AN205254

F²MC-16LX/FX, EasyCODE Integration into Softune Workbench v3

This application note describes the integration and usage of the third-party software EasyCODE with Cypress F²MC-16 Family SOFTUNE Workbench. The EasyCODE software can be used in Softune as an editor and debugger. You can use existing Projects and templates in combination with Softune and EasyCODE.

Contents

1 Introduction....1
1.1 Prerequisites....1
1.2 EasyCODE....1
2 Installation....3
2.1 Softune installation....3
2.2 EasyCODE installation....3
2.3 Softune configuration....4
2.4 EasyCODE configuration....6
3 Using Softune with EasyCODE....7
3.1 Editor mode....7
3.2 Debug mode....7
4 Information on the WWW....7
Document History....9

1 Introduction

This application note describes the integration and usage of the third-party software EasyCODE with Cypress F²MC-16 Family Softune Workbench.

The EasyCODE software can be used in Softune as an editor and debugger. You can use existing Projects and templates in combination with Softune and EasyCODE.

1.1 Prerequisites

You will need the following software components to complete the installation and configuration process:

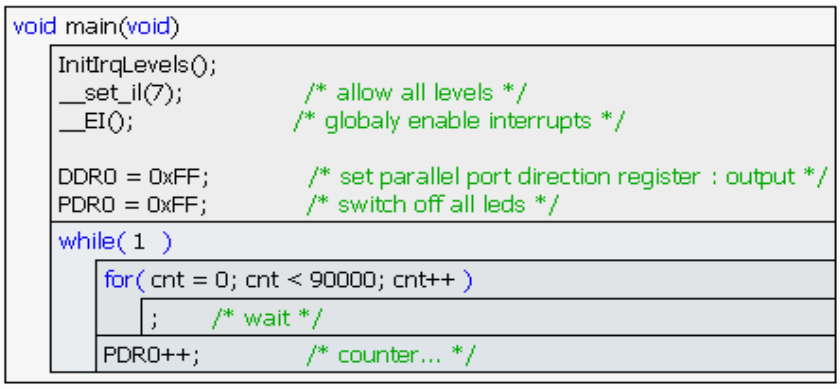
- Please asks your distributor for the latest Microcontroller DVD
- EasyCODE software
- EasyCODE communication module update
  - EasyCODE Website
    (Downloads->Integrations->Softune)

1.2 EasyCODE

EasyCODE is a way to program software projects with the help of graphical symbols. The goal of EasyCODE is to have a better overview about complex software projects. The structured view of small functions and even big programs should help you getting to the point.

The EasyCODE software uses the widespread structogram technique from the system development. The structogram is a graphical way of planning programs and functions. The code blocks are represented by adequate drawn blocks.S

Sample code snippet in EasyCODE and in a normal editor:



As you can see the function “main” is one big block that contains the sub-blocks. There is one directive Block containing the function calls for the Interrupts und Port I/Os. The while-loop is the next big block. It’s “L”-shape encloses the code which will be executed, while the loop is running. Inside the while-loop is a for-loop that also has an L-shape and also encloses the looped code. The directive-block containing “PDR0++” will not be looped by the for-loop, because it is outside of the L-shape, but it will be looped by the while-loop.

The same code snippet in a normal editor:

```
void main(void)
{
        InitIrqLevels();
        __set_il(7);                    /* allow all levels */
        __EI();                         /* globally enable interrupts */

        DDR0 = 0xFF;                    /* set parallel port direction register : output */
        PDR0 = 0xFF;                    /* switch off all leds */
        while(1)
        {
                for(cnt = 0; cnt < 90000; cnt++);       /* wait */
                PDR0++;                                 /* counter... */
        }
}
```

The coding process is a bit different in EasyCODE then in a normal environment. First you drag and drop the needed code block. Then you customize the block e.g. if-condition and so on. You can nest blocks by dropping them into a previously placed block, building the structogram of your application. EasyCODE will save the file as a plain c-file which then can be compiled.

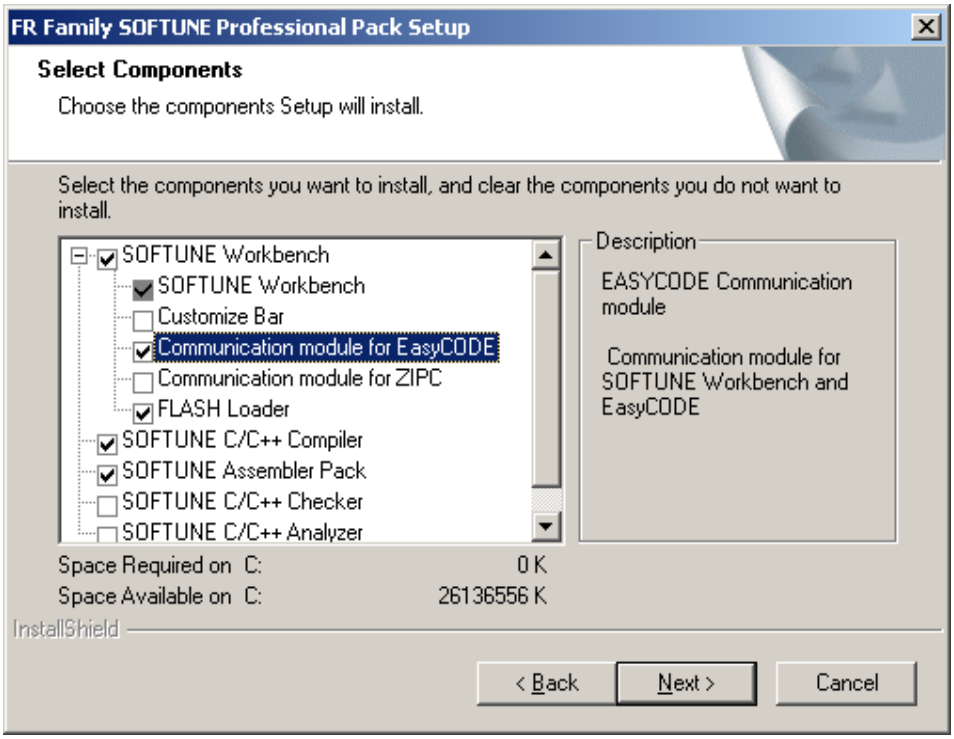
You can open and edit existing source files that you created earlier without EasyCODE. There are no conversions or additional information’s needed to start with these source files in EasyCODE.

But be informed that once edited and saved with EasyCODE your file formatting will not be the same as before.

2 Installation

2.1 Softune installation

Before you can use EasyCODE with Softune you will have to install Softune in a proper way. Start the Setup-Routine and proceed to the component selection. Now you have the opportunity to choose the needed software packages. For the integration process you will have to select the “Communication module for EasyCODE”.



Proceed with the installation and make sure everything works fine. If you don’t have already installed EasyCODE you will have to do this now.

2.2 EasyCODE installation

Refer to the EasyCODE manual for additional information on installing the software. For the integration into Softune you don’t need a special EasyCODE installation. A standard setup will work fine.

Next step is the installation of a “Communication module update” which is provided by EasyCODE. You can download it from their website: http://www.easycode-software.com/home.html

Look for a file called “d_int_softune.zip“, under Downloads->Integrations->Softune. Or use this direct link to the file: ftp://ftp.bkr.de/EasyCODE/integration/d_int_softune.zip

Unzip the contents and execute the setup routine. Install the update into your Softune root folder. This is e.g. “C:\Softune\”.

After a successful installation you must have a file called “Easy-cpp.ini” in your Softune root folder, “C:\Softune\bin\Easy-cpp.ini”.

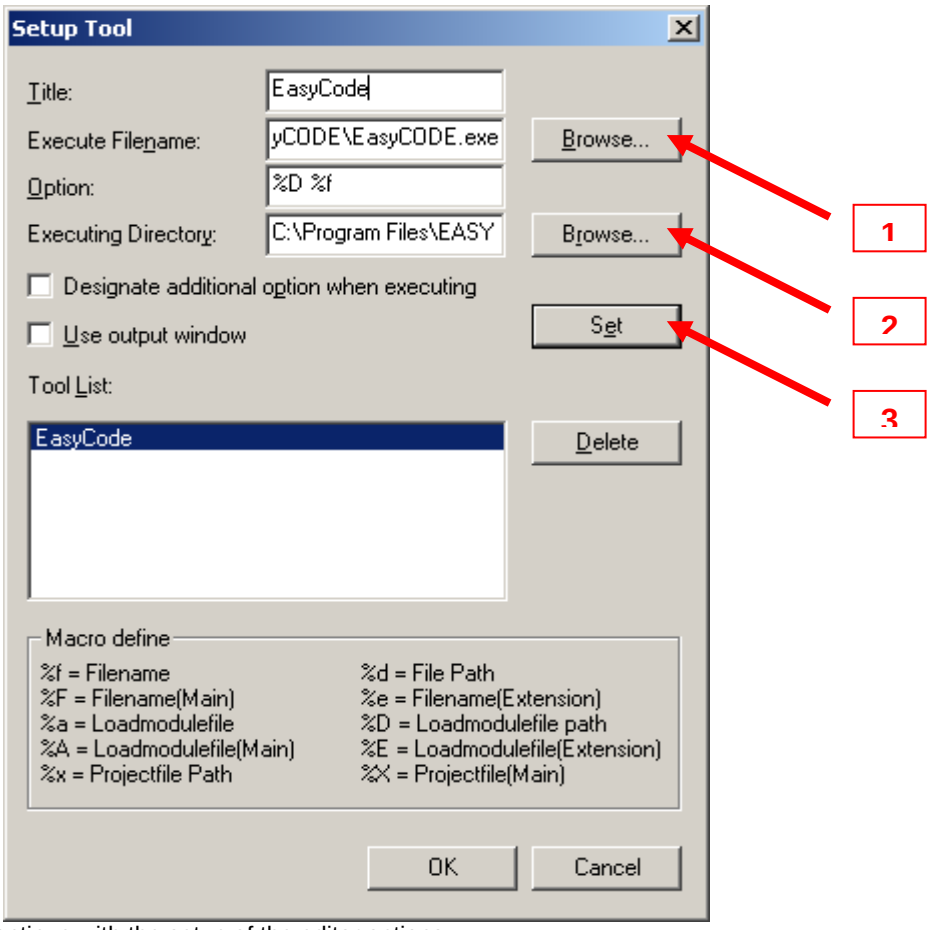
2.3 Softune configuration

Start your fresh installed Softune Workbench. You now will have to set up the EasyCODE software as an additional tool and as the standard editor.

Select Setup->Tool and create a new entry, in the tool List, called EasyCODE. To do so fill out the fields like in the screenshot.

Give the tool-profile a recognizable name e.g. EasyCODE. The “Execute Filename” should be the fully qualified path to your EasyCODE editor executable, e.g. “C:\Program Files\ EasyCODE\EasyCODE.exe”. The “Option” field holds the commands which are passed to the EasyCODE editor, “%D %f” is the correct setting for this. The “Executing Directory” is the directory where the Executable editor file is located. This should be the same directory like above, but without the filename added. You can simply use the Browse-button to locate the executable and the path.

Press the “Set”-Button to add and save the Settings you made.

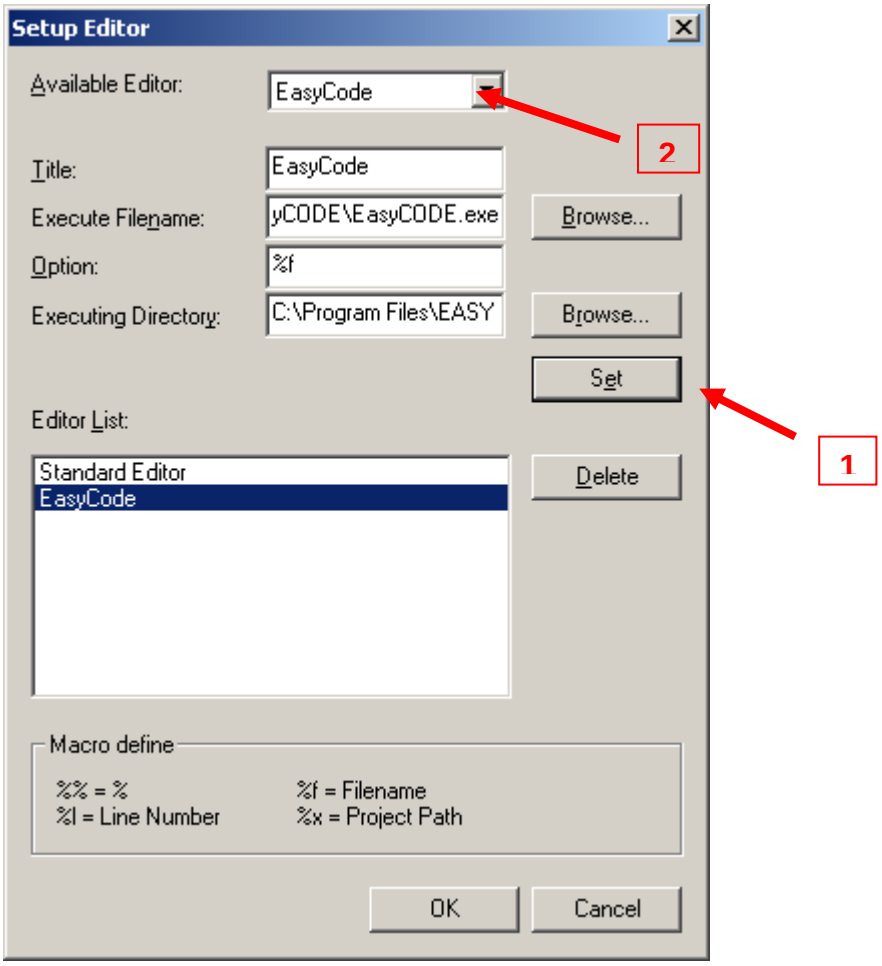


Leave this dialog and continue with the setup of the editor options.

The next step is the integration of the EasyCODE editor into Softune; this procedure will set the EasyCODE software as the standard editor.

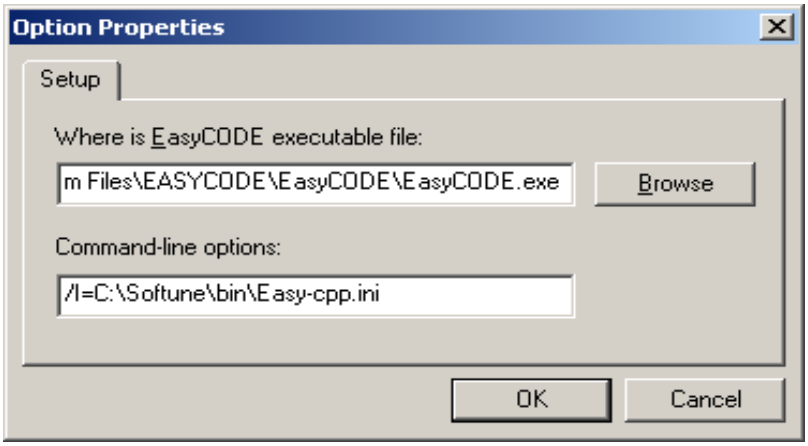
Select Setup->Editor and create a new Editor List entry. Fill out the dialog-fields like shown in the screenshot below.

Start with the title, give it a recognizable name like EasyCODE. For the “Executable Filename”, the easiest way is to press the Browse-button and browse for the EasyCODE executable in your EasyCODE installation directory, e.g. “C:\Program Files\EasyCODE\EasyCODE.exe”. The “Option”-field takes a “%f” which will pass the actual filename to the editor. The “Executing Directory” is the directory where the EasyCODE Editor is located. Use the Browse-button to get the location. This should be the same directory like above, but without the filename of the EasyCODE directory.



After you made the correct changes you will have to hit the “Set”-Button first, to insert the EasyCODE-Editor configuration to the List. After you have clicked the set-button you should be able to select the EasyCODE entry from the drop-down to make it your default editor. Quit the dialog to save your changes.

Last step to enable EasyCODE in the Softune Workbench is to configure the EasyCODE plug-in. Select “Project->EasyCODE->option” and add the path to your EasyCODE editor. Press the Browse-button and search for the EasyCODE editor executable in the EasyCODE installation dir, e.g. “C:\Program Files\EasyCODE\EasyCODE.exe”



The Command-line-options should be filled out correctly by the Workbench. If not double-check the installation of the “Communication module for EasyCODE”.

2.4 EasyCODE configuration

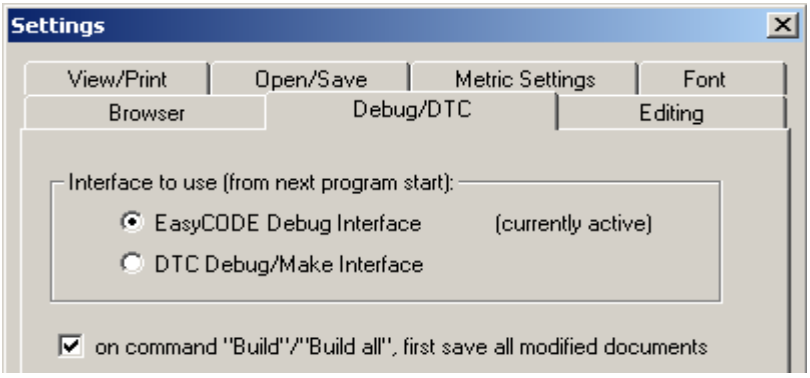
Before you can use EasyCODE to debug your code you will have to add some configuration entries to the EasyCODE configuration.

Start the “EasyCODE Configuration”-tool, and double click on the “CPP” Entry. Now select Import->Import ini file and choose the “Easy-cpp.ini” from your Softune binary folder e.g. “C:\Softune\bin\Easy-cpp.ini”.

After successfully importing the additional configuration options, press the Save-button and exit the EasyCODE configuration.

Next start the EasyCODE editor once to commit the configuration changes. A popup should show up and inform you about the changes you made.

If you want to use EasyCODE to debug your code you will have to select this in the EasyCODE editor configuration. Go to Options->Settings->Debug/DTC and select the “EasyCODE Debug Interface”.



Exit the EasyCODE editor. Now you are ready to start working with, your EasyCODE enabled, Softune Workbench.

3 Using Softune with EasyCODE

There are two ways you can use your EasyCODE software out of Softune.

1. As a simple editor for source files
2. As a debugging interface

3.1 Editor mode

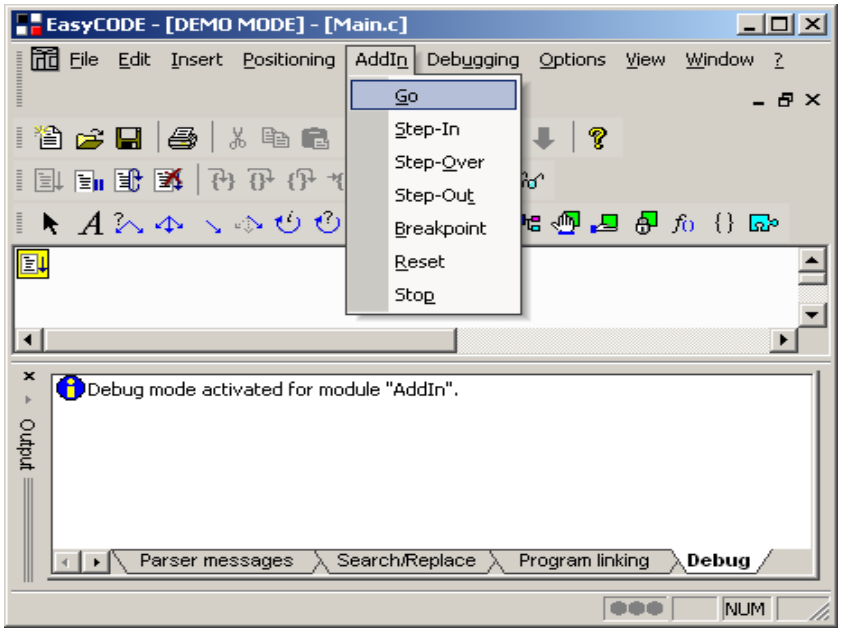
If you followed the application note correctly you have already set up EasyCODE as your default editor. You can now open your existing project or template file in the Softune Workbench. Every time you open or double-click a source file it will open EasyCODE and display the file.

You can edit and save the file in EasyCODE. If you want build your project you have to switch back to Softune.

3.2 Debug mode

Open the Softune Workbench and open your project. Now start the debug mode of the workbench by selecting one of the existing debug-profiles.

Make sure all EasyCODE windows have been closed before to prevent errors. After that start the EasyCODE editor via Project->EasyCODE->Start, in the Softune Workbench.



A new EasyCODE window will open and a message in the output window of the EasyCODE editor should confirm the debug mode.

Now open a source code file in the Softune Workbench, the EasyCODE editor should popup. As you can see a new menu has been created called “Add In”. This menu enables you to navigate the Softune simulator out of the EasyCODE Software.

Be informed that a breakpoint which is set in the Softune Workbench will not be shown in the EasyCODE editor. However the Debugger will stop at it.

For more detailed information about EasyCODE refer to the EasyCODE documentation or the manufacturer’s website.

4 Information on the WWW

Information about Cypress Products can be found on the following Internet pages:

Microcontrollers (8-, 16- and 32bit), Graphics Controllers
Datasheets and Hardware Manuals, Support Tools (Hard- and Software)

http://www.cypress.com/cypress-microcontrollers

Document History

Document Title: AN205254 - F²MC-16LX/FX, EasyCODE Integration into Softune workbench v3

Document Number: 002-05254

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	MKEA	03/23/2007	V1.0 First release (HWe, DGo)
*A	5067205	MKEA	04/06/2016	Converted Spansion Application Note “MCU-AN-300037-E-V11” to Cypress format
*B	5843389	AESATP12	08/03/2017	Updated logo and copyright.

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

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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

© Cypress Semiconductor Corporation, 2007-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.