Universidad Nacional de La Plata
Facultad de Ingeniería

Trabajo practico 1-C

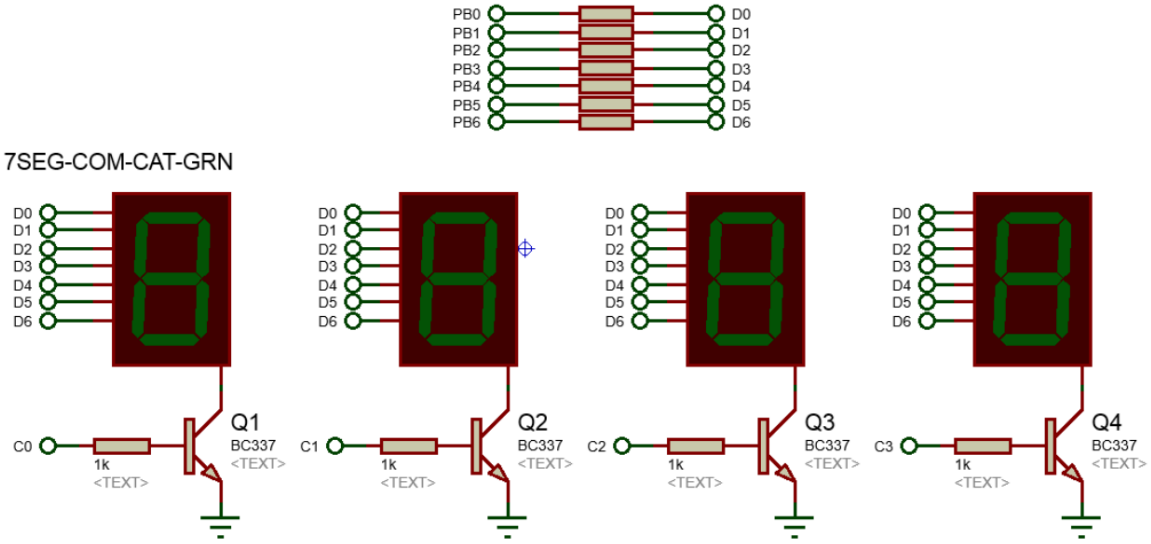
Cronometro 4 dígitos

Gallo, Molinuevo Francisco

Pérez, Francisco

Problema:

3. Realice un programa para implementar un reloj cronometro de 4 dígitos (decimas de segundos, segundos, décimas y centésimas) que se actualiza cada 10ms. Para mostrar el valor actual de conteo deberá implementar la técnica de multiplicación de los dígitos con el circuito mostrado en la figura y con una temporización adecuada. Elija un retardo adecuado para la visualización en el simulador (no será en tiempo real ya que el tiempo del simulador dependerá de los recursos disponibles en la computadora donde se ejecute).



Interpretación:

Junto con el ATMEGA328 y cuatro display de 7 segmentos se debe simular un cronometro.

Se utilizarán varios puertos como salida. Primero, se utilizará el Puerto B del 0 al 6 para mandar a cada display el número que le corresponde.
Como los display de 7 segmentos son independientes de cada uno, es decir, un display solo puede representar un número del 0 al 9, debemos multiplexar el número a representar inicialmente. Además, se tiene que tener en cuenta que solo 1 display puede estar encendido a la vez y mostrar el número que le corresponde. Ya que, como se mencionó, el puerto de salida es compartido. Se tiene que sincronizar que display encender y que valor mandar al puerto B. Para esto, se empleará 4 pines del puerto C.
Se utilizará 4 variables donde en cada una se guardará el valor de uno los 4 dígitos

Solución:

Multiplexacion:

Para llevar a cabo el cronometro, se utilizará un contador el cual puede tomar valores desde el 0 al 9999, donde cada valor debe ser visualizado en los 4 display a disposición.

Como los display necesitan una codificación a BCD (rango entre 0 y 9), se debe obtener cada digito del entero y guardarlos en varias variables para luego mandar cada número a su display correcto.

Por ejemplo, si se tiene el valor 3204, se debe obtener de ese número, el 4, el 0, el 2 y el 3 por separado, guardarlos en variables distintas y mandar cada uno a cada display.

La solución es un algoritmo que los obtiene y los guarda en una variable por separado (ver Pseudocodigo 1.)

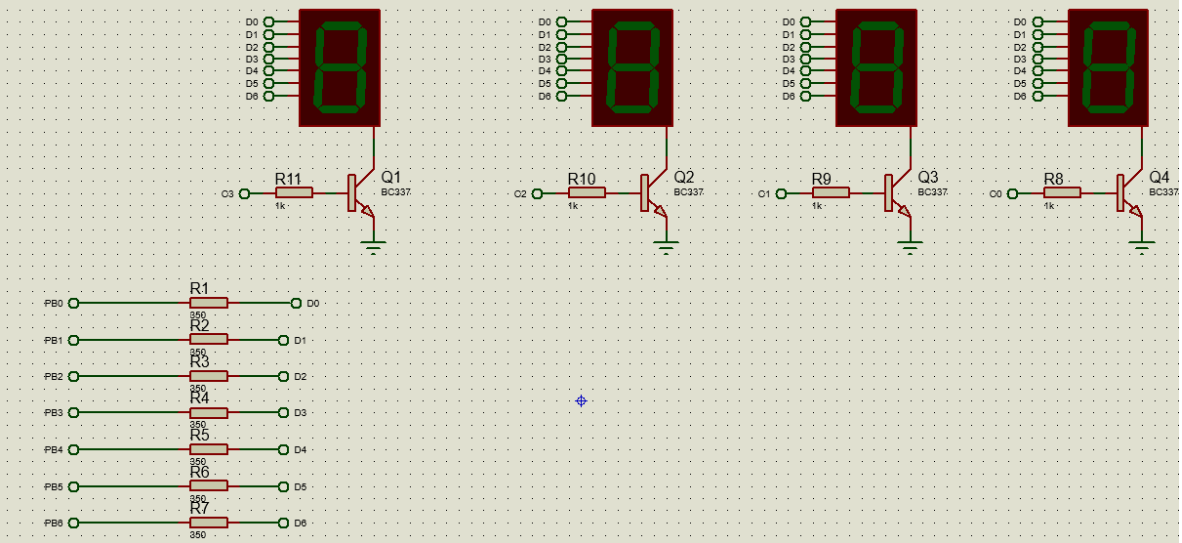
```
Obtener el resto del entero para obtener el digito
Guardarlo en una variable

Si el número es distinto de 0 (Numero > 9)
   Dividir numero por 10 y sacar el primer digito
   Obtener el resto del entero al dividirlo por 10
   Guardar décima en otra variable

   Si el número es distinto de 0 (Numero > 99)
      Dividir numero por 10 y sacarnos la decima
      Dividir el numero por 10 y obtener el resto
      Guardar centena en otra variable

      Si el número es distinto de 0 (numero > 1000)
         Dividir numero por 10 y sacar la centena
         Dividir el numero por 10 y obtener el resto
         Guardar milésima en otra variable
```

Pseudocodigo 1: separarDigitos()

De esta forma podemos tener 4 variables donde estas representen cada digito del número inicial.

Podemos construir un arreglo de 10 números (0..9) donde en cada posición están los números codificados en BCD. Esto facilita que señal mandar al puerto B solo consultando este vector en la posición del número que queremos representar.

Ahora, se debe solucionar el problema de la incapacidad de mandar cada variable a un puerto distinto y que los display muestren su número correspondiente en el mismo instante de tiempo. (Ver ilustración 1.)



Ilustracion 1.

(PBx es la conexión al puerto B)

De la *ilustración 1.* se puede observar como cada entrada de los distintos display tiene una conexión al puerto B, por lo tanto, esta es compartida. Si mandásemos una señal en B y todos los display están encendidos, se verá el mismo número repetido 4 veces lo cual no es lo esperado.

Se utilizará otro puerto (en este caso el C) para que, en un cierto instante de tiempo se muestre solo un display encendido (con su número correspondiente), luego, el display siguiente y así. (Ver *Pseudocodigo 1.2*)

```
Prender solo display 1
Mandar al puerto B el primer digito en BCD
Prender solo display 2
Mandar al puerto B el segundo digito en BCD
Prender solo display 3
Mandar al puerto B el tercer digito en BCD
Prender solo display 4
Mandar al puerto B el cuarto digito en BCD
```

Pseudocodigo 1.2: showDisplays()

La frecuencia en lo que sucede el prendido y apagado de display es tan rápida que el ojo humano no puede percibir este cambio, lo que dará la sensación de que todos los display siempre están encendidos.

El puerto C será configurado como salida, en este caso será suficiente utilizar solo 4 pines.

La solución circuital es que, en un instante de tiempo, si en un pin de C, tiene un valor alto, el display conectado a ese pin estará encendido, mientras que, si tiene un valor bajo, este estará apagado (ver ilustración 1.1)

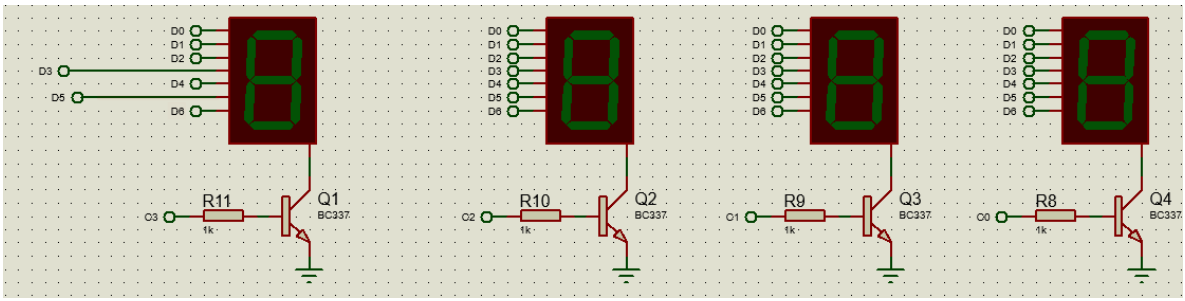


Ilustración 1.1

Se puede observar las conexiones, donde C0, C1, C2, C3 son los pines correspondientes del puerto C del microcontrolador.

Programa

```
Inicializamos contador

Lazo infinito
  Obtenemos cada digito de la variable contador
  Incrementamos contador
  Mostramos en el display cada variable
```

Pseudocodigo 1.3 main.c

Como comentario, el obtener cada digito y mostrar en el display son funciones donde su Pseudocodigo ya fue explicado con anterioridad.

Conclusion:

Cuando se exporta la solución del programa a Proteus, se puede ver que funciona correctamente la sincronización de mostrar cada display en un instante de tiempo y su correspondiente digito a representar, la visualización en Proteus no es acorde a los tiempos que se utilizan en el programa ya que depende de cada computadora en donde se ejecute. Para poder visualizarlo mejor en Proteus hay que modificar los delay. Modificando estos tiempos ya no representaría un cronometro en segundos, pero podría a ayudar a verificar que la lógica.

Anexo

```
#include <avr/io.h>
#define F_CPU 16000000UL // Defino la frecuencia de oscilador en 16MHz
#include <util/delay.h>

const float DELAY = 2.5;
const uint8_t BCDTABLE[10] = {0b00111111, 0b00000110, 0b01011011, 0b01001111,
0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01101111};

// Obtener centesima de segundo, segundo, decima y centesima del contador;
void separarDigitos(uint16_t* contador, uint8_t* digito1, uint8_t* digito2, uint8_t*
digito3, uint8_t* digito4) {
		uint16_t aux = 0;
		aux = *contador;
		if(aux != 0){
			*digito1 = aux % 10;
			aux = aux / 10;
		}
		if(aux !=0){
			*digito2 = aux % 10;
			aux = aux / 10;
		}
		if(aux != 0){
			*digito3 = aux % 10;
			aux = aux / 10;
		}
		if(aux != 0){
			*digito4 = aux % 10;
		}
	}

// Por cada DELAY ms, muestro el valor que le corresponde a cada digito en el
dispositivo de 7 segmentos.
void showDisplays(int digito1, int digito2, int digito3, int digito4){
		PORTB = BCDTABLE[digito1];
		PORTC = 0b00000001;
		_delay_ms(DELAY);
		PORTB = BCDTABLE[digito2];
		PORTC = 0b00000010;
		_delay_ms(DELAY);
		PORTB = BCDTABLE[digito3];
		PORTC = 0b00000100;
		_delay_ms(DELAY);
		PORTB = BCDTABLE[digito4];
		PORTC = 0b00001000;
		_delay_ms(DELAY);
		PORTB = BCDTABLE[digito1];
		PORTC = 0b00000001;
	}
```

```c
int main(void)
{
	uint16_t contador = 0;
	uint8_t digito1 = 0;
	uint8_t digito2 = 0;
	uint8_t digito3 = 0;
	uint8_t digito4 = 0;
	uint8_t start = 0;

	DDRB = 0xFF; // Seteamos todos los puertos de B como Output
	DDRC = 0x0F; // Seteamos los primero 4 puertos de C como Output
	// Los primeros 4 puertos los utilizamos para definir que segmento mostramos
en ese instante de tiempo

    while (1)
    {
		separarDigitos(&contador, &digito1, &digito2, &digito3, &digito4);
		contador++; //Como tenemos 4 delay de 2,5 ms. La suma nos da 10ms.
Como es la unica seccion en donde se utiliza el delay() esta variable contador se
incrementara cada 10 ms como pide el enunciado.
		showDisplays(digito1, digito2, digito3, digito4);
	}

}
```