Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



THIS SPEC IS OBSOLETE

Spec Number: 001-35940

Spec Title: PSoC® 1 Migrating and Cloning Projects in PSoC Designer™ 5.2 – AN2131

Sunset Owner: RJVB

Replaced By: None

PSoC® 1 Migrating and Cloning Projects in PSoC Designer™ 5.2

Author: Andrew Best

Associated Project: No

Associated Part Family: CY8C20xxx, CY8C21xxx,CY8C22xxx, CY8C23xxx, CY8C24xxx, CY8C25xxx, CY8C26xxx, CY8C27xxx,CY8C28xxx, CY8C29xxx

Software Version: PSoC® Designer™ 5.2

Related Application Notes: None

OBSOLETE

Abstract

AN2131 covers the process of cloning a project in PSoC® Designer™ to migrate to a different PSoC 1 family. This Application Note walks you through the steps of cloning your project and provides some tips for getting your project up and running. Special detail is given for converting your project from a CY8C25xxx/26xxx to the CY8C27xxx, which is a popular use for this process.

Introduction

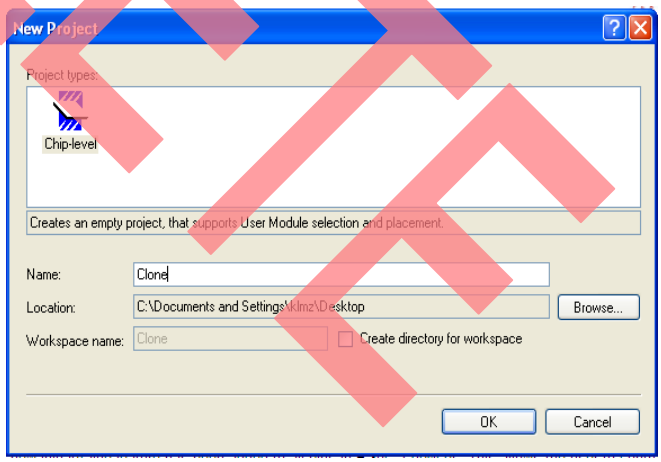
Enhancements to newer PSoC families, the need for more resources, or the need to update a project to the latest PSoC Designer version may have convinced you to migrate your project to a new part. If so, you are going to need to clone. Cloning a project creates a new project in a separate directory and copies your existing loadable configurations and source code. Placed User Modules and their parameters, block and global connections, and pin-drive settings remain intact as much as is possible in the cloning process. There may, however, be some User Module parameters and interconnections that do not map directly from part to part. Step 2 provides some ways to painlessly re-construct these settings.

Warning It is always recommended that you backup your project before cloning or opening the project in a new version of PSoC Designer. You need to copy the entire project folder to another location or create a .zip file archive of the project folder. Printing the Configuration Data Sheet and .lst file is also recommended for archiving your project configuration and source code. At the bottom of the .lst file you will find the version number of PSoC Designer in which the project was created. PSoC projects are not backwards compatible and there is no guarantee that your project will work in an older version of PSoC Designer once it has been opened in 5.2.

Step 1: Creating a Project Clone

In PSoC Designer 5.2 select New Project from the File menu. Type the name of your new project clone and browse to the directory where you want the project. Click on the OK button.

Figure 1. New Project Dialog Box



On the next dialog box browse to the .SOC file of the project you would like to clone. You may then select your new target device by clicking on the View Catalog button.

Figure 2. Parts Catalog

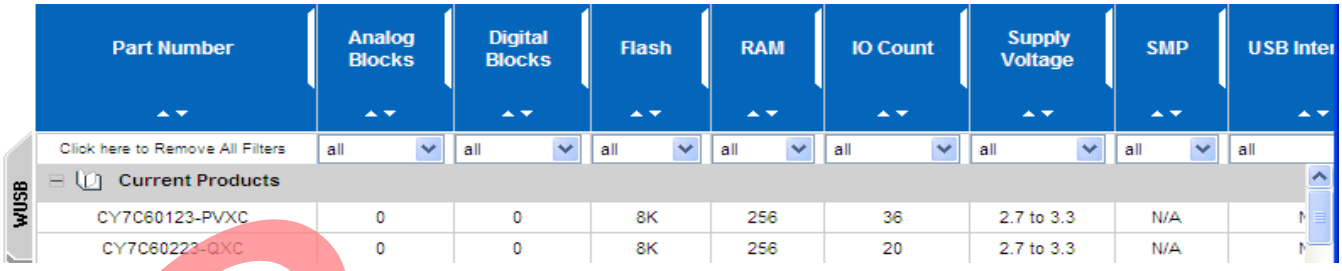
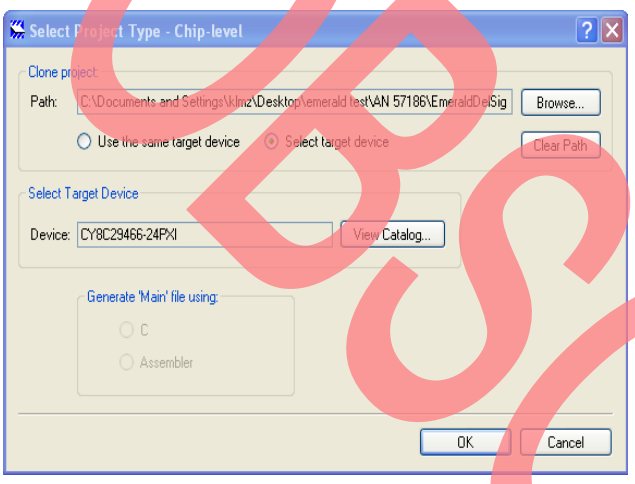


Figure 3. Existing Project Path



Clicking on the View Catalog button opens the Parts Catalog dialog box. It provides an easy way to compare the specifications of the PSoC parts such as the IO pin count, number of configurable blocks, and the amount of code space.

After you have selected the part, click on the OK button and the cloning process will take place. This process could take a few minutes with large projects.

If you are cloning from an older version of PSoC Designer, a message box will appear informing you that the project was created in an old database version. You need to update your project to get the correct *boot.tpl* file.

Select the 'Replace the boot.tpl' radio button. A new *boot.tpl* file will be placed in the project directory and the old file will be moved to the `\backup` directory the first time you generate source.

PSoC Designer 5.2 support custom code banners that define where the custom code has been written. These banners allow the software to automatically copy these customizations when performing future updates by selecting *Preserve Custom Code Sections*.

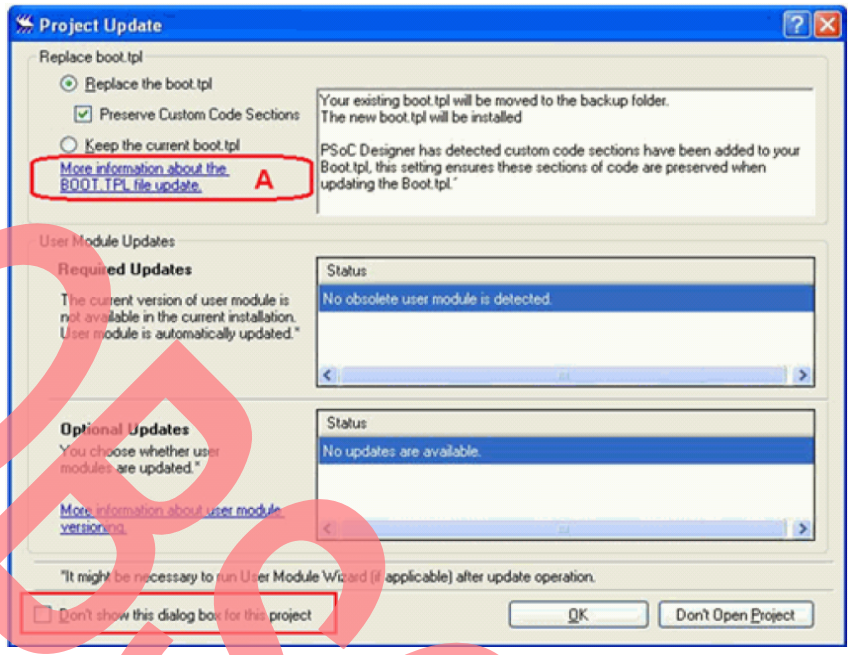
The following example shows how to migrate customized code from your existing boot.tpl file to the updated version. This step will require some manual edits to your existing file, but will result in a more robust file for future updates.

NOTE: putting the merge section banners at wrong line in original boot.tpl may result in broken boot.tpl after update.

1. During project update select 'Don't Open Project' button.
2. Go to project directory.
3. Open Boot.tpl file in text editor.
4. Add the '@PsoC_BOOT_ISR_UserCode_START@' banner at the start *and '@PsoC_BOOT_ISR_UserCode_END@' banner at the end of your custom ISR code as shown in the following example:*

OBSOLETE

Figure 4. Old version update dialog




```
IF (TOOLCHAIN & HITECH)
; jmp __Start ;C compiler fills in this vector
ELSE
jmp __Start ;First instruction executed following a
Reset
ENDIF
;@PsoC_BOOT_ISR_UserCode_START@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
org 04h ;Low Voltage Detect (LVD) Interrupt Vector
halt ;Stop execution if power falls too low
org 08h ;Analog Column 0 Interrupt Vector
`@INTERRUPT_2`
reti
;
; other contents of ISR Table
;
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PsoC_BOOT_ISR_UserCode_END@ (Do not change this line.)
```

5. Add the '@PsoC_BOOT_LOADCFG_UserCode_START@' banner at the start and '@PsoC_BOOT_LOADCFG_UserCode_END@' banner at the end of your custom user code, but before LoadConfigInit function as shown in the following example:

```
;@PsoC_BOOT_LOADCFG_UserCode_START@ (Do not change this line.)
;---------------------------------------------------
; Insert your custom code below this banner
;---------------------------------------------------
;---------------------------------------------------
; Insert your custom code above this banner
;---------------------------------------------------
;@PsoC_ BOOT_LOADCFG_UserCode_END@ (Do not change this line.)
;-------------------------
; Load Base Configuration
;-------------------------
```

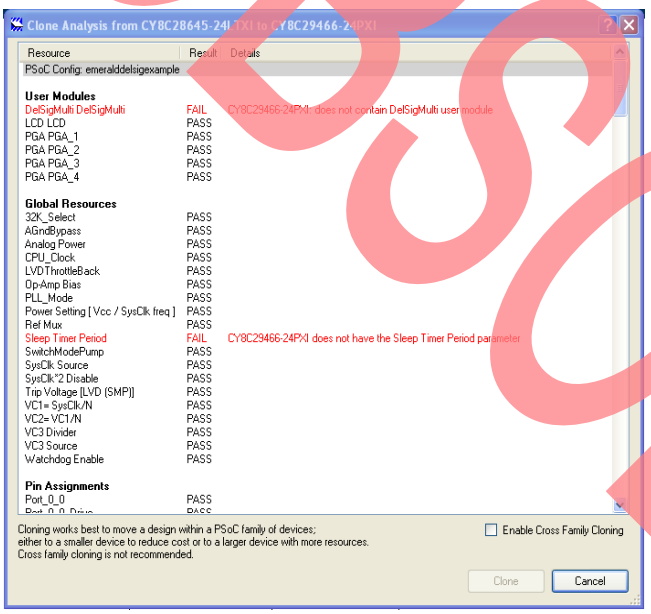
OBSOLETE

```
; Load global parameter settings and load the user modules in the
; base configuration. Exceptions: (1) Leave CPU Speed fast as possible
; to minimize start up time; (2) We may still need to play with the
; Sleep Timer.
;
lcall LoadConfigInit
; other boot.tpl code
```

6. Save Boot.tpl file in text editor.
7. Open project again and choose 'Preserve Custom Code Sections'.

If "Preserve Custom Code Sections" option is selected, the existing boot.tpl will be updated automatically to the newer version and code located in the merge sections is preserved. The old version of boot.tpl is copied to a folder named "Backup" and file name of backed-up boot.tpl includes the timestamp when update was done.

Figure 4. Clone Analysis Dialog



The Clone Analysis dialog, shown above, will be shown when you are cloning a project to a different target device. This dialog will show you what modules the cloning process will be able to transfer and which ones are not available in the target part. The reason for the module not passing the cloning process is also given. To continue check "Enable Cross Family Cloning" and hit Clone. All of the outdated User Module API source files will automatically be updated in your project the first time you generate source. Because of the difference in resources between devices not all user modules will always be able to be cloned.

The **int.asm* files for each User Module will be replaced with newer versions and your original files will be moved to the \backup directory the first time you generate source. Any custom interrupt routine code that was in the **int.asm* file needs to be manually cut and pasted from the files in the `\backup` directory into the new versions. It is recommended that you copy only your custom-written code and paste it in between the commented headings for user code in the new **int.asm* file. This will assist in future updates so the updating can be done automatically.

Step 2: Verifying Project Settings

Any differences between the User Modules for the source part and the target part will cause parameters to be unset in the project clone. Also, the difference in location of the digital communication blocks could cause any communication User Modules to be "unplaced."

The parameters with red question marks could not be mapped during the cloning process and an appropriate value should be chosen. There are three typical methods of determining which User Module parameters need to be reset.

Method 1: Using the Configuration Data Sheet

Remember to backup your old project if it has never been opened in 5.2. Open your original project and select Configuration Data Sheet from the View menu. This Data Sheet shows all of a User Module parameters, global resource settings and register values that are written for each of the configurations in your project. You can print this data sheet and use it to compare the settings in the project clone to your old settings for each User Module. The pin settings and global parameters should be verified as well.

Once it has been generated, the configuration datasheet is self contained in its own folder and can be viewed independently of PSoC Designer by opening *configreport.xml* in Internet Explorer.

Method 2: Using the Design Rule Checker

The Design Rule Checker (DRC) is a new feature that can be used to detect unset or invalid settings in your project. The DRC will run once automatically when you clone your project and it can be accessed at any time from the Tools >> Add in Tools menu. Rules of the following categories will be tested on the project: Top Level, Device Family, Base Device, and User Module with severity level from 1 to 5. You can customize which rules are performed under the Design Rule Checker tab under Tools >> Options.

Note Running the Design Rule Checker does not change any settings or code in your project; it simply provides a notification of potential problems with your project. For unset User Module parameters you will receive a message such as the following:

```
Level 5 Warning - Configuration
MagCardReader, User Module BitTimer_T1:
Clock value has not been initialized
```

OBSOLETE

Method 3: Side-by-Side Projects

With your new project clone open, open another instance of PSoC Designer then access your original project. The two projects can be compared and you can see exactly what was done during the cloning process.

Project Migration Tips

Interrupt Service Routines

If the User Module **int.asm* files for your project have not been updated you will have compilation errors because the name of the routine has changed or has had an underscore prepended to the name. To replace any User Module file in your project with the current version, simply right click on the file icon in the file source tree and select Remove from Project. Removing a file from a project automatically moves the file into the `\backup` directory. Clicking the Generate Source icon will add the latest version of the file to your project. Any custom interrupt routine code that was in the **int.asm* file needs to be manually cut and pasted from the files in the `\backup` directory into the new versions. It is not recommended that you use your old **int.asm* file directly but you may do so by appropriately changing the name or prepending an underscore.

Include File Name Changes

The old version of your project may have had include files named *ProjectNameGlobalParams.inc* and *ProjectNameAPI.h*. These have changed to *GlobalParams.inc* and *PSoCAPI.h* in newer versions of Designer.

You will need to change the file names wherever they were used in your source files. Also, there is a new file generated called *PSoCAPI.inc* that mirrors *PSoCAPI.h*.

Changes to *boot.tpl*

As previously stated, custom changes that you made to *boot.tpl* need to be manually migrated from the `\backup` directory or using the method suggested in Step 1 to retain the custom sections of the boot.tpl. Do not directly use your old *boot.tpl* file under any circumstances.

Changes to Register Names and Masks

Some register names are different between parts. For example, the CMP_CR register in the CY8C25xxx/26xxx is called CMP_CR0 in the CY8C27xxx. Because of this, if your migrated project accesses register with changed names, syntax errors will result.

Clock Sync Parameter

The Clock Sync User Module parameter is very important and it must be chosen correctly for your User Modules to function. Use the following guidelines to select the appropriate value, depending on what you are using as the clock source for the block:

Use SysClk Direct

This setting will clock the block directly from the SysClk (24 MHz). It is the only way to use the SysClk (24 MHz) to clock a PSoC block. Trying to use VC1/1, VC2/1 with VC1/1, or VC3/1 is not allowed.

Sync to SysClk

This setting is for any SysClk (24 MHz)-based clock. Use this setting for VC1, VC2, and VC3 driven by SysClk, as well as clocking from any digital block with a SysClk-based source clock.

Also use this setting for the broadcast bus with a source based on SysClk or a row input or row output with a source based on SysClk. This is the most common setting.

Sync to SysClk*2

This setting is for any SysClk*2 (48 MHz)-based clock. Use this setting for VC3 driven by SysClk*2 as well as clocking from any digital block with a SysClk*2-based source clock. Also use this setting for the broadcast bus with a source based on SysClk*2 or a row input or row output with a source based on SysClk*2.

Unsynchronized

Use this setting for SysClk*2 (48 MHz) or any asynchronous clock input. Trying to use any other setting will not work for a SysClk*2 (48 MHz) clock source.

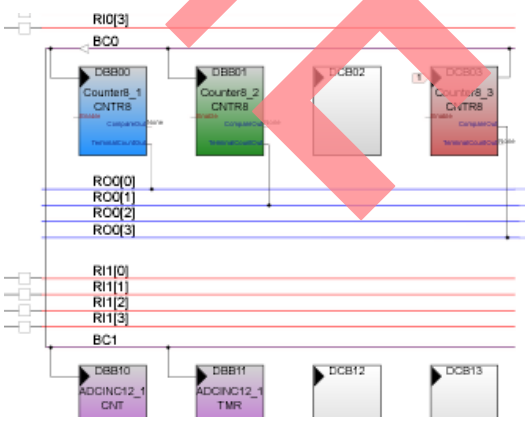
DBA03 Broadcast

Any User Modules that used the DBA03 Broadcast will no longer be connected. The best way to route clocks between digital blocks in the CY8C27xxx part is with the Row Broadcasts.

Click on the Row 0 Broadcast and select DCB03. Then for all of the User Modules that used the DBA03 Broadcast, select Row 0 Broadcast. The Row 0 Broadcast can be selected to drive the Row 1 Broadcast also by clicking on the Row 1 Broadcast.

You can then use the Row 1 Broadcast to route from DCB03 to the User Modules in Row 1.

Figure 5. Row Broadcast



PROFIT

Previous Has Been Removed

Any User Module parameter that was set to Previous will not be set in the project clone. For clarity, Previous has been replaced with the specific block name. You will need to set this parameter wherever Previous was used.

Change in Communication Block Location

The location of the communication blocks is different between some part families. The only way to resolve this difference is to swap the location of the User Modules that need communication blocks with the ones that do not need them. The increased interconnectivity of the more advanced part families should allow you to easily reconnect these User Modules. The old parameters for unplaced communication User Modules will be restored when you place the User Module in its new location.

VC3

24V1 and 24V2 are named VC1 and VC2 in the latest versions of PSoC Designer. VC3 is an additional clock resource that could come in handy in your project. Its source can be chosen as SysClk (24 MHz), SysClk*2 (48 MHz), VC1, or VC2. The divider can be chosen between 1 and 256 (see Clock Sync Parameter section for restrictions). If you are currently using a counter as a dedicated clock divider, you could probably use VC3 instead and free up an additional digital block.

Drive Modes

The latest part families have more settings for the pin drive modes and the default setting has changed from Pull Down to High Z Analog. Any unused pin or analog input should be set to Analog High Z for the drive mode. Details about the new drive mode settings are explained in GPIO section under the Core Architecture heading of the Device Family Data Sheets. The data sheets can be found under Help >> Documentation in PSoC Designer.

Pinout Differences

Pinouts may be different between part families. For example, Ports 3 and 4 have swapped pins in the CY8C27xxx parts with respect to the CY8C26xxx parts. This will affect any 44- or 48-pin project. It should not be a problem to change the routing because the row inputs and outputs can connect to either the odd or even global busses.

A change in pinout has also occurred in the 8-pin CY8C27xxx part. Pin 1 has changed from P0[7] to P0[5] and pin 2 has changed from P0[5] to P0[3]. This change was made so that all four Analog Output Buffers could be driven to pins whereas the Column 0 output buffer could not be used on the CY8C25122 part. If you were using the Column 1 output buffer to drive P0[5], you will need to change your project to use Column 0 with P0[3] or change the P0[5] wiring externally from pin 2 to pin 1.

Conclusion

The cloning feature of PSoC Designer makes the migration process very straightforward. With the increased interconnectivity and better analog performance of the new part families, you will never look back.

OBSOLETE

Document History

Document Title: PSoC® 1 Migrating and Cloning Projects in PSoC Designer™ 5.2 – AN2131

Document Number: 001-35940

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1514403	MKEA	09/27/2007	OLD APP. NOTE: Obtain spec. # for note to be added to spec. system. Update copyright. Add source disclaimer, revision disclaimer, Samples Request Form link, PSoC App. Note Index link. pdf has been stamped. **This note had no technical updates. There is an associated project but it was not updated.**
*A	3120306	MKEA	12/24/2010	Updated Software Version in page 1 as PSoC Designer™ 5.1 Generalized the information to all migration/cloning projects from specifically CY8C25xxx/26xxx to CY8C27xxx migration.
*B	3173031	KLMZ	02/14/2011	Updated the title, abstract, and deleted references and the associated files.
*C	3498937	MSUR	01/17/2012	Updated software version to PSoC Designer™ 5.2 in title and all other places. Updated Figure 4 and the corresponding section to reflect the changes in PD 5.2 for supporting Custom boot.tpl sections
*D	4241199	RJVB	01/09/2014	Obsolete document.

OBSOLETE

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

OBSOLETE

PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," and PSoC Designer are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

© Cypress Semiconductor Corporation, 2007-2014. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.
Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
Use may be limited by and subject to the applicable Cypress software license agreement.