



## King Mongkut's University of Technology Thonburi

Department of Electronics and Telecommunication Engineering Faculty of Engineering  
EIE/ENE 335 Digital Circuit and Microprocessor Lab for the 3<sup>rd</sup> year student

### Experiment: 1-Wire® interface (DS1820)

#### Objectives

- How to use
  - the NuMicro™ NUC100 series driver to do the fast application software development
  - DS1820

#### Background Theory

##### DS1820

The DS18S20 Digital Thermometer provides 9-bit centigrade temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18S20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of  $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$  and is accurate to  $\pm 0.5^{\circ}\text{C}$  over the range of  $-10^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ . In addition, the DS18S20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

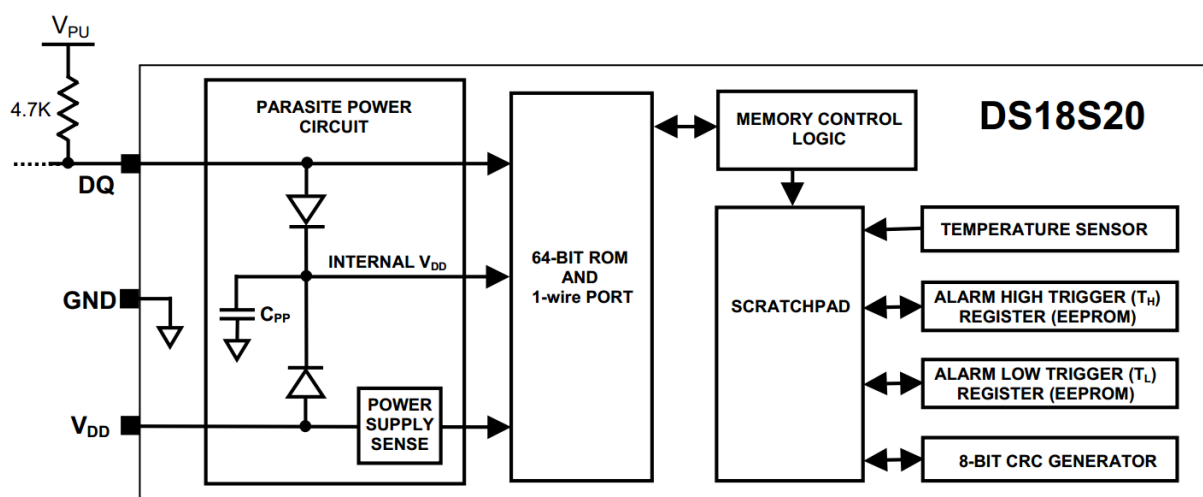


Figure 1 DS18S20 BLOCK DIAGRAM

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

Figure 2 TEMPERATURE REGISTER FORMAT

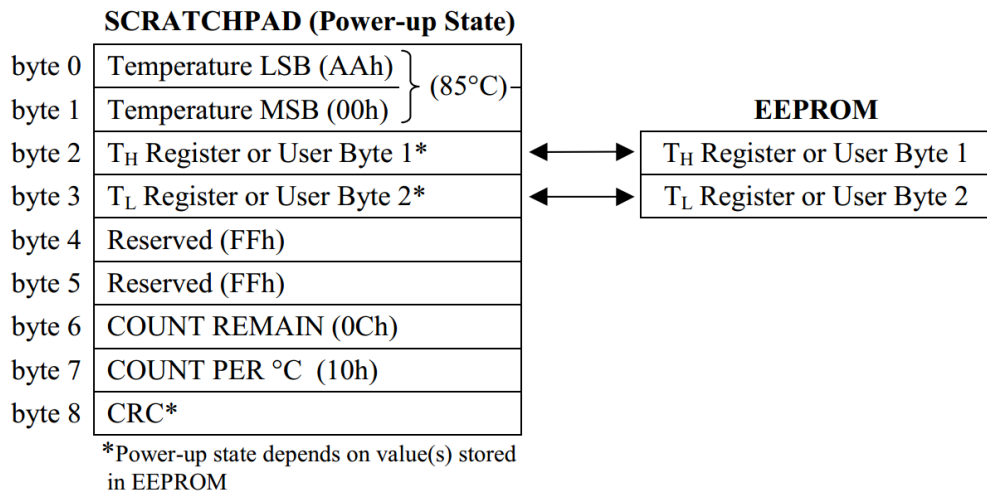


Figure 3 DS18S20 MEMORY MAP

### 1-WIRE BUS SYSTEM

The 1-Wire bus system uses a single bus master to control one or more slave devices. The DS18S20 is always a slave. When there is only one slave on the bus, the system is referred to as a “single-drop” system; the system is “multidrop” if there are multiple slaves on the bus.

### TRANSACTION SEQUENCE

The transaction sequence for accessing the DS18S20 is as follows:

Step 1. Initialization

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

Step 3. DS18S20 Function Command (followed by any required data exchange)

It is very important to follow this sequence every time the DS18S20 is accessed, as the DS18S20 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.

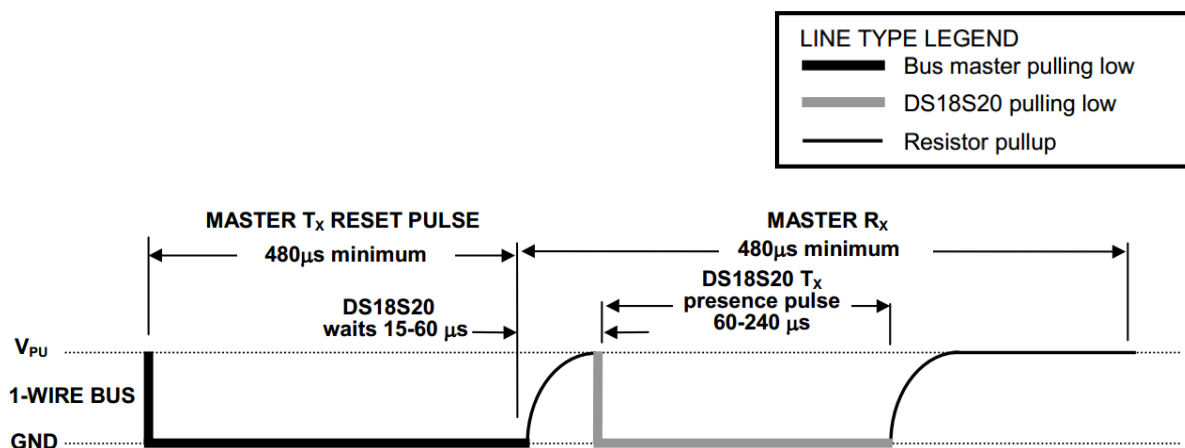


Figure 4 INITIALIZATION TIMING

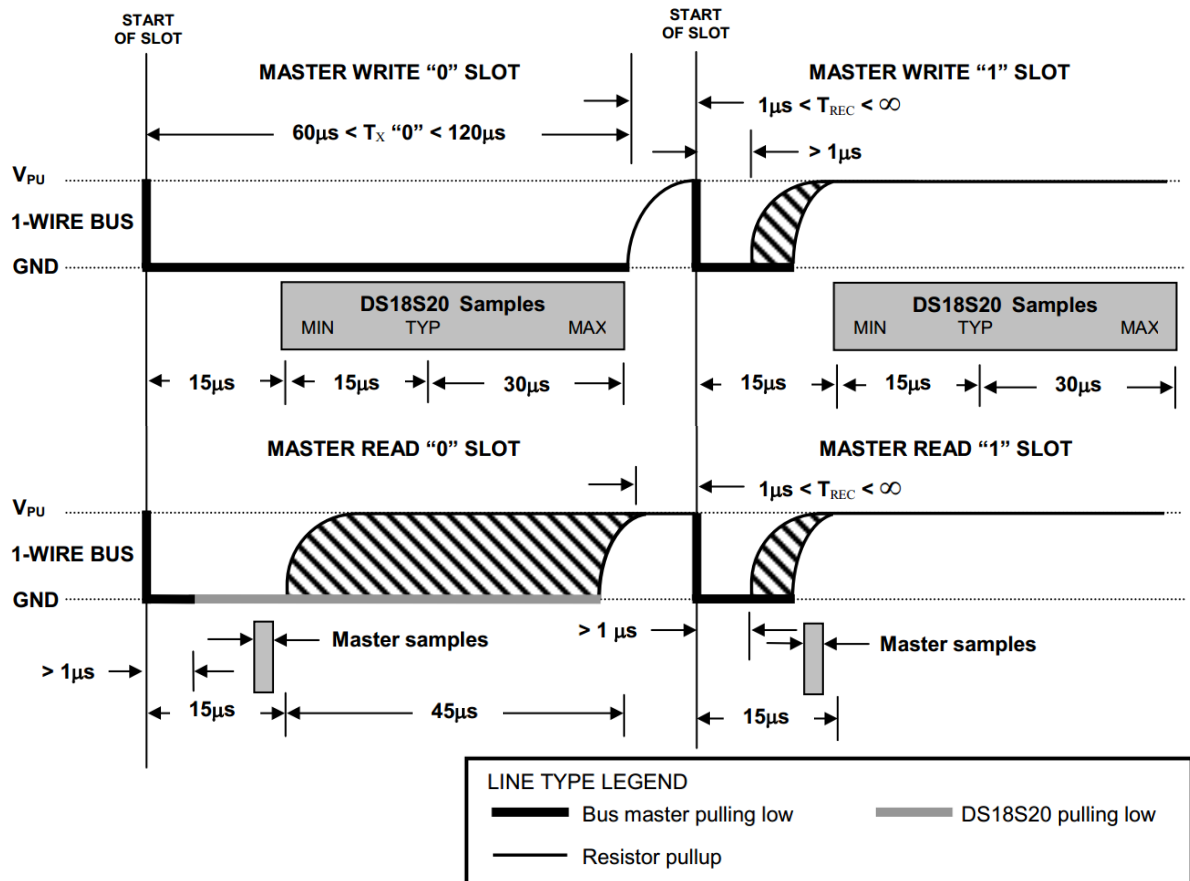


Figure 5 READ/WRITE TIME SLOT TIMING DIAGRAM

### Equipment required

- Nu\_LB-002 (Nuvoton learning board)
- The PCF8574(with DS1820) board

### Reference:

1. [Nu\\_LB-002 Rev 2.1 User's Manual](#)
2. [NuMicro™ NUC130\\_140 Technical Reference Manual EN V2.02](#)
3. [NuMicro™ NUC100 Series Driver Reference Guide V1.05.002](#)
4. [DS1820 datasheet](#)

```

172 //-----MAIN
173 int32_t main (void) {
174     UNLOCKREG();
175     DrvSYS_Open(48000000);
176     LOCKREG();
177
178     Initial_panel(); //call initial panel function
179     clr_all_panel();
180     print_lcd(0, "DS1820 Onewire");
181
182     DrvGPIO_Open(E_GPE, 8, E_IO_QUASI);
183     InitTIMER3();
184
185     while (1) {
186         _NOP();
187     }
188 }
189

```

Figure 6 a main program

## Lab09\_OneWire

## Procedure: 1-WIRE

1. Replace the content of the 'Smpl\_Start\_Kit.c' with the '1-Wire' lab file.
2. Connect Pin4 on the PCF8591 board to GPE8.
3. Compile the project, and run the program.
4. Study the program and work on assignments in the class.

```

150 //-----Timer3_IRQ
151 void TMR3_IRQHandler(void) // Timer0 interrupt subroutine
152 {
153     int8_t ds1820Temp;
154     //uint16_t Timer3Counter = 0;
155     char lcd2_buffer[18] = "Timer3:";
156     char lcd3_buffer[18] = "T = C";
157
158     sprintf(lcd2_buffer+7, " %d", Timer3Counter);
159     print_lcd(2, lcd2_buffer);
160     Timer3Counter++;
161
162     // to initiate a temperature measurement and A-to-D conversion
163     OneWireTxSkipROMConvert();
164     DrvSYS_Delay(100);
165     ds1820Temp = OneWireReadByteTemperature();
166     sprintf(lcd3_buffer+4, "%d C", ds1820Temp);
167     print_lcd(3, lcd3_buffer);
168
169     TIMERS->TISR.TIF = 1; // Write 1 to clear the interrupt flag
170 }

```

Figure 7 TMR3\_IRQHandler function

```

82 //-----OneWireTxSkipROMConvert
83 void OneWireTxSkipROMConvert(void) {
84     int8_t i;
85     uint8_t dataByte = 0xCC; // skip ROM
86
87     GPIOE->DOUT &= 0xFEFF; // Master send Reset
88     DrvSYS_Delay(500);
89     GPIOE->DOUT |= 0x0100;
90     DrvSYS_Delay(200);
91
92     for (i=0; i<8; i++) { // skip ROM
93         if ((dataByte & 0x01 == 0x01)) {
94             GPIOE->DOUT &= 0xFEFF; // send '1'
95             DrvSYS_Delay(3); // low > 1 microsec.
96             GPIOE->DOUT |= 0x0100;
97             DrvSYS_Delay(60);
98         } else {
99             GPIOE->DOUT &= 0xFEFF; // send '0'
100             DrvSYS_Delay(60); // low > 60 microsec.
101             GPIOE->DOUT |= 0x0100;
102             DrvSYS_Delay(2);
103         }
104         dataByte >>= 1;
105     }
106
107     dataByte = 0x44; // convert Temperature
108     for (i=0; i<8; i++) {
109         if ((dataByte & 0x01 == 0x01)) {
110             GPIOE->DOUT &= 0xFEFF; // send '1'
111             DrvSYS_Delay(3); // low > 1 microsec.
112             GPIOE->DOUT |= 0x0100;
113             DrvSYS_Delay(60);
114         } else {
115             GPIOE->DOUT &= 0xFEFF; // send '0'
116             DrvSYS_Delay(60); // low > 60 microsec.
117             GPIOE->DOUT |= 0x0100;
118             DrvSYS_Delay(2);
119         }
120         dataByte >>= 1;
121     }
}

```

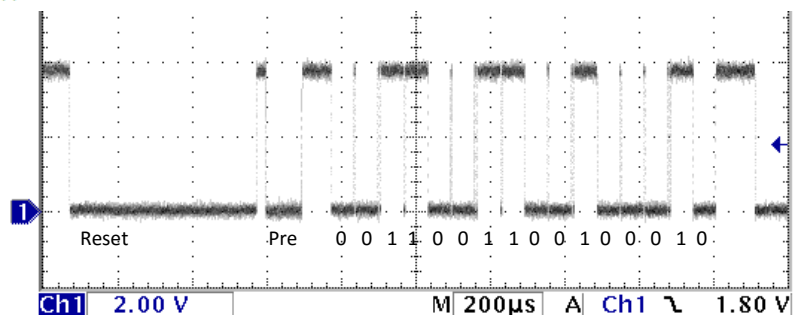


Figure 8 an OneWireTxSkipROMConvert function

## Lab09\_OneWire

```

21 //-----OneWireReadByteTemperature
22 int8_t OneWireReadByteTemperature(void) {
23     int8_t i;
24     int8_t dataByte = 0xCC; // skip ROM
25
26     GPIOE->DOUT &= 0xFEFF; // Master send Reset
27     DrvSYS_Delay(500);
28     GPIOE->DOUT |= 0x0100;
29     DrvSYS_Delay(200); // wait for presence pulse
30
31     for (i=0;i<8;i++) { // skip ROM
32         if ((dataByte&0x01 == 0x01)) {
33             GPIOE->DOUT &= 0xFEFF; // send '1'
34             DrvSYS_Delay(3); // low > 1 microsec.
35             GPIOE->DOUT |= 0x0100;
36             DrvSYS_Delay(60);
37         } else {
38             GPIOE->DOUT &= 0xFEFF; // send '0'
39             DrvSYS_Delay(60); // low > 60 microsec.
40             GPIOE->DOUT |= 0x0100;
41             DrvSYS_Delay(2);
42         }
43         dataByte >>= 1;
44     }
45
46     dataByte = 0xBE; // ReadScratchpad
47     for (i=0;i<8;i++) {
48         if ((dataByte&0x01 == 0x01)) {
49             GPIOE->DOUT &= 0xFEFF; // send '1'
50             DrvSYS_Delay(3); // low > 1 microsec.
51             GPIOE->DOUT |= 0x0100;
52             DrvSYS_Delay(60);
53         } else {
54             GPIOE->DOUT &= 0xFEFF; // send '0'
55             DrvSYS_Delay(60); // low > 60 microsec.
56             GPIOE->DOUT |= 0x0100;
57             DrvSYS_Delay(2);
58         }
59         dataByte >>= 1;
60     }
61
62     // read 8 bits (byte0 scratchpad)
63     DrvSYS_Delay(100);
64     for (i=0;i<8;i++) {
65         GPIOE->DOUT &= 0xFEFF; //
66         DrvSYS_Delay(2); // low > 1 microsec.
67         GPIOE->DOUT |= 0x0100;
68         // Read
69         DrvSYS_Delay(12);
70         if ((GPIOE->PIN &= 0x0100) == 0x0100) {
71             dataByte >>= 1;
72             dataByte |= 0x80;
73         } else {
74             dataByte >>= 1;
75             dataByte &= 0x7F;
76         }
77         DrvSYS_Delay(60);
78     }
79     dataByte >>= 1;
80     return dataByte;
81 }

```

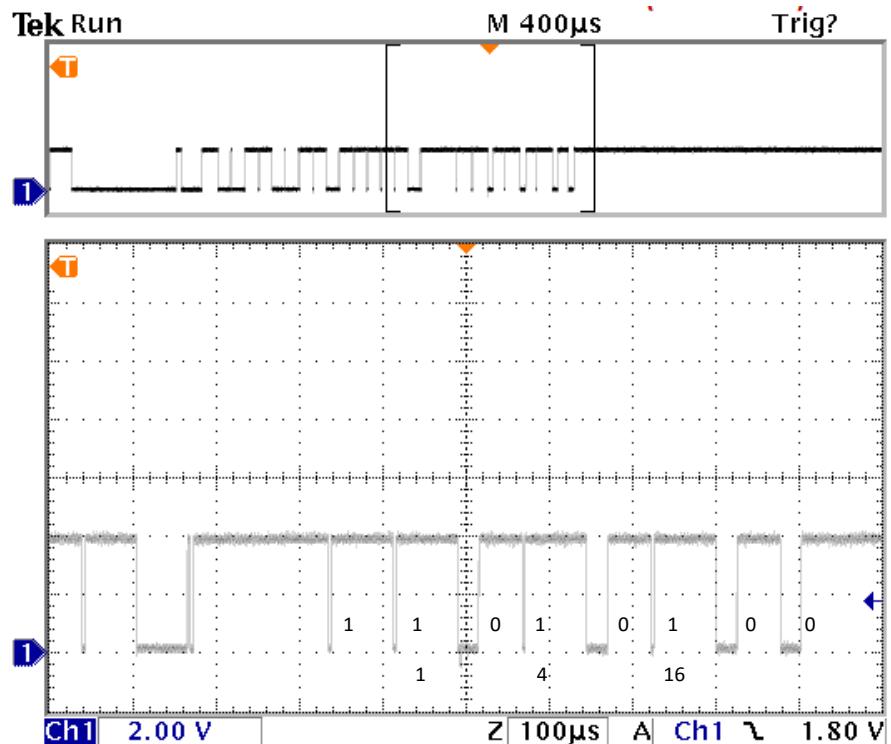


Figure 9 an OneWireReadByteTemperature function (21.5 C)

Lab09\_OneWire

Assignment(s)

Summarize what you suppose to learn in this class.