UNIVERSIDAD DE COSTA RICA

ESCUELA DE INGENIERÍA ELÉCTRICA

IE0624
Laboratorio de Microcontroladores

Laboratorio II

GPIOs, Timers y FSM

Estudiantes:
Kendall Saborio Picado - B87103
Alexander Rojas Brenes - B86869

Profesor:
Ing. Marco Villalta Fallas

Fecha de entrega: 14/04/2021

Índice

1. INTRODUCCIÓN

Dentro del vasto terreno de la electrónica digital y la computación embebida, los microcontroladores destacan como elementos cruciales. Estos dispositivos, al reunir en un solo chip un procesador, memoria, periféricos de entrada/salida y otras funcionalidades, se posiciona como la columna vertebral de una amplia gama de aplicaciones tecnológicas. Su capacidad para gestionar y coordinar diversas tareas en tiempo real los convierte en componentes esenciales en la actual era digital, donde la demanda de sistemas compactos, rápidos y energéticamente eficientes es cada vez mayor. El objetivo de este laboratorio es construir un simulador de un cruce peatonal de unidireccional, utilizando como base el microcontrolador ATtiny4313, así como el lenguaje de programación C y algunos elementos pasivos y activos que complementan el diseño.

A lo largo de este informe se documenta la forma en la que fueron puestos en práctica una serie de conceptos básicos de GPIO, *timers* y máquinas de estado finito. Lo anterior se logró utilizando el lenguaje de programación C para la construcción de la lógica computacional de fondo, el simulador SimulIDE para diseñar el circuito y realizar las pruebas de funcionamiento, además de los conocimiento adquiridos a lo largo de la carrera sobre circuitos lineales y electrónica. Asimismo, se detalla el proceso de cálculo de las magnitudes físicas de los componentes utilizados, así como la lógica de programación utilizada en las funciones construidas para llegar al resultado final. Por último, se muestran los resultados obtenidos en los que puede verse un simulador de cruce peatonal completamente funcional y apegado a las especificaciones solicitadas en el enunciado. Para consultar el trabajo realizado, puede consultarse el siguiente repositorio de trabajo: `https://gitlab.com/arbre29/laboratorio_de_microcontroladores.git`. El código en C, archivo de simulación .simu, el archivo READ.md y el Makefile puede consultarse en la ruta `Laboratorio_02/src`.

2. NOTA TEÓRICA

Antes de iniciar con el uso y manipulación del lenguaje de programación C para la creación de los algoritmos objetivo del presente laboratorio, es necesario tener a mano una serie de conceptos importantes, los cuales se explican a continuación:

2.1. Microcontrolador ATtiny4313

Tal como se detalla en la hoja del fabricante [1], este componente corresponde a un microcontrolador de 8 bits, de alto rendimiento y de baja potencia, producido por la empresa Atmel. Cuenta con 120 instrucciones, una memoria flash con capacidad de 4 kb, una SRAM de 258 bytes y una memoria EEPROM de 258 bytes. Asimismo, cuenta con 18 GPIO programables, 2 timers (uno de 8 y otro de 16 bits), cuatro canales PWM, además de USI y una USART *full duplex*. En la Figura 1 puede observarse la distribución de pines del microcontrolador y sus respectivas funciones:

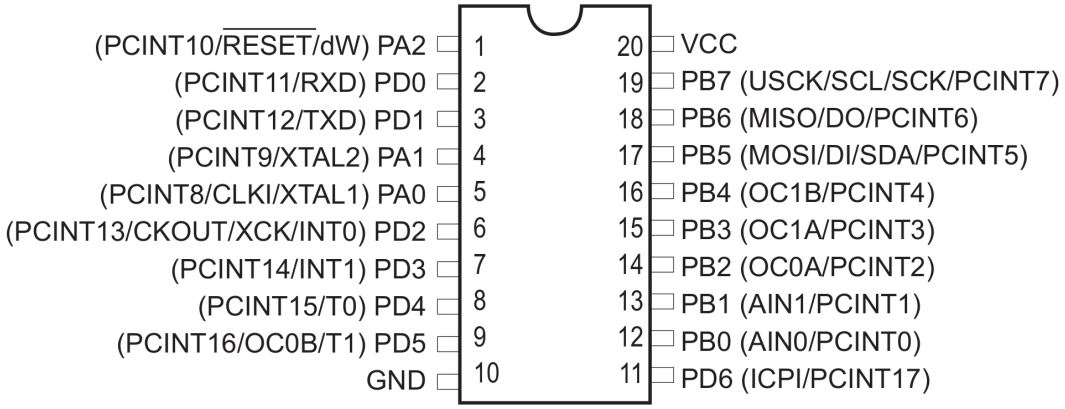


Figura 1: Diagrama de pines y sus respectivas funciones. (Fuente: Imagen tomada de [1])

Para el presente laboratorio, sólo serán de utilidad los pines que tienen la función de GPIO para la entrada y salida de información, además de las lineas de interrupción y temporización. Para más información general del dispositivo, observar el Anexo 5.1. Por otro lado, en la Figura 2 puede consultarse el diagrama de bloques del microcontrolador en el cual puede consultarse a alto nivel la conexión interna de este (no incluye el CPU):

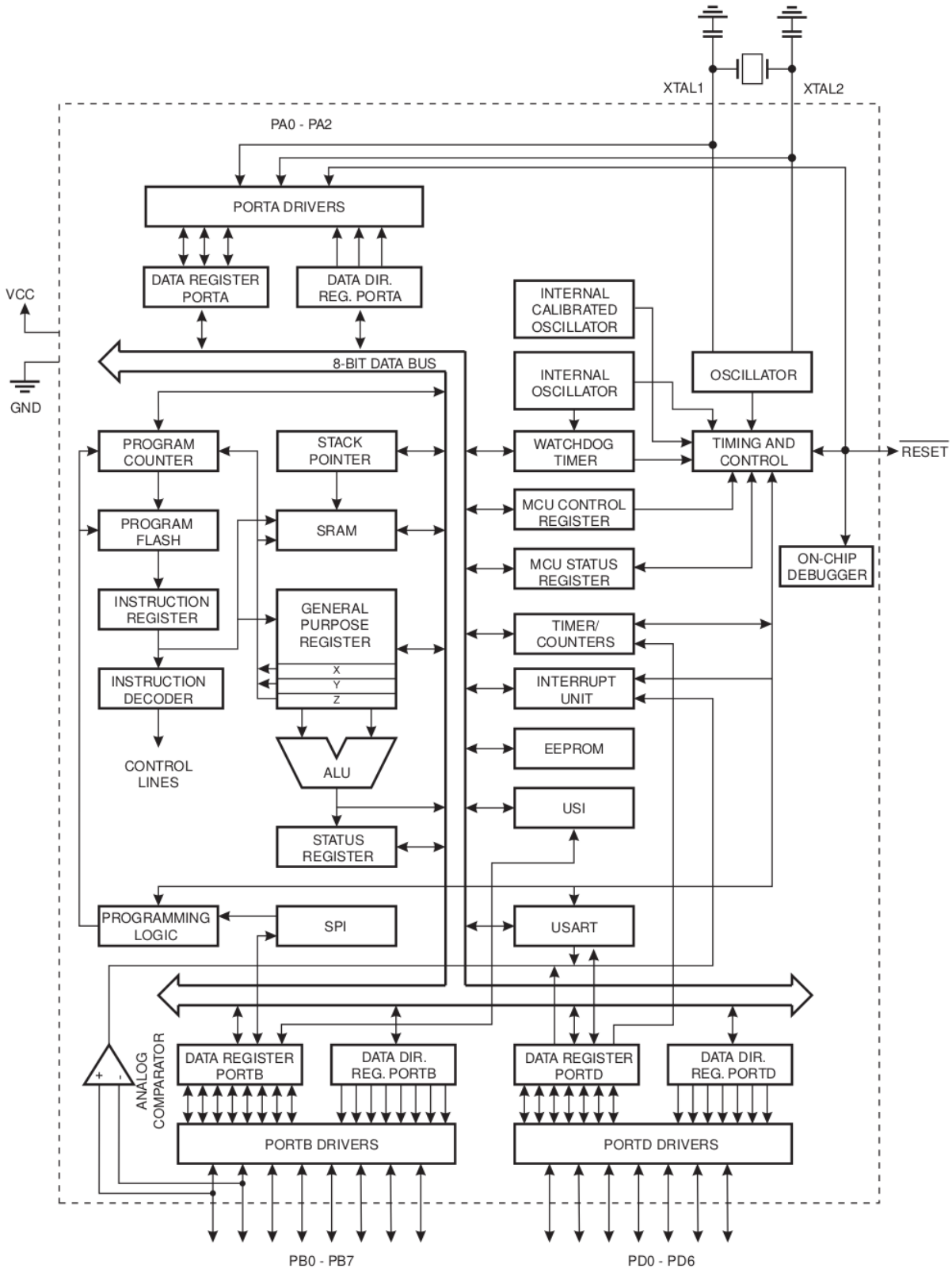


Figura 2: Diagrama de bloques del ATtiny4313. (Fuente: Imagen tomada de [1])

2.1.1. Registros de importancia

Para controlar la operación deseada del microcontrolador, es necesario colocar los valores adecuados en los distintos registros que contiene el dispositivo. Para el caso particular del presente laboratorio, los principales registros de interrupción se detallan a continuación, utilizando como base la hoja del fabricante [1]:

- **DDRB - Port B Data Direction Register**: este registro permite configurar los pines del puerto B como entradas o salidas (si se define un pin en 1 es una salida, de lo contrario es un entrada). En la Figura 3, puede verse la composición del registro:

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 3: Composición del *Port B Data Direction Register*. (Fuente: Imagen tomada de [1])

- **PORTB - Port B Data Register**: corresponde a un puerto I/O bidireccional de 8 bits con resistencias de pull-up internas. Permiten determinar el estado de cada uno de los pines del puerto B. A continuación en la Figura 4, pueden observarse los detalles de configuración de este registro:

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 4: Composición del *Port B Data Register*. (Fuente: Imagen tomada de [1])

- **GIMSK -**: este permite habilitar o deshabilitar algunos tipos de interrupciones. Para el caso particular de este laboratorio, son importantes las interrupciones externas (INT0 e INT1). Los detalles del registro pueden verse en la Figura 5 a continuación:

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	INT1	INT0	PCIE0	PCIE2	PCIE1	–	–	–	GIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Figura 5: Composición del *General Interrupt Mask Register*. (Fuente: Imagen tomada de [1])

- **MCUCR - MCU Control Register**: este registro contiene bits de control que habilitan distintos modos de interpretar interrupciones. Su composición puede observarse en la Figura 6

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 6: Composición del *MCU Control Register*. (Fuente: Imagen tomada de [1])

En el caso de que la interrupción externa INT0 (del registro anterior) esté habilitada, este registro cuenta con distintas configuraciones, las cuales pueden observarse en la Tabla 1:

Tabla 1: Control de detección de interrupciones de INT0. (Tomada de [1]).

ISC01	ISC00	Descripción
0	0	El nivel bajo de INT0 genera una petición de interrupción.
0	1	Cualquier cambio lógico en INT0 genera una petición de interrupción.
1	0	El flanco decreciente de INT0 genera una petición de interrupción.
1	1	El flanco creciente de INT0 genera una petición de interrupción.

Por otro lado, los principales registros de *timing* (para el contador 0 - Timer0) para este laboratorio, pueden consultarse a continuación, utilizando como base la hoja del fabricante [1]:

- **TCNT0 - Timer/Counter**: este registro da acceso directo al contador de 8 bits de la unidad *Timer/Counter* y este acceso puede ser tanto de lectura como de escritura. En la Figura 7 puede verse su composición:

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 7: Composición del *Timer/Counter*. (Fuente: Imagen tomada de [1])

- **TCCR0A - Timer/Counter Control Register A**: este registro permite configurar la secuencia de conteo para el contador 0 y el modo de comparación. Su composición puede observarse en la Figura 8:

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 8: Composición del *Timer/Counter Control Register A*. (Fuente: Imagen tomada de [1])

La secuencia de conteo es determinada por los bits WGM01 y WGM00, además del bit WGM02 de registro CCR0B (el siguiente). Los modos de comparación son determinados por combinaciones de los bits de secuencia de conteo y los birs COM0A1, COM0A0, COM0B1 y COM0B0.

- **TCCR0B - Timer/Counter Control Register B**: este registro permite configurar la fuente de reloj, la cual es controlada por los bits CS02:0. Su composición puede observarse en la Figura 9:

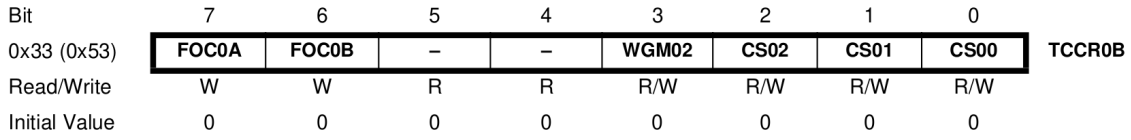


Figura 9: Composición del *Timer/Counter Control Register B*. (Fuente: Imagen tomada de [1])

La distintas configuraciones posibles pueden observarse en la Tabla 2:

Tabla 2: Descripción de los bits de selección de reloj. (Tomada de [1]).

CS02	CS01	CS00	Descripción
0	0	0	Sin fuente de reloj (Timer/Counter detenido).
0	0	1	$clk_{I/O}$ / (Sin *prescaling*).
0	1	0	$clk_{I/O}$ / 8 (Del *prescaler*).
0	1	1	$clk_{I/O}$ / 64 (Del *prescaler*).
1	0	0	$clk_{I/O}$ / 256 (Del *prescaler*).
1	0	1	$clk_{I/O}$ / 1024 (Del *prescaler*).
1	1	0	Fuente de reloj externa en el pin T0. Reloj en flanco decreciente.
1	1	1	Fuente de reloj externa en el pin T0. Reloj en flanco creciente.

- **TIMSK - Timer/Counter Interrupt Mask Register**: este registro permite la selección del tipo de interrupciones causadas por el contador 0 (*Timer/Counter Compare Match* A y B, además de interrupción por *overflow*). Los detalles del registro pueden verse en la Figura 10:

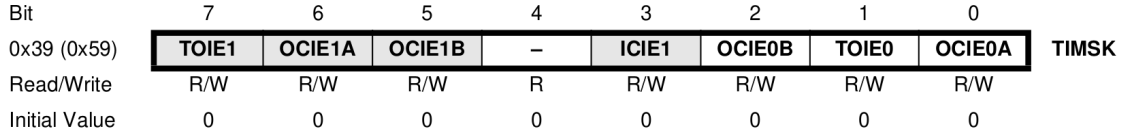


Figura 10: Composición del *Timer/Counter Interrupt Mask Register*. (Fuente: Imagen tomada de [1])

2.2. Resistencia de *pull-down*

Del enunciado del laboratorio, se sabe que el objetivo de este es construir un simulador de un paso peatonal que cuente con dos botones de entrada con los que el usuario pueda solicitar una parada del tránsito vehicular. De lo anterior se puede deducir que al menos uno de los pines del microcontrolador debe ser configurado como entrada y que esta debe permanecer en un estado determinado, para que sólo en caso de presionar el botón (se obtiene el inverso del estado anterior), se active su funcionamiento. Para ese diseño, se quiere que el estado por defecto sea en bajo para proteger el circuito constantemente y que el estado en alto (al presionar el botón) active el circuito.

Una resistencia de pull-down permite lograr el comportamiento deseado para el caso expuesto. En la Figura 11 puede observarse la disposición adecuada construida en SimulIDE según se detalla en [2]:

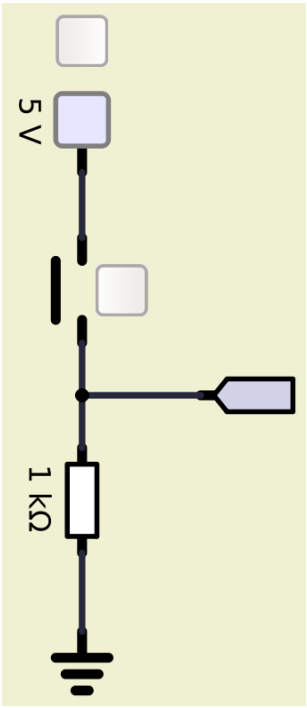


Figura 11: Ejemplo de resistencia de *pull-down* y *switch* en SimulIDE.

En la Figura 11, puede observarse como el pulsador está conectado entre la fuente de tensión y el pin del MCU. Cuando el interruptor es presionado, la entrada del microcontrolador está en un valor lógico alto, pero cuando el interruptor está abierto, la resistencia *pull-down* "jala" la tensión de entrada a la tierra (valor lógico cero). Según se explica en [2], la resistencia *pull-down* debe ser mayor que la impedancia de salida del dispositivo, ya que de lo contrario podría ser capaz de bajar la tensión demasiado y la tensión de entrada en el pin se mantendría en un valor bajo lógico constante. Para calcular esta, se aplica la ley de Ohm de la siguiente forma:

$$Z_{MCU} = \frac{V_{DD}}{I_{max}} \tag{1}$$

Donde V_{DD} corresponde a la tensión máxima entregada por cada pin del MCU y I_{max} la corriente máxima en el mismo caso, de esta forma se tiene que:

$$Z_{MCU} = \frac{5}{20 \times 10^{-3}} = 250\ \Omega \tag{2}$$

Para este caso particular, se tomará su valor como 1 kΩ para asegurarse de que no haya ningún problema.

2.3. Circuito para lidiar con el *switch bounce*

Tal como se detalla en [3], cerrar un *switch*, puede parecer que este hace contacto de manera absoluta e inmediata, sin embargo, este dispositivo no deja de ser mecánico y el contacto ocurre de manera oscilante, por lo que una serie de interferencias son ingresadas a la señal transmitida. Para solucionar este problema, puede utilizarse un circuito RC a la salida del interruptor, de tal forma que se filtren los picos producidos por las oscilaciones y suavicen la señal que será transmitida. Lo anterior es útil para evitar falsas pulsaciones. Un ejemplo del tipo de circuito a construir es el que puede observarse en la Figura 12:

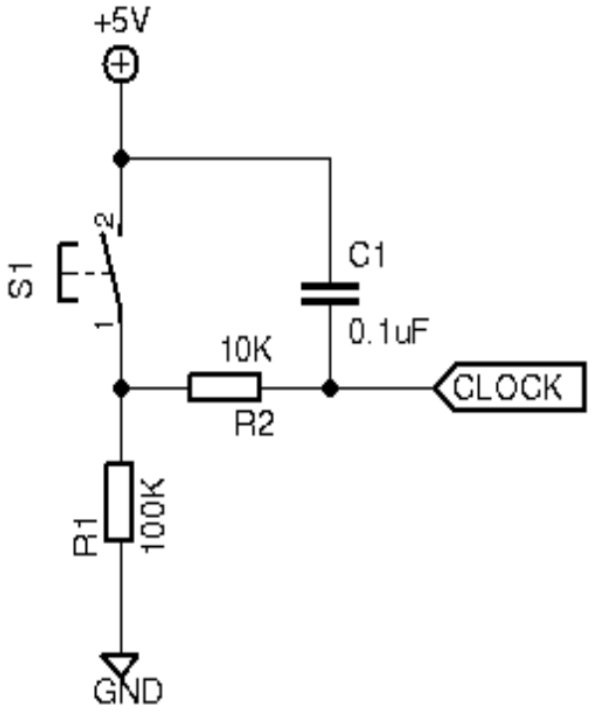


Figura 12: Ejemplo de circuito utilizado para evitar las oscilaciones producidas por un *switch* que combina un circuito RC con un resistor de *pull-down*. (Fuente: Imagen tomada de [3])

Para hacer los cálculos adecuados de capacitancia y resistencia, se inicia definiendo un tiempo de carga y descarga adecuado. En este caso, como no se trata de una acción que necesite de actividad inmediata, se puede trabajar en el orden de los milisegundos, en este caso, de los 100 ms. Como bien se sabe:

$$\tau = R \cdot C \tag{3}$$

Por lo que si ya se tiene un valor de tiempo, basta con seleccionar una de las dos variables restantes para despejar la otra. Si se toma un valor de resistencia de 5 kΩ, entonces despejando:

$$C = \frac{\tau}{R} = \frac{10 \times 10^{-3}}{5 \times 10^{3}} = 2\ \mu F \tag{4}$$

Para apegarse a los valores disponibles en el mercado, se elige un capacitor de 2.2 μF.

El circuito resultante construido en SimulIDE puede consultarse en la Figura 13 a continuación:

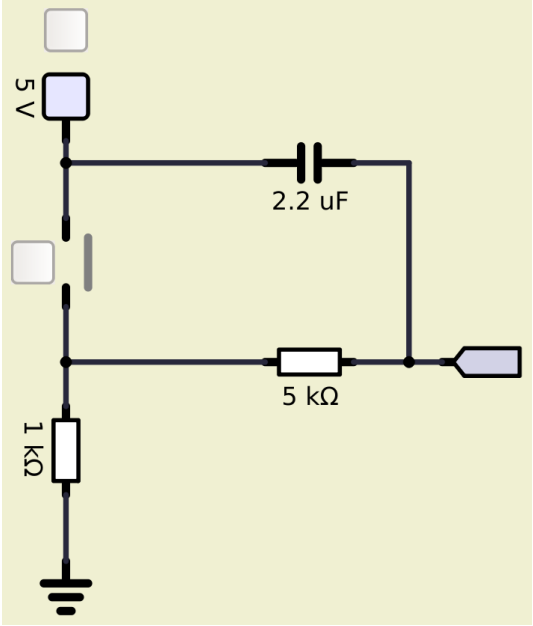


Figura 13: Circuito utilizado para evitar las oscilaciones producidas por un *switch* que combina un circuito RC con un resistor de *pull-down*.

Del enunciado se sabe que para simular un paso peatonal, serán necesarios dos botones, uno para cada semáforo peatonal. A pesar de ser botones distintos, la funcionalidad que activan (interrupción externa en INT0 que modifica la FSM) en la misma, por lo que deben colocarse en paralelo. El circuito resultante construido en SimulIDE puede consultarse en la Figura 14 a continuación:

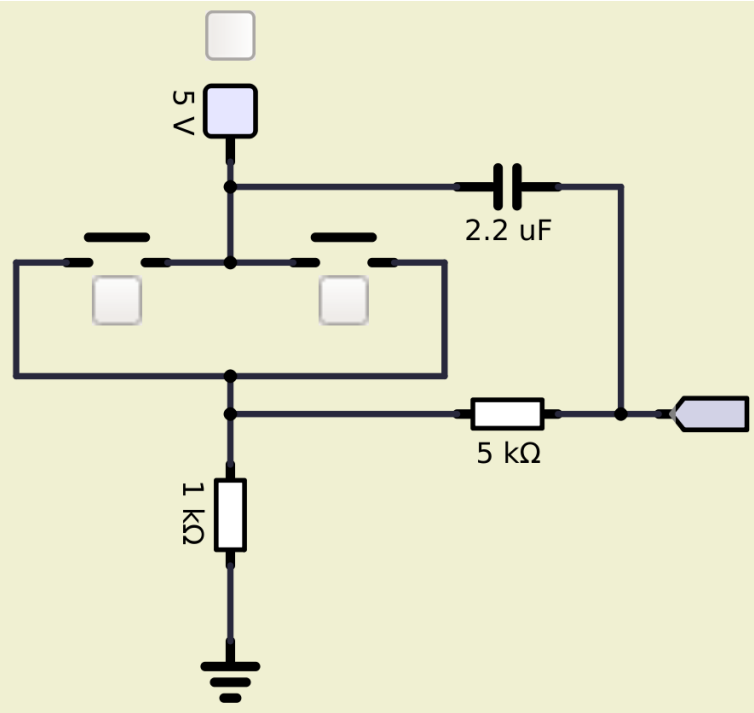


Figura 14: Circuito utilizado para evitar las oscilaciones producidas por dos *switches* que combina un circuito RC con un resistor de *pull-down*.

2.4. Diseño del circuito simulador de un paso peatonal

Según se indica en el enunciado, el presente laboratorio tiene como objetivo simular el funcionamiento de un paso peatonal de una calle unidireccional que cuenta con dos semáforos peatonales y un semáforo vehicular. La Figura 15 muestra la disposición de los elementos mencionados:

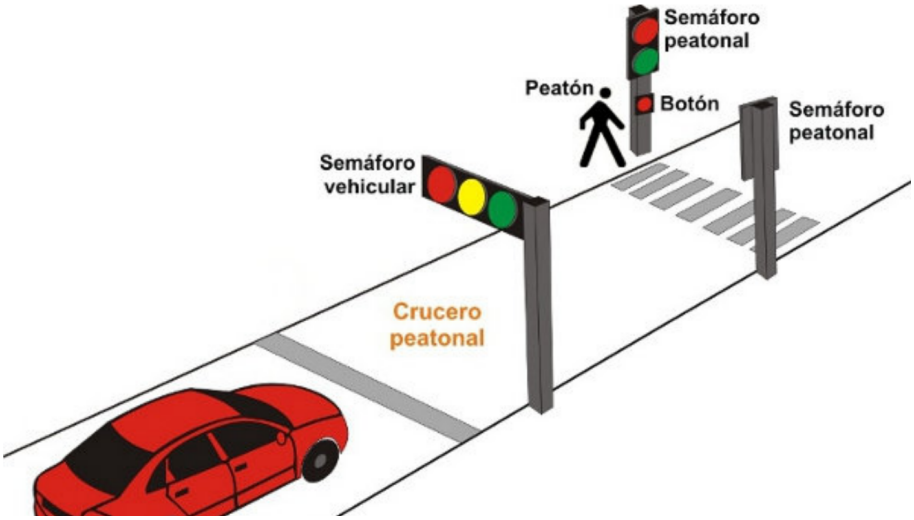


Figura 15: Cruce peatonal.

De la Figura 15, se puede deducir que habrán 7 LEDs conectados entre alguno de los pines del ATtiny4313 y la tierra, por lo que estos necesitan de resistencias de protección. Para calcular el valor de estas, se debe tomar en cuenta que la tensión máxima de operación es de 5 V y la corriente máxima suministrada es de 20 mA, información tomada de la hoja del fabricante [1]. Para obtener el valor de la resistencia de protección utilizando la ley de Kirchhoff de esta forma:

$$R = \frac{V_{DD} - V_{LED}}{I_{max}} \tag{5}$$

Como se trata de una simulación, se tomará V_{LED} = 2.4 V (tensión de umbral del simulador) para todos los casos. Para no presionar el sistema, se realizarán los cálculos utilizando 15 mA. De esta forma:

$$R = \frac{5 - 2{,}4}{15 \times 10^{-3}} = 173{,}33\ \Omega \tag{6}$$

Es este caso, se selecciona un valor estándar de 180 Ω.

2.5. Lista de componentes y precios

En la Tabla 3 pueden consultarse los componentes utilizados y sus precios, tomando como referencia los precios del sitio web de la tienda de componentes MicroJPM (https://www.microjpm.com/):

Tabla 3: Lista de componentes.

Componente	Cantidad	Precio
ATtiny4313	1	$1,80
Pulsador	2	$0,40
Capacitor de 2.2 μF	1	$0,35
Resistor de 5 kΩ	1	$0,07
Resistor de 1 kΩ	1	$0,07
Resistores de 180 Ω	7	$0,35
LED verdes	3	$0.51
LED rojos	3	$0.51
LED amarillo	1	$0,17
Total	-	$4.23

Según la Tabla 3, para realizar este proyecto son necesario ₡2,121.73, según el valor del dólar al momento de escribir el presente informe.

2.6. Máquinas de estado finitas (FSM)

Una Maquina de estado Finito es esencialmente un programa que representa una secuencia de instrucciones a ser ejecutadas, donde cada instrucción depende del estado actual de la máquina y

del actual estimulo. Las posibles entradas al sistema son una secuencia de símbolos seleccionadas desde un conjunto finito I de símbolos de entrada, y las salidas resultantes son secuencias de símbolos escogidas desde un conjunto finito Z de símbolos de salida [4]. Estas maquinas pueden ser síncronas o asíncronas, cuando necesitan o no la intervención de un pulso de reloj. Existen dos tipos principales de maquinas de estado:

- **Maquina de estado de Mealy**: Las salidas de la máquina de estados no solo dependen de los estados, sino también de las entradas al sistema, que se representan definiendo salidas de la máquina en las transiciones [5].
- **Maquina de estado de Moore**: Las salidas de la máquina de estados solo dependen del estado del sistema, que se representa definiendo salidas de la máquina en los estados, según se detalla en [5].

En la Figura 16 puede consultarse un ejemplo de máquina de estado:

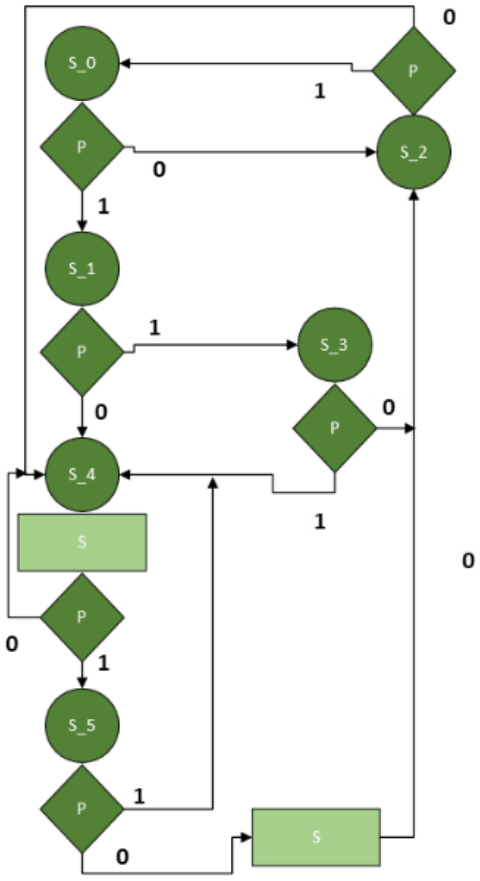


Figura 16: Diagrama FSM ejemplo (Elaboración propia).

3. DESARROLLO Y ANÁLISIS DE RESULTADOS

A continuación se muestran los métodos de desarrollo utilizados para resolver cada uno de los problemas planteados, así como la funciones utilizadas, el método de operación a partir de diagramas de flujo y los resultados obtenidos:

3.1. Desarrollo de la solución

A continuación podrá consultarse una descripción de alto nivel de cada una de las funciones implementadas para conseguir los objetivos del presente laboratorio:

3.1.1. Función `init()`

Esta función se encarga de configurar e inicializar los registros necesarios para permitir el funcionamiento esperado del microcontrolador. Al inicio de la función, se le da valor a alguna variables globales y se habilitan las interrupciones globales haciendo el llamado de la función `sei()`. Además, se configuran los pines del puerto B que se utilizarán, tanto su estado como su modo. Lo siguiente es la configuración de la interrupción externa INT0, la cual permite controla el cruce a través de un pulsador. Finalmente se configura el temporizador 0 utilizando el modo de *Clear Timer on Compare Match*.

3.1.2. Rutina de interrupción externa

Esta función se activa cuando se produce una interrupción externa en el microcontrolador, en este caso específico, cuando ocurre una interrupción en el pin INT0. El objetivo de esta función es almacenar el una variable global llamada `btn_pushed` un 1, cuando el botón conectado al pin PD2 (pin 6) ha sido presionado. Lo anterior permite la activación de la máquina de estados más adelante.

3.1.3. Rutina de interrupción de temporizador

Esta función se activa cada vez que ocurre un CTC y permite controlar el tiempo del programa, ya que tras *n* (en este caso es 60) comparaciones exitosos, suma una unidad a la variable global de segundos. Lo anterior permite que la máquina de estados realice las transiciones en los momentos adecuados. Asimismo, hace posible los parpadeos de los estados en los que esto es necesario.

3.1.4. Función `FSM()`

Esta función contiene la máquina de estados que permite las transiciones de estado en función de distintas condiciones que dependen del estado actual. En la Figura 17 se muestra el diagrama de estados:

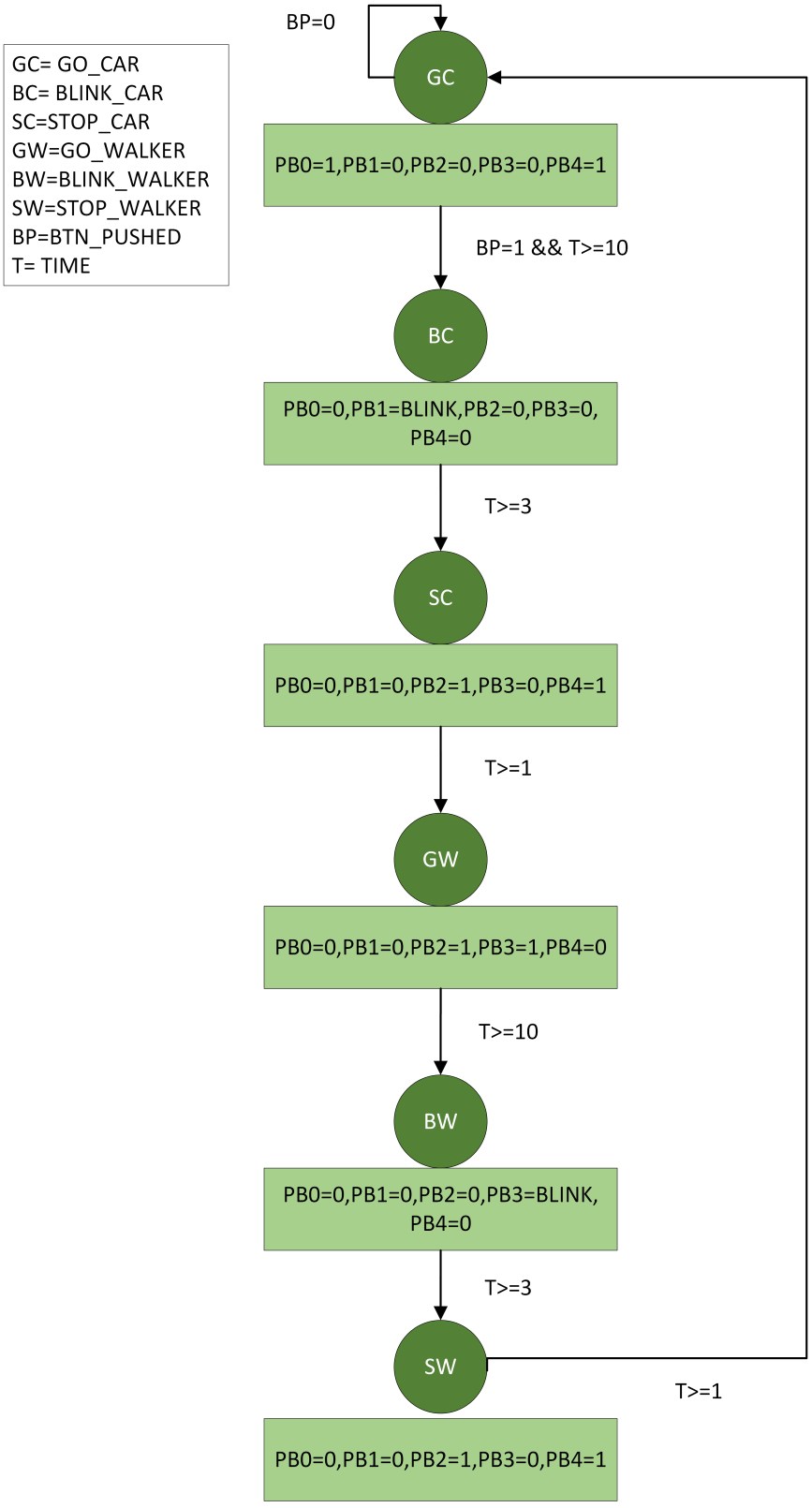


Figura 17: Diagrama FSM.

3.1.5. Función `main()`

La función principal solamente hace el llamado de la función `init()` para configurar adecuadamente el ATtiny4313 y `FSM()` para iniciar con las transiciones de estado según sea el caso.

3.1.6. Construcción del circuito utilizado

Para iniciar con la construcción de este circuito, es necesario tomar en cuenta los métodos de protección descritos en las secciones 2.2 y 2.3. Una vez que ambos fueron unidos y construidos, se conectó la salida de este circuito al PD2 (pin 6 de la Figura 1), ya que este se trata del pin encargado de detectar las interrupciones externas a través de INT0. El siguiente paso es la construcción de los semáforos. Los tres LED del semáforo vehicular necesitan de pines independientes, ya que su operación es distinta a la de los semáforos peatonales. En el caso de estos últimos, como su funcionamiento está sincronizado, sólo es necesario utilizar dos pines (uno para los LED verdes y otro para los rojos). Debido a la cantidad de pines necesarios, se selecciona el puerto B, es el que cuenta con la mayor cantidad de pines de I/O. El método de cálculo de las resistencias de cada LED puede observarse en la sección 2.4. El circuito resultante tras implementar todas as consideraciones anteriores y utilizando *tunnels* para hacer la disposición del circuito más clara, puede consultarse en la Figura 18:

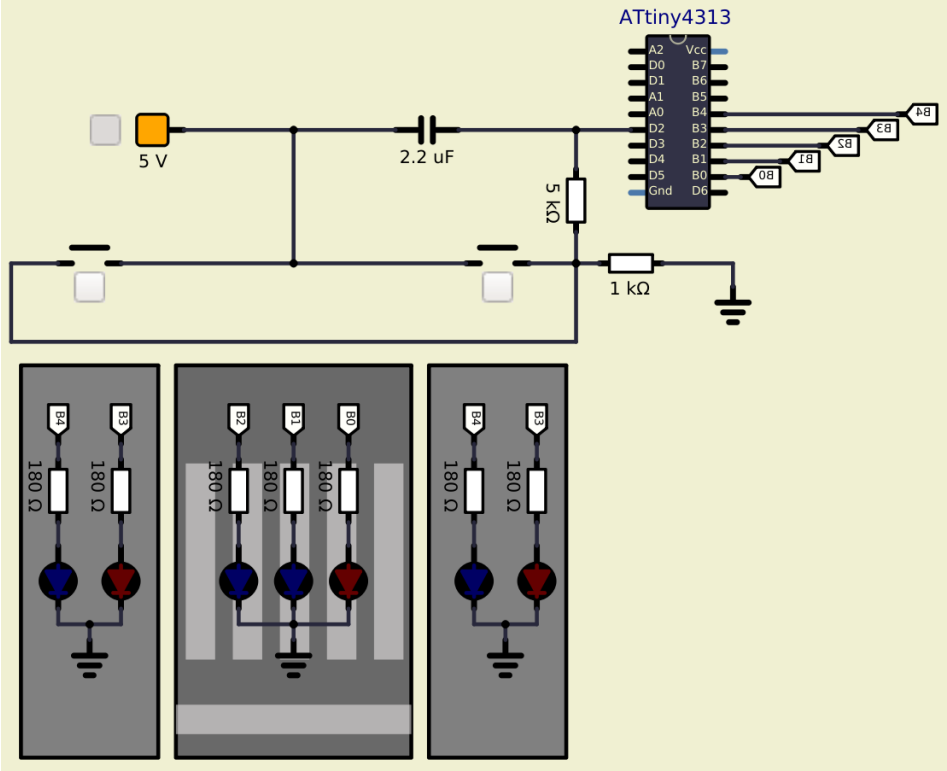


Figura 18: Circuito final tras los cálculos de magnitudes y la implementación de las consideraciones necesarias.

3.2. Análisis de los resultados

3.2.1. Funcionamiento del simulador del cruce peatonal según su estado

A continuación se hará una demostración del funcionamiento de la simulación, mediante la demostración de cada uno de los 6 estados posibles de la FSM:

- Estado ***GO_CAR*** [0]: este corresponde al estado inicial y también es el estado por defecto. En este, la luz verde del semáforo vehicular y las rojas de los semáforos peatonales están encendidas. Los detalles pueden verse en la Figura 19:

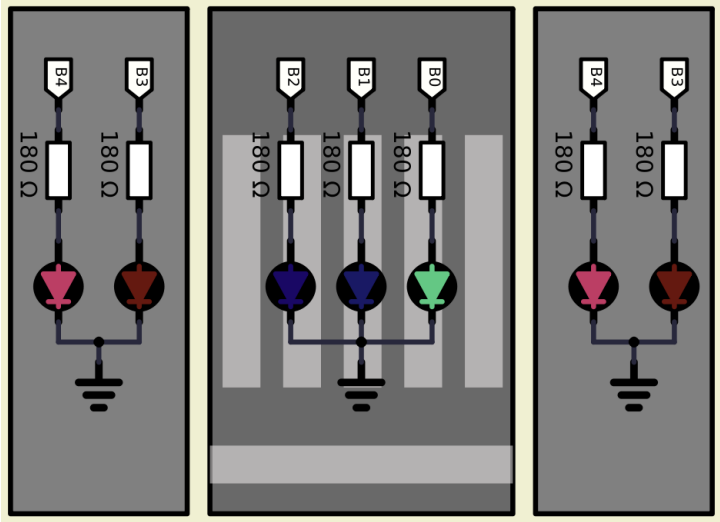


Figura 19: Estado GO_CAR de la FSM.

- Estado ***BLINK_CAR*** [1]: en este estado la luz amarilla parpadea del semáforo vehicular parpadea, mientras que las luces rojas del semáforo peatonal están encendidas. Con las Figura 20 y 21 se muestra el parpadeo:

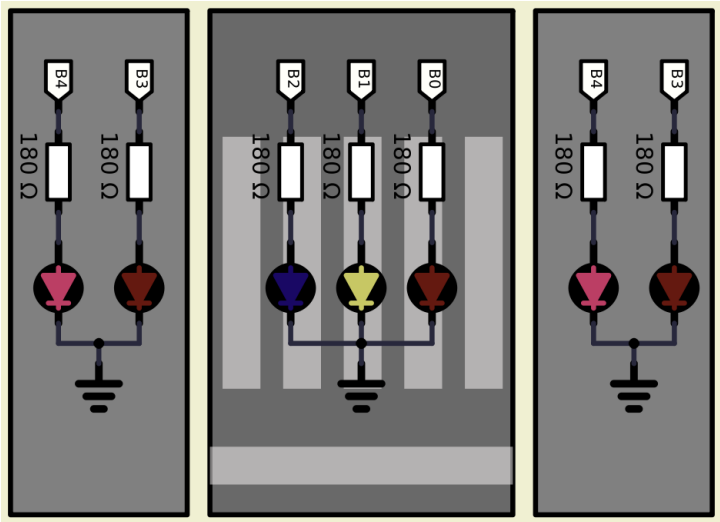


Figura 20: Estado BLINK_CAR de la FSM, luz amarilla encendida.

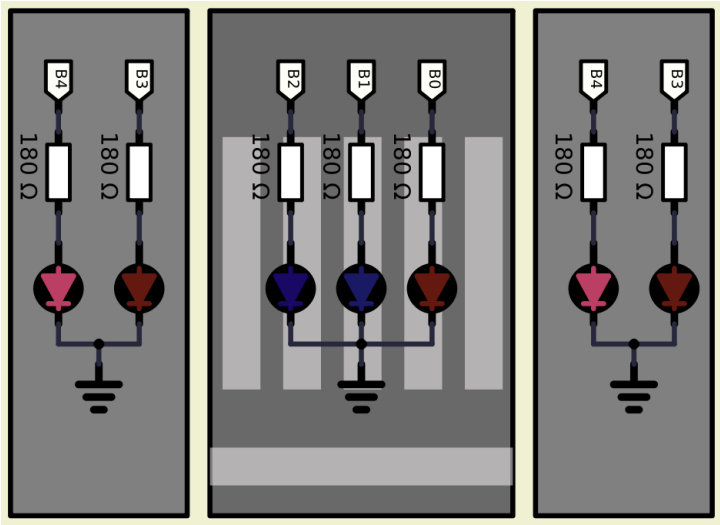


Figura 21: Estado BLINK_CAR de la FSM, luz amarilla apagada.

- Estado ***STOP_CAR*** [2]: este corresponde un estado de transición en el que la luz roja del semáforo vehicular y las luces rojas del semáforo peatonal están encendidas al mismo tiempo. Los detalles pueden verse en la Figura 22:

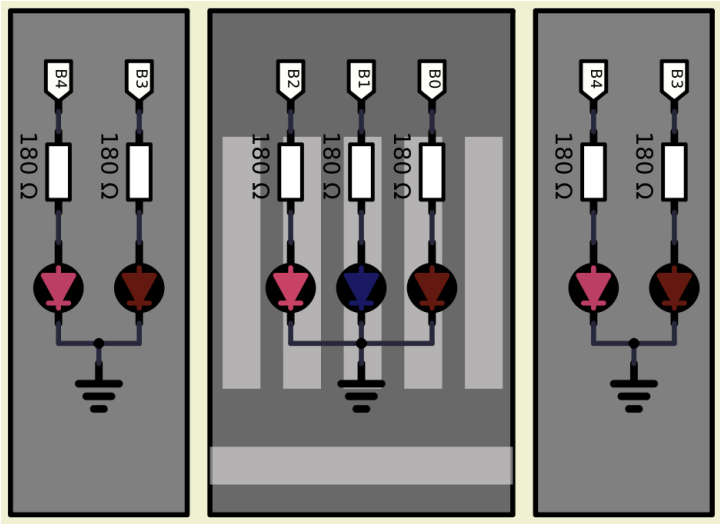


Figura 22: Estado STOP_CAR de la FSM.

- Estado ***GO_WALKER*** [3]: en este estado, las luces verdes del semáforo peatonal y la luz roja del semáforo vehicular están encendidas. Los detalles pueden verse en la Figura 23:

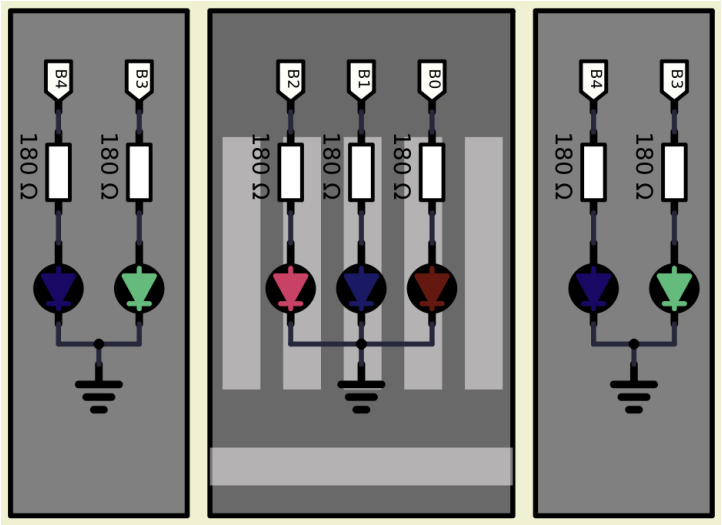


Figura 23: Estado GO_WALKER de la FSM.

- Estado ***BLINK_WALKER*** [4]: en este estado las luces verdes del semáforo peatonal parpadean mientras la luz roja del semáforo vehicular está encendida. Con las Figura 24 y 25 se muestra el parpadeo:

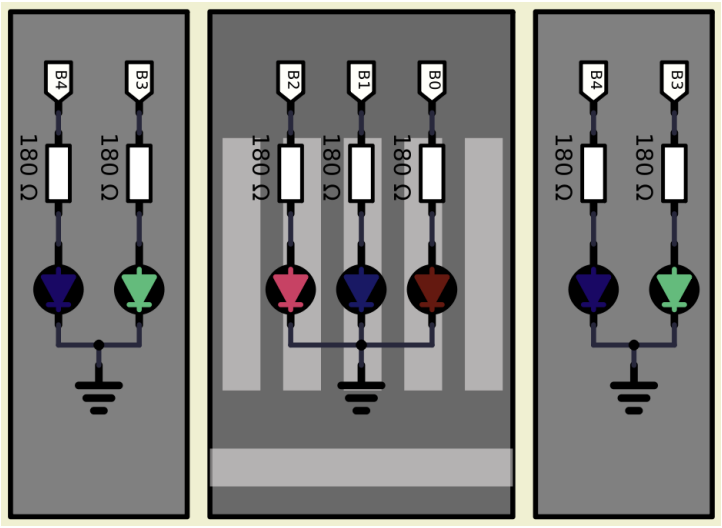


Figura 24: Estado BLINK_WALKER de la FSM, luces verdes encendidas.

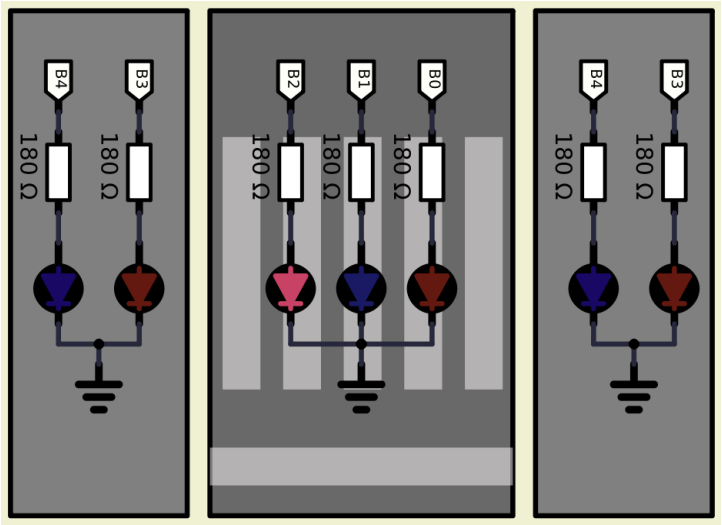


Figura 25: Estado BLINK_WALKER de la FSM, luces verdes apagadas.

- Estado ***STOP_WALKER*** [5]: este corresponde un estado de transición en el que la luz roja del semáforo vehicular y las luces rojas del semáforo peatonal están encendidas al mismo tiempo. Los detalles pueden verse en la Figura 26:

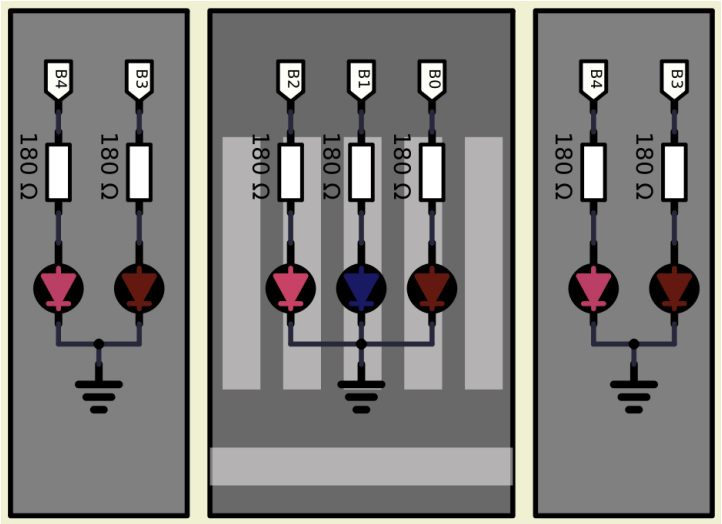


Figura 26: Estado STOP_WALKER de la FSM.

Como puede observarse en las 6 figuras anteriores, los LED se encienden de manera adecuada según sea el estado en el que se encuentre el proceso. Para verificar el fila aleatoriedad de la implementación, se insta a la persona lectora del presente a probar la simulación por su cuenta.

Una vez que el comportamiento a nivel de LED fue comprobado, puede analizarse también el comportamiento de la se señales obtenidas utilizando el osciloscopio. La configuración usada en el osciloscopio fué Time Div 2.5seg, Time pos 0.00ps y Volt div 500mV. Las señales mostradas son:

- LDPV: Luz de Paso Vehículos.

- LDPV-A: Luz de Paso Vehículos - Amarilla.
- LDVD: Luz de Vehículos Detenidos.
- LDPP: Luz de Paso de Peatones.
- LDPD: Luz de Peatones Detenidos.

En la Figura 27 se muestra el inicio de la simulación, el estado iniciales (GO_CAR), por lo que se puede esperar una LDPV y LDPD en alto, una LDPV-A,LDVD y LDPP en bajo esto se mantiene durante 10 segundos hasta que se acciona el botón. Inmediatamente al accionar se establece el siguiente estado (BLINK_CAR), por lo que se puede observar como LDPV se coloca en bajo y la luz Amarilla comienza a parpadear durante 3 segundos, dando aviso a los conductores que la LDVD pronto se encenderá.

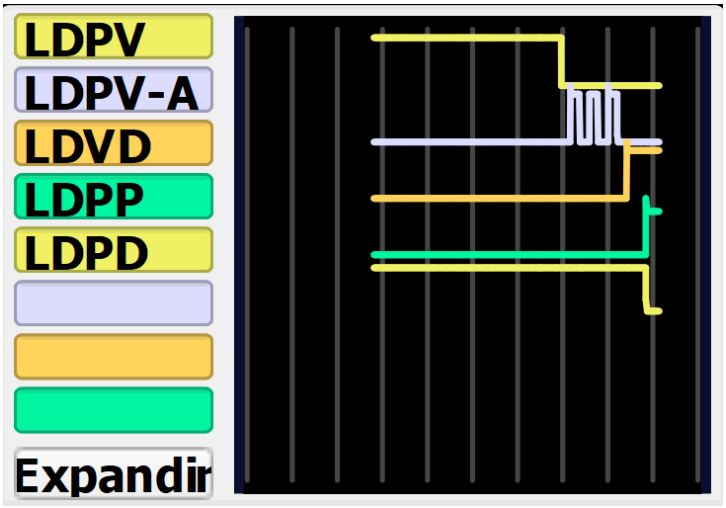


Figura 27: Análisis de ondas en Osciloscopio(Primeros 10 segundos).

Al terminar de parpadear LDPV-A, se pasa al siguiente estado (STOP_CAR) coloca en alto LDVD y 1 segundo después se procede a encender la LDPP y apagar la LDPD (GO_WALKER). En la Figura 28 puede verse lo anterior:

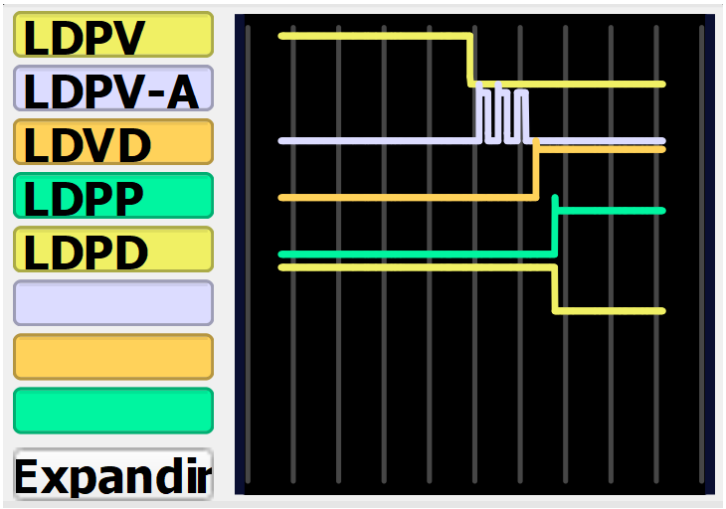


Figura 28: Análisis de ondas en Osciloscopio(Justo luego de 10 segundos).

Esta condición de pase de peatones se mantiene durante 13 segundos, sin embargo, al segundo 10 entra el estado (BLINK_WALKER) y se procede a parpadear durante 3 segundos el led de LDPP indicando a los peatones que pronto se encenderá LDPD. Justo luego de parpadear entra el estado (STOP_WALKER) se enciende LDPD y 1 segundo después entra el estado (GO_CAR) se Apaga LDVD y de enciende LDPV de forma simultanea. En la Figura 29 puede verse lo anterior:

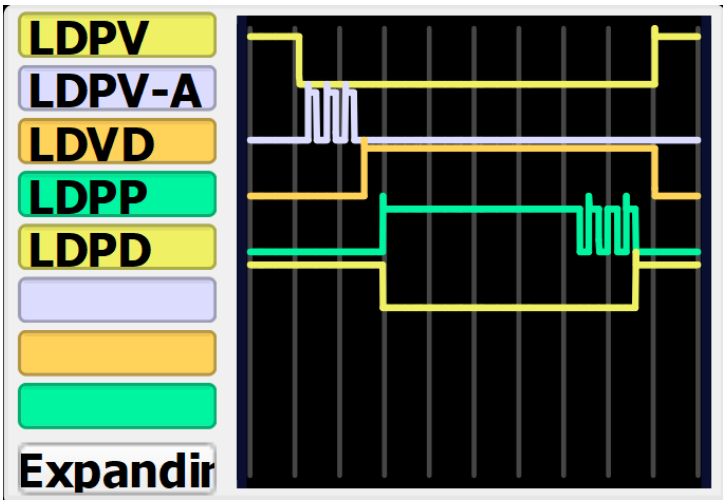


Figura 29: Análisis de ondas en Osciloscopio(Completo).

Finalmente, luego de este análisis de funcionamiento utilizando el osciloscopio, se puede decir que el funcionamiento cumple con el diagrama de temporización propuesto en el enunciado.

4. CONCLUSIONES Y RECOMENDACIONES

A lo largo del desarrollo del informe se pusieron a prueba una serie de conceptos y prácticas importantes para el manejo básico del microcontrolador ATtiny4313, así como sus GPIO, *timers* y el uso de máquinas de estado finito. A continuación algunas conclusiones y recomendaciones recolectadas a lo largo de trabajo realizado:

- La simulación del cruce peatonal refleja de manera coherente los estados de la Máquina de Estados Finitos (FSM), garantizando una representación coherente con lo esperado de la secuencia de luces tanto vehiculares como peatonales.
- La verificación del comportamiento de las señales mediante el osciloscopio confirma la correcta secuencia temporal y activación de las diferentes luces del cruce, validando así la precisión y fiabilidad del simulador, respecto al diagrama de tiempos dado en el enunciado.
- Al trabajar con interrupciones y temporizadores, es de suma importancia comprender con profundidad la forma en la que estos operan en general y para el caso específico del microcontrolador con el que se esté trabajando, ya que el método de programación que debe utilizarse no es tan intuitivo como al momento de hacer uso de retrasos.
- Es muy importante no subestimar la información contenida en las hojas del fabricante. En estas está contenida toda la información necesaria para realizar un diseño seguro, eficiente y estable.
- El presente diseño podría ser extendido y mejorado, agregando la evaluación de casos como la acción que se debe tomar si ambos botones del paso peatonal se presionan a la vez o el caso en el que el cruce sea bidireccional.

REFERENCIAS BIBLIOGRÁFICAS

[1] Atmel, *8-bit Microcontroller with 2/4K Bytes In-System Programmable Flash*. 8246B–AVR–09/11, 2011. Disponible en: https://ww1.microchip.com/downloads/en/DeviceDoc/doc8246.pdf.

[2] EEPower, "Pull-up and Pull-down Resistors." https://eepower.com/resistor-guide/resistor-applications/pull-up-resistor-pull-down-resistor/#, s.f. Accessed: April 01, 2024.

[3] All About Circuits, "Switch Bounce and How to Deal with It." https://www.allaboutcircuits.com/technical-articles/%20switch-bounce-how-to-deal-with-it/, 2015. Accessed: April 01, 2024.

[4] M. Alavi, S. Aliaga, and M. Murga, "Máquinas de estado finito," *Revista de Investigación Estudiantil Iluminate*, vol. 8, p. 41, 2016.

[5] MathWorks, "Introducción a las state machines." https://la.mathworks.com/discovery/state-machine.html. Accessed: April 11, 2024.

5. Anexos

5.1. Hoja del fabricante del ATtiny2313 Atmel: información general

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
- Data and Non-volatile Program and Data Memories
  - 2/4K Bytes of In-System Self Programmable Flash
    - Endurance 10,000 Write/Erase Cycles
  - 128/256 Bytes In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 128/256 Bytes Internal SRAM
  - Programming Lock for Flash Program and EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Modes
  - Four PWM Channels
  - On-chip Analog Comparator
  - Programmable Watchdog Timer with On-chip Oscillator
  - USI – Universal Serial Interface
  - Full Duplex USART
- Special Microcontroller Features
  - debugWIRE On-chip Debugging
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Low-power Idle, Power-down, and Standby Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit
  - Internal Calibrated Oscillator
- I/O and Packages
  - 18 Programmable I/O Lines
  - 20-pin PDIP, 20-pin SOIC, 20-pad MLF/VQFN
- Operating Voltage
  - 1.8 – 5.5V
- Speed Grades
  - 0 – 4 MHz @ 1.8 – 5.5V
  - 0 – 10 MHz @ 2.7 – 5.5V
  - 0 – 20 MHz @ 4.5 – 5.5V
- Industrial Temperature Range: -40°C to +85°C
- Low Power Consumption
  - Active Mode
    - 190 µA at 1.8V and 1MHz
  - Idle Mode
    - 24 µA at 1.8V and 1MHz
  - Power-down Mode
    - 0.1 µA at 1.8V and +25°C

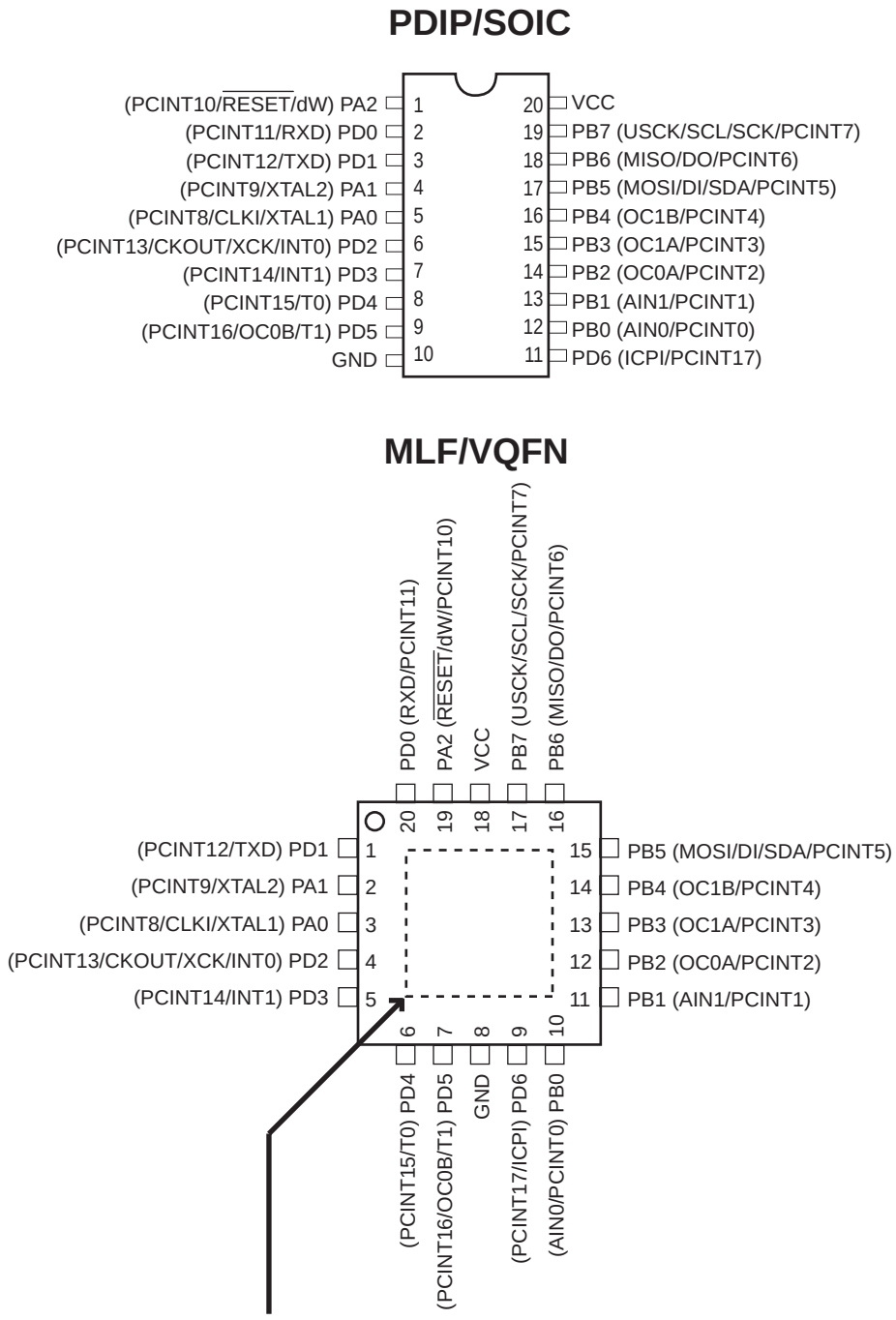
8-bit AVR® Microcontroller with 2/4K Bytes In-System Programmable Flash

ATtiny2313A
ATtiny4313

Rev. 8246B–AVR–09/11

1. Pin Configurations

Figure 1-1. Pinout ATtiny2313A/4313



NOTE: Bottom pad should be soldered to ground.

1.1 Pin Descriptions

1.1.1 VCC

Digital supply voltage.

1.1.2 GND

Ground.

1.1.3 Port A (PA2..PA0)

Port A is a 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability, except PA2 which has the $\overline{\text{RESET}}$ capability. To use pin PA2 as I/O pin, instead of RESET pin, program (“0”) RSTDISBL fuse. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATtiny2313A/4313 as listed on page 62.

1.1.4 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATtiny2313A/4313 as listed on page 63.

1.1.5 Port D (PD6..PD0)

Port D is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATtiny2313A/4313 as listed on page 67.

1.1.6 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running and provided that the reset pin has not been disabled. The minimum pulse length is given in Table 22-3 on page 201. Shorter pulses are not guaranteed to generate a reset. The Reset Input is an alternate function for PA2 and dW.

The reset pin can also be used as a (weak) I/O pin.

1.1.7 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit. XTAL1 is an alternate function for PA0.

1.1.8 XTAL2

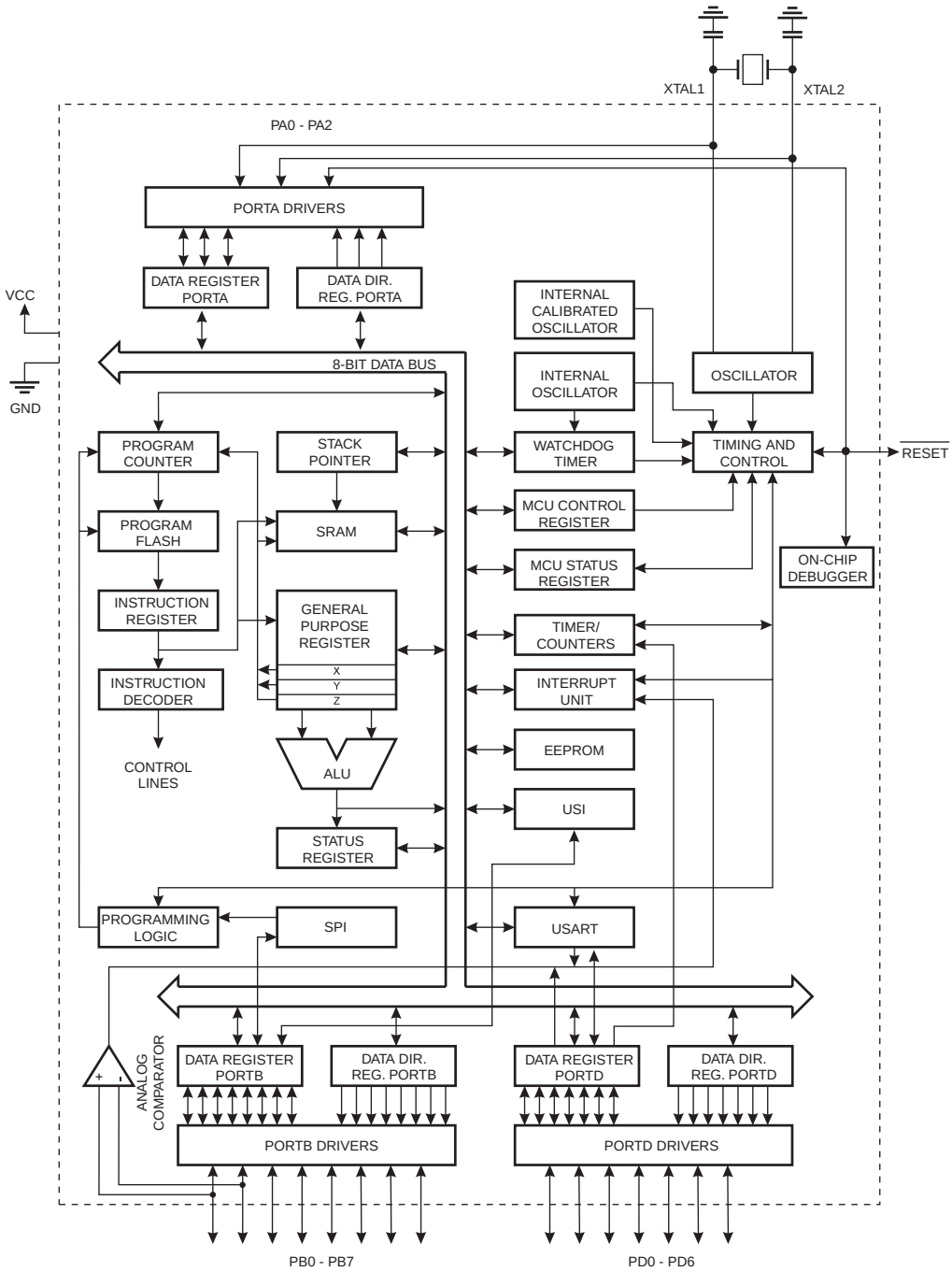
Output from the inverting Oscillator amplifier. XTAL2 is an alternate function for PA1.

2. Overview

The ATtiny2313A/4313 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny2313A/4313 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATtiny2313A/4313 provides the following features: 2/4K bytes of In-System Programmable Flash, 128/256 bytes EEPROM, 128/256 bytes SRAM, 18 general purpose I/O lines, 32 general purpose working registers, a single-wire Interface for On-chip Debugging, two flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, Universal Serial Interface with Start Condition Detector, a programmable Watchdog Timer with internal Oscillator, and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, or by a conventional non-volatile memory programmer. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATtiny2313A/4313 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATtiny2313A/4313 AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

2.2 Comparison Between ATtiny2313A and ATtiny4313

The ATtiny2313A and ATtiny4313 differ only in memory sizes. Table 2-1 summarizes the different memory sizes for the two devices.

Table 2-1. Memory Size Summary

Device	Flash	EEPROM	RAM
ATtiny2313A	2K Bytes	128 Bytes	128 Bytes
ATtiny4313	4K Bytes	256 Bytes	256 Bytes

3. About

3.1 Resources

A comprehensive set of drivers, application notes, data sheets and descriptions on development tools are available for download at http://www.atmel.com/avr.

3.2 Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

For I/O Registers located in the extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically, this means “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”. Note that not all AVR devices include an extended I/O map.

3.3 Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.