**AD1Pmod and DA2Pmod in FreeRTOS**

Benjamin A. Blouin

*benjamin.blouin@temple.edu*

**Summary**

This lab is a rehash of the previous two labs using the ADC and DAC but doing the both using FreeRTOS. The example given in the lab includes a C-style program to be run as-is, which shows the functionality of the ADC and DAC data transfer. Then this program is implemented in FreeRTOS using a task for each component, passing the signal through. The last lab requirement implements a filtering task, the same filter designed from a previous lab. The ADC takes in a generated signal, pass the data through a queue to the filtering task, which then passes the filtered data to the DAC for output to an oscilloscope. The Digilent EEBoard is used as a waveform-generator and oscilloscope.

**Introduction**

**Discussion**

**Conclusions**

**Appendices**

**Video**

**Github**

**Code**

//

//

#include <stdio.h>

// FreeRTOS includes

#include <FreeRTOS.h>

#include <task.h>

#include <queue.h>

// Xilinx includes

#include <xparameters.h>

#include <xil\_io.h>

#include <xil\_printf.h>

#include <sleep.h>

#include <assert.h>

#include <stdlib.h>

//AD1Pmod from Address Editor in Vivado, first IP

#define AD1acq 0x43C00000 //AD1 acquisition - output

#define AD1dav 0x43C00004 //AD1 data available - input

#define AD1dat1 0x43C00008 //AD1 channel 1 data - input

//DAC2Pmod from Address Editor in Vivado, second IP

#define DA2acq 0x43C10000 //DA2 acquisition - output

#define DA2dav 0x43C10004 //DA2 data available - input

#define DA2dat1 0x43C10008 //DA2 channel 1 data - output

// tasks prototypes

static void prvAD1task(void\* pvParameters);

static void prvDA2task(void\* pvParameters);

static void prvFiltertask(void\* pvParameters);

// declare tasks handles

static TaskHandle\_t xAD1Task;

static TaskHandle\_t xDA2Task;

static TaskHandle\_t xFilterTask;

// declare queues

static QueueHandle\_t xTransferQueue = NULL;

// ADC and DAC global vars

int32\_t adcdav; //ADC data available

int32\_t dacdav; //DAC data available

int main(int argc, char\*\* argv) {

xTaskCreate(prvAD1task,

"AD1Task",

configMINIMAL\_STACK\_SIZE,

NULL,

tskIDLE\_PRIORITY+1,

&xAD1Task);

xTaskCreate(prvDA2task,

"DA2Task",

configMINIMAL\_STACK\_SIZE,

NULL,

tskIDLE\_PRIORITY+2,

&xDA2Task);

xTaskCreate(prvFiltertask,

"FilterTask",

configMINIMAL\_STACK\_SIZE,

NULL,

tskIDLE\_PRIORITY+3,

&xFilterTask);

// create the Queue that will be used to pass the data

xTransferQueue = xQueueCreate(1,

sizeof(int32\_t));

// make sure that the Queue was created

//

configASSERT(xTransferQueue);

// start the task Scheduler

vTaskStartScheduler();

while(1);

return 0;

}

static void prvAD1task(void\* pvParameters) {

static int32\_t data;

while(1) {

//ADC acquire

//

Xil\_Out32(AD1acq,1);

while (adcdav == 0) {

adcdav=Xil\_In32(AD1dav);

}

//ADC stop acquire

Xil\_Out32(AD1acq, 0);

//input ADC data

data = Xil\_In32(AD1dat1);

//wait for reset

//

while (adcdav==1) {

adcdav = Xil\_In32(AD1dav);

}

// pass the data to the queue

xQueueSend(xTransferQueue,

&data,

0UL);

vTaskResume(xFilterTask);

}

}

static void prvFiltertask(void\* pvParameters){

static int32\_t filter\_data;

static int32\_t filtered\_data;

static int32\_t buffer[4096];

static int8\_t init\_flag,first\_flag,continue\_flag,n;

while(1){

if(n==4){init\_flag=1;first\_flag=1;}

xQueueReceive(xTransferQueue,

&filter\_data,

portMAX\_DELAY);

if(init\_flag==0){

buffer[n]=filter\_data;

n++;

}

if(first\_flag==1 && n==4){

filtered\_data = buffer[0]+3\*buffer[1]+3\*buffer[2]+buffer[3]+4\*buffer[4];

first\_flag = 0;

continue\_flag = 1;

}

if(continue\_flag==1&&n==4){

buffer[4]=buffer[3];

buffer[3]=buffer[2];

buffer[2]=buffer[1];

buffer[1]=buffer[0];

buffer[0]=filter\_data;

filtered\_data = 0.2621\*(buffer[0]+3\*buffer[1]+3\*buffer[2]+buffer[3]+4\*buffer[4]);

}

xQueueSend(xTransferQueue,

&filtered\_data,

0UL);

vTaskResume(xDA2Task);

vTaskSuspend(NULL);

}

}

static void prvDA2task(void\* pvParameters) {

static int32\_t incoming\_data;

while(1) {

// get the data from the Queue

xQueueReceive(xTransferQueue,

&incoming\_data,

portMAX\_DELAY);

//output DAC data

//xil\_printf("data %u\n",incoming\_data);

Xil\_Out32(DA2dat1, incoming\_data);

//DAC acquire

Xil\_Out32(DA2acq,1);

//DAC data output?

while (dacdav==0) {

dacdav=Xil\_In32(DA2dav);

}

//stop DAC acquire

Xil\_Out32(DA2acq,0);

//wait for reset

while(dacdav==1) {

dacdav=Xil\_In32(DA2dav);

}

}

}