

Informe de laboratorios de

Circuitos Digitales 3

Profesor: Laprovita, Agustín

Alumnos: Alberoni, Cristian

Gutiérrez, Agustín

Fecha: 19 de Junio 2014

1. Índice
2. Introducción

En el presente informe se detalla la resolución de los laboratorios propuestos en la materia Circuitos Digitales 3 de la carrera Ingeniería en Electrónica, los cuales tienen como objetivo que los alumnos adquieran experiencia real en la programación de plataformas basadas en microcontroladores, adquieran la habilidad de consultar manuales y hojas de datos de dispositivos programables y conocimientos en aspectos como la utilización de puertos de E/S digital y su implementación, utilizar una interfaz de comunicación de datos digitales serial para intercambiar información entre distintos sistemas computacionales en forma segura, configurar y controlar módulos ADC embebidos en dispositivos programables y por ultimo comprender el principio de funcionamiento de los bloques responsables de la generación y captación de señales de control en tiempo real y sincronismos.

1. Descripción de plataforma utilizada

Para la implementación de los laboratorios se utilizó el EduKit Ez430 Board (Ver Figura 3.1), él cual es una interfaz para el microcontrolador MSP430f2274 de Texas Instruments (Ver Figura 3.2), este microcontrolador consiste en varios recursos con diferentes conjuntos de periféricos específicos para diversas aplicaciones.

Su arquitectura combinada con los cinco modos de bajo consumo está optimizado para lograr la vida extendida de la batería en aplicaciones de mediciones portátiles.

El dispositivo presenta un CPU de 16 bits RISC, registros de 16 bits y generadores constantes que contribuyen a la máxima eficiencia del código. La serie MSP430x22xx es un microcontrolador con dos temporizadores integrados de 16 bits, una interfaz de comunicación serial universal un conversor A/D de 10 bits con referencia integrada y un controlador de transferencia de datos (DTC), dos amplificadores operacionales de propósitos generales y 32 pines de entrada/salida.



Figura 3.1: EduKit Ez430 Board

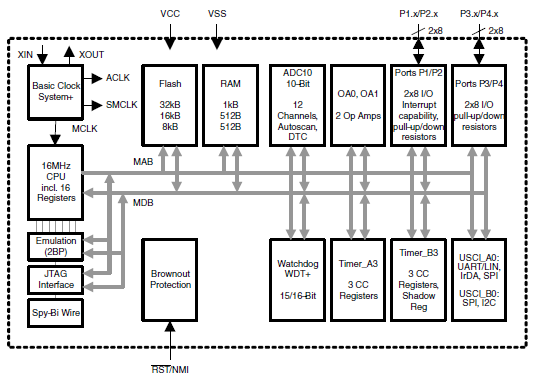


Figura 3.3: Diagrama de bloques

funcionales MSP430x22x4

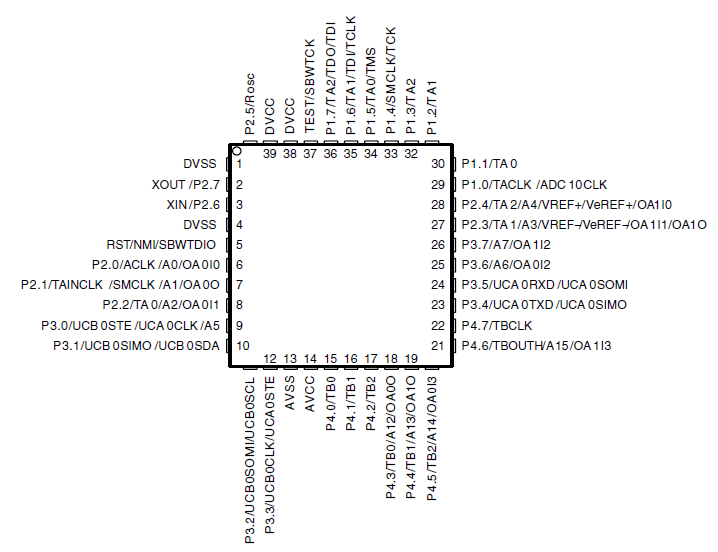


Figura 3.2: Pinout MSP430x22x4

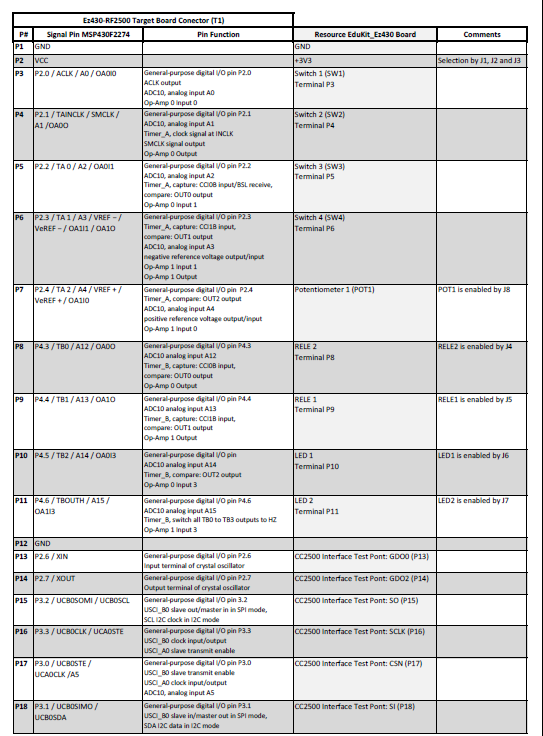


Figura 3.4

1. Resolución de guías de laboratorio
   1. Guía de laboratorio 1: Microcontroladores: puertos E/S digitales (GPIO)

Elementos de hardware necesarios:

• 2 pulsadores (SW1 y SW2)

• 2 leds (LED1 y LED2)

• Plataforma programable (microcontrolador) con puertos de E/S digital (GPIO)

Escribir un programa en lenguaje C que, mediante la utilización del módulo GPIO de la plataforma, realice las operaciones de configuración inicial y de control para utilizar los elementos antes mencionados de acuerdo a las siguientes especificaciones:

1. Mientras SW1 este presionado LED1 esta encendido. Al soltar SW1, LED1 se apaga.
2. Mientras SW2 este presionado LED2 esta encendido. Al soltar SW2, LED2 se apaga.
3. Si mientras SW1 está presionado, se presiona SW2, ambos LEDs deben permanecer encendidos al soltar los pulsadores.
4. Si mientras SW2 está presionado, se presiona SW1, ambos LEDs deben apagarse.

Se pide:

1. Implementar el programa que realice las operaciones descriptas mediante “pooling” a los registros de status correspondientes.
2. Implementar el mismo programa pero mediante la utilización del recurso de Interrupción asociado al módulo GPIO.

* Implementación mediante “pooling”:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp430f2274.h"

#include "CPU.h"

#include "GPIO.h"

#define TRUE 1

#define FALSE 0

voidmain( void )

{

int Led1Flag,Led2Flag = 0;

CPU\_Init();

GPIO\_Init();

while(1){

if (!(SW\_PIN & SW1)){

if (!(SW\_PIN & SW2)){

if(!Led1Flag && !Led2Flag){

Led1Flag = 1;

Led2Flag = 1;

LED\_POUT |= LED\_VERDE + LED\_ROJO;

}

else{

if(!Led1Flag){

Led1Flag = 1;

LED\_POUT |= LED\_VERDE;

}

}

}

else{

if(Led1Flag){

Led1Flag = 0;

LED\_POUT &= ~LED\_VERDE;

}

}

}

if (!(SW\_PIN & SW2)){

if(!(SW\_PIN & SW1)){

if(Led1Flag && Led2Flag){

Led1Flag = 0;

Led2Flag = 0;

LED\_POUT &= ~( LED\_VERDE + LED\_ROJO );

}

else{

if(!Led2Flag){

Led2Flag = 1;

LED\_POUT |= LED\_ROJO;

}

}

}

else{

if(Led2Flag){

Led2Flag = 0;

LED\_POUT |= LED\_ROJO;

}

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//GPIO.h

#define SW1 0x01

#define SW2 0x02

#define LED\_ROJO 0x20

#define LED\_VERDE 0x40

#define SW\_PDIR P2DIR

#define SW\_PIN P2IN

#define SW\_PREN P2REN

#define SW\_POUT P2OUT

#define LED\_PDIR P4DIR

#define LED\_POUT P4OUT

voidGPIO\_Init(void);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Biblioteca de configuracion de los GPIO

\*/

#include "msp430f2272.h"

#include "GPIO.h"

void GPIO\_Init(void){

SW\_PDIR &= ~(SW1 + SW2);

//Habilito Res de pullup

SW\_PREN |= SW1 + SW2;

SW\_POUT |= SW1 + SW2;

LED\_PDIR |= LED\_ROJO + LED\_VERDE;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Biblioteca de configuracionbasica de la CPU

voidCPU\_Init(void);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp430f2272.h"

#include "CPU.h"

voidCPU\_Init(){

WDTCTL = WDTPW + WDTHOLD; //Detengo el Watchdog

DCOCTL = 0; //Frecuencia DCO (8MHz)

BCSCTL1 = CALBC1\_8MHZ;

DCOCTL = CALDCO\_8MHZ;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Implementación mediante interrupciones asociadas al módulo GPIO:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp430f2274.h"

#include "CPU.h"

#include "GPIO.h"

char flag1 = 0;

char flag2 = 0;

voidLed\_test(void);

voidmain( void ){

CPU\_Init();

GPIO\_Init();

\_\_enable\_interrupt();

while(1){

}

}

#pragma vector= PORT2\_VECTOR

\_\_interruptvoid PORT2\_ISR(void){

if(SW\_PXIFG&SW1){

if(SW\_PXIES&SW1){ //Chequeo si está activo el flanco descendente

if(!(flag1) && !(flag2)){

LED\_POUT |= LED\_ROJO;

SW\_PXIES &=~SW1;

}

else{

if(flag1 && !(flag2)){

LED\_POUT &= ~(LED\_ROJO + LED\_AMARILLO);

SW\_PXIES |= SW1 + SW2;

flag1=0;flag2=0;

}

}

}

else{

if(!(flag1) || !(flag2)){

LED\_POUT &=~ LED\_ROJO;

SW\_PXIES |= SW1;

}

}

SW\_PXIFG &=~SW1;

}

if(SW\_PXIFG&SW2){

if(SW\_PXIES&SW2){ //Chequeo si está activo el flanco descendente

if(flag1 && flag2){

flag1=1;

flag2=0;

}

if(!(flag1) && !(flag2)){

LED\_POUT |= LED\_AMARILLO;

SW\_PXIES &=~SW2;

}

if(!(SW\_PXIES&SW1)){

LED\_POUT |= LED\_ROJO;

SW\_PXIES |=SW2 + SW1;

flag2=1;flag1=1;

}

}

else{

LED\_POUT &= ~LED\_AMARILLO;

SW\_PXIES |= SW2;

}

SW\_PXIFG &=~SW2;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//GPIO.h

#define SW1 BIT0

#define SW2 BIT1

#define LED\_ROJO 0x20

#define LED\_AMARILLO 0x40

#define SW\_PDIR P2DIR

#define LED\_PDIR P4DIR

#define LED\_POUT P4OUT

#define SW\_PREN P2REN

#define SW\_PXIFG P2IFG

#define SW\_PXIE P2IE

#define SW\_PXIES P2IES

#define SW\_PIN P2IN

#define SW\_POUT P2OUT

voidGPIO\_Init(void);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Biblioteca de configuracion de los GPIO

\*/

#include "msp430f2272.h"

#include "GPIO.h"

voidGPIO\_Init(void){

//Todos los pines como salidas y los reseteo

SW\_PDIR &= ~(SW1 + SW2);

SW\_PREN |= SW1 + SW2; //Pull-up enable

SW\_POUT |= SW1 + SW2; //Pull-up control

SW\_PXIES |= SW1 + SW2; //Flanco descendente

SW\_PXIFG &= ~(SW1+ SW2); //Bajo las flag - Por seguridad

SW\_PXIE |= SW1 + SW2; // Habilito interrupcion de P2

LED\_PDIR |= LED\_ROJO + LED\_AMARILLO;

LED\_POUT &= ~(LED\_ROJO + LED\_AMARILLO);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Biblioteca de configuracionbasica de la CPU

voidCPU\_Init(void);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp430f2272.h"

#include "CPU.h"

voidCPU\_Init(){

WDTCTL = WDTPW + WDTHOLD; //Detengo el Watchdog

DCOCTL = 0; //Frecuencia DCO (8MHz)

BCSCTL1 = CALBC1\_8MHZ;

DCOCTL = CALDCO\_8MHZ;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* 1. Guía de laboratorio 2: Microcontroladores: módulo USART

Elementos de hardware necesarios:

• PC con puerto serie RS232 o Adaptador USB-RS232

• Software de Terminal Serial (HyperTerminal o similar)

• Plataforma programable (microcontrolador) con módulo USART (UART) e interfaz externa.

Se pide:

Escribir un programa en lenguaje C que, mediante la utilización del módulo UART de la plataforma, realice las operaciones de configuración inicial y de control para cumplir con los siguientes requerimientos:

1. Mediante el programa Terminal de la PC se deben enviar datos en formato ASCII por puerto serial al microcontrolador para que este realice operaciones de suma y resta, y devuelva los resultados por esta misma interfaz para ser visualizados en la Terminal.
2. Los datos enviados desde la PC deben respetar el siguiente formato:

[N0][N1][N2][OP][M0][M1][M2][CR]

Dónde: N0, N1 y N2 son 3 números en formato ASCII que representan un número de 3 dígitos (1° sumando). De la misma forma M0, M1 y M2 representan el 2° sumando. OP es el carácter que determina la operación a realizar: ‘+’ para suma y ‘-‘ para la resta. Y CR es el código ASCII de la tecla “Enter” (0x13).

Ejemplo: si se desea realizar la operación 23+400 se debe enviar la siguiente secuencia: [‘0’][‘2’][‘3’][‘+’][‘4’][‘0’][‘0’][CR]

1. El resultado debe ser enviado desde el microcontrolador a la PC en el siguiente formato:

[R0][R1][R2][R3][CR]

1. Los datos deben ser enviados por la interfaz serial configurada sin control de flujo de datos, a 9600bps, 8 bit de datos, 1 bit de stop y paridad par.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | Parity | Stop |

Se pide:

1. Implementar el programa que realice las operaciones descriptas mediante “pooling” a los registros de status correspondientes.
2. Implementar el mismo programa pero mediante la utilización del recurso de Interrupción asociado al módulo UART.

* Implementación mediante “pooling”:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Practico 2 con Pooling

\*/

#include "msp430f2274.h"

#include "TI\_MSP\_UART.h"

charstring[] = {'C','A','L','C','U','L','A','D','O','R','A','\n','\r'};

char string2[]="Resultado: ";

chardato\_rx[7]={0};

char dato\_rx2[7]={0};

unsignedint i= 0;

unsignedintvalidateFlag=0;

int error=0;

intmain( void )

{

// Stop watchdogtimer to prevent time outreset

WDTCTL = WDTPW + WDTHOLD;

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

Init\_UART();

UART\_Tx\_string(string,(sizeof(string)-sizeof(char))/sizeof(char));

UART\_Tx\_char('\r');

UART\_Tx\_char('\n');

while(1){

if(UART\_IFG &UART\_UCAxRXIFG){

dato\_rx[i] = UART\_UCAxRXBUF;

i++;

if(i>=8){ //Validar trama completa recibida

i=0;

validateFlag = 1;

}

}

if (validateFlag){

validateFlag=0;

for(char a=0; a<7; a++){dato\_rx2[a]=dato\_rx[a];}

error = UART\_Rx\_Validate(dato\_rx2);

if (error >= 1) UART\_Tx\_ErrorMsg(error);

else{ UART\_Tx\_string(string2,0); UART\_Rx\_Calc(dato\_rx2); UART\_Tx\_char('\r');

UART\_Tx\_char('\n');}

i=0;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Implementación mediante interrupciones asociadas al módulo UART:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp430f2274.h"

#include "TI\_MSP\_UART.h"

charstring[] = {'C','A','L','C','U','L','A','D','O','R','A','\n','\r'};

char string2[]="Resultado: ";

chardato\_rx[7]={0};

char dato\_rx2[7]={0};

unsignedint i= 0;

unsignedintvalidateFlag=0;

int error=0;

intmain( void )

{

// Stop watchdogtimer to prevent time outreset

WDTCTL = WDTPW + WDTHOLD;

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

Init\_UART();

\_\_enable\_interrupt();

UART\_Tx\_string(string,(sizeof(string)-sizeof(char))/sizeof(char));

UART\_Tx\_char('\r');

UART\_Tx\_char('\n');

while(1){

if (validateFlag){

validateFlag=0;

for(char a=0; a<7; a++){dato\_rx2[a]=dato\_rx[a];}

error = UART\_Rx\_Validate(dato\_rx2);

if (error >= 1) UART\_Tx\_ErrorMsg(error);

else{ UART\_Tx\_string(string2,0); UART\_Rx\_Calc(dato\_rx2); UART\_Tx\_char('\r');

UART\_Tx\_char('\n');}

i=0;

}

}

}

#pragma vector=USCIAB0RX\_VECTOR

\_\_interruptvoidISR\_UartRx(void){

if(UART\_IFG &UART\_UCAxRXIFG){

dato\_rx[i] = UART\_UCAxRXBUF;

i++;

if(i>=7){ //Validar trama completa recibida

i=0;

validateFlag = 1;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* 1. Guía de laboratorio 3: Microcontroladores: Conversores A/D (ADC)

Elementos de hardware necesarios:

• Potenciómetro lineal de 10KΩ

• PC con puerto serie RS232 o Adaptador USB-RS232

• Software de Terminal Serial (HyperTerminal o similar)

• Plataforma programable (microcontrolador) con módulo USART (UART), interfaz externa y módulo ADC.

Se pide:

Escribir un programa en lenguaje C que, mediante la utilización del módulo conversor A/D y el módulo UART de la plataforma, realice las operaciones de configuración inicial y de control para cumplir con los siguientes requerimientos:

1. Se debe iniciar una conversión del valor de tensión analógico (variable con la posición del potenciómetro) cada vez que se reciba por el puerto serie UART el carácter ASCII ‘C’ o ‘c’.
2. El resultado de la conversión en formato digital debe ser procesado, utilizando la función de transferencia del ADC (dependiente de la configuración del mismo: resolución, niveles de referencia, etc), para ser convertido en un numero codificado en ASCII que represente el nivel de tensión del punto medio del potenciómetro expresado en V con 2 dígitos decimales. (Ejemplo: 2.34V)
3. Este número codificado en ASCII con el nivel de tensión convertido debe ser enviado por puerto serie UART para ser visualizado mediante un programa Terminal desde la PC.
4. Los datos enviados desde a la PC deben respetar el siguiente formato:

[N][’.’][D0][D1][‘V’][CR]

Dónde: N es la unidad del valor medido en formato ASCII. D0 y D1 son los dos decimales del valor medido también en ASCII. Y CR es el código ASCII de la tecla “Enter” (0x13).

Ejemplo: si el resultado obtenido es 2.34V, debe enviarse la siguiente secuencia: [‘2’][‘.’][‘3’][‘4’][‘V’][CR]

1. Los datos deben ser enviados por la interfaz serial configurada sin control de flujo de datos, a 9600bps, 8 bit de datos, 1 bit de stop y paridad par.

* Implementación del programa:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp430f2274.h"

#include "TI\_MSP\_ADC10.h"

#include "TI\_MSP\_UART.h"

unsignedint valor=0;

chardato\_rx=0;

intmain( void )

{

// Stop watchdogtimer to prevent time outreset

WDTCTL = WDTPW + WDTHOLD;

// Configuracion de clock

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

Init\_UART();

Init\_ADC10();

\_\_enable\_interrupt();

while(1){

if (dato\_rx == 'c' || dato\_rx == 'C'){

valor=AD10\_Convert(INCH\_4);

UART\_Tx\_string("Valor de conversion: ",0);

UART\_Tx\_Voltage(valor);

UART\_Tx\_char('\n');

UART\_Tx\_char('\r');

dato\_rx=0;

}

}

}

#pragma vector=USCIAB0RX\_VECTOR

\_\_interruptvoidISR\_UartRx(void){

if(UART\_IFG &UART\_UCAxRXIFG){

dato\_rx = UART\_UCAxRXBUF;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* 1. Guía de laboratorio 4: Microcontroladores: Modulo Timer

Elementos de hardware necesarios:

• Potenciómetro lineal de 10KΩ

• Led

• Osciloscopio

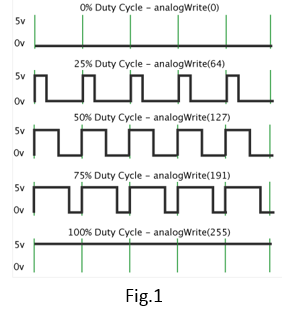
• Plataforma programable (microcontrolador) con móduloTimer (de comparación y captura) y módulo ADC.

Se pide:

Escribir un programa en lenguaje C que, mediante la utilización del conversor A/D y el móduloTimer de comparación y captura, realice las operaciones de configuración inicial y de control para cumplir con los siguientes requerimientos:

1. Se debe realizar una rutina de control por “pooling” a los registros de status del móduloTimer para hacer destellar al Led cada un periodo de tiempo fijo. El mismo debe permanecer encendido durante 100mseg. cada periodos de 2 seg.
2. Al mismo tiempo, mediante el recurso de interrupción del Timer, se debe generar una señal de sincronismo de 100Hz con ancho de pulso variable (Fig. 1) entre el 10% (10% del periodo en estado “1” y 90% en estado “0”) y el 90% (90% del periodo en estado “1” y 10% en estado “0”) por un pin digital de salida. El ancho del pulso será determinado por la lectura del nivel de tensión del potenciómetro conectado al módulo ADC:

* Desde 0V -> 10% del ciclo
* Hasta Vcc -> 90% del ciclo



Aclaración:

Este ejercicio fue implementado de manera diferente a lo solicitado en la consigna. Se generó un PWM de frecuencia 100 Hz con ancho de pulso variable desde 10% al 90%, tal como se pide en la consigna, pero en vez de utilizar el Timer y hacer que dentro de esta interrupción se generase el pulso variable por software, se estudió como configurar de manera adecuada los módulos de captura y comparación asociados al Timer para hacer que este genere un PWM por hardware utilizando las configuraciones de los registros del micro. La salida de la señal se encuentra en el pin OUT asociado al comparador, que para nuestro caso es la bornera 6 del EduKit.

En concreto, se utilizó el Modulo de comparación y captura 0 y el 1, con el Timer configurado en “up mode” y el registro de control del comparador configurado de manera tal que el TACCR0 marque la base de tiempo y que el TACCR1 marque el tiempo del duty cycle. En el programa se ven las configuraciones necesarias y se encuentran comentadas de manera de entender mejor su funcionamiento.

* Implementación del programa:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp430f2274.h"

#include "GPIO\_EduKit.h"

#include "TI\_MSP\_ADC10.h"

#include "PWM.h"

#define T1mSEG 10

#define T1SEG 100

#define T2SEG 210

unsignedintAD\_result;

char mensaje[4]={0};

unsignedinttimerLed=0;

unsignedint timerLed2=0;

unsignedinttimeVal=0;

intmain( void )

{

// Stop watchdogtimer to prevent time outreset

WDTCTL = WDTPW + WDTHOLD;

/\* configureinternaldigitally control led oscillator \*/

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

/\* configuracion de pin de salida del PWM\*/

P2DIR |= BIT3;

P2SEL |= BIT3;

/\* configuracion de ADC \*/

ADC10\_Init();

ADC10AE0 |= BIT4; //Pin de entrada para ADC

GPIO\_EduKit\_Init();

// Configuracion Inicial Timer A

/\*Con el modulo 0 y 1 de CCP \*/

TACTL = (TASSEL\_2 + ID\_3 + MC\_1); //MC\_3 up/downmode

TACCR0 = 1250; //-->10ms base de tiempo

TACCR1 = 125; //-->1ms equivalente al 10% del duty

TACCTL1 = (CM\_0 + CCIS\_2+ OUTMOD\_7 + CCIE);

\_\_enable\_interrupt();

while(1){

if(timerLed2<T1mSEG){LED\_POUT |= LED\_2;}

if(timerLed2>=T1mSEG){LED\_POUT &= ~LED\_2;}

if (timerLed2>=T2SEG){timerLed2=0;}

timeVal = PWM\_ADC2time(ADC10\_Convert(INCH\_4));

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Funcion: TimerA\_ISR\_CCR1

// ISR de CCR1-2 del Timer A

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#pragma vector = TIMERA1\_VECTOR

\_\_interruptvoid TimerA\_ISR\_CCR1(void){

if(TACCTL1 & CCIFG){

TACCTL1 &= ~CCIFG; //Bajo la bandela

timerLed++;

timerLed2++;

TACCR1 = timeVal;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Conclusión
2. Bibliografía