嵌入式系统设计及应用
实验报告

姓名：李炳辉

学号：2140504139

班级：自动化 46

指导教师：刘瑞玲

嵌入式实验中对器件的了解和操作需要阅读器件手册和实验板原理图，做完整个实验，可知具体到RL78/G13 嵌入式实验着重要阅读的是芯片手册（r01uh0146cj0200_rl78g13.pdf）、温度测量器件手册（DS18B20.pdf）、LCD 器件手册（ST7920 资料.pdf）及原理图（G13 DEMO 板原理图.pdf）。芯片手册详尽的讲解了各个功能模块的使用和配置方法，配合已有的官方例程可以更快地了解模块的具体操作方法。

官方的入门指导视频（RL78 G13 DEMO BOARD 入门指导教程）讲解了CubesuitePlus 软件的基本使用方法以及如何新建工程，正式进行嵌入式各项实验前需要先观看此视频并模仿其过程。

实验项目 1：学习 RENESAS RL78/G13 嵌入式微控制器开发环境

实验目的： 掌握 RL78/G13 的集成编译环境 CubeSuite Plus 和仿真调试工具 EZ-CUBE 的使用方法。

实验基本要求： 能够熟练使用 RENESAS 嵌入式设备的集成编译环境 CubeSuite Plus 和仿真调试工具 EZ-CUBE。

实验内容提要： 熟悉 RENESAS 嵌入式设备的编程环境 CubeSuite Plus 和调试工具 EZ-CUBE，能够根据实验要求在编程环境下设计相应的工程项目，包括文件定义、变量定义、程序结构设计、算法实现等；在 EZ-CUBE 环境下，掌握程序的调试步骤以及排除程序中的错误等。

具体任务： 1）学习视频资料和“RL78 G13 Demo 板使用指南.pdf”，创建新工程，分别用 Delay（）函数和 Timer 模块实现跑马灯功能，使 D3、D4、D5 二极管能够按照一定的规律循环点亮或熄灭；

按照入门指导视频建立新工程后，按照两种方式使二极管闪灭。两种方式均是在控制时间，只是精度不同。第一种如视频中演示，是以嵌套循环执行空操作来完成延时，这种方式虽然时间上并不准确，但在不严格要求时间控制的场合，这是非常简单方便的一种延时方式；第二种就是使用定时器模块来完成时间控制。

微机原理和嵌入式课程中均对定时器有讲解，简单来说就是一个计数器（按照所设定的时钟频率来计数，根据时钟频率和定时器预期时间可以计算出应该给此计数器设置多大的装载值）。定时器到了设定时间后会产生一个中断，我们可在此中断中执行一些简单的语句。CubesuitePlus 软件的生成代码功能中可以方便的配置定时器。具体过程如下：

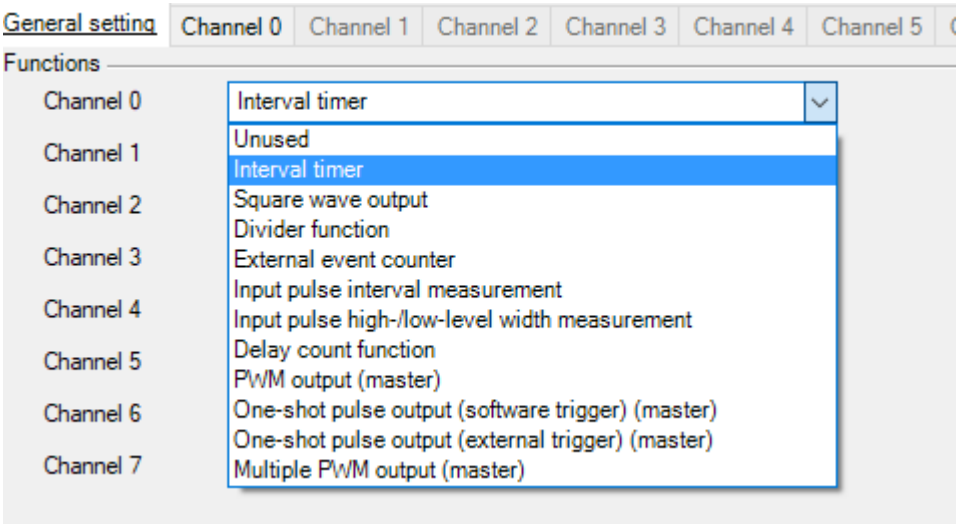

图 1.1

选择代码生成器中的 Timer 模块，任意选择一个通道将模式选为 Interval timer

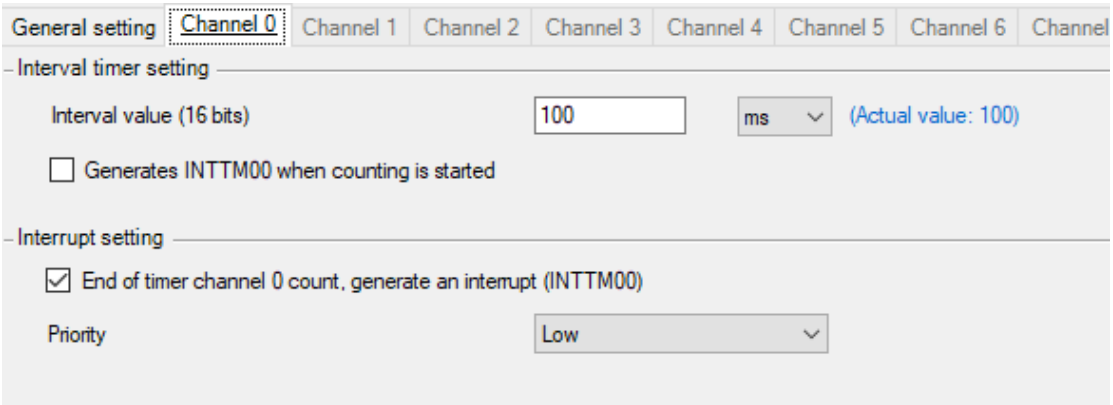

图 1.2

点击所选通道的标签可以转到具体设置，Interval value 设置定时器时间，可以选择不同时间单位，输入不同时间值。下面的 Priority 设置的是中断的优先级，一般无需设置，当开启了多个定时器通道时需要注意是否有一些中断在多个中断同时到来时需要优先执行，此时可按需求设置不同的优先级。设置完成后点击 Generate Code 生成配置代码。

生成的代码中模块的初始化已经写好我们无需再去专门初始化，只需在合适的位置调用开始计数函数（此例中为 R_TAU0_Channel0_Start（））即可。当程序运行完开始计数函数，就会每隔一段时间（我们设定的时间）进一次中断。因此，可以在中断函数（__interrupt static void r_tau0_channel0_interrupt(void)，在 r_cg_timer_user.c 文件）中写入二极管的控制，每次执行中断函数就将显示的灯换为下一个灯（比如将赋给 P4 的值左移一位），做到流水灯效果。

这里需要说明的一个额外问题就是看门狗的问题，在我们的整个嵌入式实验中因为任务都不至于过于复杂，可以将看门狗功能禁用。看门狗本身也是一个

定时器，在一定时间内如果不喂狗（也就是重置数值），看门狗就会认为程序出错跑飞了，会将 CPU 重置，我们的代码运行时就有可能显示出很奇怪的结果，为了避免忘记喂狗出现结果异常，在生成代码时要将看门狗禁用（代码生成器中点击 Watchdog Timer，选择 Unused）。

实验项目 2：处理器接口模块设计实验

实验目的：掌握处理器各接口模块的使用和编程方法。

实验基本要求：通过对各接口模块参考示例程序的学习，了解其工作原理，并能根据实验要求修改或扩充示例程序及功能。

实验内容提要：能够编写嵌入式 C 语言并完成：蜂鸣器按照某种规律工作；7 段 LED 显示、LCD128*64 显示；键盘输入；定时中断的功能。

具体任务：2）学习 Buzzle 模块的调用和编程，能够重建该工程，并控制蜂鸣器发出不同节奏和频率的声音。（注意：要保证蜂鸣器能正常输出声音，输出频率设置无需太高，1KHz～10KHZ 即可）

从原理图中芯片引脚总图可知，此单片机共有两个蜂鸣器输出引脚（如图 2.1），即 P140/PCLBUZ0 和 P141/PCLBUZ1，实验中使用其中一个输出引脚即可完成目标任务，当然也可两个一起使用做出更加复杂的变化效果。这里选择配置 P140/PCLBUZ0。

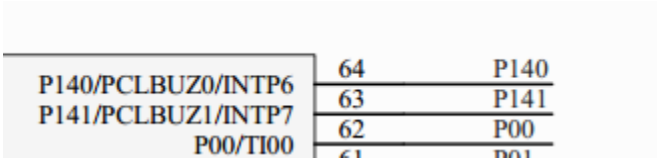

图 2.1

配置过程如下：

点击代码生成器中的 Clock Output/Buzzer Output，选择 PCLBUZ0 标签。

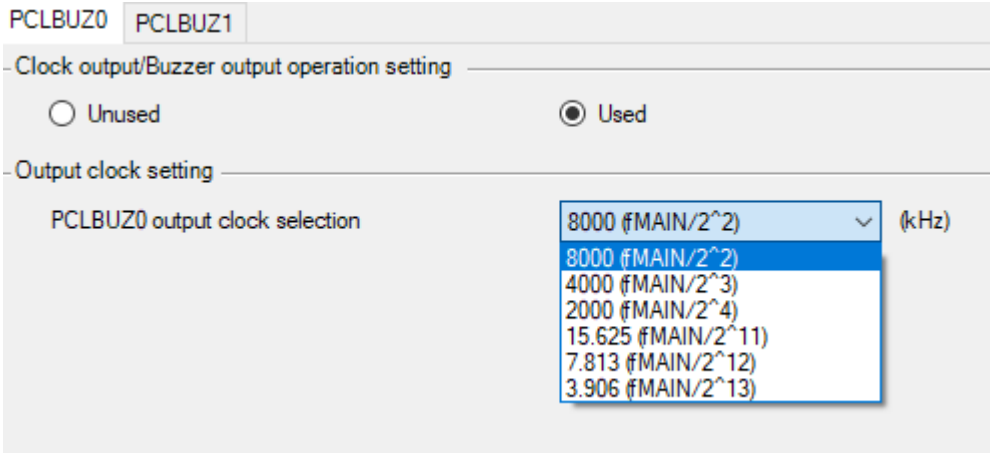

图 2.2

启用蜂鸣器，并选择初始频率。点击 Generate Code 生成代码。同样，初始

化代码已自动生成，我们无须关心。只需调用 R_PCLBUZ0_Start()开启蜂鸣器或 R_PCLBUZ0_Stop()关闭蜂鸣器。

除了像以上自己新建工程，在熟悉软件后也可以备份一份例程，直接在原例程上更改代码，只要了解需要更改的是哪些文件，都不会影响实际结果。

CubesuitePlus 的生成代码功能虽然可以很方便的配置我们需要的模块，但是新生成的代码会覆盖掉原来的配置代码，所以写入自己的代码时要写在用户代码区，用户区的代码在生成新代码时不会被覆盖掉。所有/ Start user code.....generated here */和/* End user code……generated here */之间的区域都是用户代码区，只是官方按照添加全局变量，添加函数内容，添加头文件等不同内容将用户代码区分了几大块，以便清晰有序。我们只要找到相应区域添加内容即可。**所有自己添加的代码都应写在用户代码区，以便可以随时更改配置重新生成配置代码而不必担心自己的代码被覆盖！***

我们需要的结果是单片机蜂鸣器输出具有一定节奏并能变换音调的音频片段，蜂鸣器的音调由该引脚的输出频率控制，节奏由蜂鸣器输出的时间决定——即蜂鸣器开启和关闭的时间决定，所以只需设置好这两点就可得到想要的结果。

有了工程后，接下来就了解如何更改输出频率和输出时间。

查看芯片手册第九章，可知控制电路的框图如下：

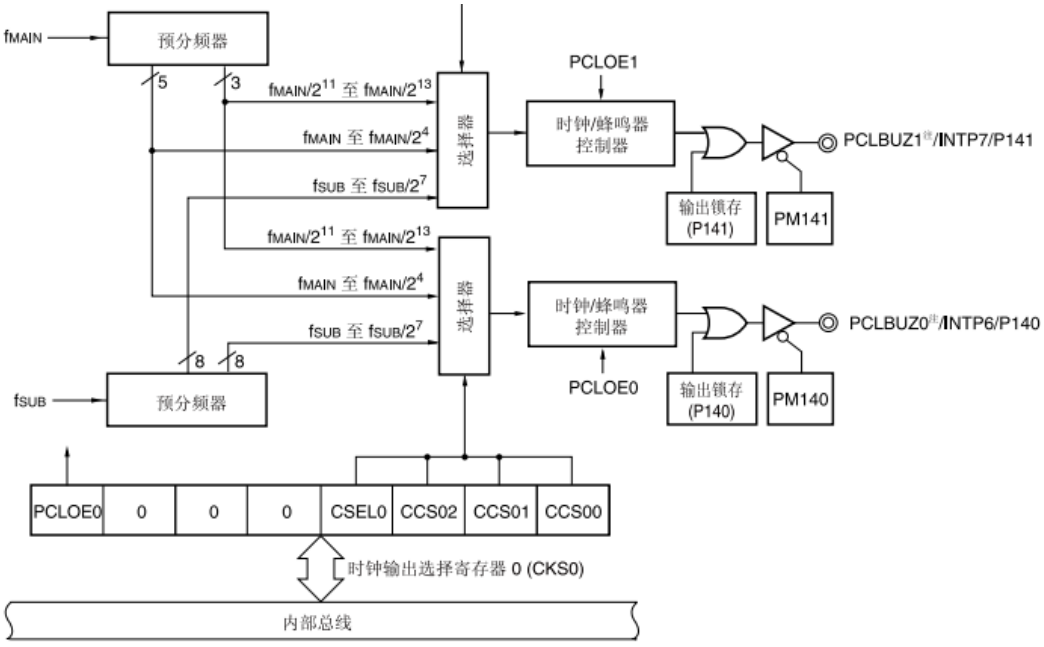

图 2.3

由图 1.2 可见 CKS0 寄存器控制着 PCLBUZ0 输出的开启和关闭以及输出频率的设置。该寄存器具体格式如图 1.3 所示：

地址：FFFA5H (CKS0), FFFA6H (CKS1)　复位后：00H　R/W

符号	<7>	6	5	4	3	2	1	0
CKSn	PCLOEn	0	0	0	CSELn	CCSn2	CCSn1	CCSn0

PCLOEn	PCLBUZn 引脚的输出允许/禁止的指定
0	禁止输出 (默认)
1	允许输出

CSELn	CCSn2	CCSn1	CCSn0	PCLBUZn 引脚输出时钟选择				
					f_{MAIN} = 5 MHz	f_{MAIN} = 10 MHz	f_{MAIN} = 20 MHz	f_{MAIN} = 32 MHz
0	0	0	0	f_{MAIN}	5 MHz	10 MHz*	禁止设置*	禁止设置*
0	0	0	1	$f_{MAIN}/2$	2.5 MHz	5 MHz	10 MHz*	16 MHz*
0	0	1	0	$f_{MAIN}/2^2$	1.25 MHz	2.5 MHz	5 MHz	8 MHz*
0	0	1	1	$f_{MAIN}/2^3$	625 kHz	1.25 MHz	2.5 MHz	4 MHz
0	1	0	0	$f_{MAIN}/2^4$	312.5 kHz	625 kHz	1.25 MHz	2 MHz
0	1	0	1	$f_{MAIN}/2^{11}$	2.44 kHz	4.88 kHz	9.76 kHz	15.63 kHz
0	1	1	0	$f_{MAIN}/2^{12}$	1.22 kHz	2.44 kHz	4.88 kHz	7.81 kHz
0	1	1	1	$f_{MAIN}/2^{13}$	610 Hz	1.22 kHz	2.44 kHz	3.91 kHz
1	0	0	0	f_{SUB}	32.768 kHz			
1	0	0	1	$f_{SUB}/2$	16.384 kHz			

图 2.4

这时可以看到我们需要控制的就是 CKS0 的位 7 以及位 3~0。CKS0 的位 7 控制蜂鸣器输出开启和关闭，位 3 至位 0 控制蜂鸣器输出的频率。

图 1.2 显示了可选的频率是怎样从系统主频分出来的。图 1.3 显示了 CKS0 的位 3 至位 0 的不同取值对应的不同频率。

目前，已经知道可以给寄存器 CKS0 赋值来控制开关及频率（比如 CKS0=0x00，选择输出频率为 f_{MAIN}，但此时不输出，开启输出后将以所设置频率输出），所差的就是开关的时间控制。时间控制最简单的方式就是前文所述的嵌套循环延时方式，也可以使用定时器模块进行更为精确延时。至此，便可以按照自己的设计编写实验主要程序。

一个基本的流程为：

a) 设置 CKS0 低四位控制频率
b) 开启蜂鸣器并延时一段时间
c) 关闭蜂鸣器并延时一段时间

重复循环以上步骤即可演奏“音乐”。

3）学习数码管（数字 LED）模块的使用和编程，能够实现从 0-20 顺序循环显示。

查看数码管的原理图，图示如下：

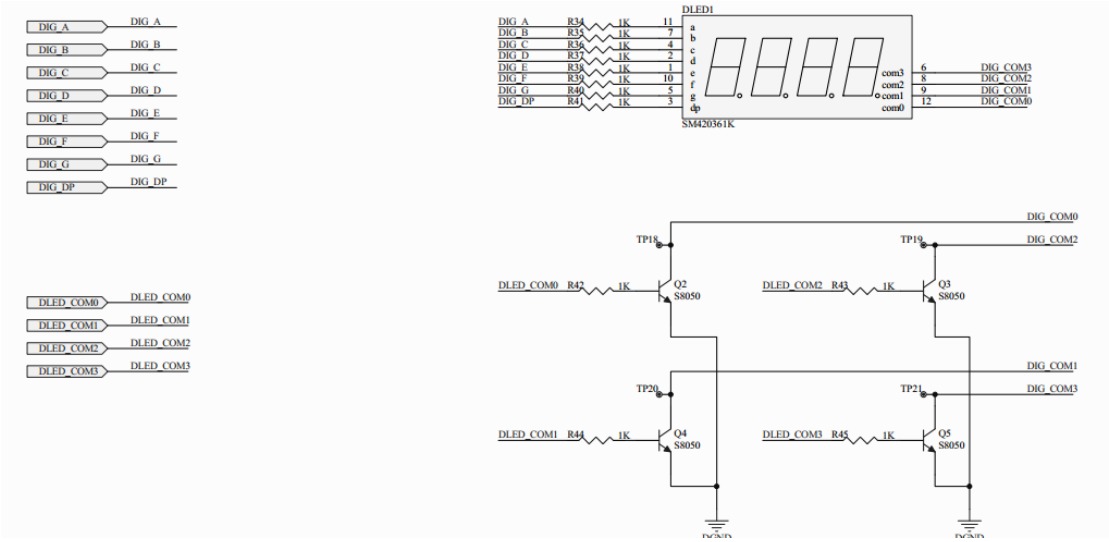

图 3.1

图中显示了所需控制的引脚名，这里只有名字没有编号，对比原理图第一页（如图 3.2 所示）可知具体的引脚是哪些。

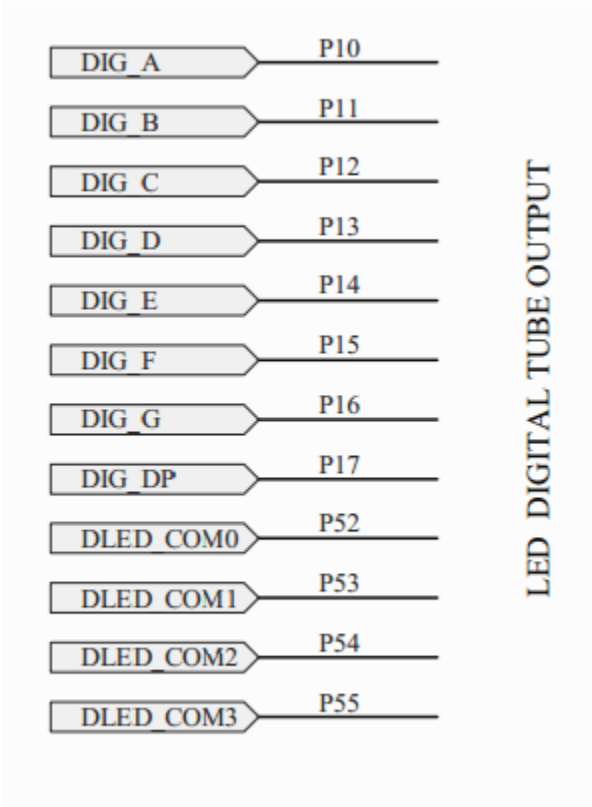

图 3.2

图 3.1 中，DLED_COM0~DLED_COM3 分别控制着四个数码管的使能与否，即 DLED_COMx 为高电平则相应的数码管 x 可显示，而数码管显示的图形由 DIG_A~DIG_DP 这 8 个引脚的电平控制。

例程中，r_cg_port.c 中显示了原理图中所涉及的引脚的配置。我们也可以自己新建工程在代码生成器中按照入门指导视频中所讲述的引脚配置方法去配置

这些引脚。

了解完引脚相关的问题，接下来要正常显示一个数字就需要了解数码管的使能控制和数字的映射两个方面。例程中的 Getcode()函数（在文件 r_cg_digtalLED.c 中）是数字变量到数码管显示的映射，dispaly()函数是使能显示控制函数。

Getcode()函数传入一个无符号 8 位整型变量，通过 switch-case 结构映射为 7 段数码管显示此字符所需的 16 进制数返回给用户，将此 16 进制数赋给控制 DIG_A~DIG_DP 这 8 个引脚电平的寄存器（即寄存器 P1），就可以在数码管使能时显示相应的符号。我们可以更改 Getcode()中的映射来自定义显示的图案。

设置好寄存器 P1 的值后，我们要控制数码管的使能来正确显示字符。从例程的 dispaly()函数可以看到，同时只能有 1 个数码管是使能状态，这是因为四个数码管共用一个输出源——即都使用 P1 寄存器设置的引脚电平，所以如果多个数码管同时使能，它们都显示同一个字符。为了显示不同字符，只能在时间上分开。

每个数码管都应该显示一定时间再关闭，这样才有一定强度的显示效果，例程中的定时器和 flag_digital_led 就是为了实现此目的。每次调用显示函数都会换一个数码管显示，为了防止一个数码管显示时间过短，所以在主函数循环中利用 flag_digital_led 判断是否调用显示函数，flag_digital_led 通过定时器中断按照固定的时间间隔使能，从而达到每个数码管显示一段固定时间后再关闭换成另一个数码管显示。

至于数字从 0~20 变化，可以配置一个 1 秒时长的定时器，每次进中断将变量值加 1，到了边界值就将变量重新置 0，主程序中就一直循环显示字符，这其实已经是一个简陋的秒表。

流程总结如下：

a) 定义一个变量，初值为 0
b) 定义一个变量 flag，作为是否调用显示函数的判断标志
c) 开启 1s 定时器，在定时器中断中对变量加 1，判断变量是否超过阈值，若超过将变量置 0
d) 开启一个短时间定时器（比如 100 微秒），在定时器中断中使能标志
e) 主函数中一直循环调用数码管显示函数，调用前判断 flag 是否使能，若是则调用。

数码管显示函数主要操作如下：

i. 将所有数码管关闭
ii. 将变量中此次需要显示的位（例如个位）分离出来传入 Getcode()得到映射值
iii. 将映射值赋给 P1
iv. 使能对应的数码管（例如与个位对应的数码管）
v. 将 flag 失能

*****选做题目***：**通过温度传感器检测当前温度值，用液晶屏或数码管显示。

温度模块的使用没有例程，需要阅读温度测量器件的数据手册自己编写相关代码。

因为要编写一个模块的代码，所以最好都写在一个新的文件中，这样如果以后要用在别的工程中，可以只把这个模块的文件复制过去。新建文件过程如下：

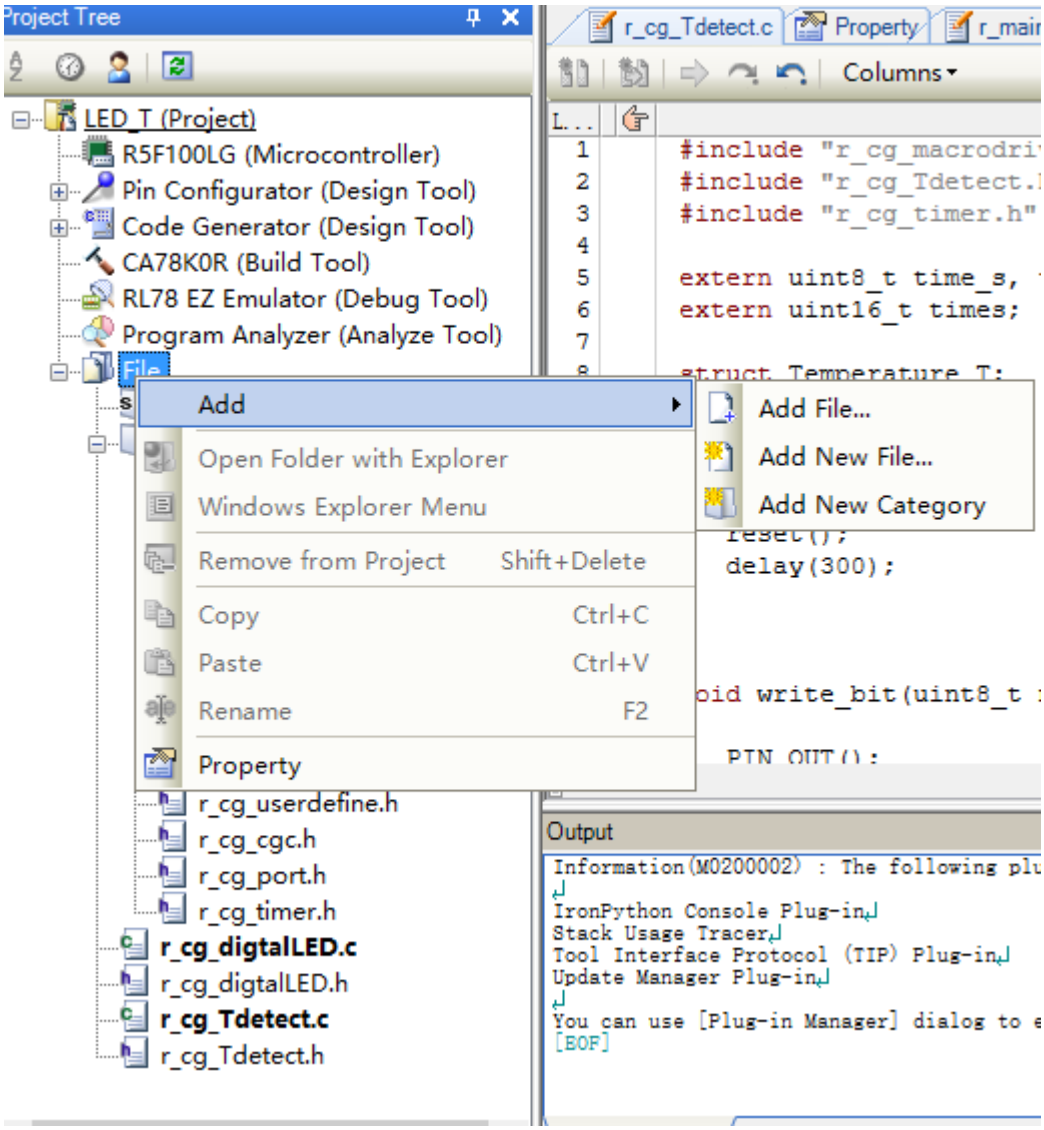

图 4.1

首先，如图 4.1 在工程视窗中右键 File，选择 Add—Add New File

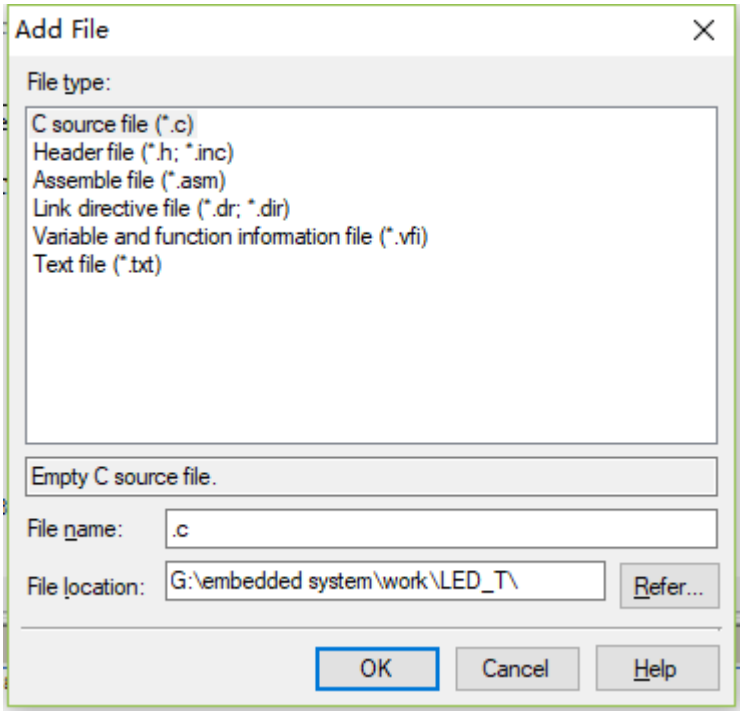

图 4.2

在弹出的对话框中选择.c 格式文件，输入一个名字后确认添加。

依照同样的过程再添加一个.h 格式文件，输入和刚才.c 文件一样的命名。

在头文件（.h 文件）中添加如图 4.3 中几行代码（按照自己的文件名添加，此例中添加的头文件命名为 r_cg_Tdetect.h）。

```
#ifndef R_CG_TDETECT_H_
#define R_CG_TDETECT_H_

#endif
```

图 4.3

最后在新添加的.c 文件中加入#include "r_cg_Tdetect.h"。之后这个模块所有的主要程序都写在 c 文件中，函数声明可以都放在相应的头文件中，在其他文件中若要调用此 c 文件中的函数，可在要调用的 c 文件的开头包含此 c 文件相对应的头文件即可。

手册中表明此模块共有四种测量精度（如图 4.4），默认是最高精度，四种精度下的温度测量转换时间不同，由于我们没有严格的时间限制，所以就用默认的精度即可。

R1	R0	Resolution	Max Conversion Time	
0	0	9-bit	93.75 ms	(t_{CONV}/8)
0	1	10-bit	187.5 ms	(t_{CONV}/4)
1	0	11-bit	375 ms	(t_{CONV}/2)
1	1	12-bit	750 ms	(t_{CONV})

图 4.4

要使用此温度转换器件，首先需要了解如何与其通信。通过器件手册可知此器件有两种接入电路的方式，对照电路板原理图，在此单片机上温度器件接成了外部供电模式，所以我们全部按照外部供电模式去控制器件。器件手册上清楚地说明了我们唯一需要控制的就是数据总线——即数据引脚 P00（如图 4.5）的输入/输出模式以及输出时高低电平持续的时间，只要满足其手册上定义的各功能的时序，就可以完成相应的通信任务。

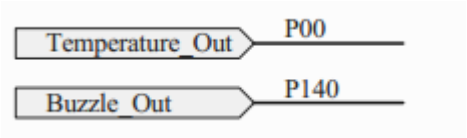

图 4.5

器件手册指出此器件工作过程分为三步，分别为器件初始化、接收 ROM 命令并执行相应动作、接收功能命令并执行相应动作，继而转向器件初始化开始下一循环。以下分别说明。

初始化时序：

INITIALIZATION TIMING Figure 13

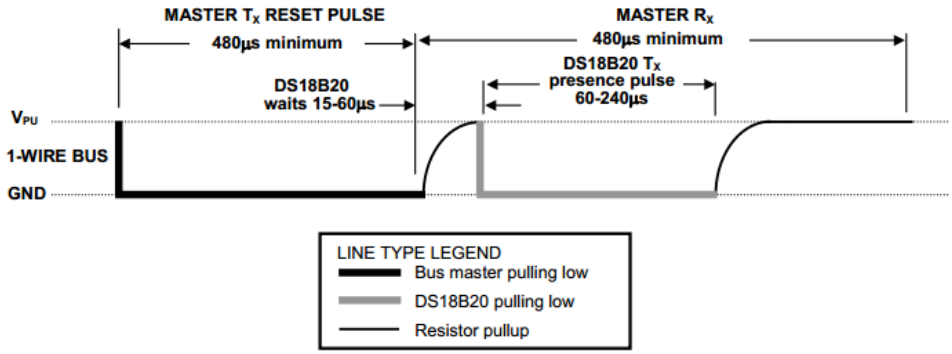

图 4.6

如图 4.6，在编写初始化函数时，数据引脚需要输出低电平至少 480 微秒后释放总线等待温度模块再次将总线拉低一定时间，之后总线恢复高电平即表明初始化完成。在此次实验中可以在程序中使用延时函数延时一段时间等待从机完成初始化而不必验证从机是否拉低总线 60-240 微秒——因为只有一个从机，只要硬件不出问题在主机释放总线后从机必定将完成初始化。可编写初始化代码如下：

```
void reset(void)
{
    PIN_OUT();//输出模式
    LOW();//输出低电平
    delay(500); //延时
    PIN_IN();//输入模式（释放总线）
}
//初始化函数
void init(void)
{
     reset();//复位函数
     delay(300);
```

其中部分宏定义如下：

```
#define PIN_OUT() (PM0 = 0x00)
#define PIN_IN() (PM0 = 0x01)
#define LOW()    (P0.0=0)
#define HIGH() (P0.0=1)
```

接下来需要向器件发送 ROM COMMAND 和 FUNCTION COMMAND，并在器件完成相应动作后读取所需内容。所以需要编写位读函数和位写函数，至于读字节和写字节在位读写函数基础上扩充即可。

器件手册中写时序如下：

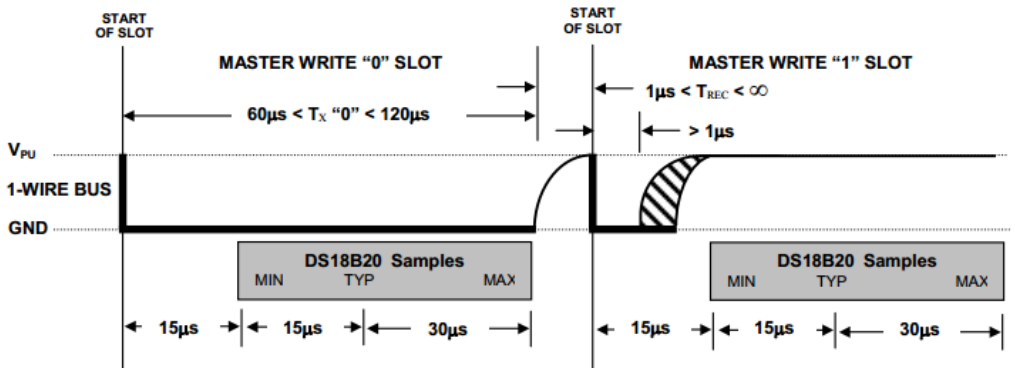

图 4.7

从图 4.7 中知每写一位需要至少 60 微秒的持续期，在两次写操作之间要释放总线超过 1 微秒。写 1 的时候要在写操作开始后 1 微秒~15 微秒之间拉高总线。位写代码如下：

```
//位写函数
void write_bit(uint8_t num)
{
    PIN_OUT();
    LOW();
    delay(1);//延时1微秒
    //若写1则释放总线使总线电平拉高；否则，延时60微秒以上再释放总线
    if(num==1) PIN_IN();
    delay(65);
    PIN_IN();
}
```

器件手册中读时序如下：

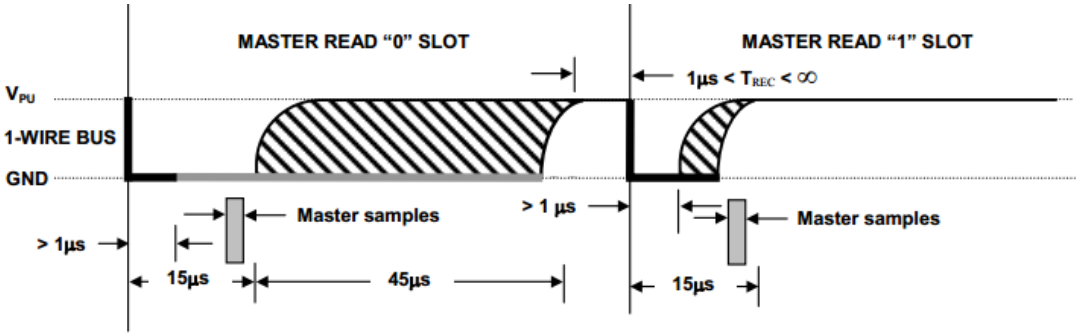

图 4.8

可以看到读一位的持续期至少 60 微秒，两次操作之间至少间隔 1 微秒，读时隙开始的标志是总线被拉低。这里要注意图中所示的采样时间限制，虽然持续期为 60 微秒但采样要在读时隙开始后的 1~15 微秒内。采样即为将数据引脚设为输入模式并读取电平。代码编写如下：

```
//位读函数
uint8_t read_bit(void)
{
    uint8_t num;
    PIN_OUT();
    LOW();//拉低总线开始读时隙
    delay(3);
    PIN_IN();//释放总线准备读数据
    delay(3);
    num = P0.0;//读出数据
    delay(65);
    return num;
}
```

完成以上函数的编写，基本的功能就有了，发送命令码时按位发送即可，也可以写一个函数，传入参数是 8 位命令码，函数内部调用位写函数发送，这样封装好的函数是很方便的。

此模块的流程和命令码有关，使用时参考器件手册（DS18B20）13,14 页发送需要的命令即可。

4）学习 LCD 液晶屏模块的使用和编程，能够实现 2-3 个静态画面循环显示，含中文、英文、数字等符号，能够通过变量控制 LCD 显示内容和位置，能够显示一个简单的图形图片。

LCD 模块的例程写的很清楚，通过例程代码的阅读就可以大致掌握此器件的使用方法，配合模块手册的一些讲解可以更快地了解此器件，并且相关的函数都在例程中写好，我们学会调用修改就可以。就像温度模块一样，此器件需要初始化，写命令等等才能正常使用，只是初始化方式和引脚的变化不同，这里可以不去详细地关心这些，例程中都已写好，想了解整个过程可以仔细阅读器件手册和代码内容。以下仅对一些使用中的要点进行说明。

此显示屏可以显示常见的各种字符，只需把相关字符的数码写入内存即可。在写入内存时需要首先指明地址再写字符码，就是需要使用例程中的 lcd_write()函数发送一次地址码，再发送字符码。因为内存地址和屏幕位置一一对应，我们可以通过控制发送的地址来控制显示位置。例程中写好了一个可以一次发送多个字符的函数 lcd_display()，传入的参数就是屏幕位置和字符串指针，我们可以搭配使用 lcd_write()和 lcd_display()来灵活控制显示的位置。

关于屏幕位置和内存关系这里还有一点需要注意：此显示屏可以显示四行的内容，每行 8 个字，对应内存地址的 80H~9FH，但是 80H~8FH 对应的是屏幕第一行和第三行，其余对应屏幕第二行和第四行，如下图所示。

图 5.1

所以在控制显示位置时，要注意地址的切换。

此器件一个地址对应 16 位存储空间，字母、数字类字符只有 8 位编码，所以一个地址可以写入两个此类字符，汉字的字符使用 16 位编码，所以一个地址只能写入一个汉字，并且汉字的 16 位编码必须写入一个地址中（例如不能将 16 位编码连续写入 80H 的后 8 位和 81H 的前 8 位），否则会显示乱码。

在显示图片时需要用到扩展指令集（0x36），然后基本的方法就是写入行、列地址确定位置，再写入数据，1 则点描黑，0 则为空白，我们可以使用其他软件将图片转换为点阵用在这里显示。由于内容较多，关于 LCD 图片显示等一些方法的使用可以参考 http://blog.sina.com.cn/s/blog_61b6e08b01016xif.html 和器件的数据手册，里面图文并茂比较详细，这里不赘述。

画面循环十分简单，在一个大循环中顺序执行显示各画面，显示时延时一定时间以便画面停留，在切换画面时先清屏再显示新画面即可。总结起来就是在一遍循环中显示画面 1、延时、清屏、显示画面 2、延时、清屏等等。一个简单示例如下：

```
while (1U)
{
 lcd_write(0x01,0);//LCD清屏
 switch(page)
   {
      case 1:
            lcd_display(0,"        Welcome       "
                  "     My     Friend    "
                  "       ('v' )~~        "
                  "       2017.09.27    ");//显示画面1
             page++;break;
       case 2:
            Lcd_test();//显示画面2
            page--;break;
   }
   lcd_delay(20000);//延时，使一个画面有足够的显示
时间
}
```

因为 LCD 的内容较多，在实验时可以在例程中多设置断点，通过调试来了解那些语句的实际效果和意义，可以更快入门。

5）学习键盘模块的使用和编程，能够定义键盘功能，例如数字 0~9，符号＋、－、×、÷、＝和小数点等。在键盘按下后能够在 LCD 屏幕上依次显示输入的内

容，能够在数码管上显示定义的数字。

原理图中，按键电路如下：

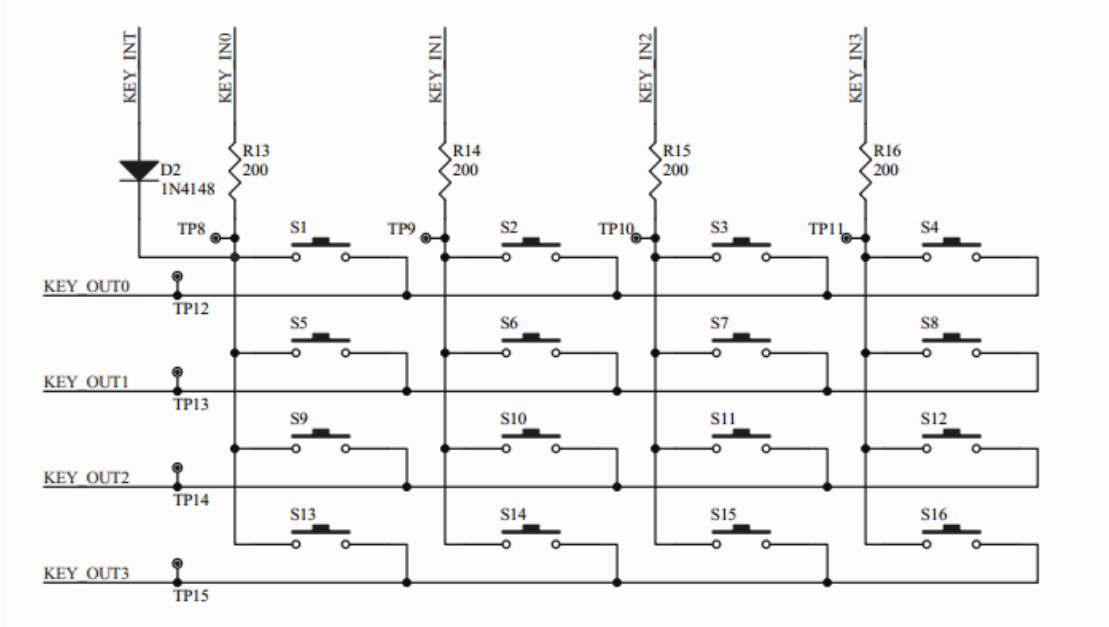

图 6.1

例程的 Keyboard_scan()函数思路就是轮流将 KEY_OUT0 至 KEY_OUT3 置低电平，之后检测 KEY_IN0 至 KEY_IN3 中哪一个变为低电平来确认 S1~S16 中哪个按键被按下。比如先给 KEY_OUT0 引脚置低电平，然后轮流查询 KEY_IN0~KEY_IN3，测试哪一个引脚变为了低电平，若 KEY_IN2 为低电平，通过图 6.1 可以看到这时是 S3 按键被按下，若无引脚变为低电平则先将所有引脚置为高电平，再将 KEY_OUT1 置为低电平继续测试，以此类推。检测到某个按键被按下时，就返回相应的数值代表此按键被按下，所以总共有 1~16 这 16 个数值。我们可以定义一个映射函数来定义这 16 个键的含义，传入参数是按键返回值，返回相应的映射值，此处为了后面综合实验计算器的使用，定义面板如下图：

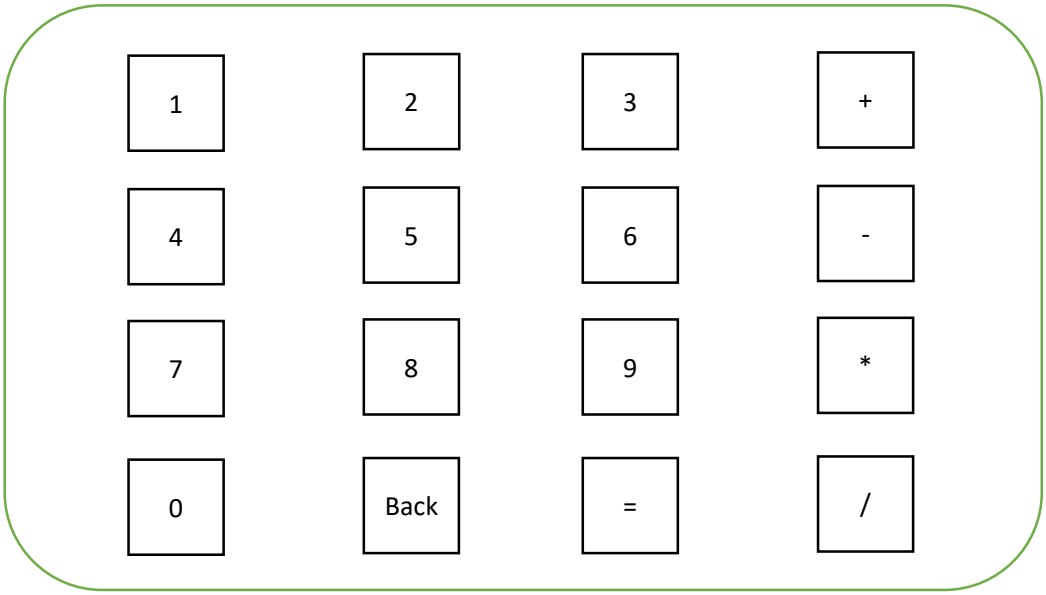

图 6.2

与面板对应的映射函数如下：

```
unsigned char key_num(uint8_t    n)
{
    unsigned char p;
    switch(n)
    {
        case 1: p='1';break;
        case 2: p='2';break;
        case 3: p='3';break;
        case 4: p='+';break;
        case 5: p='4';break;
        case 6: p='5';break;
        case 7: p='6';break;
        case 8: p='-';break;
        case 9: p='7';break;
        case 10: p='8';break;
        case 11: p='9';break;
        case 12: p='*';break;
        case 13: p='0';break;
        case 14: p='b';break;
        case 15: p='=';break;
        case 16: p='/';break;
    }
    return(p);
}
```

最后，显示就很简单了，把按键的返回值传入映射函数得到映射值，将映射值用前面学过的 LCD 相关函数发送到屏幕显示即可。

实验项目 3：综合设计实验

实验目的： 较全面掌握 RL78/G13 系列嵌入式微控制器的程序设计技术。

实验基本要求：利用基础模块设计实验的积累，完成较为复杂的综合实验任务。

实验内容提要：实现具有简单人机界面的加、减、乘、除计算器；分别采用定时中断和定时器，设计秒表和倒计时表；设计实现电子日历和时钟。

具体任务：6）制作简易计算器，在第 5 个实验基础上，实现具有简单人机界面的加、减、乘、除计算，并在 LCD 上显示输入内容及计算结果；

7）制作秒表和倒计时表，通过按键操作、通过 LCD 显示，倒计时结束时蜂鸣器发出提示音；要求：通过键盘实现“启动”、“停止”、“数值设定” 等功能，在液晶屏上有“2-4 位数字显示”。

8）制作电子日历和时钟，在液晶屏上显示当前日期年月日、在液

晶屏（或数码管）上显示当前时钟，能够通过键盘设定年月日和时间，数值能够自动正确累加。

这里直接做成一个系统，通过菜单调用各模块。对于菜单调用式的系统，常见的可以设计为上下键选择菜单或数字键选择菜单，上下键切换的方式需要频繁的按上或下键来选择相应选项，不够灵活便捷，这里选择数字键方式，即每个选项对应一个数字，按下相应键就调用相应选项。

实现此功能可以选用 switch-case 结构。显示界面每行对应一个选项，从上到下依次为 1,2,3,4，当按下相应按键后，通过 switch 选择相应分支来执行相应函数，跳转到分支函数里后清屏重新绘制界面或执行相应功能就做到了菜单调用。结构函数如下：

```
//主菜单，在main()函数中循环运行
void main_menu(void)
{
    while(1)
    {

        lcd_display(0,"1.秒表计时          "
                      "2.计算器            "
                      "3.时钟与日历        ");//菜单显示
        Keyboard_scan();//按键扫描
        switch(key_num(num_keyboard))//key_num将按键值映射为自定义面板字符，通过switch选择不同分支
        {

            case '1': num_keyboard = 0;//按键值置0
                      lcd_delay(2000);//延时一段时间防止运行过快，手指还未抬起就被子函数中的按键函数再次扫描
                      time_menu();//秒表计时菜单子函数
                      Lcd_Clear();//清屏
                      break;
            case '2': num_keyboard = 0;
                      lcd_delay(2000);
                      EvaluateExpression();//计算器子函数
                      Lcd_Clear();
                      break;
            case '3': num_keyboard = 0;
                      lcd_delay(2000);
                      tim_calendar();//时钟日历子函数
                      Lcd_Clear();
                      break;
            case 'b': num_keyboard = 0;
                          return;
        }
        num_keyboard = 0;
```

```
    }
}
```

这里需要注意一点，由于按键扫描函数执行速度很快，在得到按键值进入相应分支后且在进入分支里的函数前要延时一段时间（上面 lcd_delay(2000)），防止快速进入分支函数里后造成按键连按现象。

有了这个整体框架后，接下来就是填充各个分支里的功能函数。

①秒表与倒计时

秒表和倒计时功能都比较简单，使用时长为 1 秒的定时器即可。秒表每次进中断后时间加 1，在秒表显示界面不断刷新 LCD 屏幕使之动态显示，并且在秒表函数里要不断执行按键扫描函数以便可以随时退回上一界面。

倒计时表逻辑稍微复杂一些，需要多加一个设置选项。在程序初始时，倒计时时间设置为 60 秒，编写一个设置时间子函数用来手动调整倒计时时间，在进入此函数后，扫描按键，当扫描到按键 2 或 8 时，将倒计时初始时间加 1 或减 1（将表示倒计时时间的变量做加减计算），扫描到“=”键确认设置，之后可以选择开始倒计时分支函数。倒计时原理与秒表一样，只是在定时器中断里要做时间减 1 操作，当检测到时间为 0 时启动蜂鸣器一段时间。设置倒计时时间部分代码如下：

```
Keyboard_scan();//扫描按键
if(key_num(num_keyboard)=='2')//按键2代表调高时间，即时间加1
{
    num_keyboard=0;
    lcd_delay(2000);
    t0++;//时间加1
    if(t0>60) t0=1;//限制倒计时初始值在1~60s之间
}
```

②计算器

计算器的功能决定着算法的复杂度，这里由于是简易计算器，并不需要复杂的功能，再加上按键的限制，最终选择可以多个数多符号同时计算的功能，也就是说可以计算多个数的加减乘除，这就需要在设计中能够存储多个数字及算数符号，并且判断符号计算的优先级。这可以使用栈的方式来实现。算法主要流程如下：

a) 输入字符 char，若为数字字符则进入数字处理（即转换为数字并等待输入下一字符），若仍为数字字符则转换为数字直到输入为算数符则将此数字压入数字栈中并跳转到算数符处理 b)；
b) 若输入为算数符，则将 char 与算数符栈顶的符号比较优先级，若优先级高则执行 c)，若优先级低则转向 d)；
c) 入栈继续输入字符，按字符类型转向 a)或 b)；
d) 将算数符栈顶符号出栈，将操作数栈栈顶的两个数出栈，这三个变量一并送入计算函数中得到结果数并压入操作数栈中，然后继续将 char 与当前算数符栈的栈顶符号比较，按优先级结果转向 c)或 d)。

实现方法中比较优先级的操作需要编写一个映射函数和一个预定好的优先级表。优先级表使用一个二维数组来存储，行和列分别代表各个运算符（顺序

是+ - * / =），内容为大于小于等于三种符号，符号就代表所在行和列这两种运算符的比较结果。映射函数的作用是将传入的两个运算符映射为对应的行号和列号，并从数组中读出对应比较结果。优先级数组和映射函数具体如下所示：

```
char operand[5][5]={
   {'>','>','<','<','>'},
   {'>','>','<','<','>'},
   {'>','>','>','>','>'},
   {'>','>','>','>','>'},
   {'<','<','<','<','='},
};//存储优先级比较结果的数组
char precede(char d, char c){//算数符优先级比较函数

   uint8_t    num1,num2;
   num1 = transform(d);
   num2 = transform(c);//将传入的两个运算符映射为对应的行号和列号
    return operand[num1][num2];//返回比较结果
}
```

流程中的计算函数如下：

```
float Operate(float a,char theta,float b){//传入浮点数a,b，算数符theta
   float result;
   switch(theta){//根据算数符选择相应计算分支
   case '+':
            result = a+b;
            break;
   case '-':
            result = a-b;
            break;
   case '*':
            result = a*b;
            break;
   case '/':
            result = a/b;
            break;
   }
   return result;
}
```

此算法只是计算器的一种实现思路，功能设定不一样，算法就有很大变化。而且就此算法来说，栈的实现可以定义一个栈的结构体和相应的一套函数操作，也可以用数组和指针来实现，方法很多。

显示方面在前面的实验中单独了解完 LCD 模块后就没什么难度，只要每次输入字符后将其写入 LCD 显示即可。

③日期和时钟

日期和时钟同秒表的任务类似，子菜单中有设置选项，可以设置时间和日期，这里时间设置是设计成了直接输入数字（将键盘输入值赋给时间变量）；日期设置设计成了按 2 或 8 键来加减初始日期值（与倒计时初始值设置是一个道理），并且按 5 键来切换设置的目标（即设置年或月或日）。显示界面不断刷新动态显示即可。

综合实验操作方法：菜单界面，按数字键选择菜单对应选项；倒计时设置使用 2 或 8 键增加或减少时间，‘=’键确认设置；时钟设置，选择对应选项后，按数字键输入时间，“=”键确认设置；日期设置，5 键切换设置项目，2 或 8 键增加或减少设置值，无需确认。所有的界面，按 b 键都可返回上一界面。

综合实验总的来说复杂度虽然高一些，但是并不难，只是需要灵活地把前面学过的各个模块结合起来，而且实现目的的方法很多，最后结果正常都是好的。

实验总结

嵌入式实验刚入手时比较麻烦，因为任务量很大，要看很多内容才能熟悉软件的使用方法和嵌入式的编程方式，但渐渐了解系统之后就方便很多，软件中带有的生成代码功能也让配置硬件变得简单，只要对照着数据手册和例程就不难学会这些基本的内容。虽然需要投入的时间比较多，但在这些模块熟悉之后就可以按照自己想法搭建系统也是很让人兴奋的一件事——尤其是在成功工作时。此次实验中学到了许多知识，也锻炼了自己调试代码的能力，这也是很需要毅力的一件事，同时也感受到了单片机的强大能力，是非常有意义的一个实验。