PRECISION32™ SOFTWARE DEVELOPMENT KIT CODE EXAMPLES OVERVIEW

1. Introduction

The Precision32™ code examples are part of the Software Development Kit (SDK) installed with the Precision32 software package available at www.silabs.com/32bit-software. The code examples are simple and complete examples that illustrate and highlight the peripheral modes and features. They are also modular so code can be copied into an application base, making peripheral-specific code development easy.

Figure 1 shows the Precision32 firmware layer block diagram.

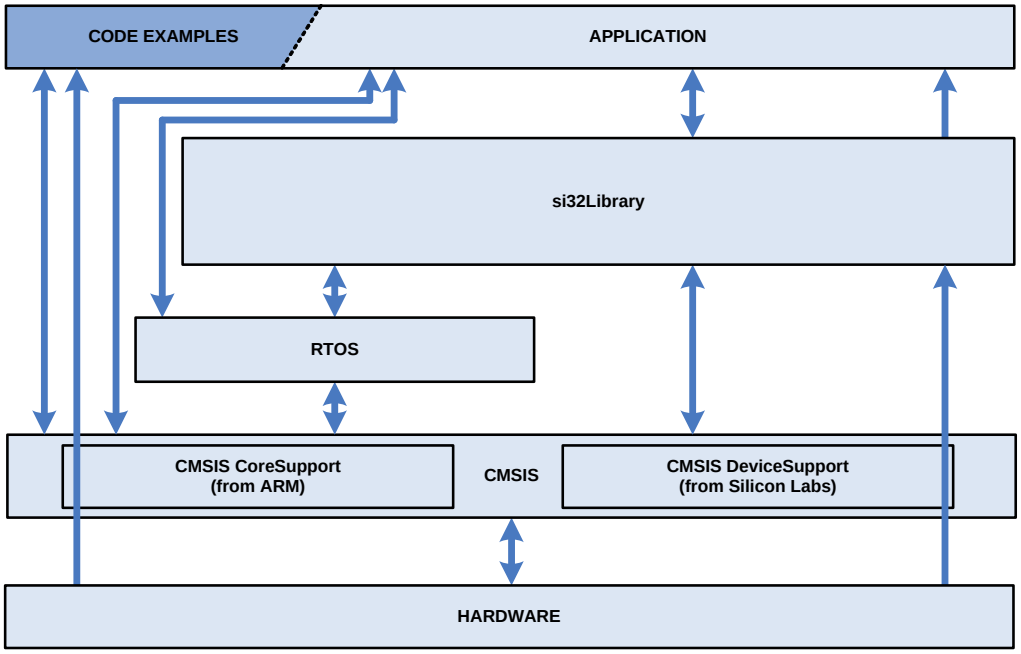


Figure 1. Firmware Layer Block Diagram

2. Relevant Documentation

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

- AN664: Precision32™ CMSIS and HAL User's Guide
- AN667: Getting Started with the Silicon Labs Precision32™ IDE
- AN670: Getting Started with the Silicon Labs Precision32™ AppBuilder
- AN673: Precision32™ Software Development Kit (SDK) Overview

Copyright © 2012 by Silicon Laboratories

3. Code Example Organization

The code examples are separated by device and are located in **si32-x.y\Examples\device**, where **x** is the major SDK version, and **y** is the minor SDK version. Each code example has its own folder named with the peripheral or example name.

The IDE can import these code examples using the instructions in the application note, “AN667: Getting Started with the Silicon Labs Precision32 IDE”. Sub-directories under the example folder include example projects for uVision and IAR.

Figure 2 displays the code example organization on disk.

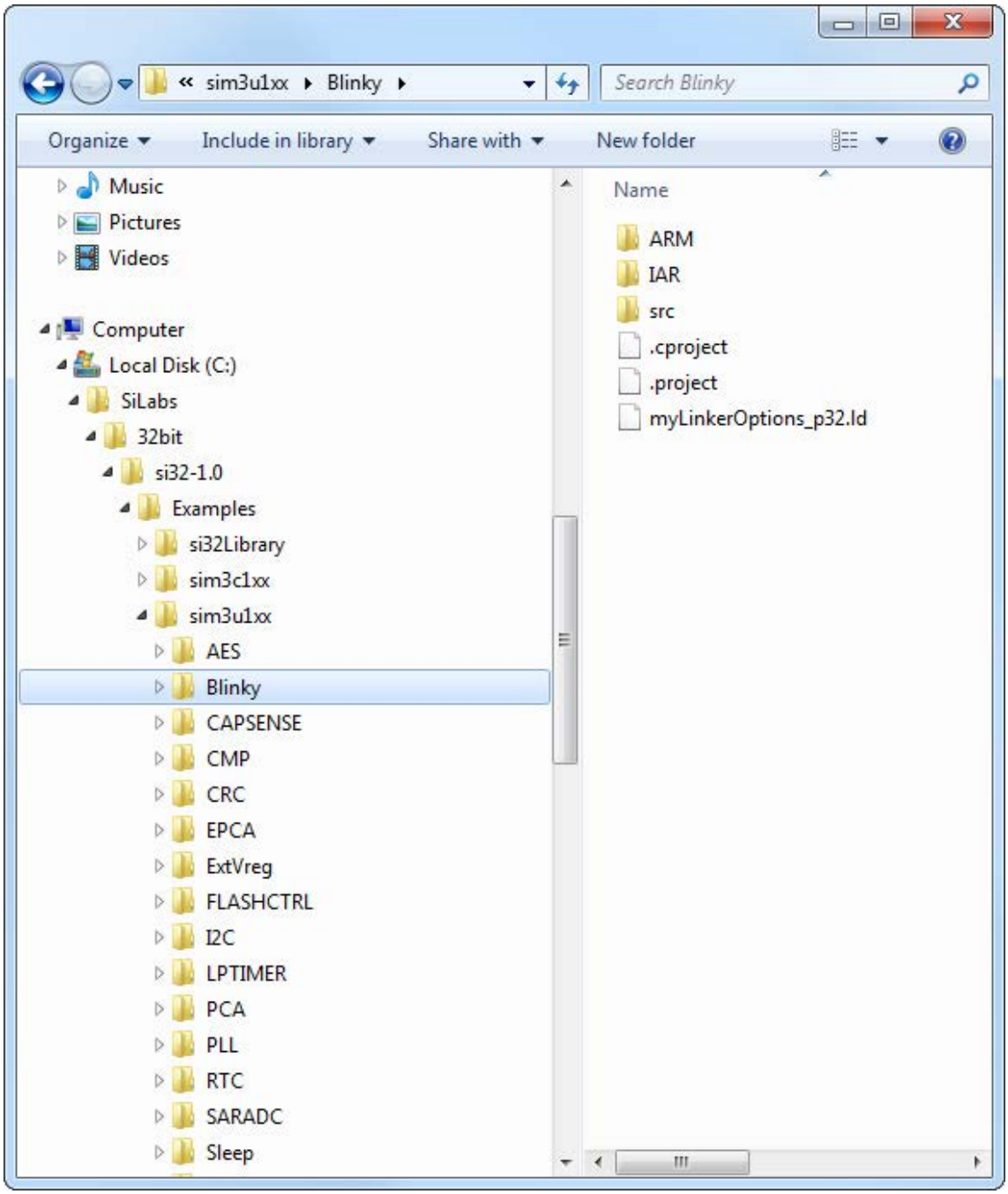


Figure 2. Code Example Organization

4. Detailed Documentation

The detailed documentation for each code example is in the Readme file in the example folders.This file includes a basic description of the example, the resources and clock speeds used, notes on the example and supported modes, and detailed how-to-use steps.

The **Blinky_Readme.txt** file shown in Figure 3 is installed in **si32-x.y\Example\sim3u1xx\Blinky** for SiM3U1xx devices after installing the Precision32 software package.

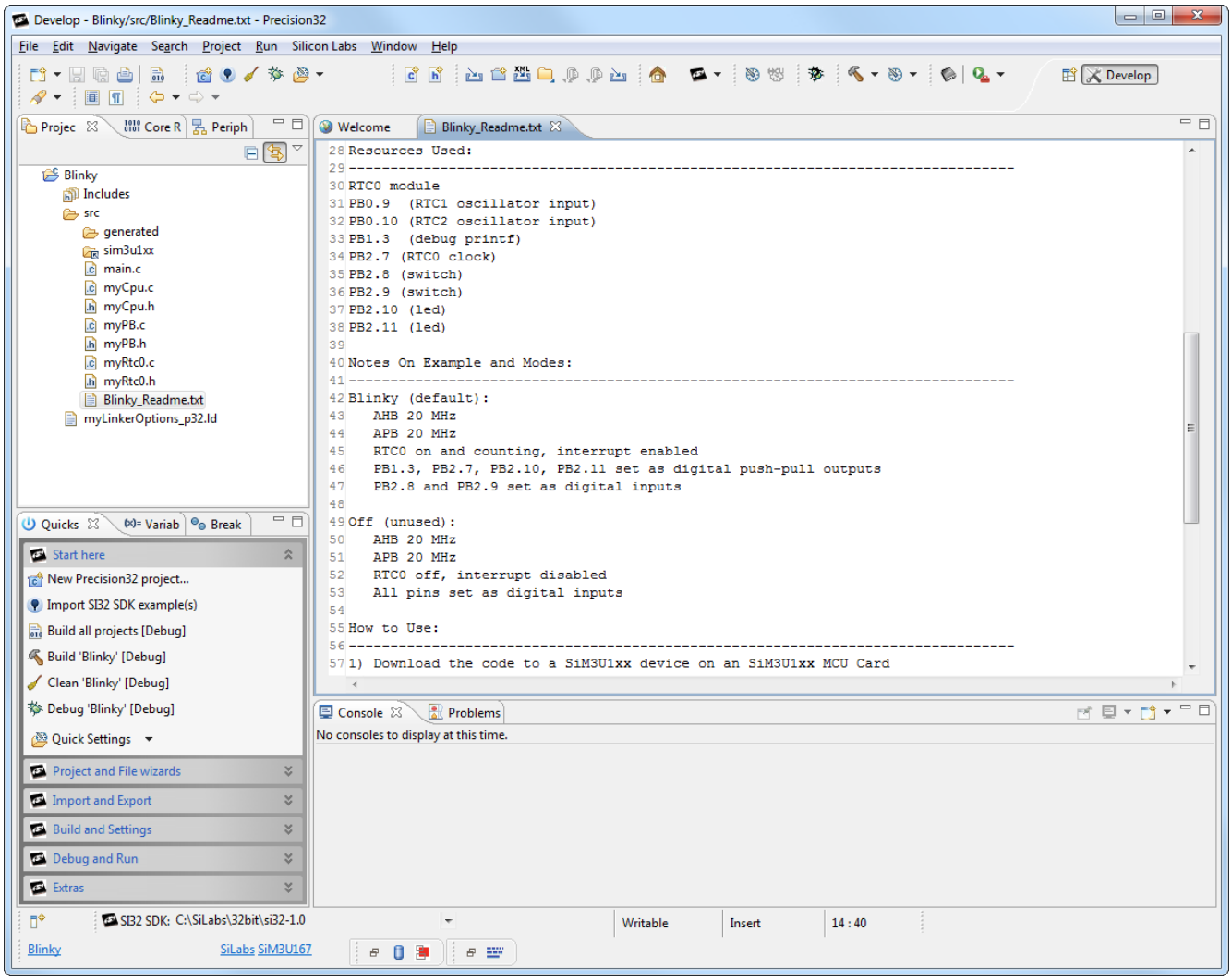


Figure 3. Code Example Documentation

5. SiM3U1xx Blinky Example

Blinky uses the SiM3U1xx RTC0 module and SysTick to toggle two LEDs. PB2.10 (LED on the SiM3U1xx MCU card) toggles every 500 ms using the RTC counting 10 ms intervals. PB2.11 (also an LED) toggles every second using SysTick. The example also reads the switches on the MCU card (PB2.8 and PB2.9) every 500 ms and prints their status to the **Console** view (semi-hosting) by default.

The code examples follow the Silicon Labs AppBuilder file generation structure: generated files are included in the **src\generated** folder with the **g-** prefix, and application files are in the **src** folder with **my-** prefix. All of the modes implemented in the example are called from **main**. In the Blinky example, **main** calls **gModes_enter_my_default_mode()** from **gModes.c**, which then calls **gCpu_enter_default_config()** in **gCpu.c**, and so forth.

These code examples use the Silicon Labs Hardware Access Layer (HAL) macros by default.

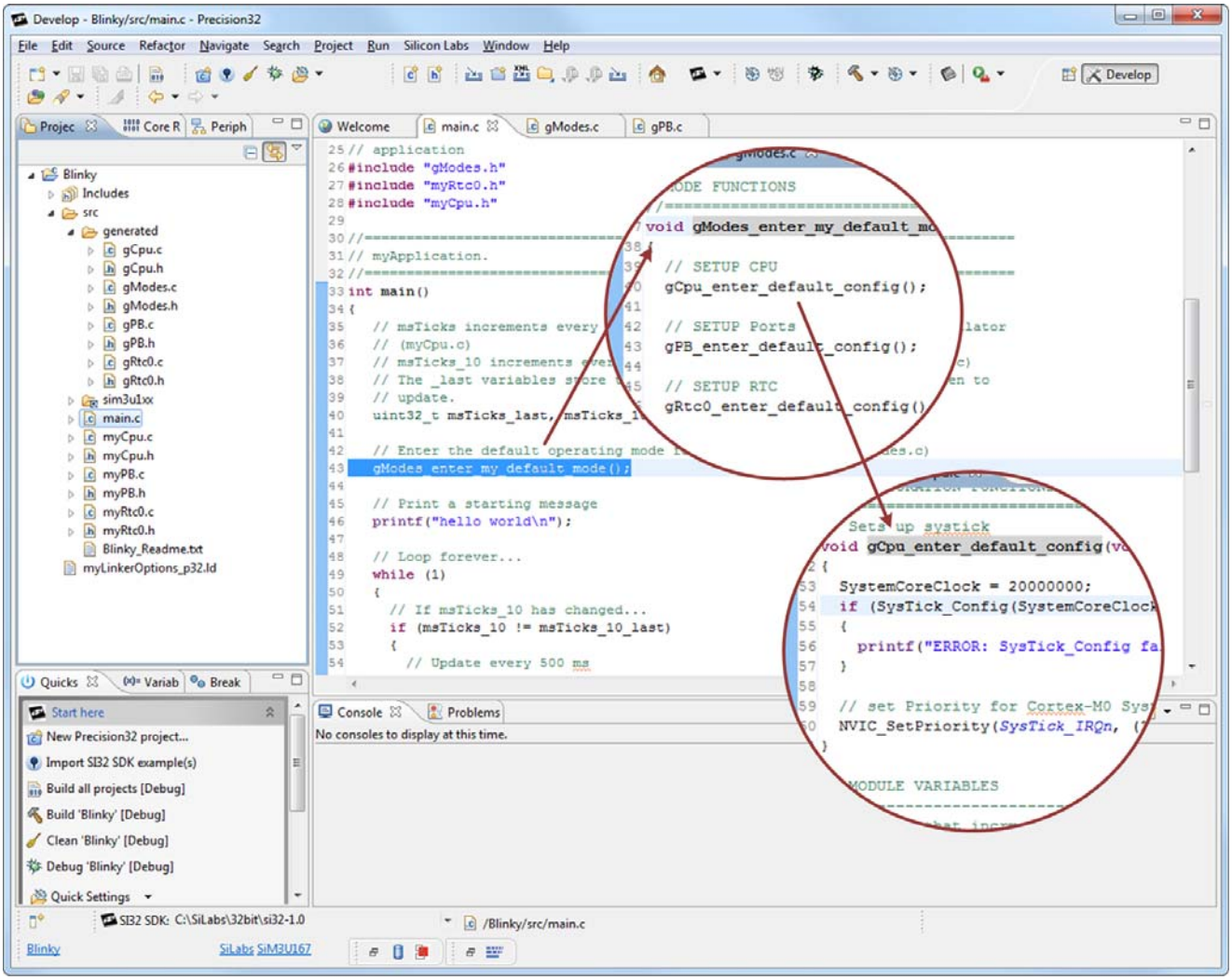


Figure 4. Blinky Example

5.1. Blinky Application-Specific Files

5.1.1. main.c

The **main.c** file calls functions from the **gModes.c** and the port HAL as shown in Figure 5. This file also includes all the main application code that sits in a while(1) loop, toggling the LED pins or reading the switches and printing their status to the **Console** view.

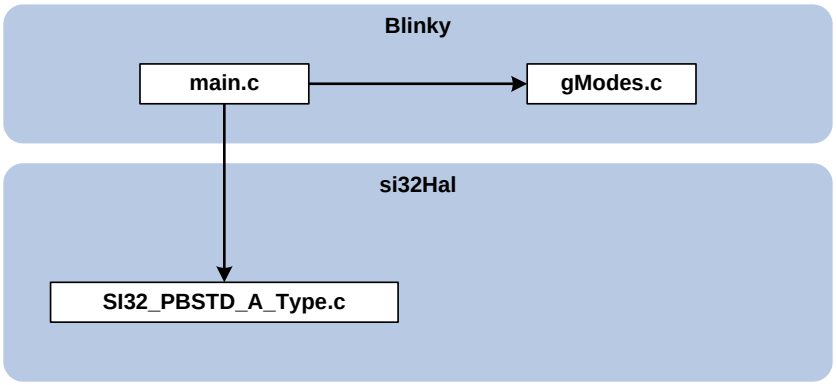


Figure 5. main.c Dependencies

5.1.2. myCpu.c

This file has the application-specific implementation of **mySystemInit()** called by **system_sim3u1xx.c** in the HAL. For Blinky, this function:

- Disables the Watchdog Timer.
- Enables APB to the Port Bank modules.
- Sets the Serial Wire Viewer pin (PB1.3) to push-pull.

Figure 6 shows the dependencies for **myCpu.c**.

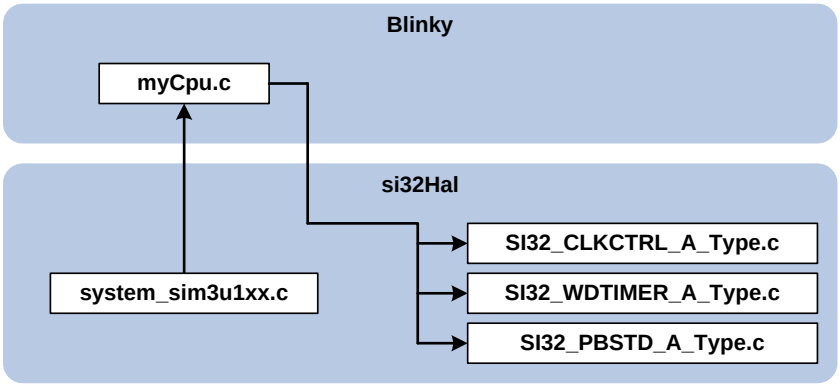


Figure 6. myCpu.c Dependencies

5.1.3. myPB.c

This file doesn't include any application-specific code for the Blinky example.

5.1.4. myRtc0.c

The **myRtc0.c** file includes the second-level handlers for the RTC0 Alarm 0 and RTC0 oscillator fail interrupts. These second-level handlers are called from the first-level handlers in gRtc0.c.

The second-level Alarm 0 handler:

- Reads the RTC0 counter value.
- Sets the RTC0 Alarm 0 value with a new value equal to the current counter value + 10 ms.
- Increments **msTicks_10**, which keeps track of the 10 ms intervals.
- Clears the Alarm 0 interrupt in the RTC0 module.

The second-level oscillator fail handler sits in a while(1) loop to indicate an unrecoverable error condition.

Figure 6 shows the dependencies for **myRtc0.c**.

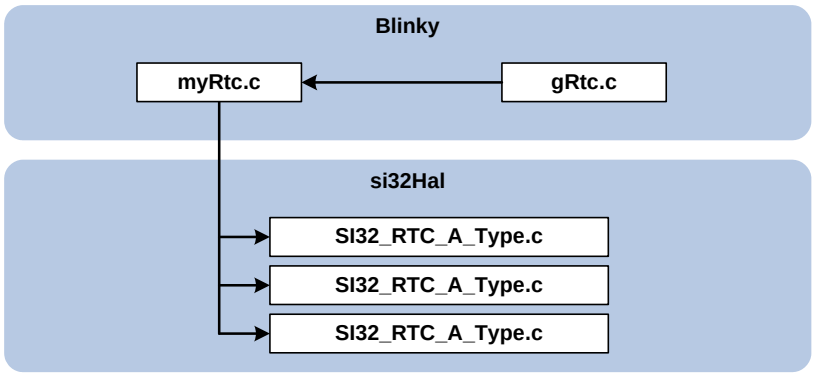


Figure 7. myRtc0.c Dependencies

5.2. Blinky AppBuilder-Generated Files

5.2.1. gCpu.c

This file contains the SysTick handler, which increments the **msTicks** variable, and the **gCpu_enter_default_config()** function, which is called from **gModes_enter_my_default_mode()** in **gModes.c**. This function sets the **SystemCoreClock** variable to 20 MHz, since Blinky uses the Low Power Oscillator, and sets the SysTick timer to trigger every millisecond.

Figure 8 shows the dependencies for **gCpu.c**.

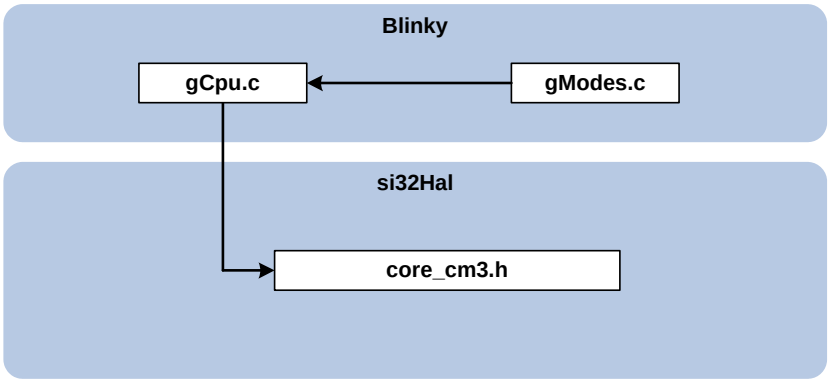


Figure 8. gCpu.c Dependencies

5.2.2. gModes.c

The **gModes.c** file has two functions: **gModes_enter_my_default_mode()** and **gModes_enter_my_off_mode()**.

The default mode function places the device in the default mode after a reset and calls **gCpu_enter_default_config()** from **gCpu.c**, **gPB_enter_default_config()** from **gPB.c**, and **gRtc0_enter_default_config()** from **gRtc0.c**.

The off mode function is not currently called from anywhere in the project and includes calls to **gRtc0_enter_off_config()** from **gRtc0.c** and **gPB_enter_off_config()** from **gPB.c**.

Figure 9 illustrates the dependencies diagram for **gModes.c**.

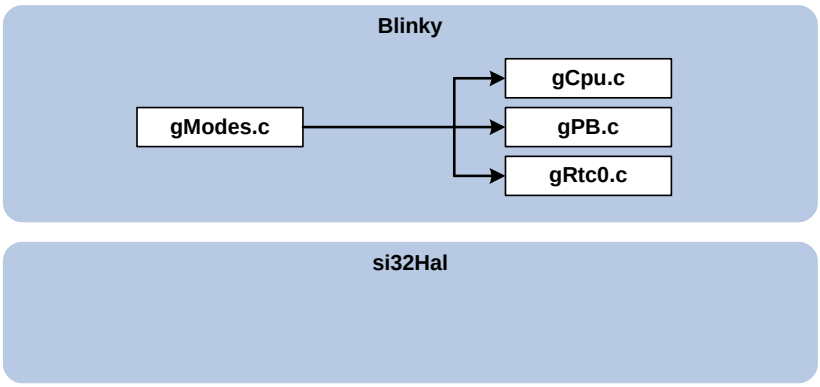


Figure 9. gModes.c Dependencies

5.2.3. gPB.c

This file includes two functions: **gPB_enter_off_config()** and **gPB_enter_default_config()**.

The **gPB_enter_off_config()** function is called by **gModes_enter_my_off_mode()** in **gModes.c** and sets all the pins to digital input mode, disables crossbar 1, and disables the APB clock to the port registers.

The **gPB_enter_default_config()** function:

- Enables the APB clock to the port registers.
- Sets the SWV pin to push-pull.
- Enables crossbar 1.
- Sets the PB2.10 and PB2.11 pins to push-pull.
- Sets the PB2.8 and PB2.9 pins to digital input mode.
- Configures the PB0.9 and PB0.10 RTC0 oscillator input pins as analog inputs.
- Configures PB2.7 to output the RTC0 clock by setting PB2.7 to push-pull, skipping the PB2.0-PB2.6 pins on crossbar 1, and enabling the oscillator output on crossbar 1.

Figure 10 shows the dependencies for **gPB.c**.

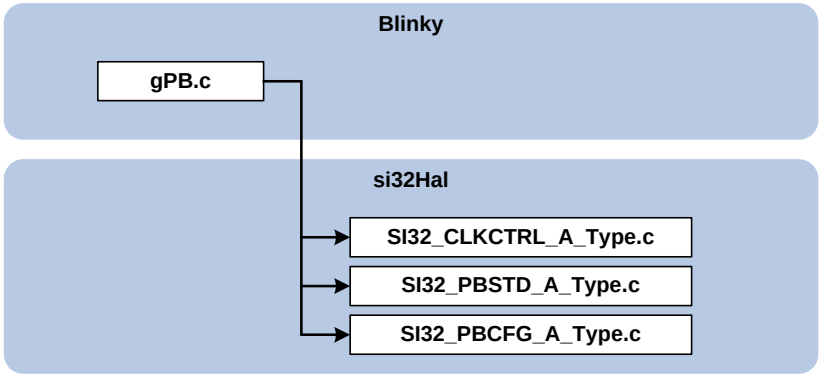


Figure 10. gPB.c Dependencies

5.2.4. gRtc0.c

The **gRtc0.c** file includes the first-level interrupt handlers for the RTC0 oscillator fail (**RTC0FAIL_IRQHandler()**) and Alarm 0 (**RTC0ALRM_IRQHandler()**). These first-level interrupt handlers just call the second-level handlers and must use the handler names defined in **sim3u1xx.h**, the device-specific header file.

In addition to the first-level interrupt handlers, **gRtc0.c** has two functions: **gRtc_enter_off_config()** and **gRtc0_enter_default_config()**.

The **gRtc_enter_off_config()** function disables the Alarm 0 interrupt in the RTC0 module, clears any pending oscillator fail or Alarm 0 interrupts in the NVIC, and disables these two interrupts in the NVIC. In addition, the function stops the RTC0 timer, disables the RTC0 module, and disables the APB clock to RTC0.

The **gRtc0_enter_default_config()** function:

- Enables the APB clock to the RTC0 module.
- Enables the RTC module and configures it for crystal oscillator mode.
- Sets the initial Alarm 0 value for 10 ms.
- Clears any pending interrupts and enables the interrupts in the NVIC.
- Enables the Alarm 0 interrupt in the RTC0 module.
- Enables the RTC0 output.

Figure 11 shows the dependencies for **gRtc0.c**.

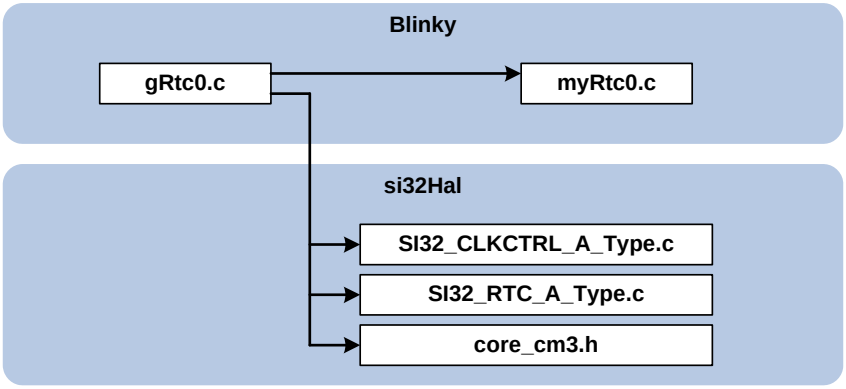
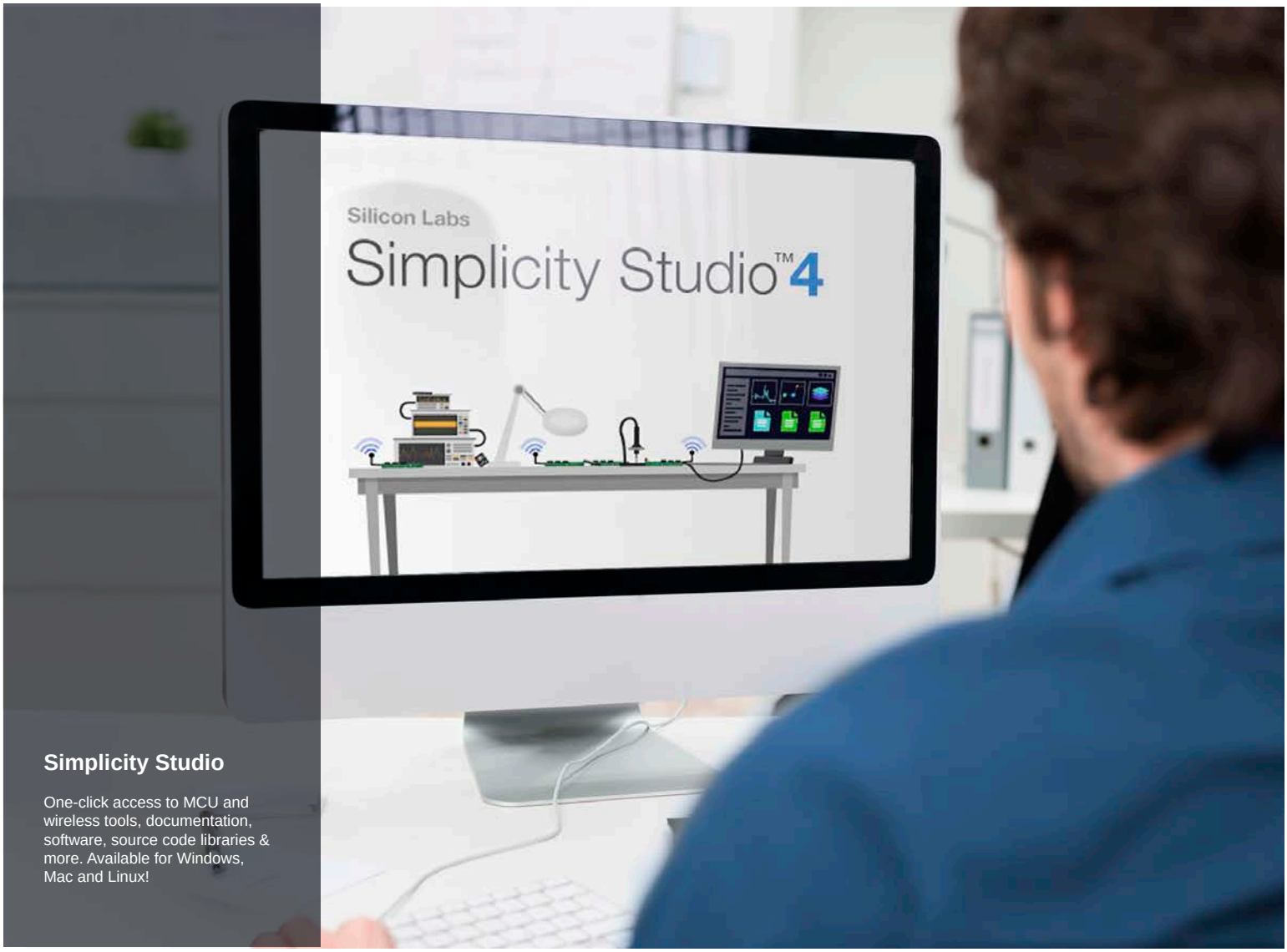


Figure 11. gRtc0.c Dependencies



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Disclaimer
Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information
Silicon Laboratories Inc.® , Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOmodem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com