USER'S GUIDE

Apollo EVK

Ultra-Low Power Apollo SoC Family

A-SOCAP1-UGGA01EN v1.3

Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO 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. "TYPICAL" PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING "TYPICALS" MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER'S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Revision	Date	Description
1.0	June 2015	Initial release
1.1	April 2016	More information added about functionality for each board. Board errata added. List of software examples for each board added.
1.2	May 2016	Additional errata item added to address extra power consumption created by R65 and the EVK sensor board. Minor corrections to Bluetooth Low Energy board errata text.
1.3	January 6, 2023	Updated document template

Reference Documents

Document ID	Description

Table of Contents

List of Figures

SECTION

1

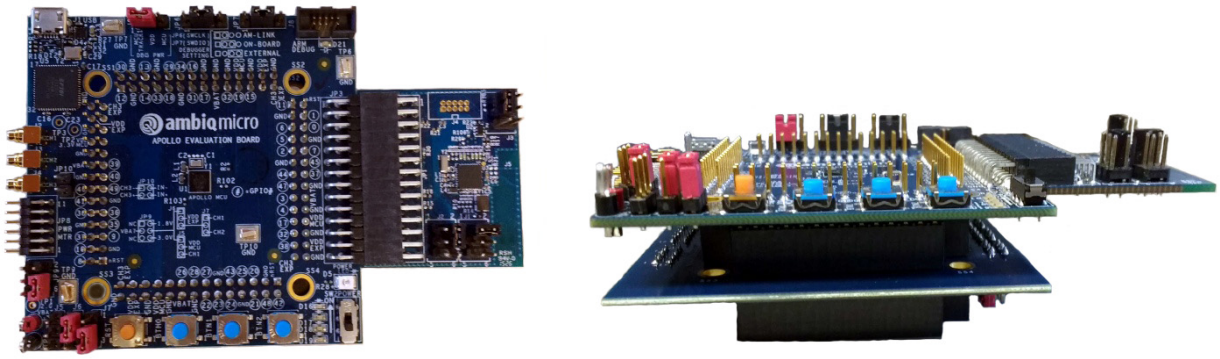
Introduction

This document guides users in setting up their Apollo evaluation system to get started with executing code examples, measuring power consumption in various configurations, and begin software development.

An Apollo EVK evaluation system consists of 3 boards: the Apollo EVK base board, Apollo EVK sensor board, and Apollo EVK Bluetooth Low Energy board. The EVK system facilitates development of sensor and Bluetooth applications utilizing the revolutionary low power consumption of the Apollo SoC.

Figure 1-1: Apollo EVK

Figure 1-2: Apollo EVK, with all boards assembled together for evaluation



(a) Top view of assembled EVK (b) Side view of assembled EVK

An Apollo EVB evaluation system consists of just the Apollo EVK base board. The Apollo EVB allows for rapid prototyping of embedded applications that take advantage of the Apollo SoCs low energy consumption, rich set of peripherals, and high performance Arm® Cortex®-M4F features.

Figure 1-3: Apollo EVB

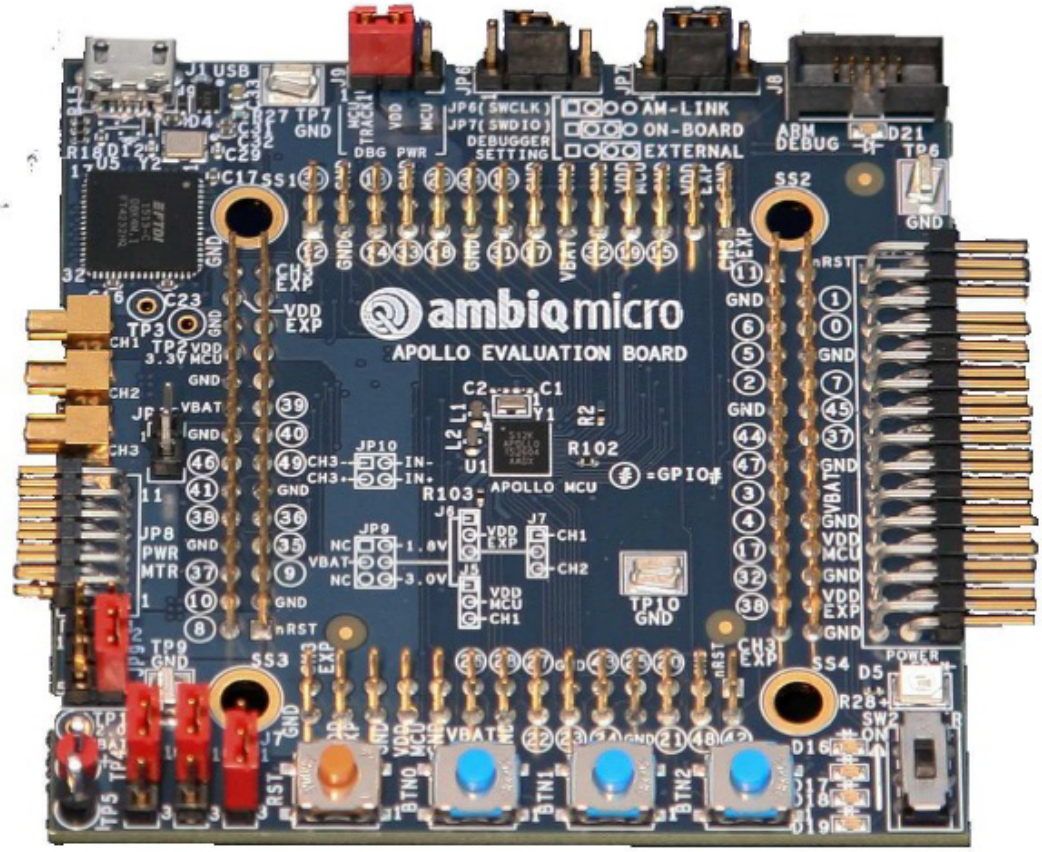
SECTION

2

Apollo Base Board

The Apollo base board is the primary evaluation platform for the Apollo SoC. The board features the APOLLO-512-KBR part in the center, and various header pins that function to break out all key pins as well as select power supplies, and debugger options via jumpers.

Figure 2-1: Apollo Base Board

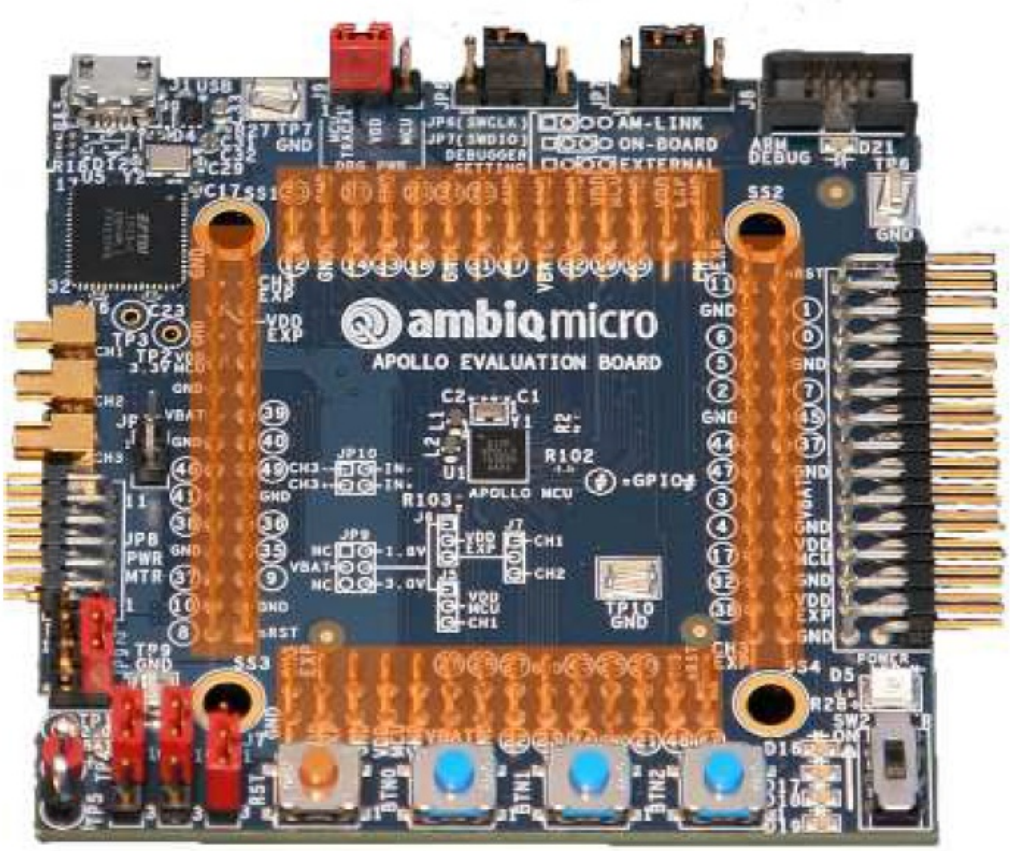


2.1 Expansion Headers

On the base board are four, 2x14 pin headers (JP1, JP2, JP4, and JP5), one on each of the four sides of the SoC that expose all pins for easy access. JP3 is a right-angle 2x14 header that replicates the connections of JP2 for side-loaded connections. Silkscreen on the base board indicates the functionality of each pin on these header. Numbers shown in a circle on the silkscreen indicate a GPIO (e.g., 6 in a circle indicates a pin is connected to GPIO6).

On an Apollo EVK, the sensor board connects to JP1, JP2, JP4, and JP5. The Bluetooth Low Energy radio board connects to JP3.

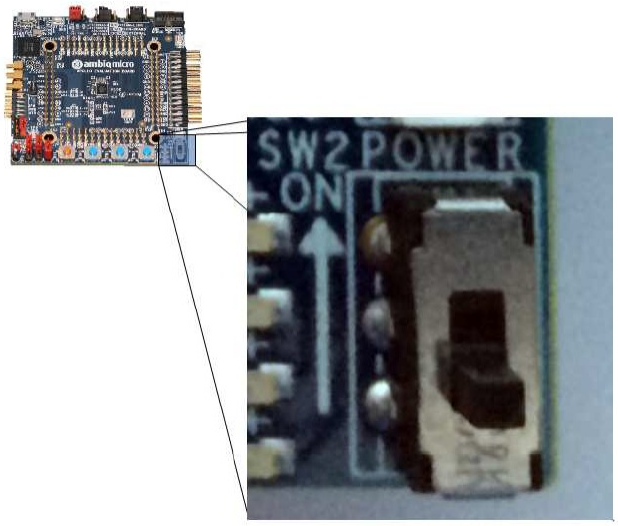
Figure 2-2: Apollo Base Board



2.2 Supplies and Power Measurements

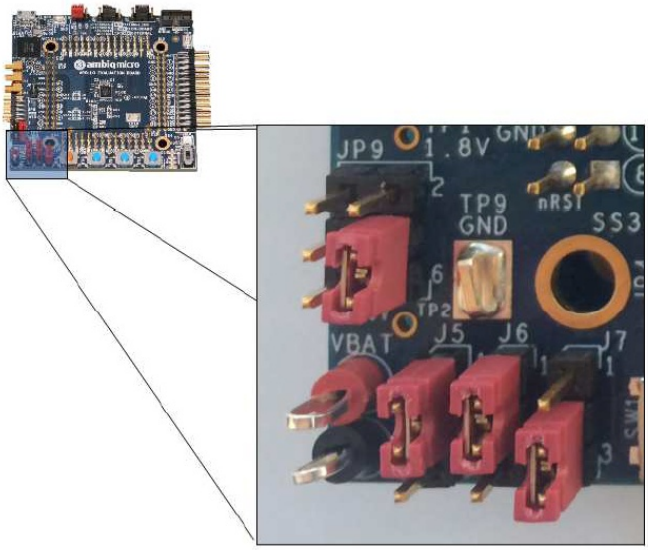
The default power supply input for the EVK base board is the USB connection to a PC (J1). The power switch (SW2) in the bottom right-hand corner of the board must be set to the on position for USB power to be applied to the board.

Figure 2-3: Apollo Base Board



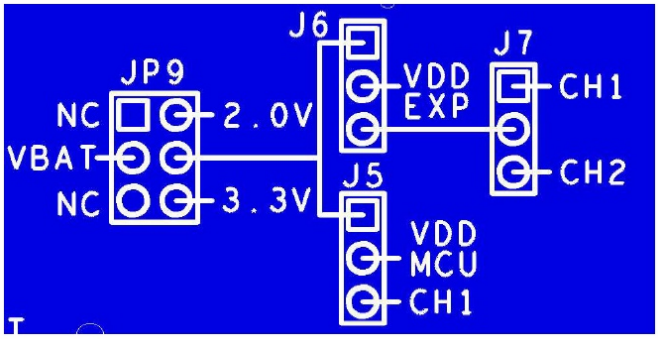
Once switched on, the 5V USB supply is then fed into 2 different LDO voltage regulators that can be used to power the Apollo SoC. One voltage regulator produces 2.0V (U12), and the other has an output of 3.3V (U13). If a user prefers an SoC supply voltage other than 2.0V or 3.3V, there are 2 test loops in the bottom left-hand corner of the board (TP4 and TP5) that are labeled VBAT. These can be used to power the SoC from a bench power supply, or battery source. The voltage at VBAT must not exceed the absolute maximum limits for the Apollo SoC.

Figure 2-4: Apollo Base Board Power Jumpers



The jumper settings for JP9 and J5 determine the supply routed to the SoC. On J5, pin 2 connects to VDD SoC which is the main MCU supply net. By default, a jumper shorts VDD SoC to pin 1 on J5, which connects it to JP9. On JP9, VBAT, 2.0 V, or 3.3 V can be routed to J5 by using a jumper to short pin 4 to one of several other pins, as shown in Figure 2-5.

Figure 2-5: Apollo Base Board Power Jumpers Silkscreen



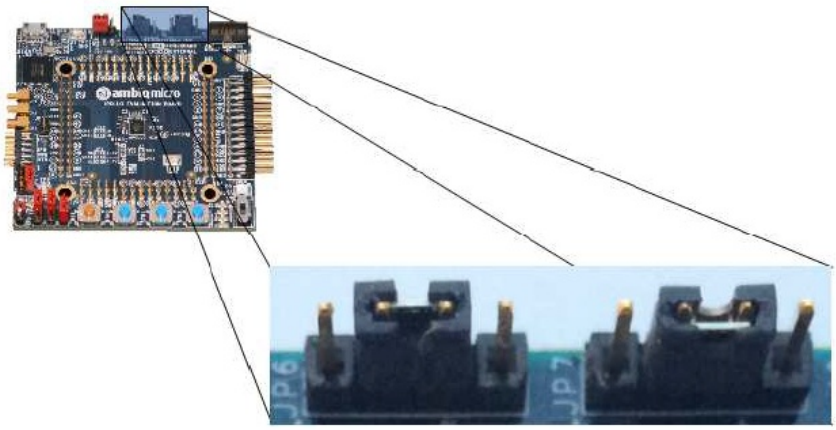
To measure power going into the SoC, remove the jumper on J5 and connect an ammeter in series with pins 1 and 2. This connection measures power exclusively into the SoC, and not to any other peripherals on the base board or other boards connected to it.

2.3 Debugger Connections

The Apollo EVK base board supports 3 different kinds of debugger connections:

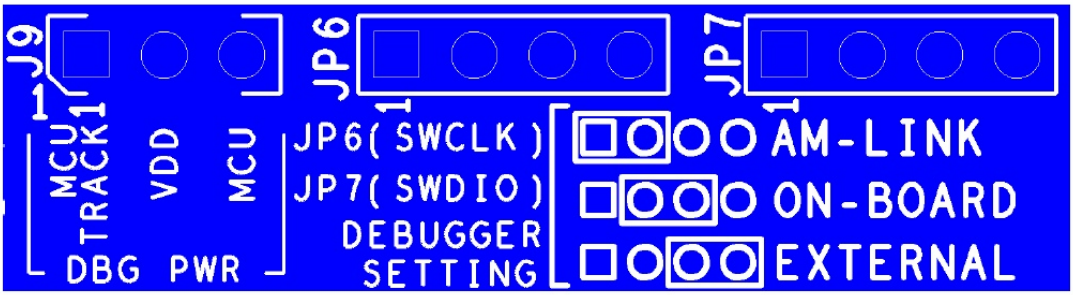
1. On-board, USB-based debugging
2. Use of an external debugging device like the ULINK2 from Keil, or I-jet from IAR Systems
3. Putting the board in AM-LINK mode, which allows the on-board USB debugger to be used to debug a separate, off-board device.

Figure 2-6: Apollo Base Board Debug Jumpers



Debugger options are selected by changing jumpers on the base board connected to 3 header: JP6, JP7, and J9. JP6 controls the SWCLK connection, JP7 controls SWDIO. J9 configures the power supply used by the debugger. The silkscreen just below the header/jumpers indicate the possible positions and functionality, as shown in Figure 2-7.

Figure 2-7: Apollo Base Board Debug Jumpers Silkscreen



By default, the base board is configured for option 1 above, using a USB connection to a PC running a debug driver (OpenOCD or Keil via AGDI). In this configuration, JP6 and JP7 have pins 2 and 3 shorted together.

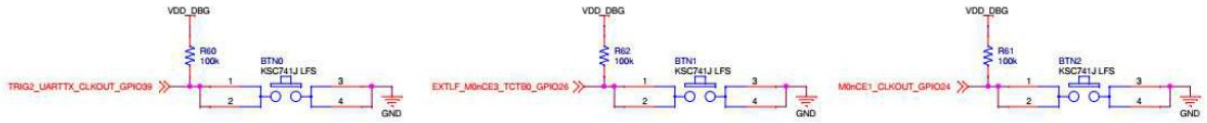
To use an external debug device move the jumpers on JP6 and JP7 to the right, connecting pins 3 and 4 on each. The debug device should connect to the Arm debug connector on the board (J8). For AM-LINK, move the jumpers to the left on JP6 and JP7 connecting pins 1 and 2 on each. In AM-LINK mode, the

on-board USB-based debugger can be connected to another board via the Arm debug connector.

J9 is normally configured to power the debugger from a regulator on the base board (U16) designed to track the SoC power supply but reduce by 200 mV to prevent leakage through pull up/down resistors, or back-powering of the SoC. The

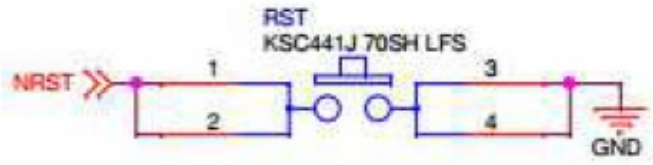
jumper on J9 can also be changed to power the debugger from the same supply as the SoC. Using the VDD SoC option will result in debugger power consumption being measured on J5, in addition to SoC consumption.

Figure 2-8: Schematic for Apollo Base Board Buttons (BTN0 through BTN2)



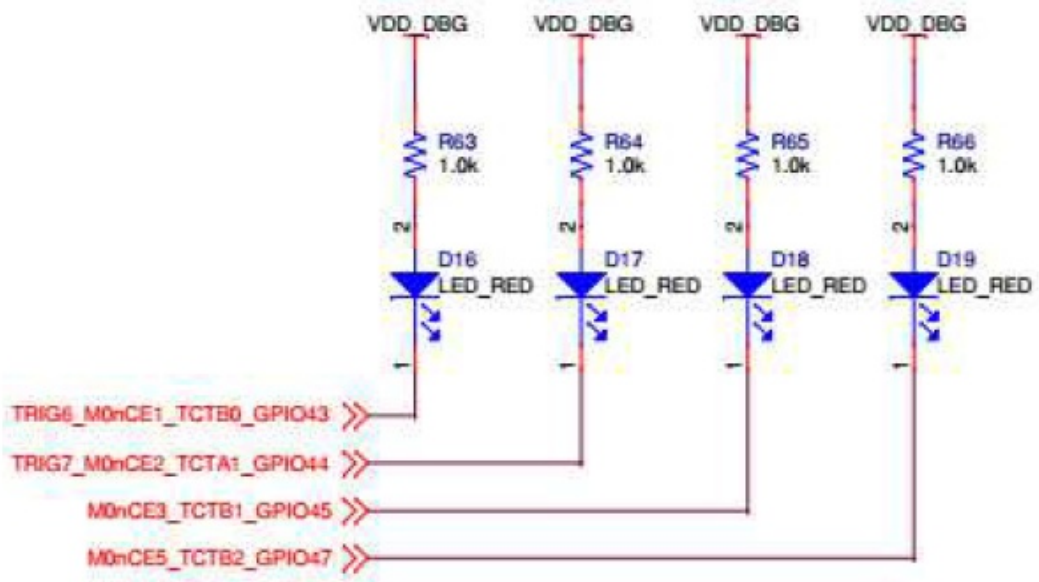
The nRST pin of the Apollo MCU is connected to an orange button (RST), which will connect the pin to ground when pressed to assert a reset.

Figure 2-9: Schematic for Apollo Base Board Reset Buttons



GPIO43, GPIO44, GPIO45, and GPIO47 are connected to the cathode of 4 red LEDs. Each LED has a 1k pullup resistor, which is connected to the power supply configured by J9, just like the buttons.

Figure 2-10: Schematic for Apollo Base Board LEDs



2.4 Design Files

Schematic, BOM, and layout information for the Apollo base board are freely available on the Ambiq website. Go to https://www.support.ambiq.com to register an account and gain access to the files.

2.5 Software Examples

Software examples designed specifically for evaluation of the Apollo SoC on the EVK base board ship with the AmbiqSuite SDK, available with Ambiq Control Center from the Ambiq website at: https://www.support.ambiq.com

Key code examples for Apollo low power evaluation include **`deepsleep`**, **`deepsleep_wake`**, **`coremark`**, and **`ulpbench`**.

A complete list of available code examples, designed to work with the EVK base board is as follows:

- **adc_vbatt**: This example initializes the ADC, and a timer. Two times per second it reads the VBATT voltage divider and temperature sensor and prints them. It monitors button 0 and if pressed, it turns on the BATT LOAD resistor. One should monitor SoC current to see when the load is on or off.
- **binary_counter**: This example increments a variable on every timer interrupt. The global variable is used to set the state of the LEDs. The example sleeps otherwise.
- **bootloader_targetA**: This is an example that is intended to be downloaded by a boot host into an Apollo chip running either ios boot or the secure boot loader. It is set to build at 0x8000 instead of 0x0. This example prints a "Hello World" style message marked TARGET A over SWO at 1M baud. To see the output of this program, run AMFlash, and configure the console for SWO. The example sleeps after it is done printing. It generates a different repeating output message than **bootloader_targetB**.
  - **bootloader_targetA** is an example that is intended to be downloaded by a boot host into an Apollo chip running either ios boot or the secure bootloader. It is set to build at 0x8000 instead of 0x0. **bootloader_targetA** is an example that is intended to be downloaded by a boot host into an Apollo chip running either ios boot or the secure boot loader. It is set to build at 0x8000 instead of 0x0. Use the bash script **generate_boot_image.sh** to get an OTA file for download.
  - **bootloader_targetA** and **bootloader_targetB** are nearly identical programs. The host program should download targetA into a fresh Apollo SoC with a brand new boot loader in it. After a reset it should be continuously printing "TARGET A: I was downloaded via the boot loader". In order to test the host ability to over ride an existing program, one should download **bootloader_targetB** to get "TARGET B: I was downloaded via the boot loader" in order to confirm a successful update operation.
- **bootloader_targetB**: This is an example that is intended to be downloaded by a boot host into an Apollo chip running either ios boot or the secure boot loader. It is set to build at 0x8000 instead of 0x0. This example prints a "Hello World" style message marked TARGET B over SWO at 1M baud. To see the output of this program, run AMFlash, and configure the console for SWO. The example sleeps after it is done printing. It generates a different repeating output message than **bootloader_targetA**.
  - **bootloader_targetB** is an example that is intended to be downloaded by a boot host into an Apollo chip running either ios boot or the secure bootloader. It is set to build at 0x8000 instead of 0x0. **bootloader_targetB** is an example that is intended to be downloaded by a boot host into an Apollo chip running either ios boot or the secure boot loader. It is set to build at 0x8000 instead of 0x0. Use the bash script generate boot image.sh to get an OTA file for download.

  - **bootloader_targetA** and **bootloader_targetB** are nearly identical programs. The host program should download targetA into a fresh Apollo SoC with a brand new boot loader in it. After a reset it should be continuously printing "TARGET A: I was downloaded via the boot loader". In order to test the host ability to over ride an existing program, one should download **bootloader_targetB** to get "TARGET B: I was downloaded via the boot loader" in order to confirm a successful update operation.

- **clkout**: This example enables the LFRC to a CLKOUT pin then uses GPIO polling to track its rising edge and toggle an LED at 1/2 hertz.
- **coremark**: This example runs the official EEMBC COREMARK test.
- **deepsleep**: This example configures the device to go into a deep sleep mode. Once in sleep mode the device has no ability to wake up. This example is merely to provide the opportunity to measure deepsleep current without interrupts interfering with the measurement.
- **deepsleep_wake**: This example configures the device to go into a deep sleep mode. Once in deep sleep the device has the ability to wake from button 0 or the RTC configured to interrupt every second. If the SoC woke from a button press, it will toggle LED0. If the SoC woke from the RTC, it will toggle LED1.
- **flash_write**: This example shows how to modify the internal Flash using HAL flash helper functions.This example works on instance 1 of the Flash, i.e. the portion of the Flash above 256 KB.
- **hello_fault**: This example demonstrates the extended hard fault handler which can assist the user in decoding a fault condition. The handler pulls the registers that the Cortex M4 automatically loads onto the stack and combines them with various other fault information into a single data structure saved locally. It can optionally print out the fault data structure (assuming the stdio printf has previously been enabled and is still enabled at the time of the fault).
- **hello_world**: This example prints a "Hello World" message with some device info over SWO at 1M baud. To see the output of this program, run AMFlash, and configure the console for SWO. The example sleeps after it is done printing.
- **hello_world_uart**: This example prints a "Hello World" message with some device info over UART at 115200 baud. To see the output of this program, run AMFlash, and configure the console for UART. The example sleeps after it is done printing.
- **iomi2c_host_side**: This is the main program. Refer to **iomi2c_host_side_driver** for the details of the actions an Android or Nucleus kernel driver has to do to talk to the Apollo slave for sensor hub applications. This application simply provides stimulus for the driver code in **iomi2c_host_side_driver**.c. This examples passes messages to the slave board where they are echoed back. See the **i2cios_hub** application as an example slave device to talk with.
- **ios_boot**: I/O Slave (I^2C or SPI) based boot loader for sensor hub like devices. This bootloader is intended to reside permanently in the beginning of flash onan Apollo SoC used as a sensor hub like device, i.e. attached to a host appli-

cation processor. The AP can download new applications to the flash in the Apollo processor whenever it desires. This bootloader implementation support the I/O slave and be conditionally compiled to use either I^2C mode or SPI mode of the I/O slave.

- **`iosi2c_hub`**: This is the main program. Refer to **`iosi2c_hub_driver`** for the details of the actions an Android or Nucleus kernel driver has to do to talk to the Apollo slave for sensor hub applications. See the **`iomi2c_host_side`** application for a host example that can be used with this example.
- **`itm_printf`** : This example walks through the ASCII table (starting at character 033('!') and ending on 126("~')) and prints the character to the ITM. This output can be decoded by running AM Flash and configuring the console for SWO at 1M Baud. This example works by configuring a timer and printing a new character after ever interrupt and sleeps in between timer interrupts.
- **`pwm_gen`**: This example shows one way to vary the brightness of an LED using timers in PWM mode.
- **`reset_states`**: This example shows a simple configuration of the watchdog. It will print a banner message, configure the watchdog for both interrupt and reset generation, and immediately start the watchdog timer. The watchdog ISR provided will 'pet' the watchdog four times, printing a notification message from the ISR each time. On the fifth interrupt, the watchdog will not be pet, so the 'reset' action will eventually be allowed to occur. On the sixth timeout event, the WDT should issue a system reset, and the program should start over from the beginning.
- **`rtc_print`**: This example demonstrates how to interface with the RTC and prints the time over SWO. The example works by configuring a timer interrupt which will periodically wake the core from deep sleep. After every interrupt, it prints the current RTC time.
- **`timers`**: This example demonstrates how to setup the ctimer for counting and interrupts. It toggles LED 0 every interrupt.
- **`timer_plot`**: This example plots the value of a variable using AM Flash. This works by configuring a timer interrupt, starting the timer, tracking in a variable the number of times interrupts occur, and then plotting various bits from that count. The value plotted depends on the axis.
- **`uart_printf`** : This example walks through the ASCII table (starting at character 033('!') and ending on 126(")) and prints the character to the UART. This output can be decoded by running AM Flash and configuring the console for UART at 115200 Baud. This example works by configuring a timer and printing a new character after ever interrupt and sleeps in between timer interrupts.
- **`uart2spi`**: This example accepts STXETX protocol packets from the buffered UART and turns them into SPI transactions on I/O Master 0. In addition, it monitors one GPIO line for interrupts from an Apollo's I/O slave. It will also handle the reset/SPICLK protocol to force a boot loader in to BL mode or in to application mode. It is intended to run the HOST EMULATION side of a pair of Apollo EVK boards. The other one is the sensor hub with boot loader installed.

- **`ulpbench`**: This example runs the official EEMBC ULPBench Core Profile.
- **`ulpbench_lfrc`**: This example runs the EEMBC ULPBench Core Profile clocking the RTC module with the internal LFRC.
- **`vcomp_interrupts`**: This example initializes the voltage comparator, enables its interrupts then captures transition interrupts and signal them to the base level via a very simplistic method where it then reports the transitions on the printf stream. When connected to a suitable RC network on the control pins, this example creates a relaxation oscillator. The Apollo EVK base board contains one example of a relaxation oscillator circuit. One can monitor the voltage waveform on pin 18 to see the relaxation oscillator charge/discharge cycle.
- **`watchdog`**: This example shows a simple configuration of the watchdog. It will print a banner message, configure the watchdog for both interrupt and reset generation, and immediately start the watchdog timer. The watchdog ISR provided will 'pet' the watchdog four times, printing a notification message from the ISR each time. On the fifth interrupt, the watchdog will not be pet, so the 'reset' action will eventually be allowed to occur. On the sixth timeout event, the WDT should issue a system reset, and the program should start over from the beginning.

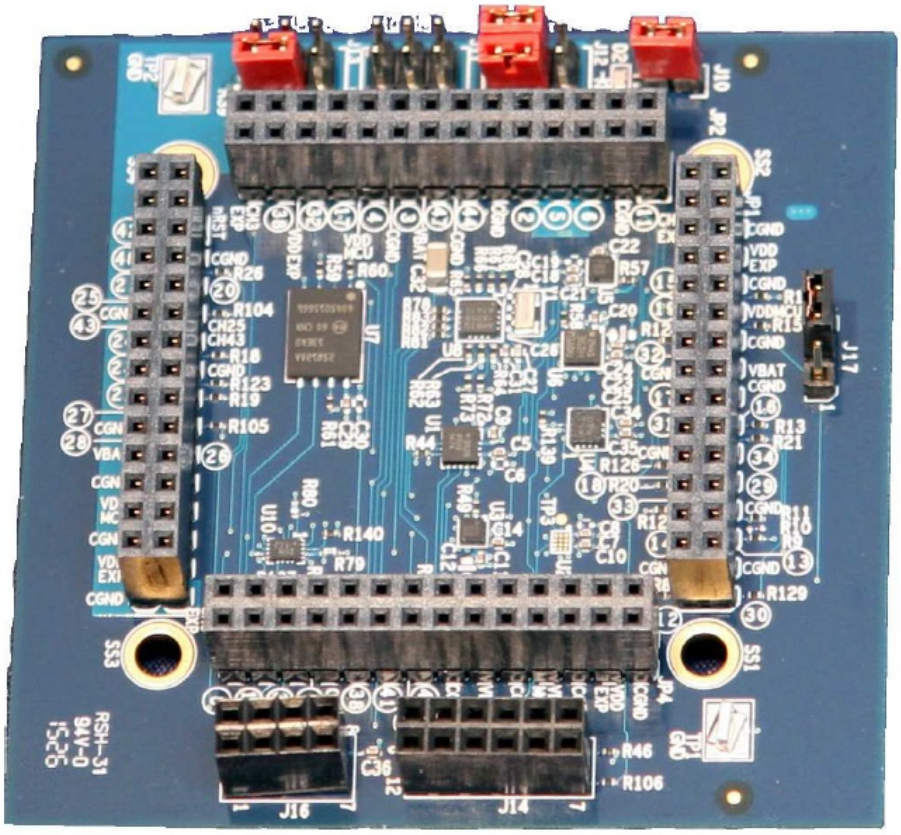
SECTION

3

Apollo Sensor Board

The Apollo sensor board serves as a vehicle for rapid prototyping of sensor-based applications. It connects to the Apollo base board via JP1, JP3, JP4, and JP5 for a quick connection to SoC peripheral pins, GPIOs, power, and ground signals. Included accelerometer, gyroscope, and magnetometer devices allow for up to 9 axes of sensor data collection and processing.

Figure 3-1: Apollo EVK Sensor Board



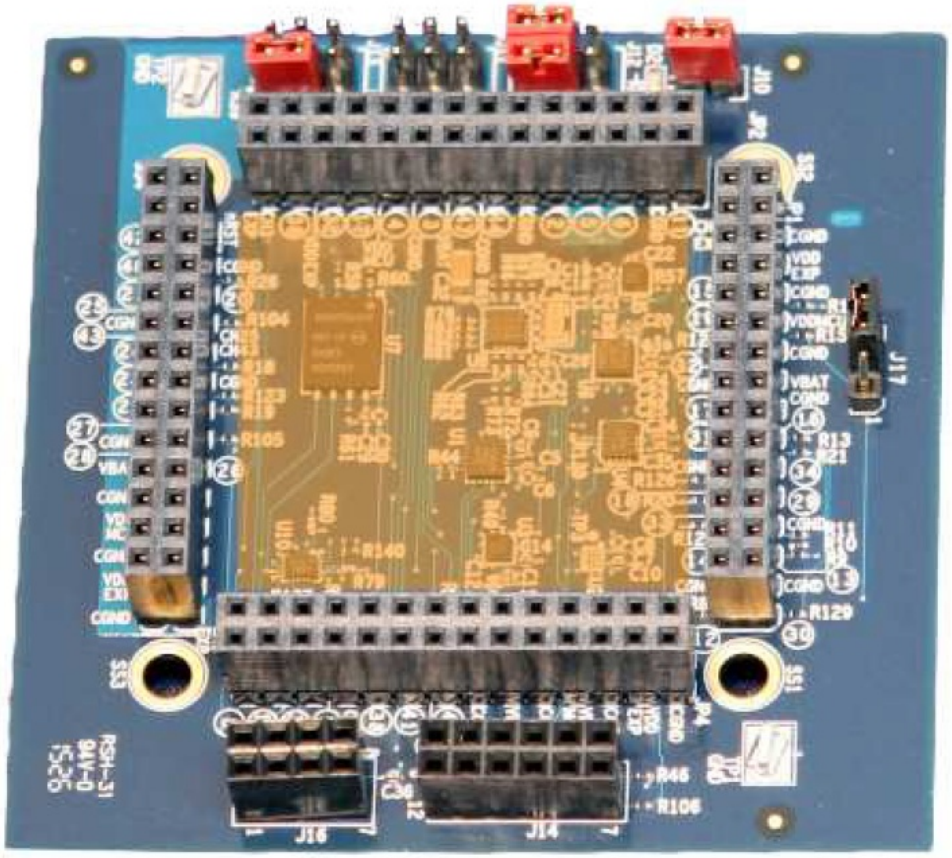
3.1 On-board Devices

On the Apollo sensor board are 5 different sensors:

- Analog Devices ADXL362, and ST Micro LIS2DH12 accelerometers
- Bosch BMI160 combination accelerometer and gyroscope
- ST Micro L3GD20H gyroscope
- ST Micro LIS3MDL magnetometer

In addition to these sensors are a 128 Mb SPI flash memory, and the AM1805 ultra-low power RTC.

Figure 3-2: Apollo EVK Sensor Board with Sensor, Flash Memory, and RTC Highlighted

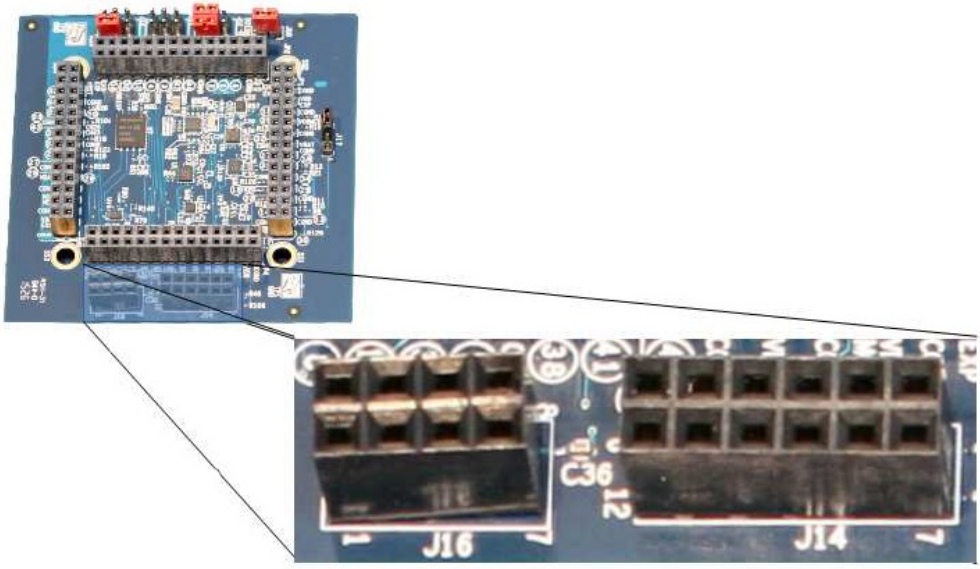


3.2 Expansion Header

The Apollo sensor board replicates the expansion header connections of the base board for JP1, JP2, JP4, and JP5. It does not include a right-angle header connection (JP3 on the base board).

Two expansion sockets are also included on the board. J14 provides SPI connections suitable for connection to Digilent PMOD I/O interface boards. J16 provides similar connections but for I^2C. With J14 and J16, users can extend the sensor capabilities of the board to include dozens of devices.

Figure 3-3: Apollo EVK Sensor Board PMOD Connection



3.3 Supplies and Power Measurements

Figure 3-4: Apollo EVK Sensor Board Power Supply and Measurement Jumpers

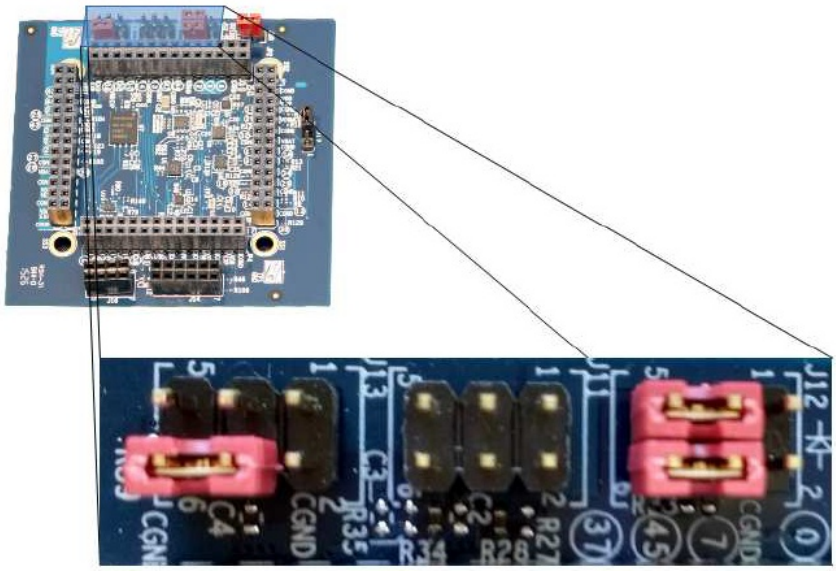


Figure 3-4: Apollo EVK sensor board power supply and measurement jumpers J12 selects the power supply for the devices on the sensor board. By default, pins 3 and 5 are connected together which connects VDD EXP (coming from the Apollo base board) to VDD IO (the power net for all devices on the sensor board). Pins 4 and 6 are also connected by default, which connects VDD EXP to the power supply for the on-board LED (D2) and SPI Flash (U7).

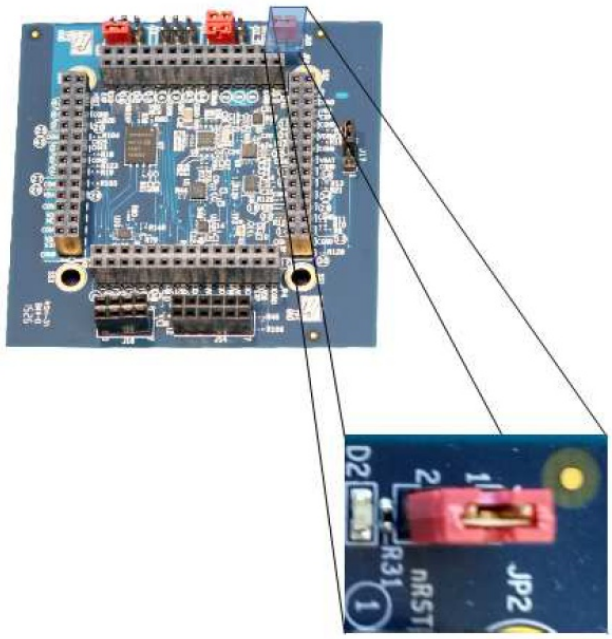
J11 can be used for powering sensors via an Apollo high side power switch GPIO (GPIO 3 or GPIO4). There is also a connection to CH3 EXP, which will be used in the future for certain power measurement features.

The last supply-related header is J13, which allows the ground for all sensors on the board to be connected to Apollos low side GPIO power switch (GPIO11). Its also possible to connect the sensor board ground to CH3 EXP, for a future power measurement feature. By default, the ground coming from the Apollo base board (CGND, pin 6), and the ground of the sensor devices on the board (GND, pin 4) are shorted together via a jumper.

3.4 LED

A single LED, D2, on the sensor board indicates power is successfully applied to the board. A jumper connected to J10 enables the LED to be used by default but can be pulled to prevent the LED from interfering with power measurements.

Figure 3-5: Apollo EVK Sensor Board LED and LED Power Jumper



3.5 Design Files

Schematic, BOM, and layout information for the Apollo sensor board are freely available on the Ambiq website. Go to https://support.ambiq.com to register an account and gain access to the files.

3.6 Software Examples

Software examples designed specifically for evaluation of the Apollo in conjunction with the devices on the sensor board ship with the AmbiqSuite SDK, available with Ambiq Control Center from the Ambiq website at: https://support.ambiq.com

Key code examples for sensor low power evaluation include **`adxl362_read`**, **`am1805_time`**, **`bmi160_read`**, and **`data_plot`**.

A complete list of available code examples, designed to work with the EVK sensor board is as follows:

- **`adxl362_read`**: Configures the adxl362 to sample at 100Hz and set its watermark based on the **`ADXL362_SAMPLE_SIZE`** define. When the adxl362 FIFO hits its watermark, an interrupt line asserts causing the Apollo SoC to interrupt and begin draining the FIFO while sleeping and periodically waking to empty the internal IOM. The samples are plotted over the ITM. While moving the EVK board, use AM Flash (click **Show Plot Window**) to view the real-time plot.

- **`am1805_time`**: Example that sets the initial time on the AM1805, configures the countdown timer for 500ms, which is used to wake up Apollo. Once Apollo is awake, the app reads and prints the time from the AM1805 on the ITM port at 1 MBaud.
- **`bmi160_read`**: Configures the bmi160 to sample at 100Hz and set its watermark based on the **`BMI160_SAMPLE_SIZE`** define. When the bmi160 FIFO hits its watermark (data-ready if sample size equals 1), an interrupt line rises causing the Apollo SoC to interrupt and begin draining the FIFO while sleeping and periodically waking to empty the internal IOM. The samples are plotted over the ITM. Use AM FLash to view the real-time plot.
- **`data_plot`**: This FreeRTOS example configures the adxl362 to sample at 400Hz and when ready reads the samples from the FIFO and sends them over ITM to be plotted using AMFlash.
- **`l3gd20h_read`**: Configures the l3gd20h to sample at 100Hz and set its watermark based on the **`L3GD20H_SAMPLE_SIZE`** define. When the l3gd20h FIFO hits its watermark (data-ready if sample size equals 1), an interrupt line rises causing the Apollo SoC to interrupt and begin draining the FIFO while sleeping and periodically waking to empty the internal IOM. The samples are plotted over the ITM. Use AM FLash to view the real-time plot.
- **`lis2dh12_read`**: Configures the lis2dh12 to sample at 100Hz and set its watermark based on the **`LIS2DH12_SAMPLE_SIZE`** define. When the lis2dh12 FIFO-hits its watermark (data-ready if sample size equals 1), an interrupt line rises causing the Apollo SoC to interrupt and begin draining the FIFO while sleeping and periodically waking to empty the internal IOM. The samples are plotted over the ITM. Use AM FLash to view the real-time plot.
- **`lis3mdl_read`**: Configures the lis3mdl to sample at 100Hz and interrupt on data-ready causing the Apollo SoC to interrupt and retrieve the sample. The samples are plotted over the ITM. While moving the EVK board, use AM Flash (click **Show Plot Window**) to view the real-time plot.
- **`spiflash_example`**: This example attempts to perform a series of read and write operations to an external spiflash. Success or failure of each operation is reported over ITM.

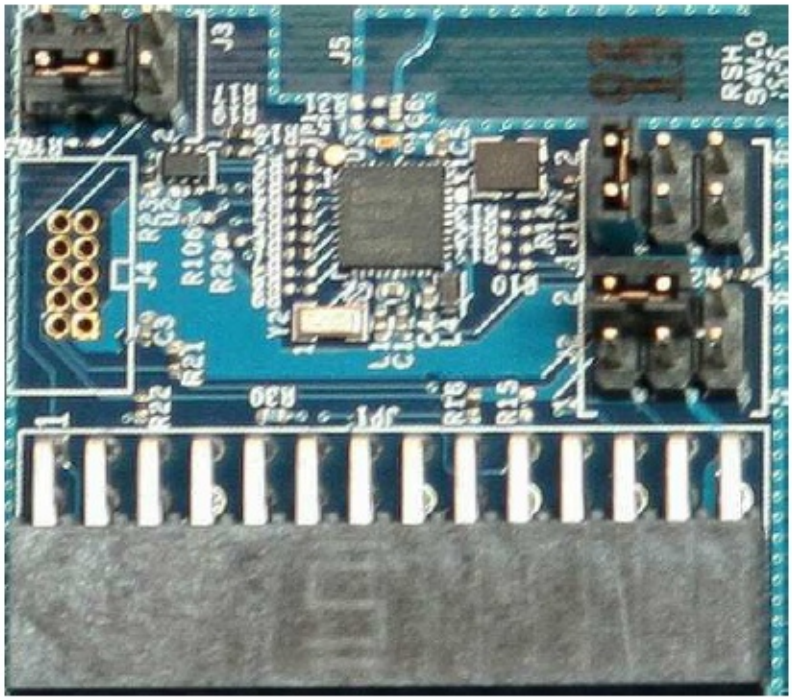
SECTION

4

Apollo Bluetooth Low Energy Radio Board

Users can explore Bluetooth Low-Energy connectivity using the Apollo Bluetooth Low Energy radio board. It connects to Apollo via the right-angle, JP2 connector so the on-board PCB antenna has a clear space for transmission and reception. JP2 also provides a set of GPIO and power connections for robust control of the radio device.

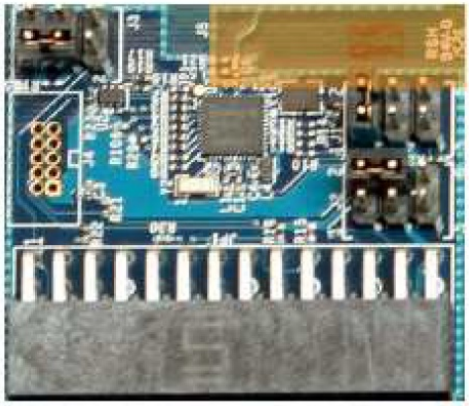
Figure 4-1: Apollo EVK Bluetooth Low Energy Board



4.1 Bluetooth Low Energy Radio and Antenna

The Bluetooth Low Energy radio board includes a DA14581 low power Bluetooth smart SoC. On the PCB is an inverted-F PCB antenna (J5) for easy Bluetooth Low Energy reception and transmission with minimal BOM cost.

Figure 4-2: Apollo EVK Bluetooth Low Energy Board, with PCB Antenna Highlighted

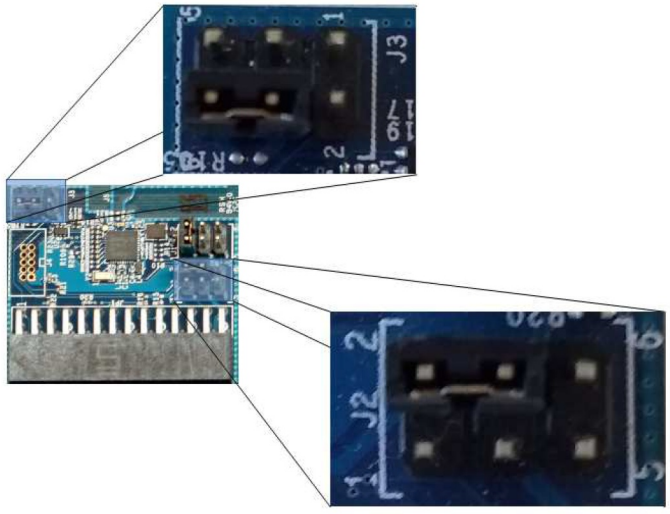


4.2 Supplies and Power Measurements

The J2 jumper on the radio board selects the power supply for the DA14581. By default, a jumper between pins 2 and 4 connects Apollos GPIO3 high side power switch GPIO to the SoCs power supply. The VDD EXP supply from the Apollo base board can also be selected as a power supply, or GPIO4.

J3 selects the ground for the DA14581. By default, a jumper shorts the Apollo base boards ground (CGND, pin 6) to the SoCs ground (GND, pin 4). The SoCs ground can also be connected to Apollos low side GPIO power switch (GPIO11).

Figure 4-3: Apollo EVK Bluetooth Low Energy Board, J2 and J3 Jumpers



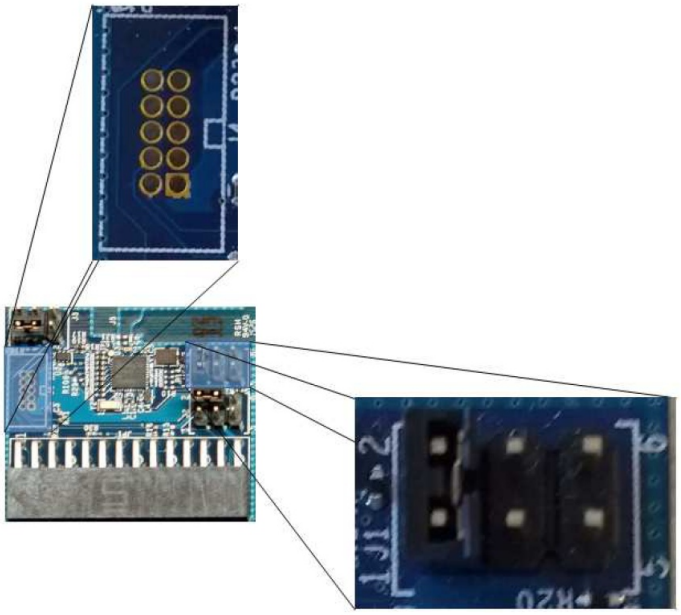
In the future CH3 EXP can also be selected by J2 or J3 for a future power measurement feature.

4.3 Software Debugging and Reset

Connector J4 is not installed by default on the Apollo Bluetooth Low Energy board, but can optionally be installed by the user to add software debug capability to the board. The DA14581 contains an SoC core that can be debugged using this connector.

Header J1 selects the reset signal source to the DA14581. A jumper selects Apollo GPIO17 as the default reset signal, but the reset signal on J4 can be selected instead by shorting pins 5 and 6 together.

Figure 4-4: Apollo EVK Bluetooth Low Energy Board, Software Debug Header and JI Reset Jumper



4.4 Design Files

Schematic, BOM, and layout information for the Apollo Bluetooth Low Energy radio board are freely available on the Ambiq website. Go to https://support.ambiq.com to register an account and gain access to the files.

4.5 Software Examples

Software examples designed specifically for evaluation of the Apollo in conjunction with the DA14581 ship with the AmbiqSuite SDK, available with Ambiq Control Center from the Ambiq website at: https://support.ambiq.com

The key code example for Bluetooth Low Energy evaluation is exactle fit.

A complete list of available code examples, designed to work with the EVK Bluetooth Low Energy board is as follows:

- **`exactle_fit`**: This example application implements the standard Bluetooth Low Energy HRP profile using the ExactLE stack and the Dialog DA14581 Bluetooth Low Energy radio. This application is able to communicate with standard heart-rate applications running on recent model mobile devices. In this example implementation, the heart rate value is reported as a constant "78", and the "kCals consumed" value is reported as a single incrementing integer value. In a real application, these values could be supplied by a heart-rate sensor and context-tracking software.
- **`dsps_print`**: This example uses the Dialog Serial Port Service Profile to send UART data to a mobile device over a Bluetooth Low Energy connection.

SECTION

5

Errata

This section describes errata that affect functionality of the Apollo EVK and EVB.

5.1 Increased Deep Sleep Current on Base Board

5.1.1 Issue Description

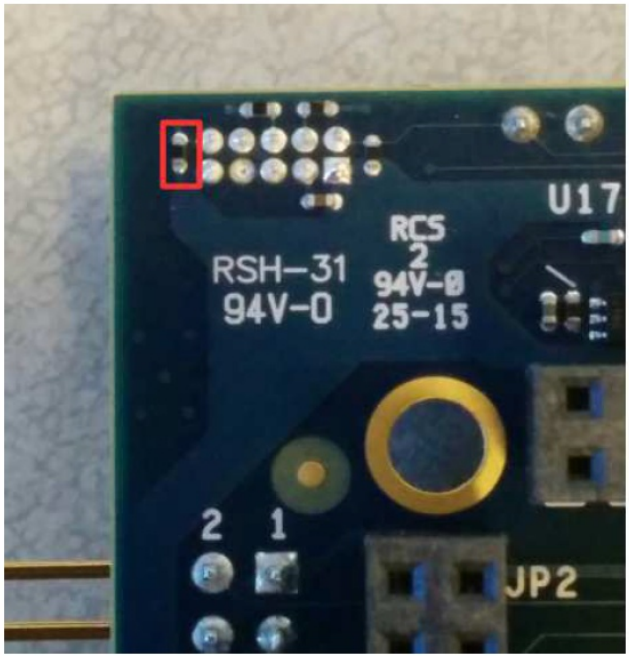
When placing the Apollo SoC into the deepsleep mode, for instance using the deepsleep code example, the current consumption of the SoC may be twice the value expected: between 200 nA and 300 nA. Base boards revision 3.0.0 or earlier may exhibit this issue.

A pullup resistor connected to the reset pin pulls up to the debugger voltage supply, which is lower than the SoC supply by about 200 mV. The SoC has its own internal pullup on this pin, tied to the SoC supply. As a result, leakage from the SoC's supply to the lower debugger supply creates a constant additional current that is most noticeable when the SoC is in its lowest power state.

5.1.2 Workaround

Customers who encounter this issue should remove resistor R1 on the Apollo base board. Located in the top left-hand corner on the bottom side of the board next to J8.

Figure 5-1: Location of R1 highlighted on bottom side of Apollo base board, next to J8



5.1.3 Resolution

A new base board revision, 3.1.0, will be released that does not have this resistor populated by default on the BOM to prevent the leakage condition.

5.2 Incorrect Silkscreen Indicating SoC Supply Voltages on Base Board

5.2.1 Issue Description

Apollo EVK base board revisions 3.0.0 and earlier display 1.8 V, and 3.0 V on the silkscreen for JP9 instead of 2.0 V and 3.3 V. The BOM is configured on these boards to provide 2.0 V and 3.3 V, but the silkscreen is in error.

5.2.2 Workaround

Customers should continue to use JP9 as needed, but take note that the 1.8 V indication on JP9 actually provides 2.0 V, and 3.0 V is actually 3.3 V.

5.2.3 Resolution

A new base board revision, 3.1.0, will be released that corrects this error in the silkscreen.

5.3 Increased System Current on Bluetooth Low Energy Board

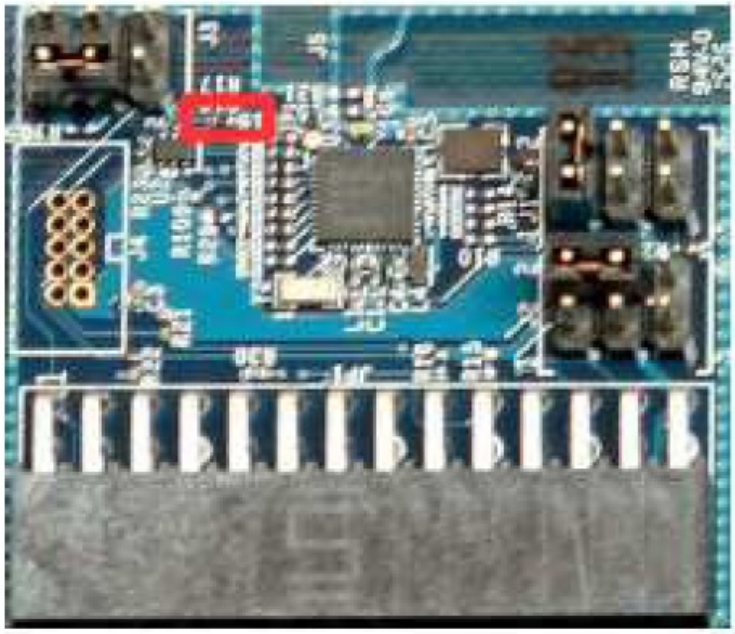
5.3.1 Issue Description

When measuring power consumption using the Bluetooth Low Energy board connected to the Apollo base board, overall current consumption is higher than expected at around 30 µA. A pulldown resistor (R17), included on the Bluetooth Low Energy board's reset inverter circuit creates a leakage path to ground when GPIO6 is held high. Since GPIO6 is part of the UART bus connecting Apollo to the Bluetooth Low Energy for HCI communication, this line is high for a significant portion of time creating near-constant leakage current.

5.3.2 Workaround

Customers who encounter this issue should remove resistor R17 on the Apollo EVK Bluetooth Low Energy board. R17 is located on the top side of the board, directly beneath J3.

Figure 5-2: Location of R17 highlighted on top side of Apollo EVK BLE Board



5.3.3 Resolution

A new Bluetooth Low Energy board revision, 2.1.0, will be released that does not have this component populated by default which will prevent the leakage going forward.

5.4 Increased system current on sensor board

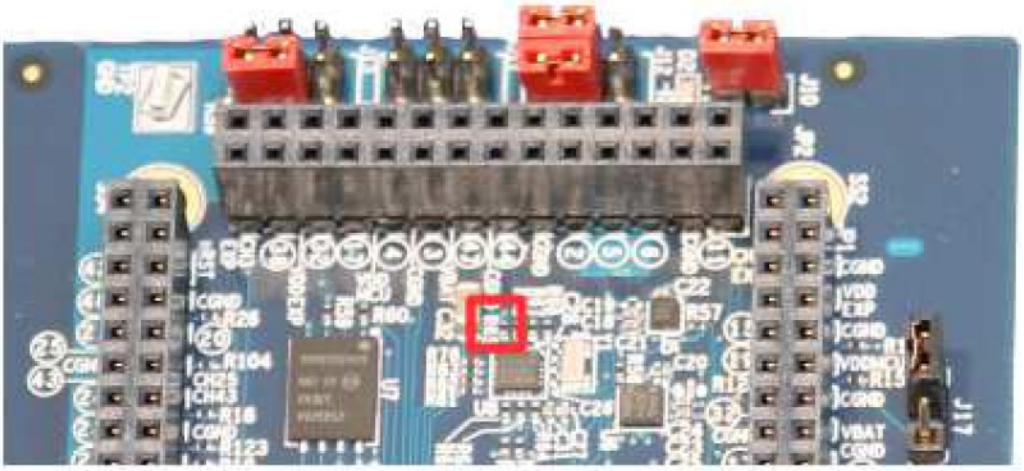
5.4.1 Issue Description

When measuring system power on the sensor board, power consumption can be higher than expected at around 340 µA from the 3.0 V supply. A pullup resistor (R65) connects the PSW pin of the AM1805 RTC to VDD IO. Upon startup, the RTC activates the power switch on the PSW pin pulling this node to ground which creates the additional leakage through R65.

5.4.2 Workaround

Customers who encounter this issue should remove resistor R65 on the Apollo EVK sensor board. R65 is located directly above the AM1805 (U8).

Figure 5-3: Location of R65 Highlighted on the Apollo EVK Sensor Board



5.4.3 Resolution

A new sensor board revision, 2.2.0, will be released that does not have this component populated by default which will prevent the leakage going forward.

6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730
www.ambiq.com
sales@ambiq.com
+1 (512) 879-2850

A-SOCAP1-UGGA01EN v1.3
January 2023