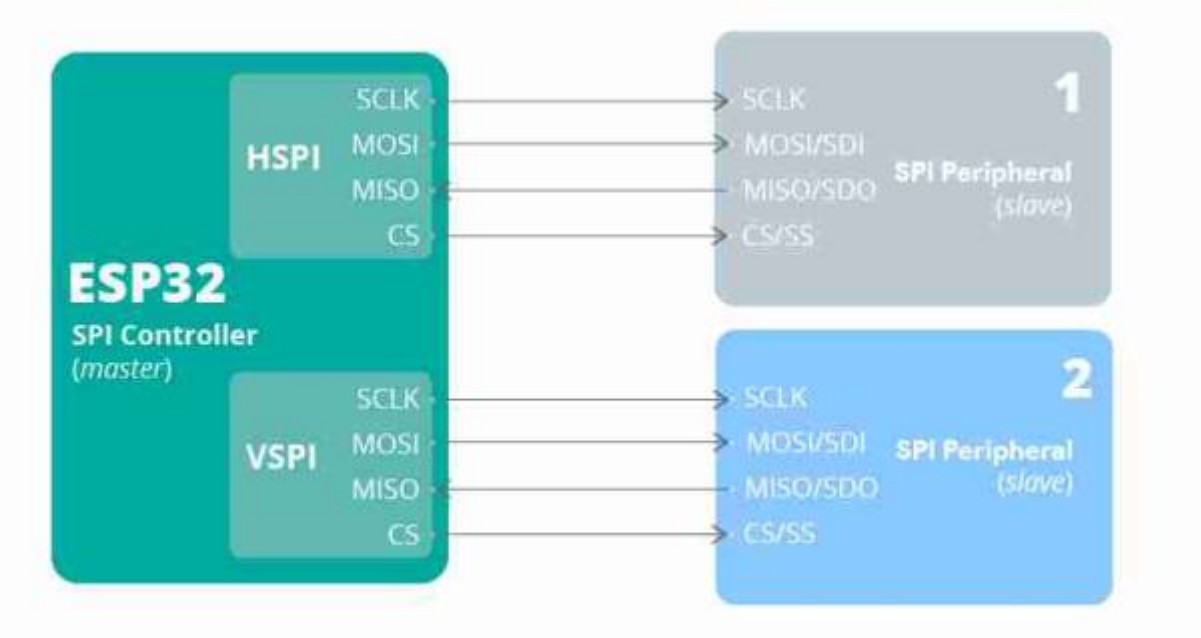
Multiples Peripheriques SPI sur ESP32

ESP32 utilisant deux interfaces de bus SPI (utilisez simultanément HSPI et VSPI)

Pour communiquer simultanément avec plusieurs périphériques SPI, vous pouvez utiliser les deux bus SPI ESP32 (HSPI et VSPI). Vous pouvez utiliser les broches HSPI et VSPI par défaut ou utiliser des broches personnalisées.



En bref, pour utiliser HSPI et VSPI simultanément, il vous suffit de le faire.

1) Tout d'abord, assurez-vous d'inclure la bibliothèque SPI dans votre code.

```
#include <SPI.h>
```

2) Initialiser deux Classe SPI objets avec des noms différents, un sur le bus HSPI et un autre sur le bus VSPI. Par exemple:

```
vspi = new SPIClass(VSPI);
hspi = new SPIClass(HSPI);
```

3) Appeler le beginr() méthode sur ces objets.

```
vspi.begin();
hspi.begin();
```

Vous pouvez transmettre des broches personnalisées au begin() méthode si nécessaire.

```
vspi.begin(VSPI_CLK, VSPI_MISO, VSPI_MOSI, VSPI_SS);
hspi.begin(HSPI_CLK, HSPI_MISO, HSPI_MOSI, HSPI_SS);
```

4) Enfin, vous devez également définir les broches SS comme sorties. Par exemple:

```
  pinMode(VSPI_SS, OUTPUT);
  pinMode(HSPI_SS, OUTPUT);
```

Ensuite, utilisez les commandes habituelles pour interagir avec les appareils SPI, que vous utilisiez une bibliothèque de capteurs ou les méthodes de la bibliothèque SPI.

Vous pouvez trouver un exemple d'utilisation de plusieurs bus SPI sur le Bibliothèque SPI arduino-esp32. Voir l'exemple ci-dessous :

exemple_Multiple_SPI_ESP32.ino

```
/* L'ESP32 dispose de quatre bus SPi, mais pour le moment, seuls deux d'entre eux
 * ils sont disponibles à utiliser, HSPI et VSPI. Simplement en utilisant l'API SPI
 * comme illustré dans les exemples Arduino utilisera VSPI, laissant HSPI inutilisé.
 *
 * Cependant, si nous initialisons simplement deux instances de la classe SPI pour les deux
 * de ces bus, les deux peuvent être utilisés. Cependant, lorsque vous les utilisez simplement, l'Arduino
 * way only sera effectivement sorti à la fois.
 *
 * La capture de l'analyseur logique se trouve dans le même dossier que cet exemple
 * "multiple_bus_output.png"
 *
 * créé le 30/04/2018 par Alistair Symonds
 */
#comprendre


// Définir ALTERNATE_PINS pour utiliser des broches GPIO non standard pour le bus SPI

#ifdef ALTERNATE_PINS
  #define VSPI_MISO 2
  #define VSPI_MOSI 4
  #define VSPI_SCLK 0
  #define VSPI_SS 33

  #define HSPI_MISO 26
  #define HSPI_MOSI 27
  #define HSPI_SCLK 25
  #define HSPI_SS 32
#autre
  #define VSPI_MISO MISO
  #define VSPI_MOSI MOSI
```

```
  #define VSPI_SCLK SCK
  #define VSPI_SS SS

  #define HSPI_MISO 12
  #define HSPI_MOSI 13
  #define HSPI_SCLK 14
  #define HSPI_SS 15
#fin si

#si CONFIG_IDF_TARGET_ESP32S2 || CONFIG_IDF_TARGET_ESP32S3
#définir VSPI FSPI
#fin si

statique const int spiClk = 1000000 ; // 1MHz

//pointeurs non initialisés vers des objets SPI
SPIClass * vspi = NULL ;
SPIClass * hspi = NULL ;

void setup() {
  //initialise deux instances de la SPIClass attachées respectivement à
VSPI et HSPI
  vspi = nouvelle SPIClass(VSPI);
  hspi = nouvelle SPIClass(HSPI);

  //horloge miso mosi ss

#ifndef ALTERNATE_PINS
  //initialise vspi avec les broches par défaut
  //SCLK = 18, MISO = 19, MOSI = 23, SS = 5
  vspi->begin();
#autre
  //routez alternativement à travers les broches GPIO de votre choix
  vspi->begin(VSPI_SCLK, VSPI_MISO, VSPI_MOSI, VSPI_SS); //SCLK, MISO,
MOSI, SS
#fin si

#ifndef ALTERNATE_PINS
  //initialise hspi avec les broches par défaut
  //SCLK = 14, MISO = 12, MOSI = 13, SS = 15
  hspi->begin();
#autre
  // alternativement route via les broches GPIO
  hspi->begin(HSPI_SCLK, HSPI_MISO, HSPI_MOSI, HSPI_SS); //SCLK, MISO,
MOSI, SS
#fin si

  // configurer les broches de sélection esclave en tant que sorties en
tant qu'API Arduino
  // ne gère pas automatiquement l'abaissement du SS
  pinMode(vspi->pinSS(), SORTIE); //VSPISS
```

```
  pinMode(hspi->pinSS(), SORTIE); //HSPISS

}

// la fonction de boucle s'exécute encore et encore jusqu'à la mise hors tension ou la réinitialisation
boucle vide() {
  //utiliser les bus SPI
  spiCommand(vspi, 0b01010101); // données indésirables pour illustrer l'utilisation
  spiCommand(hspi, 0b11001100);
  retard(100);
}

void spiCommand(SPIClass *spi, byte data) {
  // utilisez-le comme vous le feriez avec l'API SPI arduino habituelle
  spi->beginTransaction(SPISettings(spiClk, MSBFIRST, SPI_MODE0));
  digitalWrite(spi->pinSS(), LOW); // tirez SS lentement pour préparer l'autre extrémité pour le transfert
  spi->transfert(données);
  digitalWrite(spi->pinSS(), HIGH); // tirez ss vers le haut pour signifier la fin du transfert de données
  spi->endTransaction();
}
```