

Sensing - Temperature Measurement with a 1-Wire Digital Sensor

AN2163

Author: Onur Ozbek

Associated Project: Yes

Associated Part Family: CY8C22xxx, CY8C24xxx, CY8C25xxx, CY8C26xxx, CY8C27xxx

[GET FREE SAMPLES HERE](#)

Software Version: PSoC Designer™ 4.0

Associated Application Notes: AN2094

Application Note Abstract

This Application Note implements the temperature measurement at 1°C resolution using a Dallas Semiconductor DS1821 digital temperature sensor with a 1-wire digital interface. It also shows how to implement a “1-Wire Bus Protocol” communication in ‘C’ programming language between slave and master devices using just one wire.

Introduction

The temperature sensor has two operating modes, **1-wire mode** and **thermostat mode**. In this Application Note, 1-wire mode of the sensor is used to measure temperature. The operating mode is determined by changing the settings in the configuration register of the sensor stored in EEPROM.

In 1-wire mode, the related pin (DQ) of the sensor is configured as a 1-wire port for communication with PSoC® using the protocol explained later in this document. The sensor outputs temperature in degrees Celsius and stores it in 2’s-complement format in the single-byte temperature register shown in Table 1. The most significant bit is used as a flag to show whether the temperature is positive or negative. For examples of digital output data and corresponding temperature readings see Table 2.

Table 1. Temperature Register

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
S	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰

Table 2. Temperature/Data Relationships

Temperature	Digital Output	Digital Output
+125°C	01111101	7Dh
+85°C	01010101	55h
+25°C	00011001	19h
0°C	00000000	00h
-1°C	11111111	FFh
-25°C	11100111	E7h
-55°C	11001001	C9h

Status/Configuration Register

The status/configuration register, stored in EEPROM of the digital sensor (Table 3), adjusts the operation mode and options.

Table 3. Status/Configuration Register

Bit 7	Bit 6	Bit5	Bit4	Bit3	Bit 2	Bit1	Bit0
D	1	NVB	THF	THL	T/R	POL	1Shot
R	R	R	R/W	R/W	R/W	R/W	R/W

The sensor output temperature is stored in 2’s-complement format in the temperature register and can be accessed when the sensor is in **1-wire mode**. This mode can be configured by selecting **T/R=0** in the status/configuration register. The sensor can be configured to take continuous temperature measurements (continuous-conversion mode) or single measurements (one-shot mode). The desired configuration can be achieved by setting the 1Shot bit in the status/configuration register:

1Shot=0 → -continuous-conversion mode

1Shot=1 → one-shot mode

In one-shot mode, PSoC can monitor the **D** bit in the configuration register to determine the conversion status:

D=0 → conversion in progress

D=1 → conversion complete

The **D** bit does not provide conversion status in continuous-conversion mode since measurements are constantly in progress.

THL, TLF, and POL bits, which are not used for measuring temperature, as well as **T/R** and **1Shot** bits in the status/configuration register, are stored in EEPROM of the sensor. Writing to these bits can take up to 10 ms. To avoid data corruption, no write-to-memory should be initiated while a write-to-memory is in progress. Reading the **NVB** bit in the status/configuration register monitors write status:

NVB=1 → write to EEPROM memory in progress

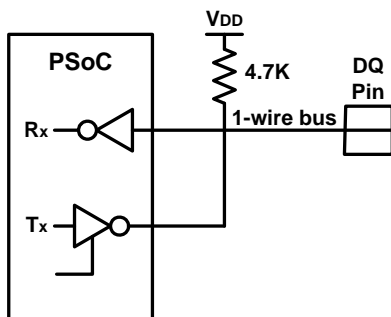
NVB=0 → EEPROM memory is not busy

Hardware Configuration

The 1-wire bus has only has a single data line: PSoC and the sensor interface to the data line via an open drain. This allows each device to “release” the data line when the device is not transmitting so that the bus is available for use by the other device.

The 1-wire bus requires an external Pull-Up resistor of approximately 4.7 kΩ; thus, the idle state for the 1-wire bus is high. However, it is possible to build this register in the PSoC using PSoC’s pin drives. Figure 1 shows the hardware configuration.

Figure 1. Hardware Configuration



Using PSoC’s configurable and flexible pin-out feature, a single pin can be used to implement the 1-wire bus protocol, realizing both read and write with a single pin.

In order to change configuration of the I/O pin, Port Drive Mode 0 Registers (**PRTxDM0**) and Port Drive Mode 1 Registers (**PRTxDM1**) should be controlled together (see Table 4). For more information please look at AN2094 “PSoC I/O Pin-Port Configuration.”

Table 4. Port Drive Mode

DM1	DM0	Output
0	0	Resistive Pull Down
0	1	Strong Drive (for Output)
1	0	High Z (for Input)
1	1	Resistive Pull up

Port2[3] is selected as a data line (DQ) and can be changed using the definitions below, available in the *define.h* file included with this project.

```
#define DQ PRT2DR
```

```
#define pin pin3
```

In order to eliminate the external resistor, this pin should be in Pull-Up mode. To control its configuration dynamically, the following function, *makepullup*, is defined in example Code 1.

If this code is to be used in the CY8C27xxx series of chips, the comment line for PRT2DM2 register should be uncommented.

Code 1. “makepullup” Function

```
=void makepullup (void)
{
    M8C_SetBank1; //Jump to Bank1
    PRT2DM0=0x08; //00001000
    PRT2DM1=0x08; //00001000
    //PRT2DM2=0x00; //00000000 Uncomment this
    //line if CY8C27xxx is used
    M8C_SetBank0; //Jump back to Bank0
    return;
}
```

Note In the CY8C27xxx series, three registers control the pin drive configuration. This is shown in Table 5.

Table 5. Drive Modes

DM2	DM1	DM0	Mode
0	0	0	Resistive Pull Down
0	0	1	Strong Drive
0	1	0	High Impedance
0	1	1	Resistive Pull Up
1	0	0	Open Drain, Drives High
1	0	1	Slow Strong Drive
1	1	0	High Impedance Analog
1	1	1	Open Drain, Drives Low

Transaction Sequence

The transaction sequence for accessing the sensor with the 1-wire port is:

1. Initialization.
2. Sensor Function Commands.
3. Data Transmitted/Received.

Initialization

All transactions on the 1-wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by PSoC followed by a presence pulse transmitted by the digital sensor. The presence pulse lets PSoC know that the sensor is on the bus and ready to operate. Timing for the reset and presence pulses is detailed in Figure 10 in the Appendix.

Read/Write Time Slots

PSoC writes data to the digital sensor during write time slots and reads data from the digital sensor during read time slots. One bit of data is transmitted over the 1-wire bus per time slot.

Write Time Slots

There are two types of write time slots, "Write 1" and "Write 0." time slots. These correspond to logic 1 and logic 0, respectively. All write time slots must be a minimum of 60 μ s duration with a minimum 1 μ s recovery time between individual write slots. PSoC initiates both types of write time slots pulling the 1-wire bus low. See Figure 11 in the Appendix.

To generate a Write **1** time slot, once the 1-wire bus is low, PSoC must release a 1-wire bus within 15 μ s. When the bus is released, the external Pull-Up resistor will pull the bus high. To generate a Write **0** time slot, after pulling the 1-wire

bus low, PSoC must continue to hold the bus low for the duration of the time slot, at least 60 μ s.

The digital sensor samples the 1-wire bus during a window that lasts from 15 μ s to 60 μ s after PSoC initiates the write time slot. If the bus is high during the sampling time, a **1** is written to the sensor. If the line is low, a **0** is written to the sensor.

Read Time Slots

The digital sensor can only transmit data to PSoC when PSoC is issuing a read command. This is so the sensor can provide the requested data (see Table 6). All read time slots must be a minimum of 60 μ s in duration with a minimum of a 1 μ s recovery time between slots.

PSoC initiates the read time slot by pulling the 1-wire bus low for a minimum of 1 μ s and then releasing the bus. See Figure 11 in the Appendix. After PSoC initiates the read time slot, the digital sensor will begin transmitting a **1** or **0** on the bus. The sensor transmits a **1** by leaving the bus high and then transmits a **0** by pulling the bus low. All output data from the sensor is valid for 15 μ s after the falling edge that initiated the read time slot. Therefore, PSoC must release the bus and then sample the bus state within 15 μ s from the start of the slot.

Sensor Function Commands

The digital sensor function commands allow the master to communicate with, and configure, the digital sensor.

The digital sensor can only transmit data to PSoC when issuing commands. The digital sensor function commands are related to temperature measurement as summarized in Table 6.

Table 6. The Digital Sensor Function Command

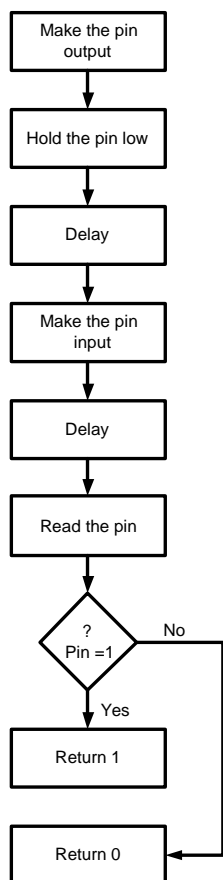
Command	Description	Protocol	1-Wire Bus Activity After Command is Issued
Read Temperature	Reads last converted temperature value from temperature register.	AAh	PSoC receives 8-bit temperature value from sensor.
Start Convert T	Initiates temperature conversion.	EEh	None.
Stop Convert T	Halts temperature conversion.	22h	None.
Write Status	Writes data to status/configuration register.	0Ch	PSoC transmits 8-bit status/configuration value to digital sensor.
Read Status	Reads data from status/configuration register.	ACh	PSoC receives 8-bit status/configuration value from digital sensor.

Data Transmitted/Received

Read One Bit

To read the bit from the digital sensor, it is key to choose the duration of delays. This is shown in Figure 11 of the Appendix. A flowchart and the function for reading one bit are shown ahead.

Figure 2. Single Bit Read Flowchart



Code 2. Single Bit Read Function

```

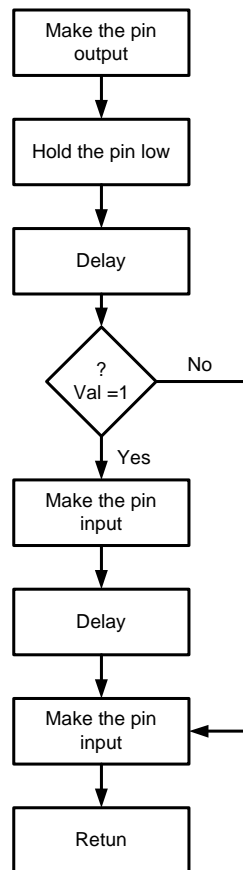
BYTE ds_r_bit(void)
{
    makeoutput();
    output_low(DQ,pin);
    asm("nop\n" "nop\n" "nop\n"); // 1
us delay
    makeinput();
    DelayFunction(3); // 11 us delay

    if(input(DQ,pin)) {return(true);}
    else
{return(false);}
}
  
```

Write One Bit

In order to write a bit to the digital sensor, the key point is also the duration of delay. The function created for this issue must take a value to be written to the sensor. The flowchart and the function for writing one bit are shown below.

Figure 3. Single Bit Write Flowchart



Code 3. Single Bit Write Function

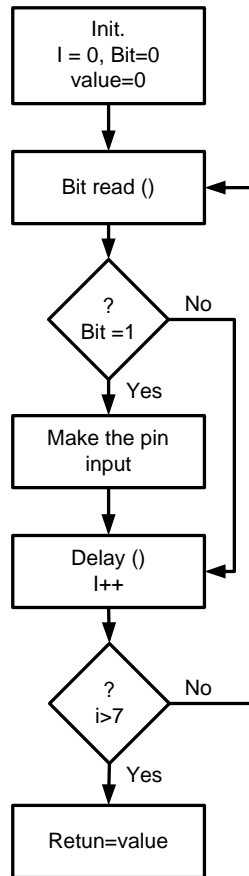
```

void ds_w_bit(BYTE bitval)
{
    makeoutput();
    output_low(DQ,pin);
    DelayuFunction(4); // 15 us delay
    if(bitval)
    {
        makeinput();
    }
    DelayuFunction(14); // 36 us delay
    makeinput();
    return;
}
  
```

Read One Byte

One byte can be read using the function created to read one bit. The flowchart and the function of reading one byte are shown below.

Figure 4. Single Byte Read Flowchart



Code 4. Single Byte Read Function

```

BYTE ds_r_byte(void)
{
    int value=0;
    char i=0;
    char bit=0;

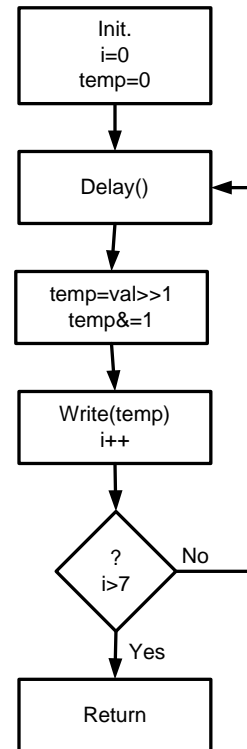
    for(i=0;i<8;i++)
    {
        bit=ds_r_bit();
        if(bit) value|=(1<<i);
        DelayuFunction(21); // 50 us
    }

    return(value);
}
  
```

Write One Byte

One byte can be written using the function created to write one bit. The flowchart and the function for writing one byte are shown below.

Figure 5. Single Byte Write Flowchart



Code 5. Single Byte Write Function

```

void ds_w_byte(BYTE val)
{
    BYTE i,temp;
    for (i=0; i<8; i++)
    {
        DelayuFunction(18); // 45 us
        temp=val>>i;
        temp&=1;
        ds_w_bit(temp);
    }

    return;
}
  
```

Delay Function

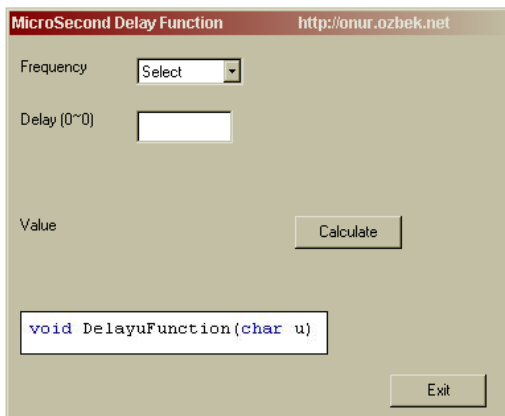
In this project, it is necessary to continuously calculate the delay value in microseconds. Therefore, a program that calculates the delay value has been written.

The format of the delay function is:

```
void DelayuFunction(char u)
```

Thus, the program calculates the 'u' value. A screen shot of the program is shown in Figure 6.

Figure 6. Delay Function Calculation Program Interface

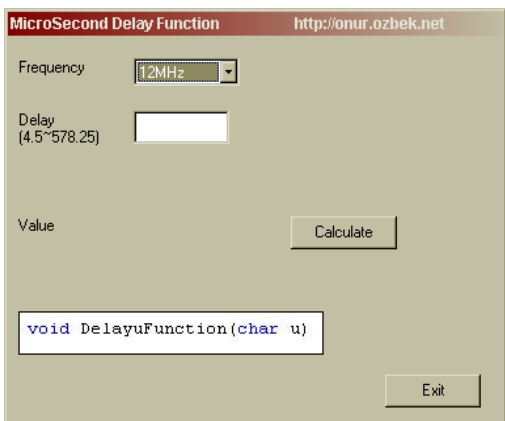


For example, if the CPU frequency of the PSoC is 12 MHz and desired delay value is 50 μ s:

1. From the frequency drop-down menu, Select 12 MHz.

The range (in microseconds) will appear in the parenthesis near the 'Delay' field (Figure 7).

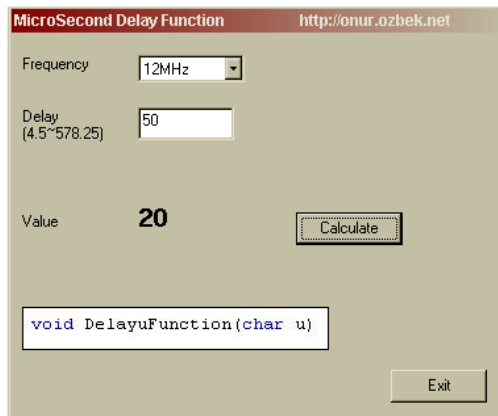
Figure 7. Delay Range in Delay Function Calculation Program



2. Enter the 50 in the Delay field.
3. Click Calculate.

The value will appear (see Figure 8).

Figure 8. Delay Value



4. Click Exit to quit.

The function for 50 μ s delay is:

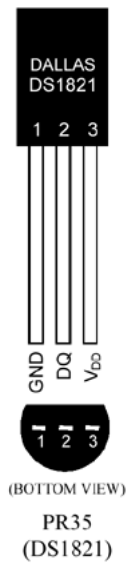
```
DelayuFunction(20); // 50 us delay.
```

Summary

In this Application Note, the 1-wire bus protocol has been discussed and a method has been offered to realize communication between a slave and master. As an application example, a digital temperature sensor has been selected. Using one pin of PSoC, the temperature is read and displayed to the LCD screen.

Appendix

Figure 9. The Digital Sensor and Pin Description (from the DS1821 Data Sheet)



PIN DESCRIPTION

- GND - Ground
- DQ - Data In/Out and Thermostat Output
- V_{DD} - Power Supply Voltage
- NC - No Connect

Figure 10. Timing Diagram for Initialization (from the DS1821 Data Sheet)

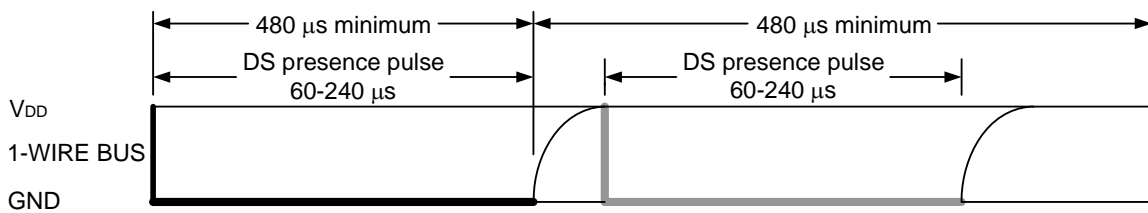
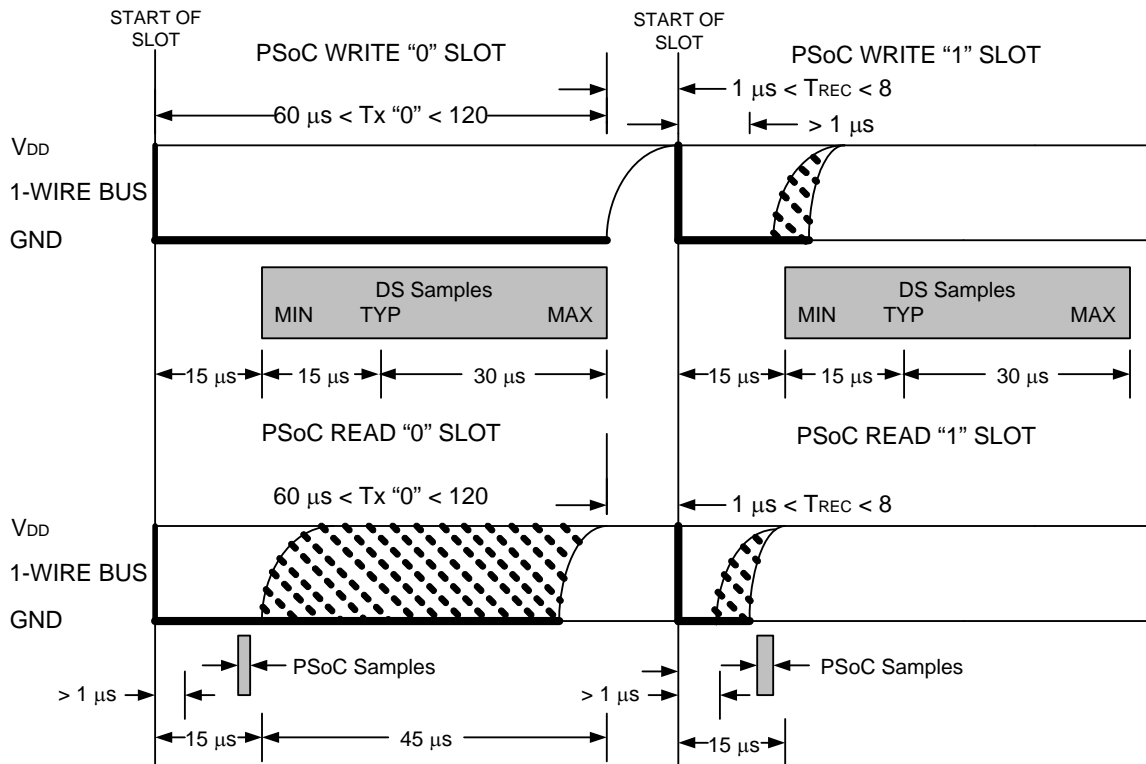


Figure 11. Timing Diagram for Read/Write Slots (from the DS1821 Data Sheet)



About the Author

Name: Onur Ozbek
Title: Electronic Engineer (BSc. EE)
Background: He graduated from Yeditepe University in Istanbul. Currently, attending a master program at Istanbul Technical University and working as an FAE at DesTEK.
Contact: onur@destekelektronik.com

In March of 2007, Cypress recataloged all of its Application Notes using a new documentation number and revision code. This new documentation number and revision code (001-xxxxx, beginning with rev. **), located in the footer of the document, will be used in all subsequent revisions.

PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," PSoC Designer, and PSoC Express are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
<http://www.cypress.com/>

© Cypress Semiconductor Corporation, 2004-2007. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.