

Cortex-M4原理与实践实验报告

题目： ADC操作实验

作者姓名 章金皓

指导教师 付明磊

专业班级 自动化1901

学 院 信息工程学院

提交日期 2021年11月日

一、实验目的

1.理解TM4C1294KCPDT的ADC模块原理。

2.理解ADC如何启动，如何判断ADC转换结束。

3.学会ADC模块的设计

二、实验设备

1.计算机一台，操作系统为WindosXP或Windos7，装有CCSv6.0软件。

2.EK-TM4C1294XL实验开发板一块。

3.USB连接线一条。

三、实验原理和流程

1.TM4C1294 ADC模块简介

 TM4C1294 微控制器内置两个相同的模数转换器（ADC）模块。这两个模块（ADC0和ADC1）共用相同的20 个模拟输入通道。两个ADC 模块的工作相互独立，因此可同时执行不同的采样序列、随时对任一模拟输入通道进行采样、并各自产生不同的中断和触发事件。图1给出了ADC 模块中控制寄存器及数据寄存器的配置情况。

图 1 ADC模块框图

2.本实验原理图

 滚轮的电阻值为0-10K，AD通道为通道0（PD7），可以通过拨动滚轮得到不同的输入电压。

图 3 ADC电路原理图（2）

图 2 ADC电路原理图（1）

AD 通道为通道4（PE3），PE3口对应的是温度传感器实验，外接不同的温度使传感器产生不同的电压，电压与温度的关系如式x-x，通过AD 测得当前的电压，然后再根据公式转换为当前的温度。

温度传感器的监测范围为-55度到+130度，

流程图：



四、实验代码、注释

实验代码：

#include "stdint.h"

#include "stdbool.h"

#include "driverlib/debug.h"//调试

#include "inc/hw\_types.h"//数据类型设置，寄存器访问封装

#include "inc/hw\_memmap.h"//基址

#include "driverlib/fpu.h"

#include "driverlib/sysctl.h"//使能寄存器

#include "driverlib/rom.h"

#include "driverlib/pin\_map.h"

#include "grlib/grlib.h"

#include "driverlib/gpio.h"

#include "driverlib/adc.h"

#include "inc/hw\_gpio.h"

#include "inc/tm4c1294ncpdt.h"

#include "driverlib/uart.h"

#include "utils/uartstdio.h"

volatile uint32\_t ui32SysClock;

void InitConsole(void)

{

// 由于UART0使用PA0,PA1两个引脚，因此需要使能GPIOA模块

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

// 因为有引脚复用，所以要对PA0和PA1两个引脚的功能进行选择

// 这里将它们选择为执行UART0模块的功能

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

// 对于PA0和PA1两个引脚，在将它们作为UART功能使用之前，需要对它们做一些有关UART的配置

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

// UART的标准初始化

UARTStdioConfig(0, 115200, ui32SysClock);

}

int main(void)

{

uint32\_t ulADC0\_Value;

//设置系统时钟

ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN |SYSCTL\_USE\_PLL |SYSCTL\_CFG\_VCO\_480), 16000000);

//初始化UART

InitConsole();

// 初始化ADC0/PE3为温度测量

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

GPIOPinTypeADC(GPIO\_PORTE\_BASE, GPIO\_PIN\_3);//PE3

// 配置ADC采集序列

ADCSequenceConfigure(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

// 对应通道0

ADCSequenceStepConfigure(ADC0\_BASE, 3, 0, ADC\_CTL\_CH2 | ADC\_CTL\_END | ADC\_CTL\_IE);

// 使能ADC采集序列

ADCSequenceEnable(ADC0\_BASE, 3);

ADCIntClear(ADC0\_BASE, 3);

while(1)

{

// 触发采集

ADCProcessorTrigger(ADC0\_BASE, 3);

// 等待采集结束

while(!ADCIntStatus(ADC0\_BASE, 3, false)) ;

// 获取采集结果

ADCSequenceDataGet(ADC0\_BASE, 3, &ulADC0\_Value);

SysCtlDelay(2\*50000000/3000);

int temp=(1.8663-ulADC0\_Value\*3.3/4096)/0.01169;

UARTprintf("%04d \n ", temp );

}

实验现象：

五、思考题

4.1 修改程序，将程序修改为检测滚轮的采样程序。

设计思路：将滚轮对应端口进行使能与初始化，选择检测滚轮相应的通道4（PD7）。

实现代码：

#include "stdint.h"

#include "stdbool.h"

#include "driverlib/debug.h"//调试

#include "inc/hw\_types.h"//数据类型设置，寄存器访问封装

#include "inc/hw\_memmap.h"//基址

#include "driverlib/fpu.h"

#include "driverlib/sysctl.h"//使能寄存器

#include "driverlib/rom.h"

#include "driverlib/pin\_map.h"

#include "grlib/grlib.h"

#include "driverlib/gpio.h"

#include "driverlib/adc.h"

#include "inc/hw\_gpio.h"

#include "inc/tm4c1294ncpdt.h"

#include "driverlib/uart.h"

#include "utils/uartstdio.h"

volatile uint32\_t ui32SysClock;

void InitConsole(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

UARTStdioConfig(0, 115200, ui32SysClock);

}

int main(void)

{

uint32\_t ulADC0\_Value;

//设置系统时钟

ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN |SYSCTL\_USE\_PLL |SYSCTL\_CFG\_VCO\_480), 16000000);

//初始化UART

InitConsole();

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

GPIOPinTypeADC(GPIO\_PORTD\_BASE, GPIO\_PIN\_7);//PD7为滚轮输入 ADCSequenceConfigure(ADC0\_BASE, 0, ADC\_TRIGGER\_PROCESSOR, 0);

// Sample channel 0 (ADC\_CTL\_CH0) in

ADCSequenceStepConfigure(ADC0\_BASE, 3, 0, ADC\_CTL\_CH4 | ADC\_CTL\_END | ADC\_CTL\_IE);

// 使能ADC采集序列

ADCSequenceEnable(ADC0\_BASE, 3);

ADCIntClear(ADC0\_BASE, 3);

// ADCIntEnable(ADC0\_BASE, 0);

while(1)

{

// 触发采集

ADCProcessorTrigger(ADC0\_BASE, 3);

// 等待采集结束

while(!ADCIntStatus(ADC0\_BASE, 3, false)) ;

// 获取采集结果

ADCSequenceDataGet(ADC0\_BASE, 3, &ulADC0\_Value);

SysCtlDelay(2\*50000000/3000);

int k=0;

k=ulADC0\_Value\*270/4096;

UARTprintf("滚轮值：%04d 角度：%d \n ",ulADC0\_Value,k);

}

}

实验现象：

4.2修改程序，将程序修改为检测温度并显示到米字管的程序。

（为了减少报告长度，初始化参考实验五）

实现代码：

#define Delay(x) SysCtlDelay(x\*10000) //

#define DELAYTIME 50

volatile uint32\_t ui32SysClock;

int

main(void)

{

//设置系统时钟

ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN |SYSCTL\_USE\_PLL |SYSCTL\_CFG\_VCO\_480), 16000000);

I2C0GPIOBEnable();//配置I2C0模块的IO引脚

I2C0DeviceInit();//配置PCA9557芯片中连接米字管的各引脚为输出

// 由于UART0使用PA0,PA1两个引脚，因此需要使能GPIOA模块

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

// 因为有引脚复用，所以要对PA0和PA1两个引脚的功能进行选择

// 这里将它们选择为执行UART0模块的功能

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

// 对于PA0和PA1两个引脚，在将它们作为UART功能使用之前，需要对它们做一些有关UART的配置

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

// UART的标准初始化

UARTStdioConfig(0, 115200, ui32SysClock);

uint32\_t ulADC0\_Value;

//设置系统时钟

// 初始化ADC0/PE3为温度测量

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

GPIOPinTypeADC(GPIO\_PORTE\_BASE, GPIO\_PIN\_3);

// 配置ADC采集序列

ADCSequenceConfigure(ADC0\_BASE, 0, ADC\_TRIGGER\_PROCESSOR, 0);

// 对应通道0

ADCSequenceStepConfigure(ADC0\_BASE, 3, 0, ADC\_CTL\_CH0 | ADC\_CTL\_END | ADC\_CTL\_IE);

// 使能ADC采集序列

ADCSequenceEnable(ADC0\_BASE, 3);

ADCIntClear(ADC0\_BASE, 3);

uint8\_t temp\_low = 0;

uint8\_t temp\_high = 0;

while(1)

{

// 触发采集

ADCProcessorTrigger(ADC0\_BASE, 3);

// 等待采集结束

while(!ADCIntStatus(ADC0\_BASE, 3, false)) ;

// 获取采集结果

ADCSequenceDataGet(ADC0\_BASE, 3, &ulADC0\_Value);

SysCtlDelay(2\*50000000/3000);

int temp=(1.8663-ulADC0\_Value\*3.3/4096)/0.01169;

UARTprintf("%04d \n ", temp );

temp\_high = (uint8\_t)(temp % 10);

temp\_low = (uint8\_t)(temp / 10 % 10);

//高位

I2C0TubeSelSet(0x3A);

I2C0TubeLowSet(tubeCodeTable[temp\_low][0]);

I2C0TubeHighSet(tubeCodeTable[temp\_low][1]);

Delay(1);

I2C0TubeSelSet(~0x00);

I2C0TubeLowSet(0x00);

I2C0TubeHighSet(0x00);

Delay(1);

//低位

I2C0TubeSelSet(0x36);

I2C0TubeLowSet(tubeCodeTable[temp\_high][0]); //0001 1010

I2C0TubeHighSet(tubeCodeTable[temp\_high][1]); //0010 0110

Delay(1);

I2C0TubeSelSet(~0x00);

I2C0TubeLowSet(0x00);

I2C0TubeHighSet(0x00);

Delay(1);

}

}

实验现象：



六、实验体会与心得

1）ADC的用途十分广泛，在自然环境中，信号大多以模拟量的形式存在，而ADC就是数字量和模拟量的桥梁，这就可以让单片机去很快得处理模拟信号，比如温度，长度，速度等。ADC实质是测量电压，所以只要能将待测量转换成电压量，就可以用ADC测量，经过转换即可求得待测量。

2）串口的使用也应规范，TXRX交叉接线(此次实验直接USB线代替)，发送和接收的波特率得保持一致。调试的时候可以使用ccs自带的窗口显示接收到的值，也可以用其他的串口调试助手来查看。同时我在实验的时候发现，下载口线接触不良时会影响接收到的数据。