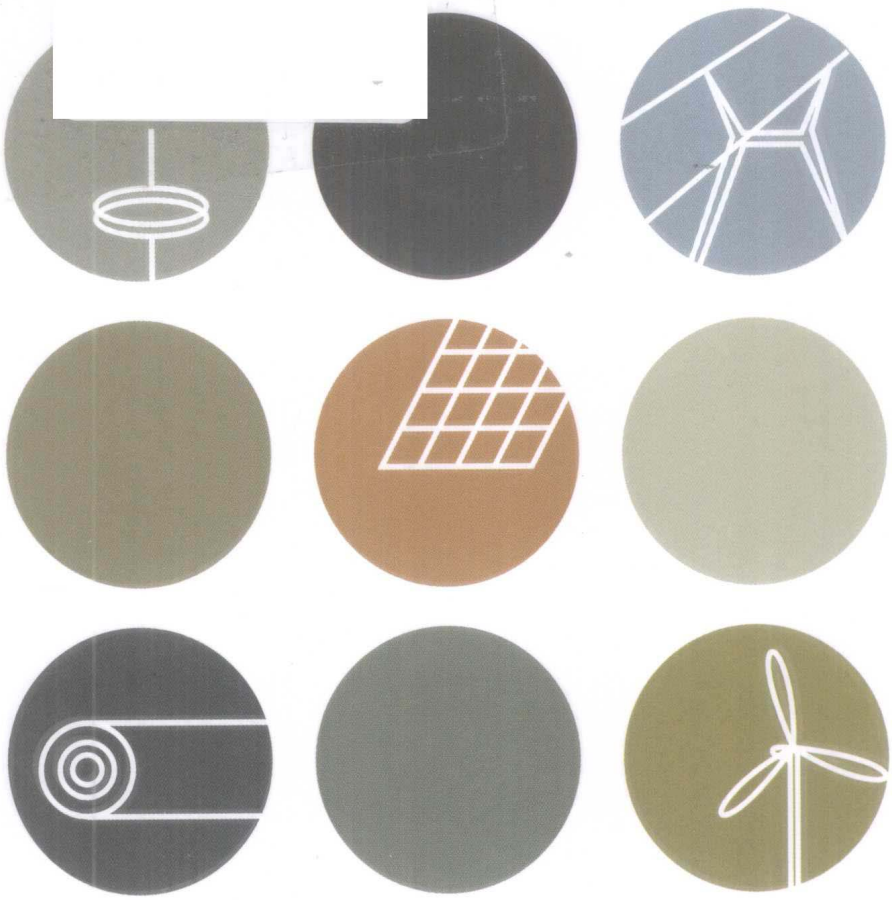
DSP及其电气与自动化工程应用

徐科军 陶维青 汪海宁 等编著



北京航空航天大学出版社

DSP及其电气与自动化工程应用

徐科军　陶维青　汪海宁　等编著

北京航空航天大学出版社

内 容 简 介

TI公司C2000系列DSP不仅具有一般DSP芯片高速的数据处理能力，而且与单片机一样在片内集成了丰富的外设，具备很强的控制能力，特别适用于先进传感技术、电机控制和数字电源等领域。本书共分6章，以作者的科研工作为基础，以实际应用为目的，通过关键技术和系统实例，较为系统和全面地介绍了C2000系列DSP在自动化和电气工程中的应用。

本书可供从事自动控制、电气工程、计算机应用和仪器仪表等专业的科研和工程技术人员参考，也可以作为相关专业本科生和研究生的参考书。

图书在版编目(CIP)数据

DSP及其电气与自动化工程应用/徐科军等编著. --
北京：北京航空航天大学出版社，2010.9
ISBN 978-7-5124-0126-6

Ⅰ. ①D… Ⅱ. ①徐… Ⅲ. ①数字信号—信号处理—应用—电气工程—自动化技术 Ⅳ. ①TN911.72②TM

中国版本图书馆CIP数据核字(2010)第116562号

DSP及其电气与自动化工程应用
徐科军 陶维青 汪海宁 等编著
责任编辑 李松山
*
北京航空航天大学出版社出版发行
北京市海淀区学院路37号(邮编100191) http://www.buaapress.com.cn
发行部电话：(010)82317024 传真：(010)82328026
读者信箱：emsbook@gmail.com 邮购电话：(010)82316936
北京时代华都印刷有限公司印装 各地书店经销
*
开本：787 mm×1 092 mm 1/16 印张：27.25 字数：698千字
2010年9月第1版 2010年9月第1次印刷 印数：5 000册
ISBN 978-7-5124-0126-6 定价：49.00元

前言

TI(美国德州仪器)公司的C2000系列DSP(数字信号处理器)不仅具有一般DSP所拥有的高速的数据处理能力,而且与单片机一样,在片内集成了丰富的外设,具备单片机所拥有的事务性处理能力,所以,被称为数字信号控制器(DSC),特别适合于先进传感技术、电机控制和数字电源等方面的应用。近年来,合肥工业大学电气与自动化工程学院将TI公司C2000系列DSP应用于传感器与自动化仪表、电气传动、电力系统、新能源利用和电机控制等方面,完成国家863计划、国家科技支撑计划、国家自然科学基金、省部委基金和企业委托项目,取得了一系列科研成果,很多成果已经被有关单位采用,部分成果已经在企业投入生产应用,取得了较好的社会和经济效益。本书就是在这样的背景下,以我们学院的科研工作为基础,以实际应用为目的,通过关键技术和系统实例,较为系统和全面地介绍了TI公司C2000系列DSP在自动化和电气工程中的应用。

全书共分为6章。第1章介绍C2000系列DSP在传感器与自动化仪表中的应用,第2章介绍C2000系列DSP在电气传动中的应用,第3章介绍C2000系列DSP在太阳能开发中的应用,第4章介绍C2000系列DSP在风能利用中的应用,第5章介绍C2000系列DSP在电力系统监控中的应用,第6章介绍C2000系列DSP在电机控制中的应用。每章的结构大致为:概述部分,简要介绍应用DSP技术的必要性;DSP应用中的若干关键技术和共性技术;若干应用系统举例。为了方便读者阅读,书中给出了部分电路的原理图、部分程序的流程图和程序源代码,并加以注释。考虑到介绍TI公司C2000系列DSP工作原理、体系结构、指令系统和开发工具的书籍已经出版了较多,所以,在本书中没有安排类似C2000系列DSP概述的章节。

本书的作者均为合肥工业大学电气与自动化工程学院的教师。第1章由徐科军编写,第2章由杜燕编写,第3章由汪海宁编写,第4章由杨淑英编写,第5章由陶维青编写,第6章由王海欣编写。苏建徽教授审阅了第2章和第3章,张兴教授审阅了第4章。本书由徐科军策划。

合肥工业大学电气与自动化工程学院在C2000系列DSP的应用中,得到TI公司的大力支持,在此表示感谢!

由于DSP技术发展非常迅速,我们所做的工作有限,书中难免存在疏漏和不足之处,敬请读者批评指正。有兴趣的读者,可发送邮件到dsplab@hfut.edu.cn与作者进一步交流,也可以发送邮件到emsbook@gmail.com与本书策划编辑进行交流。

作　者
于合肥工业大学电气与自动化工程学院
2010年8月

目　录

第1章

DSP在传感器与自动化仪表中的应用

1.1 概 述

传感器是将非电量转换成电量的器件和装置，是仪器仪表的核心部件，在自动化工程和电气工程中担任温度、压力、物位、流量、成分、电压、电流等的检测任务，为实现生产的自动控制和电气设备的正常运行提供必要的数据和信息。随着工业生产的发展和科学技术的进步，人们对检测提出了更高的要求，即要求能从含有噪声的信号中提取更多的信息，要求能够测试非电量的动态变化过程。为此，就必须将一些新方法应用于传感器输出信号的处理中和传感器本身特性的校正中。而DSP具有实现这些新方法的显著优势——运算速度快、体积小、含有丰富的外设。所以，我们将DSP技术用于传感器与检测技术，并取得较好效果。本章将介绍DSP在传感器与自动化仪表中的应用情况。首先介绍DSP在传感器与自动化仪表应用中的关键技术，如采集数据、频谱分析、频率测量、数字滤波、人机接口和输出电路，然后介绍3个应用系统：数字涡街流量计、传感器动态校正系统和数字科氏质量流量计。

1.2 模拟信号采集

在电气与自动化工程中，电压和电流信号一般都是以模拟信号的形式出现的。另外，像温度、压力、物位、流量等过程量，经过传感器检测后，大部分也为模拟信号。为了对电气与自动化工程中的这些重要参数进行数字信号处理和控制，就需要采用模数转换器(ADC)将这些模拟量转换成数字量。本节介绍如何用DSP来实现模拟量的采集和转换。由于C2000系列DSP中内置了ADC模块，当对数据采集的精度要求不高时，可以采用DSP内部的ADC。但是，当内置ADC的精度不满足有些应用场合的要求时，就必须配置外部的ADC。所以，本节介绍3个例子：用TMS320LF2407A内置的ADC进行数据采集，用TMS320F2812内置的ADC进行数据采集，用外置的ADC进行数据采集。

1.2.1 基于 TMS320LF2407A 内置 ADC 的数据采集

1. 内置 ADC 模块简介

TMS320LF2407A 具有内置采样和保持器(S/H)的 10 位 ADC,具有最短达 375 ns 的转换时间(S/H+转换);一共具有 16 个模拟输入通道(ADCIN0～ACDIN15)。A/D 的转换可以采用多个触发源启动转换(SOC),包括:软件立即启动(用 SOC SEQn 位);事件管理器 A(在 EVA 中有多个事件源可以启动转换)或者事件管理器 B(在 EVB 中有多个事件源可以启动转换),也可以使用外部 ADCSOC 引脚启动。芯片具有输入通道的自动排序能力,一次最多可执行 16 个通道的"自动转换",而每次要转换的通道可以通过编程来选择。排序器可工作在启动/停止模式下,允许多个按时间排序的触发源同步转换,转换结果存储在可单独访问的 16 个结果寄存器(RESULT0～RESULT15)中。采样保持时间可以独立设置,用于适应不同内阻的信号源[1]。

2. 硬件原理图

在图 1.2.1 中,模拟信号从 J5(音频信号输入口)或者 J6 输入,放大器 U12A 与周围电路构成的加法器,在放大器输入端 2 的信号与在输入端 3 的信号经过放大器叠加,在 AD2 处送至 DSP 内部的 ADC 进行 A/D 转换。AD2 之后的两个二极管是过压保护,使电压在 0～3.3 V 之间。

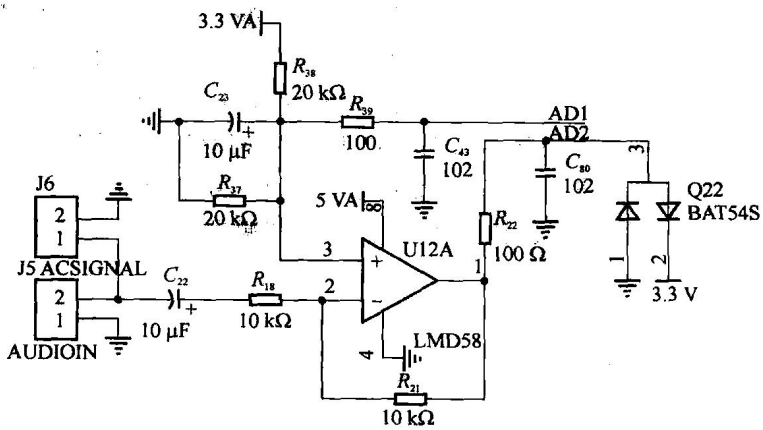


图 1.2.1 AD 采样电路硬件原理图

放大器正向输入端 3 的电压为

$$V_3=\frac{R_{37}}{R_{37}+R_{38}}3.3\ \text{V}=1.65\ \text{V}$$

放大器输出端 1 的电压为

$$V_1=V_3-V_2=1.65-V_2$$

这相当于把输入的正弦信号抬高了 1.65 V,使 AD2 的值在 0～3.3 V 之间。

3. 软件流程图

AD 采样流程图如图 1.2.2 所示。程序开始是关总中断，然后进行系统初始化，使能 EVA、ADC 模块的时钟。定时器 1 初始化，设置周期中断标志触发 A/D 转换。ADC 模块的初始化，设置排序器工作在启动/停止模式下，双排序器工作模式。初始化定时器 1 的周期寄存器，连续递增计数，给计数器赋初值。规定采样通道的顺序以及每次采样的通道数。然后，开启定时器 1，计数器开始工作。当计数器的值等于周期寄存器的值时重载。空循环，等待 AD 中断。一旦确定是 AD 中断，就转入中断服务子程序。从 PIVR(外设中断向量寄存器)中读取中断向量地址，从而转到相应的中断服务子程序的地址入口。将 AD 采样结果保存在结果寄存器的高 10 位。清除排序器中断位，复位排序器以使其指向 CONV00 处等待触发。开总中断，中断返回继续循环。

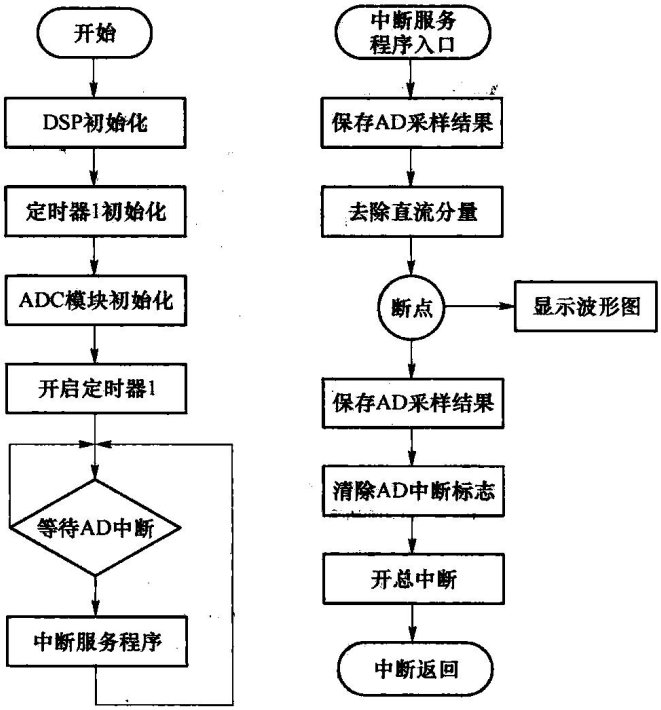


图 1.2.2　软件流程图

4. 源程序及注释

```
/*********************************************************************/
//外部晶振为 20 MHz
//AD 采集试验，采用排序器 SEQ1 来完成
//采用周期中断标志位触发 ADC，0.1 ms 触发一次 AD 采样
/*********************************************************************/
#include "regs2407.h"          //各种寄存器的定义
#include "Memcpy.h"
/************  SETUP for the MCRA Register  ************/
#define MCRA15                  0  // 0:IOPB7   1:TCLKIN
```

```
#define MCRA14                  0  // 0:IOPB6  1:TDIR
#define MCRA13                  0  // 0:IOPB5  1:T2PWM
#define MCRA12                  0  // 0:IOPB4  1:T1PWM
#define MCRA11                  0  // 0:IOPB3  1:PWM6
#define MCRA10                  0  // 0:IOPB2  1:PWM5
#define MCRA9                   0  // 0:IOPB1  1:PWM4
#define MCRA8                   0  // 0:IOPB0  1:PWM3
#define MCRA7                   0  // 0:IOPA7  1:PWM2
#define MCRA6                   0  // 0:IOPA6  1:PWM1
#define MCRA5                   0  // 0:IOPA5  1:CAP3
#define MCRA4                   0  // 0:IOPA4  1:CAP2/QEP2
#define MCRA3                   0  // 0:IOPA3  1:CAP1/QEP1
#define MCRA2                   0  // 0:IOPA2  1:XINT1, 该位无关紧要
#define MCRA1                   0  // 0:IOPA1  1:SCIRXD
#define MCRA0                   0  // 0:IOPA0  1:SCITXD
/*************************************************************/
/************** SETUP for the MCRB-Register  **************/
#define MCRB9                   0  // 0:IOPD1  1:XINT2/EXTSOC
#define MCRB8                   1  // 0:CKLKOUT 1:IOPD0, 该位无关紧要
#define MCRB7                   0  // 0:IOPC7  1:CANRX ,CAN 通信
#define MCRB6                   0  // 0:IOPC6  1:CANTX, CAN 通信
#define MCRB5                   0  // 0:IOPC5  1:SPISTE
#define MCRB4                   0  // 0:IOPC4  1:SPICLK
#define MCRB3                   0  // 0:IOPC3  1:SPISOMI
#define MCRB2                   0  // 0:IOPC2  1:SPISIMO
#define MCRB1                   1  // 0:IOPC1  1:BIO
#define MCRB0                   1  // 0:IOPC0  1:W/R
/*************************************************************/
/************** SETUP for the MCRC-Register  **************/
#define MCRC13                  0  // 0:IOPF5  1:TCLKIN2
#define MCRC12                  0  // 0:IOPF4  1:TDIR2
#define MCRC11                  0  // 0:IOPF3  1:T4PWM/T4CMP
#define MCRC10                  0  // 0:IOPF2  1:T3PWM/T3CMP
#define MCRC9                   0  // 0:IOPF1  1:CAP6
#define MCRC8                   0  // 0:IOPF0  1:CAP5/QEP3
#define MCRC7                   0  // 0:IOPE7  1:CAP4/QEP2
#define MCRC6                   0  // 0:IOPE6  1 :  PWM12
#define MCRC5                   0  // 0:IOPE5  1:PWM11
#define MCRC4                   0  // 0:IOPE4  1 :  PWM10
#define MCRC3                   0  // 0:IOPE3  1:PWM9
#define MCRC2                   0  // 0:IOPE2  1 :  PWM8
#define MCRC1                   0  // 0:IOPE1  1 :  PWM7
#define MCRC0                   0  // 0:IOPE0  1:CLKOUT
/*************************************************************/
/************** SETUP for the WDCR-Register  **************/
#define WDDIS                   1  // 0:看门狗被使能 1:看门狗被禁止
#define WDCHK2                  1  // 0:系统复位 1:正常运行
#define WDCHK1                  0  // 0:正常运行 1:系统复位
#define WDCHK0                  1  // 0:系统复位 1:正常运行
#define WDSP                    7  //看门狗预定标位选择 7:/64
```

```
/*************************************************************************/
/*************  SETUP for the SCSR1-Register  ********************/
#define CLKSRC                  0  // 0:输出 CPU 的时钟 20 MHz
#define LPM                     0  // 0:CPU 进入 IDLE1 模式
#define CLK_PS                  1  // 001:锁相环时钟预定标  ×2
#define ADC_CLKEN               1  // 1:使能 ADC 模块
#define SCI_CLKEN               0  // 0:禁止 SCI 模块
#define SPI_CLKEN               0  // 0:禁止 SPI 模块
#define CAN_CLKEN               0  // 0:禁止 CAN 模块
#define EVB_CLKEN               0  // 0:禁止 EVB 模块
#define EVA_CLKEN               1  // 1:GPT1 初始化 ADC
#define ILLADR                  1  // 1:无效地址检测位
/*************************************************************************/
/*************  SETUP for the ADCTRL1-Register  ******************/
#define RESET                   0  // 14:1 Resets entire ADC Module
#define SOFTFREE                2  // 13-12:10 complete ADC before halt
#define ACQ_PS                  7  // 11-8:Acquisition time 16 x TCLK
#define CPS                     1  // 7:1 ADC logic Clock = CLK/2
#define CONT_RUN                0  // 6 :0 Continuous run
#define INT_PRI                 0  // 5 :0 High ADC interrupt priority
#define SEQ_CASC                0  // 4 :0 Dual-Sequencer mode
#define CAL_ENA                 0  // 3 :0 Calibration mode disabled
#define BRG_ENA                 0  // 2    :0 Full reference Volt. to ADC
#define HILO                    0  // 1    :0 VREFLO as Test Voltage
#define STEST_ENA               0  // 0    :0 Self-Test mode disabled
/*************************************************************************/
/*************  SETUP for the ADCTRL2-Register ******************/
#define EVB_SOC_SEQ             0  // 15   :1 EVB starts Cascaded Sequ.
#define RST_SEQ1                0  // 14   :1 Reset Sequencer 1
#define SOC_SEQ1                0  // 13   :0 Clears a pending SOC trig
#define INT_ENA_SEQ1            1  // 11-10:1 Interrupt Mode 1
#define  EVA_SOC_SEQ1           1  // 8:1 EVA starts Sequencer1
#define  EXT_SOC_SEQ1           0  // 7:1 ADCSOC Pin starts Sequencer1
#define  RST_SEQ2               0  // 6:1 Reset Sequencer 2
#define  SOC_SEQ2               0  // 5:0 Clears a pending SOC trig
#define  INT_ENA_SEQ2           0  // 3-2:0 Interrupt disabled
#define  EVB_SOC_SEQ2           0  // 8:1 EVB starts Sequencer1
/*************  SETUP for the GPTCON-Register ********************/
#define GPTCONA_T2TOADC         0  // 10-9 : 0 no ADC-Start by GPT2-Event
#define GPTCONA_T1TOADC         2  // 8-7 : 2 ADC-Start by GPT1-Event
#define GPTCONA_TCOMPOE         0  // 6:0 disable all 2 GPT compare outs
#define GPTCONA_T2PIN           0  //3-2: Polar of GPT2 comp out = forced low
#define GPTCONA_T1PIN           0  //1-0 : Pol. of GPT1 comp out = forced low
/*************************************************************************/
/*************  SETUP for the T1CON-Register ********************/
#define T1CON_FREESOFT          0  // 15-14: 0 stop on JTAG suspend
#define T1CON_TMODE             2  // 12-11: 2 Continuous up counting
#define T1CON_TPS               3  // 10-8 : 3 CPUCLK/8
#define T1CON_TENABLE           1  // 6 : 1 Timer1 enable
#define T1CON_TCLKS             0  // 5-4 : 0 Clock source: internal
```

```
#define T1CON_TCLD          1   // 3-2: 1 reload when 0 or = T1PR
#define T1CON_TECMPR        0    // 1 : disable timer compare
/*****************************************************************/
/************** SETUP for the WSGR-Register  ********************/
#define BVIS                0   // 10-9 : 00 Bus visibility OFF
#define ISWS                0   // 8-6 : 000 0 Waitstates for IO
#define DSWS                0   // 5-3 : 000 0 Waitstates data
#define PSWS                0   // 2-0 : 000 0 Waitstaes code
/************** SETUP for the IMR-Register  ********************/
#define INT6                0   // 5 : Level INT6 is masked
#define INT5                0   // 4 : Level INT5 is masked
#define INT4                0   // 3 : Level INT4 is masked
#define INT3                0   // 2 : Level INT3 is masked
#define INT2                0   // 1 : Level INT2 is masked
#define INT1                1   // 0 : Level INT1 is unmasked
/*****************************************************************/
#define SYS_FRE (unsigned long)20000000   //20 MHz
#define SAMPLE_FRE 10000
//采样频率是 10 kHz,范围在(SYS_FRE/T1_PSCL)/65535 与(SYS_FRE/T1_PSCL)之间
#define T1_PSCL   (1 << T1CON_TPS)
#define T1_PRD    ((SYS_FRE/T1_PSCL)/SAMPLE_FRE )
//T1_PSCL 为 8,所以,定时器 1 的周期寄存器的值为 250
unsigned int ADC0_result[1000],ADC1_result[1000],ADC2_result[1000];
unsigned int ADC3_result[1000];
unsigned int i = 0,j = 0;
void c_dummy1(void)
{
    while(1);          // Dummy ISR used to trap spurious interrupts
}
interrupt void ADC_ISR(void)
{
    if((PIVR-0x0004) == 0)
//ADC 中断,PIVR 为外设中断向量寄存器,0004h 是 ADC 模块的中断地址向量
  {
    ADC0_result[i] = RESULT0 >> 6;
    ADC1_result[i] = RESULT1 >> 6;
    ADC2_result[i] = RESULT2 >> 6;//AD2
    ADC3_result[i] = RESULT3 >> 6;//AD10
//把结果保存到结果寄存器中,低 6 位为保留位
   i++;
    if(i >= 1000)
  {
    i = 0;AVE = 0;
    for(i = 0;i < 1000;i++)
      {AVE = AVE + ADC2_result[i];}
    if(i >= 1000)
      i = 0; AVE = AVE/1000 ;
    for(i = 0;i <= 1000;i++)
      {ADC2_result[i] = (ADC2_result[i] - AVE) * 100;}
  }
```

```
//将转换好的信号的直流分量去掉,并扩大输出信号的幅值,AVE 是直流量
      if(i >= 1000)
          i = 0;
//在此设断点,当计算 1 000 点后,完成去除直流分量的处理
      ADCTRL2 | = 0x0200;          //清除 ADC 排序器 1 中断标志位
      ADCTRL2 | = 0x4000;
//在启动/停止模式下,排序器必须复位以使排序器指针指到 CONV00
    }
}
void main(void)
{
    asm (" setc INTM");            //关闭总中断
    asm (" clrc SXM");             //禁止符号位扩展
    asm (" clrc OVM");             //累加器结果正常溢出
    asm (" clrc CNF");             //B0 块映射为数据存储器
    WSGR = ((BVIS << 9) + (ISWS << 6) + (DSWS << 3) + PSWS);
    //设置等待状态产生器寄存器 WSGR
    WDCR = ((WDDIS << 6) + (WDCHK2 << 5) + (WDCHK1 << 4) + (WDCHK0 << 3) + WDSP
    //初始化 WD 定时器控制寄存器。看门狗被禁止,系统正常运行,64 分频
    SCSR1 = ((CLKSRC << 14) + (LPM << 12) + (CLK_PS << 9) + (ADC_CLKEN << 7) +
            (SCI_CLKEN << 6) + (SPI_CLKEN << 5) + (CAN_CLKEN << 4) +
            (EVB_CLKEN << 3) + (EVA_CLKEN << 2) + ILLADR);
    //初始化系统寄存器。输出 CPU 时钟,使能 ADC 模块、EVA 模块的时钟
    MCRC = ((MCRC13 << 13) + (MCRC12 << 12) + (MCRC11 << 11) + (MCRC10 << 10) +
            (MCRC9 << 9) + (MCRC8 << 8) + (MCRC7 << 7) + (MCRC6 << 6) +
            (MCRC5 << 5) + (MCRC4 << 4) + (MCRC3 << 3) + (MCRC2 << 2) +
            (MCRC1 << 1) + MCRC0);
    //初始化 I/O 复用寄存器 C
    MCRB = ((MCRB9 << 9) + (MCRB8 << 8) +
            (MCRB7 << 7) + (MCRB6 << 6) + (MCRB5 << 5) + (MCRB4 << 4) +
            (MCRB3 << 3) + (MCRB2 << 2) + (MCRB1 << 1) + MCRB0);
    //初始化 I/O 复用寄存器 B
    MCRA = ((MCRA15 << 15) + (MCRA14 << 14) + (MCRA13 << 13) + (MCRA12 << 12) +
            (MCRA11 << 11) + (MCRA10 << 10) + (MCRA9 << 9) + (MCRA8 << 8) +
            (MCRA7 << 7) + (MCRA6 << 6) + (MCRA5 << 5) + (MCRA4 << 4) +
            (MCRA3 << 3) + (MCRA2 << 2) + (MCRA1 << 1) + MCRA0);
    //初始化 I/O 复用寄存器 A
    GPTCONA = ((GPTCONA_T2TOADC << 9) + (GPTCONA_T1TOADC << 7) +
            (GPTCONA_TCOMPOE << 6) + (GPTCONA_T2PIN << 2) +
            (GPTCONA_T1PIN);
    //初始化全局通用定时器控制寄存器,设置周期中断标志启动 ADC
    T1PR = T1_PRD;                 //初始化定时器 1 的周期寄存器
    T1CNT = 0x0000;                //给计数寄存器赋初值
    T1CON = ((T1CON_FREESOFT << 14) +
            (T1CON_TMODE << 11) +
            (T1CON_TPS << 8) +
            (T1CON_TCLKS << 4) +
            (T1CON_TCLD << 2) +
            (T1CON_TECMPR << 1));
    //初始化定时器 1 的控制寄存器,连续增计数,计数器值等于零或等于周期寄存器值时重载
```