
Khởi đầu luôn khó khăn, hãy để chúng tôi giúp bạn

*Câu chuyện bắt đầu từ một cậu bé,*
*và một ý tưởng*
*có thể*
*làm thay đổi thế giới...*

PAY IT FORWARD

*Đó là khi bạn giúp đỡ 3 người bạn không quen biết,*
*dù là bằng thời gian,*
*hay công sức,*
*hay kinh nghiệm,*
*hay kiến thức,*
*hay tiền bạc, ...*
*của mình.*

*Mà không chờ đợi một sự báo ân nào.*

*Chỉ cần mỗi người trong 3 người đó,*
*lại đem những gì mình có, mà người khác cần,*
*tiếp tục giúp đỡ thêm 3 người nữa.*

*Chính những người-giúp-đỡ, và người-được-giúp-đỡ,*
*sẽ là những người góp phần thay đổi thế giới...*

*Một thế giới sẻ chia kiến thức - và yêu thương ...*

# PAY IT FORWARD ...

Chúng tôi không sáng tạo ra câu nói này.

Pay it forward...

Hãy tri ân người giúp mình bằng cách giúp đỡ người khác
Cho đi không phải để nhận lại.

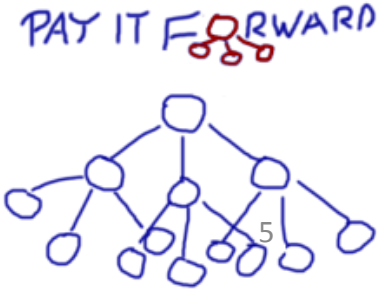

# Serial Peripheral Interface (SPI)

28/10/2014

payitforward.edu.vn

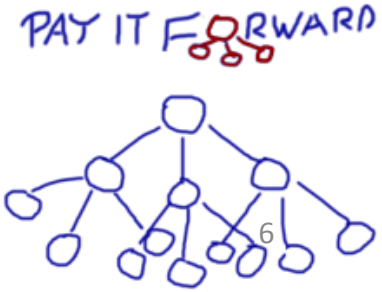
TIVA ARM Cortex-M4
TM4C123G Tutorial

- Cấu hình
- Một số hàm
- Chương trình mẫu



Bốn module SSI, mỗi module với:

- Chế độ SPI, MICROWIRE hay TI SSI
- Hoạt động chế độ Master hay Slave
- Có thể điều chỉnh tốc độ xung nhịp bit
- Có thể chia tần số (pre-scaler)
- Kích thước khung truyền từ 4 đến 16-bits
- Tx và Rx FIFO riêng ( 8x16-bits )
- Hỗ trợ ngắt và µDMA



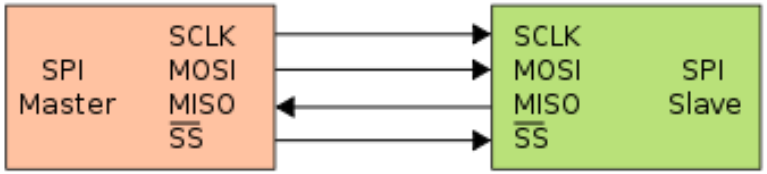
SPI bus gồm 4 tín hiệu logic

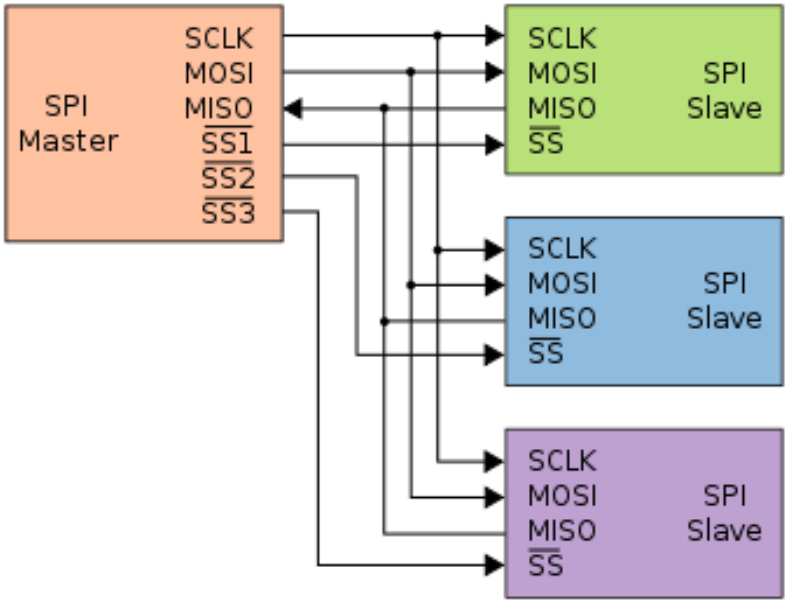
SCLK : Serial Clock (ngõ ra của master)

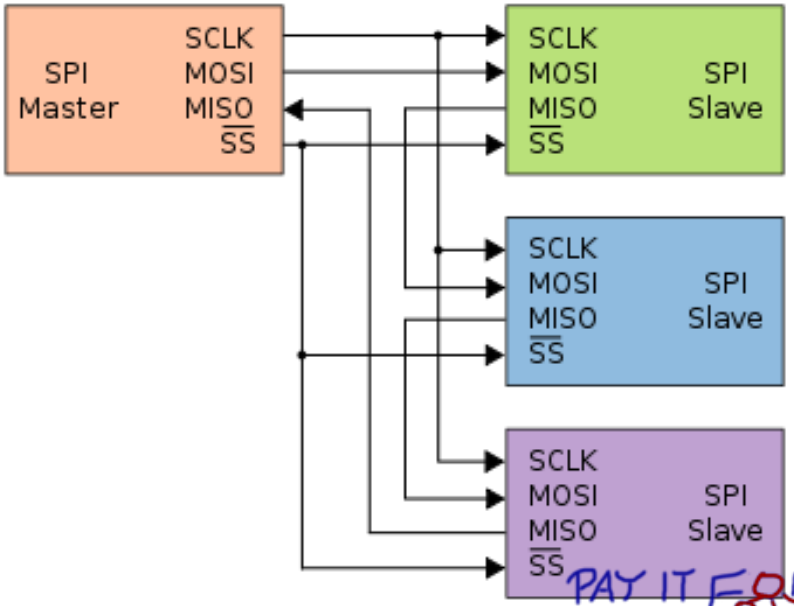
MOSI : Master Output, Slave Input

MISO : Master Input, Slave Output

SS : Slave Select (tích cực thấp, ngõ ra của master)



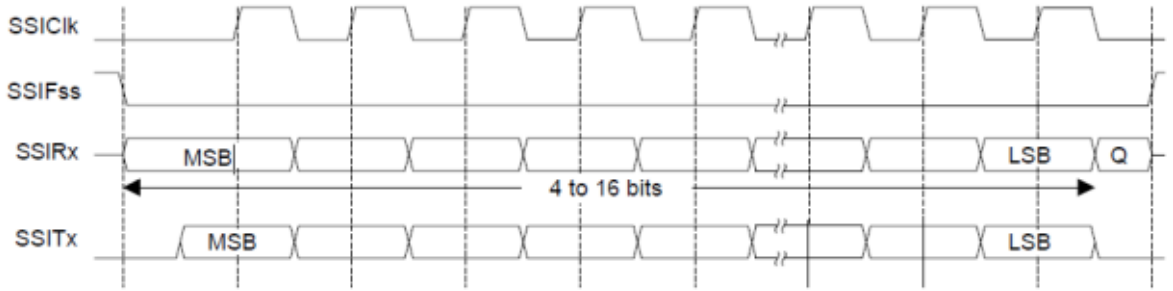



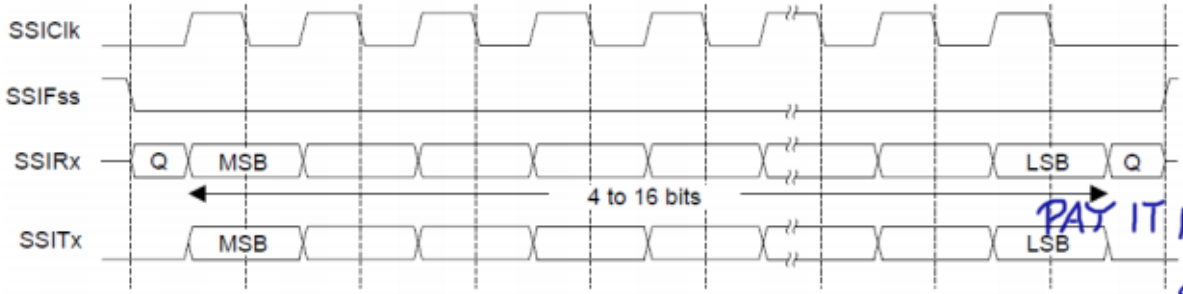

Giao tiếp 4 dây. Song công
Thanh ghi SSIFss để chọn chip
Trạng thái nghỉ và pha clock được lập trình thông qua bit SPO và SPH (SSI_FRF_MOTO_MODE_(0 đến 3))

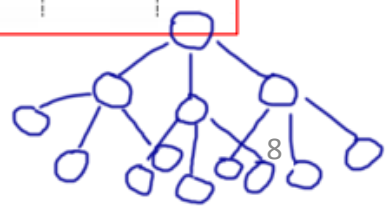
- SPO = 0: SSIClk ở mức thấp khi không hoạt động. SPO = 1: ngược lại
- SPH = 0: Dữ liệu được lấy mẫu ở lần chuyển trạng thái thứ nhất của SSIClk. SPH = 1: lấy mẫu ở lần thứ 2.

**SPO = 0**
**SPH = 0**
**Single**
**Transfer**




**SPO = 0**
**SPH = 1**
**Single**
**Transfer**



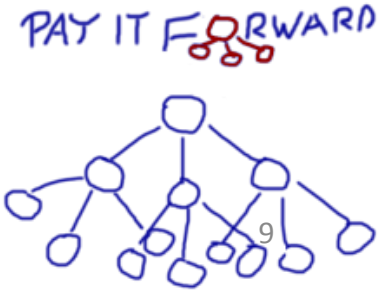



# NGẮT SSI

Mỗi module bao gồm một ngắt.
Điều kiện ngắt bao gồm:

- FIFO truyền còn ít hơn hoặc bằng một nửa.
- FIFO nhận nhiều hơn hoặc bằng một nửa.
- Hết thời gian chờ của FIFO nhận.
- Tràn FIFO nhận.
- Kết thúc truyền.
- DMA nhận hoàn tất việc truyền.
- DMA truyền hoàn tất việc truyền.

Các ngắt trên có thể được kích hoạt độc lập. Code xử lí phải kiểm tra để xác định nguồn ngắt SSI và xóa cờ.



# HOẠT ĐỘNG CỦA SSI µDMA

Kênh Tx và Rx riêng.

Khi kích hoạt, SSI sẽ xác nhận yêu cầu của DMA trên mỗi kênh khi FIFO truyền hoặc nhận có thể truyền dữ liệu.

Với kênh Rx: Một yêu cầu đơn được tạo ra khi có bất kì dữ liệu nào trong FIFO Rx.

Với kênh Tx: Một yêu cầu đơn được tạo ra khi có ít nhất một vị trí trống trong FIFO truyền.

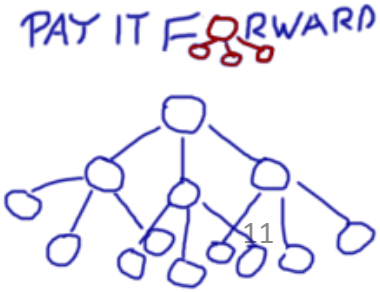
# CẤU HÌNH THANH GHI

Kích hoạt SSI để sử dụng

```
SysCtlPeripheralEnable(uint32_t ui32Peripheral);
```

Tiếp theo là kích hoạt cổng GPIO nào cần sử dụng

```
SysCtlPeripheralEnable(uint32_t ui32Peripheral);
```

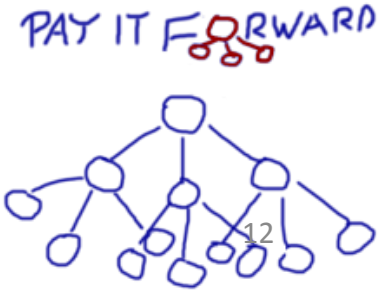


# CẤU HÌNH THANH GHI

Tiếp theo cấu hình cho các chân chức năng cho SSI

```
GPIOPinConfigure(uint32_t ui32PinConfig);
```

Tùy theo mục đích mà chọn chân cho phù hợp.

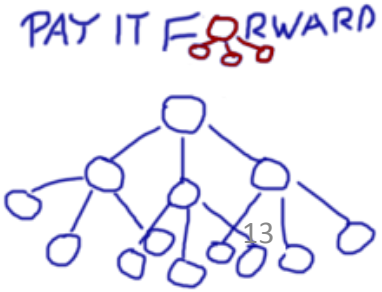


# CẤU HÌNH THANH GHI

Hàm sau cấp quyền điều khiển  những chân này cho SSI. Xem datasheet để biết chức năng nào được dùng trên mỗi chân. Trong ví dụ này:

PA5 - SSI0Tx
PA4 - SSI0Rx
PA3 - SSI0Fss
PA2 - SSI0CLK

```
GPIOPinTypeSSI(uint32_t ui32Port, uint8_t ui8Pins);
```



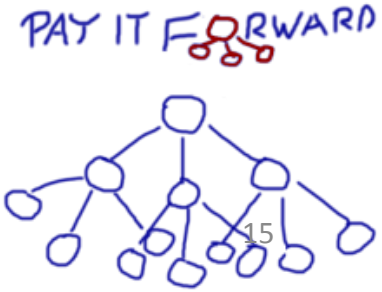
## Table 15-1. SSI Signals (64LQFP)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| SSI0Clk | 19 | PA2 (2) | I/O | TTL | SSI module 0 clock |
| SSI0Fss | 20 | PA3 (2) | I/O | TTL | SSI module 0 frame signal |
| SSI0Rx | 21 | PA4 (2) | I | TTL | SSI module 0 receive |
| SSI0Tx | 22 | PA5 (2) | O | TTL | SSI module 0 transmit |
| SSI1Clk | 30<br>61 | PF2 (2)<br>PD0 (2) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | 31<br>62 | PF3 (2)<br>PD1 (2) | I/O | TTL | SSI module 1 frame signal. |
| SSI1Rx | 28<br>63 | PF0 (2)<br>PD2 (2) | I | TTL | SSI module 1 receive. |
| SSI1Tx | 29<br>64 | PF1 (2)<br>PD3 (2) | O | TTL | SSI module 1 transmit. |
| SSI2Clk | 58 | PB4 (2) | I/O | TTL | SSI module 2 clock. |
| SSI2Fss | 57 | PB5 (2) | I/O | TTL | SSI module 2 frame signal. |
| SSI2Rx | 1 | PB6 (2) | I | TTL | SSI module 2 receive. |
| SSI2Tx | 4 | PB7 (2) | O | TTL | SSI module 2 transmit. |
| SSI3Clk | 61 | PD0 (1) | I/O | TTL | SSI module 3 clock. |
| SSI3Fss | 62 | PD1 (1) | I/O | TTL | SSI module 3 frame signal. |
| SSI3Rx | 63 | PD2 (1) | I | TTL | SSI module 3 receive. |
| SSI3Tx | 64 | PD3 (1) | O | TTL | SSI module 3 transmit. |

# CẤU HÌNH THANH GHI

Cấu hình và kích hoạt cổng SSI:

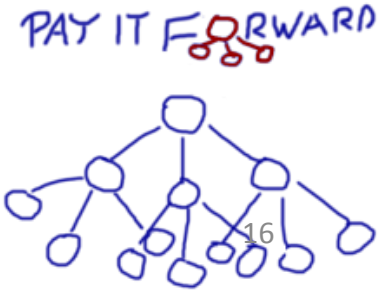
```
SSIConfigSetExpClk(uint32_t ui32Base, uint32_t ui32SSIClk,
                   uint32_t ui32Protocol, uint32_t ui32Mode,
                                uint32_t ui32BitRate,
                                uint32_t ui32DataWidth);
```



# CẤU HÌNH THANH GHI

Cuối cùng là kích hoạt module SSI

```
SSIEnable(uint32_t ui32Base);
```



# MỘT SỐ HÀM KHÁC

```
SSIDataGetNonBlocking(uint32_t ui32Base, uint32_t *pui32Data);
```

true khi có dữ liệu trả về, false khi không có dữ liệu trả về

```
SSIDataPut(uint32_t ui32Base, uint32_t ui32Data);
```

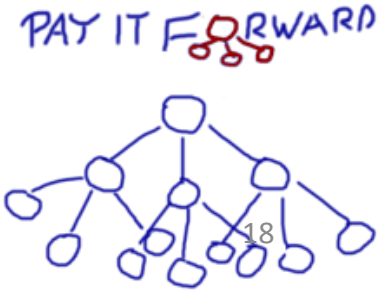
```
SSIDataGet(uint32_t ui32Base, uint32_t *pui32Data);
```

```
SSIBusy(uint32_t ui32Base);
```
true nếu đang truyền dữ liệu, thường được dùng để kiểm tra SSI truyền hết dữ liệu hay chưa.

# CHƯƠNG TRÌNH MẪU

Chương trình sau chỉ ra cách để cấu hình SSI0 thành SPI master. Ví dụ này sẽ gửi 3 byte dữ liệu và hỏi FIFO cho tới khi nào nhận được 3 byte

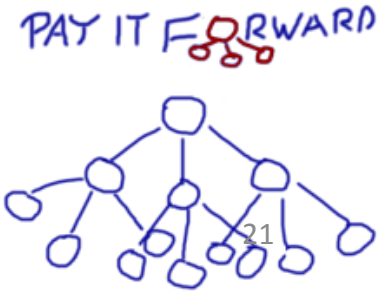
```
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/ssi.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
```



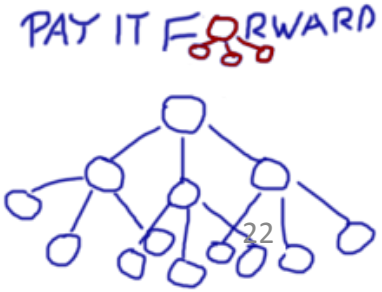
```
// Số byte gửi và nhận
#define NUM_SSI_DATA   3

// Cấu hình UART (các bạn tham khảo bài giảng về UART ;) )
void
InitConsole(void)
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
    GPIOPinTypeUART(GPIO_PORTA_BASE,
                                GPIO_PIN_0 | GPIO_PIN_1);
    UARTStdioConfig(0, 115200, 16000000);
}
```

```
int
main(void)
{
    uint32_t pui32DataTx[NUM_SSI_DATA];
    uint32_t pui32DataRx[NUM_SSI_DATA];
    uint32_t ui32Index;


  // Cấu hình xung clock
    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC |
                   SYSCTL_OSC_MAIN |  SYSCTL_XTAL_16MHZ);

  // Cấu hình giao tiếp nối tiếp (chỉ dùng cho ví dụ này)
    InitConsole();
```

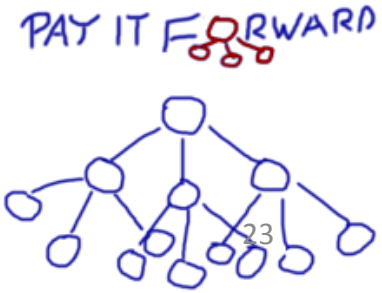
```
// Hiển thị cài đặt trên giao diện
    UARTprintf("SSI ->\n");
    UARTprintf("  Mode: SPI\n");
    UARTprintf("  Data: 8-bit\n\n");

// Kích hoạt SSI0
  SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI0);

// Kích hoạt cổng GPIO
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

// Cấu hình các chân cần sử dụng
  GPIOPinConfigure(GPIO_PA2_SSI0CLK);
  GPIOPinConfigure(GPIO_PA3_SSI0FSS);
  GPIOPinConfigure(GPIO_PA4_SSI0RX);
  GPIOPinConfigure(GPIO_PA5_SSI0TX);
```



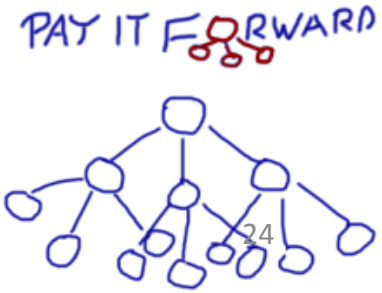
```
// Cấu hình cài đặt GPIO cho các chân SSI
 GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_5 | GPIO_PIN_4
                                | GPIO_PIN_3 | GPIO_PIN_2);

// Cấu hình cổng SSI ở chế độ master
    SSIConfigSetExpClk(SSI0_BASE, SysCtlClockGet(),
       SSI_FRF_MOTO_MODE_0,SSI_MODE_MASTER, 1000000, 8);

// Kích hoạt module SSI0
    SSIEnable(SSI0_BASE);

// Câu này để đọc dữ liệu còn lại trong cổng SSI, đảm bảo FIFOs đang trống
    while(SSIDataGetNonBlocking(SSI0_BASE, &pui32DataRx[0]))
    {
    }

        // Dữ liệu gửi
            pui32DataTx[0] = 'p';
            pui32DataTx[1] = 'i';
            pui32DataTx[2] = 'f';
```



```
// Hiển thị dấu hiệu thể hiện SSI đang truyền dữ liệu
    UARTprintf("Sent:\n  ");

// Hiển thị dữ liệu đang được truyền
        UARTprintf("'%c' ", pui32DataTx[ui32Index]);

// Gửi 3 byte dữ liệu
    for(ui32Index = 0; ui32Index < NUM_SSI_DATA; ui32Index++)
    {
        SSIDataPut(SSI0_BASE, pui32DataTx[ui32Index]);
    }

// Chờ SSI0 kết thúc quá trình truyền
    while(SSIBusy(SSI0_BASE))
    {
    }
```

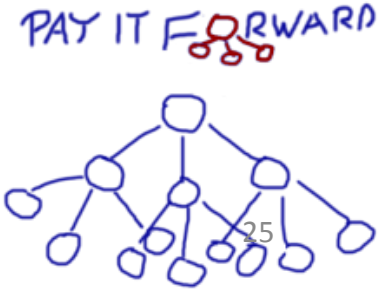


```
// Hiển thị dấu hiệu thể hiện SSI đang nhận dữ liệu
    UARTprintf("Receive:\n  ");

// Nhận 3 byte dữ liệu
    for(ui32Index = 0; ui32Index < NUM_SSI_DATA; ui32Index++)
    {
        SSIDataGet(SSI0_BASE, &pui32DataRx[ui32Index]);

 // Do sử dụng 8 bit dữ liệu, loại bỏ phần MSB
        pui32DataRx[ui32Index] &= 0x00FF;

 // Hiển thị dữ liệu đang nhận
        UARTprintf("'%c' ", pui32DataRx[ui32Index]);
    }
    return(0);
}
```



# Tài liệu tham khảo

[1] Peripheral Driver Library.pdf
[2] TM4C123G_LaunchPad_Workshop_Workbook.pdf
[3] tm4c123gh6pge.pdf
[4] TivaWare for C Series Software



PAY IT FORWARD

payitforward.edu.vn