Arduino wire.h(IIC) library

Wire.begin()

函数

Wire.begin(address)

说明

对库进行初始化，加入I2C总线作为从机或主机。 这通常只被调用一次。

begin()：无输入参数，表示以 Master 形式加入总线。

begin(address)：有输入参数，表示以从机形式加入总线，设备地址为address（7-bit)

参数

address：7位从机地址（可选），如果不详细描述（不进行设定），加入的总线被看做主机。

返回值

无

Wire.beginTransmission(address)

说明

根据已给地址，开始向I2C的从机进行传输。随后，调用函数 write() 对传输的字节进行排列，调用函数 endTransmission() 进行传输 。

参数

address：传输指向的设备的7位比特地址

返回值

无

注释

beginTransmission 函数用于启动一次 Master write to Slave 操作。值得注意的是这个函数的调用并**不会产生**

Start 信号 和发送 Slave Address，仅是实现通知 Arduino后面要启动 Master write to Slave 操作。

beginTransmission 函数调用后，（再调用 write 函数进行数据写入）， 最后再调用 endTransmission 函数方能

产生 Start 信号 和发送 Slave Address 及通讯时序。

Wire.endTransmission()

说明

停止对从机的传输，传输开始时调用 beginTransmission()，传输的字节由 write() 排列。

对于1.0.1的Arduino，为了和某些I2C设备兼容，requestFrom()接收一个布尔类型参数并改变其行为。

如果为真（true）时，requestFrom()在请求之后发送一个停止的指令，释放I2C总线。

如果为假（false）时，requestFrom()在请求之后发送一个重新启动的指令。总线不会被释放，这可以防止其他主机同时也发送请求。这允许一个主设备，在被控制时可发送多个请求。

默认值为真（true）。

函数

Wire.endTransmission()
Wire.endTransmission(stop)

参数

stop：布尔值。为真（true）时将在请求后发送停止指令并释放总线。 为假（false）时将在请求后发送重新启动的指令，保持连接状态。

返回值

字节，它表示的传输的状态：

- 0：成功
- 1：数据太长，传送缓冲区溢出
- 2：传送地址时接收到NACK（非应答信号）
- 3：传送数据时接收到NACK（非应答信号）
- 4：其他错误

  有个地方需要注意的：当通讯过程中，出现异常后，异常后的 write 操作将被终止，直接结束通讯，具体的是否出现异常，只需要看 endTransmission 的返回值即可。

write()

说明

由从机中写入数据，回应主机的请求，或排列将要从主机传输给从机的字节（在beginTransimission()和endTransmission()中调用）。

函数

Wire.write(value)
Wire.write(string)
Wire.write(data,length)

参数

val: 以单个字节形式发送的值
str: 以一串字节的形式发送的字符串
data：以字节形式发送的数组
length：传输的字节数

返回值

byte:write()将返回写入的字节数，但是否读取这个返回值是可选的

Example:

```
#include <Wire.h>

byte val = 0;

void setup()
{
  Wire.begin(); // 初始化，加入I2C总线
}

void loop()
{
  Wire.beginTransmission(44); //开始传输， 发送 #44 (0x2c)
  //设备具体地址在数据表
     Wire.write(val);              // 以字节形式发送val值
  Wire.endTransmission();     // 停止传输

  val++;         // val值加一
  if(val == 64) // 当val到达最大值64时
  {
    val = 0;     // 从最小值0重新开始
  }
  delay(500);
}
```

Wire.requestFrom()

说明

主机中使用，向从机请求数据。这些字节可以通过以下两个函数进行检索available()和 read()

对于1.0.1的Arduino，为了和某些I2C设备兼容，requestFrom()接收一个布尔类型参数并改变其行为。

如果为真(true)时，requestFrom()在请求之后发送一个停止的指令，释放I2C总线。

如果为假true时，requestFrom()在请求之后发送一个重新启动的指令。总线不会被释放，这可以防止其他主机同时也发送的请求。这允许一个主设备，在被控制时可发送多个请求。

默认值为真(true)。

函数

Wire.requestFrom(address, quantity)
Wire.requestFrom(address, quantity, stop)

参数

address： 7比特地址，向该地址发送请求
quantity：请求的字节个数
stop：布尔值。为真（true）时将在请求后发送停止指令并释放总线。 为假（false）时将在请求后发送重新启动的指令，保持连接状态。

返回值

byte : the number of bytes returned from the slave device

Wire.available()

说明

当和函数receive()一起使用时，返回可被调用的字节数 。调用函数 requestFrom()后，可在主机中调用此函数。

或者在 onReceive()函数内部的从机中调用。

available 函数用于统计 Master Read From Slave 操作后， read 缓存区剩余的字节数。每当缓存区的数据被读走

1 个字节，available 函数的返回值减一。通常 available 函数会搭配着 read 函数使用。

参数

无

返回值

可读取的字节数。

read()

说明

读取一个由从机发送给主机，或由主机发送给从机的字节，在调用 requestFrom() 函数后。

read 函数用于在 Master Read From Slave 操作后，读取缓存区的数据。

Wire.read()

参数

无

返回值

返回下一个接收到的字节

Example

```
#include "Wire.h"

void setup()
{
  Wire.begin();         //加入I2C总线（地址可选为主机）
  Serial.begin(9600);  // 初始化串口输出
}

void loop()
{
  Wire.requestFrom(2, 6);     // 向从机 #2请求6个字节

  while(Wire.available())     // 从机发送的数据可以少于所请求的
  {
    char c = Wire.read();     // 接收一个字节，并设置为字符类型
```

```
    Serial.print(c);         // 串口打印该字符
  }

  delay(500);
}
```

Wire.onReceive(handler)

说明

注册一个函数，当从机接收到主机所发送的数据时调用

参数

handler：当从机接收到数据时被调用的函数。应该采取单精度整数型参数（主机读取的字节数），并无任何返回值，如： void MyHandler(int numBytes)

返回值

无

Wire.onRequest(handler)

说明

注册一个函数,当主机向这个从机请求数据时调用该函数

参数

handler：要调用的函数，不带任何参数，并且没有任何返回值，例如： void MyHandler()

返回值

无

示例代码

```
#include <Wire.h> // I2C head file

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  // Initiate the Wire library and join the I2C bus as a master or Slave.
  Wire.begin();
  Serial.print("Ready to Read TFmini\r\n");
  delay(10);
}

void loop() {
  // put your main code here, to run repeatedly:
  byte i = 0;
  byte rx_Num = 0;  // the bytes of received by I2C
  byte rx_buf[7] = {0}; // received buffer by I2C

  Wire.beginTransmission(7); // Begin a transmission to the I2C Slave device with the given address.
  Wire.write(1); // Reg's Address_H
```

```
  Wire.write(2); // Reg's Address_L
  Wire.write(7); // Data Length
  Wire.endTransmission(0);  // Send a START Sign

  // Wire.requestFrom（AA,BB）;receive the data form slave.
  // AA: Slave Address ; BB: Data Bytes
  rx_Num = Wire.requestFrom(0x07, 7);

  // Wire.available: Retuens the number of bytes available for retrieval
with read().
  while( Wire.available())
  {
      rx_buf[i] = Wire.read(); // received one byte
      i++;
  }

}

```