Communication - 1-Wire User Modules (Introduction)

AN2166

Author: Wes Randall

Associated Project: Yes
Associated Part Families: CY8C21xxx, CY8C24xxx, CY8C27x43
CY28xxx, CY29x66
Software Version: PSoC Designer™ 5.0 SP6
Associated Application Notes: AN51234, AN50987

Application Note Abstract

The Dallas Semiconductor 1-Wire® and iButton® products are a family of devices that communicate over a single wire and a ground. Most devices have a unique 8-byte ROM code, which can be used as an address. Multiple addressable devices may be attached to the same signal wire simultaneously. The 1-Wire user modules provide easy access to these devices. This application note introduces the basic functions of the 1-Wire user modules and gives simple examples of their operation.

Introduction

1-Wire user modules are provided in two forms. The OneWire user module uses two digital blocks to perform the timing and bit-level input and output. The user modules described here implement the master role. The OneWireSW User Module is implemented entirely in software. It uses the CPU to perform all of the timing and functions.

Common Features

- Bit and byte read/write functions
- 8 and 16 bit cyclic redundancy check (CRC) functions
- Support for parasite power
- 1-Wire search functions

Software Features

- No digital blocks (requires 12 MHz CPU speed)
- Only one device pin used

Hardware Features

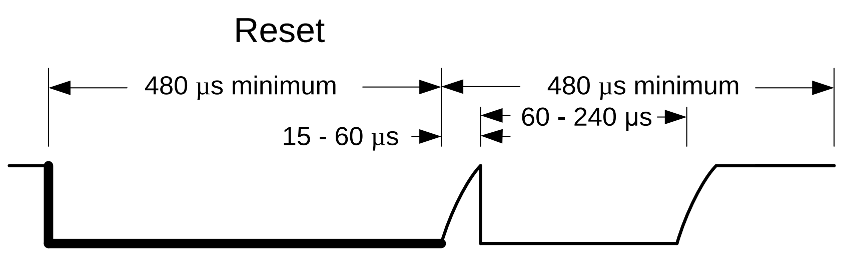
- Any CPU speed (requires 3 MHz clock input)
- Support for parasite power with 6 MHz or higher CPU speed
- Support for overdrive speed
- Interrupts do not disrupt communication

Installation

The OneWire(hardware) user module is included with PSoC Designer™ as of PD5.0 SP6 and requires no special user intervention for use.

The OneWireSW(software) user module is not included with PSoC Designer. It must be downloaded from the website www.psocdeveloper.com and installed separately. It is stored in a file called *OneWire.zip*.

Figure 1. Reset Timing with Presence Detect



[+] Feedback

1-Wire Overview

A 1-Wire data line is an open-drain with a resistive pull up. In this case, the pull up is internal to the PSoC®. The PSoC initiates all signaling. The signal types used by the user module are:

- reset with presence detect
- write 0
- write 1
- read 0
- read 1

Reset

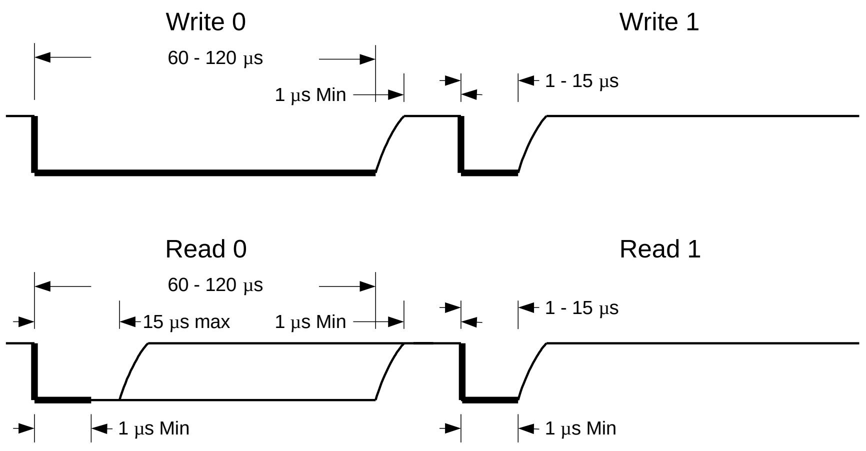
Communication with devices starts with a reset. The master pulls the line low for a minimum of 480 µs and then releases it, enabling the pull up to return the line high. The devices indicate their presence by waiting 15 to 60 µs after the line returns high, then pulling the line low for 60 to 240 µs. The reset takes place during a time slot of at least 960 µs. Figure 1 on page 1 shows the timing of the reset and presence pulse. Bold lines indicate where the PSoC is driving the line low.

Write and Read

Write and read operations take place during a time slot of 60 to 120 µs with a minimum 1 µs recovery time. To write a 0, the master pulls the line low and holds it through the end of the time slot. To write a 1, the master pulls the line low and releases it within 15 µs. The pull up returns the line high for the remainder of the time slot. A read cycle takes place after an instruction that requests data from a device. The master pulls the line low for a minimum of 1 µs and then releases it. At this point the active device outputs a 1 or 0. The device outputs a 1 by letting the pull up return the line high. The device outputs a 0 by holding the line low for 15 µs after the falling edge from the master and releasing it by the end of the time slot. Figure 2 shows the timing of write and read time slots. Bold lines indicate where the PSoC is driving the line low.

The byte functions are made of multiple calls to the bit functions. The least significant bit is transmitted or received first.

Figure 2. Write and Read Timing



Example Project

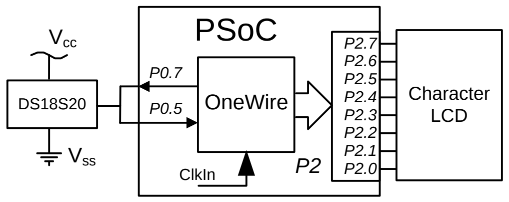
An example project is provided for the CY8C27xxx family. This project reads the temperature from a DS18S20 temperature sensor and outputs the value to a character LCD on port 2. Register and data format information for the DS18S20 is included in the Appendix. Several different models of temperature sensors are available. The different models tend to have different data formats.

Consult the Dallas Semiconductor data sheets for more information on the DS18S20 and other 1-Wire devices.

Hardware Project

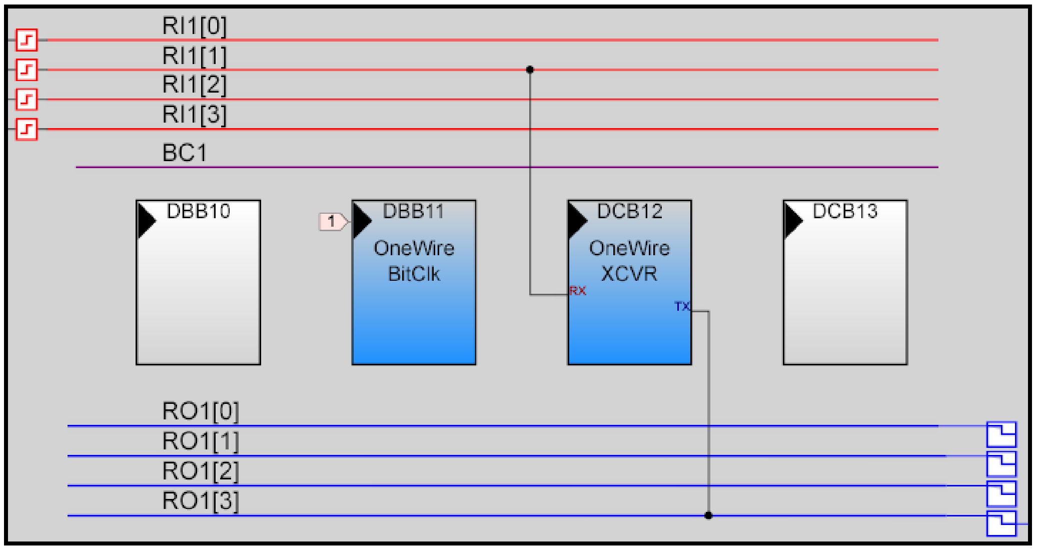
The OneWire(hardware) example project is meant to operate in a system that is configured as shown in Figure 3.

Figure 3. Hardware Example Project Block Diagram



The hardware version of the user module, **OneWire**, uses two digital blocks. Figure 4 shows the block diagram for the hardware example project. The two blocks used in the hardware version of the user module are called BitClk and XCVR. The BitClk block is a programmable divider that generates the clock signal used by the XCVR block. The BitClk block may be either a basic or a communication block. The XCVR block is the output and input block for the 1-Wire data.

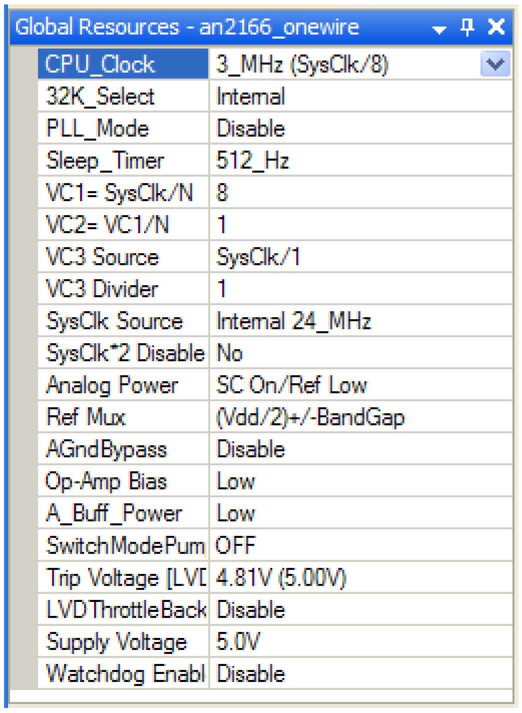
Figure 4. Hardware Example Project Block Placement



The XCVR block must be a communication block. If the blocks are placed so that the BitClk block is in front of and adjacent to the XCVR block, the clock input of the XCVR block can be made by chaining it to the BitClk block. In this manner the interconnect resources are conserved. Figure 4 shows the block placement used in the example projects.

The global resources for the project are shown in Figure 5. The **CPU_Clock** speed used is not important. However, the delay functions will be accurate only when the **CPU_Clock** parameter is set to 1.5 MHz or higher. If the CPU clock is slower, the delay functions yield longer delays. The clock source to the BitClk block in the example is **VC1**. The **VC1** parameter is set to 8 for a clock frequency of 3 MHz, the required speed for the user module.

Figure 5. Global Resources



Global Resources - an2166_onewire

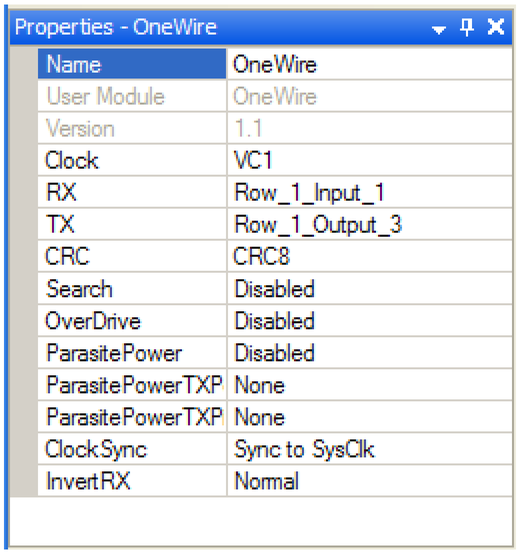
CPU_Clock	3_MHz (SysClk/8)
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	8
VC2= VC1/N	1
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal 24_MHz
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePum	OFF
Trip Voltage [LVD	4.81V (5.00V)
LVDThrottleBack	Disable
Supply Voltage	5.0V
Watchdog Enabl	Disable

The user module parameters are as follows. **ClkIn** is the input to the BitClk block. Here it is set to **VC1**. The **BitClkOut** is the output of the BitClk block. Here it is set to **None** because the blocks are adjacent and may be chained. The **BitClkIn** is the input to the XCVR block. It is set to **DBB11**, which is the location of the BitClk block. The **RX** parameter is the input to the XCVR block. The value **Row_1_Input_1** will reach the input **Port_0_5**. The **TX** parameter is the input to the XCVR block. The value **Row_1_Output_3** will reach the output **Port_0_7**. Note that the hardware version of the user module uses two pins. These two pins must be connected together externally. The output pin must be set to use the pull up mode. Two pins are necessary because pins connected to digital blocks cannot simultaneously be both inputs and outputs.

The remaining parameters are optional except for the **InvertRX** parameter. It must be set to **Normal**. The optional parameters may be disabled to save RAM and program space. The **CRC16** parameter determines whether the 16-bit CRC functions are available for data-integrity checking. The **Search** parameter determines whether the 1-Wire search functions are available. These are needed when more than one 1-Wire device is present and the ROM numbers are unknown. The **OverDrive** parameter can be used to enable overdrive speeds, which some 1-Wire devices support. The overdrive speed is about eight times faster than the normal speed. The **ParasitePower** parameter enables the parasite power functions.

Figure 6 shows the user module parameters for **OneWire**.

Figure 6. OneWire User Module Parameters

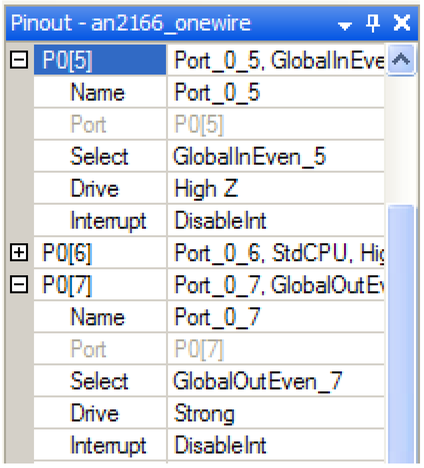


Properties - OneWire

Name	OneWire
User Module	OneWire
Version	1.1
Clock	VC1
RX	Row_1_Input_1
TX	Row_1_Output_3
CRC	CRC8
Search	Disabled
OverDrive	Disabled
ParasitePower	Disabled
ParasitePowerTXP	None
ParasitePowerTXP	None
ClockSync	Sync to SysClk
InvertRX	Normal

The pin configuration is shown in Figure 7. Port _0_5 is configured as an input by setting the Select parameter to GlobalInEven_5. Port_0_7 is configured as an output by setting the **Select** parameter to GlobalOutEven_7. The drive mode is set to Pull Up. Port_1_0 through Port_1_7 must have drive strength of Pull Down or Strong to drive the LEDs.

Figure 7. I/O Pin Parameters



Pinout - an2166_onewire

⊟ P0[5]	Port_0_5, GlobalInEve
Name	Port_0_5
Port	P0[5]
Select	GlobalInEven_5
Drive	High Z
Interrupt	DisableInt
⊞ P0[6]	Port_0_6, StdCPU, Hi
⊟ P0[7]	Port_0_7, GlobalOutE
Name	Port_0_7
Port	P0[7]
Select	GlobalOutEven_7
Drive	Strong
Interrupt	DisableInt

Application Programming Interface (API)

Code 1 lists the functions used in the example projects. The instance name in the projects is **OW**. The entire set of functions is described in the user module data sheets.

Table 1. HW Example Project Functions

Function	Action
OneWire_Start	Initializes the 1-Wire interface.
OneWire_Reset	Performs a 1-Wire reset. Returns true if one or more devices are present.
OneWire_WriteByte	Writes a byte to one or more 1-Wire device.
OneWire_ReadByte	Returns a byte from a 1-Wire device.

HW Project Code in C

Code 1.HW Project C Code

```
#include <m8c.h>
#include "PSoCAPI.h"
#include "disclaimer.h"

char cTens(char cInput);
char cOnes(char cInput);

char cTemp;
char bfTempSign;
char cWait;
char acPrintString[ ];

void main()
{
  OneWire_Start();
  LCD_Start();
  LCD_Position(0,0);
  LCD_PrCString("Temp:");

  LCD_Position(0,12);
  LCD_PrCString("C");
  while(1)
  {
    OneWire_fReset();
    OneWire_WriteByte(0xCC);
    // Skip ROM
    OneWire_WriteByte(0x44);
    // Start Conversion
    // Delay
    cWait = 0xff;
    while(cWait--);

    OneWire_fReset();
    OneWire_WriteByte(0xCC);
    // Skip ROM
    OneWire_WriteByte(0xBE);
    // Read Scratch Pad

    cTemp = OneWire_bReadByte();
    bfTempSign = OneWire_bReadByte();
    LCD_Position(0,6);

    //Detect negative
    if(bfTempSign)
    {
      acPrintString[0] = 45; //'-'
      cTemp = ~cTemp;
    }
    else
    {
      acPrintString[0] = 43; //'+'
    }

//Format string for printing on LCD
    acPrintString[1] =
0x30+cTens(cTemp>>1);
    acPrintString[2] =
0x30+cOnes(cTemp>>1);
    acPrintString[3] = 0; //null character

    LCD_PrString(acPrintString);

    //add LSB of data
    if(cTemp<<7)
      LCD_PrCString(".5");
    else
      LCD_PrCString(".0");
  } // end for loop
} // end main
```

Note Helper functions are not included here.

Software Project

The OneWireSW User Module and example code may be downloaded at www.psocdeveloper.com in the download area.

Summary

This application note, along with the example project, gives an introduction to using Dallas Semiconductor 1-Wire devices with the PSoC user modules, **OneWireSW** and **OneWire**. These user modules provide a solid base for projects using 1-Wire devices. The Dallas Semiconductor web site www.maxim-ic.com, is an excellent source of more information on available 1-Wire devices and their operation.

About the Author

Name:	Wes Randall
Title:	Staff Design Engineer Cypress Semiconductor
Contact:	xwr@cypress.com

Document History

Document Title: Communication - 1-Wire User Modules (Introduction)

Document Number: 001-35321

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1513664	VED	9/26/2007	Created
*A	2521399	VED	6/25/2008	Updated to reflect template changes. Updated copyright. Added source disclaimer, revision disclaimer, Samples Request Form link. This note had no technical updates.
*B	2797308	MAXK	11/03/2009	Updated to most recent AN template. Updated for PD5.0 SP6. Added LCD for temp display. Removed SW example code.

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