AVR-BLE Hardware

AVR-BLE Hardware User Guide

Preface

The AVR-BLE Development Board is a small and easily expandable demonstration and development platform for Bluetooth® Low Energy (BLE) solutions based on the AVR® microcontroller architecture. It is designed to demonstrate that the design of a typical BLE application can be simplified by partitioning the task into three blocks:

- Smart – represented by the ATmega3208 microcontroller
- Secure – represented by the ATECC608A secure element
- Connected – represented by the RN4870 BLE module

In addition, the AVR-BLE Development Board features the following elements:

- The on-board debugger (PKoB nano) supplies full programming and debugging support through Atmel Studio/ Microchip MPLAB® X IDE. It also provides access to a serial port interface (serial to USB bridge) and two logic analyzer channels (debug GPIO).
- A mikroBUS™ socket enables the ability to expand the board capabilities with the selection from 450+ sensors and actuators options offered by MikroElektronika (www.mikroe.com) via a growing portfolio of Click board™.

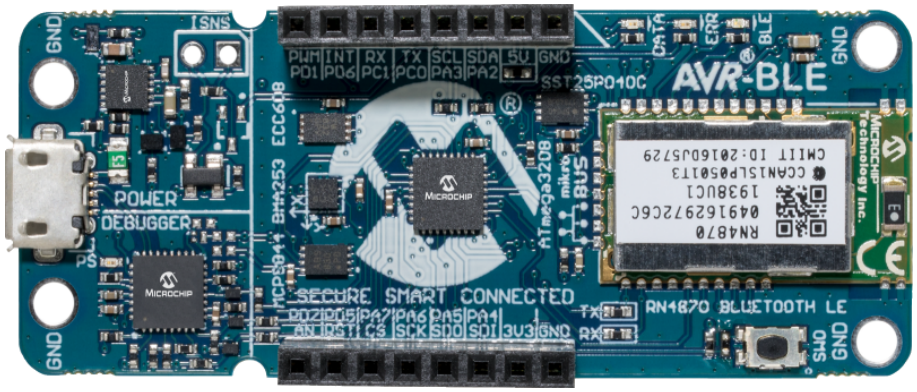


Table of Contents

1. Introduction

1.1 Features

- ATmega3208 AVR Microcontroller
- Two User LEDs (Data and Error)
- Mechanical Button
- RN4870 Bluetooth Low Energy (BLE) Module
- MCP9844 Temperature Sensor
- BMA253 Acceleration Sensor
- ATECC608A CryptoAuthentication™ Device
- SST25PF040CT 4Mb Serial Flash
- mikroBUS Socket
- On-board Debugger
  - Board identification in Atmel Studio/Microchip MPLAB® X IDE
  - Programming and debugging
  - Virtual serial port (USB CDC)
  - Two logic analyzer channels (DGI GPIO)
- USB or Battery Powered

1.2 Board Overview

The AVR-BLE development board is a hardware platform that is being used to evaluate the ATmega3208 AVR microcontroller and RN4870 BLE module.

Figure 1-1. AVR-BLE Development Board Front Side

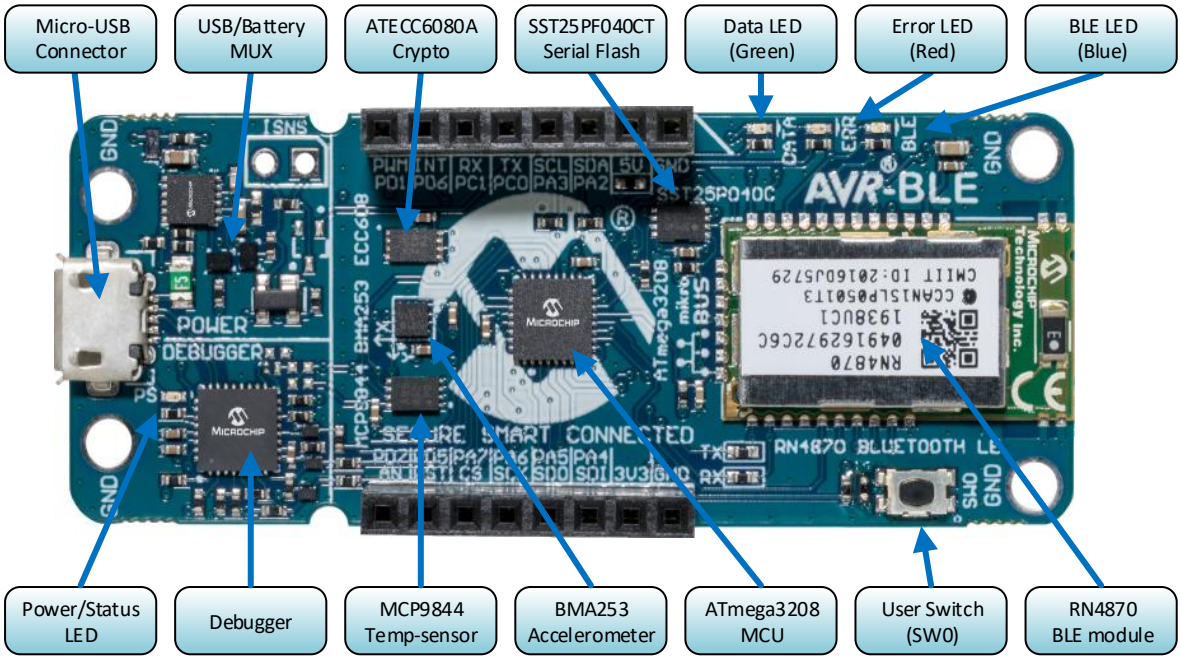


Figure 1-2. AVR-BLE Development Board Back Side

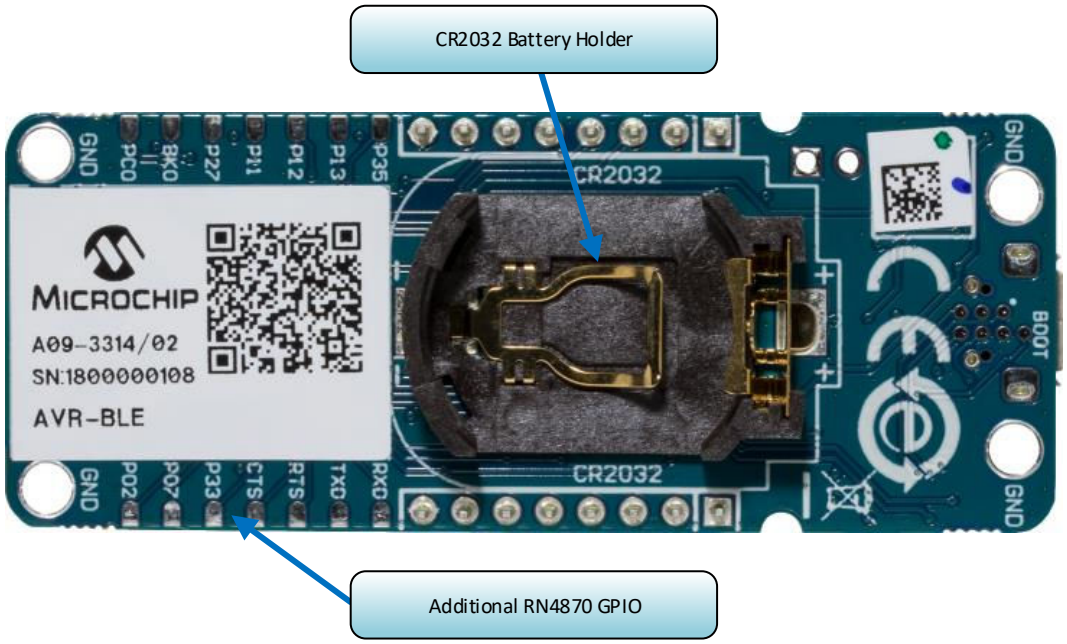
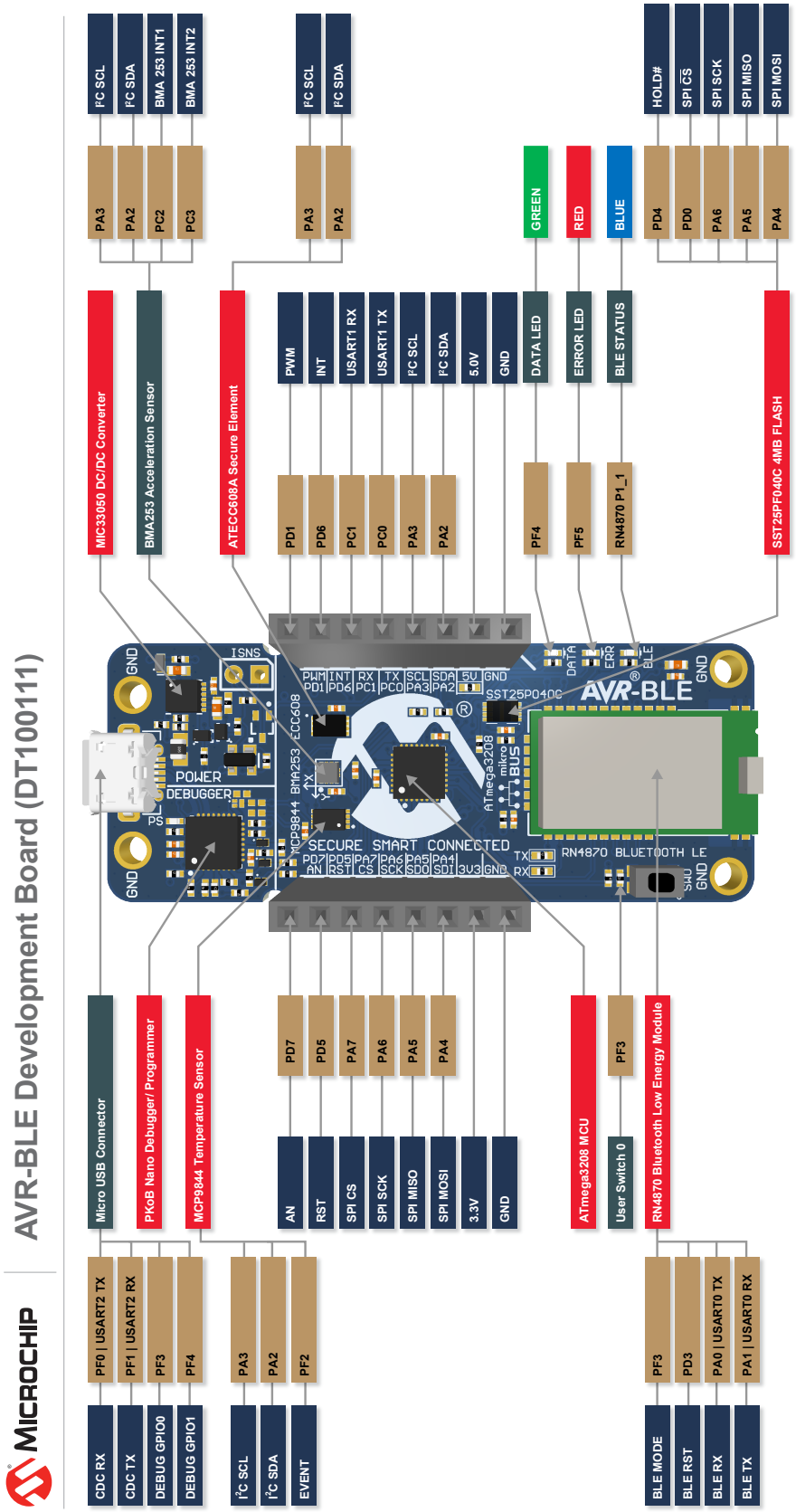


Figure 1-3. AVR-BLE Quick Reference Overview



2. Getting Started

2.1 Quick Start

Demo Application

Out of the box, the AVR-BLE board comes programmed with the avr-lightblue-explorer-demo. This application can be used to demonstrate a number of the board features using the LightBlue® app by Punch Through.

1. Download the latest demo firmware from the releases section on avr-lightblue-explorer-demo GitHub page.
2. Connect a USB cable (Standard-A to Micro-B or Micro-AB) between the PC and the debug USB port on the board.
3. Program the AVR-BLE with the downloaded `.hex` file through the drag-and-drop feature.
4. Download the LightBlue® app for iOS or Android.
5. Power the board through a Micro-USB cable or CR2032 battery.
6. Open the LightBlue® app and select the AVR-BLE peripheral.
7. Use the custom interface to explore the board.

Info: The AVR-BLE will show up in the LightBlue® app as AVR-BLE_xxxx, where xxxx are the last two bytes of the RN4870 BLE module's Bluetooth MAC address. This makes it possible to distinguish between multiple AVR-BLE boards.

Communication between the demo application and the LightBlue® app is done by using a protocol based on ASCII packets. Refer to the protocol chapter on the avr-lightblue-explorer-demo page for a list of commands with examples, as well as the full source code for the project.

Development Requirements

MPLAB® X IDE:

- MPLAB X IDE v5.30 or later
- XC8 Compiler v2.10 or later

For help with installation, view the MPLAB X installation guide.

Build an Application

View the default source code that is pre-loaded onto the development board. Explore, modify, and build off this source code to create a custom application.

1. View the source code at the avr-lightblue-explorer-demo GitHub page.
2. Read through the README.md to get more information on how to expand the solution.
3. Download the project from GitHub and open it in the latest version of MPLAB® X IDE.
4. Connect a USB cable (Standard-A to Micro-B or Micro-AB) between the Windows, Mac or Linux device, and the debug USB port on the AVR-BLE. The board will be identified in the kit window in MPLAB® X IDE.
5. Explore, modify, and build off the source code.
6. Make and program the device. Select the PKoB nano serial number as the debug tool when prompted.

Driver Installation

When the board connects to your computer for the first time, the operating system will perform a driver software installation. The driver file supports both 32- and 64-bit versions of Microsoft® Windows® XP, Windows Vista®, Windows 7, Windows 8, and Windows 10. The drivers for the board are included with both MPLAB® X IDE and Atmel Studio 7.

Kit Window

Once the board is powered, the green status LED will lit, and both MPLAB® X IDE and Atmel Studio 7 will auto-detect which boards are connected. The Kit Window in MPLAB® X IDE and Atmel Studio 7 will present relevant information

like data sheets and board documentation. The ATmega3208 device on the AVR-BLE board is programmed and debugged by the on-board debugger and, therefore, no external programmer or debugger tool is required.

Tip: If closed the Kit Window in MPLAB® X IDE can be reopened through the menu bar *Window > Kit Window*.

2.2 Design Documentation and Relevant Links

The following list contains links to the most relevant documents and software for the AVR-BLE board:

- **MPLAB® X IDE** - MPLAB X IDE is a software program that runs on a PC (Windows®, Mac OS®, Linux®) to develop applications for Microchip microcontrollers and digital signal controllers. It is called an Integrated Development Environment (IDE) because it provides a single integrated “environment” to develop code for embedded microcontrollers.
- **Atmel Studio** - Free IDE for the development of C/C++ and assembler code for microcontrollers.
- **IAR Embedded Workbench® for AVR®** - This is a commercial C/C++ compiler that is available for AVR microcontrollers. There is a 30-day evaluation version as well as a 4 KB code-size-limited kick-start version available from their website.
- **MPLAB® XC Compilers** - MPLAB® XC8 C Compiler is available as a free, unrestricted-use download. Microchips MPLAB® XC8 C Compiler is a comprehensive solution for your project’s software development on Windows®, macOS® or Linux®. MPLAB® XC8 supports all 8-bit PIC® and AVR® microcontrollers (MCUs).
- **MPLAB® Xpress Cloud-based IDE** - MPLAB Xpress Cloud-Based IDE is an online development environment that contains the most popular features of our award-winning MPLAB X IDE. This simplified and distilled application is a faithful reproduction of our desktop-based program, which allows users to easily transition between the two environments.
- **MPLAB® Code Configurator** - MPLAB Code Configurator (MCC) is a free software plug-in that provides a graphical interface to configure peripherals and functions specific to your application.
- **Atmel START** - Atmel START is an online tool that hosts code examples, helps the user to select and configure software components, and tailor your embedded application in a usable and optimized manner.
- **Microchip Sample Store** - Microchip sample store where you can order samples of devices.
- **MPLAB Data Visualizer** - MPLAB Data Visualizer is a program used for processing and visualizing data. The Data Visualizer can receive data from various sources such as serial ports and on-board debugger’s Data Gateway Interface, as found on Curiosity Nano and Xplained Pro boards.
- **Studio Data Visualizer** - Studio Data Visualizer is a program used for processing and visualizing data. The Data Visualizer can receive data from various sources such as serial ports, on-board debugger’s Data Gateway Interface as found on Curiosity Nano and Xplained Pro boards, and power data from the Power Debugger.
- **Microchip PIC® and AVR® Examples** - Microchip PIC and AVR Device Examples is a collection of examples and labs that use Microchip development boards to showcase the use of PIC and AVR device peripherals.
- **Microchip PIC® and AVR® Solutions** - Microchip PIC and AVR Device Solutions contains complete applications for use with Microchip development boards, ready to be adapted and extended.
- **AVR-BLE website** - Board information, latest user guide and design documentation.
- **AVR-BLE on Microchip Direct** - Purchase this board on Microchip Direct.

3. Hardware User Guide

3.1 On-Board Debugger Overview

AVR-BLE contains an on-board debugger for programming and debugging. The on-board debugger is a composite USB device consisting of several interfaces:

- A debugger that can program and debug the ATmega3208 in both MPLAB® X IDE and Atmel Studio 7
- A mass storage device that allows drag-and-drop programming of the ATmega3208
- A virtual serial port (CDC) that is connected to a Universal Asynchronous Receiver/Transmitter (UART) on the ATmega3208, and provides an easy way to communicate with the target application through terminal software
- A Data Gateway Interface (DGI) for code instrumentation with logic analyzer channels (debug GPIO) to visualize program flow

The on-board debugger controls a Power and Status LED (marked PS) on the AVR-BLE board. The table below shows how the LED is controlled in different operation modes.

Table 3-1. On-Board Debugger LED Control

Operation Mode	Power and Status LED
Boot Loader mode	The LED blinks slowly during power-up
Power-up	The LED is ON
Normal operation	The LED is ON
Programming	Activity indicator: The LED blinks slowly during programming/debugging
Drag-and-drop programming	**Success:** The LED blinks slowly for 2 sec. **Failure:** The LED blinks rapidly for 2 sec.
Fault	The LED blinks rapidly if a power fault is detected
Sleep/Off	The LED is OFF. The on-board debugger is either in a sleep mode or powered down. This can occur if the board is externally powered.

Info: Slow blinking is approximately 1 Hz, and rapid blinking is approximately 5 Hz.

3.1.1 Debugger

The on-board debugger on the AVR-BLE board appears as a Human Interface Device (HID) on the host computer's USB subsystem. The debugger supports full-featured programming and debugging of the ATmega3208 using both MPLAB® X IDE and Atmel Studio 7, as well as some third-party IDEs.

Remember: Keep the debugger's firmware up-to-date. Firmware upgrades automatically when using MPLAB® X IDE or Atmel Studio 7.

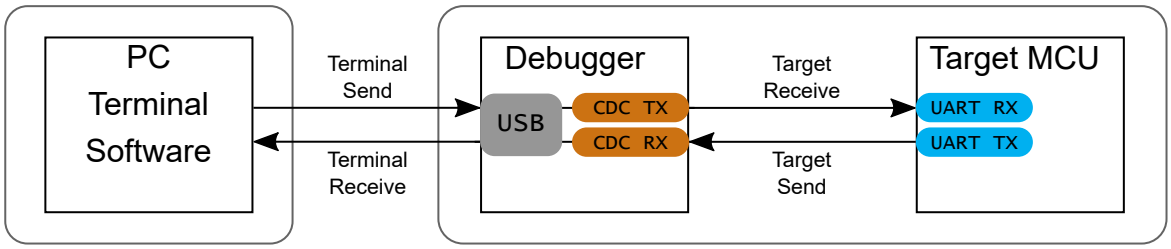
3.1.2 Virtual Serial Port (CDC)

The virtual serial port (CDC) is a general purpose serial bridge between a host PC and a target device.

3.1.2.1 Overview

The on-board debugger implements a composite USB device that includes a standard Communications Device Class (CDC) interface, which appears on the host as a virtual serial port. The CDC can be used to stream arbitrary data in both directions between the host computer and the target: All characters sent through the virtual serial port on the host computer will be transmitted as UART on the debugger's CDC TX pin, and UART characters captured on the debugger's CDC RX pin will be returned to the host computer through the virtual serial port.

Figure 3-1. CDC Connection



Info: As shown in Figure 3-1, the debugger's CDC TX pin is connected to a UART RX pin on the target for receiving characters from the host computer. Similarly, the debugger's CDC RX pin is connected to a UART TX pin on the target for transmitting characters to the host computer.

3.1.2.2 Operating System Support

On Windows machines, the CDC will enumerate as *Curiosity Virtual COM Port* and appear in the Ports section of the Windows Device Manager. The COM port number can also be found there.

Info: On older Windows systems, the CDC requires a USB. This driver is included in installations of both MPLAB® X IDE and Atmel Studio 7.

On Linux machines, the CDC will enumerate and appear as `/dev/ttyACM#`.

Info: tty* devices belong to the "dialout" group in Linux, so it may be necessary to become a member of that group to have permissions to access the CDC.

On MAC machines, the CDC will enumerate and appear as `/dev/tty.usbmodem#`. Depending on which terminal program is used, it will appear in the available list of modems as `usbmodem#`.

Info: For all operating systems: Be sure to use a terminal emulator that supports DTR signaling. See 3.1.2.4 Signaling.

3.1.2.3 Limitations

Not all UART features are implemented in the on-board debugger CDC. The constraints are outlined here:

- **Baud rate**: Must be in the range of 1200 bps to 500 kbps. Any baud rate outside this range will be set to the closest limit, without warning. Baud rate can be changed on-the-fly.
- **Character format**: Only 8-bit characters are supported.

- **Parity**: Can be odd, even, or none.
- **Hardware flow control**: Not supported.
- **Stop bits**: One or two bits are supported.

3.1.2.4 Signaling

During USB enumeration, the host OS will start both communication and data pipes of the CDC interface. At this point, it is possible to set and read back the baud rate and other UART parameters of the CDC, but data sending and receiving will not be enabled.

When a terminal connects on the host, it must assert the DTR signal. As this is a virtual control signal implemented on the USB interface, it is not physically present on the board. Asserting the DTR signal from the host will indicate to the on-board debugger that a CDC session is active. The debugger will then enable its level shifters (if available) and start the CDC data send and receive mechanisms.

Deasserting DTR in debugger firmware version 1.20 or earlier has the following behavior:

- Debugger UART receiver is disabled, so no further data will be transferred to the host computer
- Debugger UART transmitter will continue to send data that is queued for sending, but no new data is accepted from the host computer
- Level shifters (if available) are not disabled, so the debugger CDC TX line remains driven

Deasserting DTR in debugger firmware version 1.21 or later has the following behavior:

- Debugger UART receiver is disabled, so no further data will be transferred to the host computer
- Debugger UART transmitter will continue to send data that is queued for sending, but no new data is accepted from the host computer
- Once the ongoing transmission is complete, level shifters (if available) are disabled, so the debugger CDC TX line will become high-impedance

Remember: Set up the terminal emulator to assert the DTR signal. Without the signal, the on-board debugger will not send or receive any data through its UART.

Tip: The on-board debugger's CDC TX pin will not be driven until the CDC interface is enabled by the host computer. Also, there are no external pull-up resistors on the CDC lines connecting the debugger and the target, which means that during power-up, these lines are floating. To avoid any glitches resulting in unpredictable behavior like framing errors, the target device should enable the internal pull-up resistor on the pin connected to the debugger's CDC TX pin.

3.1.2.5 Advanced Use

CDC Override Mode

In normal operation, the on-board debugger is a true UART bridge between the host and the device. However, in certain use cases, the on-board debugger can override the basic operating mode and use the CDC TX and RX pins for other purposes.

Dropping a text file into the on-board debugger's mass storage drive can be used to send characters out of the debugger's CDC TX pin. The filename and extension are trivial, but the text file must start with the characters:

```
CMD:SEND_UART=
```

Debugger firmware version 1.20 or earlier has the following limitations:

- The maximum message length is 50 characters – all remaining data in the frame are ignored
- The default baud rate used in this mode is 9600 bps, but if the CDC is already active or has been configured, the previously used baud rate still applies

Debugger firmware version 1.21 and later has the following limitations/features:

- The maximum message length may vary depending on the MSC/SCSI layer timeouts on the host computer and/or operating system. A single SCSI frame of 512 bytes (498 characters of payload) is ensured, and files of up to 4 KB will work on most systems. The transfer will complete on the first NULL character encountered in the file.
- The baud rate used is always 9600 bps for the default command:

```
CMD:SEND_UART=
```

- The CDC override mode should not be used simultaneously with data transfer over the CDC/terminal. If a CDC terminal session is active at the time a file is received via CDC override mode, it will be suspended for the duration of the operation and resumed once complete.
- Additional commands are supported with explicit baud rates:

```
CMD:SEND_9600=
```

```
CMD:SEND_115200=
```

```
CMD:SEND_460800=
```

USB-Level Framing Considerations

Sending data from the host to the CDC can be done byte-wise or in blocks, which will be chunked into 64-byte USB frames. Each such frame will be queued up for sending to the debugger's CDC TX pin. Transferring a small amount of data per frame can be inefficient, particularly at low baud rates, as the on-board debugger buffers frames and not bytes. A maximum of four 64-byte frames can be active at any time. The on-board debugger will throttle the incoming frames accordingly. Sending full 64-byte frames containing data is the most efficient method.

When receiving data on the debugger's CDC RX pin, the on-board debugger will queue up the incoming bytes into 64-byte frames, which are sent to the USB queue for transmission to the host when they are full. Incomplete frames are also pushed to the USB queue at approximately 100 ms intervals, triggered by USB start-of-frame tokens. Up to eight 64-byte frames can be active at any time.

If the host (or the software running on it) fails to receive data fast enough, an overrun will occur. When this happens, the last-filled buffer frame will be recycled instead of being sent to the USB queue, and a full data frame will be lost. To prevent this occurrence, the user must ensure that the CDC data pipe is being read continuously, or the incoming data rate must be reduced.

3.1.3 Mass Storage Device

The on-board debugger includes a simple Mass Storage Device implementation, which is accessible for read/write operations via the host operating system to which it is connected.

It provides:

- Read access to basic text and HTML files for detailed kit information and support
- Write access for programming Intel® HEX formatted files into the target device's memory
- Write access for simple text files for utility purposes

3.1.3.1 Mass Storage Device Implementation

The on-board debugger implements a highly optimized variant of the FAT12 file system that has several limitations, partly due to the nature of FAT12 itself and optimizations made to fulfill its purpose for its embedded application.

The Curiosity Nano USB device is USB Chapter 9-compliant as a mass storage device but does not, in any way, fulfill the expectations of a general purpose mass storage device. This behavior is intentional.

When using the Windows operating system, the on-board debugger enumerates as a Curiosity Nano USB Device that can be found in the disk drives section of the device manager. The CURIOSITY drive appears in the file manager and claims the next available drive letter in the system.

The CURIOSITY drive contains approximately one MB of free space, and this does not reflect the size of the target device's Flash in any way. When programming an Intel® HEX file, the binary data are encoded in ASCII with metadata providing a large overhead, so one MB is a trivially chosen value for disk size.

It is not possible to format the CURIOSITY drive. When programming a file to the target, the filename may appear in the disk directory listing. This is merely the operating system's view of the directory, which in reality, has not been updated. It is not possible to read out the file contents. Removing and replugging the board will return the file system to its original state, but the target will still contain the application that has been previously programmed.

To erase the target device, copy a text file starting with "`CMD:ERASE`" onto the disk.

By default, the CURIOSITY drive contains several read-only files for generating icons as well as reporting status and linking to further information:

- `AUTORUN.ICO` – icon file for the Microchip logo
- `AUTORUN.INF` – system file required for Windows Explorer to show the icon file
- `CLICK-ME.HTM` – redirect to the AVR-BLE web demo application
- `KIT-INFO.HTM` – redirect to the development board website
- `KIT-INFO.TXT` – a text file containing details about the board's debugger firmware version, board name, USB serial number, device, and drag-and-drop support
- `STATUS.TXT` – a text file containing the programming status of the board

Info: `STATUS.TXT` is dynamically updated by the on-board debugger. The contents may be cached by the OS and, therefore, do not reflect the correct status.

3.1.3.2 Limitations of Drag-and-Drop Programming

Lock Bits
Lock bits included in the hex file will be ignored when using drag-and-drop programming. To program lock bits, use MPLAB® X IDE or Atmel Studio 7.

Enabling CRC Check in Fuses
It is not advisable to enable the CRC check in the target device's fuses when using drag-and-drop programming. This because a subsequent chip erase (which does not affect fuse bits) will effect a CRC mismatch, and the application will fail to boot. To recover a target from this state, a chip erase must be done using MPLAB® X IDE or Atmel Studio 7, which will automatically clear the CRC fuses after erasing.

3.1.3.3 Special Commands

Several utility commands are supported by copying text files to the mass storage disk. The filename or extension is irrelevant – the command handler reacts to content only.

Table 3-2. Special File Commands

Command Content	Description
`CMD:ERASE`	Executes a chip erase of the target
`CMD:SEND_UART=`	Sends a string of characters to the CDC UART. See "CDC Override Mode."

`CMD:SEND_9600=`
`CMD:SEND_115200=`
`CMD:SEND_460800=`
	Sends a string of characters to the CDC UART at the baud rate specified. Note that only the baud rates explicitly specified here are supported! See "CDC Override Mode" (Debugger firmware v1.21 or newer.)

`CMD:RESET`	Resets the target device by entering Programming mode and then exiting Programming mode immediately afterward. Exact timing can vary according to the programming interface of the target device. (Debugger firmware v1.16 or newer.)

Info: The commands listed here are triggered by the content sent to the mass storage emulated disk, and no feedback is provided in the case of either success or failure.

3.1.4 Data Gateway Interface (DGI)

Data Gateway Interface (DGI) is a USB interface for transporting raw and timestamped data between on-board debuggers and host computer-based visualization tools. MPLAB Data Visualizer is used on the host computer to display debug GPIO data. It is available as a plug-in for MPLAB® X IDE or a stand-alone application that can be used in parallel with MPLAB® X IDE or Atmel Studio 7.

Although DGI encompasses several physical data interfaces, the AVR-BLE implementation includes logic analyzer channels:

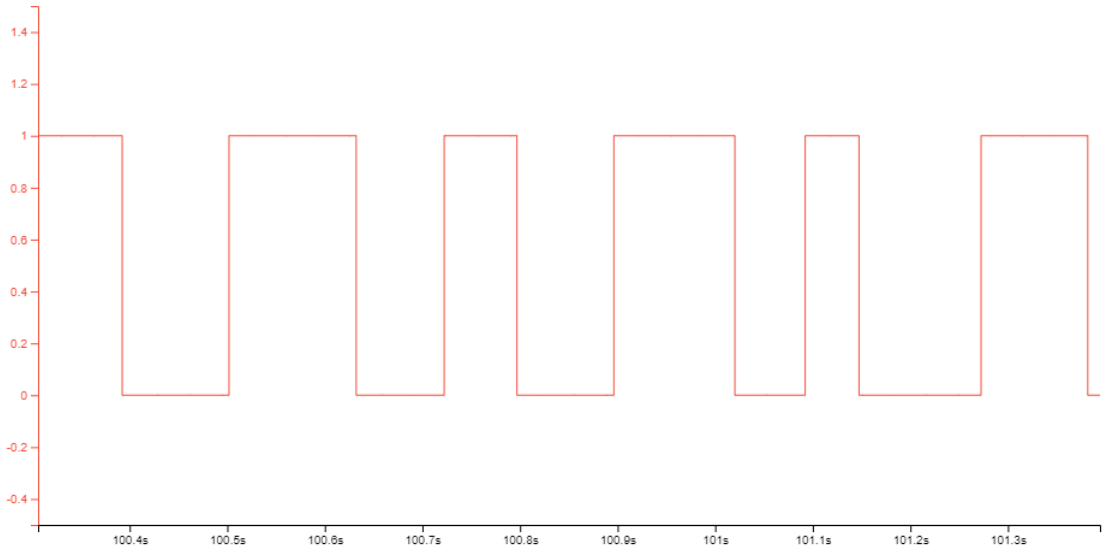
- Two debug GPIO channels (also known as DGI GPIO)

3.1.4.1 Debug GPIO

Debug GPIO channels are timestamped digital signal lines connecting the target application to a host computer visualization application. They are typically used to plot the occurrence of low-frequency events on a time-axis – for example, when certain application state transitions occur.

The figure below shows the monitoring of the digital state of a mechanical switch connected to a debug GPIO in MPLAB Data Visualizer.

Figure 3-2. Monitoring Debug GPIO with MPLAB® Data Visualizer



Debug GPIO channels are timestamped, so the resolution of DGI GPIO events is determined by the resolution of the DGI timestamp module.

Important: Although bursts of higher-frequency signals can be captured, the useful frequency range of signals for which debug GPIO can be used is up to about 2 kHz. Attempting to capture signals above this frequency will result in data saturation and overflow, which may cause the DGI session to be aborted.

3.1.4.2 Timestamping

DGI sources are timestamped as they are captured by the debugger. The timestamp counter implemented in the Curiosity Nano debugger increments at 2 MHz frequency, providing a timestamp resolution of a half microsecond.

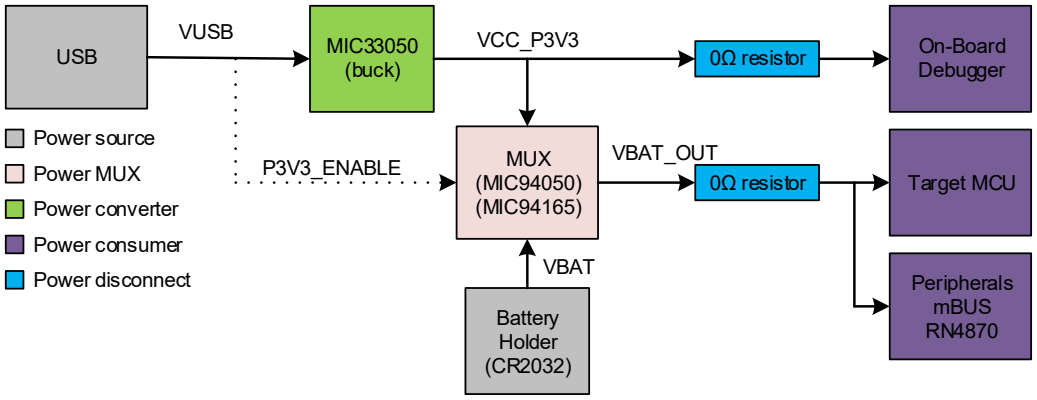
3.2 Power Supply

The board can be powered through the USB port or by a CR2032 battery. It will automatically switch to the battery if USB power is not available. While powered through USB, the board generates 3.3V for the debugger, ATmega3208, and peripherals. During battery operation, the ATmega3208 and peripherals run directly on the battery voltage, while the debugger is not powered.

Current drawn from the USB port is limited to 500 mA by a PTC resettable fuse.

Important: When powering the AVR-BLE board with a CR2032 battery, it is important to leave the ATmega3208 pins that connect to the CDC UART in Tri-State (Input) mode. This is to prevent the debugger from getting powered through its GPIO.

Figure 3-3. Power Supply Block Diagram



Info: On the mikroBUS socket, the +5V rail is powered from the USB port. Consequently, +5V will not be available when the board is powered from a battery.

3.3 Low-Power Operation

To achieve the lowest power consumption of the board, the following considerations must be taken:

- Set the MCP9844 in Shutdown mode
  - Set bit 8 (SHDN) in the 16-bit CONFIG register (address 0x01)
- Set the BMA253 in Deep Suspend mode
  - Set bit 5 (deep suspend) in the 8-bit PMU_LPW register (address 0x11)
- Set the RN4870 in Sleep mode
  - Set the RX_IND pin high (PD2 on the ATmega3208)
  - Send the "`O,0\r`" command to the RN4870
- Set unused ATmega3208 I/O pins as input and disable the digital input buffer

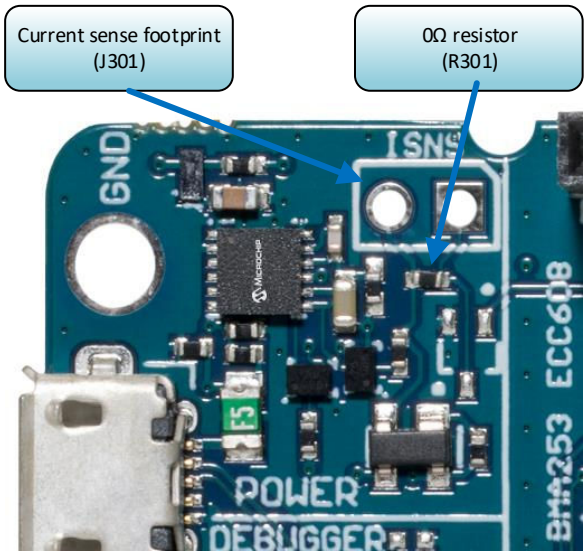
Important: USART pins PF0 and PF1 are connected directly to the on-board debugger. It is important to tri-state the USART pins when the board is powered from a CR2032 battery to prevent powering the debugger through its I/O pins. Doing so will increase the power consumption and cause undefined behavior from the on-board debugger.

Info: The load switch U300 in the power MUX can leak up to 1 µA when the board is powered from a battery. By modifying the board and removing resistor R303 (0Ω), U300 can be disconnected. Be warned that a board modified this way can no longer be powered from USB, and consequently, neither programmed nor debugged using the on-board debugger until the 0Ω resistor is reconnected.

3.4 Target Current Measurement

Power to the ATmega3208 and its peripherals is connected from the on-board power supply through a 0Ω resistor (R301) in parallel with a 100-mil *Current sense* pin header footprint marked with “ISNS” in silkscreen (J301). To measure the power consumption of the ATmega3208 and other peripherals on the board, de-solder the 0Ω resistor and connect an ammeter over the *Current sense* footprint.

Figure 3-4. Current sense footprint



Tip: A 100-mil pin header can be soldered into the *Current sense* (J301) footprint for easy connection of an ammeter. Once the ammeter is not needed anymore, place a jumper cap on the pin header.

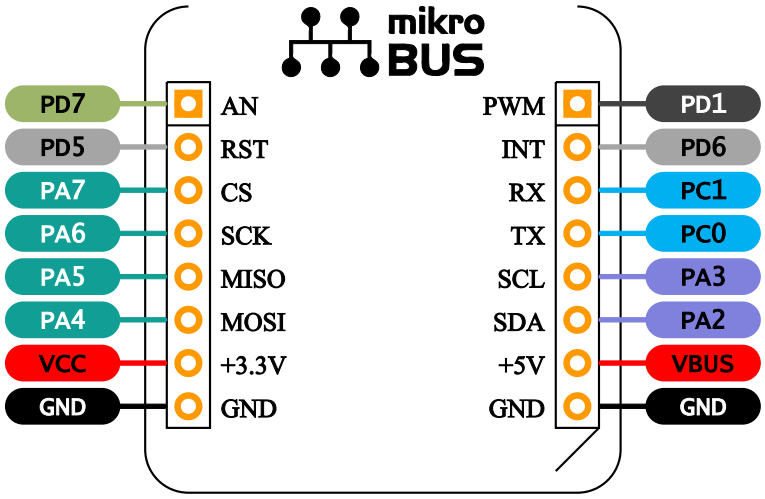
3.5 Peripherals

3.5.1 ATmega3208

Microchip ATmega3208 is a microcontroller featuring the AVR® processor with hardware multiplier – running at up to 20 MHz and with 32 KB Flash, 4 KB SRAM, and 256 bytes of electrically erasable programmable read-only memory (EEPROM) in a 28- or 32-pin package. It uses the latest Core Independent Peripherals (CIPs) with low-power features, including Event System, accurate analog features, and advanced peripherals.

3.5.2 mikroBUS Socket

Figure 3-5. mikroBUS Socket Pinout



The AVR-BLE board features a mikroBUS socket for expanding the functionality of the development board using MikroElektronika Click Boards and other mikroBUS add-on boards. The socket is populated with two 1x8 2.54 mm pitch female headers and is ready to mount add-on boards.

Table 3-3. mikroBUS Socket Pinout

mikroBUS Socket Pin	ATmega3208 Pin	Function	Shared Functionality
AN	PD7	ADC AIN7	—
RST	PD5	GPIO	—
CS	PA7	SPI0 CS	—
SCK	PA6	SPI0 SCK	SST25PF040CT
MISO	PA5	SPI0 MISO	SST25PF040CT
MOSI	PA4	SPI0 MOSI	SST25PF040CT
+3.3V	V_{DD}	VCC_TARGET	—
GND	GND	Ground	—
PWM	PD1	TCA0 WO1	—
INT	PD6	GPIO	—
RX	PC1	UART1 RX	—
TX	PC0	UART1 TX	—
SCL	PA3	TWI0 SCL	MCP9844, BMA253 and ATECC608A
SDA	PA2	TWI0 SDA	MCP9844, BMA253 and ATECC608A
+5V	—	VBUS	—
GND	GND	Ground	—

Info: VBUS is powered from USB. Consequently, +5V will not be available while the board is powered from a battery.

Info: VCC_TARGET will have the battery voltage when the board is powered from a battery, which can be less that +3.3V.

3.5.3 RN4870 BLE Module

The RN4870 is a Bluetooth® Low Energy (BLE) module that integrates a Bluetooth® 5.0 baseband controller, on-board Bluetooth stack, digital and analog I/O, and RF power amplifier into one solution.

Additional Features:

- Range up to 50m
- Operating Voltage Range: 1.9V to 3.6V
- TX / RX Mode Peak Current: 10 mA (typical)
- Low-Power Mode Current: 60 µA (typical)
- Shutdown Current: 2.9 µA (max)

The RN4870 BLE module is connected to the ATmega3208 through UART as well as three GPIOs for control and configuring of the module. The RST signals resets the module, while the RX_IND signal is used to wake the module from Low-Power mode. The MODE signal, available from the ATmega3208, the debugger as well as by a physical switch, allows the module to be put in a “Test Mode” where the RN4870 firmware can be updated.

The module has one of its GPIO pins connected to an LED. By default, this will indicate connection status, but the user can configure it for a number of other functions. Many of the other RN4870 GPIO pins are available as pads around the label on the back side of the AVR-BLE board, as can be seen in Figure 1-2.

Info: Some RN4870 settings have been changed during manufacturing of AVR-BLE. Using the `S-` command, the device name has been changed to “AVR-BLE”. In addition, the communication settings have been configured for ATmega3208 UART0 settings of 9600,8,N,1.

Table 3-4. RN4870 Connections

RN4870 Pin	ATmega3208 Pin	Function	Shared Functionality
RX	PA0	UART0 TX	—
TX	PA1	UART0 RX	—
RST	PD3	GPIO	—
P2_0 / MODE	PF3	GPIO	SW0 and on-board debugger
P3_3 / RX_IND	PD3	GPIO	—
P1_1 / STATUS1	—	BLE Connection LED	BLE LED

Info: The RST and MODE signals are pulled up by external resistors.

3.5.4 ATECC608A Secure Element

The ATECC608A is a secure element from the Microchip CryptoAuthentication portfolio with advanced Elliptic Curve Cryptography (ECC) capabilities. With ECDH and ECDSA being built right in, this device is ideal for the rapidly growing Internet of Things (IoT) market by easily supplying the full range of security, such as confidentiality, data integrity, and authentication to systems with MCU or MPUs running encryption/decryption algorithms. Similar to all Microchip CryptoAuthentication products, the ATECC608A employs ultra-secure, hardware-based cryptographic key storage and cryptographic countermeasures that eliminate any potential backdoors linked to software weaknesses.

The ATECC608A CryptoAuthentication device on the AVR-BLE board can be used to authenticate the board with other hardware for secure IoT communication.

Info: 7-bit I^2C address: 0x58.

Table 3-5. ATECC608A Connections

ATECC608A Pin	ATmega3208 Pin	Function	Shared Functionality
SDA	PA2	TWI0 SDA	MCP9844, BMA253 and mikroBUS
SCL	PA3	TWI0 SCL	MCP9844, BMA253 and mikroBUS

3.5.5 SST25PF040CT Serial Flash

The SST25PF040CT is a 4 Mbit Serial Flash with extended operating voltage range and low-power consumption.

Additional Features:

- Operating Voltage Range: 2.3V to 3.6V
- Active Read Current: 5 mA (typical)
- Power-Down Standby Current: 3 µA (typical)

The SST25PF040CT Serial Flash is connected to the ATmega3208 through SPI and a GPIO for the HOLD signal.

Info: The Flash is SPI Mode 0 and Mode 3 compatible, and supports clock speeds up to 40 MHz.

Table 3-6. SST25PF040CT Connections

SST25PF040CT Pin	ATmega3208 Pin	Function	Shared Functionality
CS	PD0	GPIO	—
SCK	PA6	SPI0 SCK	mikroBUS
MISO	PA5	SPI0 MISO	mikroBUS
MOSI	PA4	SPI0 MOSI	mikroBUS
HOLD#	PD4	GPIO	—

3.5.6 MCP9844 Temperature Sensor

The MCP9844 digital temperature sensor converts circuit board temperatures between -40°C and +125°C to a digital word with ±1°C/±3°C (typical/maximum) accuracy.

Additional features:

- Accuracy:
  - ±0.2°C/±1°C (typical/maximum) from +75°C to +95°C
  - ±0.5°C/±2°C (typical/maximum) from +40°C to +125°C
  - ±1°C/±3°C (typical/maximum) from -40°C to +125°C
- User Selectable Measurement Resolution:
  - 0.5°C, 0.25°C, 0.125°C, 0.0625°C
- User Programmable Temperature Limits:
  - Temperature Window Limit
  - Critical Temperature Limit
- User Programmable Temperature Alert Output
- Operating Voltage Range:
  - 1.7V to 3.6V
- Operating Current:
  - 100 µA (typical)
- Shutdown Current:
  - 0.2 µA (typical)

The MCP9844 temperature sensor is connected to the ATmega3208 through I^2C and a GPIO for the user-configurable event output.

Info: 7-bit I^2C address: 0x18.

Table 3-7. MCP9844 Connections

MCP9844 Pin	ATmega3208 Pin	Function	Shared Functionality
SDA	PA2	TWI0 SDA	ATECC608A, BMA253 and mikroBUS
SCL	PA3	TWI0 SCL	ATECC608A, BMA253 and mikroBUS
Event	PF2	ASYNC External Interrupt	—

3.5.7 BMA253 Acceleration Sensor

The Bosch BMA253 is a low-g acceleration sensor with digital output for measurements of acceleration in three perpendicular axes.

Additional Features:

- 12-Bit Sensitivity
- User Selectable Acceleration Ranges: ±2g, ±4g, ±8g, ±16g
- On-Chip 32 Frame First-In First-Out (FIFO)
- Motion Triggered Interrupts:
  - New Data
  - Any Motion Detection
  - Single/Double Tap Sensing
  - Orientation Recognition
  - Flat Detection
  - Low/High-g Detection
  - Inactivity Detection
- Operating Voltage Range: 1.62V to 3.6V

- Operating Current (Normal mode): 130 µA (typical)
- Shutdown Current (Deep Suspend mode): 1 µA (typical)

The BMA253 acceleration sensor is connected to the ATmega3208 through I^2C and two GPIOs for the user configurable interrupt outputs.

Info: 7-bit I^2C address: 0x19

Table 3-8. BMA253 Connections

BMA253 Pin	ATmega3208 Pin	Function	Shared Functionality
SDA	PA2	TWI0 SDA	MCP9844, ATECC608A and mikroBUS
SCL	PA3	TWI0 SCL	MCP9844, ATECC608A and mikroBUS
INT1	PC2	ASYNC External Interrupt	—
INT2	PC3	External Interrupt	—

3.5.8 LEDs

There are two user LEDs available on the AVR-BLE board that can be controlled by either GPIO or PWM. In addition, there is one LED connected directly to the BLE module. The LEDs can be activated by driving their connected I/O lines to GND.

Table 3-9. LED Connections

LED	ATmega3208 Pin	Function	Shared Functionality
Green Data LED	PF4	TCA0 WO4	On-board debugger
Red Error LED	PF5	TCA0 WO5	—
Blue BLE LED	—	Connected to BLE module	RN4870

3.5.9 Mechanical Switch

The AVR-BLE board has one mechanical switch. This is a generic user-configurable switch that will drive the connected I/O line to ground (GND) when it is pressed. An external resistor pulls the signal high when the switch is not pressed.

Holding the switch during power-up can be used to put the Bluetooth module in Configuration mode. See 3.5.3 RN4870 BLE Module for more information.

Table 3-10. Mechanical Switch Connection

Switch	ATmega3208 Pin	Function	Shared Functionality
SW0	PF3	User switch	RN4870 and on-board debugger

Info: The SW0 signal is pulled up by an external resistor.

3.5.10 On-Board Debugger Implementation

AVR-BLE features an on-board debugger that can be used to program and debug the ATmega3208 using UPDI. The on-board debugger also includes a virtual serial port (CDC) interface over UART and debug GPIO. Both MPLAB® X IDE and Atmel Studio 7 can be used as a front-end for the on-board debugger for programming and debugging. MPLAB Data Visualizer can be used as a front-end for the CDC and debug GPIO.

3.5.10.1 On-Board Debugger Connections

The table below shows the connections between the target and the debugger section. All connections between the target and the debugger are tri-stated as long as the debugger is not actively using the interface. Hence, since there are little contaminations of the signals, the pins can be configured to anything the user wants.

For further information on how to use the capabilities of the on-board debugger, see 3.1 On-Board Debugger Overview.

Table 3-11. On-Board Debugger Connections

ATmega3208 Pin	Debugger Pin	Function	Shared Functionality
PF1	CDC TX	UART2 RX (ATmega3208 RX line)	—
PF0	CDC RX	UART2 TX (ATmega3208 TX line)	—
UPDI	DBG0	UPDI	—
PF4	DBG1	DEBUG GPIO1	Data LED
PF3	DBG2	DEBUG GPIO0	SW0 and RN4870

4. Hardware Revision History and Known Issues

This user guide is provides information about the latest available revision of the board. The following sections contain information about known issues, a revision history of older revisions, and how older revisions differ from the latest revision.

4.1 Identifying Product ID and Revision

There are two ways to find the revision and product identifier of the AVR-BLE: Either by utilizing the MPLAB® X IDE or Atmel Studio 7 Kit Window or by looking at the sticker on the bottom side of the PCB.

By connecting AVR-BLE to a computer with MPLAB® X IDE or Atmel Studio 7 running, the Kit Window will pop up. The first six digits of the serial number, listed under kit information, contain the product identifier and revision.

Tip: If closed the Kit Window can be opened in MPLAB® X IDE through the menu bar *Window > Kit Window*.

The same information is found on the sticker on the bottom side of the PCB. Most boards will have the identifier and revision printed in plain text as A09-nnnn\rr, where “nnnn” is the identifier, and “rr” is the revision. Boards with limited space have a sticker with only a data matrix code, containing the product identifier, revision, and serial number.

The serial number string has the following format:

"nnnnrrssssssssss"

n = product identifier

r = revision

s = serial number

The product identifier for AVR-BLE is A09-3314.

4.2 Revision 3

Revision 3 is functionally the same as revision 2, but features a RN4870 BLE module with firmware version 1.40 (part number RN4870-V/RM140).

In the silkscreen, the part number of the serial flash chip is wrong. It should be SST25PF040C, but shows SST25P040C.

4.3 Revision 2

Revision 2 is the initial released revision of the board. It features a RN4870 BLE module with firmware version 1.30 (part number RN4870-I/RM130).

5. Document Revision History

Revision	Date	Description
B	10/2020	Document updated with the latest information. Steps describing how to download and install the latest demonstration firmware added to the Quick Start section.
A	03/2020	Initial document release

6. Appendix

6.1 Schematics

Figure 6-1. AVR-BLE Target schematic

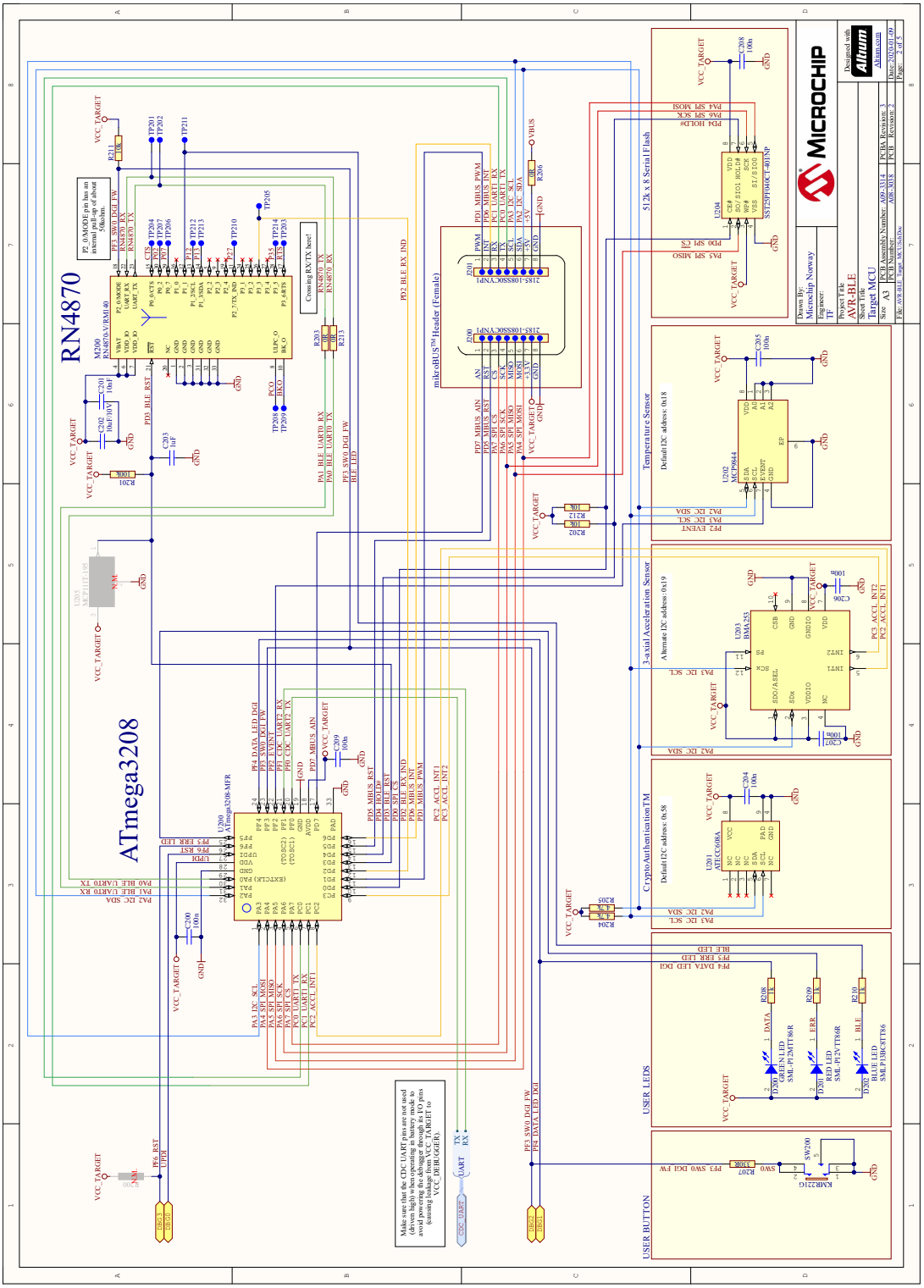


Figure 6-2. AVR-BLE Power Schematic

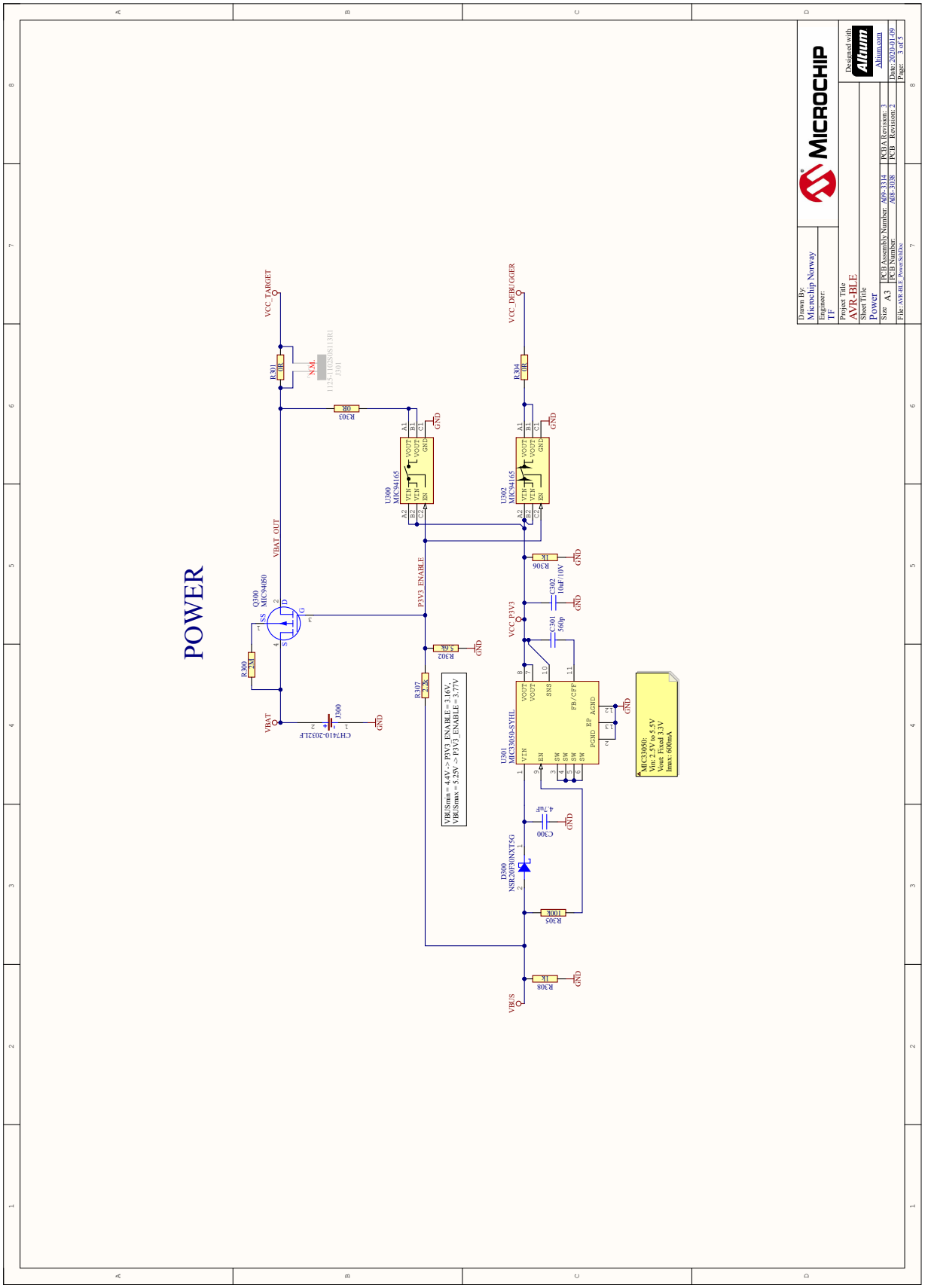
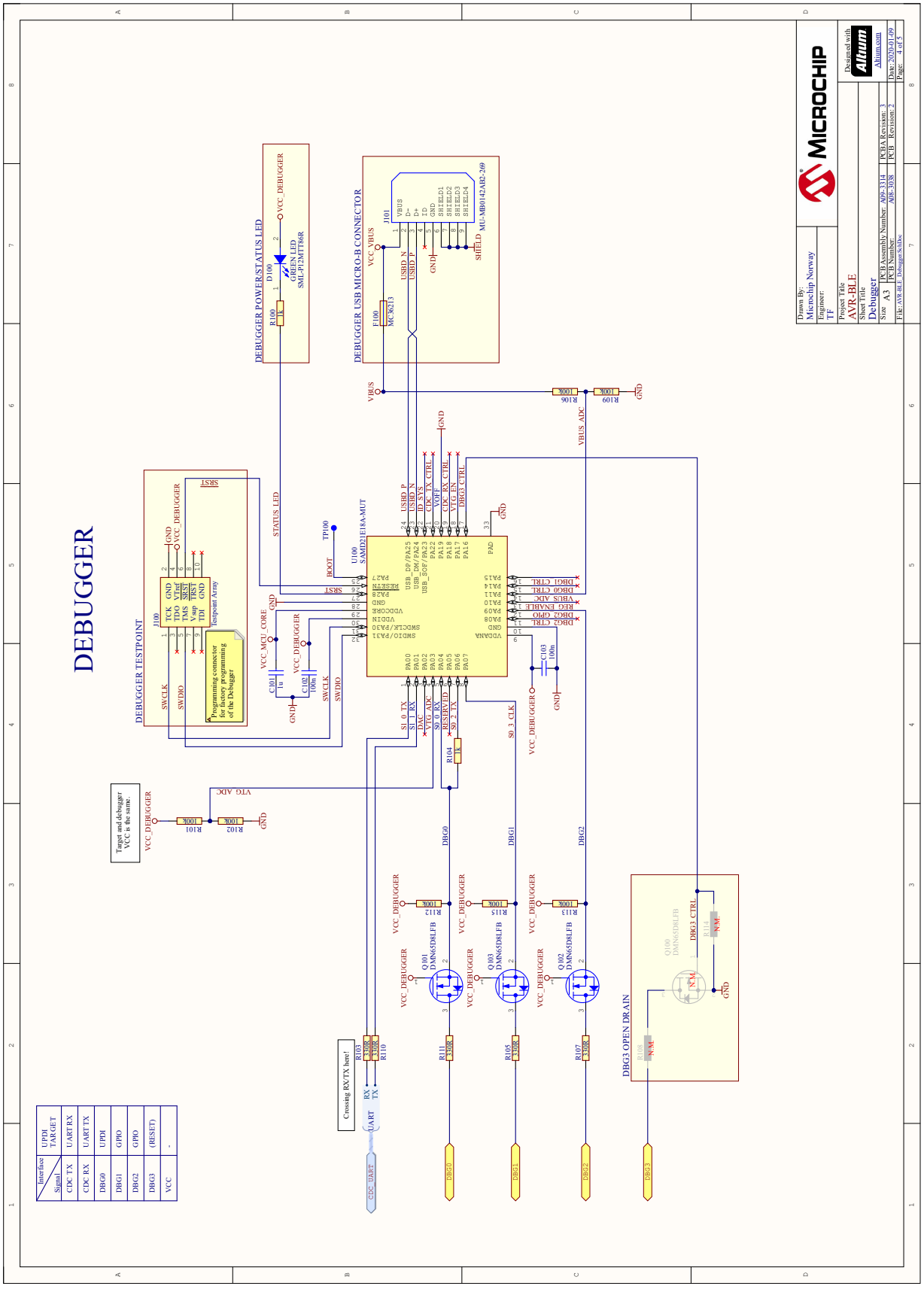


Figure 6-3. AVR-BLE Debugger Schematic



6.2 Assembly Drawing

Figure 6-4. AVR-BLE Assembly Drawing Top

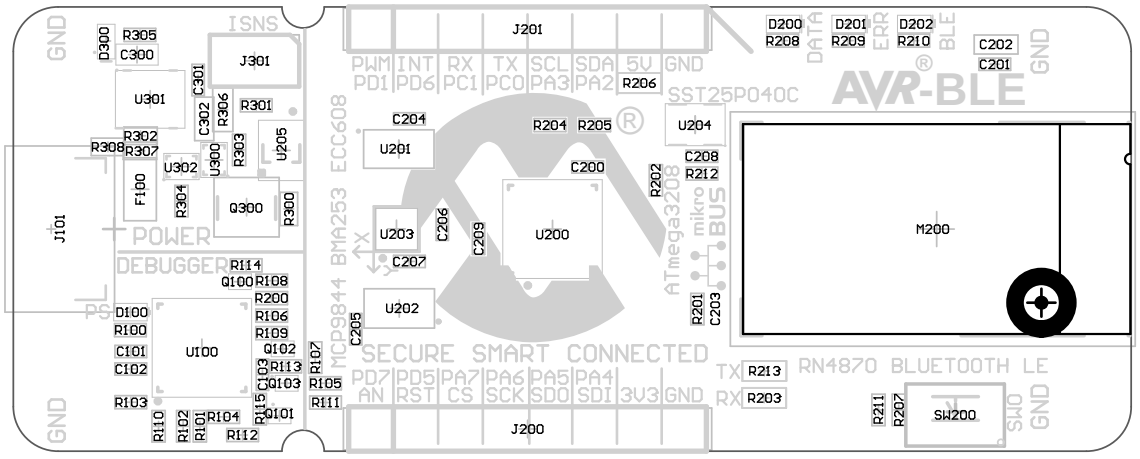
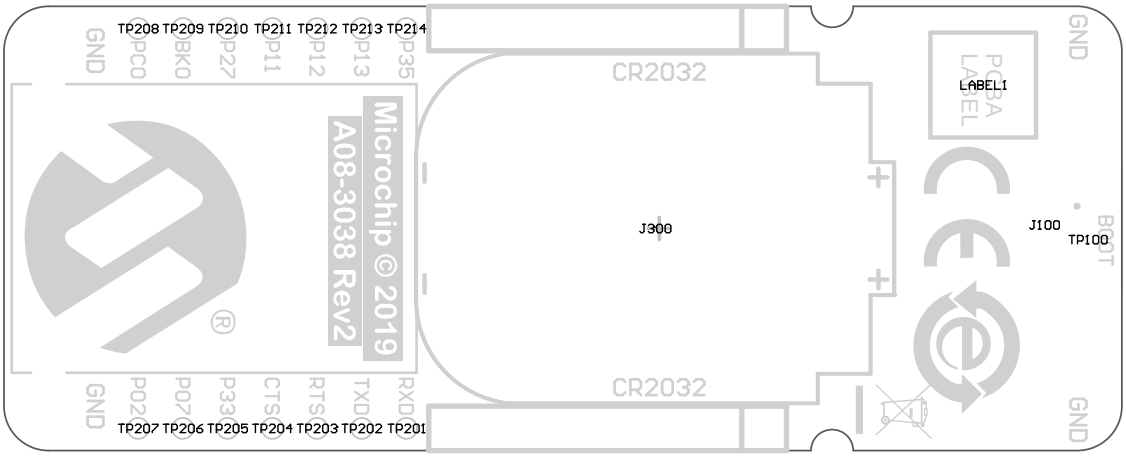


Figure 6-5. AVR-BLE Assembly Drawing Bottom



The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-7032-8

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820