

1-Wire User Modules (Introduction)

By: Wes Randall

Associated Project: Yes

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

PSoC Designer Version: 4.1

The Dallas Semiconductor 1-Wire® and iButton®_[1] 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 **OneWireSW** User Module is implemented entirely in software. It uses the CPU to perform all of the timing and functions. 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.

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 (12 MHz or higher for CY8C25/26xxx)
- Support for overdrive speed
- Interrupts do not disrupt communication

Installation

The 1-Wire user modules currently are not included with PSoC Designer. They must be downloaded and installed separately. The user modules may be downloaded from the web site www.psocdeveloper.com in the download area. Both modules and example projects are together in a file called *OneWire.zip*. The user modules and example projects are contained in a folder called "OneWire." This folder must be moved into the Data directory of PSoC Designer (e.g., ProgramFiles\Cypress Microsystems\PSoC Designer\Data\OneWire). The user modules appear in the "Digital Comm" category after PSoC Designer is restarted. The example projects folder, named "Example Projects", should be moved to a folder such as "My Documents."

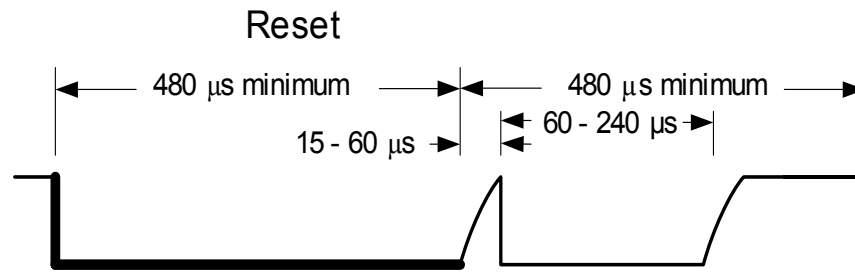


Figure 1. Reset Timing with Presence Detect

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 μ s and then releases it, allowing the pull up to return the line high. The devices indicate their presence by waiting 15 to 60 μ s after the line returns high, then pulling the line low for 60 to 240 μ s. The reset takes place during a time slot of at least 960 μ s. Figure 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 μ s with a minimum 1 μ s 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 μ s. 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 μ s and then releases it. At this point the active device will output 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 μ s 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.

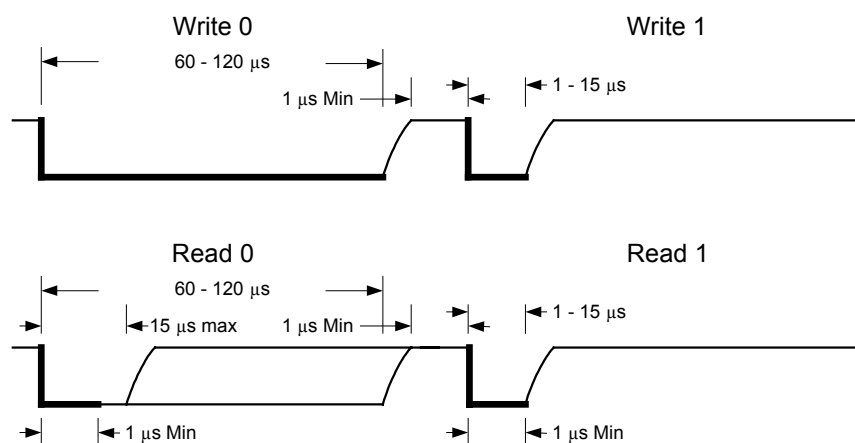


Figure 2. Write and Read Timing

Example Projects

Example projects are provided for the CY8C27xxx family. These projects read the temperature from a DS18S20 temperature sensor and output the value to eight LEDs in a continuous loop on port 1. 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.

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

Software Project

The software version of the example project is called OWSWapp27C. The main code is written in C. The CY8C27xxx hardware example projects are called OWHWapp27A and OWHWapp27C. They are written in assembly and C, respectively. Figure 3 shows the block diagram for the software example project.

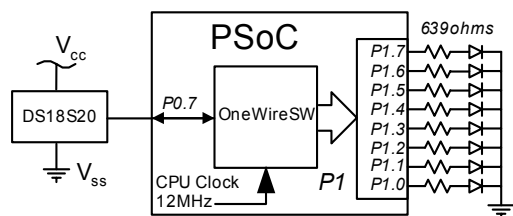


Figure 3. Software Example Project Block Diagram

The only global resource parameter that is required for the software 1-Wire user module, OneWireSW, is CPU_Clock. It must be set to 12 MHz for correct timing. Figure 4 shows the global resource selections.

Global Resources	
CPU_Clock	12_MHz (SysClk/2)
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	1
VC2= VC1/N	1
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal 24_MHz

Figure 4. Global Resources

Figure 5 shows the user module parameters for **OneWireSW**. The parameter **DQIOPort** sets the port to be used for the 1-Wire bus. The **DQPin** parameter sets the pin within the port set by **DQIOPort**. The drive mode of the pin set in **DQPin** will be set to Pull Up. **CRC16** and **Search** are not used in the example project and can be disabled to save RAM and program space. The **CRC16** parameter determines if the 16-bit CRC functions will be available for data-integrity checking. Not all 1-Wire devices use a 16-bit CRC. The **Search** parameter determines whether the 1-Wire search functions will be available. These are needed when more than one 1-Wire device is present and the ROM numbers are unknown.

User Module Parameters	
DQIOPort	Port_0
DQPin	Port_0_7
CRC16	Enable
Search	Enable

Figure 5. OneWireSW User Module Parameters

Hardware Project

The hardware version of the user module, **OneWire**, uses two digital blocks. Figure 6 shows the block diagram for the hardware example project.

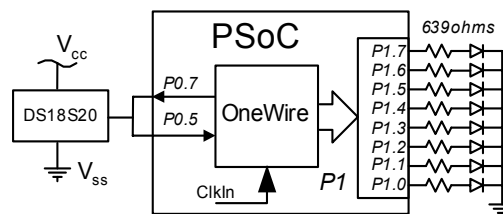


Figure 6. Hardware Example Project Block Diagram

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.

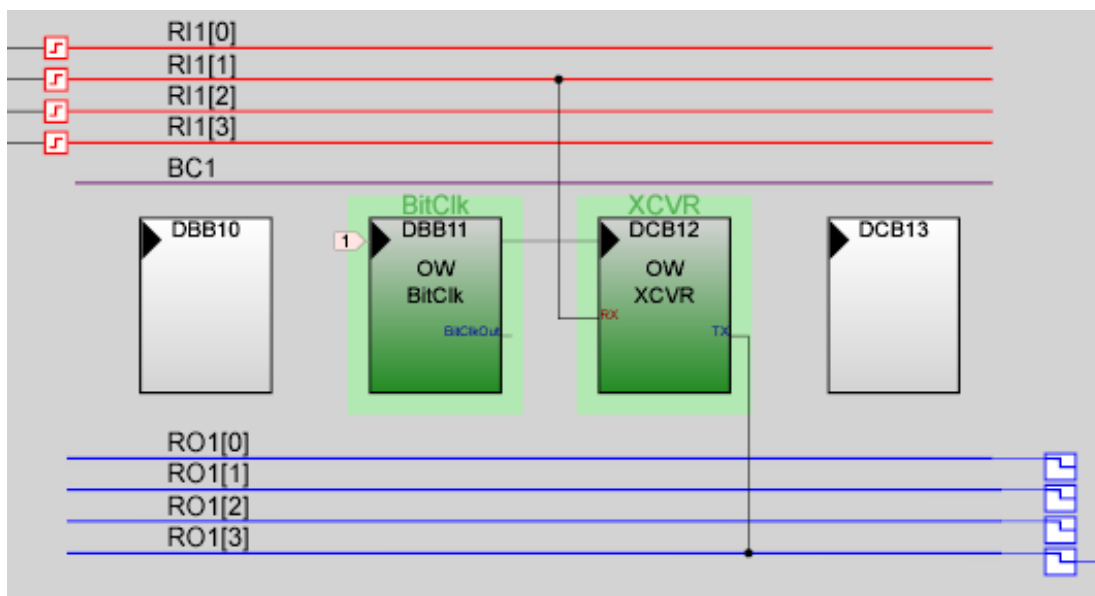


Figure 7. Hardware Example Project Block Placement

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 7 shows the block placement used in the example projects.

The global resources for **OneWire** are shown in Figure 8. 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.

Global Resources	
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

Figure 8. Global Resources

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** since 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 Pull Up mode. Two pins are necessary since 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 9 shows the user module parameters for **OneWire**.

User Module Parameters	
ClkIn	VC1
BitClkOut	None
BitClkIn	DBB11
RX	Row_1_Input_1
TX	Row_1_Output_3
CRC16	Disable
Search	Disable
OverDrive	Disable
ParasitePower	Disable
InvertRX	Normal

Figure 9. OneWire User Module Parameters

The pin configuration is shown in Figure 10. **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** need to have drive strength of Pull Down or Strong to drive the LEDs.

Name	Port	Select	Drive	Interrupt
Port_0_0	P0[0]	StdCPU	Pull Down	DisableInt
Port_0_1	P0[1]	StdCPU	Pull Down	DisableInt
Port_0_2	P0[2]	StdCPU	Pull Down	DisableInt
Port_0_3	P0[3]	StdCPU	Pull Down	DisableInt
Port_0_4	P0[4]	StdCPU	Pull Down	DisableInt
Port_0_5	P0[5]	GlobalInEv	High Z	DisableInt
Port_0_6	P0[6]	StdCPU	Pull Down	DisableInt
Port_0_7	P0[7]	GlobalOutE	Pull Up	DisableInt
Port_1_0	P1[0]	StdCPU	Pull Down	DisableInt
Port_1_1	P1[1]	StdCPU	Pull Down	DisableInt
Port_1_2	P1[2]	StdCPU	Pull Down	DisableInt
Port_1_3	P1[3]	StdCPU	Pull Down	DisableInt
Port_1_4	P1[4]	StdCPU	Pull Down	DisableInt

Figure 10. I/O Pin Parameters

Application Programming Interface (API)

Table 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. Example Project Functions

Function	Action
<i>OW_Start</i>	Initializes the 1-Wire interface.
<i>OW_Reset</i>	Performs a 1-Wire reset. Returns true if one or more devices are present.
<i>OW_WriteByte</i>	Writes a byte to one or more 1-Wire device.
<i>OW_ReadByte</i>	Returns a byte from a 1-Wire device.
<i>OW_Delay10m</i>	Waits 10 ms, then returns.
<i>OW_Delay10mTimes</i>	Calls <i>OW_Delay10m</i> the specified number of times (1-255).

Project Code in Assembly

```

;-----;
Assembly main line
;-----
include "m8c.inc"
; part specific constants and macros
include "PSOCAPI.inc"
; PSoC API definitions for all User Modules

export _main

_main:

    ; Insert your main assembly code here.
    call OW_Start
    M8C_EnableGInt

Loop:
    call OW_Reset
    mov A, CCh
; Skip ROM
    call OW_WriteByte
    mov A, 44h
; Start conversion
    call OW_WriteByte
    call Delay750m
; Wait for conversion to finish
    call OW_Reset
    mov A, CCh
; Skip ROM
    call OW_WriteByte
    mov A, BEh
; Read scratchpad
    call OW_WriteByte
    call OW_ReadByte
; Read Temperature LSB
    mov REG[PRT1DR], A
; Send to Port 1
    jmp Loop
    ret

Delay750m:
    mov A, 4Bh
    call OW_Delay10mTimes
    ret

```

Code 1. Project Assembly Code

Project Code in C

```

//-----
// C main line
//-----

#include <m8c.h>
#include "PSoCAPI.h"
// This is a generated file that contains the
include
// files for the User Modules in this project.
After
// Config>>Generate Application has been run,
this
// line can be uncommented.
void main()
{
    OW_Start();
    for (;1;)
    {
        OW_Reset();
        OW_WriteByte(0xCC);
// Skip ROM
        OW_WriteByte(0x44);
// Start Conversion
        OW_Delay10mTimes(75);
        OW_Reset();
        OW_WriteByte(0xCC);
// Skip ROM
        OW_WriteByte(0xBE);
// Read Scratch Pad

        PRT1DR = OW_ReadByte();

    } // end for loop
} // end main

```

Code 2. Project C Code

Conclusion

This Application Note, along with the example projects, 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 at 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

[1] 1-Wire® and iButton® are registered trademarks of Dallas Semiconductor.

Appendix: DS18S20 Register Formats and Memory Map

Table 2. Temperature and Alarm Register Format

Temperature Format (°C)								
	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}
MS Byte	S	S	S	S	S	S	S	S
T_L T_H	S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Table 3. Scratchpad Memory Map

Scratchpad Memory Map	
Byte 0	Temperature LSB
Byte 1	Temperature MSB
Byte 2	T_H Register
Byte 3	T_L Register
Byte 4	Reserved
Byte 5	Reserved
Byte 6	Count Remain
Byte 7	Count per °C
Byte 8	CRC

Table 4. 64-Bit ROM Code

64-Bit ROM Code		
MSB		LSB
8-Bit CRC	48-Bit Serial Number	8-Bit Family Code (10h)

Cypress MicroSystems, Inc.
 2700 162nd Street SW, Building D
 Lynnwood, WA 98037
 Phone: 800.669.0557
 Fax: 425.787.4641

<http://www.cypress.com/> / <http://www.cypress.com/support/mysupport.cfm>

Copyright © 2004 Cypress MicroSystems, Inc. All rights reserved.

PSoC™ (Programmable System-on-Chip) and PSoC Designer are trademarks of Cypress MicroSystems, Inc.

^[1] 1-Wire® and iButton® are registered trademarks of Dallas Semiconductor.

All other trademarks or registered trademarks referenced herein are property of the respective corporations.

The information contained herein is subject to change without notice.