Parcial Materia Microcontroladores II

Universidad: Catolica Argentina

Alumno (nombre completo): Aaron Amaro Stafuza Michlig

Fecha: 29 de mayo 2020

Profesores: Ing. Ricardo Vecchio / Ing. Diego Colodrero

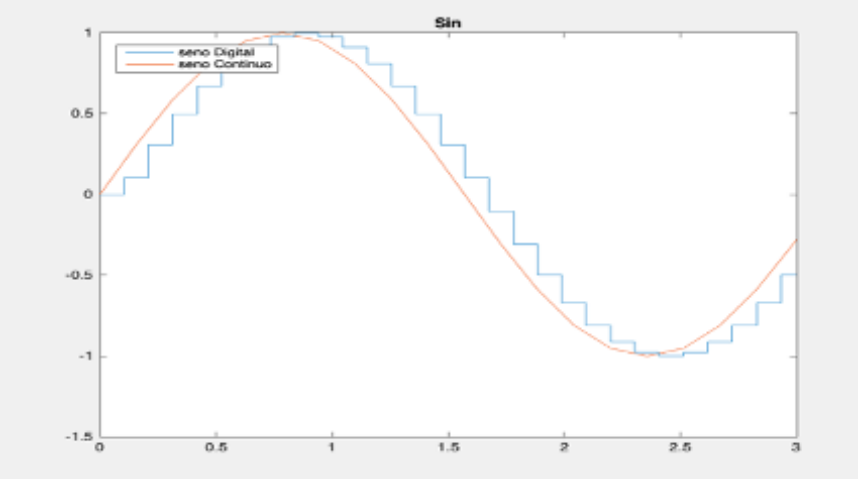
1) Dspic30f4013, modulo USART, escribir código para configurarlo a 9600 bits/s modo 8N1
2) Obtener los coeficientes numd y dend (utilizando comandos de matlab) de la función de transferencia discreta para implementar la función y= seno (2t) ante una exitación impulso unitario. Muestrear 30 veces por ciclo la señal sintetizada (utilizar 'zoh'). Mostrar la respuesta al escalón en un matlab (.m y .xls) (continua y discreta superpuesta)
3) Escribir un código para hallar el máximo de un vector utilizando las librerías del modulo DSP e imprimirlo.
   Vsrc={0.9 0.6 12.3 -9.4 -0.4 -0.03 0.904 1}
   Cuantos TCY lleva determinarlo (considerando exclusivamente a la función correspondiente del DSP)?
4) DSPIC33F: Los DMA Channels almacenan los datos del Periferico X antes de enviarlos a la DPSRAM. V / F explique.
5) DSPIC33F: La CPU se comunica con la SRAM mediante el X-bus únicamente. V / F
6) DSPIC33F: Que bit y de que registro hay que setear para iniciar un DMA request manual?
7) DSPIC33F: Una vez seteado el bit del punto 6)…Quien se encarga de limpia ese bit?
8) Escriba un programa para calcular cuántos TCY toma ejecutar la siguiente instrucción: **printf ("Yo jamas aprobare este examen\r\n");** e imprima el numero de TCY necesarios. Como mejoraría a este número de TCY utilizando el módulo DSP?
9) DSPIC33F: En que registro se encuentra y cuál es el bit que fija la cantidad de muestreos/conversión del A/D hasta que se genere un interrupt? Qué valor debe tener (en binario) para que genere una interrupción cada 12th secuencias de muestreos/conversión consecutivas?
10) DSPIC33F: Que sucede si fuerzo un valor cero al bit DONE del A/D mientras una conversión está en proceso?

NOTA: Escribir todo el código de los programas en el main() del IDE y pegar eso en el Word examen (no enviar el proyecto completo). Copy-pastear los resultados obtenidos en la ventana SIM Uart1 si corresponde y aclarar.
Pueden enviar el .m y el simulink del matlab o pegar sus resultados en un Word (aconsejo esto último porque son muy básicos).

Consejo: yo integraría todo en un Word y luego lo paso a pdf en https://www.ilovepdf.com/es pueden elegir en la web Word a pdf…

Punto 2)

```
freq=1/pi;
T=1/freq;
w=2*pi*freq;
muest=30;
Ts=T/muest;
t=linspace(0,pi,muest+1);
num=[w];
den=[1 0 w^2];
sys=tf(num,den);
[yc,tc]=impulse(sys);
[numd,dend]=c2dm(num,den,Ts,'zoh')
[y]=dimpulse((1/Ts)*numd,dend,muest+1);
stairs(t,y)
hold on
plot(tc,yc)
title('Sin')
xlim([0,3])
legend({'seno Digital','seno Continuo'},'Location','northwest')
format long
```




```
numd =

                   0   0.010926199633097   0.010926199633097

dend =

   1.000000000000000  -1.956295201467611   1.000000000000000
```

Punto 3)

```
#define MIPS 30
#include <p30F4013.h>         // si se cambia de DSC cambiar tb p30FXXXX.gld
linker
#include <stdio.h>
#include <string.h>
#include <dsp.h>
#include <time.h>           // tic toc
#include "terminal.h"         // hiperterminal colores
#include "delays.h"

_FOSC(CSW_FSCM_OFF & XT_PLL16);  //Run this project using an external crystal
                                 //routed via the PLL in 16x multiplier mode
                                 //For the 7.3728 MHz crystal we will derive a
                                 //throughput of 7.3728e+6*16/4 = 29.4
MIPS(Fcy)
                                 //,~33.9 nanoseconds instruction cycle
time(Tcy).
_FWDT(WDT_OFF);                  //Turn off the Watch-Dog Timer.
_FBORPOR(MCLR_EN & PWRT_OFF);    //Enable MCLR reset pin and turn off the
                                 //power-up timers.
_FGS(CODE_PROT_OFF);             //Disable Code Protection

////-----------------------------------------
clock_t start, stop;
fractional strb1[8] = {Q15(0.9/12.3), Q15(0.6/12.3),
Q15(12.3/12.3),Q15(-9.4/12.3),Q15(-0.4/12.3),Q15(-0.03/12.3),Q15(0.904/12.3),
Q15(1/12.3)};
fractional bandera;
float resultado;
int max;
void maxi_DSP(void);
int main(void) {
    maxi_DSP();
    return (EXIT_SUCCESS);
}
void maxi_DSP(void){
    static double count=0;
    start=clock();
    bandera=VectorMax(8,strb1,&max);
    stop=clock();       //Deja de contar TCY
    count=stop-start-19.0;
    resultado=Fract2Float(bandera);
    resultado=resultado*12.3;
    printf("\nEl maximo es: %.3f\n",resultado);
    printf("La demora del dsp es: %.1f TCY\n",count);
}
```

```
El maximo es: 12.300
La demora del dsp es: 63.0 TCY
```

PUNTO 5)

VERDADERO.

Punto 9)
Saque una captura del registro que es el siguiente:

REGISTER 21-2: ADxCON2: ADCx CONTROL REGISTER 2 (where x = 1 or 2)

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
VCFG<2:0>			—	—	CSCNA	CHPS<1:0>	
bit 15							bit 8

R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI<3:0>				BUFM	ALTS
bit 7							bit 0

El SMPI<3:0> lo que hace es espificarme cuantas muestras de adquisiones/ conversiones se terminaran antes que ocurra la interrupción.

El bit BUFS es un bit de estado que indica que mitad del bufer esta escribiendo actualmente el modulo ADC. Si se divide el buffer, la aplicación de software tiene mas tiempo para leer la información del buffer.

Punto 10)

REGISTER 21-1: ADxCON1: ADCx CONTROL REGISTER 1 (where x = 1 or 2)

R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
ADON	—	ADSIDL	ADDMABM	—	AD12B	FORM<1:0>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0 HC,HS	R/C-0 HC, HS
SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE
bit 7							bit 0

El bit DONE es un bit de estado de conversión del ADC.
Las configuraciones del mismo son:
En 1 se completa el ciclo de conversion de ADC.
En 0 la conversión del ADC no esta iniciada o se encuentra en proceso.

Desde el software puedo escribir un 0 para borrar el estado hecho. Si borramos este bit no afectara ninguna operación que se encuentre en progreso, lo que generará va a ser un borrado automático por hardware al inicio de una nueva conversión.

PUNTO 4)
FALSO.
Cada canal de un DMA se puede configurar para transferir datos desde un periférico a la DPSRAM o desde la DPSRAM a un periférico. Si se configura el bit de Dirección de transferencia (DIR) en DMA, los datos se leen desde el periférico y la escritura se dirige al desplazamiento de la dirección de memoria DPSRAM DMA.

Entonces lo que ocurre es que yo leo por el periférico y lo almaceno en DPSRAM.

PUNTO 1)

```
#include "xc.h"
#include <p30F4013.h>
#define MIPS 30
#define B9 LATBbits.LATB9
#define B10 LATBbits.LATB10
#define B11 LATBbits.LATB11
#define B12 LATBbits.LATB12

_FOSC(CSW_FSCM_OFF & FRC_PLL16);
_FWDT(WDT_OFF);

void configuart();
void delay_ms (unsigned long delay_count);
void delay_us (unsigned int delay_count);

int main(void)
{
    ADPCFG = 0xFFFB;
    TRISBbits.TRISB9 =0;
    TRISBbits.TRISB10 = 0;
    TRISBbits.TRISB11 = 0;
    TRISBbits.TRISB12 = 0;
    TRISDbits.TRISD8 = 0;
    TRISDbits.TRISD9 = 0;
    TRISDbits.TRISD2 = 0;
    delay_ms(2000); //esto solo es para encender un led de
    B9=1;           //verificacion
    delay_ms(2000);
    B9=0;
    configuart();
    IEC0bits.U1RXIE = 0;
    IEC0bits.U1TXIE = 1;
    while (1){
        U1TXREG='h';
        delay_us(5000);
        U1TXREG='o';
        delay_us(5000);
        U1TXREG='l';
        delay_us(5000);
        U1TXREG='a';
        delay_us(5000);
        U1TXREG='\n';
        delay_us(5000);
        U1TXREG='\r';
        delay_us(5000);
        U1TXREG='\r';
        delay_us(5000);
    }
    return 0;
}

 void __attribute__((interrupt, no_auto_psv)) _U1TXInterrupt( void ) // la
interrupcion por si queremos hacer algo, limpiamos la
{                                                                    // el
interrupt flag
    IFS0bits.U1TXIF=0;
}
```

```
void configuart(void)
{
    OSCTUN=0;
    RCONbits.SWDTEN=0;
    U1MODEbits.UARTEN = 0;// Disable UART
    U1MODEbits.STSEL = 0; // 1-stop bit
    U1MODEbits.PDSEL = 0; // No Parity, 8-data bits
    U1MODEbits.ABAUD = 0; // Auto-Baud Disabled
    U1BRG = 194;          //9600
    U1STAbits.UTXISEL = 0;         // Interrupcion cuando mandamos un
caracter
    U1STAbits.URXISEL = 0;         // Interrupcion cuando recibimos un
caracter
    U1MODEbits.UARTEN = 1;         // Enable UART
    U1STAbits.UTXEN = 1;           // Enable UART Tx, solo activamos el envio
}

void delay_ms (unsigned long delay_count)

      {
      delay_count=delay_count*MIPS*100;
      while(delay_count--);
      }


void delay_us (unsigned int delay_count)

      {
      delay_count=delay_count*MIPS/4;

      while(delay_count--);
      }
```

PUNTO 6)

Hay que setear el registro DMAxREQ, el bit 15 FORCE.

REGISTER 8-2: DMAxREQ: DMA CHANNEL x IRQ SELECT REGISTER

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
FORCE(1)	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	IRQSEL6(2)	IRQSEL5(2)	IRQSEL4(2)	IRQSEL3(2)	IRQSEL2(2)	IRQSEL1(2)	IRQSEL0(2)
bit 7							bit 0

FORCE: bit de transferencia de DMA de fuerza.
1 = Forzar una sola transferencia DMA (modo manual)
0 = Iniciación automática de transferencia DMA por solicitud DMA.

Entonces para iniciar un DMA request, seteo el FORCE en 0.

PUNTO 7)

El usuario no puede borrar el bit FORCE. Una vez completada la transferencia DMA el hardware se encarga de limpiar el bit FORCE.