Design Aids - PSoC IO Pin-Port Configuration

AN2094

Author: Mehmet Zeki SONMEZ
Associated Project: Yes
Associated Part Family: CY8C24x23A, CY8C24x94, CY8C21x34, CY8C20x34,CY8C21x23, CY8C27x43, CY8C29x66, CY7C64215, CYWUSB6953
GET FREE SAMPLES HERE
Software Version: PSoC Designer™ 4.4
Associated Application Notes: None

Application Note Abstract

This tutorial focuses on reconfiguring port pins using firmware.

Introduction

IO pin configuration is performed using two methods. The first method is to define the configuration as part of the initialization in the Device Editor. The easiest way to configure the pins is if the pin configuration is fixed at all times.

The second method is to use firmware to reconfigure a pin. Pin configuration may be changed at any time by using assembler or C code.

Device Editor Configuration

IO pins may be configured by using the Pinout View mode of the Device Editor. Inside the Pinout View mode, a table appears in the lower left corner of the PSoC Designer interface. This table is shown in Figure 1.

Figure 1. Pinout Table ('Select' List)

Name	Port	Select	Drive	Interrupt
Port_0_0	P0[0]	StdCPU	High Z Analog	DisableInt
Port_0_1	P0[1]	AnalogInput		DisableInt
Port_0_2	P0[2]	Default		DisableInt
Port_0_3	P0[3]	GlobalInEven_0		DisableInt
Port_0_4	P0[4]	GlobalOutEven_0		DisableInt
Port_0_5	P0[5]	StdCPU	High Z Analog	DisableInt
Port_0_6	P0[6]	StdCPU	High Z Analog	DisableInt
Port_0_7	P0[7]	StdCPU	High Z Analog	DisableInt
Port_1_0	P1[0]	StdCPU	High Z Analog	DisableInt
Port_1_1	P1[1]	StdCPU	High Z Analog	DisableInt
Port_1_2	P1[2]	StdCPU	High Z Analog	DisableInt
Port_1_3	P1[3]	StdCPU	High Z Analog	DisableInt
Port_1_4	P1[4]	StdCPU	High Z Analog	DisableInt
Port_1_5	P1[5]	StdCPU	High Z Analog	DisableInt
Port_1_6	P1[6]	StdCPU	High Z Analog	DisableInt

In the first column of Figure 1, the name of the pin is shown. For example, Port_2_5 means pin 5 of port 2, which corresponds to pin 6 of a 28 pin CY8C27443 device. You can rename the pin to any other meaningful name.

In the second column of the same figure, the ID of the port is shown.

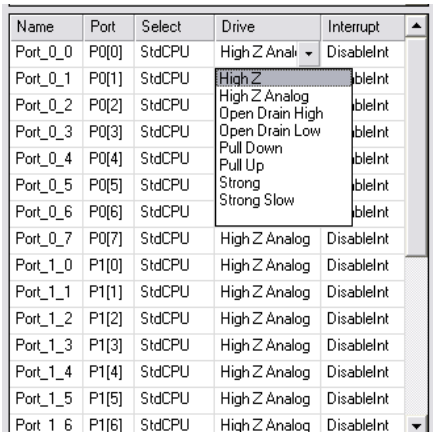
In the third column, the type of the pin is shown. A breakdown of pin types follows:

- **AnalogInput**. Only Port 0 and Port 2 have additional analog input and analog output options. AnalogInput gets analog signals from the outside world and connects to the analog column input MUX or to PSoC blocks directly. For example, if you use an ADC, you must configure at least one of the pins as AnalogInput to get analog signals from the outside world.
- **AnalogOutputBuf**. Only Port 0 has additional analog output options.
- **Default**. The global bus is not connected and the drive strength is High Z Analog.
- **StdCPU**. Normal IO through the port. This is controlled by the CPU.
- **Global_IN, Global_OUT**. Global inputs and outputs provide capability to route clock and data signals to the digital PSoC blocks. If you configure a pin as a Global_IN (input) or Global_OUT (output), then that pin can talk to the digital blocks. For example, if the Global_IN is selected, then this selection connects that particular pin to the Global_INPUT bus. This bus is then used as an input to the **digital PSoC blocks**.

Apart from the previously mentioned pin types, there are pins that have special features, and are listed. For example, P1[0] and P1[1] have XtalOut and XtalIn, P1[4] has ExtSysClk, P1[5] and P1[7] have I2C_SDA and I2C_SCL, and so on.

In the fourth column of Figure 2, the drive property of the pin is shown.

Figure 2. Pinout Table ('Drive' List)



You can configure the pin as input or output by selecting the drive property. A breakdown of the drive properties follows:

High Z. Use High Z for the pin to be input digital.

High Z Analog. This mode is similar to the High Z state, but the digital input section is disabled. Select this mode to use this pin as an analog input or output. This mode must be selected if the pin is unused. Each GPIO pin has a Schmitt trigger cell which interfaces with the internal digital data bus. In High Z Analog mode, this Schmitt trigger cell is disabled. This prevents oscillations at the Schmitt trigger output when the analog signal on the pin is at threshold levels.

Open Drain High. In this mode, the HIGH output is driven with a strong drive. The LOW output is High Z.

Open Drain Low. In this mode, the LOW output is driven with a strong drive and HIGH output is open. This mode is suitable for I2C bus where external pull up resistors are used.

Strong. Use the Strong mode if your pin is output. The pin has a low impedance connection to VSS and VDD. Do not use the Strong mode if the pin is an input.

Pull Down. In this mode, HIGH output is driven strong, and LOW output is through an internal pull down resistor of approximately 5.6K. This mode is used as an input, for example with a switch connected to VCC. This mode is also used as output.

Pull Up. This mode is the opposite of the Pull Down mode. In this mode, HIGH output is driven strong and LOW output is through an internal pull down resistor of approximately 5.6K. This mode is used as an input, for example with a switch connected to GND. When used as input, the corresponding bit in the PRTxDR register must be set to enable the pull up resistor. Once the pull up resistor is enabled, the state of the pin is read using the PRTxDR register. This mode is also used as output.

Strong Slow. This mode is similar to the Strong mode, but the slope of the output is slightly controlled so that high harmonics are not present when the output switches.

Selecting the pins and drive properties as described in this section completes the pin configuration.

The fifth column shown in Figure 2 sets the interrupt type of the pins.

Code Level Configuration

Another method to configure IO pins is to directly modify certain registers in the code using assembly or C language. This method allows you to configure IO ports dynamically during program execution.

Note If the pin configuration is fixed, then code is not required to configure the pins. PSoC Designer automatically generates startup code to configure the pins according to the settings in the Device Editor.

There are three registers for each port that sets the drive mode of every port pin. They are called PRTxDM0, PRTxDM1, and PRTxDM2 registers. The bits in the three registers together configure a particular pin. For example, bit0 of PRT0DM0, PRT0DM1, and PRT0DM2 controls the P0[0] drive mode.

The following tables list the combination of these registers and the corresponding drive mode.

Table 1. Drive Mode Configuration

DM2	DM1	DM0	Drive Mode	Data = 0	Data = 1
0	0	0	Resistive Pull Down	Resistive	Strong
0	0	1	Strong Drive	Strong	Strong
0	1	0	High Impedance	High-Z	High-Z
0	1	1	Resistive Pull Up	Strong	Resistive
1	0	0	Open Drain, Drives High	High-Z	Strong (Slow)
1	0	1	Slow Strong Drive	Strong (Slow)	Strong (Slow)
1	1	0	High Impedance Analog	High-Z	High-Z
1	1	1	Open Drain, Drives Low	Strong (Slow)	High-Z

DM2: Drive Mode 1 (Change DM2 value by using PRTxDM2 Registers)

DM1: Drive Mode 1 (Change DM1 value by using PRTxDM1 Registers)

DM0: Drive Mode 0 (Change DM0 value by using PRTxDM0 Registers)

Table 2. Port m Drive Mode 0 Registers (PRTxDM0)

Control Pin #	Port_x_7	Port_x_6	Port_x_5	Port_x_4	Port_x_3	Port_x_2	Port_x_1	Port_x_0
Bit Name	DM0[7]	DM0[6]	DM0[5]	DM0[4]	DM0[3]	DM0[2]	DM0[1]	DM0[0]

Note Port Drive Mode 0 Registers are 8 bits wide. All bits are Read or Write.

Port 0 Drive Mode **0** Register (PRT0DM**0**, Address = Bank 1, 00h)
Port 1 Drive Mode **0** Register (PRT1DM**0**, Address = Bank 1, 04h)
Port 2 Drive Mode **0** Register (PRT2DM**0**, Address = Bank 1, 08h)
Port 3 Drive Mode **0** Register (PRT3DM**0**, Address = Bank 1, 0Ch)
Port 4 Drive Mode **0** Register (PRT4DM**0**, Address = Bank 1, 10h)
Port 5 Drive Mode **0** Register (PRT5DM**0**, Address = Bank 1, 14h)
Port 6 Drive Mode **0** Register (PRT5DM**0**, Address = Bank 1, 18h)
Port 7 Drive Mode **0** Register (PRT5DM**0**, Address = Bank 1, 1Ch)

Table 3. Port m Drive Mode 1 Registers (PRTxDM1)

Control Pin #	Port_x_7	Port_x_6	Port_x_5	Port_x_4	Port_x_3	Port_x_2	Port_x_1	Port_x_0
Bit Name	DM1[7]	DM1[6]	DM1[5]	DM1[4]	DM1[3]	DM1[2]	DM1[1]	DM1[0]

Note Port Drive Mode 1 Registers are 8-bits wide. All bits are Read or Write.

Port 0 Drive Mode **1** Register (PRT0DM**1**, Address = Bank 1, 01h)
Port 1 Drive Mode **1** Register (PRT1DM**1**, Address = Bank 1, 05h)
Port 2 Drive Mode **1** Register (PRT2DM**1**, Address = Bank 1, 09h)
Port 3 Drive Mode **1** Register (PRT3DM**1**, Address = Bank 1, 0Dh)
Port 4 Drive Mode **1** Register (PRT4DM**1**, Address = Bank 1, 11h)
Port 5 Drive Mode **1** Register (PRT5DM**1**, Address = Bank 1, 15h)
Port 6 Drive Mode **1** Register (PRT6DM**1**, Address = Bank 1, 19h)
Port 7 Drive Mode **1** Register (PRT7DM**1**, Address = Bank 1, 1Dh)

Table 4. Port m Drive Mode 2 Registers (PRTxDM2)

Control Pin #	Port_x_7	Port_x_6	Port_x_5	Port_x_4	Port_x_3	Port_x_2	Port_x_1	Port_x_0
Bit Name	DM2[7]	DM2[6]	DM2[5]	DM2[4]	DM2[3]	DM2[2]	DM2[1]	DM2[0]

Note Port Drive Mode 2 Registers are 8-bits wide. All bits are Read or Write.

Port 0 Drive Mode **2** Register (PRT0DM**2**, Address = Bank 0, 03h)
Port 1 Drive Mode **2** Register (PRT1DM**2**, Address = Bank 0, 07h)
Port 2 Drive Mode **2** Register (PRT2DM**2**, Address = Bank 0, 0Bh)
Port 3 Drive Mode **2** Register (PRT3DM**2**, Address = Bank 0, 0Fh)
Port 4 Drive Mode **2** Register (PRT4DM**2**, Address = Bank 0, 13h)
Port 5 Drive Mode **2** Register (PRT5DM**2**, Address = Bank 0, 17h)
Port 6 Drive Mode **2** Register (PRT6DM**2**, Address = Bank 0, 1Bh)
Port 7 Drive Mode **2** Register (PRT7DM**2**, Address = Bank 0, 1Fh)

Port x Drive Mode 0, Port x Drive Mode 1 and Port x Drive Mode 2 registers together control the IO configuration of pins (ports) as shown in Table 1 on page 3.

DM2[0] **DM1**[0] **DM0**[0] control pin 0 of port x

DM2[1] **DM1**[1] **DM0**[1] control pin 1 of port x

DM2[2] **DM1**[2] **DM0**[2] control pin 2 of port x

DM2[3] **DM1**[3] **DM0**[3] control pin 3 of port x

DM2[4] **DM1**[4] **DM0**[4] control pin 4 of port x

DM2[5] **DM1**[5] **DM0**[5] control pin 5 of port x

DM2[6] **DM1**[6] **DM0**[6] control pin 6 of port x

DM2[7] **DM1**[7] **DM0**[7] control pin 7 of port x

For example, to configure Port_2_5 then you must write to the DM2[5], DM1[5] and DM0[5] bits of PORT 2 DRIVE MODE registers (PRT2DM2, PRT2DM1, PRT2DM0).

For example, configure P2_0 to be a Strong drive. The following steps must be performed.

From Table 1 on page 3 find the DM2, DM1, DM0 values for **Strong**. (If you want a pin to be output, you may choose Strong. Do not choose High Z for digital outputs.)

DM2 = 0 **DM1** = 0 and **DM0** = 1 for **Strong** drive.

Use assembly or C instructions to set bit 5 of PRT2DM0 and clear Bit5 of PRT2DM1 and PRT2DM2 registers.

Assembly Example

```
M8C_SetBank1
or   reg[PRT2DM0], 0x20
and  reg[PRT2DM1], ~0x20
M8C_SetBank0
and  reg[PRT2DM2], ~0x20
```

In the previous assembly example, the first line is a call to the M8C_SetBank1 macro, which switches the register bank to '1'. This is done because PRT2DM0 and PRT2DM1 are in register bank 1. Then using the "or" instruction and using a mask of 0x20, bit 5 of PRT2DM0 register is set. Then using "and" instruction and a mask of inverse of 0x20, bit 5 of the PRT2DM1 register is cleared. Using M8C_SetBank0, it is switched back to register bank '0', and using the "and" instruction and a mask of inverse of 0x20, bit 5 or the PRT2DM2 register is cleared. The "or" and "and" instructions are read, modify or write instructions. The content of the register is first read, a "or" or "and" operation is done on the value and then the result is written back to the same register. With this method, particular bits are modified without affecting the others.

C Example

```
PRT2DM0 |= 0x20;
PRT2DM1 &= ~0x20
PRT2DM2 &= ~0x20;
}
```

In C, the code becomes much simpler. The switching of the banks is taken care of by the C compiler. The "bitwise AND" (&=) or the "bitwise OR" (|=) must be used with the corresponding masks on the registers

Reading and Writing to a Port

After configuring IO ports, the PORT DATA registers are used to write or read data. PORT DATA registers are 8 bits wide. The bytes show the value read from pins of a particular port or the content of these registers is written directly to the port.

Port 0 Data Register (PRT0DR, Address = Bank 0, 00h)

Port 1 Data Register (PRT1DR, Address = Bank 0, 04h)

Port 2 Data Register (PRT2DR, Address = Bank 0, 08h)

Port 3 Data Register (PRT3DR, Address = Bank 0, 0Ch)

Port 4 Data Register (PRT4DR, Address = Bank 0, 10h)

Port 5 Data Register (PRT5DR, Address = Bank 0, 14h)

Port 6 Data Register (PRT6DR, Address = Bank 0, 18h)

Port 7 Data Register (PRT7DR, Address = Bank 0, 1Ch)

To write to a particular port pin, use the corresponding mask and bitwise "AND" or "OR" operation. For example, to set and clear P3[4]:

Assembly Example

```
or reg[PRT3DR], 0x10 ; Set P3[4]
and reg[PRT3DR], ~0x10 ; Clear P3[4]
```

C Example

```
PRT3DR |= 0x10;
PRT3DR &= ~0x10;
```

To read from a port pin, read from the PRTxDR register and use the corresponding bit mask. For example, to check the status of P2[3]:

Assembly Example

```
mov A, reg[PRT2DR]
and A,0x08
jnz PinHigh
; Code to process Pin cleared state
PinHigh:
; Code to process Pin set state
```

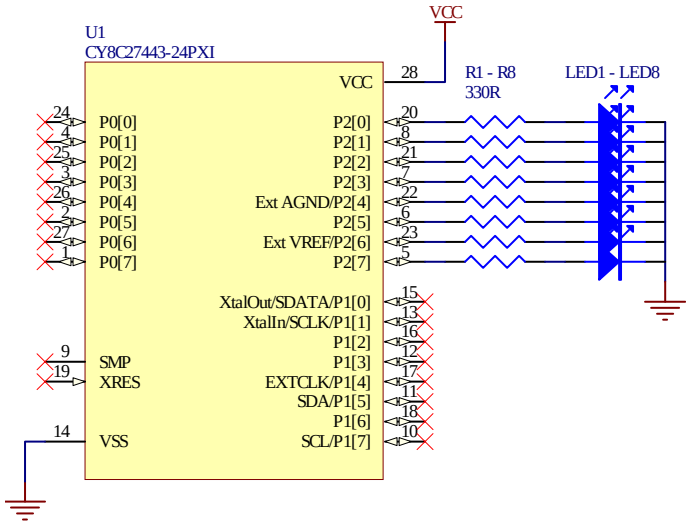
C Example

```
if (PRT2DR & 0x08)
{
   // Code to process Pin Set state
}
Else
{
   // Code to process Pin cleared state
```

Example Project

Example projects in assembly and C language have been created to demonstrate the reconfiguration of pins. Port 2 is reconfigured. Eight LEDs are connected to Port 2 and the effects of the pin drive modes are observed by checking the status of the LEDs. The schematic diagram of the test setup is shown in Figure 3.

Figure 3. Schematic Diagram of the Test Setup



Appendix

Assembly Example - Source Code

```
; Place a break point here
; Configure the pins of Port2 to Strong drive mode
; DM2=0 DM1=0 DM0=1
   M8C_SetBank1
   mov reg[PRT2DM0], FFh
   mov reg[PRT2DM1], 00h
   M8C_SetBank0
   mov reg[PRT2DM2], 00h

; Now we are writing FFh to the port 2
; Write FFh to Port2.
   mov reg[PRT2DR], FFh
   nop
; Now you see 8 LEDs of PSoC PUP Board are ON

; Let us turn Off P2[7], P2[6], P2[3] and P2[2]
; For this we perform an "AND" operation with the inverse of
; the bit mask CCh
   and reg[PRT2DR],~CCh
   nop
; Now you see 0011 0011 on the LEDs

; Let us switch Off all LEDs
   mov reg[PRT2DR], 00h

; Let's configure pins as input! (High Z with Digital Input Enabled)
; DM2=0 DM1=1 DM0=0
   M8C_SetBank1
   mov reg[PRT2DM0], 00h  ;Remember that DM0=0 DM1=1 for High Z.
   mov reg[PRT2DM1], FFh
   M8C_SetBank0
   mov reg[PRT2DM2], 00h
   nop
; Now all LEDs are switched Off! Because all pins are input now.
; You do not need to write data to the port again after a new configuration.

; Let's configure pins as pull down
; DM2=0 DM1=0 DM0=0
   M8C_SetBank1
   mov reg[PRT2DM0], 00h  ;Remember that DM0=0 DM1=0 for Pull Down.
   mov reg[PRT2DM1], 00h
   M8C_SetBank0
   mov reg[PRT2DM2], 00h
   ; Let's write FFh to the port data registers
   mov reg[PRT2DR], FFh
   nop
; Now you see all LEDs are ON. Same with STRONG Drive.

; Let's configure pins as pull up.

   M8C_SetBank1
   mov reg[PRT2DM0], FFh         ;Remember that DM0=1 DM1=1 for Pull Up.
   mov reg[PRT2DM1], FFh
   M8C_SetBank0
   mov reg[PRT2DM2], 00h
   nop
; Now you see all LEDs are ON but they are not bright.
; This is because of the internal 5.6K resistor. It limits the output current.
```

C Example - Source Code

```
//-------------------------------------------------------------------------
// C main line
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

void main()
{
// Place a break point here
// Configure the pins of Port2 to Strong drive mode
// DM2=0 DM1=0 DM0=1
   PRT2DM0 = 0xFF;
   PRT2DM1 = 0x00;
   PRT2DM2 = 0x00;

// Let's write 0xFF to Port2
   PRT2DR = 0xFF;
// Place breakpoint here
   asm("nop");
// Now you see 8 LEDs of PSoC PUP Board are ON

// Let us turn Off P2[7], P2[6], P2[3] and P2[2]
// For this we perform an "AND" operation with the inverse of
// the bit mask 0xCC
       PRT2DR &= ~0xCC;
// Place breakpoint here
   asm("nop");
// Now you see 0011 0011 on the LEDs

// Let us switch Off all LEDs
   PRT2DR = 0x00;

// Let's configure pins as input! (High Z with Digital Input Enabled)
// DM2=0 DM1=1 DM0=0
   PRT2DM0 = 0x00;
   PRT2DM1 = 0xFF;
   PRT2DM2 = 0x00;
// Place breakpoint here
   asm("nop");
// Now all LEDs are switched Off! Because all pins are input now.
// You do not need to write data to the port again after a new configuration.

// Let's configure pins as pull down
// DM2=0 DM1=0 DM0=0
   PRT2DM0 = 0x00;
   PRT2DM1 = 0x00;
   PRT2DM2 = 0x00;

// Let's write 0xFF to the port data registers
   PRT2DR = 0xFF;
// Place breakpoint here
   asm("nop");
// Now you see all LEDs are ON. Same with STRONG Drive.

// Let's configure pins as pull up.
   PRT2DM0 = 0xFF;
   PRT2DM1 = 0xFF;
   PRT2DM2 = 0x00;
// Place breakpoint here
   asm("nop");
// Now you see all LEDs are ON but they are not bright.
// This is because of the internal 5.6K resistor. It limits the output current.
}
```

About the Author

Name: Mehmet Zeki SONMEZ

Title: Yeditepe University: Electrical and Electronics Engineering Student, and Student Assistant (2006).

Background: M.Zeki Sonmez is interested in the design of both analog and digital (including microcontrollers) circuits. In the near future his area of interest will be RF Identification.

Web Site: http://www.sonmicro.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.

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