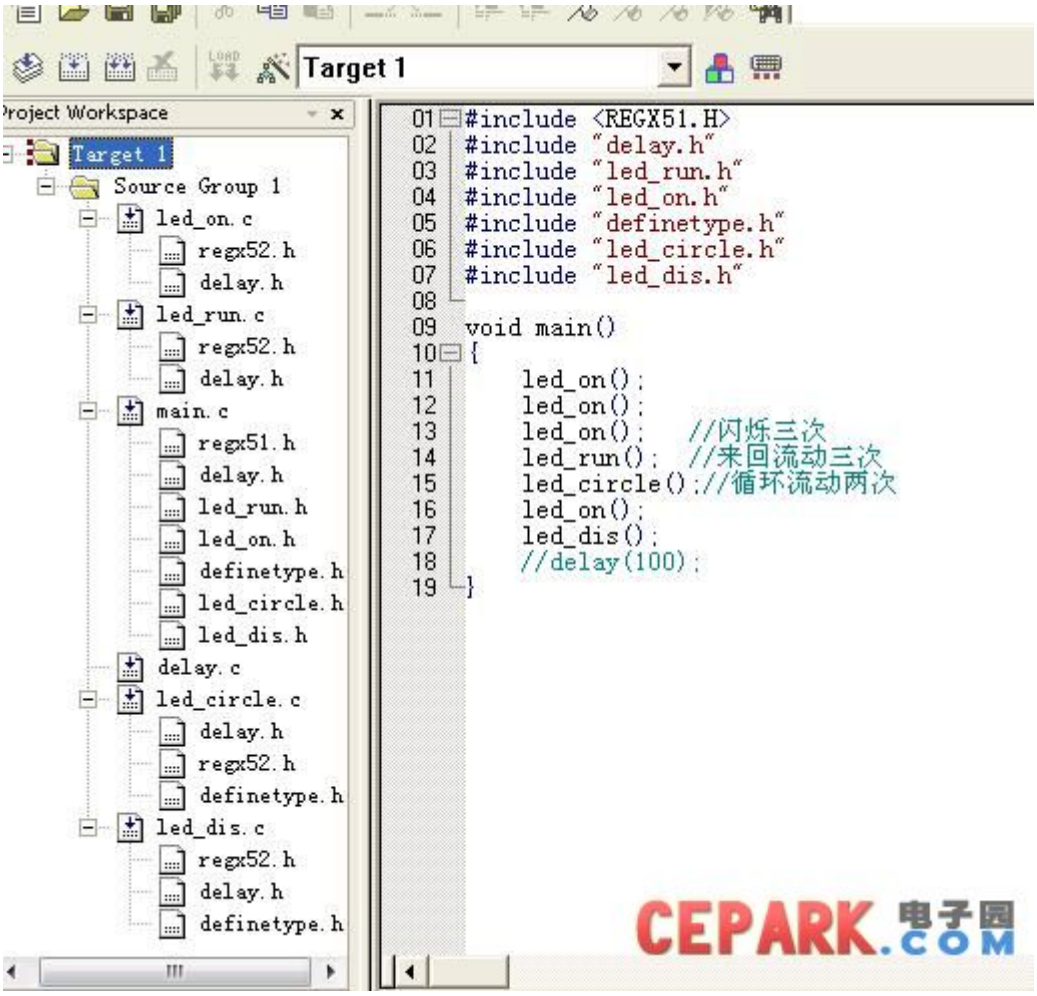
在 KEIL 中的模块化程序写法

在使用 KEIL 的时候，我们习惯上在一个.c 的文件中把自己要写的东西按照自己思路的顺序进行顺序书写。这样是很普遍的写法，当程序比较短的时候比如几十行或者一百多行，是没有什么问题的。但是当程序很长的时候，比如你要用到 LCD 显示数据，就有几个 LCD 相关的函数，然后你想在 LCD 上显示温度，那么就要有 DS18B20 相关的操作，这又有几个相关的函数，如果你还想加上去 DS1302 的时间显示功能，那么又要多很多函数。这样的话一个程序下来几百行是很正常的事情，对于自己写的程序可能在自己的脑海中比较清晰，不会太乱，但是当把自己写的程序交给别人来看的时候，别人往往会看的云里雾里，经常会看着看着就不知道你写的是什么了。

如果大家写过类似电子钟这样的比较长的程序的话，肯定在网上下载过相关的程序看过，有没有觉得别人的程序看起来让自己觉得很郁闷呢？呵呵。现在我们来介绍一种在 KEIL 中 C 语言的模块写法。这样的写法其实也是很好用的，一些稍长的程序中经常见到。结构如下图所示：

图一



是不是看起来不陌生？这就对了。其实如果学过 PC 机的 C 语言的话，对多文件的编译比较熟悉那么这个就不是什么问题了，因为这基本上是相同的。如果您是高手对此很熟悉的话，那么请略过本文；如果您是对此不是很熟悉并对此有点兴趣，那么本文或许对您有所帮助。如果在本文中有讲的不对的地方请跟帖提出。或者在我的主页给我留言进行交流。

这个教程不大容易用文字的形式来讲清楚，如果用视频来做的话效果应该好的多，但是俺没这个条件（俺普通话不好怕吓到大家，哈哈）。可能一帖会写不完，另外打字是件很痛苦的事情，所以这个请见谅。下面正式开始。

我们主的目的是学习模块化的写法，所以功能是次要的，熟悉了这个写法以后功能是大家自己控制的，我们现在将以 LED 灯的控制为例子讲起。

这样，我们先建立三个.c 的文件，分别命名为 main.c、delay.c 和 led_on.c，并将在建立文件的时候尽量做到看到文件名即能看出程序的功能，这样的话比较直观，不容易混乱。然后将这三个文件都添加进工程。（这个不能不会吧？）

在 delay.c 中我们加入如下代码：

```
void delay1s()
{
    unsigned int m,n;
    for(m=1000;m>0;m--)
    for(n=20;n>0;n--);
}
```

当然这个延时函数的实际延时时间并不是一秒，我们暂且不用管它，知道他是起延时作用的就可以了。在 led_on.c 这个文件中我们加入如下代码：

```
void led_on()
{
   P0=0x00;
   delay1s();
   P0=0xff;
   delay1s();
```

```
}
```

然后在 main.c 函数中我们添加如下代码：

```
void main()
{
   led_on();
}
```

这个程序的功能简单的很，就是实现 LED 的闪烁。

下面问题来了，就是如何将这三个 C 文件关联起来。

其实在单个.c 文件的程序中，我们在写程序的时候第一件事就是写上#include <reg52.h>，如果你是一个好学者，你一定问过为什么要这样写一句话，要是你上过辅导班，老师一定跟你讲 reg52.h 是头文件，这句话的作用的把头文件包含进来。当然这是很正确的，你可以打开 reg52.h，看一下里面的内容，里面包含了关于 51 单片机的一些定义，如果在这个文件中遗漏的东西可以使用命令 sfr 来在 C 文件中定义，如在 STC89C52 中实用扩展 RAM 的时候会用到一个寄存器你可以添加到这个文件中或者在 C 文件中用 sfr 定义。进一步想一下，一个包含命令可以把一个文件包含进来，那么用不同的头文件包含不就可以把更多的文件包含进来了吗？是不是有点思路了？

先讲到这里，下次看一下具体如何将三个文件关联起来。

我们接着上一次的讨论一下如何将三个 c 文件关联起来，在单文档的程序中我们使用#include 这个命令将单片机的头文件与我们的程序关联起来。同理我们也将以头文件的形式把我们建立的源程序关联起来。

首先，我们需要一个新文档，这个文档的建立有两种方法（以 delay1s 函数为例）。第一种，在工程目录下建立一个 delay1s.txt 然后将其改名为 delay1s.h。因为都是同编码的所以不会出现乱码，然后在工程中将其打开。第二种方法是直接在工程中新建一个文档，然后保存的时候将名字保存为 delay1s.h 即可。如果是需要添加很多文件的话建议使用第一种方法，这是个人建议。

其次，我们需要编写 delay1s.h 这个文件的内容，其内容如下：

```
#ifndef _DELAY1S_H_
#define _DELAY1S_H_
```

```
void delay1s();//延时函数


#endif
```

这个是头文件的定义，作用是声明了 delay1s()函数，因为如果在别的函数中如果我们需要用到 delay1s()函数的话，若不事先声明则在编译的时候会出错。对于#ifndef……#define……#endif；这个结构大概的意思就是说如果没有定义（宏定义）一个字符串，那么我们就定义它，然后执行后面的语句，如果定义过了那么就跳过不执行任何语句。

关于为什么要使用这么一个定义方法，比如在 led_on()函数中我们调用了 delay1s()函数，然后在 main()函数中我们也调用了 delay()函数，那么，在 led_on()函数中我就就要包含头文件 delay1s.h,然后在 main()函数中也要包含 delay1s.h,若主函数中我们调用过 led_on(),那么在编译的时候,遇到 delay1s()和 led_on()的时候就会对 delay1s.h 进行两次解释，那么就会出现错误。若有以上预处理命令的话，那么在第二次的时候这个_DELAY1S_H_已经被定义过了，那么就不会出现重复定义的问题。这就是它的作用。但是注意，在编译器进行编译的时候头文件不参与编译。

再次，我们建立一个 led_on.h，起代码内容如下：

```
#ifndef _LED_ON_H_

#define _LED_ON_H_


   void led_on();//灯闪烁


#endif
```

作用同 delay1s.h,不理解的话可以再看一下上面的解释。

最后，将我们上次说的三个函数补充完整。

在 led_on()函数中，我们用到了 51 单片机的一些寄存器的定义，所以我们要包含 reg52.h,而且我们用到了 delay1s()函数，所以我们要包含 delay1s.h，故 led_on()函数的代码如下：

```
#include <reg52.h>
#include “delay1s.h” //注意这里没有分号
void led_on()
{
   P0=0x00;
   delay1s();
   P0=0xff;
   delay1s();
}
```

Main 函数的代码如下：

```
#include <reg52.h>
#include “delay1s.h”
void main()
{
   led_on();
   delay1s();//在这里其实只有第一句就可以了，这句是不必要的
   led_on();//这也是不必要的
}
```

在这个函数中，为了再次说明一下#ifndef……#define……#endif 这个结构的定义，大家可以把所有的.h 文件中的这个结构去掉，然后编译一下看一下效果。

到这里相信大家对于这种模块化的写法就有大概的了解了，如果我们想添加新功能的时候，比如我们要添加一个流水灯的功能，那么，我们只需要增加一个 led_circle.c 和 led_circle.h，然后按照上述步骤添加进工程即可，程序的其他部分不需要任何改动。显然这是很方便的。其实函数的声明可以使用 extern 关键字，C 语言中默认都是这个类型的，所以可以不用写。

如果还有说的不清楚的请提出来，我们一起讨论。由于这些东西都是手动输

入的所以难免会有错误，如果各位朋友在看这个教程的时候发现有哪里表达错误或者是不妥当的地方，欢迎指出，我会及时改正，以免误导别人，呵呵。

最后附上一个参考例程，这个可以直接在CEPARK的51板子上看到实验效果。希望这个对大家能有所帮助。

本文下载地址：http://bbs.cepark.com/register.php?fromuid=2975

BY：kidcao1987