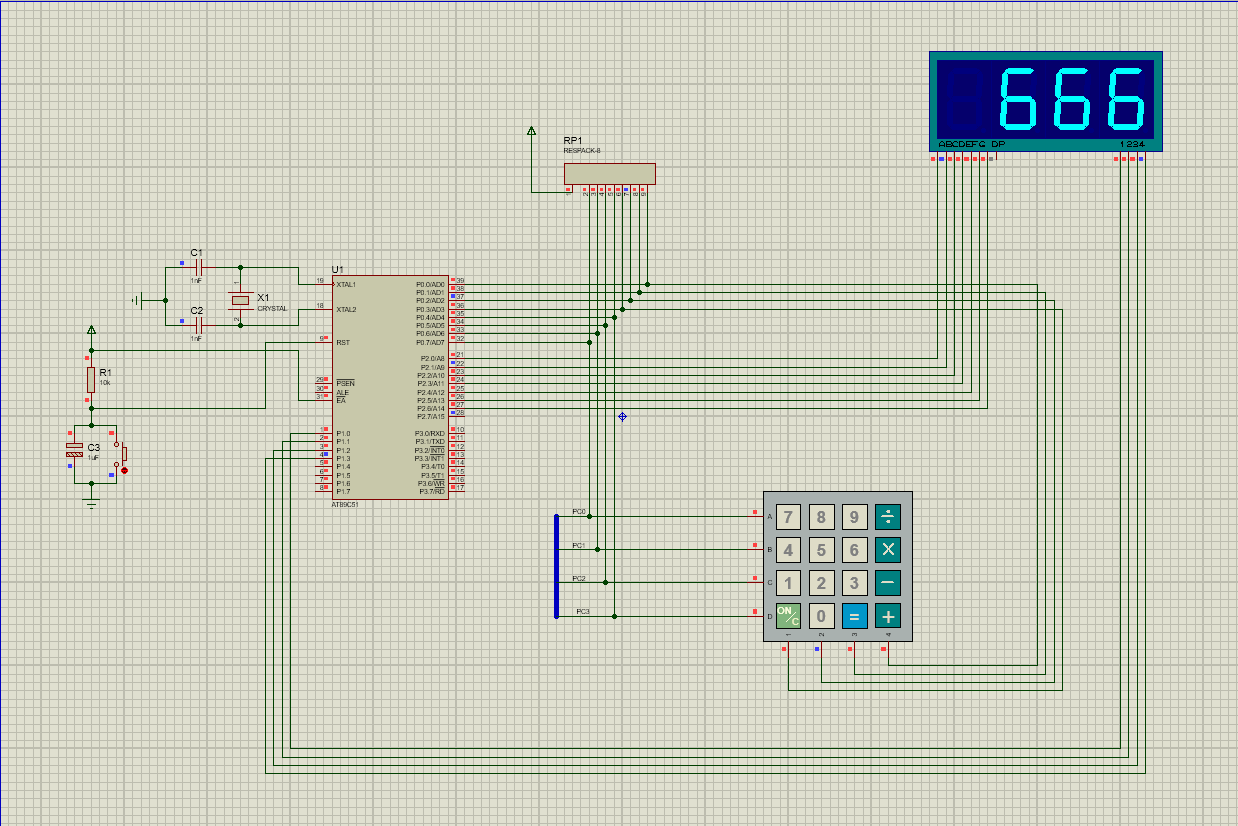
**实验报告二 键盘和显示仿真**

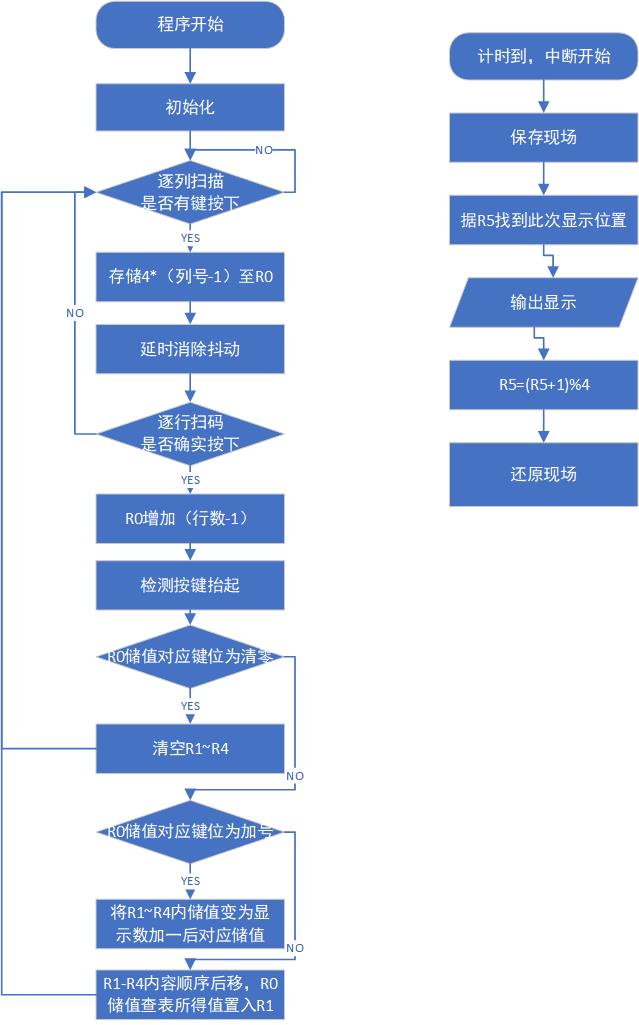
一、电路图



根据所给样例改造完成，将透明连线的4\*4按钮组换为keypad-smallcal，两个数码管换为四位数码管组，原件1-4口连P1.0-P1.3控制哪位数码管显示，a-g口连P0控制显示

二、程序分析

1、流程图的形式给出程序设计的思路



1. 代码及必要注释。

四张图，分别为主函数，调用函数，以及中断，在start前还有语句未被粘贴如下

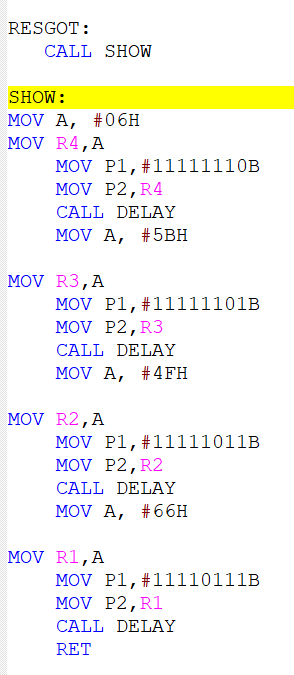
org 0000h

jmp Start

org 000Bh

JMP LABEL1

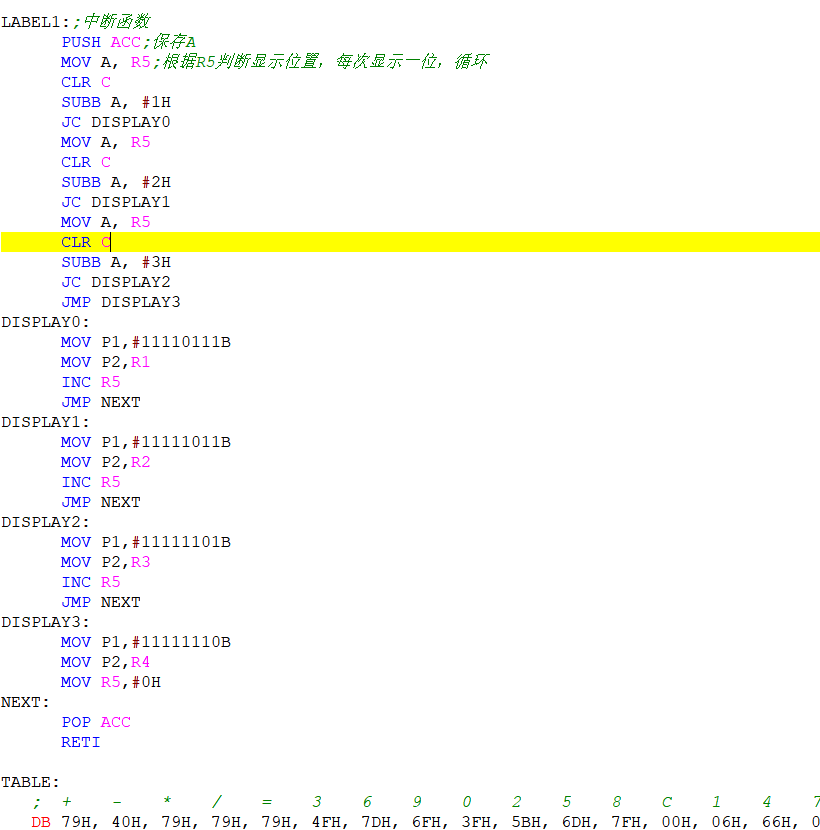
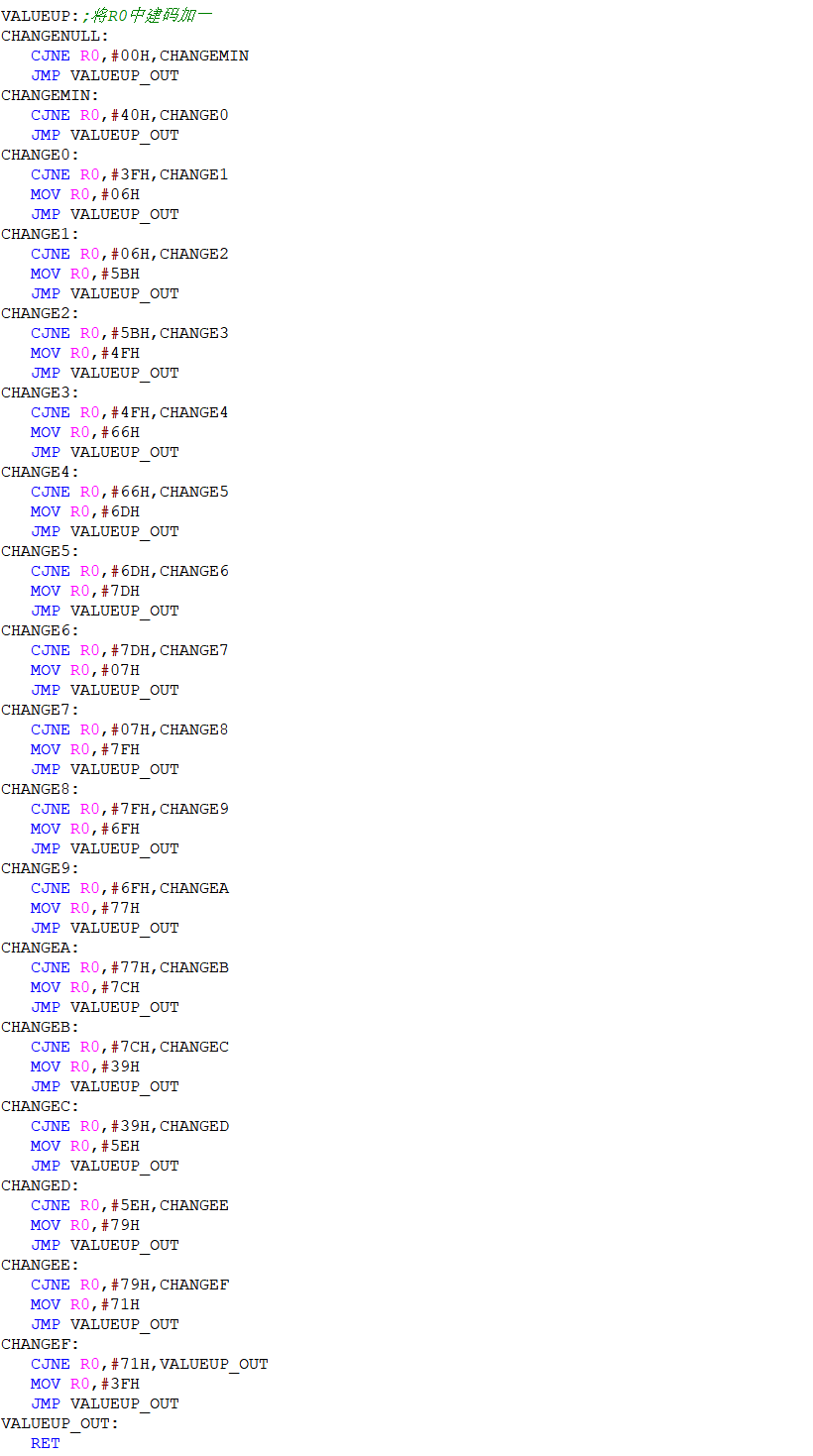
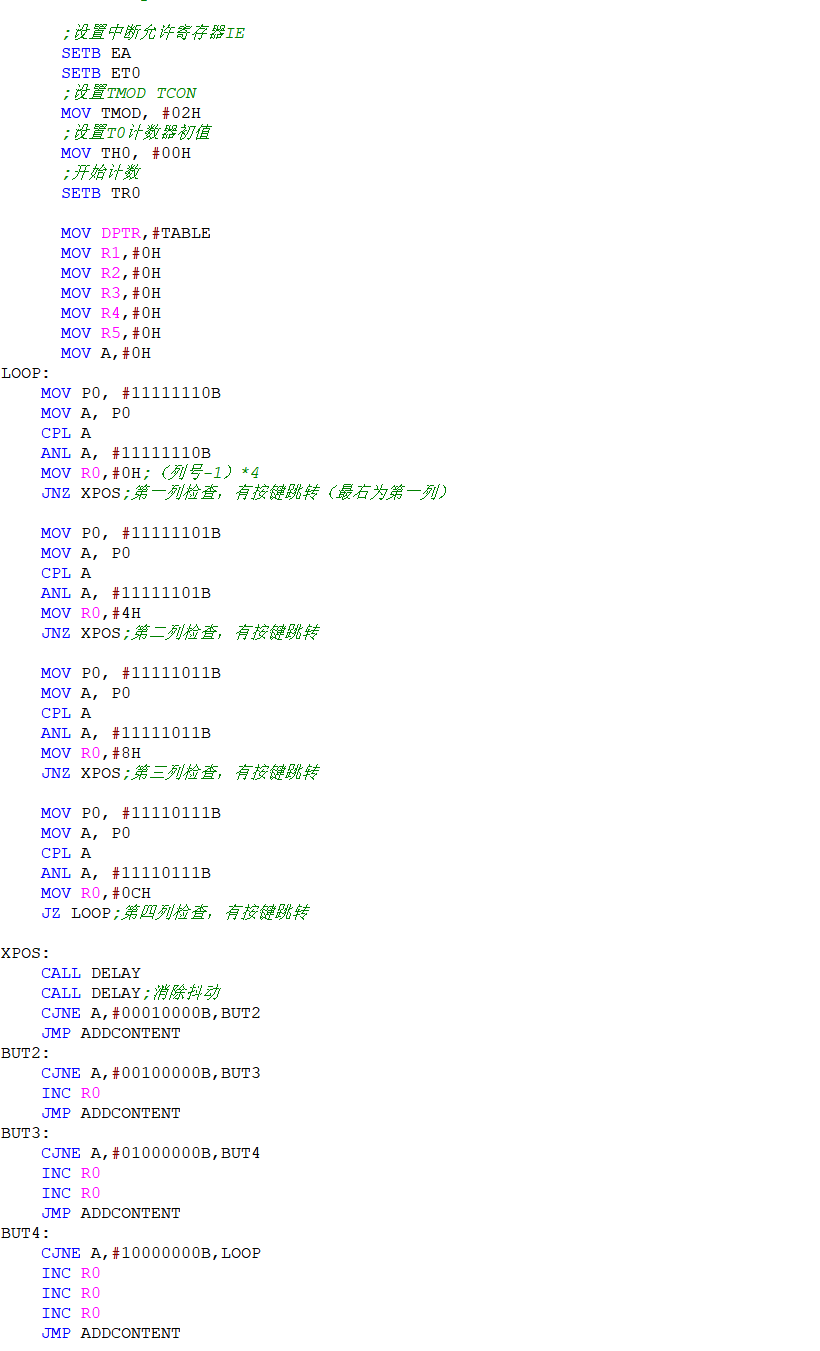
实验要求一延时扫描显示:



实验要求二中断显示，此程序即中断显示

实验要求三要求按下按键后所有显示加一，已内置，按下samllcal的加号后会实现（16进制循环加）

实验要求四要求不同按键对应值，已实现，数字对应数字，C键清理，减号为“-”（不会受到要求三加号影响），加号为要求三所述，其余符号会显示E（此为学习通演示视频中所示结果，不过学习通中按键显示E应表示error，此处将其当做十六进制E，会受加号影响加一）

1. 
2. 思考题（见PPT）

1：分析键盘矩阵的接口方式和工作流程

答：键盘矩阵只需（行数+列数）个引脚便可控制（行数\*列数）个按键（此次使用4\*4键盘），工作流程即先确认行（列）位置再确认列（行）位置，此时可得其唯一坐标。

具体例：本次实验中，先对关于列的四个引脚分别输出低电平，其余口高电平，若关于行的四个引脚未接受到低电平，则说明该列无键按下，如有，则说明对应列的对应行处按钮被按下

2：在步骤2，不同定时器时间会对实验效果产生什么影响

答：将定时器时间设置过长可能会导致显示明显的闪烁（因为实际为四位轮流闪烁），同时，可能会导致显示的延迟（即按键抬起后需要时间反应）。虽在模拟仿真中无法看出，定时器时间越短，由于向数码管输出显示频率提升，数码管应会更亮一些

3：在实际使用中，键盘按键会产生抖动，如何在程序中去除抖动影响

答：此次所附程序中已有简单消除抖动，可以见流程图了解。在收到列信号确认行号前，先进行一个短延时，延时后若信号消失则不予理会，可以保证抖动会被消除不计入。同时，可以采用硬件措施如R-C滤波电路或加上RS去抖电路。

4：当键盘中有多个键同时按下，如何得到全部的按键代码

答：仅得到所有按键代码仍还简单，在一轮输入中，对应列进行电平输入后，可以对当列的四个按键都进行一次是否按下的检查，根据输入列的电平信号（如1110）与得到的信号（如当列两个按键按下，0101）结合可以得到当列对应按键代码，可以进行一个分离处理（如得到1110 0111和1110 1101），也可以就如此进行存储保留，后续再进行处理，逐列如此扫描确认保存即可得到所有按键代码

但是要将所有按键按普通单个按键按下抬起进行保存，还需对原程序进行改进

首先因原程序检测按键抬起时会进行循环，主程序扫描无法继续进行，若它列中有按键按下不会扫描，所以可以改进将扫描与抬起检测按键存入进行分离，或者在中断中检测抬起。此处提出一种解决方法：

主程序中进行循环列扫描，每列扫描数据（后4位二进制数）会分别存入四个变量（初始四个为无按键按下时数），每次扫描将所得数与先前变量异或（可得哪位有变化，变化位为1其余全变0），再与现在所得数进行与操作，此时所得如有某位为1，则相应按键恰抬起，存入按键。

例：对某列，此时对应变量存储值为1111，按下一个键，扫描得1110，1111与1110异或得0001再与1110和操作得0000无弹起，抬起后，扫描得1111.与1110异或得0001再与1111和操作得0001说明对应位弹起。多按键按下时也可确定是否弹起，随后处理，

需注意，多键同时抬起可能存在两个位为1，处理时注意，还有若处理程序太长，可能导致按键按下未能被扫描，此情况最好主程序扫描与处理程序并行运行。

四、问题分析

1、实验过程中遇到的问题及解决方法。

一：在编写确定按键位置的代码时，先是想用一种先向行的四个接口输入低电平以确认列号，再向列输入低电平以确认行号，以此得到位置的方法，但实际运行不起效

解决方法：逐列输入低电平确认当列是否有按键按下，此方法对特定口仅进行输入或仅进行输入，实际运用可行，而且确实比起初版方案在应对更高层次要求（如重键情况下的分别按键位置寻找）要有优势。

注：经搜索并未查到能证实keypad-smallcal组件内部与所给演示文件键盘矩阵不同的证据，但此方法确实似乎在原本的电路上可行而变为smallcal后无法实现，猜测其内部有不同电路，或者可能是由于相同口会被作为输入和输出口轮换使用所致。

二：在抬起检测函数中，一开始使用类似演示文件中的方法来确认是否有按键抬起（对行/列接口四个都输入低电平），发现无法实际实现作用

解决方法：此问题实质与问题一原因应相同，最后通过检测对应列的信号是否重新变为无按键时状态解决

三：一开始写中断时无法进入中断

解决方法：尝试不同的初始化方案，并上网搜索解决

四：在自己电脑上下载仿真软件使用时库出现问题，无法正确运行

解决方法：上网询问处理后解决

1. 实验的收获或感想。（可选）

此次实验难度徒增不少，运用的语句种类也比上一次实验中的要多不少，存储数据的要求也上升了，因而也感受到了该种的些许问题（不过也许是自己的问题……）——麻烦，许多的语句只可作用于给定的极少量寄存器，导致在操作前需要颠来倒去，长度上升不少，push操作似乎也仅可以对ACC进行，导致在数据存入时比起先前所学的汇编也有些麻烦之处，不过这次的实验也是一个适应的过程，相信未来的自己能够更加灵活地运用该语言。

除此之外处理键盘带来的麻烦也不少，除去在“实验过程中遇到的问题及解决方法”中的问题外，还遇到个问题便是对于键码的处理。

此次开始时使用了单表映射，确认行号列号后直接根据（（列号-1）\*4+行号-1）确认唯一的表中位置，取出相应码值，所以table的表现如程序截图最后一行中所致，相连的数值实际在表中位置并不相连，导致在后续知道要实现各个位数字加一要求后，需要根据每个数写不同代码来重新计算码值，导致代码冗长，但是如果开始就写双表映射，一个表用于确认键盘上相应位置对应的按键内容，再一个表根据这些内容在一个有序的表中查找对应的数码管码值，那么在实现加的时候，就可以直接简单而通用地加一查找即可。这个经历让我知道了写程序目光还是长远点好……要考虑到后续可能添加要求会导致的程序改动，尽力写出易与其他功能对接的程序。

这次代码实现难度较大，我们没有在意逐步的要求，起手就将从最难的步骤4进行设计导致我们在制作过程中碰到了不小的麻烦。我们先将图接好然后程序分两部分进行编写。我在编写中断部分前，反复阅读了学习通上有关中断的定时器的资料好几遍，并且将相关信息上网整合搜索，最后设计出来第一版初始化程序设置控制字，但是接入系统后并没能如愿以偿地成功进入中断。我再次反复研究，并且参考同学的成功案例，我发现其实不需要设置每一个控制字，将自己确定需要的控制字设置完毕即可，因为有的控制在的设置我并不了解，不能轻易就将不关心的控制字设置为0或1，应该保留其初始值，系统其他功能有可能会用到。改完这一步部分后就可以成功进入中断了。但是进入中断后我们发现程序依然无法正常显示，我们又发现了识别按键抬起的函数有问题，并不能照搬照抄样例代码中的抬起函数，是由于键盘内部结构不同导致的。在处理完这两个大问题之后我们的程序就基本可以正常显示了。过程中我们还有遇到了一些由于不熟悉硬件和对应汇编语言导致的小问题，比如说一些指令必须借助指定寄存器完成，还有两个人工作衔接的时候有重复使用寄存器的需要逐一压栈，完成后再弹栈，压栈和弹栈也并不是所有寄存器都可以进行的。在完成最难的步骤四后，我将原本代码删改微调便轻松的完成了步骤1和步骤2。在这一段实验当中，我切实的感受到了自己经验的积累和能力的提升。