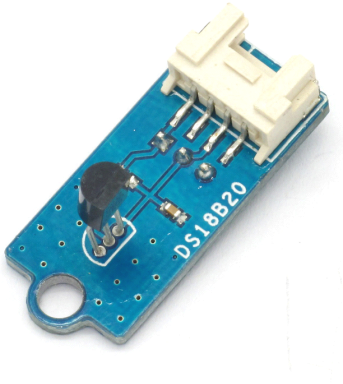
DS18B20 Electronic Brick of Digital Temperature Sensor

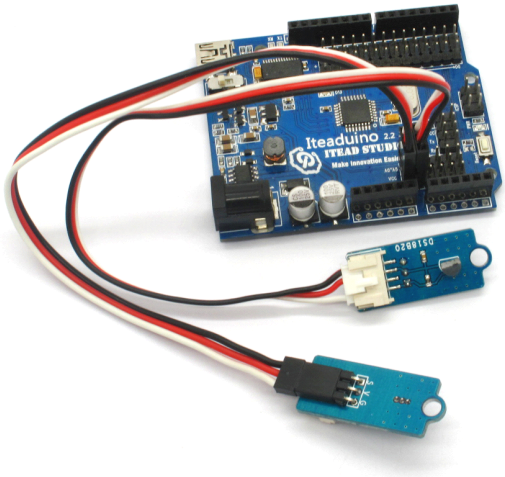
Overview



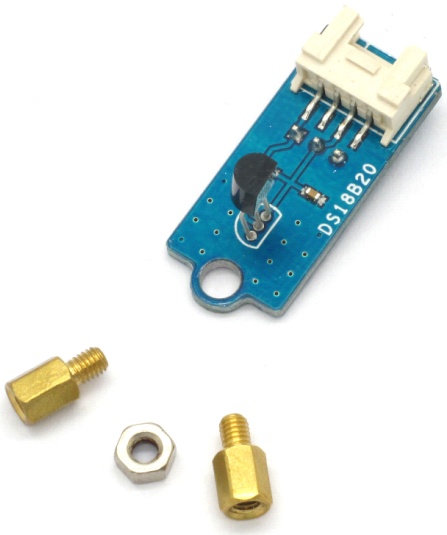
What is an electronic brick? An electronic brick is an electronic module which can be assembled like Lego bricks simply by plugging in and pulling out. Compared to traditional universal boards and circuit modules assembled with various electronic components, electronic brick has standardized interfaces, plug and play, simplifying construction of prototype circuit on one’ s own. There are many types of electronic bricks, and we provide more than twenty types with different functions including buttons, sensors, Bluetooth modules, etc, whose functions cover from sensor to motor drive, from Ethernet to wireless communication via Bluetooth, and so on. We will continue to add more types to meet the various needs of different projects.
DS18B20 is a single-bus digital temperature sensor with features of ultra-small size, wide range of applicable voltage and measurable temperature and high resolution, etc. , which can be applied to temperature control, industrial systems, consumer products, thermometers or any thermal measurement system.

Features

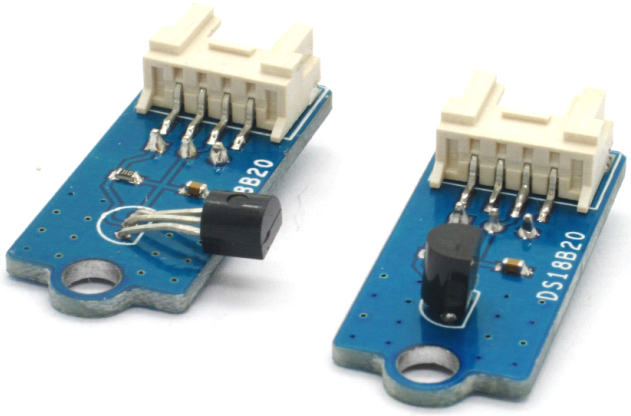
1. Plug and play, easy to use. Compatible with the mainstream 2.54 interfaces and 4-Pin Grove interfaces in the market.



2. With use of M4 standard fixed holes, compatible with M4-standard kits such as Lego and Makeblock



3. Rotatable detecting direction for better adaption



Specifications

PCB size	33.2mm X 14.0mm X 1.6mm
Working voltage	3.3 or 5V DC
Operating voltage	3.3 or 5V DC
Measurable range	-55~+125℃
Measurement resolution	9~12bits (programmable)
Compatible interfaces	2.54 3-pin interface and 4-pin Grove interface(1)

Note 1 : S for digital input/output port, V and G for voltage at the common collector and ground respectively

DC electrical characteristics

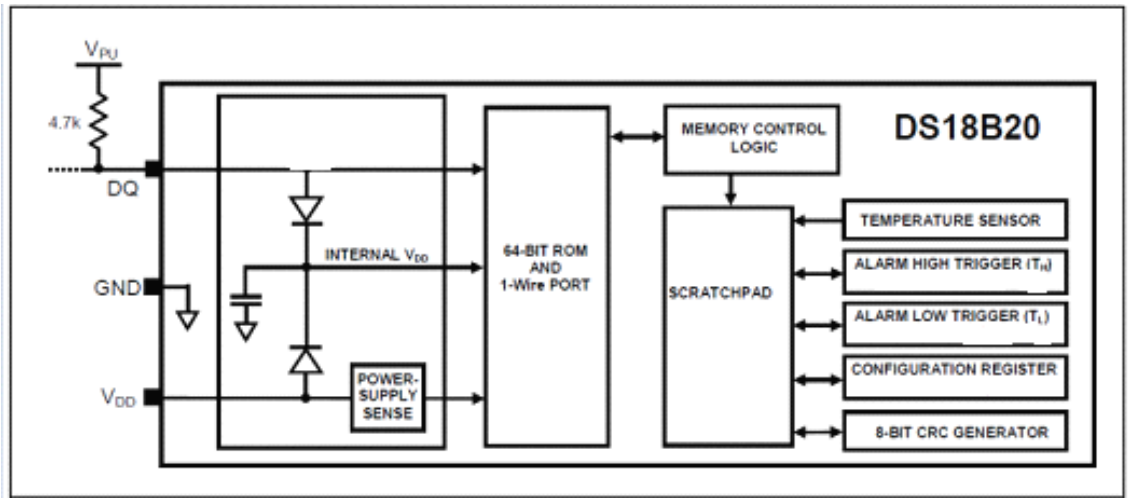
Parameter		Min.	Typical	Max.	Unit
Working voltage		3	5	5.5	VDC
Working current (VCC=5V , T=25℃)		-	1	1.5	mA
Sampling interval		1	-	-	s
Accuracy	-10~+85℃	-	-	±0.5	℃
	-55~+125℃	-	-	±2	℃
Logic input low level V_{IL}		-0.3	-	+0.8	V
Logic input high level V_{IH}		2.2	-	5.5/VCC+0.3	V

AC electrical characteristics

Parameter		Min.	Typical	Max.	Unit
Temperature conversion time t_{CONV}	9-bit accuracy	-	-	93.75	ms
	10-bit accuracy	-	-	187.5	ms
	11-bit accuracy	-	-	375	ms
	12-bit accuracy	-	-	750	ms
Time for strong pullup on t_{SPON}		-	-	10	us
Time slot t_{SLOT}		60	-	120	us
Recovery time t_{REC}		1	-	-	us
Write 0 low time t_{LOW0}		60	-	120	us
Write 1 low time t_{LOW1}		1	-	15	us
Read data valid t_{RDV}		-	-	15	us
Reset time high t_{RSTH}		480	-	-	us
Reset time low t_{RSTL}		480	-	-	us
Presence-detect high t_{PDHIGH}		15	-	60	us
Presence-detect low t_{PDLOW}		60	-	240	us
Capacitance $C_{IN/OUT}$		-	-	25	pF

Instruction

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. The figure below shows a block diagram of DS18B20. The scratchpad memory contains the 2-byte temperature register that stores the datal output from the temperature sensor. In addition, the scratchpad provides access to direct temperature alarm trigger registers (TH and TL) and the 1-byte configuration register. The configuration register allows the user to set the resolution of the temperature-to-digital conversion to 9, 10, 11, or 12 bits. The TH, TL, and configuration registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.



Operation—Measuring temperature

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low-power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a [44h] command. After that, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state. If the DS18B20 is powered by an external supply, the master can issue "read time slots" after the Convert T command and the DS18B20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18B20 is powered with parasite power, there will be no response unless the bus is pulled high by a strong pull-up during the entire temperature conversion.

Temperature Register Format

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

Temperature/Data Relationship

Temperature (℃)	Digital output (Binary)	Digital output (Hex)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

** The default power-on reset value of the temperature register is +85°C*

Operation—Alarm signaling

After the DS18B20 performs a temperature conversion, the temperature value is compared to the user-defined alarm trigger values stored in the 1-byte TH and TL registers. The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The TH and TL registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. How TH and TL can be accessed through bytes 2 and 3 of the scratchpad will be explained in the Memory section.

TH and TL Register Format

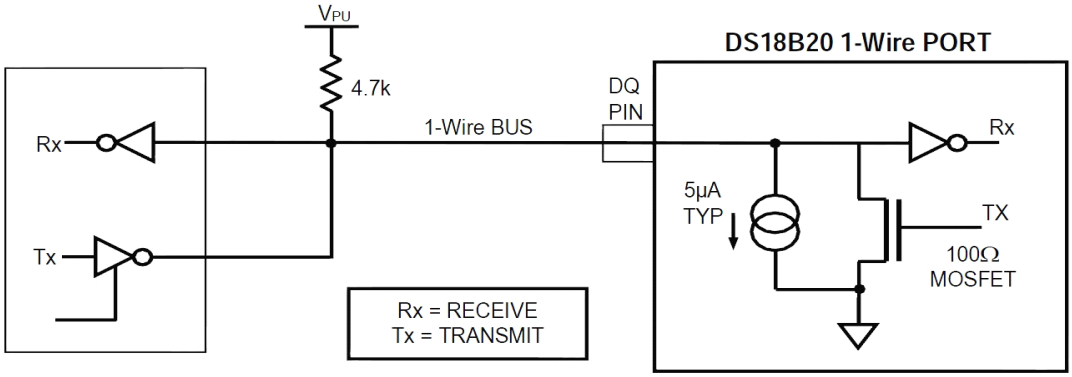
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Only bits 11 through 4 of the temperature registers are used in the TH and TL comparison since TH and TL are 8-bit registers. If the measured temperature is lower than or equal to TL or higher than or equal to TH, an alarm condition exists and an alarm flag is set inside the DS18B20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18B20s on the bus by issuing an Alarm Search [ECh] command. Any DS18B20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18B20s have experienced an alarm condition. If an alarm condition exists and the TH or TL settings have changed, another temperature conversion should be done to validate the alarm condition.

1-Wire Bus System

Hardware configuration



Transaction sequence

The transaction sequence for accessing the DS18B20 via 1-wire bus port is as follows:

Step 1. Initialization

Step 2. ROM Command

Step 3. DS18B20 Function Command

It is very important to follow this sequence every time the DS18B20 is accessed, as the DS18B20 will not respond if any steps in the sequence are missing or out of order. While after issuing ROM commands of Search [F0h] and Alarm Search [ECh] commands, the master must return to Step 1 in the sequence.

Initialization

All transactions on the 1-Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the bus master know that DS18B20 are on the bus and are ready to operate.

ROM commands

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64-bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-Wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18B20 function command.

SEARCH ROM [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master performs search of ROM codes through several Search ROM cycles to identify all of the slave devices. If there is only one slave on the bus, the simpler Read ROM command (see below) can be used in place of the Search ROM process. For Book of Standards, please refer to www.ibutton.com/ibuttons/standard.pdf. After every Search ROM command, the bus master must return to Step 1 in the transaction sequence.

READ ROM [33h]

This command can only be used when there is one DS18B20 on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM procedure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

MATCH ROM [55h]

The match ROM command followed by a 64-bit ROM code sequence allows the bus master to address a specific DS18B20 on a multi-drop bus. Only the DS18B2 that exactly matches the 64-bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

SKIP ROM [CCh]

This command allows the master to use function command without providing 64-bit ROM code. For example, the master can perform temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command.
Note that the Read Scratchpad [BEh] command can follow the Skip ROM command only if there is a single slave device on the bus. In this case, time is saved by allowing the master to read from the slave without sending the device's 64-bit ROM code. A Skip ROM command will cause a data collision on the bus if there is more than one slave on the bus since multiple devices attempt to transmit signals simultaneously.

ALARM SEARCH [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18B20s experienced an alarm condition during the most recent temperature conversion. After every Alarm Search cycle, the bus master must return to Step 1 in the transaction sequence.

DS18B20 FUNCTION COMMANDS

After the bus master has used a ROM command to address the DS18B20 with which it wishes to communicate, the master can issue one of the DS18B20 function commands. These commands allow the master to write and to read from the DS18B20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The details of DS18B20 function commands can be described below.

CONVERT T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10µs (max.) after this command is issued the master must enable a strong pull-up on the 1-Wire bus for the duration of the conversion (tCONV). If the DS18B20 is powered by an external supply, the master can issue read time slots after the Convert T command and the DS18B20 will respond by transmitting a 0 while the temperature conversion is in progress and a 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pull-up during the conversion.

WRITE SCRATCHPAD [4Eh]

This command allows the master to write 3 bytes of data to the DS18B20’ s scratchpad. The first data byte is written into the TH register (byte 2 of the scratchpad), the second byte is written into the TL register (byte 3), and the third byte is written into the configuration register (byte 4). Data must be transmitted least valid bit first. All three bytes MUST be written before the master issues a reset command, otherwise the data may be corrupted.

READ SCRATCHPAD [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least valid bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

COPY SCRATCHPAD [48h]

This command copies the contents of the scratchpad TH, TL and configuration registers (bytes 2, 3 and 4) to EEPROM. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pull-up on the 1-Wire bus for at least 10ms.

RECALL E2 [B8h]

This command recalls the alarm trigger values (TH and TL) and configuration data from EEPROM and places them in the scratchpad memory. The master device can issue read time slots following the Recall E2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up in DS18B20, so valid data is available in the scratchpad as soon as power is applied to the device.

READ POWER SUPPLY [B4h]

The master device issues this command followed by a read time slot to determine if any DS18B20s on the bus are using parasite power. During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high.

1-WIRE SIGNALING

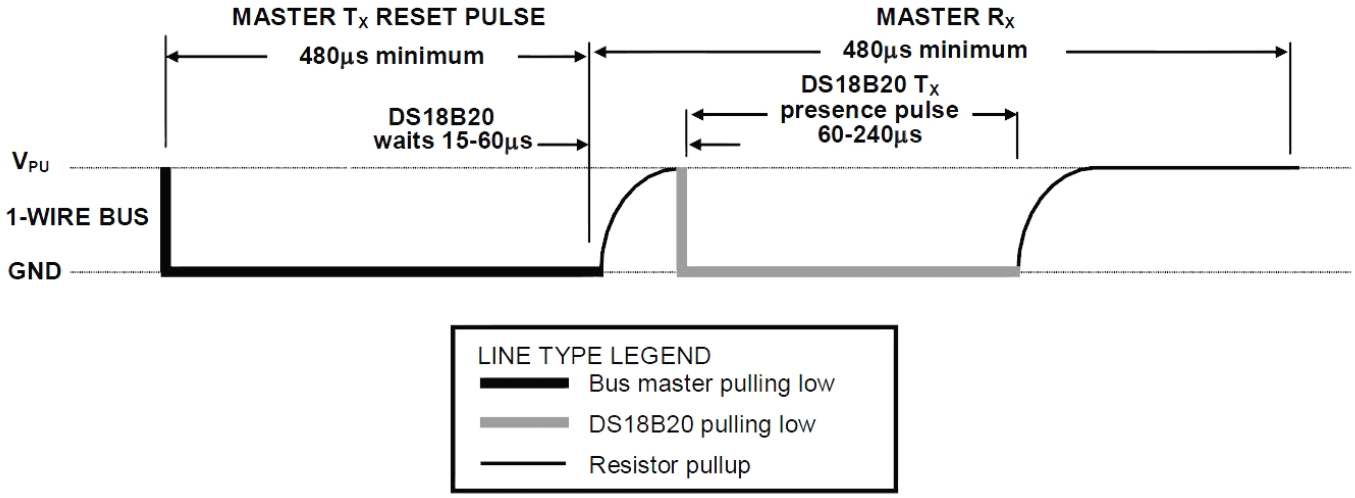
The DS18B20 uses a strict 1-Wire communication protocol to ensure data integrity. Several signal types are defined by this protocol: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. The bus master initiates all these signals except the presence pulse.

INITIALIZATION Sequence—RESET AND PRESENCE PULSES

All communication with the DS18B20 begins with an initialization sequence which is illustrated in the figure below. When the DS18B20 sends the presence pulse in response to the reset pulse, it is indicating that DS18B20 is ready to send and to receive data.

During the initialization sequence the bus master transmits (TX) the reset pulse by pulling the 1-Wire bus low for a minimum of 480μs. The bus master then releases the bus and goes into receive mode (RX) and pulls up to high level by a 5k pull-up resistor. When the DS18B20 detects the rising edge of I/O pin, it waits for 15μs to 60μs and then transmits a presence pulse consisting of low level signals of 60μs to 240μs.

Initialization sequence diagram:



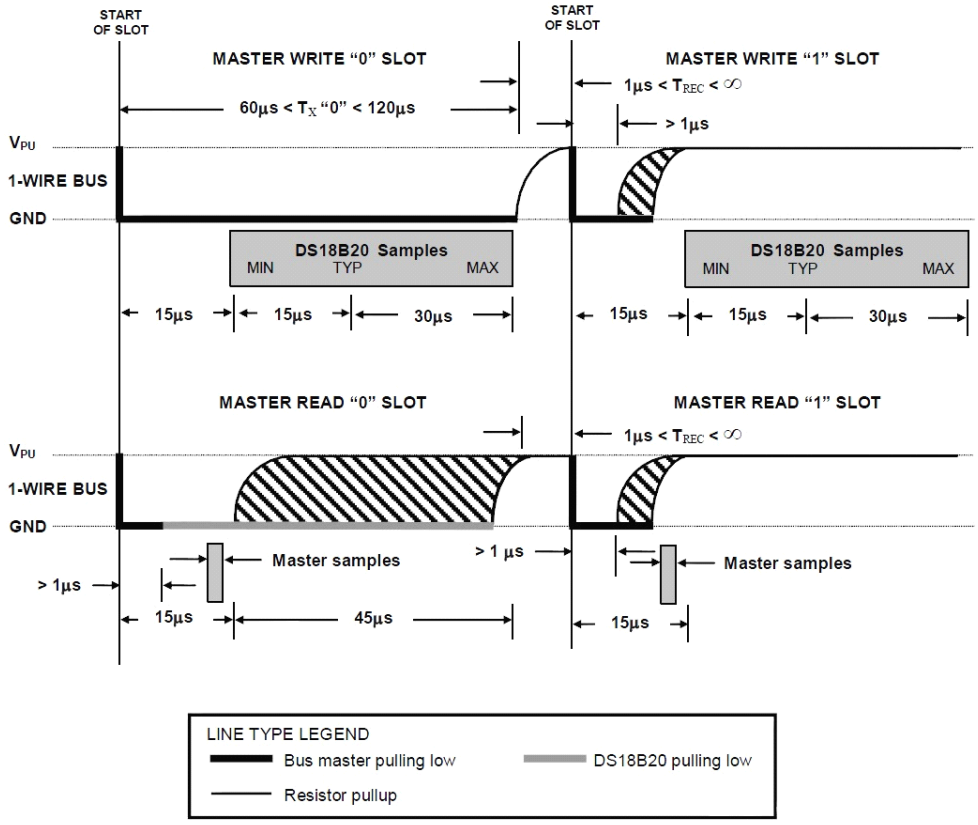
WRITE TIME SLOTS

There are two types of write time slots: "Write 1" time slots and "Write 0" time slots. The bus master uses a Write 1 time slot to write a logic 1 to the DS18B20 and a Write 0 time slot to write a logic 0 to the DS18B20. All time slots writing must last for at least 60μs with a minimum of 1μs recovery time between two write intervals. Both types of write time slots are initiated by the master pulling the data bus from high level down to low level.

To generate a write time slot, the bus master must pull the data bus to low level and then release the 1-Wire bus within 15 μs after write time slot begins. When the bus is released, the 5kΩ pull-up resistor will pull the bus high. To generate a Write 0 time slot, the bus master must pull down the bus to low level and hold (for at least 60μs).

DS18B20 will sample from I/O cable during a window that lasts from 15μs to 60μs after the master initiates the write time slot. If the bus is high during the sampling window, a 1 is written to the DS18B20. If the line is low, a 0 is written to the DS18B20.

Read/write time slot diagram:

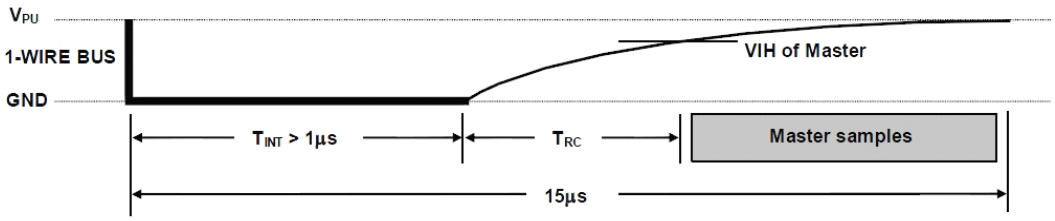


READ TIME SLOTS

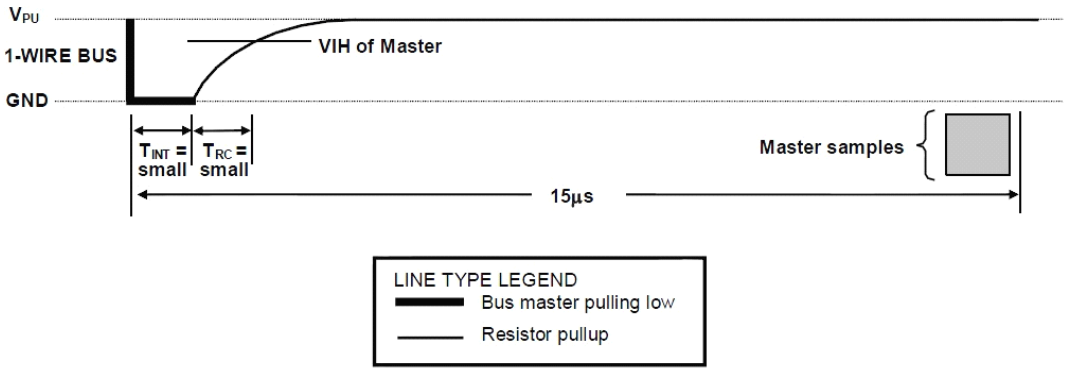
The DS18B20 can only transmit data to the master when the master issues read time slots. Therefore, the master must generate read time slots immediately after issuing a Read Scratchpad [BEh] or Read Power Supply [B4h] command, so that the DS18B20 can provide the request data. In addition, the master can generate read time slots after issuing Convert T [44h] or Recall E2 [B8h] commands, details as explained in the DS18B20 Function Commands section.

All read time slots must last for at least 60µs with a minimum of 1µs recovery time between two slots. A read time slot is initiated by the master device pulling the data bus low for a minimum of 1µs and then releasing the bus. After the master initiates the read time slot, the DS18B20 will begin transmitting a 1 or 0 on bus. After finishing transmitting a 0, the DS18B20 will release the bus, and the bus will be pulled back to its high idle state by the pull-up resister. Output data from the DS18B20 is valid for 15µs after the falling edge that initiated the read time slot. Therefore, the master must stop driving I/O pin to low level 15us to read the status of I/O pin.

Sum of TINIT, TRC, and TSAMPLE must be less than 15µs for a read time slot. Figure 16 shows that system timing margin is maximized by keeping TINIT and TRC as short as possible and by locating the master sample time during read time slots towards the end of the 15µs period.



Recommend master read 1 time slot:



DEMO

Connect S port of DS18B20 electronic brick to D0 port of Arduino board, and we will use the following program to read register information and temperature value.

```
#include <OneWire.h>

// OneWire DS18S20, DS18B20, DS1822 Temperature Example
// http://www.pjrc.com/teensy/td_libs_OneWire.html
// The DallasTemperature library can do all this work for you!
// http://milesburton.com/Dallas_Temperature_Control_Library

OneWire  ds(10);  // on pin 10

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

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

```
  float celsius, fahrenheit;

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

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

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

  // the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      Serial.println("   Chip = DS18S20");   // or old DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("   Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
      Serial.println("   Chip = DS1822");
      type_s = 0;
      break;
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return;
  }

  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("  Data = ");
  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();

  // convert the data to actual temperature

  unsigned int raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // count remain gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw << 3;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw << 2; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw << 1; // 11 bit res, 375 ms
    // default is 12 bit resolution, 750 ms conversion time
  }
  celsius = (float)raw / 16.0;
  fahrenheit = celsius * 1.8 + 32.0;
  Serial.print("  Temperature = ");
  Serial.print(celsius);
  Serial.print(" Celsius, ");
  Serial.print(fahrenheit);
  Serial.println(" Fahrenheit");
}
```

Revision record

Version	Description	Date	Written by
v1.0	Initial edition	26th, April, 2013	Stan Lee