Escuela de Ingeniería en Electrónica

Licenciatura en Ingeniería en Electrónica

EL3313 Taller de Diseño Digital

Lab. 3: Interfaces con periféricos

Fiorela Chavarría Castrillo

William Sánchez Ramírez

Rolando Vega Marino

Alajuela, Costa Rica

Abril 2024

Las interfaces con periféricos son elementos que facilitan la comunicación entre dispositivos. Estas interfaces actúan como puentes que conectan componentes electrónicos, permitiendo que intercambien datos y se coordinen entre sí para ejecutar tareas específicas. Uno de los aspectos de las interfaces con periféricos es su capacidad de establecer protocolos de comunicación estándar que regulen la transferencia de datos. Estos protocolos definen cómo se envían, reciben y procesan los datos entre dispositivos.

Dentro de las interfaces con periféricos, se encuentran la interfaz SPI y la interfaz UART. Ambas son interfaces de comunicación en serie. Esto significa que transmiten datos un bit a la vez, secuencialmente, a través de un único canal de comunicación. Es común el uso de la comunicación en serie para sistemas electrónicos debido a la eficiencia en la transmisión de datos, principalmente en distancias cortas o medias y cuando se requiere una alta velocidad de transferencia.

SPI

La interfaz SPI (Serial Peripheral Interface) opera bajo un protocolo maestro-esclavo (master-slave) y utiliza cuatro cables, los cuales son asignados como: SCLK, MOSI (master out, slave in), MISO (master in, slave out) y SS' (slave select; activo BAJO) [1]. Funciona de manera síncrona, utilizando un reloj compartido, y transmite datos secuenciales a través de la misma línea. En este contexto, el dispositivo maestro controla la comunicación y posee líneas distintas con propósitos específicos. Además, provee una comunicación full-duplex, lo que significa que tanto el esclavo como el maestro pueden transmitir señales simultáneamente, recibiendo y enviando datos al mismo tiempo [2].

La cantidad de bits transmitidos en SPI se deriva en la cantidad de pulsos de reloj necesarios para la transferencia. Las líneas de comunicación de la interfaz SPI son SCLK para el reloj, MOSI para enviar datos del maestro al esclavo, MISO para enviar datos del esclavo al maestro, y SS' para seleccionar un esclavo específico.

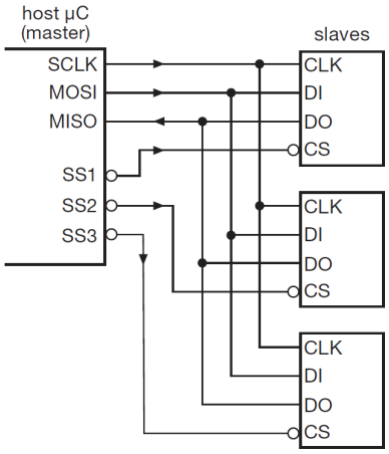

Fig. 1. Configuración de buses en SPI: señal de reloj y líneas de datos son compartidas, con líneas individuales de SS' activando las entradas de chip-select de los esclavos correspondientes [1].

En la comunicación SPI, cada intercambio de información entre dispositivos comienza cuando la línea de selección de esclavo SS' experimenta un cambio hacia un estado lógico bajo, generalmente interpretado como una señal activa baja. La configuración de la relación entre líneas de selección de esclavo, datos y reloj se define mediante los parámetros de polaridad del reloj (CPOL) y fase del reloj (CPHA) [2]. Estos parámetros dan lugar a cuatros posibles configuraciones de reloj conocidos como modos de SPI e identificados como Modo 0, Modo 1, Modo 2 y Modo 3.

La polaridad del reloj se refiere al estado en el que se encuentra la señal del reloj cuando no está activa, pudiendo ser alta o baja. Por otro lado, la fase del reloj determina el momento en que se registra la información en el pin de salida. Esta acción puede ocurrir en el flanco de bajada, cuando la señal cambia de alto a bajo, o en el flanco de subida, cuando la señal cambia de bajo a alto [3].

Modos de SPI

En [2] se detallan los modos en la configuración del reloj en interfaces como la SPI.

- **(Polaridad de reloj no invertida)** Reloj en nivel lógico bajo cuando la selección de esclavo pasa a nivel lógico bajo

  - *Modo 0* **(CPOL=0, CPHA=0)**

    Los datos se muestrean en el flanco ascendente del pulso del reloj y se desplazan en el flanco descendente del pulso de reloj.

  - *Modo 1* **(CPOL=0, CPHA=1)**

    Los datos se muestrean en el flanco descendente del impulso de reloj y se desplazan hacia fuera en el flanco ascendente del impulso de reloj.

- **(Polaridad de reloj invertida)** Reloj en nivel lógico alto cuando la selección de esclavo pasa a nivel lógico bajo

  - *Modo 2* **(CPOL=1, CPHA=0)**

    Los datos se muestrean en el flanco descendente del impulso del reloj y se desplazan hacia afuera en el flanco ascendente del impulso de reloj.

  - *Modo 3* **(CPOL=1, CPHA=1)**

    Los datos se muestrean en el flanco ascendente del impulso de reloj y se desplazan hacia fuera en el flanco descendente del impulso de reloj.

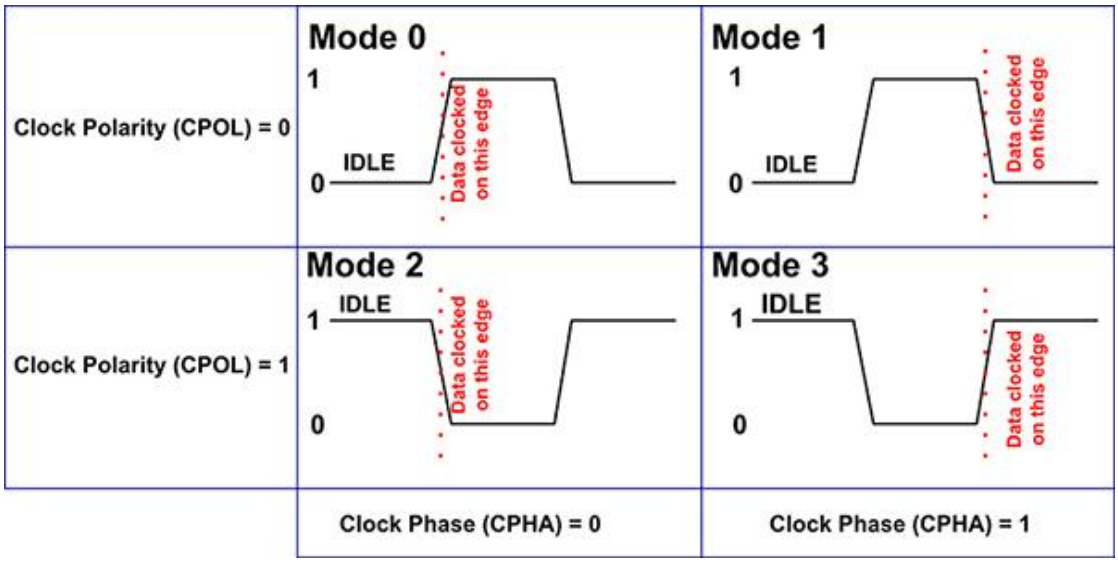

Fig. 2. La selección del modo de cronómetro de la SPI fija el borde de reloj activo en el que se toman muestras de los datos [3].

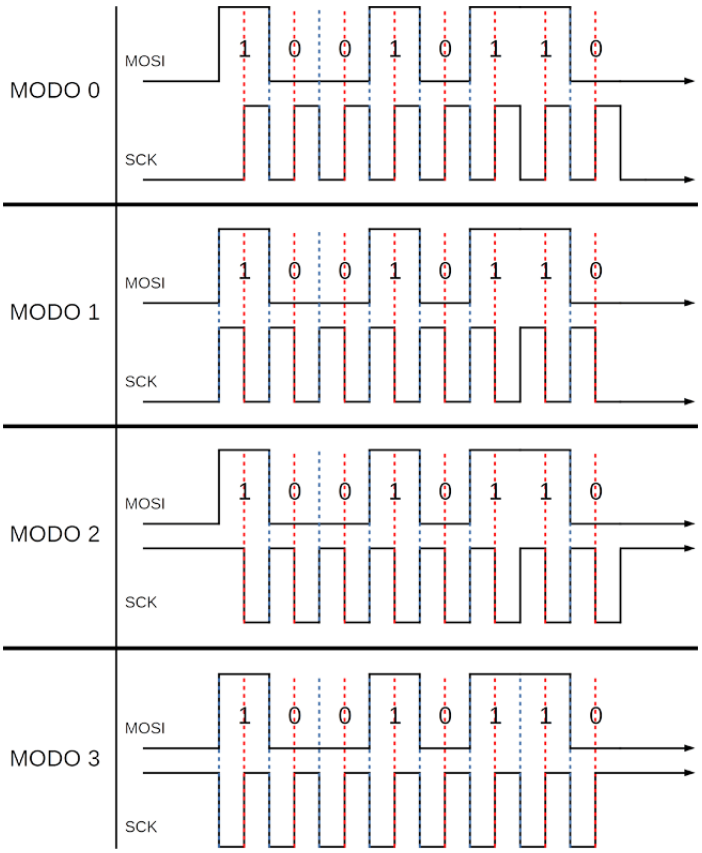

Fig. 3. Señales de reloj y datos para los modos. En azul los flancos en que se actualiza. En rojo los flancos en que el esclavo lee el valor [4].

UART

La comunicación en serie UART (Universal Asynchronous Receiver/Transmitter) es un protocolo de comunicación asíncrona, lo que significa que no requiere un reloj compartido. Actúa como un puente entre interfaces paralelas y en serie permitiendo la transmisión de datos en ambas direcciones. Para establecer la comunicación UART, se utilizan dos líneas principales: una para transmitir datos (Tx) y otra para recibirlos (Rx). Además, se requiere una conexión a tierra común para completar el circuito [5].

La UART puede operar en varios modos, como simplex (unidireccional), half-duplex (bidireccional, pero solo un dispositivo puede) y full-duplex (bidireccional, donde ambos dispositivos pueden enviar y recibir datos simultáneamente) [5].

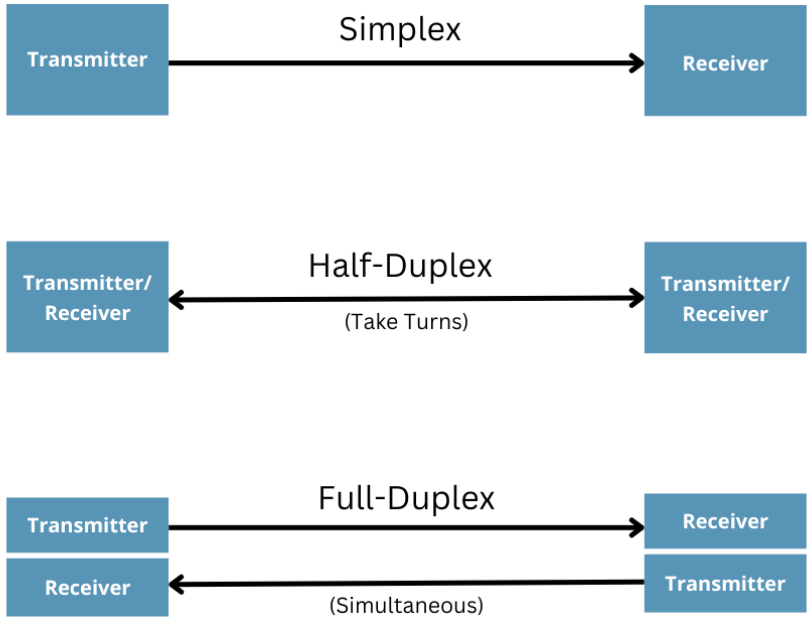

Fig. 4. Ejemplos de modos de operación según la dirección en que transmiten o reciben los datos [6].

Al configurar la UART, se ajustan parámetros como Baut Rate, número de bits de datos, la paridad y el número de bits de parada.

Baut rate

El baut rate, o velocidad de transmisión, indica la frecuencia con la que una señal cambia de estado, lo que significa que determina cuándo comienza un bit y cuándo termina otro. Lo cual representa el número de veces por segundo que una señal de comunicaciones serie cambia su estado, ya sea un nivel de voltaje, una frecuencia o un ángulo de fase de frecuencia [7]. Esta medida determina los bits individuales dentro de un flujo de datos, especialmente cuando varios bits tienen el mismo valor numérico que su predecesor.

Bits de inicio y parada

El bit de inicio marca el principio de una transmisión de datos. Se distingue por un cambio de nivel de voltaje de alto a bajo en la señal de transmisión, lo que indica al receptor que se espera la llegada de un nuevo conjunto de datos. Por otro lado, el bit de parada se sitúa al final de la transmisión y se caracteriza por un retorno al nivel de voltaje alto. Su función es delimitar la conclusión de los datos transmitidos y proporcionar al receptor un punto claro para el final del mensaje. Juntos, estos bits forman una trama de datos que ayuda al receptor a identificar e interpretar correctamente la información transmitida [5].

Bit de paridad

El bit de paridad funciona para verificar la integridad de los datos, lo que significa que detecta errores de transmisión durante la comunicación. Este bit se inserta entre los bits de datos y el bit de parada en una trama de datos UART.

La paridad puede ser de dos tipos: par o impar. En el caso de la paridad par, el bit de paridad se establece de manera que el número total de unos en el marco sea par. Por otro lado, en la paridad impar, el bit de paridad se ajusta para que el número total de unos sea impar. Al recibir los datos, el receptor verifica si la cantidad de unos en el marco coincide con el tipo de paridad especificado. Si no coinciden, se asume que ha habido un error en la transmisión y se puede tomar una acción correctiva, como solicitar una retransmisión de los datos. Sin embargo, es importante tener en cuenta que el bit de paridad solo puede detectar la presencia de un único bit erróneo en el marco de datos y no puede corregir errores [5].

Para poder utilizar los puertos serie en el sistema operativo Windows se buscan las opciones de software como emuladores de terminal, ya sea, PuTTY, RealTerm o algún otro programa que nos permita la correcta comunicación entre la computadora y los puertos serie.

La idea para configurar estos programas es que hay que tener en cuenta los valores que se configuraron anteriormente en el sistema, ya sea, velocidad de transmisión, bits de parada, bits de datos, etc.

Referencias

[1] P. Horowitz and W. Hill, *The art of electronics*, 3rd ed. Cambridge, England: Cambridge University Press, 2015.

[2] *Allaboutcircuits.com*. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/spi-serial-peripheral-interface/. [Accessed: 22-Apr-2024].

[3] *Digikey.com*. [Online]. Available: https://www.digikey.com/es/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices. [Accessed: 23-Apr-2024].

[4] J. G. Carmenate, “SPI con Arduino y sensor BMP280,” *Programarfacil Arduino y Home Assistant*, 06-Apr-2021. .

[5] Rohde and Schwarz International, “Qué es UART,” *Rohde-schwarz.com*. [Online]. Available: https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart_254524.html. [Accessed: 23-Apr-2024].

[6] J. Hopkins, “Difference between half-duplex vs full-duplex,” *Total Phase Blog*, 19-Oct-2022. [Online]. Available: https://www.totalphase.com/blog/2022/10/difference-between-half-duplex-vs-full-duplex/. [Accessed: 23-Apr-2024].

[7] “IBM Documentation,” *Ibm.com*, 24-Mar-2023. [Online]. Available: https://www.ibm.com/docs/es/aix/7.2?topic=parameters-baud-rate. [Accessed: 23-Apr-2024].