PIC16F877A (MPLAP-XC8)

1. GPIO
2. INTERUPT
3. Tổng quan về ngắt:

* Trong máy tinhs, ngắt (INTERRUPT) là một tín hiệu được gửi đến bộ xử lí yêu cầu tạm dừng thực thi chương trình hiện tại và chuyển xang thực thi một chương trình khác. Ngắt có thể gây ra bởi phần cứng và phần mềm
* Phân loại các loại ngắt:
  + Ngắt ngoài (EXTERNAL INTERRUPT)
  + Ngắt timer (TIMER 0, TIMER 1, TIMER 2)
  + Ngắt ADC
  + Ngắt PWM

1. Cấu trúc một chương trình ngắt

Main(){

// cấu hình thực hiện ngắt

While(1){

// câu lệnh

}

}

// hàm ngắt

Isr(){

If( flag == 1){ // kiểm tra xem trạng thái của cờ tràn

// Thực hiện các câu lệnh muốn thực hiện ngắt

Flag = 0; // đặt lại cờ tràn về giá trị = 0

}

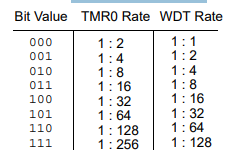
}

1. EXTERNAL INTERRUPT : ngắt ngoài

* Chân cho phép ngắt ngoài : RB0/INT
* Các thanh ghi sử dụng
* INTCON :
  + GIE(Global Interrupt Enable) : bit cho phép ghắt toàn cục
    - = 1: cho phép ngắt
    - = 0: không cho phép
  + PEIE(Peripheral Interrupt Enable): bit cho phép ngắt ngoại vi
    - = 1: ngắt trên các các thiết bị ngoại vi sẽ được bật
    - = 0: ngắt trên các các thiết bị ngoại vi sẽ được tắt
  + INTE(: RB0/INT External Interrupt Enable): bit cho phép ngắt ngoài trên RB0/INT
    - = 1: cho phép RB0/INT ngắt ngoài
    - = 0: không cho phép RB0/INT ngắt ngoài
  + INTF(: RB0/INT External Interrupt Flag bit): cờ tràn ngắt ngoài trên RB0/INT
    - =1: cờ được bật (RB0/INT ngắt ngoài đã sảy ra ): cờ tràn cần phải đặt lại
    - =0: ngất ngoài RB0/INT chưa được thực hiện
* OPTIONAL\_REG :
  + RBPU(PORTB Pull-up Enable):bit bật tắc pull-up cho PORTB: // chân đảo
    - = 0: bật pull-up cho PORTB ( giá trị mặc định )
    - = 1: tắt pull-up cho PORB
  + INTEDG(Interrupt Edge Select): bit xác định loại cạnh xung đầu vào ngắt ngoài sẽ đượ thực hiện
    - = 1: xung sườn lên
    - = 0: xung sườn xuống

1. TIMER

* Tổng quan timer
* ứng dụng của timer
* thanh ghi được sử dụng :
* OPTIONAL\_REG:
  + TOCS( TMR0 Clock Source Select): bit chọn nguồn xung
    - = 1: chọn xung ngoại từ RA4/TOCKI
    - = 0: chọn xung nội thanh anh
  + PSA (Prescaler Assignment ): bit cho phép bộ chia
    - = 1: bộ chia trước được sử dụng cho WDT
    - = 0: bộ chia trước được sử dụng cho TIMER0
  + PS2 – PS0 (: Prescaler Rate Select): bit chọn bộ chia trước



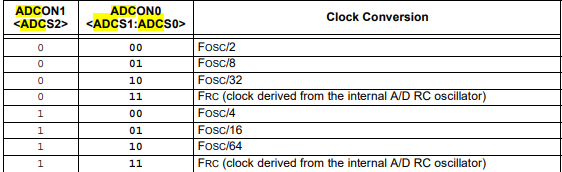
* INTCON :
  + GIE(Global Interrupt Enable) : bit cho phép ghắt toàn cục
    - = 1: cho phép ngắt
    - = 0: không cho phép
  + PEIE(Peripheral Interrupt Enable): bit cho phép ngắt ngoại vi
    - = 1: ngắt trên các các thiết bị ngoại vi sẽ được bật
    - = 0: ngắt trên các các thiết bị ngoại vi sẽ được tắt
  + TMR0IE :
    - = 1: cho phép sử dụng TIMER0
    - = 0: không cho phép sử dụng TIMER0
  + TMR0F : cờ báo tràn
    - = 1: giá trị TMR0 đã tràn
    - = 0: giá trị trên thanh ghi TMR0 chưa tràn
* Công thức tính toán giá trị thanh ghi TMR0
  + TMR0 là thanh ghi 8bit ghi giá trị bắt đầu đếm của TIMER0: có thể đếm lên đến 255 hoặc đếm xuống giá trị 0 là kết thúc một chu kỳ timer 0
  + Công thức:
    - Delay\_time = (255 – TMR0)\* prescalar\*machine\_cycle

Trong đó:

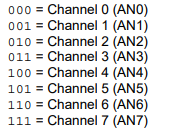
* + - delay\_time : là thời gian mong muốn thực hiện được một chu kỳ timer0
    - TMR0 là giá trị cần tìm
    - Prescalar là giá trị bộ chia trước
    - Machine\_cycle là giá trị một chu kỳ máy = 4/f0

1. ADC

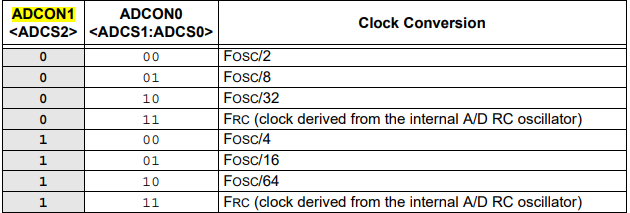
* Tổng quan
* ứng dụng ADC
* các thanh ghi được sử dụng
* INTCON :
  + GIE(Global Interrupt Enable) : bit cho phép ghắt toàn cục
    - = 1: cho phép ngắt
    - = 0: không cho phép
  + PEIE(Peripheral Interrupt Enable): bit cho phép ngắt ngoại vi
    - = 1: ngắt trên các các thiết bị ngoại vi sẽ được bật
    - = 0: ngắt trên các các thiết bị ngoại vi sẽ được tắt
* ADCON0:
  + ADCS1-ADCS0<bit 7- bit 6>( A/D Conversion Clock Select ): bit chọn xung chuyển đổi ADC



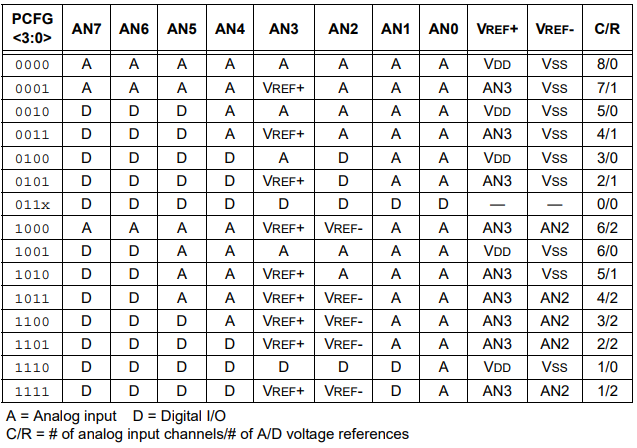
* + CHS2-CHS0(: Analog Channel Select bits): bit chọn chân analog



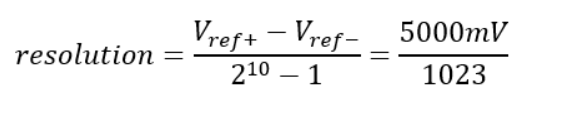
* + GO/DONE: chỉ được bật khi bắt đầu muốn thực hiện quá trìn adc
    - = 1: bắt đầu quá trình chuyển đổi ADC
    - =0: không thực hiện chuyển đổi ADC
  + ADON:
    - = 1: Modul ADC được mở
    - = 0: Modul ADC tắt
* ADCON1:
  + ADFM: bit chọn kết quả quá trình lưu dưới dạng
    - =1: 8 MSB được lưu ở ADRESH, 2 LSM được lưu ở ADRESL
    - =0: 8 LSM được lưu ở ADRESH, 2MBS được lưu ở ADRESH
  + ADCS2: bit



* + PCFG3 – PCFG0: bit lựa chọn chức năng của các chân AN7-AN0



* PIR1:
  + ADIE:
    - = 1:
    - = 0:
  + ADIF:
    - = 1:
    - = 0:
* Độ phân giải :

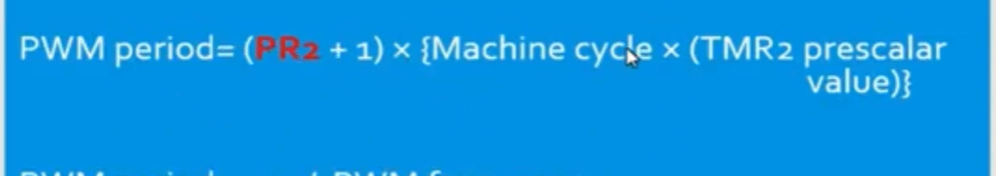


* Kết quả của quá trình chuyển đổi được lưu vào 2 thanh ghi ADRESH và ADRESL
  + - Kết quả = (ADRESH << 8) | ADRESL (8 bit MSB được lưu vào ADRESH)
    - Kết quả = (ADRESH << 2) | ADRESL (8 bit LSB được lưu vào ADRESL)

1. PWM

* Tổng quan
* ứng dụng
* các thanh ghi được sử dụng
* các bước thực hiện tùy chỉnh xung

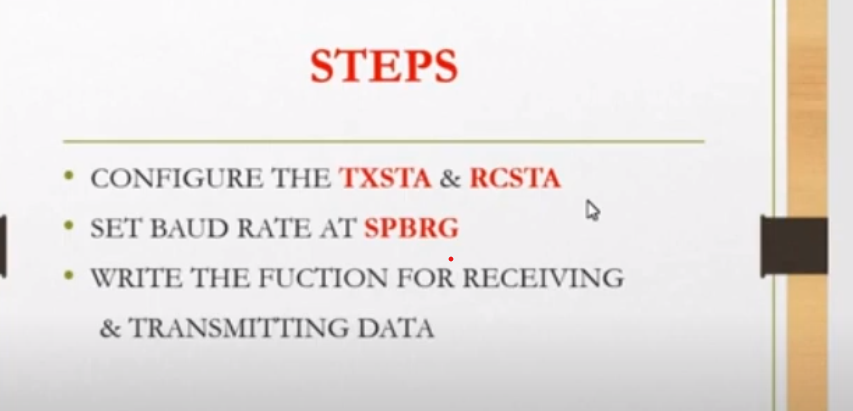
1. bật chế độ timer2 t2con
2. tạo bộ chia trước t2con
3. bât modul pwm ccpxcon
4. tính giá trị pr2



1. set vào bit giá trị ccp2con & ccpr1l
2. eeprom

* Trong pic có bộ nhớ eeprom có dung lượng 256 byte
* ứng dụng
* cách lập trình: các thanh ghi được thực hiện
* EEADR : thanh ghi lưu trữ địa chỉ của dữ liệu trong eeprom
* EEDATA: thanh ghi chứa dữ liệu
* EECON1
  + EEPGD : bit cho phép truy cập vào bộ nhớ chương trình(1)/ bộ nhớ dữ liệu(0)
  + WEN : bit cho phép ghi trên eeprom (1) / không được phép
  + WR : bit thể hiện việc hoàn thành việc ghi bit này không thể xóa dc trên phần mềm
  + RD : bit thể hiện việc hoàn thành việc đọc dữ lệu và tương tự với WR bit này chỉ được phép set trên phần mềm
  + PIR2bits.EEIF : cờ tràn báo hiệu việc ghi dữ liệu
* EECON2 nhận 2 giá trị 0x55 và 0xAA

1. Uart truyền dữ liệu nối tiếp bất đồng bộ

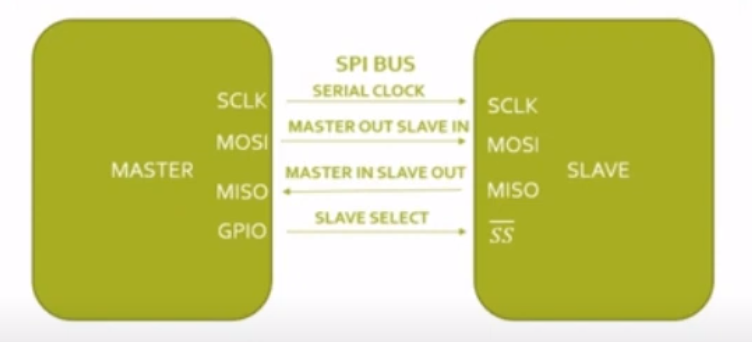


TXSTA :

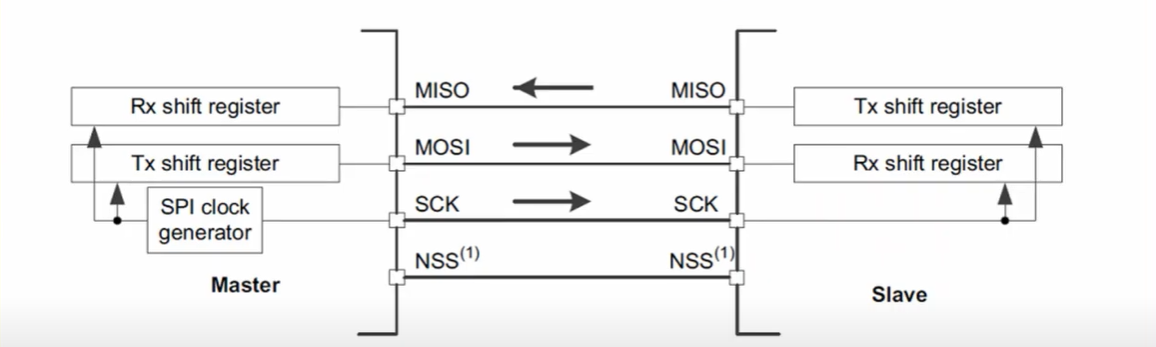
* + SYNC : lựa chọn truyền đồng bộ(1) / bất đồng bộ(0)
  + TXEN: bit cho phép chuyền nhận
  + BRGH : bit lựa chọn tốc độ truyền với truyền bất đồng bộ

1. Spi

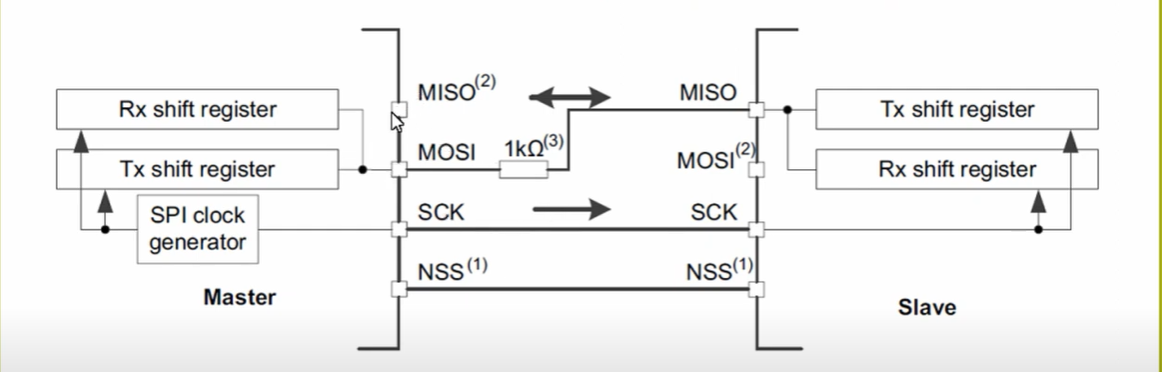
* Là giao thức truyền nhận giữa một master và một hoặc nhiều slave
* Sử dụng các pin: SCLK, MOSI, MISO , SS



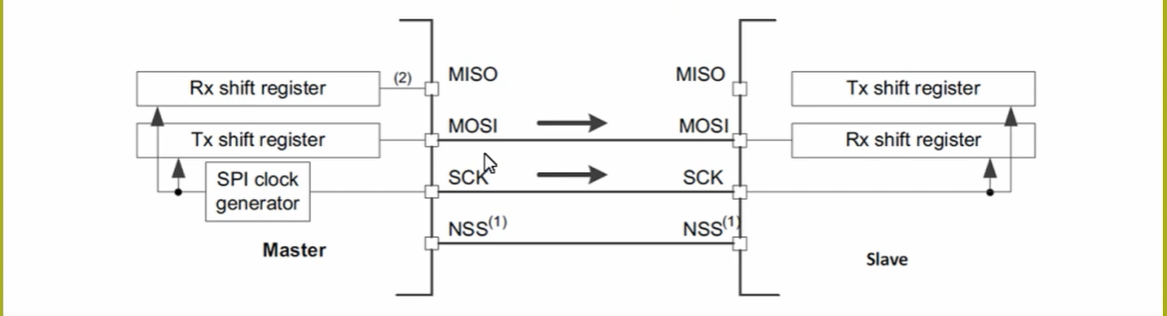
* + SCLK : là đồng hộ xung được cung cấp bởi master
  + MOSI: là chân Master in/slave out có chức năng truyền dữ liệu ở chế độ chính và nhận dữ liệu ở chế độ phụ
  + MISO: là chân Master out/slave in có chức năng nhận dữ liệu ở chế độ chính và truyền dư liệu ở chế độ phụ
  + SS là chân chọn slave đối với trường hợp một master giao tiếp với nhiều slave(pin nhận giá trị 0 thì slave dc chọn 1 thì không được chọn)
* Note:
  + việc đầu tiên khi thực hiện giao tiếp với nhiều slave là chọn slave giao tiếp với master
  + tùy vào ứng dụng thì giao tiếp spi có thế giảm bớt số lượng bus chỉ giữ lại bit cần thiết
* Biến thể của spi
  + Giao tiếp song song hoàn toàn(full-duplex communication): là giao tiếp master-slave tại một thời điểm trong cùng một thiết bi master và slave kết nối 4 bus



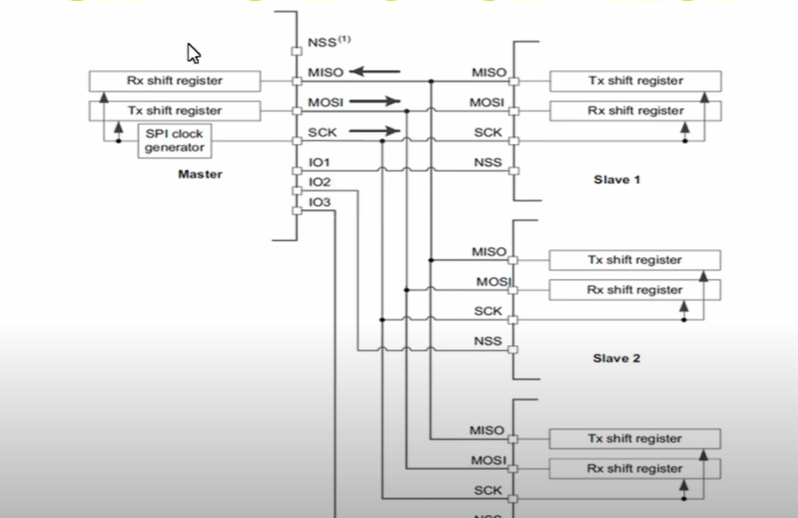
* + Giao tiếp bán song song(half-duplex communication) : cách mắc MOSI-master sẽ được nối với MISO-slave điều này khiến giao tiếp giữa master và slave diễn ra không đồng thời tại một thời điểm



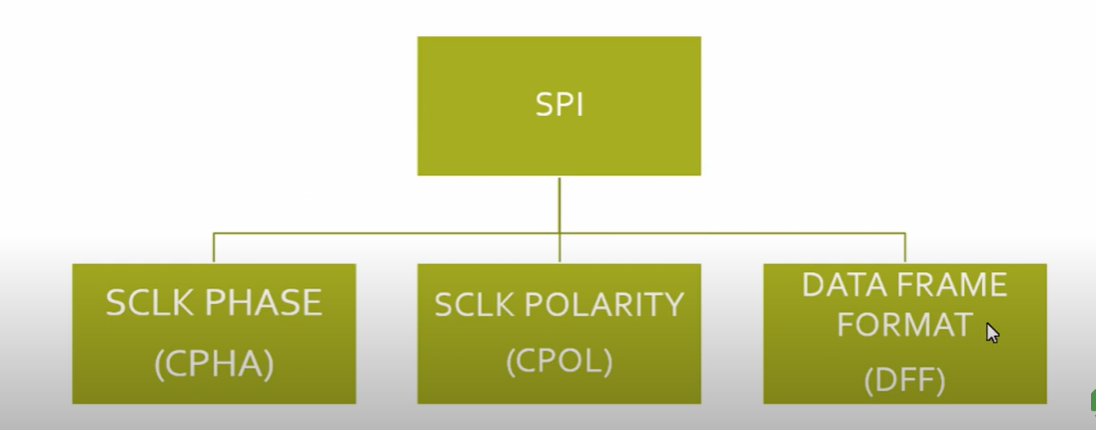
* + Giao tiếp đơn giản(simplex communication): cách mắc chỉ có dây MOSI-master nối với MOSI-slave ở giao tiếp này master chỉ đảm nhiệm vai trò chuyền, slave chỉ nhận



* Sơ đồ master-slave:



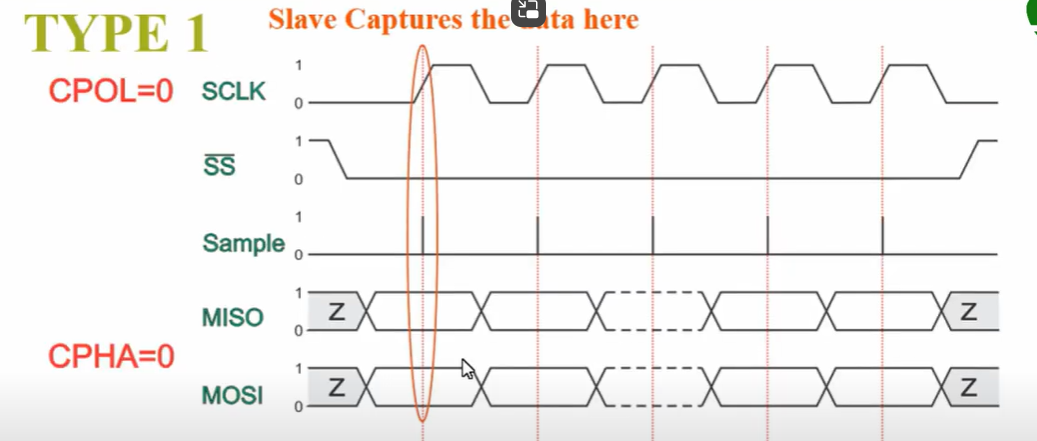
* Định dạng truyền thông spi và kiểu dữ liệu



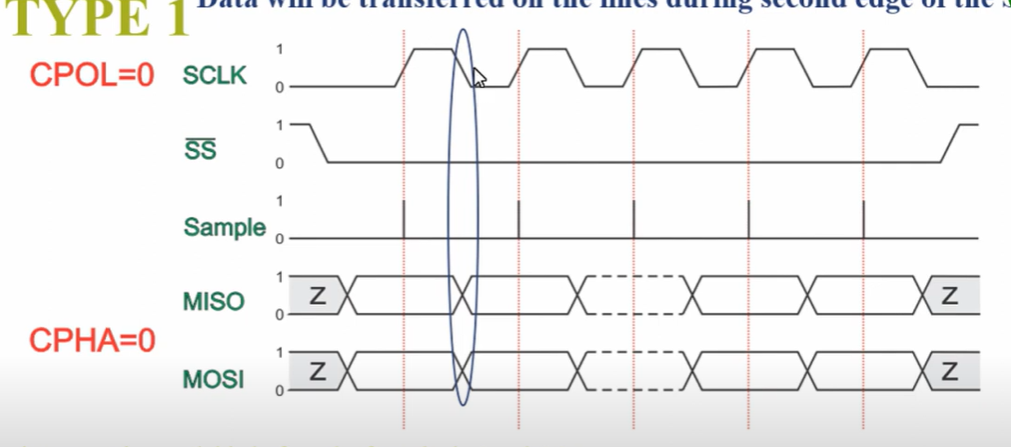
* + Spi phụ thuộc vào 3 tham số trên ngoài ra còn có
    - Data toggling: là sự thay đổi dữ liệu (vidu là sự chuyển đổi giữa mức 0 -> 1
    - Data sampling: là giá trị lấy mẫu thường



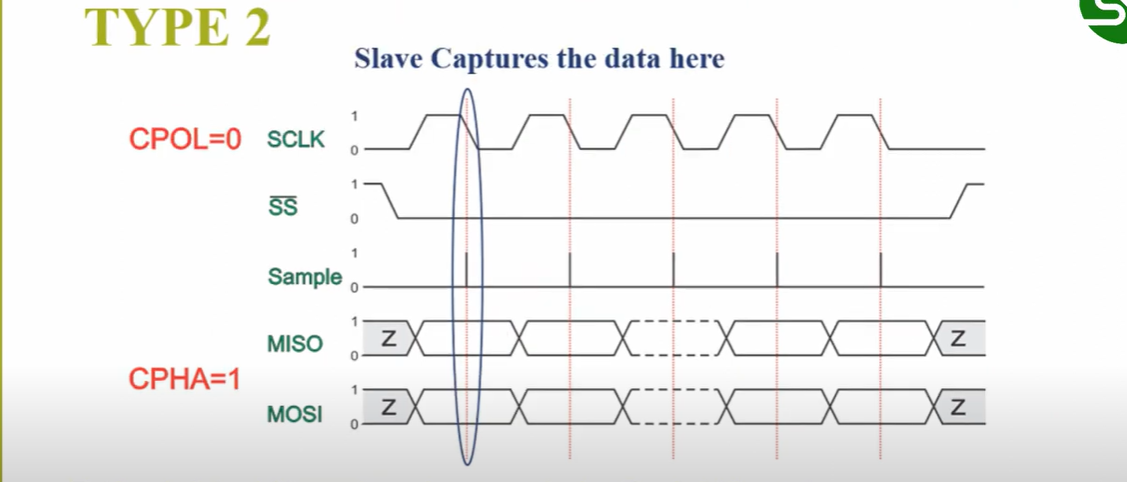
* + - CPOL là giá trị thể hiện trạng thái lí tưởng của mặt đất
      * Nếu CPOL = 0 -> SCLK = 0;
      * Nếu CPOL = 1 -> SCLK = 1;
    - Cpha là giá trị xác định quá trình chuyển đổi đồng hồ mà tại đó dữ liệu được lấy mẫu
      * Nếu cphase = 1: dữ liệu được lấy mẫu ở cạnh thử nghiệm của xung
      * Nếu cphase = 0: dữ liệu được lấy mẫu ở cạnh đầu của xung

Ex: 

* + Khi cpol = 0, cpha = 0 thì slave ghi lại dữ liệu(lấy mẫu) ở cạnh đầu của xung sclk



* + Tại cạnh xung xuống thứ 2 thì dữ liệu bị chuyển đổi ở 2 dòng MISO và MOSI



* Thanh ghi sử dụng

Sspcon

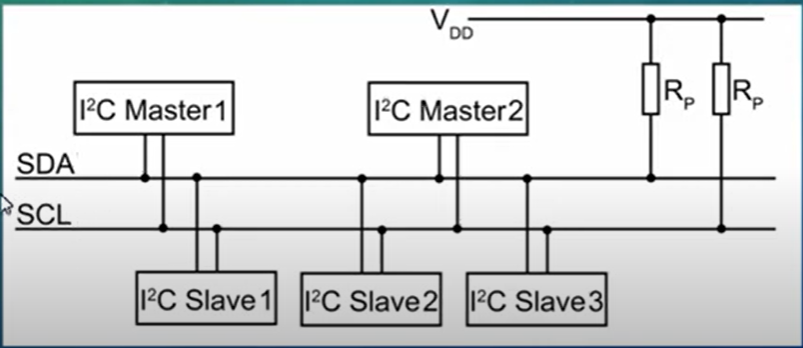
Sspen

Cmp

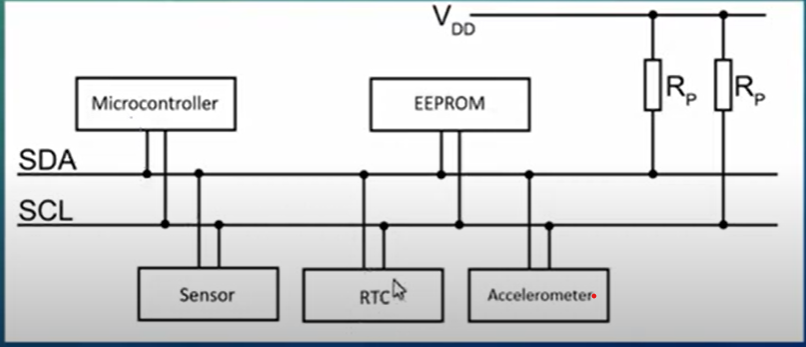
Sspstat bit 6 7

1. I2C

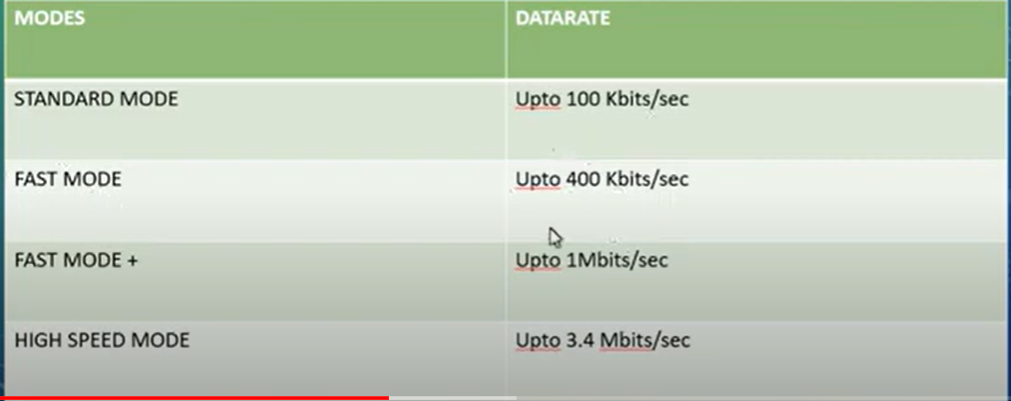
* I2C là chuẩn giao tiếp thường dung giữa các thiết bị có khoảng cách gần vd: các thiết bị nằm trên cùng một pcb
* Sơ đồ khối của I2C



* Các master và slave kết nối thông qua duy nhất 2 bus dữ liệu SDA và SCL
* Ngoài ra thì 2 bus trên kết nối với nguồn thông qua 2 trở
* Trong giao tiếp I2C: master và slave giao tiếp với nhau thông qua địa chỉ

Vd 

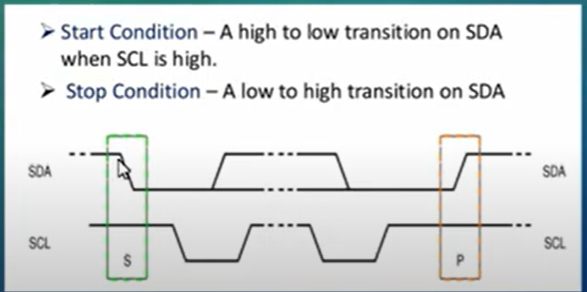
* + Mcu muốn lấy giá trị cảm biến của sensor thì phải gọi được địa chỉ đúng của sensor …
  + RTC muốn chuyền dữ liệu cho mcu thì cũng phải gọi địa chỉ của mcu
* SDA và SCL lines:
  + Cả hai đường đều là 2 chiều. chúng cùng kết nối với nguồn thông qua 1 trở kéo lên -> cả hai đường sẽ nhận mức cao ngay cả khi chúng không được thiết lập
* Tùy chỉnh của I2C



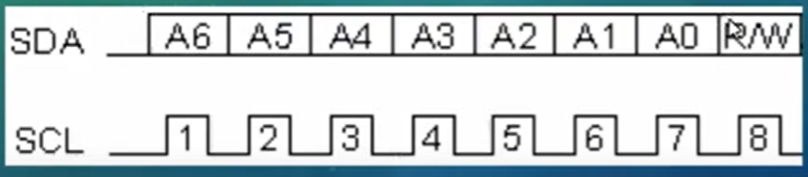
* I2C



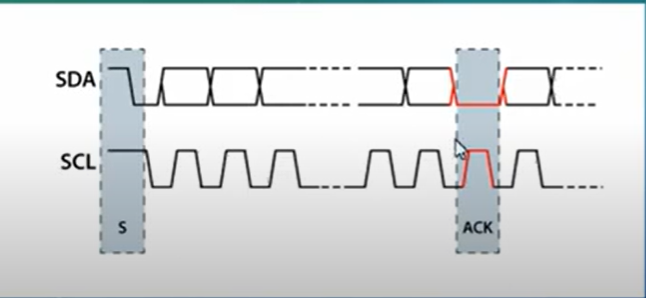
* + Master sẽ truyền bít S (Start)
  + Sau bit S là 7 bit địa chỉ (A0->A6) và 1 bit cấp phép sử dụng data(R/~W) của slave mà master muốn truy cập
    - R/~W: 1-> data read/ 0-> data write
  + Master sẽ truyền data qua 8bit (D0 -> D7) sau khi nhận được bit xác nhận (ACK) từ slave
  + Slave xác nhận việc truyền dữ liệu hoàn tất bằng bit ACK
  + Master sẽ kết thúc bằng bit p(stop)
* I2C parameters
  + Start/Stop condition



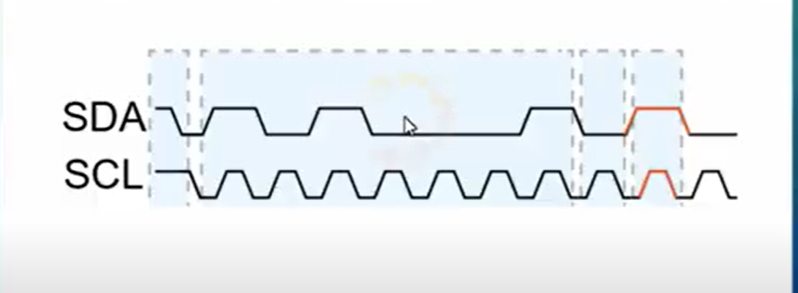
* I2c address phase



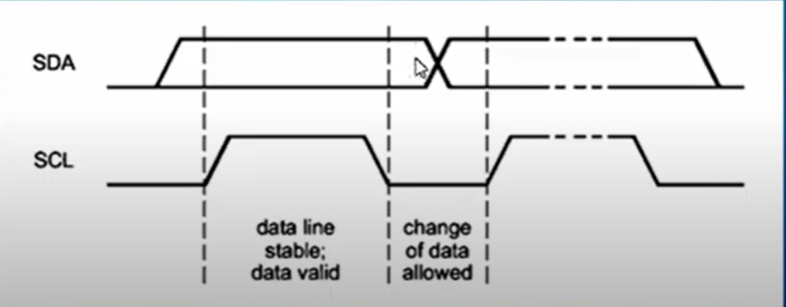
* I2C ack/nack
  + Sau khi truyền được 8bit master sẽ giải phòng SDA
  + Sau khi truyền được 8 bit master sẽ đời chu kỳ xung nhịp thứ 9 nếu slave kéo SDA xuống thấp thi master sẽ coi đó là sự xác nhận còn không thì sẽ không xác nhận
  + Ack



* + Nack



* I2C data validity



* Data được chuyền ổn định khi scl mức cao
* Data được thay đổi khi scl mức thấp (data không được nhận)
* I2c write operator



* + Khi truyền nhiều hơn 1byte dữ liệu thì sau bit ACK thứ 2 master sẽ ko truyền bít stop(P) mà tiếp tục truyền dữ liệu tiếp
* I2c read operator



* + Tương tự với read
* I2c repeat start
  + Vấn đề này sảy ra khi có nhiều master cùng kết nối với 1 slave-

Cấu hình i2c

Write

Read

Start bit

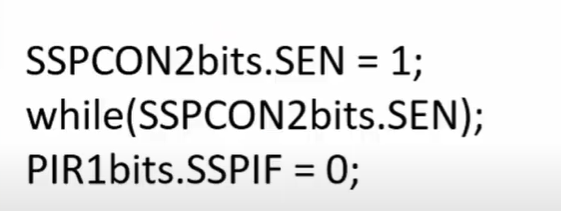
Stop bit

Restart

Ack

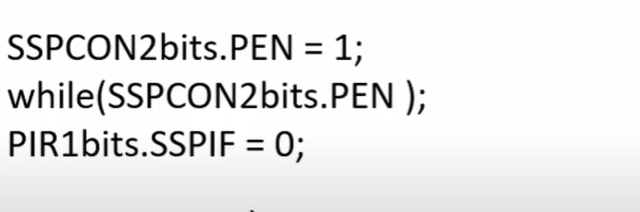
Nack

Start bit

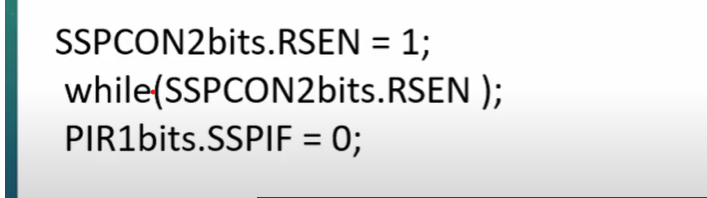


* Sen là bit cho phép
* = 1 khởi tạo trên chân sda và tự động được clear bởi phần cứng
* = 0
* Sspif

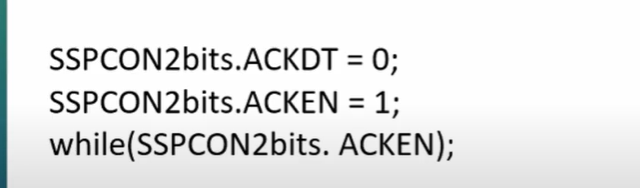
Stop bit



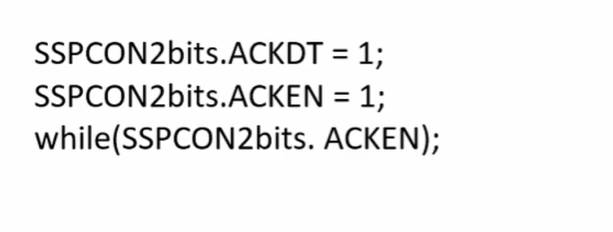
Repeat start



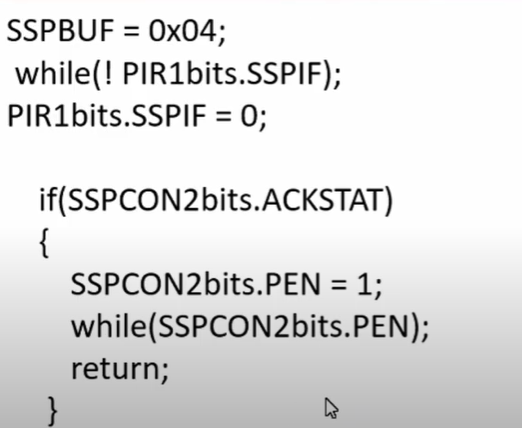
Ack



N ack



Write



Read

