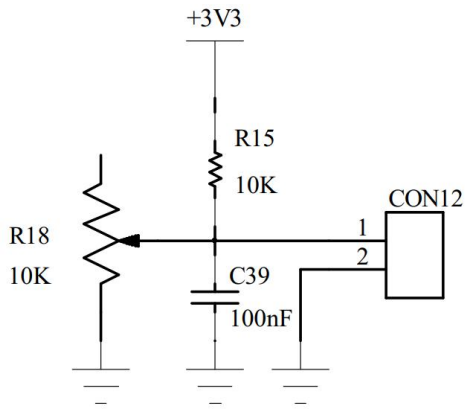
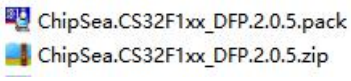


图 8　可调电位器

Pack 包安装和例程的编译下载的基本设置

CS32F103 最新 pack 包是 2.0.5， 把 后辍名称. pack 强制改成 .zip 解压，就可以得到例程文件。 双击 pack 包文件，可以进入 pack 包安装过程。



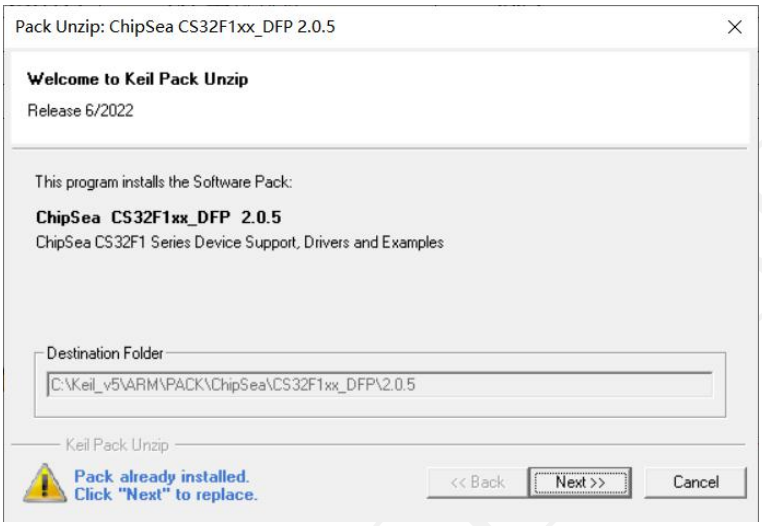


图 9 安装 Pack 包

安装完 Pack 后，要确认 KEIL 工程选的是最新的 Pack 版本。

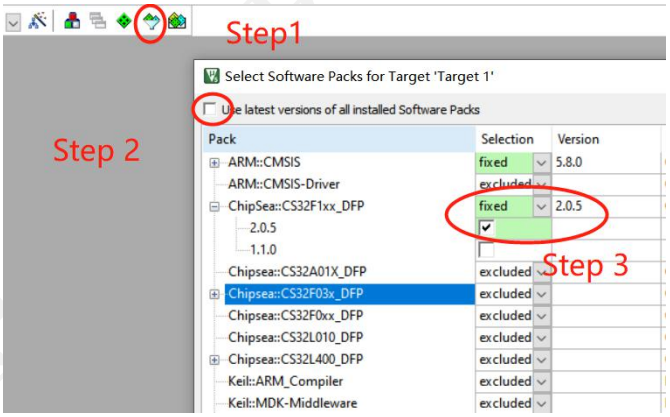


图 10 Pack 包的版本选择

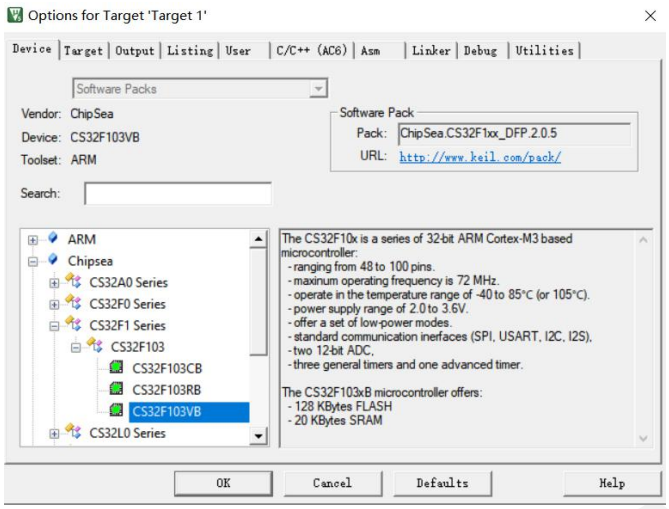


图 11　Device 中选择对应的型号

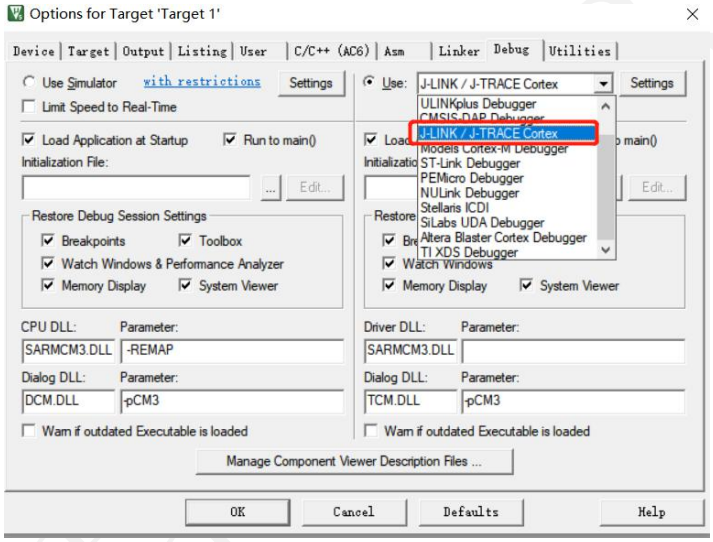


图 12　选择 J-Link 调试器

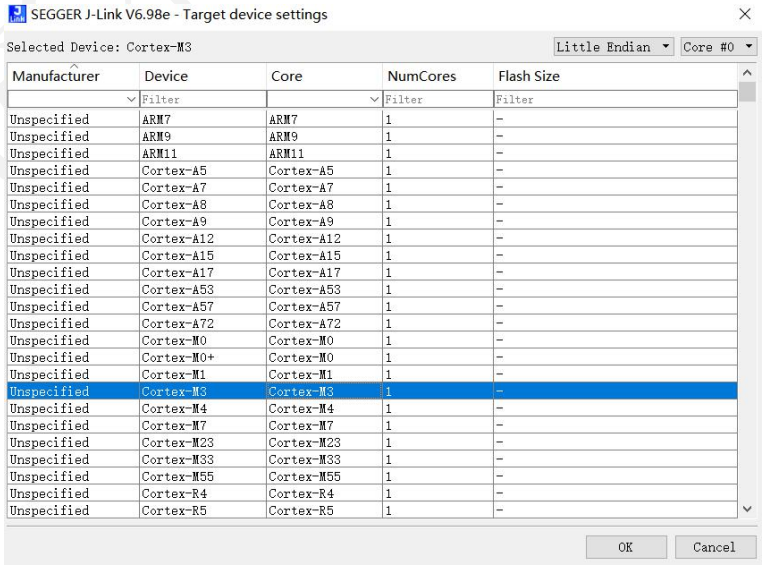


图 13　弹出型号选择，点 Cortex-M3

打开 GPIO_TOGGLE 例程

路径： ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\GPIO\GPIO_TOGGLE\Project

程序编译下载后，按下复位 B2 键， LED1 LED2 闪烁起来。

注意：此时开发板不做任何跳线， 只需要通过 J-Link 用排线连上开发板， MCU 就可以正常运行，LED1 LED2 可以正常闪烁起来。 J-Link 的 VCC 直接连接到 MCU 来供电，没有通过 LDO 来给 MCU 供电。

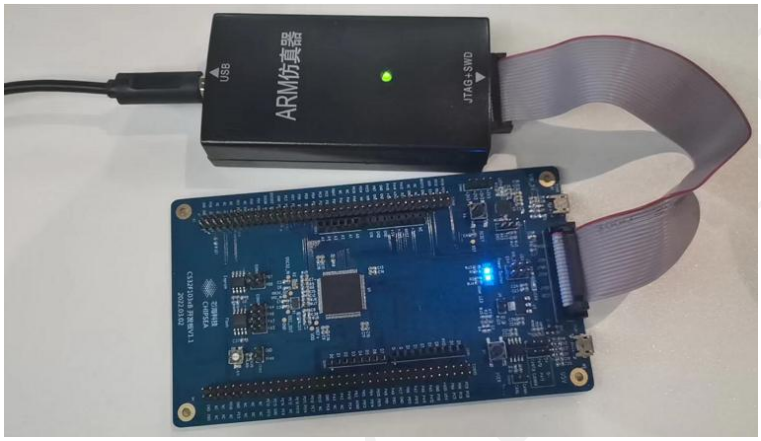


图 14　GPIO 闪灯例程

2 例程 - ADC_Base

例程：ADC_Base

路径：ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\ADC\ADC_Base\Project

硬件配置：ADC: PC0 (CON3.6， 可以将 PC0 连接到 CON12.1, 调节电阻分压，观察串口打印数据的变化)

UART3: PB11 (115200)

(可以通过 Micro USB 连接 CON6， 用板载 USB 转串口 观察 Printf 数据，也可以直接用 USB 转串口工具，接 J2.2)

说明 MCU 采集 PC0 上的电压并输出到串口。

Printf 信息如下：

```
AD conversion start...
ADC voltage = 1421 mV.
ADC voltage = 1420 mV.
ADC voltage = 1421 mV.
ADC voltage = 1420 mV.
ADC voltage = 1420 mV.
ADC voltage = 1421 mV.
ADC voltage = 1419 mV.
```

Main 函数如下：

```
int main(void)
{
  uint16_t value = 0 ;
  uint16_t voltage = 0;
  uint32_t i = 0xFFFFFF;

  /* Configure USART3 as the printing port. */
  usart_config();

  /* Configure ADC input pin(PC0). */
  adc_gpio_init();

  /* Configure ADC1. */
  adc_bsp_init();

  printf("AD conversion start...\n\r");

  while(1)
  {
    /* Start ADC1 Software Conversion. */
    __ADC_REG_CONV_START(ADC1);

    /* Wait for the conversion to complete. */
    while(RESET == __ADC_FLAG_STATUS_GET(ADC1, ADC_FLAG_EOC));

    /* Calculate the voltage value. */
    value = __ADC_CONV_VALUE_GET(ADC1);
    voltage = ((value * 3300) / 0xFFF);

    printf("ADC voltage = %d mV.\n\r", voltage);

    /* a short delay */
    while(i --);
    i = 0xFFFFF;
  }
}
```

UART 配置：PB10 PB11 配置成 USART3, 115200

```
void usart_config(void)
{
  usart_config_t ptr_usart;

  /* Enable clocks. */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOB);
  __RCU_APB1_CLK_ENABLE(RCU_APB1_PERI_USART3);
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_AFIO);

  /* Configure PB10(TX) and PB11(RX). */
  gpio_mode_config(GPIOB, GPIO_PIN_10, GPIO_MODE_OUT_AFPP(GPIO_SPEED_HIGH));
  gpio_mode_config(GPIOB, GPIO_PIN_11, GPIO_MODE_IN_PU);

  /* Configure the basic information of USART3. */
  usart_def_init(USART3);
  ptr_usart.baud_rate = 115200;
  ptr_usart.data_width = USART_DATA_WIDTH_8;
  ptr_usart.flow_control = USART_FLOW_CONTROL_NONE;
  ptr_usart.parity_check = USART_PARITY_NONE;
  ptr_usart.stop_bits = USART_STOP_BIT_1;
  ptr_usart.transceiver_mode = USART_MODE_TX_RX;
  usart_init(USART3, &ptr_usart);

  /* Enable USART3. */
  __USART_ENABLE(USART3);
}

int fputc(int ch, FILE *f)
{
  (void) f;
  while(__USART_FLAG_STATUS_GET(USART3, TC) == RESET);
  __USART_DATA_SEND(USART3, (uint8_t) ch);

  return ch;
}
```

ADC 配置：PC0 配置成 ADC ch10

```
void adc_gpio_init(void)
{
  /* Configure PC0 as an analog input port(ADC channel 10). */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOC);
  gpio_mode_config(GPIOC, GPIO_PIN_0, GPIO_MODE_IN_ANALOG);
}

void adc_bsp_init(void)
{
  /* Configure ADC1 clock. */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_ADC1);
  rcu_adcclk_config(RCU_ADCCLK_SEL_PCLK2_DIV8);

  adc_cfg_t ptr_cfg;

  /* Configure the basic information of ADC1. */
  __ADC_DEF_INIT(ADC1);
  adc_struct_init(&ptr_cfg);
  ptr_cfg.ext_trigger = ADC_EXT_TRIGGER_SWSTART;
  adc_init(ADC1, &ptr_cfg);

  /* ADC1 regular channel10 configuration. */
  adc_regular_channel_config(ADC1, ADC_CHANNEL_10, ADC_SAMPLE_TIME_55_5_CYCLE, 1);

  /* Enable ADC1. */
  __ADC_ENABLE(ADC1);

  /* Enable ADC1 reset calibration register. */
  __ADC_RESET_CALI(ADC1);
  /* Check the end of ADC1 reset calibration register. */
  while(__ADC_RESET_CALI_STATUS_GET(ADC1));

  /* Start ADC1 calibration. */
  __ADC_CALI_START(ADC1);
  /* Check the end of ADC1 calibration. */
  while(__ADC_CALI_STATUS_GET(ADC1));
}
```

3 例程 - CRC_Calculation

例程：ADC_Base

路径：CS32F1xx_DFP.2.0.5\Boards\Example\CRC\CRC_Calculation\Project

硬件配置：UART3: PB11 (115200)

(可以通过 Micro USB 连接 CON6， 用板载 USB 转串口 观察 Printf 数据，也可以直接用 USB 转串口工具，接 J2.2)

说明：MCU 计算数据的 CRC 并输出到串口。

Printf 信息如下：

```
\0CRC calculation result: 379e9f06{
```

Main 函数如下：

```
int main(void)
{
  /* Enable CRC clock */
  __RCU_AHB_CLK_ENABLE(RCU_AHB_PERI_CRC);

  usart_config();

  /* Compute the CRC of "data_buf" */
  crc_value = crc_data_buffer_calc((uint32_t *)data_buf, BUFFER_SIZE);

  printf("CRC calculation result: %x\r\n", crc_value);

  while(1)
  {
  }
}
```

4 例程 - EXTI

例程：EXIT

路径：ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\EXTI\Project

硬件配置：KEY: PC13

LED: LED1 LED2 PB14 PB15

说明：代码编译下载后，复位 MCU， 每次按下按键 B1, 开发板上 LED 1 LED2 会同时翻转。

Main 函数如下：

```
int main(void)
{
  /* nvic configuration */
  nvic_config();

  /* Initialization LED */
  led_init();

  exti_init();

  while(1)
  {
    delay(LED_DELAY);
  }
}
```

外部中断服务如下：

```
void EXTI15_10_IRQHandler(void)
{
  if(__EXTI_FLAG_STATUS_GET(EXTI_LINE_13) != RESET)
  {
    __EXTI_FLAG_CLEAR(EXTI_LINE_13);
    led1_toggle();
    led2_toggle();
  }
}
```

LED 初始化， 外部中断初化函数如下：

```
void led_init(void)
{
  /* Enable the clock */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOB);

  gpio_mode_config(GPIOB,GPIO_PIN_15,GPIO_MODE_OUT_PP(GPIO_SPEED_HIGH));
  gpio_mode_config(GPIOB,GPIO_PIN_14,GPIO_MODE_OUT_PP(GPIO_SPEED_HIGH));

  __GPIO_PIN_RESET(GPIOB,GPIO_PIN_15);   //LED1
  __GPIO_PIN_RESET(GPIOB,GPIO_PIN_14);   //LED2
}

void led1_toggle(void)
{
  GPIOB->DO ^= GPIO_PIN_15;
}

void led2_toggle(void)
{
  GPIOB->DO ^= GPIO_PIN_14;
}

void nvic_config(void)
{
 nvic_init_t nvic_struct = {0};
```

```
 nvic_priority_group_config(NVIC_PriorityGroup_2);

 /* Enable and configure interrupt channel */
 nvic_struct.nvic_irqchannel = IRQn_EXTI15_10;
 nvic_struct.nvic_irq_pre_priority = 0;
 nvic_struct.nvic_irq_sub_priority = 1;
 nvic_struct.nvic_irq_enable = ENABLE;
 nvic_init(&nvic_struct);
}

void exti_init(void)
{
  /* Enable the GPIOC clock */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOC);
  /* Enable the AFIO clock */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_AFIO);

  /* PC13 -- button */
  gpio_mode_config(GPIOC, GPIO_PIN_13, GPIO_MODE_IN_FLOAT);

  /* Config exti line to pin */
  gpio_exti_pin_config(GPIO_EXTI_EVEVT_PORT_GPIOC, GPIO_EXTI_EVENT_PIN13);

  /* Config rising detect */
  __EXTI_EDGE_ENABLE(EXTI_EDGE_RISING,EXTI_LINE_13);

  /* Enable the interrupt */
  __EXTI_INTR_ENABLE(EXTI_LINE_13);
}
```

5 例程 - GPIO_TOGGLE

例程：GPIO_TOGGLE

路径：ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\GPIO\GPIO_TOGGLE

硬件配置：LED: LED1 LED2 PB14 PB15

说明：代码编译下载，复位 MCU， 开发板上 LED1 LED2 会同步快速闪动起来。

Main 函数， LED 初始化函数如下：

```
}int main(void)
{
  /* Initialization LED */
  led_init();

  while(1)
  {
    delay(LED_DELAY);
    led1_toggle();
    led2_toggle();
  }
}

void led_init(void)
{
  /* Enable the clock */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOB);

  gpio_mode_config(GPIOB,GPIO_PIN_15,GPIO_MODE_OUT_PP(GPIO_SPEED_HIGH));
  gpio_mode_config(GPIOB,GPIO_PIN_14,GPIO_MODE_OUT_PP(GPIO_SPEED_HIGH));

    GPIO_PIN_RESET(GPIOB,GPIO_PIN_15);    //LED1
    GPIO_PIN_RESET(GPIOB,GPIO_PIN_14);    //LED2
}

void led1_toggle(void)
{
  GPIOB->DO ^= GPIO_PIN_15;
}

void led2_toggle(void)
{
  GPIOB->DO ^= GPIO_PIN_14;
}
```

6 例程 - RTC_RealTime

例程：RTC_RealTime

路径： ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\RTC\RTC_RealTime\Project

硬件配置：RTC

UART3: PB11 (115200)

(可以通过 Micro USB 连接 CON6， 用板载 USB 转串口 观察 Printf 数据，也可以直接用 USB 转串口工具，接 J2.2)

说明：代码编译下载，复位 MCU， 串口助手显示时间每一秒刷新一次，秒走满 60 清 0，分钟加一。

Printf 信息如下：

```
TIME: 08 : 00 : 00
TIME: 08 : 00 : 01
TIME: 08 : 00 : 02
TIME: 08 : 00 : 03
TIME: 08 : 00 : 04
```

Main 函数， RTC 配置函数如下：

```
int main(void)
{
  uint32_t hour = 0;
  uint32_t min = 0;
  uint32_t second = 0;
  uint32_t cnt_value = 0;

  /* Configure USART3 */
  usart_config();
  /* Configure RTC */
  rtc_config();
  /* Configure NVIC */
  nvic_config();

  while(1)
  {
    if(second_flag == 1)
    {
      /* Time is 23:59:59 */
      if(__RTC_COUNTER_GET() == TIME_RST)
      {
        /* Reset RTC Counter */
        rtc_counter_set(0);
        /* Wait until last write operation on RTC registers has finished */
        while(__RTC_FLAG_STATUS_GET(RTC_FLAG_OPERATION_COMPLETE) == RESET);
      }

      cnt_value = __RTC_COUNTER_GET();

      /* Compute  hours */
      hour = (cnt_value / 3600);
      /* Compute minutes */
      min = ((cnt_value % 3600) / 60);
      /* Compute seconds */
      second = ((cnt_value % 3600) % 60);

      printf("TIME: %0.2d : %0.2d : %0.2d \r\n", hour, min, second);
```

```
        second_flag = 0;
      }
    }
}

void rtc_config(void)
{
  /* Enable PMU clock */
  __RCU_APB1_CLK_ENABLE(RCU_APB1_PERI_PMU);

  /* Allow access to RTC Domain */
  pmu_vbat_domain_write_config(ENABLE);

  /* Enable LXT clock */
  __RCU_FUNC_ENABLE(LXT_CLK);
  /* Wait till LXT is ready */
  while( RESET == rcu_clkready_reset_flag_get(RCU_FLAG_LXT_STABLE));

  /* Select LXT as RTC Clock Source */
  rcu_rtcclk_config(RCU_RTCCLK_SEL_LXT);

  /* Enable RTC Clock */
  __RCU_RTC_CLK_ENABLE();

  /* Wait for RTC registers synchronization */
  rtc_wait_for_synchronize();
  /* Wait until last write operation on RTC registers has finished */
  while(__RTC_FLAG_STATUS_GET(RTC_FLAG_OPERATION_COMPLETE) == RESET);

  /* Enable the RTC Second interrupt*/
  __RTC_INTERRUPT_ENABLE(RTC_INTERRUPT_SECOND);
  /* Wait until last write operation on RTC registers has finished */
  while(__RTC_FLAG_STATUS_GET(RTC_FLAG_OPERATION_COMPLETE) == RESET);

  /* RTC period = RTCCLK/RTC_PR = (32.768 KHz)/(32767+1) */
  rtc_prescaler_set(32767);
  /* Wait until last write operation on RTC registers has finished */
  while(__RTC_FLAG_STATUS_GET(RTC_FLAG_OPERATION_COMPLETE) == RESET);

  /* Time starts at 8 o'clock in the morning */
  rtc_counter_set(AM_8);
  /* Wait until last write operation on RTC registers has finished */
  while(__RTC_FLAG_STATUS_GET(RTC_FLAG_OPERATION_COMPLETE) == RESET);
}
```

7 例程 - I2C_Eeprom

例程：I2C_Eeprom

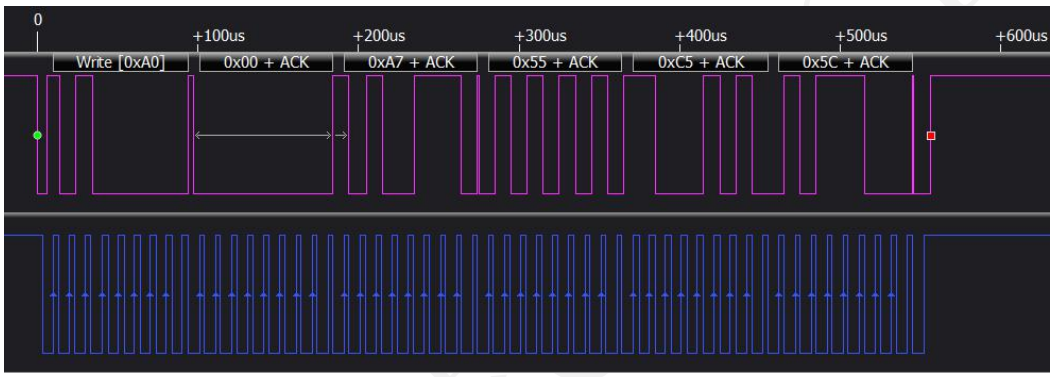
路径：ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\I2C\I2C_Eeprom\project

硬件配置：I2C1(I2C1 SCL:PB6 SDA: PB7)

通过 CON5 跳线连接板载 EEPEOM AT24C02

LED: LED2

说明：MCU 循环读写数据到 EEPROM, 第个循环 LED2 翻转一次。逻辑分析仪抓 I2C 时序如下：



Main 函数 以及 I2C 配置函数如下：

```
int main(void)
{
  uint8_t data[4]={0xA7,0x55,0xC5,0x5C};
  uint8_t i= 0;
  volatile uint8_t readBuf[4]={0x00};

  /*LED init*/
  led_init();
  /*delay init*/
  systick_delay_init();

  /*I2C configure*/
  i2c_configure();
  /*gpio configure*/
  gpio_configure();

  while(1)
  {
    /*Eeprom write*/
    ret = write_at24c02(0x00,data,4);
    /*Delay*/
    systick_delay_ms(10);
    /*Eeprom read*/
    ret = read_at24c02(0x00,readBuf,4);
    /*Delay*/
    systick_delay_ms(10);
    /*LED toggle*/
    led2_toggle();

    for(i = 0; i < 4; i++)
    {
      readBuf[i] = 0;
    }
  }
}
```

```
void gpio_configure(void)
{
  /* Enable the clock */
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOB);
        /*SCL*/
  gpio_mode_config(GPIOB,GPIO_PIN_6,GPIO_MODE_OUT_AFOD(GPIO_SPEED_MEDIUM));
        /*SDA*/
  gpio_mode_config(GPIOB,GPIO_PIN_7,GPIO_MODE_OUT_AFOD(GPIO_SPEED_MEDIUM));
}

void i2c_configure(void)
{
  i2c_config_t i2c_struct;

  __RCU_APB1_CLK_ENABLE(RCU_APB1_PERI_I2C1);

//   __I2C_SW_RESET(I2C1);
//   __I2C_NOT_RESET(I2C1);

  /* initialize the I2C mode */
  i2c_struct.mode = I2C_MODE_I2C;
  /* initialize the I2C speed 100KHz */
  i2c_struct.speed = 100000;
  /* initialize the I2C duty cycle Tlow/Thigh = 2 */
  i2c_struct.duty_cycle = I2C_DUTY_CYCLE_2;
  /* initialize the I2C address1 */
  i2c_struct.address1 = 0xA0;
  /* initialize the I2C ack enable */
  i2c_struct.ack = I2C_ACK_ENABLE;
  /* initialize the I2C address mode */
  i2c_struct.addr_mode = I2C_ADDRESS_MODE_7BITS;

  i2c_init(I2C1,&i2c_struct);

  /*Enable I2C*/
   __I2C_ENABLE(I2C1);
}
```

8 例程 - SPI_FLASH

例程：SPI_FLASH

路径：ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\SPI\SPI_FLASH\Project

硬件配置：SPI (SPI1: PI_SCK: PA5 SPI_MISO: PA6 SPI_MOSI: PA7 CS: PA4)

通过 CON8 跳线连接 板载 SPI FLASH W25X16

UART3: PB11 (115200)

(可以通过 Micro USB 连接 CON6， 用板载 USB 转串口 观察 Printf 数据，也可以直接用 USB 转串口工具，接 J2.2)

说明：MCU 循环读写 SPI FLASH，并对比数据，通过串口提示是否读写成功。

Printf 信息如下：

```
\0SPI FLASH W25Q64 operation Success!
```

Main 函数，及 SPI 配置函数如下：

```
int main(void)
{
  uint8_t spi_wbuf[] = "SPI test!";
  uint8_t spi_rbuf[16];
  uint8_t tx_buffer[60];
  uint8_t print_len = 0;

  uint8_t len = 0;

  memset(spi_rbuf,0,16);
  memset(tx_buffer,0,60);
  len = sizeof(spi_wbuf);

  usart_config();
  spi_flash_init();
  delay_us( 20000 );

  spi_flash_wait_sector_erase(0x00000);

  spi_flash_buffer_write(spi_wbuf, 0x00000, len);
  spi_flash_buffer_read(spi_rbuf, 0x00000, len);
  if(buf_compare(spi_wbuf, spi_rbuf, len) == 1)
  {
    print_len = (uint8_t)sprintf((char *)tx_buffer, "SPI FLASH W25Q64 operation Success!");
    usart_send(tx_buffer, print_len);
  }
  else
  {
    print_len = (uint8_t)sprintf((char *)tx_buffer, "SPI FLASH W25Q64 operation Failed!");
    usart_send(tx_buffer,print_len);
  }

  while(1)
  {

  }
}

void spi_flash_init(void)
{
  spi_config_t  spi_config_struct;

  __SPI_DEF_INIT(SPI1);
```

```
    __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOA);
    __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_SPI1);

    // SPI GPIO Config
    gpio_mode_config(GPIOA, GPIO_PIN_5, GPIO_MODE_OUT_AFPP(GPIO_SPEED_HIGH)); //SD_SPI_SCK
    gpio_mode_config(GPIOA, GPIO_PIN_6, GPIO_MODE_IN_PU);                     //SD_SPI_MISO
    gpio_mode_config(GPIOA, GPIO_PIN_7, GPIO_MODE_OUT_AFPP(GPIO_SPEED_HIGH)); //SD_SPI_MOSI

    gpio_mode_config(GPIOA, GPIO_PIN_4, GPIO_MODE_OUT_PP(GPIO_SPEED_HIGH));

    __GPIO_PIN_SET(GPIOA, GPIO_PIN_4);

    //SPI Config
    spi_config_struct.spi_direct = SPI_DIR_2LINES_FULL_DUPLEX;
    spi_config_struct.spi_mode = SPI_MODE_MASTER;
    spi_config_struct.data_width = SPI_DATA_WIDTH_8BIT;
    spi_config_struct.spi_cpol = SPI_CPOL_HIGH;
    spi_config_struct.spi_cpha = SPI_CPHA_2EDGE;
    spi_config_struct.spi_nss = SPI_SSM_SW;
    spi_config_struct.spi_predivid = SPI_BAUD_RATE_PDIV_32;
    spi_config_struct.first_bit = SPI_FIRST_BIT_MSB;
    spi_config_struct.crc_polynomial = 7;
    spi_init(SPI1, &spi_config_struct);

    spi_software_nss_config(SPI1,SPI_SOFTWARE_NSS_SET);

    __SPI_ENABLE(SPI1);
}
```

9 例程 - SysTick\Interrupt

例程：SysTick\Interrupt

路径：ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\SysTick\Interrupt\Project

硬件配置：Systick

LED: LED1 LED2 PB14 PB15

说明： 代码编译下载，复位 MCU。 LED1 LED2 同步闪烁起来。通过 Systick 延时来定时翻转 LED.

Main 函数，Systick 中断服务函数，延时函数如下：

```
int main(void)
{
  /* Init RCU configuration */
  rcu_init();

  /* Init LED */
  led_init();

  if (SysTick_Config(SystemCoreClock / 1000))
  {
    /* error */
    while (1);
  }
  while(1)
  {
    systick_delay_int(500);
    led1_toggle();
    led2_toggle();
  }
}

void SysTick_Handler(void)
{
  /* User code */
  systick_decrement();
}

void systick_delay_int(__IO uint32_t count)
{
  delay_ticks = count;

  while(delay_ticks != 0);
}

void systick_decrement(void)
{
  if(delay_ticks != 0)
  {
    delay_ticks --;
  }
}
```

10 例程 - FWDT

例程：FWDT

路径：ChipSea.CS32F1xx_DFP.2.0.5\Boards\Example\FWDT\Project

硬件配置：FWDT

LED: LED2

说明：代码编译下载，复位 MCU。 LED2 会亮起来。如果增加延时 systick_delay_ms(300); MCU 会一直复位 LED2 不会亮。

Main 函数，FWDT 配置函数如下：

```
int main(void)
{
  /* Initialization LED */
  led_init();
  systick_delay_init();
  fwdt_init(0xFFF,FWDT_PRESCALER_4);
  while(1)
  {

    fwdt_reload_counter();
    systick_delay_ms(200);
//    systick_delay_ms(300);   // 如果增加这段代码，会造成 FWDT 超时， LED 不会亮
    led1_on();
    led2_on();

  }
}

void fwdt_init(uint16_t reload_value,uint8_t prescaler_value)
{
  fwdt_write_access_enable_ctrl(FWDT_WRITE_ACCESS_ENABLE);
  fwdt_prescaler_set(prescaler_value);
  fwdt_reload_set(reload_value);
  fwdt_enable();
}
```

11 例程 - TIM\PWM_Output

例程：TIM\PWM_Output

路径：CS32F1xx_DFP.2.0.5\Boards\Example\TIM\PWM_Output\Project

硬件配置：TIM2 PA0

说明：代码编译下载，复位 MCU。 PA0 上会有 PWM 输出，周期为 1 秒，点空比为 25% LED2 会亮起来。

Main 函数，TIM2 配置函数如下：

```
int main(void)
{
  gpio_config();

  tim_config();

  while(1);
}

void tim_config(void)
{
  tim_base_t ptr_time = {0};
  tim_choc_t ptr_choc = {0};

  /* PCLK1 = HCLK/2 */
  rcu_pclk1_config(RCU_HCLK_DIV_2);
  /* PCLK2 = HCLK/2 */
  rcu_pclk2_config(RCU_HCLK_DIV_2);

  /* TIM clock enable */
  __RCU_APB1_CLK_ENABLE(RCU_APB1_PERI_TIM3);
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_TIM1);
  __RCU_APB1_CLK_ENABLE(RCU_APB1_PERI_TIM2);

  /* Time base configuration */
  ptr_time.clk_div = TIM_CLK_DIV1;
  ptr_time.cnt_mode = TIM_CNT_MODE_UP;
  ptr_time.period = 10000;
  ptr_time.pre_div = 7199;
  tim_base_init(TIM2, &ptr_time);

  /* PWM1 Mode configuration: Channel1 */
  ptr_choc.oc_mode = TIM_OCMODE_PWM1;
  ptr_choc.output_state = TIM_OUTPUT_EN;
  ptr_choc.channel = TIM_CHANNEL_1;
  ptr_choc.polarity = TIM_OUTPUT_POLARITY_HIGH;
  ptr_choc.pulse = 2500;
  tim_choc_init(TIM2, &ptr_choc);

  tim_choc_preload_set(TIM2, TIM_CHANNEL_1, TIM_CHXOC_PRELOAD_ENABLE);

  __TIM_FUNC_ENABLE(TIM2, UVAL_SHADOW);

  /* TIM enable */
  __TIM_ENABLE(TIM2);
}

void gpio_config(void)
{
  __RCU_APB2_CLK_ENABLE(RCU_APB2_PERI_GPIOA);
  gpio_mode_config(GPIOA, GPIO_PIN_0, GPIO_MODE_OUT_AFPP(GPIO_SPEED_HIGH));
}
```

免责声明和版权公告

本文档中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

本文档可能引用了第三方的信息，所有引用的信息均为“按现状”提供，芯海科技不对信息的准确性、真实性做任何保证。

芯海科技不对本文档的内容做任何保证，包括内容的适销性、是否适用于特定用途，也不提供任何其他芯海科技提案、规格书或样品在他处提到的任何保证。

芯海科技不对本文档是否侵犯第三方权利做任何保证，也不对使用本文档内信息导致的任何侵犯知识产权的行为负责。本文档在此未以禁止反言或其他方式授予任何知识产权许可，不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。蓝牙标志是 Bluetooth SIG 的注册商标。

文档中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

芯海科技
CHIPSEA
股票代码:688595