片内存储器

1. 实验目的

- 掌握 STC15W4K32S4 系列单片机片内存储器类型及各类型存储器之间的区别。
- 掌握单片机读写内部 EEPROM、内部 FLASH 时的寄存器配置及程序设计。

2. 实验内容

- 编写程序实现对片内 EEPROM 读写的程序设计。（针对有片内 EEPROM 的型号）
- 编写程序实现对片内 FLASH 读写的程序设计。

3. 硬件设计

3.1. 有关存储器概念介绍

RAM（全称是 Random Access Memory）随机存储内存常称之为随机存储器，这种存储器在断电时将丢失其存储内容，故主要用于存储短时间使用的变量或程序。他的优势是可以随时读写，而且速度很快，通常作为操作系统或其他正在运行中的程序的临时数据存储媒介。

ROM（全称是 Read-Only Memory）只读内存常称之为只读存储器，是一种只能读出事先所存数据的固态半导体存储器，他不像随机存储器那样可以快速、方便地改写存储于其中的数据。他的优势是所存数据稳定，断电后存储的数据也不会丢失。

EPROM（全称是 Erasable Programmable Read-Only Memory）可擦除可编程只读存储器是一种可重写的存储器，并且其内容在掉电的时候也不会丢失。换句话说，他是非易失性的。他通过 EPROM 编程器进行编程，EPROM 编程器能够提供比正常工作电压更高的电压对 EPROM 编程。一旦经过编程，EPROM 只有在强紫外线的照射下才能够进行擦除。为了进行擦除，EPROM 的陶瓷封装上具有一个小的石英窗口，这个石英窗口一般情况下使用不透明的粘带覆盖，当擦除时将这个粘带揭掉，然后放置在强紫外线下大约 20 分钟。

EEPROM（全称是 Electrically Erasable Programmable Read-Only Memory）电可擦除可编程只读存储器的内容在掉电的时候也不会丢失。在平常情况下，EEPROM 与 EPROM 一样是只读的，需要写入时，在指定的引脚加上一个高电压即可写入或擦除，而且其擦除的速度极快。通常 EEPROM 芯片又分为串行 EEPROM 和并行 EEPROM 两种，串行 EEPROM 在读写时数据的输入/输出是通过 2 线、3 线、4 线或 SPI 总线等接口方式进行的，而并行 EEPROM 的数据输入/输出则是通过并行总线进行的。

FLASH 闪存结合了 ROM 和 RAM 的长处，不仅具备电可擦除可编程（EEPROM）的性能，还不会断电丢失数据、同时可以快速读取数据，所以现如今大多数单片机片内都集成了这种存储器。又因为 FLASH 相对于 EEPROM（又称为 E2PROM）成本要低得多，所以单片机片内往往会集成较大空间的 FLASH 存储器供用户使用。单片机的应用中常常将开发调试成功后的应用程序存储在程序存储器中，而 FLASH 存储器恰恰是非常理想的程序存储器。

FRAM（全称是 Ferroelectric Random Access Memory）铁电体随机存储内存常称之为铁电存储器，这种存储器的核心技术是铁电晶体材料，这一特殊材料使得铁电存储器同时拥有随机存储器(RAM)和非易失性存储器的特性。

根据存储器是否在单片机内部，存储器可分为片内存储器和片外存储器。下面各存储器都是片外存储器。

图 1：外部存储器实物图

✧ 注：根据存储器与单片机之间的接口是并行还是串行，又可将存储器分为串行存储器和并行存储器。

3.2. STC15W4K32S4 系列单片机片内存储器介绍

STC15W4K32S4 系列单片机片内有 4096 字节（一般为方便描述会说成 4K 字节）的数据存储器、16KB~61KB 空间不等的 FLASH 程序存储器和 2KB~42KB 空间不等的 E2PROM 程序存储器。如下表所示。

表 1：STC15W4K32S4 系列单片机片内存储器

序号	单片机型号	SRAM	FLASH	E2PROM	备注
1	STC15W4K16S4	4K 字节	16K 字节	42K 字节	无仿真功能
2	STC15W4K32S4	4K 字节	32K 字节	26K 字节	无仿真功能
3	STC15W4K40S4	4K 字节	40K 字节	18K 字节	无仿真功能
4	STC15W4K48S4	4K 字节	48K 字节	10K 字节	无仿真功能
5	STC15W4K56S4	4K 字节	56K 字节	2K 字节	无仿真功能
6	IAP15W4K58S4	4K 字节	58K 字节	0K 字节	有仿真功能
7	IAP15W4K61S4	4K 字节	61K 字节	0K 字节	有仿真功能
8	IRC15W4K63S4	4K 字节	63.5K 字节	0K 字节	有仿真功能

■ SRAM 和 DRAM 区别

SRAM（全称是 Static Random Access Memory）静态随机存储内存常称之为静态存储器，所谓的“静态”，是指这种存储器只要保持通电，里面储存的数据就可以恒常保持。

DRAM（全称是 Dynamic Random Access Memory）动态随机存储内存常称之为动态存储器，所谓的“动态”，是指这种存储器每隔一段时间，要刷新充电一次，否则内部的数据即会消失。

SRAM 和 DRAM 都属于 RAM 存储器，在断电后存储的数据会全部丢失。SRAM 优势是只需刷新充电一次，而 DRAM 则每隔一段时间就要刷新充电一次。SRAM 劣势是相对 DRAM 集成度较低，功耗大，体积大，价格高。

✧ 注：STC15W4K32S4 单片机片内集成的是 SRAM。

■ STC15W4K32S4 系列单片机片内 SRAM

与程序存储器对应的另外一种存储器叫数据存储器，数据存储器可暂存运行期间的数据、中间结果、缓冲和标志位等，数据存储器一般选择 RAM 来存储数据。

STC15W4K32S4 系列单片机片内 RAM 的类型是静态存储器 SRAM，片内 SRAM 大小是 4096 字节，该 4096 字节 SRAM 在物理和逻辑上都分为 2 个地址空间：内部 RAM（256 字节）和内部扩展 RAM（3840 字节）。

STC15W4K32S4 系列单片机 256 字节的内部 RAM 分 3 个部分：低 128 字节 RAM（与传统 8051 兼容）、高 128 字节 RAM（Intel 在 8052 中扩展了高 128 字节 RAM）及特殊功能寄存器区。高 128 字节 RAM 与特殊功能寄存器区貌似共用相同的地址范围，但物理上是独立的，使用时通过不同的寻址方式加以区分。如下图所示。

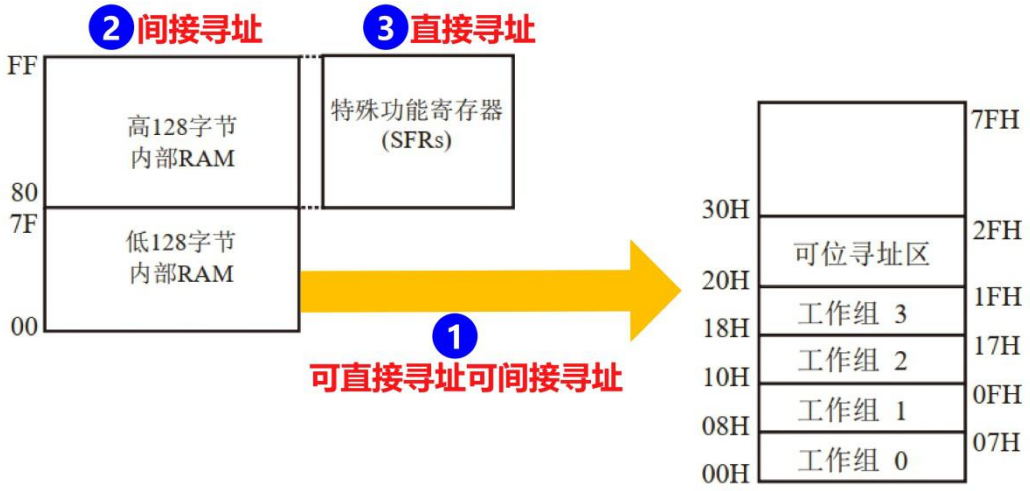

图 2：STC15W4K32S4 系列单片机内部 RAM 分配示意图

✧ 注：直接寻址、间接寻址、立即寻址，是 CPU 在通过总线与内存交互时存在不同的交互方法，而产生的三种概念词。这里不详细介绍，学过汇编的同学会非常容易理解他们之间的区别。

STC15W4K32S4 系列单片机片内除了集成 256 字节 SRAM 外，内部还集成了 3840 字节的 SRAM（常称之为内部扩展 RAM），地址范围是：0000H~0EFFH。访问内部扩展 RAM 的方法和传统 8051 单片机访问外部扩展 RAM 的方法相同，但是 STC15W4K32S4 系列单片机访问内部扩展 RAM 不会影响 P0 口、P2 口、WR 引脚、RD 引脚和 ALE 引脚。

STC15W4K32S4 系列单片机片内扩展 RAM 是否可以访问受辅助寄存器的 B1 位控制，如下图所示。

AUXR : Auxiliary register (不可位寻址)

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	name	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2

① 寄存器名

② 内部/外部RAM控制位

EXTRAM：内部/外部RAM存取控制位。
0：允许使用逻辑上在片外、物理上在片内的扩展RAM；
1：禁止使用逻辑上在片外、物理上在片内的扩展RAM。

图 3：辅助寄存器 AUXR

✧ 注：一般 EXTRAM 位默认是 0，即允许使用逻辑上在片外、物理上在片内的扩展 SRAM。

■ STC15W4K32S4 系列单片机片内 E2PROM

在项目应用中，往往会有一些很重要的数据是需要掉电不丢失的，这些数据需要被保存到非易失性的存储器里面，这些非易失性的存储器可以是单片机片内的非易失性存储器，也可以是单片机片外的非易失性存储器。STC15W4K32S4 系列单片机片内集成了容量不等的 E2PROM（带仿真功能的 IAP 命名的没有片内 E2PROM），其与程序空间是分开的，擦写次数可达 10 万以上，可用于用户存放一些掉电不丢失的重要数据。

STC15W4K32S4 系列单片机片内 E2PROM 是以扇区为单位进行操作的，每个扇区包含 512 字节。使用片内 E2PROM 时，同一次修改的数据建议放在同一个扇区，需要不同时修改的数据不要放在同一扇区，每个扇区不是一定要用满的。

表 2：STC15W4K32S4 系列单片机片内 E2PROM 空间大小及地址

序号	单片机型号	E2PROM	扇区数	E2PROM 起始扇区首地址	E2PROM 结束扇区末尾地址	备注
1	STC15W4K16S4	42K 字节	84	0000h	A7FFh	这里的 E2PROM 起始扇区首地址和结束扇区末尾地址都是针对用 IAP 字节读时。（使用 MOVC 指令读取时地址不同）
2	STC15W4K32S4	26K 字节	52	0000h	67FFh	
3	STC15W4K40S4	18K 字节	36	0000h	47FFh	
4	STC15W4K48S4	10K 字节	20	0000h	27FFh	
5	STC15W4K56S4	2K 字节	4	0000h	07FFh	
6	IAP15W4K58S4	0K 字节	116	0000h	E7FFh	无片内 E2PROM，用户可将 FLASH 当做 E2PROM 使用。
7	IAP15W4K61S4	0K 字节	112	0000h	F3FFh	
8	IRC15W4K63S4	0K 字节	127	0000h	FDFFh	

✧ 注：用户在操作用户程序区的程序 FLASH 时，切勿将有效程序擦除。

STC15W4K32S4 系列单片机片内 E2PROM 低压时不宜操作，所以在使用 STC-ISP 软件给单片机下载程序时，建议选择“低压时禁止 EEPROM 操作”选项，如下图所示。

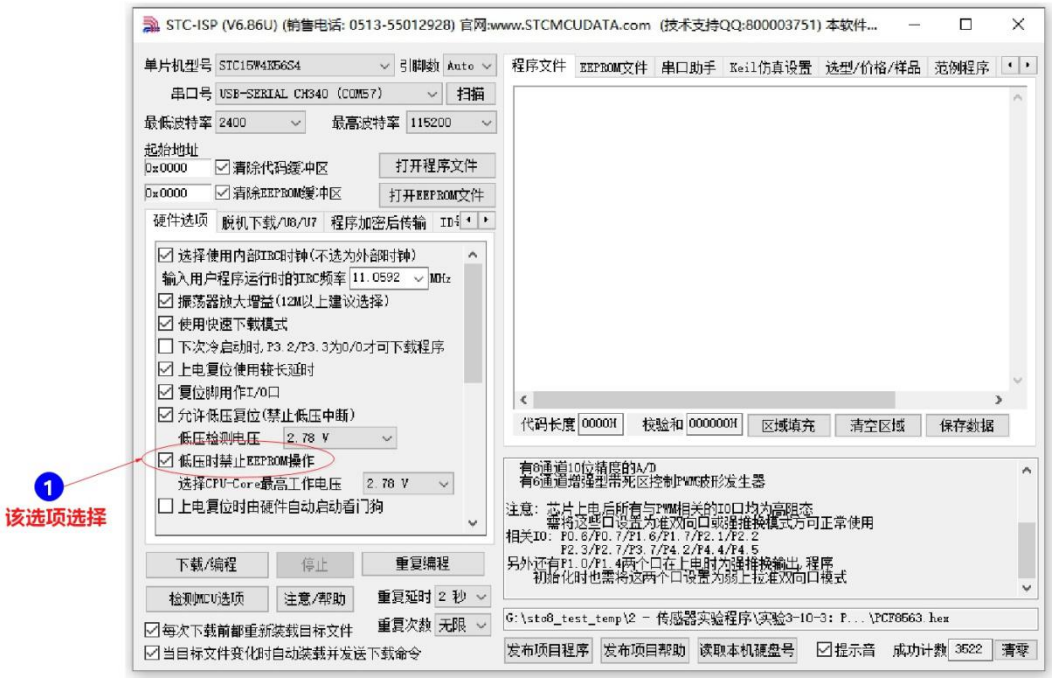

图 4：STC-ISP 下载界面选项注意事项

4. 软件设计

4.1. 片内存储器寄存器汇集

STC15W4K32S4 系列单片机进行片内存储器操作时会用到 8 个寄存器，如下表所示：

表 3：STC15W4K32S4 系列片内存储器操作使用寄存器汇总

序号	寄存器名	读/写	功能描述
1	AUXR	读/写	辅助寄存器。
2	PCON	读/写	电源控制寄存器。
3	IAP_DATA	读/写	ISP/IAP 数据寄存器。
4	IAP_CMD	读/写	ISP/IAP 命令寄存器。
5	IAP_TRIG	读/写	ISP/IAP 命令触发寄存器。
6	IAP_CONTR	读/写	ISP/IAP 控制寄存器。
7	IAP_ADDRH	读/写	ISP/IAP 地址寄存器。
8	IAP_ADDRL	读/写	

✧ 注：在操作 STC15W4K32S4 系列单片机片内 E2PROM 或 FLASH 时，一定要谨慎定义操作地址（即 ISP/IAP 地址寄存器取值），以免对不该操作或者无效的区域进行操作。

4.2. 寄存器解析

4.2.1. 电源控制寄存器 PCON

电源控制寄存器 PCON 的 LVDF 位为单片机低压检测标志位，该位标志单片机工作电压 VCC 是否低于低压检测门槛电压。当 VCC 低于低压检测门槛电压时，LVDF 位被置 1，一旦该位置 1 需软件清零。

电源控制寄存器

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
PCON	87H	name	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL

① 寄存器名

② 低压检测标志位

LVDF: 低压检测标志位，当工作电压Vcc低于低压检测门槛电压时，该位置1。

图 5：电源控制寄存器 PCON

4.2.2. ISP/IAP 命令寄存器 IAP_CMD

ISP/IAP 命令寄存器 IAP_CMD 的 MS0 位和 MS1 位组合用来选择对用户的应用程序区操作的模式。

ISP/IAP命令寄存器

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IAP_CMD	C5H	name	-	-	-	-	-	-	MS1	MS0

① 寄存器名

② 模式选择位

MS1	MS0	命令 / 操作 模式选择
0	0	Standby 待机模式，无ISP操作
0	1	从用户的应用程序区对"Data Flash/EEPROM区"进行字节读
1	0	从用户的应用程序区对"Data Flash/EEPROM区"进行字节编程
1	1	从用户的应用程序区对"Data Flash/EEPROM区"进行扇区擦除

图 6：ISP/IAP 命令寄存器 IAP_CMD

✧ 注:STC15W4KxxS4 单片机程序在用户应用程序区,仅可以对数据FLASH区(E2PROM)进行字节读、字节编程和擦除扇区等操作。IAP15W4KxxS4 单片机可在用户应用程序区修改用户应用程序区。

4.2.3. ISP/IAP 命令触发寄存器 IAP_TRIG

ISP/IAP 命令触发寄存器 IAP_TRIG： 用于触发对用户的应用程序区的读、写、擦除操作。每次进行读、写或擦除操作时，需先使能 IAPEN 位(即将 IAPEN 位置 1)，再对 IAP_TRIG 寄存器先写 0x5A，再写 0xA5，这样操作命令才会生效。

ISP/IAP命令触发寄存器

符号	地址	B7	B6	B5	B4	B3	B2	B1	B0
IAP_TRIG	C6H								

① 寄存器名

② 触发数据写入

图 7：ISP/IAP 命令触发寄存器

4.2.4. ISP/IAP 控制寄存器 IAP_CONTR

ISP/IAP 控制寄存器 IAP_CONTR 主要是配置对用户的应用程序区的读、写、擦除操作时的功能选择，IAPEN 是操作总开关，SWBS 位和 SWRST 位配合使用，用于控制软件复位及开始执行程序区，CMD_FAIL 位用于指示操作是否成功，不成功则会自动置 1（需软件清零）。

ISP/IAP控制寄存器（不可位寻址）

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IAP_CONTR	C7H	name	IAPEN	SWBS	SWRST	CMD_FAIL	-	WT2	WT1	WT0

1 寄存器名

2 使能控制位
IAPEN: ISP/IAP功能允许位。0：禁止IAP读/写/擦除Data Flash/EEPROM
1：允许IAP读/写/擦除Data Flash/EEPROM

3 软件复位选择控制位
SWBS: 软件选择复位后从用户应用程序区启动(送0)，还是从系统ISP监控程序区启动(送1)。要与SWRST直接配合才可以实现

4 软件复位控制位
SWRST: 0：不操作； 1：软件控制产生复位，单片机自动复位。

5 操作失败标志位
CMD_FAIL: 如果IAP地址(由IAP地址寄存器IAP_ADDRH和IAP_ADDRL的值决定)指向了非法地址或无效地址，且送了ISP/IAP命令，并对IAP_TRIG送5Ah/A5h触发失败，则CMD_FAIL为1，需由软件清零。

6 组合设置等待时间

设置等待时间			CPU等待时间(多少个CPU工作时钟)			
WT2	WT1	WT0	Read/读(2个时钟)	Program/编程(=55us)	Sector Erase 扇区擦除(=21ms)	Recommended System Clock 跟等待参数对应的推荐系统时钟
1	1	1	2个时钟	55个时钟	21012个时钟	≥ 1MHz
1	1	0	2个时钟	110个时钟	42024个时钟	≥ 2MHz
1	0	1	2个时钟	165个时钟	63036个时钟	≥ 3MHz
1	0	0	2个时钟	330个时钟	126072个时钟	≥ 6MHz
0	1	1	2个时钟	660个时钟	252144个时钟	≥ 12MHz
0	1	0	2个时钟	1100个时钟	420240个时钟	≥ 20MHz
0	0	1	2个时钟	1320个时钟	504288个时钟	≥ 24MHz
0	0	0	2个时钟	1760个时钟	672384个时钟	≥ 30MHz

图 8：ISP/IAP 控制寄存器

✧ 注：STC15W4K32S4 系列单片机针对不同系统时钟推荐了等待时间设置值，用户须知该等待时间是硬件自动完成的，不需要用户在软件里面加额外的软件延时。另外，如果用户使用的是 MOVC 指令进行读取操作，则 CPU 不需要等待时间的。

4.3. 片内 EEPROM 读写 - 单个字节（STC15W4K56S4）

✧ 注：本节的实验源码是在“实验 2-8-1：串口 1 收发实验（P3.0 和 P3.1）”的基础上修改。本节对应的实验源码是：“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”。

4.3.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 4：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	uart	.c	包含与用户 uart 有关的用户自定义函数。
2	eeprom	.c	E2PROM 有关的用户自定义函数。

3	delay	.c	包含用户自定义延时函数。

4.3.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "uart.h"
3. #include "eeprom.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 5：头文件包含路径

序号	路径	描述
1	..\ Source	uart.h、eeprom.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

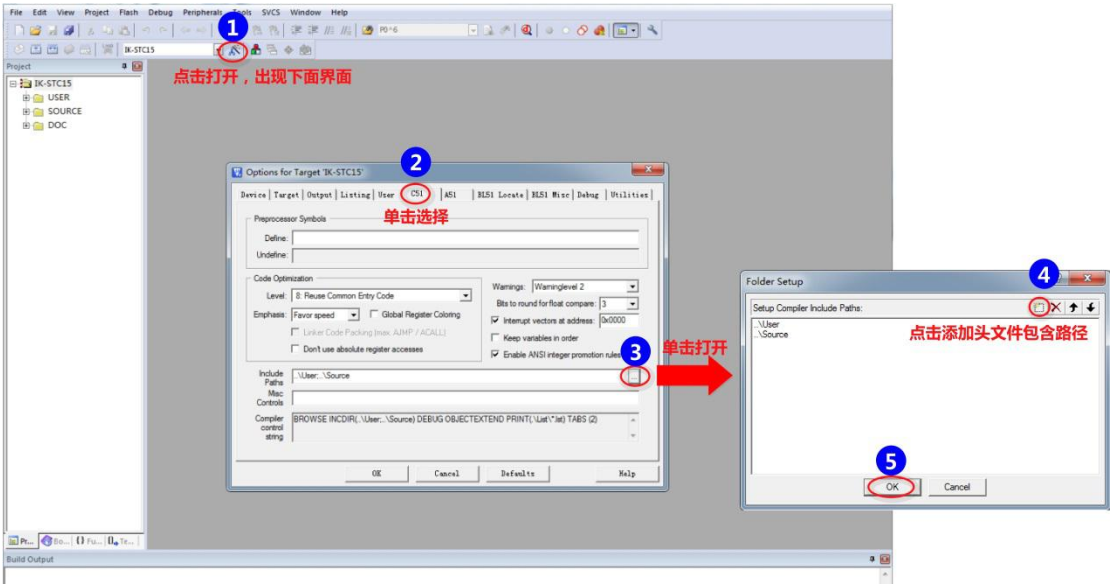

图 9：添加头文件包含路径

4.3.3. 编写代码

首先，在 eeprom.c 文件中编写操作 E2PROM 外设会用到的函数，如下表所示。

表 6：E2PROM 相关用户函数汇集

序号	函数名	功能描述
1	Disable_EEPROM	禁止对用户的应用程序区的读、写、擦除操作。

2	EEPROM_read_n	读取指定地址的 n 个字节数据。
3	EEPROM_SectorErase	擦除指定地址扇区。
4	EEPROM_write_n	写入 n 个字节数据到指定地址。

关于每个 E2PROM 相关用户函数，下面详细给出代码。

程序清单：禁止访问 ISP/IAP 函数

```
/*************************************************************************
 * 描  述 : 禁止访问 ISP/IAP
 * 入  参 : 无
 * 返回值 : 无
 *************************************************************************/
void Disable_EEPROM(void)
{
    IAP_CONTR = 0x00;      //对 ISP/IAP 控制寄存器赋值，IAPEN 位为 0，禁止 ISP/IAP 操作
    IAP_CMD   = 0x00;          //清零 ISP/IAP 命令寄存器各位，去除 ISP/IAP 命令
    IAP_TRIG  = 0x00;        //清零 ISP/IAP 命令触发寄存器各位，防止 ISP/IAP 命令误触发
    IAP_ADDRH = 0x80;          //ISP/IAP 地址寄存器高 8 位赋值
    IAP_ADDRL = 0x00;      //ISP/IAP 地址寄存器低 8 位赋值，指向非 EEPROM 区，防止误操作
}
```

程序清单：读取指定地址的 n 个字节数据函数

```
/*************************************************************************
 * 描  述 : 从指定 EEPROM 首地址读出 n 个字节放指定的缓冲
 * 入  参 : EE_address:  读出 EEPROM 的首地址
 *           DataAddress: 读出数据放缓冲的首地址
 *           number:      读出的字节长度
 * 返回值 : 无
 *************************************************************************/
void EEPROM_read_n(uint16 EE_address,uint8 *DataAddress,uint16 number)
{
    EA = 0;                         //禁止总中断
    IAP_CONTR |= 0x80;          //IAPEN 位置 0，允许 ISP/IAP 操作
    IAP_CONTR |= 0x04;          //WT2 位置 1，用于设置 CPU 等待时间
    IAP_CONTR &= 0xFC;          //WT0 和 WT1 位置 0，用于设置 CPU 等待时间
    IAP_CMD = 0x01;             //MS0 位置 1，MS1 位置 0，对 ISP/IAP 进行字节读操作
    do
    {
        IAP_ADDRH = EE_address / 256;       //ISP/IAP 地址寄存器高 8 位赋值
        IAP_ADDRL = EE_address % 256;       //ISP/IAP 地址寄存器低 8 位赋值
        IAP_TRIG = 0x5A;                    //ISP/IAP 命令触发寄赋值为 5AH
        IAP_TRIG = 0xA5;                    //ISP/IAP 命令触发寄赋值为 A5H
        _nop_();                                //空命令
```

```
22.         *DataAddress = IAP_DATA;            //读出 ISP/IAP 数据寄存器的值送往指定缓存
23.         EE_address++;
24.         DataAddress++;
25.     }while(--number);
26.     Disable_EEPROM();               //禁止访问 ISP/IAP
27.     EA = 1;                           //开启总中断
28. }
```

程序清单：擦除指定地址扇区函数

```
1.  /*************************************************************************
2.   * 描   述 : 把指定地址的 EEPROM 扇区擦除
3.   * 入   参 : EE_address:  读出 EEPROM 的首地址
4.              DataAddress: 读出数据放缓冲的首地址.
5.              number:      读出的字节长度.
6.   * 返回值 : 无
7.   *************************************************************************/
8.  void EEPROM_SectorErase(uint16 EE_address)
9.  {
10.     EA = 0;                             //禁止总中断
11.     IAP_ADDRH = EE_address / 256;       //ISP/IAP 地址寄存器高 8 位赋值
12.     IAP_ADDRL = EE_address % 256;       //ISP/IAP 地址寄存器低 8 位赋值
13.     IAP_CONTR |= 0x80;                 //IAPEN 位置 0，允许 ISP/IAP 操作
14.     IAP_CONTR |= 0x04;                 //WT2 位置 1，用于设置 CPU 等待时间
15.     IAP_CONTR &= 0xFC;                 //WT0 和 WT1 位置 0，用于设置 CPU 等待时间
16.     IAP_CMD = 0x03;                   //MS0 位置 1，MS1 位置 1，对 ISP/IAP 进行扇区擦除操作
17.     IAP_TRIG = 0x5A;                   //ISP/IAP 命令触发寄赋值为 5AH
18.     IAP_TRIG = 0xA5;                   //ISP/IAP 命令触发寄赋值为 A5H
19.     _nop_();                           //空命令
20.     Disable_EEPROM();                  //禁止访问 ISP/IAP
21.     EA = 1;                            //重新允许中断
22. }
```

程序清单：写入 n 个字节数据到指定地址函数

```
1.  /*************************************************************************
2.   * 描   述 : 向指定 EEPROM 首地址写入 n 个字节数据
3.   * 入   参 : EE_address:  写入 EEPROM 的首地址
4.              DataAddress: 写入源数据的缓冲的首地址
5.              number:      写入的字节长度
6.   * 返回值 : 无
7.   *************************************************************************/
8.  void EEPROM_write_n(uint16 EE_address,uint8 *DataAddress,uint16 number)
9.  {
10.     EA = 0;                           //禁止总中断
```

```
    IAP_CONTR |= 0x80;          //IAPEN 位置 0，允许 ISP/IAP 操作
    IAP_CONTR |= 0x04;          //WT2 位置 1，用于设置 CPU 等待时间
    IAP_CONTR &= 0xFC;          //WT0 和 WT1 位置 0，用于设置 CPU 等待时间
    IAP_CMD = 0x02;             //MS0 位置 0，MS1 位置 1，对 ISP/IAP 进行字节写操作
    do
    {
        IAP_ADDRH = EE_address / 256;       //ISP/IAP 地址寄存器高 8 位赋值
        IAP_ADDRL = EE_address % 256;       //ISP/IAP 地址寄存器低 8 位赋值
        IAP_DATA = *DataAddress;            //将指定缓存数据送往 ISP/IAP 数据寄存器
        IAP_TRIG = 0x5A;                    //ISP/IAP 命令触发寄赋值为 5AH
        IAP_TRIG = 0xA5;                    //ISP/IAP 命令触发寄赋值为 A5H
        _nop_();                            //空命令
        EE_address++;
        DataAddress++;
    }while(--number);
    Disable_EEPROM();               //禁止访问 ISP/IAP
    EA = 1;                           //开启总中断
}
```

✧ 注：以上关于 E2PROM 的用户自定义函数有对 EA 的操作，即操作 E2PROM 相关寄存器之前关掉总中断 EA，之后再打开总中断 EA。这在官方手册提供的例程中并没有，这个仅供用户参考。

然后，在主函数中对串口 1 进行初始化，通过串口 1 发送不同的命令实现对单片机片内 E2PROM 的单字节读、写及扇区擦除等操作。（注意读写的首地址）

代码清单：主函数

```
int main()
{
//////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////
    P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口

    Uart1_Init();                                   //串口 1 初始化
    EA = 1;                                         //使能总中断
    delay_ms(10);                                   //初始化后延时

  while (1)
    {
     if(WriteFLAG)                      //写模式
     {
```

```
        WriteFLAG=0;                        //写标志变量清零,发送一次
        EEPROM_write_n(0x0000,scan,1);     //在 EEPROM 的首地址为 0x0000 处写入 1 个字节
        SendDataByUart1(0x33);              //串口 1 发送数据 0x33
         }
         if(ReadFLAG)                       //读模式
         {
              ReadFLAG=0;                           //读标志变量清零,发送一次
              EEPROM_read_n(0x0000,buffer,1);       //在 EEPROM 的首地址为 0x0000 处读
取 1 个字节存入 buffer 数组中
        SendStringByUart1_n(buffer,1);        //串口 1 发送 buffer 中存的数据
         }
         if(ClearFLAG)                      //扇区擦除模式
         {
              ClearFLAG=0;                          //清除标志变量清零,发送一次
              EEPROM_SectorErase(0x0000);       //对 EEPROM 的首地址为 0x0000 处的扇区
进行扇区擦除
        SendDataByUart1(0x00);              //串口 1 发送数据 0x00
         }
    }
}
```

4.3.4. 硬件连接

略。（开发板单片机型号选择 STC15W4K56S4）

4.3.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\EEPROM\project”目录下的工程“EEPROM.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“EEPROM.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 FF。

   2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 33。

   3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，

会在接收端显示 FF。

4）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板断电再通电后，在串口调试助手发送端发"3"，会在接收端显示 33。说明掉电数据不丢失。

5）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板重新下载该程序后，在串口调试助手发送端发"3"，会在接收端显示 33。说明重新下载程序数据不丢失。

4.4. 片内 EEPROM 读写 - 多个字节（STC15W4K56S4）

✧ 注：本节的实验源码是在“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”的基础上修改。本节对应的实验源码是：“实验 2-11-2：片内 EEPROM 读写 - 多个字节（STC15W4K56S4）”。

4.4.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”部分。

4.4.2. 编写代码

在 eeprom.c 文件中编写操作 E2PROM 外设会用到的函数，请参考“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”部分。

然后，在主函数中对串口 1 进行初始化，通过串口 1 发送不同的命令实现对单片机片内 E2PROM 的多字节读、写及扇区擦除等操作。（注意读写的首地址）

代码清单：主函数

```
1.  /*************************************************************************
2.   * 描  述 : 主函数
3.   * 入  参 : 无
4.   * 返回值 : 无
5.   *************************************************************************/
6.  int main()
7.  {
8.  ///////////////////////////////////////////////
9.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
10. //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
11. //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
12. //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
13. ///////////////////////////////////////////////
14.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                    //设置P3.0~P3.1为准双向口
15.
16.     Uart1_Init();                                    //串口 1 初始化
17.     EA = 1;                                          //使能总中断
18.   delay_ms(10);                                     //初始化后延时
19.
20.   while (1)
```

```
    {
     if(WriteFLAG)                    //写模式
         {
              WriteFLAG=0;                        //写标志变量清零,发送一次
              EEPROM_write_n(0x0000,scan,5);          //在 EEPROM 的首地址为 0x0000 处写入 5 个字节
         SendDataByUart1(0x33);              //串口 1 发送数据 0x33
          }
          if(ReadFLAG)                    //读模式
          {
              ReadFLAG=0;                         //读标志变量清零,发送一次
              EEPROM_read_n(0x0000,buffer,5);          //在 EEPROM 的首地址为 0x0000 处读取 5 个字节存入 buffer 数组中
         SendStringByUart1_n(buffer,5);       //串口 1 发送 buffer 中存的数据
          }
          if(ClearFLAG)                   //扇区擦除模式
          {
              ClearFLAG=0;                        //清除标志变量清零,发送一次
              EEPROM_SectorErase(0x0000);             //对 EEPROM 的首地址为 0x0000 处的扇区进行扇区擦除
         SendDataByUart1(0x00);               //串口 1 发送数据 0x00
          }
    }
}
```

4.4.3. 硬件连接

略。（开发板单片机型号选择 STC15W4K56S4）。

4.4.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-11-2：片内 EEPROM 读写 - 多个字节（STC15W4K56S4）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\EEPROM\project”目录下的工程“EEPROM.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“EEPROM.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 FF。

2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 11 22 33 44 55。

3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，会在接收端显示 FF FF FF FF FF。

4）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板断电再通电后，在串口调试助手发送端发"3"，会在接收端显示 11 22 33 44 55。说明掉电数据不丢失。

5）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板重新下载该程序后，在串口调试助手发送端发"3"，会在接收端显示 11 22 33 44 55。说明重新下载程序数据不丢失。

4.5.片内 FLASH 读写 - 单个字节（IAP15W4K58S4）

✧ 注：本节的实验源码是在“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”的基础上修改。本节对应的实验源码是:“实验 2-11-3:片内 FLASH 读写 - 单个字节（IAP15W4K58S4）”。

4.5.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”部分。

4.5.2. 编写代码

在 eeprom.c 文件中编写操作 E2PROM 外设会用到的函数，请参考“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”部分。

然后，在主函数中对串口 1 进行初始化，通过串口 1 发送不同的命令实现对单片机片内 FLASH 的单字节读、写及扇区擦除等操作。（注意读写的首地址）

代码清单：主函数

```
1.  int main()
2.  {
3.  ///////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ///////////////////////////////////////////////
9.      P3M1 &= 0xFC;   P3M0 &= 0xFC;                    //设置 P3.0~P3.1 为准双向口
10.
11.     Uart1_Init();                                    //串口 1 初始化
12.     EA = 1;                                          //使能总中断
13.   delay_ms(10);                                     //初始化后延时
14.
15.   while (1)
```

```
    {
     if(WriteFLAG)                          //写模式
         {
              WriteFLAG=0;                              //写标志变量清零,发送一次
              EEPROM_write_n(0xE000,scan,1);          //在 FLASH 的首地址为 0xE000 处写入 1 个字节
         SendDataByUart1(0x33);              //串口 1 发送数据 0x33
          }
          if(ReadFLAG)                          //读模式
          {
              ReadFLAG=0;                               //读标志变量清零,发送一次
              EEPROM_read_n(0xE000,buffer,1);          //在 FLASH 的首地址为 0xE000 处读取 1 个字节存入 buffer 数组中
         SendStringByUart1_n(buffer,1);       //串口 1 发送 buffer 中存的数据
          }
          if(ClearFLAG)                         //扇区擦除模式
          {
              ClearFLAG=0;                              //清除标志变量清零,发送一次
              EEPROM_SectorErase(0xE000);             //对 FLASH 的首地址为 0xE000 处的扇区进行扇区擦除
         SendDataByUart1(0x00);                //串口 1 发送数据 0x00
          }
    }
}
```

4.5.3. 硬件连接

略。（开发板单片机型号选择 IAP15W4K58S4）。

4.5.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-11-3：片内 FLASH 读写 - 单个字节（IAP15W4K58S4）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\FLASH\project”目录下的工程“FLASH.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“FLASH.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 FF。

2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 33。

3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，会在接收端显示 FF。

4）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板断电再通电后，在串口调试助手发送端发"3"，会在接收端显示 33。说明掉电数据不丢失。

5）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板重新下载该程序后，在串口调试助手发送端发"3"，会在接收端显示 FF。说明重新下载程序数据丢失。

4.6.片内 EEPROM 读写 - 多个字节（IAP15W4K58S4）

✧ 注：本节的实验源码是在“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”的基础上修改。本节对应的实验源码是:“实验 2-11-4:片内 FLASH 读写 - 多个字节（IAP15W4K58S4）”。

4.6.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”部分。

4.6.2. 编写代码

在 eeprom.c 文件中编写操作 E2PROM 外设会用到的函数，请参考“实验 2-11-1：片内 EEPROM 读写 - 单个字节（STC15W4K56S4）”部分。

然后，在主函数中对串口 1 进行初始化，通过串口 1 发送不同的命令实现对单片机片内 FLASH 的多字节读、写及扇区擦除等操作。（注意读写的首地址）

代码清单：主函数

```
1.  int main()
2.  {
3.  ////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ////////////////////////////////////////////////
9.      P3M1 &= 0xFC;   P3M0 &= 0xFC;                    //设置 P3.0~P3.1 为准双向口
10.
11.     Uart1_Init();                                    //串口 1 初始化
12.     EA = 1;                                          //使能总中断
13.   delay_ms(10);                                     //初始化后延时
14.
15.   while (1)
16.     {
17.      if(WriteFLAG)                    //写模式
```

```
        {
            WriteFLAG=0;                        //写标志变量清零,发送一次
            EEPROM_write_n(0xE000,scan,5);      //在 FLASH 的首地址为 0xE000 处写入 5 个字节
        SendDataByUart1(0x33);              //串口 1 发送数据 0x33
        }
        if(ReadFLAG)                        //读模式
        {
            ReadFLAG=0;                         //读标志变量清零,发送一次
            EEPROM_read_n(0xE000,buffer,5);     //在 FLASH 的首地址为 0xE000 处读取 5 个字节存入 buffer 数组中
        SendStringByUart1_n(buffer,5);      //串口 1 发送 buffer 中存的数据
        }
        if(ClearFLAG)                       //扇区擦除模式
        {
            ClearFLAG=0;                        //清除标志变量清零,发送一次
            EEPROM_SectorErase(0xE000);         //对 FLASH 的首地址为 0xE000 处的扇区进行扇区擦除
        SendDataByUart1(0x00);              //串口 1 发送数据 0x00
        }
    }
}
```

4.6.3. 硬件连接

略。（开发板单片机型号选择 IAP15W4K58S4）。

4.6.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-11-4：片内 FLASH 读写 - 多个字节（IAP15W4K58S4）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\FLASH\project”目录下的工程“FLASH.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“FLASH.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 FF。

   2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 11 22 33 44 55。

3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，会在接收端显示 FF FF FF FF FF。

4）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板断电再通电后，在串口调试助手发送端发"3"，会在接收端显示 11 22 33 44 55。说明掉电数据不丢失。

5）在串口调试助手发送端发"2"，会在接收端显示 33，给开发板重新下载该程序后，在串口调试助手发送端发"3"，会在接收端显示 FF FF FF FF FF。说明重新下载程序数据丢失。