Práctica 4

- **Manejo de la sección de E/S del micocrontrolador ATmega1280/2560**

Objetivo: Mediante esta práctica el alumno analizará la implementación de retardos por software, así como también se familiarizará con la configuración y uso de puertos.

Equipo: - Computadora Personal con AVR Studio, AVRDUDE y tarjeta TJuino
Teoría: 1) Investigación a cerca de ensamblador en línea para GCC.
2) Convención de llamadas a funciones en C en GCC para AVR (avr-gcc).
3) Programación en lenguaje C en microcontroladores
4) Técnicas de anti-rebote de botones táctiles.

Descripción:

Implementar un programa en base al Listado 1, el cual revisa el estado del botón conectado en **PK0**, y dependiendo de su duración envía ciertos datos. La transmisión está definida por el Listado 2, enviándolo por el pin **PE1**.

Listado 1:

```
#define SetBitPort(port, bit) __asm__ ( ?? )
#define ClrBitPort(port, bit) __asm__ ( ?? )

//Press States
#define NOT_PRESSED 0
#define SHORT_PRESSED 1
#define LONG_PRESSED 2

#define BTN_PIN PK0
#define SEND_PIN PE1

extern void delay_103us(void);
extern void delay(uint16_t mseg);
void InitPorts(void);
uint8_t check_Btn(void);
void sendData(uint8_t *data);
void sendByte(uint8_t data);

uint8_t
data[2][14]={{0x53,0x68,0x6f,0x72,0x74,0x20,0x70,0x72,0x65,0x73,0x73,0x0D,0
x0A,0x00},{0x4c,0x6f,0x6e,0x67,0x20,0x70,0x72,0x65,0x73,0x73,0x21
,0x0D,0x0A,0x00}};

int main(void){
	UCSR0B &= ~(1<<TXEN0); // Disable UART-TX

	InitPorts();
	while(1){
		switch(check_Btn()){
			case SHORT_PRESSED: sendData(data[0]);
					    break;
			case LONG_PRESSED:  sendData(data[1]);
					    break;
		}

	}
}
```

Macros a implementar:

1. `SetBitPort(port, bit)`
   Macro que inserta la instrucción de ensamblador **SBI**, mediante *inline assembly*.

2. `ClrBitPort(port, bit)`
   Macro que inserta la instrucción de ensamblador **CBI**, mediante *inline assembly*.

Funciones a implementar:

3. `void delay_103us(void);`
   Función que debe de tardarse exactamente 103 *us* en retornar.

4. `void delay(uint16_t mseg);`
   Función que debe de tardarse **n** *ms* en retornar, según se especifique en el parámetro de entrada. Con una exactitud de ±5 *us*.

5. `void InitPorts(void);`
   Inicialización requerida de los puertos utilizados en esta práctica. El pin PE1 debe dejarse en un nivel alto.

6. `uint8_t check_Btn(void);`
   Retorna el estado del botón, detectando entre `NOT_PRESSED`, `SHORT_PRESSED` y `LONG_PRESSED`. Donde el umbral para una larga duración es cualquiera que sea mayor a 1 seg. Implementar técnicas de anti-rebote.

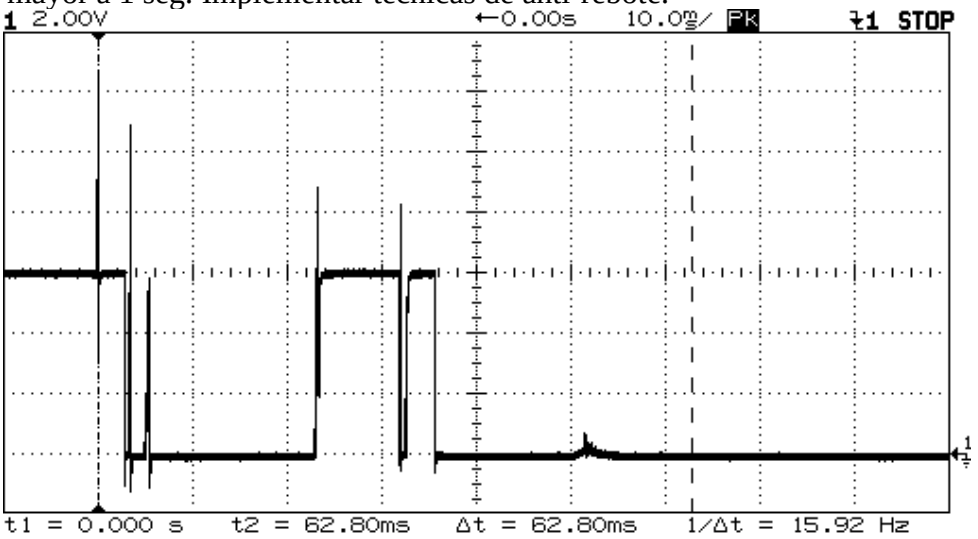


Fig. 1. Ejemplo del rebote mecánico de un botón.

7. `void sendData(uint8_t *data);`
   Recorre el arreglo que recibe por parámetro, y lo envía mediante `sendByte()` hasta encontrar la cadena vacía.

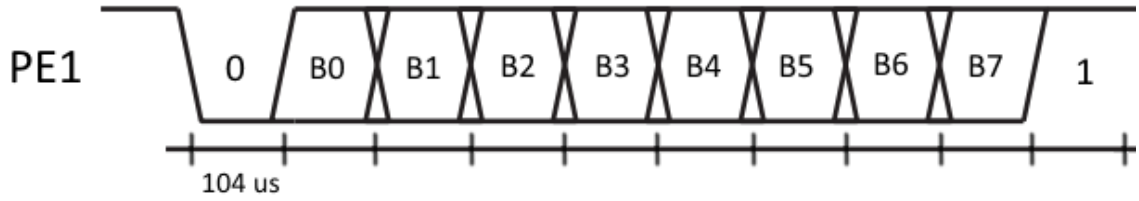
8. `void sendByte(uint8_t data);`
   Función que implementa el Listado 2.

Listado 2:

```
;**************** Algoritmo sendByte ********************
Consideraciones:
El bit PE1 ya está programado como salida y se le ha escrito un 1 lógico

inicio:
  Paso 1: Poner en bajo (0 lógico) a PE1
  Paso 2: Esperar un retardo de 103uS. (Tx start bit)
  Paso 3: Pasar el bit LSB (menos significativo) a PE1.
  Paso 4: Esperar un retardo de 103uS. (Tx bit N del byte)
  Paso 5: Hacer corrimiento a la derecha el dato recibido.
  Paso 6: Si no es el último bit ir al paso 3, de lo contrario continuar
          Nota: El último bit es cuando el MSB original ha llegado al LSB
  Paso 7: Poner PE1 alto (1 lógico). Nota: se deja como estaba al inicio
  Paso 8: Esperar un retardo de 103uS. (Tx stop bit)
fin:
```

En el siguiente diagrama se presenta el comportamiento del algoritmo anterior:



Comentarios y Conclusiones

Bibliografía y Referencias