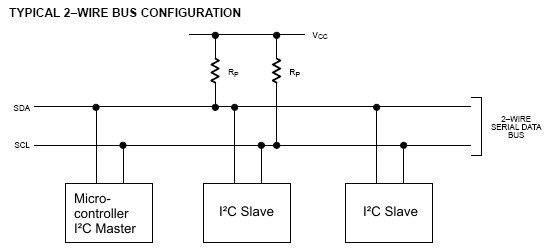
Bài 6: I2C Sofware & I2C Hardware

I. I2C Software

# Cấu hình GPIO

I2C chỉ sử dụng hai dây để truyền dữ liệu giữa các thiết bị:

* SDA (Serial Data) - đường truyền cho master và slave để gửi và nhận dữ liệu.
* SCL (Serial Clock) - đường mang tín hiệu xung nhịp.



Các chân có thể vừa là ngõ vào, vừa là ngõ ra tùy thuộc vào Master và slave khi truyền.

#define I2C\_SCL GPIO\_Pin\_6

#define I2C\_SDA GPIO\_Pin\_7

#define I2C\_GPIO GPIOB

void RCC\_Config(){

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2, ENABLE);

}

void GPIO\_Config(){

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_OD;

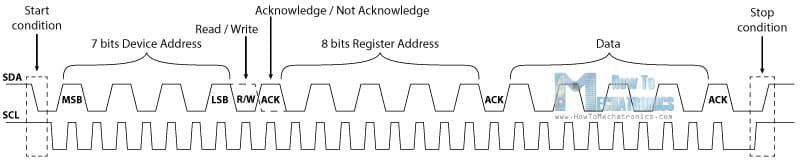
GPIO\_InitStructure.GPIO\_Pin = I2C\_SDA| I2C\_SCL;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(I2C\_GPIO, &GPIO\_InitStructure);

}

# I2C Software code



Ở trạng thái không truyền, các đừng SDA, SCL đều ở mức cao:

void I2C\_Config(){

WRITE\_SDA\_1;

delay\_us(1);

WRITE\_SCL\_1;

delay\_us(1);

}

Massage bắt đầu với 1 tín hiệu start: SDA được kéo xuống 0 trong khi SCL vẫn mức 1. Các macro dưới đây nhằm mục đích đơn giản hóa việc gọi hàm.

#define WRITE\_SDA\_0 GPIO\_ResetBits(I2C\_GPIO, I2C\_SDA)

#define WRITE\_SDA\_1 GPIO\_SetBits(I2C\_GPIO, I2C\_SDA)

#define WRITE\_SCL\_0 GPIO\_ResetBits(I2C\_GPIO, I2C\_SCL)

#define WRITE\_SCL\_1 GPIO\_SetBits(I2C\_GPIO, I2C\_SCL)

#define READ\_SDA\_VAL GPIO\_ReadInputDataBit(I2C\_GPIO, I2C\_SDA)

Frame truyền bắt đầu bằng tín hiệu Start, Start: SDA kéo xuống 0 trước SCL 1 khoảng delay nhỏ. Kết thúc bằng tín hiệu stop: SCL kéo lên 1 trước SDA 1 khoảng delay nhỏ.

void I2C\_Start(){

WRITE\_SCL\_1;

delay\_us(3);

WRITE\_SDA\_1;

delay\_us(3);

WRITE\_SDA\_0; //SDA reset to 0 before SCL.

delay\_us(3);

WRITE\_SCL\_0;

delay\_us(3);

}

void I2C\_Stop(){

WRITE\_SDA\_0;

delay\_us(3);

WRITE\_SCL\_1; //SCL set to 1 before SDA.

delay\_us(3);

WRITE\_SDA\_1;

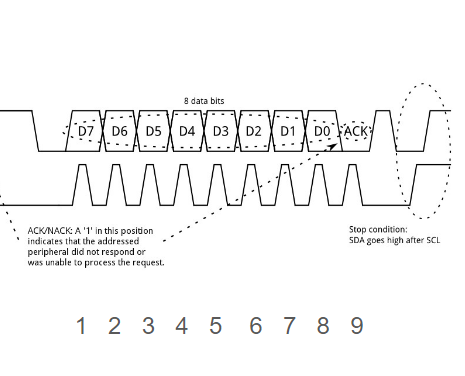
delay\_us(3);

}

### Hàm truyền:

Hàm truyền sẽ truyền lần lượt 8 bit trong byte dữ liệu.

* + Truyền 1 bit.
  + Tạo clock;
  + Dịch 1 bit.
* Chờ nhận lại ACK ở xung thứ 9.



status I2C\_Write(uint8\_t u8Data){

uint8\_t i;

status stRet;

for(int i=0; i< 8; i++){ //Write byte data.

if (u8Data & 0x80) {

WRITE\_SDA\_1;

} else {

WRITE\_SDA\_0;

}

delay\_us(3);

WRITE\_SCL\_1;

delay\_us(5);

WRITE\_SCL\_0;

delay\_us(2);

u8Data <<= 1;

}

WRITE\_SDA\_1;

delay\_us(3);

WRITE\_SCL\_1; //

delay\_us(3);

if (READ\_SDA\_VAL) {

stRet = NOT\_OK;

} else {

stRet = OK;

}

delay\_us(2);

WRITE\_SCL\_0;

delay\_us(5);

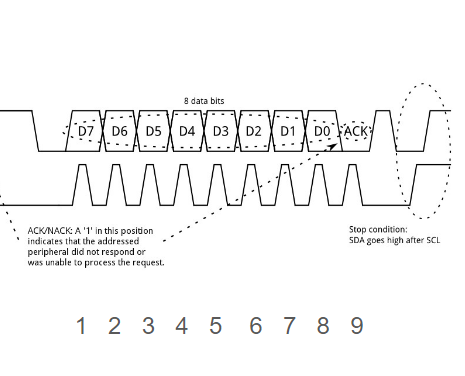
return stRet;

}

### Hàm nhận

Hàm nhận sẽ nhận lần lượt 8 bit trong byte dữ liệu.

* Kéo SDA lên 1
  + Đọc data trên SDA, ghi vào biến.
  + Dịch 1 bit.
* Gửi lại 1 tín hiệu ACK ở xung thứ 9.



uint8\_t I2C\_Read(ACK\_Bit \_ACK){

uint8\_t i;

uint8\_t u8Ret = 0x00;

WRITE\_SDA\_1;

delay\_us(3);

for (i = 0; i < 8; ++i) {

u8Ret <<= 1;

WRITE\_SCL\_1;

delay\_us(3);

if (READ\_SDA\_VAL) {

u8Ret |= 0x01;

}

delay\_us(2);

WRITE\_SCL\_0;

delay\_us(5);

}

if (\_ACK) {

WRITE\_SDA\_0;

} else {

WRITE\_SDA\_1;

}

delay\_us(3);

WRITE\_SCL\_1;

delay\_us(5);

WRITE\_SCL\_0;

delay\_us(5);

return u8Ret;

}

II. I2C Hardware

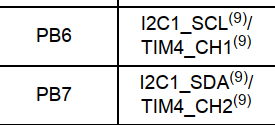
# Cấu hình GPIO

Các chân của từng bộ I2C được cài đặt sẵn trong datasheet.

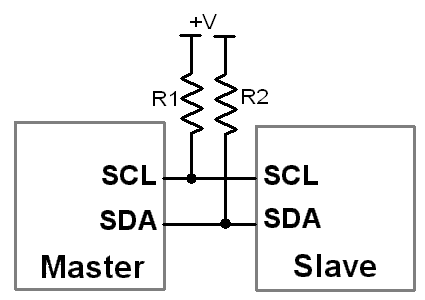
#define I2C\_SCL GPIO\_Pin\_6

#define I2C\_SDA GPIO\_Pin\_7

#define I2C1\_GPIO GPIOB



* SCL: Output
* SDA: Input/Output
* Vì có trở kéo lên nên sẽ hoạt động ở chế độ OD.



void GPIO\_Config(void) {

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6 | GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_OD;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

}

# Cấu hình I2C.

Tương tự các ngoại vi khác, các tham số I2C được cấu hình trong Struct I2C\_InitTypeDef:

* I2C\_Mode: Cấu hình chế độ hoạt động cho I2C:
  + I2C\_Mode\_I2C: Chế độ I2C FM(Fast Mode);
  + I2C\_Mode\_SMBusDevice&I2C\_Mode\_SMBusHost: Chế độ SM(Slow Mode).
* I2C\_ClockSpeed: Cấu hình clock cho I2C, tối đa 100khz với SM và 400khz ở FM.
* I2C\_DutyCycle: Cấu hình chu kì nhiệm vụ của xung:
  + I2C\_DutyCycle\_2: Thời gian xung thấp/ xung cao =2;
  + I2C\_DutyCycle\_16\_9: Thời gian xung thấp/ xung cao =16/9;
* I2C\_OwnAddress1: Cấu hình địa chỉ slave.
* I2C\_Ack: Cấu hình ACK, có sử dụng ACK hay không.
* I2C\_AcknowledgedAddress: Cấu hình số bit địa chỉ. 7 hoặc 10 bit
* Hàm **I2C\_Send7bitAddress(I2C\_TypeDef\* I2Cx, uint8\_t Address, uint8\_t I2C\_Direction)**, gửi đi 7 bit address để xác định slave cần giao tiếp. Hướng truyền được xác định bởi I2C\_Direction để thêm bit RW.
* Hàm **I2C\_SendData(I2C\_TypeDef\* I2Cx, uint8\_t Data)** gửi đi 8 bit data.
* Hàm **I2C\_ReceiveData(I2C\_TypeDef\* I2Cx)** trả về 8 bit data.
* Hàm **I2C\_CheckEvent(I2C\_TypeDef\* I2Cx, uint32\_t I2C\_EVENT)** trả về kết quả kiểm tra I2C\_EVENT tương ứng:
* Hàm **I2C\_CheckEvent(I2C\_TypeDef\* I2Cx, uint32\_t I2C\_EVENT)** trả về kết quả kiểm tra **I2C\_EVENT** tương ứng:
  + **I2C\_EVENT\_MASTER\_MODE\_SELECT**: Đợi Bus I2C về chế độ rảnh.
  + **I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED**: Đợi xác nhận của Slave với yêu cầu ghi của Master.
  + **I2C\_EVENT\_MASTER\_RECEIVER\_MODE\_SELECTED**: Đợi xác nhận của Slave với yêu cầu đọc của Master.
  + **I2C\_EVENT\_MASTER\_BYTE\_TRANSMITTED**: Đợi truyền xong 1 byte data từ Master.
  + **I2C\_EVENT\_MASTER\_BYTE\_RECEIVED**: Đợi Master nhận đủ 1 byte data.

Bắt đầu truyền nhận, bộ I2C sẽ tạo 1 tín hiệu start. Đợi tín hiệu báo Bus sẵn sàng.

* Gửi 7 bit địa chỉ để xác định slave. Đợi Slave xác nhận.

I2C\_GenerateSTART(I2C1, ENABLE);

//Waiting for flag

**while**(!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_MODE\_SELECT));

I2C\_Send7bitAddress(I2C1, 0x44, I2C\_Direction\_Transmitter);

//And check the transmitting

**while**(!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_TRANSMITTER\_MODE\_SELECTED));

* Gửi/đọc các byte data, Đợi truyền xong.

void Send\_I2C\_Data(uint8\_t data)

{

I2C\_SendData(I2C1, data);

// wait for the data trasnmitted flag

while(!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_BYTE\_TRANSMITTED));

}

uint8\_t Read\_I2C\_Data(){

uint8\_t data = I2C\_ReceiveData(I2C1);

while(!I2C\_CheckEvent(I2C1, I2C\_EVENT\_MASTER\_BYTE\_RECEIVED));

return data;

}

* Sau đó kết thúc bằng tín hiệu stop.

