A thick dark green vertical bar runs down the left side of the page. A green arrow-shaped banner points to the right from this bar, containing the date. Below the banner, several thin, curved lines in dark green and light grey sweep upwards from the bottom left corner.

15-12-2021

Interrupciones externas

Martínez Coronel Brayan Yosafat

3CM17 Introducción a los microcontroladores
FERNANDO AGUILAR SÁNCHEZ

Objetivo

Al término de la sesión, los integrantes del equipo contarán con la habilidad para manejar las interrupciones del Microcontrolador.

Introducción teórica

Las interrupciones externas son activadas por los pines INT1 y INT0. Si son habilitadas, las interrupciones se activarán aun cuando los pines INT0/INT1 se configuren como salidas.

Esta característica proporciona una manera de generar una interrupción por software. Las interrupciones externas pueden ser activadas por un flanco de bajada, subida o por un nivel bajo. Esto es establecido como se indica en la especificación para el registro de control MCU (MCUCR). Cuando la interrupción externa se habilita y se configura como activo por nivel, la interrupción se activará mientras el pin se mantenga a nivel bajo.

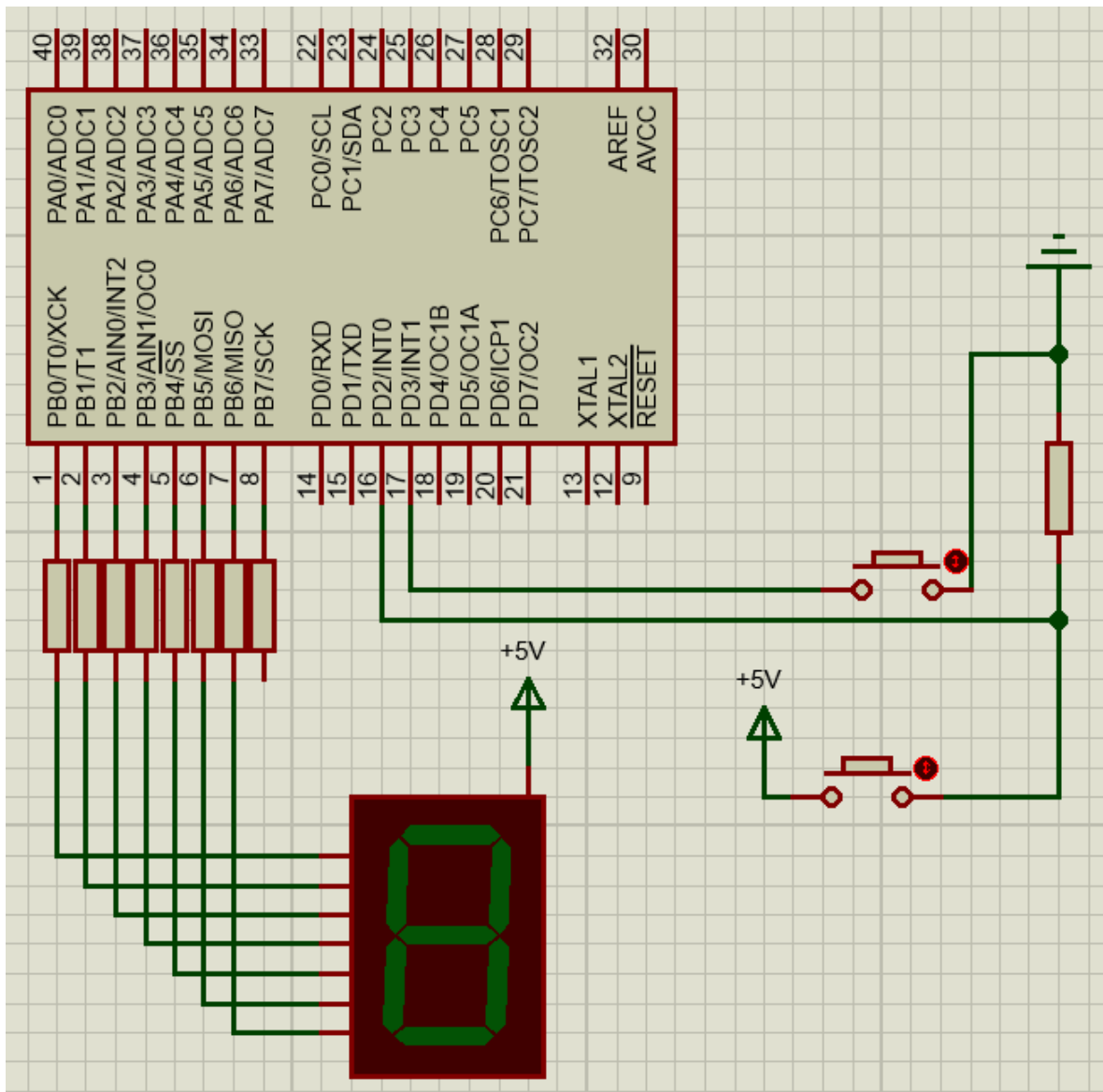
Las interrupciones externas se establecen como se describe en la especificación para el registro de control MCU (MCUCR).

Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 1 display cátodo común
- 2 push button
- 1 resistor de 1K

Desarrollo experimental

1.- Diseñe un programa que cuando haya un flanco de subida en INT0 se incremente el conteo del display que podrá contar de 0 a 9, y que cuando haya un nivel lógico de 0 en INT1 se decremente el valor del display.



Estructura del programa

/*****

This program was created by the CodeWizardAVR V3.46a

Automatic Program Generator

© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.

<http://www.hpinfotech.ro>

Project :

Version :

Date : 07/12/2021

Author :
Company :
Comments:

Chip type : ATmega8535
Program type : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model : Small
External RAM size : 0
Data Stack size : 128

*****/

```
#include <mega8535.h>
```

```
char indice = 0;
```

```
// External Interrupt 0 service routine
```

```
interrupt [EXT_INT0] void ext_int0_isr(void)
```

```
{
```

```
    indice++;
```

```
    if (indice == 10) indice = 0;
```

```
}
```

```
// External Interrupt 1 service routine
```

```
interrupt [EXT_INT1] void ext_int1_isr(void)
```

```
{
```

```
    if (indice == 0) indice = 9;
```

```
    else indice--;
```

```
}
```

```
// Declare your global variables here
```

```
const char mem[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};
```

```
void main(void)
```

```
{
```

```
// Declare your local variables here
```

```

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) |
(0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) |
(1<<DDB1) | (1<<DDB0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) |
(0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF

```

```
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01)
| (0<<CS00);
TCNT0=0x00;
OCR0=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
(0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

```
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
```

```
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21)
| (0<<CS20);
TCNT2=0x00;
OCR2=0x00;
```

```
// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);
```

```
// External Interrupt(s) initialization
```

```
// INT0: On
```

```
// INT0 Mode: Rising Edge
```

```
// INT1: On
```

```
// INT1 Mode: Low level
```

```
// INT2: Off
```

```
GICR|=(1<<INT1) | (1<<INT0) | (0<<INT2);
```

```
MCUCR=(0<<ISC11) | (0<<ISC10) | (1<<ISC01) | (1<<ISC00);
```

```
MCUCSR=(0<<ISC2);
```

```
GIFR=(1<<INTF1) | (1<<INTF0) | (0<<INTF2);
```

```
// USART initialization
```

```
// USART disabled
```

```
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) |
(0<<RXB8) | (0<<TXB8);
```

```
// Analog Comparator initialization
```

```
// Analog Comparator: Off
```

```
// The Analog Comparator's positive input is
```

```
// connected to the AIN0 pin
```

```
// The Analog Comparator's negative input is
```

```
// connected to the AIN1 pin
```

```
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
```

```
SFIOR=(0<<ACME);
```

```
// ADC initialization
```

```
// ADC disabled
```

```
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) |  
(0<<ADPS1) | (0<<ADPS0);
```

```
// SPI initialization
```

```
// SPI disabled
```

```
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) |  
(0<<SPR1) | (0<<SPR0);
```

```
// TWI initialization
```

```
// TWI disabled
```

```
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);
```

```
// Globally enable interrupts
```

```
#asm("sei")
```

```
while (1)
```

```
{
```

```
    PORTB = ~mem[indice];
```

```
}
```

```
}
```

Observaciones y Conclusiones

Sinceramente es mucho más fácil utilizar las interrupciones que a manejar los flancos uno mismo. Eso resulta muy útil si solo tenemos dos botones, además de que se ve mucho más ordenado de esta forma, me gusta cómo hemos avanzado hasta ahora, son un montón de prácticas y no puedo creer que hallamos hecho todas.

Bibliografía

Interrupciones

externas:

http://www.sc.ehu.es/sbweb/webcentro/automatica/web_avr/archivos/Manual_AT90S8515/Arquitectura/interrupt_externas&tiempo_respu.htm