Tutorial 3 - UART

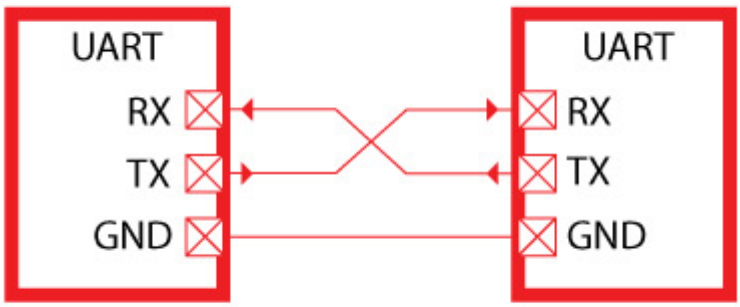
Author: Peter Tse
Contact: hntse@ust.hk

Introduction

UART, which stands for **Universal Asynchronous Receiver-Transmitter**, is a protocol for serial communication. We use UART for general data transmission, such as in Bluetooth. You actually are using UART at the beginning - you are using UART1 to flash your program all this time.

Network

There are usually 3 pins for UART communication.



- **TX**: Transmit, used for sending data
- **RX**: Receive, used for receiving data
- **GND**: Ground

There is probably also **VCC** (power supply) for the UART module as well.

Do note that **TX** should connect to **RX** and vice versa.

The data transfer speed, also known as **baud rate**, is the numbers of bits that can be transferred in a second. The higher the number, the greater the speed. In robotics team, we usually use 115200 as our baud rate.

During transmission, data are splits into **bytes** (groups of 8 bits). You don't need to worry about this when you are sending data since our library would already split up the data, yet when you are receiving data, since you do not know the structure of the data beforehand, you could only receive data byte by byte.

UART in STM32F1

UART Initialization

Just like other modules in the board, you need to initialize the UART before using it.

Prototype:

```
// COM: COM port; br: baud rate
void uart_init(COM_TypeDef COM, u32 br);
```

Usage:

```
uart_init(COM3, 115200);
```

Note the definition of `COM_TypeDef`:

```
typedef enum
{
    COM_NULL = -1,       //disabled
    COM1 = 0,            //UART1
    COM3 = 1,            //UART2
} COM_TypeDef;
```

It support using COM1 and COM3 but since you are using COM1 to flash your program, we can only use COM3 for Bluetooth communication.

Sending Data

Prototype:

```
// COM: COM port; tx_buf, ...: string format (same as printf(...))
void uart_tx(COM_TypeDef COM, const char * tx_buf, ...);
```

Usage:

```
uart_tx(COM3, "Hello"); // sends the string "Hello"
uart_tx(COM3, "%d", 3); // sends the number 3
```

Receiving Data

Note that when the board is receiving data, one can only receive it byte by byte. Since the board can receive data anytime during its program execution, it is impractical to have something like this.

```
u8 data; // u8 is equivalent to one byte
while (1) {
    data = uart_rx_byte(COM3); // for reading one single byte in COM3
    /* ... */
}
```

Since the function `uart_rx_byte()` is not run immediately when data is received, data may be delayed and even lost. Therefore, to resolve this issue, we introduce the concept of **interrupt** and **listener**.

Interrupts and listeners

We can configure the board such that as data is received, we pause the current program execution and directly jumps to the functions of reading the data. Such mechanisms is called **interrupts** and the functions for data reading is called **listeners**. Here is how to setup the interrupts and listeners.

Prototype:

```
typedef void on_receive_listener(const uint8_t byte);
// COM: COM port, listener: listener function with parameter const uint8_t and return type void
void uart_interrupt_init(COM_TypeDef COM, on_receive_listener *listener);
```

Usage:

- Define listener function outside main scope.

```
// Listener functions for handling data received during interrupts
// Function name does not matter
void UARTListener(const uint8_t byte) {
    /* ... */
}
```

- Initialize the listener function inside main scope.

```
// sets up UARTListener to be the listener function for COM3
uart_interrupt_init(COM3, UARTListener);
// Side note:
// uart_interrupt_init(COM3, &UARTListener);
// would work as well (note the &)
```

UART in Windows

Bluetooth

You will be using Bluetooth connection between the robot and your laptop in your internal competition. This section is about how to setup Bluetooth UART connection with your computer.

1. Connect the Bluetooth to the mainboard.
2. Connect your laptop to the Bluetooth.
3. Go to 'More Bluetooth options'.
4. Select 'COM Ports' tab and see which port is stated 'Outgoing'.
5. Note this port and use it in Coolterm.
6. In Coolterm, go to 'Options/Serial Port' and select the correct port and baud rate (115200).
7. You may want to turn on 'Local Echo' in 'Options/Terminal'
8. You can now send data to the board using UART.

TTL

You can also directly connect to the UART pins in the board through TTL. In this case, the UART of the board connects to one of your USB ports.

1. Go to 'Device Manager' to check your COM port.
2. Repeat step 5-8 in Bluetooth section.

Classwork

Echoing

You may use the following task to test if you are able to communicate with the MCU properly. In this task, you will be echoing data inputted through your computer (e.g. Coolterm) and TTL.

Implement that whenever a character is inputted through Coolterm (or other software), return a text like in the sample I/O provided below. (lines start with `>` are outputs from MCU, otherwise they are inputs)

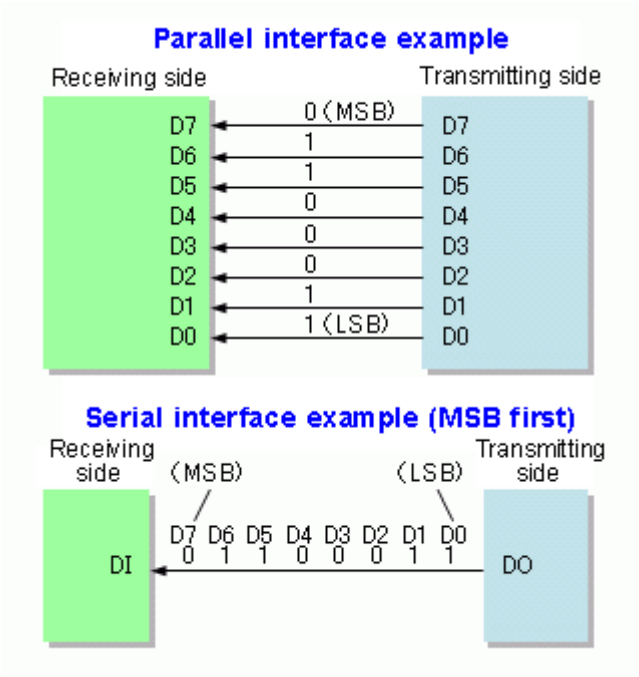
```
c
>You entered: c
#
>You entered: #
```

Extra Information

Here are some information about communication protocols

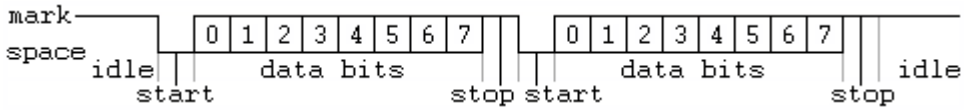
- Serial/Parallel communication

  It is the process of sending data one bit at a time sequentially. It is in contrast with *parallel* communication in which data are sent in parallel in multiple ports.



- Asynchronous/Synchronous

It is a method of serial communication in which when data is sent, a *START* bit and a *STOP* bit are sent, which is in contrast to the *synchronous* serial communication in which a *CLOCK* is used to control the data cycle.

Method	Image
Asynchronous	
Synchronous