基于 SDK 的 C51 工程开发教程

简介: 51 单片机以及程序设计的需求较大,除了一些 51 单片机的小项目外,一些需要联网、需要涉及到较多的外围器件以及需要较为复杂的任务逻辑时,传统的裸机 51 程序编写变得也比较烦心,模块代码的不确定性、逻辑的复杂等等,造成项目编写思维的困难以及后期调试的复杂等等。

实现成熟模块代码的可重用、提供各种软件服务以及程序的模块化设计等等, 就非常的重要了。

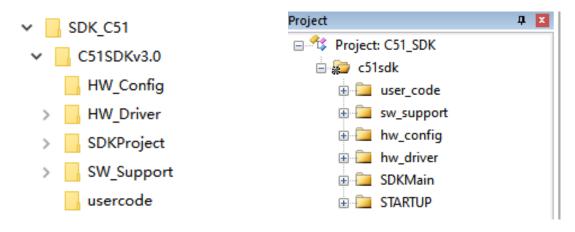
SDK 介绍: SDK 主要争对 STC89C51\52、AT89C51\52 系列的单片机,就是为了实现成熟模块代码的可重用、提供各种软件服务以及程序的模块化设计等等。

SDK 提供 max1241、ad9850、ad0832、1cd1602、1cd12864、esp8266 等芯片或模块的成熟代码,实现对芯片或模块相关的具体操作逻辑的屏蔽,向上层提供统一的接口,例如调用初始化函数就能初始化模块,调用读函数就能读取模块的数据等等。

SDK 还对芯片底层进行一定的屏蔽。将硬件定时器、串口、外部中断、管脚配置等进行函数 封装,实现对芯片底层进行屏蔽,方便程序的开发与调试等。

SDK 同时也提供软件服务。例如通用软件延时、软件定时器、软件捕获器等等。

SDK 文件结构与 KEIL 工程结构如下图:



C51SDKv3.0: SDK 包所在文件

HW Config: 与51、52系列单片机硬件相关的代码,例如定时器、串口等等。

HW Driver: 单片机系统外围芯片和模块的驱动文件,例如 ad0832 等等的驱动文件。

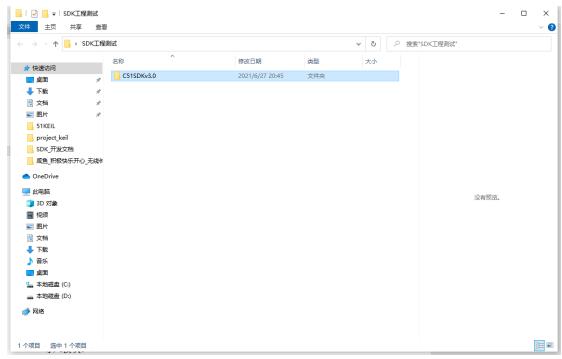
SDKProject: SDK 工程文件。

SW Support: SDK 提供的软件服务,包含软件定时器等等。

Usercode: 用户代码文件。

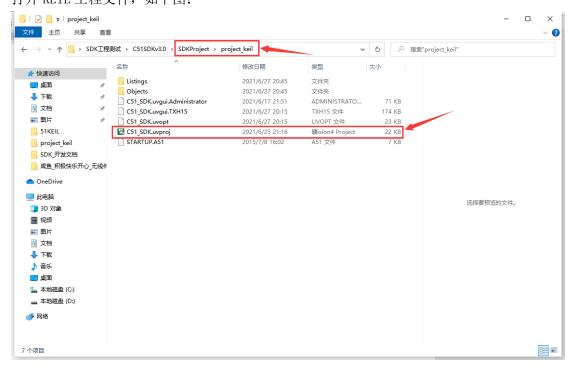
新建工程:

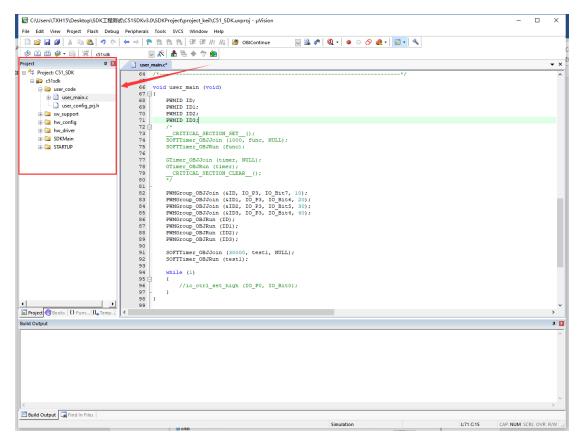
复制 C51SDKv3.0 这个 SDK 包到自己工程中



名字可自己修改。

进入刚刚拷贝好的文件夹,打开 SDKProject\project_keil 文件夹;打开 KEIL 工程文件,如下图:





这表示已经进入工程

创建用户文件:

SDK 包中的 usercode 是用户代码的存放文件夹。

usercode 文件夹下有三个已经建好的文件(user_main.c、user_main.h、user_config_prj.h) 这三个文件不要删除和修改名称。

user_main.c: 是用户编写代码的文件,中间有一个user_main()函数,这个是用户主函数,相当与裸机中的main函数,所有用户代码主逻辑都应在这儿开始,且应有一个无限循环体。user_config_prj.h: 工程配置文件,不可删除和修改名字。

这个文件夹下除了已有的三个文件,用户还可以在这个文件夹下新建文件和文件夹,包含自己所需的代码模块等等。但要在 user_main.c 或 user_main.h 中去包含到这些模块。

其对应的 KEIL 工程如下图:

```
駅 C:\Users\TXH15\Desktop\SDK工程測試\C51SDKv3.0\SDKProject\project_keil\C51_SDK.uvproj - μVision
                                                                                                                                                                                                                                                     File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
  user_main.c*

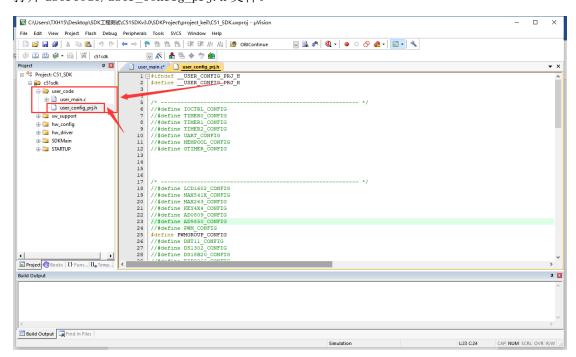
☐ <sup>4</sup> Project: C51_SDK
   S Project C51_SUK
Size_C15ak
Size_C15ak
Size_Code
Size_Code
Size_Code
Size_Code
Size_Code
Size_Code
Size_Code
                                                                       user main (void)
                                                                    FWHID ID;
FWHID ID1;
FWHID ID2;
FWHID ID3;

'CRITICAL_SECTION_SET__();
SOFTTIMET_OBJJOin (1000, func, NULL);
SOFTTIMET_OBJJOin (func);
                                                       hw_config
hw_driver
SDKMain
STARTUP
                                                                    GTimer_OBJJoin (timer, NULL);
GTimer_OBJRun (timer);
__CRITICAL_SECTION_CLEAR__();
*/
                                                                    FWMSroup_OBJJoin (&ID, IO_P3, IO_Bit7, 10);
FWMSroup_OBJJoin (&IDI, IO_P3, IO_Bit6, 20);
FWMSroup_OBJJoin (&IDI, IO_P3, IO_Bit6, 20);
FWMSroup_OBJJoin (&IDI, IO_P3, IO_Bit5, 30);
FWMSroup_OBJJOIn (&IDI, IO_P3, IO_Bit4, 40);
FWMSroup_OBJRAN (IDI);
FWMSroup_OBJRAN (IDI);
FWMSroup_OBJRAN (IDI);
FWMSroup_OBJRAN (IDI);
                                                                     SOFTTimer_OBJRun (30000, test1, NULL);
                                                                          //io_ctrl_set_high (IO_P0, IO_Bit0);
Build Output
                                                                                                                                                                                                                                                              Ù 🔣
```

如需添加更多的模块代码,可将文件添加至 usercode 目录下,并在 KEIL 工程中 usercode 导入新加的代码。

导入模块:

打开 usercode/user_config_prj.h 文件。



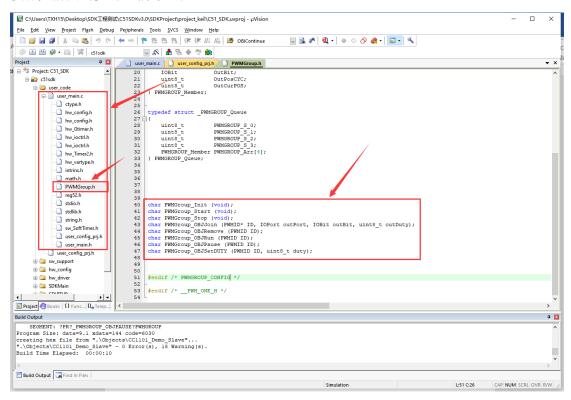
文件中定义了很多的 CONFIG 宏。

要使用某个已有模块,只需要取消相应宏的备注就行。

此工程就打开了 PWMGroup 组件,就可以使用 PWM 组的模块,现多组 PWM 的输出。

开始编写代码:

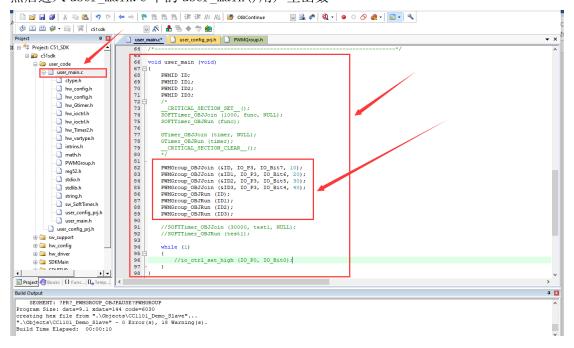
执行完前面的步骤,编译工程。



就会出现如上图所示的头文件

打开选中模块头文件,文件中有模块的相关接口函数。可以查找到自己所需的 API。

然后进入 user_main. c 下的 user_main()用户主函数



在 user_main. c 文件下编写任务代码,实现相应的功能。 图中代码就实现了一个 4 路自定义端口输出占空比分别为 10%、20%、30%、40%的 PWM 波。