Embedded System Design with MCU and FPGA

LAB04 0858706 黃品嫚

Goal

Let us know more about serial communications.

Problems

1. List 6 or more functions in the Serial library. Explain their usage.

- Serial Library is used for communication between the Arduino board and a computer or other devices.
- There are 21 functions in the Serial Library, I only list six functions which are more commonly used: [1]
  - **begin()**: sets the data rate in bits per second (baud) for serial data transmission.

    syntax:

    ```
    Serial.begin(speed)
    Serial.begin(speed, config)
    ```

    example:

    ```
    void setup() {
        Serial.begin(9600);
    }
    ```
  - **end()**: disables serial communication, allowing the RX and TX pins to be used for general input and output.

    syntax:

    ```
    Serial.end()
    ```
  - **print()**: prints data to the serial port as human-readable ASCII text.

    syntax:

    ```
    Serial.print(val)
    Serial.print(val, format)
    ```

    example:

    ```
    void loop() {
        Serial.print("For Example.");
    }
    ```
  - **println()**: prints data to the serial port as human-readable ASCII text followed by a carriage return character and a newline character.

    syntax:

    ```
    Serial.println(val)
    Serial.println(val, format)
    ```

    example:

    ```
    void loop() {
    ```

```
        Serial.println("For Example.");
    }
```

- **read()**: reads incoming serial data.

  syntax:

```
    Serial.read()
```

  example:

```
    void loop() {
        incomingByte = Serial.read();
    }
```

- **write()**: writes binary data to the serial port.

  syntax:

```
    Serial.write(val)
    Serial.write(str)
    Serial.write(buf, len)
```

  example:

```
    void loop() {
        Serial.write(45);
        Serial.write("hello");
    }
```

2. List 3 or more functions in the SPI library. Explain their usage.

- SPI is used to communicate with one or more peripheral devices quickly over short distances.
- There are 10 functions in the SPI Library: (the functions in bold are more commonly used) [2]
  - SPISettings: is used to configure the SPI port for our SPI device.
  - **begin()**: initializes the SPI bus by setting SCK, MOSI, and SS to outputs, pulling SCK and MOSI low, and SS high.
  - **end()**: disables the SPI bus.
  - beginTransaction(): initializes the SPI bus using the defined SPISettings.
  - endTransaction(): stop using the SPI bus. Normally this is called after de-asserting the chip select, to allow other libraries to use the SPI bus.
  - **setBitOrder()**: sets the order of the bits shifted out of and into the SPI bus, either LSBFIRST or MSBFIRST.
  - **setClockDivider()**: sets the SPI clock divider relative to the system clock.

    parameters:

    (1) SPI_CLOCK_DIV4
    (2) SPI_CLOCK_DIV8
    (3) SPI_CLOCK_DIV16
    (4) SPI_CLOCK_DIV32
    (5) SPI_CLOCK_DIV64
    (6) SPI_CLOCK_DIV128

- **setDataMode()**: sets the SPI data mode: that is, clock polarity and phase.
- **transfer()**: SPI transfer is based on a simultaneous send and receive; the received data is returned in receivedVal.
- usingInterrupt(): this allows SPI.beginTransaction() to prevent usage conflicts.

3. List 6 or more functions in the Wire library. Explain their usage.

- Wire Library allows us to communicate with I2C / TWI devices.
- There are 10 functions in the Wire Library: [3]
  - **begin()**: initiate the Wire library and join the I2C bus as a master or slave. This should normally be called only once.
  - **requestFrom()**: used by the master to request bytes from a slave device. The bytes may then be retrieved with the available() and read() functions.
  - **beginTransmission()**: begin transmission to the I2C slave device with the given address.
  - **endTransmission()**: ends a transmission to a slave device that was begun by beginTransmission() and transmits the bytes that were queued by write().
  - **write()**: writes data from a slave device in response to a request from a master
  - **available()**: returns the number of bytes available for retrieval with read().
  - **read()**: reads a byte that was transmitted from a slave device to a master after a call to requestFrom() or was transmitted from a master to a slave. read() inherits from the Stream utility class.
  - **SetClock()**: this function modifies the clock frequency for I2C communication.
  - **onReceive()**: registers a function to be called when a slave device receives a transmission from a master.
  - **onRequest()**: register a function to be called when a master requests data from this slave device.

4. Which protocol could support the longest distance communication?

- In this week's class, we learned three types of communication, SPI, I2C, and UART. First, I will compare these three communication methods. [5]

Features	UART	SPI	I2C
Full Name	Universal Asynchronous Receiver/Transmitter	Serial Peripheral Interface	Inter-Integrated Circuit
Distance	Lower about 50 feet	Highest	Higher
Type of Communication	Asynchronous	Synchronous	Synchronous
Number of masters	Not Application	One	One or more than One
Hardware complexity	Lesser	Less	More
Data Rate	Maximum data rate supported is about 230 Kbps to 460kbps.	Usually supports about 10 Mbps to 20 Mbps	I2C supports 100 kbps, 400 kbps, 3.4 Mbps. Some variants also supports 10 Kbps and 1 Mbps.

- From the above table, we could know that SPI has the longest distance communication. But all of these methods have a serious drawback: they do not work at large distances. There is a way to solve this problem, connecting two Arduino boards using a long cable and **RS-485** interface. [4]
- The maximum range of the RS-485 module is 1200 meters. The transmission speed over such a long wire will be about 60 kb/s, which is a relatively good speed for transmitting the sensor's data. RS-485 supports a data transfer rate of 30Mbps maximum.

Features	UART	SPI	I2C
Full Name	Universal Asynchronous Receiver/Transmitter	Serial Peripheral Interface	Inter-Integrated Circuit
Distance	Lower about 50 feet	Highest	Higher
Type of Communication	Asynchronous	Synchronous	Synchronous
Number of masters	Not Application	One	One or more than One
Hardware complexity	Lesser	Less	More
Data Rate	Maximum data rate supported is about 230 Kbps to 460kbps.	Usually supports about 10 Mbps to 20 Mbps	I2C supports 100 kbps, 400 kbps, 3.4 Mbps. Some variants also supports 10 Kbps and 1 Mbps.

5. Which protocol could support the fastest communication?

- From the above table, we could know that SPI supports the fastest communication.
- However, the RS-485 supports a higher data transfer rate of 30Mbps maximum.

6. Which protocol could support the largest amount of devices?

Features	UART	SPI	I2C
Full Name	Universal Asynchronous Receiver/Transmitter	Serial Peripheral Interface	Inter-Integrated Circuit
Distance	Lower about 50 feet	Highest	Higher
Type of Communication	Asynchronous	Synchronous	Synchronous
Number of masters	Not Application	One	One or more than One
Hardware complexity	Lesser	Less	More
Data Rate	Maximum data rate supported is about 230 Kbps to 460kbps.	Usually supports about 10 Mbps to 20 Mbps	I2C supports 100 kbps, 400 kbps, 3.4 Mbps. Some variants also supports 10 Kbps and 1 Mbps.

- From the above table, we could know that I2C supports the largest of devices.

Reference

[1] https://www.arduino.cc/reference/en/language/functions/communication/serial/
[2] https://www.arduino.cc/en/Reference/SPI
[3] https://www.arduino.cc/en/Reference/Wire
[4] https://circuitdigest.com/microcontroller-projects/rs485-serial-communication-between-arduino-uno-and-arduino-nano
[5] https://www.rfwireless-world.com/Terminology/UART-vs-SPI-vs-I2C.html