SPC1168 读写 AT24C02 EEPROM 使用指南

概述

AT24C02 是一个 2K Bit 的串行 EEPROM 存储器(掉电不丢失),内部含有 256 个字节。在 AT24C02 里面有一个 8 字节的页写缓冲器，具有 I2C 接口和用于硬件数据保护的写保护引脚（WP），最大写周期为 10ms。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-06-20	CanChai	Outdated	首次发布。
C/0	2024-03-26	Jiali Zhou	Released	修改排版格式。

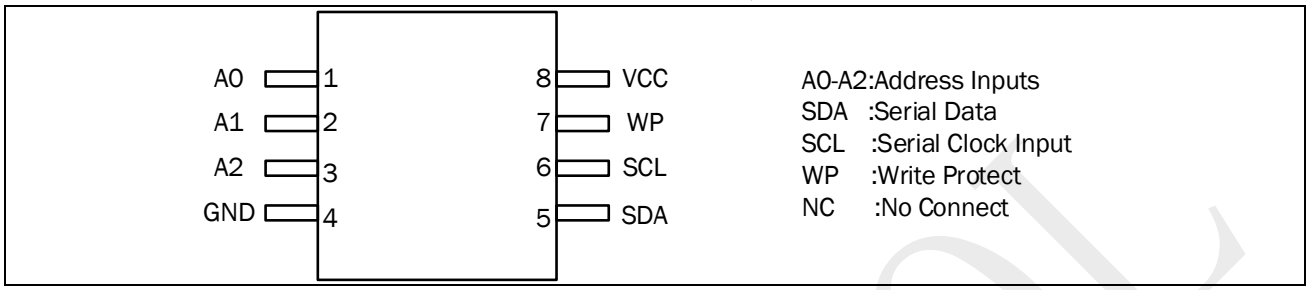
术语或缩写

术语或缩写	描述
/	/

1 使用方式简介

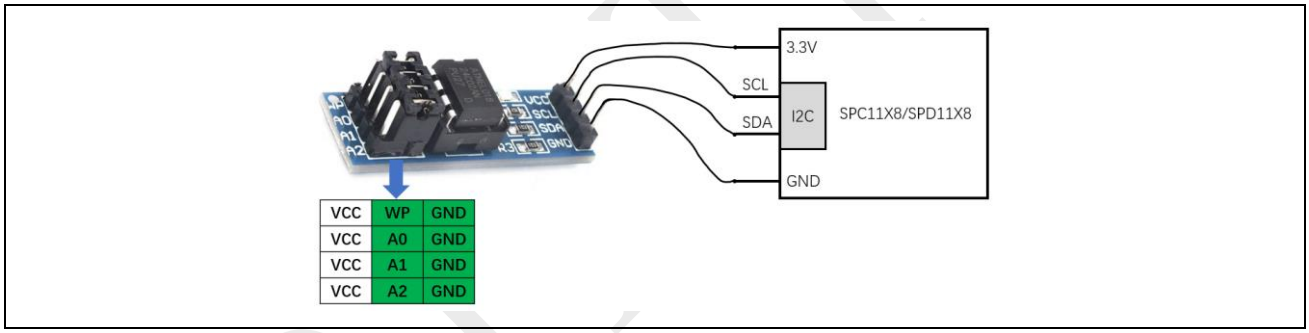
AT24C02 的引脚配置如图 1-1 所示。

图 1-1：24C02 引脚配置



SPC1168 内置有 IIC 单元，可与 EEPROM（AT24C02）建立 I2C 通信，硬件连接示意如图 1-2 所示。

图 1-2：硬件链接示意图



在 AT24Cxx 的驱动库中，已经有下列驱动函数可供调动，可方便用户使用和理解。表 1-1 和表 1-2 中 AT24Cxx 表示 EEPROM 模块编号，取值为 AT24C02。

表 1-1：AT24C02 宏定义列表

宏名	功能及参数说明
EEPROM_PAGE_SIZE	配置 AT24C02 的页字节数
EEPROM_SPEED	配置 I2C 协议速率
EEPROM_TIMEOUT	配置 AT24C02 超时检测
GPIOx_SCL_PIN	选择 GPIOx 引脚为 SCL
I2C_SCL_FUNC	配置 GPIOx 引脚为 SCL 功能
GPIOx_SDA_PIN	选择 GPIOx 引脚为 SDA
I2C_SDA_FUNC	配置 GPIOx 引脚为 SDA 功能
EEPROM_DEBUG_ON	使能 AT24C02 的调试打印功能
EEPROM_INFO(fmt,arg...)	打印 AT24C02 输出信息
EEPROM_ERROR(fmt,arg...)	打印 AT24C02 输出出错信息

EEPROM_DEBUG(fmt,arg...)	打印 AT24C02 输出调试信息
EEPROM_DEVICE_ADDR	配置 AT24C02 的设备地址
EEPROM_DEVICE_ADDR_A0	配置 AT24C02 的引脚 A0
EEPROM_DEVICE_ADDR_A1	配置 AT24C02 的引脚 A2
EEPROM_DEVICE_ADDR_A2	配置 AT24C02 的引脚 A2

表 1-2：AT24C02 函数定义列表

函数名	功能及参数说明
ErrorStatus AT24C02_Init(void)	初始化 24C02 模块
ErrorStatus AT24C02_WriteByte(uint8_t u8Addr, uint8_t u8Data)	向 24C02 发送一个字节数据 u8Addr：要写入的 24C02 存储地址 u8Data：要发送的字节数据
ErrorStatus AT24C02_ReadByte(uint8_t u8Addr, uint8_t* pu8Data)	接收 24C02 一个字节数据 u8Addr：要读取的 24C02 存储地址 pu8Data：接收数据缓冲区地址
ErrorStatus AT24C02_WritePage(uint8_t* pu8Data, uint8_t u8Addr, uint8_t u8Size)	向 24C02 发送一页（8 字节）数据 pu8Data：发送数据缓冲区地址 u8Addr：要写入的 24C02 存储地址 u8Size：发送数据数量
ErrorStatus AT24C02_Write(uint8_t* pu8Data, uint8_t u8Addr, uint16_t u16Size)	向 24C02 发送多字节数据 pu8Data：发送数据缓冲区地址 u8Addr：要写入的 24C02 存储地址 u16Size：发送数据数量
ErrorStatus AT24C02_Read(uint8_t* pu8Data, uint8_t u8Addr, uint16_t u16Size)	从 24C02 接收多字节数据 pu8Data：接收数据缓冲区地址 u8Addr：要读取的 24C02 存储地址 u16Size：发送数据数量

2 应用示例

2.1 AT24C02 单字节读写示例

按单字节读写 AT24C02 模块示例代码如下：

Example Code

```
#define AT24C02_START_ADDR    0x00
uint8_t au8WriteByteBuf[1] = {0xAB};
uint8_t au8ReadByteBuf[1] = {0x00};

int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();
  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
  Delay_Init();

  /* Configure GPIO pin as UART function */
  GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);

  /* UART Init */
  UART_Init(UART, 38400);

  /* AT24C02 Init */
  AT24C02_Init();

/* Read and write a byte data to AT24C02 */
  AT24C02_Byte_Test();

  while(1){}
}
```

在 AT24C02_Byte_Test 测试函数中，对 AT24C02 进行单字节数据的读写和数据比对。

Example Code

```
ErrorStatus AT24C02_Byte_Test(void)
{
    ErrorStatus eStatus;
    EEPROM_INFO("Write a byte data: ");

    printf("0x%02X \n", au8WriteByteBuf[0]);

    /* Send one byte of data to 24C02 */
    eStatus = AT24C02_WriteByte(EEPROM_PAGE_SIZE, au8WriteByteBuf[0]);
    if(eStatus == ERROR)
    {
        EEPROM_DEBUG("[Write Data ERROR]\n");
        return ERROR;
    }
    else
    {
        EEPROM_DEBUG("[Write Data SUCCESS]\n");
    }

    EEPROM_INFO("Read a byte data: ");

    /* Read one byte of data from 24C02 */
    AT24C02_ReadByte(EEPROM_PAGE_SIZE, au8ReadByteBuf);

    printf("0x%02X \n", au8ReadByteBuf[0]);

    /*  To check the data sent and recieved are the same */
    if(au8ReadByteBuf[0] != au8WriteByteBuf[0])
    {
        EEPROM_INFO("ERROR: AT24C02 inconsistent data was written and read");
        return ERROR;
    }
    else
    {
        EEPROM_INFO("AT24C02 read and write a byte test successful\n");
        return SUCCESS;
    }
}
```

2.2 AT24C02 多字节读写示例

按多字节读写 AT24C02 模块示例代码如下：

Example Code

```
#define AT24C02_START_ADDR 0x00
uint8_t au8WriteBuf[256];
uint8_t au8ReadBuf[256];

int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();
  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
  Delay_Init();

  /* Configure GPIO pin as UART function */
  GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);

  /* UART Init */
  UART_Init(UART, 38400);

  /* AT24C02 Init */
  AT24C02_Init();

  /* Reads and writes multiple bytes data to AT24C02 */
  AT24C02_MultiByte_Test();

  while(1){}
}
```

在 AT24C02_MultiByte_Test 测试函数中，对 AT24C02 进行多字节数据的读写和数据比对。

Example Code

```
ErrorStatus AT24C02_MultiByte_Test(void)
{
    uint16_t i;
    ErrorStatus eStatus;

    EEPROM_INFO("Write data:");

    /* Initialize au8WriteBuf and au8ReadBuf*/
    for ( i = 0; i < 256; i++ )
    {
        au8WriteBuf[i] = i;
        au8ReadBuf[i] = 0xFF;

        printf("0x%02X ", au8WriteBuf[i]);

        if(i % 16 == 15)
        {
            printf("\n\r");
        }
    }
```

```
    /* Write 256 bytes of data sequentially to AT24C02 */
    eStatus = AT24C02_Write(au8WriteBuf, AT24C02_START_ADDR, 256);
    if(eStatus == ERROR)
    {
        EEPROM_DEBUG("[Write Data Error]\n");
        return ERROR;
    }
    else
    {
        EEPROM_DEBUG("[Write Data Success]\n");
    }

    EEPROM_INFO("Read data:");

    /* Read 256 bytes of data sequentially from AT24C02 */
    AT24C02_Read(au8ReadBuf, AT24C02_START_ADDR, 256);

    for (i = 0; i < 256; i++)
    {
        if(au8ReadBuf[i] != au8WriteBuf[i])
        {
            printf("0x%02X ", au8ReadBuf[i]);
            EEPROM_INFO("ERROR: AT24C02 inconsistent data was written and read");

            return ERROR;
        }

        printf("0x%02X ", au8ReadBuf[i]);

        if(i % 16 == 15)
        {
            printf("\n\r");
        }
    }

    EEPROM_INFO("AT24C02 read and write all test successful\n");

    return SUCCESS;
}
```