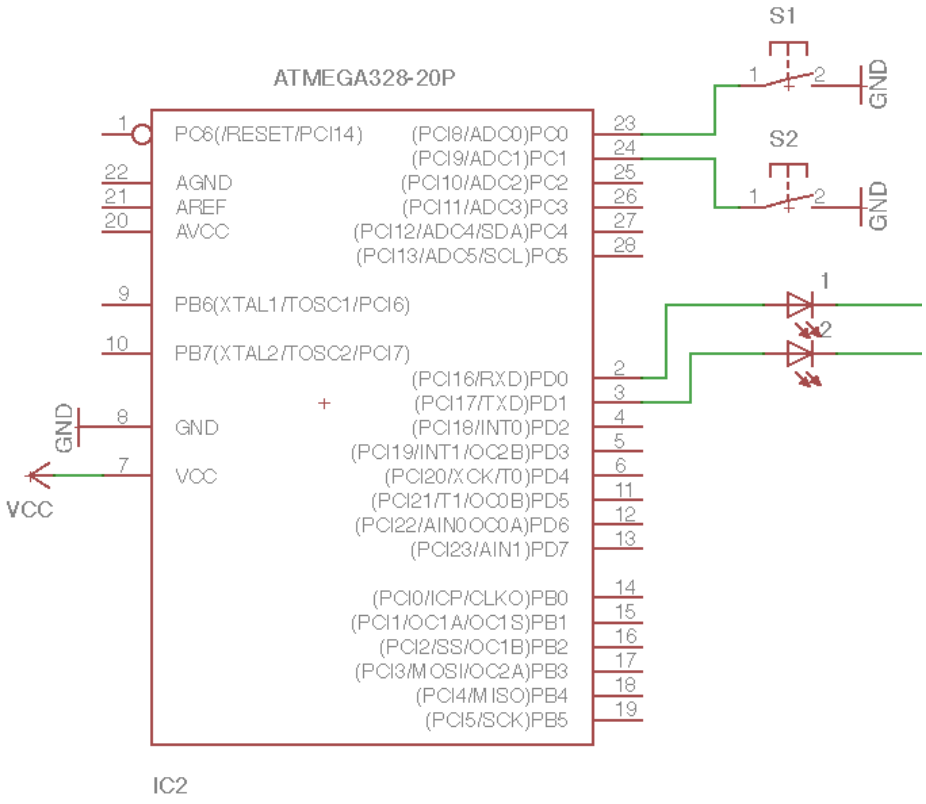
UNIVERSIDAD DE SANTIAGO DE CH
FACULTAD DE INGENIERÍA
Departamento de Ingeniería Mecánica
Área de Procesos Mecánicos

Correas transportadoras

Se desea construir un sistema de control para dos correas transportadoras independientes, cada una accionada por un motor paso a paso, de 200 pasos/rev, y un engranaje solidario de 30cm de diámetro. Al presionar el switch 1, el motor 1 debe girar lo necesario para permitir que la primera correa transportadora se mueva 3 metros, y en caso de ser presionado nuevamente, detenerse. Al presionar el switch 2, la segunda correa debe moverse hasta que el switch sea presionado nuevamente.

En el siguiente esquemático se muestra el circuito utilizado



Se pide:

- Configurar los puertos correspondientes, incluyendo resistencias Pull-Up
- Solucionar mediante el uso del timer0 del microcontrolador, utilizando un prescaler con un valor de 64 y utilizar interrupciones por overflow. Los pasos deben tener un delay de 50ms.
- Realizar debounce de ambos switch, sin utilizar librería delay.h
- Ambos motores deben poder funcionar de manera simultánea. (Utilizar Procesos)

Andres Utrera Soto

Solución:

```
#include <avr/io.h>
#include <avr/interrupt.h>//Libreria interrupciones

#define N_Procesos 4//Define numero de procesos a usar

#define PIN_botones PINC
#define Puerto_botones PORTC
#define S1 PC0
#define S2 PC1

#define DDR_motores DDRD
#define Puerto_motores PORTD
#define Motor1 PD0
#define Motor2 PD1

unsigned char S1_deb = 0;//Contadores debounce
unsigned char S2_deb = 0;

typedef struct//Define estructura de un proceso
{
	unsigned int IDTASK;
	unsigned long TIMER;
	unsigned long DELTA;
} PROC_STRUCT;

static volatile PROC_STRUCT Procesos[N_Procesos];//Crea numero de procesos

void config_procesos()
{
	Procesos[0].IDTASK = 0;//SW1
	Procesos[0].TIMER = 0;
	Procesos[0].DELTA = 13;//aprox 13ms


	Procesos[1].IDTASK = 0;//SW2
	Procesos[1].TIMER = 0;
	Procesos[1].DELTA = 13;


	Procesos[2].IDTASK = 0;//Motor 1
	Procesos[2].TIMER = 0;
	Procesos[2].DELTA = 1;

	Procesos[3].IDTASK = 0;//Motor 2
	Procesos[3].TIMER = 0;
	Procesos[3].DELTA = 1;
}

ISR(TIMER0_OVF_vect)//Interrupcion overflow Timer0
{
	for(unsigned char i=0;i <= N_Procesos-1;i++)
	{
		Procesos[i].TIMER++;//Suma 1 a .Timer en cada proceso
	}
}
```

```
void setup_hardware()
{
	Puerto_botones |= _BV(S1) |_BV(S2);
	DDR_motores |= _BV(Motor1) |_BV(Motor2);
}

void setup_timer()
{
	TCCR0B |= _BV(CS00) |_BV(CS01);//Prescaler 64
	TIMSK0 |= _BV(TOIE0);//Activa Interrupcion timer0
	sei();//Interrupciones globales
}


int main(void)
{
	setup_hardware();
	setup_timer();
	config_procesos();
	while (1)
	{
		if(Procesos[0].TIMER >= Procesos[0].DELTA)//Debounce switch 1, ejecuta cada "delta"
segundos
		{
			Procesos[0].TIMER = 0;//Reinicia contador de interrupciones
			if(bit_is_clear(PIN_botones,S1))//Si boton es presionado:
			{
				if(S1_deb < 250)
				{
					S1_deb++;//Cuenta mientras este presionado, con un maximo de 250,
para no hacer overflow
				}
			} else//Si el boton se suelta, o si no esta presionado
			{
				if(S1_deb > 35)//Si S1_deb logro contar al menos hasta 35, realizar accion
				{
					//Hacer algo
					if(Procesos[0].IDTASK == 0)//IDTASK de procesos 0 permite conocer si
el boton fue activado correctamente (Por debounce)
					{
						Procesos[0].IDTASK = 1;
					} else
					{
						Procesos[0].IDTASK = 0;
					}
				}
				S1_deb = 0;//Reinicia la variable contador.
			}
		}
```

```
		if(Procesos[1].TIMER >= Procesos[1].DELTA)//Debounce switch 2, lo mismo que en switch 1
		{
			Procesos[1].TIMER = 0;
			if(bit_is_clear(PIN_botones,S2))
			{
				if(S2_deb < 250)
				{
					S2_deb++;
				}
			} else
			{
				if(S2_deb > 35)
				{
					//Hacer algo
					if(Procesos[1].IDTASK == 0)//IDTASK de procesos 1 permite conocer si el boton fue activado correctamente (Por debounce)
					{
						Procesos[1].IDTASK = 1;
					} else
					{
						Procesos[1].IDTASK = 0;
					}

				}
				S2_deb = 0;
			}
		}


		if(Procesos[0].IDTASK == 1)//Si el switch 1 fue accionado correctamente:
		{
			if(Procesos[2].TIMER >= Procesos[2].DELTA)//Realiza lo que esta dentro de este if cada "delta" segundos
			{
				Procesos[2].TIMER = 0;
				if(Procesos[2].IDTASK <= 637*2)//Cantidad de pulsos a ejecutar para recorrer 3 metros, y *2 ya que son 2 ciclos del micro por cada paso (High - low > 1 paso)
				{
					Puerto_motores ^= _BV(Motor1);
					Procesos[2].IDTASK++;//Cuenta los pasos a ejecutar, dados por el arco a rrecorrer
				} else
				{
					Procesos[2].IDTASK = 0;//Reinicia contador de pasos
					Procesos[0].IDTASK = 0;//Desactiva "estado" del switch
				}

			}
		} else
		{
			Puerto_motores &=~_BV(Motor1);//Si switch 1 pasa a estar inactivo, apaga el motor
			Procesos[2].IDTASK = 0;
		}
```

```
		if(Procesos[1].IDTASK == 1)//Si switch 2 fue presionado:
		{
			if(Procesos[3].TIMER >= Procesos[3].DELTA)//Ejecuta cada "Delta" segundos
			{
				Procesos[3].TIMER = 0;
				if(Procesos[3].IDTASK == 0)//Si esta apagado, enciende
				{
					Puerto_motores |= _BV(Motor2);
					Procesos[3].IDTASK = 1;//cambia a "estado encendido"
				} else
				{
					Puerto_motores &= ~_BV(Motor2);//Si esta encendido, apaga
					Procesos[3].IDTASK = 0;//Cambia a "estado apagado"
				}

			}
		} else
		{
			Puerto_motores &= ~_BV(Motor2);//Si switch 2 pasa a estar inactivo, apaga el motor
		}


	}
}
```

Calculo contador TIMER para 50ms:

$$\frac{F\_CPU}{PRESCALER} = Fercuencia \text{ "efectiva" } timer$$

$$\frac{16000000}{64}\left[\frac{Cuentas}{segundo}\right] * \frac{1}{256}\left[\frac{1}{\frac{Cuentas}{Interrupcion}}\right] = 976{,}56\left[\frac{interrupciones}{segundo}\right]$$

Luego, con regla de 3:

$$976{,}56 > 1[s]$$

$$x > 50 * 10^{-3}[s]$$

Luego, x es:

$$x = 49$$

Calculo pasos motor para 3 metros:

$$200\frac{pasos}{revolucion} * \frac{1\ revolucion}{2 * \pi * 0.15\ [m]} * 3[m] = 636{,}62\ pasos\ \approx 637\ pasos$$