Técnicas Digitales II

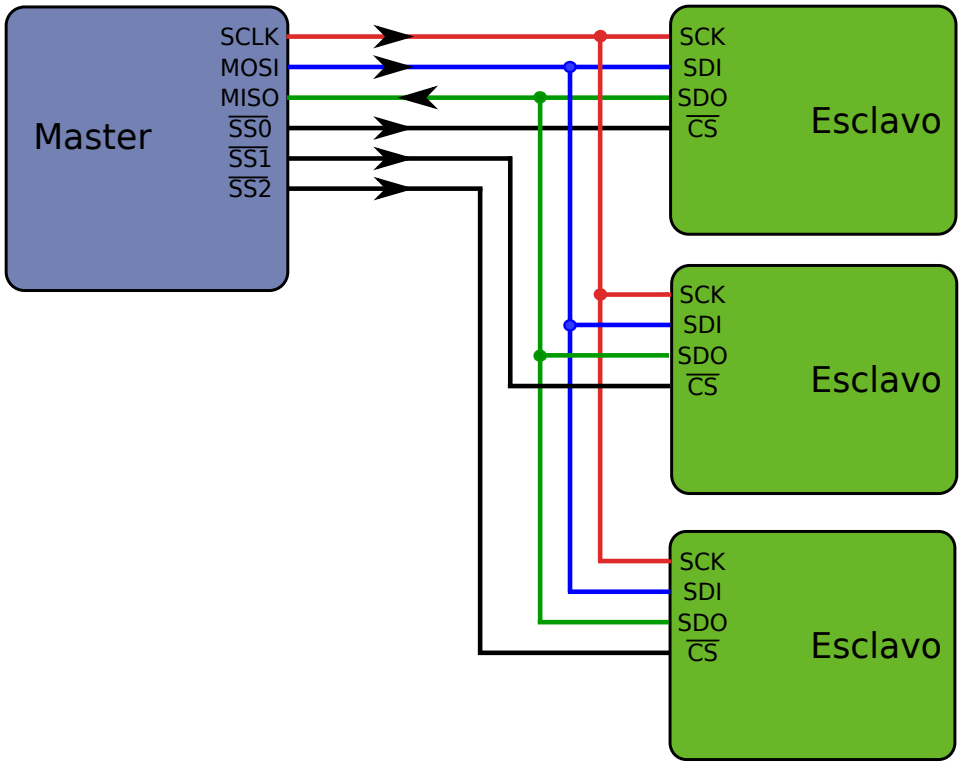
SPI (Serial Peripheral Interface)

Dr.Ing.Steiner Guillermo

Descripción General

- Estándar de facto desarrollado por Motorola finales de 1980.
- Es un protocolo de comunicación Serie Síncrono.
- Utilizado para comunicación de dispositivos dentro de la placa.
- Protocolo maestro / esclavo de simple maestro.
- Permite múltiple esclavos mediante la selección individual con una linea $\overline{SS}$ (Slave select).
- El bus es full-duplex de cuatro lineas de señales y una masa.

Esquema de la Red

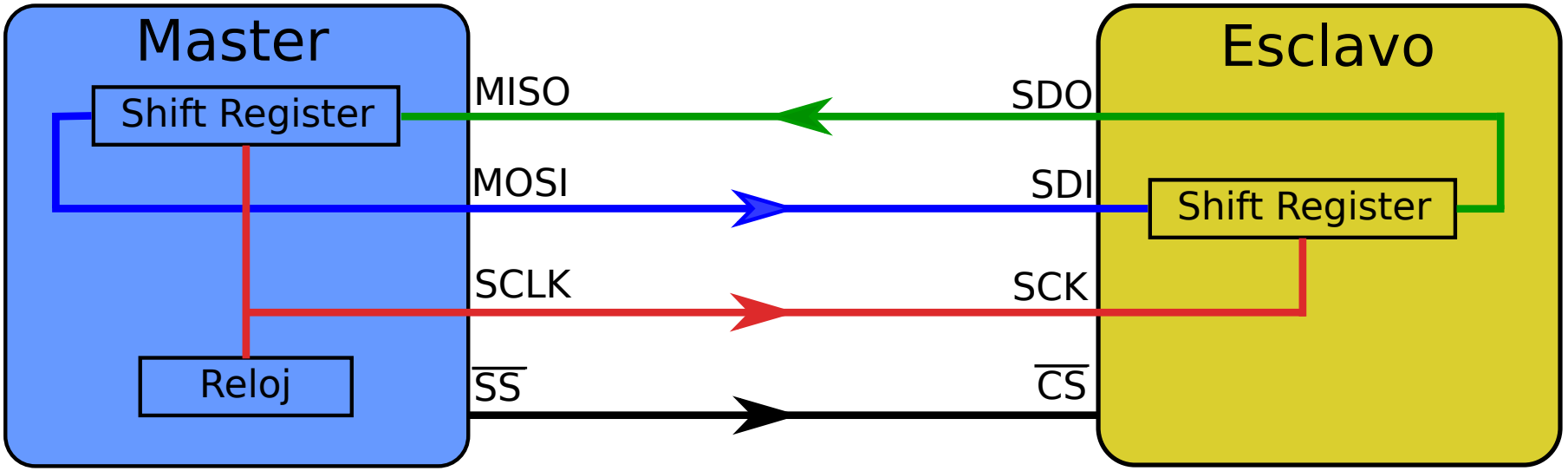


Señales

- 4 Lineas que permiten la comunicación
  - SCLK o SCK (Serial Clock) Reloj de sincronismo entregado por el master.
  - MOSI o SDO (Master Output Slave Input) Salida de dato del master.
  - MISO o SDI (Master Input Slave Output) Salida de dato del esclavo.
  - $\overline{SS}$ (Slave Select) Habilitador del esclavo.

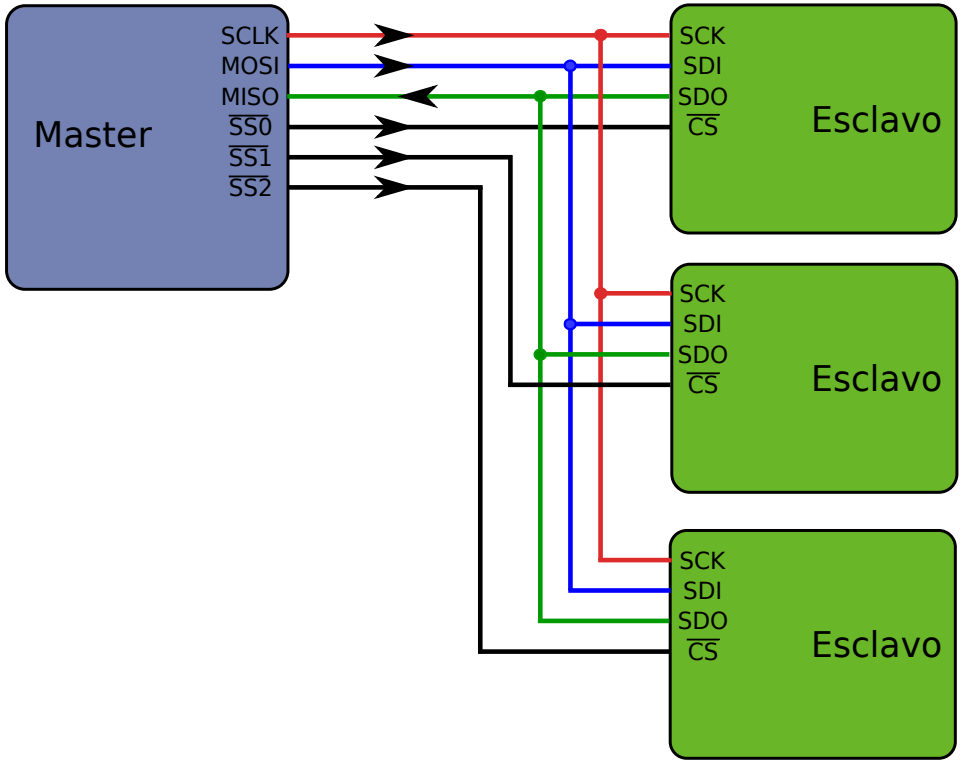
Transmisión de datos

- El master selecciona el esclavo con un 0 en en $\overline{SS}$
- Si el esclavo seleccionado requiere un tiempo de espera, el master debe esperar al menos ese tiempo
- Durante cada ciclo el master envía un bit por MOSI y el esclavo lo lee, en ese momento el esclavo envía un bit por MISO para que el master lo lea.



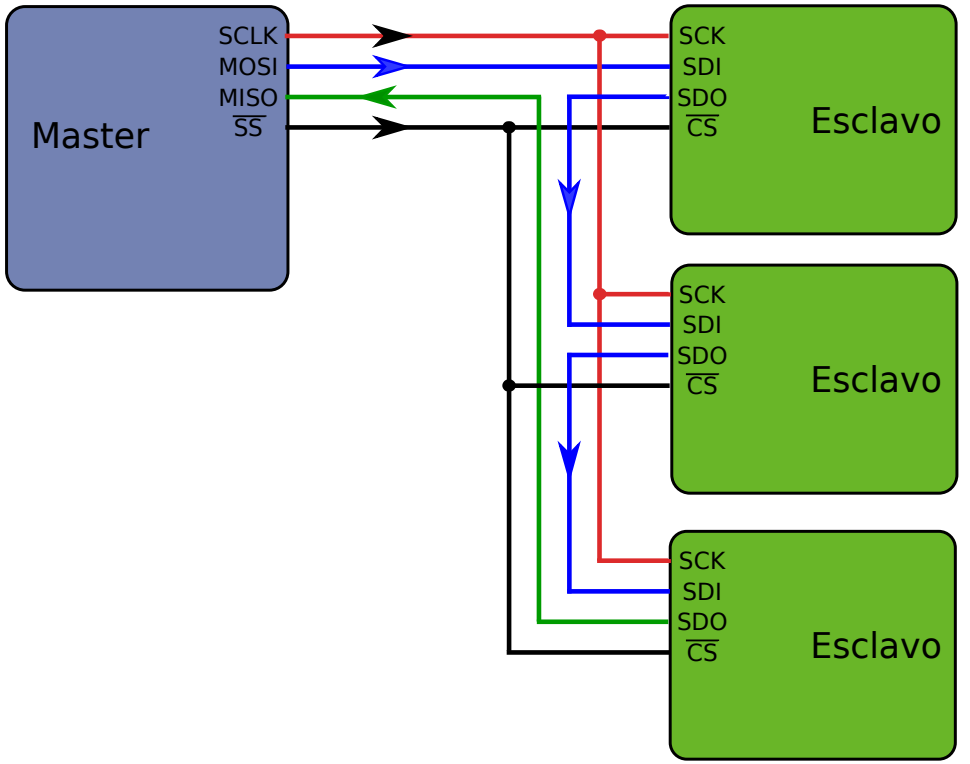
Configuración de Esclavos

- Esclavos Independientes
  - Cada esclavo tiene su selector.
  - Es la configuración mas usada.
  - Requiere que los esclavos no carguen la linea cuando están deshabilitados.



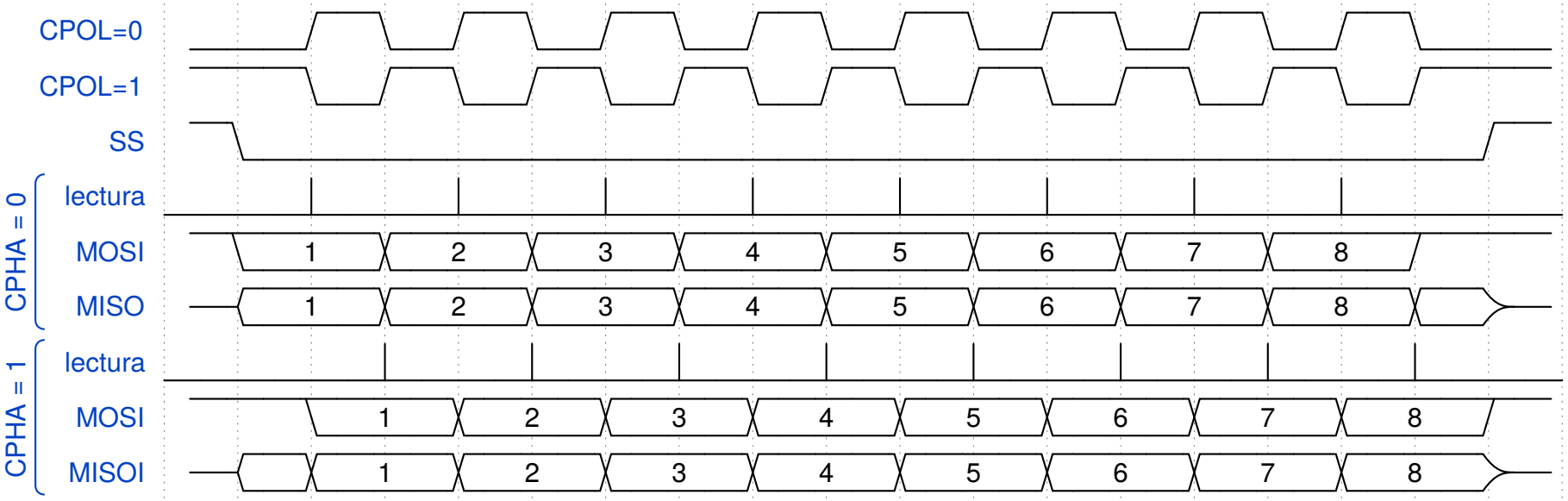
Configuración de Esclavos

- Cadena (daisy chain)
  - Una sola linea va atravesando cada esclavo.
  - Un bit ingresa a un esclavo y la salida de su shift, es enviada al próximo.
  - Requiere menos cables.
  - Utilizado donde no es necesario recibir datos.



Polaridad de Reloj y Fase

- Según donde se realice la lectura podemos encontrar 4 maneras diferentes de configurar el reloj en el SPI.
- CPOL=0 clock inactivo en 0, CPOL=1 clock inactivo en 1.
- CPHA=0 se utiliza el 1er semiciclo, con CPHA=1 el 2do



Ventajas

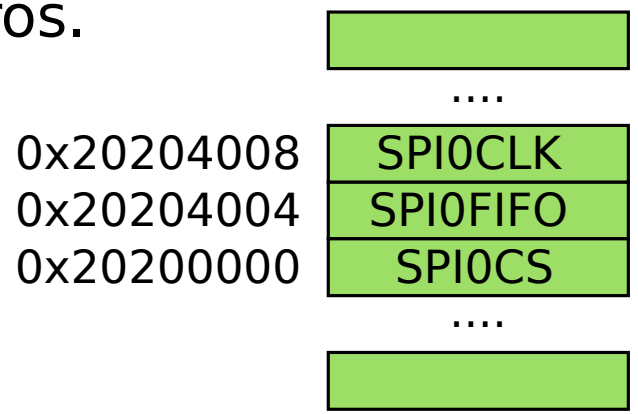
- Full duplex.
- Push-pull driver, a diferencia del open drain de algunos protocolos seriales, permitiendo trabajar a mayor velocidad.
- El protocolo no esta limitado a una cantidad fija de bits.
- No requiere interfaz ni elementos externo como resistencias.
- Pocas lineas comparada con el paralelo por ejemplo.
- Señales unidireccionales (permite Aislamiento galvánico).
- No limitado por el clock.
- Implementación simple de Software.

Desventajas

- Mayor requerimiento de pines que otros protocolos seriales.
- Requiere señales dedicadas para seleccionar un dispositivo.
- Sin control de flujo por parte del esclavo.
- Sin reconocimiento del esclavo, ni chequeo de errores (el master no puede conoce el resultado de una transmisión).
- Típicamente un master.
- No hay estándar formal, existiendo múltiple variaciones.
- Interrupciones deben ser implementadas de manera externa.
- No permite el agregado dinámico de nodos.

SPI en BCM2835

- Esta SoC posee 3 SPI master y 1 SPI esclavo.
- Aquí se presentará el SPI master port 0, accesible en la **GPIO** pins 11:9
- Los pines deben ser configurado con **GPFSEL** en el modo **ALT0**.
- Este puerto tiene asociado 3 registros.



SPI Registro SPI0CS

- Es el registro de control del SPI.
- Se utiliza para activar el SPI y establecer atributos como la polaridad del reloj.
- Todos inician en 0.
- Algunos campos (interrupciones por ej) no son presentados en la tabla pero pueden observarse en el datasheet.

Bit	Nombre	Función	Para 0	Para 1
16	DONE	Transferencia Realizada	Transf. en progreso	Transf. completada
7	TA	Transferencia Activada	SPI desactivo	SPI activo
3	CPOL	Polaridad del Reloj	Reloj inactivo en bajo	Reloj inactivo en alto
2	CPHA	Fase del Reloj	1era trans. del CLK a la mitad del bit de dato	1era trans. del CLK al comienzo del bit de dato

SPI Registro SPI0FIFO

- Este registro es escrito para transmitir un dato.
- Y leído para tomar el dato recibido.

SPI Registro SPI0CLK

- Configura la frecuencia del reloj.
- El valor de la frecuencia se obtiene al dividir 250MHz (reloj de periféricos) por la potencia de 2 almacenada en el registro.

SPI0CLK	Frecuencia del SPI
2	125000 kHz
4	62500 kHz
8	31250 kHz
16	15625 kHz
32	7812 kHz
64	3906 kHz
128	1953 kHz
256	976 kHz
512	488 kHz
1024	244 kHz
2048	122 kHz

Driver EasyPIO + SPI

```
#define BCM2835_PERI_BASE        0x20000000
#define SPI0_BASE                (BCM2835_PERI_BASE + 0x204000)
typedef struct
{
    unsigned CS       :2;
    unsigned CPHA     :1;
    unsigned CPOL     :1;
     ...
    unsigned TA       :1;
     ...
    unsigned DONE     :1;
     ...
}spi0csbits;
spi = (volatile int *)reg_map;

#define SPI0CSbits (* (volatile spi0csbits*) (spi + 0))
#define SPI0CS (* (volatile unsigned int *) (spi + 0))

#define SPI0FIFO (* (volatile unsigned int *) (spi + 1))
#define SPI0CLK (* (volatile unsigned int *) (spi + 2))
```

Obtenido de mmap

Driver EasyPIO + SPI

- La función de inicializar configura los pines de E/S del **SPI0** como **ALT0** (modo SPI)
- Configura la frecuencia con el parámetro **freq**.
- Y configura el SPI de acuerdo a la entrada **settings**

```
void spiInit(int freq, int settings) {
    pinMode(8, ALT0);     // CEOb
    pinMode(9, ALT0);     // MISO
    pinMode(10, ALT0);    // MOSI
    pinMode(11, ALT0);    // SCLK
    SPI0CLK = 250000000/freq; // Set SPI a la frecuencia determinada
    SPI0CS = settings;
    SPI0CSbits.TA = 1;    // Turn SPI on
}
```

Driver EasyPIO + SPI

- Esta función envía el dato pasado como parámetro y luego permanece en un bucle hasta que un dato es recibido.
- El dato recibido se verifica con el bit **DONE** de **SPI0CS**.

```
char spiSendReceive(char send){
    SPI0FIFO = send; // Envia un dato el esclavo
    while (!SPI0CSbits.DONE); // Espera hasta que el dato sea recibido
    return SPI0FIFO; // retorna el dato recibido
}
```

Driver EasyPIO + SPI

- Finalmente un ejemplo del uso.

```
#include "EasyPIO.h"
void main(void) {
    char received;
    pioInit(); // Inicializa el SPI
    spiInit(244000, 0); // 244 kHz de clk y configuración por defecto
    received = spiSendReceive('A'); // Envía una “A” y espera el result.
}
```

- Bibliografía

  Harris & Harris. Digital design and computer architecture: ARM edition. Elsevier, 2015. Capítulo 9.

¿ Preguntas ?