Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0624 – Laboratorio de Microcontroladores
II ciclo 2024

Laboratorio # 2

GPIOs, Timers y FSM

Leonardo Serrano Arias C17484
Lorena Solís Extteny B97657

Profesor: Marco Villalta

18 de setiembre, 2024

Índice

Índice de figuras

Introducción

En este laboratorio, se desarrolló un juego de memoria llamado Simón Dice, utilizando el microcontrolador ATtiny4313, LEDs, botones y otros componentes electrónicos. El objetivo principal fue implementar una máquina de estados finitos (FSM) que controlara la lógica del juego, utilizando temporizadores e interrupciones para gestionar las señales de los LEDs y la lectura de los botones. El jugador debe memorizar y reproducir una secuencia de luces que incrementa en dificultad a medida que avanza. La duración de cada luz disminuye conforme la secuencia se alarga, y el juego termina cuando el jugador comete un error al reproducir la secuencia. La FSM facilitó su programación, ya que permitió gestionar las salidas en diferentes momentos del tiempo de manera más estructurada y la sincronización mediante interrupciones temporales garantizó la fluidez en la ejecución del juego y la lectura de los botones.

El repositorio utilizado para el desarrollo de este laboratorio se encuentra en la siguiente dirección: `https://github.com/Leonardo-SA/IE-0624_II_2024_C17484_B97657.git`

Nota teórica

ATTiny 4313

El **ATtiny 313** [1] es un microcontrolador de la familia AVR, basado en una arquitectura RISC de 8 bits. Es conocido por su eficiencia energética, bajo costo y capacidad para ser programado y controlado fácilmente en proyectos embebidos. Este cuenta con 18 pines I/O programables, estos pines se dividen en tres puertos los **A** de 3-bits, los **B** de 8-bits y los **D** de 7-bits. Estos puertos tienen acceso a distintos registros del microcontrolador, de manera que es posible realizar múltiples funciones. Particularmente, se utilizaron, por ejemplo, para el control de salidas para gestionar los pines que controlan los LEDs, encendiéndolos y apagándolos según las secuencias de juego y conteo utilizando el reloj interno, donde se utilizó el temporizador interno del dispositivo para generar interrupciones basadas en tiempos, como la duración de encendido de los LEDs. A continuación, se muestra el diagrama de pines y bloques del microcontrolador que se extrajo de la datasheet.

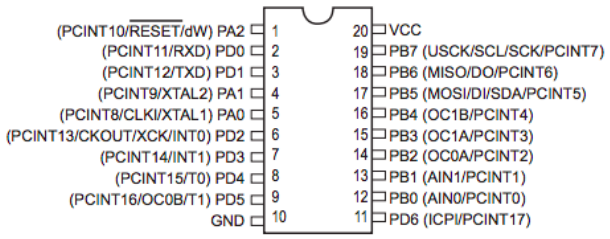


Figura 1: Diagrama con la especificación de los pines del microcontrolador ATtiny4313 [1]

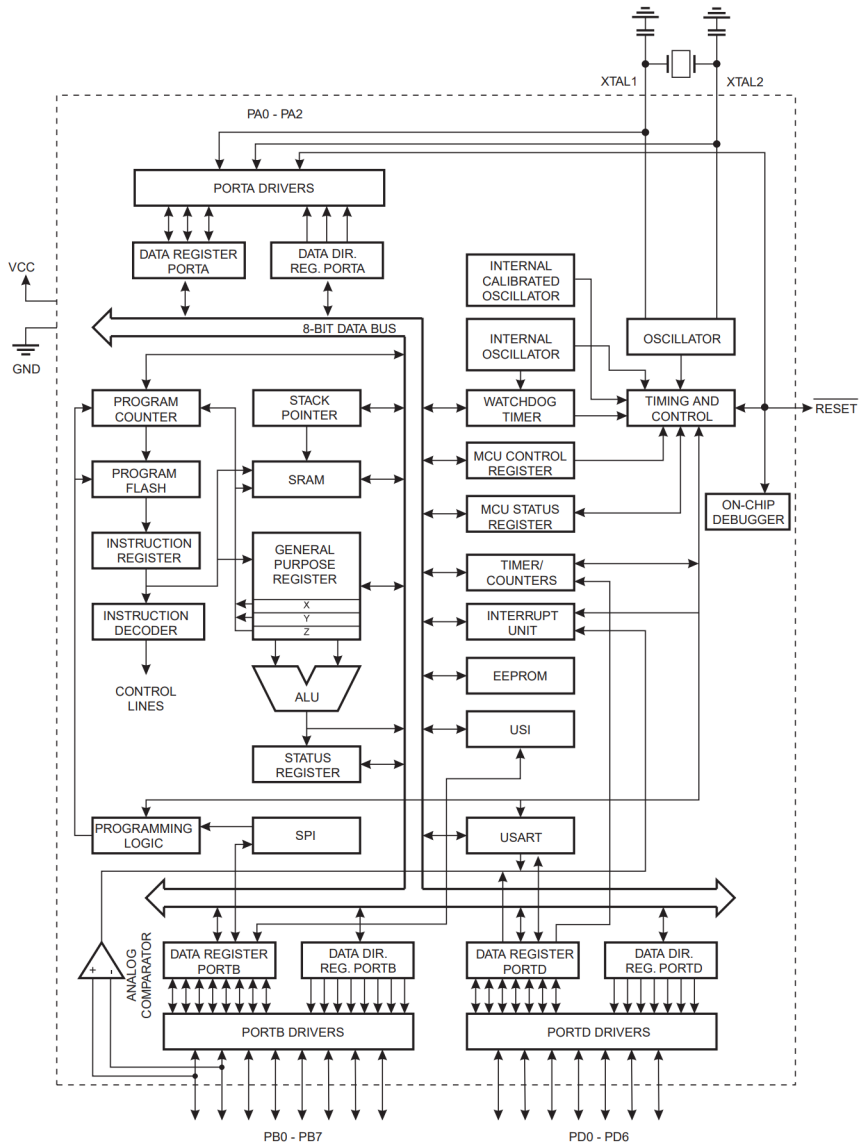


Figura 2: Diagrama de bloques del microcontrolador ATtiny4313 [1]

Seguidamente, se muestran especificaciones importantes que serán necesarias para la implementación del juego en el microcontrolador:

22.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

Figura 3: Características eléctricas del ATtiny4313[1]

Registros

El ATtiny4313 permite gestionar interrupciones tanto en flancos ascendentes como descendentes, consisten en señales internas o externas que indican al sistema que debe detenerse o realizar una tarea diferente a la que estaba siendo realizada [2]. En este laboratorio, se configuraron interrupciones para los cuatro botones, lo que permite cambiar el estado del juego de forma inmediata en respuesta a las acciones del jugador, sin que se interrumpa el flujo normal del programa. Se utilizaron específicamente las interrupciones **INT0**, **INT1**, **PCMSK1**, y **PCMSK2**, ya que estos registros habilitan las interrupciones en los pines correspondientes. El registro **GIMSK** se encarga de activar las interrupciones en los pines de entrada, mientras que **MCUCR** configura el flanco (ascendente o descendente) que desencadena la interrupción.

Para la temporización de las interrupciones, se empleó el temporizador interno controlado por el registro **TIMSK**. Con la ayuda del prescaler, se creó un contador que mide el tiempo en segundos. Los registros **TCCR0A**, **TCCR0B** y **TCNT0** también fueron clave: TCCR0A establece el modo de operación del temporizador, TCCR0B divide la frecuencia del temporizador hasta 1024 veces, y TCNT0 inicia el conteo del temporizador.

En cuanto al manejo de los leds se utlizaron los registros **DDRB** y **PORTB**. DDRB configura los pines del puerto B como salidas para los LEDs, mientras que PORTB controla el encendido y apagado de los LEDs conectados al puerto B.

Configuración de los registros para entradas/salidas y manejo de interrupciones

PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	–	–	–	–	–	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 4: Configuración del registro PCMSK1[1]

PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	–	PCINT17	PCINT16	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCMSK2
Read/Write	R	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 5: Configuración del registro PCMSK2 [1]

GIMSK – General Interrupt Mask Register

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 6: Configuración del registro GMISK[1]

MCUCR – MCU Control Register

The External Interrupt Control Register contains control bits for interrupt sense control.

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 7: Configuración del registro MCUCR[1]

TIMSK – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 8: Configuración del registro TIMSK[1]

10.3.6 DDRB – Port B Data Direction Register

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 9: Configuración del registro DDRB[1]

10.3.5 PORTB – Port B Data Register

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 10: Configuración del registro PORTB[1]

11.9.3 TCNT0 – Timer/Counter Register

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 11: Configuración del registro TCNT0[1]

TCCR0A – Timer/Counter Control Register A

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 12: Configuración del registro TCCR0A[1]

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 13: Configuración del registro TCCR0B[1]

En las figuras 4, 5, 6, 7, 8, 9, 10, 12 y 13 es posible observar la configuración de los registros mencionados con su respectiva cantidad de bits y nombre correspondiente, los cuales fueron fundamentales para la realización del programa del juego.

Librerías

Para este laboratorio se utilizaron dos librerías que contiene **AVR**, como se muestra acontinuación:

avr/interrupt.h

Esta biblioteca ofrece una serie de macros y funciones para manejar y controlar las interrupciones en los microcontroladores AVR. Las interrupciones permiten que el microcontrolador reaccione de inmediato a eventos externos o internos, sin que el código principal tenga que monitorear continuamente esos eventos. Entre las funciones principales de esta librería se encuentran la habilitación y deshabilitación global de las interrupciones, así como la definición de los vectores de interrupción, que son los controladores que se ejecutan cuando ocurre una interrupción. [3]

avr/io.h

Esta biblioteca proporciona acceso a todos los registros y funcionalidades de bajo nivel del hardware de los microcontroladores AVR. Define macros y funciones para interactuar con los pines de entrada/salida (GPIOs), temporizadores, contadores y otros periféricos del microcontrolador. A través de esta librería, puedes configurar y controlar los puertos, así como ajustar los registros de control para habilitar o deshabilitar funcionalidades específicas del hardware, entre otras tareas.[4]

Conceptos importantes

Rebote de un pulsador

Cada vez que se activa un botón en un sistema electrónico, no siempre hay una transición suave entre los estados de encendido y apagado. Los interruptores mecánicos, en particular, son susceptibles a un fenómeno conocido como switch bounce, donde la señal de conmutación exhibe un comportamiento oscilatorio cuando se acciona el interruptor o botón. Esto puede resultar de una vibración mecánica cuando un interruptor cambia su contacto y colisiona con el otro polo en el ensamblaje del interruptor. [6]

Hardware

A continuación, en la figura 14 es posible observar la configuración que se realizó para el circuito de este programa:

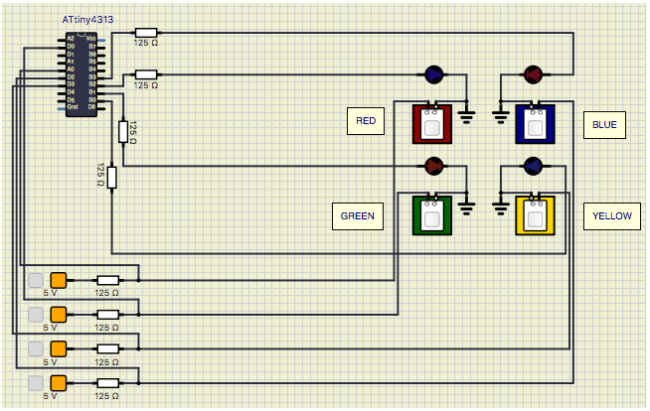


Figura 14: Diseño realiza para el circuito del juego

El diseño del juego Simón se implementa utilizando 4 botones, 4 leds, 8 resistencias y 4 fuentes de tensión de 5V. Las resistencias se utilizan para evitar rebotes del pulso en los botones, de igual manera en los LEDs. Específicamente, se tiene 4 pines de entrada (botones), al igual que de salida(LEDs). Los pines **PD3, PA0, PD2 y PD0** son los que corresponde a los botones de entrada: amarillo, rojo, azul y verde respectivamente. Por otra parte, se tiene los pines **PB0, PB1, PB2 Y PB3** que controlan los LEDs, es decir, los pines de salida: amarillo, verde, rojo y azul respectivamente.

Para determinar el valor de las resistencias, se aplicó la Ley de Ohm, considerando que la tensión de la fuente es de **5V** y que, como se muestra en la figura 3, la corriente es de 40 mA. A partir de estos datos, se obtiene lo siguiente:

$$V = I \cdot R$$

$$R = \frac{V}{I}$$

$$\boxed{R = \frac{5V}{40 \cdot 10^{-3} A} = 125, \Omega}$$

Este valor de resistencia se seleccionó también para los LEDs, asegurando que la corriente que reciben no exceda los 40 mA, previniendo así sobrecargas y prolongando la vida útil de los componentes.

Tabla de componentes electrónicos

Posteriormente, se hizo un calculo aproximado del costo de los componentes para poder elaborar este circuito físicamente, por lo tanto se utilizó Amazon [5] como referencia del precio de los productos:

Componente	**Cantidad**	**Precio ($)**
ATtiny4313	1	1.25
Source 5V	4	5.95
Resistencias de 125 Ω	8	6.99
LED's Rojo, verde, amarillo y azul	4	5.69
Botones	4	5.99
TOTAL		25,87

Tabla 1: Componentes electrónicos del diseño con precio [5]

Desarrollo

Diagrama de flujo

Para poder trabajar con una **FSM** es necesario tener estados, por lo tanto, por medio de este diagrama se estableció el funcionamiento esperado del software con los estados necesarios para poder realizar todos los requerimientos del programa. Por ende, se muestra la siguiente propuesta de seis estados para la máquina:

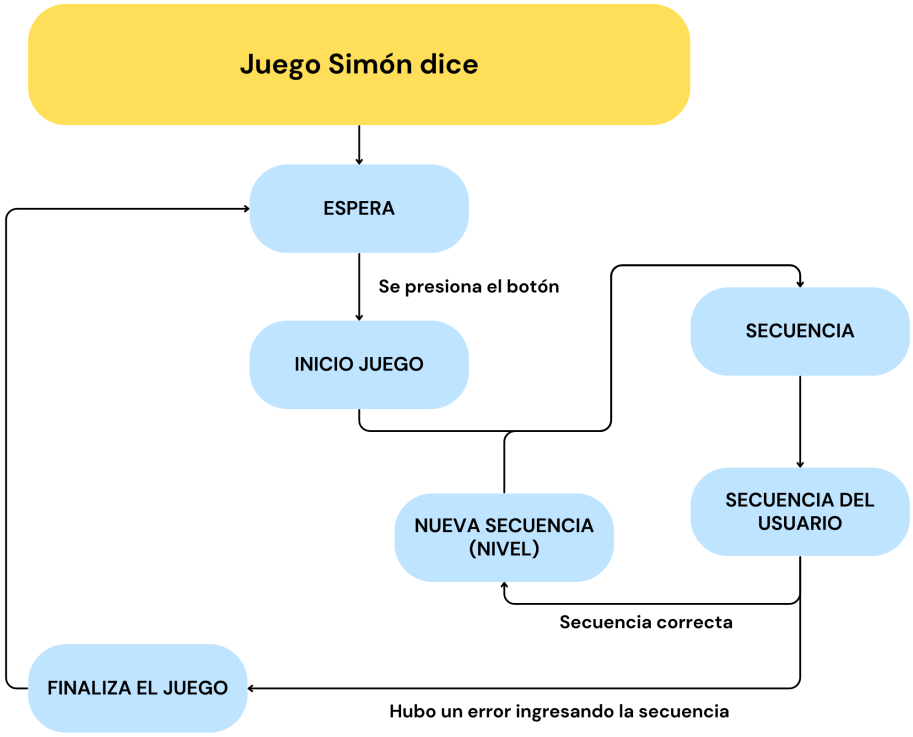


Figura 15: Diagrama de flujo del circuito de acuerdo a los estados

- **START (Inicio del juego)**: Este estado se activa cuando el juego es iniciado presionan alguno de los botones.El sistema prepara las variables y estructuras necesarias para comenzar el juego, como inicializar la secuencia de luces o LEDs que deberá ser mostrada al usuario. Se pasa al estado **GAME_SEQUENCE** una vez que el sistema ha preparado todo y el usuario ha indicado estar listo (por ejemplo, al presionar un botón).
- **GAME_SEQUENCE (Mostrar secuencia del juego)**: En este estado, el sistema presenta al usuario una secuencia visual, por medio de los LEDs que se encienden en un orden específico. El propósito de esta secuencia es que el usuario la memorice para reproducirla más tarde. Durante este estado, los LEDs parpadean siguiendo la secuencia predefinida o aleatoria generada por el juego. **Transición**: Una vez que la secuencia ha sido mostrada por completo, el juego pasa al estado **USER_SEQUENCE** para permitir al usuario reproducirla.
- **USER_SEQUENCE (Input del usuario)**: Aquí es donde el usuario interactúa directamente con el sistema, introduciendo la secuencia que recuerda a través de botones. El sistema compara cada entrada del usuario con la secuencia mostrada previamente en **GAME_SEQUENCE**. Si el usuario ingresa correctamente toda la secuencia, se pasa

al siguiente nivel, incrementando la dificultad. Si el usuario se equivoca en algún paso, el juego finalizará. Si el usuario ingresa la secuencia correctamente, el estado cambia a **LEVEL**. Si el usuario comete un error, se transita al estado **END**.

- **LEVEL (Nivel)**: Este estado indica que el jugador ha completado correctamente la secuencia del nivel actual y se sigue al próximo nivel. El juego incrementa la dificultad del nivel, generalmente añadiendo más pasos a la secuencia. Se prepara una nueva secuencia que incluirá todos los pasos anteriores, además de un paso nuevo. Después de preparar la nueva secuencia, el juego regresa al estado **GAME_SEQUENCE** para mostrar la secuencia ampliada al usuario.
- **END (Termina el juego)**: El estado **END** es activado cuando el usuario comete un error al reproducir la secuencia. Para indicar el final del juego, todos los LEDs parpadean de manera simultánea.
- **WAIT_TIME (Espera)**: Finalmente, este estado espera la interacción del usuario para continuar. El sistema esta pendiente de recibir la entrada del usuario, como cuando se presiona el botón para iniciar. Este estado ocurre antes de que el juego comience. Una vez que el usuario presiona el botón, el sistema pasa al estado **START**.

Funcionamiento del circuito

El circuito se simuló en el programa SimulIDE de forma que se pudiera implementar la funcionalidad de un Simón Dice simple a partir del uso de un microcontrolador ATtiny4313. Al comenzar la simulación se espera la pulsación de cualquier botón para comenzar a mostrar una secuencia de colores con el objetivo de que el usuario pueda recordarlos posteriormente y presionar los botones respectivos, incrementando la dificultad cada vez.

Al presionar el botón se parpadean todos los leds 2 veces como indicación al usuario, tal como se observa en la siguiente figura.

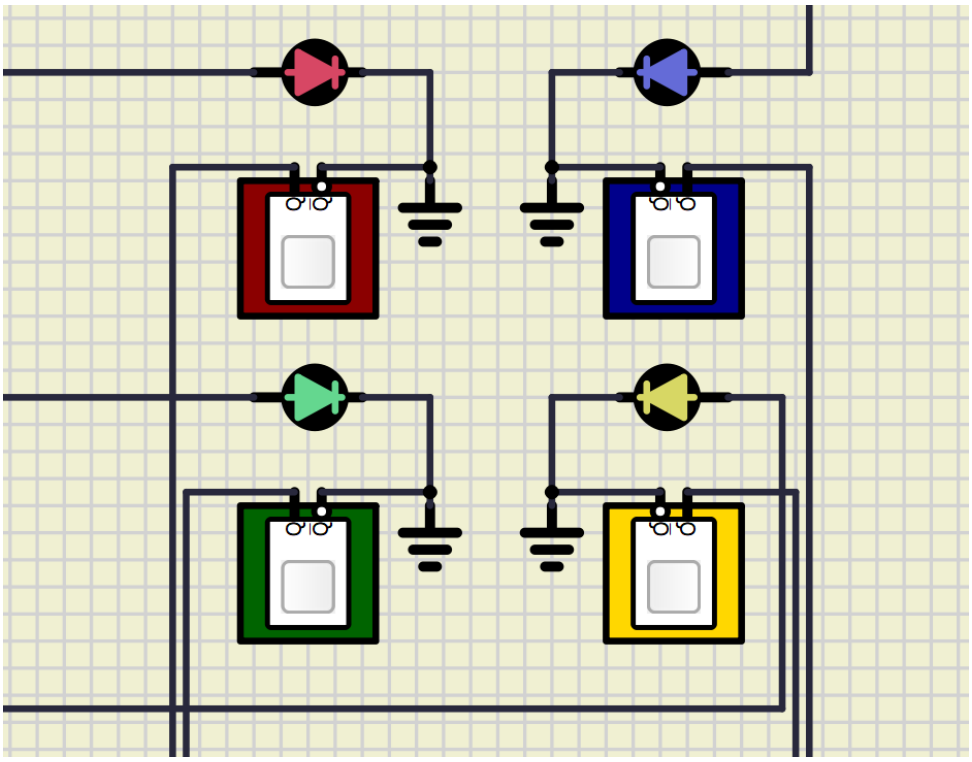


Figura 16: Parpadeo de todos los leds como indicación al usuario del inicio del juego

Posteriormente, se muestra una secuencia de 2 colores. Una vez que el usuario presiona los botones correctamente, se continúa con una secuencia en la que se añade un color más y se reduce el tiempo de encendido de los leds en 200ms.

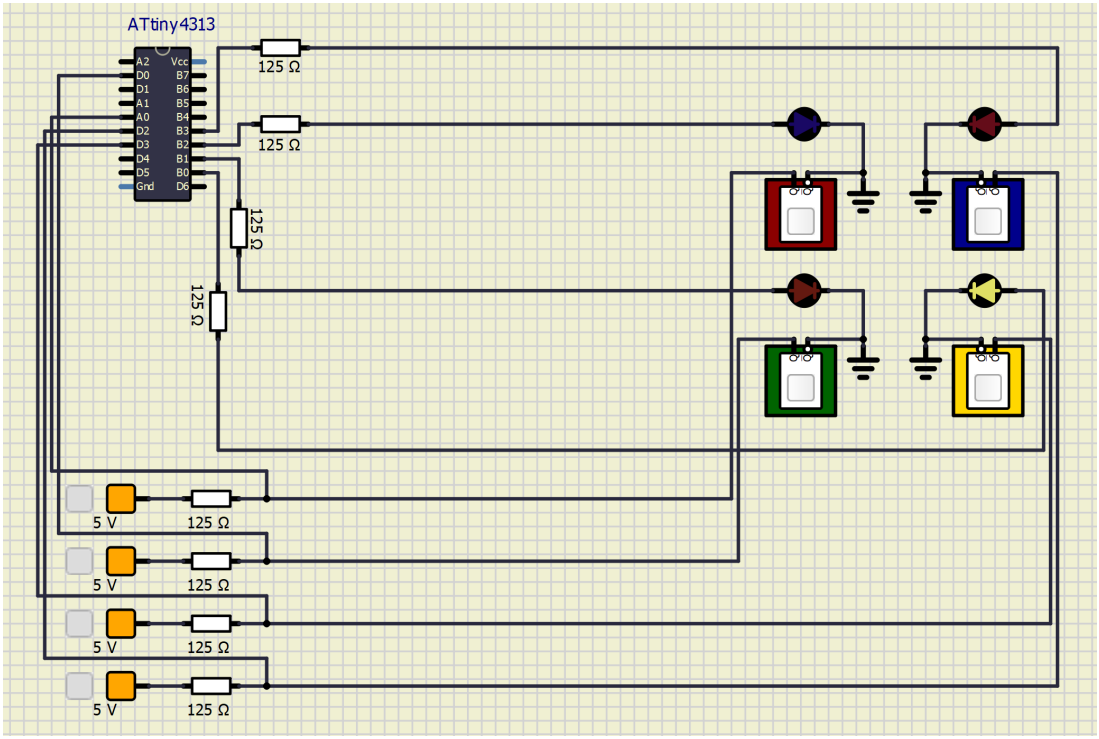


Figura 17: Parpadeo del led amarillo en una secuencia

Si el usuario continúa acertando las secuencias, se sigue incrementando el nivel (añadiendo un color más y reduciendo el tiempo en el que el led está encendido). En el momento en el que el usuario se equivoque con algún color, parpadearán todos los leds 3 veces como indicación al usuario de que el juego ha finalizado y se entra en un estado en el que se espera a que el usuario presione algún otro botón para volver a comenzar.

Análisis electrónico

Este circuito utiliza un microcontrolador ATtiny4313 para controlar cuatro LEDs (verde, rojo, azul y amarillo) con la ayuda de botones, implementando una máquina de estados finita (FSM). Cada LED está conectado en serie con una resistencia de 125Ω, como se puede ver en el diseño del circuito. Esto es necesario para proteger tanto los LEDs como el microcontrolador al limitar la corriente. El control de los LEDs se realiza mediante los pines de salida del microcontrolador, mientras que los botones están conectados a las interrupciones del microcontrolador para iniciar diferentes secuencias de color.

En términos de diseño eléctrico, los LEDs no soportan corrientes mayores a 40 mA, por lo que se utilizaron resistencias de 125Ω. Esto garantiza que, a un voltaje de 5V, la corriente a través de los LEDs se mantenga en un nivel seguro. Utilizando la ley de Ohm, se espera una corriente menor a 40 mA cuando se aplica un voltaje de 5V. El cálculo se realiza considerando que el voltaje nominal de los LEDs es de 5V, y las resistencias seleccionadas aseguran la limitación adecuada de corriente.

En el circuitos, los voltajes medidos (como los 2.41V vistos en la imagen) indican que los LEDs están operando en un estado de encendido parcial, probablemente debido a las características de los ciclos de PWM (modulación por ancho de pulso) o al estado de transición

dentro de la FSM. En este estado, no todos los LEDs están alimentados completamente, lo que explica el valor de voltaje intermedio medido en la imagen.

En el caso de los botones, se utiliza una fuente de 5V para alimentar el circuito. Cada botón está configurado para activar una interrupción en el microcontrolador cuando se presiona, lo que permite a la FSM actualizar el estado de los LEDs según la secuencia predefinida. Las resistencias de los botones también limitan la corriente que pasa a través de los circuitos de entrada del microcontrolador, protegiendo los pines de entrada.

Se utilizan dos tipos de funciones de interrupción: una controlada por los botones y otra por el temporizador. Para el control de botones, se emplea una variable auxiliar llamada color que contiene un valor entre 0 y 3 para representar los colores asociados a los botones, o un valor de 5 para indicar que ningún color ha sido seleccionado. Cuando se presiona un botón, la función correspondiente asigna el valor 0, 1, 2 o 3, dependiendo del color seleccionado. En cuanto al temporizador, se configura para operar con una frecuencia de 60 Hz. Esto significa que el temporizador genera una interrupción cada 1/60 de segundo, y cada vez que el contador interno llega a 60 ciclos, ha pasado un segundo completo. Adicionalmente, hay contadores que permiten medir tiempos intermedios, como 200 ms, para controlar con mayor precisión el parpadeo de los LEDs y la duración de los ciclos en la FSM.

Este diseño garantiza un control eficiente de las luces y una correcta protección de los componentes electrónicos. Además, al incluir las mediciones de voltaje y corriente esperada, se puede garantizar que los LEDs no se sobrecarguen, y que el sistema funcione dentro de los parámetros de seguridad recomendados.

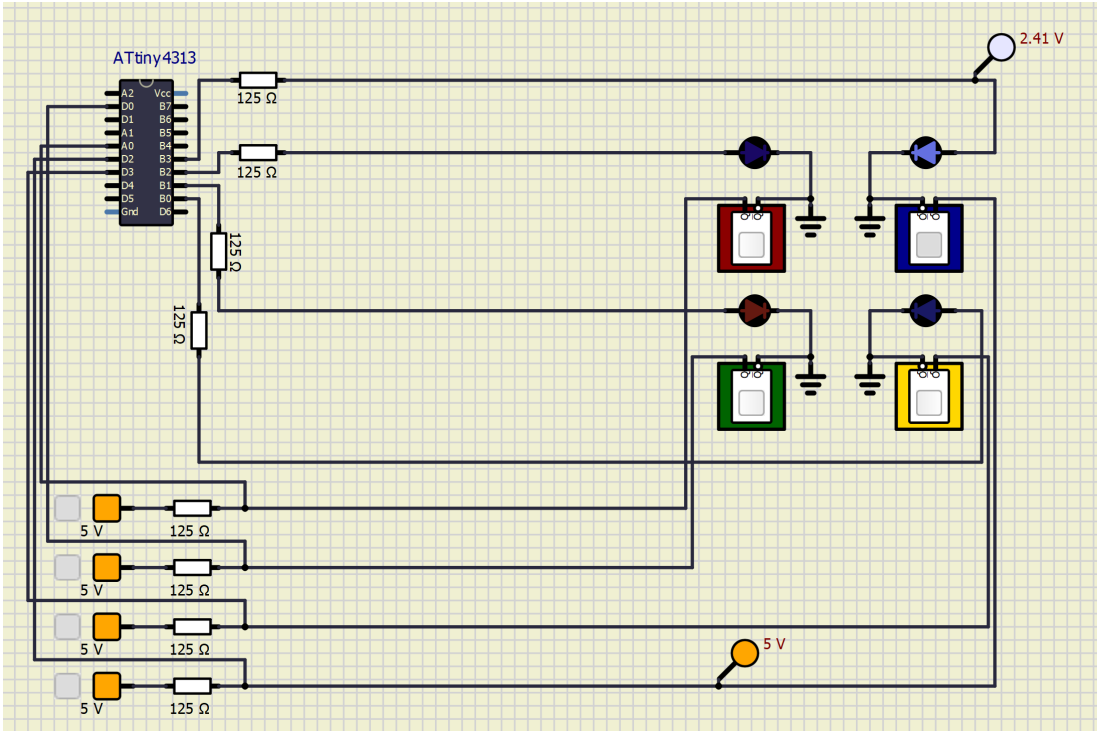


Figura 18: Medición de tensiones al presionar el botón azul y encender esta led

Conclusiones y recomendaciones

En este proyecto, se logró implementar un juego de memoria tipo 'Simón Dice' utilizando el microcontrolador ATtiny4313, controlando cuatro LEDs y cuatro botones mediante una máquina de estados finita (FSM). Considerar el sistema como una FSM facilitó su programación, ya que permitió gestionar las salidas en diferentes momentos del tiempo de manera más estructu-

rada. Es recomendable organizar el código en funciones y módulos más pequeños y específicos para facilitar futuras modificaciones y mejoras, influyendo en la legibilidad y reutilización.

El diseño del circuito fue cuidadosamente desarrollado para asegurar la protección de los componentes, utilizando resistencias adecuadas para limitar la corriente a través de los LEDs y evitar sobrecargas.

El uso de un prescaler ayudó a reducir la frecuencia a la que opera el microcontrolador, logrando los tiempos exactos requeridos para el programa. La sincronización mediante interrupciones temporales garantizó la fluidez en la ejecución del juego y la lectura de los botones, permitiendo el desarrollo correcto del circuito, el uso esperado de los botones y el incremento de la dificultad de la secuencia de luces conforme el usuario progresaba, sin sobrecargar al microcontrolador.

Referencias

[1] ATMEL. ATtiny2313A/4313 Data Sheet. https://www.microchip.com/en-us/product/attiny4313#document-table

[2] LinkedIn. (2023). What are some examples of interrupts and timers in a microcontroller?. https://www.linkedin.com/advice/1/what-some-examples-interrupts-timersmicrocontroller#:~:text=Interrupts%20are%20signals%20that%20tell,serial%20communication%2C%20or%20other%20peripherals.

[3] TU Chemnitz. (s.f.). Interrupts in AVR. https://www-user.tu-chemnitz.de/~heha/hsn/chm/avr-libc.chm/avr-interrupts.html

[4] Lund University. (2010). avr/io.h Library in AVR Libc User Manual. https://www.eit.lth.se/fileadmin/eit/courses/edi021/avr-libc-user-manual/group__avr__io.html

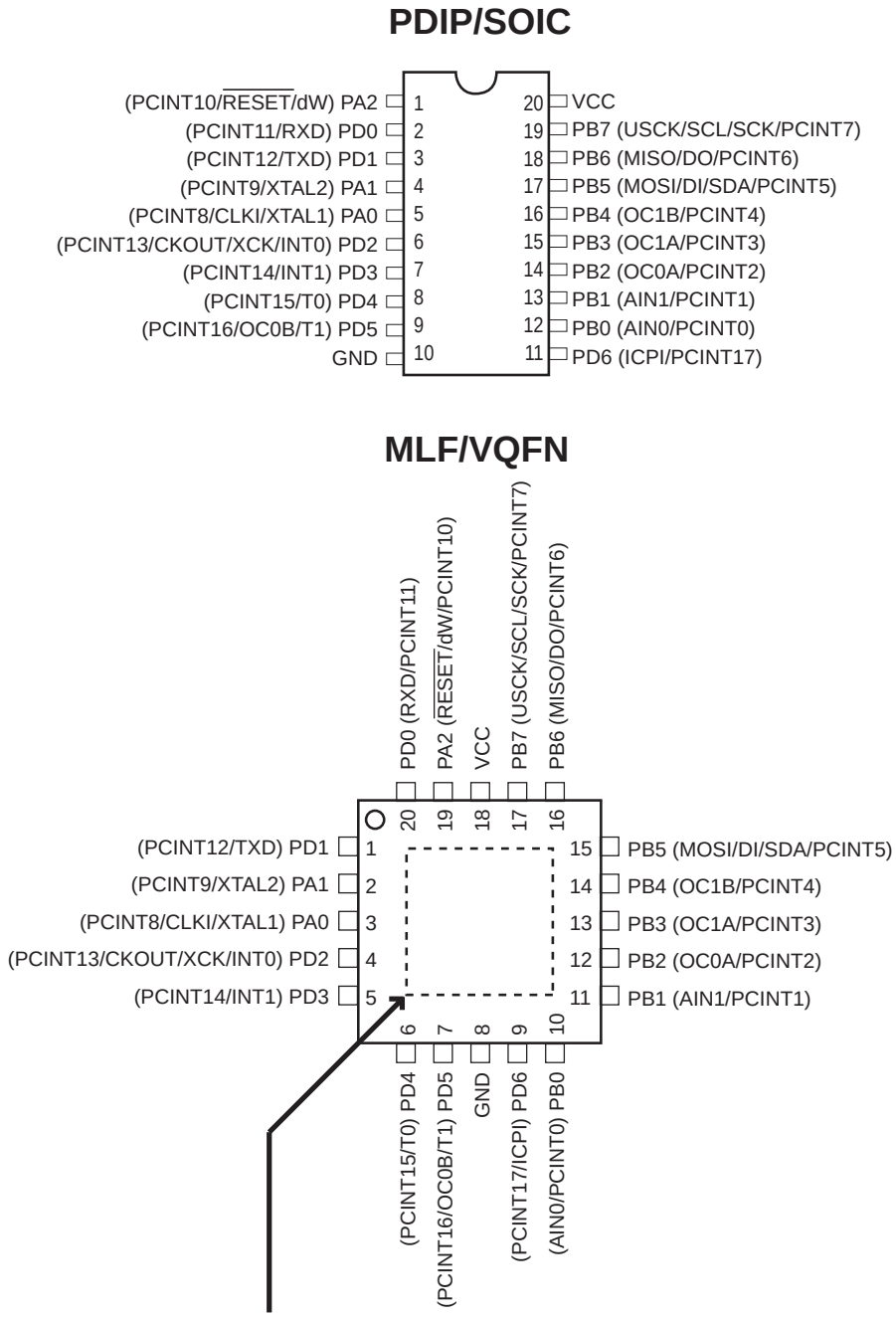
[5] Amazon. https://www.amazon.com

[6] Cadence, "How to Reduce Switch Bounce", s.f. [Online]. Disponible en: https://resources.pcb.cadence.com/blog/how-to-reduce-switch-bounce

Apéndices

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

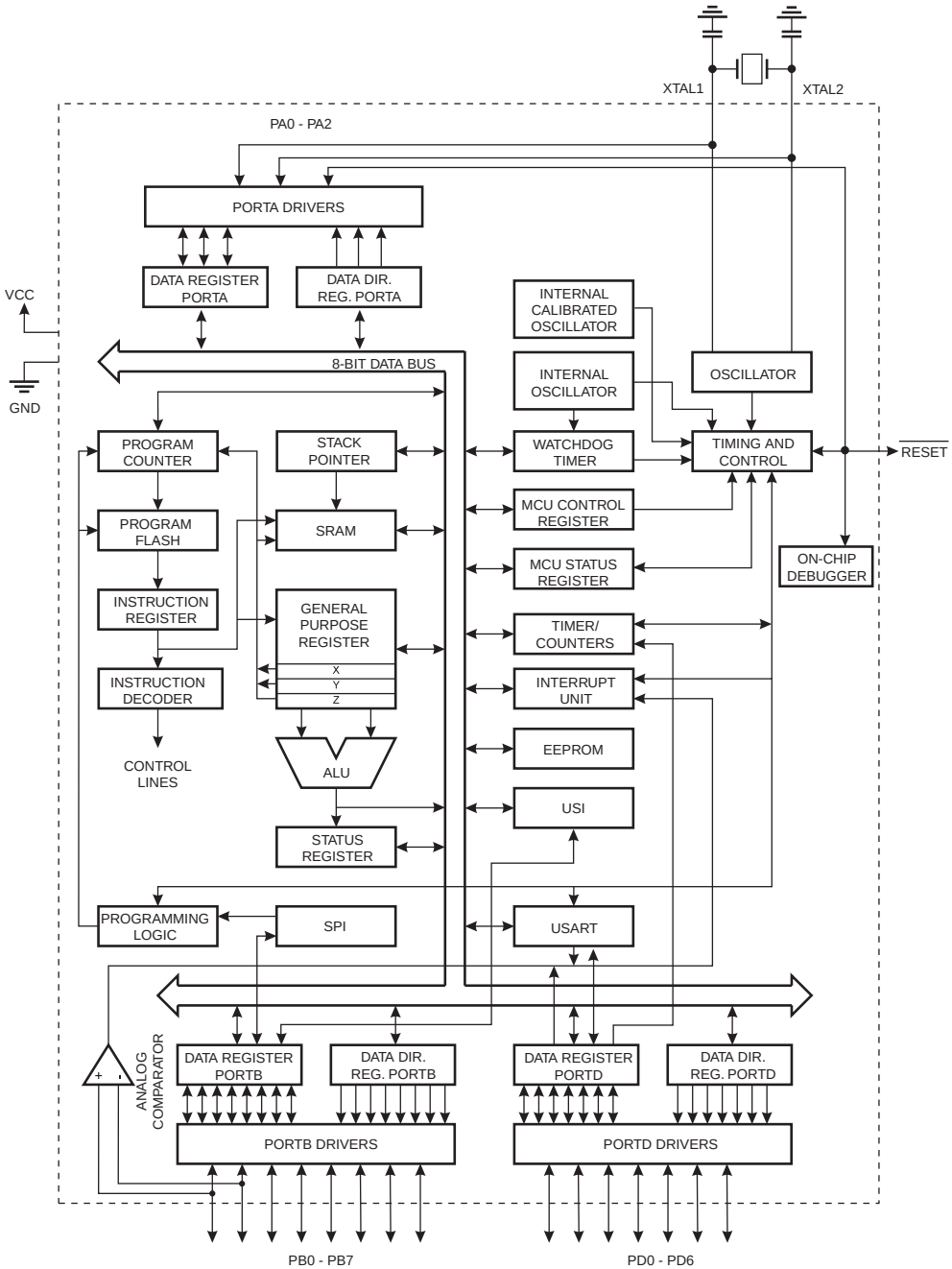
Output from the inverting Oscillator amplifier. XTAL2 is an alternate function for PA1.

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



9.3 Register Description

9.3.1 MCUCR – MCU Control Register

The External Interrupt Control Register contains control bits for interrupt sense control.

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	

- **Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 9-2. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt

Table 9-2. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

- **Bits 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 9-3. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 9-3. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

9.3.2 GIMSK – General Interrupt Mask Register

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	

- **Bits 2..0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control bits (ISC11 and ISC10) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

- **Bit 6 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control bits (ISC01 and ISC00) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

- **Bit 5 – PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 Register.

- **Bit 4 – PCIE2: Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT17..11 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI2 Interrupt Vector. PCINT17..11 pins are enabled individually by the PCMSK2 Register.

- **Bit 3 – PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT10..8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT10..8 pins are enabled individually by the PCMSK1 Register.

9.3.3 GIFR – General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A)	INTF1	INTF0	PCIF0	PCIF2	PCIF1	–	–	–	GIFR
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	

- **Bits 2..0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 7 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

- **Bit 6 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

- **Bit 5 – PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF becomes set (one). If the I-bit in SREG and the PCIE0 bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 4 – PCIF2: Pin Change Interrupt Flag 2**

When a logic change on any PCINT17..11 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 3 – PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT10..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

9.3.4 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	–	PCINT17	PCINT16	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCMSK2
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

These bits are reserved and will always read as zero.

- **Bits 6..0 – PCINT17..11: Pin Change Enable Mask 17..11**

Each PCINT17..11 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT17..11 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT17..11 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

9.3.5 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	–	–	–	–	–	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bits 2..0 – PCINT10..8: Pin Change Enable Mask 10..8**

Each PCINT10..8 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT10..8 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT10..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

9.3.6 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
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	

- **Bits 7..0 – PCINT7..0: Pin Change Enable Mask 7..0**

Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

Table 10-10. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1/ PCINT14	PD2/INT0/XCK/CKOUT/ PCINT13	PD1/TXD/ PCINT12	PD0/RXD/PCINT11
PUOE	0	0	TXD_OE	RXD_OE
PUOV	0	0	0	PORTD0 • $\overline{\text{PUD}}$
DDOE	0	0	TXD_OE	RXD_EN
DDOV	0	0	1	0
PVOE	0	XCKO_PVOE	TXD_OE	0
PVOV	0	XCKO_PVOV	TXD_PVOV	0
PTOE	0	0	0	0
DIEOE	INT1 Enable + PCINT14	INT0 Enable/ XCK Input Enable/PCINT13	PCINT12	PCINT11
DIEOV	PCINT14	PCINT13	PCINT12	PCINT11
DI	INT1 Input/ PCINT14	INT0 Input/XCK Input/ PCINT13	PCINT12	RXD Input/PCINT11
AIO	–	–	–	–

10.3 Register Description

10.3.1 MCUCR – MCU Control Register

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	

- **Bit 7 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See “Configuring the Pin” on page 56 for more details about this feature.

10.3.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	**PORTA2**	**PORTA1**	**PORTA0**	**PORTA**
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	–	–	–	–	–	**DDA2**	**DDA1**	**DDA0**	**DDRA**
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.4 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x19 (0x39)	–	–	–	–	–	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

10.3.5 PORTB – Port B Data Register

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	

10.3.6 DDRB – Port B Data Direction Register

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	

10.3.7 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

10.3.8 PORTD – Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x12 (0x32)	–	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.9 DDRD – Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x11 (0x31)	–	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.10 PIND – Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x10 (0x30)	–	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	