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.

FM3, FM4 Trace with µVision and ULINK Pro

Target Product: All FM3, FM4 series with trace ports

This application notes describes how setup the trace functionality with KEIL's µVision tool chain using the ULINK Pro JTAG/TRACE adapter.

Contents

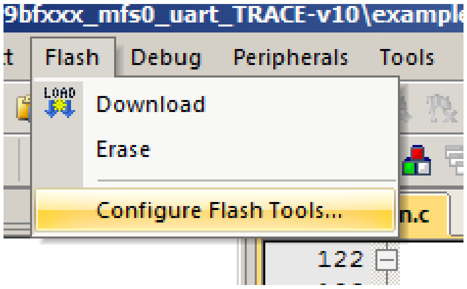
1 Introduction

This application notes describes how setup the trace functionality with KEIL's µVision tool chain using the ULINK Pro JTAG/TRACE adapter.

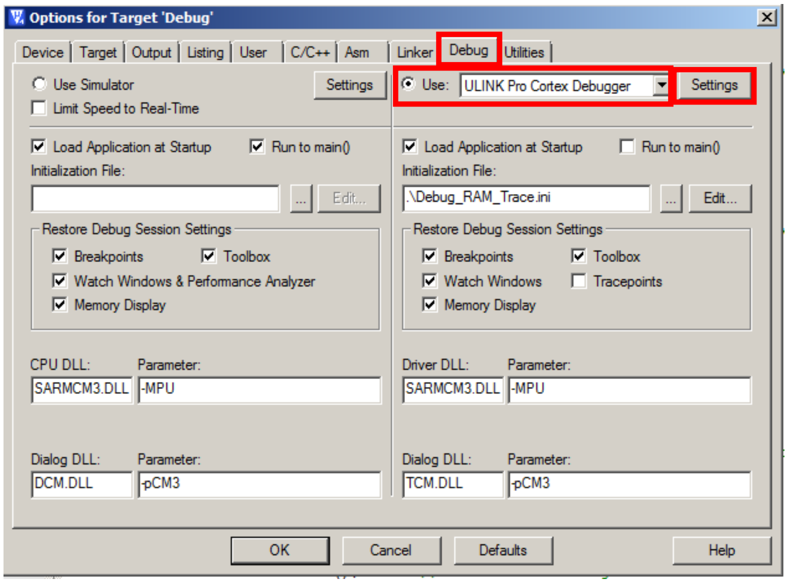
2 µVision Configuration

2.1 Configure Flash Tools…

Follow the menu path **Flash → Configure Flash Tools**… The Options for Target <*Name*> window opens.



In the Options for Target <*Name*> dialog, go to the **Debug** tab. Select the **ULINK Pro Cortex Debugger** option, and then click **Settings**.

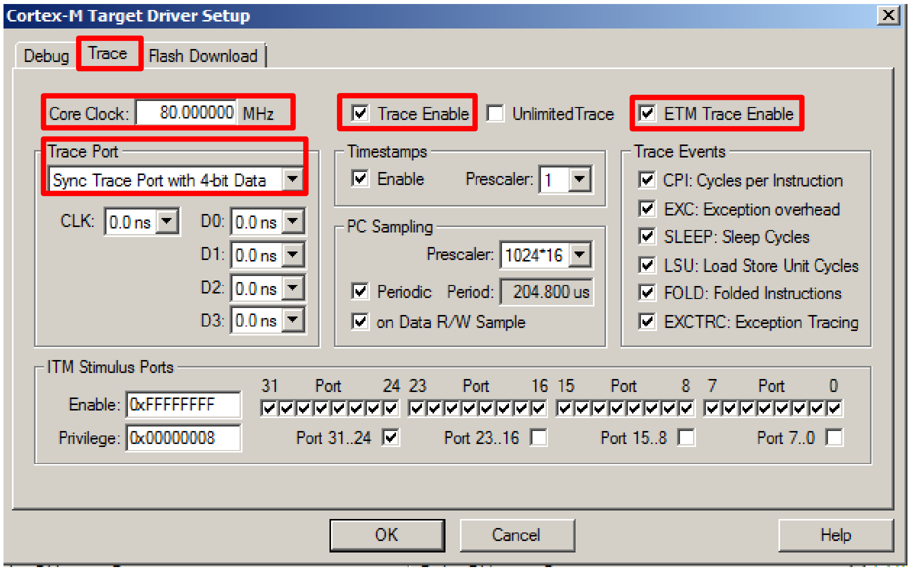


Options for Target 'Debug'

○ Use Simulator
☐ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display
CPU DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: DCM.DLL Parameter: -pCM3

● Use: ULINK Pro Cortex Debugger
☑ Load Application at Startup ☐ Run to main()
Initialization File: .\Debug_RAM_Trace.ini
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows ☐ Tracepoints
☑ Memory Display
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM3

The Cortex-M Target Driver Setup window opens. Go to the **Trace** tab.



Cortex-M Target Driver Setup

Core Clock: 80.000000 MHz
☑ Trace Enable ☐ Unlimited Trace ☑ ETM Trace Enable

Trace Port: Sync Trace Port with 4-bit Data
CLK: 0.0 ns
D0: 0.0 ns
D1: 0.0 ns
D2: 0.0 ns
D3: 0.0 ns

Timestamps: ☑ Enable Prescaler: 1

PC Sampling: Prescaler: 1024*16
☑ Periodic Period: 204.800 us
☑ on Data R/W Sample

Trace Events:
☑ CPI: Cycles per Instruction
☑ EXC: Exception overhead
☑ SLEEP: Sleep Cycles
☑ LSU: Load Store Unit Cycles
☑ FOLD: Folded Instructions
☑ EXCTRC: Exception Tracing

ITM Stimulus Ports
Enable: 0xFFFFFFFF
31 Port 24 23 Port 16 15 Port 8 7 Port 0
☑☑☑☑☑☑☑☑ ☑☑☑☑☑☑☑☑ ☑☑☑☑☑☑☑☑ ☑☑☑☑☑☑☑☑
Privilege: 0x00000008 Port 31..24 ☑ Port 23..16 ☐ Port 15..8 ☐ Port 7..0 ☐

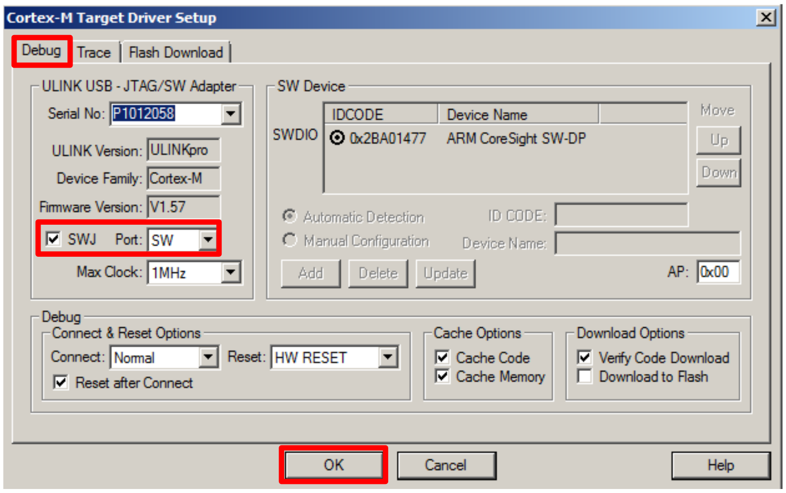
In this configuration window, adjust the following mandatory settings:

- **Core Clock** (for example, FM3: 80 MHz, FM4: 160 MHz)
- **Trace Port**: Sync Trace Port with 4-bit Data
- **Trace Enable**
- **ETM Trace Enable**

The remaining settings depend on your requirements.

2.2 Debugger Configuration

In the Cortex-M Target Driver Setup dialog, go to the **Debug** tab. Select the **SWJ** option, and select **SW** from the **Port** drop-down list to adjust the ULINK Pro to serial wire debug. Confirm the settings and click **OK**.



ULINK USB - JTAG/SW Adapter
Serial No: P1012058
ULINK Version: ULINKpro
Device Family: Cortex-M
Firmware Version: V1.57
☑ SWJ Port: SW
Max Clock: 1MHz

SW Device

	IDCODE	Device Name
SWDIO	● 0x2BA01477	ARM CoreSight SW-DP

● Automatic Detection ID CODE:
○ Manual Configuration Device Name:
AP: 0x00

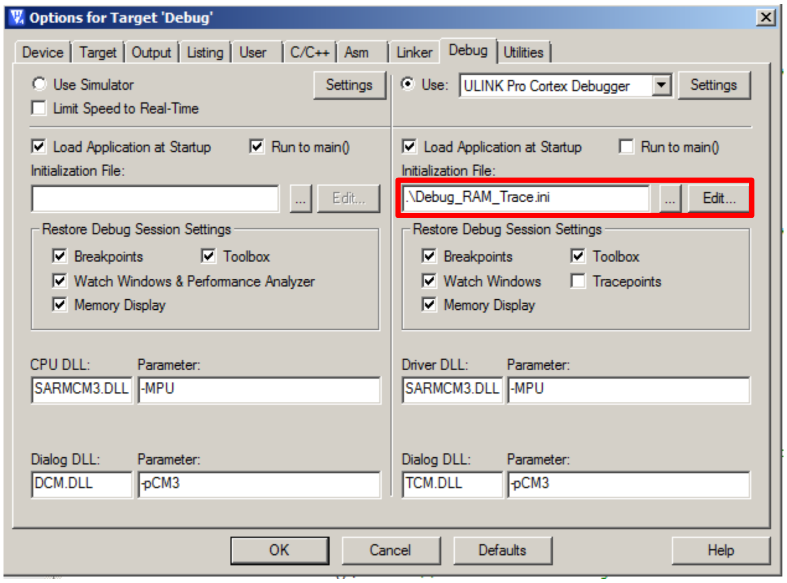
Debug
Connect & Reset Options
Connect: Normal Reset: HW RESET
☑ Reset after Connect

Cache Options
☑ Cache Code
☑ Cache Memory

Download Options
☑ Verify Code Download
□ Download to Flash

2.3 Trace Port Enable

Adjust the debug *ini* script for trace usage.



○ Use Simulator
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:

Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display

CPU DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: DCM.DLL Parameter: -pCM3

● Use: ULINK Pro Cortex Debugger
☑ Load Application at Startup □ Run to main()
Initialization File: .\Debug_RAM_Trace.ini

Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows □ Tracepoints
☑ Memory Display

Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM3

A typical debug initialization script may look like the following code. The necessary trace ports and pins enable is highlighted in dark red bold characters. These lines enable the port pins for trace. Note that at Port 0 also the JTAG pins have to be kept (write `0x000003FF`).

Example for FM4 (MBF568R):

```
/* Initialization Script for Debug RAM */

FUNC void Setup (void) {
  SP = _RDWORD(0x1FFF8000);           // Setup Stack Pointer
  PC = _RDWORD(0x1FFF8004);           // Setup Program Counter
  _WDWORD(0xE000ED08, 0x1FFF8000);    // Setup Vector Table Offset Register

  _WDWORD(0x4006F500, 0xF07FFFFF);    // ADE.AN19...23 = 0
  _WDWORD(0x40063000, 0x000003FF);    // PFRs for Port P00 to P09
  _WBYTE(0x40063603, 0x03);           // TRC0E, TRC1E of EPFR00
}

load %L incremental

Setup();                              // Setup for Running

g, main
```

Example for FM3 (MBF506N):

```
/* Initialization Script for Debug RAM */

FUNC void Setup (void) {
  SP = _RDWORD(0x1FFF8000);           // Setup Stack Pointer
  PC = _RDWORD(0x1FFF8004);           // Setup Program Counter
  _WDWORD(0xE000ED08, 0x1FFF8000);    // Setup Vector Table Offset Register

  _WDWORD(0x40033000, 0x000003FF);    // PFRs for Port P00 to P09
  _WBYTE(0x40033603, 0x03);           // TRC0E, TRC1E of EPFR00
}

load %L incremental

Setup();                              // Setup for Running

g, main
```

Note: FM3 and FM4 have different GPIO base addresses. Table 1 lists the differences.

Table 1. Differences between GPIO Base Addresses

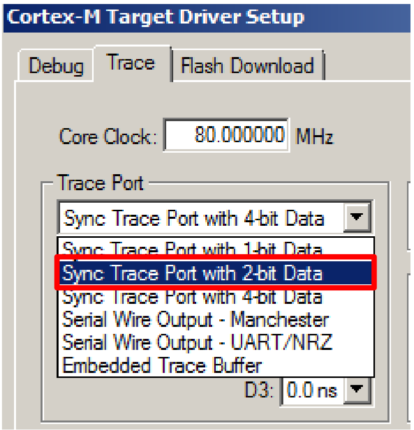
Register	FM3 address	FM4 address
EPFR0	0x40033600	0x40063600
PFR0	0x40033000	0x40063000
ADE	0x4003F000	0x4006F000

Note, that the JTAG port pins must be preserved, so that the access to EPFR00 should be done for the upper byte where the trace enable bits are located. Thus, the offset for the EPFR00 base address is 3.

Note: If the trace pins share analog inputs, these analog inputs must be switched OFF using the GPIO's ADE register. Check the pin layout to identify the analog inputs that your device share with these trace pins.

2.3.1 Trace Port with 2-Bit Data

If only TRACECLK, TRACED0, TRACED1 should be used by writing `0x01` to `0x40033603`, **Sync Trace Port with 2-bit Data** must be selected in the Cortex-M Target Driver Setup dialog.

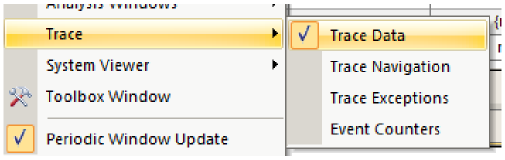

Cortex-M Target Driver Setup
Debug
Trace
Flash Download
Core Clock: 80.000000 MHz
Trace Port
Sync Trace Port with 4-bit Data
Sync Trace Port with 1-bit Data
Sync Trace Port with 2-bit Data
Sync Trace Port with 4-bit Data
Serial Wire Output - Manchester
Serial Wire Output - UART/NRZ
Embedded Trace Buffer
D3: 0.0 ns

3 Debug Session with µVision

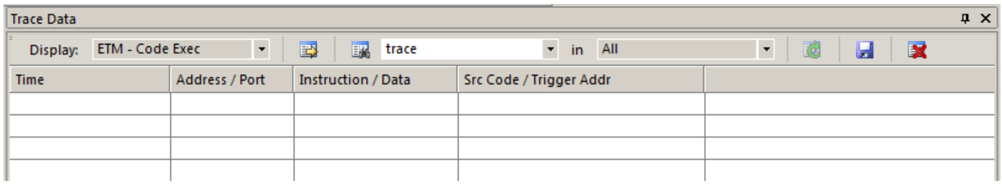
3.1 Trace Window

Now the IDE and the MCU are prepared for trace usage.

Start the debug session with the µVision IDE. Follow the menu path **View → Trace → Trace Data**.

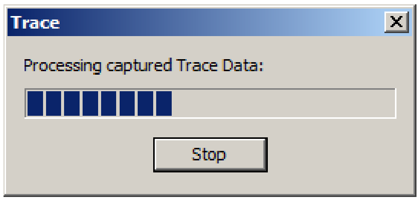


The Trace Data window opens.



Select **ETM – Code Exec HLL** from the **Display** drop-down list. The ITM items use a reduced trace functionality via the JTAG port.

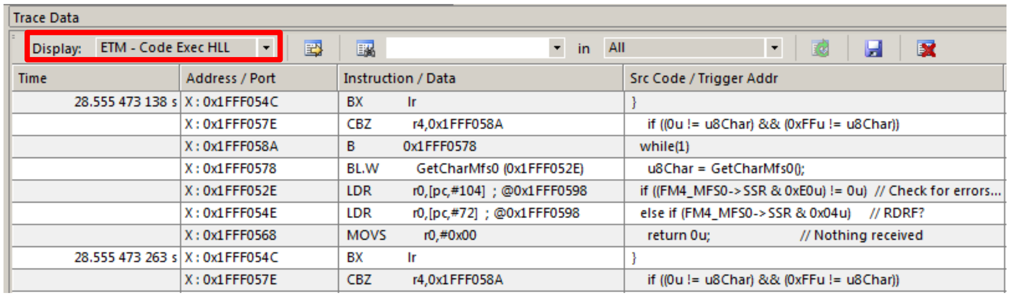
A pop up window will occur after stopping the application.



Trace

Processing captured Trace Data:

After code execution and the application is stopped, the trace window will look like the following example.



Trace Data

Display: ETM - Code Exec HLL

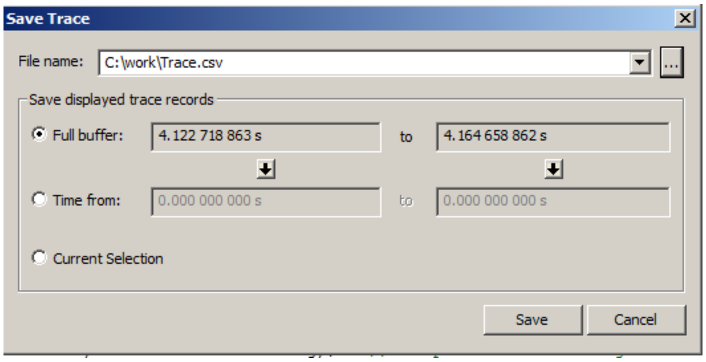
in All

Time	Address / Port	Instruction / Data	Src Code / Trigger Addr
28.555 473 138 s	X : 0x1FFF054C	BX lr	}
	X : 0x1FFF057E	CBZ r4,0x1FFF058A	if ((0u != u8Char) && (0xFFu != u8Char))
	X : 0x1FFF058A	B 0x1FFF0578	while(1)
	X : 0x1FFF0578	BL.W GetCharMfs0 (0x1FFF052E)	u8Char = GetCharMfs0();
	X : 0x1FFF052E	LDR r0,[pc,#104] ; @0x1FFF0598	if ((FM4_MFS0->SSR & 0xE0u) != 0u) // Check for errors...
	X : 0x1FFF054E	LDR r0,[pc,#72] ; @0x1FFF0598	else if (FM4_MFS0->SSR & 0x04u) // RDRF?
	X : 0x1FFF0568	MOVS r0,#0x00	return 0u; // Nothing received
28.555 473 263 s	X : 0x1FFF054C	BX lr	}
	X : 0x1FFF057E	CBZ r4,0x1FFF058A	if ((0u != u8Char) && (0xFFu != u8Char))

3.2 Saving Trace Data

You can explore the trace data to a CSV file using the save icon.

You can either save the full buffer for a time interval or a selection of the trace window. Note that the save process may take longer, depending of the size of the trace data.



Save Trace

File name: C:\work\Trace.csv

Save displayed trace records

● Full buffer: 4.122 718 863 s to 4.164 658 862 s

○ Time from: 0.000 000 000 s to 0.000 000 000 s

○ Current Selection

The CSV file looks like the following excerpt.

```
# C:/work/Trace.csv : uVision Trace Data Export - CSV
#
#
Index,Time (in s),Address / Port,Instruction / Data,Src Code / Trigger Addr,Function
"0","0.736783500",X : 0x1FFF058A," B            0x1FFF0578","   while(1)","main"
"1","0.736783500",X : 0x1FFF0578," BL.W         GetCharMfs0 (0x1FFF052E)","     u8Char = GetCharMfs0();","main"
"2","0.736783500",X :  0x1FFF052E," LDR          r0,[pc,#104]  ; @0x1FFF0598","   if ((FM4_MFS0->SSR & 0xE0u) != 0u)  // Check for
errors PE, ORE, FRE","GetCharMfs0"
"3","0.736783500",X : 0x1FFF054E," LDR          r0,[pc,#72]  ; @0x1FFF0598","   else if (FM4_MFS0->SSR & 0x04u)     //
RDRF?","GetCharMfs0"
"4","0.736783500",X : 0x1FFF0568," MOVS         r0,#0x00","     return 0u;                        // Nothing received","GetCharMfs0"
"5","0.736783500",X : 0x1FFF054C," BX           lr"," }","GetCharMfs0"
"6","0.736783625",X : 0x1FFF057E," CBZ          r4,0x1FFF058A","     if ((0u != u8Char) && (0xFFu != u8Char))","main"
```

Document History

Document Title: AN204432 - FM3, FM4 Trace with µVision and ULINK Pro

Document Number: 002-04432

Revision	ECN	Submission Date	Description of Change
**	–	03/06/2014	Initial Release
*A	5034126	12/02/2015	Converted Spansion Application Note "FM3_FM4_AN706-00082-1v0-E" to Cypress format
*B	5874553	09/06/2017	Updated logo and copyright.
*C	6469227	01/31/2019	Sunset Review Updated Template Minor Text Edits
*D	6835006	03/20/2020	Adjusted the width of the tables containing examples FM4 (MBF568R) and FM3 (MBF506N).

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 MCU

Cypress Developer Community

Community | Code Examples | 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

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. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. 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. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, 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 any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.