

PSoC™ Designer:
Integrated Development Environment
User Guide

User Guide Revision 1.19 (Cypress Revision *C)
PSoC Designer 4.1
Spec.# 38-12002
Last Revised: January 21, 2004
Cypress MicroSystems, Inc.



CYPRESS MICROSYSTEMS

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

http://www.cypress.com/ http://www.cypress.com/aboutus/sales_locations.cfm
http://www.cypress.com/support/mysupport.cfm

Copyright © 2002-2004 Cypress MicroSystems, Inc. All rights reserved.
PSoC™ (Programmable System-on-Chip) is a trademark of Cypress MicroSystems, Inc.

Athlon is a trademark of Advanced Micro Devices, Inc.
Acrobat and Reader are registered trademarks of Adobe Systems Incorporated.
Copyright © 1999-2002 iMAGEcraft Creations Inc. All rights reserved.
InstallShield® is a registered trademark/service mark of InstallShield Software Corporation in USA and other countries.
All Intel products referenced herein are either trademarks or registered trademarks of Intel Corporation.
All Microsoft products referenced herein are either trademarks or registered trademarks of Microsoft Corp.
All O&O products referenced herein are either trademarks or registered trademarks of O&O Software.
All Quatech products referenced herein are either trademarks or registered trademarks of Quatech, Inc.
All RAXCO products referenced herein are either trademarks or registered trademarks of Raxco Software.
All SIIG products referenced herein are either trademarks or registered trademarks of SIIG, Inc.
All Symantec products referenced herein are either trademarks or registered trademarks of Symantec Corporation.

The information contained herein is subject to change without notice.

Table of Contents

List of Tables v
List of Figures vii
What's New With PSoC Designer 4.1 ix
Two-Minute Overview 1
Documentation Conventions 2
Notation Standards 3
Section 1. Introduction 5
1.1 Purpose 5
1.2 Support 5
1.3 Section Overview 6
1.4 Product Upgrades 6
Section 2. Installation 7
2.1 Hardware Requirement Checklist 7
2.2 Software Requirement Checklist 8
2.3 Performance Factors 8
2.4 Installing the System 9
2.5 Project Update 17
Section 3. Using the IDE 21
3.1 System Diagram 21
3.2 File Types and Extensions 21
3.3 Project Manager 25
3.4 Edit Windows 31
3.5 Output Status Window 32
3.6 Project Settings 33
3.7 Tools Options 44
Section 4. Creating a Project 49
4.1 Create a Project 49
4.2 Project Methods 53
4.3 Project Backup Folder 57
Section 5. Device Editor 59

5.1 Navigating Device Editor 59
5.2 Selecting User Modules 64
5.3 Placing User Modules 70
5.4 Deploying Interconnectivity (Connecting User Modules) 79
5.5 Specifying Pin-out 89
5.6 Tracking Device Space 98
5.7 Design Rule Checker 99
5.8 Generating Application Files 100

Section 6. Dynamic Re-configuration 109

6.1 Add Configuration 109
6.2 Export and Import Designs 111
6.3 Delete Configuration 115
6.4 Global Parameters and Dynamic Re-configuration 116
6.5 Pin Settings and Dynamic Re-configuration 116
6.6 Code Generation and Dynamic Re-configuration 118
6.7 Application Editor and Dynamic Re-configuration 121
6.8 Debugger and Dynamic Re-configuration 121

Section 7. Application Editor 125

7.1 File Definitions and Recommendations 125
7.2 Additional Generated Files 127
7.3 Modifying Files 129
7.4 Adding Files 130
7.5 Removing Files 131
7.6 Full Application File Search 132

Section 8. Assembler 135

8.1 Accessing the Assembler 135
8.2 The Microprocessor 135
8.3 Assembly File Syntax 137
8.4 List File Format 138
8.5 Assembler Directives 138
8.6 Instruction Set 139
8.7 Compiling/Assembling Files 140

Section 9. Builder 141

9.1 Building a Project 141
9.2 C Compiler 142
9.3 Linker/Loader 143
9.4 Librarian 144

Section 10. Debugger 145

10.1 Debugger Components 145
10.2 Connecting to the ICE 147
10.3 Downloading to Pod 151
10.4 Debug Strategies 152
10.5 Menu Options 166
10.6 Programming the Part 167

Section 11. Flash Program Memory Protection 170

11.1 FPMP and PSoC Designer 170
11.2 flashsecurity.txt and Application Editor 171
11.3 FPMP File Errors 173
11.4 Example 173

Appendix A. Troubleshooting 175
Appendix B. Glossary 177
Index 179

List of Tables

Table 1: Documentation Conventions 2
Table 2: Internal Registers 3
Table 3: Assembler Directives 3
Table 4: File Types and Extensions 22
Table 5: Default Memory Organization 39
Table 6: Navigating Device Interface 63
Table 7: Source Files Generated by Generate Application 101
Table 8: boot.asm Interrupt Names 105
Table 9: Menu Options for Modifying Source Files 129
Table 10: Destination of AND Instruction 137
Table 11: Keyword Types 137
Table 12: Assembler Directives 138
Table 13: Instruction Set Notation 139
Table 14: Debugging Menu Options 166
Table 15: Header to Device Pin Connections 168
Table 16: Flash Program Memory Protection Options 170
Table 17: Parallel Port Options 176
Table 18: Terminology 177

List of Figures

Figure 1: Welcome Screen 9
Figure 2: PSoC Designer Setup Wizard 10
Figure 3: License Agreement 11
Figure 4: Choose Destination Location 12
Figure 5: Select Program Folder 13
Figure 6: Start Copying Files 14
Figure 7: Setup Status 15
Figure 8: PSoC Designer Installation Wizard Completed 16
Figure 9: Database Version Detection 17
Figure 10: Outdated User Modules 18
Figure 11: PSoC Designer Subsystems 21
Figure 12: Source Tree 24
Figure 13: Device Editor Subsystem 26
Figure 14: Application Editor Subsystem 28
Figure 15: Debugger Subsystem 30
Figure 16: Cascaded Windows 31
Figure 17: Window Options 31
Figure 18: Output Status Window 32
Figure 19: Project Settings Compiler Tab 33
Figure 20: Project Settings Device Editor Tab 37
Figure 21: Project Settings Linker Tab 38
Figure 22: Project Settings Programmer Tab 42
Figure 23: Project Settings Debugger Tab 43
Figure 24: Options Dialog Box 44
Figure 25: Start Dialog Box 49
Figure 26: New Project Dialog Box 50
Figure 27: New Configuration Dialog Box 51
Figure 28: Parts Catalog Dialog Box 52
Figure 29: Existing Configuration Dialog Box 54
Figure 30: Design-Based Configuration Dialog Box 56
Figure 31: Design Import Status 56
Figure 32: Zoom/Pan Pop-up Window 60
Figure 33: Interconnect View 61
Figure 34: Device Interface View 62
Figure 35: Device Interface Pan/Zoom Toolbar 62
Figure 36: User Module Selection View in Toolbar 64
Figure 37: User Module Options 65
Figure 38: User Module Data 66

Figure 39: User Module Toolbar 67
Figure 40: User Module Selections 69
Figure 41: Interconnect View in Toolbar 70
Figure 42: Selected (yet-to-be Placed) User Modules 71
Figure 43: Placed User Modules 74
Figure 44: Filter User Module Options 76
Figure 45: User Module Parameters 78
Figure 46: Global Resources 79
Figure 47: Digital Interconnect Row Input 85
Figure 48: Synchronization Options for Digital Interconnect Row Inputs 86
Figure 49: Digital Interconnect Row Output 87
Figure 50: Logical Operations in Digital Interconnect Row Output 88
Figure 51: Digital Interconnect Row Global Output 89
Figure 52: Interconnect View in Device Editor Toolbar 90
Figure 53: Device Pin-Out 91
Figure 54: PSoC Block Resource Meter 99
Figure 55: Output Status During Application Generation 101
Figure 56: PWM16_1.h 102
Figure 57: Timer32 on Four Digital PSoC Blocks 104
Figure 58: Dynamic Re-configuration Toolbar 109
Figure 59: Export Design Dialog Box 112
Figure 60: Design Browser 114
Figure 61: Add New File 131
Figure 62: Find in Files Dialog Box 132
Figure 63: Output Status Window 142
Figure 64: Basic Development Kit Components 146
Figure 65: ICE Front Panel 147
Figure 66: Pod, Pod Foot, and Demonstration Board 149
Figure 67: Trace 153
Figure 68: Trace Save As 154
Figure 69: Debug Breakpoints 155
Figure 70: Debug ASM Watch Properties 156
Figure 71: Watch/Global Name Window 157
Figure 72: Array Type Variables 159
Figure 73: Stack Overflow Event 164
Figure 74: Debugger Toolbar 166
Figure 75: 28-Pin DIP Programming Board 167
Figure 76: 28-Pin SOIC (Small Outline IC) Programming Board 167
Figure 77: flashsecurity.txt in Source Tree 171
Figure 78: Snip-it of flashsecurity.txt 172
Figure 79: FPMP Error in Output Status Window 173
Figure 80: Unprotected Flash at 3C80h 174

What's New With PSoC Designer 4.1

- USB Dongle support for PSoC Designer v. 4.1 running on Windows XP, 2000 or Me. The existing ICE may be connected to a USB port through the USB Dongle. Drivers are automatically installed during installation of PSoC Designer v. 4.1. An ICE used with the USB Dongle has the same functionality as connecting via paralell port. Paralell port ICE connection is still supported.
- Support has been added for CY8C24xxx, CY3208-POD downloading and debugging.
- The User Module Project Update dialog box has been removed. User code "markers" are provided in User Module *int.asm* files for automatic updates during code generation/build. The *int.asm* files are always regenerated during code generation.
- The ICE connection setting has moved to the Project Settings dialog box and is now project specific. A text area is provided for friendly names of the connection port. USB dongle ports can be selected from the drop-down as well.
- System performance is faster when you access and close PSoC Designer, open a project, generate source code, and load Device Editor Interconnection View.
- Progress-indicator bars and/or text have been added when you access PSoC Designer, open a project and generate source code.
- New connection options have been added under User Module Parameters in the device interface and existing connection issues have been fixed.
- You can now vertically resize the parameter grids in the Interconnection View of Device Editor.

Two-Minute Overview

This two-minute overview of *PSoC™ Designer: Integrated Development Environment User Guide* was purposefully placed up front for you advanced engineers who are ready to configure and program the chip but need a *quick* point in the right direction. **Make sure you have the latest version of the software**. (Now we only have a minute and-a-half left.)

Overview	35 seconds	You have the device, PSoC Designer, and the vision… This guide provides: ▪ installation procedures, ▪ interface overview, ▪ instructions for creating a project, ▪ instructions for configuring the device, ▪ instructions for dynamically re-configuring the project, ▪ instructions for editing assembly-source files, ▪ instructions for compiling files, ▪ instructions for building the project, ▪ instructions for debugging the project, ▪ instructions for Flash memory protection.
Basics	30 seconds	PSoC Designer contains three subsystems: Device Editor, Application Editor, and Debugger. Start by creating a project: 1. Create a project.  2. Configure device in Device Editor.  3. Edit source files in Application Editor.  4. Debug project in Debugger. 
Quick Reference	15 seconds	Click a hyperlink to reference key material: Section 2. Section 3. Section 4. Appendix A.
Bottom Line	10 seconds	Programmable System-on-Chip PSoC Designer empowers you to customize the functionality you desire into the device.

 Time's up…

Documentation Conventions

Following, are easily identifiable conventions used throughout the PSoC Designer suite of product documentation.

Table 1: Documentation Conventions

Convention	Usage

`Courier Size 12`	Displays source code:
	`>cmasm testfile -t 4`
	`CMASM Version 2.20`
	`For C series Microcontrollers`
	`I 2000 Cypress MicroSystems Inc.`
	`Complete!`
	`>`

`Courier Size 12`	Displays file locations: `C:\ …cd\icc\`
Italics	Displays file names: *sourcefile.rom*
[**bracketed, bold**]	Displays keyboard commands: [**Enter**] or [**Ctrl**] [**C**]
File >> Open	Displays menu paths: Edit >> Cut

Notation Standards

Following, are notation standards used throughout the PSoC Designer suite of product documentation.

Table 2: Internal Registers

Notation	Description
A	Primary Accumulator
CF	Carry Flag
expr	Expression
F	Flags (ZF, CF, and Others)
k	Operand 1 Value
k_1 k_2	First Operand of 2 Operands Second Operand of 2 Operands
K	Operand 2 Value
PC	PCH, PCL
SP	Stack Pointer
X	X Register
ZF	Zero Flag

Table 3: Assembler Directives

Symbol	Assembler Directive
AREA	Area
ASCIZ	NULL Terminated ASCII String
BLK	RAM Byte Block
BLKW	RAM Word Block
DB	Define Byte
DS	Define ASCII String
DSU	Define UNICODE String
DW	Define Word
DWL	Define Word with Little Endian Ordering
ELSE	Alternative Result of IF…ELSE…ENDIF

Table 3: Assembler Directives, continued

ENDIF	End of IF…ELSE…ENDIF
EQU	Equate Label to Valuable Value
EXPORT	Export
IF	Conditional Assembly
INCLUDE	Include Source File
.LITERAL, .ENDLITERAL	Prevent Code Compression of Data
MACRO/ENDM	Macro Definition Start/End
ORG	Area Origin
.SECTION, .ENDSECTION	Section for Dead-Code Elimination
Suspend - OR F,0 Resume - ADD SP,0	Suspend and Resume Code Compressor

Section 1. Introduction

1.1 Purpose

The *PSoC Designer: Integrated Development Environment User Guide* will guide you from start to finish on utilizing PSoC Designer to configure, program, compile, build, emulate, and debug your customized system that runs from the PSoC Device.

For details on compiling and assembling, see:

- *PSoC Designer: C Language Compiler User Guide*
- *PSoC Designer: Assembly Language User Guide*
- *PSoC Designer: ICE Connection Troubleshooting Guide*
- *PSoC Designer: ICE USB Dongle Installation Guide*
- *In-System Serial Programming (ISSP) CY3207ISSP User Guide*
- CY8C25122, CY8C26233, CY8C26443, CY8C26643 Device Family Data Sheet for Silicon Revision D
- CY8C27143, CY8C27243, CY8C27443, CY8C27543, CY8C27643 PSoC Mixed-Signal Array Data Sheet
- CY8C24123, CY8C24223, CY8C24423 PSoC Mixed-Signal Array Data Sheet
- CY8C22113, CY8C22213 PSoC Mixed-Signal Array Data Sheet

Together, these documents comprise the PSoC Designer documentation suite.

1.2 Support

Free support for PSoC Designer is available online at http://www.cypress.com. Resources include Training Seminars, Discussion Forums, Application Notes, PSoC Consultants, TightLink Technical Support Email/Knowledge Base, and Application Support Technicians.

1.3 Section Overview

Section 1.	Describes the purpose of this guide, overviews each section, and gives product upgrade and support information.
Section 2.	Lists system hardware and software requirements, runs through the installation procedure, and explains how to update existing projects.
Section 3.	Discusses the functional format of the system interface.
Section 4.	Describes how to create a project.
Section 5.	Details how to select and place User Modules, implement interconnectivity, specify pin-out, track device space, and generate application files.
Section 6.	Details how to add multiple configurations to a single PSoC Project.
Section 7.	Describes all source-editing options.
Section 8.	Details assembly-language source and compiling/assembling project files.
Section 9.	Describes how to build a project and details transparent linker/loader and librarian functionality.
Section 10.	Describes connecting to the In-Circuit Emulator (ICE) and debugging and programming the project.
Section 11.	Describes the feature of modifying the default security settings for Flash memory.

1.4 Product Upgrades

Cypress MicroSystems provides scheduled upgrades and version enhancements for PSoC Designer *free of charge*. You can order the upgrades from your distributor on CD-ROM or, better yet, download them directly from the Cypress MicroSystems PSoC web site under PSoC >> Software & Drivers at http://www.cypress.com/.

Also provided at the web site are critical updates to system documentation. To stay current with system functionality you can find documentation updates under PSoC >> More Resources, again at http://www.cypress.com/.

Section 2. Installation

In this section you will learn recommended hardware and software requirements and how to optimally run PSoC Designer as well as how to install the system and update existing projects to compatibility.

2.1 Hardware Requirement Checklist

The following hardware specifications are required to run PSoC Designer:

System Requirements	Minimum	Recommended
Processor Speed	750 MHz	1 GHz
MB of RAM	256 MB	512 MB
MB of Free Hard Drive Space	150 MB	200 MB
SVGA Monitor Graphics (High Color 16-Bit)	1024x768	1280x1024
CD-ROM Drive	x	x
Available EPP Parallel Port for In-Circuit Emulator	x	x
Rev. AB or Later - Emulation Pod for CY8C27xxx	x	x
Rev. G or Later - Emulation Pod for CY8C25/26xxx	x	x
ICE, CAT5 Patch Cable, Parallel Port Cable, and Power Adapter **Supplied in Kit**	x	x
PSoC Pup Board, Device Programmer, Replacement Feet, and Samples **Supplied in Development Kit** with Universal Pod (with Mask) Purchase	x	x

2.2 Software Requirement Checklist

The following software is required to run PSoC Designer:

Windows® 98, NT 4.x, 2000, Me, or XP

Microsoft Internet Explorer 6.x with MSXML Parser v. 3.0 or Higher

Adobe® Acrobat® Reader

Adobe® SVG Viewer 3.0

2.3 Performance Factors

PSoC Designer is a data-driven application with heavy use of data-intensive calculations and file I/O operations. To enhance performance for certain user tasks such as code generation, building/linking, and other key functions, you can implement one, some or all of the following options:

- Do not open projects across a network. Run them from your resident machine.
- Close down other applications and/or background tasks.
- Although PSoC Designer runs on a minimal configuration, faster processors have the greatest impact on performance. Increasing RAM and CPU speed provides some improvement.
- Because Windows Operating System hard-drive files become fragmented over time, reading operations can be greatly enhanced by defragmenting the hard drive. This can be done with Window's built-in defragmenter (under Start >> Programs >> Accessories >> System Tools >> Disk Defragmenter).

  For even better results, use Norton (Utilities™) Speed Disk, RAXCO PerfectDisk™, or O&O® Defrag.

2.4 Installing the System

Installing PSoC Designer on Windows NT/2000/XP requires user to have local Administrator permission.

To install PSoC Designer, do the following (estimated time is 2-4 minutes):

1. Place Cypress MicroSystems PSoC Designer CD-ROM in drive.

2. At the Welcome screen, single-click **Next**.

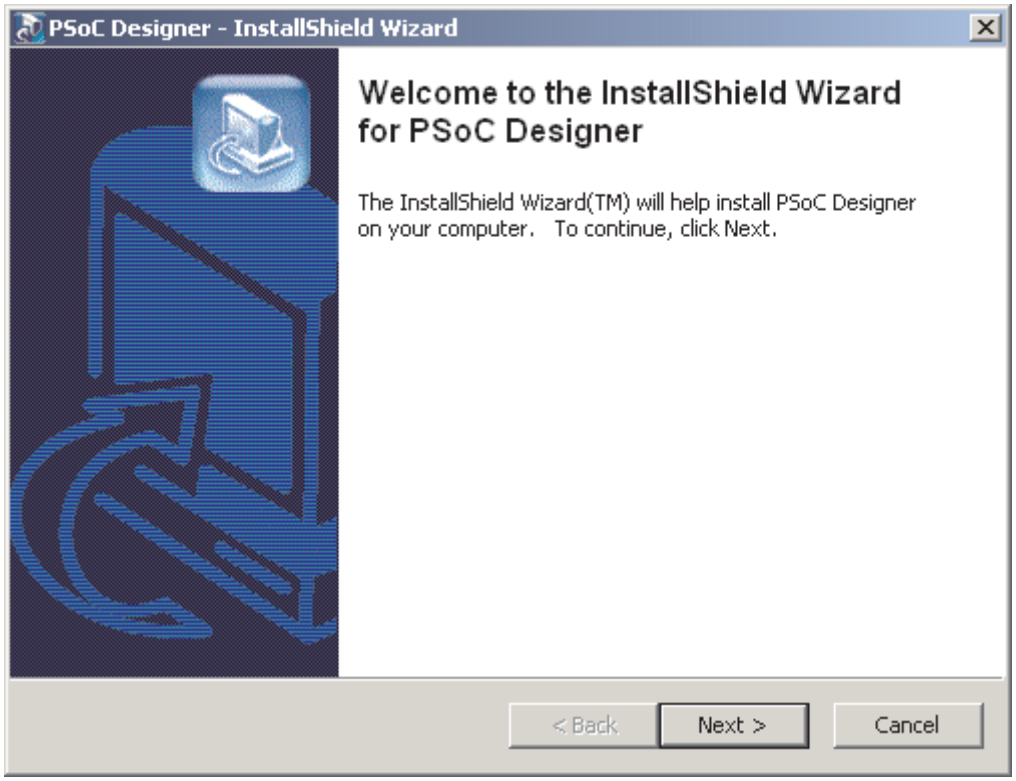


Figure 1: Welcome Screen

(If the Welcome screen does not automatically appear, click Start >> Run and Browse your CD drive for *PSoC Designer.exe*. Once located, click **OK**. The Welcome screen will appear.) The system will extract files and then run InstallShield® Wizard.

3. At the PSoC Designer Setup Wizard, click **Next**.

Click **Back** at any time during installation if you need to view or modify the previous screen. Click **Cancel** at any time to halt installation.

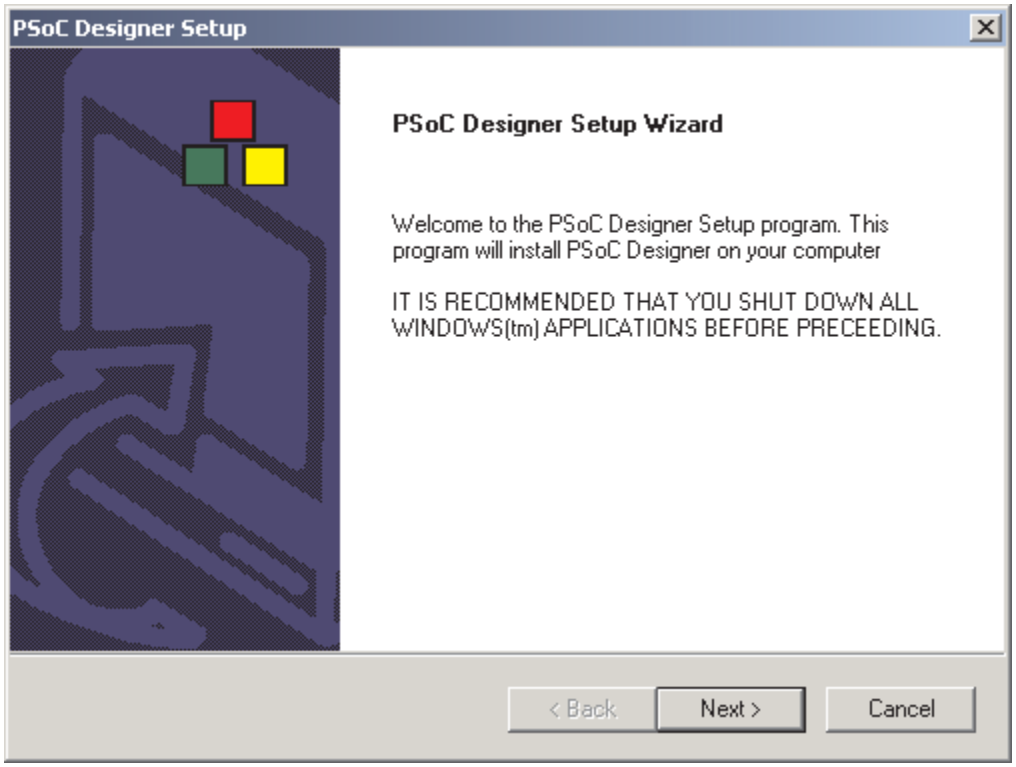


Figure 2: PSoC Designer Setup Wizard

4. At the License Agreement screen, scroll or use [**Page Down**] to view the terms of the agreement. When satisfied, single-click **Yes**.

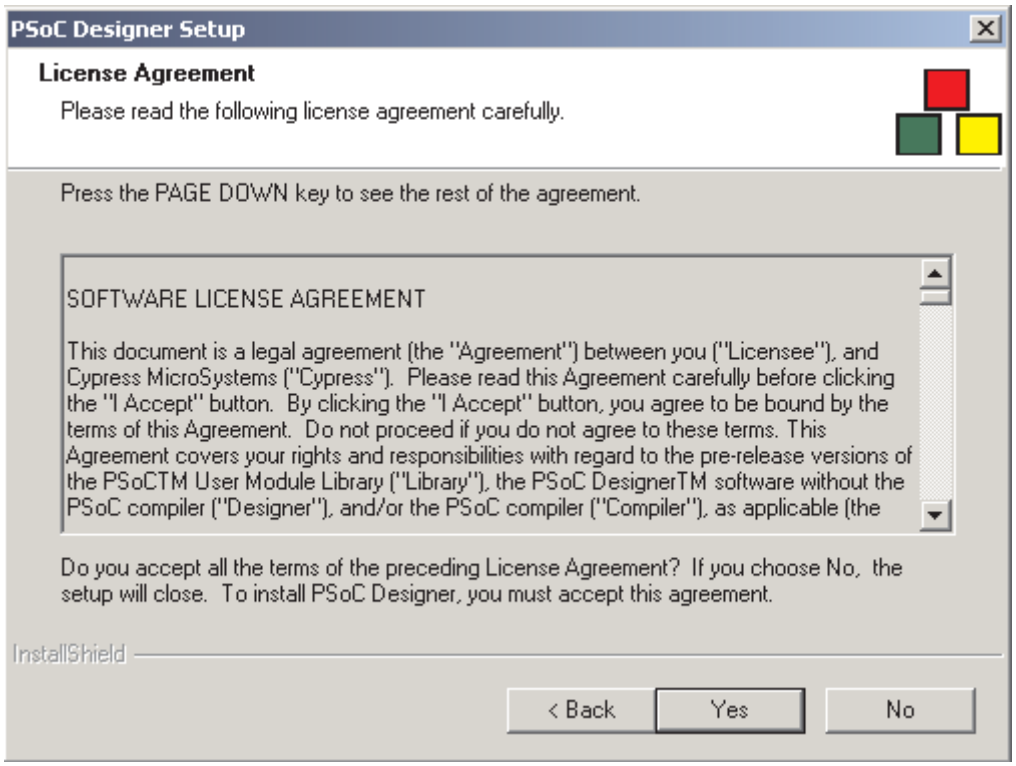


Figure 3: License Agreement

5. At the Choose Destination Location screen, click **Next** to install the system to the default directory path of `C:\Cypress MicroSystems\PSoC Designer`. If you wish to choose an alternative location, click **Browse** and select a different directory path.

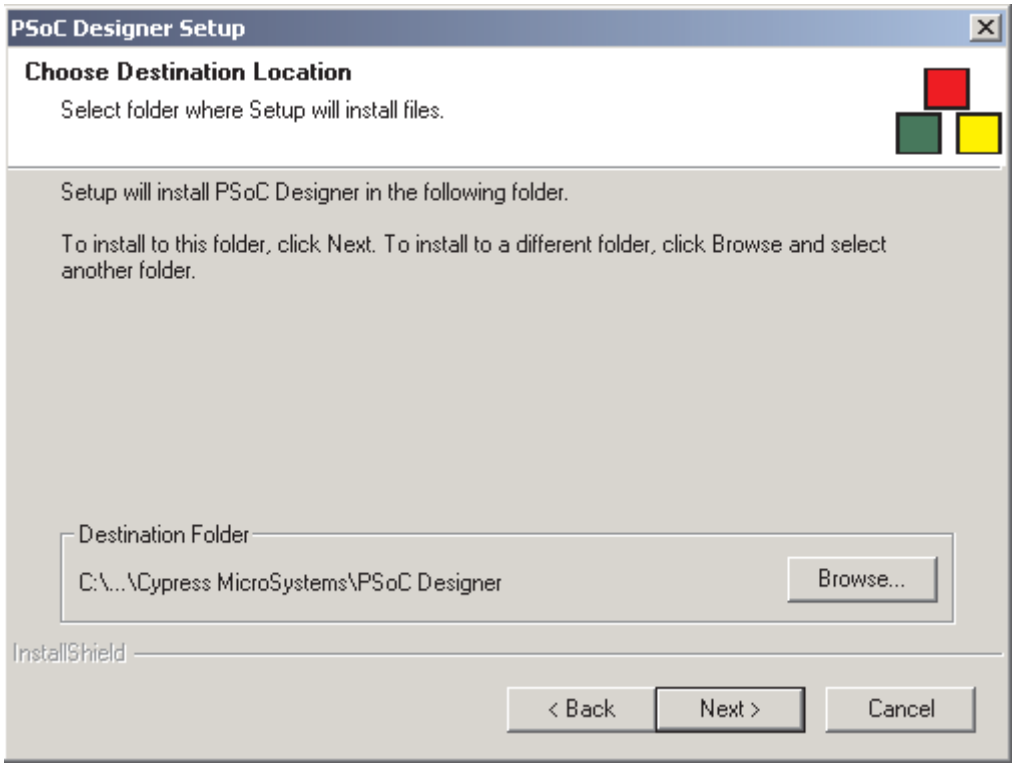


Figure 4: Choose Destination Location

6. At the Select Program Folder screen, click **Next** to accept the default folder of Cypress MicroSystems in which to add system icons. If you prefer, you can type a new folder name in the Program Folders field or select an existing folder from the Existing Folders field.

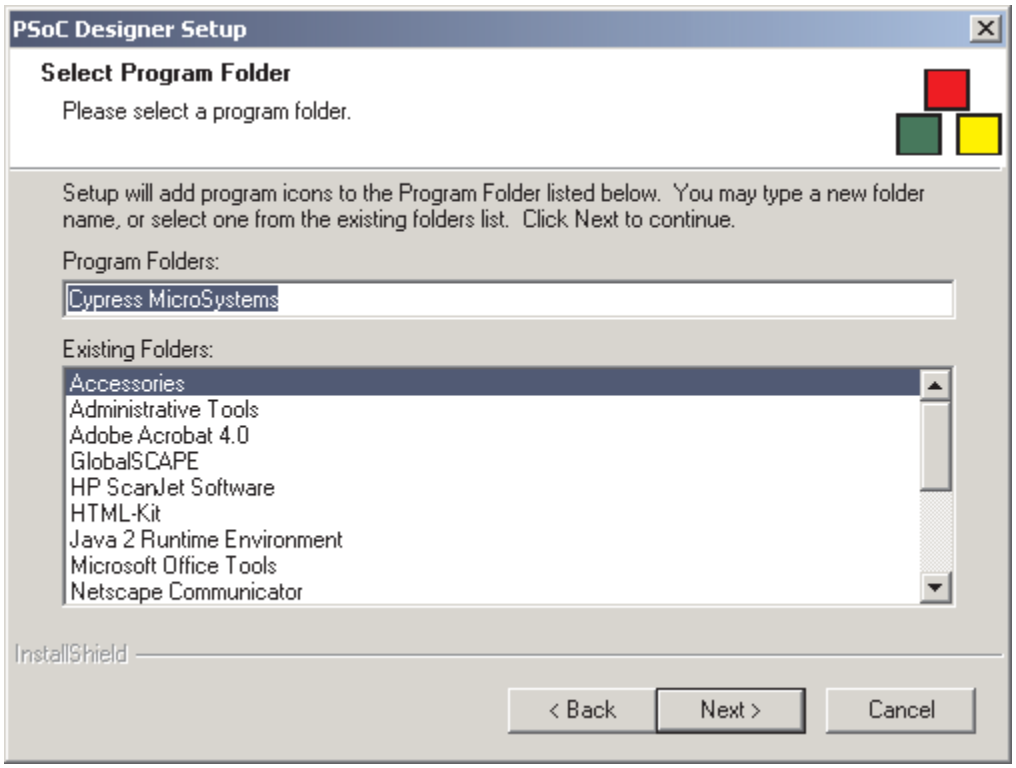


Figure 5: Select Program Folder

7. At the Start Copying Files screen, scroll to review your current settings. If you are satisfied, click **Next**. If not, click **Back** to return to view and/or modify settings in a previous screen.

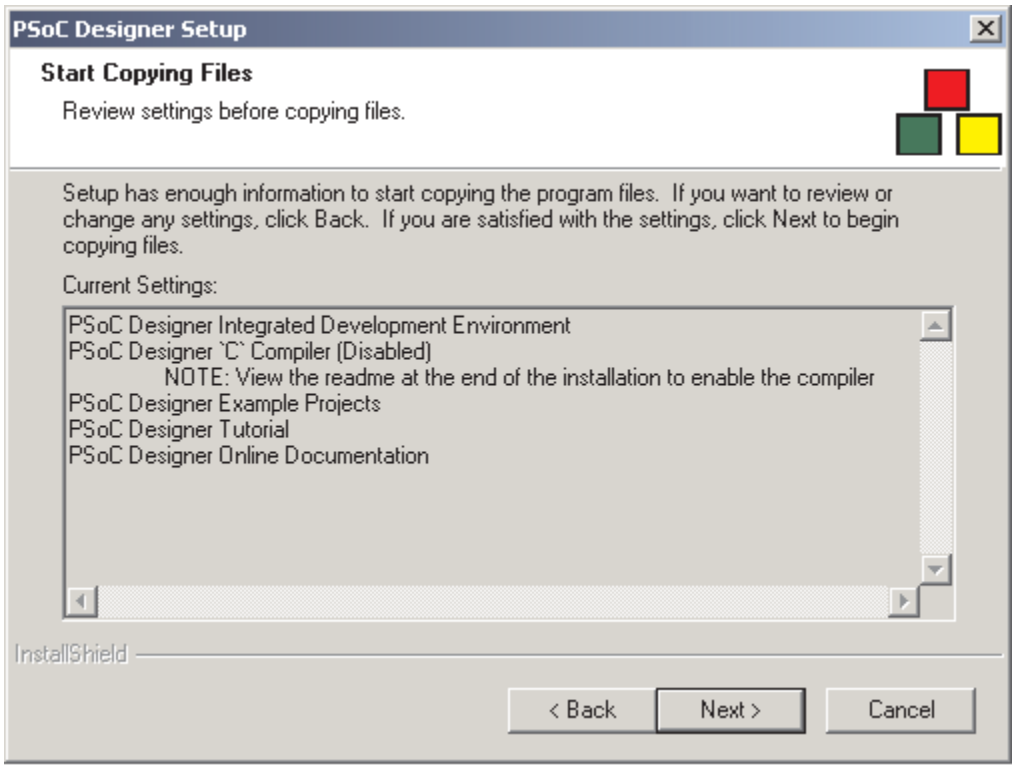


Figure 6: Start Copying Files

8. At the Setup Status screen, you will see a status, in percentage complete, of system installation. Click **Cancel** if you wish to cancel installation at this time.

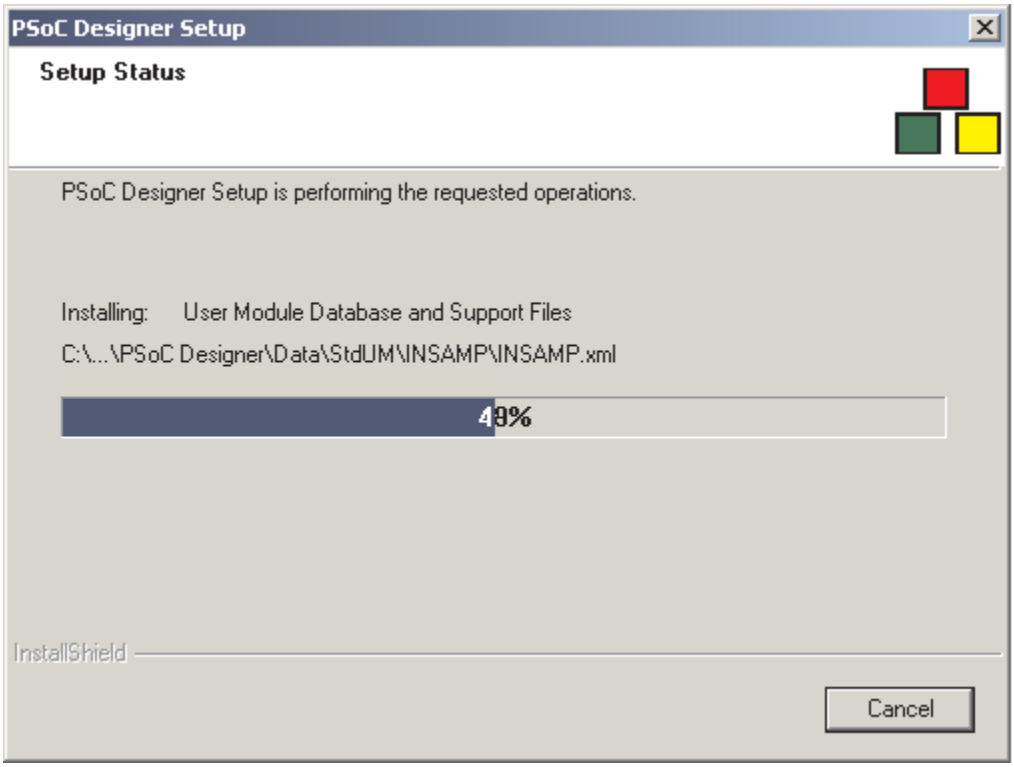


Figure 7: Setup Status

9. At the PSoC Designer Installation Wizard Completed screen, click a check in the applicable box if you wish to view release notes or instructions on enabling the C Compiler. When finished, click **Finish**.

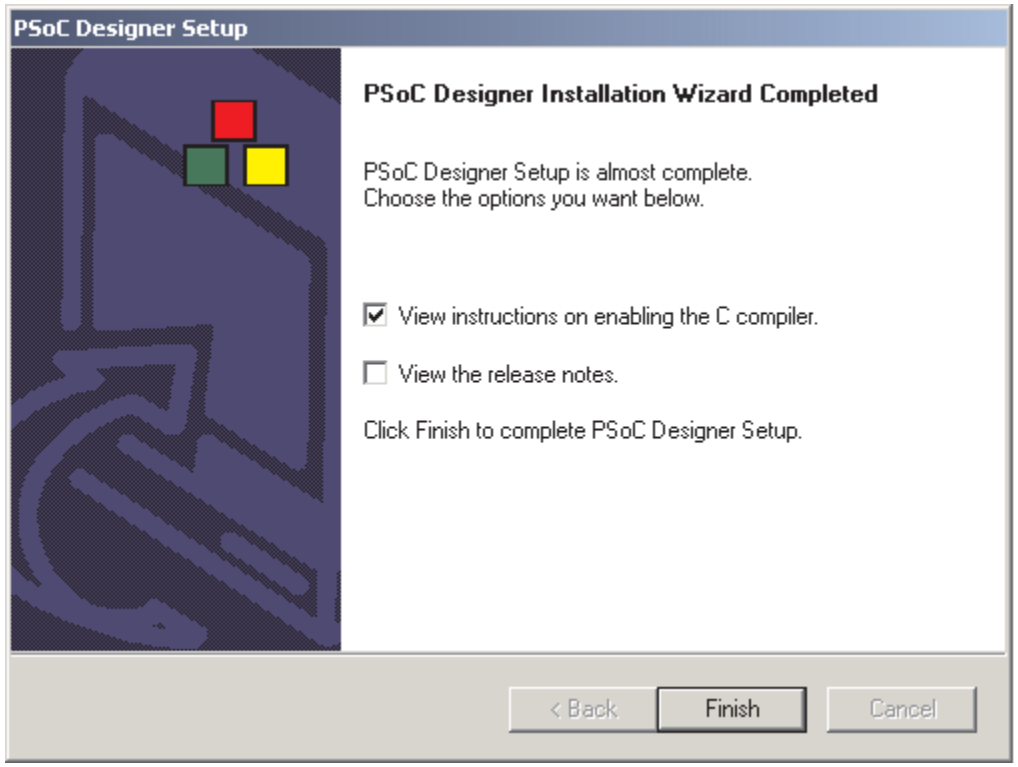


Figure 8: PSoC Designer Installation Wizard Completed

At this time, you can reboot your PC. If you wish to connect the In-Circuit Emulator to your PC and PSoC Designer, go to Section 10.

When you first access the Interconnect View of Device Editor after installation, you will see a pop-up window to jump start your familiarity with the device interface. Click the “x” in the upper-right corner to exit and a check in the “Don’t show this window again” box if you do not want the window to show again. You can re-enable the window by checking Enable Interconnect navigation info under Tools >> Options >> Device Editor tab.

2.5 Project Update

If you download a new version of PSoC Designer you will most likely need to update existing projects (created with an earlier version of PSoC Designer).

If you installed PSoC Designer v. 3.00 and ran Project Update, you will be prompted to run it again if you upgrade to 3.xx or higher because *boot.tpl* has changed.

To update a PSoC project for compatibility, execute the following steps:

1. Open PSoC Designer.
2. Access the project you believe needs to be updated.

At this point, PSoC Designer will “check” if the project is compatible with the new version of PSoC Designer.

3. If your project needs to be updated, you will receive a message accordingly. Click **Update** (or you can update later by selecting, “Update later...”).

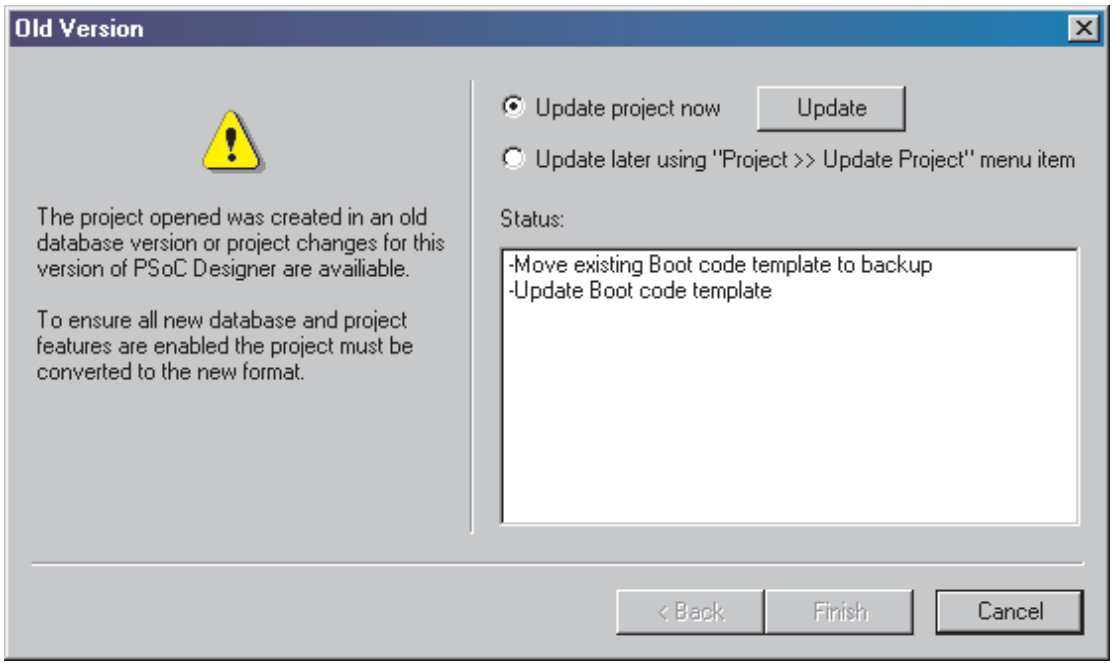


Figure 9: Database Version Detection

4. Once the update has occurred, click **Finish** and your project will be compatible with the current version of PSoC Designer.

If there are User Modules that have a version number different from the application, a dialog box will appear after the Project Update dialog. If this project source is generated, check the backup files for any changes that might be relevant.

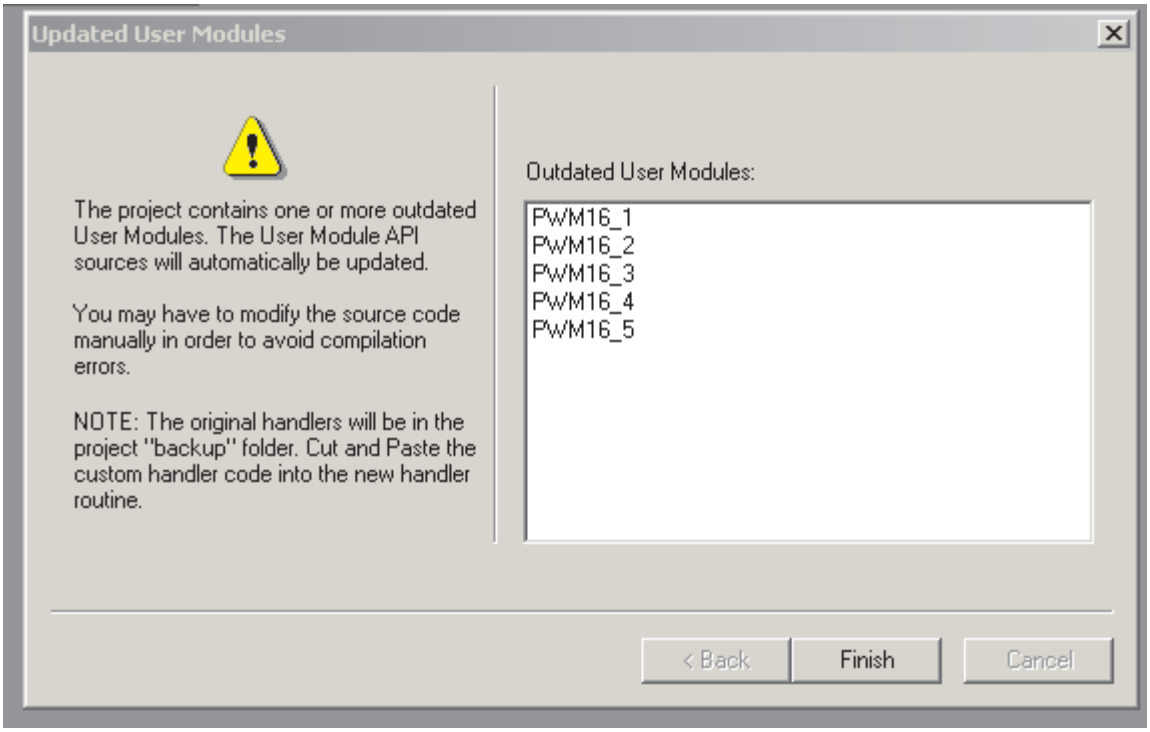


Figure 10: Outdated User Modules

Project Update has updated *boot.tpl* (the template for *boot.asm*). If you have modified *boot.tpl* in your previous version of PSoC Designer (including version 3.00), Project Update placed your custom file in the `\Backup` folder of your project directory. You can use this file for reference to manually modify the new *boot.tpl* installed by Project Update.

To change the part for a project from CY8C25xxx/26xxx to CY8C27xxx and beyond, you must clone. See 4.2.2 for details. If you are migrating a CY8C25xxx/26xxx part project to CY8C27xxx, you must also manually use an _underscore label on all interrupt subroutines in each *int.asm* file. Note that there will be one **int.asm* file per placed User Module. Updating User Modules places **int.asm* files in the `\Backup` folder.

Migration from 4.0 to 4.1 is automatic for most User Modules. If the old project contains user code in **int.asm* files, check the copy of the old file in the `\Backup` folder for user code markers. User code markers are comment lines that contain the text `@PSoC_UserCode_xxx@` and `@PSoC_UserCode_END@`. If the old file contains these markers, then the update is automatic. If the markers are not present, the user code must be manually copied from the old file to the newly generated file. The user code must be placed in appropriate locations between valid user code markers. Once the user code is copied, the code will be carried forward every time the User Module source files are generated.

Section 3. Using the IDE

In this section you will learn fundamentals of the system interface as well as how to use project settings and tool options.

3.1 System Diagram

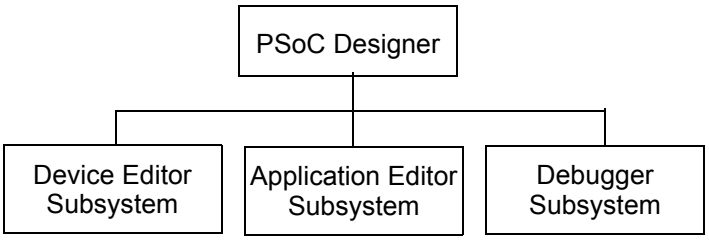


Figure 11: PSoC Designer Subsystems

To start PSoC Designer, go to Start >> Programs >> Cypress MicroSystems >> PSoC Designer.

3.2 File Types and Extensions

When you create a project (see Section 4.), a root directory with three folders will be generated at the location specified by you. The name of the root directory will be the project name and the names of the three folders are lib (Librarian), obj (Objects), and output (for files generated by a project build).

The lib folder contains system Library Source files.

The obj folder contains intermediate files generated during the compiling/assembling of .c and assembly-source files.

The output folder contains the project .rom file (used for debugging), the listing file, and other files that contain debug information.

Upon installation and subsequent system use you will have access to the following files:

Table 4: File Types and Extensions

Type	Extension	Location	Description
Library/ Archive	.a	…\lib\libpsoc.a but Libraries can be anyplace	A collection of object files, created by *ilibw.exe*.
Assembly Source[1]	.asm	Source Files\ Library Source in source tree	Editable assembly-language source file (created initially, added, or generated for APIs).
C Source[1]	.c	Source Files in source tree	C compiler-language file that can be added to the project.
CFG File	.cfg	Folder under project directory	Project configuration file that can be imported and exported for Dynamic Reconfiguration
Debug Symbols	.dbg	…\output folder under project directory	Generated during the build process. Used by the Debugger subsystem.
HEX File	.hex	…\output folder under project directory	Output file in Intel HEX format generated during the build process.
C Header	.h	C Headers in source tree	Editable c-language include file (generated for APIs)
ASM Include[1]	.inc	ASM Include Headers in source tree	Editable assembly-language include file (generated for APIs).
Relative Source Listing	.lis	…\obj folder under project directory	Relative address listing file generated by the assembler.
Full Program Listing	.lst	…\output folder under project directory	Full program listing. Used by the Single-Step ASM function.
Make	.mk	Menu under Project >> Open local.mk file	Customize the build/Make process for a particular PSoC Designer project.
Address Map	.mp	…\output folder under project directory	Generated during the build process. Identifies global symbol addresses and other attributes of output.
Object Module	.o	…\obj folder under project directory	Intermediate, relocatable object file generated during assembly/compilation.
Template[1]	.tpl	Installation directory under …\Templates then copied to project directory	Template files used to generate project files (*boot.tpl* >> *boot.asm*).

Table 4: File Types and Extensions, continued

ROM File	.rom	…\output folder under project directory	Output file in raw binary image generated by device configuration, placed in *PSoc-Config.asm*, and updated during the build process. This file alone will be down-loaded to the ICE for project debugging.
Assembly of C	.s	Found near the C file	Assembly generated from the C source code.
Project Database[1]	.soc	Project directory	Project file accessed under File >> Open Project.
Text Document	.txt	Project directory	Text document that contains system information.
WNP File	.wnp	Project directory	Persistence file unique to PSoC Designer. Contains project information restored each time the project is opened.
XML Document[1]	.xml	Project directory	Device resource file.

1 If you are using a version control system to track project process, copy the above checked files including *m8c.inc* (as the only .inc file) and not including *boot.asm* (as it is recreated during the device configuration process). Also include any **INT.asm* files that have been modified. All other project files will be regenerated during the device application configuration process .

Most of these files are editable and appear in the left frame of the system interface inside the folder bearing the project name.

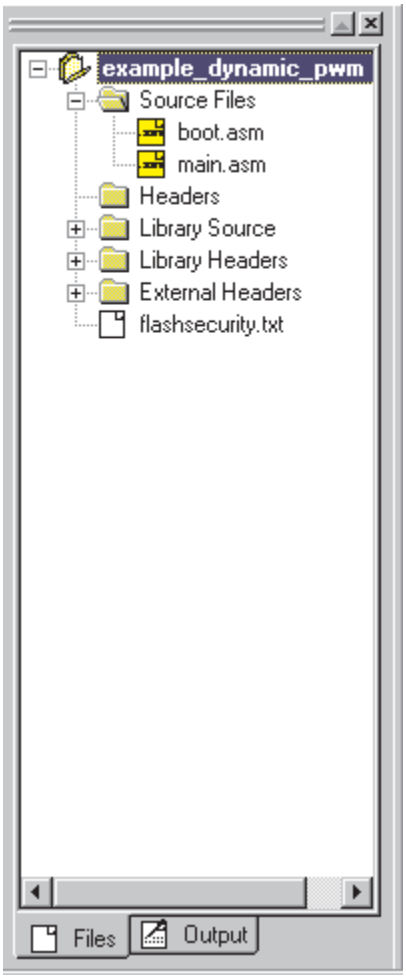


Figure 12: Source Tree

The project file system (source tree) is set up identical to the standard Windows file system.

The Source Files folder contains assembly-language code and C Compiler files generated by the system and you.

The Headers and Library Headers folders contain intermediate files added by device configurations and you.

Library Source contains the project configuration .asm as well other project-specific reference files generated by device configuration.

To access and edit files simply double-click the target file.

Open files appear in the main window (to the right of the source tree).

For more details regarding files and recommended usage see Section 7.

To access the project source tree any time while in any subsystem, click the **Project View** icon .

If you are viewing the source tree in the Debugger subsystem, you will see an Output tab. In the Output tab you can access the project .lst and .mp files. Because these files are generated output from your assembled and linked source, they are Read Only.

You can also access the Output tab on the source tree in Application Editor by clicking Tools >> Options >> Editor tab and uncheck “Enable Output...”

3.3 Project Manager

PSoC Designer contains three subsystems; Device Editor, Application Editor, and Debugger. The interface is split into several active windows that differ depending on which subsystem you are in.

If you are in the Device Editor subsystem, by default you will see a User Module window, a User Module selection window, a resource manager window, and two User Module information windows, which include a block diagram and data sheet for the chosen modules .

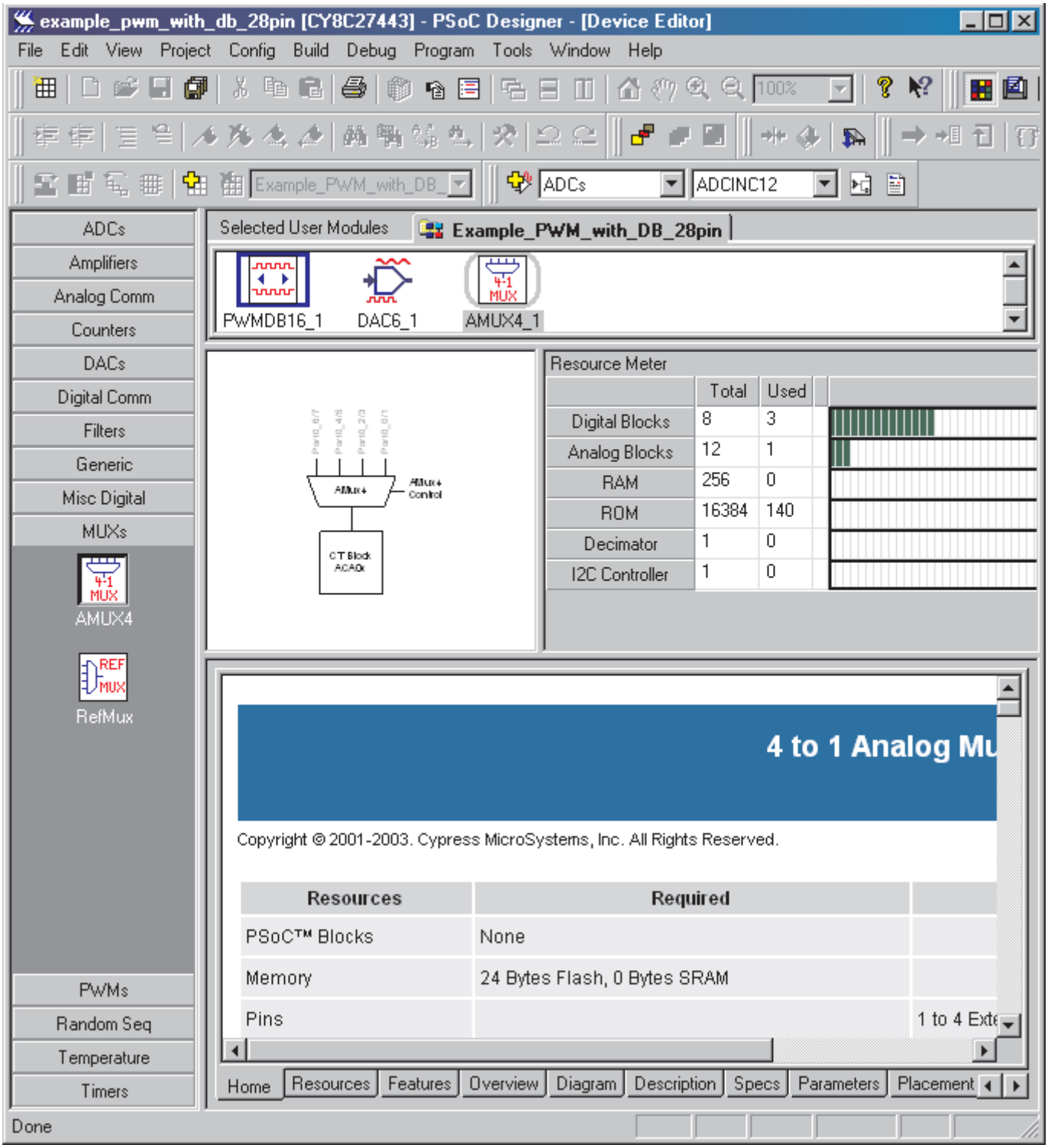


Figure 13: Device Editor Subsystem

To resize any of the windows, just hover your mouse over the dividing line until your pointer becomes a two-sided arrow then drag up or down, left or right.

The amount of information and functionality in the Device Editor subsystem is quite extensive. For further details see Section 5.

If you are in the Application Editor subsystem , you will see the project files (source tree) window, the open source-file editing window, and the Output Sta-

tus window, where error messages appear if there are code problems when files are compiled and built.

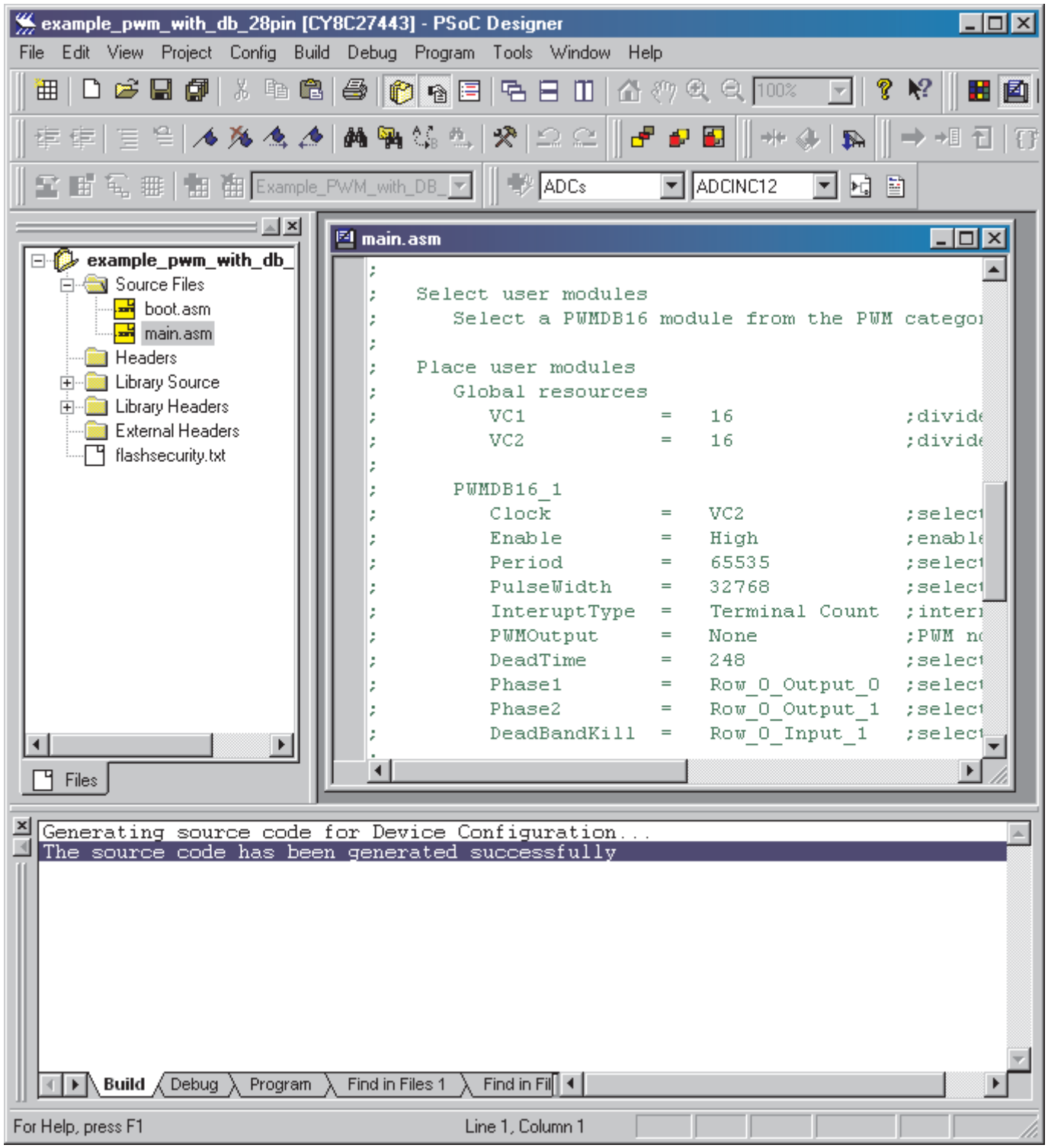


Figure 14: Application Editor Subsystem

To access the Output Status window any time while in any subsystem, click the **Output View** icon .

Last, if you are in the Debugger subsystem, you will see the same active windows as in Application Editor plus CPU register, RAM/Bank/Flash data register, and Watch Variable windows.

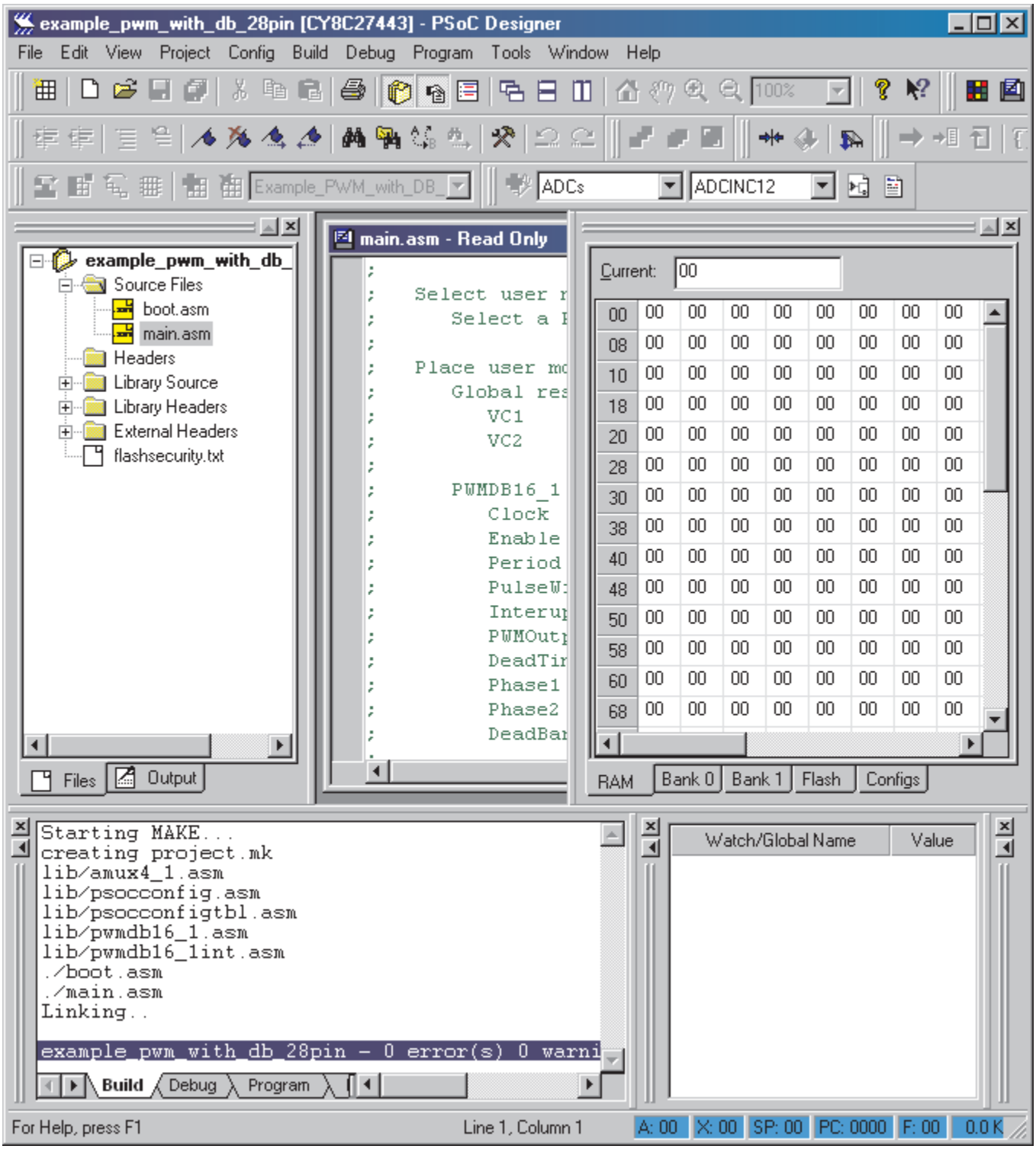


Figure 15: Debugger Subsystem

Click the “arrow” in the upper corner of an active window to expand or collapse the placement. Click the “x” to close. (Use the **View** menu to re-open closed windows.)

As you move between subsystems, you will notice different options being enabled or disabled in the toolbar and menus as applicable to functionality.

3.4 Edit Windows

As mentioned earlier, open assembly language and C Compiler source files reside in the main frame of Application Editor (see Figure 14:). Each file can be opened as a separate window.

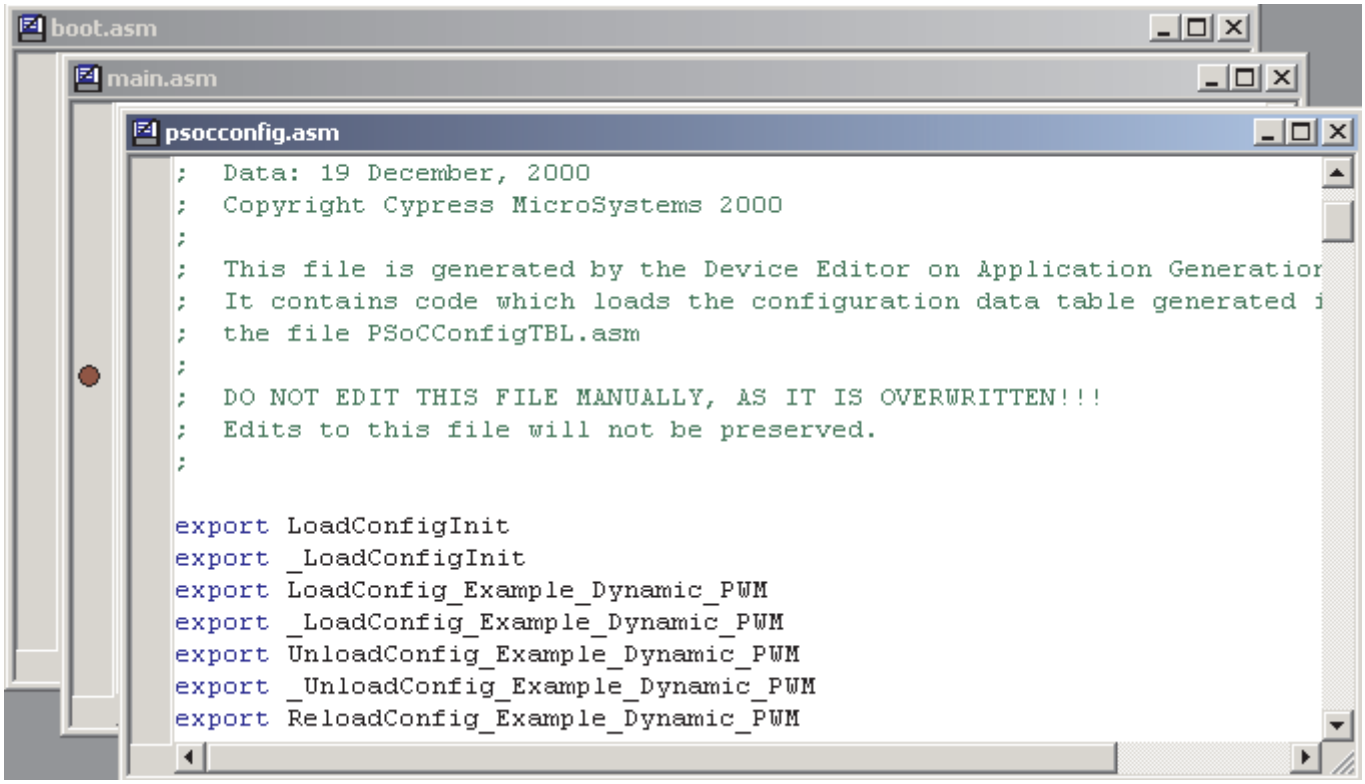


Figure 16: Cascaded Windows

You can use the Window menu or icons to cascade or tile your window display. Options include Cascade, Tile Vertically, and Tile Horizontally.



Figure 17: Window Options

Open files are accessible from within the Debugger subsystem under the **Window** menu but are displayed by default in Application Editor. This is where you add and edit the assembly language and C Compiler source files of your project. See Section 7. for further details on this subsystem.

3.5 Output Status Window

The Output Status (or error-tracking) window of Application Editor is where the status of file compiling/assembling and project building resides.

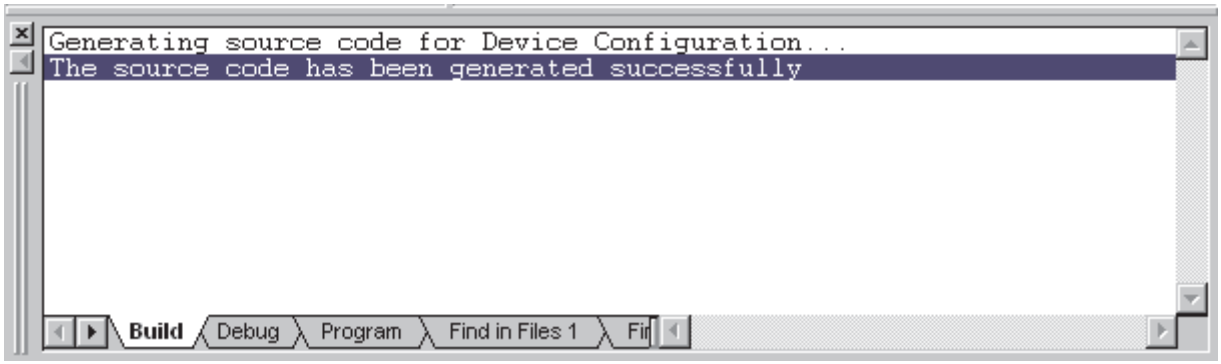


Figure 18: Output Status Window

Each time you compile/assemble files or build the project, the Output Status window is cleared and the current compilation status entered as the process occurs.

When compiling or building is complete, you will the see the number of errors. Zero errors signify that the compilation/assemblage or build was successful. One or more errors indicate problems with one or more files. Such errors include *missing input data* and *undeclared identifier*. For a list of all identified compile and build errors with solutions see Section 8. Compile/Assemble Error Messages in *PSoC Designer: Assembly Language User Guide*. For further details on compiling and building see Section 9. in this user guide.

In PSoC Designer version 3.2x or higher you can right-click your mouse in the Output Status window and either copy or clear specified contents.

In PSoC Designer version 3.2x or higher you can toggle the “verbose” messages in the build (resulting from the “make” utility) by clicking Tools >> Options >> Builder tab and checking or unchecking “Use verbose build messages,” depending on your need. For details about this utility see *make.pdf* in `\Documentation\Supporting Documents` of the PSoC Designer installation directory.

In PSoC Designer version 4.0x or higher you will see Design Rule Checker results in the Output Status window.

3.6 Project Settings

3.6.1 Compiler

In the Project Settings dialog box you can enable the PSoC Designer C Compiler macro features. To access the dialog box, click Project >> Settings, ImageCraft Compiler tab.

Anything added to the Project Settings, ImageCraft Compiler (tab) dialog box is used when a ‘C’ source file is compiled.

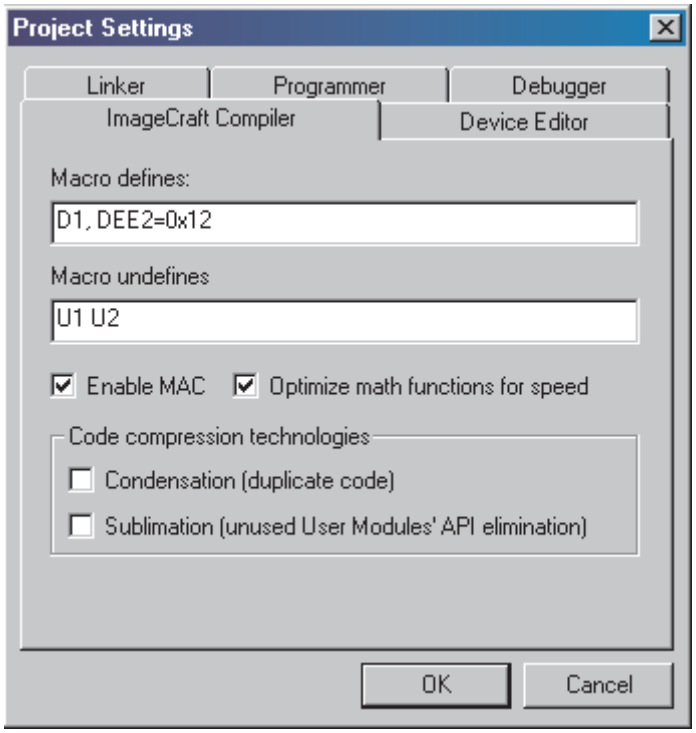


Figure 19: Project Settings Compiler Tab

1. **Macro defines**
   Macro defines are a pre-processor action; things entered for macro defines are expanded in the source file. For example, we could enter DEE2=0x12 in the Macro define field of the Project Settings dialog box. Let’s say that we have a ‘C’ source file with the following instructions:

```
    if (nTotal > 1000)
{
    nTransformBase = DEE2;
```

```
}
```

When this source file is compiled, the code that would assign a number to “nTransformBase” would use the absolute value of DEE2 (e.g., 0x12). The benefit is that the source file does not have to change if a different constant is needed, only the macro define value would change.

Legal Defines: The compiler’s pre-processor expects a macro defined by using a compiler “switch”. This switch is defined in the *PSoC Designer: C Language Compiler User Guide* as “-D”. When you want to add Macro defines using PSoC Designer, they **do not** need to enter the switch, because it (PSoC Designer) knows the information in the Macro define field of the Project Settings dialog box are macro defines

The Macro defines, can have the following syntax:

```
-D<name>[=value]
```

`<name>` is required. `[=value]` is optional. Because we do not need the –D in the Macro define field of the Project Settings dialog box, we could use the following examples:

```
FOO
FOO=12
FOO=BAZ
```

Spaces can be used before and after the equal sign.

Strings in Macros: It is typical for the pre-processor to strip quotations marks in a macro definition. For example:

```
FOO = “CLUB”  … won’t work like #define FOO “CLUB”.
```

You can use a string macro with the help of some source file additions. For example, you enter a macro definition as:

```
FOO=CLUB
```

Your source file should contain a definition to add the quote marks. In this case, in your ‘C’ file you would enter:

```
#define CLUB “CLUB”
```

You can still get the benefits of the macro definition by using the macro name. For example:

`cstrcpy( sSomeArray,  FOO);` … Copy the string `"CLUB"` to an array

Therefore, you can have a set of strings that the macro definition can use. Continuing with the above example, you can have the following string definitions in your 'C' source file:

```
#define CLUB "CLUB"
#define BUNGALOW "BUNGALOW"
#define HOUSE "HOUSE"
```

The macro definition can use any of the above values, for example:

```
FOO=HOUSE
```

Your 'C' code does not have to change, for example:

`cstrcpy( sSomeArray,  FOO);` … Copy the string `"HOUSE"` to an array.

Multiple macro definitions can be used in the Macro define field. The rule is to separate the macros with "white" space (e.g., a space character, one or more) or a comma (,).

2. **Macro undefines**
   The Macro undefines field "blocks" a Macro defines. The syntax for a macro undefine is:

   ```
   <name>
   ```

   No value can be entered. Undefines are useful to temporarily take away a macro. You can add multiple undefines using the same mechanism described in the Macro defines section.

3. **Enable MAC**
   The MAC (or Multiplier/Accumulator) is built-in hardware that performs 8-bit multiplication operations quickly. A problem can crop up if the MAC is being used in both non-interrupt and interrupt code. It would be possible to have non-interrupt code begin executing the instructions that set up the MAC operation, then be interrupted by an Interrupt Service Routine that also uses the MAC. This "simultaneous" use of the MAC will likely cause one (the non-interrupt code) of the code domains of the MAC to get invalid results. This option enables or disables 'C' code generation for using the MAC. If the MAC code generation is disabled, software (library) routines for multiplication and addition will be inserted into the code.

The compiler switch to disable the C code generation for the MAC is: `Wf-Osize`.

4. **Optimize math functions for speed**
   This enable/disable applies to 16-bit multiplications. If you want to optimize math functions for speed, you will have larger code size for the math operations. The compiler's code generation effectively "rolls" out virtually the same code when multiplying multiple values. When this feature is disabled, the 'C' code generation "parameterizes" the values that will be multiplied and calls a "generic" multiply routine. The code is smaller in this case, but due to the (library) function calls, the execution time will be longer.

   The compiler switch to optimize math functions for speed is on by default, however, to optimize the math functions for size the switch is: `Wf-Osize`.

5. **Code compression technologies**
   Certain code compression technologies apply when either 'Condensation' or 'Sublimation' is selected.

   The following applies if **Condensation (duplicate code)** is checked:

   The Code Compressor replaces duplicate code "blocks" with a "call" to a single instance of the code. It also optimizes long calls or jumps (LCALL or LJMP) to relative offset calls or jumps (CALL or JMP).

   Code compression occurs after linking the entire code image. The Code Compressor uses the binary image of the program as its input for finding duplicate code blocks. Therefore, it works on source code written in 'C' or assembly or both.

   To enable, click a check in the **Condensation (duplicate code)** field. This must be done on a per-project basis.

   The following applies if **Sublimation (unused User Modules' API elimination)** is checked:

   This is an optimization feature of PSoC Designer. If you check this field, upon a build the system will go in and remove all "dead code" from the APIs in effort to free up space.

   There are many conditions and rules for the Code Compressor. Please review the following documentation before enabling:

   - *PSoC Designer: C Language Compiler User Guide*

- *PSoC Designer: Assembly Language User Guide*

3.6.2 Device Editor

In the Project Settings dialog box you can enable options for the PSoC Designer Device Editor. To access the dialog box, click Project >> Settings, Device Editor tab.

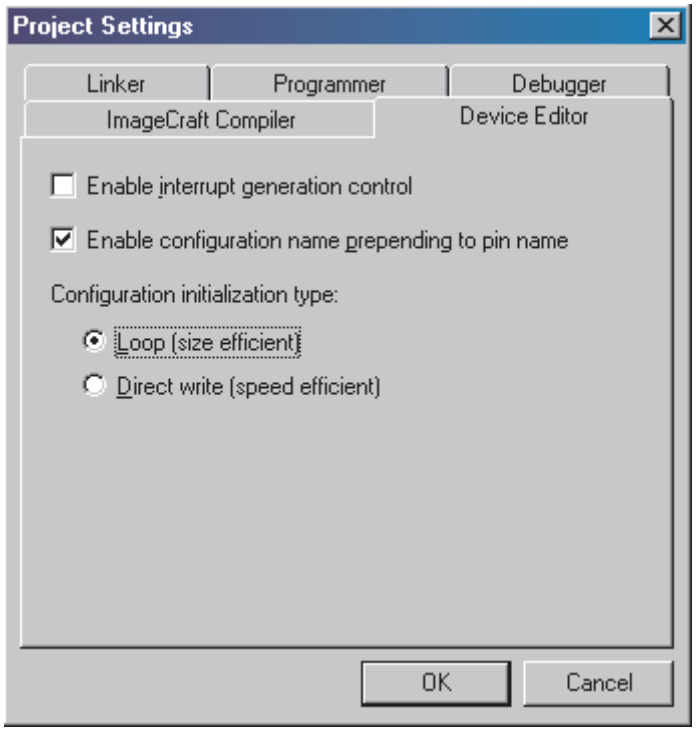


Figure 20: Project Settings Device Editor Tab

There are three Device Editor Project Settings:

1. **Enable interrupt generation control**
   By default, Device Editor generates an interrupt for each User Module that is “placed.” A check indicates you want to determine which User Modules employ an interrupt. You can possibly shorten application run-time.

2. **Enable configuration name prepending to pin name**
   When pins are “used” (and this field is checked), code generation prepends the configuration name to the pin name for all references to the pin in the include files. A pin is considered "used" when at least one of the following settings for the pin is *not* set to the default:
   Setting: Default Value
   Custom Name: Port_m_n

Interrupt Mode: DisableInt

The files that will be pre-pended are *psocgpioint.h* and *psocgpioint.inc*.

3. **Configuration initialization type**
   PSoC Designer initializes to a loop (to save program space) by default. Click “Direct write” if you prefer to speed up execution time by writing directly to the register.

3.6.3 Linker

In the Project Settings dialog box you can enable options for the PSoC Designer Linker. To access the dialog box, click Project >> Settings, Linker tab.

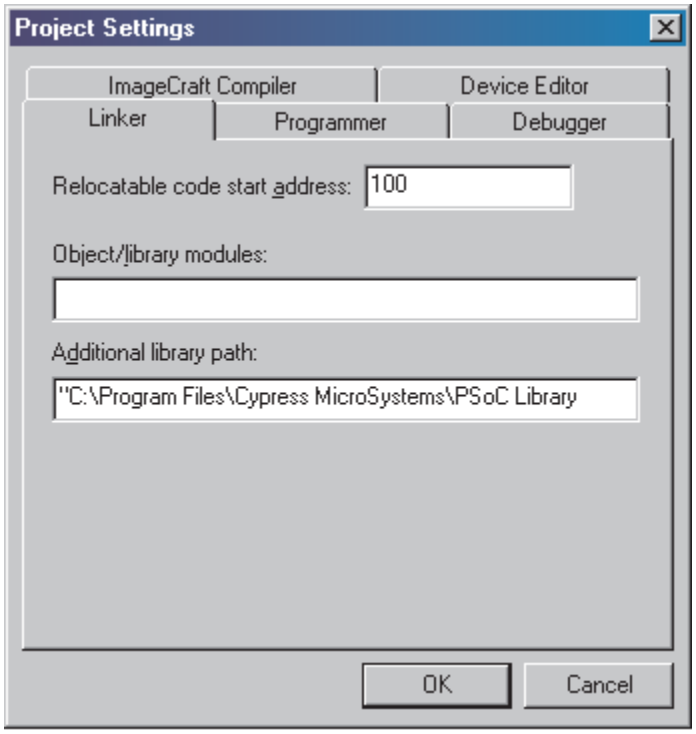


Figure 21: Project Settings Linker Tab

1. **Relocatable code start address**
   The Relocatable code start address setting allows you to set the starting location for the “text” area. The “text” area is the primary relocatable code space used by ‘C’ and User Module APIs. The following diagram shows the default memory organization for a newly created PSoC Designer project:

Table 5: Default Memory Organization

0	Area TOP Absolute Code Segment Reset Interrupt Vectors
~0xB8	Unused Code Space
0x100	Area Text Relocatable Code Segment User and User Module API Code
ROM Size	

The unused code space area will fluctuate, based on custom changes to the startup (boot) code in the TOP area. By inspecting the project-listing file, you can adjust the start of the text area. Some developers may wish to place “boot loader” code or ROM tables in the unused code space area, so the text area must be adjusted accordingly.

2. **Object/library modules**
   The Object/library module setting on the Linker tab of the Project Settings dialog box provides a way to link code from other places. The benefit for using this setting allows the developer to centralize a set of tested code that may be used by more than one PSoC Designer project configuration. (See Section 6. for details on employing multiple configurations per PSoC MCU project.) Linking additional libraries and object files is a relatively common thing.

   It is intended that the object or library filename be entered in this setting field. It is legal to use long file names, but these must be enclosed in double quotes. Following are examples:

   `"Sample Long Filename.a"` >> Valid
   `"Sample BadName.a` >> Invalid - No ending quote
   `foo.o` >> Valid

   Multiple object/library module names can be entered. Each name must be separated by a space, (one or more) characters, or a comma.

3. **Additional library paths**
   The Additional library path setting adds one or more directories to the list of directories searched for object and library files. Directories are searched only until the specified object/library file is found. It is intended that valid directory paths are entered, however if incorrect paths are entered you will receive an error message. The linker searches for the object/library files in

the following order:

--PSoC Project Directory
--First library path listed in the Additional library path field
--Second library path listed in the Additional library path field, and so on…

It is not necessary to add the PSoC Project Directory to the Additional library path, because a search for an object/library file is done in this directory first.

It is intended that one or more paths be entered into this field. It is legal to use long path names, but these must be enclosed in double quotes. It is not expected that a terminating backslash (\) be added to the end of the path. Multiple paths can be entered on the settings line by separating them with a comma or space(s).

Creating a Library: PSoC Designer does not provide a user interface to create/maintain an external library. Libraries can be created from a console (DOS) shell. This shell environment lends itself to scripting, so that the library can be rebuilt, appended, and maintained. PSoC Designer is distributed with a librarian as well as an assembler and compiler.

The first step is to operate in a shell environment. A path to the librarian, assembler, and compiler tools should be created for the console to execute. The path to these tools can be set in a variety of ways. Typically, it resides in a “Tools” folder beneath the PSoC installation directory (e.g. `C:\Program Files\Cypress MicroSystems\PSoC Designer\Tools`).

You must have an error-free object file (typically an .o file type) to add or update to a library. Object files are created after you assemble or compile a file. To create an object file from an assembly source file you would enter the following:

```
iasm8c –I”include path” filename.asm
```

To create an object file from a ‘C’ source file you would enter the following:

```
icc -e -c [options] –I”include path” –I”another include path" filename.c
```

The compiler switches are listed in the *PSoC Designer: C Language Compiler User Guide*. Do not use the –g switch to include debug information.

Once you have successfully obtained an object file you can use the librarian. The librarian is invoked with the following examples:

```
ilibw -a libname.a foo.o >> add or update library module foo.o
ilibw -d libname.a bar.o >> remove library module bar.o
ilibw -t libname.a >> list the library modules
ilibw -x libname.a foo.o >> extract the library module foo.o
```

The library (archive) file will be created, if it did not previously exist, when the add/update librarian action is invoked.

4. **Enable 24 MHz alignment shift**
   Some of the silicon contained an errata condition that involved unpredictable results when code execution occurred through certain code bytes at certain code locations, while running at 24 MHz. This option enables/disables the software “fix up” for this condition.

   The alignment shift is implemented by adjusting (or incrementing) the start of the relocatable area (text area) by the correct number of bytes that will avoid the 24 MHz condition.

   The Build output status window will display messages that show the number of errata conditions, the number of code shifts applied, and the success or failure of the fix up.

   This option setting is disabled by default. You should enable it if you know you will be running at 24 MHz anytime during program execution.

This setting is only available for the CY8C25xxx/26xxx device family. It does not appear for CY8C27xxx parts and future PSoC device families.

5. **Enable silicon errata warnings**
   This setting enables or disables warning messages associated with silicon errata. The warning messages appear in the Build tab of the Output Status window when certain project settings may be susceptible to an errata condition.

   There are currently two errata warnings for the silicon; the 24 MHz condition described in the previous section and a RAM-I/O aliasing errata associated with a UART (RX block) User Module.

This setting is only available for the CY8C25xxx/26xxx device family. It does not appear for CY8C27xxx parts and future PSoC device families.

The RAM-I/O aliasing erratum is described in Application Note UART

Receiver Errata Workaround AN2013 at http://www.cypress.com. This warning will be displayed in the Build tab of the Output Status window when a UART User Module is part of the project.

3.6.4 Programmer

In the Project Settings dialog box you can enable options for the PSoC Designer Programmer. To access the dialog box, click Project >> Settings, Programmer tab.

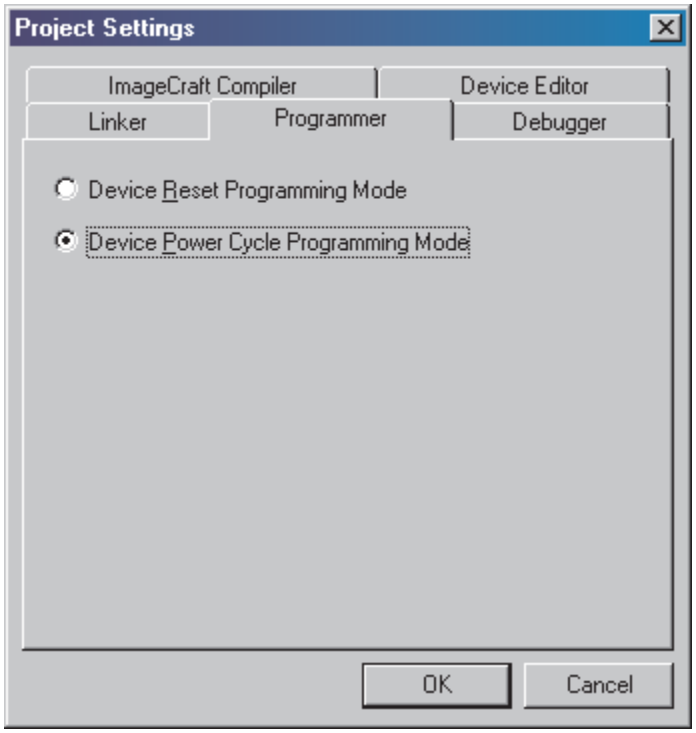


Figure 22: Project Settings Programmer Tab

Device Power Cycle is the default programming mode. Click Device Reset Programming Mode for slightly faster programming or situations where a power cycle is not feasible (ISP programming of a target board with an independent power supply).

Device Power Cycle Programming Mode is required for 8-pin parts, as they do not have a reset pin.

3.6.5 Debugger

In the Project Settings dialog box you can enable options for the PSoC Designer Debugger. To access the dialog box, click Project >> Settings, Debugger tab.

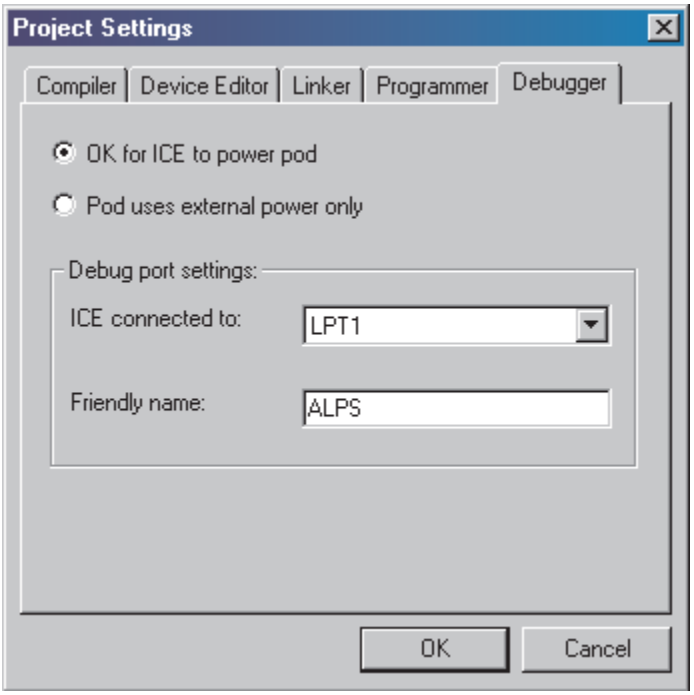


Figure 23: Project Settings Debugger Tab

By default, if the emulation pod is in an unpowered target system, the Cypress MicroSystems' In-Circuit Emulator (ICE) will attempt to power the pod during a connect operation. The ICE supplies 5 V to the pod, which can damage some 3.3 V systems. To prevent the ICE from supplying an over-voltage to an inadvertently unpowered 3.3 V system, click "Pod uses external power only."

This dialog box now provides a drop-down to select the Debug port settings.

The drop-down for the "ICE connected to" entry is a dynamic list, primarily for USB ports. In other words, PSoC USB ICE ports can be added and removed at any time, unlike parallel LPTx ports that are relatively fixed peripherals. Many PSoC USB ICE ports can be used on a single workstation. Each PSoC USB ICE or Dongle contains an electronic serial number, which is displayed in the list when the PSoC USB ICE port is connected via USB. The USB port is removed from the drop-down when the port is disconnected from USB. Utilizing the dynamic nature of this drop-down can help determine what PSoC USB port is connected, since there are no external markings on the USB ICE or ICE Dongle.

The Friendly name field can contain any combination of characters and is used to associate a user-friendly name with a specific port. Port selection is saved on a per-project basis. The friendly name associated with a specific port is a global setting that does not change from project to project.

You will have to change the default port, LPT1, if the ICE is in conflict with a printer (or another port-dependent device) or if you have installed a PCI/ PCMCIA dedicated port card.

3.7 Tools Options

Under Tools >> Options there are several “preference” options for PSoC Designer including Builder, Compiler, Debugger, Device Editor, Editor, Toolbars, Design Rule Checker. Each option is described ahead.

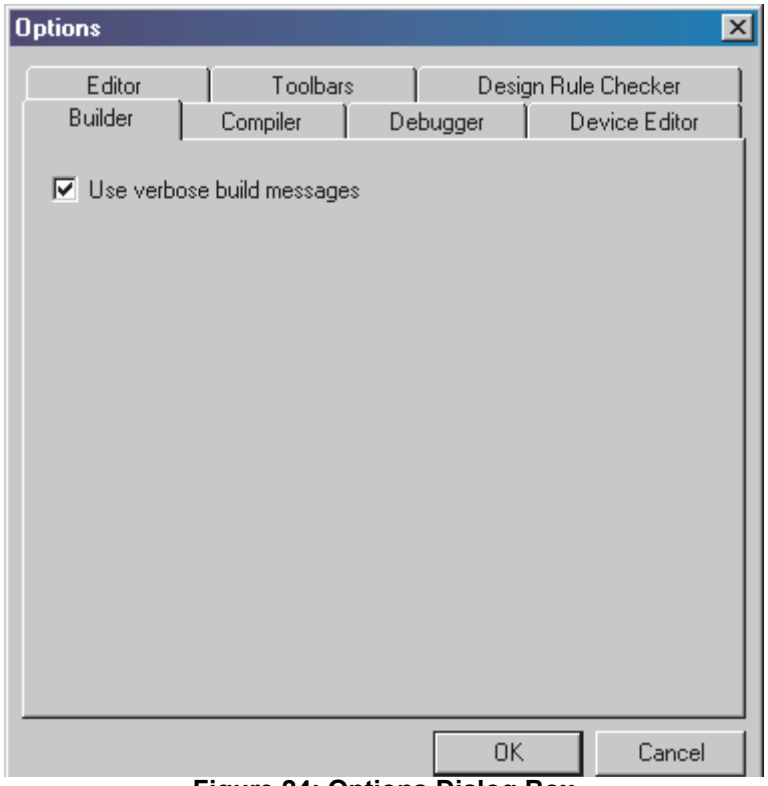


Figure 24: Options Dialog Box

3.7.1 Builder

You can toggle the “verbose” messages in the Output Status window during a build (resulting from the “make” utility). Click Tools >> Options >> Builder tab and check or uncheck “Use verbose build messages,” depending on your need.

3.7.2 Compiler

Enabling the compiler is done within PSoC Designer. To accomplish this, execute the following steps:

1. Access Tools >> Options >> Compiler tab.

2. Enter your *key code*.

You have this *key code* if you purchased the C Language Compiler License when you received PSoC Designer (by download, mail, or through a distributor).

3. Scroll to read the License Agreement, then click a check to accept the License Agreement and hit **OK**.

To view the version details for the iMAGEcraft Compiler, click **Version**. When finished, click the “x” in the upper-right to close.

If, for some reason, you have not received a *key code* or are uncertain of how to proceed, contact the Cypress MicroSystems Application Hotline at 425.787.4814.

3.7.3 Debugger

Under Tools >> Options >> Debugger tab you can specify trace log options.

To specify trace log options, click a check in either PC Only, PC/Registers, or PC/Timestamp.

PC Only mode lists the PC value and instruction only. PC/Registers mode lists the PC, instruction, data, A register, X register, SP register, F register, and ICE external input. PC/Timestamp mode lists the PC, instruction, A register, ICE external input, and timestamp.

These log options can also be specified at Debug >> Trace Mode...

In this tab you can also specify to Use default ICE connection for all projects. With the addition of a USB connection to the ICE, it could be quite possible that multiple USB and or parallel port (LPT) ICE connections could be used from a single developer workstation. There is another potential inconvenience if projects are shared between different workstations since the ICE port setting in the project may not be consistent after the project is moved. Enabling the ‘Use default ICE connection for all ports” setting will allow the developer who works with only one ICE to avoid resetting the ICE port setting in the project. The project setting is not changed in the project database (.soc file) so the

project could be passed back to the original developer without an impact to their ICE port setting.

3.7.4 Device Editor

If you download a new version of PSoC Designer you will most likely need to update existing projects (created with an earlier version of PSoC Designer). This is done automatically by the Project Update feature.

Under Tools >> Options >> Device Editor tab you can specify automation of the Design Rule Checker. If you wish the Design Rule Checker to run automatically after application generation, click a check at Always run Design Rule Checker after application generation. See 5.7 for details.

To re-enable the Interconnect View navigation tip window, check Enable Interconnect navigation info.

3.7.5 Editor

Under Tools >> Options >> Editor tab you can specify system save options and window persistence (i.e., whether or not PSoC Designer retains access to certain windows upon re-opens).

In the “Save options” field, check which group of project files you want automatically reloaded upon a build even if they have been externally modified. Options include Source files, Header files, Library source files, Library header files, and Other files. See Table 4: for project file specifics.

In the “Window persistence” field, check “Reload documents when opening project” if you would like PSoC Designer to retrieve the documents that were open during a previous project session upon future project access.

In “Re-activate previously active document when switching to” of the “Window persistence” field, check Application Editor and/or Debugger if you would like PSoC Designer to re-activate the document that was previously open in either subsystem.

Check “Insert spaces instead of tabs” if you would like PSoC Designer to insert into your source file 5 spaces per [**Tab**] rather than a 5-space jump. This allows you to type source anywhere inside a “tab.”

Finally, you can uncheck “Enable Output file tree...” if you want the Output tab to be accessible on the source tree in Application Editor.

3.7.6 Toolbars

Under Tools >> Options >> Toolbars tab you can check or uncheck which toolbars you want to show or hide.

3.7.7 Design Rule Checker

Under Tools >> Options >> Design Rule Checker tab you can specify the level of severity at which to run Design Rule Checker. The lower the number, the more critical the category-specific rules. See 5.7 for particulars.

Section 4. Creating a Project

In this section you will learn how to create a project and select the device package and pin count. You can create a new project from the initial dialog box or from within an existing project. PSoC Designer provides a wizard to guide you through either process.

4.1 Create a Project

In order to program the desired functionality into the device, you need to first create a project directory in which the files and device configurations can reside.

1. To access the New Project wizard dialog box you can either click the **New Project** icon  or select Start new project from the Start dialog box upon system entry.

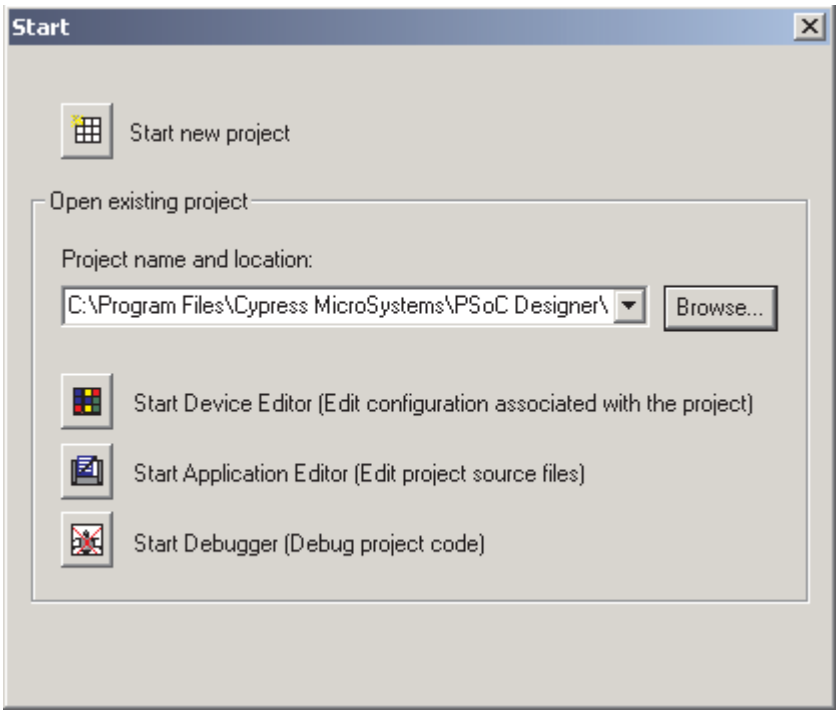


Figure 25: Start Dialog Box

2. Once inside the New Project dialog box, click once on a method, type a Project name, and either type or **Browse** to designate a project directory.

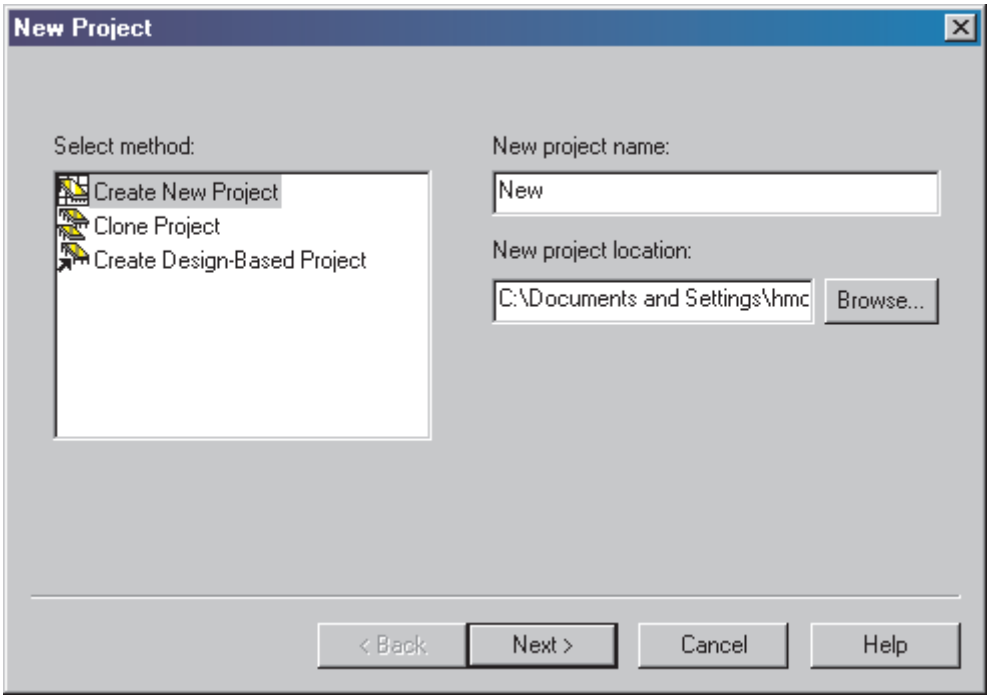


Figure 26: New Project Dialog Box

You can either create a new project, clone an existing project, or apply a design from an existing project. For detailed definitions of each method (Create New Project, Clone Project, Create Design-Based Project), jump ahead in this section to 4.2.

Avoid use of the following characters in path names (they are problematic): \ / : * ? " < > | & + , ; = [ ] % $ ` '.

3. When finished, click **Next**.

At any time you can click **Back** to return to the previous dialog box, **Cancel** to cancel operation, or **Help** to view context-sensitive help.

Once you click **Next**, you will see the New Configuration dialog box.

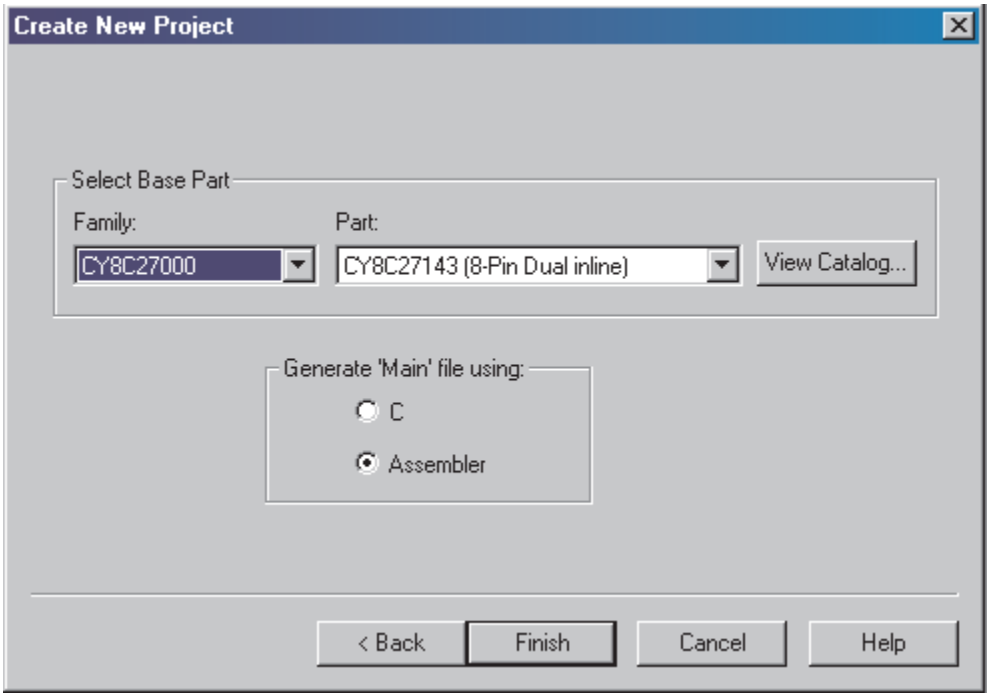


Figure 27: New Configuration Dialog Box

4. Click the drop-arrow in the Select Base Part field and select a part. Click **View Catalog** to access a detailed list of available parts. Highlight your part of choice and click **Select** to save your selection and exit the dialog box.

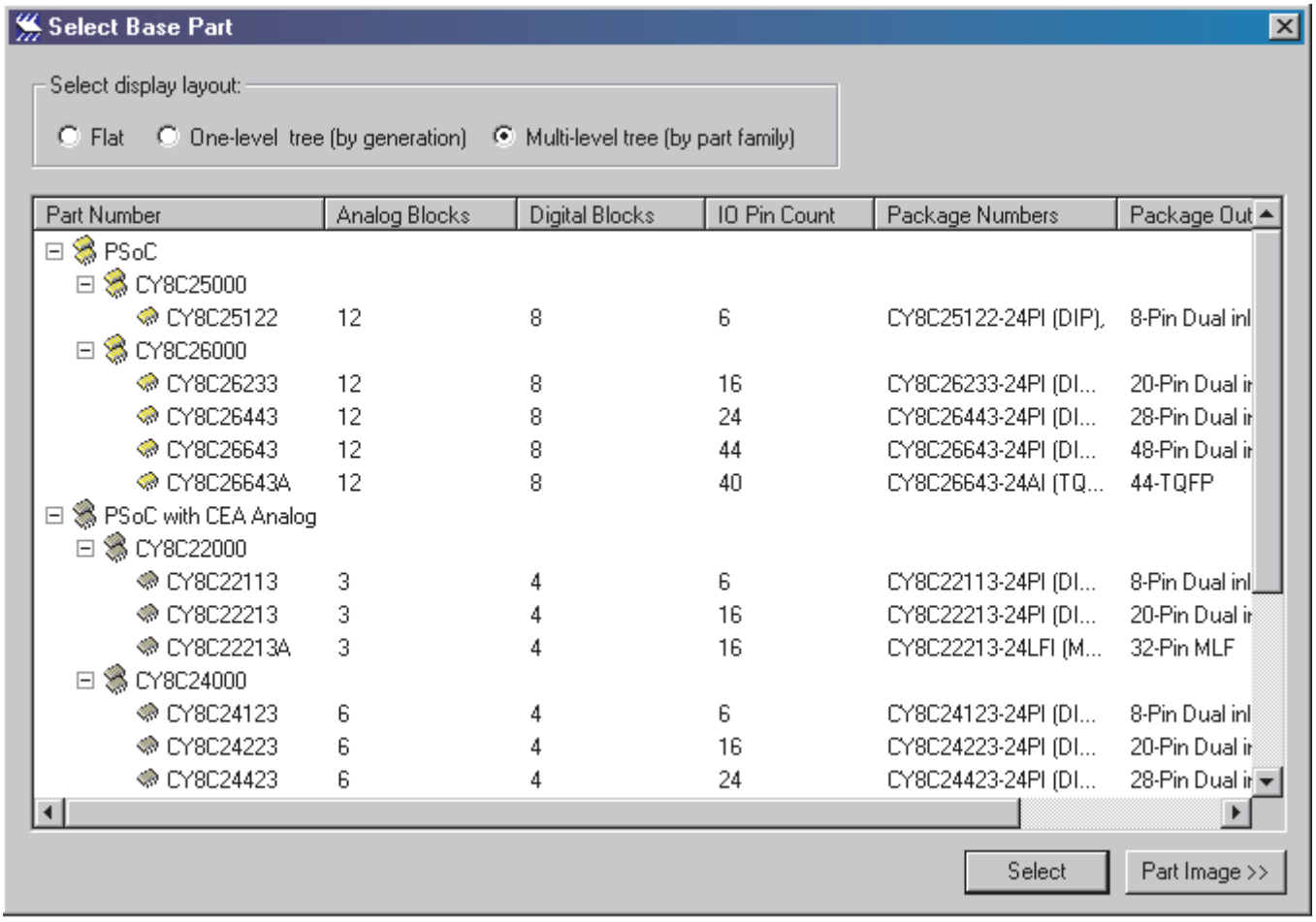


Figure 28: Parts Catalog Dialog Box

You have several options in this dialog box including layout display, viewing part image, and sorting part selection (by clicking on a chosen column title).

5. Once you have selected a part, click C or Assembler to designate the source in which you want the system to generate the “main” file. (Note that C will only be an option if the C Compiler has been enabled in your version of PSoC Designer. See 3.7.2 in this guide or *PSoC Designer: C Language Compiler User Guide* for enabling instructions.)

6. Click **Finish**.

Your project directory with folders is created and can be seen in the source tree (left frame) of the Application Editor subsystem.

4.2 Project Methods

Following is a definition for each method to help you decide the best option for your project:

4.2.1 Create New Project

Creating a new project is described in 4.1 earlier in this section. PSoC Designer will provide a “blank” project configuration for which you select a part to be configured. Once you create your project, select your part, and click **Finish**, you are taken directly to the Device Editor subsystem where you choose and configure User Modules then generate the “new” device configuration. See Section 5. for details on device configuration.

4.2.2 Clone Project

You can clone an existing project at any point of its existence; before, during, or after device configuration, assembly-source programming, or project debugging. Cloning copies the existing project but allows users to change the base part.

If you wish to change parts within a part family in the middle of project design (say from CY8C27143 to CY8C27243), you must use the clone method. If you are migrating from CY8C25xxx/26xxx to CY8C27xxx and beyond, cloning is the only way. See the Project Migration Application Note under Help >> Documentation, Migrating Projects to CY8C27.

If you are migrating a CY8C25xxx/26xxx part project to CY8C27xxx, you must manually use an _underscore label on all interrupt subroutines in each *int.asm* file. Note that there will be one *int.asm* file per placed User Module. Updating User Modules (2.5) places **int.asm* files in the `\Backup` folder.

To clone an existing project, perform the following steps:

1. Start as if you are creating a new project but in the New Project dialog box, click Clone Project in the Select method field (refer back to Figure 26:). Type a Project name and either type or **Browse** to designate a project directory.

2. Click **Next**.

3. Once you click **Next**, you will be asked if you wish to create a new directory for the cloned project with its new name. Click **Yes**.

4. In the Existing Configuration dialog box, **Browse** (or type) to identify the existing directory of the project you wish to clone.

   If you wish to specify an alternative part (device), do so in the Select Base Part drop-down.

5. Finally, select the subsystem in which you would like to begin: Device Editor, Application Editor, or Debugger.

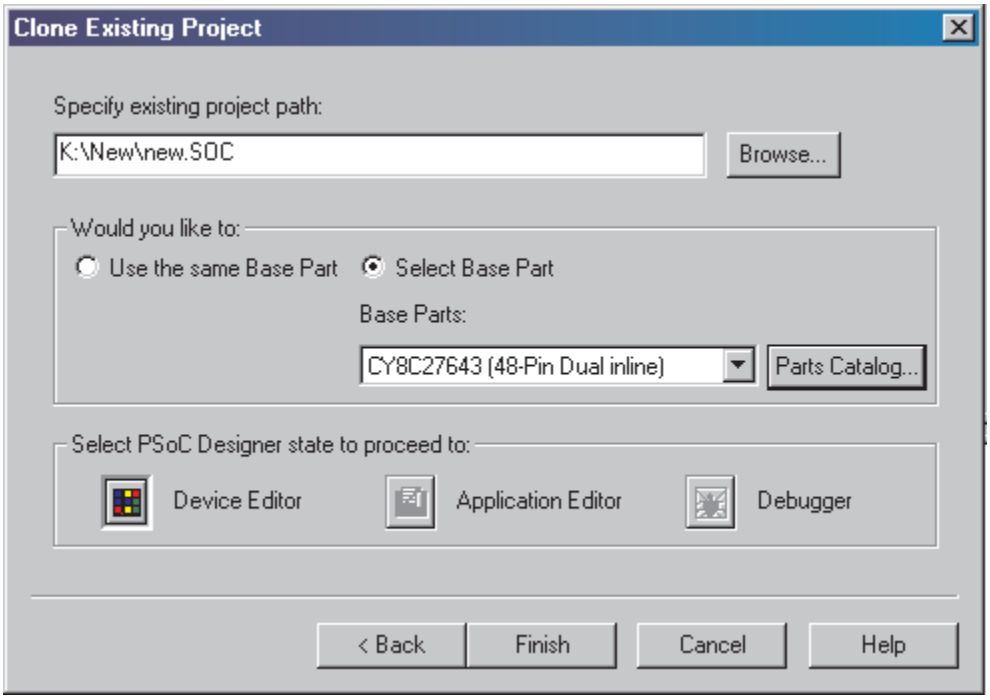


Figure 29: Existing Configuration Dialog Box

If the base part is different from the base part of the cloned project, then the pin settings in the resulting project will be set to the default values.

6. When finished, click **Finish**. Your new project directory will be created and can be seen in the source tree.

If you wish to move an existing project from one directory to another, use the cloning method to create a new “cloned” project in the new directory (rather than employing a physical move).

4.2.3 Design-Based Project

Creating a design-based project is directly related to 6.2. You can create a project based on exported designs. A design is a single (or collection of) exist-

ing, loadable configuration(s) from a project. A loadable configuration consists of one or more “placed” User Modules with module parameters, Global Resources, set pin-outs, and generated application files. PSoC projects can consist of one or multiple loadable configurations. This feature allows you to efficiently use and re-use configurations, thus saving design time and resources.

For complete details on using multiple configurations in a single application, see Section 6.

To create a design-based configuration project, perform the following steps:

1. Start as if you are creating a new project but in the New Project dialog box, click Create Design-Based Project in the Select method field (refer back to Figure 26:). Type a Project name and either type or **Browse** to designate a project directory.

2. Click **Next**.

3. Once you click **Next**, you will be asked if you wish to create a new directory for the project with its new name. Click **Yes**.

4. In the Design-Based Project dialog box, click **Design Browser** to identify the design out of which you wish to create the project.

   See 6.2 to learn how to navigate the Design Browser as well as export and import design configurations.

   If you wish to specify an alternative part (device), do so in the Select Base Part drop-down.

   Specify a programming language for the project ”main” file.

Finally, select the subsystem in which you would like to begin: Device Editor, Application Editor, or Debugger.

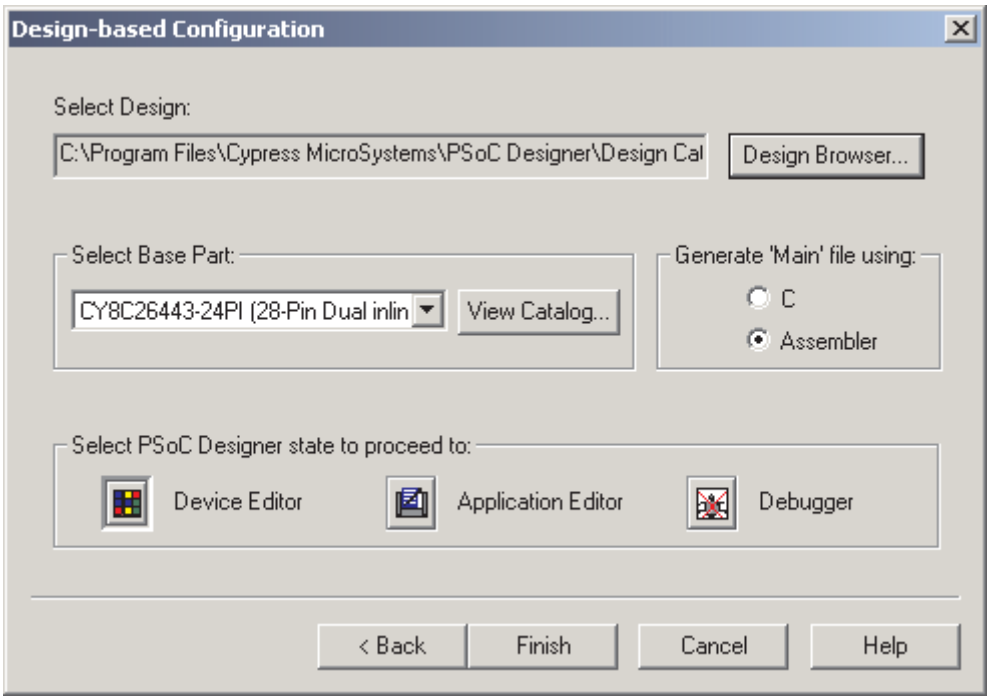


Figure 30: Design-Based Configuration Dialog Box

5. When finished, click **Finish**. Your new project directory will be created and the project will display.

Importing make take a few minutes depending on the size of the design.

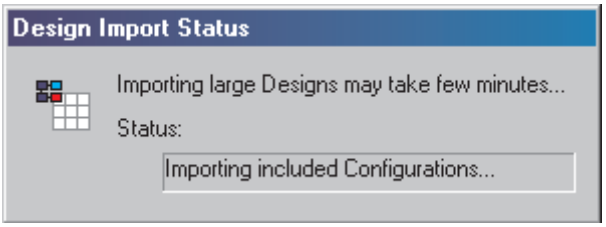


Figure 31: Design Import Status

4.3 Project Backup Folder

PSoC Designer maintains a backup folder in the project directory for files that have been removed from the source tree. This includes files that are manually removed and files that are removed due to cloning or code generation. The backup folder only retains the version of the file that was last removed. The files are named identically to the original project file and the `\lib` directory is not retained, i.e., library files are placed directly under the backup folder.

Section 5. Device Editor

In this section you will learn how to navigate, select User Modules, configure and place User Modules on PSoC blocks, make interconnections, set pinouts, track resources usage, invoke Design Rule Checker and generate application files.

5.1 Navigating Device Editor

Becoming familiar with the interface and knowing your options will expedite the design process.

When you first access Device Editor, you will see the following pop-up window to jump start your familiarity with the device interface.

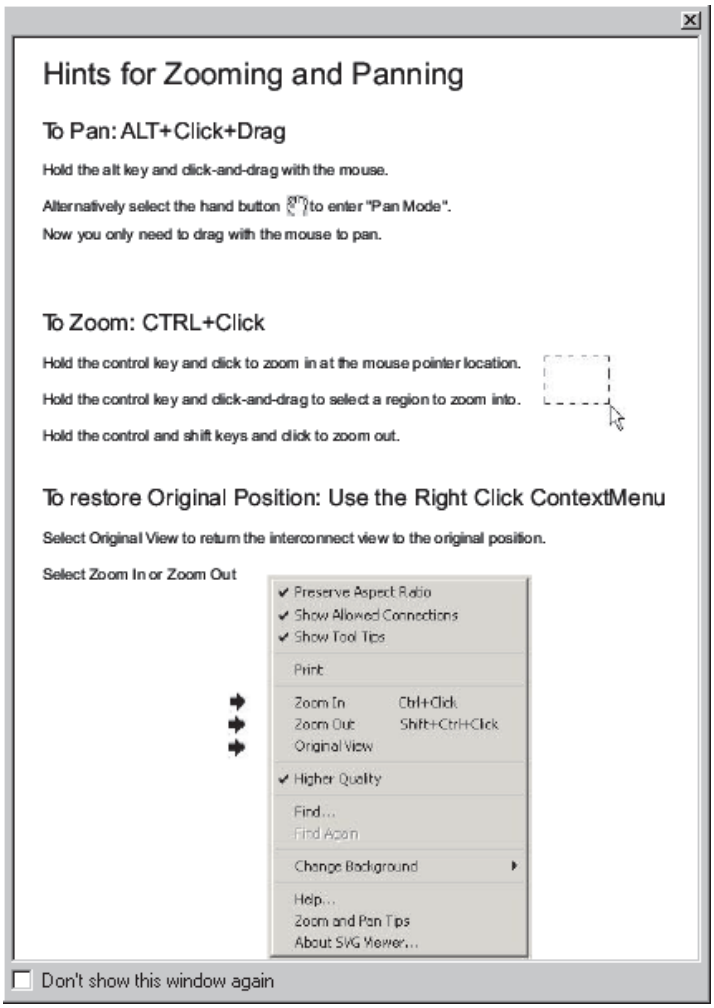


Figure 32: Zoom/Pan Pop-up Window

Click the “x” in the upper-right corner to exit and a check in the “Don’t show this window again” box if you do not want the window to show again.

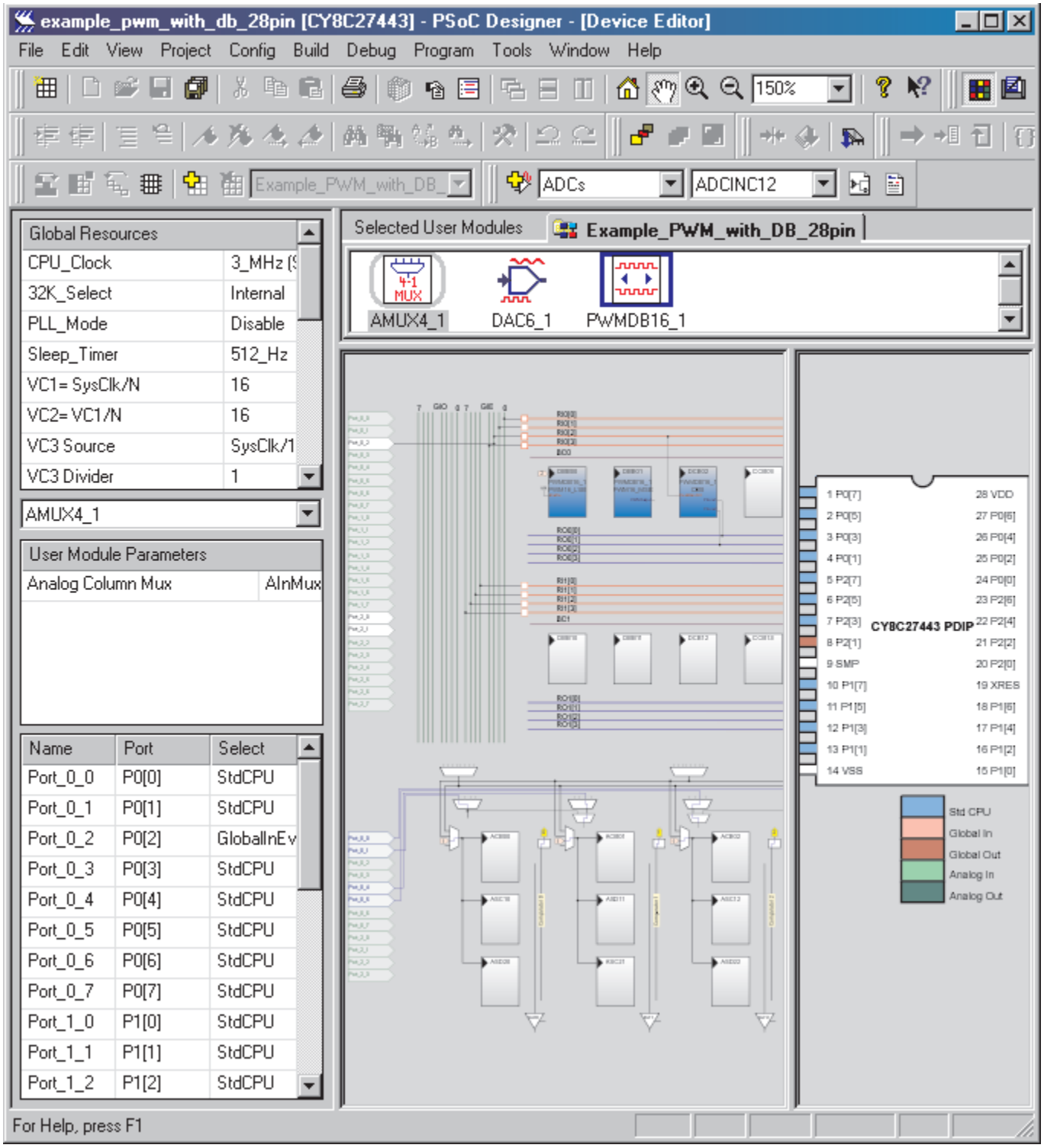


Figure 33: Interconnect View

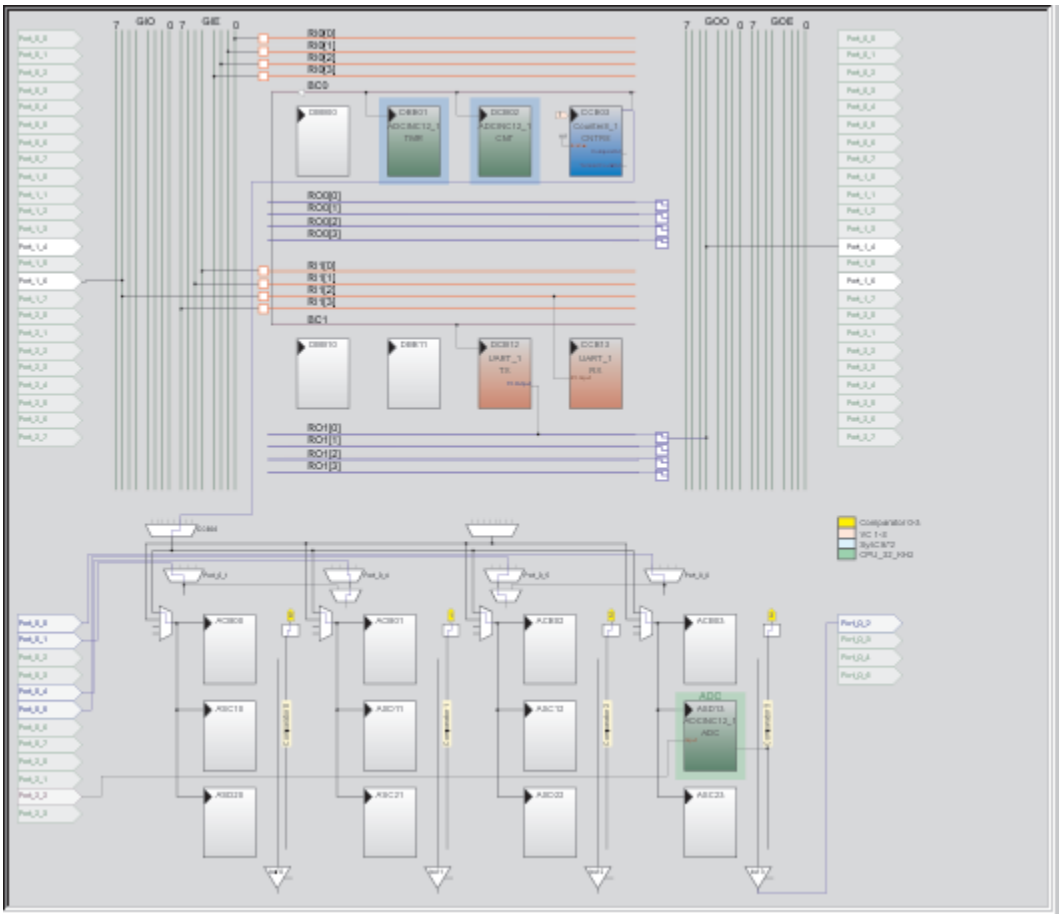


Figure 34: Device Interface View

As you slowly move your mouse around the device interface, the tooltips will guide you as will the Legend.

Use the Pan/Zoom toolbar to click and navigate your way around.



Figure 35: Device Interface Pan/Zoom Toolbar

Click the Hand icon to “pan.” Click the Plus or Minus icons to zoom in or out. Use the drop-arrow to specify a size and the Home icon to return to your original view.

Note that you can “pan” either frame of the Interconnect View Figure 33:, the device interface and the pin-out.

Following is a description of additional options available to navigate the device interface:

Table 6: Navigating Device Interface

Feature	Action	Description
Pan (with Mouse)	Hold down [**Alt**] and click and drag your mouse to pan the interface.	Panning moves the focal point of the view up, down, right or left.
Pan (with Arrow Keys)	Enable [**Scroll Lock**] and you can pan the interface with the arrow keys.	Panning moves the focal point of the view up, down, right or left.
Zoom In (with Mouse)	Hold down [**Ctrl**] and click your mouse to zoom in at the specified location.	Zooming in enlarges the view.
Zoom Out (with Mouse)	Hold down [**Ctrl**] [**Shift**] and click your mouse to zoom out.	Zooming out shrinks the view.
Zoom a Region	Hold down [**Ctrl**] and click and drag your mouse over a region to zoom into.	Zooming in on a region enlarges a specified portion of the interface view.
Preserve Aspect Ratio	Right click your mouse in the interface and select Preserve Aspect Ratio from the menu.	Preserving the aspect ratio maintains the proportion of your view as you pan and zoom around the interface. Note that this will remain on until you remove the check from the menu.
Show Allowed Connections	Right click your mouse in the interface and select Show Allowed Connections from the menu.	Showing allowed connections highlights all possible connections with a red/green fan each time you click a drop-down list of options. Note that this will remain on until you remove the check from the menu.
Show Tool Tips	Right click your mouse in the interface and select Show Tool Tips from the menu.	Showing tool tips enables definitions as you hover your mouse over images in the interface. Note that this will remain on until you remove the check from the menu.
Print	Right click your mouse in the interface and select Print from the menu.	Printing allows you to print the device interface.
Zoom In	Right click your mouse in the interface and select Zoom In from the menu.	Zooming in enlarges the view.

Table 6: Navigating Device Interface, continued

Feature	Action	Description
Zoom Out	Right click your mouse in the interface and select Zoom Out from the menu.	Zooming out shrinks the view.
Original View	Right click your mouse in the interface and select Original View from the menu. You can also use the menu at View >> Interconnection View.	Original View takes you back to the original, default view of the device interface.
Higher Quality	Right click your mouse in the interface and select Higher Quality from the menu.	Higher Quality displays the interface at a higher quality but loads slower. Note that this will remain on until you remove the check from the menu. Removing the check will display a lower quality view but load faster.
Find	Right click your mouse in the interface and select Find from the menu.	Find brings up a standard Find dialog box where you can search the entire interface, regardless of how much of it is currently in view. The result of the search is highlighted, and panned (but not zoomed) into view.
Find Again	Right click your mouse in the interface and select Find Again from the menu.	Find Again automatically launches a search of the word or string for which you previously searched.
Help	Right click your mouse in the interface and select Help from the menu.	Help launches the help system pointing to **Navigating Device Interface**.
About SVG Viewer	Right click your mouse in the interface and select About SVG Viewer from the menu.	Adobe SVG Viewer is the engine that drives the device interface. Here you will see version and trademark information.

5.2 Selecting User Modules

Selecting applicable User Modules is the first step (after creating a project) to configuring your target device. A User Module is a pre-configured function that once placed and programmed will work as a peripheral on the target device

To access Device Editor, click the **Device Editor** icon .

By default, you will be in the User Module Selection View mode of the subsystem.

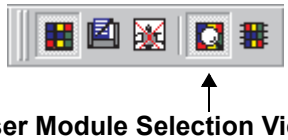


Figure 36: User Module Selection View in Toolbar

In the left frame you will see the User Module Options.



Figure 37: User Module Options

To view the individual User Modules, click one of the set titles (i.e., Timers, Counters, PWMs, etc.) and scroll to see pre-configured options.

By right-clicking your mouse on a User Module icon you can select the module (for eventual placement), view its revision properties, or change the icon from large to small (and back again).

In the User Module Selection View, you can view configuration data related to an individual User Module. To view device/module data, single-click a User Module from within a set title. In other active windows you will see all related configuration data. The following figure shows data related to a DTMF Dialer.

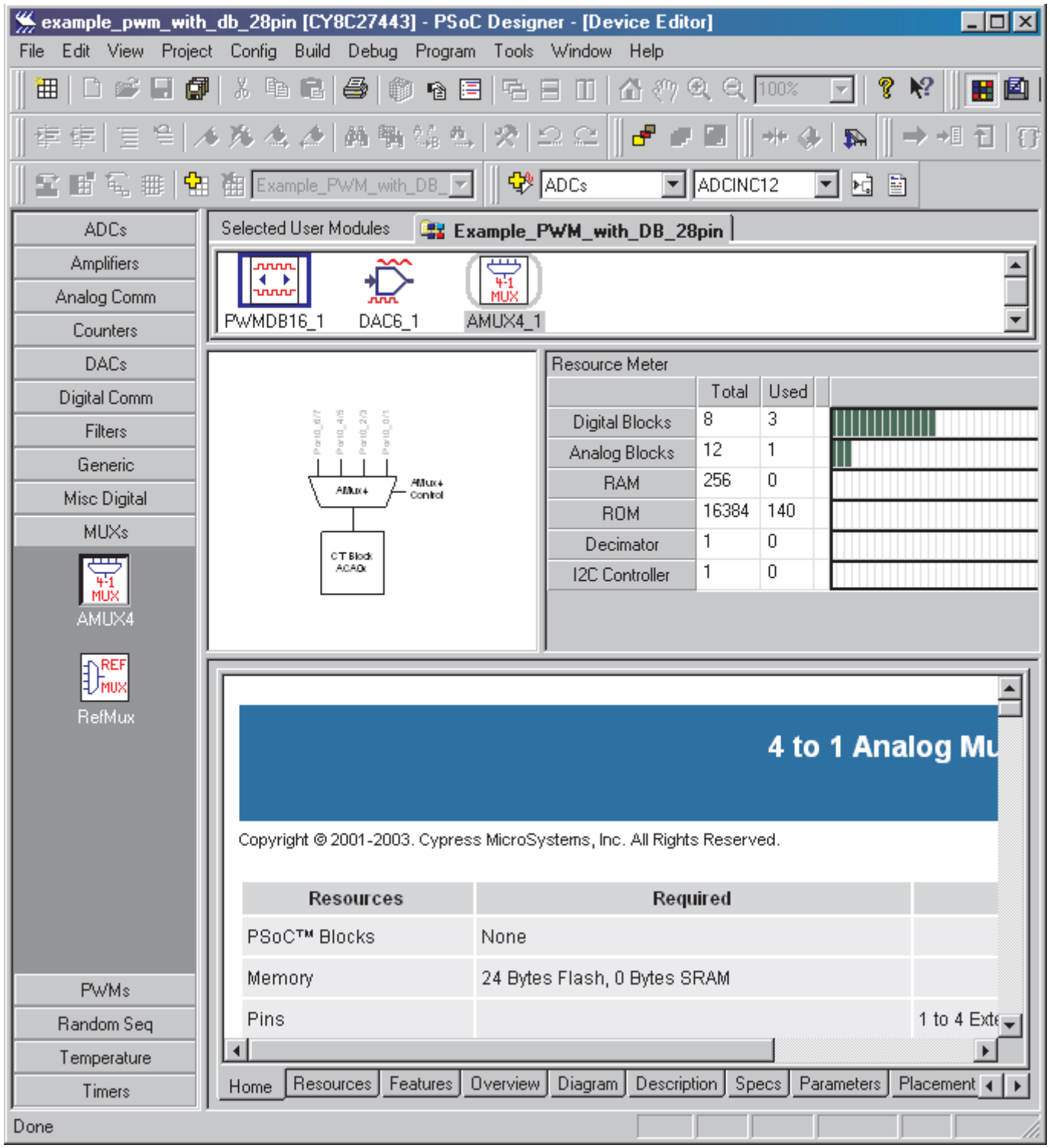


Figure 38: User Module Data

In the lower active window, click the tab options (Resources, Overview, Diagram, Features, etc.) to view additional information regarding a chosen User Module.

If you right-click your mouse anywhere inside the data sheet, you can view or print its .pdf as well as refresh the content and return to the top.

In the User Module Toolbar click the first drop-arrow to select a category and the second drop-arrow to select a module from within the specified category. Then, click the **User Module Block Diagram** icon or **User Module Data Sheet** icon to view floating windows of the diagram and data sheet, respectively. This option is available from within all three subsystems; Application Editor, Device Editor and Debugger.

The User Module toolbar can also be used in the Interconnect View.



Figure 39: User Module Toolbar

Once you have viewed and decided upon User Modules, you are ready to officially select them. To select a User Module, perform the following steps:

1. Double-click a User Module in the left frame (or right-click your mouse on it and choose Select). It will then appear in the upper active window (in User

Module Section View).

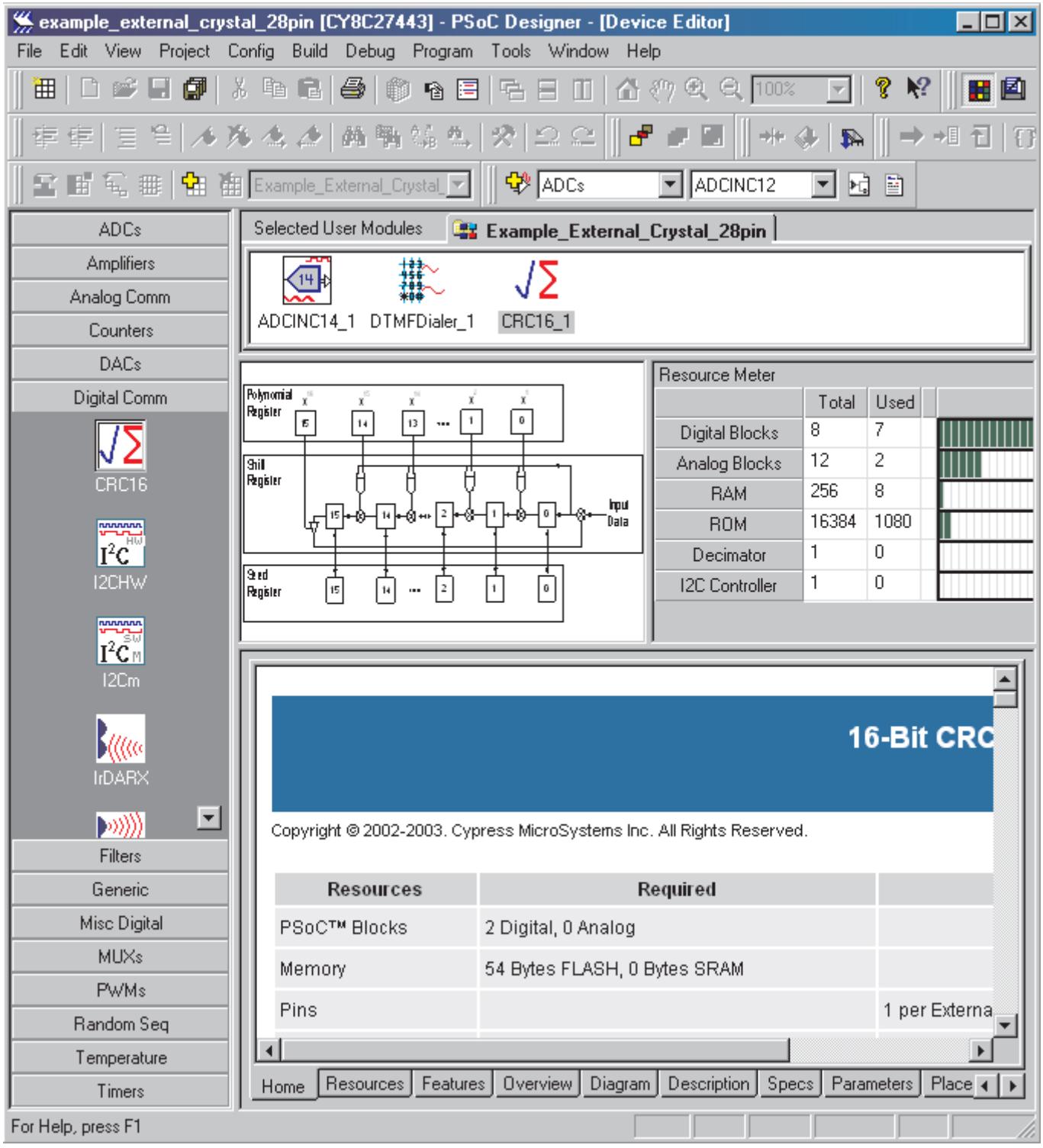


Figure 40: User Module Selections

2. Repeat the process for each individual User Module you wish to select.

If, at any time, you would like to change the User Module instance name, right-click on the module, select Rename, and type a new name. The User Module instance name identifies multiple User Modules of the same type in a single project.

For each User Module you add, the system updates the data in the Resource Manager window with the number of occupied PSoC blocks, along with RAM and ROM usage used by the current set of “selected” User Modules. If you select a User Module that requires more resources than are currently available, PSoC Designer will not allow the selection.

If User Modules are already placed, then there are some cases when User Module selection will fail even if it appears that sufficient PSoC blocks remain unallocated. In such cases, the “placed” User Modules are occupying resources that the “selected” User Module requires.

At any time during device configuration you can add and remove User Modules to and from your device.

To remove User Modules from your collection, click on the User Module you wish to remove and click the [**Delete**] key (or right-click the User Module itself and select Delete from the menu). This will not remove User Modules from PSoC Designer, just from your collection.

In addition to the User Module Toolbar, you can view a specific User Module diagram or data sheet by right-clicking on a “selected” User Module and making a selection.

5.3 Placing User Modules

Placing selected User Modules on PSoC blocks is the second step to configuring your target device. PSoC blocks, as defined in PSoC Designer, are the analog and digital peripheral blocks of a device that are customized by the placement and configuration of User Modules.

To place User Modules, click the **Interconnect View** icon in the Device Editor toolbar.



Figure 41: Interconnect View in Toolbar

In the left frame you will see User Module Parameters and Global Resources. In the upper window you will see your selected User Modules. In the main window you will see the device interface as well as device pin-out.

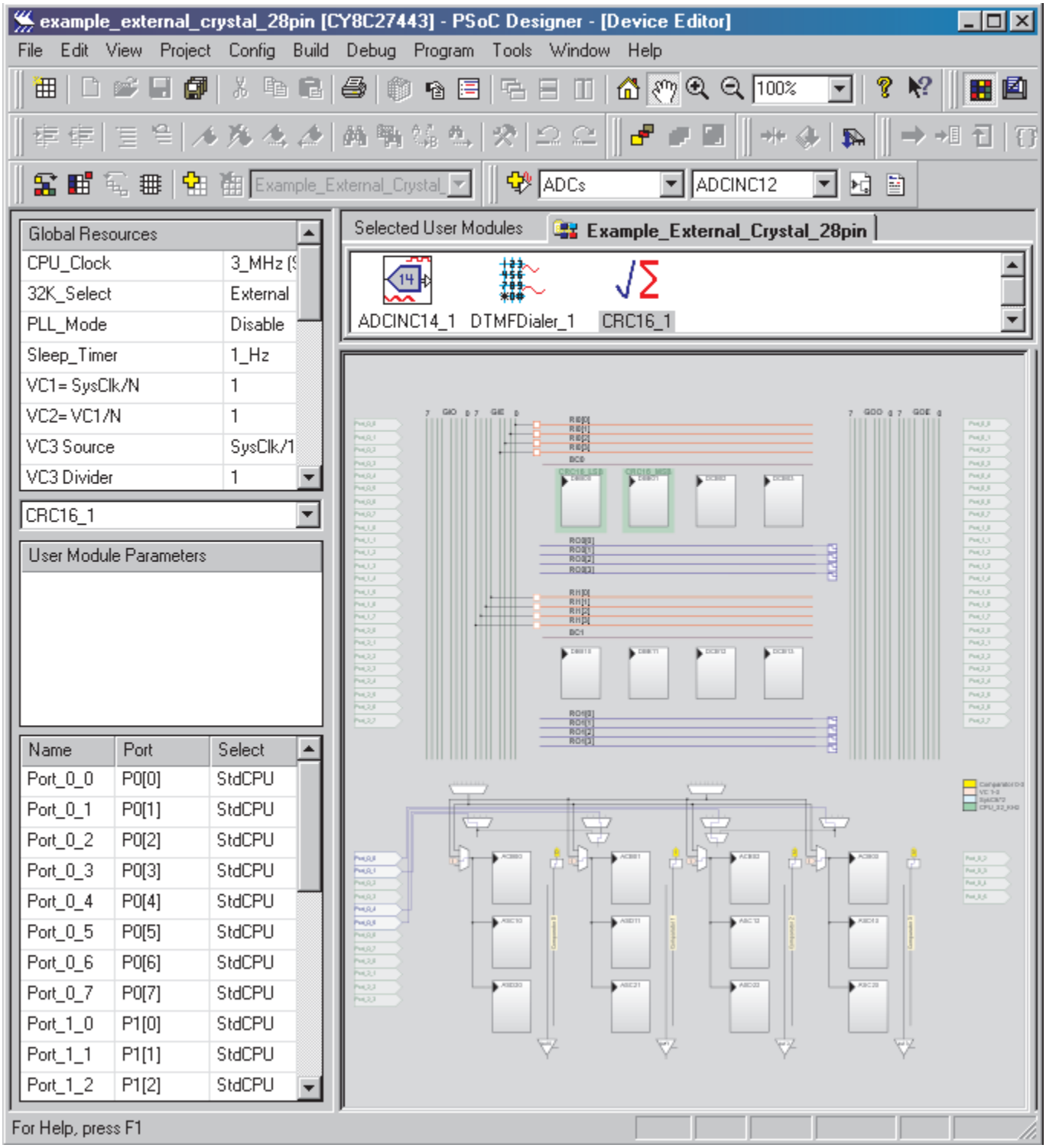


Figure 42: Selected (yet-to-be Placed) User Modules

5.3.1 Placing a User Module

To place a User Module perform the following steps:

1. Single-click on a selected User Module.

   When you click the module, the first available location on the device is highlighted in the device interface. If the User Module requires more than one group of PSoC blocks, then the groups will be highlighted in green (analog) or blue (digital). Also, the User Module reference name for the block appears above the blocks that are currently active (i.e., clicked on).

2. Click the **Next Allowed Placement** icon  to advance the highlights to the next available location (analog identified as green, digital as blue). Do this until you have identified the exact location for the User Module.

The Next Allowed Placement action shows the next possible set of PSoC blocks a User Module may be placed, regardless of any currently placed User Modules. If placement of the User Module is not possible in the highlighted location due to lack of resources, a “Resource Allocation” message will flash in the lower-left corner of PSoC Designer. Placement may not be possible if another User Module occupies the PSoC block, or if a placed User Module is using another resource which the highlighted User Module requires.

3. When you have identified the location, click the **Place User Module** icon , or right-click and select Place.

Once you have placed the module, it will appear on the device, color-coded, bearing the designated name on the chosen PSoC block.

To print or view your placements in the browser, right-click anywhere in the gray area of the device interface and select Print.

If you want to clear all User Module “placements” (i.e., remove them from their location on the PSoC blocks), click Config >> Clear All Placements. If you want to unplace one particular module, right-click on it and select Unplace or click the **Undo Place User Module** icon . This will not remove User Modules from PSoC Designer or from your collection, just placement.

If, at any time, you would like to name or rename User Modules, right-click on the module, select Rename, and type a new name.

Some modules are not “placed” on PSoC blocks (i.e., LCD, I^2C Master, I^2C Slave). Under User Module Parameters you designate a port, which then appears highlighted in the device interface with the User Module name.

4. Repeat this process for all selected User Modules.

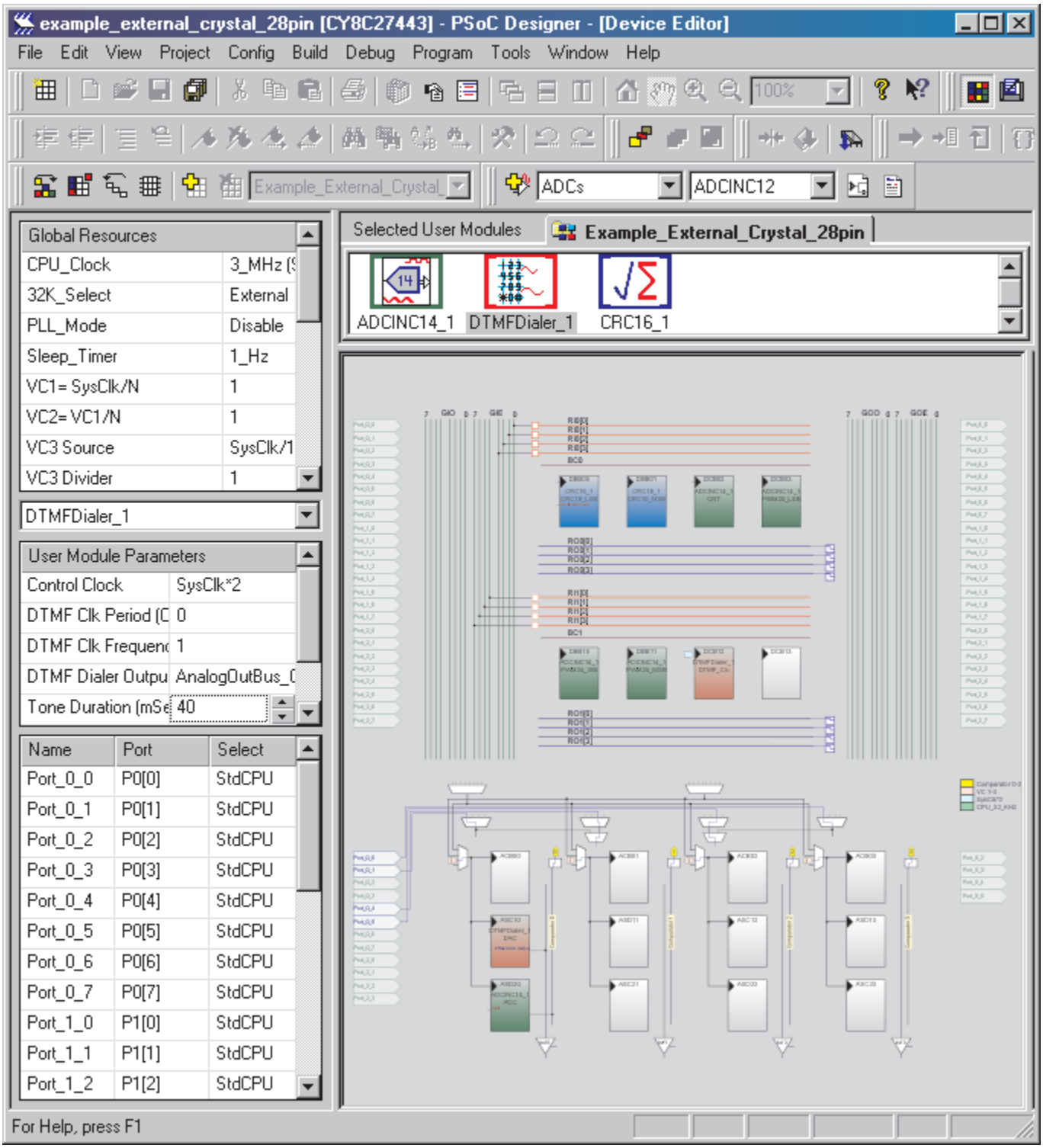


Figure 43: Placed User Modules

Additional options must be specified if you have placed a Band-Pass Filter (BPF2) and/or Low-Pass Filter (LPF2). Right-click on the filter module and select User Module Selection Options and then Filter Design Wizard. Proceed, filling in and making choices according to your application. You can also access these menu items by right-clicking the module on the PSoC block inside the device interface.

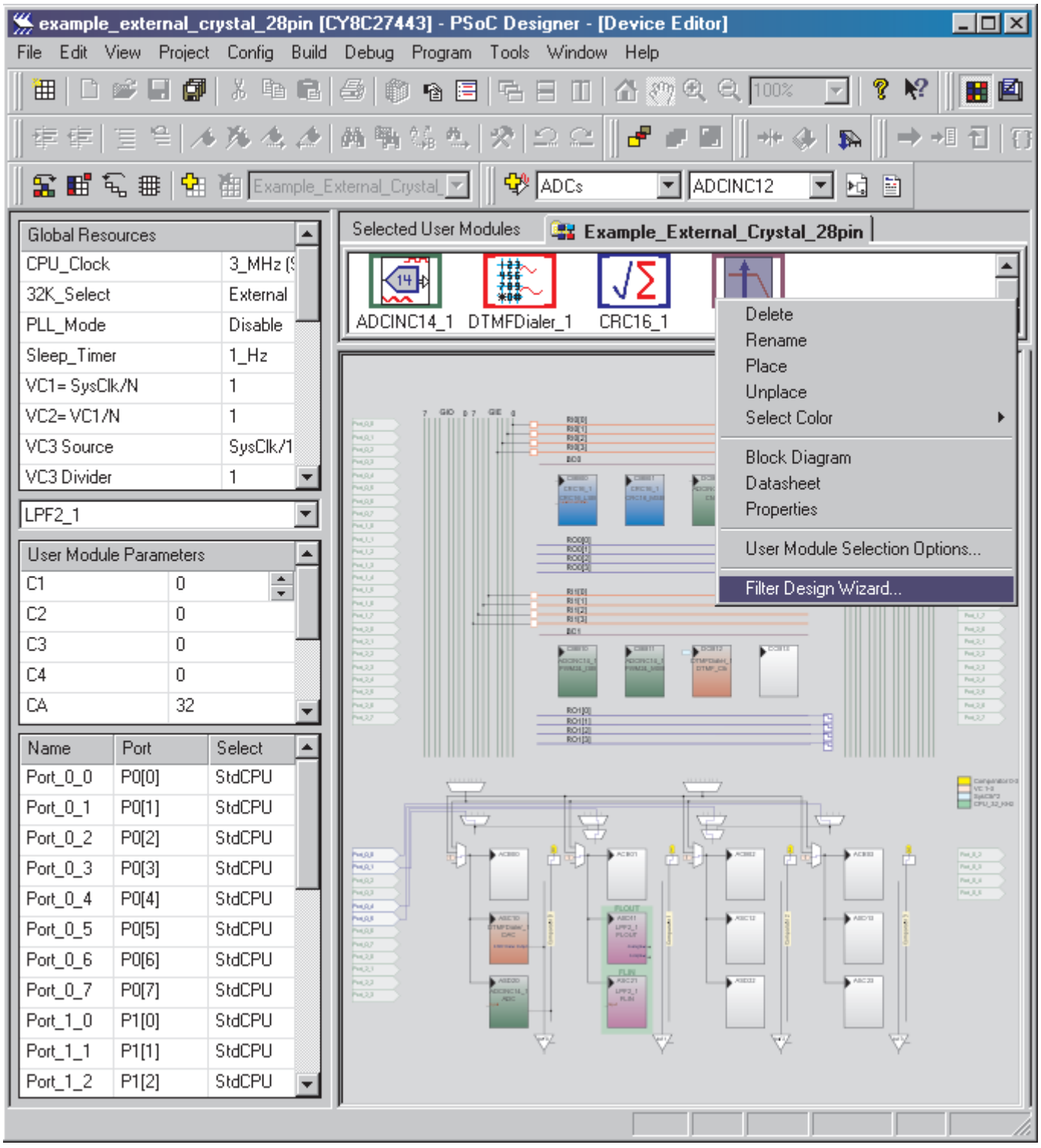


Figure 44: Filter User Module Options

If you add or remove User Modules after you have generated application files, you will need to re-generate the application files (as well as reconfigure required settings). For further details, see **Generating Application Files** later in this section.

The I^2CHW User Module will be “unplaced” if your project was created in PSoC Designer 4.0 *Beta* release. Right-click on the module icon and select User Module Selection Options. One of the topology selections must be chosen in order to place the I^2CHW User Module.

5.3.2 User Module Parameters

Setting User Module Parameters connects the User Module to the external pins and other User Modules. Connecting to User Modules is accomplished through the output and input parameters of the PSoC Blocks. The interconnection buses provide interconnection paths between the external pins and to other digital User Modules.

Connections are shown as lines between elements, special symbols, or flag connectors. The flag connectors are used when the connection is made to a point where drawing a line could result in a cluttered display, with the legend indicating the origin of the connection. Connections to pins are shown as lines from interconnection buses. The interconnection bus structure depends on the PSoC device selected and can consist of a single level of buses, or two levels of buses, between the digital PSoC blocks and the pins.

Connections between analog PSoC blocks and pins are accomplished through the analog input muxes and output buses. The muxes connect directly to pins and there is always only one level of output buses between analog PSoC blocks and output pins.

The pin names are duplicated in several places, since they are multifunction, and they are highlighted when used with lines showing their current connection state. The location of the pin to which a line is drawn indicates the usage of the pin. Lines drawn to the pins on the left edge indicate that the pins are used as inputs, while the right edge indicates usage as output. Pins in the upper groups indicate connection to the digital network, while lower groups indicate analog connections. When lines are drawn to multiple locations to the same pin, this indicates that the shown combination is electrically valid.

When you single-click a User Module you can view its parameters under User Module Parameters (Interconnect View).

To view all settings for a selected User Module, use the drop-down list in the lower-left corner of the left frame.

Once you place that User Module, the parameters will be updated with applicable module names. You can now make selections from the updated drop-down lists as to the configuration for the PSoC blocks associated with the User Module.

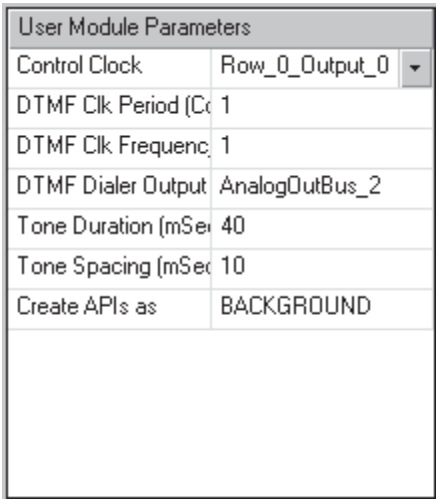


Figure 45: User Module Parameters

1. To update all User Module parameters click each drop-arrow (in parameter value fields) and make applicable selections.

Some parameters are specified integer values. You set these values, as the non-integer values, by clicking the up/down arrows. But for integer values, you can also double-click the value and type over. If you enter a value that is out of range, you will see a dialog box specifying the acceptable range. (Click **OK** to close the dialog box.)

2. Repeat this process for all placed User Modules.

5.3.3 Global Resources

Global Resources are hardware settings that determine the underlying operation of the part (for the entire application). Such settings include the CPU_Clock. For example, this setting designates the speed at which the M8C processes. High MHz equals fast processing and low MHz equals slower processing. High takes more power, low takes less power. Therefore, when you set the value, you must strike a balance between speed and power to optimize your implementation.

Note that Global Resource options differ slightly per device family (i.e., resource options for CY8C25xxx/26xxx parts are different than the options for CY8C27xxx parts).

1. To update global resources for the project, click each drop-arrow (in parameter value fields) and make applicable selections.

Similar to User Module Parameters, some parameters in Global Resources are specified integer values (such as 24V1 and 24V2). You set these values by clicking the up/down arrows or double-clicking the value and typing over. If you enter a value that is out of range, you will see a dialog box specifying the acceptable range. (Click **OK** to close the dialog box.)

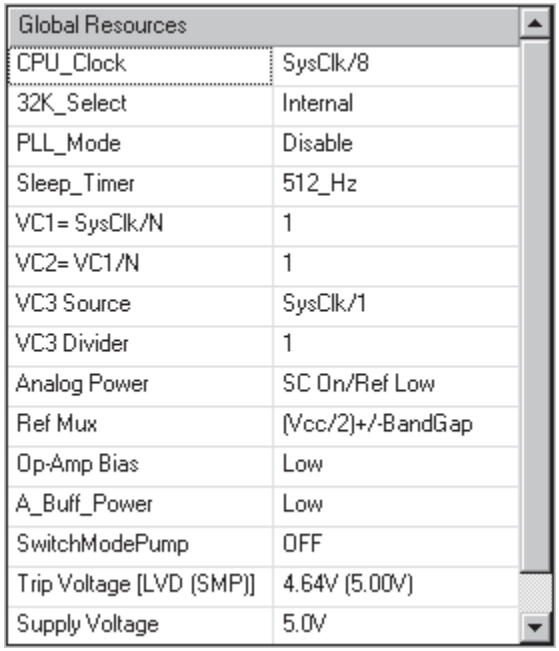


Figure 46: Global Resources

The current settings for the Global Parameters can be saved as default settings. This is done by choosing the menu item Config >> Global Resources >> Update Default Values. You can also right-click on any Global Resource name and select “Update Default Values.” This action saves all Global Resource settings to the `\Preferences` directory under the PSoC Designer installation path. These settings can then be used by any other PSoC Designer project by choosing the menu item Config >> Global Resources >> Restore Default Values, or by right-clicking on any Global Resource name and selecting “Restore Default Values.” If no custom default values are saved, then the menu item and the right-click to “Restore Default Settings” will restore the factory default Global Resource Settings.

5.4 Deploying Interconnectivity (Connecting User Modules)

Specifying interconnections between the User Modules on the PSoC blocks can be done as you place or after you place each User Module. Interconnec-

tivity between User Modules enables communication between PSoC blocks (which are the analog and digital peripheral blocks of a device that are customized by the placement and configuration of User Modules).

Connecting to User Modules is accomplished through the output and input parameters of the PSoC Blocks. The interconnection buses provide interconnection paths between the external pins and to other digital User Modules.

Connections are shown as lines between elements, special symbols, or flag connectors. The flag connectors are used when the connection is made to a point where drawing a line could result in a cluttered display, with the legend indicating the origin of the connection. Connections to pins are shown as lines from interconnection buses. The interconnection bus structure depends on the PSoC device selected and can consist of a single level of buses, or two levels of buses, between the digital PSoC blocks and the pins.

Connections between analog PSoC blocks and pins are accomplished through the analog input muxes and output buses. The muxes connect directly to pins and there is always only one level of output buses between analog PSoC blocks and output pins.

The pin names are duplicated in several places, since they are multifunction, and they are highlighted when used with lines showing their current connection state. The location of the pin to which a line is drawn indicates the usage of the pin. Lines drawn to the pins on the left edge indicate that the pins are used as inputs, while the right edge indicates usage as output. Pins in the upper groups indicate connection to the digital network, while lower groups indicate analog connections. When lines are drawn to multiple locations to the same pin, this indicates that the shown combination is electrically valid.

To specify interconnections, click the **Interconnect View** icon in the Device Editor toolbar.

User Module interconnections consist of connections to surrounding PSoC blocks, output bus, input bus, internal system clocks and references, external pins, and analog output buffers. Multiplexers may also be configured to route signals throughout the PSoC block architecture.

Digital PSoC blocks are connected through the Global_IN and Global_OUT buses to external pins and to other digital PSoC blocks. There are 8 Global_IN and 8 Global_OUT bus lines, numbered 0 through 7. For external pin connections, the number of the Global bus line corresponds to the bit number of the associated port. For example, Global_IN_0 can connect to pins associated with Port_0_0, Port_1_0, Port_2_0, etc. The Global_OUT buses can drive the inputs to other digital PSoC blocks. However, all Global_OUT lines do not reach all digital PSoC blocks. Refer to the CY8C25xxx/26xxx Device Family Data Sheet for details on the global bus interconnections.

When setting output parameters to the Global_OUT lines, only one PSoC block can drive a single Global_OUT line at a time. Global_OUT lines that are being used by a User Module are not available to other User Modules for output. For example, if two timer User Modules are placed and the first timer is set to use Global_OUT_1 for output, attempting to set the output for the second timer to Global_OUT_1 will fail.

To save current configurations in Device Editor, click File >> Save Project.

Global In

Global In connections apply to a PSoC device as follows:

- CY8C25xxx/26xxx as Global In: Input Port Connections.
- CY8C27xxx and beyond as Global In Odd and Global In Even: Input Port Connections and Global Connections.

To set Global In connections, execute the following:

1. Click on the target Globalxxx vertical line.
2. Click the global input to output connection (if active) then click the port.

You will see a line connecting the digital input port to the global vertical line.

Global Out

Global Out connections apply to a PSoC device as follows:

- CY8C25xxx/26xxx as Global Out: Output Port Connections.
- CY8C27xxx and beyond as Global Out Odd and Global Out Even: Output Port Connections and Global Connections.

To set Global Out connections, execute the following:

1. Click on the target Globalxxx vertical line.
2. Click the global input to output connection (if active) then click the port.

You will see a line connecting the digital output port to the global vertical line.

Analog Clock Select

To set Analog Clock Select connections, execute the following:

1. Click on the target AnalogClock_x_Select Mux.
2. Select a DBAxx or DBBxx PSoC block (as applies).

You will see a line from the right side of DBxxx to the input of the AnalogClock_x_Select Mux. The mux switch shows a connection to this input.

Analog Column Clock

To set Analog Column Clock connections, execute the following:

1. Click on the target AnalogColumn_Clock_x Mux.
2. Select a device-specific option from the menu.

You will see that the AnalogColumn_Clock_x Mux has a line connecting your chosen option to the mux output.

Analog Column Input Mux

To set Analog Column Input Mux connections, execute the following:

1. Click on the target AnalogColumn_InputMUX_x.
2. Select a port from the menu.

You will see a connection between the output of AnalogColumn_InputMUX_x and the analog input port.

Analog Column Input Select

To set Analog Column Input Select connections, execute the following:

1. Click on the target AnalogColumn_InputSelect_x.
2. Select appropriate AnalogColumn_InputMUX_x from the menu.

You will see that your chosen AnalogColumn_InputSelect_x has a line inside that connects the output of AnalogColumn_InputMUX_x to its output.

Analog Output Buffer

The Analog Output Buffers can be connected to the associated port pin or turned off. To set Analog Output Buffer connections, execute the following:

1. Click on the target AnalogOutBuf_x.
2. Select a port from the menu.

You will see a line that connects the AnalogOutBuf_x triangle to the analog output port.

Selection of Clock Input for a Digital Block

Note that the name “Clock Input” is determined by a specified User-Module parameter.

To set clock input connections on a digital block, execute the following:

1. Click the clock input triangle on the digital block where your target User Module has been placed.

The clock input triangle is not active for all blocks when a User Module uses more than one block.

2. Select an option from the menu.

You will see your chosen input option displayed next to the clock input triangle. Your choice option will also appear in the Control Clock field under User Module Parameters (where you can click the drop-arrow to change your selection).

Selection of Enable Input for a Digital Block

Note that the name “Enable Input” is determined by a specified User-Module parameter.

To set the Enable Input connection on a digital block, execute the following:

1. Click the Enable text label on the digital block where your target User Module has been placed.

2. Select an option from the menu.

You will see your chosen input option displayed next to the Enable text label. Your choice option will also appear in the Enable field under User Module Parameters (where you can click the drop-arrow to change your selection).

Selection of Output for a Digital Block

Note that the name “Output” is determined by a specified User-Module parameter.

To set output connections on a digital block, execute the following:

1. Click the Output text label on the on the digital block where your target User Module has been placed.

2. Select an option from the menu (None, Global_OUT_x for CY8C25xxx/26xxx, or Row_x_Output_x for CY8C27xxx and beyond).

You will see your chosen option displayed (with a connection) next to the Output text label. Your choice option will also appear in the Output field under User Module Parameters (where you can click the drop-arrow to change your selection).

Selection of RBotMux for a CT Analog Block

Note that the name “RBotMux” is determined by a specified User-Module parameter.

To select an RBotMux for a CT analog block, execute the following:

1. Click the RBotMux text label on the analog block where your target User Module has been placed.

2. Select an option from the menu.

You will see your chosen option displayed next to the RBotMux text label. Your choice option will also appear in the RBotMux field under User Module Parameters (where you can click the drop-arrow to change your selection).

You can execute this same procedure when the NMux, PMux, AnalogBus or CompBus CT Analog Block apply, as well as for ACMux, BMux, AnalogBus or CompBus *SC* Analog Blocks.

Row Broadcast

Row Broadcast connections only apply to CY8C27xxx parts and beyond.

1. Click the Row_0_Broadcast (BC0) or Row_1_Broadcast (BC1) horizontal line.

2. Select an option from the menu.

You will see a line connecting to a digital PSoC block or to the other Row Broadcast, depending on the option you chose.

Comparator Analog LUT

Comparator Analog LUT connections only apply to CY8C27xxx parts and beyond.

1. Click the AnalogLUT_x box. (Its symbol is identified in the Comparator x line along each column of analog PSoC blocks.)

2. Select an option from the menu.

You will see connections on the device interface reflecting your A or B selection with associated symbol.

5.4.1 Digital Interconnect Row Input Window

Digital Interconnect Row Input Window connections only apply to CY8C27xxx parts and beyond.

Connection to Global Input

To set a Connection to Global Input, execute the following:

1. Click on the white box or the line of the target Row_x_Input_x. (A tool tip will appear to identify your selection.)

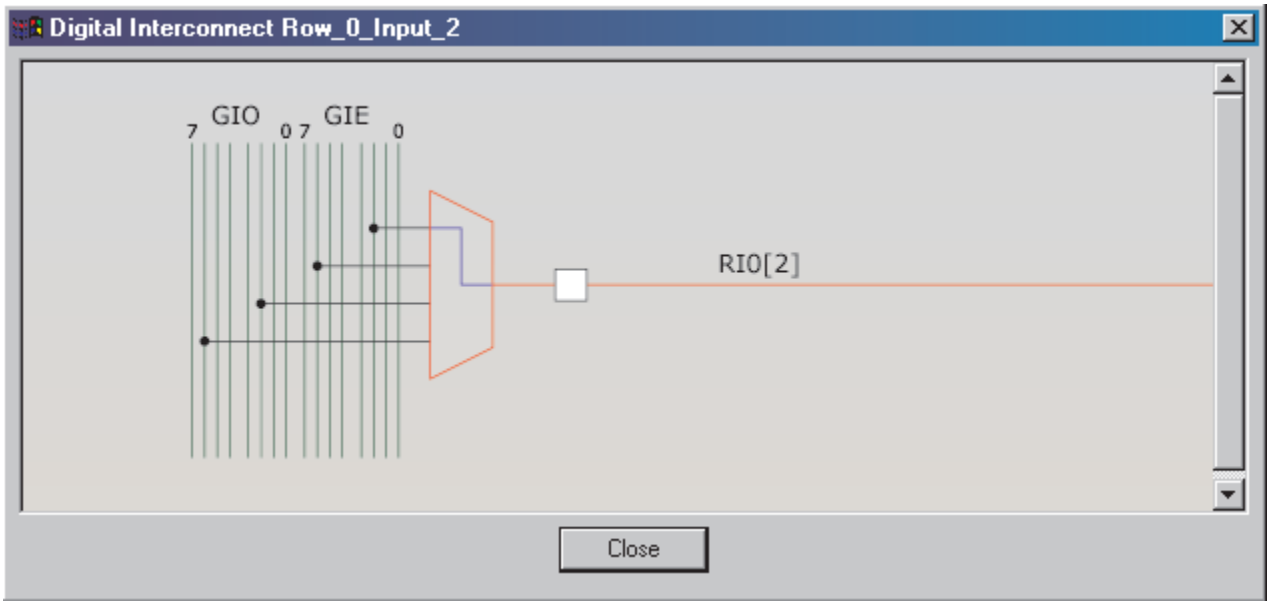


Figure 47: Digital Interconnect Row Input

2. Click on the Row_x_Input_x Mux in the Digital Interconnect Row Input floating window and select a Global Input from the menu. (You will immediately see a connection from the mux to the Global Input vertical line.)

The Digital Interconnect Row Input floating window provides further detail on the device to advance comprehension and better decisions.

3. Click the **Close** button when finished.

In this floating window you can also click the white box to toggle the Synchronization value for Row_x_Input_x. Options include SysClk_Sync and Async.

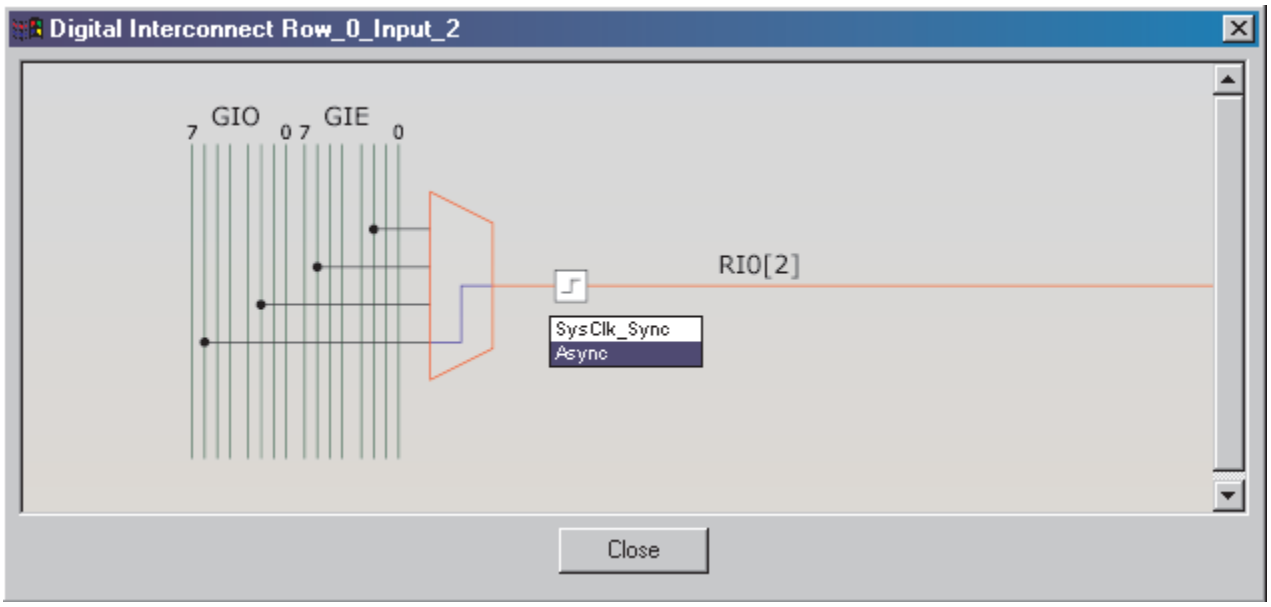


Figure 48: Synchronization Options for Digital Interconnect Row Inputs

5.4.2 Digital Interconnect Row Output Window

Digital Interconnect Row Output Window connections only apply to CY8C27xxx parts and beyond.

Row Logic Table Input

To set Row Logic Table Input connections, execute the following:

1. Click on the target Row_x_Output_x Logic Table Box.

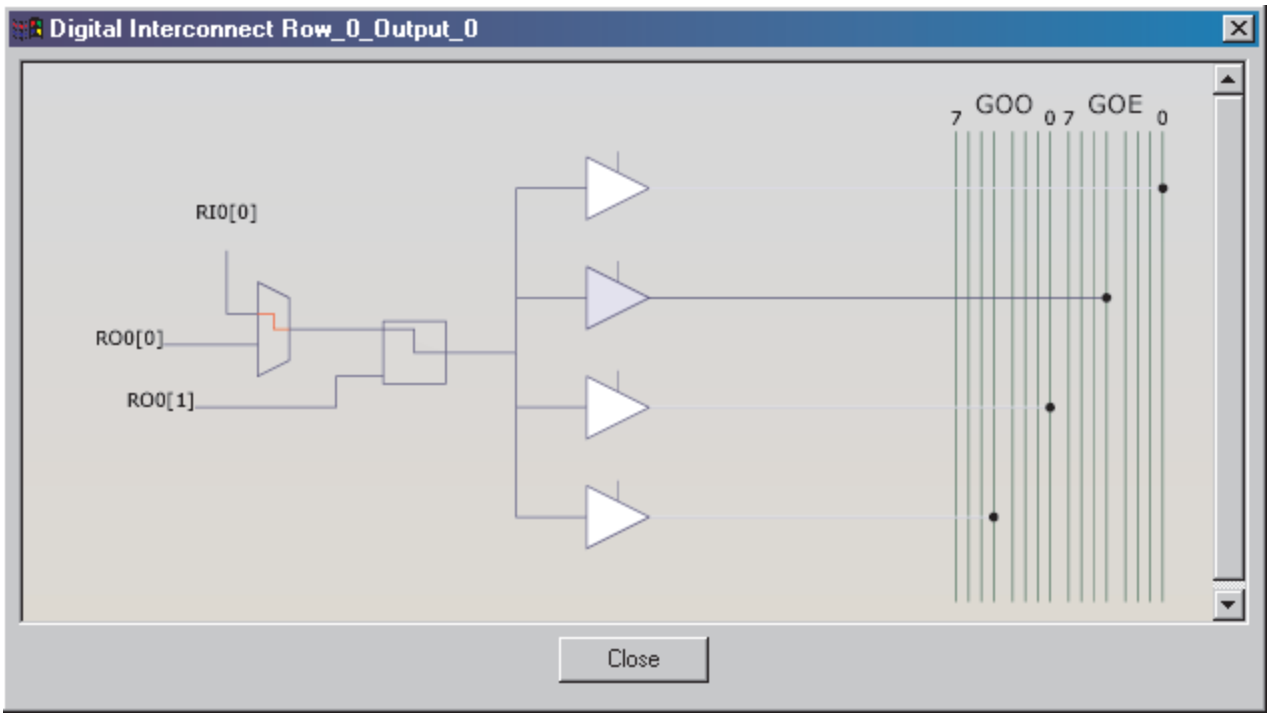


Figure 49: Digital Interconnect Row Output

2. Click on the Row_x_LogicTable_Input_0 Mux in the Digital Interconnect Row Output floating window and select an input or output option from the menu.

3. Click the **Close** button when finished. (You will see connections on the device interface reflecting your row input or output selection.)

Row Logic Table Select

To set Row Logic Table Select connections, execute the following:

1. Click on the target Row_x_Output_x Logic Table Box.

2. Click on the Row_x_LogicTable_Select_x logical operation box in the Digital Interconnect Row Output floating window and select an option from the menu.

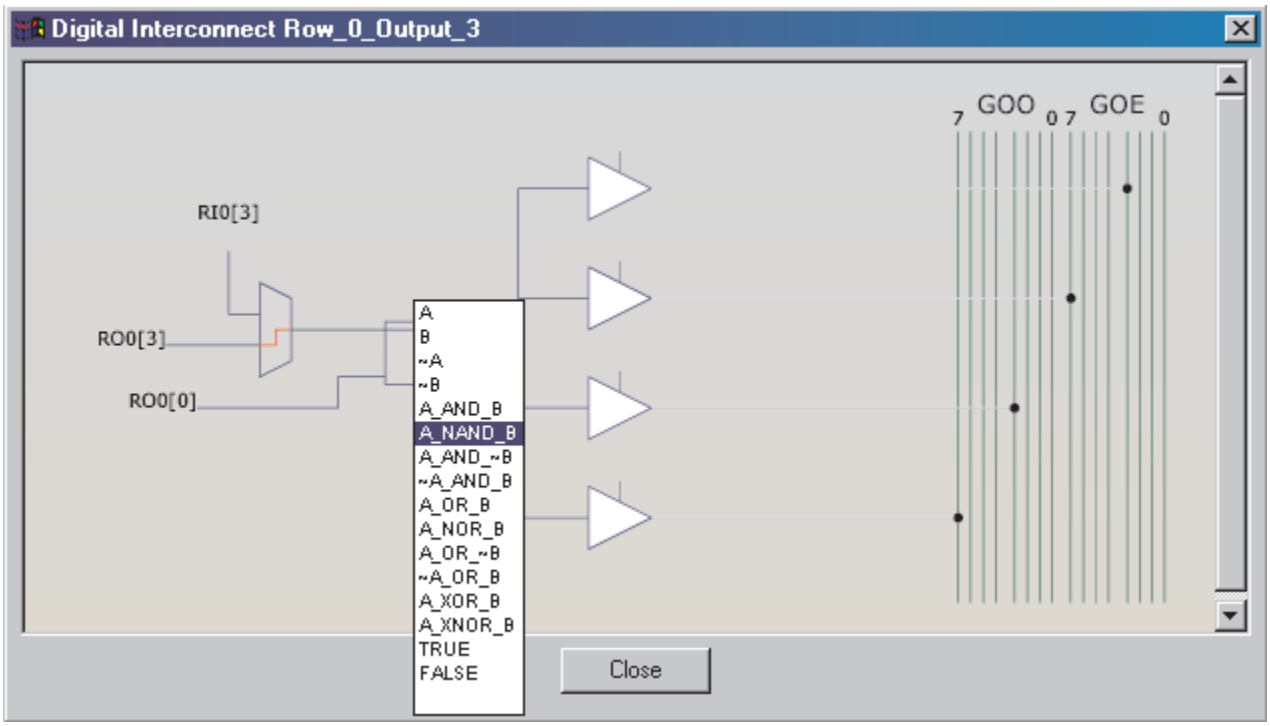


Figure 50: Logical Operations in Digital Interconnect Row Output

3. Click the **Close** button when finished. (You will see connections on the device interface reflecting your A or B input selection with associated symbol.)

Connection to Global Output

To set Connections to Global Output, execute the following:

1. Click on the target Row_x_Output_x Logic Table Box.

2. Click on the target Row_x_Output_x_Drive_x triangle in the Digital Interconnect Row Output floating window and select an option from the menu.

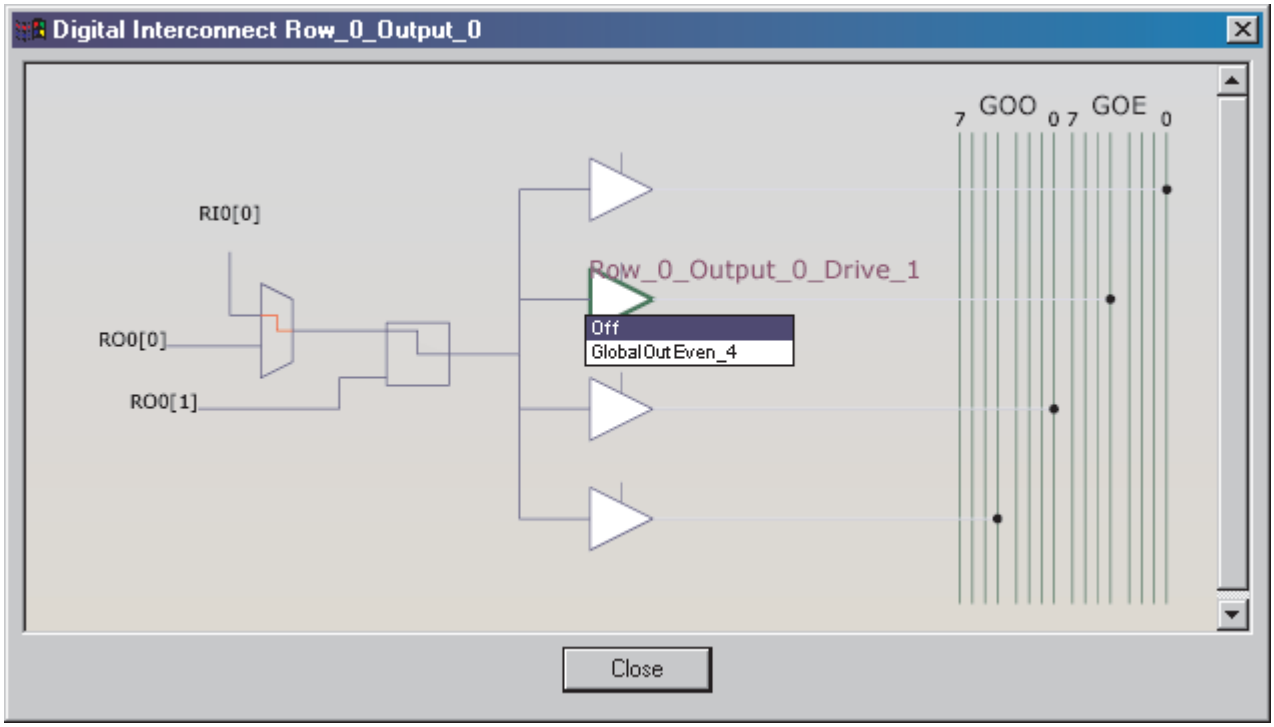


Figure 51: Digital Interconnect Row Global Output

3. Click the **Close** button when finished. (You will see a connection from the Row_x_Output_x Logic Table Box to the chosen GlobalOutEven_x vertical line.)

Once you open the Digital Interconnect Row Output Window, you can select Row Logic Table Input, Row Logic Table Select, and Connections to Global Output without closing the window.

5.5 Specifying Pin-out

Specifying the pin-out for each PSoC block is the next step to configuring your target device. When you specify a PSoC block to a pin-out you are making a physical connection between the software configuration and the hardware.

To access the device pin-out, click the **Interconnect View** icon in the Device Editor toolbar.



Figure 52: Interconnect View in Device Editor Toolbar

In the middle frame you will see the pin-out bearing the number of pins of your target device. Your specific device was chosen when you specified a base part during the creation of your project.

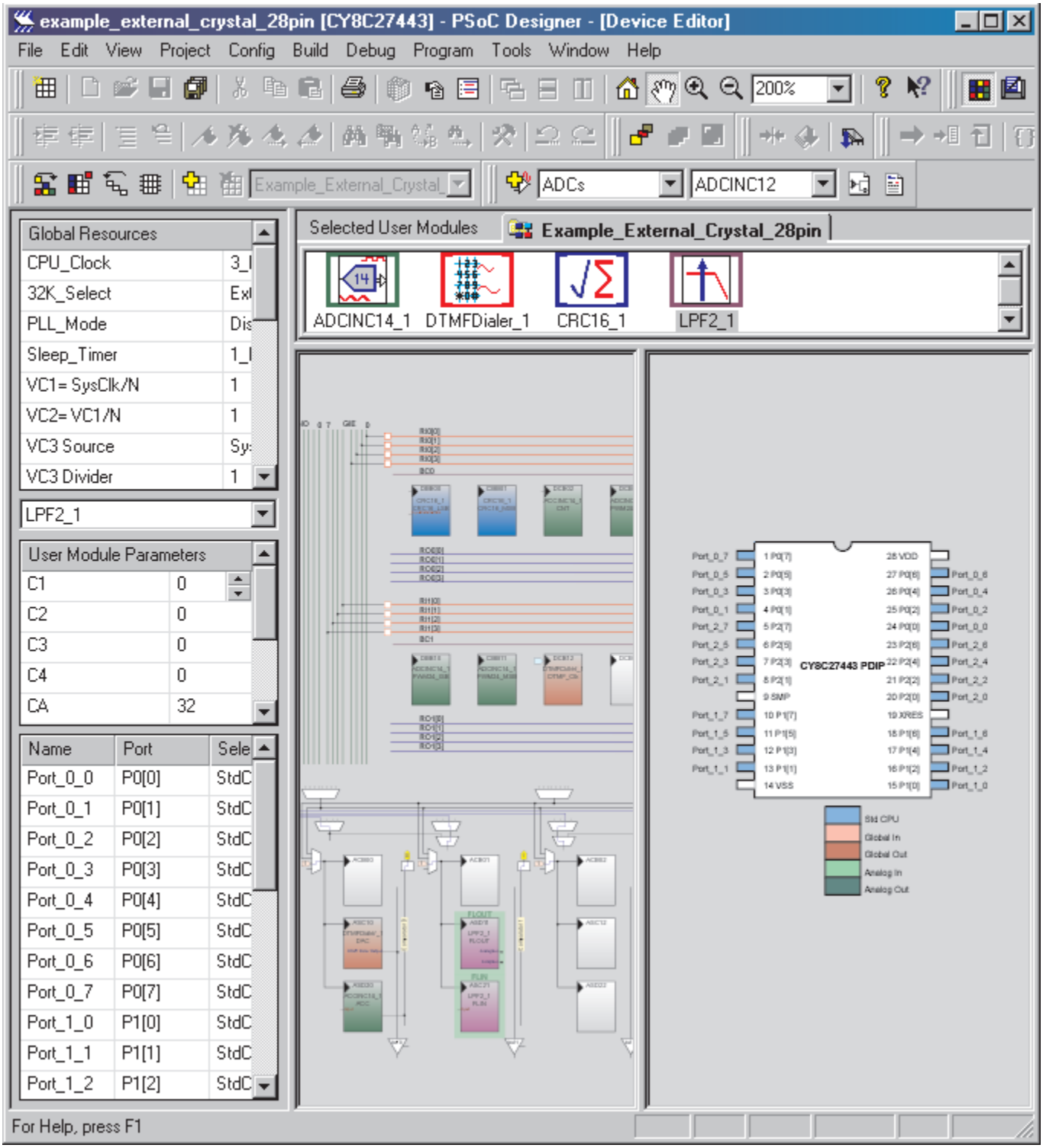


Figure 53: Device Pin-Out

Care should be taken when connecting to pins. The pin settings can be modified either by setting elements to connect to pins, or by setting the pin directly.

Setting the pin directly connects the pin to the appropriate element and disconnects from any other element. If multiple connections to the same pin are desired, connections must be made from the element to the pin. For example, suppose a connection to a pin, an analog input mux and an analog output buffer, simultaneously, is desired. Port_0_2 can connect to the analog input mux for column 1 and to the analog output buffer for column 3. The connections must be made from the analog input mux and the analog output buffer.

Setting the pin to "Default" disconnects the pin from both digital buses, but does not affect analog connections.

Another difference in pin settings between PSoC Designer v. 3.21 and v. 4.0 is that 3.21 coupled digital output bus connections with a drive level so that the setting also set a drive level. For example, Port_0_0 had settings that included "Global_OUT_0 (Strong)," "Global_OUT_0 (Pull Down)," and "Global_OUT_0 (Pull Up)." In 4.0, the setting has been simplified to a single setting with the drive level set independently. Port_0_0 has a single setting "Global_OUT_0" and the drive level is set independently. Projects created in earlier versions of PSoC Designer are automatically adjusted to this new convention without changing the drive setting as saved in the project.

5.5.1 Port Connection

Port connections can be done in one location, three ways; using the port icons in the device interface, device pin-out, or port-related fields all inside Interconnect View.

Analog Input

To set Analog Input connections, execute the following:

1. Click on the target Port_0_x.
2. Select the port from the menu.
3. Select Analog Input from the second menu.

On the device you will see the new designation color coded according to the legend along side the device. The port name and selection will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

You can execute the same procedure for Analog Output Buffer only select AnalogOutBuf_3 for step number 3.

Default Input

To set Default Input connections, execute the following:

1. Click on the target Port_x_x.
2. Select the port from the menu.
3. Select Default from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name, the Select column value of StdCPU, and the drive mode of High Z Analog will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections).

Global_IN_x

Global_IN_x connections apply to a PSoC device as follows:

- CY8C25xxx/26xxx as Global_IN_x.
- CY8C27xxx and beyond as GlobalIn[Odd/Even]_x.

To set Global_IN_x connections, execute the following:

1. Click on the target Port_x_x.
2. Select the port from the menu.
3. Select the device-specific Global IN option from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name, the Select column value of your chosen option, and the drive mode of High Z will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections).

You will also see a line between the digital input port and the Global IN vertical line.

Global_OUT_x

Global_OUT_x connections apply to a PSoC device as follows:

- CY8C25xxx/26xxx as Global_OUT_x.
- CY8C27xxx and beyond as GlobalOUT[Odd/Even]_x.

To set Global_OUT_x connections, execute the following:

1. Click on the target Port_x_x.

2. Select the port from the menu.

3. Select the device-specific Global OUT option from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name, the Select column value of your chosen option, and the drive mode of Strong will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections).

You will also see a line between the Global OUT vertical line and the digital output port.

StdCPU

To set StdCPU connections, execute the following:

1. Click on the target Port_x_x.

2. Select the port from the menu.

3. Select StdCPU from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name and StdCPU will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

XtalOut

To set the XtalOut connection, execute the following:

1. Click on Port_1_0 (P1[0]).

2. Select Port_1_0 from the menu.

3. Select XtalOut from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name, XtalOut, and the drive mode of High Z will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

XtalIn

To set the XtalIn connection, execute the following:

1. Click on Port_1_1 (P1[1]).

2. Select Port_1_1 from the menu.

3. Select XtalIn from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name, XtalIn, and the drive mode of High Z will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

ExternalGND

To set the ExternalGND connection, execute the following:

1. Click on Port_2_4 (P2[4]).

2. Select Port_2_4 from the menu.

3. Select ExternalGND from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name and ExternalGND will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

In the device interface you will see that all lines from Port_2_4 have disappeared.

Ext Ref

To set the Ext Ref connection, execute the following:

1. Click on Port_2_6 (P2[6]).

2. Select Port_2_6 from the menu.

3. Select Ext Ref from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name and Ext Ref will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

In the device interface you will see that all lines from Port_2_6 have disappeared.

I2C SDA

To set the I2C SDA connection, execute the following (this connection is only available for CY8C27xxx parts):

1. Click on Port_1_5 (P1[5]).
2. Select Port_1_5 from the menu.
3. Select I2C SDA from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name, I2C SDA, and the drive mode of Open Drain High will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

In the device interface you will see that all lines from Port_1_5 have disappeared.

I2C SCL

To set the I2C SCL connection, execute the following (this connection is only available for CY8C27xxx parts):

1. Click on Port_1_7 (P1[7]).
2. Select Port_1_7 from the menu.
3. Select I2C SCL from the second menu.

On the device you will see the designation color coded according to the legend along side the device. The port name, I2C SCL, and the drive mode of Open Drain High will also appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrow to change your selection).

In the device interface you will see that all lines from Port_1_7 have disappeared.

If, at any time, you wish to restore default pin-out, click the Restore Default Pin-out icon .

5.5.2 Port Drive

Port drive modes can be specified in one location, three ways; using the port icons in the device interface, the port icons on the device pin-out or in the Drive field of Interconnect View.

Port drive modes apply to a PSoC device as follows:

- CY8C25xxx/26xxx options include High Z, Pull Down, Pull Up, and Strong.
- CY8C27xxx and beyond options include High Z, High Z Analog, Open Drain High, Open Drain Low, Pull Down, Pull Up, Strong, and Strong Slow.

To specify a port drive mode, execute the following:

1. Click on the target Port_x_x.
2. Select the target Port_x_x_Drive from the menu.
3. Select the port drive mode option from the second menu.

The port name and the drive mode of your choice will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections)

5.5.3 Port Interrupt

ChangeFromRead

To specify ChangeFromRead interrupt, execute the following:

1. Click on the target Port_x_x.
2. Select the target Port_x_x_Interrupt from the menu.
3. Select ChangeFromRead from the second menu.

The port name and ChangeFromRead will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections)

DisableInt

To disable interrupts, execute the following:

1. Click on the target Port_x_x.
2. Select the target Port_x_x_Interrupt from the menu.
3. Select DisableInt from the second menu.

The port name and DisableInt will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections)

FallingEdge

To specify FallingEdge interrupt, execute the following:

1. Click on the target Port_x_x.
2. Select the target Port_x_x_Interrupt from the menu.
3. Select FallingEdge from the second menu.

The port name and FallingEdge will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections)

RisingEdge

To specify RisingEdge interrupt, execute the following:

1. Click on the target Port_x_x.
2. Select the target Port_x_x_Interrupt from the menu.
3. Select RisingEdge from the second menu.

The port name and RisingEdge will appear in the port-related fields underneath User Module Parameters (where you can click the drop-arrows to change your selections)

5.6 Tracking Device Space

Tracking the available space and memory of configurations for your device is something you do intermittently during the whole process of configuring your target device. Device space and memory resources need to be monitored so you are aware, on an ongoing basis, of the capacity and limitations you are working with on the MCU.

You can monitor device space and memory from within the User Module Selection View of Device Editor. Resources are updated as each User Module is selected (before actual placement).

In the far-right frame of the Select User Module mode of the subsystem, you see a table to track Analog Blocks, Digital Blocks, RAM, and ROM. As you place User Modules, you can view how many analog and digital PSoC blocks you have available and how many you have used. You will also notice a live graph tracking the PSoC blocks you have used by percentage.

RAM and ROM monitors track the amount RAM and ROM required to employ each selected User Module.

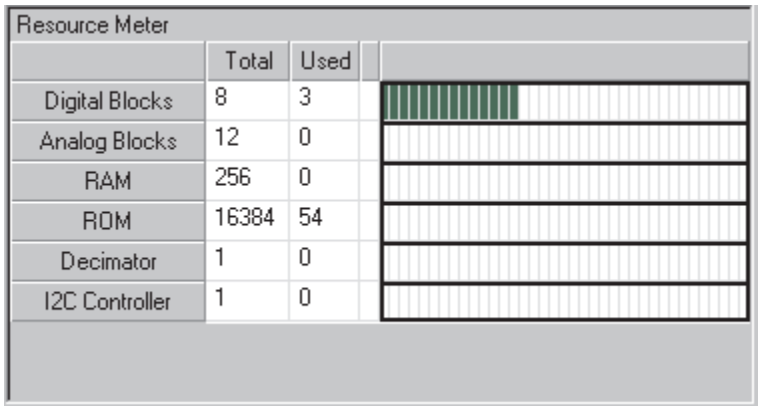


Figure 54: PSoC Block Resource Meter

5.7 Design Rule Checker

Design Rule Checker (DRC) operates on a collection of pre-determined rules associated with elements in a project database. Once invoked, the DRC runs and then communicates the results of a “rule” evaluation.

The DRC is designed to point out potential errors or “rule” violations in your project that might eventually pose problems. Design Rule Checker will not impose limitations or prevent you from proceeding with your project “as is.” It simply gives you a “heads up” on PSoC User Module, software and hardware elements of which you may or may not have been aware when configuring and sourcing your device. Bottomline, it is an additional tool to provide support for user-configuration consistency via an error-proof project.

Sample rules include:

- PLL and no External Crystal
- 24 MHz and 3V Operation
- 48 MHz and 3V for Digital Clock Operation
- Un-set Parameters/Connections
- P0[1] and P0[0] Pins not High-Z with External Crystal
- 3.3V Indicating ICE is 5V Supply Only
- Global Bus with Signal Pulse Width < 1/12 MHz
- Phase Consistency Between Output to Input of SCblocks

- PWM/Counter/Timer with Pulse Width > Period
- Inappropriate Ground and Reference Level Selections

This list is not comprehensive but merely a sample. The PSoC Designer collection of rules is being updated and added to on an ongoing basis.

5.7.1 Running Design Rule Checker

Execute the following to run DRC:

1. Go to Tools >> Design Rule Checker or click the DRC icon.
2. In a matter of seconds, you can review the results of the rule evaluation in the Build tab of the Output Status window.

You can specify specifics regarding level of rule checking and result detail under Tools >> Options >> Design Rule Checker tab.

Note that DRC can be run at any time or any number of times during project development. To designate it to run automatically each time you generate application files, go to Tools >> Options >> Device Editor tab.

5.8 Generating Application Files

Generating application files is the final step to configuring your target device. When you generate application files, PSoC Designer takes all device configurations and updates existing assembly-source and C compiler code and generates API and ISR shells. Read ahead in this section for details regarding APIs and ISRs. At this time, the system also creates a data sheet based on your part configurations that can be accessed in the Device Editor.

You can view the data sheet by clicking View >> Datasheet. The configuration data sheet is self contained in its own folder in the project directory and can be viewed independently of PSoC Designer by opening *configreport.xml* in Internet Explorer. (If you need to move or send someone the file, you must move/send the entire directory of `\ConfigDataSheet`.)

Once this process is complete, you can enter Application Editor and begin programming the desired functionality into your (now configured) device. For further details regarding programming, see Section 7. and Section 8.

You can generate application files from within any of the three Device Editor modes; Select User Module, Place User Module, or Specify Pin-out.

To generate application files, click the **Generate Configuration** icon . This process is transparent to you and takes less than a minute.

After the process runs and completes, you can check the official status in the Output Status window.

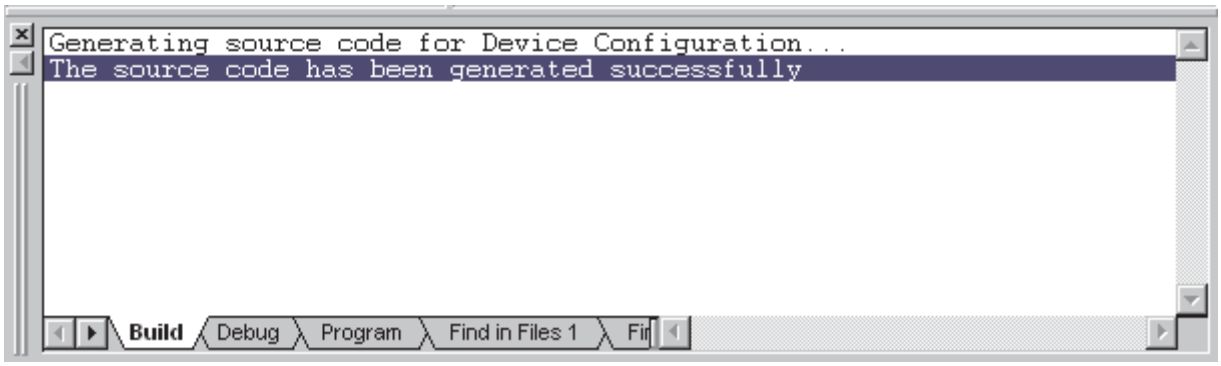


Figure 55: Output Status During Application Generation

Once application source files have been successfully generated, click the sub-system you would like to enter; Device Editor, Application Editor, or Debugger.

It is important to note that if you modify any device configurations, you must re-generate the application files before you resume source programming.

5.8.1 Source Files Generated by Generate Application Files

Table 7: Source Files Generated by Generate Application

Name	Overwritten	Description
…/lib/PSoCCOnfig.asm	Yes	Configuration loaded upon system access
…/lib/PSoCCOnfigTBL.asm	Yes	Contains chip configuration
Boot.asm	Yes	Boot code and initial Interrupt Table
…/lib/<User module name>.asm	Yes	User Module API source
…/lib/<User module name>.h	Yes	User Module API ‘C’ include
…/lib/<User module name>.inc	Yes	User Module API assembly include
…/lib/<User module name>INT.asm	No	User Module interrupt .asm file (if needed)
…/lib/<Project name>API.h	Yes	Project API include
…/lib/<Project Name>_GlobalParameters.inc	Yes	Project parameters include file
main.asm	No	Main code

If you undo placement of a User Module but leave it in your selected collection and generate application files, associated .asm files will remain (just not be updated). If you undo placement and delete a User Module from your collection then generate application files, all associated .asm files will be deleted (removed from source tree project files).

5.8.2 APIs and ISRs

APIs (Application Programming Interfaces) and ISRs (Interrupt Service Routines) are also generated during the device configuration process in the form of *INT.asm, .h, and .inc files. *These files provide the device-interface and interrupt-activity framework for source programming*. See the following example of an .h file for configurations of a 16-bit PWM (Pulse Width Modulator) created during application-code generation:

pwm16_1.h

```
#include <m8c.h>

/* Create pragmas to support proper argument and return value passing */
#pragma fastcall  PWM16_1_EnableInt
#pragma fastcall  PWM16_1_DisableInt
#pragma fastcall  PWM16_1_Start
#pragma fastcall  PWM16_1_Stop
#pragma fastcall  PWM16_1_WritePeriod
#pragma fastcall  PWM16_1_WritePulseWidth
#pragma fastcall wPWM16_1_ReadPulseWidth
#pragma fastcall wPWM16_1_ReadCounter

/***************************************************
* Prototypes of PWM API. For a definition of
* functions see PWM16_1.inc.
***************************************************/
extern void  PWM16_1_EnableInt(void);
extern void  PWM16_1_DisableInt(void);
extern void  PWM16_1_Start(void);
extern void  PWM16_1_Stop(void);
extern void  PWM16_1_WritePeriod(WORD wPeriod);
extern void  PWM16_1_WritePulseWidth(WORD wPulseWidth);
extern WORD wPWM16_1_ReadPulseWidth(void);
extern WORD wPWM16_1_ReadCounter(void);

// end of file
```

Figure 56: PWM16_1.h

Once you have generated your device configuration application code, the files for APIs and ISRs can be found in the source tree of Application Editor under the Library Source and Library Header folders.

This user-transparent action occurs each time device application code is generated. If you modify any ISR file and then re-generate your application, changes **will not** be overwritten if they are placed between user code markers included in the **int.asm* file. Source code outside of the user code marker regions is overwritten and is always re-generated. However, if a User Module is renamed and the application is re-generated, any user modifications within the user code markers are not updated with the instance name. Any use of the User Module instance name within user code markers must be manually updated.

5.8.3 Working with ISRs

The CPU has 16 interrupts: 4 fixed function and 12 configurable. The fixed function interrupts are:

- Reset
- Supply Monitor
- GPIO
- Sleep Timer

The configurable PSoC block interrupts include eight (8) digital blocks and four (4) analog column blocks. The definition (e.g. interrupt vector action) of a configurable PSoC block interrupt depends on the User Module that occupies that block.

The Device Editor handles the details of getting User Module parameters into source code so that the project will be configured correctly upon startup and expose subroutines that make for ease-of-use. Exposing subroutines that make User Module parameters easy to use involves PSoC Designer adding files to your project. These files are known as Application Program Interfaces (APIs). Typically, one of these User Module files added to your project is an interrupt handler.

Aside from adding API files to your project, the Device Editor also inserts a call or jump to the User Module's interrupt handler in the startup source file, *boot.asm*.

5.8.4 Interrupt Vectors and the Device Editor

Following is an example of how an interrupt handler is dispatched in the startup code, using a device from the CY8C27xxx part family. Shown below is

Timer32 User Module mapped to PSoC blocks 00, 01, 02, and 03. An interrupt is generated by the hardware when terminal count is reached. The last PSoC Block (or MSB byte) of Timer32 generates the terminal count interrupt.

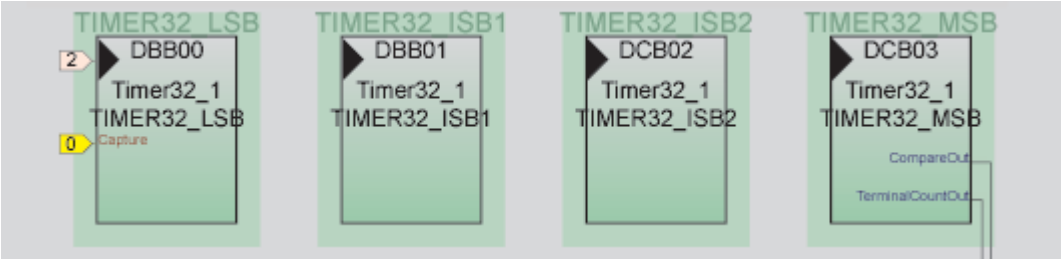


Figure 57: Timer32 on Four Digital PSoC Blocks

Upon device application generation ( ) code is produced for the Timer32_1 User Module in *boot.asm*. The startup code is also altered with the addition of the call to the timer interrupt handler.

```
org 0                ; Reset Interrupt Vector
jmp __start          ; First instruction executed following Reset
org 04h              ; Supply Monitor Interrupt Vector
   // call     void_handler
reti
org 08h              ; PSoC Block DBB00 Interrupt Vector
   // call     void_handler
reti
org 0Ch              ; PSoC Block DBB01 Interrupt Vector
   // call     void_handler
reti
org 10h              ; PSoC Block DCB02 Interrupt Vector
   // call     void_handler
reti
org 14h              ; PSoC Block DCB03 Interrupt Vector
ljmp     Timer32_1INT
reti
org 18h              ; PSoC Block DBB10 Interrupt Vector
   // call     void_handler
reti
org 1Ch              ; PSoC Block DBB11 Interrupt Vector
   // call     void_handler
reti
org 20h              ; PSoC Block DCB12 Interrupt Vector
   // call     void_handler
reti
org 24h              ; PSoC Block DCB13 Interrupt Vector
   // call     void_handler
reti
org 28h              ; Analog Column 0 Interrupt Vector
   // call     void_handler
reti
org 2Ch              ; Analog Column 1 Interrupt Vector
```

```
   // call     void_handler
reti
org 30h              ; Analog Column 2 Interrupt Vector
   // call     void_handler
reti
org 34h              ; Analog Column 3 Interrupt Vector
   // call     void_handler
reti
org 38h              ; GPIO Interrupt Vector
   // call     void_handler
reti
```

The following table shows how *boot.asm* vector names map to fixed and PSoC block (configurable) interrupts:

Table 8: *boot.asm* Interrupt Names

boot.asm Interrupt Name	Data Sheet Interrupt Name	Type
__start	Reset	Fixed
Interrupt1	Supply Monitor	Fixed
Interrupt2	DBB00	PSoC Block
Interrupt3	DBB01	PSoC Block
Interrupt4	DCB02	PSoC Block
Interrupt5	DCB03	PSoC Block
Interrupt6	DBB10	PSoC Block
Interrupt7	DBB11	PSoC Block
Interrupt8	DCB12	PSoC Block
Interrupt9	DCB13	PSoC Block
Interrupt10	Analog Column 0	PSoC Block
Interrupt11	Analog Column 1	PSoC Block
Interrupt12	Analog Column 2	PSoC Block
Interrupt13	Analog Column 3	PSoC Block
Interrupt14	GPIO	Fixed
Interrupt15	Sleep Timer	Fixed

From our example, Interrupt5 corresponds to DCB03. There are no interrupt handlers at DBB00, DBB01, and DCB02 (Interrupt2, Interrupt3, and Interrupt4) because a 32-bit Timer User Module only requires the interrupt at the end of the chain.

In many cases the actual interrupt handling code is “stubbed” out. You can modify the content of this stubbed handler to suit your needs. Any subsequent device re-configuration will not overwrite your work in the handler.

5.8.5 A Word About boot.asm

When device configuration application files are generated, *boot.asm* is updated. Among other things, this file includes a jump table for interrupt handlers. (Additional details regarding this file are up ahead in section 7.1.)

The entries in the interrupt table are handled automatically for interrupts employed by User Modules. For example, a Timer8 User Module uses an interrupt. The interrupt-vector number depends on which PSoC block is assigned to the Timer8 instance; vector 2 for PSoC digital block 0, vector 3 for block 1, etc.

During the device configuration process, the ISR name is added to the appropriate interrupt-vector number. The interrupt handler is included in a file that is named *instance_nameint.asm*, where *instance_name* is the name given to the User Module. For example, if the User Module is named Timer8_1, then the ISR source file is named *Timer8_1INT.asm*. All API files generated during the device configuration process follow this naming convention. Following are the API files that would be generated for a User Module named Timer8_1:

- *Timer8_1.inc*
- *Timer8_1.h*
- *Timer8_1.asm*
- *Timer8_1INT.asm*

The *boot.asm* file is based on a file named *boot.tpl*. You can make changes to *boot.tpl* and those changes will be reflected in *boot.asm* whenever the application is generated. Do not change any strings with the form `` `@INTERRUPT_nn` where nn = 0 to 15 ``. These substitution strings are used when device configuration application files are generated. However, you can replace substitution strings if you safely define the interrupt vector and install your own handler.

If you install an interrupt handler and make changes directly to *boot.asm*, the changes will not be preserved if application generation is executed after the changes are made. If you make changes to *boot.asm* that you do not want overwritten, hard code the change in *boot.tpl* (template for *boot.asm*).

If there is no interrupt handler for a particular interrupt vector, the comment string `"// call void_handler"` is inserted in place of substitution string.

5.8.6 Configuration Data Sheet

Once you have configured your device and generated application files, you can produce, view, or print a data sheet based on how you configured your project device.

1. To produce and view a data sheet, click the View Data Sheet icon or access View >> Data Sheet from the menu. This opens an independent browser window that shows the current data sheet. (This can be done in any of the three modes of the PSoC Designer application.)

If changes were made in the Device Editor and a “Generate Application” was not performed, then PSoC Designer will show a dialog box asking for verification before performing Generate Application. If you select “No,” then the data sheet will reflect the configuration before the changes were made.

2. To print the data sheet click the standard Print icon or File >> Print.

The configuration data sheet is self contained in its own folder in the project directory and can be viewed independently of PSoC Designer by opening *configreport.xml* in Internet Explorer. (If you need to move or send someone the file, you must move/send the entire directory of `\ConfigDataSheet`.)

Section 6. Dynamic Re-configuration

Dynamic Re-configuration allows for PSoC applications to dynamically load and unload configurations. With this feature, your single PSoC device can have multiple functions.

If a PSoC project does not use multiple configurations, PSoC Designer and its usages behave as in previous version releases.

Upon installing and launching the upgrade version of PSoC Designer that contains Dynamic Re-configuration capabilities, by default you can select the example project that has multiple configurations.

In the Start dialog box, click anyone of the subsystems (preferably Device Editor) and `C:\Program Files\Cypress MicroSystems\PSoC Designer\Examples\Example_Dynamic_PWM\Example_Dynamic_PWM.SOC` will open. Using the example project you can easily sample features of Dynamic Re-configuration.

6.1 Add Configuration

To add loadable configurations to your PSoC project, execute the following steps:

This assumes you have PSoC Designer, Device Editor, and your target project open.

1. From the menu, click Config >> Loadable Configuration >> New. (Or click the left-most icon in the Dynamic Re-configuration toolbar.)

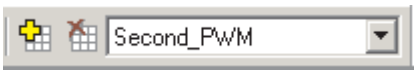


Figure 58: Dynamic Re-configuration Toolbar

The name of the new configuration is Config1 (and each additional configuration will take on consecutive numbering, i.e., Config2, Config3, Config4…).

2. Upon the addition of a new configuration you will see a new tab directly below the Dynamic Re-configuration toolbar as well as the drop-arrow selection in the Dynamic Re-configuration toolbar, both bearing the name Config1. Left-click to move between tabs (project configurations). Note that whatever tab you are on dictates the project configuration, regardless of the view. All views are the settings or configuration for the project configuration of the current tab.

There will always be at least one tab with the project name when a project is created. This tab represents the base configuration and has special characteristics. The base configuration cannot be deleted but can be exported. The new configuration will, by default, have global settings and pin settings identical to the base configuration.

   To change the name (to be configuration or project specific), double-click (or right-click) the tab and type the new name. The new name will appear on the tab and in the drop-arrow selection in the Dynamic Re-configuration toolbar. This is how you select your working configuration.

You will be in and can work in the corresponding configuration (User Modules, User Module Parameters, Global Resources…) of the configuration named in the tab (and drop-arrow selection). That is the currently “loaded” working configuration.

3. Proceed, as usual, with the configuration process, i.e., selecting and placing User Modules, setting up parameters, and specifying pin-out.

One strict requirement for Dynamic Re-configuration is that User-Module instance names must be unique across all configurations. This requirement eliminates confusion in code generation. Otherwise, all other icon and menu-item functions are identical to projects that do not employ additional configurations.

Additional configuration tabs will appear in alphabetical order from left to right (beginning after the base configuration tab).

6.2 Export and Import Designs

Exporting and importing designs can be done using the Design Browser feature in PSoC Designer. A design is a single (or collection of) existing, loadable configuration(s) from a project. A loadable configuration consists of one or more “placed” User Modules with module parameters, Global Resources, set pin-outs, and generated application files. PSoC projects can consist of one or multiple loadable configurations.

Exporting and importing designs allows you to efficiently use and re-use tested and proven configurations, thus saving design time and resources.

You can also create new projects based on exported designs. See 4.2.3 for details.

6.2.1 Export Design

To export a design, perform the following steps (this assumes you have PSoC Designer, Device Editor, and your target project open):

1. From the menu, click Config >> Export Design.

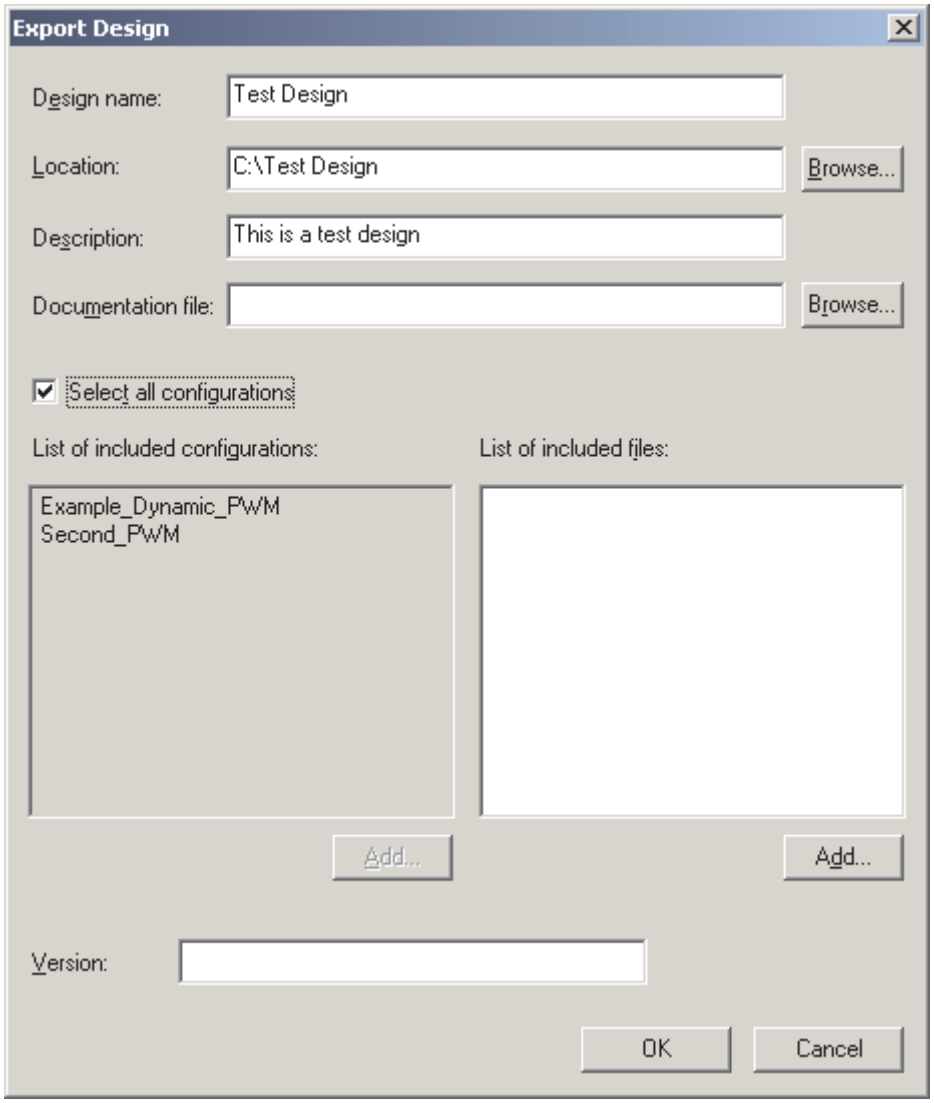


Figure 59: Export Design Dialog Box

2. In the Design name field of the Export Design dialog box, type a name. In the Location field below, a .cfg file with your chosen name will be created and added to the current project directory.

3. In the Description field, type a description of the design configuration.

4. In the Documentation file field, click **Browse** to identify an informational, design-related file. This field is not required.

5. In the List of included configurations field, click **Add**, select configurations, and click **OK**.

Click a check in the Select all configurations field if you wish to export the entire project (i.e., selecting all configurations in the project). You can also select all configurations by holding the [**Shift**] key and dragging your mouse down or you can select randomly by holding the [**Ctrl**] key and clicking selections.

6. In the List of included files field, click **Add**. Browse to identify all configuration-related files. Again, use the [**Shift**] and [**Ctrl**] methods to choose multiple files.

If you select a file that bears the same name as the file in your current project, you will be prompted to modify.

7. In the Version field, type the version of this design configuration. This is for your own internal tracking.

8. Click **OK**.

Your exported design (.cfg) can now be imported (added) to a new or existing project via Design Browser. See 6.2.2 for details.

6.2.2 Design Browser (Import Design)

In order to import an existing design (.cfg file), you must have previously exported the design you currently want to import (i.e., generated the .cfg). See 6.2.1 for details.

Again, designs brought into your project with Design Browser can have dynamic re-configurations or a single configuration.

To import a design to your project, execute the following steps (this assumes you have PSoC Designer, Device Editor, and your target project open):

1. From the menu, click Config >> Design Browser.

   PSoC Designer launches Design Browser.

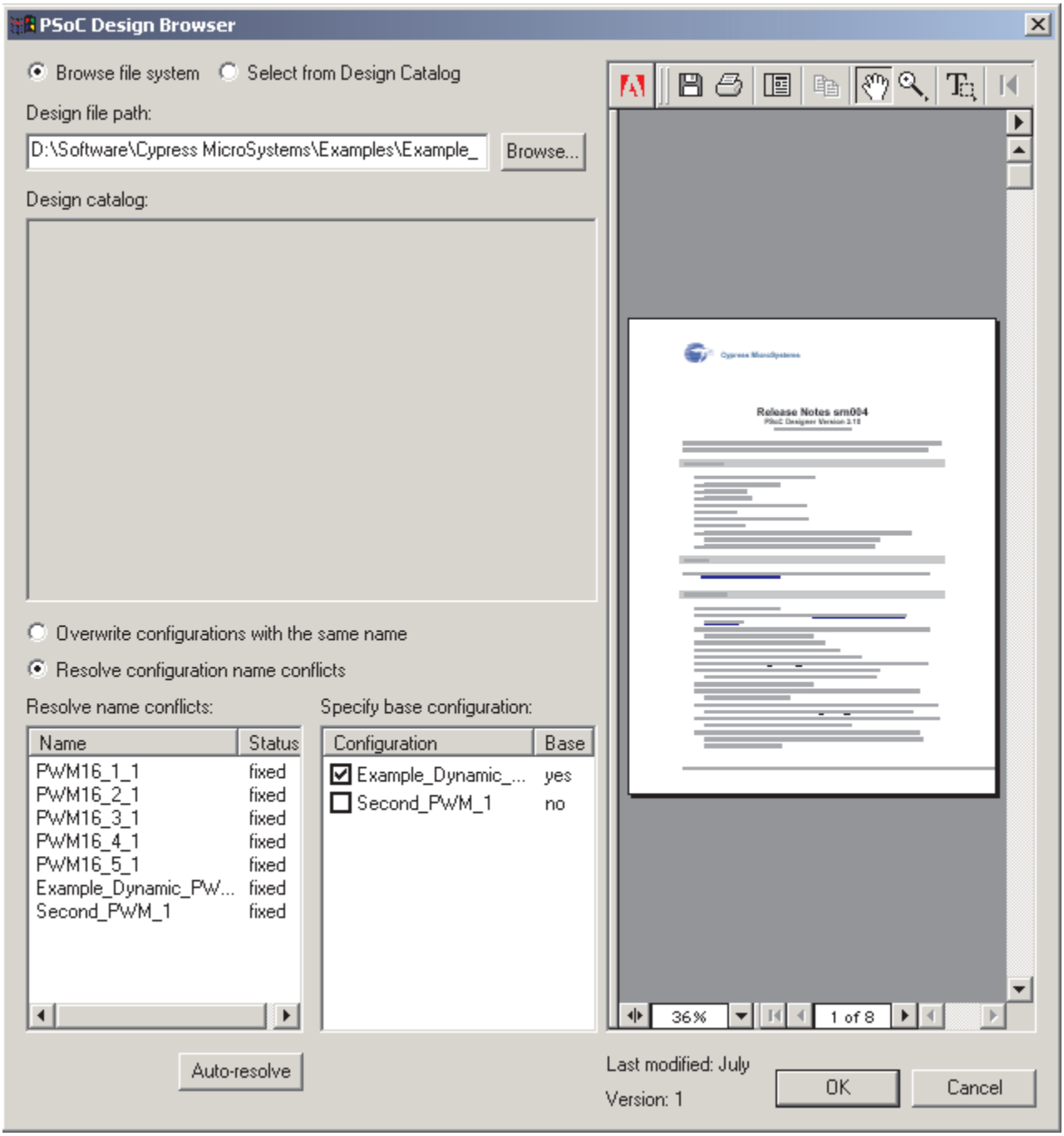


Figure 60: Design Browser

2. Click Browse file system or Select from Design Catalog.

   Browse file system: In the Design file path field of Design Browser, type

the file path or click **Browse** to locate the .cfg for the design you wish to import (add to your open project).

Select from Design Catalog: Design Browser displays all designs located in the Design Catalog folder in the PSoC Designer installation directory. Here, navigation is similar to navigation of the source tree. Click once on the target design.

Design Catalog contains sample designs provided to you upon installation of PSoC Designer.

Once you have made a selection, in the right frame you will see your configuration-related file corresponding to the chosen .cfg. You can view this file using the standard features of the application it launches.

3. Click whether you want to “overwrite” or “resolve” configuration name conflicts. Note that you must resolve all User Module name conflicts.

   If you choose to “resolve,” you can either double-click each name in the Name conflicts field and type a new name or click the **Auto-resolve** button. Auto-resolve instantly adds an “_1” to each name. Once you have resolved the conflict, the Status field will show the name status as “fixed.”

If you choose not to resolve, PSoC Designer will proceed. Be aware, there may be overlapping User Module placement issues.

4. In the Included configurations field, check which loadable configuration you want to designate as the base configuration for a dynamically re-configurable project. If you are not employing multiple configurations, simply check the chosen configuration. If you already have a base configuration in your target project, you should not check any of these boxes.

5. When finished, click **Import**/**OK**. Click **Close** to close Design Browser.

Once your design has been imported (added), its loadable configurations will be active in your current project.

6.3 Delete Configuration

To delete a loadable configuration from your PSoC project, execute the following steps:

This assumes you have PSoC Designer, Device Editor, and your target project open.

1. From the menu, click Config >> Loadable Configuration >> Delete, click the icon immediately to the left of the drop-arrow selection in the Dynamic Re-configuration toolbar, *or* right-click the tab of the configuration you wish to delete and select Delete.

2. Once you have selected to delete the configuration, you will be asked to confirm. Click **Yes**. (If you wish to cancel “delete” click **No**.)

Once you delete a configuration, the associated source files will be removed from the project (if application files had been generated).

6.3.1 Rename Configuration

To rename a loadable configuration in your PSoC project, execute the following steps:

This assumes you have PSoC Designer, Device Editor, and the project in which you want to rename a loadable configuration open.

1. From the menu, click Config >> Loadable Configuration >> Rename (or double-click the target configuration tab).

2. Inside the configuration tab, type the new name.

3. Hit [**Enter**] or click your cursor somewhere outside the tab.

6.4 Global Parameters and Dynamic Re-configuration

When employing Dynamic Re-configuration, global parameters are set in the same manner as single configurations. However, changes to base configuration global parameters are propagated to all additional configurations. Therefore, global parameter changes made to an additional configuration are to be done locally to that particular configuration. The code generation (Application Generation icon) considers global-parameter changes made to different configurations, but these changes should be made cautiously to prevent unexpected results.

6.5 Pin Settings and Dynamic Re-configuration

When employing Dynamic Re-configuration, port pin settings are similar to global parameters in that all settings in the base configuration are propagated to additional configurations. When manually set, port pin settings become local to the configuration.

6.5.1 Port Pin Interrupts

To set port pin interrupts, execute the following steps:

1. Access Interconnect View of Device Editor.

2. Click the drop-arrow or tab that corresponds to the configuration view for which you wish to set port pin interrupts.

3. The pin interrupt can be set in two places; in the Pin Parameter Grid under the Interrupt column, or through the pop-up menu that appears when a pin in the pin-out diagram is clicked.

   In the Pin Parameter Grid, access the drop-down list by clicking the drop-arrow in the Interrupt column and highlighting your selection.

   In the pop-up menu on the diagram, the interrupt setting appears in the first list along with the select and drive options. Clicking the Port Pin Interrupt option brings up the same drop-arrow selection as in the Pin Parameter Grid. Double-click your choice.

The default pin interrupt setting is “Disable.” If all pin interrupts are set to “Disable,” there is no additional code generated for the pin interrupts. If at least one pin is set to a value other than “Disable,” code generation performs some additional operations. In *boot.asm*, the vector table is modified so that the GPIO interrupt vector has an entry with the name ProjectName_GPIO_ISR. Following are additional files that are generated:

- *PSoCGPIOINT.asm*
- *PSoCGPIOINT.inc*

PSoCGPIOINT.asm contains an export and a placeholder so you can enter its pin interrupt handling code. Enter user code between the user code markers where appropriate. This file is re-generated for each code generation cycle, but the user code will be carried forward if it is within the user code markers.

When opening an old project that contains a *PSoCGPIOINT.asm* file where user code is entered, the user code must be copied from the backup copy in the `\Backup` folder into the newly generated *PSoCGPIOINT.asm file*.

PSoCGPIOINT.inc contains equates that are useful in writing the pin interrupt handling code. For each pin (with enabled interrupt or custom name), a set of equates are generated that define symbols for the data address and bit, and for the interrupt mask address and bit associated with the pin. The naming convention for the equates is:

- *CustomPinName_Data_ADDR*
- *CustomPinName_MASK*
- *CustomPinName_IntEn_ADDR*
- *CustomPinName_Bypass_ADDR*
- *CustomPinName_DriveMode_0_ADDR*
- *CustomPinName_DriveMode_1_ADDR*
- *CustomPinName_IntCtrl_0_ADDR*
- *CustomPinName_IntCtrl_1_ADDR*

The *CustomPinName* used in the substitution is replaced by the name entered for the pin during code generation. Custom pin naming allows you to change the name of the pin. The name field is included in the pin parameter area of the pin-out diagram.

The Name column in the Pin Parameter Grid shows the names assigned to each of the pins. The default name shows the port and bit number. Double-click the name field and type the custom name. Note that the name must not include any embedded spaces.

The effect of the name is primarily used in code generation when the pin interrupt is enabled. The pin name is appended to the equates that are used to represent the address and bit position associated with the pin for interrupt enabling and disabling, as well as testing the state of the port data.

6.6 Code Generation and Dynamic Re-configuration

When additional configurations are present, there is a considerable difference in code generation and the files generated. The User Module files are generated identically to previous versions. Differences are described ahead.

6.6.1 PSoCConfig.asm

The static *PSoCConfig.asm* contains exports and code for:

- LoadConfigInit: Initial configuration-loading function
- *LoadConfig_projectname*: Load configuration function

And code only for:

- LoadConfig: General load registers from a table

For projects with additional configurations, a variable is added to the bottom of the file that tracks the configurations that are loaded. The LoadConfig function

does not change at all. The *LoadConfig_projectname* function includes a line that sets the appropriate bit in the active configuration status variable. The name of this variable is fixed for all projects. Additional variables that shadow the “write only” registers are added when required.

Additional functions named *LoadConfig_configurationname*, are generated with exports that load the respective configuration. These functions are the equivalent of the *LoadConfig_projectname* function, including the setting of the bit in the active configuration status variable. The only difference is that *LoadConfig_configurationname* loads values from *LoadConfigTBL_configurationname_Bankn* and there is some additional code that manages the values of any global registers that are changed in the configuration relative to the base configuration.

For each *LoadConfig_xxx* function, an *UnloadConfig_xxx* function is generated and exported to unload each configuration, including the base configuration. The *UnloadConfig_xxx_Bankn* are similar to the *LoadConfig_xxx* functions except that they load an *UnloadConfigTBL_xxx_Bankn* and clear a bit in the active configuration status variable. In these functions, the global registers are restored to a state that depends on the currently active configuration.

With regard to the base configuration, *UnloadConfig_xxx* and *ReloadConfig_xxx* functions are also generated. These functions load and unload only User Modules contained in the base configuration. When the base configuration is unloaded, the *ReloadConfig_xxx* function must be used to restore the base configuration User Modules. The *ReloadConfig_xxx* function ensures the integrity of the “write only” shadow registers. Respective load tables are generated for these functions in *PSoCConfigTBL.asm*.

An additional unload function is generated as *UnloadConfig_Total*. The *UnloadConfig_Total* function loads tables *UnloadConfigTBL_Total_Bank0* and *UnloadConfigTBL_Total_Bank1*. These tables include the unload registers and values for all PSoC blocks. The active configuration status variable is also set to 0. The global registers are not set by this function.

The name of the base configuration will match the name of the project. The project name will be changed to match the base configuration name if you change the name of the base configuration (from the project name).

A “C” callable version of each function is defined and exported so that these functions can be called from a “C” program.

6.6.2 PSoCConfigTBL.asm

PSoCConfigTBL.asm contains the personalization data tables used by the functions defined in *PSoCConfig.asm*. For static configurations, there are only two tables defined; *LoadConfigTBL_projectname_Bank0* and

LoadConfigTBL_projectname_Bank1, which support the *LoadConfig_projectname* function. These tables personalize the entire global register set and all registers associated with PSoC blocks that are used by User Modules placed in the project.

For projects with additional configurations, a pair of tables are generated for each *LoadConfig_xxx* function generated in *PSoCConfig.asm*. The naming convention follows the same pattern as *LoadConfig_xxx* and uses two tables; *LoadConfigTBL_xxx_Bank0*, and *LoadConfigTBL_xxx_Bank1*. *UnloadConfigTBL_xxx_Bank0* and *UnloadConfigTBL_xxx_Bank1* are used by *UnloadConfig_xxx*. The labels for these tables are exported at the top of the file.

The tables for the additional configurations' **loading** function differ from the base configuration load table in that the additional configuration tables only include those registers associated with PSoC blocks that are used by User Modules placed in the project and only those global registers with settings that differ from the base configuration. If the additional configuration has no changes to the global parameters or pin settings, only the placed User Module registers are included in the tables.

The tables for additional configurations' **unloading** functions include registers that deactivate any PSoC blocks that were used by placed User Modules, and all global registers which were modified when the configuration was loaded. The registers and the values for the PSoC blocks are determined by a list in the device description for bitfields to set when unloading a User Module, and are set according to the type of PSoC block. The exceptions are the *UnloadConfigTBL_Total_Bankn* tables, which include the registers for unloading all PSoC blocks.

6.6.3 boot.asm

The *boot.asm* file is generated similarly to a project that has no additional configurations unless there are one or more configurations that have User Modules placed in such a way that common interrupt vectors are used between configurations. In this case, the vector entry in the interrupt vector table will show the line "`ljmp Dispatch_INTERRUPT_n`" instead of a User Module defined Interrupt Service Routine.

6.6.4 New Files

There are three new files that are generated when additional configurations are present in a project:

- *PSoCDynamic.inc*
- *PSoCDynamic.asm*

- *PSoCDynamicINT.asm*

The names of the files may change in future releases, but their functions will be the same regardless of the actual file names.

The *PSoCDynamic.inc* file is always generated. It contains a set of equates that represent the bit position in the active configuration status variable, and the offset to index the byte in which the status bit resides if the number of configurations exceeds eight. A third equate for each configuration indicates an integer index representing the ordinal value of the configuration.

The *PSoCDynamic.asm* file is always generated. It contains exports and functions that test whether a configuration is loaded or not. The naming convention for these functions is *IsOverlayNameLoaded.*

The *PSoCDynamicINT.asm* file is generated only when the User Module placement between configurations results in both configurations using a common interrupt vector. The reference to `Dispatch_INTERRUPT_n` function is resolved in this file. For each conflicting interrupt vector, one of these ISR dispatch sets is generated. The ISR dispatch has a code section that tests the configuration that is active and loads the appropriate table offset into a jump table immediately following the code. The length of the jump table and the number of tests depends on the number of User Modules that need the common vector rather than the total number of configurations. The number of conflicts can equal the number of configurations if each configuration utilizes the common interrupt vector. Generally, there will be fewer interrupt conflicts on a per-vector basis.

6.7 Application Editor and Dynamic Re-configuration

There are no direct changes in Application Editor with regards to Dynamic Reconfiguration. The additional files generated are placed in the Library Source and Library Headers folders of the source tree. Library source files that are associated with an additional configuration are shown under a folder with the name of the configuration. This partitions the files so that the source tree view is not excessively long.

6.8 Debugger and Dynamic Re-configuration

PSoC Designer Debugger subsystem now displays currently loaded configuration names and IO register labels during Debugger halts. The IO register grid labels are compiled from the labels for all currently loaded configurations.

The names of loaded configurations are displayed in a new Debugger view. The view is a new tab titled “Config” below the memory map (which already contains RAM, IO Banks 0,1, and Flash tabs).

The IO register labels modify the existing IO register bank grids. The only difference is that in addition to setting IO register labels on entry to the Debugger, labels are updated on M8C halts if the set of loaded User Modules has changed since the last halt.

The Debugger obtains the active configuration names from the runtime configuration data stored in M8C RAM. This data is maintained by the `LoadConfig` and `UnloadConfig` routines generated by the Device Editor.

6.8.1 Active Configuration Display

The set of currently active configurations is displayed in the “Config” tab of the memory map during Debugger halts. The display lists all project configurations with the status for each currently loaded configuration marked “Active.”

Note that the display is not valid immediately after a reset. The PSoC initialization code must run before the “Config” tab display is valid.

6.8.2 Active Configuration IO Register Labels

The Debugger IO register bank labels (Bank 0, Bank 1) are updated to match the User Modules defined in the currently active configurations.

6.8.3 Active Configuration Limitations

The new displays are based on a bitmap of loaded configurations maintained by the `LoadConfig` and `UnloadConfig` routines, which are generated by the Device Editor. This bitmap can get out-of-sync with the actual device configuration in several ways:

- The bitmap’s RAM area can be accidentally overwritten.
- If overlapping (conflicting) configurations are loaded at the same time, the register labels will be scrambled.
- If an overlapping configuration is loaded and then unloaded, register labels from the original configuration will be used, even though some PSoC blocks will have been cleared by the last `UnloadConfig` routine.

6.8.4 Active Configuration Display Test

The new display features can be tested with a single project that loads and unloads configurations containing overlapping User Modules.

The test project should have a base configuration that defines one or more User Modules and several overlay configurations, some conflicting and some not conflicting. The main line of the project should load and unload configura-

tions in various combinations. After each load and unload, the status of the active configuration display and the register label display should be checked.

Checking the displays after loading and unloading conflicting configurations is recommended.

Section 7. Application Editor

In this section you will learn definitions and recommended usage of critical files as well as how to modify files generated by PSoC Designer, add new files, and remove unwanted files.

Before you begin adding and modifying files, it is recommended that you take a few moments to navigate Application Editor, take inventory of your current files, and map out what you plan to do and how you plan to do it.

7.1 File Definitions and Recommendations

Once you have finished configuring your device and generating application code, you are ready to program the desired functionality into the device. This is done in the Application Editor subsystem. Application Editor is where all source-code programming (editing and adding files) takes place.

To access Application Editor, click the **Application Editor** icon . See Project Manager in section 3 to review subsystem navigation.

As discussed in Section 3., you will see the file source tree in the left frame. The files you see were generated when the project was created, and updated after device configuration. See 3.2 for general facts about these files.

The following definitions of critical system files should further your system knowledge and better prepare you for carrying out your vision for the MCU:

boot.asm: This startup file resides in the source tree under Source Files and is key because it defines the boot sequence. Device Editor uses a template (*boot.tpl*) from the software installation …\Templates directory to create *boot.asm*. Changes made to *boot.asm* will get overwritten when the code gets regenerated (application generation ). Following are components of the boot sequence:

- Originates reset vector for code that begins after the device is powered up.

- Holds interrupt table for code that is executed when an interrupt occurs.

Device configuration initialization will always occur because *boot.asm* is generated with a call to the configuration load function before the jump to main. In the case of Dynamic Re-configuration, the base configuration is automatically loaded, while other configurations must be loaded by the embedded developer.

- Calls device configuration initialization to enforce quick device configuration after reset.
- Creates a proper ‘C’ environment because ‘C’ code requires certain types of initializations.

The ‘C’ initialization code will occur even if the application is built using only assembly code.

- Calls `main` (or `_main`) to begin executing code.

boot.asm will be re-generated every time device configurations change and application files generated. This is done to ensure that interrupt handlers are consistent with the configuration. If you make changes to *boot.asm* that you do not want overwritten, hard code the change in *boot.tpl* (template for *boot.asm*).

Policy dictates that this file belongs to PSoC Designer. Again, it is created from a template. Keep in mind, it is highly recommended that you *do not* modify the contents of this file.

main.asm: This file resides under Source Files and is key because it holds the “`_main`” label that is referenced from the boot sequence. *main.asm* is created to resolve the external reference from the boot sequence. Upon new project creation, the “`_main`” function contains a simple “forever” loop.

This file can be removed and replaced with a ‘C’ “main” if you determine that most of the application should be written in ‘C’ source. It is only created once, when a new project is created. No additional policies or recommendations are attached to this file.

PSocConfig.asm: This is always a required Library Source file because it contains the configuration that is loaded upon system access. Initially, and probably for a very brief moment, there is not a configuration. Therefore this file contains a function label (`LoadConfig`), to satisfy the boot-sequence reference, as well as a return.

The function to load the configuration (`LoadConfig`) is called from the boot sequence. PSoC Designer will overwrite *PSocConfig.asm* when a device configuration has changed and application files regenerated - *no exceptions*.

> You must not re-configure or modify any aspect of a device configuration if you wish to preserve changes that you have made to *PSocConfig.asm*.

You can, however, keep a copy of any changes and reapply them after a re-configuration. Remember, because this is a Library Source file, it is added/replaced to *libPsoc.a*.

> If you wish to manipulate bits, all part register values reside in this file for your reference.

7.2 Additional Generated Files

In PSoC Designer versions 3.2x and higher, additional files are generated in association with User Modules and Dynamic Re-configuration.

This feature adds files and content to current files to provide you additional access to configuration information. Primarily, this additional information is associated with the Write-Only shadow registers generated by User Modules and Dynamic Re-configuration. These shadow registers must be used when manipulating bits within the Write-Only registers so that the data present in the register at any time is consistent with the operation of the User Module, or Dynamic Re-configuration, or user code.

In addition to the primary reason for this feature, files are also generated to make configuration data more accessible for use from C programs.

Following is the list of additional files:

- *Psocgpioint.inc* - Additional Information
- *Psocgpioint.h* - New File
- *Globalparams.h* - New File

Globalparams.h has the same contents as *projectname_globalparams.inc*, except it also has `#define` statements.

Psocgpioint.inc contains additional information pertaining to Write-Only register shadows. If a pin group is defined in a register set for which register shadows are allocated, then a set of three macros are defined for each register shadow to read, set, or clear the particular bit within the register associated with the pin. The names of the macros are keyed to the custom name assigned to the pin and are as follows...

- *GetCustomName_registerName*
- *SetCustomName_registerName*
- *ClearCustomName_registerName*

...where *CustomName* is the custom name set for the pin, and *registerName* is the associated register name for which a register shadow is allocated.

The *registerName* registers vary with the chip device description and include all registers associated with the GPIO ports. For the CY8C25xxx/26xxx device family, these registers include:

- Bypass
- DriveMode_0
- DriveMode_1
- IntCtrl_0
- intCtrl_1
- IntEn

For the CY8C27xxx device family, registers include:

- GlobalSelect
- DriveMode_0
- DriveMode_1
- DriveMode_2
- IntCtrl_0
- IntCtrl_1
- IntEn

The register shadow allocation is determined by User Modules and Dynamic Re-configuration. As the register allocation changes, the macro generation will change accordingly.

Psogpioint.h contains the same information as *Psocgpioint.inc* except in a form needed for C code. In the case of the register shadows, this file does not generate macros, but rather defines a symbol that allows manipulation of the shadow as a Global Variable. For each register shadow associated with a custom pin definition, a variable named *CustomName _registerNameShadow* is defined, where *CustomName* and *registerName* are the same as previously defined for *Psocgpioint.inc*. The variable name can then be used to manipulate

the shadow register. For example, to set a pin value to 1 within the port, do the following:

```
CustomName registerNameShadow |= CustomName_MASK;
CustomName_registerName_ADDR = CustomName_registerNameShadow;
```

7.3 Modifying Files

When you are ready to program and modify assembly-language source files, double-click the target file from under the source tree in the left frame. The open file will then appear in the main active window.

Open files are accessible from within the Debugger subsystem under the Window menu as Read Only but are displayed editable by default in Application Editor.

You can have as many files open as you wish (or that your computer will allow). For further details regarding open files and active windows, refer back to Section 3.

Following, is a description of the menu options available for modifying source files:

Table 9: Menu Options for Modifying Source Files

Icon	Option	Menu	Shortcut	Feature
	Application Editor	View >> Application Editor		Enables source tree and files for editing
	Compile/Assemble	Build >> Compile/Assemble	**[Ctrl] [F7]**	Compiles/assembles the most prominent open, active file (.c or .asm)
	Build	Build >> Build	**[F7]**	Builds entire project and links applicable files
	New File	File >> New	**[Ctrl] [N]**	Adds a new file to the project
	Open File	File >> Open	**[Ctrl] [O]**	Opens an existing file in the project
	Indent			Indents specified text
	Outdent			Outdents specified text
	Comment			Comments selected text
	Uncomment			Uncomments selected text

Table 9: Menu Options for Modifying Source Files, continued

	Toggle Bookmark			Toggles the bookmark: Sets/removes user-defined bookmarks used to navigate source files
	Clear Bookmark			Clears all user-defined bookmarks
	Next Bookmark			Goes to next bookmark
	Previous Bookmark			Goes to previous bookmark
	Find Text	Edit >> Find	[**Ctrl**] [**F**]	Find specified text
	Replace Text	Edit >> Replace	[**Ctrl**] [**H**]	Replace specified text
	Find in Files	Edit >> Find in Files		Find specified text in specified file(s)
	Repeat Replace			Repeats last replace
	Set Editor Options			Set options for editor
	Undo	Edit >> Undo	[**Ctrl**] [**Z**]	Undo last action
	Redo	Edit >> Redo	[**Ctrl**] [**Y**]	Redo last action

Note that in all source files the maximum number of characters allowed per line is 2,048. The maximum per word is 256. These limits are imposed by the PSoC Designer development software.

7.4 Adding Files

Adding files to your project, or for use with other projects, is essential for complete, well-balanced device functionality. To add a file:

1. Click on the New File icon, or select File >> New.
2. In the New File dialog box, select a file from the File type field. (For general facts about these files refer back to 3.2.)
3. In the File name field, type the name for the file.

4. In the Location field you will see that your current project directory is the default destination for your file. Uncheck the Add to current project field and click Browse to identify a different location if you do not want the default.

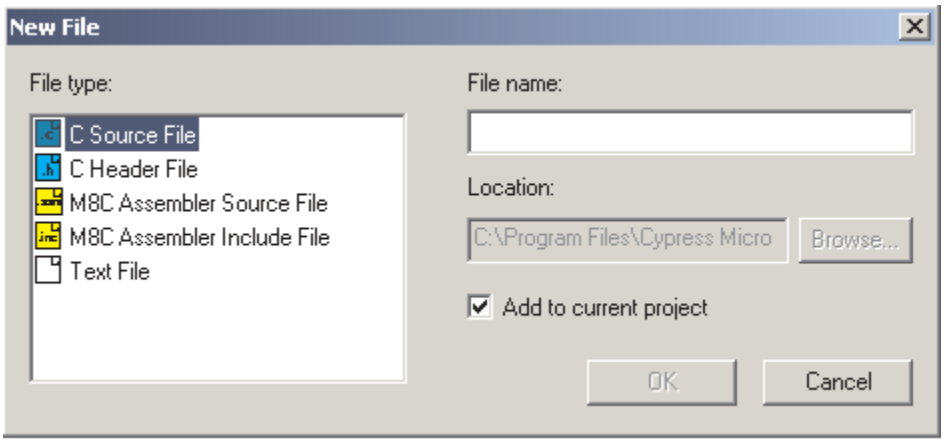


Figure 61: Add New File

The **Browse** button will only be enabled if you uncheck the Add to current project field.

5. When finished, click **OK**. Click **Cancel** if you wish to cancel the operation.

Your new file will be added to the file source tree and appear in the main active window.

You are also able to add other source files to your project (either C or assembly) even if these source files already exist somewhere else on your computer. Do this by accessing Project >> Add to Project >> Files and identifying the source file (by locating the file with the file dialog). Keep in mind that you will be adding a *copy* of your original file to the project, not the original itself. See *PSoC Designer: C Compiler Language User Guide* for further options and guidelines.

For high-level guidance on programming assembly-language source files, see Section 8. in this user guide. For comprehensive details, see *PSoC Designer: Assembly Language User Guide*.

7.5 Removing Files

You can remove files from your project one of three ways. Either comment-out an unwanted file through code manipulation, remove the file by clicking once to highlight it in the source tree and accessing Project >> Remove from Project, or right-click the file in the source tree and select Remove from Project

([**Delete**] key). The second and third actions remove the file permanently, whereas the first action simply bypasses it. All three ways are acceptable to PSoC Designer.

7.6 Full Application File Search

In addition to the standard Find/Replace feature in PSoC Designer, you can now search for specific text inside specific files. To search for text in any single file or combination of multiple files, execute the following:

1. Click Edit >> Find in Files or the Find in Files icon .

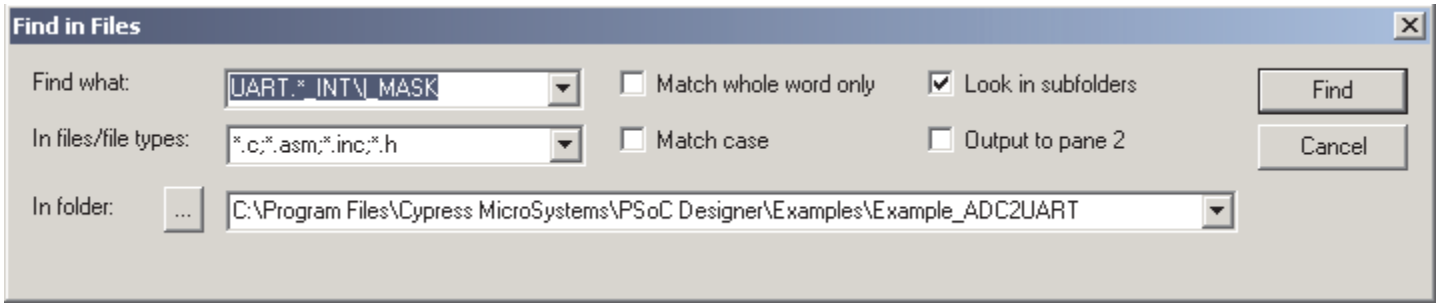


Figure 62: Find in Files Dialog Box

2. In the Find what field of the Find in Files dialog box, type or click the drop-arrow to choose a previously “searched” word.

Here, you can search by standard grep (Global Regular Expression Print) methods. grep searches the input files for lines containing a match to a given pattern list. For options, see *grep.pdf* in the `\Documentation\Supporting Documents` directory for PSoC Designer.

3. In the In files/file type field, type or click the drop-arrow to choose a previously “searched” file or file type. Separate multiple files by semi-colon.

4. Click the ... box if you wish to search a different project directory than the directory of your currently open project. (Or click the drop-arrow to choose a previously “searched” directory.)

5. Click a check in the specifics: Match whole word, Match case, Look in subfolders, and Output to pane 2 (Find in Files 2 tab of the Output Status window).

6. When finished, click **Find**. Click **Cancel** to close Find in Files dialog box.

To cancel an active search, re-open the Find in Files dialog box.

In the Find in Files 1 (or Find in Files 2) tab of the Output Status window you will see the files resulting from your search. Double click to open the file. Inside the file in the left margin there is a yellow arrow indicating the text for which you were looking.

Section 8. Assembler

In this section you will receive high-level guidance on programming assembly-language source files for the PSoC Device.

For comprehensive details, see *PSoC Designer: Assembly Language User Guide*.

8.1 Accessing the Assembler

The Assembler is an application accessed from within PSoC Designer, much like the C Compiler. This application is run as a batch process. It operates on assembly-language source, constructed by you, to produce executable code. This code is then compiled and built into a single executable file that can be downloaded into the In-Circuit Emulator (ICE), where the functionality of the PSoC device can be emulated and debugged.

To access assembly-language source files, click the **Application Editor** icon in the toolbar.

The project source files appear in the left frame (source tree). Double-click individual files to appear in the main active window where you can add and modify code using the enabled edit icons.

8.2 The Microprocessor

The MCU is an enhanced 8-bit microprocessor core. It supports 8-bit operations and has been optimized to be small and fast.

The Internal registers are: the Accumulator 'A'; the Flag Register ‘F’; the Index Register ‘X’; the Stack Pointer 'SP'; and the Program Counter 'PC'. All registers are 8 bits wide except 'PC' which is composed of two 8-bit registers (PCH and PCL) which together form a 16-bit register.

8.2.1 Address Spaces

There are three separate address spaces implemented in the Assembler: Register space (REG), data RAM space, and program memory space.

The Register space is accessed through the MOV and LOGICAL instructions. There are 8 address bits available to access the Register space, plus an extended address bit via the flag register bit 4.

The data RAM space contains the data/program stack, and space for variable storage. All the read and write instructions, as well as instructions which operate on the stacks, use data RAM space. Data RAM addresses are 8 bits wide, although for RAM sizes 128 bytes or smaller, not all bits are used. The Extended Address flag bits (XA[2:0]) are used to address beyond the first 256 bytes of RAM. Depending on the memory size implemented on a particular device, any or all of the Extended Address bits may not be implemented. These 3 bits provide an 11-bit RAM address for addressing up to 2 kilobytes as 8 pages of 256 bytes each. The flag register must be manipulated to change RAM page addresses.

All stack operations force XA[2:0] on the bus to be zero (leaving flag values intact) so that the stack is constrained to the first 256-byte page.

The program memory space is organized into 256 byte pages, such that the PCH register contains the memory page number and the PCL register contains the offset into that memory page. The M8C automatically advances PCH when a page boundary needs to be crossed. The user need not be concerned with program memory page boundaries, as they are invisible within the programming module. The one exception to this is that non-jump instructions ending on a page boundary will take an extra cycle to complete. Jump instructions are not affected in this manner.

8.2.2 Instruction Format

Instruction addressing is divided into two groups: (1) Logic, arithmetic, and data movement functions (unconditional); (2) jump and call instructions, including INDEX (conditional).

Logic, arithmetic, and data movement functions are one-, two-, or three-byte instructions. The first byte of the instruction contains the opcode for that instruction. In two-byte instructions, the second byte contains either a data value or an address.

Most jumps, plus CALL and INDEX, are 2-byte instructions. The opcode is contained in the upper 4 bits of the first instruction byte, and the destination address is stored in the remaining 12 bits. For memory sizes larger than 4 kilobytes, a three-byte format is used in Big Endian format.

8.2.3 Addressing Modes

Ten addressing modes are supported; Source Immediate, Source Direct, Source Indexed, Destination Direct, Destination Indexed, Destination Direct

Immediate, Destination Indexed Immediate, Destination Direct Direct, Source Indirect Post Increment, and Destination Indirect Post Increment. For examples of each see *PSoC Designer: Assembly Language User Guide*.

8.2.4 Destination of Instruction Results

The result of a given instruction is stored in the entity, which is placed next to the opcode in the assembly code. This allows for a given result to be stored in a location other than the accumulator. Direct and Indexed addressed Data RAM locations, as well as the X register, are additional destinations for some instructions. The AND instruction is a good illustration of this feature (i2 == second instruction byte, i3 == third instruction byte):

Table 10: Destination of AND Instruction

Syntax	Operation
AND A, expr	acc ← acc & i2
AND A, [expr]	acc ← acc & [i2]
AND A, [X + expr]	acc ← acc & [x + i2]
AND [expr], A	[i2] ← acc & [i2]
AND [X + expr], A	[x + i2] ← acc & [x + i2]
AND [expr], expr	[i2] ← i3 & [i2]
AND [X + expr], expr	[x + i2] ← i3 & [x + i2]

The ordering of the entities within the instruction determines where the result of the instruction is stored.

8.3 Assembly File Syntax

Assembly language instructions reside in source files with .asm extensions in the source tree of Application Editor. Each line of the source file may contain five keyword-types of information. The following table gives critical details about each keyword-type:

Table 11: Keyword Types

Keyword Type	Critical Details
Label	A symbolic name followed by a colon (:).
Mnemonic	An assembly language keyword.
Operands	Follows the Mnemonic.
Expression	Is usually addressing modes with labels and must be enclosed by parentheses.
Comment	Can follow Operands or Expressions and start in any column if the first non-space character is either a C++ style comment (//) or semi-colon (;).

Instructions in an assembly file have one operation on a single line. For readability, separate each keyword-type by tabbing once or twice (approximately 5-10 white spaces).

See *PSoC Designer: Assembly Language User Guide* for type definitions and an example of assembly-file syntax.

8.4 List File Format

When you build a project (using all assembly files), a listing file with an .lst extension is created. The listing shows how the assembly program is mapped into a section of code beginning at address 0. The linking (building) process will resolve the final addresses. This file also provides a listing of errors and warnings, and a reference table of labels.

Also generated during a build (in addition to the .lst) are .rom, .mp, .dbg, and .hex files. The .rom is used for debugging and programming. The .mp contains global symbol addresses and other attributes of output. And the .dbg and .hex files are good troubleshooting resources.

.lst files can be viewed after a project “build” in the Debugger subsystem under the Output tab of the source tree (see Figure 12:).

8.5 Assembler Directives

The PSoC Designer Assembler allows the assembler directives listed below:

Table 12: Assembler Directives

Symbol	Assembler Directive
AREA	Area
ASCIZ	NULL Terminated ASCII String
BLK	RAM Byte Block
BLKW	RAM Word Block
DB	Define Byte
DS	Define ASCII String
DSU	Define UNICODE String
DW	Define Word
DWL	Define Word with Little Endian Ordering
ELSE	Alternative Result of IF…ELSE…ENDIF
ENDIF	End of IF…ELSE…ENDIF
EQU	Equate Label to Valuable Value
EXPORT	Export

Table 12: Assembler Directives, continued

IF	Conditional Assembly
INCLUDE	Include Source File
.LITERAL, .ENDLITERAL	Prevent Code Compression of Data
MACRO/ENDM	Macro Definition Start/End
ORG	Area Origin
.SECTION, .ENDSECTION	Section for Dead-Code Elimination
Suspend - OR F,0 Resume - ADD SP,0	Suspend and Resume Code Compressor

See *PSoC Designer: Assembly Language User Guide* for descriptions and sample listings of supported assembler directives.

8.6 Instruction Set

All instructions are 1, 2, or 3 bytes wide and fetched from program memory, in a separate address space from data memory. The first byte of an instruction is an 8-bit constant, referred to as the Opcode. Depending on the instruction, there can be one or two succeeding bytes that encode address or operand information.

The following notation is used for the instructions:

Table 13: Instruction Set Notation

Notation	Description
A	Primary Accumulator
CF	Carry Flag
expr	Expression
F	Flags (ZF, CF, and Others)
k	Operand 1 Value
k_1 k_2	First Operand of 2 Operands Second Operand of 2 Operands
PC	PCH, PCL
SP	Stack Pointer
X	X Register
ZF	Zero Flag

To access a complete instruction in detail within PSoC Designer, click your cursor on the target instruction in the file and hit [**F1**].

See *PSoC Designer: Assembly Language User Guide* for the complete instruction set.

8.7 Compiling/Assembling Files

Once you have finished programming all assembly-language source (in addition to any .c source), you are ready to compile/assemble the group of files. Compiling translates source code into object code. (The Linker then combines modules and gives real values to symbolic addresses, thereby producing machine code.) Each time you compile/assemble, the most prominent, open source file will be compiled.

PSoC Designer can decipher the difference between .c and assembly language files and compile/assemble accordingly.

To compile the source files for the current project, click the **Compile/Assemble** icon in the toolbar. Compiling must be done in Application Editor.

PSoC Designer now employs a “make” utility. Each time you click the Compile/Assemble or Build icon, the utility automatically determines which files of a large application (manual or generated) have been modified and need to be recompiled, then issues commands to recompile them. For further details, see *make.pdf* in the `\Documentation\Supporting Documents` directory for PSoC Designer.

As discussed in section 3, the Output Status (or error-tracking) window of Application Editor is where the status of file compiling/assembling resides.

Each time you compile/assemble files, the Output Status window is cleared and the current status entered as the process occurs.

When compiling is complete, you will the see the number of errors. Zero errors signify that the compilation/assemblage was successful. One or more errors indicate problems with one or more files. This process reveals syntax errors. Such errors include `missing input data` and `undeclared identifier`. For a list of all identified compile (and build) errors with solutions see Section 8. Compiling/Assembling Files in *PSoC Designer: Assembly Language User Guide*. For further details on compiling and building see Section 9. in this user guide.

At any time you can ensure a “clean” compile/assemble (or build) by accessing Project >> Clean then clicking the Compile/Assemble or Build icon. The “clean” will delete all `lib\libPSoc.a`, `obj\*.o`, and `lib\obj\*.o` files. These files are regenerated upon a compile or build (in addition to normal compile and build activity).

Section 9. Builder

In this section you will learn details of building a project and of the C Compiler as well as basic, transparent functions of the system Linker/Loader and Librarian.

For comprehensive details on the C Compiler, see *PSoC Designer: C Language Compiler User Guide*.

9.1 Building a Project

It is not necessary to compile all the source files individually before you build the entire project. Building your project will compile/assemble all source files and selectively assemble library source files. The build process performs the compile/assemble of project files then links to all the project's object modules (and libraries), creating a .rom file that can be downloaded for debugging.

If the debugger is started, following source code changes, but the project build is not performed, the debugger will force a build to ensure the .rom file is synchronized with the source changes.

To build the current project, click the **Build** icon in the toolbar (or Build >> Build from the menu or just [**F7**]).

PSoC Designer now employs a "make" utility. Each time you click the Compile/Assemble or Build icon, the utility automatically determines which files of a large application (manual or generated) have been modified and need to be recompiled, then issues commands to recompile them. For further details, see *make.pdf* in the `\Documentation\Supporting Documents` directory for PSoC Designer.

The initial build action of compiling/assembling causes object modules to be created (assuming no errors occur) and places them in the …`\project name\obj` or …`\project name\lib\obj` folder. The link/build action uses the object modules to produce the final project image in the …`\project name\output` folder.

As mentioned, the .rom file can be downloaded to the ICE. Other files also reside in the folder to provide reference for the Debugger subsystem and you, as the program developer. Such files include .dbg, the entire program listing .lst, and memory map .mp.

At any time you can ensure a “clean” build by accessing Project >> Clean then clicking the Build icon. The “clean” will delete all `lib\libPSoc.a`, `obj\*.o`, and `lib\obj\*.o` files. These files are regenerated upon a build (in addition to normal build activity).

Each time you build your project, the Output Status window is cleared and the current status entered as the process occurs.

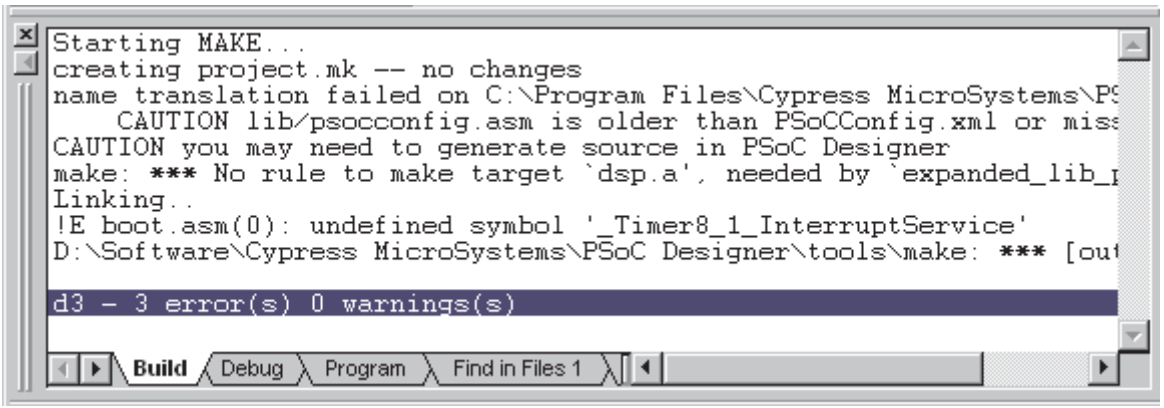


Figure 63: Output Status Window

To save all open files in Application Editor, click File >> Save All.

When the build is complete, you will the see the number of errors and warnings. Zero errors signify that the build was successful. One or more errors indicate problems with one or more files and therefore, the program image (.rom file) will not be available to download to the ICE.

For a list of all identified compile and build errors with solutions see *PSoC Designer: Assembly Language User Guide*.

9.2 C Compiler

The Cypress MicroSystems PSoC family of devices and PSoC Designer support a high-level C language compiler by iMAGEcraft. Even if you have never worked in standard C language before, this system resource enables you to quickly create a complete C program for a PSoC device. The C language compiler in PSoC Designer allows users more design flexibility.

The embedded, optimizing C compiler provides all the features of C, but is tailored to PSoC Designer architecture. It includes a built-in macro assembler allowing assembly-language code to be seamlessly merged with C code. The link libraries use absolute addressing, or can be compiled in relative mode and linked with other software modules to get absolute addressing.

The compiler compiles each .c source file to an .asm assembly file. The PSoC Designer Assembler then translates each .asm (either those produced by the compiler or assembly files that have been added) into a relocatable object file, .o. After all the files have been translated into object files, the builder/linker combines them together to form an executable file.

The PSoC Designer C Compiler comes complete with embedded libraries providing port and bus operations, standard keypad and display support, and extended math functionality. For comprehensive details on the C compiler, see *PSoC Designer: C Language Compiler User Guide*.

9.3 Linker/Loader

The linking and loading functions in the build process of PSoC Designer are transparent to the user. As discussed earlier in this section, building your project links all the programmed functionality of the source files (including device configuration) and loads it into a .rom file, which is the file you download for debugging.

The linking process links intermediate object and library files generated during compilation/assemblage, checks for unresolved labels, and then loads results into a .rom and a .lst file as well as assorted .o and .dbg files. Again, for descriptions of these files, refer to 3.2.

9.3.1 Customized Linker Actions

It is possible to customize the actions of the Linker when a PSoC Designer “build” does not provide the user interface to support these actions.

A file called *custom.lkp* can be created in the root folder of the project, which can contain Linker commands (see the Command Line Compiler Overview section in *PSoC Designer: C Language Compiler User Guide*).

The file name must be *custom.lkp*. Be aware that in some cases, creating a text file and renaming it will still preserve the .txt file extension (e.g. *custom.lkp.txt*). If this occurs, your custom commands will not be used. The “make“ file process reads the contents of *custom.lkp* and amends those commands to the Linker action.

A typical use for employing the *custom.lkp* capability would be to define a custom relocatable code AREA. Using a custom AREA and the *custom.lkp* file allows you to set a specific starting address for this AREA.

For example, if you wish to create code in a separate code AREA that should be located in the upper 2k of the Flash, you could use this feature. For the sake of this example, let's call the custom code AREA ‘BootLoader’. If you were developing code in 'C' for the 'BootLoader' AREA you would use the following pragma in your 'C' source file:

```
#pragma text:BootLoader         // switch the code below from
                              // AREA text to BootLoader
// ... Add your Code ...
#pragma text:text                 // switch back to the text
                              // AREA
```

If you were developing code in assembly you would use the AREA directive as follows:

```
AREA BootLoader(rom,rel)
; ... Add your Code ...
AREA text ; reset the code AREA
```

Now that you have code that should be located in the 'BootLoader' AREA, you can add your custom Linker commands to *custom.lkp*. For this example, you would enter the following line in the *custom.lkp* file:

```
-bBootLoader:0x3800.0x3FFF
```

You can verify that your custom Linker settings were used by checking the 'Use verbose build messages' field in the Builder tab under the Tools >> Options menu. You can “build” the project then view the Linker settings in the Build tab of the Output Status window (or check the location of the BootLoader AREA in the .mp file).

9.4 Librarian

The library and archiving features of PSoC Designer provide system storage and reference.

There are two types of Librarian files; Library Source and Library Headers, which can be found in the source tree. Source file types include archived and assembly language such as *libPSoc.a* and *PSocConfig.asm*. Header files are intermediate reference/include files created during application-code generation and compilation. Both types are generated and used by PSoC Designer and are unique to each specific project. See File Definitions and Recommendations in Section 7. for recommended usage.

Section 10. Debugger

In this section you will learn how to connect and download your project to the In-Circuit Emulator (ICE), debug and perfect functionality, and program the part. For additional information about the ICE and the Development Kit, refer to Application Note AN2018 Care and Feeding of ICE Pods under PSoC >> Application Notes at http://www.cypress.com/.

10.1 Debugger Components

The PSoC Designer Debugger provides in-circuit emulation that allows you to test the project in a hardware environment while viewing and debugging

device activity in a software environment. The Basic Development Kit contains the following components:



Figure 64: Basic Development Kit Components

Note that the CAT5 Patch Cable should be no longer than 1 ft. It must also have 8 wires in order to connect from the ICE to the Pod (some data CAT5 cables only have 4 wires).

The following diagram depicts the ICE front panel:

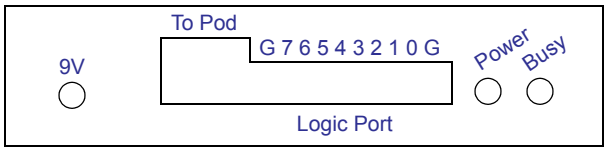


Figure 65: ICE Front Panel

9V	Power Adapter Input
To Pod	CAT5 Cable Input
Logic Ports	G = Ground 0-7 = Input Pins 7 = External Trigger Pin
LEDs	Power: ICE Unit Powered Up Busy: ICE Unit In Use

10.2 Connecting to the ICE

The purpose of this section is to facilitate establishing a connection to the In-Circuit Emulator (ICE). The PSoC ICE provides significant debugging functionality that requires full two-way communication over the ICE to operate. There are several steps in the connection process, including both setting up the hardware, and making the communications connection in the software. Making the software connection on your computer may require changes in the BIOS settings.

Some recent laptops do not support EPP and Bi-directional modes in the BIOS needed for full two-way communication over the ICE. A relatively easy method that bypasses the need for changing the BIOS settings is to install a parallel port card. This has the added benefit of providing a dedicated port to the ICE without potential conflicts with other applications or printers a user may have on their computer. Section 6. Alternate Parallel Port Cards in the *PSoC Designer: ICE Troubleshooting Guide* details parallel port cards for both desktops and laptops that have been tested for compliance with the ICE.

New with PSoC Designer v. 4.1 is the USB Dongle. The USB Dongle allows the standard parallel port ICE to connect to a USB 1.1 or 2.0 port. The dongle has a parallel cable connector on one end and a USB "B" connector on the other. The ICE may be plugged directly into the dongle's connector or a parallel cable may be used. The USB "B" connector is connected to your PC through a standard USB "A" -to- "B" cable.

In order for the dongle to be recognized and configured automatically, it should be plugged in after installing PSoC Designer v. 4.1. Once the dongle has been recognized, it will be available in the Project Settings dialog box under the

Debugger tab. Using the USB Dongle, you can connect to the ICE and debug with the same functionality as with an LPTx parallel port connection.

We want to assist you in troubleshooting any problems with the ICE connection. If the information in this user guide or the *PSoC Designer: ICE Troubleshooting Guide* is not sufficient to resolve any issues, please use the following resources:

TightLink Technical Support System
You can enter a support request in this system with a guaranteed response-time of four hours:

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

Support Forums
View and participate in discussion threads about a wide variety of PSoC device topics:

http://www.cypress.com/forums/

10.2.1 Connecting the Hardware

Installing PSoC Designer on Windows NT/2000/XP requires user to have local Administrator permission.

To physically connect your computer to the ICE (and related hardware), perform the following steps:

1. Locate the parallel interface cable, ICE, power adapter, CAT5 Patch cable, Pod, and Pup.

2. Plug the parallel interface cable into the LPT1 port (back of computer).

If your PC's main connection to its printer is through LPT1, you may need to temporarily re-route printing to an alternate port, the network, or a file. This is done through Control Panel >> Printers.

3. Plug the other end of the parallel interface cable into the ICE.

4. Plug the power adapter into the ICE (and AC receptor).

5. Plug the CAT5 Patch cable into the ICE and the Pod.

6. Connect the Pup to the Pod (if you are planning to run one of the tutorial/ demonstration projects).

If you are using your own circuit board, plug the Pod into your board, turn on board power, *then* connect the Pod to the ICE via the CAT5 cable. The ICE will automatically determine the power source.

Following is a closer look at some of the components:

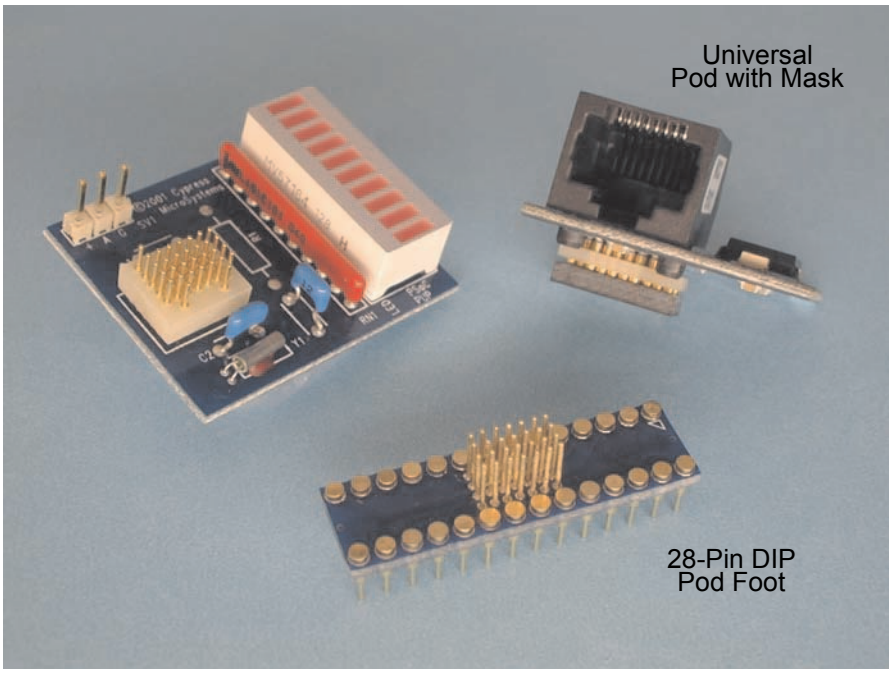


Figure 66: Pod, Pod Foot, and Demonstration Board

7. Reboot your machine and launch BIOS during boot up by pressing [**F2**] or [**Delete**].

If [**F2**] or [**Delete**] do not launch your BIOS, see Section 5 How to Access YOUR PC BIOS in the *PSoC Designer: ICE Connection Troubleshooting User Guide* to identify the BIOS for your particular PC.

8. In BIOS Setup, select EPP mode, as this setting works most often (for both desktops and laptops).

   Because the BIOS settings vary per machine, the correct mode cannot be known in advance and may take some trial and error. Options include EPP, ECP, EPP+ECP, and Bi-directional.

9. Save the settings, exit the BIOS, reboot, and launch PSoC Designer.

10.2.2 Connecting the Software

Once you have made the physical connection, you are ready to make the internal connection from PSoC Designer to the ICE. The ICE enables communication and debugging between PSoC Designer and the Pod. To connect to the ICE from inside PSoC Designer, perform the following steps:

1. Confirm that the Pod is connected to the ICE with the CAT5 Patch cable (< 1 ft. in length).

2. Confirm that the parallel port connection is secure from the ICE to the PC.

3. Confirm that the ICE is powered from the adaptor (yellow LED on, green LED off).

4. Open a project (Example projects are also available).

5. Click the Debugger icon to access the Debugger subsystem (using example project, Example_PWM_28-pin, from the ...\Examples directory of PSoC Designer).

6. Click the **Connect** icon.

Upon successful connection, you will receive notification in the Output tab of the status window and a green indicator displaying Connected will appear in the lower-right corner of the subsystem.

If you receive notification that the ICE did not connect several things can be the cause:

1. It has been documented that some computers running Windows 2000 can be put into the correct mode, but even when machines are rebooted once, twice, or several times, the ICE will still not connect the first time PSoC Designer launched. If this scenario is familiar, do not modify the mode (EPP, ECP, or Bi-directional) because this could be correct. Simply reboot, launch PSoC Designer, and try connecting.

2. When using Windows NT, 2000, or XP with versions of PSoC Designer 2.16 or earlier, the machine needs to be rebooted twice in order for the parallel port driver to initiate the connection for the first time. For all later versions of PSoC Designer, the machine only needs to be re-booted when the installation requests a reboot with the selection of the BIOS parallel port.

3. The correct mode has not been selected from the BIOS for the system, in which case the machine will need to be rebooted and another mode tried. Try Bi-directional, EPP+ECP, then ECP (usually for Windows 98 systems).

4. Other hardware/applications may be accessing the parallel port. For instance, printers or a full version of Adobe Acrobat can interfere with the use of the parallel port. You can redirect prints to a file. Another way to circumvent other applications interfering with the accessibility of the port is to obtain a PCI parallel card to provide a dedicated parallel port. For further details, see *PSoC Designer: ICE Connection Troubleshooting User Guide*.

If you are not interested in a dedicated port, verify that no other hardware, such as a printer or scanner, is configured to access the same LPT port as the ICE.

5. There can be issues connecting the In-Circuit Emulator (ICE) to a Gateway Solo 9500. This problem can be solved if you upgrade the Gateway Solo 9500 BIOS, as described in the *PSoC Designer: ICE Connection Troubleshooting User Guide*.

6. Failed Hardware: Although all hardware is tested by Cypress MicroSystems before leaving the factory, it is possible to have a faulty parallel cable, CAT5 cable, or Pod. Try swapping parallel cables or Pods if possible. Swapping the CAT5 cable is not advised. (The ICE requires CAT5 cables 1 foot or less in length with all 8 wires connected. Some patch cables contain only 4 wires.)

If you are still experiencing problems, contact the Cypress MicroSystems Applications Engineering Hotline at 425.787.4814. Also, consult *PSoC Designer: ICE Connection Troubleshooting User Guide*.

10.3 Downloading to Pod

Before you can begin a debug session you need to download your project .rom file to the Pod. By doing this, you are loading the ROM addressing data into the emulation bondout device (chip on the Pod). To download the .rom file to the Pod:

1. Click the Download to Emulator (Pod) icon .

A general rule to follow before downloading is to make sure there is not a part in the programming socket of the Pod. Otherwise, debug sessions may fail.

The system downloads the project .rom file located in the …\output folder of your project directory. A progress indicator will report download status.

The Pod can now be directly connected to and debugged on your specific circuit board.

You will receive the following message if your project cannot be debugged with the current Pod:

“Unable to connect to ICE due to pod incompatibility with project.

Found pod:

Silicon: 0008 Family: 0 Die: 0 Rev: AC

Expected pod(s):

Silicon: 0011 Family: 1 Die: 3 Rev: AA
Silicon: 0015 Family: 1 Die: 2 Rev: AA
Silicon: 000E Family: 1 Die: 0 Rev: AA
Silicon: 000E Family: 1 Die: 0 Rev: AB

Check the POD revision code.”

10.4 Debug Strategies

Debugger commands allow you to read and write program and data memory, read and write I/O registers, read and write CPU registers and RAM, set and clear breakpoints, and provide program run, halt, and step control.

In the status bar of the Debugger subsystem you will find ICE connection indication, debugger target state information, and (in blue) **A**ccumulator, **X**, **S**tack **P**oint, **P**rogram **C**ounter, and **F**lag register values.

10.4.1 Trace

This feature of PSoC Designer enables you to track and log device activity at either a high or detailed level. Such activity includes register values, data memory, and time stamps.

The Trace window displays a continuous, configurable listing of project symbols and operations from the last breakpoint. (The trace shows symbolic rather than address data to enhance readability.) Each time program execution

starts, the trace buffer is cleared. When the trace buffer becomes full it continues to operate and overwrite old data.

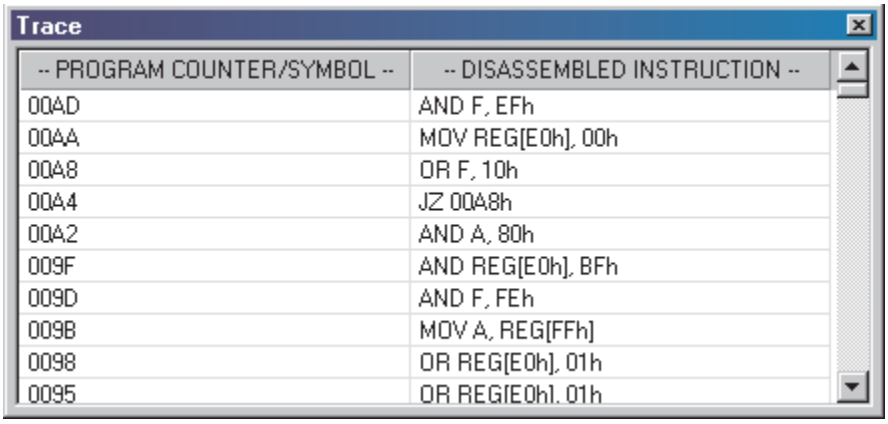


Figure 67: Trace

To help with troubleshooting, you can view read-only versions of your application source files inside the Debugger subsystem. If the project source tree is not showing in the left frame, click View >> Project and double-click any file you would like to view, or click the Project View icon .

The Trace window is displayed when Trace is chosen from the Debug menu (or the icon selected). It is configured by selecting either Debug >> Trace Mode or Tools >> Options from the menu. Configuration options include PC Only, PC/Registers, or PC/Timestamp.

PC Only mode lists the PC value and instruction only. PC/Registers mode lists the PC, instruction, data, A register, X register, SP register, F register, and ICE external input. PC/Timestamp mode lists the PC, instruction, A register, ICE external input, and timestamp.

The ICE external input value is the binary representation of the 8 center pins on the 10-pin ICE header. The right and left outside pins are connected to ground while the inputs accept a 5-volt TTL level signal. The timestamp is displayed as a 32-bit relative count of clock cycles from the CPU clock source.

The default size of trace is 256 kilobytes. This provides 128 K trace instructions in trace mode 1 and 32 K trace instructions in modes 2 and 3.

If you right-click your mouse inside the Trace dialog box you can copy or save the trace results as .txt, .xml, or another file type of choice.

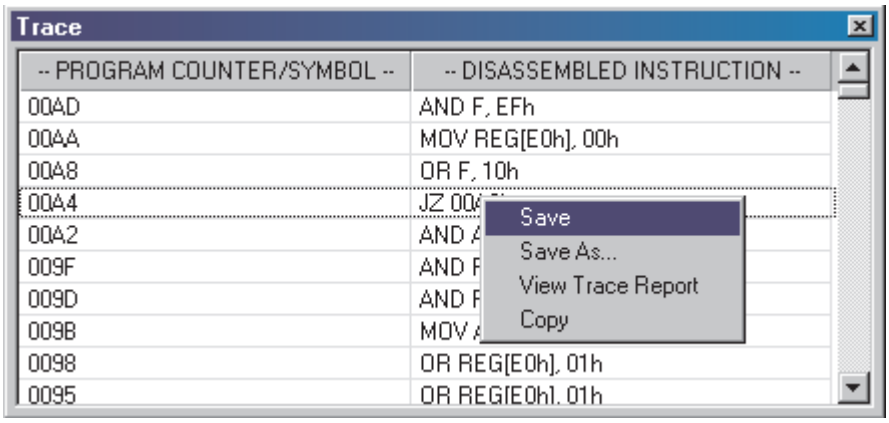


Figure 68: Trace Save As

You can also view the results in spreadsheet format inside your browser by clicking View Trace Report.

When viewing the Trace Report, at the top of your browser you will see the file path to which PSoC Designer saved the spreadsheet for later access.

Note that the trace output under reports by two instructions from the current actual Program Counter location when halted.

Contents of Trace Log Entries

The following entries are logged before the instruction is executed:

- PC Register
- A Register
- Data Bus
- External Signals

10.4.2 Breakpoints

This feature of PSoC Designer allows you to stop program execution at predetermined address locations. When a break is encountered, the program is halted at the address of the break, without executing the address's code. Once halted, the program can be restarted using the available menu/icon options.

To set breakpoints, first open the file you wish to debug. Do this from the source tree. (If your project file source tree is currently not showing, click View

>> Project.) Breakpoints are selected and deselected by clicking your mouse at targeted points in the left margin of the open file. You can view and remove active breakpoints in the Breakpoints dialog box accessed through Debug >> Breakpoints.

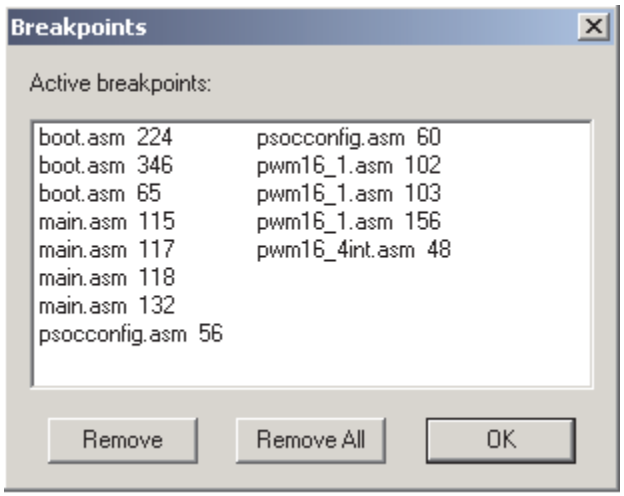


Figure 69: Debug Breakpoints

You can view the exact line and column for each breakpoint (or wherever you click your cursor in the file) across the bottom of PSoC Designer.

To set a bookmark in a source file for your own personal reference in Application Editor or Debugger, click the blue-flag icon in the toolbar. Use the other flag icons to advance to and/or clear bookmarks.

10.4.3 CPU and Register Views

There are five accessible “watch” windows that are readable and write-able during debugging. They are CPU Registers, Bank Registers 0,1, RAM, and FLASH (accessed at View >> Debug Windows).

The CPU Register window allows you to examine and change the contents. Click the drop-arrow to access a register then double-click and type over the value.

CPU register values can be viewed in blue across the bottom of PSoC Designer.

Each register is viewable by clicking the applicable lower tab of the Registers window. Double-click on a location and enter a new value to update the location’s value.

The current status of all locations can be saved to a .txt file by right-clicking at the top of the window and selecting Save or Save As.

Use caution when changing register values (they can alter hardware functions).

10.4.4 Watch Variables

Watch Variables can be set at Debug >> Watch Variables (or by right-clicking Add in the Watch/Global Name window). In the ASM Watch Properties dialog box you can specify the address you wish to view, the label for the location, the data type located at the address, the location as either RAM or FLASH space, and a display preference of either decimal or hexadecimal. You can also select Global Variables.

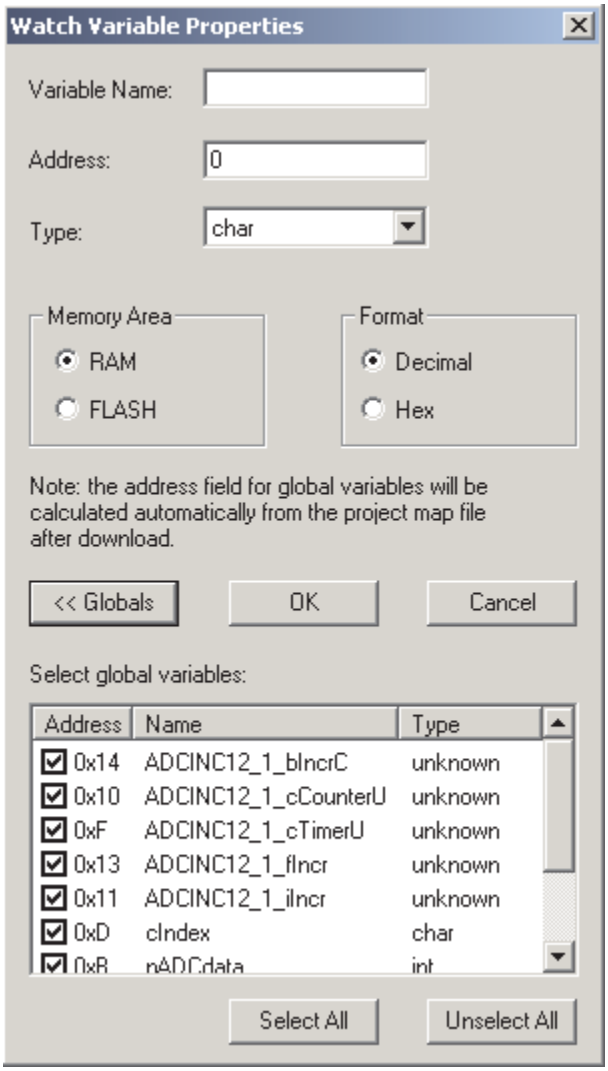


Figure 70: Debug ASM Watch Properties

Right-click Add, Delete, or Properties in the Watch/Global Name window to add, delete, or modify values. Note that if you change a variable type (or other settings in the window) and close the project, the next time you access that project the variable types and settings will be the same.

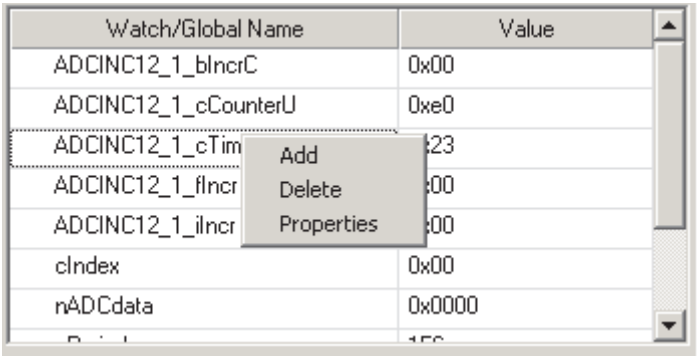


Figure 71: Watch/Global Name Window

10.4.5 Local Watch Variables

The Local Name window appears in the lower-right corner of PSoC Designer by default upon access of the Debugger subsystem.

For better viewing, you can click, hold, drag and drop the Local Name window anywhere inside the Debugger subsystem. This is also true of the Watch/Global Name window.

You will not see data in the Local Name window under the following conditions:

- Not connected to the ICE
- Connected to the ICE but have not downloaded the .rom
- Connected to the ICE and have downloaded the .rom but the program is “sitting” at the reset vector

Once you have a program with Local Variables and you have halted in a function in which they are contained, you will see all variable (names) in the Local Name window.

Following a program download, the default "radix" is set to Hexadecimal for all watch variable values, other than Float. Right-click inside the Local Name window to display the option, "Toggle Display Radix." Click to toggle the values in the appropriate radix (HEX or Decimal).

You can cut, copy, paste, and delete values (not names) by doubling-clicking to highlight then righting-clicking to choose an option.

The default action for using the “Delete” option is to set the value to zero (including Floating point types).

You cannot change Local Variable names.

You can adjust the column width of the Local Name window by dragging the heading row column dividers.

The Local Variables will not be "alive" when the program has halted at the initial function scope, for example:

```
void cgentest_009(void)
{------------------------- HALT, NO Locals
uInt32 u32var0;
uInt32 u32var1
 -.-.-.-.-.-.-.-.-.-. etc -.-.-.-.-.-.-.-.-.-.
```

Local Variables will be displayed in the Local Name window once the debugger halts at the first line of the function.

As discussed under 10.5 you can use the **Single Step** icon to step through the project .lst file.

10.4.6 Array Types Added to Global and Local Watch Variables

Array types have been added to both Global and Local Watch Variables. Array types can only be declared from 'C'. For example:

```
//-------------------------------
char sC[5];
signed char signedC[5];
int siI[5];
unsigned int uiI[5];
float fA[5];
long slL[5];
unsigned long uslL[5];
//-------------------------------
```

…shows declarations for all the supported array types.

The following shows the Watch/Global Name window after the array types were added:

Watch/Global Name	Value
sC	0x00, 0x00, 0x00, 0x00, 0x00
sil	0x0000, 0x0000, 0x0000, 0x0000, 0x0000
signedC	0x00, 0x00, 0x00, 0x00, 0x00
slL	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000
uil	0x0000, 0x0000, 0x0000, 0x0000, 0x0000
uslL	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

Figure 72: Array Type Variables

The elements are displayed horizontally, separated by commas in both the Watch/Global Name and Local Name windows. The radix can be changed from decimal to hexadecimal for all array types except "floats."

10.4.7 Dynamic Event Points Overview

The Events window is selectable from Debug >> Events and allows you to perform complex debugging by configuring conditional breaks and traces.

While breakpoints allow you to select a program location and halt, Dynamic Event Points provide multiple sequences of logical combinations and have multiple potential actions. Breakpoints allow you to select locations within a program to stop, look around, and determine, “how did I get here?” However, debugging is enhanced by the ability to stop and collect information about the target program based upon specified conditions. An example is the scenario “when variable **OutputV** gets set to zero, turn trace buffer on.”

Dynamic Event Points help simplify the debugging process by providing this capability. They monitor the processor to determine a match with logical operations of: Program Counter (PC), data bus, data address, instruction type, external logic signals, X Register, Accumulator, Stack Pointer, and Flags.

Typically, breakpoints have one logical input (PC) and one action (Break). Dynamic Event Points, on the other hand, differ from breakpoints in that they trigger actions when the specified logical condition occurs. An event point can trigger the following actions: break, turn trace on or off, decrement the input counter, initiate an external trigger, trigger the trace buffer, and enable an event sequence.

Dynamic Event Points can also be run in parallel. Several event threads can be selected to run in parallel or sequentially. The examples included in the Appendix A: Dynamic Event Points demonstrate this capability.

In summary, Dynamic Event Points provide you the ability to:

- Define complex breakpoints
- Characterize multiple test cases to be monitored and logically sequenced
- Perform any of the following actions: break, turn the trace on, turn the trace off, or set an external trigger

10.4.8 Configuring Events

Use the **Events** icon  to enable or disable event settings anytime during a debug session. To configure "events," execute the following procedure:

1. Access the Debugger Events dialog box at Debug >> Events.
2. Click your cursor in the first row, labeled "0," of the dialog box.
3. Below the rows, click a check to enable 8-bit Thread and/or 16-bit Thread.

Enabling both thread options brings the Combinatorial Operator field to life.

4. Once you have enabled the chosen thread, fill in the applicable thread fields (i.e., Low Compare, Input Select, High Compare, Input Mask), as well as state logic fields (i.e., Next State, Match Count).

Match Count can be used to specify the number of times an event task will occur before it performs the selected action.

As you make your selection in the Input Select drop-down, you will see details in the grayed-out, scrollable box below.

5. When finished, click **Apply**. The individual event is now configured and its information will appear at row "0."

If you forget to apply your entries, you will be prompted to save. Click **Yes** or **No**.

6. Click row "1" and repeat steps 3-5 to configure another event. Repeat this process for each additional event. (You can configure up to 65 events.)

To clear all events in the dialog box, click **Clear All**. To disable all events in the dialog box, click **Disable All**.

Click **Close** to exit the dialog box. All entries will be saved.

The input mask for 8-bit threads is applied to the high and low range comparison values as well as to the input select value.

This is done to support range comparisons on sub-sets of the bits in the input select value. All comparisons take place within the bits specified by the input mask. Other bits are ignored.

The range values are masked during event editing when the thread states are saved by the **Apply** button or by switching to a thread state. For example, if the entered low compare value is 06 hex and the input mask value is 05 hex, the low compare value after the mask is applied will be 04 hex. The input select value is masked at run time.

For complete training on debugging and Dynamic Event Points, try Tele-Training Module 3: Getting Started Debugging. Review and sign up under Support >> Tele-Training at http://www.cypress.com/.

As you “run” events, you can view messages regarding the status in the Debug tab of the Output Status window. For instance, if you check Break as part of an event, “Hit Event state break” will appear in the Output Status window as the debugger hits the event.

10.4.9 Typical Event Uses

There are many potential uses for events:

- Find a stack overflow

See Stack Overflow Errors under Invalid Memory Reference in PSoC Designer Online Help System at Help >> Help Topics.

- Detect `jmp` or `call` out of program

See Code that will Corrupt Stack under Invalid Memory Reference in PSoC Designer Online Help System at Help >> Help Topics.

- Trace a specific range of code
- Find when a register is written (with optional matching data value)
- Drive an external signal on interrupt(s)
- Measure interrupt latency
- Break the “n”th time a line of code is executed (match count)

- Break on Carry Flag status
- Break on signals from customer target board
- Wait for certain number of instructions
- Count sleep periods
- Break on specific data in Accumulator on certain instructions (PC)
- Collect trace-data reads or writes to specified register
- Find memory write

10.4.10 Event Examples

Following are a few pre-set examples for common events.

Find Memory Write

Let's say you want to break on a memory write to address 20h:

1. Access the Debugger Events dialog box at Debug >> Events.
2. Turn on the 8-bit thread by checking the "Enable 8-Bit Thread" box.
3. Pick "BITFIELD" in the Input Select box.

The little help box in the lower left describes the BITFIELD input. Bit 1 is the "ram write" bit.

4. We only want to look at bit 1. This means we'll need to mask off the other bits. The Input Mask field lets you knock off bits that you don't care about. It's an 8-bit number - set bits in the input mask to mark the bits that you care about (e.g., 0f to get the low four bits, ff to get all eight bits, 01 to get the low bit, 80 to get the high bit, c0 to get the top two bits, etc). Pick 02 for the input mask to get the ram write bit.
5. Any bits that are masked off need to be zeroes in the "compare" fields. You can see in the help box that the bit that we care about is "active low" - it is a 0 when the ram write is happening. This means we can just set both compare values to 0.
6. Turn on the 16-bit thread with the enable checkbox.
7. Pick "MEM_DA" for the input select. (If you picked MEM_DA_DB, you could check the address and the data value.)

8. Set both compare values to 20, your desired address.

9. Pick "AND" for your Combinatorial Operator.

10. Hit the "Break" checkbox. This will make the debugger halt when it sees the ram write.

11. Click **Apply**.

Stack Overflow

Let's say you want to create an event to break when the Stack Pointer reaches FF.

1. Access the Debugger Events dialog box at Debug >> Events.

2. Turn on the 16-bit thread by checking the "Enable 16-Bit Thread" box.

3. Set the Input select drop-down to SP.

4. Set both the Low compare and the High compare to values to "00FF."

5. Check the Break check box in the Static Logic fields.

6. Click **Apply**.

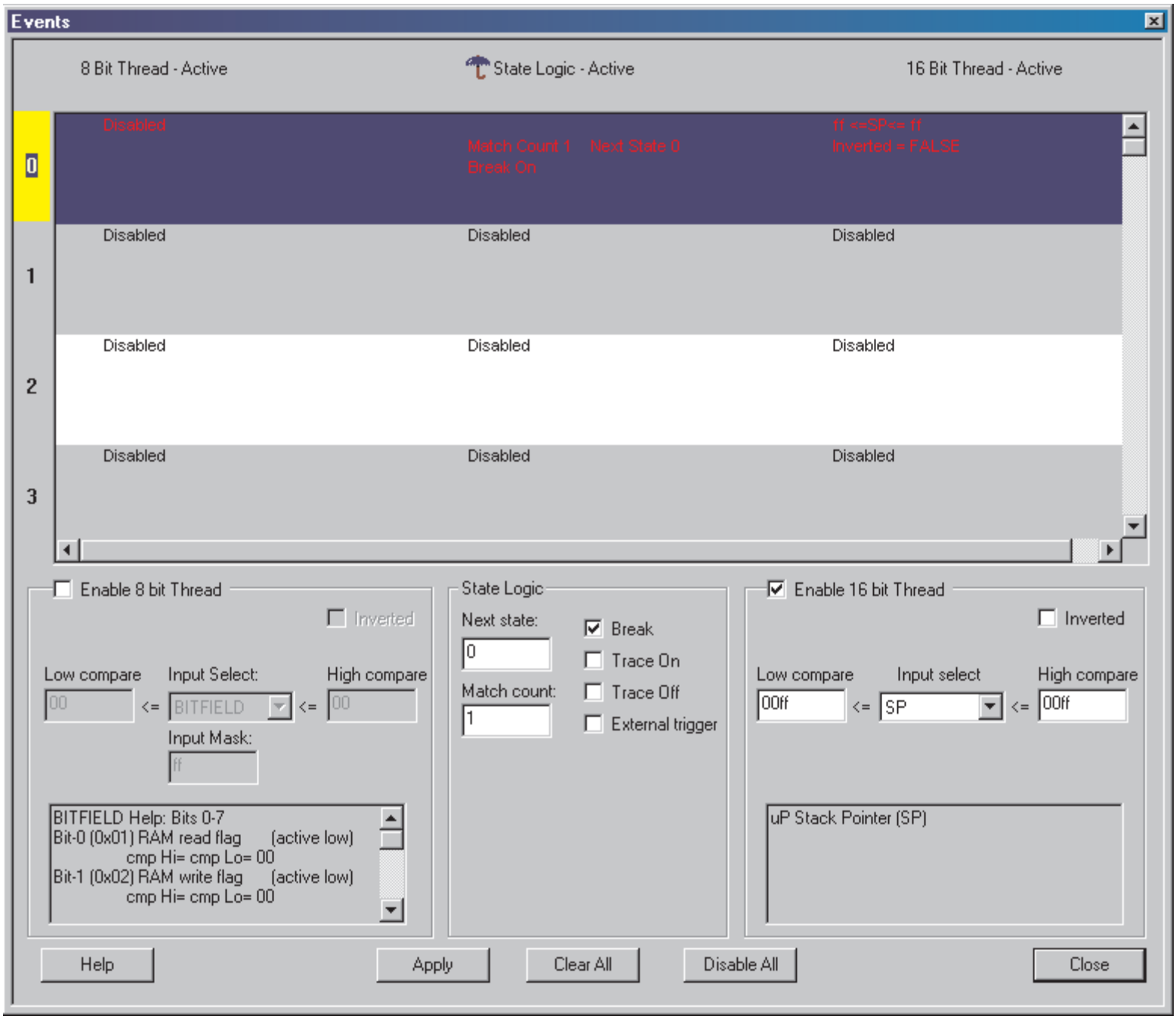


Figure 73: Stack Overflow Event

7. Click **Close**.

Register A Value, Trace On and Off, and Match Count

Let's say you want to create an event to turn the Trace Off at PC16=0000, turn the Trace On when Register A gets the value 0x32, and turn the Trace Off and Break after Register A gets the value 0x32 ten times.

1. Enter the Debugger subsystem.
2. Connect to the ICE and download target project .rom file.
3. Access the Debugger Events dialog box at Debug >> Events.
4. Click on the State 0 event.
5. Turn on the 16-bit thread by checking the "Enable 16-Bit Thread" box.
6. Set Input select to PC16, Low compare to 0000 and High compare to 0000.
7. Under State Logic set Next state to 1 and check Trace Off.
8. Click **Apply** to save.
9. Click on the State 1 event.
10. Turn on the 8-bit thread by checking the "Enable 8-Bit Thread" box.
11. Set Input select to A, Low compare and High compare to 32, and leave the Input Mask at ff.
12. Under State Logic set Next state to 2 and check Trace On and Break.
13. Click **Apply** to save.
14. Click on the State 2 event.
15. Turn on the 8-bit thread by checking the "Enable 8-Bit Thread" box.
16. Set Input select to A, Low compare and High compare to 32, and leave the Input Mask at ff.
17. Under State Logic set Next state to 3, the Match Count to 10, and check Trace Off and Break.
18. Click **Apply** to save.
19. Click **Close**.

10.5 Menu Options

The PSoC Designer Debugger toolbar is shown below:



Figure 74: Debugger Toolbar

Following, is a description of the debugging menu options:

Table 14: Debugging Menu Options

Icon	Menu/Tool Tip	Shortcut	Feature
	Debugger		Enables Debugger subsystem
	Connect		Connect PSoC Designer to ICE
	Download to Emulator		Download project .rom file to hardware emulator (Pod). This file holds all device configurations and source-code functionality
	Program Part		This programs the chip by placing and storing ROM data in the FLASH memory
	Start/Go	[**F5**]	Start debugger
	Stop/Halt	[**F6**]	Stop debugger
	Reset	[**Ctrl**] [**Shift**] [**F5**]	Reset device to 0 and restart debugger
	Step Into	[**F11**]	Step into next statement
	Step Out	[**Shift**] [**F11**]	Step out of current function
	Step Over	[**F10**]	Step over next statement
	Single Step		Single-step through .lst files of projects using both ‘C’ and assembly.
	Activate Trace	[**Ctrl**] [**F**]	Activate MCU-trace debugging feature
	Enable/Disable Events	Click drop-arrow and check option.	Enables or disables Event settings during debugging.
	Toggle Breakpoint		Toggles the breakpoint: Sets/removes user-defined breakpoints for use in the Debugger subsystem

10.6 Programming the Part

Programming the part occurs once debugging is complete. By doing this, you are storing the ROM data directly in the FLASH memory of the part. The Cypress MicroSystems device can be reprogrammed multiple times due to its FLASH Program Memory. Following is the Pod Programming Socket, which is connected to the CAT5 Patch Cable:

Only the five required serial programming pins are available on the Programming Socket. These required pins are the same pins that make up the Serial Programming Header (V_{cc}, V_{ss}, X_{RES}, P1[1] SCLK, and P1[0] SDATA). The Programming Socket cannot be used for emulation.

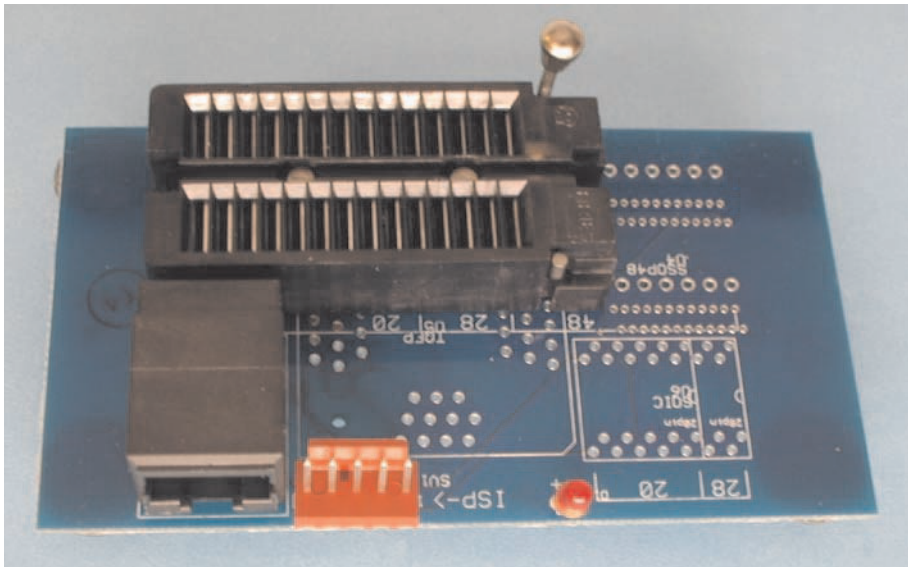


Figure 75: 28-Pin DIP Programming Board

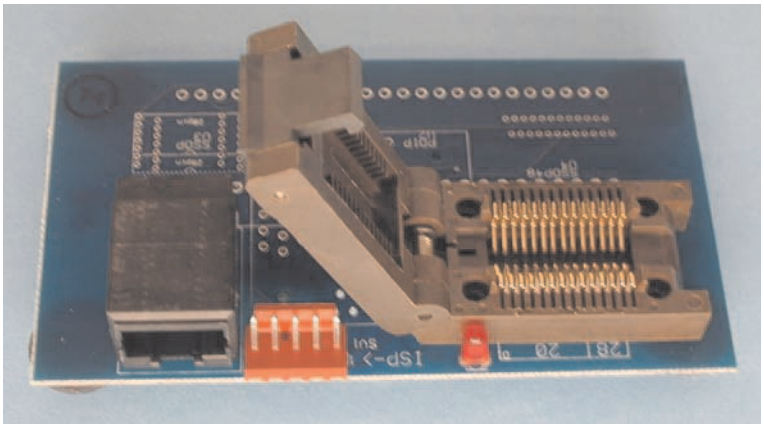


Figure 76: 28-Pin SOIC (Small Outline IC) Programming Board

Make sure the Pod is not connected to a circuit board (your development board or the PSoC Pup) when you program the part. Otherwise, programming (the part) may fail.

To program the part, perform the following steps:

1. Place the part in the Programming Socket on the Pod. (Note the position of Pin 1 on the Programming Socket to ensure correct operation.)

2. Click on the Program Part icon  and select the .rom file from the `...\output` folder of your project directory.

Alternatively, the device can be programmed on the target board using the Serial Programming Header on the Pod. The five connections that must be made from the Serial Programming Header to the pins on the target device are listed below:

Table 15: Header to Device Pin Connections

Header Pin	Device Pin
1	V_{cc}
2	V_{ss}
3	X_{RES}
4	P1 [1]/SCLK
5	P1 [0]/SDATA

It is important to note that there is a limit to the amount of current that can be supplied to the V_{cc} pin from the emulator Pod (500 mA at 5V). If you draw greater current through the V_{cc} pin on the programming header, this could damage the emulator. You must supply the connections on the target board for serial programming in the system.

Once part programming is complete, you can test it directly on your development circuit board.

Section 11. Flash Program Memory Protection

Users have the option to define the access to the Flash memory. Flash Program Memory Protection (FPMP) allows the user to select one of four protection modes for each 64-byte block within the Flash, based on the particular application. The protection mechanism is implemented using the System Supervisor Call instruction (SSC). When this command is executed, two bits within the data programmed into the Flash will select the protection mode. The following table lists the available protection options:

The *flashsecurity.txt* file for 2.xx and higher projects using Flash writes must be set to the correct protection modes. The defaults are set to full protect mode. In order to operate, blocks of Flash memory being used must be set to a mode which enables internal Flash writes to the designated blocks of memory. See 11.2 and 11.4 for details on modifying *flashsecurity.txt*.

Table 16: Flash Program Memory Protection Options

Mode Bits	Mode Name	External Read	External Write	Internal Write
00	Unprotected	Enabled	Enabled	Enabled
01	Factory Upgrade	Disabled	Enabled	Enabled
10	Field Upgrade	Disabled	Disabled	Enabled
11	Full Protection	Disabled	Disabled	Disabled

Mode Bit 10, Field Upgrade, is default.

11.1 FPMP and PSoC Designer

PSoC Designer now has a “rudimentary” mechanism that enables the user to set security modes in their Flash Program Memory. The security (or protection) can be set from within PSoC Designer on a “per” project basis.

A simple text file called “*flashsecurity.txt*” is used as the medium for the Flash security. This text file contains comments describing how to alter the Flash security. PSoC Designer validates the correctness of the Flash security data before it is used.

11.2 flashsecurity.txt and Application Editor

The FPMP file, *flashsecurity.txt*, is added to each new project and appears in the source tree.

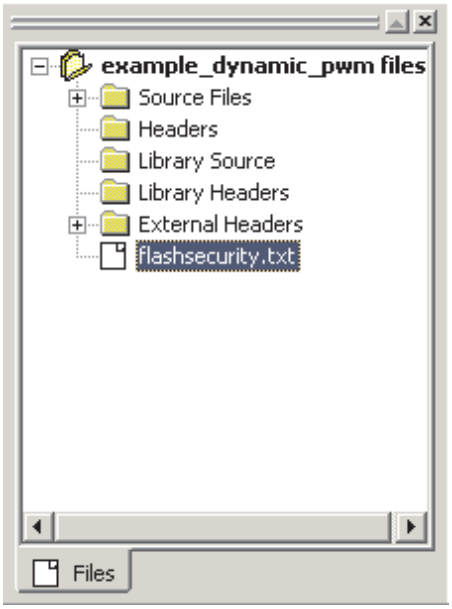


Figure 77: flashsecurity.txt in Source Tree

PSoC Designer will also add the FPMP file to a cloned project. This is especially useful when “cloning” projects that were created with earlier versions of PSoC Designer because earlier versions did not carry this feature. Note that if you do clone a project created in an earlier version of PSoC Designer, you will be prompted to update your project. See 2.5.

If, for some reason, *flashsecurity.txt* is missing or was deleted from the project, the default behavior is to apply Mode Bit 11 Full Protection to the entire program memory.

The following information contains instructions on modifying *flashsecurity.txt* and appears at the beginning of this file in PSoC Designer:

```
; Edit this file to adjust the Flash security for this project.
; Flash security is provided by marking a 64-byte block with a
; character that corresponds to the type of security for that block,
; given:
;
; W: Full (Write protected)
; R: Field Upgrade (Read protected)
; U: Unprotected
; F: Factory
```

```
; Note #1: Protection characters can be entered in upper or lower
; case.
; Note #2: Refer to the Flash Program Memory Protection section in the
; Data Sheet.

; Various parts with different Flash sizes can be used with this file.
; Security settings for Flash areas beyond the part limit will be
; ignored.
; Comments may be added similar to an assembly language comment, by
; using the semicolon (;) followed by your comment. The comment
; extends to the end of the line.
```

Following is an example of *flashsecuriy.txt*:

flashsecurity.txt

```
0 40 80 C0 100 140 180 1C0 200 240 280 2C0 300 340 380 3C0 (+) Base Address

W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 0
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 400
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 800
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address C00
End 4K parts
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 1000
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 1400
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 1800
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 1C00
End 8K parts
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 2000
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 2400
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 2800
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 2C00
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 3000
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 3400
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 3800
W  W  W  W  W   W   W   W   W   W   W   W   W   W   W   W ;    Base Address 3C00
End 16K parts
```

Figure 78: Snip-it of flashsecurity.txt

11.3 FPMP File Errors

If you edit or enter the wrong information in the *flashsecurity.txt* file, a dialog box will appear informing you of the situation when you are downloading to the ICE or programming a part. You will also see a message in the Build tab of the Output Status window.

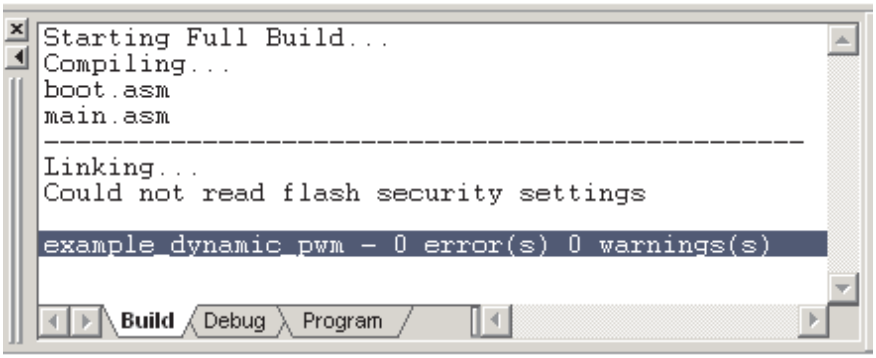


Figure 79: FPMP Error in Output Status Window

11.4 Example

If you have a Flash data table that can be changed using a Flash write routine, you might have assembly code that looks like this:

```
area Table (ROM, ABS)
org 3C80h
widgetTable:
export WidgetTable
db 57h ; W
db 49h ; I
db 44h ; D
db 47h ; G
db 45h ; E
db 54h ; T
; …. More table entries continue
```

You would then like to unprotect the Flash block associated with this table at address 3C80h. You would make your change in the *flashsecurity.txt* file as such:

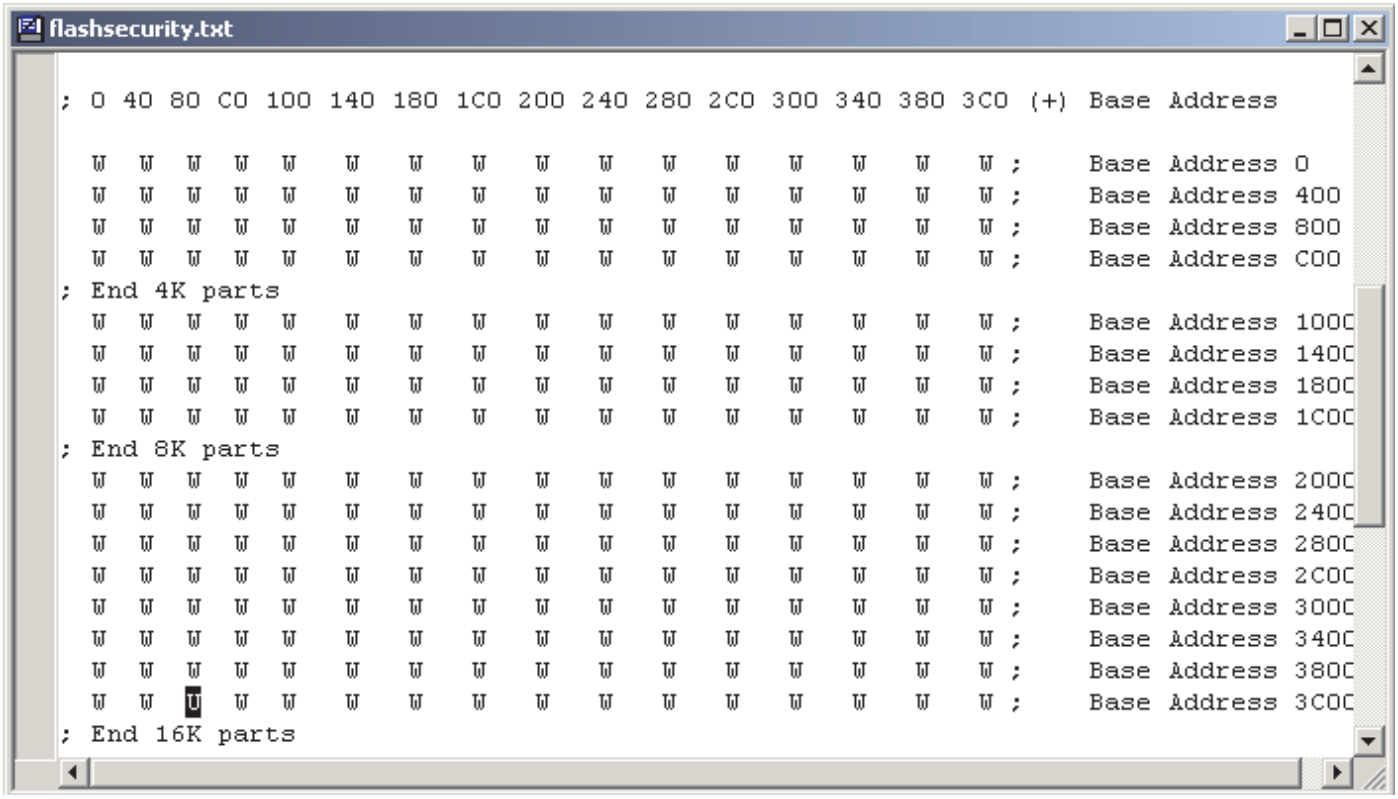


Figure 80: Unprotected Flash at 3C80h

Appendix A. Troubleshooting

Following are solutions for some potential system problems:

1. **During installation of PSoC Designer I receive an error message stating, "You can not expand the support files."**

   Possible Cause: This error message occurs by starting the installation process and not pressing any **Next** buttons, then starting the installation process again, resulting in two instances of the installation.

   Resolution: Proceed gracefully through installation to completion. You can uninstall PSoC Designer after a successful installation by running installation again and selecting Remove.

2. **Upon opening Application Editor after an “upgrade” installation of PSoC Designer, my screen (system window) is blank. The source tree and status window are not in view.**

   Possible Cause: This possibly occurs if you upgrade PSoC Designer on Windows 2000.

   Resolution: Access Application Editor . Click your cursor at the far-left of the screen and drag the source tree into view. Repeat at the bottom of your screen to drag the status window into view.

3. **I would like to install a PCI or PCMCIA parallel card to provide a Dedicated Parallel Port.**

   Symptom: For whatever reason, I want/need a dedicated parallel port.

   Possible Cause: Not applicable.

   Resolution: In the event that all efforts to get the onboard parallel port to work with the Cypress MicroSystems PSoC ICE have failed (or if you prefer a dedicated parallel port), adding an after-market parallel port is an alternative. Cypress MicroSystems has tested two parallel port cards with systems that have not connected using the onboard parallel port. Both cards have proven to work with these systems. One of these solutions is compatible with PCI-bus

based PCs and the second uses the PCMCIA port available on many portable computers.

Table 17: Parallel Port Options

PC Type	Port	Parallel Port Option
Desktop	PCI	SIIG®, Inc. Cyberparallel PCI Model IO1839, Part# JJ-P00112 http://www.siig.com
Portable	PCMCIA	Quatech, SPP-100 Enhanced Parallel Port Type II PCMCIA Card http://www.quatech.com

Follow the manufacturer instructions to install and configure the parallel port to EPP mode. Both of these cards include drivers that support Windows 98/ 98SE/NT/Me/XP and 2000.

PSoC Designer version 2.16 or later is required to make use of a second parallel port. If the parallel port card is installed as LPT2, it must be designated from within PSoC Designer. To select an alternate parallel port, perform the following:

1. Click Tools >> Options.

2. Inside the Options dialog box, select the “Debugger” tab.

3. Use the drop-down menu labeled “ICE board debug port connected to” to select the correct port. In most cases, the default onboard parallel port will be “LPT1” and the additional port that you just installed will be “LPT2.” Select the appropriate port, most likely “LPT2.”

4. After the correct port is selected, press **OK** and try to connect the ICE in using the “Connect” icon .

If the ICE still does not connect, make sure the ICE is connected to the correct parallel port and the Pod is connected to the ICE. Also, verify that the parallel port was installed correctly per the manufacturer instructions. The PC may need to be restarted after installation of the parallel port. If the PC is restarted, verify the correct parallel port is selected when re-entering PSoC Designer.

Appendix B. Glossary

The following system and industry terminology is used throughout the PSoC Designer suite of product documentation.

Table 18: Terminology

Term	Definition
Active Windows	Subsystem-related windows that are open and workable
Analog PSoC Blocks	Basic programmable op-amp circuits. There are SC (switched capacitor) and CT (continuous time) blocks. These blocks can be interconnected to provide ADCs, DACs, multi-pole filters, gain stages, and much more
API	Application Programming Interface. APIs for source programming are created during application code generation in Device Editor
Application Editor	PSoC Designer subsystem where users edit and program C Compiler and assembly-language source files
Assemble (Combined with Compiling)	Assembling, in PSoC Designer, translates all relative-addressed code into a single .rom file with absolute addressing
Build/Link	Building your project in PSoC Designer links all the programmed functionality of the source files and loads it into a .rom file, which is the file you download for debugging and programming
Compile (Combined with Assembling)	Compiling, in PSoC Designer, takes the most prominent, open file and translates the code into object source code with relative addresses
Debugger	PSoC Designer subsystem where users debug and perfect project functionality
Design (Export/ Import)	One or more loadable configurations that can be exported from a project then imported and used in a new or existing project. A loadable configuration consists of one or more “placed” User Modules with module parameters, Global Resources, set pin-outs, and generated application files.
Design Browser	Venue to identify re-usable designs for import to PSoC Designer projects
Device Editor	PSoC Designer subsystem where users choose/configure their device

Table 18: Terminology, continued

Digital PSoC Blocks	8-bit logic blocks that can be given a personality. The personality can be to act as a counter, timer, serial receiver, serial transmitter, CRC generator, pseudo-random number generator, or SPI.
Dynamic Re-configuration	Dynamic Re-configuration allows for microcontroller applications to dynamically load and unload configurations. With this feature, your single PSoC MCU can have multiple functions
Family of Devices	CY8C25xxx/CY8C26xxx family of devices consists of five pin-outs; 8, 20, 28, 44, and 48
ICE	In-Circuit Emulator that allows users to test the project in a hardware environment (Pod) while viewing and debugging device activity in a software environment (PSoC Designer)
IDE	Integrated Development Environment (for PSoC Designer)
ISR	Interrupt Service Routine. ISR shells for source programming are created during application code generation in Device Editor
Link/Build	Linking your project in PSoC Designer links all programmed functionality of the source files (with absolute addressing) and loads it into a .rom file, which is the file you download for debugging and programming
M8C	Enhanced 8-bit microprocessor core (CY8C2xxxx family of devices) that supports 8-bit operations and is optimized to be small and fast
MCU	Acronym for microcontroller unit
Pod	Part of the ICE that emulates functionality, in which debugging occurs
PSoC™	Programmable System-on-Chip
PSoC Blocks	Analog and digital peripheral blocks of a device that are customized by the placement and configuration of User Modules
PSoC Designer	Integrated Development Environment for Cypress MicroSystems' Programmable System-on-Chip technology
Source Tree	Project file system displayed by default in left frame of Application Editor
Subsystem	PSoC Designer has three subsystems; Device Editor, Application Editor, and Debugger
USB Dongle	New port connection to work the ICE in PSoC Designer v. 4.1.
User Module	Accessible, pre-configured function that once placed and programmed will work as a peripheral in the target device

Index

Symbols

"Make" Utility 32, 140

A

APIs and ISRs 102
Application Editor 125
  Adding Files 130
  Additional Generated Files 127
  File Definitions and Recommendations 125
  flashsecurity.txt 171
  Full Application File Search 132
  Modifying Files 129
  Removing Files 131
  Standard grep 132
Assembler 135, 135
  "Clean" Compile/Assemble/Build 140
  Accessing 135
  Address Spaces 135
  Addressing Modes 136
  Assembler Directives 138
  Compiling/Assembling Files 140
  Destination of Instruction Results 137
  File Syntax 137
  Instruction Format 136
  Instruction Set 139
  List File Format 138
  The Microprocessor 135
Assembler Directives 3, 138

B

boot.asm 106
Builder 141, 141
Building a Project 141

C

C Compiler 142
Clone Project 53
  Change Parts 53
  Migrate from Part 53
  Move Project 54
Configuration Data Sheet 100, 107
Configuration Methods 53
Create a Project 49
Create New Project 53
Custom boot.tpl 18

D

Database Version Detection 17
Debugger 145
  Breakpoints 154
  Configuring Events 160
  Connecting the Hardware 148
  Connecting the Software 150
  Connecting to the ICE 147
  CPU and Register Views 155
  Debug Strategies 152
  Downloading to Pod 151
  Dynamic Event Points 159
  Header to Device Pin Connections 168
  ICE Front Panel 147
  Local Watch Variables 157
  Menu Options 166
  Programming the Part 167
  Trace 152
  Trace Log Entries 154
  Watch Variables 156
Debugger Components 145
Dedicated Port Card 44
Design Browser 113
Design Catalog 115
  Auto-resolve 115
Design Rule Checker 99
  Running Design Rule Checker 100
Design-Based Project 54
Development Kit 146
Device Editor 59
  Interrupt Vectors 103
Device Power Cycle Programming Mode 42
Device Reset Programming Mode 42
Documentation Conventions 2
Dynamic Re-configuration 109
  Add Configuration 109

Application Editor 121
Code Generation 118
Debugger 121
Delete Configuration 115
Design Browser (Import) 113
Export 111
Export and Import Design Configurations 111
Global Parameters 116
Pin Settings 116
Port Pin Interrupts 117
Rename Configuration 116

E

Edit Windows 31
Enable Output File Tree 46
Enabling the Compiler 45
Event Examples 162
Find Memory Write 162
Register A Value, Trace On and Off, and Match Count 165
Stack Overflow 163

F

File Types and Extensions 21
Headers 24
lib (Librarian) File Folder 21
Library Headers 24
Library Source 21, 24
obj (Objects) File Folder 21
output File Folder 21
Source Files 24
Flash Program Memory Protection 170
Flash Program Memory Protection Options 170

G

Generating Application Files 100
Global Variables 156
Glossary 177

I

ICE
Connecting to the ICE 147
ICE Front Panel 147
Local Watch Variables 148, 157
Power 43
Insert Spaces Instead of Tabs 46
Installation 7
Installing the System 9
Interconnect View 70
Interconnections
Analog Clock Select 81
Analog Column Clock 82
Analog Column Input Mux 82
Analog Column Input Select 82
Analog Output Buffer 82
Comparator Analog LUT 84
Connection to Global Input 85
Connection to Global Output 88
Digital Interconnect Row Input Window 85
Digital Interconnect Row Output Window 86
Global In 81
Global Out 81
Row Broadcast 84
Row Logic Table Input 86
Row Logic Table Select 87
Selection of ACMux, BMux, AnalogBus and CompBus for a SC Analog Block 84
Selection of Clock Input for a Digital Block 83
Selection of Enable Input for a Digital Block 83
Selection of NMux, PMux, AnalogBus and CompBus for a CT Analog Block 84
Selection of Output for a Digital Block 83
Selection of RBotMux for a CT Analog Block 84
Synchronization 86
Internal Registers 3
Introduction 5
ISRs 103

L

Librarian 144
Librarian, Project Settings
Additional Library Paths 39, 39
Library Headers 144
Library Source 144
Linker, Project Settings 38
Additional Library Paths 39
Creating a Library 40
Default Memory Organization 39
Enable 24 MHz Alignment Shift 41
Object/Library Modules 39
Relocatable Code Start Address 38

Silicon Errata Warnings 41
Linker/Loader 143
Customized Linker Actions 143

N

Navigating Device Editor 59
Notation Standards 3

O

Output Status Window 32
Output Tab 25
Output View 29

P

Pan/Zoom Toolbar 62
Pod Error 152
Pod Uses External Power Only 43
Port Connection 92
Port Connections
Analog Input 92
Analog Output Buffer 92
Default Input 93
Ext Ref 95
ExternalGND 95
Global_IN_x 93
Global_OUT_x 93
I2C SDA 95, 96
StdCPU 94
XtalIn 94
XtalOut 94
Port Drive 96
Port Interrupt 97
ChangeFromRead 97
DisableInt 97
FallingEdge 98
RisingEdge 98
Product Upgrades 6
Project
Creation 49
Project Backup Folder 57
Project Compatibility 17
Project Manager 25
Project Settings 33
Debugger 43
Device Editor 37
ImageCraft Compiler 33
Linker 38
Programmer 42
Project Settings, Device Editor
Configuration Initialization Type 38
Enable Configuration Name Prepending to Pin Name 37
Interrupt Generation, Enable 37
Project Update
User Modules 46
Project View 25
Purpose 5

R

Re-activate Document 46
Requirement Checklist
Hardware 7
Performance Factors 8
Software 8
Resize Windows 27
Resource Manager 70

S

Save Options 46
Section Overview 6
Setting Bookmarks 155
Shadow Registers 127
Source Files Generated by Generate Application Files 101
Support 5
System Interface 21

T

Toolbars 47
Tools Options 44
Builder 44
Compiler 45
Debugger 45
Design Rule Checker 47
Device Editor 46
Editor 46
Toolbars 47
Trace Log Options 45
Troubleshooting 175
Dedicated Parallel Port 175
Two-Minute Overview 1

U

User Module Selection View 64
User Modules
Clear Placements 72
Deploying Interconnectivity 79
Global Resources 78
Interconnect View 90
Name and Rename 72
Next Allowed Placement 72

Parameters 77
Pin-out View 90
Placing 70
Print Configuration 72
Remove 70
Restore Default Pin-out 96
Restore Global Resource Default Settings 79
Selecting 64
Specifying Pin-out 89
Toolbar 67
Tracking Device Space 98
Undo Place 72

V

Version Control System 23

W

Watch Variables
Array Types 158
Global 156
Local 157
Window Options 31
Window Persistence 46
Write-Only Register Shadows 127

Document Revision History

Document Title: PSoC Designer: Integrated Development Environment User Guide **Document Number**: 38-12002				
Revision	**ECN #**	**Issue Date**	**Origin of Change**	**Description of Change**
**	115168	4/23/2002	New release of PSoC Designer version 3.20.	New document to CY Document Control (Revision **). Revision 1.16 for CMS customers.
*A			New release of PSoC Designer version 3.21.	Updates to export configurations and Design Browser (import).
*B			New release of PSoC Designer version 4.0.	--Code Compressor --UI windows for Device Editor --Dynamic Rule Checking --Event Examples --Debugger features --Update system requirements
*C			New release of PSoC Designer version 4.1.	--USB Dongle...
Distribution: External/Public **Posting**: None				