

## AN2033

**Author:** Darrin Vallis

**Associated Project:** Yes

**Associated Part Family:** CY8C25xxx, CY8C26xxx

[GET FREE SAMPLES HERE](#)

**Software Version:** NA

**Associated Application Notes:** None

### Application Note Abstract

This Application Note reveals programming tips to manipulate I/O port bits of the PSoC® microcontroller.

### Introduction

Cypress MicroSystems Programmable System on Chip (PSoC™) microcontroller provides highly flexible I/O with every pin having configurable drive modes, interrupts and global routing connections.

### Assembly vs. C

Efficient software is vital for embedded system design, especially with respect to I/O routines. PSoC MCU I/O ports are highly configurable, and directly accessible by firmware bit operations in PSoC Designer. This provides maximum flexibility for programmers. However, it can require code that is difficult to read.

Consider this common example: A hardware engineer wants to clear bit 6 of I/O port 0 (P0[6]). With a PSoC microcontroller, they would write the following assembly code:

```
export _main

_main:
    and REG[PRT0DR], 0xBF
    ret
```

One line is easy to read, but a complex project can become increasingly difficult to read by others over time. The C version of the code would look like:

```
#include "m8c.h"
void main()
{
    PRT0DR&=0xBF;
}
```

### Simple Macro to Flip Bits

A desirable feature would be to flip an I/O port bit with a simple statement like `P0.0 = 1`. Using C macro programming this is easily accomplished.

Consider the following C code:

```
#include "m8c.h"

#define Port0_0(b) (PRT0DR = (b==0) ? \
(PRT0DR&0xFE) : (PRT0DR|0x01))

void main()
{
    Port0_0(1); // Set Port 0, bit 0 to
    1
}
```

The C Compiler pre-processor in PSoC Designer replaces this `Port0_0(1)` call in `main()` with the macro definition before compiling. The assembly code generated is:

```
OR    REG[0],1
```

So, we get an efficient translation into assembly from a very readable C format. All that remains is generating a header file to implement the call for each bit of all device ports. (See *ports.h* in Appendix A.)

## Macro to Set or Clear

It is also useful to set or clear all bits of a given port. Here's an example:

```
#include "m8c.h"
#include "ports.h"

void main()
{
    Port0(1); //Set all bits of Port0
to 1
}
```

## Summary

To recap, the format to set bits with this new macro header is:

```
PortX_Y(b) : PortNumber=X[7..0],
             Bit=Y[7..0], b[1,0]
```

```
PortX(b)   : PortNumber=X[7..0], b[1,0]
```

Based on package size, not all parts support all 8 ports. Voila! You now have a slick method of flipping bits in a PSoC microcontroller C application.

## Appendix. ports.h

```

#define Port0(b)    (PRT0DR = (b==0) ? 0x00 : 0xFF)
#define Port0_0(b) (PRT0DR = (b==0) ? (PRT0DR&0xFE) : (PRT0DR|0x01))
#define Port0_1(b) (PRT0DR = (b==0) ? (PRT0DR&0xFD) : (PRT0DR|0x02))
#define Port0_2(b) (PRT0DR = (b==0) ? (PRT0DR&0xFB) : (PRT0DR|0x04))
#define Port0_3(b) (PRT0DR = (b==0) ? (PRT0DR&0xF7) : (PRT0DR|0x08))
#define Port0_4(b) (PRT0DR = (b==0) ? (PRT0DR&0xEF) : (PRT0DR|0x10))
#define Port0_5(b) (PRT0DR = (b==0) ? (PRT0DR&0xDF) : (PRT0DR|0x20))
#define Port0_6(b) (PRT0DR = (b==0) ? (PRT0DR&0xBF) : (PRT0DR|0x40))
#define Port0_7(b) (PRT0DR = (b==0) ? (PRT0DR&0x7F) : (PRT0DR|0x80))

#define Port1(b)    (PRT1DR = (b==0) ? 0x00 : 0xFF)
#define Port1_0(b) (PRT1DR = (b==0) ? (PRT1DR&0xFE) : (PRT1DR|0x01))
#define Port1_1(b) (PRT1DR = (b==0) ? (PRT1DR&0xFD) : (PRT1DR|0x02))
#define Port1_2(b) (PRT1DR = (b==0) ? (PRT1DR&0xFB) : (PRT1DR|0x04))
#define Port1_3(b) (PRT1DR = (b==0) ? (PRT1DR&0xF7) : (PRT1DR|0x08))
#define Port1_4(b) (PRT1DR = (b==0) ? (PRT1DR&0xEF) : (PRT1DR|0x10))
#define Port1_5(b) (PRT1DR = (b==0) ? (PRT1DR&0xDF) : (PRT1DR|0x20))
#define Port1_6(b) (PRT1DR = (b==0) ? (PRT1DR&0xBF) : (PRT1DR|0x40))
#define Port1_7(b) (PRT1DR = (b==0) ? (PRT1DR&0x7F) : (PRT1DR|0x80))

#define Port2(b)    (PRT2DR = (b==0) ? 0x00 : 0xFF)
#define Port2_0(b) (PRT2DR = (b==0) ? (PRT2DR&0xFE) : (PRT2DR|0x01))
#define Port2_1(b) (PRT2DR = (b==0) ? (PRT2DR&0xFD) : (PRT2DR|0x02))
#define Port2_2(b) (PRT2DR = (b==0) ? (PRT2DR&0xFB) : (PRT2DR|0x04))
#define Port2_3(b) (PRT2DR = (b==0) ? (PRT2DR&0xF7) : (PRT2DR|0x08))
#define Port2_4(b) (PRT2DR = (b==0) ? (PRT2DR&0xEF) : (PRT2DR|0x10))
#define Port2_5(b) (PRT2DR = (b==0) ? (PRT2DR&0xDF) : (PRT2DR|0x20))
#define Port2_6(b) (PRT2DR = (b==0) ? (PRT2DR&0xBF) : (PRT2DR|0x40))
#define Port2_7(b) (PRT2DR = (b==0) ? (PRT2DR&0x7F) : (PRT2DR|0x80))

#define Port3(b)    (PRT3DR = (b==0) ? 0x00 : 0xFF)
#define Port3_0(b) (PRT3DR = (b==0) ? (PRT3DR&0xFE) : (PRT3DR|0x01))
#define Port3_1(b) (PRT3DR = (b==0) ? (PRT3DR&0xFD) : (PRT3DR|0x02))
#define Port3_2(b) (PRT3DR = (b==0) ? (PRT3DR&0xFB) : (PRT3DR|0x04))
#define Port3_3(b) (PRT3DR = (b==0) ? (PRT3DR&0xF7) : (PRT3DR|0x08))
#define Port3_4(b) (PRT3DR = (b==0) ? (PRT3DR&0xEF) : (PRT3DR|0x10))
#define Port3_5(b) (PRT3DR = (b==0) ? (PRT3DR&0xDF) : (PRT3DR|0x20))
#define Port3_6(b) (PRT3DR = (b==0) ? (PRT3DR&0xBF) : (PRT3DR|0x40))
#define Port3_7(b) (PRT3DR = (b==0) ? (PRT3DR&0x7F) : (PRT3DR|0x80))

#define Port4(b)    (PRT4DR = (b==0) ? 0x00 : 0xFF)
#define Port4_0(b) (PRT4DR = (b==0) ? (PRT4DR&0xFE) : (PRT4DR|0x01))
#define Port4_1(b) (PRT4DR = (b==0) ? (PRT4DR&0xFD) : (PRT4DR|0x02))
#define Port4_2(b) (PRT4DR = (b==0) ? (PRT4DR&0xFB) : (PRT4DR|0x04))
#define Port4_3(b) (PRT4DR = (b==0) ? (PRT4DR&0xF7) : (PRT4DR|0x08))
#define Port4_4(b) (PRT4DR = (b==0) ? (PRT4DR&0xEF) : (PRT4DR|0x10))
#define Port4_5(b) (PRT4DR = (b==0) ? (PRT4DR&0xDF) : (PRT4DR|0x20))
#define Port4_6(b) (PRT4DR = (b==0) ? (PRT4DR&0xBF) : (PRT4DR|0x40))
#define Port4_7(b) (PRT4DR = (b==0) ? (PRT4DR&0x7F) : (PRT4DR|0x80))

#define Port5(b)    (PRT5DR = (b==0) ? 0x00 : 0xFF)
#define Port5_0(b) (PRT5DR = (b==0) ? (PRT5DR&0xFE) : (PRT5DR|0x01))
#define Port5_1(b) (PRT5DR = (b==0) ? (PRT5DR&0xFD) : (PRT5DR|0x02))
#define Port5_2(b) (PRT5DR = (b==0) ? (PRT5DR&0xFB) : (PRT5DR|0x04))
#define Port5_3(b) (PRT5DR = (b==0) ? (PRT5DR&0xF7) : (PRT5DR|0x08))
#define Port5_4(b) (PRT5DR = (b==0) ? (PRT5DR&0xEF) : (PRT5DR|0x10))
#define Port5_5(b) (PRT5DR = (b==0) ? (PRT5DR&0xDF) : (PRT5DR|0x20))
#define Port5_6(b) (PRT5DR = (b==0) ? (PRT5DR&0xBF) : (PRT5DR|0x40))
#define Port5_7(b) (PRT5DR = (b==0) ? (PRT5DR&0x7F) : (PRT5DR|0x80))

```

```

#define Port6(b) (PRT6DR = (b==0) ? 0x00 : 0xFF)
#define Port6_0(b) (PRT6DR = (b==0) ? (PRT6DR&0xFE) : (PRT6DR|0x01))
#define Port6_1(b) (PRT6DR = (b==0) ? (PRT6DR&0xFD) : (PRT6DR|0x02))
#define Port6_2(b) (PRT6DR = (b==0) ? (PRT6DR&0xFB) : (PRT6DR|0x04))
#define Port6_3(b) (PRT6DR = (b==0) ? (PRT6DR&0xF7) : (PRT6DR|0x08))
#define Port6_4(b) (PRT6DR = (b==0) ? (PRT6DR&0xEF) : (PRT6DR|0x10))
#define Port6_5(b) (PRT6DR = (b==0) ? (PRT6DR&0xDF) : (PRT6DR|0x20))
#define Port6_6(b) (PRT6DR = (b==0) ? (PRT6DR&0xBF) : (PRT6DR|0x40))
#define Port6_7(b) (PRT6DR = (b==0) ? (PRT6DR&0x7F) : (PRT6DR|0x80))

#define Port7(b) (PRT7DR = (b==0) ? 0x00 : 0xFF)
#define Port7_0(b) (PRT7DR = (b==0) ? (PRT7DR&0xFE) : (PRT7DR|0x01))
#define Port7_1(b) (PRT7DR = (b==0) ? (PRT7DR&0xFD) : (PRT7DR|0x02))
#define Port7_2(b) (PRT7DR = (b==0) ? (PRT7DR&0xFB) : (PRT7DR|0x04))
#define Port7_3(b) (PRT7DR = (b==0) ? (PRT7DR&0xF7) : (PRT7DR|0x08))
#define Port7_4(b) (PRT7DR = (b==0) ? (PRT7DR&0xEF) : (PRT7DR|0x10))
#define Port7_5(b) (PRT7DR = (b==0) ? (PRT7DR&0xDF) : (PRT7DR|0x20))
#define Port7_6(b) (PRT7DR = (b==0) ? (PRT7DR&0xBF) : (PRT7DR|0x40))
#define Port7_7(b) (PRT7DR = (b==0) ? (PRT7DR&0x7F) : (PRT7DR|0x80))

```

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, 2002-2007. 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.