《DSP 教学实验系统》实验报告

课程名称：**DSP** 技术工程与应用

学生姓名：尉前进

学　　号：**1170400423**

实验日期：**2020.5.24**

实验成绩：

教师评语：

哈尔滨工业大学自动控制实验中心

实验目录

实验四 模数转换（A\D）

一．实验目的

1．通过实验熟悉F2812A 的定时器。

2．掌握F2812A 片内AD 的控制方法。

二．实验设备

计算机，ICETEK-F2812-EDU 实验箱（仿真器+ICETEK-F2812-A 系统板+相关连线及电源）。

三．实验原理

1．TMS320F2812A 芯片自带模数转换模块特性

- 12 位模数转换模块ADC，快速转换时间运行在25mhz，ADC 时钟或12.5MSPS。

-16 个模拟输入通道（AIN0—AIN15）。

-内置双采样-保持器

-采样幅度：0-3v，**切记输入ad 的信号不要超过这个范围，否则会烧坏2812 芯片的。**

2．模数模块介绍

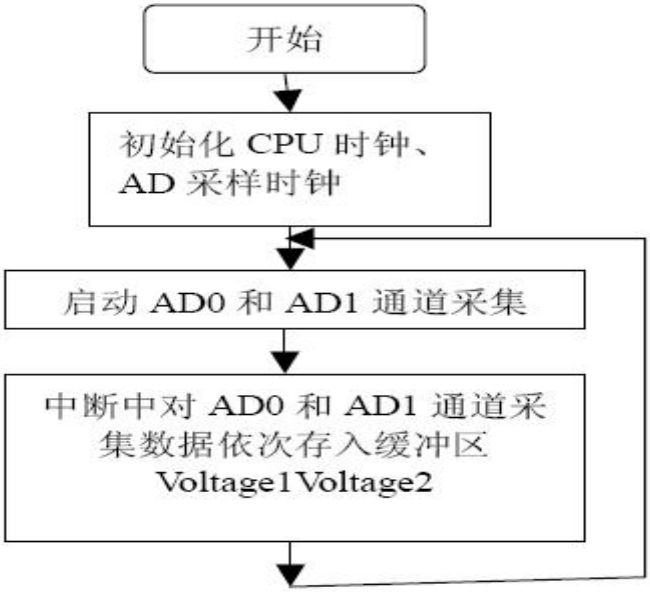
ADC 模块有16 个通道，可配置为两个独立的8 通道模块以方便为事件管理器A 和B服务。两个独立的8 通道模块可以级连组成16 通道模块。虽然有多个输入通道和两个序列器，但在ADC 内部只有一个转换器，同一时刻只有1 路ad 进行转换数据。

3．模数转换的程序控制

模数转换相对于计算机来说是一个较为缓慢的过程。一般采用中断方式启动转换或保存结果，这样在CPU 忙于其他工作时可以少占用处理时间。设计转换程序应首先考虑处理过程如何与模数转换的时间相匹配，根据实际需要选择适当的触发转换的手段，也要能及时地保存结果。关于TMS320F2812A DSP 芯片内的A/D 转换器的详细结构和控制方法，请参见文档

spru060a.pdf。

4．实验程序流程图



四．实验步骤：见实验指导书

五．问题与思考

1 写出 ccs 软件设置过程,说明 A/D 采集作用及原理。

（1）CCS 配置 ADC 寄存器过程：

①需要设置采样窗口大小、排序器工作方式、每次转换通道数目、通道编程；

②转换模式、触发方式、中断使能与工作模式、排序器复位；

③时钟分频、电源控制。

（2）A/D 采集的作用

将输入的模拟量转化为数字量，根据每一个采样时刻的数值复现输入信号，可以确定输入信号的幅值、频率等信息。

2．打开 ADC.C 文件，结合课上的学习请给如下加“//”语句加上注释。

```
main()
{
InitSysCtrl();//初始化系统控制
    DINT;// 关中断
    InitPieCtrl();//初始化 Pie 寄存器

    IER = 0x0000;// 紧张所有中断
    IFR = 0x0000;
    InitPieVectTable();//初始化 Pie 中断向量表
```

3 打开 ADC.C 文件，结合课上的学习，请分别说明以下两段代码的意义。

```
  //
    PieCtrlRegs.PIEIER1.bit.INTx6 = 1;
    IER |= M_INT1; // 打开中断 1
    EINT;                // 启用全局中断
    ERTM;                //打开全局实时中断
    LoopCount = 0;
    ConversionCount = 0;
```

```
// 重新初始化下一个ADC序列
  AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;          // 复位SEQ1
  AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;        // 清除SEQ1中断标志位
  PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
```

实验五 发光二极管阵列显示实验

一．实验目的

通过实验学习使用F2812 DSP 的扩展扩展端口控制外围设备的方法，了解发光二极管阵列的控制编程方法。

二．实验设备

计算机，ICETEK-F2812-EDU 实验箱。

三．实验原理

ICETEK-F2812-A 是一块以TMS320F2812DSP 为核心的DSP 扩展评估板，它通过扩展接口与实验箱的显示/控制模块连接，可以控制其各种外围设备。发光二极管显示阵列的显示是由扩展扩展端口控制，DSP 须将显示的图形按列的顺序存储起来(8×8 点阵，8 个字节，高位在下方，低位在上方)，然后定时刷新控制显示。具体方法是，将以下控制字按先后顺序，每两个为一组发送到全局控制寄存器的第6-4 位和端口0x108005，发送完毕后，隔不太长的时间(以人眼观察不闪烁的时间间隔)再发送一遍。由于位值为“0”时点亮，所以需要将显示的数据取反。

000B,第8 列数据取反；001B,第7 列数据取反；

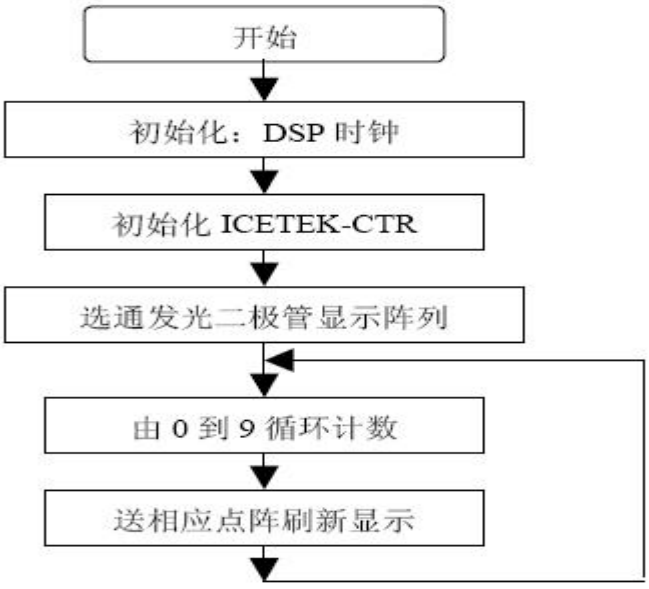
010B,第6 列数据取反；011B,第5 列数据取反；

100B,第4 列数据取反；101B,第3 列数据取反；

110B,第2 列数据取反；111B,第1 列数据取反。

注意：在使用前须在扩展端口108007 写入控制字0x0C1，以打开此设备。关闭时写0x0C0。

试验程序流程图



四．实验步骤：参见实验指导书

五．问题与思考

1 实验中发光阵列二极管排列方法？

1 实验中发光阵列二极管排列方法？

8×8 点阵，8 个字节，高位在下方，低位在上方。

2 读取程序，试在程序中怎样修改延时时间？

将 main 函数 for 循环中的 Delay 函数的延时时间，可以从 256 改成 2560.

```
for (;;)
{
    SetLEDArray(nCount);
    Delay(2560);           //原为 Delay(256);
    nCount++;
    nCount%=10;
}
```

3 修改程序定时器时间，运行结果如何？

每个状态维持的时间变短，数字变化速度变快。

4 发光管显示“HIT”时，写出修改的部分程序。

（1）将 ledkey 二维数组修改为 3 行 8 列，内容修改如下：

```
unsigned char ledkeyhit[3][8]=
{
{0xFF,0xFF,0x18,0x18,0x18,0x18,0xFF,0xFF},   //H
{0x00,0x00,0x00,0xFF,0xFF,0x00,0x00,0x00},   //I
{0x03,0x03,0x03,0xFF,0xFF,0x03,0x03,0x03}    //T
};
```

（2）将主函数 for 循环中的 nCount%=10 改成 nCount%=3

```
for (;;)
{
    SetLEDArray(nCount);
    Delay(2560);           //原为 Delay(256);
    nCount++;
    nCount%=10;
}
```

实验七　异步串口通信

一. 实验目的

1．了解ICETEK-F2812-A 评估板上扩展标准RS-232 串行通信接口的原理和方法。

2．学会对串行通信芯片的配置编程。

3．学习设计异步通信程序。

二. 实验设备

计算机，ICETEK-F2812-EDU 实验箱（或ICETEK 仿真器+ICETEK-F2812-A 系统板+相关连线及电源）。

三. 实验原理

1．异步串行通信原理（见spru051a.pdf）

2．ICETEK-F2812-A 评估板异步串口设计

在板上加上Max232 部分即可。驱动电路主要完成将输出的0-3.3V 电平转换成异步串口电平的工作，转换电平的工作由MAX232 芯片完成。

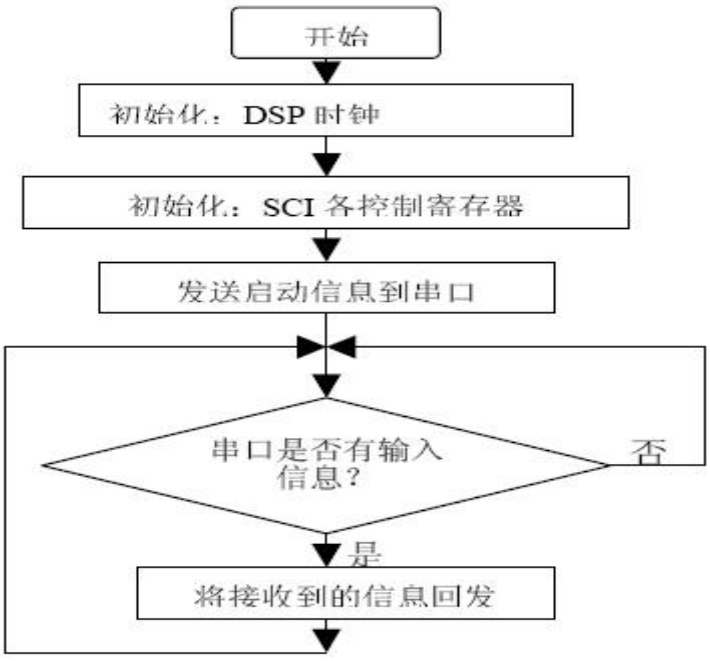
3．串行通信接口设置

*串行通信接口波特率计算

内部生成的串行时钟是由低速的外部时钟LSPCLK 频率和波特率选择寄存器决定。对于一个给定的设备时钟，SCI 通过波特率选择器的16 位值从64k 开始的不同串行时钟频率中选择一个时钟。

理想的波特率	BRR值
2400	7A0H
4800	3D0H
9600	1E7H
19200	F3H
38400	79H

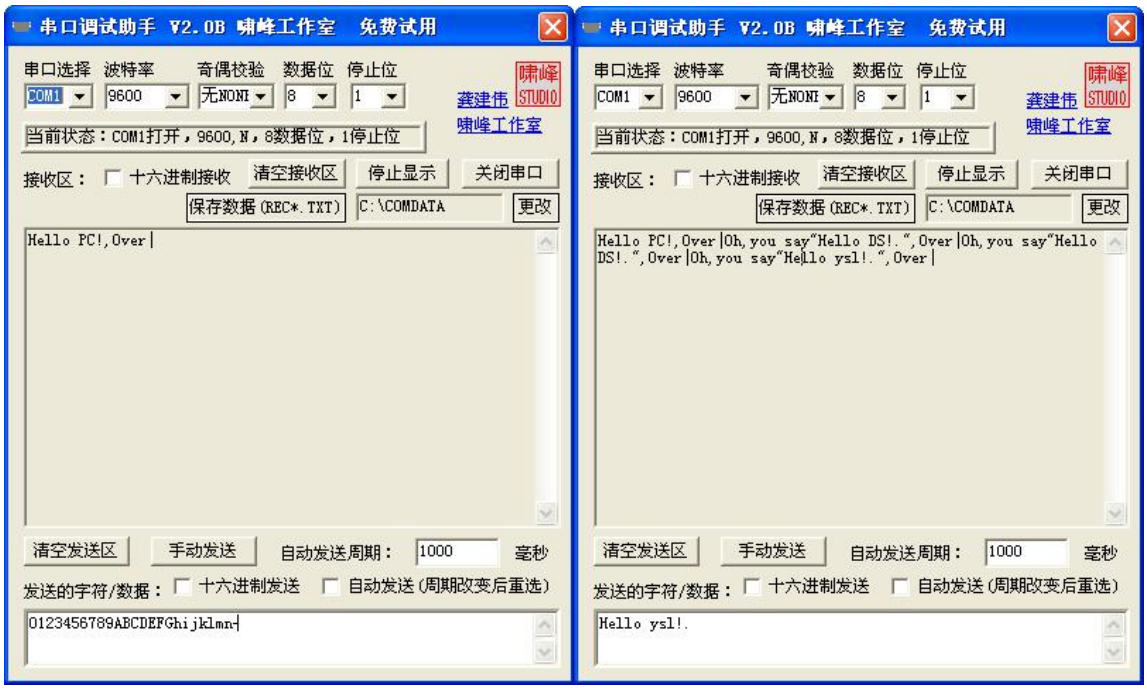
4．实验程序流程图：



四. 实验步骤：参见实验指导书

五. 问题与思考

1 给出源程序中串口发送的显示信息。描述查询方式和中断方式的特征结构，并给出相应语句。



（1）查询方式特征结构：循环结构中出现bReceive=1时，发送接收字符串"Oh,you say"再将接收到字符串发送回来。

```
if ( bReceive==1 )
      {for ( i=0;i<10;i++ )
          {
              scia_xmit(cAnswer[i]);
              while(SciaRegs.SCIFFTX.bit.TXFFST !=0) { }
          }

          scia_xmit('\"');
          for ( i=0;i<nLen;i++ )
          {

              scia_xmit(cBuffer[i]);
              while(SciaRegs.SCIFFTX.bit.TXFFST !=0) { }
          }

          scia_xmit('\"');
          wait(1024);
          for ( i=9;i<16;i++ )
          {
              scia_xmit(cString[i]);
              while(SciaRegs.SCIFFTX.bit.TXFFST !=0) { }
          }
```

```
        }
        k=0; bReceive=0;
        while(1)
         {
            while(SciaRegs.SCIFFRX.bit.RXFIFST ==0) { } // 如果接受寄存器不为 0 则跳
出
            ReceivedChar = SciaRegs.SCIRXBUF.all;
            cBuffer[k]=ReceivedChar;

            if ( ReceivedChar=='.')
            {
                cBuffer[k+1]='\0';
                nLen=k+1;
                bReceive=1;
                break;
            }
            k++; k%=16;
}
```

（2）中断方式特征结构：运用中断处理函数做接收数据的处理。

```
interrupt void scia_rx_isr(void)
{
    ReceivedChar = SciaRegs.SCIRXBUF.all;
    cBuffer[k]=ReceivedChar;
    if ( ReceivedChar=='.')
    {
        cBuffer[k+1]='\0';
        nLen=k+1;
        bReceive=1;
    }
    ReceivedChar=0;
    k++; k%=16;
    SciaRegs.SCIFFRX.bit.RXFFINTCLR = 1;
    PieCtrlRegs.PIEACK.all = M_INT9;  //清除第9组中断的响应标志位
}
```

2 怎样用中断方式设计程序完成异步串行通信，根据所给程序流程图编写相应程序。

```
#include "DSP281x_Device.h"
#include "DSP281x_Examples.h"

// Prototype statements for functions found within this file.
void scia_loopback_init(void);
void scia_fifo_init(void);
void scia_xmit(int a);
void error(int);
void wait(int nWait);
```

```
interrupt void scia_rx_isr(void);
interrupt void scia_tx_isr(void);
void allinit(void);
void step1(void);
void step2(void);
void step3(void);
void step4(void);
void step5(void);
// Global counts used in this example
Uint16 LoopCount;
Uint16 ErrorCount;
char cString[17]={ "Hello
PC!,Over|" },cReceive,cBuffer[17],cAnswer[16]={"Oh,you say"};
int k=0;
int i,nLen,bReceive=0;
char ReceivedChar;
int test=0;

void main(void)
{
    step1();
    step2();
    step3();
    //串口基础设置
    SciaRegs.SCICCR.all =0x0007;   // 1 stop bit,  No loopback
                               // No parity,8 char bits,
                               // async mode, idle-line protocol
    SciaRegs.SCICTL1.all =0x0003;  // reset SCI, enable RX, internal SCICLK,
                               // enable TX, RX ERR, SLEEP, TXWAKE
    SciaRegs.SCICTL2.all =0x0002;  //enable RX/BK INT, Disable TX INT ENA
    SciaRegs.SCIHBAUD    =0x0001;
    SciaRegs.SCILBAUD    =0x00e7;
    step4();
    step5();
    LoopCount = 0;
    ErrorCount = 0;
    for ( i=0;i<16;i++ )
    {
        scia_xmit(cString[i]);
        while(SciaRegs.SCIFFTX.bit.TXFFST !=0) { }
              //wait(1024);
    }
    for(;;)
    {
      if ( bReceive==1 )
      {
```

```
        for ( i=0;i<10;i++ )
            {
                scia_xmit(cAnswer[i]);
                while(SciaRegs.SCIFFTX.bit.TXFFST !=0) { }
            }

            scia_xmit('\"');
            for ( i=0;i<nLen;i++ )
            {

                scia_xmit(cBuffer[i]);
                while(SciaRegs.SCIFFTX.bit.TXFFST !=0) { }
            }

            scia_xmit('\"');
            wait(1024);
            for ( i=9;i<16;i++ )
            {
                scia_xmit(cString[i]);
                while(SciaRegs.SCIFFTX.bit.TXFFST !=0) { }
            }
            bReceive=0;
            k=0;
        }
    }
}

void error(int ErrorFlag)
{
     ErrorCount++;
}
// Transmit a character from the SCI'
void scia_xmit(int a)
{
    SciaRegs.SCITXBUF=a;
}

void wait(int nWait)
{
    int i,j,k=0;
    for ( i=0;i<nWait;i++ )
        for ( j=0;j<64;j++ )
            k++;
}

//中断服务函数
```

```
interrupt void scia_rx_isr(void)
{
    ReceivedChar = SciaRegs.SCIRXBUF.all;
    cBuffer[k]=ReceivedChar;
    if ( ReceivedChar=='.')
    {
           cBuffer[k+1]='\0';
           nLen=k+1;
           bReceive=1;
    }
    ReceivedChar=0;
    k++; k%=16;
    SciaRegs.SCIFFRX.bit.RXFFINTCLR = 1;
    PieCtrlRegs.PIEACK.all = M_INT9;  //清除第9组中断的响应标志位
}

void step1(void)
{
    InitSysCtrl();
}

void step2(void)
{
    EALLOW;
    //************SCIA group RS422
    GpioMuxRegs.GPFMUX.bit.SCITXDA_GPIOF4=1;    // Select GPIOs to be SciA
    GpioMuxRegs.GPFMUX.bit.SCIRXDA_GPIOF5=1;
    EDIS;
}

void step3(void)
{
    DINT;//关中断
    InitPieCtrl();//初始化pie寄存器
    IER = 0x0000;//禁止所有的中断
    IFR = 0x0000;
    InitPieVectTable();//初始化pie中断向量表

    EALLOW;  // This is needed to write to EALLOW protected registers
    PieVectTable.RXAINT=&scia_rx_isr;//指定中断服务子程序,422 sciB rx
    EDIS;
}

void step4(void)
{
    SciaRegs.SCICCR.bit.LOOPBKENA =0; // disable loop back
```

```
    SciaRegs.SCIFFTX.all=0xE040;
    SciaRegs.SCIFFRX.all=0x2026;      //enable接收FIFO匹配中断6LELVE
    SciaRegs.SCIFFRX.bit.RXFFIL=1;
    SciaRegs.SCIFFCT.all=0x0;      //禁止波特率校验
    SciaRegs.SCICTL1.all =0x0063;     // Relinquish SCI from Reset
    SciaRegs.SCIFFTX.bit.TXINTCLR = 1;//清除中断标志位
}

void step5(void)
{
    PieCtrlRegs.PIECRTL.bit.ENPIE=1;
    //ENABLE THE PIE BLOCK the PIE: Group 9 interrupt 1
    IER |= M_INT9;
    PieCtrlRegs.PIEIER9.bit.INTx1 = 1;//   使能422接收中断
    PieCtrlRegs.PIEIER9.bit.INTx2 = 0;// 使能422发送中断
    EINT;   // Enable Global interrupt INTM
    ERTM;   // Enable Global realtime interrupt DBGM
}
```

实验十二　PID 算法闭环电机控制

一．实验目的

1．　掌握利用 ICETEK-F2812-A 评估板与ICETEK-CTR 板上带速度反馈的直流电机B 的连接和控制原理。

2．熟悉F2812DSP 的通用IO 端口和定时器的编程使用。

3．学习利用数字PID 控制算法控制电机转速。

二．实验设备

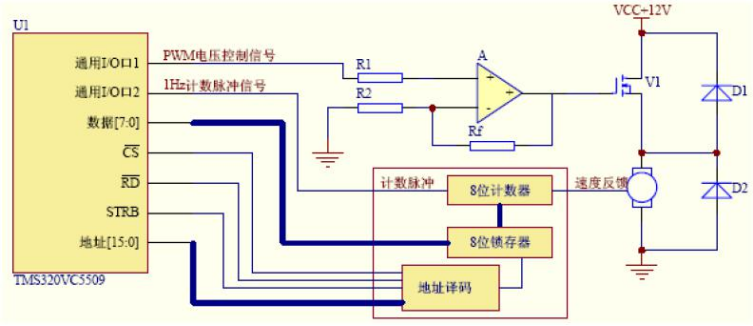
计算机，ICETEK-F2812-EDU 实验箱。

三．实验原理

1．直流电机测速原理

直流电机B：在ICETEK-CTR 板上有一个带速度反馈的直流电机B，它的额定工作电压为+12V，额定转速为6500 转，带有速度反馈线路，反馈信号为方波脉冲，其频率与转速成正比(电机转动一圈产生两个脉冲)。

电机闭环控制系统：如图在DSP 系统板的控制下形成闭环速度控制系统，DSP 发送的PWM 波控制直流电机的转速，通过速度反馈，DSP 可实时读取当前速度值，利用DSP 中运行的控制程序根据速度读数控制 PWM 的脉宽，从而实现闭环调速控制。



直流电机设计原理

2．数字PID 控制器

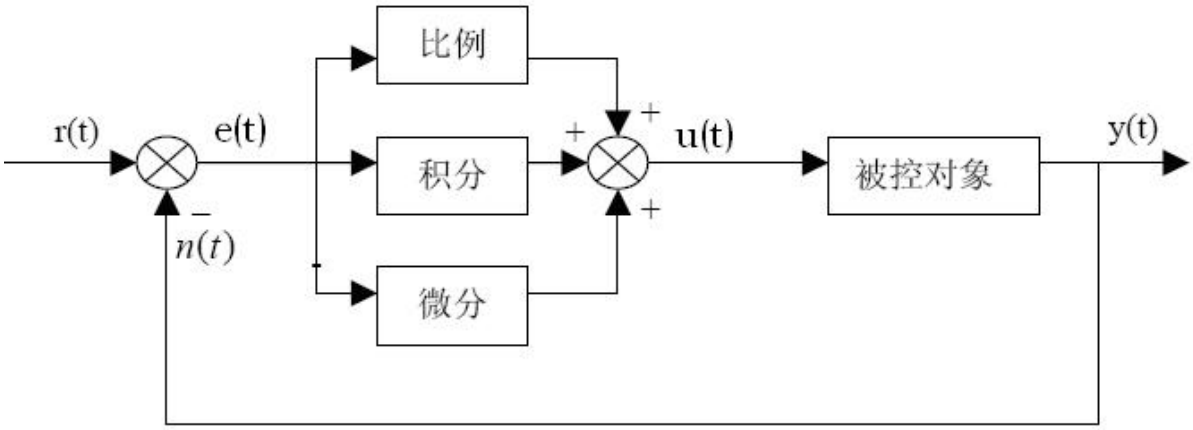
将偏差的比例(P)、积分(I)和微分(D)通过线性组合构成控制量，用这一控制量对被控对象进行控制，这样的控制器称PID 控制器。

(1)模拟PID 控制原理

模拟PID 控制系统原理图如图所示。该系统由模拟PID 控制器和被控对象组成。图中，r(t)是给定值，y(t)是系统的实际输出值，给定值与实际输出值构成控制偏差e(t)：e(t)=r(t)-y(t)，e(t)作为PID 控制器的输入，u(t)作为PID 控制器的输出和被控对象的输入。所以模拟PID 控制器的控制规律为

$$u(t) = K_P[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D\frac{de(t)}{dt}] + u_0 \qquad (1)$$

其中： K_P ——比例系数，T_I ——积分常数，T_D ——微分常数，u_0 ——控制常量



PID各环节作用：

比例环节的作用是对偏差瞬间做出快速反应。偏差一旦产生，控制器立即产生控制作用，使控制量向减少偏差的方向变化。控制作用的强弱取决于比例系数$_PK$ ， $_PK$ 越大，控制越强，但过大的$_PK$ 会导致系统震荡，破坏系统的稳定性。

积分环节的作用是把偏差的积累作为输出。在控制过程中，只要有偏差存在，积分环节的输出就会不断增大。直到偏差e(t)=0，输出的u(t)才可能维持在某一常量，使系统在给定值r(t)不变的条件下趋于稳态。积分环节的调节作用虽然会消除静态误差，但也会降低系统的响应速度，增加系统的超调量。积分常数$_IT$ 越大，积分的积累作用越弱。增大积分常数$_IT$会减慢静态误差的消除过程，但可以减少超调量，提高系统的稳定性。所以，必须根据实际控制的具体要求来确定$_IT$ 。

微分环节的作用是阻止偏差的变化。它是根据偏差的变化趋势(变化速度)进行控制。偏差变化越快，微分控制器的输出越大，并能在偏差值鞭打之前进行修正。微分作用的引入，将有助于减小超调量，克服震荡，使系统趋于稳定。但微分的作用对输入信号的噪声很敏感，对那些噪声大的系统一般不用微分，或在微分起作用之前先对输入信号进行滤波。适当地选择微分常数$_DT$ ，可以使微分的作用达到最优。

(2)数字PID 控制算法

由于计算机的出现，计算机进入了控制领域。人们将模拟 PID 控制规律引入到计算机中来。由于计算机控制是一种采样控制，它只能根据采样使可的偏差计算控制量，而不能象模拟控制那样连续输出控制量，进行连续控制。由于这一特点，式(1-1)中的积分和微分项不能直接使用，不许进行离散化处理。离散化处理的方法为：以 T 作为采样周期，k 作为采样序号，则离散采样时间 kT 对应着连续时间 t，用求和的形式代替积分，用增量的形式代替微分，可得到如下表达式：

$$u_k = K_P[e_k + \frac{T}{T_I}\sum_{j=0}^{k} e_j + \frac{T_D}{T}(e_k - e_{k-1})] + u_0 \qquad (2)$$

如果只需要计算控制量的增量Δu_k，可以使用增量式PID 控制算法。由式(1-3)可得控制器在第k-1 个采样时刻的输出值为：

$$u_{k-1} = K_P[e_{k-1} + \frac{T}{T_I}\sum_{j=0}^{k-1} e_j + \frac{T_D}{T}(e_{k-1} - e_{k-2})] + u_0 \qquad (3)$$

将(2)与(3)相减，就可以得到增量式 PID 控制算法公式为

$$\Delta u_k = u_k - u_{k-1} = K_P[e_k - e_{k-1} + \frac{T}{T_I}e_k + \frac{T_D}{T}(e_k - 2e_{k-1} + e_{k-2})] =$$

$$K_P(1 + \frac{T}{T_I} + \frac{T_D}{T})e_k - K_P(1 + \frac{2T_D}{T})e_{k-1} + K_P\frac{T_D}{T}e_{k-2} =$$

$$Ae_k + Be_{k-1} + Ce_{k-2}$$

式中：$A = K_P(1 + \frac{T}{T_I} + \frac{T_D}{T})$，$B = -K_P(1 + 2\frac{T_D}{T})$，$C = K_P\frac{T_D}{T}$

由上式可以看出，如果计算机控制系统采用恒定的采样周期T,一旦确定了A、B、C，只要用前后3 次测量值的偏差，就可以由上式求出控制增量。

(3)程序设计

控制环节：系统维护一个全局变量pwm，计算控制电机绕组电压的PWM 波形的占空比，取值越大，通过电机的电流越多，电机加速，反之，占空比越小电机越慢。

采样环节：由于电机速度反馈信号频率为几十赫兹，所以设计测量周期较长，这样保证偶然偏差值较少发生，但系统“反应”较慢。采样脉冲设计为1 赫兹的方波信号。测量的结果为电机转动圈数，比如测速结果为84，则实际转速为84 转/秒。

计算环节：利用公式(1-6)，取A=0.6、B=0.2、C=0.1，直接由测速结果计算出占空比调节增量，计算中限制了增量的最大值不能超过10，以免引起太大的电流波动。

显示：在每次调节电机转速时刷新显示各参数。“设定”为实验者指定电机转速，单位为“转/秒”；“测速”为通过采样环节得到的电机速度测量值，单位也为“转/秒”；“误差”指速度测量值与设定值之差；“调整”为通过PID 算法得到并付诸调整的调整值，调整对象为

占空比；“占空比”当前采用的占空比；“输入”通过小键盘输入新的转速设置，按“Enter”键生效。

9． 序流程图：见下页

四．实验步骤

1．实验准备

(1)连接实验设备：

(2)将ICETEK-CTR 板的供电电源开关拨动到“开”的位置。

2．设置Code Composer Studio 2.21 在硬件仿真(Emulator)方式下运行

3．启动Code Composer Studio 2.21，选择菜单Debug→Reset CPU。

4．打开工程文件

工程目录C:\ICETEK-F2812-A-EDUlab\DSP281x_examples\lab0604-PID浏览PID.c 文件的内容，理解各语句作用。

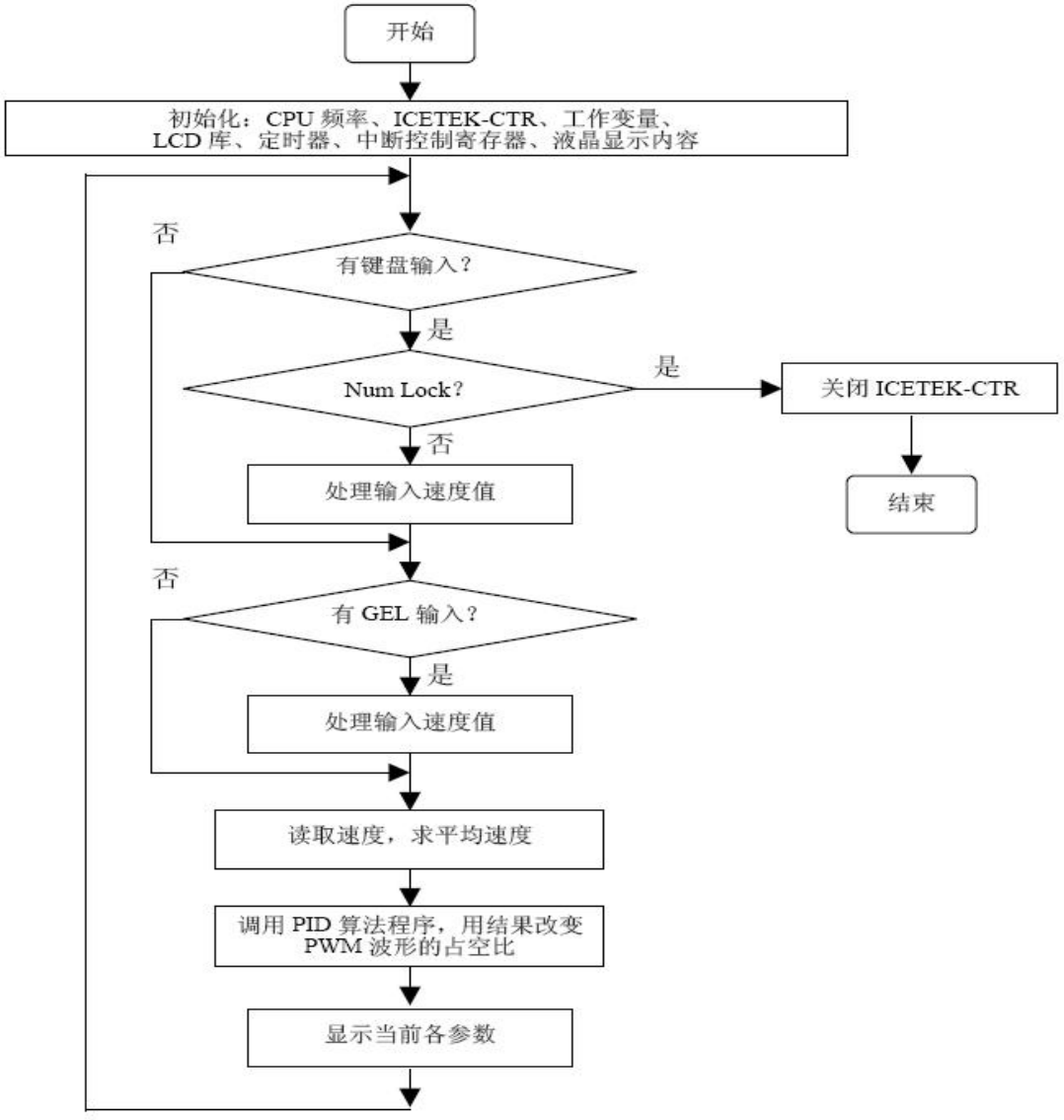
5．编译、下载、运行程序观察结果

-单击“Debug”菜单，“Run”项，运行程序。观察ICETEK-CTR 上液晶上的显示。

6．设置电机转速

按小键盘上数字键，修改转速(单位：转/秒)，输入在液晶显示屏上可观察，按回车生效。继续观察液晶显示屏中参数在PID 控制算法下的变化。

7．结束程序运行，退出CCS。



五．问题与思考

1 程序原稳定参数，当float a=0.8f,b=0.3f,c=0.2f时，电机转速始终在70转/秒上下波动，当修改后的参数为float a=0.6f,b=0.2f,c=0.1f时，电机转速始终在50转/秒上下波动.随着时间的变化转速趋向于设定值。参数a,b,c代表着PID算法里的哪些控制量？具体的作用？

a代表Kp,具体作用是增大控制器输出，减小系统的输出误差，提高精度
b代表Ki,具体作用是进一步提高系统的精度
c代表Kd,具体作用是增大阻尼比，减小超调量

2. 结合课上知识，源程序中InitSysCtrl();语句在本案例中对哪几部分进行了初始化
关闭看门狗
初始化 PLL 时钟
初始化外设时钟