Communication SPI sur ESP32

Houssem-eddine LAHMER

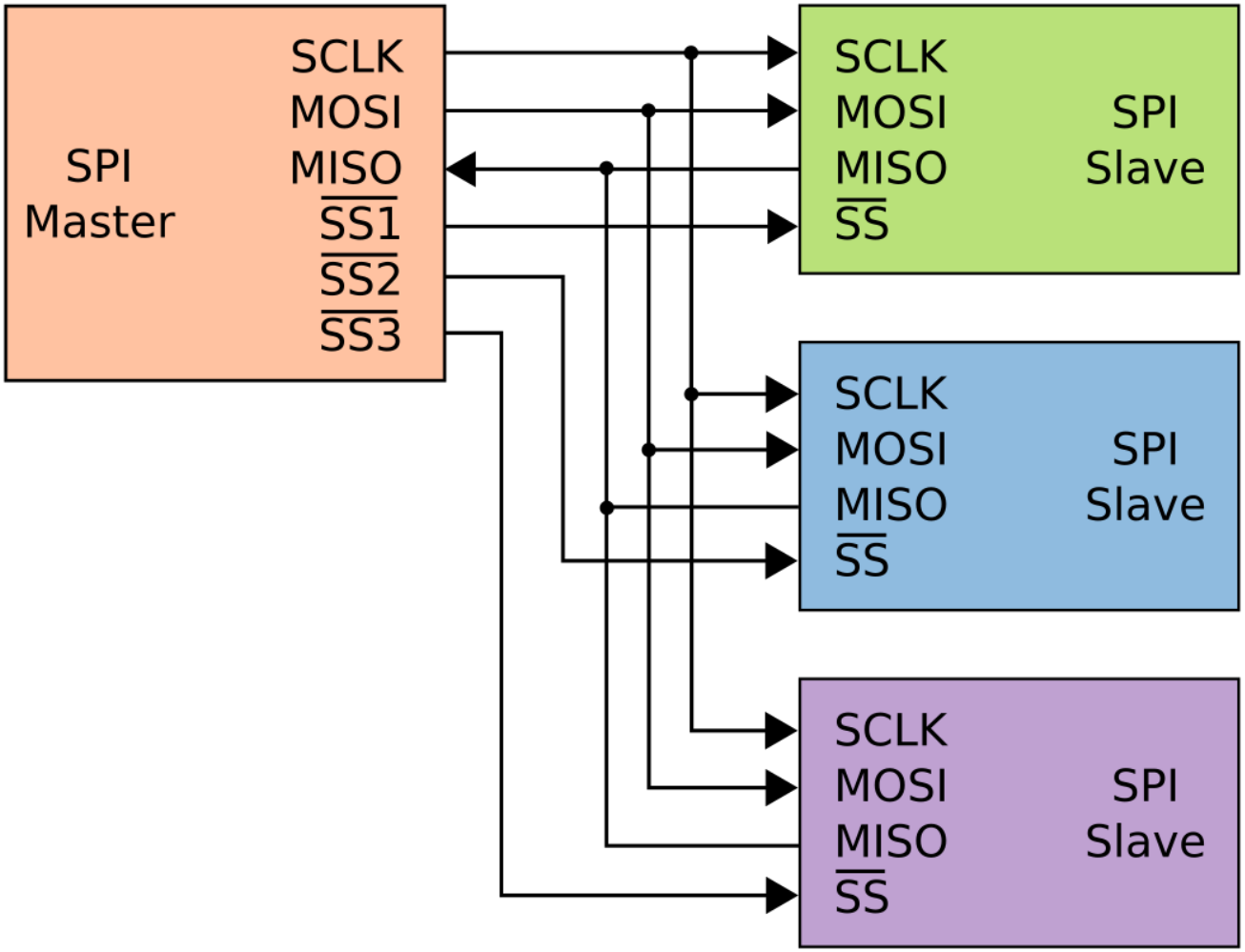
4 mai 2025

Plan du cours

Qu'est-ce que le SPI ?

- SPI (Serial Peripheral Interface)
- Protocole série synchrone maître-esclave
- Utilise quatre lignes : MOSI, MISO, SCK, CS
- Très haute vitesse, large débit de données

Qu'est-ce que le SPI ?



Brochage SPI – Signaux et connexions

- **MOSI** – ligne de données du maître vers l'esclave.
- **MISO** – ligne de données de l'esclave vers le maître.
- **SCLK** – horloge générée par le maître pour synchroniser les bits.
- **SS / CS** – signal de sélection de l'esclave ; actif bas (LOW) pour engager la communication.

Remarques

- Plusieurs esclaves : chaque esclave nécessite une ligne SS/CS dédiée.
- Niveau logique : veillez à adapter les niveaux (3.3V vs 5V) via un convertisseur de niveau si nécessaire.
- Modes SPI : quatre modes (CPOL, CPHA) définissent le point d'échantillonnage des données.

Avantages et inconvénients du SPI

Avantages

- Débit élevé (MHz)
- Full duplex
- Simple côté logiciel

Inconvénients

- 4 lignes minimum (+ broche CS par esclave)
- Pas d'adressage natif
- Pas de détection d'erreur intégrée

Contrôleurs SPI sur ESP32

- ESP32, ESP32-S2, ESP32-S3 : 4 contrôleurs SPI (SPI0, SPI1, HSPI, VSPI)
- ESP32-C3 : 3 contrôleurs (SPI0, HSPI, VSPI)
- SPI0 et SPI1 réservés pour la flash interne
- HSPI et VSPI disponibles pour l'utilisateur

Brochages par défaut

Bus	MOSI	MISO	SCK	CS
HSPI	13	12	14	15
VSPI	23	19	18	5

Table – Affectations par défaut SPI (ESP32)

Réassignation possible via `SPIClass.begin`

Initialisation basique en Arduino

```
1  #include <SPI.h>
2
3  void setup() {
4    SPI.begin();          // utilise HSPI par d faut
5    // SPI.begin(SCK, MISO, MOSI, SS);
6  }
7
8  void loop() {
9    // ...
10 }
```

Définition manuelle d'un bus SPI

```
1  #include <SPI.h>
2
3  #define HSPI_CLK 14
4  #define HSPI_MISO 12
5  #define HSPI_MOSI 13
6  #define HSPI_SS   15
7
8  SPIClass hspi = SPIClass(HSPI);
9
10 void setup() {
11   hspi.begin(HSPI_CLK, HSPI_MISO, HSPI_MOSI, HSPI_SS);
12 }
13
14 void loop() {
15   // ...
16 }
```

Envoyer et recevoir un octet

```
byte dataToSend = 0xAA;
byte received;

// S lection du p riph rique
digitalWrite(HSPI_SS, LOW);
// Envoi et r ception simultan s
received = hspi.transfer(dataToSend);
// Fin de communication
digitalWrite(HSPI_SS, HIGH);
```

Présentation de Wokwi pour SPI

- Simulateur en ligne ESP32 périphériques SPI
- Visualisation des signaux MOSI/MISO/SCK/CS
- Test sans matériel physique

Exemple Wokwi : lecture d'une mémoire SPI

```
1  #include <SPI.h>
2  #define CS_PIN 15
3
4  void setup() {
5    Serial.begin(115200);
6    SPI.begin();
7    pinMode(CS_PIN, OUTPUT);
8  }
9
10 void loop() {
11   digitalWrite(CS_PIN, LOW);
12   byte id = SPI.transfer(0x9F); // commande Read ID
13   digitalWrite(CS_PIN, HIGH);
14   Serial.print("Device ID: 0x");
15   Serial.println(id, HEX);
16   delay(1000);
17 }
```

Exercice 1 : SD Card SPI

- Connecter un module SD (MOSI=13, MISO=12, SCK=14, CS=15)
- Lire la liste des fichiers via la bibliothèque SD
- Afficher sur le moniteur série

Exercice 2 : Écran TFT SPI

- Ajouter un écran TFT SPI
- Initialiser l'affichage et dessiner un texte
- Gérer plusieurs périphériques CS distincts

Résumé

- Protocole SPI et ses caractéristiques
- Configuration des bus HSPI/VSPI sur ESP32
- Transferts d'octets et simulations Wokwi
- Applications pratiques (SD, TFT)

Questions ?

Merci pour votre attention !