INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

DIVISIÓN DE INGENIERÍA MECATRÓNICA

Asignatura: Microcontroladores

Docente: Juan Merlín Chontal

Semestre: Séptimo **Grupo:** 711-A

Periodo Escolar: septiembre 2024 – diciembre 2024.

Unidad 4: Programación de periféricos del microcontrolador.

Tema 4.3.2 Configuración y programación como TWI (I2C)

Alumnos:

Raúl Marcial Arres

Jade Yael Chagala Jimenez

Fernando Quino Cortez

German Lael Chapol Toga

Alejandro Pava Catemaxca

San Andrés Tuxtla, Veracruz a 28 de Noviembre del 2024

INVESTIGACION

1. Introducción al protocolo SPI

El **SPI (Serial Peripheral Interface)** es un protocolo de comunicación síncrono, rápido y eficiente, ampliamente utilizado en sistemas embebidos. Se usa para conectar microcontroladores con periféricos como sensores, memorias, pantallas y otros dispositivos electrónicos.

Características principales de SPI

1. **Síncrono:** Utiliza un reloj para sincronizar la transferencia de datos.
2. **Maestro-Esclavo:** Un dispositivo actúa como maestro, generando el reloj y controlando la comunicación.
3. **Full-Duplex:** Permite la transmisión y recepción simultánea de datos.
4. **Rápido:** Permite velocidades de varios MHz dependiendo del hardware.
5. **Sin dirección:** Los datos se transfieren sin direcciones explícitas (a diferencia de I2C).

Líneas del bus SPI

1. **SCK (Serial Clock):** Línea de reloj, generada por el maestro.
2. **MOSI (Master Out, Slave In):** Línea para enviar datos del maestro al esclavo.
3. **MISO (Master In, Slave Out):** Línea para enviar datos del esclavo al maestro.
4. **SS/CS (Slave Select/Chip Select):** Línea para seleccionar un esclavo. Es activa en nivel bajo.

---

2. Configuración básica de SPI

Antes de programar un microcontrolador, es fundamental configurar varios parámetros de SPI:

2.1 Modo maestro o esclavo

- **Maestro:** Controla el reloj (SCK) y selecciona a los esclavos.
- **Esclavo:** Responde a las órdenes del maestro.

2.2 Frecuencia del reloj

El maestro define la frecuencia del reloj SPI. Los valores típicos están en el rango de kHz a MHz, dependiendo del hardware.

2.3 Polaridad y Fase (Modos SPI)

La configuración de la señal de reloj depende de dos parámetros: **CPOL** (polaridad) y **CPHA** (fase).

- **CPOL:** Determina el nivel inactivo del reloj:
  - 0: El reloj está en bajo cuando está inactivo.
  - 1: El reloj está en alto cuando está inactivo.
- **CPHA:** Determina en qué flanco del reloj se capturan los datos:
  - 0: Captura en el primer flanco (subida si CPOL=0, bajada si CPOL=1).
  - 1: Captura en el segundo flanco.

2.4 Dirección de bits

- **MSB First (Más significativo primero):** El bit más significativo se envía primero.
- **LSB First (Menos significativo primero):** El bit menos significativo se envía primero.

2.5 Tamaño de datos

El tamaño de palabra (bits por transferencia) puede ser configurado. Los valores típicos son 8 o 16 bits.

3. Conexión física del bus SPI

El bus SPI requiere que las líneas del maestro estén conectadas a las del esclavo, según la tabla:

Maestro	Esclavo
SCK	SCK
MOSI	MOSI
MISO	MISO
SS	CS
GND	GND

Notas importantes:

- Si hay varios esclavos, cada uno debe tener su propia línea de **CS**.
- Las resistencias pull-up/pull-down pueden ser necesarias dependiendo del diseño del circuito.

4. Programación de un microcontrolador con SPI

El proceso varía según la plataforma. A continuación, se explican configuraciones y ejemplos para varios microcontroladores.

4.1 Configuración en STM32 (HAL)

La librería HAL de STM32 simplifica la configuración de SPI.

Configuración del periférico SPI

El periférico SPI se inicializa configurando un SPI_HandleTypeDef.

```
SPI_HandleTypeDef hspi1;

void MX_SPI1_Init(void) {
    hspi1.Instance = SPI1;
    hspi1.Init.Mode = SPI_MODE_MASTER;                // Modo maestro
    hspi1.Init.Direction = SPI_DIRECTION_2LINES;      // Full-Duplex
    hspi1.Init.DataSize = SPI_DATASIZE_8BIT;          // Tamaño de dato: 8 bits
    hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;        // CPOL = 0
    hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;            // CPHA = 0
    hspi1.Init.NSS = SPI_NSS_SOFT;                    // CS manual
    hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;
    hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;           // MSB primero
    hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
    hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;

    if (HAL_SPI_Init(&hspi1) != HAL_OK) {
        // Error en inicialización
        Error_Handler();
    }
}
```

Transmisión y recepción

- **Enviar datos:**

```c
uint8_t data = 0xA5;
HAL_SPI_Transmit(&hspi1, &data, 1, HAL_MAX_DELAY);
```

- **Recibir datos:**

```c
uint8_t received_data;
HAL_SPI_Receive(&hspi1, &received_data, 1, HAL_MAX_DELAY);
```

- **Transmisión y recepción simultánea (Full-Duplex):**

```c
uint8_t tx_data = 0xA5;
uint8_t rx_data;
HAL_SPI_TransmitReceive(&hspi1, &tx_data, &rx_data, 1, HAL_MAX_DELAY);
```

EXPOSICIÓN

MANEJO DE SOFTWARE

4.2 Configuración en Arduino

El IDE de Arduino incluye la biblioteca SPI.h para configurar y usar el protocolo SPI.

Configuración del maestro:

```cpp
#include <SPI.h>

void setup() {
    SPI.begin();                          // Inicia SPI como maestro
    SPI.setClockDivider(SPI_CLOCK_DIV16); // Divide la frecuencia del reloj
    SPI.setDataMode(SPI_MODE0);           // CPOL=0, CPHA=0
    SPI.setBitOrder(MSBFIRST);            // MSB primero
}

void loop() {
    digitalWrite(SS, LOW);                // Seleccionar esclavo
    SPI.transfer(0xA5);                   // Enviar datos
    digitalWrite(SS, HIGH);               // Deseleccionar esclavo
}
```

Configuración del esclavo:

```cpp
#include <SPI.h>

volatile byte receivedData;

void setup() {
    pinMode(MISO, OUTPUT);              // Configura MISO como salida
    SPI.setDataMode(SPI_MODE0);         // CPOL=0, CPHA=0
    SPI.attachInterrupt();              // Activa interrupción SPI
    SPI.begin();
}

ISR(SPI_STC_vect) {                     // Interrupción al recibir datos
    receivedData = SPDR;                // Leer dato recibido
}
```

4.3 Configuración en ESP32

ESP32 utiliza la biblioteca SPI para Arduino o funciones específicas de IDF.

---

5. Depuración y pruebas

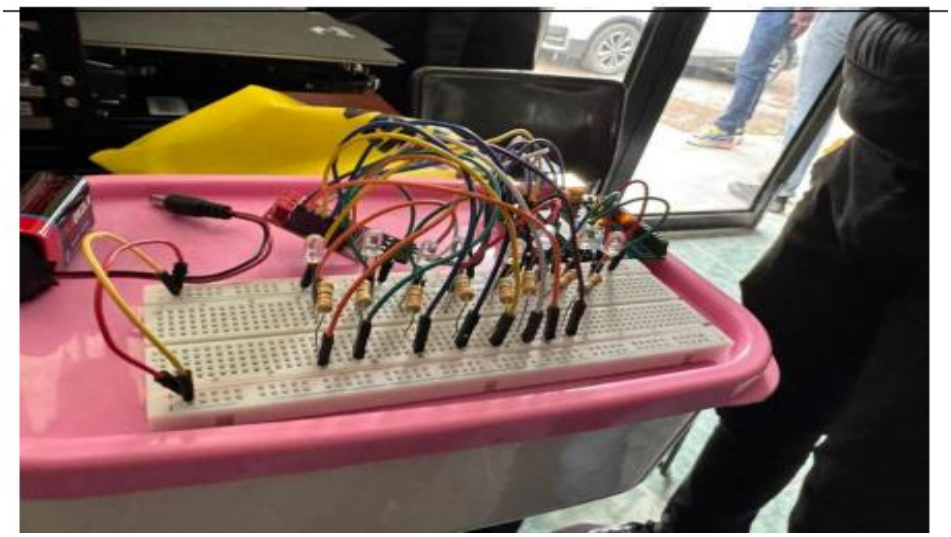
Herramientas recomendadas:

1. **Analizador lógico:** Verifica la señal de las líneas SPI.
2. **Osciloscopio:** Mide la sincronización entre SCK y los datos (MOSI/MISO).
3. **Simuladores:** Usa software como Proteus para probar el diseño.

Errores comunes y soluciones

- **Datos corruptos:** Revisa la configuración de CPOL y CPHA.
- **Fallo en la selección del esclavo:** Asegúrate de manejar correctamente el pin SS.
- **Problemas de sincronización:** Verifica la frecuencia del reloj.

IMPLEMENTACIÓN EN FISICO DEL CIRCUITO




Target