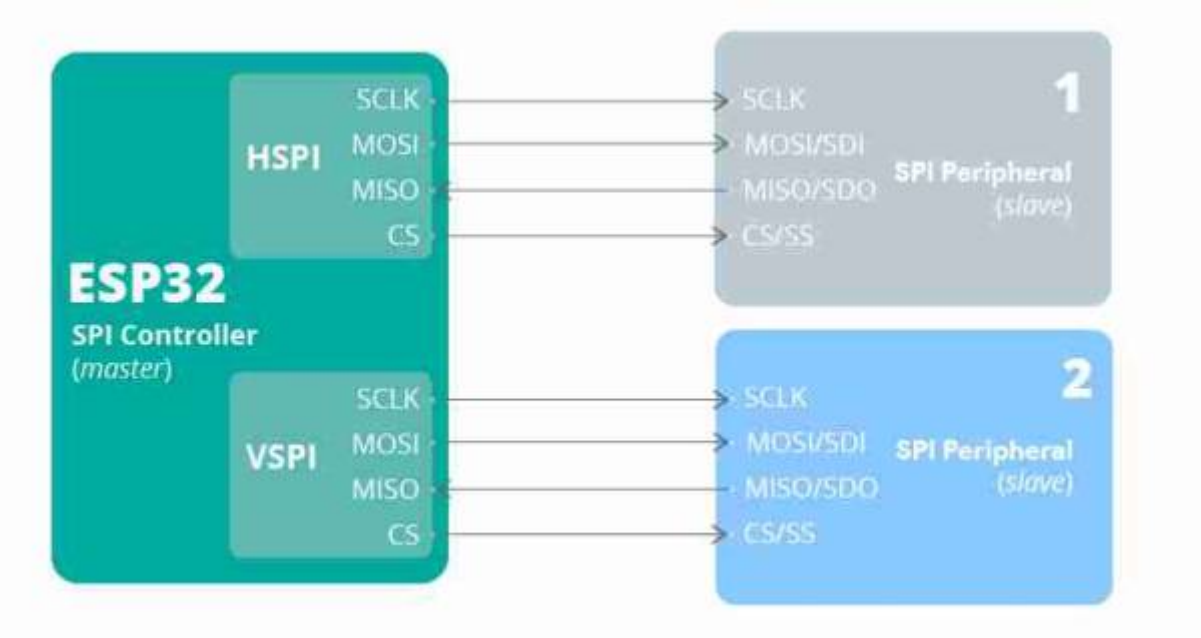
Multiples Peripheriques SPI sur ESP32

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 arduino-esp32 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 la Bibliothèque SPI arduino-esp32. Voir l'exemple ci-dessous :

exemple_Multiple_SPI_ESP32.ino

```
/* The ESP32 has four SPi buses, however as of right now only two of
 * them are available to use, HSPI and VSPI. Simply using the SPI API
 * as illustrated in Arduino examples will use VSPI, leaving HSPI
unused.
 *
 * However if we simply intialise two instance of the SPI class for
both
 * of these buses both can be used. However when just using these the
Arduino
 * way only will actually be outputting at a time.
 *
 * Logic analyser capture is in the same folder as this example as
 * "multiple_bus_output.png"
 *
 * created 30/04/2018 by Alistair Symonds
 */
#include <SPI.h>

// Define ALTERNATE_PINS to use non-standard GPIO pins for SPI bus

#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
#else
  #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
#endif

#if CONFIG_IDF_TARGET_ESP32S2 || CONFIG_IDF_TARGET_ESP32S3
#define VSPI FSPI
#endif

static const int spiClk = 1000000; // 1 MHz

//uninitalised pointers to SPI objects
SPIClass * vspi = NULL;
SPIClass * hspi = NULL;

void setup() {
  //initialise two instances of the SPIClass attached to VSPI and HSPI
respectively
  vspi = new SPIClass(VSPI);
  hspi = new SPIClass(HSPI);

  //clock miso mosi ss

#ifndef ALTERNATE_PINS
  //initialise vspi with default pins
  //SCLK = 18, MISO = 19, MOSI = 23, SS = 5
  vspi->begin();
#else
  //alternatively route through GPIO pins of your choice
  vspi->begin(VSPI_SCLK, VSPI_MISO, VSPI_MOSI, VSPI_SS); //SCLK, MISO,
MOSI, SS
#endif

#ifndef ALTERNATE_PINS
  //initialise hspi with default pins
  //SCLK = 14, MISO = 12, MOSI = 13, SS = 15
  hspi->begin();
#else
  //alternatively route through GPIO pins
  hspi->begin(HSPI_SCLK, HSPI_MISO, HSPI_MOSI, HSPI_SS); //SCLK, MISO,
MOSI, SS
#endif

  //set up slave select pins as outputs as the Arduino API
  //doesn't handle automatically pulling SS low
  pinMode(vspi->pinSS(), OUTPUT); //VSPI SS
  pinMode(hspi->pinSS(), OUTPUT); //HSPI SS

}
```

```
// the loop function runs over and over again until power down or reset
void loop() {
  //use the SPI buses
  spiCommand(vspi, 0b01010101); // junk data to illustrate usage
  spiCommand(hspi, 0b11001100);
  delay(100);
}

void spiCommand(SPIClass *spi, byte data) {
  //use it as you would the regular arduino SPI API
  spi->beginTransaction(SPISettings(spiClk, MSBFIRST, SPI_MODE0));
  digitalWrite(spi->pinSS(), LOW); //pull SS slow to prep other end for
transfer
  spi->transfer(data);
  digitalWrite(spi->pinSS(), HIGH); //pull ss high to signify end of
data transfer
  spi->endTransaction();
}
```