Wire Library

This library allows you to communicate with I2C / TWI devices. On most Arduino boards, SDA (data line) is on analog input pin 4, and SCL (clock line) is on analog input pin 5. On the Arduino Mega, SDA is digital pin 20 and SCL is 21.

As of Arduino 1.0, the library inherits from the Stream functions, making it consistent with other read/write libraries. Because of this, send() and receive() have been replaced with read() and write().

Functions

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- write()
- available()
- read()
- onReceive()
- onRequest()

Note

There are both 7- and 8-bit versions of I2C addresses. 7 bits identify the device, and the eighth bit determines if it's being written to or read from. The Wire library uses 7 bit addresses throughout. If you have a datasheet or sample code that uses 8 bit address, you'll want to drop the low bit (i.e. shift the value one bit to the right), yielding an address between 0 and 127.
Wire.begin()

Wire.begin(address)

**Description**
Initiate the Wire library and join the I2C bus as a master or slave. This should normally be called only once.

**Parameters**
address: the 7-bit slave address (optional); if not specified, join the bus as a master.

**Returns**
None
Wire.beginTransmission(address)

**Description**
Begin a transmission to the I2C slave device with the given address. Subsequently, queue bytes for transmission with the `write()` function and transmit them by calling `endTransmission()`.

**Parameters**
address: the 7-bit address of the device to transmit to

**Returns**
None

**See Also**
- Wire.write()
- Wire.endTransmission()
**Wire.endTransmission()**

**Description**

Ends a transmission to a slave device that was begun by `beginTransmission()` and transmits the bytes that were queued by `write()`.

As of Arduino 1.0.1, `endTransmission()` accepts a boolean argument changing its behavior for compatibility with certain I2C devices.

If true, `endTransmission()` sends a stop message after transmission, releasing the I2C bus.

If false, `endTransmission()` sends a restart message after transmission. The bus will not be released, which prevents another master device from transmitting between messages. This allows one master device to send multiple transmissions while in control.

The default value is true.

**Syntax**

`Wire.endTransmission()`  
`Wire.endTransmission(stop)`

**Parameters**

*stop*: boolean. true will send a stop message, releasing the bus after transmission. false will send a restart, keeping the connection active.

**Returns**

byte, which indicates the status of the transmission:  
- 0: success  
- 1: data too long to fit in transmit buffer  
- 2: received NACK on transmit of address  
- 3: received NACK on transmit of data  
- 4: other error

**See Also**

- `Wire.beginTransmission()`  
- `Wire.write()`
**Wire.requestFrom()**

**Description**

Used by the master to request bytes from a slave device. The bytes may then be retrieved with the `available()` and `read()` functions.

As of Arduino 1.0.1, `requestFrom()` accepts a boolean argument changing its behavior for compatibility with certain I2C devices.

If true, `requestFrom()` sends a stop message after the request, releasing the I2C bus.

If false, `requestFrom()` sends a restart message after the request. The bus will not be released, which prevents another master device from requesting between messages. This allows one master device to send multiple requests while in control.

The default value is true.

**Syntax**

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

**Parameters**

- **address**: the 7-bit address of the device to request bytes from

- **quantity**: the number of bytes to request

- **stop**: boolean. true will send a stop message after the request, releasing the bus. false will continually send a restart after the request, keeping the connection active.

**Returns**

None

**See Also**

- [Wire.available()](#)
- [Wire.read()](#)
**write()**

**Description**

Writes data from a slave device in response to a request from a master, or queues bytes for transmission from a master to slave device (in-between calls to `beginTransmission()` and `endTransmission()`).

**Syntax**

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

**Parameters**

- value: a value to send as a single byte
- string: a string to send as a series of bytes
- data: an array of data to send as bytes
- length: the number of bytes to transmit

**Returns**

byte: `write()` will return the number of bytes written, though reading that number is optional

**Example**

```cpp
#include <Wire.h>
byte val = 0;

void setup()
{
  Wire.begin(); // join i2c bus
}

void loop()
{
  Wire.beginTransmission(44); // transmit to device #44 (0x2c)
    // device address is specified in datasheet
  Wire.write(val); // sends value byte
  Wire.endTransmission(); // stop transmitting
  val++; // increment value
  if(val == 64) // if reached 64th position (max)
  {
    val = 0; // start over from lowest value
  }
  delay(500);
}
```
**Wire.available()**

**Description**

Returns the number of bytes available for retrieval with `receive()`. This should be called on a master device after a call to `requestFrom()` or on a slave inside the `onReceive()` handler.

`available()` inherits from the `Stream` utility class.

**Parameters**

None

**Returns**

The number of bytes available for reading.

**See Also**

- `Wire.read()`
- `Stream.available()`
**read()**

**Description**
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.

**Syntax**

```
Wire.read()
```

**Parameters**

none

**Returns**
The next byte received

**Example**

```c
#include <Wire.h>

void setup()
{
    Wire.begin(); // join i2c bus (address optional for master)
    Serial.begin(9600); // start serial for output
}

void loop()
{
    Wire.requestFrom(2, 6); // request 6 bytes from slave device #2
    while(Wire.available()) // slave may send less than requested
    {
        char c = Wire.read(); // receive a byte as character
        Serial.print(c); // print the character
    }
    delay(500);
}
```

[Get Code]

**See also**

- `WireWrite()`
- `WireAvailable()`
- `WireRequestFrom()`
- `Stream.read()`