Informe cuarta práctica: HSI

Grupo SDMI 22E

David Guerra

Roman Valls

Daniel Clemente

Se pide implementar el tacómetro de un coche, de forma que se muestre la velocidad de las ruedas por la pantalla LCD en km/h, m/s o tics/segundo.

Una opción para medir la velocidad es apuntar el valor del Timer1 cada vez que llegue un pulso HSI, y hacer la resta tiempo final - tiempo inicial para saber cuánto rato ha pasado. Pero como la rueda del coche puede girar muy rápido y producir muchos tics por segundo, no sería muy eficiente. Este enfoque será más apropiado para el anemómetro de la estación meteorológica.

El método usado es contar tics y tiempo de manera independiente, y, cada cierto tiempo, hacer la división tics/tiempo para obtener la velocidad. De esta forma, las RSI ejecutan código muy sencillo.

Nuestro código usa el Timer2 para contar, en modo "normal", por tanto sumando 1 por cada 8 flancos del Timer1. Actualizamos el contenido de la pantalla LCD sólo después de que el Timer2 se haya desbordado 5 veces, pero en cada desbordamiento aprovechamos para ir calculando el promedio, de forma que el valor mostrado es la media de las velocidades que ha habido durante los últimos 5 desbordamientos.

En vez de mostrar el valor en "número de tics por overflow", aplicamos un factor de conversión k, que permite pasar el resultado a m/s o km/h.

Se tiene en cuenta:

- Número de overflows del Timer2 que ocurren cada segundo. Tenemos 20 Mhz (20 M ciclos por segundo), que a razón de 2 ciclos por state time, son 10 M st. Contamos 1 de cada 8 st, por tanto contamos 10M / 8 = 1310720 tics. El Timer2, de 16 bits, se desborda por tanto 1310720 / (2^16) = 20 veces cada segundo.
- Número de tics por vuelta de la rueda. Con una sencilla y poco precisa, será 1 tic por vuelta.
- Perímetro de la rueda. Hemos usado 2 metros.
- Conversión de m/s a km/h.

Montaje de la placa:

Para simular los pulsos generados por un tacómetro se ha utilizado un generador de señal, éste se conecta al circuito mediante un conector BNC; al ser una señal analógica puede darse el caso de que "falsos pulsos" generen entradas en la HSI, para evitarlo se utiliza una puerta en histéresis. De esta manera nos aseguramos de que sólo se generará una HSI al haber un flanco.

Las conexiones del patillaje se han hecho de la siguiente manera:

- Conector BNC: Carcasa externa (GND) a pin 1 del JP2; pin interno a patilla 3 del trigger.
- Trigger: En la patilla 3 (entrada) se ha conectado la entrada del conector BNC; la patilla 4 (salida) se ha conectado al pin 12 del JP2 (HSI.0).

Durante la fase de pruebas del circuito nos encontramos con un problema, uno de las puertas en histéresis integradas del 40106 era defectuosa, esto provocó que al cargar el programa de prueba el circuito no funcionara correctamente (de hecho no contaba), al principio creímos que se trataba de un mal montaje, pero tras revisar varias veces las conexiones no conseguimos encontrar el fallo, la pista nos la dio un fluorescente, al acercarle el cable BNC el circuito contaba debido a la frecuencia de 50 hz del fluorescente; esto descartaba el hecho de que el circuito estuviera mal montado. La solución consistió en buscar dónde se perdía la señal de entrada con la ayuda del osciloscopio.

Hsi.c: inicializaciones

Queremos contar el número de ticks que se producen así como las interrupciones por overflow que genera el timer2. Así pues, configuramos los registros del micro como sigue:

Debemos modificar int_mask e int_mask1 para que, por un lado la HSI notifique que un dato ya está disponible en el holding register, y por otro lado activar la interrupción del timer2 que nos avisa cuándo se produce un overflow.

Adicionalmente, debemos especificar para HSI0 (escogida entre las disponibles por conveniencia con el cableado), cuándo se producirá una interrupción, esto se consigue escribiendo en IOC1.7, tal como se indica en apbuilder:

- 1= HSI FIFO Full
- 0= HSI Holding Register loaded

Sólo queremos saber cuándo esta disponible el dato capturado por la HSI. En consecuencia, escogemos poner a 0 el bit de IOC1.7.

A continuación, seleccionamos el comportamiento del timer2, activamos los siguientes parámetros:

- Clock interno.
- Contar UP.
- Valor inicial de timer2: 0.

Finalmente, configuramos la HSI0 para que cuente cada 8 flancos de subida o transiciones positivas, configuración razonable para no sobrecargar innecesariamente el microcontrolador.

El código:

```
wsr=0x00;
int_mask |= 0x04;     // HSIData Available, cal configurar IOC1.7
int_mask1 |= 0x10;    // T2OVF

wsr=0x0f;
auxiliar=ioc0;

wsr=0x00;
auxiliar |= 0x01;
ioc0=auxiliar;                // activo hsi0
auxiliar &=   0xf7;
ioc0=auxiliar;                // desactivo EXTRST del timer2

wsr=0x0f;
auxiliar=ioc1;

wsr=0x00;
auxiliar &= 0x7f;
ioc1=auxiliar;                // La Hsi Data Available (HSIINT) es generada quan
                      // es carrega el Holding Register

wsr=0x0f;
auxiliar=ioc2;
wsr=0x00;
auxiliar |= 0x02;
ioc2=auxiliar;                // Configurem el TIMER2 per comptar UP en normal mode, al rang
                      // FFFFh-0000h.

wsr=0x01;
ioc3 = ioc3 | 0x01;   // Seleccionem el clock intern pel TIMER2

wsr=0x00;
timer2=0;                     // Carreguem els valors inicials als comptadors

wsr=0x0f;
auxiliar=hsi_mode;

wsr=0x00;
auxiliar |= 0x01;     // cada transició positiva HSI.0 provocarà interrupció ...
auxiliar &= 0xfd;     // ... pero esperem 8 transicions positives (clocks)
hsi_mode = auxiliar;

/*  Inicialitzem les variables */

tics=0;
conta_ovf=0;
total=0; // Usada para hacer el promedio
```

Hsi.c: bucle principal

```
 CLRSCR_LCD();

 while(1)
      {

// En general:

//   1 mostra = N overflows
//  x tics    1 mostra    1 volta      q m      1 km    3600 s
// -------- * --------- * -------- * ------- * ------ * ------ = ?
// 1 mostra     z seg      p tics    1 volta   1000 m     1h


// En este caso:

//   1 mostra = 1 overflows
//  x tics    ops mostra    1 volta    perim m    1 km    3600 s    km
// -------- * ---------- * -------- * ------- * ------ * ------ = ----
// 1 mostra     1 seg      tpv tics   1 volta   1000 m     1h       h

// Número de overflows por segundo
#define ops 20

// Número de tics que se generan en cada vuelta de la rueda
#define tpv 1

// Perímetro de la rueda, en metros. 2*pi*radio
#define perim 2



//#define k ( (ops*perim*36)/(tpv*10) )
// Sustiyendo por los valores que aplicaremos, queda:
#define k 144

// Sin factor
// #define k 1


// En cada OVF, mirar cuántos tics nos han llegado
// 'tics' se puso a 0 en la visualización anterior

// Actualizar la pantalla cada 5 overflows
if(conta_ovf>5){

    // 'tics' tiene el número de tics por 1 overflow
    // Aplicar factor de conversión k para mostrarlo en otra escala

      conta_ovf=0;

      // escriu_valor_lcd(k*tics,8,1);
      escriu_valor_lcd(k*(total/5),8,1);

      //tics=0;
      total=0;


  }


}
```

Hsi.c: RSIs

```
void RSI_Hsi() {
      // Comptar els numero de tics que arriven
      asm {di;}
      tics++;                                 //contamos tics q van llegando
```

```
	// Leer hsi_time para cargar el siguiente valor
	wsr=0x0 ;
	auxiliar=hsi_time;

	asm {ei;}
}

void RSITimer2() {
	// Incrementa el comptador de interrupcions del timer i avisa que sa produit una interrupcio
	// i reprograma el timer2
	asm {di;}

	total+=tics;
	tics=0;

	conta_ovf++;                    //contamos overflows de tiempo, por si las HSI "tardan"

	// El timer2 sigue contando, automáticamente

	asm {ei;}
}
```

Sthsi.a96: VI's para las INTs

```
; Interrupció de "Timer 2 Overflow" (INT12)
cseg at 0d180h
        br TIMER2_OVF

; Interrupcio de "Data available" del HSI (INT2)
cseg at 0d040h
        br HSIINT
```