Getting Started with Cygnus Reach

I3 Product Design
Version 0.91
Date December 28, 2023

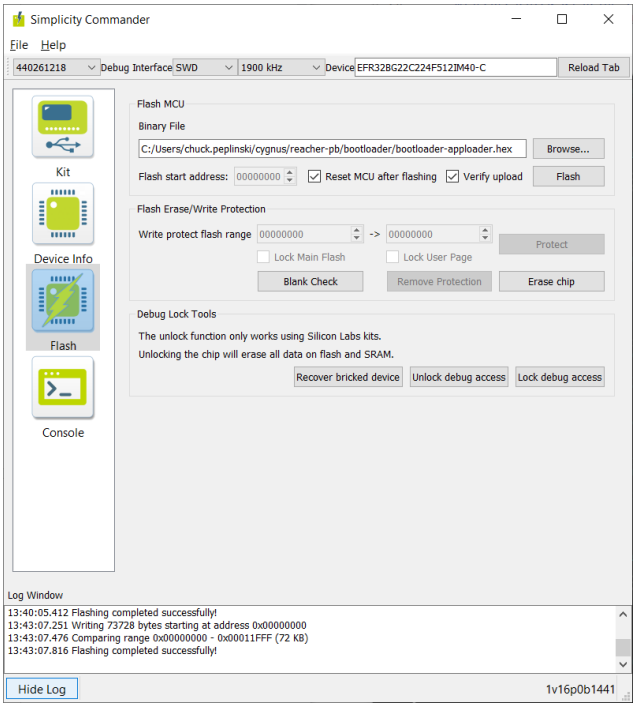
This Cygnus Reach Evaluation Kit runs on a Silicon Labs “Thunderboard”. This document briefly describes the first steps for new users.

Preparation

- Obtain the Cygnus Reach firmware package from this site on github:
    - https://github.com/cygnus-technology/reach-firmware
- Procure a Silicon Labs Thunderboard (SLTB010A). A quick google finds these common at Digikey and Mouser. The cost is around $42.
- If you are going to rebuild the Thunderboard program, install Simplicity Studio from Silicon Labs.
    https://www.silabs.com/developers/simplicity-studio
    - The code is updated to match version 5.8 with Gecko 4.4.
    - For any Linux users, be aware that the default installation can go a lot faster if you set the make to use multiple threads.
- If you are only going to flash the prebuilt binary, you can more quickly install only the Simplicity Commander tool. Google "download Simplicity Commander" and choose the version you like. They offer Windows, Mac and Linux.
- Install the Cygnus Reach application on your mobile device. You can find an appropriate version in the Google Play Store and in Apple's App Store.

Running the Demo

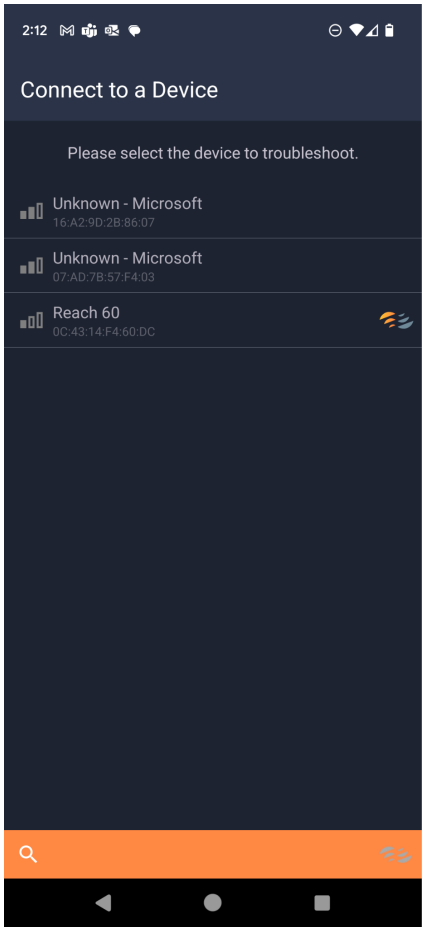
- Connect your PC to the USB port on the Thunderboard.
- Launch Simplicity Commander.



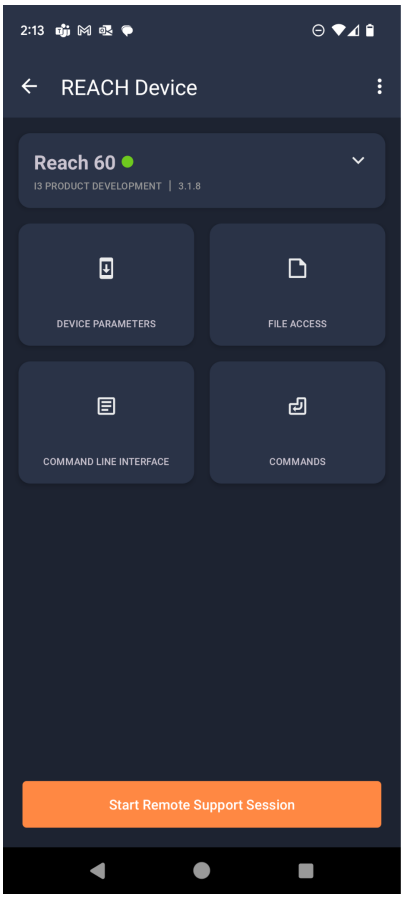
- Choose the kit in the upper left.
- Select “reacher-tboard-v320.hex” for flashing. This includes both the bootloader and the application. Separate bootloader and app hex files are also available.
- Poke the “Flash” button. The program will boot when the download is complete.
- Connect a serial terminal to the “JLink CDC UART Port” exposed by the Thunderboard. It runs at the typical 115200,N,8,1.
- When the board is running it should respond to a carriage return with a prompt and the “ver” command should print a banner. The help command prints help.
- You should start to see a console like shown here.

```
> ver
Cygnus Reach Protobuf Server, built Dec 28 2023, 11:53:24
Firmware v3.2.0
SiLabs Thunderboard hardware
Remote CLI support built but not enabled.
> help
  ver                       Gets information about the system version
  /                         Gets information about the system status
  lm                        Set log mask,
                            [string] log mask, hex number
  rcli                      Enable or disable remote CLI echo,
                            [string] 1 or 0
  phy                       Set PHY to 1 or 2 Mbps,
                            [uint32] 1 (default) or 2 (fast)
  nvm                       Manage non volatile memory
                            [string] nvm ?: (summarize), nvm clear (erase all)
  sn                        Read or set serial number in NVM
                            [string] sn ?: (display), sn clear (erase), sn N (write N)
>
```

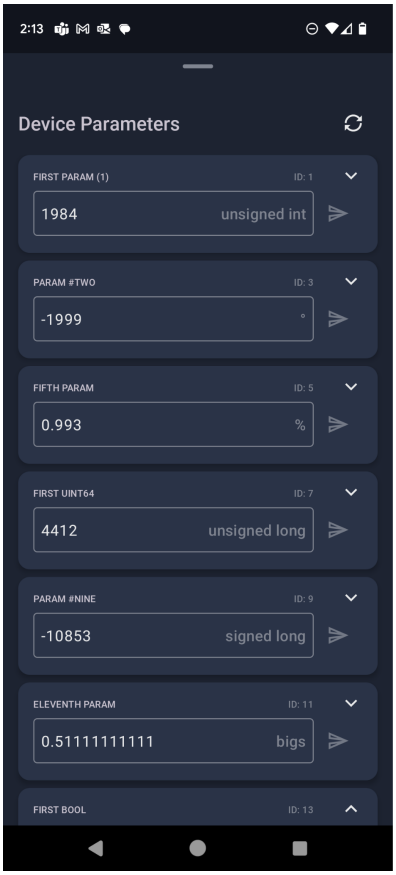
Connecting with the Cygnus App



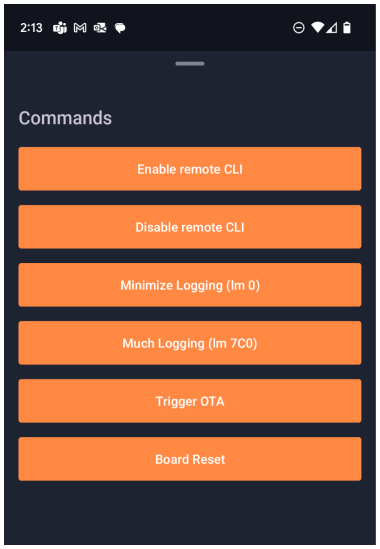
- Open the Cygnus app on your phone and select this device. You can see that it has the Cygnus logo. The app can also view the characteristics on other BLE devices.
- You can pull down the name tab to see the complete device info.



- Poke the Device Parameters button. This will bring up the basic parameter inspection frame. You can edit parameters here. Parameter ID 23 is increasing. You can refresh the screen and see that.

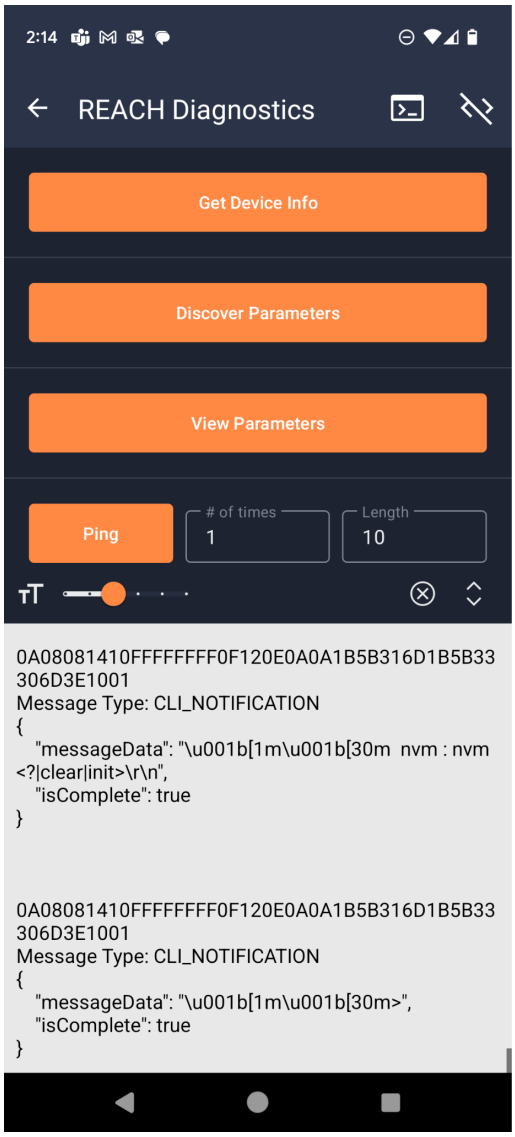


The command tab has several useful functions. You need to enable the remote CLI here to use it. You can also easily adjust the logging.

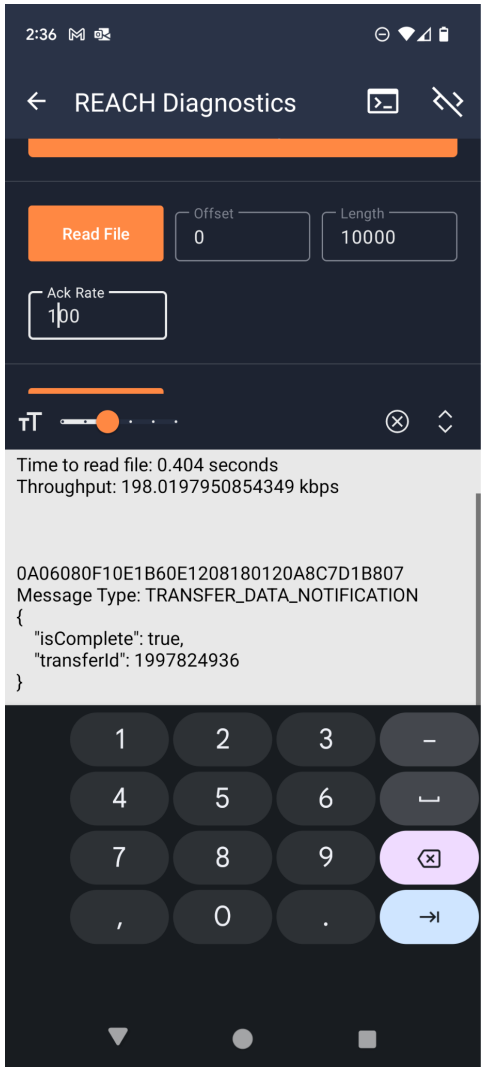


File Access and the Debug Interface

The file access features of Cygnus support the efficient transfer of larger blocks of data. The “simple” interface interacts with files on your phone and gives you the option of emailing received files. The debug interface gives you more control. Access the debug interface from the three dots in the upper right corner.



Here a button represents each of the basic message commands in the Reach protocol. The phone issues the request and the resulting action is printed on the screen. The response is shown in hex. The message is decoded into a JSON like format. The hex bytes that were sent are not displayed here. They are visible on the serial console of the device.

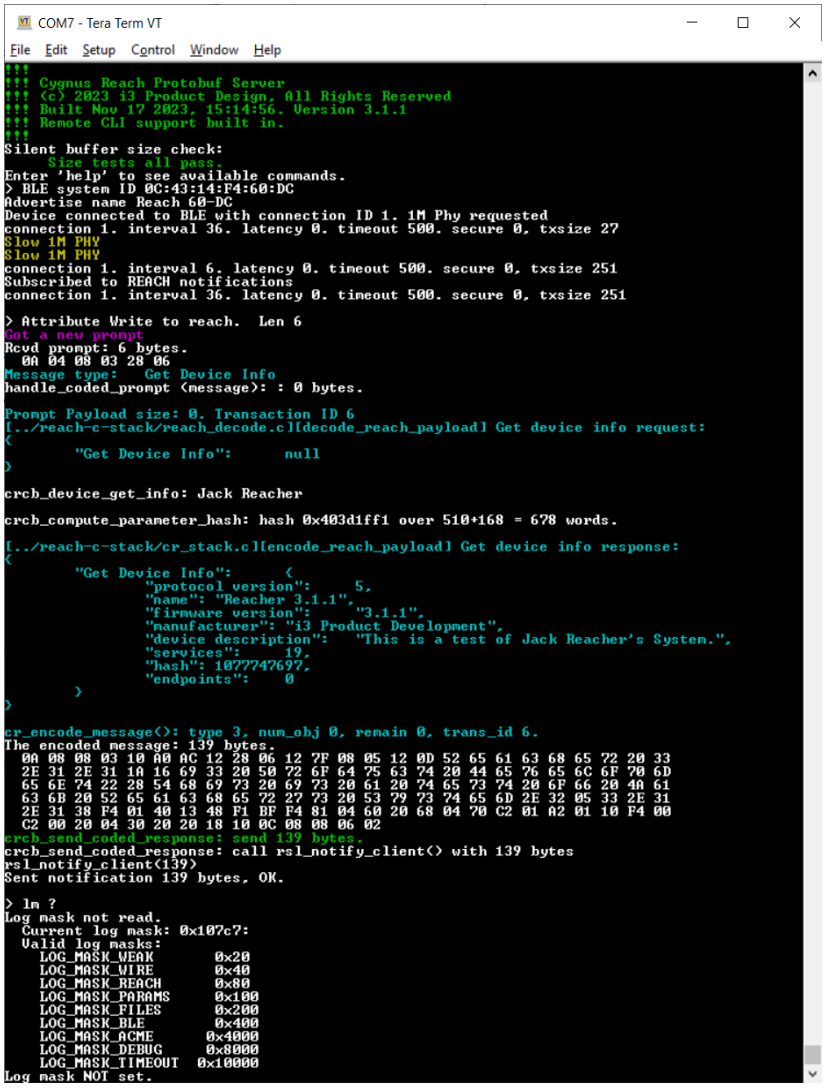


You can exercise finer control over the file operations on the debug screen. First minimize logging using command 3. Then select a fairly large size to read. These tests send synthetic data, just increasing numbers to exercise the transfer rate. You can experiment with the Acknowledgement rate and easily achieve transfer rates over 200kbps.

The Serial Console

By default you see messages on the serial console as illustrated below.

- A banner is printed at startup.
- The yellow text about the PHY indicates we are connected to the phone.
- The magenta text indicates we have received a new prompt from the phone. The hex bytes are printed. This is the “wire” log.
- The cyan text prints what is decoded or to be encoded. This is the “reach” log. In this illustration the request for device info is answered.
- Finally, the command “lm ?” shows you the bit field values of the log mask. You can enable or disable various bits of logging.



On your phone you can select “Discover Commands” and then “Send Command”. This will list the commands.

The remote command line interface (CLI) is disabled by default. When enabled the command line of the device is accessible on the phone. Rather verbose logging is enabled on the device by default. Commands are provided to minimize and maximize the logging. More finely grained control of the logging is provided through the CLI.

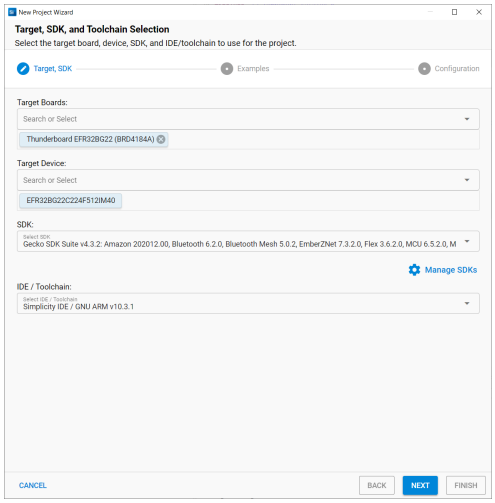
You can experiment with the file transfers. When all logging is disabled (command “lm 0” on the device console) and the “ack rate” is set relatively high (say 50 or 100), a file read transfer rate over 300kbps is easily achieved. Write transfer rates are over 100kbps.

Building the Demo

You can rebuild the application and modify it to your needs. The process is described here. It assumes that you have already run the demo on a Thunderboard.

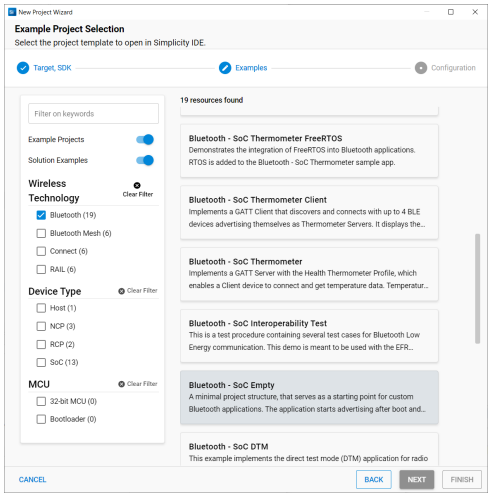
Create a new Project

Use the New menu entry to access the “Silicon Labs New Project Wizard”.



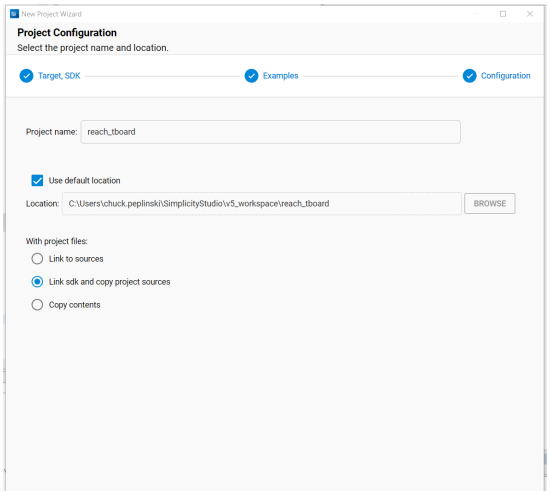
Choose the Thunderboard EFR32BG22 (BRD4184A) using the EFR32BG22C224F512IM40. If your thunderboard is attached to the USB port these should be selected automatically. You want the latest Gecko SDK, here 4.4.0, and the GCC compiler, here “Simplicity IDE / GNU ARM v12.2.1”

Click on “Next”.



Select the “Bluetooth SoC Empty” project. This will work best to duplicate the demo. Other starting points might be more appropriate for a larger project. For instance, SoC Empty has no RTOS.

Click on Next.



Rename the project to “reach_tboard”.
You can accept the default location or choose your own location.
Choose “Link SDK and copy project sources”.

More advanced users of Simplicity Studio might change these settings. They may have an impact on various configuration files.

Copy in the Reach Source

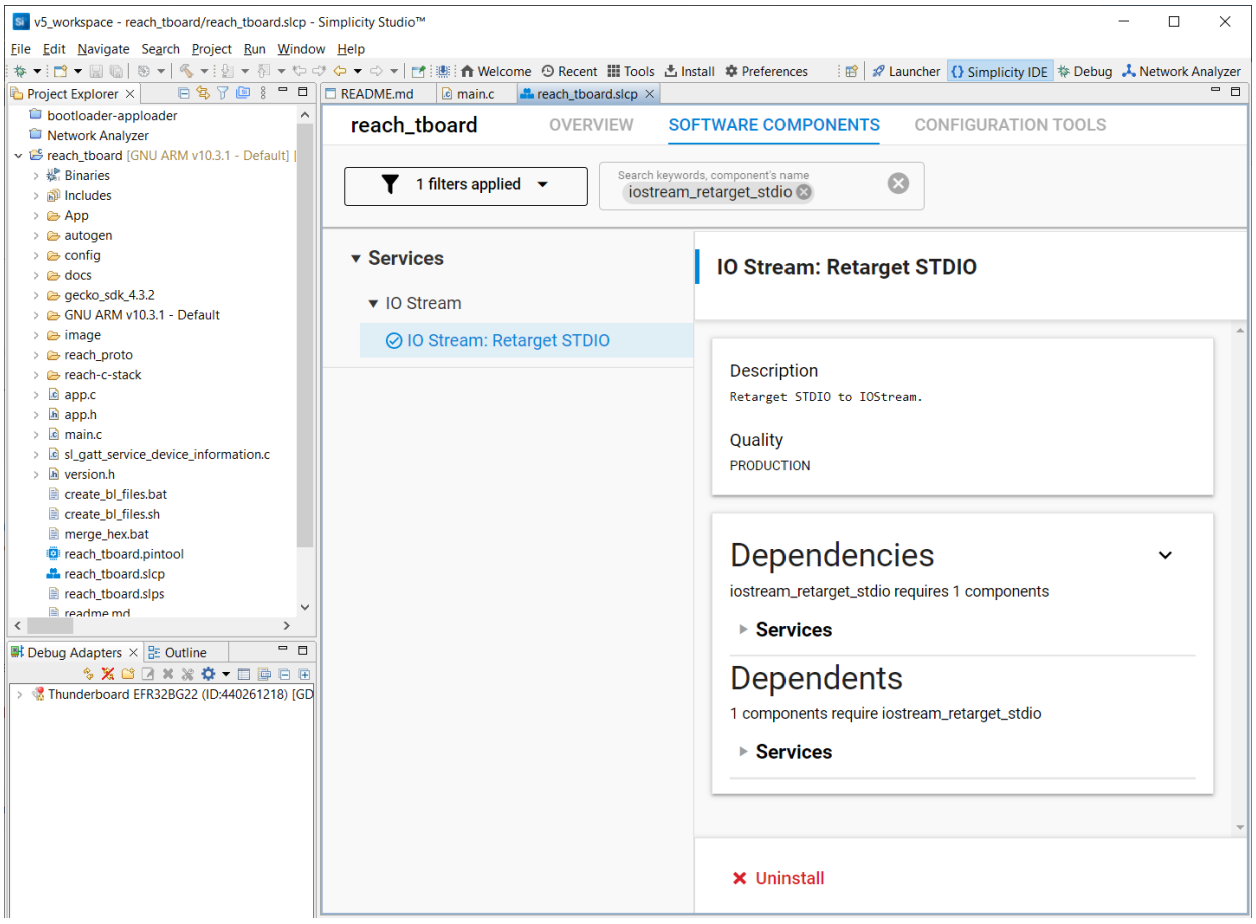
Copy the contents of the Reach source package over the new project. A handful of files are overwritten, most importantly app.c, the .pintool file and the gatt_configuration.btconf file. Most files are new to the project.

Fill in Dependencies

With the project open in Simplicity Studio, double click on reach_tboard.slcp file and open the software component manager. For each listed component, enter the name in the search box and install it with any given comments.

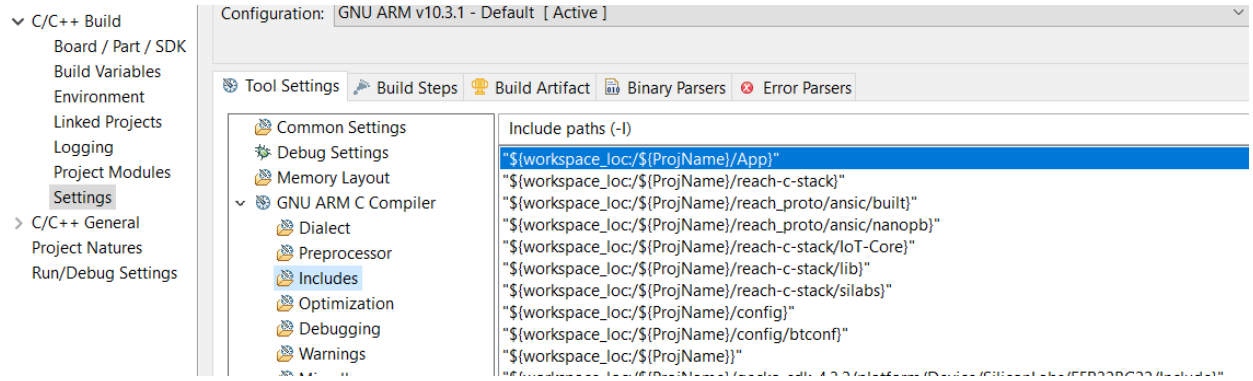
Install the following components:

1. Iostream_retarget_stdio
    a. Depends on iostream_usart_core
2. CLI Instance
    a. Create an instance named “cli”
3. Iostream_usart
    a. Create an instance named “vcom”
    b. Disable flow control.
    c. Convert \n to \r\n
4. Iostream_stdlib_config
5. Tiny Printf



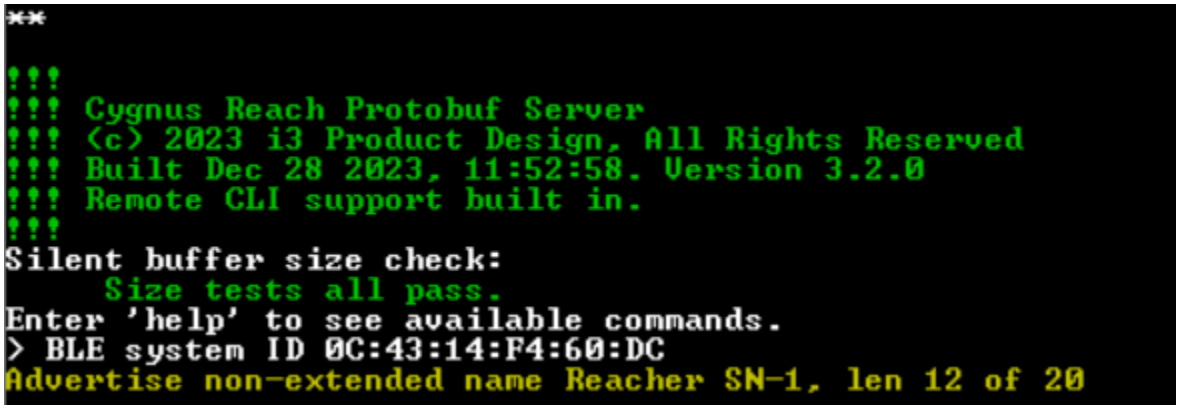
Add the necessary include paths. Right click on the project in the project explorer and select properties. Open the C/C++ build tab and choose settings. Select “Includes” for the GNU ARM C Compiler. Add the following:

1. ${workspace_loc:/${ProjName}/App}
2. ${workspace_loc:/${ProjName}/reach-c-stack}
3. ${workspace_loc:/${ProjName}/reach_proto/ansic/built}
4. ${workspace_loc:/${ProjName}/reach_proto/ansic/nanopb}
5. ${workspace_loc:/${ProjName}/reach-c-stack/IoT-Core}
6. ${workspace_loc:/${ProjName}/reach-c-stack/lib}
7. ${workspace_loc:/${ProjName}/reach-c-stack/silabs}



The project should build with this configuration.

You should be able to “Debug As” a “Silicon Labs ARM Program” by right clicking on the project in the explorer. You should see the startup banner in your serial console with the new build date.



The Cygnus app should be able to connect to you.

The advertised name will contain the last three octets of the system ID unless you store a serial number into the NVM using the “sn” command. You might also note that the first time you run the program it will print red as it initializes the non volatile storage.

Beyond the Demo

The source package includes a “Programmers Intro” document. It contains a lot more background about the Reach package which should be helpful as you customize it for your own purposes.