



# PRECISION32<sup>™</sup> IDE AND APPBUILDER DETAILED TUTORIAL AND WALKTHROUGH

## 1. Introduction

This document provides a step-by-step tutorial walkthrough for the Precision32 Development Tools (IDE and AppBuilder) using the **SiM3U1xx-B-DK** hardware and **Silicon Labs SDK version 1.0.1**. The first series in the walkthrough imports a pre-existing example into the IDE and discusses the IDE debugging basics. The second series guides the creation of a TIMER0 Blinky example using AppBuilder and adds some additional code editing and debugging tips for the IDE.





## 2. Relevant Documentation

Precision32 Application Notes are listed on the following website: www.silabs.com/32bit-appnotes.

- AN667: Getting Started with the Silicon Labs Precision32<sup>™</sup> IDE—provides a description of the IDE features and environment.
- AN670: Getting Started with the Silicon Labs Precision32<sup>TM</sup> AppBuilder—provides a description of the AppBuilder features.

### 3. Hardware Setup

To set up the hardware for this walkthrough:



Figure 2. Hardware Setup Diagram

- 1. Connect the USB Debug Adapter to J31 and connect the Debug Adapter to the PC with a USB cable.
- 2. Move the USER CONTROL MCU card switch (SW4) to ON.
- 3. Move the SYSTEM Power Select switch (SW5) to the USB position.
- 4. Connect a mini USB cable to the Device USB connector (J13). Connect the cable to the PC to power the board.





Figure 3. Shorting Block Settings

- 5. Verify the shorting blocks are installed as shown in Figure 3:
  - a. J18 top two pins
  - b. J14
  - c. J15
  - d. J17
  - e. J16



### 4. Lab Software Setup

- 1. Install the Precision32 package from www.silabs.com/32bit-software.
- 2. Open the Precision32 IDE and activate it, if it's not activated already. Instructions on how to do this are available on the Welcome page in the IDE.

## 5. Walkthrough Series #1: Running Blinky in the Precision32 IDE

#### 5.1. Objectives

The objectives of series are:

- Learn how to import an SDK example into the Precision32 IDE.
- Learn how to download a program to a device.
- Explore some of the debugging features of the IDE.

#### 5.2. Instructions

- 1. Launch the **Precision32 IDE 1.0.1** (default path: C:\SiLabs\Precision32\_v1.0.1\IDE) and select a workspace (C:\SiLabs\workspace).
- 2. Activate the IDE using the instructions on the landing page, if the IDE isn't already activated.
- 3. Set the Silicon Labs SDK path to C:\SiLabs\32bit\si32-1.0.1.





- 4. Click the Import SI32 SDK example(s) link in the Quickstart view.
- 5. Scroll down and check just the **sim3u1xx\_Blinky** example. The default path for importing projects is C:\SiLabs\32bit\si32-x.y.z\Examples, where x.y.z is the version of the SDK selected in the previous step, but the **Browse...** button can be used to navigate to a specific example directory. The **Deselect All** button is also useful for deselecting a long set of examples.
- 6. Ensure the **Copy projects into workspace** checkbox is enabled.
- 7. Click Finish.





- 8. Click the sim3u1xx\_Blinky project name.
- 9. Click the Build 'sim3u1xx\_Blinky' [Debug] link.





10. Click the Debug 'sim3u1xx\_Blinky' [Debug] link. The Debug view will open with trace information.





11. Start the program. PB2.10 and PB2.11 will blink after a small delay. The printf() output from the example will appear in the **Console** view.

	11	
Develop - sim3u1xx_Blinky/src/main.c - Precision32		x
<u>File Edit Source Refactor Navigate Search P</u>	roject <u>R</u> un Silicon Labs <u>W</u> indow <u>H</u> elp	
Project 🛛 👬 Core Re 🚼 Periphe 🛄 🗆	🍄 Debug 🛛 🛛 👘 🖉 🖉 👘 💷 📕 🖉 🕞 🗁 😥 👘 👔 🖓 🖉 🎽	
E Sim3uLxc_Blinky Sinaries Debug Composed by Second Se		
j∫n gCpu.h		
.c gModes.c	Welcome C main.c X	
a griddes.n	65 // Save the current msTicks_10 value as last seen 66 meTicks 10 last = meTicks 10:	^
ing gPB.h	67 }// if msTicks 10 changed	
ic gRtc0.c	68	
j₀ gRtc0.h	69 // If msTicks has changed	
Car sim3u1xx	7/0 1f (msTicks != msTicks_last)	
The main c	72 // Update every 1 second	
🙂 Q 🐹 🖉 V 🧠 Br 🔚 O 💖 Ex 🦳 🗆	73 if(!(msTicks % 1000))	
Start here		
	76 // invert the state of the LED griver (P2.11) 76 SI32 PBSTD & Loggle ping(SI32 PBSTD 2 0x800).	
ion New Precisionsz project	77 }	
Import SI32 SDK example(s)	78	-
🗟 Build all projects [Debug]	79 // Save current msTicks value as last seen	
K Build 'sim3u1xx_Blinky' [Debug]	80 mslicks_last = mslicks;	-
Clean 'sim3u1xx Blinky' [Debug]		
莎 Debug 'sim3u1xx_Blinky' [Debug]	📮 Console 🛿 🕂 🛣 Problems 🚺 Memory 📳 Red Trace Preview 🔗 Search 🛛 🔳 💥 🐐 🔛 💭 🦃 🛃 📮 🖉 🔤 🖛 📋 🖛 🗍 🖕 👘 🖉 🖉 🖓 🖛 Search 🖉 Search 🖉 Search Sea	
🖄 Quick Settings 🔻	hello world	^
Project and File wizards 🛛 😵	1, 1 1, 1	
Import and Export ¥	1, 1	
Build and Settings 🛛 🕹		-
Debug and Run 🛛 😵 👻	۲. (۱۰) (۱۰) (۱۰) (۱۰) (۱۰) (۱۰) (۱۰) (۱۰)	•
□ <sup>◆</sup> SI32 SDK: C:\SiLabs\32bit\si32-1.0.1	<b>▼</b>	
sim3u1xx Blinky SiLabs SiM3U167		



- 12. Pause the program.
- 13. Double-click the blue area next to a line to add a breakpoint.
- 14. Start the program again.





- 15. Highlight the msTicks\_10 variable in the main.c file, right click, and select Add Watch Expression....
- 16. Press OK in the Add Watch Expression dialog.
- 17. Select the Expression view.





18. Click the **Resume** button a few times. The program will stop each time at the breakpoint and update the value of **msTicks\_10**.





- 19. Select the Peripheral view.
- 20. Select the Memory view.
- 21. Enable the RTC-0 checkbox.
- 22. Disable the RTC by selecting **DISABLED** for the **RTCEN** bit.
- 23. Double-click on the breakpoint to remove it.
- 24. Start the program again. The PB2.10 LED will stop blinking since the RTC is no longer running. PB2.11 still blinks because it's running from the SysTick timer.





25. Click the Terminate button to end the debugging session.

	25	
Develop - sim3u1xx_Blinky/src/main.c - Precision32		×
<u>File Edit Source Refactor Navigate Search P</u>	roject <u>R</u> un Silicon Labs <u>W</u> indow <u>H</u> elp	
11 - 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	- 🖹 🖹 🚵 🖆 🛄 💭 💭 🚵 🍙 💁 🛸 - 🛞 🧐 🌞 💊 - 🕲 - 🖻 🗶 Develop	
Project 👬 Core Re 🛃 Periphe 🖄 🖓 🗖	(参 Debug 23) 後 # 25   10 変 マ	
	C sim3u1xx_Blinky Debug [C/C++ MCU Application]      Sim3u1xx_Blinky Debugger (6/4/12 2:46 PM) (Suspended)	
Address	Thread [1] (Suspended: Breakpoint hit.)	
□	arm-none-eabi-gdb (6/4/12 2:46 PM)	
R PMU-0 0x40048000	C:\SiLabs\workspace\sim3u1xx_Blinky\Debug\sim3u1xx_Blinky.axf (6/4/12 2:46 PM)	
□ ╦ RSTSRC-0 0x4002d060		
RetentionRam4 0x20000000		
🗐 🛃 SARADC-0 0x4001a000	🚱 Welcome 🛛 🔝	
□ 🔤 🔤 SARADC-1 0x4001b000	41	
□	42 // Enter the default operating mode for this application (gModes.c)	
□	43 gModes_enter_my_default_mode(); 44	
SSG-0 0x4001e000	45 // Print a starting message	
🗐 📴 StandardRam28 0x20001000 🚽	<pre>46 printf("hello world\n");</pre>	
۲. III ۲. ۲. ۲. ۲. ۲. ۲. ۲. ۲. ۲. ۲. ۲. ۲. ۲.	48 // Loop forever	
🙂 Q 🚳 = V 🤏 Br 🔡 O 🚳 Ex 🕸 📃 🗖	49 while (1)	_
	50 {	
Name Value	<pre>\$1 // II mSIICKS_10 has changed \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>	
*+y "msTicks 10" 0x5d7	53 {	
Add new expression	54 // Update every 500 ms	
	55 11 (!(ms11cx3_10 % 50)) 56 {	
		F F
	L	
	sim3u1xy, Blinky Debug IC/C++ MCU Application] C-\Sil abs\workspace\sim3u1xy, Blinky/Debug\sim3u1xy, Blinky/Debug\sim3u1xy, Blinky/Debug	
	1, 1	*
	1, 1	
	1, 1	
	1, 1	
	4	
SIB2 SDK: C:\SiLabs\32bit\si32-1.0.1	<b>▼</b>	
sim3u1xx Blinky SiLabs SiM3U167		



## 6. Walkthrough Series #2: Creating a Timer Blinky

#### 6.1. Objectives

The objectives of this series are:

- Use AppBuilder to configure the TIMER0 high 16-bit timer to generate a 12 Hz waveform on PB2.10 (red LED).
- Use the information learned in Lab 1 to download and debug the generated AppBuilder project.

#### 6.2. Lab Preparation

The default SiM3U1xx oscillator is the Low Power Oscillator at 20 MHz, but this doesn't create a 12 Hz waveform cleanly:

$$\frac{20000000}{12} = 1666666.67$$

Instead, we can use the 48 MHz USB0 oscillator. The actual overflow rate desired for the timer is 12 Hz multiplied by two, since the timer must overflow for each rising and falling edge of the waveform on the pin. This means our TIMER0 effective overflow rate is:

$$\frac{48000000}{12 \times 2} = 2000000$$

The TIMER0 module has two ways to meet this overflow rate. The module can clock from a divided version of APB. This prescaler (CLKDIV and CLKDIVRL) is an integer value and can be arbitrarily chosen between 0 and 255 (effectively creating APB divided by 256 up to APB divided by 1). In addition, the Timer count interval is 16 bits in split mode, so it can be anywhere between 0 and 65535 (setting the timer count to 0xFFFF causes the timer to overflow in one clock). We can choose these values in any way such that they equal the required count of 2000000.

If we choose to set the TIMER0 APB prescaler to 40 (for a TIMER0 clock of 1.2 MHz):

$$\frac{2000000}{40} = 50000$$

Prescaler (CLKDIVRL) value from the reference manual:

$$\mathsf{F}_{\mathsf{prescale}} = \frac{\mathsf{F}_{\mathsf{APB}}}{256 - \mathsf{CLKDIVRL}}$$

$$256-CLKDIVRL\ =\ 256-40\ =\ 216$$

16-bit reload value (HCCR) from the reference manual:

Overflow Rate = 
$$\frac{F_{TIMER}}{65536 - HCCR}$$

 $65536-HCCR\ =\ 65536-50000\ =\ 15536$ 



#### 6.3. Instructions

- 1. Launch **AppBuilder 1.0.1** (default path: C:\SiLabs\Precision32\_v1.0.1\AppBuilder).
- 2. Select New Project....
- 3. Select the project settings. SiM3U1xx→SiM3U167-B-GQ→B→1.0.1. Update the Name of the project and Location to the workspace location. Press OK.

😴 Silicon Labs AppBuilder			- 3 - X -	1		
Ele Edit View Project Help						
5 日 日 2 日 1 日 1 日 1 日 1 日 1 日 1 日 1 日 1 日						
Stores + 3 × Start Page ×		Toolbox	~ 0 ×			
2 Cyck Start Gude Silicon Labs News: New Project Silicon Labs Inhunded Tschnology. Helps Re Open Project Start Labs Company Andrea Start Start Labs Company Andrea Start	Concepts 'Light up the	Possibi * Progran				
Enor Uni  Show page on Startup  Enor Uni  Name Location Description	Family SIM3C1XX SIM3U1XX	2-Bit Proj	Device SIM3U157-B-GQ SIM3U157-B-GQ SIM3U164-B-GM SIM3U166-B-GQ SIM3U167-B-GQ SIM3U167-B-GQ	Rev SDK	SILICON	LA B S
3	Name: Location: Toolset: Compiler:	TIMER0_Blinky C:\SiLabs\works Precision32 GCC	pace	· · · · · · · · · · · · · · · · · · ·	USB, 256K Flash, 32K RAM, EM	IF, 80-pin TQFP.
	SDK Path:	C:\SiLabs\32bit\	si32-1.0.1		ОК	Cancel



- 4. Double-click on Clock Control.
- 5. Select the USB0 Oscillator.
- 6. Enable the USB0 Oscillator in the Properties window.
- 7. Click on the MUX radial button to select the USB0 oscillator as the AHB clock source. The diagram updates to show the current AHB and APB clock frequencies. After selecting the USB0 clock as the AHB source, an error appears indicating that the USB0 APB clock must be enabled.









 Enable the APB clocks to the Ports (PBCFG0, PBSTD0, PBSTD1, PBSTD2, PBSTD3, PBHD4), TIMER0, USB0, and FLASHCTRL0. The error and red highlight around USB0 disappear after enabling the USB0 APB clock.





- 9. We can remove some errors and disable SWV since we're not using it for this application. Double-click on the crossbar 0 error in the **Error List** window.
- 10. Select **Disabled** in the **Enable JTAG/SWV** drop-down menu.





- 11. Double-click on crossbar1.
- 12. Enable crossbar1.





- 13. Click on the PB2.10 column.
- 14. Set PB2.10 to Push-Pull in the Properties window.





- 15. Let's fix the FLASHCTRL0 error in the Error List. Double-click on Peripherals.
- 16. Drag and drop FLASHCTRL0 to the Peripherals Canvas.





- 17. Double-click on the error. The program will automatically jump to the **FLASHCTRL0 Properties** window.
- 18. Set the Flash Speed Mode field to 2. The error will disappear after hitting Enter.





19. Let's fix the last error in the Error List for the Watchdog Timer. Drag and drop **WDTIMER0** to the **Peripherals** Canvas.





- 20. Double-click on the error. The program will automatically jump to the WDTIMER0 Properties window.
- 21. Disable the Watchdog Timer. The **Error List** should now be empty.





22. Drag and drop TIMER0 to the Peripherals Canvas.





- 23. Click on TIMER0.
- 24. Set the **Desired Prescaler** value to **1.2 MHz.**
- 25. Select **Split Timers** for the **Timer Mode**.





- 26. Select the **Prescaler** as the TIMER0 high clock source.
- 27. Start the high timer.
- 28. Set the **Desired Overflow Frequency** to **24 Hz** (12 Hz x 2 for toggling the LED on and off). TIMER0 high will be in auto-reload mode by default.

TIMER0_Blinky - Silicon Labs	AppBuilder					x
ile <u>E</u> dit <u>V</u> iew <u>P</u> roject <u>H</u>	elp					
🗊 🛃 💕 🐰 🖬 🛍 🔊						-
oject TIMER0_Blinky Mode Transitions Clock Control Clock Control SPOrt I/O Crossbar0 Crossbar1 FPB4	V V Vefault Mode (	rossbar1 Start Page	Default Mode Peripherals 🗙	<ul> <li>▼ Toolbox</li> <li>€ LD00</li> <li>€ PMU0</li> <li>€ VMON0</li> <li>€ VREG0</li> <li>€ VREG0</li> <li>4 Timers</li> <li>€ EPCA0</li> <li>€ LPTIMER0</li> <li>€ PCA1</li> <li>€ TIMER1</li> </ul>		* # X
<ul> <li>Project Files (Read-Only)</li> <li>main.c</li> <li>myCLKCTRL.h</li> <li>myCPU.h</li> <li>myCPU.c</li> <li>myCPU.c</li> <li>myCPB.c</li> <li>myVB8.h</li> <li>myVS80.h</li> </ul>	E			Cther     AcSo     CRC0     CRC0     CCC0     CCC0     RSTSRC0     Properties     TIMER0 Periph	ral Properties	₽ ₽ ₽ ₽
C myUSB0.c myFLASHCTRL0.c myTIMER0.c myTIMER0.c myTIMER0.c myWDTIMER0.c				27 Timer Debug M High Timer Clock Source Wood Start Timer Enable Marter	ode Prescaler Auto Reload	
ror List				↓ µ × Desired Overflo	w Erequency 24 Hz	
Name	Location	Description		Actual Overflow	Frequency 24 Hz	=
				Low Timer	1.22.27 2.2.22	
				Clock Source	APB	
				Mode	Auto Reload	
				Start Timer		



 We can view the generated code by clicking on gTIMER0.c in the source files list. AppBuilder used the same values we calculated at the beginning of the lab for the prescaler (216) and reload values (15536).





- 30. Click the TIMER0 Interrupts tab.
- 31. Enable High Timer Overflow interrupts.





- 32. Click the **Export** button.
- 33. Verify Open After Export is automatically selected.
- 34. Press **OK**. The IDE will automatically open and have the AppBuilder project added to the workspace.









35. In the IDE, double-click on myTIMER0.c. The TIMER0H high overflow interrupt handler will be empty

Develop - TIMER0_Blinky/src/myTIMER0.c - Precisio	n32	• X
<u>File Edit Source Refactor Navigate Search F</u>	2roject <u>R</u> un Silicon Labs <u>W</u> indow <u>H</u> elp	
C¹ ▾ "; © à à în ă � ✔ 券 è   �   � . ≠ . ≠ . / Ø . ↓ ↓ ↓	- C 🖻 🚵 🖆 📇 📮 , 🖗 🖓 🤷 🧄 🤷 - 🥸 💖 🕸 🔦 - 🗞 - 🖽 🔀 🔀 Develop	]
Project 🛛 🚻 Core Re 🛃 Periphe 🗖 🗖	Welcome 🕞 myTIMER0.c 🕴	
	1// Copyright (c) 2012	
Src     generated     myinkerset     mycLKCTRL.c     mycLKCTRL.c     mycPU.c     myFLASHCTRL0.c     myFLASHCTRL0.h     myFLASHCTRL0.h     myFLASHCTRL0.h     myTIMER0.h     myTIMER0.h     myTIMER0.h     myTIMER0.h     myTIMER0.h     myUSB0.c     mytUSB0.c     myt	<pre>2 2 3#include "myTIMERO.h" 4 5// 6// 2nd Level Interrupt Handlers (Called from generated code) 7// 8 void TIMEROH_high_overflow_handler(void) 9 { 10 11 } 12 </pre>	
🗟 Build all projects [Debug]	4	•
🔏 Build 'TIMER0_Blinky' [Debug]	😑 Console 🖄 🖳 Problems 👖 Memory 📲 Red Trace Preview 🔗 Search 🛛 🚽 🖻	
🧹 Clean 'TIMER0_Blinky' [Debug] 😑	No consoles to display at this time.	
TIMER0 Blinky' [Debug]		
Vuick Settings 🔻		
Project and File wizards 🛛 😵		
🐼 Import and Export 🛛 😵		
Build and Settings		
Debug and Pup		
SI32 SDK: C:\SiLabs\32bit\si32-1.0.1	<ul> <li>Writable</li> <li>Smart Insert</li> <li>1:1</li> </ul>	
TIMER0 Blinky Silabs SiM3U167		

The TIMER0 high overflow handler needs to perform two functions: 1) clear the TIMER0 high overflow interrupt flag in the TIMER0 module, and 2) toggle the PB2.10 pin.



36. Add the includes to the top of the myTIMER0.c file:

```
#include <SI32_PBSTD_A_Type.h>
#include <SI32_TIMER_A_Type.h>
#include <si32_device.h>
```

37. Because we already added the header files, we can use the auto-complete feature of the IDE to help us find the correct functions. Move the cursor inside the interrupt handler, type in "SI32\_TIMER\_A\_c", and press CTRL + SPACEBAR. This will open a window that shows all the available defines and functions that start with SI32\_TIMER\_A\_c. The "clear\_high\_overflow\_interrupt" function will be the second choice. Once the correct function is selected, fill in the correct base pointer for TIMER0 (SI32\_TIMER\_0). The line of code should look like:

```
SI32_TIMER_A_clear_high_overflow_interrupt(SI32_TIMER_0);
```





38. We can use the same process to enter in the second function, but we'll search for it in the full list this time. Move to a new line, type in "SI32\_PBSTD\_A", and press CTRL + SPACEBAR. The list will show all functions and defines for the PBSTD A module. We know we want a function that will toggle the pin, so look for a function of the appropriate name (toggle\_pins). Once the correct function is selected, fill in the correct base pointer for PB2 (SI32\_PBSTD\_2) and the correct mask for PB2.10 (0x0400). The line of code should look like:

SI32\_PBSTD\_A\_toggle\_pins(SI32\_PBSTD\_2, 0x0400);





## AN719

- 39. Build the code and start a debug session. Then start the program. The LED will blink at a 12 Hz rate.
- 40. Add a breakpoint in the TIMER0 handler in myTIMER0.c.
- 41. Select the Memory view.
- 42. Select the Peripheral view.
- 43. Open the PB2 register window by selecting PBSTD-2 in the Peripheral view.
- 44. Pressing the start program button repeatedly will update the PB2 pin value in the register window.





#### 6.4. Additional Notes: Migrating a Project

The AppBuilder migrate feature allows a project switch to a different version of the SDK, a different device family, or device within the same family. To migrate a project:

- 1. Open the project in AppBuilder 1.0.1.
- 2. Go to the **Project**→**Migrate Project...** menu selection.
- 3. Check the **Change family, device, or rev** checkbox to change the device family, device within the selected family, or the revision of the device. This will prompt a message window that requests a new export location.
- 4. Changes to the SDK version do not require a new export location.
- 5. Make any other desired modifications to the Migrate Project dialog and press OK.





# AN719

If the report option was not disabled, AppBuilder will generate a report of the migration operation. This report will output a summary of the migration operation (from $\rightarrow$ to) and any potential Errors, Warnings, or Messages.

🗖 TIMERO_Blinky - Silicon Labs AppBuilder						
Eile Edit View Project Help						
🛅 🔒 💕 👗 🖻 🛍 🤊 🖲	- (° - 🕞					
Project	▼ A × Migration	Report ×	Ŧ	Toolbox	<b>~</b> ↓ ×	
Mode Transitions	Migrati	on Report				
Peripherals	SIM3016	-B-GQ SDK 1.0.1 SIM30164-B-GM SDK 1.0.1				
▲ S Port I/O	Errors	(0)				
Crossbar1 Ø PB4	▶ Warn	ings (0)				
	► Mess	ages (0)				
h gCLKCTRLh c gCLKCTRLc h gCPU.h c gCPU.c d gFLASHCTRL0.c	*			Properties	• 4 ×	
gFLASHCTRL0.h	_			Default Mode	1ode Properties 🔹	
gModes.c				Properties		
gPB.c				Common		
E g IIMERO.6 g TIMERO.h C gUSB0.c h gUSB0.h g WDTIMERO.c h gWDTIMERO.h Error List	*		<b>*</b> ∓ ×	Name	Default Mode	
Name	Location	Description				



## NOTES:



## **CONTACT INFORMATION**

Silicon Laboratories Inc.

400 West Cesar Chavez Austin, TX 78701 Tel: 1+(512) 416-8500 Fax: 1+(512) 416-9669 Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page: https://www.silabs.com/support/pages/contacttechnicalsupport.aspx and register to submit a technical support request.

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories and Silicon Labs are trademarks of Silicon Laboratories Inc. Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.

