Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

CONTROLADORES Y MICROCONTROLADORES PROGRAMABLES

Actividad fundamental 3
"TIMER"

Nombre o nombres de los integrantes junto con su matrícula:
Nahaliel Gamaliel Ríos Martínez 1884244

Ing. Jesus Daniel Garza Camarena

Semestre Febrero 2021 – Junio 2021

MN1N2

San Nicolás de los Garza, N.L. 05.05.2021

Objetivo

Utilizar los TIMER del 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

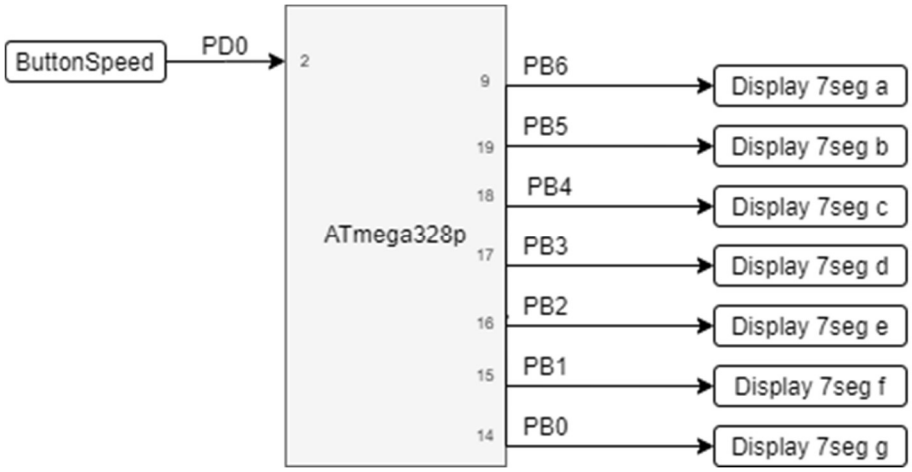
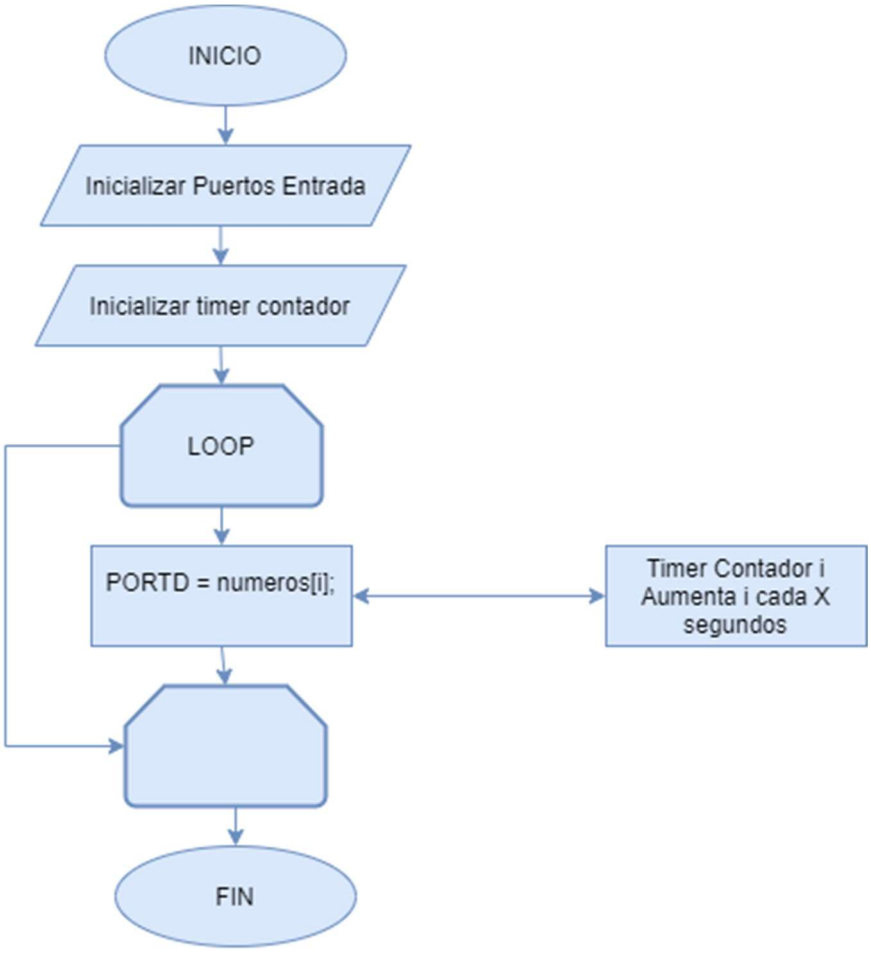


Diagrama de flujo.



Materiales utilizados

1 ATMEGA328p
1 Capasitor
2 PushButton
1 Display 7seg
1 resistencia

Código en Atmel.

```
/*****************************************************
 * LLENAR ESTE ESPACIO CON LOS SIGUIENTES DATOS:     *
 * Nombre: Nahaliel Gamaliel Rios Martinez           *
 * Hora clase: N4                                    *
 * Día: LMV                                          *
 * N° de lista: 33                                   *
 * Dispositivo: ATMEGA328P                           *
```

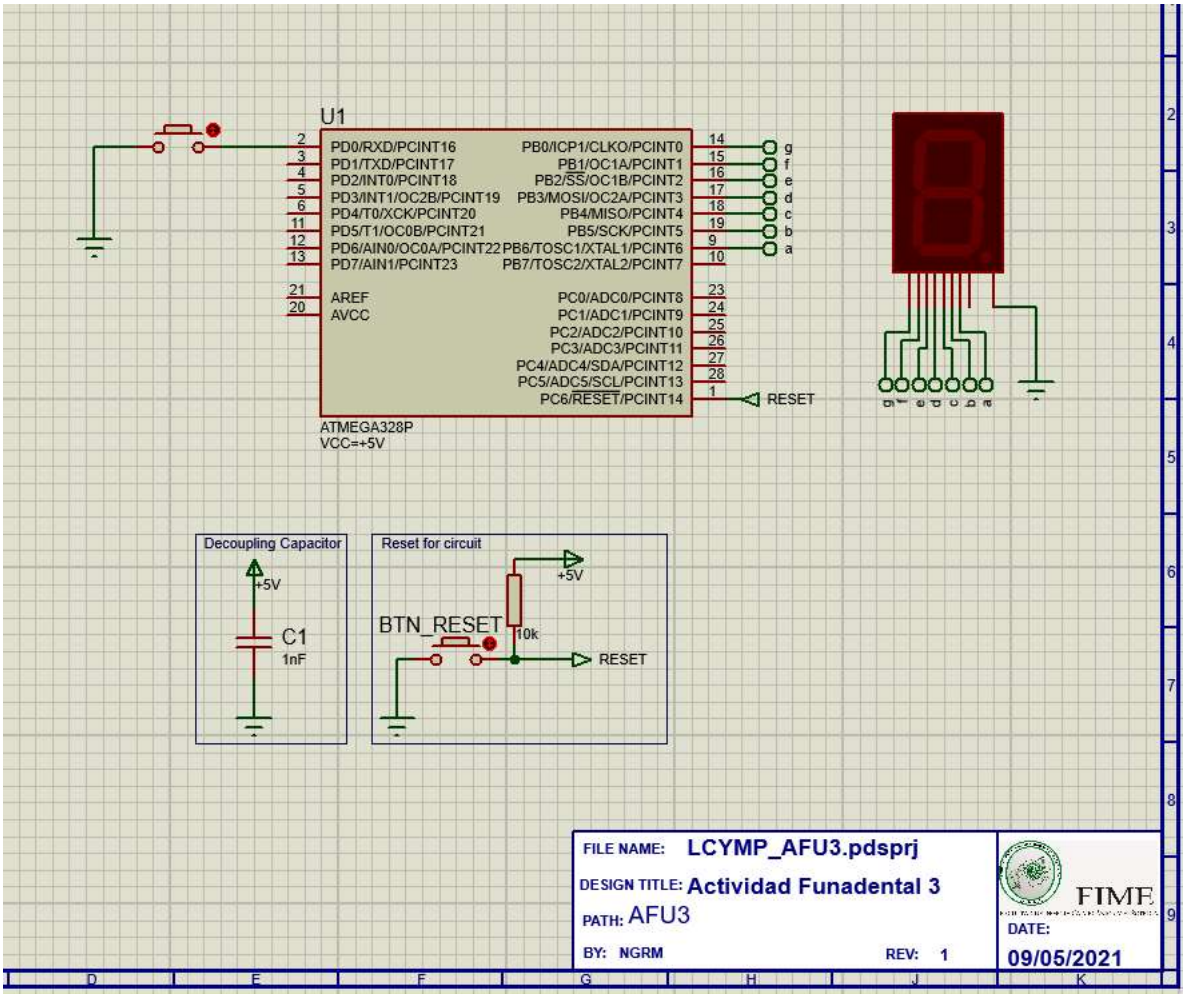
```
 * Rev: 1.0                                          *
 * Propósito de la actividad:                        *
 * Diseña y realiza un contador automático de        *
 * número 0 al 9 mostrado mediante un display 7      *
 * segmentos, controlando su velocidad con un botón  *
 * bajo las siguientes condiciones:                  *
 * - Si el botón no está presionado el contador      *
 *   incrementa cada 1 segundos.                     *
 * - Si se presiona el botón el contador incrementa  *
 *   cada 0.5 segundos.                              *
 *                                Fecha: 09.05.2021*
 ***************************************************/
/*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


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

```
/******************Variables globales********************/
#define a PINB0
#define b PINB1
#define c PINB2
#define d PINB3
#define e PINB4
#define f PINB5
#define g PINB6

#define ButtonSpeed PIND0

volatile char i = 0; //Contador para leer el arreglo de numeros
volatile char speed = 10; //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
};

/*********************Declaración de Funciones*************/
void initialize_ports(void);
void initialize_timer(void);// Función para inicializar Timer_0
void timer_on(void);// Función para encender Timer_0


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

	timer_on(); //Encendemos Timer0

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

		if (bit_is_clear(PIND,ButtonSpeed)){
			speed = 5;
		}else{
			speed = 10;
		}

		PORTB = numeros[i];
```

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

      } // END loop infinito

} // END MAIN
/******************Definición de funciones****************/
//*************************************************************
//Descripcion de lo que hace la funcion:                     *
//initialize_ports : inicializa los puertos de entrada y     *
//salida                                                     *
//*************************************************************
void initialize_ports(void)
{
      //--Entradas
      DDRD &=~ _BV(ButtonSpeed);  //INT 0 como entrada
      PORTD|=_BV(ButtonSpeed); // 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_timer_led : inicializa el timer para controlar Led
//*********************************************************************
void initialize_timer(void)
{
    //Modo de operacíon configurado como CTC
    TCCR0A &=~ (1<<WGM00);// 0 en el bit WGM00
    TCCR0A |=  (1<<WGM01);// 1 en el bit WGM01
    TCCR0B &=~ (1<<WGM02);// 0 en el bit WGM02
    OCR0A = 97; //Registro de 8 bits donde se pone el numero a comparar
    TIMSK0 |= (1<<OCIE0A);//Se pone un 1 en el bit OCIE0A del registro
    //TIMSK0 para habilitar la interrupción
}
//*********************************************************************
//timer_led_on: Enciende el timer para controlar Led
//*********************************************************************
void timer_on(void)
{

    TCNT0 = 0; // Registro de 8 bits que lleva el conteo del timer_0
    //Prescaler configurado en 1024
    TCCR0B |=  (1<<CS00);// 1 en el bit CS00
    TCCR0B &=~ (1<<CS01);// 0 en el bit CS01
    TCCR0B |=  (1<<CS02);// 1 en el bit CS02
```

```
}
//**********************************************************************
//Vectores de interrupción, Interrupt service routine (ISR)
//**********************************************************************
//**********************************************************************
//TIMER_0
//**********************************************************************
ISR (TIMER0_COMPA_vect)// Vector de interrupción para el Timer0 (0.1s)
{
   timer++;
   if(timer == speed){
          i++;
          timer = 0;
   }

}
```

Diagrama del circuito en PROTEUS.



Conclusión

En esta práctica utilizamos uno de los timers que tiene integrado el ATMEGA328P para aumentar una variable i con la que manejamos el arreglo que contiene los números que mostramos en nuestro display, para lograr el efecto de aceleración utilizamos el timer a 0.1 segundos y una variable que llamamos speed, cuando el botón no se esta presionando speed vale 10 para de esta forma esperar 1 segundo completo antes de aumentar la variable i, al momento de que se presiona el botón speed pasa a valer 5 para de esta forma esperar 0.5 segundos antes de aumentar la variable i.

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