# Documentation SpiDev

(Brouillon du 2020-07-23)

## Table des matières


1. Description....1
2. Installation de SpiDev....2
3. Pilotes SPI du noyau....2
4. Autorisation....3
5. Exemples....3
    - Sortie simple....3
    - Inverser les bits....4
    - Imprimer les octets....4
6. Constructeurs de la classe....4
    - spidev()....4
    - spidev(bus, cs)....4
7. Attributs de la classe....5
    - bits_per_word....5
    - cshigh....5
    - loop....5
    - lsbfirst....6
    - max_vitesse_hz....6
    - mode....6
    - threewire....7
8. Méthodes de la classe....7
    - close....7
    - open....7
    - readbytes....7
    - writebytes....8
    - writebytes2....8
    - xfer....9
    - xfer2....10
    - xfer3....10
9. Taille maximale du tampon SPI....11


## 1. Description

Ce module définit un type d'objet qui autorise les transactions SPI (Serial Peripheral Interface) sur les hôtes exécutant le noyau Linux. Le noyau hôte doit prendre en charge le protocole SPI et l'interface de périphérique SPI. Tous ces éléments peuvent être intégrés au noyau ou chargés dynamiquement à partir de modules.

Ce document est écrit pour tout modèle Raspberry Pi exécutant des versions récentes de Raspbian Buster et la version actuelle (juillet 2020) du système d'exploitation maintenant nommé Raspberry Pi OS. Cela pourrait s'appliquer à d'autres systèmes utilisant différentes distributions Linux avec de légères modifications.

Voir https://sigmdel.ca/michel/ha/rpi/spi_on_pi_fr.html pour plus de détails.

## 2. Installation de SpiDev

L'installation du module spidev dans un environnement virtuel Python3, se fait avec une simple commande `pip`:

```
(spipy) pi@raspberrypi:~ $ pip install spidev
```

Le module peut être installé dans l'installation par défaut de Python3 de la façon suivante :

```
pi@raspberrypi:~ $ sudo apt install python3-spidev
```

Information au sujet du module depuis `.../site-packages/spidev-3.4.dist-info/METADATA`

```
Nom : spidev
Version : 3.4
Sommaire : liaisons Python pour l'accès à l'interface spidev de Linux
Page d'accueil : http://github.com/doceme/py-spidev
```

## 3. Pilotes SPI du noyau

Par défaut, les pilotes du noyau SPI ne sont pas chargés par le système d'exploitation. Cela peut être fait de manière ponctuelle avec l'utilitaire `dtparam` :

```
pi@raspberrypi:~ $ sudo dtparam spi=on
```

Cela créera deux périphériques SPI : `/dev/spidev0.0` et `/dev/spidev0.1`.

Pour ajouter automatiquement le pilote à chaque démarrage, modifiez le fichier `/boot/config.txt` ou utilisez l'utilitaire `raspi-config`.

| Signal | Broche E/S | Broche physique |
|---|---|---|
| SPI0_MOSI | 10 | 19 |
| SPI0_MISO | 9 | 21 |
| SPI0_SCLK | 11 | 23 |
| SPI0_CEO_N | 8 | 24 |

| Signal | Broche E/S | Broche physique |
|---|---|---|
| SPI0_CE1_N | 7 | 26 |

*Connexions SPI0*

Les modèles du Raspberry Pi avec un connecteur d'entré/sortie de 40 broches ont un deuxième bus SPI qui peut être activé avec l'utilitaire `dtoverlay` :

```
pi@raspberrypi:~ $ sudo dtoverlay spi1-3cs
```

Cela créera trois périphériques : `/dev/spidev1.0` , `/dev/spidev1.1` and `/dev/spidev1.2`.

| Signal | Broche E/S | Broche physique |
|---|---|---|
| SPI1_MOSI | 20 | 38 |
| SPI1_MISO | 19 | 35 |
| SPI1_SCLK | 21 | 40 |
| SPI1_CEO_N | 18 | 22 |
| SPI1_CE1_N | 17 | 11 |
| SPI1_CE2_N | 16 | 36 |

*SPI1 connections*

D'autres modules sont disponibles avec moins de signaux de sélection de puce ( `spi1-2cs` and `spi1-cs`).

Le modèle Raspberry Pi 4 et le Compute Module ont encore plus de bus SPI.

## 4. Autorisation

Étant donné que l'interface de périphérique SPI est utilisée pour lire et écrire, les utilisateurs d'un nœud de périphérique SPI doivent disposer des autorisations `root`. Cependant, dans Raspbian Buster et Raspberry Pi OS, les membres du groupe `spi` ont accès à l'interface et l'utilisateur par défaut est membre de ce groupe. Par conséquent, il ne sera pas nécessaire d'utiliser le préfixe `sudo` lors de l'exécution d'un script Python qui importe `spidev`.

## 5. Exemples

### Sortie simple

Dans cet exemple le périphérique SPI est ouvert pour y écrire un octet (0x3A) tous les dixièmes de seconde jusqu'à ce que la combinaison de touches Ctrl+C soit saisie.

```
import spidev
import time

spi = spidev.SpiDev(0, 1)     # create spi object connecting to /dev/spidev0.1
spi.max_vitesse_hz = 250000   # set vitesse to 250 Khz

try:
   while True:                # endless loop, press Ctrl+C to exit
      spi.writebytes([0x3A])  # write one byte
      time.sleep(0.1)         # sleep for 0.1 seconds

finally:
   spi.close()                # always close the port before exit
```

### Inverser les bits

Cette fonction inversera l'ordre des bits en un octet si nécessaire.

```
def ReverseBits(byte):
   byte = ((byte & 0xF0) >> 4) | ((byte & 0x0F) << 4)
   byte = ((byte & 0xCC) >> 2) | ((byte & 0x33) << 2)
   byte = ((byte & 0xAA) >> 1) | ((byte & 0x55) << 1)
   return byte
```

### Imprimer les octets

Cette fonction imprime les valeurs d’un tableau d’octets en hexadécimal qui est plus facile à lire et qui est souvent utile lors du débogage.

```
def BytesToHex(Bytes):
   return ''.join(["0x%02X " % x for x in Bytes]).strip()
```

## 6. Constructeurs de la classe

### spidev()

Syntaxe : `myspi = spidev()`

Remarque : ceci est le constructeur de base qui ne connecte pas l'objet à un périphérique SPI du système. Il sera nécessaire d'utiliser la méthode `open()` pour associer l'objet à un périphériquel.

### spidev(bus, cs)

Syntaxe : `myspi = spidev(bus, cs)`

Remarque : il s'agit d'un constructeur surchargé qui crée l'objet et le connecte à un périphérique SPI du système. Donc

```
myspi = spidev(bus, cs);
```

et

```
myspi = spidev();
myspi.open(bus, cs);
```

sont équivalents.

Sur le Raspberry Pi, l'attribut `max_vitesse_hz` doit être défini sur une valeur raisonnable (inférieure ou égale à 32 MHz) après que l'objet `spidev` est connecté à un périphérique SPI du système soit avec le constructeur `spidev (bus, cs)` soit avec la méthode `open(bus, cs)`.

## 7. Attributs de la classe

### bits_per_word

Description : nombre de bits par mot.

Valeur par défaut : 8

Valeurs possibles : 8 .. 16

Restriction : en lecture seulement sur le Raspberry Pi.

### cshigh

Description : vrai signifie que le dispositif est sélectionné avec un niveau haut de logique, sinon un niveau bas de logique sélectionne le dispositif.

Valeur par défaut : faux (un niveau bas de logique est le signal de sélection).

### loop

Description : si vrai, la « configuration de boucle de retour » est activée.

Valeur par défaut : faux.

Restriction : en lecture seulement sur le Raspberry Pi.

Remarque : Sers à des fins de test; tout ce qui est reçu sera renvoyé en écho (cela signifie probablement que tout ce qui est reçu sur la ligne MISO est renvoyé à la ligne MOSI. Une boucle de retour sur un seul port SPI peut être effectuée en connectant ses lignes MISO et MOSI.

### lsbfirst

Description : Propriété booléenne qui spécifie si un mot est transmis avec le bit le moins significatif en premier ou non (ce qui, bien sûr, signifie qu'il serait transmis le bit le plus significatif en premier).

Valeur par défaut : faux (transmission avec le bit plus significatif en premier)

Restriction : en lecture seulement sur le Raspberry Pi.

Remarques : La valeur par défaut semble dépendre du boutisme du système. Puisque le Raspberry Pi ne peut envoyer que le bit le plus significatif en premier, il sera nécessaire de convertir le ou les octets manuellement (voir l'exemple de code pour une telle fonction) si un périphérique esclave s'attend à recevoir chaque octet avec le bit moins significatif en premier et s'il envoie ses propres données de la même manière.

## max_vitesse_hz

Description : vitesse maximale du bus SPI en Hz.

Valeur par défaut : 125000000

Valeurs possibles : 3800Hz à 32 MHz

Remarques : **La valeur par défaut de 125 MHz n'est pas viable sur un Raspberry Pi et elle doit être modifiée pour une valeur raisonnable.**

Il y a un débat sur les valeurs de vitesse autorisées, d'aucuns insistant sur le fait que la vitesse doit être une puissance de 2, d'autres soutenant qu'elle peut être un multiple de 2. Les tests confirment au moins partiellement que cette dernière position est correcte. Il était possible de régler la vitesse à 3800 Hz, ce qui semble être une limite inférieure, et à 4800 Hz. Aucune de ces valeurs n'est une puissance de 2.

## mode

Description: le mode SPI qui spécifie la polarité d'horloge et sa phase.

Valeur par défaut : 0

Valeurs possibles :

| Mode | Polarity de l'horloge (CPOL) | Phase de l'horloge phase (CPHA) |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |

### threewire

Description: si vrai, l'interface SPI est réglée en mode trois fils, ce qui signifie que les signaux MISO et MOSI partagent une seule ligne. En conséquence, le protocole devient semi-duplex.

Valeur par défaut : faux

Remarque : attribut non vérifié!

## 8. Méthodes de la classe

### close

Description : Déconnecte l'objet du périphérique SPI du système.

Syntaxe : `close()`

Renvoie : rien

### open

Description : Connecte l'object au périphérique SPI du système spécifié.

Syntaxe : `open(bus, cs)`

Renvoie : rien

Exemple :

```
import spidev

myspi = spidev.SpiDev()
myspi.open(1,2)
myspi.max_speed_hz = 1000000
```

Ce début de script ouvre le périphérique `/dev/spidev1.2` et fixe l'horloge SPI à 1 MHz.

Remarque : Bien sûr, le pilote noyau du périphérique doit être chargé.

### readbytes

Description : lit une liste de valeurs du périphérique SPI.

Syntaxe : `readbytes(len)`

Renvoie : [valeurs]

Paramètre :

`len` : nombre d'octets à lire et placer dans la liste

Exemple :

```
import spidev

myspi = spidev.SpiDev(0, 0)
myspi.max_speed_hz = 1000000
rx = myspi.readbytes(8)
print("rx", rx)
```

## writebytes

Description : écrit une liste d'octets sur le périphérique SPI.

Syntaxe : `writebytes([valeurs])`

Renvoie : rien

Paramètre :

`[valeurs]` : une liste d’octets

Exemple :

```
import spidev

myspi = spidev.SpiDev(0, 1)
myspi.max_speed_hz = 2000000
tx = [0, 1, 255]
rx = myspi.writebytes(tx)
print(« tx », tx)
print(« rx », rx)
```

Remarques : échoue si la liste de valeurs est plus grande que la taille maximale du tampon qui est de 4096 octets. Utilisez `writebytes2` avec les listes plus longues.

Il ne s'agit pas d'une transaction SPI et toutes les données lues à partir du signal MISO sont supprimées et la liste `valeurs` n'est pas modifiée.

## writebytes2

Description : écrit une liste d'octets de n'importe quelle taille sur le périphérique SPI.

Syntaxe : `writebytes2([valeurs])`

Renvoie : rien

Paramètre :

`[valeurs]` : une liste d’octets

Remarques : contrairement à `writebytes`, qui échoue si la liste de valeurs est supérieure à la taille maximale de la mémoire tampon, `writebytes2` gère les longues listes en les divisant en blocs envoyés un par un. La taille de chaque bloc est égale à la taille maximale du tampon sauf peut-être pour le dernier qui pourrait être plus petit. Le signal de sélection de puce est désactivé entre chaque bloc.

De plus, writebytes2 comprend le protocole de tampon et il peut ainsi accepter des tableaux d'octets numpy par exemple sans avoir besoin de les convertir d'abord avec `tolist()`. Cela offre de bien meilleures performances lors du transfert de trames vers des écrans SPI par exemple.

Il ne s'agit pas d'une transaction SPI, toutes les données lues à partir du signal MISO sont supprimées et la liste `valeurs` n'est pas modifiée.

### xfer

Description : effectue une transaction SPI. Une liste d'octets est écrite sur le périphérique SPI et alors que chaque octet de cette liste est envoyé, il est remplacé par les données lues simultanément depuis le dispositif esclave SPI sur la ligne MISO.

Syntaxe : `rcvd = xfer([valeurs])`

`rcvd = xfer([valeurs], vitesse)`

`rcvd = xfer([valeurs], vitesse, délai)`

`rcvd = xfer([valeurs], vitesse, délai, bits)`

Renvoie : [valeurs]

Paramètre :

`[valeurs]` : une liste d'octets

`vitesse` : fréquence de l'horloge SPI en Hz. Si non spécifiée, l'horloge SPI est définie sur l'attribut `maximum_speed_hz`. Si la vitesse est spécifiée, elle sera utilisée pour cette transaction unique et ne modifiera pas la valeur de l'attribut `maximum_speed_hz` de l'objet.

`Délai` : la ligne de sélection de puce reste affirmée pendant un court laps de temps après que toutes les données ont été transmises, cependant, si un délai est spécifié, la sélection de puce restera confirmée pendant ce temps supplémentaire (en microsecondes).

`Bits` : Le paramètre `bits` (par mot) ne peut avoir qu'une seule valeur, 8, dans le Raspberry Pi.

Remarques: par défaut, la taille de la mémoire tampon utilisée par les fonctions `xfer`, `xfer2` et `xfer3` est de 4 096 octets, mais cela peut être modifié. Voir Taille maximale du tampon SPI ci-dessous.

La fonction échoue si la liste de valeurs est plus grande que la taille maximale du tampon. Utilisez `xfer3` avec les listes plus longues.

Alors que la documentation dit que «le signal de sélection de puce sera libéré et réactivé entre les blocs», cela n'a pas beaucoup de sens, car cette fonction envoie un seul bloc de données dont la taille maximum est celle du tampon.

## xfer2

Description : effectue une transaction SPI. Une liste d'octets est écrite sur le périphérique SPI et alors que chaque octet de cette liste est envoyé, il est remplacé par les données lues simultanément depuis le dispositif esclave SPI sur la ligne MISO.

Syntaxe : `rcvd = xfer2([valeurs])`

`rcvd = xfer2([valeurs], vitesse)`

`rcvd = xfer2([valeurs], vitesse, délai)`

`rcvd = xfer2([valeurs], vitesse, délai, bit`

Remarque: La documentation indique que «contrairement à `xfer`, le signal de sélection de puce sera maintenu actif entre les blocs» mais cela ne semble pas être confirmé. En fait, il semble bien que les fonctions `xfer` et `xfer2` soient identiques.

## xfer3

Description: effectue une transaction SPI. Une liste d'octets de taille arbitraire est écrite sur le périphérique SPI et à mesure que chaque octet de cette liste est envoyé, il est remplacé par les données lues simultanément depuis le périphérique esclave SPI sur la ligne MISO.

Syntaxe : `rcvd = xfer3([valeurs])`

`rcvd = xfer3([valeurs], vitesse)`

`rcvd = xfer3([valeurs], vitesse, délai)`

```
rcvd = xfer3([valeurs], vitesse, délai, bit
```

Renvoie : [valeurs]

Paramètre :

`[valeurs]` : une liste d'octets

`vitesse` : fréquence de l'horloge SPI en Hz. Si non spécifiée, l'horloge SPI est définie sur l'attribut `maximum_speed_hz`. Si la vitesse est spécifiée, elle sera utilisée pour cette transaction unique et ne modifiera pas la valeur de l'attribut `maximum_speed_hz` de l'objet.

`Délai` : la ligne de sélection de puce reste affirmée pendant un court laps de temps après que toutes les données ont été transmises, cependant, si un délai est spécifié, la sélection de puce restera confirmée pendant ce temps supplémentaire (en microsecondes).

`Bits` : Le paramètre `bits` (par mot) ne peut avoir qu'une seule valeur, 8, dans le Raspberry Pi.

Remarque : contrairement à `xfer` et `xfer2`, qui échouent si la liste de valeurs est supérieure à la taille maximale du tampon SPI, xfer3 gère une longue liste en la divisant en bloc d'octets envoyés un par un. La taille de chaque bloc est égale à la taille maximale du tampon sauf peut-être pour le dernier qui pourrait être plus petit.

Le signal de sélection de puce est libéré entre chaque bloc pendant environ un cinquième de millisecondes et indéfiniment après l'envoi du dernier bloc de données. Si un délai est spécifié, la ligne de sélection de puce restera confirmée pendant cette période de temps à la fin de la transmission de chaque bloc de données avant d'être libérée.

## 9. Taille maximale du tampon SPI

Les données à envoyer vers un périphérique SPI à l'aide de l'une des cinq méthodes `writebytes`*`(x)`* et `xfer`*`(x)`* sont mises en mémoire tampon. Les fonctions `writebytes`, `xfer` et `xfer2` utilisent un tampon dont la taille est fixée à 4 096 octets et ne transfèrement pas plus de données.

La taille du tampon utilisé par `writebytes2` et `xfer3`, qui peut envoyer une liste de données de taille arbitraire, est spécifiée dans le fichier `/sys/module/spidev/parameters/bufsiz`.

```
pi@rasberrypi:~ $ cat /sys/module/spidev/parameters/bufsiz
4096
```

Cette valeur peut être modifiée.

Il est possible de changer la taille du tampon de manière permanente en ajoutant une option `spidev.bufsiz = xxxx` (où xxxx est la taille de tampon nécessaire) dans le fichier /boot/cmdline.txt. N'oubliez pas de conserver le contenu de ce fichier sur une seule ligne. Le système doit être redémarré pour que cette modification prenne effet.