OneWire Library

OneWire lets you access 1-wire devices made by Maxim/Dallas, such as temperature sensors and ibutton secure memory. For temperature sensors, the DallasTemperature library can be used with this library.

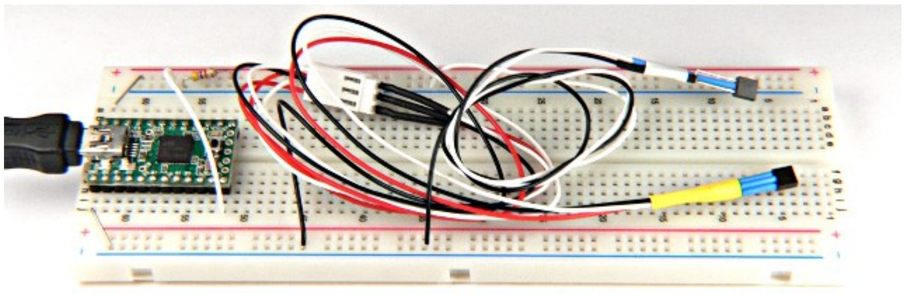
Download: Included with the Teensyduino Installer
Latest Developments on Github

OneWire communicates with 1-wire devices. To act as a 1-wire device, use the OneWireSlave library.

Hardware Requirements

OneWire requires a single 4.7K pullup resistor, connected between the pin and your power supply. When using very long wires, or with counterfeit DS18B20 chips and 3.3V power, a resistor in the 1K to 2.7K range may be required.

Then just connect each 1-wire device to the pin and ground. Some 1-wire devices can also connect to power, or get their power from the signal wire. Please refer to the specifications for the 1-wire devices you are using.



Basic Usage

OneWire myWire(pin)
: Create the OneWire object, using a specific pin. Even though you can connect many 1 wire devices to the same pin, if you have a large number, smaller groups each on their own pin can help isolate wiring problems. You can create multiple OneWire objects, one for each pin.

myWire.search(addrArray)
: Search for the next device. The addrArray is an 8 byte array. If a device is found, addrArray is filled with the device's address and true is returned. If no more devices are found, false is returned.

myWire.reset_search()
: Begin a new search. The next use of search will begin at the first device.

myWire.reset()
: Reset the 1-wire bus. Usually this is needed before communicating with any device.

myWire.select(addrArray)
: Select a device based on its address. After a reset, this is needed to choose which device you will use, and then all communication will be with that device, until another reset.

myWire.skip()
: Skip the device selection. This only works if you have a single device, but you can avoid searching and use this to immediatly access your device.

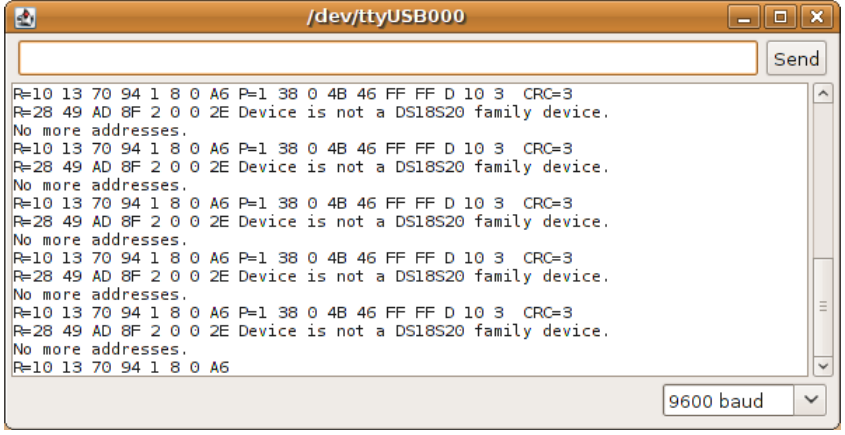
myWire.write(num);
: Write a byte.

myWire.write(num, 1);
: Write a byte, and leave power applied to the 1 wire bus.

myWire.read()
: Read a byte.

myWire.crc8(dataArray, length)
: Compute a CRC check on an array of data.

Example Program



```
R=10 13 70 94 1 8 0 A6 P=1 38 0 4B 46 FF FF D 10 3  CRC=3
R=28 49 AD 8F 2 0 0 2E Device is not a DS18S20 family device.
No more addresses.
R=10 13 70 94 1 8 0 A6 P=1 38 0 4B 46 FF FF D 10 3  CRC=3
R=28 49 AD 8F 2 0 0 2E Device is not a DS18S20 family device.
No more addresses.
R=10 13 70 94 1 8 0 A6 P=1 38 0 4B 46 FF FF D 10 3  CRC=3
R=28 49 AD 8F 2 0 0 2E Device is not a DS18S20 family device.
No more addresses.
R=10 13 70 94 1 8 0 A6 P=1 38 0 4B 46 FF FF D 10 3  CRC=3
R=28 49 AD 8F 2 0 0 2E Device is not a DS18S20 family device.
No more addresses.
R=10 13 70 94 1 8 0 A6 P=1 38 0 4B 46 FF FF D 10 3  CRC=3
R=28 49 AD 8F 2 0 0 2E Device is not a DS18S20 family device.
No more addresses.
R=10 13 70 94 1 8 0 A6
```

You can open the example code from the **File > Examples > OneWire > sample** menu.

```
#include <OneWire.h>

/* DS18S20 Temperature chip i/o */

OneWire  ds(10);  // on pin 10

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];

  if ( !ds.search(addr)) {
    Serial.print("No more addresses.\n");
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("R=");
  for( i = 0; i < 8; i++) {
    Serial.print(addr[i], HEX);
    Serial.print(" ");
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
      Serial.print("CRC is not valid!\n");
      return;
  }

  if ( addr[0] != 0x10) {
      Serial.print("Device is not a DS18S20 family device.\n");
      return;
  }

  // The DallasTemperature library can do all this work for you!

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);         // start conversion, with parasite power on at the end

  delay(1000);     // maybe 750ms is enough, maybe not
  // we might do a ds.depower() here, but the reset will take care of it.

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);         // Read Scratchpad

  Serial.print("P=");
  Serial.print(present,HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print( OneWire::crc8( data, 8), HEX);
  Serial.println();
}
```

1-Wire Information

Maxim Semiconductor's 1-Wire Products

Wikipedia explains 1-wire protocol

Tom Boyd explains 1-wire protocol

Dallas Temperature Control Library

Arduino's OneWire page (warning: has buggy version)

Weather Toys - community using 1-wire devices.

History & Credits

Jim Studt wrote OneWire in 2007, originally based on code by Derek Yerger.

Tom Pollard added CRC code which eliminated the need for a 256 byte array (in RAM).

"RJL20" added the skip function.

Robin James rewrote the search function, posting his version here.

Paul Stoffregen rewrote the I/O routines for interrupt safety, replaced search with Robin James's code, applied several small optimizations, and started calling it "version 2.0" to distinguish from the many buggy copies online.