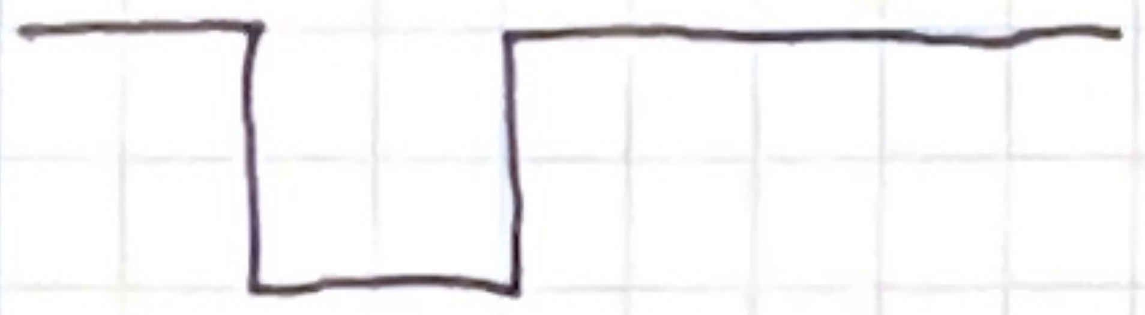
05/27/25 [illegible]

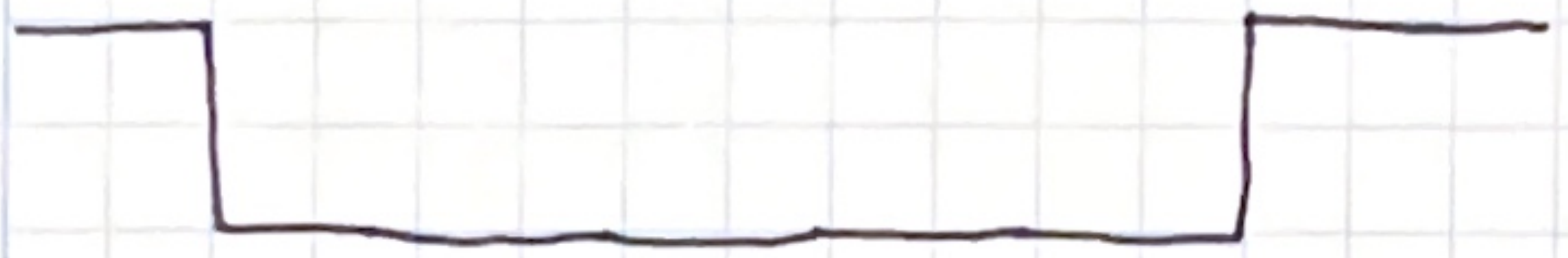
1-Wire Digital Thermometer DS18B20

Dallas 1 wire

Uses one wire to transmit 0 or 1's. The amount of time pulled low indicates 0 or 1.



=> Indicates a "1"

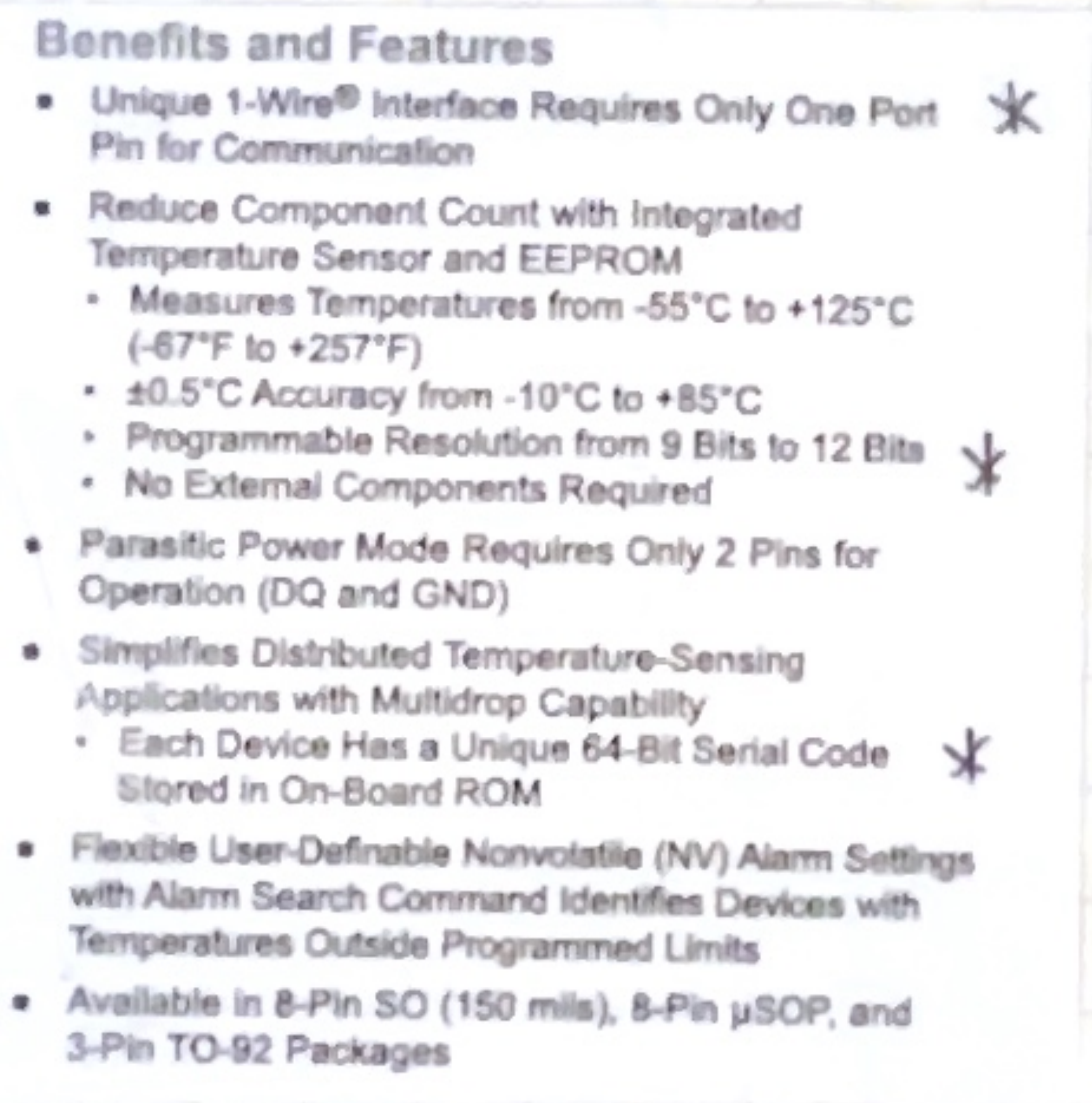


=> Indicates a "0"

The data line is often called "DQ" and needs a pull up resistor.

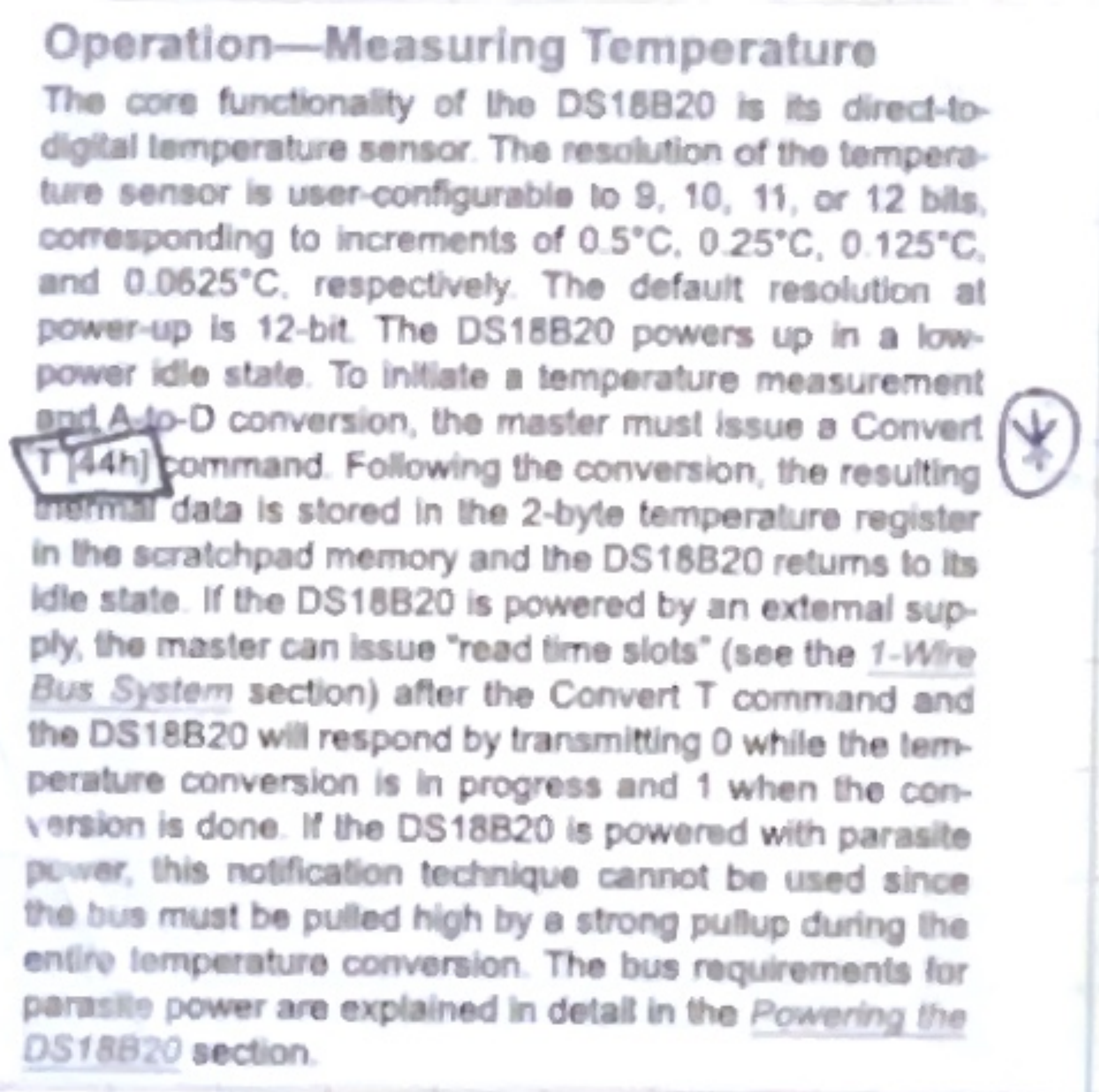
For the DS18B20: There are 3 wires.

Yellow	DQ
Blue	GND
Red	Power (3.0-5.5v)



Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication *
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits *
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM *
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages



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 Convert $\boxed{\text{T [44h]}}$ command. 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 idle state. If the DS18B20 is powered by an external supply, the master can issue "read time slots" (see the *1-Wire Bus System* section) 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, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the *Powering the DS18B20* section. *

1-Wire Digital Thermometer DS18B20

05/27/25
W [illegible]

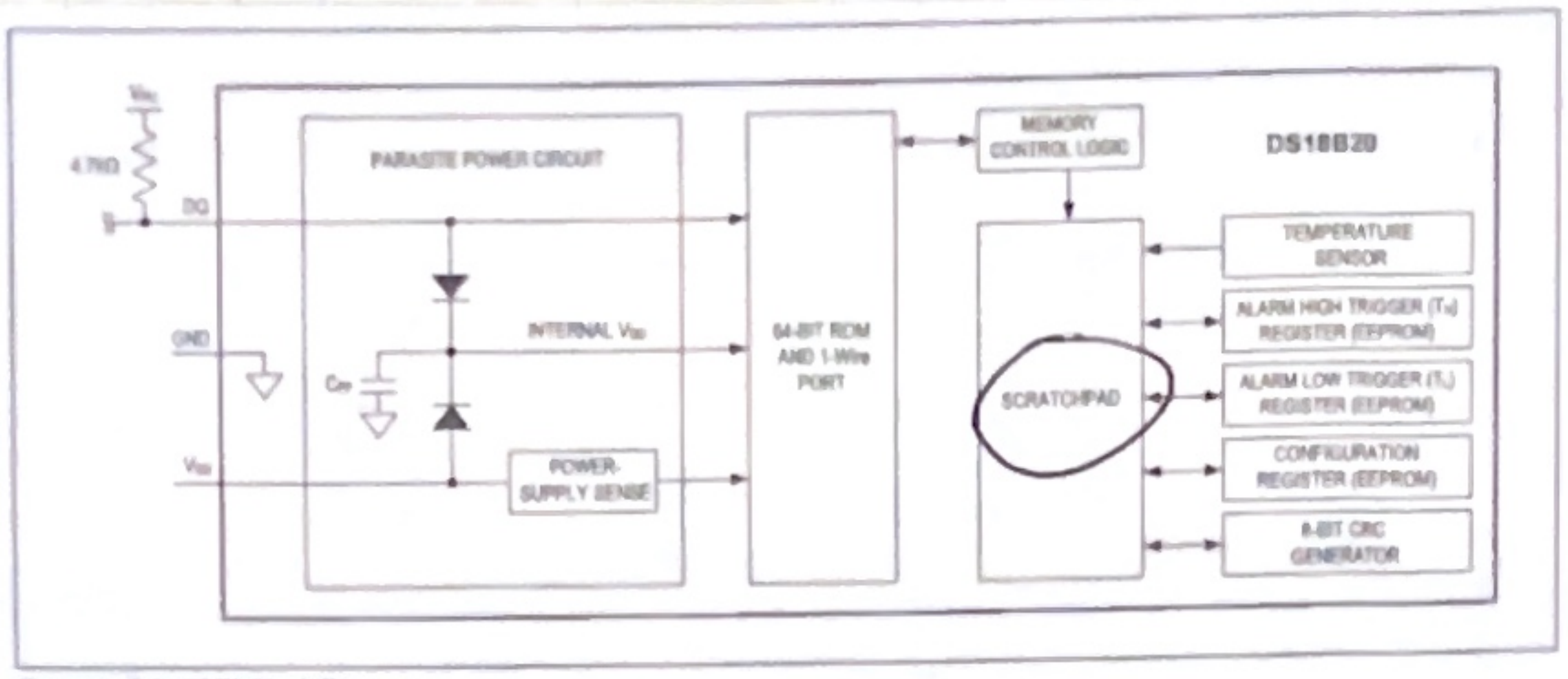
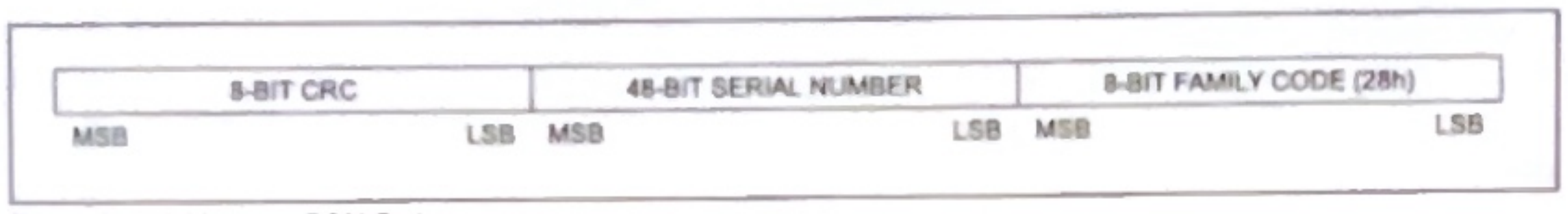


Figure 3. DS18B20 Block Diagram

All the data will flow through the "Scratch pad". This means we will write a command to the scratch pad, write another command to transfer the scratch pad over to memory, and write again to tell the chip that we want to read the data.

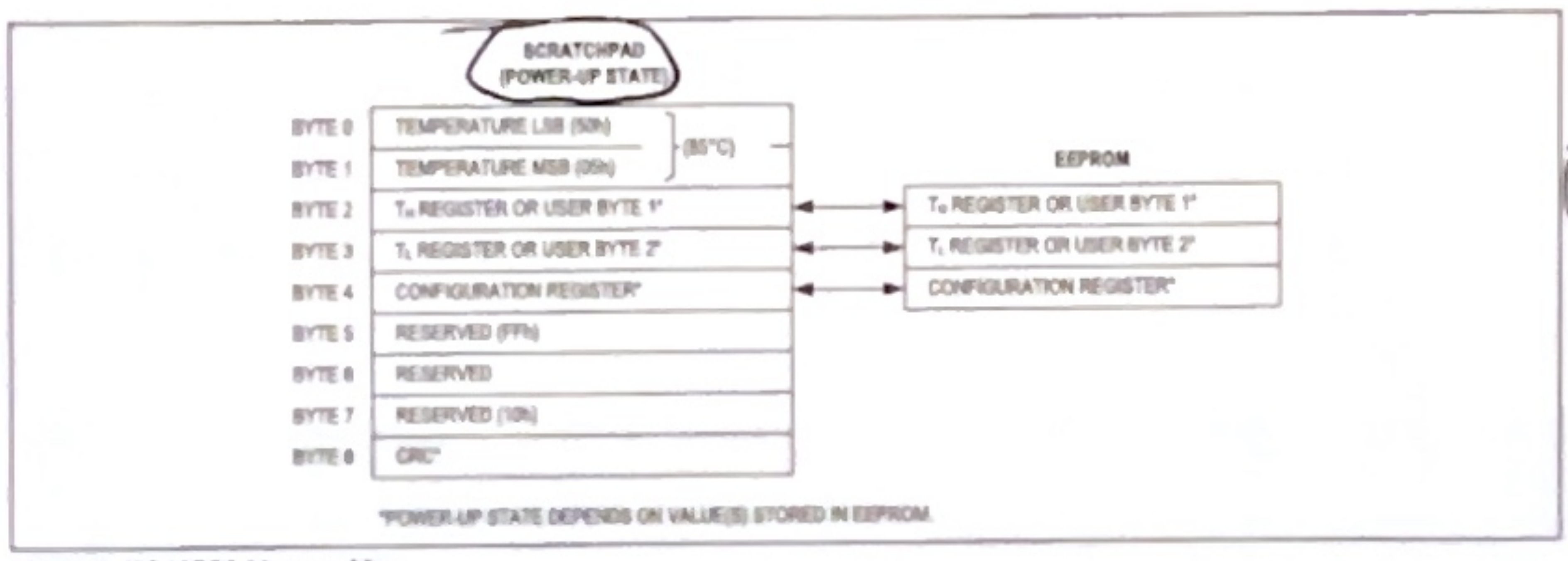


8-BIT CRC	48-BIT SERIAL NUMBER	8-BIT FAMILY CODE (28h)
MSB … LSB	MSB … LSB	MSB … LSB

Figure 8. 64-Bit Lasered ROM Code

(*)

Scratch pad Structure.



SCRATCHPAD (POWER-UP STATE)

	SCRATCHPAD		EEPROM
BYTE 0	TEMPERATURE LSB (50h) (85°C)		
BYTE 1	TEMPERATURE MSB (05h) (85°C)		
BYTE 2	T_H REGISTER OR USER BYTE 1*	↔	T_H REGISTER OR USER BYTE 1*
BYTE 3	T_L REGISTER OR USER BYTE 2*	↔	T_L REGISTER OR USER BYTE 2*
BYTE 4	CONFIGURATION REGISTER*	↔	CONFIGURATION REGISTER*
BYTE 5	RESERVED (FFh)		
BYTE 6	RESERVED		
BYTE 7	RESERVED (10h)		
BYTE 8	CRC*		

*POWER-UP STATE DEPENDS ON VALUE(S) STORED IN EEPROM.

(*)

Figure 9. DS18B20 Memory Map

05/27/25
Wesley Costa

1-wire Digital Thermometer DS18B20

Setting Resolution and Commands

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1
	0	1					

→ 3F for ~~10~~ 10-bit

Figure 10. Configuration Register

Table 2. Thermometer Resolution Configuration

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75ms	($t_{CONV}/8$)
0	1	10	187.5ms	($t_{CONV}/4$) *
1	0	11	375ms	($t_{CONV}/2$)
1	1	12	750ms	(t_{CONV})

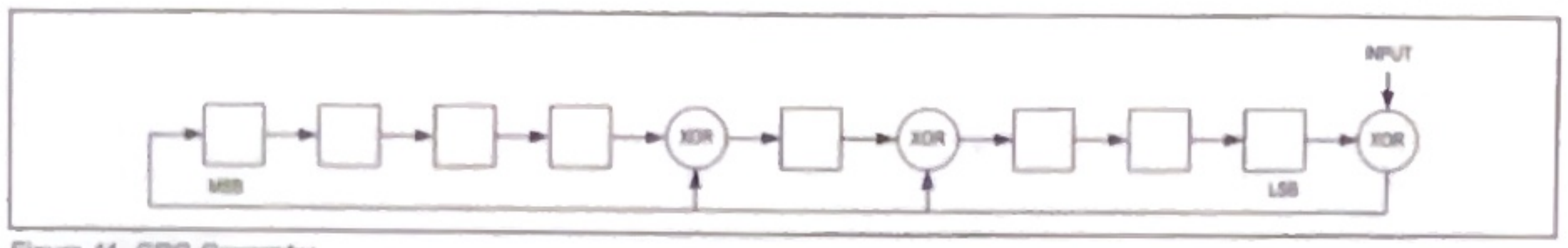


Figure 11. CRC Generator

Transaction Sequence *

The transaction sequence for accessing the DS18B20 is as follows:

Step 1. Initialization

Step 2. ROM Command (followed by any required data exchange)

Step 3. DS18B20 Function Command (followed by any required data exchange)

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. Exceptions to this rule are the Search ROM [F0h] and Alarm Search [ECh] commands. After issuing either of these ROM 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 slave devices (such as the DS18B20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the *1-Wire Signaling* section.

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. A flowchart for operation of the ROM commands is shown in Figure 13.

Table 3. DS18B20 Function Command Set

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Note 1: For parasite-powered DS18B20s, the master must enable a strong pullup on the 1-Wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
Note 2: The master can interrupt the transmission of data at any time by issuing a reset.
Note 3: All three bytes must be written before a reset is issued.

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 pullup on the 1-Wire bus for the duration of the conversion (t_{CONV}) as described in the *Powering the DS18B20* section. 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 pullup during the conversion.

Copy Scratchpad [48h]

This command copies the contents of the scratchpad T_H, T_L 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 pullup on the 1-Wire bus for at least 10ms as described in the *Powering the DS18B20* section.

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 T_H register (byte 2 of the scratchpad), the second byte is written into the T_L register (byte 3), and the third byte is written into the configuration register (byte 4). Data must be transmitted least significant bit first. All three bytes MUST be written before the master issues a reset, or the data may be corrupted.

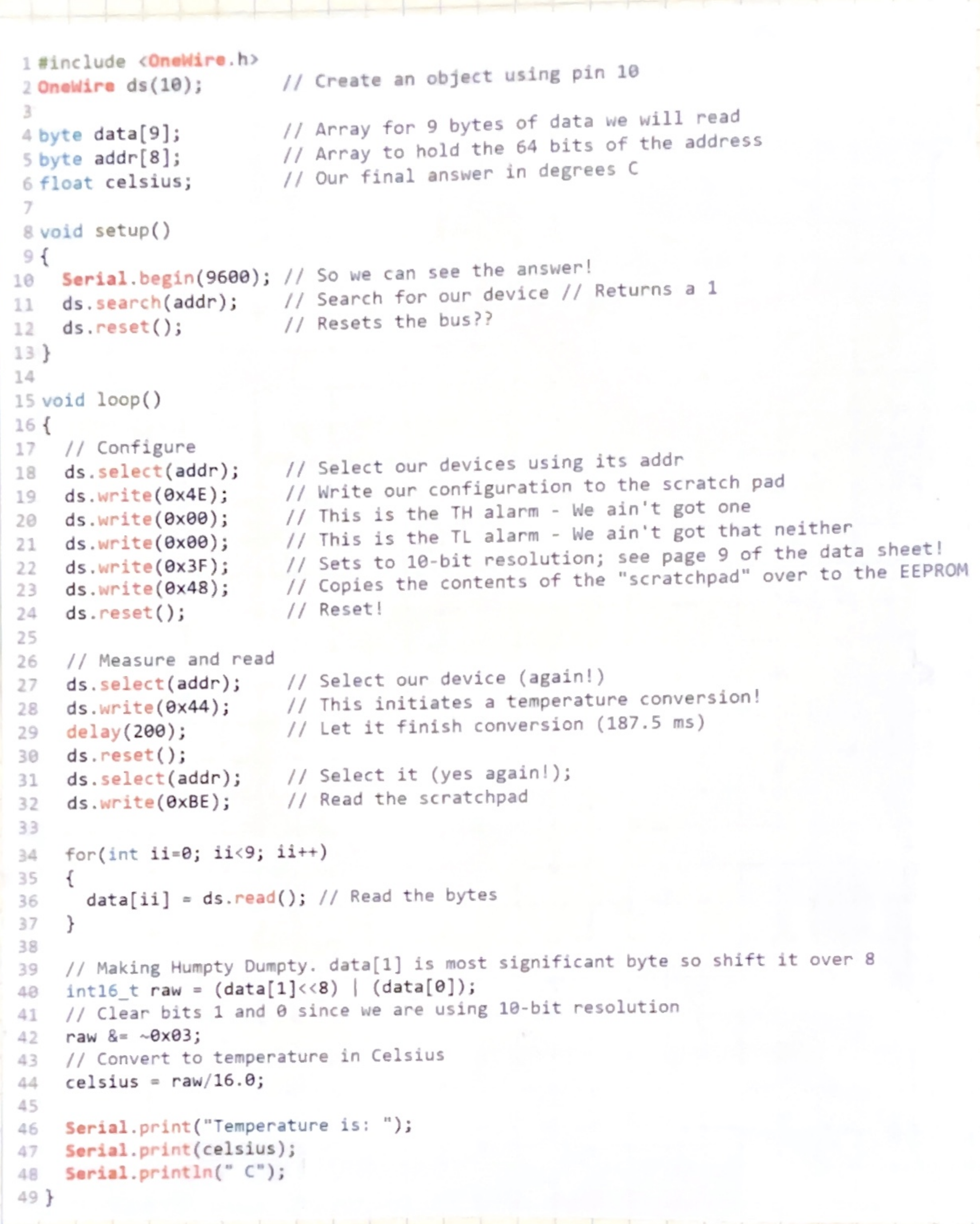
Read Scratchpad [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant 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.

1-Wire Digital Thermometer DS18B20

05/27/25
[illegible]

Code



```
#include <OneWire.h>
OneWire ds(10);        // Create an object using pin 10

byte data[9];          // Array for 9 bytes of data we will read
byte addr[8];          // Array to hold the 64 bits of the address
float celsius;         // Our final answer in degrees C

void setup()
{
  Serial.begin(9600); // So we can see the answer!
  ds.search(addr);    // Search for our device // Returns a 1
  ds.reset();         // Resets the bus??
}

void loop()
{
  // Configure
  ds.select(addr);    // Select our devices using its addr
  ds.write(0x4E);     // Write our configuration to the scratch pad
  ds.write(0x00);     // This is the TH alarm - We ain't got one
  ds.write(0x00);     // This is the TL alarm - We ain't got that neither
  ds.write(0x3F);     // Sets to 10-bit resolution; see page 9 of the data sheet!
  ds.write(0x48);     // Copies the contents of the "scratchpad" over to the EEPROM
  ds.reset();         // Reset!

  // Measure and read
  ds.select(addr);    // Select our device (again!)
  ds.write(0x44);     // This initiates a temperature conversion!
  delay(200);         // Let it finish conversion (187.5 ms)
  ds.reset();
  ds.select(addr);    // Select it (yes again!);
  ds.write(0xBE);     // Read the scratchpad

  for(int ii=0; ii<9; ii++)
  {
    data[ii] = ds.read(); // Read the bytes
  }

  // Making Humpty Dumpty. data[1] is most significant byte so shift it over 8
  int16_t raw = (data[1]<<8) | (data[0]);
  // Clear bits 1 and 0 since we are using 10-bit resolution
  raw &= ~0x03;
  // Convert to temperature in Celsius
  celsius = raw/16.0;

  Serial.print("Temperature is: ");
  Serial.print(celsius);
  Serial.println(" C");
}
```