PRÁCTICA 2

Semáforo de crucero con TM4C123C (tiva chica).

Universidad Autónoma de Querétaro

Microcontroladores.

Aguillón Olamendi Sonia P.
Elias Sánchez María Monserrath
Moreno González Alberto
Nieto Guerrero Andrea

29 de Enero del 2023

Máquina de estados.

Una máquina de estados modela el comportamiento de un solo objeto, especificando la secuencia de eventos que un objeto atraviesa durante su tiempo de vida en respuesta a los eventos.

Las máquinas de estados se definen como un conjunto de estados que sirven de intermediarios en esta relación de entradas y salidas, haciendo que las salidas dependan no solo de las señales de entradas actuales, sino también de las anteriores. De modo que una máquina de estados es una representación de un circuito secuencial particular.

Metodología práctica 2.

Configuración PLL.

Para configurar el PLL se van a seguir los siguientes pasos.

1. Establecer el uso de RCC2

El registro RCC2 se encuentra en la página 260 de la datasheet.

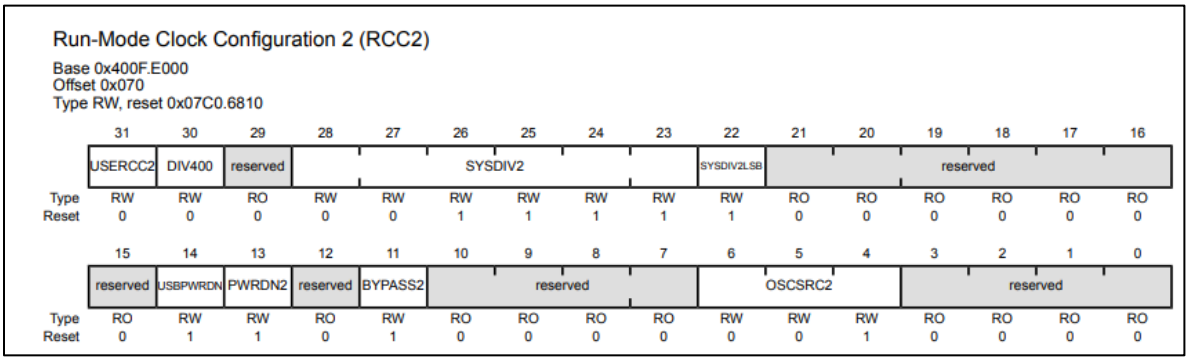

Figura 1. Registro RCC2

Para establecer su uso, basta con asignar un 1 en el bit 31 como se muestra en la figura 2.

Bit/Field	Name	Type	Reset	Description
31	USERCC2	RW	0	Use **RCC2**

Value	Description
0	The **RCC** register fields are used, and the fields in **RCC2** are ignored.
1	The **RCC2** register fields override the **RCC** register fields.

Figura 2. Bit 31 del registro RCC2.

```
SYSCTL->RCC2 |=  0x80000000;
```

Figura 3. Asignación del registro RCC2 en Visual Studio.

2. Inicializar bypass.

El bypass se establece en el mismo registro pero en el bit 11. Asignando un 1, en este bit.

11	BYPASS2	RW	1	PLL Bypass 2

Value	Description
0	The system clock is the PLL output clock divided by the divisor specified by SYSDIV2.
1	The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV2.

Figura 4. Registro RCC2.

```
SYSCTL->RCC2 |=  0x00000800;
```

Figura 5. Asignación del registro RCC2 en Visual Studio.

3. Seleccionar el valor del cristal y la fuente del oscilador

Estos valores se establecen en el registro RCC, registro que se encuentra en la página 254.

Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type RW, reset 0x078E.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			reserved
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	XTAL					OSCSRC		reserved			MOSCDIS
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

Figura 6. Registro RCC.

```
SYSCTL->RCC = (SYSCTL->RCC &~0x000007C0)   // clear XTAL field, bits 10-6
            + 0x00000540;   // 10101, configure for 16 MHz crystal
```

Figura 7. Asignación del registro RCC en Visual Studio.

```
SYSCTL->RCC2 &= ~0x00000070;  // configure for main oscillator source
```

Figura 8. Configuración de la fuente del oscilador en Visual Studio.

4. Activar el PLL
   Para activar el PLL, debemos limpiar a PWRDN, que se encuentra en el registro RCC2, en el bit 13.

```
SYSCTL->RCC2 &= ~0x00002000;
```

Figura 9. Activación del PLL en Visual Sstudio.

5. Ajustar el divisor del sistema

Primero establecemos que se va utilizar el PLL de 400Mhz, asignando un 1 en el bit 30 del registro RCC2.

30	DIV400	RW	0	Divide PLL as 400 MHz versus 200 MHz This bit, along with the SYSDIV2LSB bit, allows additional frequency choices. Value Description 0 Use SYSDIV2 as is and apply to 200 MHz predivided PLL output. See Table 5-5 on page 223 for programming guidelines. 1 Append the SYSDIV2LSB bit to the SYSDIV2 field to create a 7 bit divisor using the 400 MHz PLL output, see Table 5-6 on page 224.

Figura 10. Bit 30 del Registro RCC2.

```
SYSCTL->RCC2 |= 0x40000000;   // use 400 MHz PLL
```

Figura 11.Asignación del Bit 30 del Registro RCC2 en Visual Studio.

Ahora configuramos el reloj con una frecuencia de 80Mhz, para ello dividimos la frecuencia de fuente de reloj (400Mhz) entre 5, como se muestra en la figura 12 que se obtuvo de la página 223

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)
0x00	/1	reserved	Clock source frequency/1
0x01	/2	reserved	Clock source frequency/2
0x02	/3	66.67 MHz	Clock source frequency/3
0x03	/4	50 MHz	Clock source frequency/4
0x04	/5	40 MHz	Clock source frequency/5

Figura 12. Ejemplos de posibles frecuencias de reloj del sistema utilizando el campo SYSDIV2

```
SYSCTL->RCC2 = (SYSCTL->RCC2 &~ 0x1FC00000)  // clear system clock divider
               + (4<<22);      // configure for 80 MHz clock
```

Figura 13. Configuración del reloj a 80Mhz en Visual Studio.

6. Esperar que el PLL se interrumpa.

El estado de interrupción sin procesar del PLL, se establece en el registro RIS, que se encuentra en la página 245.

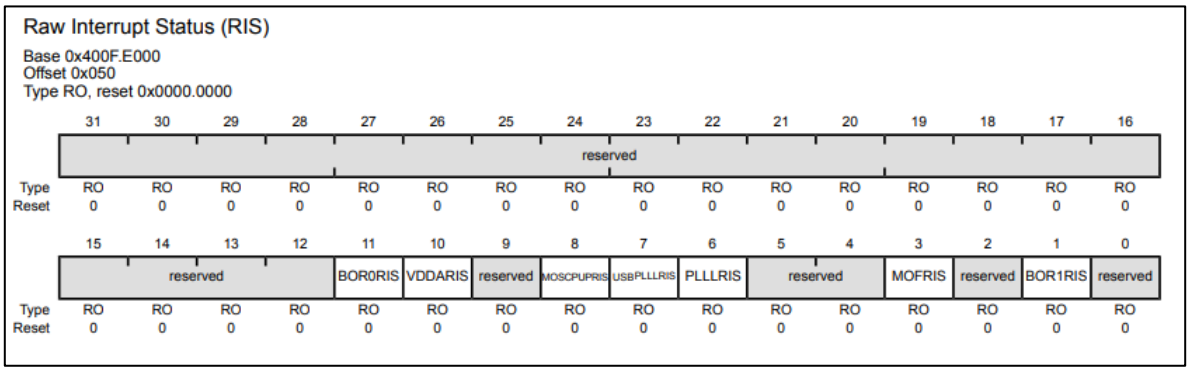

Figura 14. Registro RIS.

Para ello se asigna un 1 en el bit 6 como se muestra en la figura 15.

6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status

Value	Description
0	The PLL timer has not reached T_{READY}.
1	The PLL timer has reached T_{READY} indicating that sufficient time has passed for the PLL to lock.

Figura 15. Bit 6 del registro RIS.

```
while((SYSCTL->RIS&0x00000040)==0){};  // wait for PLLRIS bit
```

Figura 16. Asignación del registro RIS en Visual Studio.

7. Habilitar el uso del PLL.

Para habilitar el uso del PLL escribimos un 1 en el bit 11 del registro RCC2 como se muestra en la figura 17 y 18.

11	BYPASS	RW	1	PLL Bypass

Value	Description
0	The system clock is the PLL output clock divided by the divisor specified by SYSDIV.
1	The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV.

See Table 5-4 on page 223 for programming guidelines.

Note: The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Figura 17. Bit 11 del registro RCC2.

```
SYSCTL->RCC2 &= ~0x00000800;
```

Figura 18. Asignación del bit 11 del registro RCC2 en Visual Studio.

Funcionamiento del semáforo.

El semáforo que se va a implementar cuenta con una intersección de dos caminos de sentido único con la misma cantidad de tráfico: Norte y Este. En este sistema, el patrón de luces define qué camino tiene preferencia sobre el otro. Dado que es necesario un patrón de salida a las luces para permanecer en une estado, resolveremos este sistema con una FSM de Moore. La cual tendrá dos entradas (sensores automóviles en las carreteras norte y este) y seis salidas (una para cada luz de semáforo)

La máquina de estados para el semáforo de crucero se muestra en la figura 19.

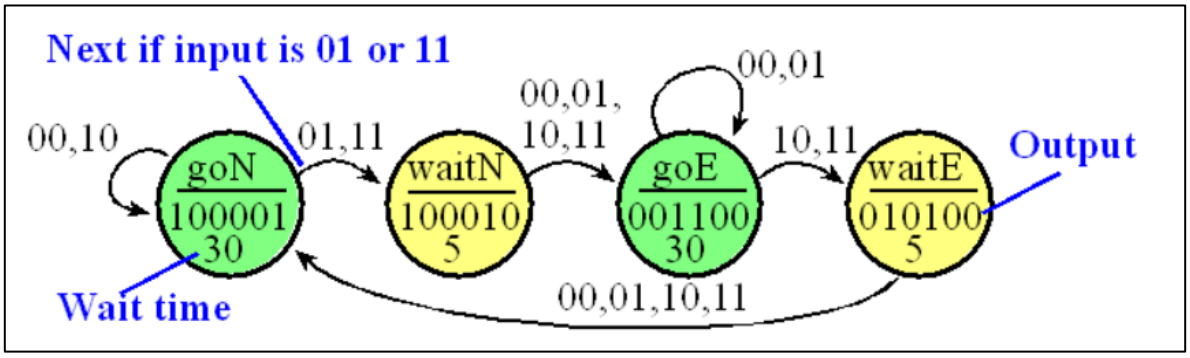

Figura 19. Máquina de estados

Para las entradas vamos a utilizar el puerto E (pines E0 Y E1) y para las salidas se va hacer uso del puerto B (pines B0-B5).

Los valores para las luces de ambos semáforos en cada uno de los estados se muestran en la siguiente tabla:

	EAST			NORTH		
	PB5	PB4	PB3	PB2	PB1	PB0
goN	1	0	0	0	0	1
waitN	1	0	0	0	1	0
goE	0	0	1	1	0	0
waitE	0	1	0	1	0	0

Estos valores no van a cambiar hasta que el Puerto E reciba valores de entrada.

El diagrama de la figura 20, describe de manera gráfica las conexiones para los pines E y B.

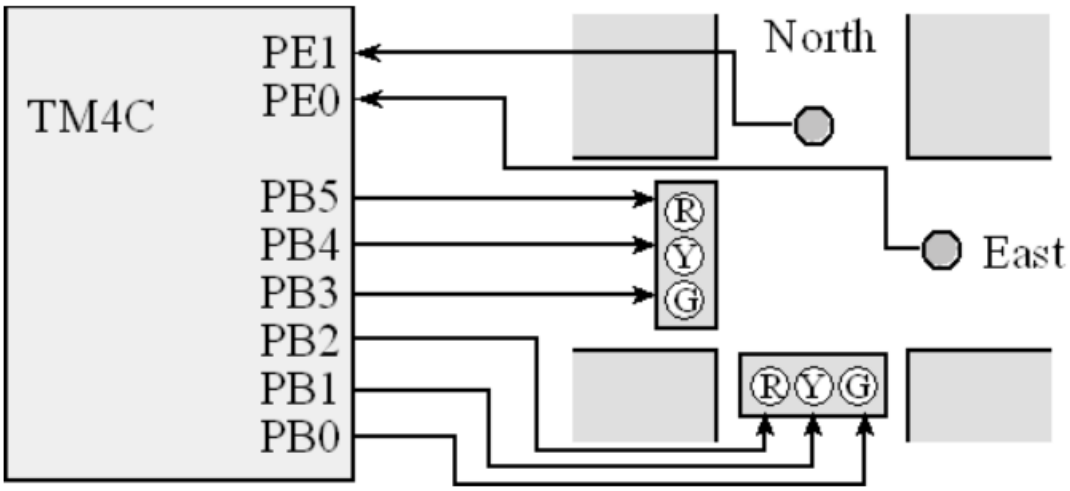

Figura 20. Conexión del puerto B y puerto E.

Ya que se explicó el funcionamiento de la práctica, se procede a codificar el algoritmo como se muestra a continuación.

Programa principal

Se comienza por definir las constantes sensor y ligth para utilizarlas en el programa.

```
#define SENSOR  (*((volatile uint32_t *)0x4002400C))
#define LIGHT   (*((volatile uint32_t *)0x400050FC))
```

Figura 21. Definición de constantes

Ahora se codifica la estructura de State.

```
struct State {

  uint32_t Out;
  uint32_t Time;
  uint32_t Next[4];};

typedef const struct State State_t;
```

Figura 22. Estructura para State

Se definen los estados goN, waitN, goE y waitE para utilizarlos

```
#define goN   0
#define waitN 1
#define goE   2
#define waitE 3
```

Figura 23. Definición de los 4 estados

Ahora se asignan los valores hexadecimales y el tiempo que conlleva cada uno de los estados:

```
State_t FSM[4]={

{0x21,300,{goN,waitN,goN,waitN}},
{0x22, 50,{goE,goE,goE,goE}},
{0x0C,300,{goE,goE,waitE,waitE}},
{0x14, 50,{goN,goN,goN,goN}}
};
```

Figura 24. Asignación de los valores para los 4 estados.

Dentro del main, asignamos los valores para el registro GPIO, de los puertos B y E, habilitando B como salidas digitales y E como entradas digitales.

```
int main(void){ volatile uint32_t delay;
  PLL_Init();       // 80 MHz, Program 10.1
  SysTick_Init();   // Program 10.2

  SYSCTL->RCGCGPIO |= 0x12;      // B E
  delay = SYSCTL->RCGCGPIO;      // no need to unlock
  GPIO_Init();
  GPIOE->DIR &= ~0x03;   // inputs on PE1-0
  GPIOE->DEN |= 0x03;    // enable digital on PE1-0
  GPIOB->DIR |= 0x3F;    // outputs on PB5-0
  GPIOB->DEN |= 0x3F;    // enable digital on PB5-0
  S = goN;
```

25. Asignación de valores para el registro GPIO.

Finalmente, dentro de un ciclo infinito, se codifica el algoritmo para la lectura de sensores y salida de las luces para cada uno de los semáforos.

```
while(1){

  LIGHT = FSM[S].Out;  // set lights
  SysTick_Wait10ms(FSM[S].Time);
  Input = SENSOR;     // read sensors
  S = FSM[S].Next[Input];
}
```

26. Lectura de sensores y salida de luces.

RESULTADOS

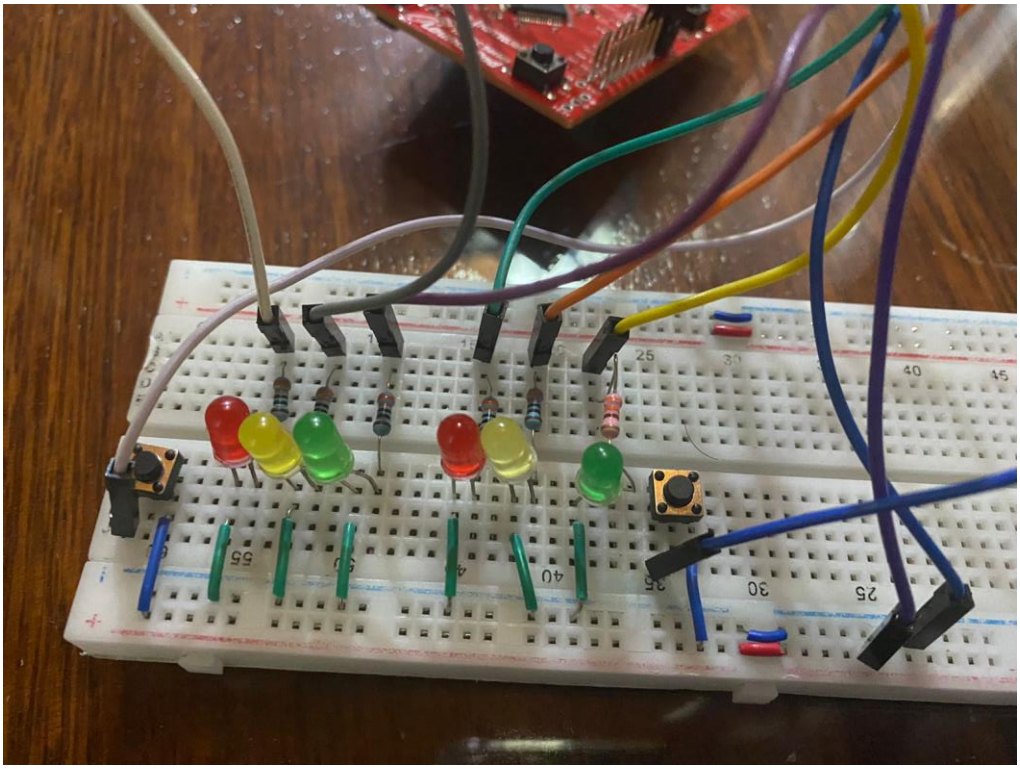

Figura 27. Conexión del circuito.

Los resultados se muestran en el vídeo en el siguiente enlace:
https://www.youtube.com/watch?v=k8XpzCwFMOg