MSP430 单片机的开发及应用

2004.9.6开始看，其实以前看过，忘记了。再看！

设计人：陈小忠

西安邮电学院电子信息工程系电子 0002 班

西安邮电学院 63#　　710061

2003 年 7 月

目录

第一章 概述

MSP430 是德州公司新开发的一类具有 16 位总线的带 FLASH 的单片机,由于其性价比和集成度高,受到广大技术开发人员的青睐.它采用 16 位的总线,外设和内存统一编址,寻址范围可达 64K,还可以外扩展存储器.具有统一的中断管理,具有丰富的片上外围模块,片内有精密硬件乘法器、两个 16 位定时器、一个 14 路的 12 位的模数转换器、一个看门狗、6 路 P 口、两路 USART 通信端口、一个比较器、一个 DCO 内部振荡器和两个外部时钟,支持 8M 的时钟.由于为 FLASH 型,则可以在线对单片机进行调试和下载,且 JTAG 口直接和 FET(FLASH EMULATION TOOL)的相连,不须另外的仿真工具,方便实用,而且,可以在超低功耗模式下工作,对环境和人体的辐射小,测量结果为 100mw 左右的功耗(电流为 14mA 左右),可靠性能好,加强电干扰运行不受影响，适应工业级的运行环境,适合与做手柄之类的自动控制的设备.我们相信 MSP430 单片机将会在工程技术应用中得以广泛应用,而且,它是通向 DSP 系列的桥梁,随着自动控制的高速化和低功耗化 , MSP430 系列将会得到越来越多人的喜爱.通过两过多月的毕业设计,我对 MSP430 有了初步了解,对内部的硬件资源和自身的汇编语法进行了实验,并开发了一个应用板,并进行了调试.鉴于时间和能力有限,没能对所有的应用一 一实验.

第二章 MSP430 F149 语言介绍

MSP430 是德州公司的新产品,有独特的开发环境和自身语言,下面是我在毕业设计中对 F149 的开发环境熟悉中遇到的一些问题的处理和汇编语言的用法及程序中遇到的问题的体会.

第一节 开发环境及程序下载

1.开发环境:在 EW23 环境下进行编程,汇编,连接,在 C—SPY 环境下进行调试,下载是在连接之后,调试之前,通过计算机的串口下载的.关于环境的操作,可以参考有关资料,其中可能遇到的问题及解决方法有:

(1) .汇编是对源程序而言的,因此必须打开一个源文件才能汇编,而连接是对一个工程文件而言的,连接是对工程文件的所有源代码(包括多个源文件)和数据的定位,因此连接必须打开一个工程文件才能连接.

(2) 连接中必须将库文件的路径改正确,且必须选定 C—SPY 的驱动方式,即在 project 中的 options 的 xlink 的 include 下修改(先选中)xcl 的库路径为 $TOOLKIT_DIR$\icc430\msp430F149A.xcl ,选择 C—SPY 的驱动 drive 为 simulator 或 FLASH EMULATION TOOL ,当没连接 430 片子时可以选 simulator,当连接 430 片子时,选 FLASH EMULATION TOOL 进行在线下载调试.

(3) 由于 430 支持汇编语言和 C 语言两种语言,因此可以在一个工程文件 中同时用两种语言,但建议用汇编语言,因为便于在调试时寻找逻辑和指令的联系及地址的定位正确与否.

(4) 在在线的 C—SPY 的调试中,单步需要将 Control 的 Reatime 前的勾取消才能进行单步测试.

(5) 在线调试时,不能将 58 管脚（复位/非屏蔽中断） 外部变高,否则,会强制退出调试环境.

2.程序下载原理及脱机工作原理:程序的在线调试是通过 JATG 口和 F149 片子的 RST、TCK、TDI、TDO、TMS 引脚按一定的时序串行的传递程序代码和数据的,调试指令的命令传递都是通过这些数据线和控制线传递的,下载时序可参见资料 1,其中的地址 0FFFEH 为复位向量的地址,它是程序遇到非屏蔽中断和程序启动的首要地址,地址中存放的是程序段开始的首地址,因此必须把程序段的首地址标号表示在中断向量中或程序伪指令的开头位置,否则,连接时将会出错,具体的表示方法在下一节中表示.程序的下载和在线调试的电源是通过计算机在 JATG 提供的,不须另外给加电源.

脱机工作时,是将 F149 的电源线上电,此时的复位时序同下载后在线复位的时序一样,只是时钟是通过 F149 内部时钟 DCO 提供的,上电后,程序将复位向量 0FFFE 中的地址装入 PC,PC 开始从程序段的首地址开始执行.脱机工作启动不需要任何操作,只需上电即可,电压要大于 1.8v,一般取 3v 左右,另外,在脱机工作时,可以给 RST 端口加一个低电平脉冲以复位从程序开始重新执行.

第二节 指令介绍

MSP430 有自身 语言,汇编语言也不同于其他类型的单片机,伪指令也是变幻魔测,但又很重要,下面是我毕业设计的一些尝试、出问题的地方.也可参见资料。

1.“ #include ”不能大写。

2.程序段前的伪指令可以套用下列模板,在以后的几章中的程序都采用此模板,只是中间的主程序变化而已:

```
#include  "MSP430x14x.h"   /*把库文件包括进来 ,这个库文件是必须的,其他的库文件视需要而定*/
 RSEG   UDATA0       /*定义数据段一般默认数据段段地址是从 0200H 开始的也可以自己定义数据段开始地址,但必须在 0200H 到 09FFH*/
  DS    0                /*表示数据段从默认的段开始,偏移地址为 0,若为 DS   N,表示数据段的偏移地址从 N 开始,此时的物理地址为(0200+N)H*/
ADINPUT EQU  00200H    /*将 0200H 地址命名为 ADINPUT,此后程序中的地址 0200H 可以用 ADINPUT 表示,便于程序的可读性,注意:标号必须顶格写*/
 A      DW      5H              /*定义 A 字变量的值为 5H,此时将会将 5H 写到数据段的当前偏移地址上,便于后面使用,变量也得顶格写*/
 RSEG   CSTACK                  /*定义堆栈段*/
  DS   0                        /*段偏移值为 0H,物理地址为默认开始地址值*/
 RSEG   CODE                    /*定义代码段 1*/
  DS  0                         /*代码段 1*/
RESET                           /*标号,表示程序段的开始地址,将被写入复位向量中*/

 MOV   #SFE(CSTACK),SP                  /*初始化堆栈指针*/

 MOV   #(WDTHOLD+WDTPW),&WDTCTL         /*停止看门狗定时器*/


 ……                                    (程序段的内容)

COMMON INTVEC                   /*表示中断向量定义*/ /*下面的伪指令都不顶格*/
  ORG    XXX 1                  / *XXX1 表示中断向量表中的具体的中断向量 1*/
  DW     YYY1                   /*YYY 是中断程序入口标号,表示中断程序首地址*/
  ORG    XXX2                   /* XXX2    表示中断向量表中的具体的中断向量 2/
  DW     YYY2                   /*YYY2 是中断程序入口标号,表示中断程序首地址*/
 ORG    RESET_VECTOR            /*复位向量,每个程序中都必须的,可以放在段开始前的伪指令中*/
  DW     RESET                  /*程序开始的地址标号*/
 END                            /*程序结束*/
```

3.几个规定:所有的标号都要顶格写,所有的变量都要顶格写,所有的伪指令和指令都不能顶格写,CALL 调用子程序是在标号前用“ # ”，而其他的转移指令中的标号前不用“ # ”，对外设的寄存器，当程序开始时，许多是复位为零的，如果要置位为 1，可以直接将每一位的名称作立即数写入，例如：指令

MOV #(WDTHOLD+WDTPW),&WDTCTL 就是将 WDTCTL 寄存器中的 WDTHOLD 和 WDTPW 位置位为高，很容易读程序内容。

4．关于几类定义的区别：EQU、=、SET、VAR、ASSIGN 都是给标号变量定义地址值的伪指令，都可以出现在程序中的任何位置，但用法不一样，= 、EQU 是定义一个永久地址标号变量，一旦定义，在程序中的这个标号将固定在定义的地址上，不能改动。而 SET、VAR、ASSIGN 是暂时的地址标号变量，可以在程序中改动，一旦定义了一个标号地址，就可以对这个标号作地址访问，但必须是在数据段。另外，DB 、DW 是定义变量在数据段当前的偏移位置，是作为数据定义的，不是作为地址定义的，例如：

AA DB 2H /*此时在数据段的当前位置写入了 2H 到存储器，以后用 AA 时就是用数据 2H，注：AA 顶格写*/可以在以后的程序中看到这些区别。

其他的指令和伪指令都可以在相关资料上查找到，以上是经常出现的问题，一般核心程序中的指令在语法上都不会有太大的问题，在此不再列举。

第三章 MSP430F149 资源的应用介绍及开发

本章将介绍 MSP430F149 的片上资源的开发和实验程序，并有详细的时序图、波形图和实验结果的数据，当然，只能是部分应用程序。

第一节 中断介绍及存储器段介绍

中断在 MSP430 中得以广泛的应用，它可以快速进入中断程序，之后返回中断前的状态，其时序为：PC 执行程序⟶中断允许置位⟶ SR 中的 GIE 置位⟶EINT（中断开） ⟶ 中断到，中断标志位（IFG）置位⟶从中断向量表中读取中断程序的入口地址，进入中断程序⟶执行中断程序⟶中断允许位复位⟶RETI 中断返回 ⟶回到原来地址。具体应用将会在应用程序中的到应用。有关中断源和中断优先级及中断允许位、中断标志位在参考资料 1 上有详细介绍。

MSP430 单片机的片上存储器共为 64K，表示为图 ：

0H—0FH SFR （特殊功能寄存器 IE、IFG、MEM）	010H—0FFH （8 位外转模块、I/O 端口）	0100H—01FFH （16 位外转模块、TIMER、ADC）	0200H—9FFH RAM 区，数据存储区，可修改访问	0A00H——0FBFH 专用 FLASH 引导	FC0H—10FFH 为信息段 1100H—FFDFH 为程序代码段 FLASH 型 ROM	FFE0H--FFFFH 中断向量地址

对存储器的访问可以用间接寻址，这对于查表处理很方便，在此举一例子： 是对存储段 200H 的 100 个数的读取和操作.

```
      ……
MAIN     MOV   #0200H,R6      /*从 200H 地址开始读出数据到 R5 中,可以加许多对 R5(即数据段的内容)进行操
                               作的 程序*/
         MOV   #100,R4        /*设取 100 个地址单元*/
LOOP1    MOV.W  0(R6),R5      /*间接寻址模式*/
         ADD    #2,R6         /  *是字操作*/
         ;.....               /  *可以加对取出的数的操作*/
         MOV.W  R5,0(R6)      / *操作完后再放回原地址*/
         SUB.B  #1,R4         /*循环 100 次*/
         CMP    #0,R4
         JNZ    LOOP1
 …….
```

实验结果为：可以从 R5 中看到数据存储器从 200H 开始的 100 个数值，在操作完后，可以在 200H 开始存储器中看到操作后的结果满足要求。

第二节 硬件乘法器

硬件乘法器不集成在 CPU 内,是独立于 CPU 运行的,运算时只需将两个操作数放进相应的地址中,就可以直接在结果寄存器中取数据,CPU 可以工作在低功耗模式,如果用间接寻址模式,可以超低工耗的乘法计算大量的表数据,这儿列举一个例子,其他的几种情况类似于此:下面为有符号数(由第一个乘数决定类型)的乘法程序的部分

```
      ……
MOV   #138H,R4        /*乘数 2 的地址为 138H,这儿用间接寻址方式*/
MOV   #-45H,&MPYS     /*装第一个有符号乘数的数值入地址,第一个乘数 MPYS 决定了*/
MOV   #35H,0(R4)      /*装第二个有乘数的数值入地址*/
MOV   RESLO,R5        /*结果低字送入 R5 中取出*/
```

```
MOV   RESHI,R6        /*结果高字送入 R6 中*/
MOV   SUMEXT,R7       /*结果扩展送入 R7 中*/
……
```

实验结果为可以在 R7、R6、R5 中看到-45*35 的结果为 FFFFFFFFF1B7H，结果正确。

硬件乘法器的软件限制可见参考资料 1,建议做乘法之前关掉中断.

第三节 P 口

MSP430F149 有 6 个 8 位的 P 口，其中 P1、P2 口占两个中断向量，共可以接 16 个中断源，还可以直接利用 P 口的输入输出寄存器，直接对外进行通信。因为所有的 P 口都是和其他外设复用的，因此在用端口之前都要用功能选择寄存器选定所用的功能是外设还是 P 口，选定之后还要在方向寄存器中确定是是输出还是输入，我实验了一个程序，前部分是实现中断功能的程序，后部分为中断程序是实现直接用 P 口对外提供一个短脉冲的程序，在我们设计的开发板中，专门利用了 P 口的输入输出功能对外存 24WCXX 和实时时钟芯片 8563 的数据通过的存取 I^2C 总线的读取和写入。还利用了 P 口向电池充电的开启电路。下面是个例子：

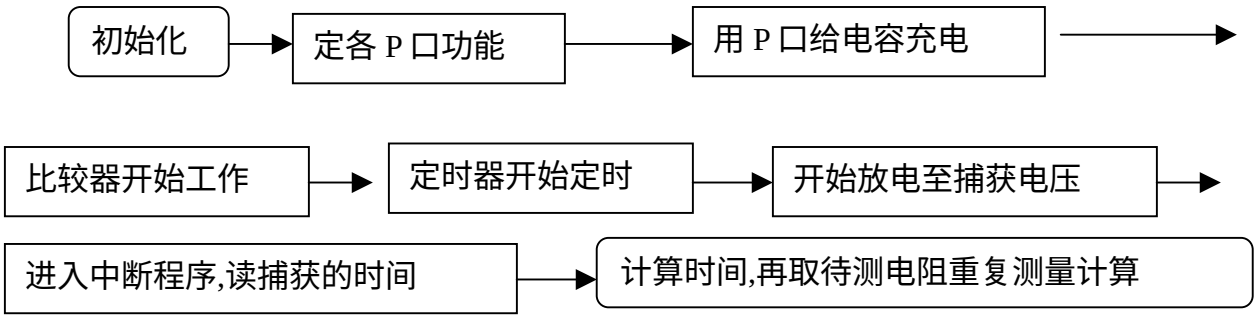
例：利用 P 口的中断功能实验：

```
    ○ ○ ○ ○ ○ ○
MAIN      MOV   #SFE(CSTACK),SP          /*初始化堆栈指针*/
          MOV   #(WDTHOLD+WDTPW),&WDTCTL/*停看门狗定时器*/
LOOP2     BIS   #GIE,SR                  /*普通中断允许*/
          EINT                           /*开中断*/
          MOV.B #000H,&P1DIR             /*定义 P1 口为输入方向*/
          MOV.B #000H,&P1SEL             /*定义 P1 口为 P 端口功能*/
          MOV.B #002H,&P1IE              /*P1.1 口为中断允许*/
          MOV.B #000H,&P1IES             /*定义 P1.1 口为上升沿产生中断*/
          JMP   LOOP2                    /*循环等待中断*/
     /*下面为中断程序*/
LOOP1     MOV.B #001H,&P1DIR             /*定义 P1.0 口为输出口*/
          MOV.B #001H,&P1OUT             /*定义 P1.0 口输出的为高电平,发光二极管灯亮*/
          MOV.B #000H,&P1IE              /*返回中断前的 PC 及其他状态*/
          MOV.B #000H,&P1OUT             /*将 P1.0 口置低,发光二极管灯灭*/
          RETI                           /*中断返回*/
          COMMON INTVEC                  /*列中断向量表*/
          ORG  PORT1_VECTOR
               DW   LOOP1                /*中断向量的入口地址为 LOOP1*/
          END
```

实验结果为：在运行中，当给 P1.1 口一个高电平时,PC 装入中断程序的地址 LOOP1,进入中断程序段,P.0 口被置高,此时发光二极管灯亮,两个指令周期之后灯灭,此后又返回中断前的地址开始执行,等待下一次中断的到来.

第四节 定时器及数模转换

MSP430 中有两个 16 位定时器，还可以利用看门狗定时器。由于定时器的是 16 位的，则可以在秒数量级上定时，且具有 2 个中断向量，便于处理各种定时中断。定时器的应用在 F149 中具有举足轻重的作用，可以利用 MSP430F149 中的定时器的比较模式产生 PWM（数字脉冲调制）波形，再经过低通滤波器产生任意函数的波形，也就是说，可以通过定时器的比较模式实现数模转换功能。另外，定时器还具有捕获模式，我们可以通过定时器的捕获功能实现各种测量，比如脉冲宽度测量，如果和比较器结合，还可以测量电阻、电容、电压、电流、温度等，可以这样说，只要能通过传感转换为时间长度的，都可以通过定时器的捕获定时功能实现值的测量。在开发板中，利用定时器，我们设计了一个 PWM 滤波输出的函数发生器。另外，我们还利用定时器的捕获功能和比较器的比较功能测电阻和电容，原理可以参见参考资料 1 中比较器的应用章节。下面是比较器测电阻的实验程序和时序： 程序和设计流图为 ：



……

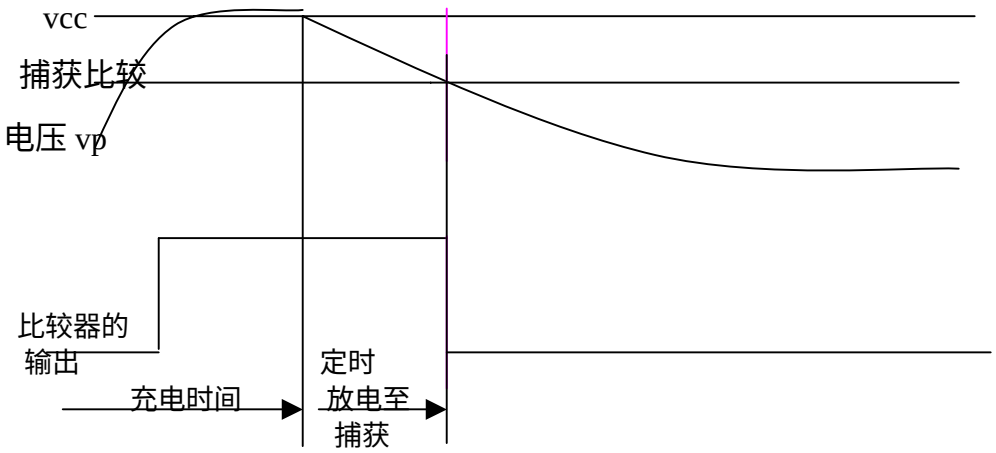
Reset

```
MOV   #SFE(CSTACK),SP            /*初始化堆栈指针*/
MOV   #(WDTHOLD+WDTPW),&WDTCTL/*停看门狗定时器*/
MOV   #GIE,SR                    /*一般中断允许*/
MOV.B #004H,&P1SEL               /*定义定时器 A 的 A1 作捕获输入*/
MOV.B #000H,&P1DIR               /*定义端口方向为输入型*/
MOV   #0FFFFH,&CCR0              /*规定定时器的最大计数值为 FFFFH*/
MOV   #000H,&CCR1              /*给捕获初始值为 0*/
MOV.B  #004H,&P2DIR            /*比较器的两个比较口为输入,输出口为输出型*/
MOV.B  #01CH,&P2SEL            /*定义了端口为比较器功能*/
MOV.B  #0FFH,&P3DIR            /*定义 P3 口输出一个高电平给电容充电*/
```

```
        MOV.B  #000H,&P3SEL          /*选择 P 口的功能*/
        MOV.B  #0FFH,&P3OUT          /*输出给电容充电*/
        EINT                         /*开中断*/
LOOP1
        MOV.B  #00CH,CACTL1          /*确定比较器的输入 0 口为外参考电压,这实验中为电容上的电压*/
        MOV.B  #00FH,CACTL2          /*确定比较器的输入 1 口为外参考电压,这实验中为捕获时刻电压,由
                                       外电源提供,可变的 ,根据电阻和电容而定*/
        MOV   #08930H,&CCTL1         /*定时器 A 的 A1 口的 CCR1 为捕获寄存器*/
        MOV   #002D2H,&TACTL         /*写控制寄存器,定时器开始计数*/
        MOV.B  #000H,&P3DIR          /*电容放电,等待放电电容上的电压降到捕获电压发生中断,此时的
                                     CCR1 中值为放电时间比例值*/
       JMP    LOOP1
CCR     BIC   #0FF0FH,&TACTL        /*停定时器*/
        MOV  &CCR1,R5                    /*从 R5 中看定时器的值,还可以送到 I/O 口上*/
        JMP  CCR                    /*程序结束*/
    COMMON INTVEC
         ORG    TIMERA1_VECTOR
          DW    CCR                      /*捕获中断向量*/
         ORG    RESET_VECTOR
          DW    Reset
    END
```

实验结果跟参考资料的充放电波形一样,波形为:

电容两端电压波形:



实验数据为

参数 / 电阻	VCC	VCP	电容	CCR1.1	CCR1.2	CCR1.3	平均定时时间	电阻测量值	电阻实际值
参考电阻	2.5	2.0	628 号	45BH	459H	45CH	45BH	0.9K	0.9K

待测电阻	2.5	2.0	628号	466H	466H	463H	465H	1K	1.01K

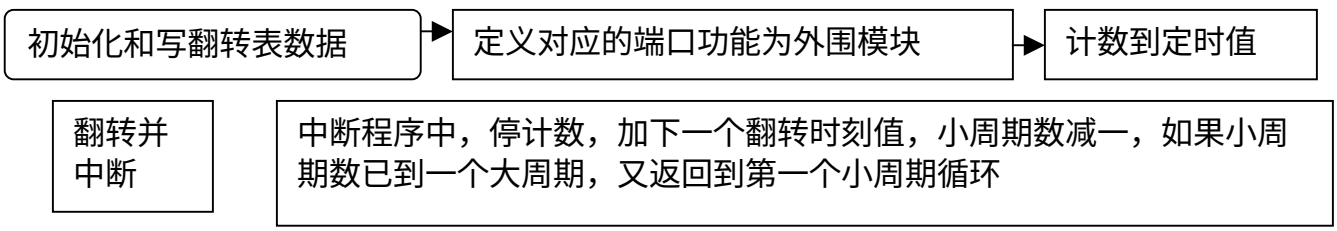
计算电阻公式为:R 测=R 参*(N 测/N 参) (其中,N 代表捕获的计数值)

本次实验的经验:电容必须选择得当,若太大可能定时器溢出中断而不是捕获中断,太小,则会为各电容的放电时间差不多,误差太大.捕获电压也必须得当,太大,可能定时时间太小,误差太大,太小,放电时间太长,可能溢出中断而不是捕获中断.这实际是一个使用范围的问题,由于 DCO 的频率太高,定时器的计数太快,如果定时器的频率低,采用大电容,则使用范围会更大一些,精度更高一些.

另外,可以用比较器和定时器的捕获用同样的原理测电容及其他的可以转换为时间的传感问题,这在实际应用中有更广泛的用途.

PWM测试 利用定时器的比较模式和输出的 PWM 形式,我们可以作出数模转换的模型和程序,这样经过低通滤波可以产生各种函数发生器.为此,我们做了一个 PWM 波的实验,原理及流图和时序及程序为:

原理为:利用输出模式的翻转特性和连续模式的 PWM 波形输出，通过 CCR0 加数据存储器 RAM 的中相互交叉“ 0 ”电平和“ 1 ”的时间间隔，成对的两个寄存器定义了占空比,而各对的和（小周期）是定值。当计数器的计数值到达 CCR0 翻转，且产生中断，转入中断程序，在中断程序中，我们给 CCR0 加上下次翻转的时间，即下次翻转时的计数长度从数据存储器中取出加到上次翻转时刻的计数值中，当返回中断后，计数器继续计数，到下次翻转和中断时，又循环继续进行。这样，就输出了占空比不断变化而又呈一种趋势的变化，经过低通滤波，即电容的充放电形成一种阶梯状的变化趋势，当计数小周期很小时，就可以得到近似的一条模拟曲线，从而实现了数模转换或函数发生器，由于小周期是任意的但必须大于 2 倍中断程序时间，则可以实现任意占空比的小周期和任意的小周期长度，又由于有多少个小周期组成一个大周期也是自由的，完全由实际需要来定，则给用户带来了很大的灵活性。下面是程序流图，由于这个 程序实验要用到对数据段表的读操作和间接寻址且用到中断向量，因此在此列出了程序的全部清单，以更完整：流程图为：



→ 又从原计数值重新开始计数，中断返回，循环执行

程序清单为：

```
#include  "msp430x14x.h"
    RSEG   UDATA0
    DW    450,50,350,150,250,250,150,350,50,450/*间隔数据表，开始地址为 200 H，数据又需要定*/
    RSEG   CSTACK
    DS   0
    RSEG   CODE
    DS  0
Rese    MOV   #SFE(CSTACK),SP          /*初始化堆栈指针*/
        MOV   #(WDTHOLD+WDTPW),&WDTCTL /*停看门狗定时器*/
        MOV.B #0FFh,&P1SEL       /*选择外部定时器功能*/
        MOV.B #0FFH,&P1DIR       /*确定方向为输出*/
        MOV   #030H,&CCR0        /*给 CCR0 一个初始值，不小于两个指令周 期的计数值*/
        BIS     #GIE,SR          /*开一般中断允许位*/
        MOV   #0200H,R6          /*将 R6 定义到数据表段开始地址*/
        MOV   #10,R4             /*取 10 个地址单元，即九个小周期*/
        MOV   #0090H,&CCTL0    /*选 CCR0 作为比较寄存器，定义输出模式为 4，且中断允许*/
        MOV   #002E0H,&TACTL   /*写控制寄存器，参数为一分频，比较模式，连续计数方式，不溢出中
断，开始计数*/
TA0   EINT                     /*开中断*/
      JMP   TA0                /*等待翻转时刻到和等待中断到，即 TAR=CCR0*/
CMPS BIC   #0FFCFH,&TACTL   /*中断程序到，停计数值，处理中断*/
        ADD   0(R6),&CCR0        /*加下一个翻转到来的时间值，间接寻址方式*/
        ADD   #2,R6              /*是字操作，加 2 是将 R6 指向下一个地址*/
        SUB.B  #1,R4             /*小周期数量减一*/
        JNZ    LOOP1             /*大周期没完，循环*/
        MOV   #0200H,R6          /*大周期完，重新开始一个大周期*/
        MOV   #10,R4
LOOP1   MOV   #002E0H,&TACTL  /*重新开始计数*/
        RETI                     /*中断返回*/
          COMMON INTVEC
            ORG    TIMERA0_VECTOR   /*定时器 A 的 0 中断向量表*/
            DW    CMPS
            ORG    RESET_VECTOR
            DW     Reset
     END
```

下面是程序中需要说明的几点问题：

1． 在中断程序中，不能在没回中断之前就用转移指令将程序跳出中断，否则，堆栈占用的空间会越来越大，数据段会出错。主要是程序段的 LOOP1 必须在中断程序里。即 CMPS … … 中断程序开始

```
        SUB.B  #1,R4                /*小周期数量减一*/
        JNZ    LOOP1                /*大周期没完，循环*/
        MOV    #0200H,R6            /*大周期完，重新开始一个大周期*/
        MOV    #10,R4
LOOP1   MOV    #002E0H,&TACTL  /*重新开始计数*/
        … …
        RETI
```

2．在程序中，只用 CCR0 而不用 CCR1 和 CCR2 的原因是 CCR0 的中断优先级高，且返回时不须软件将中断标志位清出掉，而是自动复位的，而 CCR1 和 CCR2 的中断标志须软件复位，否则中断变的不定。

3．由于不须要在计数到 0 时中断，因此将溢出中断禁止，而将 CCR0 的中断允许。

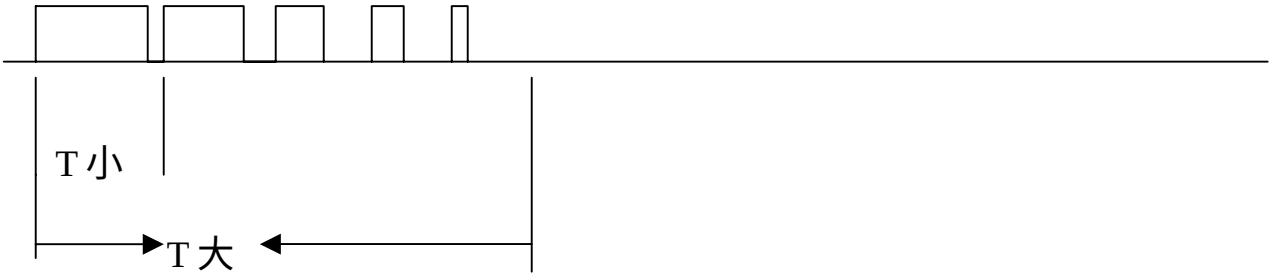
4．数据段的数据个数一般要达到 256 个才能通过 8 路 AD 转换，由于篇幅有限，只列出了 10 个，这由实际需要而定 。 如果需要的滤波形是对称的，则数据段的数据为对称就可以了。

5．定时器 A 有两个中断向量，如果要同时用 CCR0 和 CCR1 或 CCR2，需要写不同的中断程序，这一点尤其注意。

实验结果为：

PWM 波形频率：MAX：2KHZ，计数频率：8MHZ，误差：0.5%

波形图示意为:



第五节　时钟模块

MSP430F149 的时钟可以自由选择,它包括一个内部 DCO 时钟和另外两个外部时钟,内部时钟的参数见参考资料 1,其中最高可达到 1042KHZ;外部可以接两个时钟,一个可接钟表晶振或标准晶振,另一个接最高时钟频率为 8MHZ 的晶振,8M 是单片机的最高工作频率,对于晶振的选择,在参考资料一上介绍的很清楚,在此不在重复,对基础时钟的控制,只需要对相应的控制寄存器写入相应的控制位就可以产生需要的时钟,还可以从相应的端口测的时钟频率,我们做了一个实验,是控制

内部时钟的,可以从 149 的端口上测的相应的频率,只要开启时钟频率之后,时钟就继续存在到写入停止为止.

下面是主程序,由于简单,不用程序流图.

```
MAIN    MOV   #SFE(CSTACK),SP                    / *初始化堆栈指针*/
        MOV   #(WDTHOLD+WDTPW),&WDTCTL/*停看门狗寄存器*/
        BIS.B #010H,&P5DIR                       /*定义方向为输出方向*/
        BIS.B #010H,&P5SEL                       /*选择为外部模块功能*/
        BIS.B #000H,&BCSCTL2                     /*选择为 1 分频,DCO 为 MCLK 的输入时钟*/
        …….                                      /*可以在 48 管脚看到时钟频率*/
```

实验结果为时钟频率 1000KHZ,占空比为 1:1,如果调整 BCSCTL2 的控制位,可以看到频率的变化,在我们做的开发板中,要利用这个频率升压,这在后面将会用到.

第六节 USART 通信模块

通用串行同步异步通信模块是为了使 MSP430F149 多机通信用的,通过 USART 口连接 RS202 和 RS485 的驱动芯片可以实现单片机与计算机及其他的工作电平的匹配串行通信,由于 MSP430F149 具有两个通信口,因此可以分别用于 RS202 和 RS485 的串行通信.MSP430 有同步和异步两种方式,每一种方式都有独立的帧格式和控制寄存器,只需要按照需要和帧格式写入相应的寄存器就可以实现多机通信.

由于 MSP430 的波特率产生比较自由,因此异步通信模式用的比较多,在毕业设计中,我们只实验了异步通信模式,在异步通信模式中,MSP430 的波特率的产生有很独特的方式,可以实现多种波特率的产生,可以克服其他单片机的波特率受限的缺点.另外,在异步模式中,又根据需要分为线路空闲多机模式和地址位多机模式,如果只是两机通信,线路空闲比较多,用线路空闲多机模式比较好,在开发板中有一个测试程序是实现通过 RS202 与计算机超级终端串行口相连的测试程序,在此,不用多说,由于 MSP430 的波特率发生器比较特别,在此,我们着重讨论一下波特率发生器.

波特率发生器是用波特率选择寄存器和调整控制寄存器来产生串行数据位定时. 波特率 =BRCLK/(UBR+(M7+M6+M5+M4+M3+M2+M1+M0)),其中 BRCLK 为晶振频率,UBR 为分频因子的整数,即晶振频率除以波特率的整数部分,而 M7,M6,M5,M4,M3,M2,M1,M0 分别为调整位,是分别写在 UMCTL 中的,如果置位,则对应的时序时间只能波特率分频器的输入时钟扩展一个时钟周期,每接受或发送一位,在调整控制寄存器的下一位被用来决定当前位的定时时间.协议的第一位的

定时由 URB 加上 M0 决定,下一位由 UBR 加上 M1 决定,以后类推.而调整位取“ 0 ”还是“ 1 ”，取决于当前的分频因子与需要的分频因子的差距，如果大于 0.5 取“ 1 ”，如果小于 0.5 取“ 0 ”，具体实例可见参考资料 1。

第七节　　比较器模块

比较器的应用在 MSP430 中很广,可以做为可转换为电压的量的测量,这在参考资料 1 上有很详细的说明,如果加上定时器的捕获功能,比较器的用途会更广,由于比较器的应用在定时器一章已有实验证明,在此不在多述,但有几点必须说明.

1.　比较器属于硬件型的,虽然很准确,但由于有软件的控制,造成的时间误差可能很大.因此存在一段时间的振荡,这造成测量的误差大，不能很精确.

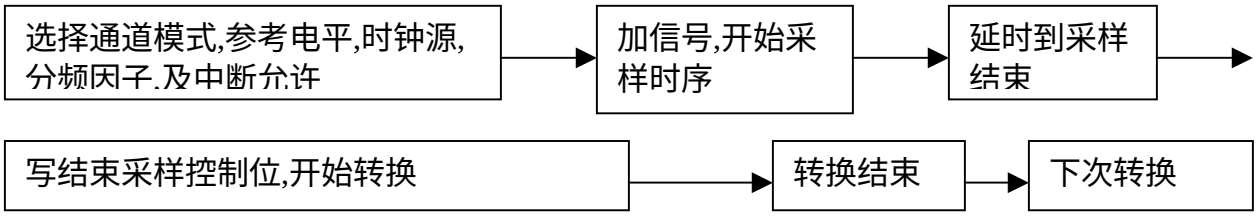
2.　比较器的参考电平很方便,可以都自由加,但不能超过片子的最高电压 3.3V ,否则不能正常工作.

比较器的应用还很多,可以很放心的用,在我们的开发板中,有比较器的输入端口,还加了电阻和稳压和嵌位的二极管对在高压和负压时的芯片保护.

第八节　模数转换模块

MSP430F149 单片机中集成了 14 路 12 位 A/D 转换,其中 8 路属于外部的信号转换,3 路是对内部参考电压的检测转换,1 路是接温控的传感电压转换,每一路转换都有一个可控制的转换存储器,而且,参考电平和时钟源都是可选择的,可以外部提供的.这给使用上带来了很大的灵活性.原理

上不同于一般积分和逐次比较等 A/D 转换原理,它的输入信号是加在 A/D 的电容网络上的,通过电容的充电来采样信号进行 A/D 转换的.其时序可以归纳为 :



A/D 转换的时序和具体的一些注意事项和参数可参见参考资料 1,但有几点必须注意的地方:

1.由于 MSP430F149 是采用加载信号到电容上充电的采样,因此必须要给一定的采样时间以能到达一定的精度和时间的不溢出,否则会出现时间溢出的中断.据测定其采样开始之后需要 13 个 ADC12CLK 周期延时.在实验时是采用的单步才能比较精确的测量，在全速时需要延时才能测量，否则采样结果为 0。

2.在采样结束和转换的开始需要一个控制过程,这就是将 ADC12CTL0 的 ENC 和 ADC12SC 同时置“ 1 ”，则表明采样结束和转换开始，在我们的测试中，是将 ADC12CTL0 的控制位重复了一次以达到开始转换。

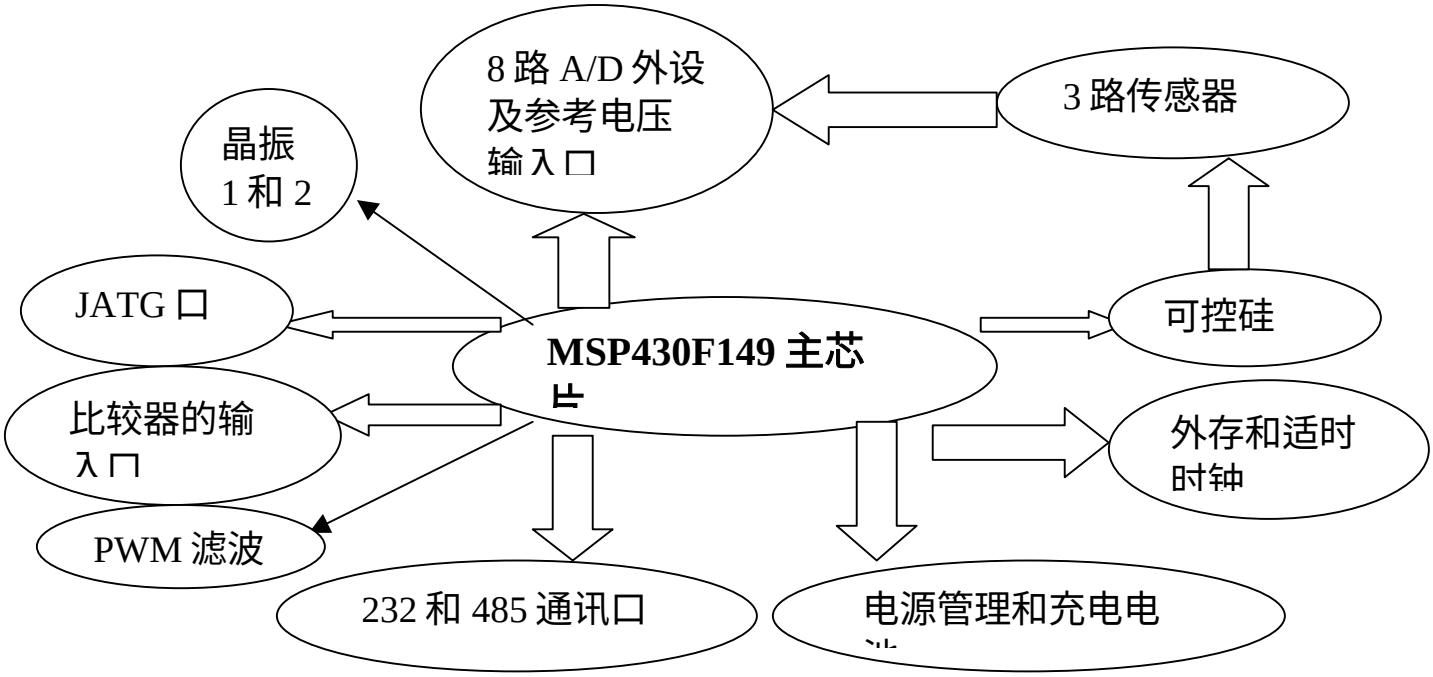
3. 用外部参考电压时，转换公式为 NADC=4095*（Vin—Vr-）/（Vr+—Vr-）。这于参考资料 1 上有不同。

4. 由于低三位是电阻性的，因此精度上需要多次测量取平均值。

5. 如果采用外参考电压，则不能认为悬空为 0v，而必须要加一个电压，即使是 0v 也必须要加地，否则不能转换。 具体的 A/D 采样程序和结果在 PCB 测试中有比较详细的结果。

第四章 MSP430F149 开发板的介绍及测试

在这次毕业设计中，我们的主要任务是设计一个芯片的开发板，以能够供以后使用，考虑到 F149 的资源，我们的设计外围模块功能图为：（电路见附）



下面将分模块介绍各模块原理（标示见原理图）及调试程序和结果：

第一节 模数转换模块

在单片机的模数应用中，经常会设计到转换为检测电流的情况，而且还会遇到起始电流不为 0mA 而为 4mA 的情况，此时需要将电流转换为电压和加电压到参考电压负端的电路，在原理图中，rz.x 和 r11x 构成分压并将电流转换为电压进入单片机采样转换,其中，d11x 中每两个组成一对，其中的一个为精密电阻以起补偿用，使 d11x 和 rz.x 组成的分压精度更高。d11x 构成过压保护和低压保护,克服一些意外毛刺等.对分压的电阻选定可以扩大转换范围,这由实际需要定,我们加了一路 0.5 分压的电路,可采样 6v 的电压.当起始电流为 0mA 时(即电流为 0mA 时转换为 0),

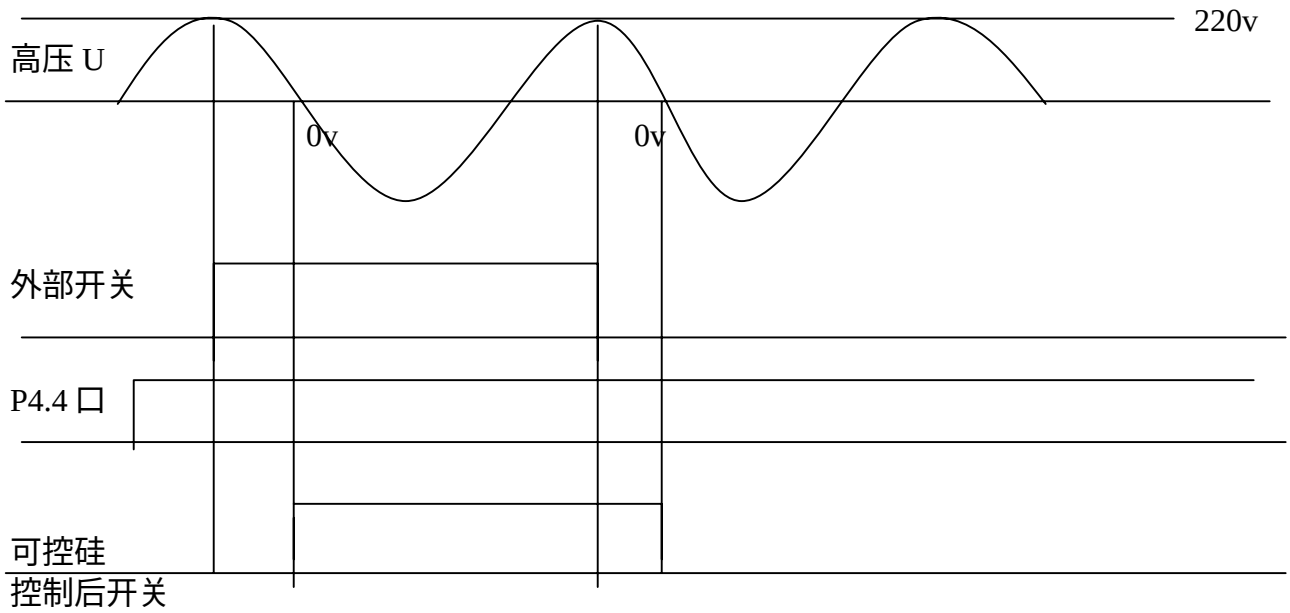
让 P1.4 口输出为高,此时三极管 Qve1 不导通,负参考电压为 0v,此时的 0mA 转换为 0,当起始电流为 4mA 时(即电流为 4mA 时转换为 0),让 P1.4 口输出为低,此时三极管 Qve1 导通,负参考电压为 Rve1 和 Rve2 分压,电阻 Rve1 和 Rve2 的值由实验得出为 13:2,负参考电压为 0.35v,测试结果和调试程序可见刘亚娇的报告.但有几点问题为:

1. 分压电阻的阻值不能太大,一般数量级为 100 欧级,因为单片机内阻不大,一般为十千欧级,且线性度不太好.但电阻太小造成电流较大,这是使用 A/D 的局限.
2. 速度不能太快否则不能转换,负参考电压不能悬空.

比较器的输入原理与 A/D 的输入原理一样,也加了分压电阻和二极管的嵌位保护电路,可以电流比较又可以电压比较,测试结果可见第三章的有关节.

第二节 传感器模块

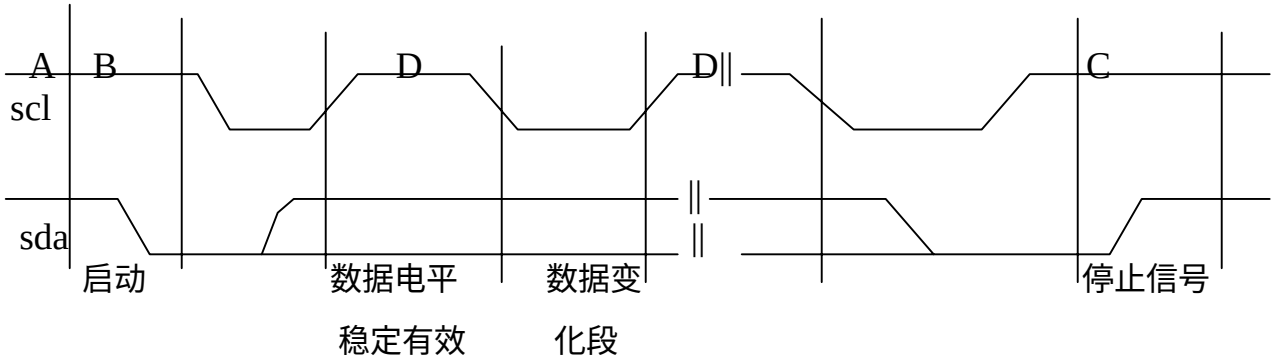
在开发板中,设计了 3 路传感器,将 220v 和更高的电压转换为可用 A/D 检测的低压,然后通过 A/D 采样,判断高压信号的稳定度后,再将 P1.4 口置高,此时二极管 DT1 和三极管 T1 导通,可控硅 TRIAC 工作,在外部高电压的工频正弦电压达到 0v 附近时将 JK1 上的 3、4 导通，3、4 外部接单片机可控负载，当断电时，利用可控硅的特性，在断开后高频正弦电压达到 0v 时才断开，这就避免了高压工作时的断电的辐射造成的环境和人体的伤害，原理图中 TR 起可控硅的电流放大作用。其测试结果的时序图可以表示为：



结果可以看到,经可控硅后,电压的开启可关闭不会发生在高压部分,这对环境和人体的辐射达到了尽可能的小.

第三节 外存和实时时钟模块

考虑到单片机有时需要实时时钟和外存,因此本设计加了一个 8563 实时时钟芯片和 24WCxx 的片外存储,采用 I^2C 总线结构传送数据和时钟,通过 P4.6 传送 SCL 时钟信号,P4.5 传送地址、控制、应答和数据信号。8563 接一个 32767HZ 的晶振产生实时时钟，R32、R31 是提拉电阻，以从 8563 和 24WCxx 中输出数据。由于本设计中只有一个片外存储，因此其芯片地址为 1010000，8563 的读的芯片地址为 1010011，写地址为 1010010，关于 I^2C 总线的帧结构为：

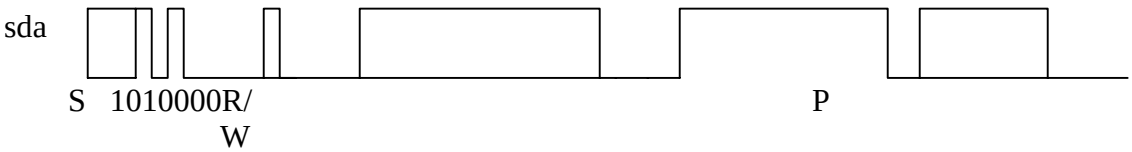


A：总线非忙阶段　　B：启动数据阶段　　C：停止数据段

D：数据有效段

其中测试时产生的字节访问帧结构为：

主机　启动　片选地址　读/写　应答　存储单元地址　应答　数据字节　应答　停止



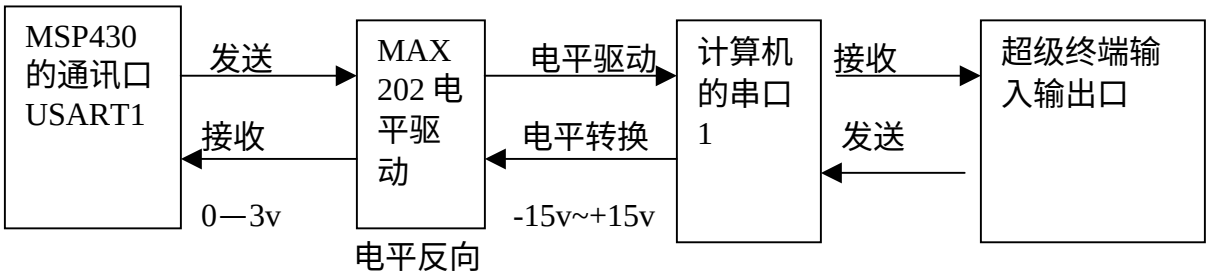
调试程序流图和帧结构的顺序一样，在这儿就不在重复，调试程序见附 eerom.s43。

调试结果为：往外存中的 10000000 地址写入 10101010 字节数据后读出仍然为 10101010，表明对外存的读写正确。对 8563 的读写一样。对 I^2C 总线的使用最重要的是时序的正确。这是编程最重要的一点。

另外，对读信号时，要注意是启动后，先写片选地址，再送写信号，等应答后送要读的地址，然后才是启动，之后是送片选地址，再送读信号，等应答之后才是地址上的数据。这与写时序是不一样的。

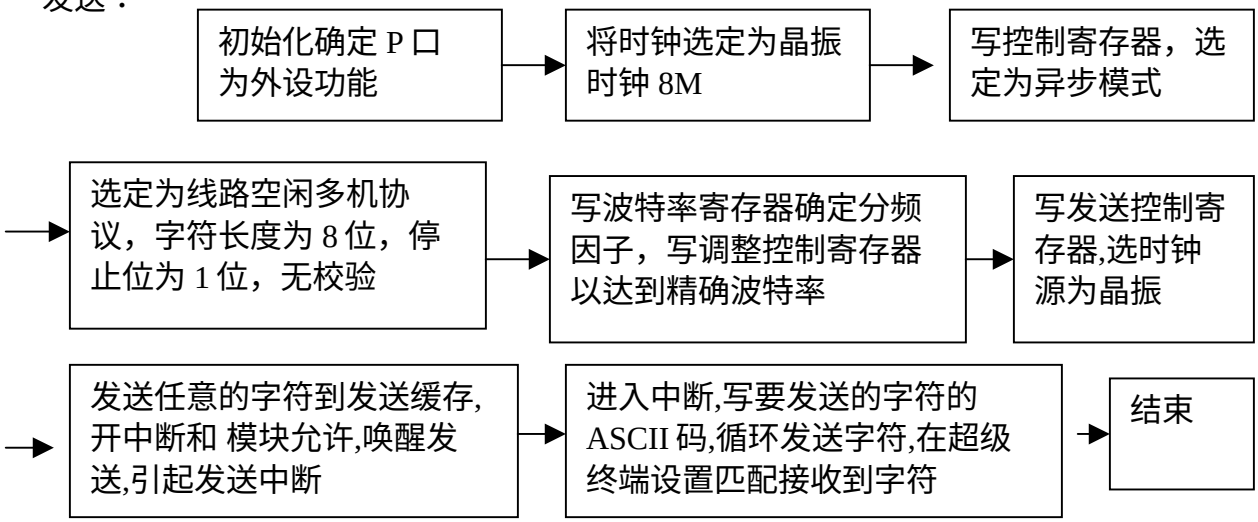
第四节 485 和 232 通讯模块

单片机通讯时需要转换电平，这就要 485 和 232 电平驱动，在开发板中，我们用了 MAX202E 和 MAX485 芯片作为驱动，MAX202 芯片可以实现短距离的通信，是属于单端驱动方式，能将-15v~+15v 的电压转换为 0≈5v 的电压,速率能达到 20K,输出电流可达到 500mA,实现了 TTL—EIA 电平的转换,由于 MAX202 芯片是全双工的,因此其使能引脚是全使能的,又由于 MAX202 芯片输出是 5v 的电压,而 430 芯片是 3v 电压,因此需要一个由 RM3 和 RM4 组成的分压电路,由于受 430 内阻和 MAX202 输出电流的影响,当 RM3 和 RM4 分别取 2 千欧和 3 千欧时分压之后的电压刚好能达到的电压 3v,因此取 RM3 和 RM4 为 2 千欧和 3 千欧.注:RS—232 电平规定为-3v— -15v 为“ 1 ”，+3v—+15v 为“ 0 ”。而 RS—485 电平属于平衡电平，是通过两路的比较决定正负的，RU4 起连接两路的作用，但由于是半双工电路，因此需要使能端口，又由于 启动电平为大于 5v，因此需要加 CM1 和 RC7 的微分电路产生一个比较大的驱动使能信号，当要使能时，P1。0 口发出一个“ 1 ”电平，则 CM1 和 RC7 的微分使输出到 MAX485 上的电平为一个尖峰信号，从而驱动。RM1，RM2 的分压电路和 RM3，RM4 的分压原理一样。MAX232 和 MAX485 分别接到 430 的 USART1 和 USART0 上，实现了多机通信。在调试中，我们是将 430 和计算机的超级终端相连实现的测试，原理为：



调试程序流图为：

发送：



调试程序见附 USART.S43,调试结果为: (时钟为 8M 的条件下)

	Ubr11	Ubr01	Umctl1	实际分频(十进制)	发送波特率	接收波特率	偏差	发送 ASCII	接收字符
1	003H	043H	000H	835	9580.838	9600	0.2%	31H	‘1’
2	00DH	005H	005H	3333.33	2399.94	2400	0.01%	32H	‘2’
3	001H	0A1H	003H	416.67	19199.48	19200	0.01%	33H	‘3’

其中,发送波特率=(8M/实际分频),偏差=(发送波特率)/(接收波特率)

调试结果可以看出发送的正确性,也可以看出用 430 可以达到的波特率的任意性,很灵活实用.但调试中的注意为:

1. 发送的字符最好为熟悉是字符,否则可能认为不正确而实际是正确的,开始我们发送的是 FFH 字符,收到的是一个符号,我们还以为发错了,而当改为 31H 时,收到 1 才知道发送正确.
2. 在超级终端需要将流量控制改为“ 无 ”.

接收流图和发送流图差不多,只是将接收控制字改动即可,在此不在重复,接收程序见附 USAR.S43,接收结果与设想的完全吻合.但要注意以下几点:

1. 只有当波特率合适时才可能发生中断请求.
2. 由于接收控制寄存器中没有时钟源的选定,因此需要在发送控制寄存器中确定时钟源.

第五节 电源管理模块及晶振模块

原理:电源是 MSP430 的很重要的部分,因为它要实现低工耗,就需要低的工作电压,为 1.8v 到 3.3v,我们一般选定为 3v,而一般的直流电压为 7.8v,电路图中的网络标号为 VIN 表示输入电压 7.8v,二极管 Dv2 为一个稳压二极管.当 7,8v 的电压输入后,需要转换 7.8v 到 3v 的电源管理,在我们的设计中,用了一个 L4940V5(电路图中为 vt1)的片子实现 7.8v 到 5v 的转换,再用了一个和 MSP430 配套的电源管理芯片 AT3221(电路图中为 vt3.3)实现从 5v 到 3v 的转换,供单片机工作的电压.由于考虑到单片机工作时的突然断电和单片机向传感器和其他外围模块提供 12v 或 24v 的电压,因此需要一个蓄电池和升压电路,当突然断电和停电后,蓄电池放电供单片机工作,在设计中,我们通过一个 5v 的电磁继电器实现对蓄电池的充放电.原理为:当上电时,P1.2 口输出高电平,三极管 Qv5 导通,继电器的 6 脚为低电平,此时继电器的 5 脚为高电平,继电器的 3、4 脚导通，电池充电，(继电器的 1 和 2 脚为常开),当断电后,继电器的 5 脚变低,继电器的 1 和 2 脚通,电池通过 1 和 2 脚放电,提供给 7.8v 电源,此时 Dv2 不导通,不会出现电的泄露,这样就避免了断电的情况.升压原理为:单片机提供 mclk 的晶振频率,推动三极管 Qv3 和 Qv4 导通和断开,提供给电感 L1 一个高频高电流(Qv3 和 Qv4 的两级放大),电感通过电磁转换实现电压的升高,再通过 DV6 的稳压,使插件 powout(CON2)
提供 12v 或 24v 的电压.

测试结果为: L4940V5 输入:7.8v,输出:5v

AT3221 输入:5v, 输出:3v

继电器 有电时充电:3、4 导通

没电时放电:1、2 导通

另外，考虑到有时需要直接从变压器提供电源，因此，电路设计中增加了一个整流模块，即通过 powin（CON4）输入从外部变压器输出的电压，经过整流桥 dd1 整流提供 7.8v 的输入电压,还通过整流桥 dd2 整流后经过 vt2 的 24v 的片子向外部模块提供 24v 或 12v 的电压,经测试基本满足条件.

外部晶振是单片机实现高速的条件,在 MSP430 的单片机中,有两个外部晶振可用,其中 LFXT1 可以接低频晶振也可接高频晶振,而 XT2 只能接高频晶振,因此,我们将 LFXT1 接 32767HZ 的晶振,XT2 接 8M 的晶振,由于单片机内部已有电容,因此在外部不需要再接电容,只需要接上晶振

就可以起振.测试时,我们将 MCLK 和 SMCLK 都定义为 XT2 晶振作为时钟源,测试结果为输出了 8M 的频率,在其他程序中,只需要将主时钟源选定为外部晶振 XT2,程序的时钟就变为了 8M,测试程序为: (其他程序中只需要写入下面程序就开始用外部晶振)

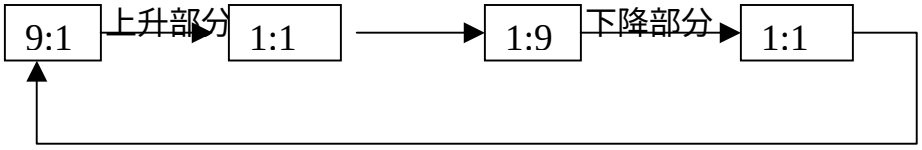
```
    ……
    BIS.B #030H,&P5DIR       /*将 mclk 和 smclk 定义为输出*/
    BIS.B #030H,&P5SEL       /*选定引脚为外部 mclk 和 smclk 输出*/
    BIC   #OSCOFF,SR         /*启动晶振*/
    PUSH  #050H
LOOP DEC   0(SP)             /*将时钟由 DCO 到 XT2 的同步延时*/
    JNZ   LOOP
    MOV.B #088H,&BCSCTL2  /*选定系统主时钟为 XT2 的 8M 晶振*/
    ……
```

第六节 PWM 波形滤波

在第三章中,我们以将原理和程序流图表述的很明白,在此不再重复,在测试中,我们测试了三角波的产生情况,其中的时钟用 8M 晶振,计数翻转的数据表为:

```
RSEG   UDATA0
    DW     450,50,400,100,350,150,300,200,250,250,200,300,150,350,100,400,50,450
    DW     150,350,100,400,150,350,200,300,250,250,300,200,350,150,400,100,450,50
```

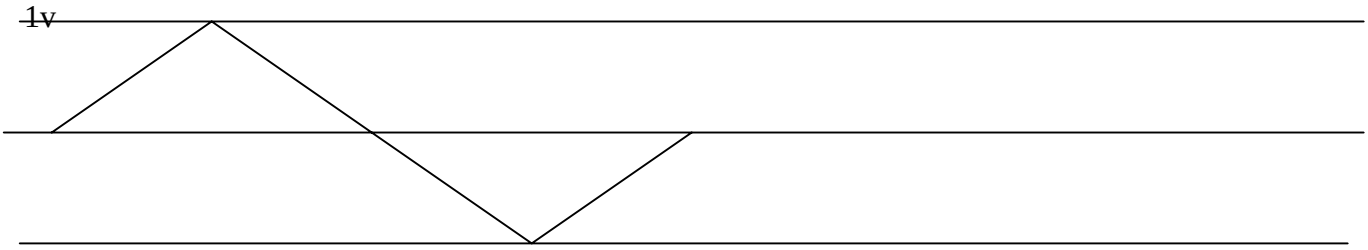
一共 36 个数,36 个点一个周期,占空比为:从 9:1 到 1:1 再到 1:9 再到 1:1



测试后滤波电路中电阻和电容及输出为:

	时钟	滤波电阻	滤波电容	波形幅度	频率
1	8M	247K	4.55NF	1v,平滑	800HZ
2	8M	220K	4.55NF	1.3v,有少量毛刺	800HZ
3	8M	200K	4.55NF	1.4v,毛刺较明显	800HZ
4	DCO	247K	4.55NF	1v,毛刺明显	800HZ
5	DCO	220K	4.55NF	1.3v,毛刺明显	800HZ

实验波形示意图为(以第一组数据产生波形为准,其中有毛刺,表明充放电过程):



从实验结果看出,如果滤波做的更好,可以产生很漂亮的波形.而且频率可以达到 800Hz 左右,用途已很大.

另外,在测试中,我们还实验了单片机的超低功耗的性能,我们用太阳能电源给单片机供电,单片机仍然能很顺利的工作,而且在下午 5:30 都还可以,实现了超低功耗.为了测试电路中对强压控制和抗干扰性,我们利用可控硅的部分进行了实验,将 JK1 的 3、4 端口串强压线包和铁钻，用 P1 口的 4 口提供一开一断的电压，控制可控硅的关闭，实现了线包的一吸一放，起到了很好的控制强压的作用，当再串上铁钻时，控制了铁钻的开闭，即使将活动的铁钻放到片子周围制造干扰，单片机仍然照常工作，实验了单片机的高抗干扰性。

参考资料:

1.《MSP430 系列 FLASH 型超低功耗 16 位单片机》 胡大可编

——北京航空航天大学出版社

2．《单片机外围器件使用手册存储器分册》 窦振中编

——北京航空航天大学出版社

附程序：

1.USART.s43

```
    #include  "MSP430x14x.h"
          RSEG   UDATA0                          /*定义数据段*/
          DS   0
          RSEG   CSTACK                          /*定义堆栈段*/
          DS   0
          RSEG   CODE                            /*定义代码段 1*/
          DS  0
RESET     MOV    #SFE(CSTACK),SP                 /*初始化堆栈指针*/
          MOV    #(WDTHOLD+WDTPW),&WDTCTL        /*停止看门狗定时器*/
          BIS.B    #030H,&P5DIR                  /*初始化端口和选择时钟源为 8M*/
          BIS.B    #030H,&P5SEL
          BIC      #OSCOFF,SR
          PUSH   #050H
LOOP      DEC      0(SP)                         /*延时实现时钟转换同步*/
          JNZ      LOOP
          BIC.B #OFIFG,&IFG1
          MOV.B    #088H,&BCSCTL2
          MOV.B   #040H,&P3DIR
          MOV.B   #0FFH,&P3SEL
          EINT                                   /*开中断*/
URT       MOV.B    #UTXE1,&ME2                   /*USART0 发送模块允许*/
          MOV      #GIE,SR                       /*中断允许*/
          MOV.B   #010H,&UCTL1                   /*控制寄存器的写入*/
          MOV.B   #02EH,&UTCTL1                  /*发送控制位的写入*/
          MOV.B   #043H,&UBR01                   /*写分频因子*/
          MOV.B   #003H,&UBR11
          MOV.B   #000H,&UMCTL1                  /*调整分频因子*/
          MOV.B   # UTXIE1,&IE2                  /*USART0 发送中断允许*/
          MOV.B   #032H,&U1TXBUF                 /*写入的发送数据*/
          JMP       URT                          /*接受中断允许*   /
FASONG MOV.B  #032H,&U1TXBUF                     /*写入的发送数据*/
           JMP   FASONG                          /*连续发字符*/
          RETI                                   /*中断返回*/
          COMMON INTVEC
          ORG    UART1TX_VECTOR                  /*  中断向量*/
            DW     FASONG
            ORG    RESET_VECTOR
            DW     RESET
      END
```

2．Usar.s43 程序（接受程序）

```
    #include  "MSP430x14x.h"
        RSEG   UDATA0
        DS   0
        RSEG   CSTACK
        DS   0
        RSEG   CODE
        DS  0
```

```
RESET   MOV      #SFE(CSTACK),SP                    /*初始化堆栈指针*/
        MOV      #(WDTHOLD+WDTPW),&WDTCTL           /*停止看门狗定时器*/
        MOV.B  #02EH,&UTCTL1                        / *选定时钟源*/
        MOV.B  #043H,&UBR01                         /*确定分频因子*/
        MOV.B  #003H,&UBR11
        MOV.B  #000H,&UMCTL1
        BIS.B    #030H,&P5DIR                      /*初始化端口功能为外设*/
        BIS.B    #030H,&P5SEL
        MOV.B  #040H,&BCSCTL1                       /*转换时钟*/
        BIC      #OSCOFF,SR
        PUSH     #050H
LOOP    DEC      0(SP)
        JNZ      LOOP
        BIC.B    #OFIFG,&IFG1
        MOV.B   #088H,&BCSCTL2
        MOV.B   #000H,&P3DIR
        MOV.B   #0FFH,&P3SEL
        MOV      #0200H,R6
        EINT
        MOV.B     #URXE1,&ME2              /*USART0 接受模块允许*/
        MOV       #GIE,SR
        MOV.B    #URXIE1,&IE2
        MOV.B    #020H,&UTCTL1
        BIC.B     #URXSE,&UTCTL1
        BIS.B     #URXSE,&UTCTL1
        MOV.B    #010H,&UCTL1              /*控制寄存器的写入*/
        MOV.B    #008H,&URCTL1
        MOV.B    #043H,&UBR01              /*写分频因子*/
        MOV.B    #003H,&UBR11
        MOV.B    #000H,&UMCTL1
        MOV.B    #URXIE1,&IE2              /*接收中断允许*/
        EINT
URT     MOV.B     #URXIE1,&IE2
        JMP        URT                     /*中断等待，接受数据*/
RECEI   MOV.B  &U1RXBUF,0(R6)              /*中断程序，接收数据到数据段*/
        MOV.B  #000H,&IE2                  /*中断禁止*/
        RETI                               /*中断返回*/
        COMMON INTVEC                      /*中断向量表*/
        ORG    UART1RX_VECTOR
        DW     RECEI
        ORG    RESET_VECTOR
        DW     RESET
      END
```

3 Eerom.s43 程序：

```
 #include  "msp430x14x.h"
        ORG    0FFFEH                      /*定义段和复位地址*/
        DW     Reset
        RSEG   UDATA0
        DS    0
        RSEG   CSTACK
        DS   0
```

```
        RSEG   CODE
        DS  0
Reset   MOV   #SFE(CSTACK),SP          /*初始化堆栈指针*/
        MOV   #(WDTHOLD+WDTPW),&WDTCTL/*停看门狗*/
        CALL  #XIE                     /*调用写子程序*/
DU1     CALL  #DU                      /*调用读子程序*/
        JMP   DU1                      /*重复调用便于观察一帧的数据*/
CUSHIHUA                               / *初始化子程序*/
        CALL  #QIDONG                  /*调用启动子程序*/
        CALL  #SONG1                   /*以下为调用芯片地址 1010000*/
        CALL  #SONG10                  /*调用 1 后的送 0 子程序*/
        CALL  #SONG1                   /*调用 0 后的送 1 子程序*/
        CALL  #SONG10                  /*调用 1 后送 0 子程序*/
        CALL  #SONG00                  /*调用 0 后送 0 子程序*/
        CALL  #SONG00
        CALL  #SONG00
        RET
XIE     CALL  #CUSHIHUA                /*调用初始化子程序*/
        CALL  #SONG00                  /*为写,因此送 0*/
        CALL  #YINDA                   /*送完地址字节 10100000,等待应答*/
        CALL  #SONG1                   /*送选定的地址 10000000*/
        CALL  #SONG10
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG00
        CALL  #YINDA                   /*送完选定的地址字节后为等待应答*/
        CALL  #SONG1                   /*送数据 11001100 到指定的地址*/
        CALL  #SONG11                  /*调用 1 后送 1 的子程序*/
        CALL  #SONG10
        CALL  #SONG00
        CALL  #SONG1
        CALL  #SONG11
        CALL  #SONG10
        CALL  #SONG00
        CALL  #YINDA                   /*等待应答*/
        MOV.B #060H,&P4DIR             /*应答后 SDA 数据线应为输出状态*/
        MOV.B #000H,&P4OUT             /*输出数据线 SDA 和 SCL 都为低,便于停止位的执行*/
        CALL  #TINGZHI                 /*调用停止子程序*/
        RET                            /*写完后返回*/
DU      CALL  #CUSHIHUA                /*读子程序,调用初始化子程序,选芯片地址为 1010000*/
        CALL  #SONG00                  /*先送写信号为 0*/
        CALL  #YINDA                   /*送完为等应答*/
        CALL  #SONG1                   /*送选中的地址为 10000000*/
        CALL  #SONG10
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG00
```

```
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG00
        CALL  #YINDA              /*送完地址,等待应答*/
        MOV.B #000H,&P4OUT        /*应答完,转为输出方式*/
        MOV.B #060H,&P4DIR
        CALL  #QIDONG1            /*又启动*/
        CALL  #SONG1              /*送选中的地址 1010000*/
        CALL  #SONG10
        CALL  #SONG1
        CALL  #SONG10
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG00
        CALL  #SONG1              /*送读信号 1*/
        CALL  #YINDA              /*等待应答*/
        CALL  #DUSHU              /*读 1000000 数据*/
        CALL  #DUSHU
        CALL  #DUSHU
        CALL  #DUSHU
        CALL  #DUSHU
        CALL  #DUSHU
        CALL  #DUSHU
        CALL  #DUSHU
        CALL  #TINGZHI            /*读完后停止*/
        RET                       /*返回*/
QIDONG  MOV.B #060H,&P4OUT        /*启动子程序,是在非忙时当 SCL 为高,SDA 由 1 到 0 跳变*/
        MOV.B #060H,&P4DIR
        MOV.B #060H,&P4OUT
        MOV.B #040H,&P4OUT
        RET
QIDONG1                           /*是在由读转为写时的启动信号*/
        MOV.B #060H,&P4DIR
        MOV.B #060H,&P4OUT
        MOV.B #040H,&P4OUT
        RET
SONG1   MOV.B #060H,&P4DIR        /*在上一个时刻是 0 时的送 1 子程序*/
        MOV.B #000H,&P4OUT
        MOV.B #020H,&P4OUT
        MOV.B #060H,&P4OUT
        CALL  #WAIT1
        RET
SONG11  MOV.B #020H,&P4OUT        /*在上一个时刻是 1 时的送 1 子程序*/
        MOV.B #060H,&P4DIR
        MOV.B #060H,&P4OUT
        CALL  #WAIT1
        RET
SONG10  MOV.B #060H,&P4DIR        /*在上一个时刻是 1 时的送 0 子程序*/
        MOV.B #020H,&P4OUT
        MOV.B #000H,&P4OUT
        MOV.B #040H,&P4OUT
```

```
          CALL  #WAIT1
          RET
SONG00   MOV.B #060H,&P4DIR    /*在上一个时刻是 0 时的送 0 子程序*/
          MOV.B #000H,&P4OUT
          MOV.B #040H,&P4OUT
          CALL  #WAIT1
          RET
WAIT1    MOV   #50H,R5          /*延时子程序*/
WW      SUB   #1,R5
          JZ    FAHUI
          JMP   WW
FAHUI    RET
YINDA    MOV.B #000H,&P4OUT      /*应答子程序,时序见论文*/
          MOV.B #020H,&P4OUT
          MOV.B #040H,&P4DIR
          MOV.B #040H,&P4OUT
          CALL  #WAIT1
          RET
TINGZHI MOV.B #000H,&P4OUT      /*停止子程序,时序见论文*/
          MOV.B #040H,&P4OUT
          MOV.B #060H,&P4OUT
          RET
DUSHU    MOV.B #000H,&P4OUT     /*读数子程序,时序见论文*/
          MOV.B #040H,&P4OUT
          CALL  #WAIT1
          RET
      END
```

4　A/D 转换程序

```
    ... ...
MAIN              MOV   #SFE(CSTACK),SP       /*初始化堆栈指针*/
                  MOV   #(WDTHOLD+WDTPW),&WDTCTL
BB                CALL   #ADCHANGE
                  JMP    BB
ADCHANG           BIS     #GIE,SR
                  EINT
                  MOV    #0098H,&ADC12CTL0
                  MOV.B  #040H,&ADC12MCTL0
                  BIS     #00004H,&ADC12CTL1
                  MOV.B  #000H,&P6DIR
                  MOV.B  #0FFH,&P6SEL
                  BIS     #ENC,&ADC12CTL0
                  BIS     #00001H,&ADC12CTL0
                  CLR     &ADC12MEM0
                  RET
                  END
```

摘要：MSP430 是一类现场 16 位数据线 FLASH 存储器的单片机，该单片机以丰富的片上资源，高速和高精度而深受广大单片机爱好者的青睐。而本设计充分利用其资源，实现了 A/D 转换、多机通信、外存和实时时钟、PWM 波形的函数发生器、比较器测量、定时器的捕获测量周期、8M 方波产生、硬件乘法器等等。通过设计的 PCB 板，可以应用到大量的工业自动控制中，实现低功耗、低辐射、低污染的控制。

关键字：MSP430 A/D 转换 USART 24WCXX 8563 比较器 PWM 定时器 硬件乘法器 电源管理 捕获定时 AAT3221 FLASH

Abstract: MSP430 is a kind of Single-Chip CPU that is presented 16 bit bus and has FLASH memory. Many people will like it because it has a lot of resources in the chip and it runs in high speed and high precision . We use its resource and realize A/D switch、correspondence between compute and Single-Chip CPU、 external memory、to come into being PWM wave、comparison and measure 、to enumerate and capture 、to come into being 8 M pulses 、 the multiplication of hardware and so on . We can use the PCB in the control of industry to realize low power and low radiation and low pollution .

KEYWORD:**MSP430 A/D switch USART 24WCXX 8563 Comparison Enumerate Pwm Multiplication of Hardware Manager of Power Enumerate and Capture AAT3221 FLASH**