SSPI (pronunciado “S-P-I”) es un protocolo serial síncrono simple que es fácil de usar y relativamente rápido. La interfaz física consta de tres pines: Serial Clock (SCK), Master Out Slave In (MOSI, también conocido como SDO) y Master In Slave Out (MISO, también conocido como SDI). SPI conecta un dispositivo maestro a un dispositivo esclavo, como se muestra en la Figura e9.6(a). El maestro produce el reloj. Inicia la comunicación enviando una serie de pulsos de reloj en SCK. Si quiere enviar datos al esclavo, pone los datos en MOSI, comenzando con el bit más significativo. El esclavo puede responder simultáneamente poniendo datos en MISO. La Figura e9.6(b) muestra las formas de onda SPI para una transmisión de datos de 8 bits. Los bits cambian en el flanco descendente de SCK y son estables para muestrear en el flanco ascendente. La interfaz SPI también puede enviar una habilitación de chip activo-bajo para alertar al receptor que los datos están llegando. El BCM2835 tiene tres puertos maestros SPI y un puerto esclavo [1].

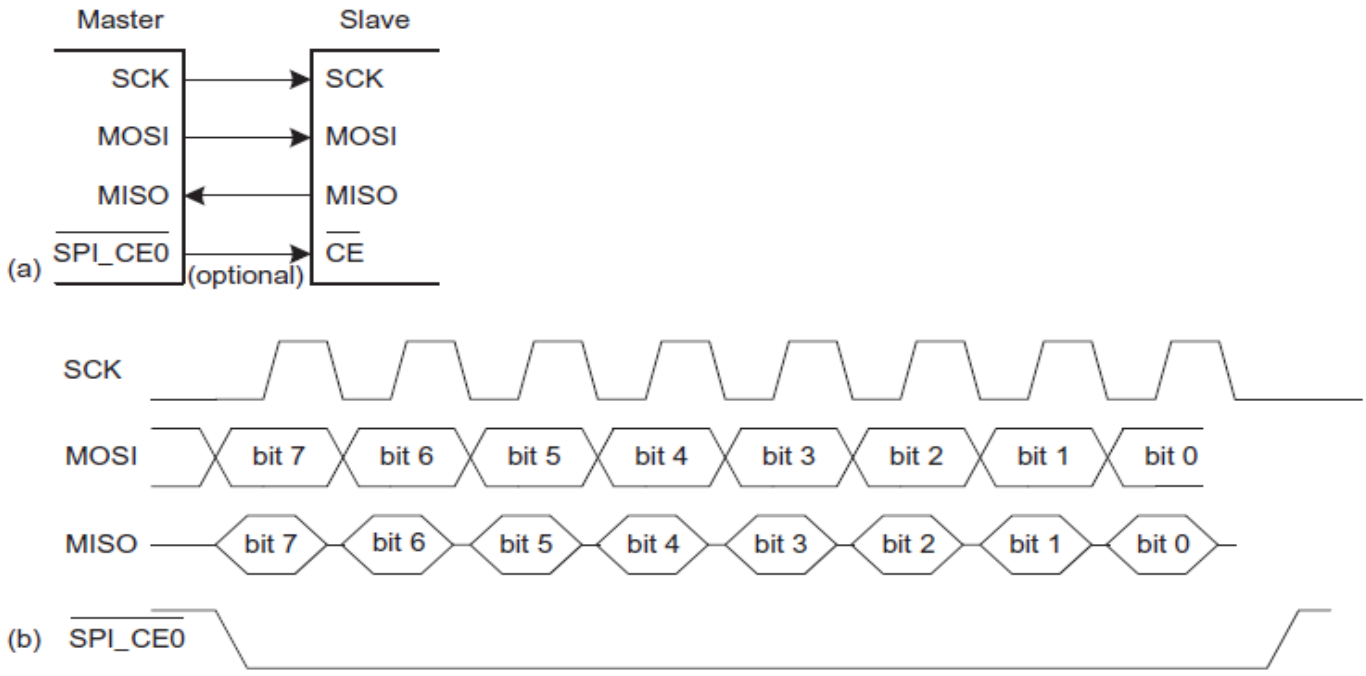

Figure e9.6 SPI connection and waveforms

Aspectos necesarios para poder diseñar un controlador maestro de SPI:

1. Conocimiento de la arquitectura del sistema SPI y sus señales de control.
2. Familiaridad con los registros asociados al puerto maestro SPI, como el registro de control (SPI0CS), el registro FIFO (SPI0FIFO) y el registro de reloj (SPI0CLK).
3. Comprensión de cómo configurar los atributos del puerto SPI, como la polaridad del reloj y la frecuencia del reloj.
4. Conocimiento sobre cómo transmitir y recibir datos a través del puerto SPI utilizando el registro FIFO.
5. Capacidad para programar en lenguajes de bajo nivel como C o ensamblador.

Diferentes modos de SPI:

El protocolo SPI tiene cuatro modos diferentes, que se diferencian en la polaridad y fase del reloj [1].

1. Modo 0: El reloj está inactivo en estado bajo (polaridad cero) y los datos se capturan en el flanco ascendente del reloj (fase uno).
2. Modo 1: El reloj está inactivo en estado bajo (polaridad cero) y los datos se capturan en el flanco descendente del reloj (fase dos).
3. Modo 2: El reloj está inactivo en estado alto (polaridad uno) y los datos se capturan en el flanco descendente del reloj (fase uno).
4. Modo 3: El reloj está inactivo en estado alto (polaridad uno) y los datos se capturan en el flanco ascendente del reloj (fase dos).

Cada modo tiene sus propias ventajas y desventajas, dependiendo de las necesidades específicas de la aplicación [1].

Comunicación UART

La comunicación serie UART (Universal Asynchronous Receiver/Transmitter) es un protocolo de comunicación de datos que se utiliza para transmitir información entre dos dispositivos. A diferencia del protocolo SPI, la comunicación UART no utiliza un reloj compartido para sincronizar la transmisión y recepción de datos. En su lugar, los dispositivos deben acordar previamente una velocidad de transmisión y generar su propio reloj local para sincronizar la transmisión y recepción de datos [1].
La comunicación UART se utiliza comúnmente en aplicaciones que requieren una transferencia de datos simple y sin procesamiento adicional, como la comunicación entre un microcontrolador y un módulo GPS o un sensor [1].

Características de configuración para la comunicación serie mediante UART.

Se describen las diferentes características de configuración necesarias para la comunicación serie mediante UART. Estas características son [1]:

1. Velocidad de transmisión (baud rate): La velocidad de transmisión se mide en baudios y representa el número de bits que se transmiten por segundo. Los dispositivos deben acordar previamente una velocidad de transmisión para asegurar que los datos se transmitan correctamente.
2. Número de bits por carácter: El número de bits utilizados para representar cada carácter transmitido. Un valor común es 8 bits.
3. Bit de paridad: Un bit adicional que se utiliza para detectar errores en la transmisión. Puede ser par, impar o ninguno.

4. Número de bits de parada: El número de bits adicionales que se utilizan para indicar el final del carácter transmitido. Un valor común es 1 bit.

Es importante que ambos dispositivos estén configurados con los mismos valores para estas características, ya que, si no lo están, los datos pueden ser garbled o ilegibles [1].

Cómo puede utilizar puertos serie en su computadora considerando el sistema operativo utilizado.

A continuación, se proporciona una descripción general de los pasos necesarios para hacerlo, considerando el sistema operativo que utilice [1]:

1. Identifique el puerto serie: En primer lugar, debe identificar el puerto serie al que está conectado su dispositivo. En Windows, puede hacer esto abriendo el Administrador de dispositivos y buscando la sección "Puertos (COM y LPT)". En Linux o macOS, puede utilizar el comando "dmesg" para ver los mensajes del kernel y buscar información sobre los puertos serie.
2. Configure PuTTY: Una vez que haya identificado el puerto serie, puede utilizar un programa como PuTTY para conectarse a él. Para hacer esto, abra PuTTY y seleccione "Serial" como tipo de conexión. Luego, configure la velocidad de transmisión (baud rate) y otros parámetros según las especificaciones de su dispositivo.
3. Conéctese al puerto serie: Después de configurar PuTTY, haga clic en "Open" para conectarse al puerto serie. Si todo está configurado correctamente, debería ver los datos transmitidos por su dispositivo en la ventana de PuTTY.

Es importante tener en cuenta que los pasos exactos pueden variar según el sistema operativo que esté utilizando y las especificaciones de su dispositivo. Consulte la documentación del fabricante o busque recursos en línea específicos para su dispositivo y sistema operativo si necesita ayuda adicional [1].

Referencias

[1] S. L. H. &. D. M. Harris, «Digital Design and Computer Architecture,» de *Digital Design and Computer Architecture*, 225 Wyman Street, Waltham, MA 02451, USA, Book Aid International, 206, pp. 531.e12-531.e23.