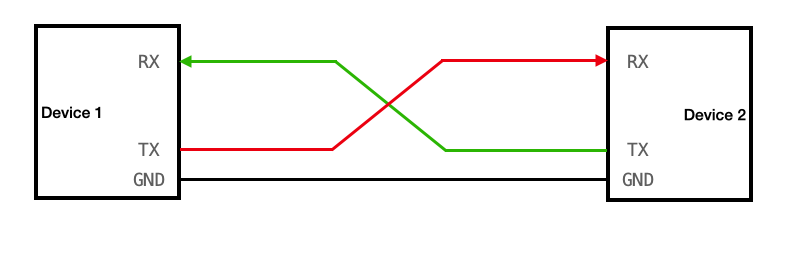
Bài 7: UART Software & UART Hardware

I. UART Software

Các dòng vi điều khiển đều tích hợp bộ truyền thông UART, trong bài này, chúng ta sẽ mô phỏng giao thức UART bằng GPIO.

# Cấu hình GPIO

UART chỉ sử dụng 2 chân để truyền, đó là TX và RX.



Xác định các chân sử dụng cho UART là bước đầu tiên. UART soft không yêu cầu các chân cụ thể nên ta có thể sử dụng chân bất kì:

#define TX\_Pin GPIO\_Pin\_9

#define RX\_Pin GPIO\_Pin\_10

#define UART\_GPIO GPIOA

TX là chân Transmit nên được cấu hình OUTPUT, RX là chân Receive nên sẽ được cấu hình INPUT.

void GPIO\_Config(){

GPIO\_InitTypeDef GPIO\_InitStructure;

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

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

GPIO\_InitStructure.GPIO\_Pin = TX\_Pin;

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

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

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

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_InitStructure.GPIO\_Pin = RX\_Pin;

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

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

}

# UART Software code

Tốc độ baudrate được xác định bởi thời gian truyền đi 1 bit. Ở bài này ta dùng tốc độ phổ thông 9600, ứng với mỗi bit là 105us.

//Baaurate = 9600bits/s >> 0.10467ms for 1 bit = 104,67 ús

//=>> time delay ~~105 us

#define BRateTime 105

Ở chế độ nghỉ (Không truyền), đường TX sẽ được giữ ở mức cao. hàm UART\_Config() thiết lập chế độ nghỉ cho đường truyền:

void UART\_Config(){

GPIO\_SetBits(UART\_GPIO, TX\_Pin);

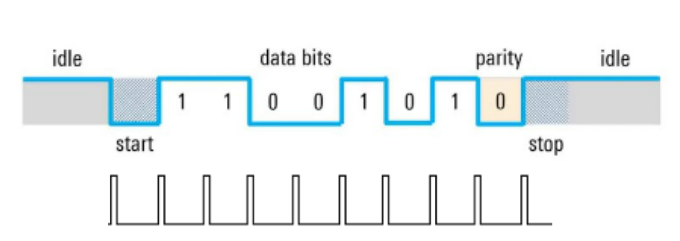
delay\_us(1);

}

### Hàm truyền:

Hàm truyền sẽ truyền lần lượt 8 bit trong byte dữ liệu, sau khi tín hiệu start được gửi đi.

* Tạo start, delay 1 period time.
  + Truyền bit dữ liệu. mỗi bi truyền trong 1 period time.
  + Dịch 1 bit.
* Tạo stop, delay tương ứng với số bit stop



void UART\_Transmit(const char DataValue)

{

// Send Start Bit

GPIO\_WriteBit(UART\_GPIO, RX\_Pin, Bit\_RESET);

delay\_us(BRateTime);

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

if( ((DataValue>>i)&0x1) == 0x1 ){

GPIO\_WriteBit(UART\_GPIO, RX\_Pin, Bit\_SET);

} else{

GPIO\_WriteBit(UART\_GPIO, RX\_Pin, Bit\_RESET);

}

delay\_us(BRateTime);

}

// Send Stop Bit

GPIO\_WriteBit(UART\_GPIO, RX\_Pin, Bit\_SET);

delay\_us(BRateTime);

}

Data truyền đi sẽ được thêm bit parity tùy theo cấu hình parity bit là chẵn/lẻ hay không dùng pairty bit:

typedef enum{

Parity\_Mode\_NONE,

Parity\_Mode\_ODD,

Parity\_Mode\_EVENT

}Parity\_Mode;

Có thể tạo bit parity bằng cách đếm số bit 1, sau đó thêm vào cuối chuỗi bit bit 0 hoặc 1 tương ứng:

uint8\_t Parity\_Generate(uint8\_t data, Parity\_Mode Mode){

uint8\_t count =0;

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

if(data & 0x01){

count++;

}

data>>=1;

}

switch(Mode){

case Parity\_Mode\_NONE:

return data;

break;

case Parity\_Mode\_ODD:

if(count%2){

return ((data<<1)|1);

} else {

return (data<<1);

}

break;

case Parity\_Mode\_EVENT:

if(!(count%2)){

return ((data<<1)|1);

} else {

return (data<<1);

}

break;

default:

return data;

break;

}

}

### Hàm nhận:

Hàm nhận sẽ nhận lần lượt 8 bit

* Chờ tín hiệu start từ thiết bị gửi.
* Delay 1,5 period time.
  + Đọc data trên RX, ghi vào biến.
  + Dịch 1 bit.
  + Delay 1 period time.
* Delay 0,5 period time và đợi stop bit.

unsigned char UART\_Receive(void)

{

unsigned char DataValue = 0;

while(GPIO\_ReadInputDataBit(UART\_GPIO, RX\_Pin) == 1);

delay\_us(BRateTime);

delay\_us(BRateTime/2);

for ( unsigned char i = 0; i < 8; i++ )

{

if ( GPIO\_ReadInputDataBit(UART\_GPIO, RX\_Pin) == 1 ){

DataValue += (1<<i);

}

delay\_us(BRateTime);

}

if ( GPIO\_ReadInputDataBit(UART\_GPIO, RX\_Pin) == 1 ){

delay\_us(BRateTime/2);

return DataValue;

}

}

Sau khi nhận được data, có thể tiến hành kiểm tra chẵn/lẻ:

uint8\_t Parity\_Check(uint8\_t data, Parity\_Mode Mode){

uint8\_t count =0;

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

if(data & 0x01){

count++;

}

data>>=1;

}

switch(Mode){

case Parity\_Mode\_NONE:

return 1;

break;

case Parity\_Mode\_ODD:

return (count%2);

break;

case Parity\_Mode\_EVENT:

return (!(count%2));

break;

default:

return 0;

break;

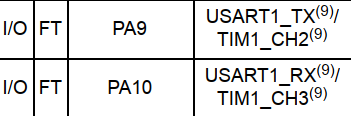
}

}

# 

II. UART Hardware

# Cấu hình GPIO



Các bộ UART trong STM32F1 được xác định sẵn các chân GPIO

Tương tự Software, TX sẽ là UOTPUT và RX sẽ là INPUT.

void GPIO\_Config(){

GPIO\_InitTypeDef GPIOInitStruct;

GPIOInitStruct.GPIO\_Pin = GPIO\_Pin\_10;

GPIOInitStruct.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOA, &GPIOInitStruct);

//

GPIOInitStruct.GPIO\_Pin = GPIO\_Pin\_9;

GPIOInitStruct.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIOInitStruct.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_Init(GPIOA, &GPIOInitStruct);

}

# Cấu hình UART

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

* USART\_Mode: Cấu hình chế độ hoạt động cho UART:
  + USART\_Mode\_Rx: Cấu hình truyền.
  + USART\_Mode\_Tx: Cấu hình nhận.
  + Có thể cấu hình cả 2 cùng lúc.
* USART\_BaudRate: Cấu hình tốc độ baudrate cho uart.
* USART\_HardwareFlowControl: Cấu hình chế độ bắt tay cho uart.
* USART\_WordLength: Cấu hình số bit mỗi lần truyền.
* USART\_StopBits: Cấu hình số lượng stopbits.
* USART\_Parity: cấu hình bit kiểm tra chẳn, lẻ.

void UART\_Config(){

//Usart

USARTInitStruct.USART\_BaudRate = 9600;

USARTInitStruct.USART\_WordLength = USART\_WordLength\_8b;

USARTInitStruct.USART\_StopBits = USART\_StopBits\_1;

USARTInitStruct.USART\_Parity = USART\_Parity\_No;

USARTInitStruct.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USARTInitStruct.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

USART\_Init(USART1, &USARTInitStruct);

USART\_Cmd(USART1,ENABLE);

}

Hàm **USART\_SendData(USART\_TypeDef\* USARTx, uint16\_t Data)**, truyền data từ UARTx. Data này đã được thêm bit chẵn/lẻ tùy cấu hình.

Hàm **USART\_ReceiveData(USART\_TypeDef\* USARTx)**, nhận data từ UARTx.

Hàm **USART\_GetFlagStatus(USART\_TypeDef\* USARTx, uint16\_t USART\_FLAG)** trả về trạng thái cờ USART\_FLAG tương ứng:

* + **USART\_FLAG\_TXE**: Cờ truyền, set lên 1 nếu quá trình truyền hoàn tất.
  + **USART\_FLAG\_RXNE**: Cờ nhận, set lên 1 nếu quá trình nhận hoàn tất.
  + **USART\_FLAG\_IDLE**: Cờ báo đường truyền đang ở chế độ Idle.
  + **USART\_FLAG\_PE**: Cờ báo lỗi Parity.

Quá trình truyền/nhận có thể mô tả như sau:

* Bắt đầu truyền/nhận, UART xóa hết data trong thanh ghi DR để đảm bảo data đúng.
* Truyền: Gửi đi từng byte data. Sau đó đợi cờ TXE bật lên.
* Nhận: Đọc data từ bộ UART, chờ cờ RNXE bật lên.
* Đối với mảng dữ liệu, lặp lại quá trình cho từng byte.