Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

#LAB. CONTROLADORES Y MICROCONTROLADORES PROGRAMABLES

#Práctica P6
" Interrupciones externas"

*Nombre o nombres de los integrantes junto con su matrícula:

#Verónica Yazmín Gómez Cruz #1884224
#Nahaliel Gamaliel Ríos Martínez #1884244

#Ing. Jesús Daniel Garza Camarena

Semestre Febrero 2021 – Junio 2021

MN1N2

San Nicolás de los Garza, N.L. #11.05.2021

Objetivo

Analizar el uso de las interrupciones externas de un Microcontrolador.

Introducción.

En lenguaje de procesadores digitales las interrupciones son señales que le indican al circuito que tiene que atender algún proceso urgente, dejando de lado temporalmente lo que esté haciendo en ese momento.

En las interrupciones controladas por E/S la CPU responde a una solicitud de servicio sólo cuando un dispositivo periférico efectúa su solicitud de manera explícita. De este modo, la CPU puede concentrarse en ejecutar el programa actual, sin tener que detenerlo innecesariamente para ver si un dispositivo necesita ser atendido.

Cuando la CPU recibe una señal de interrupción de E/S, detiene temporalmente el programa actual, confirma la interrupción y extrae de la memoria un programa especial (rutina de atención de la interrupción) adaptado al dispositivo concreto que haya generado la interrupción. Una vez generada la rutina de atención a la interrupción, la CPU continúa con aquello que estuviera haciendo. Un dispositivo especial denominado controlador de interrupciones programable (PIC, Programmable Interrupt Controller) gestiona las interrupciones de acuerdo con un mecanismo de prioridad. Este dispositivo acepta las solicitudes de servicio procedentes de los periféricos. Si dos o más dispositivos solicitan servicio al mismo tiempo, aquél que tenga asignada la prioridad más alta será servida primero, después el que tenga la siguiente prioridad más alta y así sucesivamente. Después de enviar una señal de interrupción (INTR) a la CPU, el controlador PIC proporciona a la CPU la información necesaria para “dirigir” a la CPU hacia la dirección de memoria inicial de la rutina de atención a la interrupción apropiada. Este proceso se denomina vectorización.

Para las interrupciones externas o hardware, solo hay dos pines que las soportan en los ATmega328 son las INT0 y INT1 que están mapeadas a los pines 2 y 3. Estas interrupciones se pueden configurar con disparadores en RISING o FALLING para flancos o en nivel LOW. Los disparadores son interpretados por hardware y la interrupción es muy rápida.

Diagrama de bloques

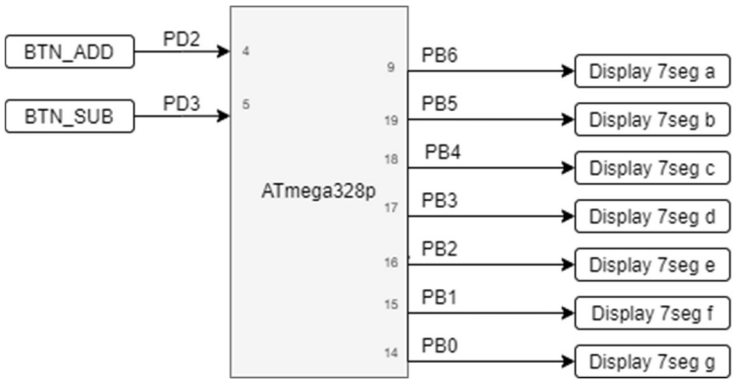
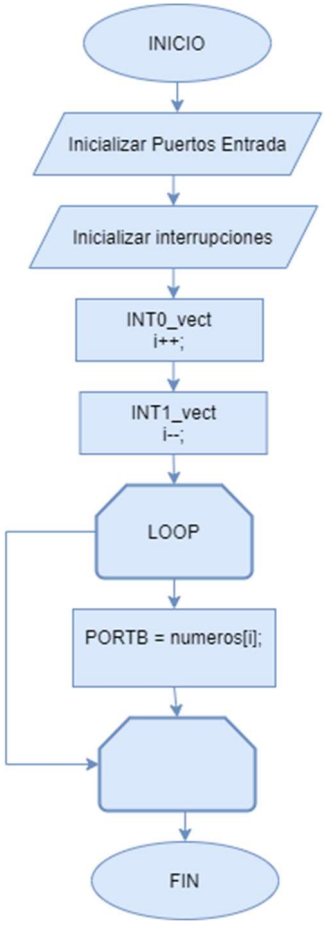


Diagrama de flujo.



Materiales utilizados

1 ATMEGA328P
3 Push Button
5 resistencaias (10K y 1 K)
2 Diodos
2 capacitores
1 Display 7 seg

Código en Atmel.

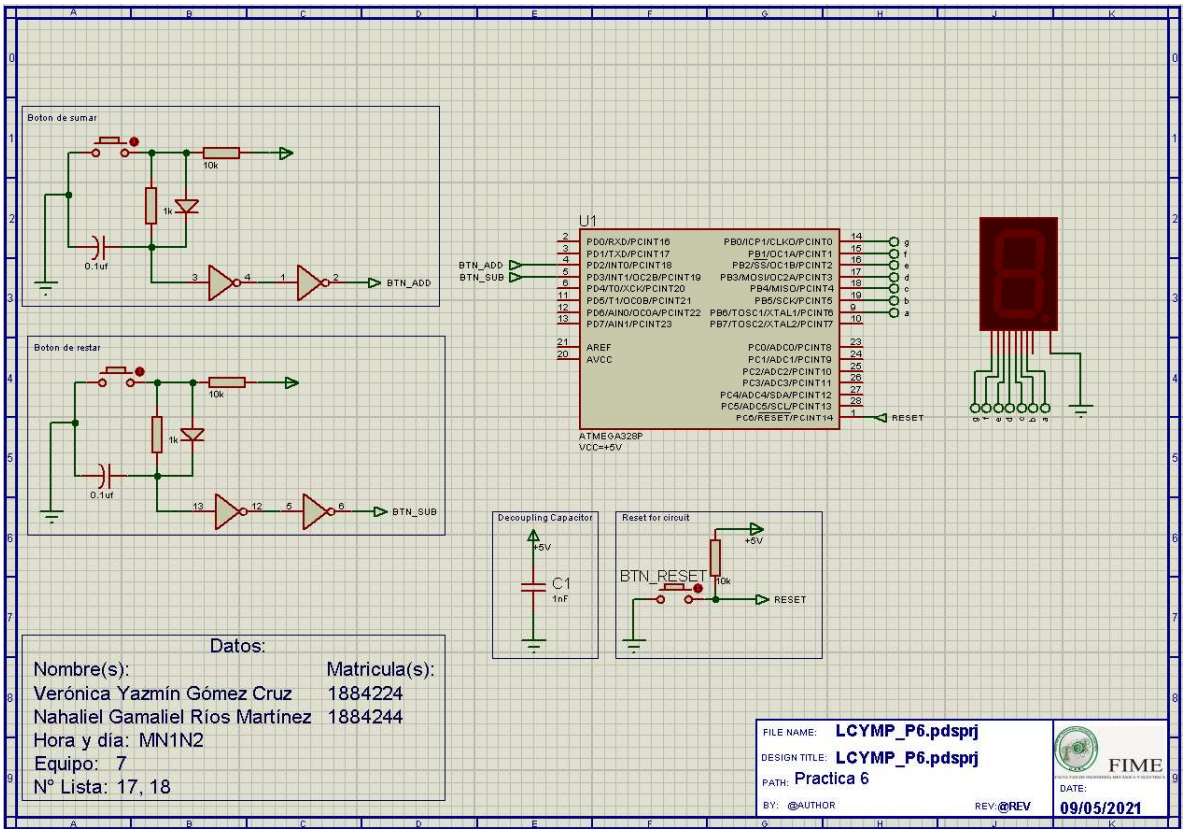
```
/*************************************************
 * LLENAR ESTE ESPACIO CON LOS SIGUIENTES DATOS:    *
 * Nombre: Verónica Yazmín Gómez Cruz               *
 *         Nahaliel Gamaliel Rios Martinez          *
 * Hora clase: N1-N2                                *
 * Día: M                                           *
 * N° de lista: 17, 18                              *
 * N° de Equipo: 7                                  *
 * Dispositivo: ATMEGA328P                          *
 * Rev: 1.0                                         *
 * Propósito de la actividad:                       *
 * Crear un contador ascendente - descendente       *
 * mostrado mediante un solo display de 7 segmentos *
 * ya sea cátodo o ánodo común mostrando números      *                          *
 * del 0 al 9.                                                                   *
 *
      *
 * Un botón de interrupción incrementa el número      *
 * y otra interrupción lo decrementa.                       *
 *
      *
 * Crear alguna manera de eliminar los                      *
 * rebotes creados por la acción mecánica                   *
 * de los botones por hardware                                     *
 *                                 Fecha: 02.05.2021 *
 *************************************************/
/*atmega328P PIN - OUT*/
/*       PIN - OUT
         atmega328P
          -------
    PC6  |1     28| PC5
    PD0  |2     27| PC4
    PD1  |3     26| PC3
    PD2  |4     25| PC2
    PD3  |5     24| PC1
    PD4  |6     23| PC0
    VCC  |7     22| GND
    GND  |8     21| AREF
    PB6  |9     20| AVCC
    PB7  |10    19| PB5
    PD5  |11    18| PB4
    PD6  |12    17| PB3
    PD7  |13    16| PB2
    PB0  |14    15| PB1
```

```
         --------
*/
/*atmega328P PIN FUNCTIONS*/
 /*
 atmega328P PIN FUNCTIONS
 pin   function              name    pin     function              name
 1     !RESET/PCINT14        PC6     15      PCINT1/OC1A           PB1
 2     RxD/PCINT16           PD0     16      PCINT2/OC1B/SS        PB2
 3     TxD/PCINT17           PD1     17      PCINT3/OC2A/MOSI      PB3
 4     INT0/PCINT18          PD2     18      PCINT4/MISO           PB4
 5     INT1/PCINT19/OC2B     PD3     19      PCINT5/SCK            PB5
 6     PCINT20               PD4     20      ANALOG VCC            AVCC
 7     +5v                   VCC     21      ANALOG REFERENCE      AREF
 8     GND                   GND     22      GND                   GND
 9     XTAL1/PCINT6          PB6     23      PCINT8/ADC0           PC0
 10    XTAL2/PCINT7          PB7     24      PCINT9/ADC1           PC1
 11    PCINT21/OC0B          PD5     25      PCINT10/ADC2          PC2
 12    PCINT22/OC0A/AIN0     PD6     26      PCINT11/ADC3          PC3
 13    PCINT23/AIN1          PD7     27      PCINT12/ADC4/SDA      PC4
 14    PCINT0/AIN1           PB0     28      PCINT13/ADC5/SCL      PC5
 */
/*******************Bibliotecas*****************************/
#include <avr/io.h>//se incluyen las Bibliotecas de E/S del AVR atmega328P
#include <avr/interrupt.h>// librería de interrupciones
#include <util/delay.h> // Librería de retardos

/*****************Macros y constantes**********************/
#define F_CPU 1000000UL //1 Mhz

/*******************Variables globales*********************/
//--Espacio para declarar variables globales
#define a PINB0
#define b PINB1
#define c PINB2
#define d PINB3
#define e PINB4
#define f PINB5
#define g PINB6

#define ButtonAdd PIND2 // INT 0
#define ButtonSub PIND3 // INT 1

volatile char i = 0; //Contador para leer el arreglo de numeros
volatile char timer = 0; //Contador para el timer

uint8_t numeros[10] = {
       //gfedcba
       0b0111111, //0
       0b0000110, //1
       0b1011011, //2
       0b1001111, //3
       0b1100110, //4
       0b1101101, //5
       0b1111101, //6
       0b1000111, //7
       0b1111111, //8
       0b1100111, //9
};
```

```
/**********************Funciones**************************/
//--Espacio para Establecer funciones
/*********************Declaración de Funciones************/
//--Espacio para declarar funciones
void initialize_ports(void); // Inicializar puertos
void initialize_interrupt(void); // Inicializar interrupciones

/******************Programa principal*******************/
int main(void)
{
//--Inicialización
    cli(); //Deshabilitamos interrupciones
       initialize_ports(); // va hacía la inicialización de puertos
       initialize_interrupt();// va hacía la inicialización del TIMER para controlar
Led
       sei(); //Habilitamos interrupciones

//--Ejecución
    while (1) //loop infinito
    {

              //PORTB |=_BV(LedIndicador); //Encender
              PORTB = numeros[i];

              //i++;
              if (i == 10) {
                     i = 0;
              }


              //_delay_ms(2000);

    } // END loop infinito

} // END MAIN
/******************Definición de funciones****************/
//*********************************************************
//initialize_ports : inicializa los puertos de entrada o   *
//salida                                                   *
//*********************************************************
void initialize_ports(void)
{
       //--Entradas
       DDRD &=~ _BV(ButtonAdd);  //INT 0 como entrada
       PORTD|=_BV(ButtonAdd); // Push button con pull - up (INT 0)

       DDRD &=~ _BV(ButtonSub); // INT 1 como entrada
       PORTD|=_BV(ButtonSub); // Push button con pull - up (INT 1)

    //--Salidas
       DDRB |=_BV(a);
       DDRB |=_BV(b);
       DDRB |=_BV(c);
       DDRB |=_BV(d);
       DDRB |=_BV(e);
       DDRB |=_BV(f);
```

```
        DDRB |=_BV(g);

        PORTB = 0x00; //-Por seguridad iniciamos en 0


}
//*********************************************************************
//initialize_ínterrupt :
//*********************************************************************
void initialize_interrupt(void)
{
//INT0
    //Modo
    EICRA &=~ (1<<ISC00);   // INT0 configurado = LOW LEVEL
    EICRA &=~ (1<<ISC01);   // INT0 configurado = LOW LEVEL
    //Activacion
    EIMSK |= (1<<INT0);     // INT0 activado
//INT1
    //Modo
    EICRA &=~ (1<<ISC10);   // INT1 configurado = LOW LEVEL
    EICRA &=~ (1<<ISC11);   // INT1 configurado = LOW LEVEL
    //Activacion
    EIMSK |= (1<<INT1);     // INT1 activado
}
//*********************************************************************
//ISR :
//*********************************************************************
ISR (INT0_vect) // Vector de interrupción INT0
{
    i++;

       if (i == 10) {
              i = 0;
       }
}
ISR (INT1_vect) // Vector de interrupción INT1
{

       if (i == 0) {
              i = 10;
       }

    i--;

}
```

Diagrama del circuito en PROTEUS.



Conclusión

En esta práctica aprendimos a utilizar las interrupciones externas que brinda el microcontrolador Atmega328p, en este caso utilizamos estas interrupciones para incrementar y decrementar una variable y con ello cambiara el valor de un dispaly de 7 segmentos, esto sin tener que codificar explícitamente en el código principal la condición del botón. Este tipo de interrupción puede ser de mucha ayuda para no estar constantemente revisando el estado de nuestras entradas y salidas en el código principal y dejar ese trabajo a los procesos del microcontrolador, con esto nuestro código principal puede estar enfocado totalmente en ciertas tareas y hacer ciertas cosas al detectar cambios en las entradas o salidas a través de la interrupción.

En esta práctica también pudimos observar el efecto de los rebotes. Cuando armamos el circuito en físico, nos dimos cuenta de que con cada pulsación del botón el display avanzaba varios números hacia adelante o hacia atrás, por lo cual es de suma importancia tener un mecanismo de anti-rebotes para evitar el mal funcionamiento del circuito. Investigando un poco más vimos que existen 2 métodos para evitar esto, el que vimos en clase que es el método por Hardware pero también existe formas de evitar esto por software, en nuestro caso tratamos de implementar el método por hardware pero nos faltaron algunos componentes necesarios para poder hacer funcionar correctamente el circuito.

Bibliografía

Parra Reynada, L. (2012). Microprcesadores. RED TERCER MILENIO S.C.

Floyd, T. L. (2006). Fundamentos de sistemas digitales. Pearson Educación.

ATmega328P. 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. DATASHEET. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf