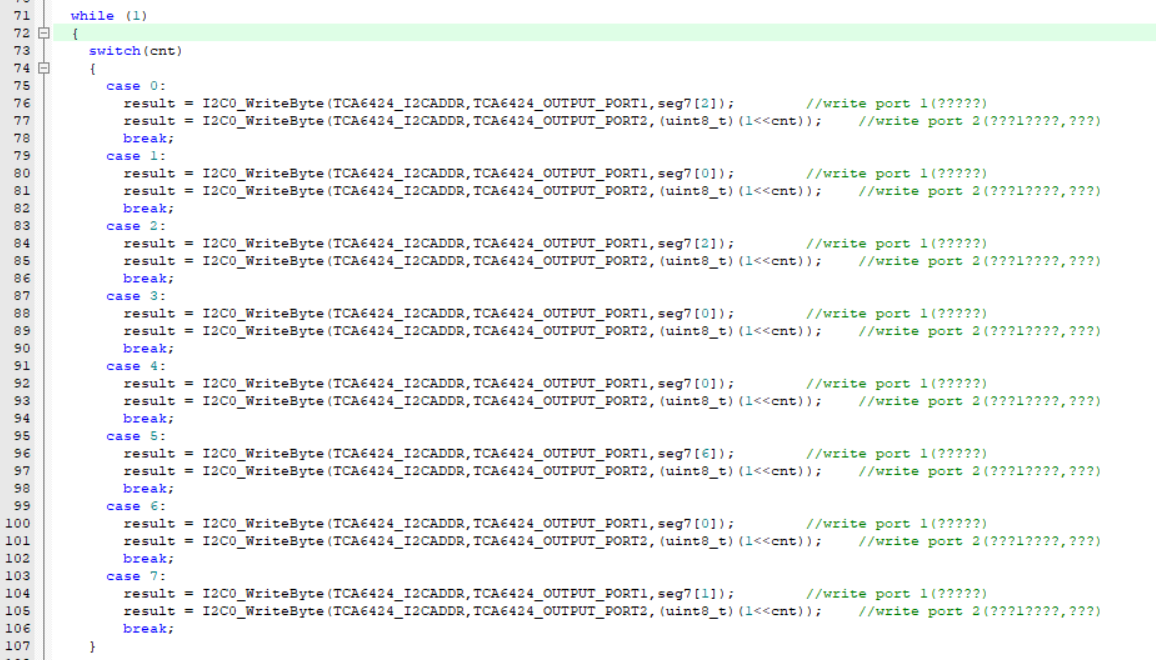
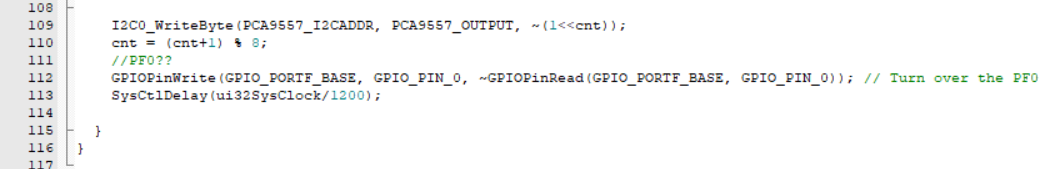
**ARM实验二 I2C GPIO扩展及SYSTICK中断实验**

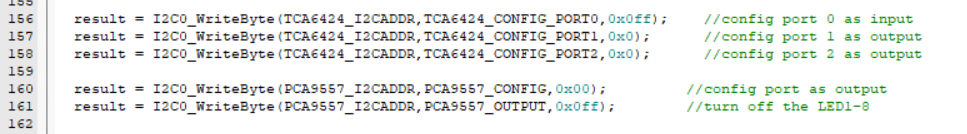
1. **实验主要程序段展示与分析**

以下展示各分项实验的关键代码，具体代码详见附加的c文件压缩包。

实验2.1：

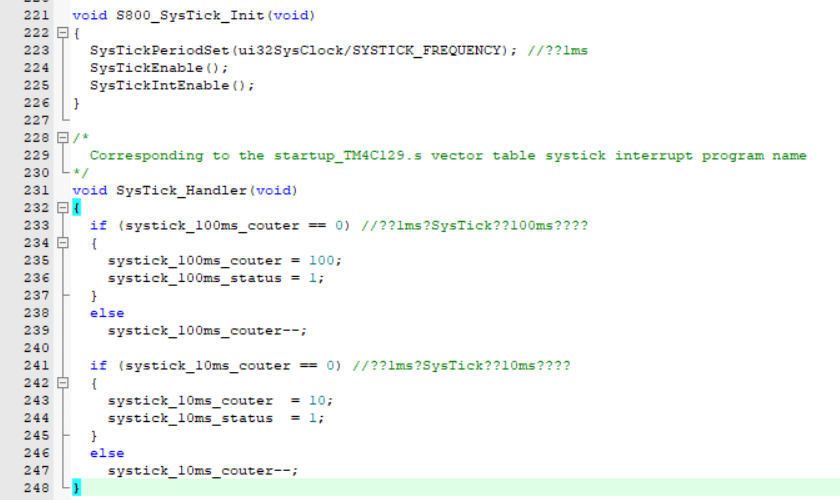






由于数码管一次只能亮一位，因此要同时显示八个字符只能利用人眼的视觉暂留效应。容易想到将全局变量cnt分为0~7共八种情况，分别对数码管八个位进行使能与输入字模。110行是同余八的操作。利用113行延时使每个数码管亮2.5ms，这样视觉上每一位都不会闪烁，亮度也不是很低。再注意一下156行开始的端口配置是否正确即可。

实验2.2：

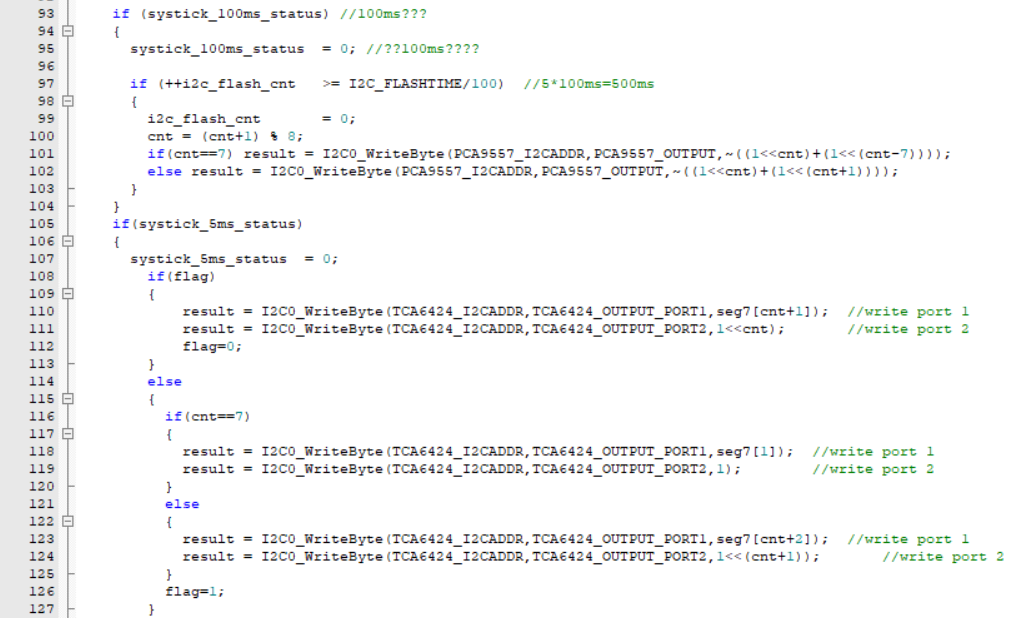


由于跑马灯需要一个频率，那不可避免会用到systick定时中断功能。如图设置好初始化和中断服务程序，引出100ms和10ms两个定时。10ms作为led\_M0的闪烁半周期，100ms用来设置跑马灯频率。

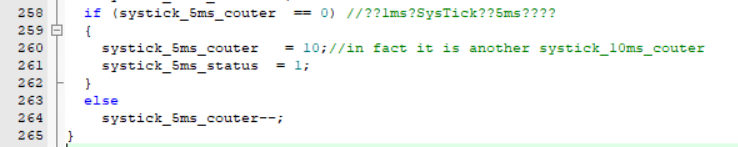


在主函数while循环内，有着和实验一中systick定时相同的操作，实现了每10ms和100ms进入一次对应的if语句。由于只需要实现两位跑马，故在112行中将同余8改为同余2。另外，由于显示1的时候led需要亮第一个和最后一个，有必要做分类讨论，这就是104行到111行的内容。

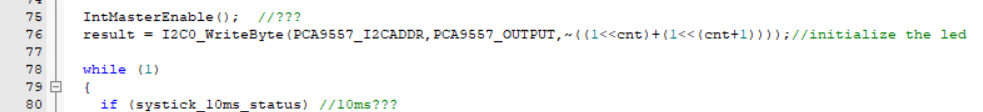
实验2.3：



实验三中，针对数码管一次只能亮一位的特性，考虑将数码管显示代码单独形成模块。93行到104行通过设定500ms定时，使得用作指示的全局变量cnt和led灯每隔0.5s变化一次。105行开始是数码管显示模块，设定10ms定时，使得两数码管轮流亮10ms，这样视觉上两数码管同时发光。其中，cnt负责决定输入哪个字模。由于跑马灯需要循环，因此需要对特殊情况分类讨论。这和实验2.2的思路类似。

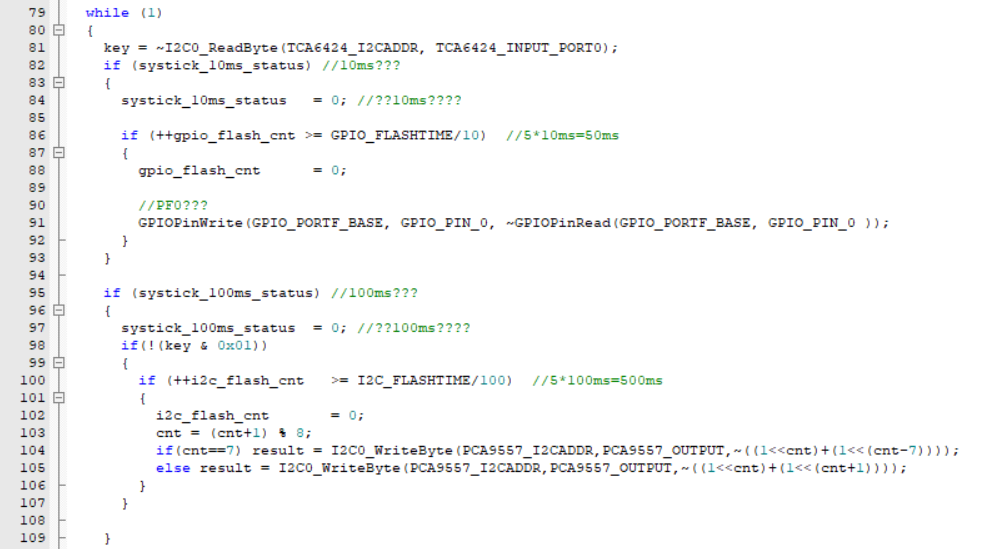


为了配合上述思路，需要在systick中断服务程序中添加实现10ms定时的语句。如果利用已有的10ms定时systick\_10ms\_status，会导致错误。因此设置新的10ms定时，因此这里的systick\_5ms\_counter和status实际上是10ms定时。



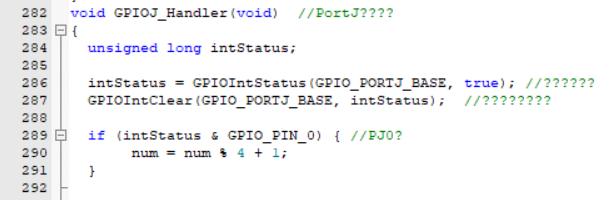
另外，在开机时需要过100ms才会执行93行的设置led操作，因此需要对led进行初始化。其实也可以直接将cnt初始值设为1，效果相同。

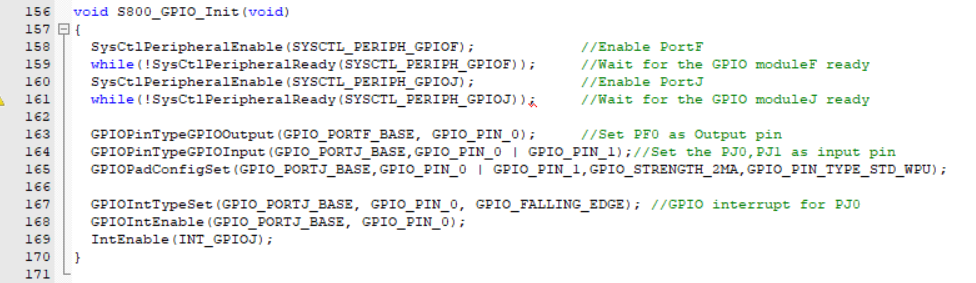
实验2.4：



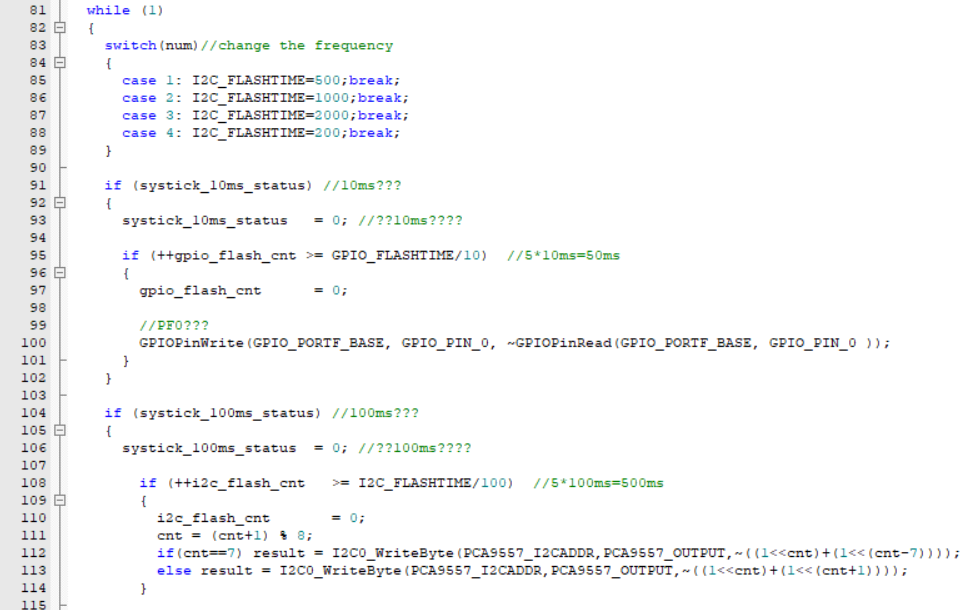
利用蓝板按键暂停跑马灯，这个操作不需要对数码管显示模块做任何改动，这是因为在本代码结构中，数码管显示模块和决定显示什么字型的模块是独立的。因此，只需要改控制cnt和led的语句即可。如图，第81行读取了按键状态，在95行的if语句块中，用98行的if判断使得在没有按下按键的时候，cnt和led正常运行，在按下按键时不执行改变语句，从而实现了暂停功能。

实验2.5：





既然需要用到红板上的按键，那么采用GPIO中断方法就是必然。如图，写好中断服务程序和中断设置。由于是四次按键为一个周期，在终端服务程序里同余4。num初始设为1。

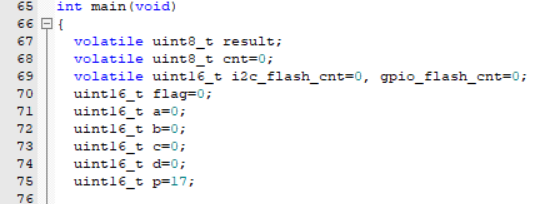


注意到108行中，是I2C\_FLASHTIME这个量在控制跑马灯频率。因此，只需要在while中判断num值以决定I2C\_FLASHTIME的大小，就能控制跑马灯频率。显而易见，I2C\_FLASHTIME越大，跑马灯变化的频率越低，周期越长。采用switch语句实现。其余代码和实验2.3类似。

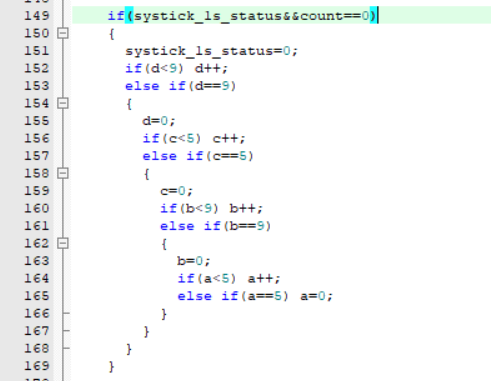
实验2.6：

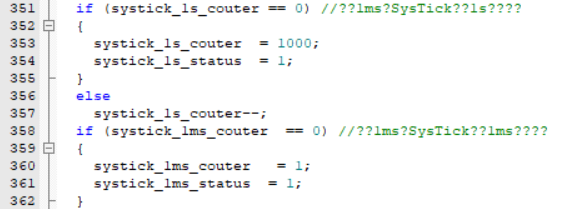


首先，由于需要显示‘-’这个新字型，在seg7数组中增加0x40字型码，位于第17位。

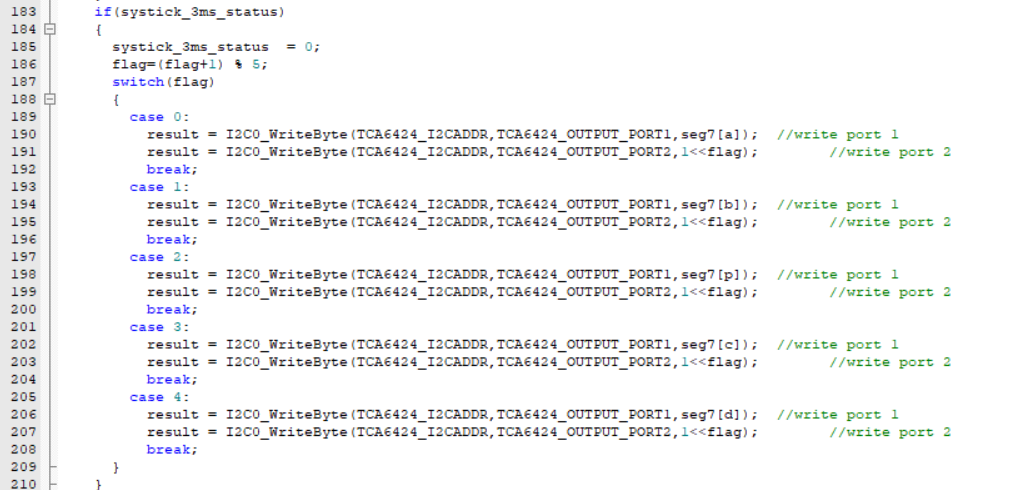


之后，增加定义a，b，c，d，p作为字型输入代码。分别代表分钟十位、个位和秒的十位、个位和‘-’。

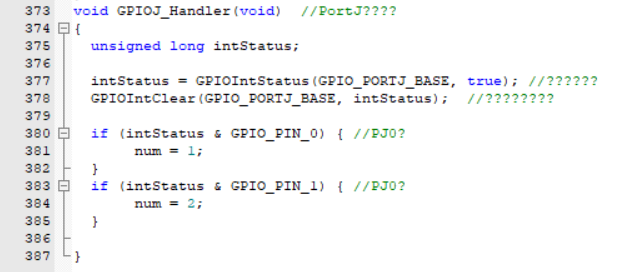




为了实现过1秒加一的功能，在systick中断服务程序中引出1s的systick定时。152行开始是进位功能的实现。

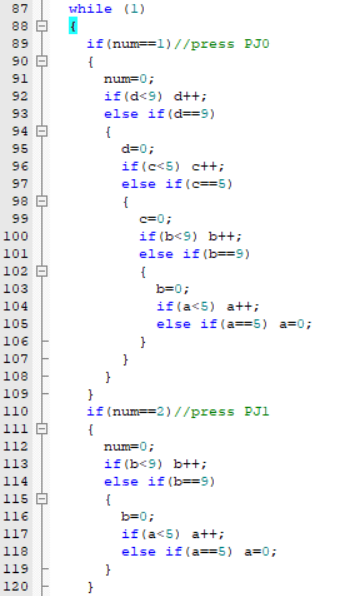


显示模块方面，由于需要显示的位数变多，10ms会出现闪烁，于是将10ms改为3ms。利用flag同余变化和switch语句决定显示哪个位。

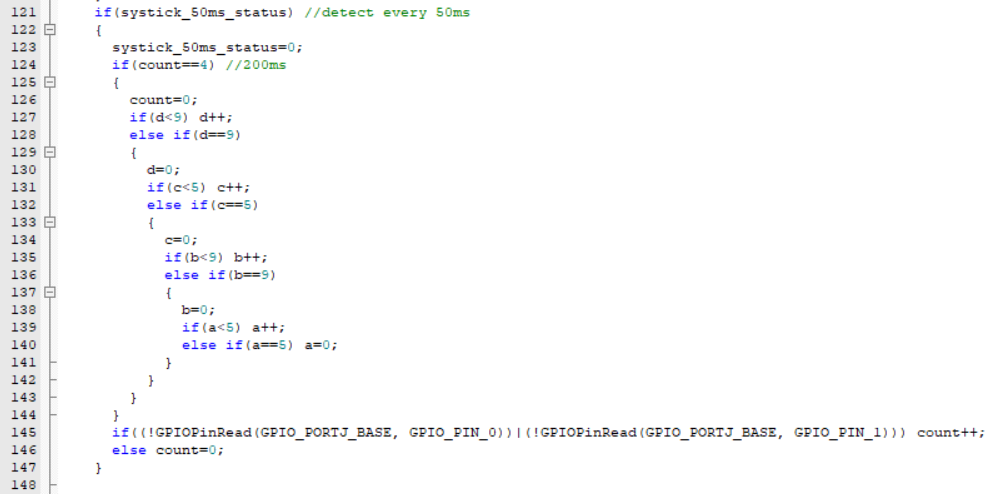




与实验2.5类似，由于需要用到红板按钮，必然采用GPIO中断。由于需要用到两个按键，因此在中断设置和服务程序中需要做一些改动。用num全局变量记录按下了哪个按键。



相应地，在while循环中判断num的值，num为1时秒数加一，为2时分钟加一。进位语句和之前类似。



最后考虑实现在按下按键不松开时，每过200ms自动加1秒的功能。尝试多种定时周期后选择每过50ms用或非语句检测一次按键状态，效果较好，在systick里设定语句与其他时长类似。当检测到按键被按下则计数器count加一，否则直接清零，确保按键被连续按下200ms才能触发操作，如145、146行所示。当count为4时，即不松手达到了200ms，触发秒数加一的操作，并清除count。

1. **讨论题**
   1. I2C是同什么类型的总线（同步/异步、串行/并行）？传输速率是多少？

答：是一种同步串行总线，传输速率：标准100Kbit/s，快速400Kbit/s，快速附加1Mbit/s，高速3.33Mbit/s。

* 1. 在同一I2C总线上最多可以连接多少PCA9557和TCA6424芯片？如果PCA9557的A2、A1管脚连接到高电平，A0管脚接到地时，它的I2C从机地址为多少？

答：最多112个。从机地址为0x1E。

* 1. 若I2C主设备传送的第一个字节为二进制数01000111，其表达的含义是什么？

答：表示从机地址和数据方向。含义为接收01000111地址从机的信息。

* 1. 若时钟源选用25M外部MOSC，则能否利用SysTick产生8s的定时？请给出设置方法或不能设置的理由。

答：可以。先设置计数初值12.5M得到500ms的systick定时，然后通过软件16分频就能够得到8s定时。

* 1. 例程2-4中3个任务的优先级为多少？程序是如何实现“任务3的优先级高于任务1，2”的要求的？

答：3>2=1。在任务3被触发时，修改其余两个任务的变量，使程序运行到另两个任务时因为不满足条件而不执行。任务3要结束时再将变量修改回来，恢复执行另两个任务。