ZRAD Z-Wave Reference Application Design

# Introduction

The Z-Wave Alliance Z-Wave Reference Application Design (ZRAD) is a simple to copy, easy to modify, Open-Source *Reference* design of a Z-Wave best-in-class RF range device. ZRAD is not a product you can purchase, but anyone is welcome to manufacture and sell a product based on ZRAD. The objective is to accelerate Time-To-Market for Z-Wave products thru a simple to follow example with detailed step-by-step instructions.

See the [README.md](https://github.com/drzwave/ZRAD/blob/main/README.md) file in the repo for the latest information on building and using ZRAD.

## Features

* Public GitHub repository
  + Open-Source Repository
  + MIT License
* KiCAD schematic & PCB layout
  + Easily imported into other tools like Altium
  + Gerbers for immediate production
* EFR32ZG23 based +20dBm
  + 512K FLASH, 64K RAM
  + ARM CM33 CPU 39MHz
  + UARTs, Timers, I2C, ADC, DAC
* CP2102N USB->UART host interface
* Standard SerialAPI Firmware
* Documentation with full details for customization and replication
* Battery powered End Device option
* Target Assembled/tested kit cost less than $50@10K (not including enclosure)
* Target BOM cost under $15@10K
* White Paper on Z-Wave Antenna Best Practices and RF range field test results

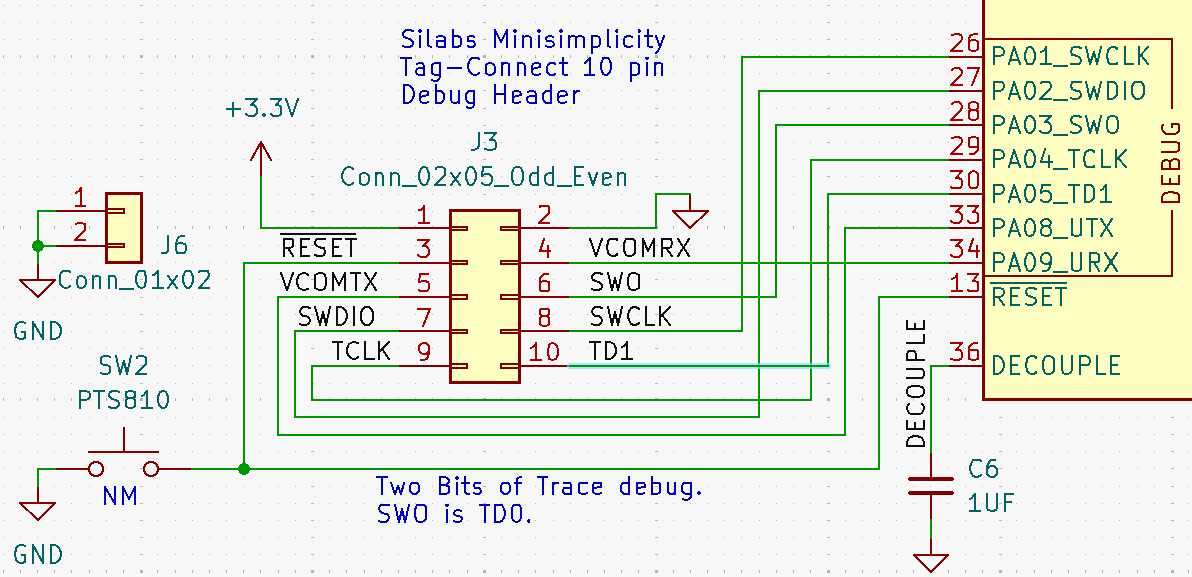
# Theory of Operation

## Z-Wave MCU

Naturally the heart of ZRAD is the Silicon Labs EFR32ZG23 SoC. The ZG23 has an ARM CM33 CPU, a long list of standard peripherals and the Z-Wave radio. The ZGM230 module is limited to +14dBm (or half the RF range) compared to the SoC at +20dBm. The SoC requires tuning the 39MHz crystal on every unit produced but this is easily done in a few seconds on the production line. The procedure for tuning is described below and scripts are available in the Repo.

The UART Rx/Tx pins are connected to the CP2102 thru 100 ohm resistors to allow the Tag-Connect debug cable to drive the pins during manufacturing. RailTest is needed to send/receive UART commands during crystal calibration. While this does not provide complete isolation between the CP2102 and the debug connector, the debug connector is typically only connected for a few seconds during final test making this a viable low-cost solution.

A Tag-Connect 10 pin header with alignment pins is used for programming and test during production and debug. Two Trace pins are wired to the header to enable Trace debugging using the adaptor board in the [etm\_zwave](https://github.com/drzwave/etm_zwave) repo. The PTI pins are not wired as Z-Wave typically uses a separate standalone Zniffer. The 2 ground pins are often needed for Trace debugging to improve signal integrity across the 10 pin ribbon cable.



The ZG23 DCDC switching power supply is utilized to convert the 3.3V to 1.8V for the VDD power supply. The USB 5.0VDC is passed thru a linear LDO to drop the supply voltage to 3.3V. Power spikes of about 130mA occur when the radio transmits at +20dBm. The LDO and USB handle that sudden change in supply current ensuring good RF range. The PAVDD is powered with 3.3V to power the RF amplifier to +20dBm. The analog supply and IOVDD are on the 3.3V to provide better analog voltage range and connectivity with other external devices. The internals of the ZG23 are powered with 1.8V. Sensitive power supplies have additional ferrite beads and capacitors to ensure clean supply voltages.

## Antenna Filter and Match

The RF path from the ZG23 to the RF50 signal is straight from the Silabs DevKit reference designs. This path filters the RF and shifts the high impedance pins to be close to 50 ohms. Between the matching circuit and the SMA connector is a standard 3 element pi filter used to match the impedance of the antenna. The values for these 3 elements are typically tuned to a specific antenna and RF region. A U.FL footprint is in this path to make tuning easier via a U.FL cable to a network analyzer. The U.FL is only mounted to bare PCBs to help with tuning. Thus, it is listed as “NM” in the schematic and is why there is a red X thru it. The SubGx1 pins are not used.



## USB-C Interface

A Silicon Labs CP2102N USB to UART chip is used to connect the SerialAPI to the host computer. A USB-C connector is used but only USB 2.0 is supported. USB3 is not supported or needed as the data rate of the UART is 115200 baud. The design follows the reference example in the CP2012 datasheet. This [article](https://hackaday.com/2023/08/07/all-about-usb-c-example-circuits/) was used as a guide. A CP2105 dual UART is pinout compatible and might be used in a future release to include support for Zniffer via PTI. Pin 12 of the CP2105 is wired to the PD04 pin of the ZG23 which can be used for 9-bit data mode of the PTI component.

## Power Supplies

USB-C provides +5V power in the typical application of a controller. The +5V is reduced to 3.3V via an LDO to power the on-chip switching regulator on the ZG23. The radio is powered from +3.3V to achieve a transmit power of +20dBm to the PAVDD pin.

When prototyping a low-power end device, the battery holder can be populated and connected to a CR123A or similar 3V battery. No reverse battery protection is provided so the battery holder must ensure the polarization. The LDO and USB devices are not populated when developing a battery powered device.

## PCB Ground Plane

A very important part of an IoT radio is the ground plane. Ideally the ground plane is perpendicular to the antenna and is half-wavelength. At 868MHz the wavelength is 34.5cm and 920Mhz is 32.6cm. The exact dimension isn’t that critical as long as it’s close. The ½ wavelength is 17cm but a square is more cost effective than a disk, thus divide by the diagonal (1.41) to get a 12cm on a side or roughly 5” square PCB. The ground plane fills the entire bottom side of the PCB to ensure there are no eddy currents or blockages. The next layer is used to distribute power. The upper two layers are used for interconnection of the ICs. All layers are flood-filled with ground as much of the area is open to provide a large ground plane for ideal RF performance. A smaller PCB size can be easily prototyped by simply making the PCB smaller with the potential of a loss in RF performance.

A related calculation is the width of the infeed trace. The width of the infeed trace is calculated using one of the many coplanar waveguide calculators such as: <https://chemandy.com/calculators/coplanar-waveguide-with-ground-calculator.htm>.

For [OSHPark](https://docs.oshpark.com/services/four-layer/), the relative dielectric constant is 3.61. The thickness of the prepreg is .2021mm. Usually the gap is set to .3mm then try various widths until the calculator comes up with close to 50 ohms. For ZRAD using OSHPark for PCBs, the calculated width is .44mm. Most of the infeed trace is 0.44mm except where it needs to narrow to reach the pins of the ZG23.

## RGB Color LED

An RGB Color LED can be used to provide visual status of any sort. Utilizing this LED requires customizing the SerialAPI. The LED can be used when ZRAD is used as an end-device for Indicator CC support. The LED is Active Low meaning the LED is ON when the pin is a zero (0V). The LEDs are driven directly from the GPIOs without extra FETs to keep costs low. This limits the brightness of the LEDs but they are sufficiently bright for most applications.

* Red = PC04 (LED3\_GPIO)
* Green = PA00 (LED1\_GPIO)
* Blue = PA10 (LED2\_GPIO)

The pins can be assigned using the Z-Wave Target Boards component in Simplicity Studio. Open the .SLCP file and select the Software Components tab. Scroll down to the Z-Wave Target Boards component and click on the gear icon. Assign the pin as shown above. The standard Switch On Off sample app will then control the green LED as the state of the switch and the blue LED will blink when in Learn mode or via Indicator CC. If SSv5 is not properly setting the configuration, see the README.MD file for the latest workaround.

## Learn Mode Switch

PC05 is connected to a SPST momentary pushbutton switch. This pin is an EM4WU pin so it can wake the chip from EM4. End Devices use this pin to enter Classic Inclusion/exclusion mode. Press and hold for more than 5 seconds to factory reset ZRAD. Controller applications do not populate this switch.

## I2C via QWIIC connector

Tons of handy and cheap I2C devices from Sparkfun use the tiny 4 pin QWIIC connector. There are sensors of every type, various displays, and GPS receivers. To add I2C to the ZRAD, install the Platform->Driver->I2C->I2CSPM component. Configure it with: Leave the Reference clock frequency at 0, Speed Mode=Fast Mode (400kbits/s), Selected Module=I2C0 (or I2C1), SCL=PB00, SDA=PB02. Reference clock frequency of 0 sets the reference frequency to the same as the I2C peripheral clock which is usually the HF clock (39MHz) (based on following the code). Once configured, the pins should show up in the PinTool. sl\_system\_init() calls sl\_driver\_init() calls sl\_i2cspm\_init\_instances() which then configures the I2C block on startup. The only function is the [I2CSPM\_Transfer](https://docs.silabs.com/mcu/5.9/efr32mg13/group-I2CSPM)(). The Z-Wave Multilevel Sensor sample app uses the I2CSPM which can be used as an example to see how it works.

## DEBUGPRINT

DEBUGPRINT can be defined and used to print messages via the MiniSimplicity headers VCOM pins. Typically, the USART is used for this purpose. First install the Services->IO Stream: USART component. Configure for 115200 baud, USART0, RTS/CTS=none, Rx=PA09, Tx=PA08. Then install Z-Wave-> Debug Print and uncomment the #define DEBUGPRINT in app.c or as a compiler symbol which will enable all printfs in every file.

## Optional Connections

Extra connections are incorporated in the design to allow prototyping of many other types of Z-Wave devices. These connectors and devices are normally NOT mounted for SerialAPI applications.

* 32KHz crystal supports Time CC. The on-chip low-speed oscillator drifts by a few minutes per day making it not usable for schedules or other time-of-day uses. The crystal reduces drift to a few seconds per day.
* Two 5 pin GPIO headers provide generic connections to other devices
* SW2 is an optional switch connected to RESET which is handy when debugging End Devices

## Crystal Calibration

See the README.md file and scroll down to the Crystal Calibration section for the procedure to calibrate the 39MHz crystal. Calibration is unique to each unit and must be performed during final test. Calibration is required for maximum RF range across environmental conditions and aging.

# Regulatory Preparation

Z-Wave products typically require regulatory approval thru appropriate test facilities. The regulatory bodies are FCC in the US, ISED in Canada, CE for Europe and others. Typically test houses want a device that will simply power up and turn the radio on in various modes. More details are in [UG523](https://www.silabs.com/documents/public/user-guides/ug523-bring-up_test-hw-development.pdf) and [UG409](https://www.silabs.com/documents/public/user-guides/ug409-railtest-users-guide.pdf). For Silicon Labs devices, the firmware required is called RailTest. Simplicity Studio has a project called RAIL – SoC RAILtest which must be customized for ZRAD. Follow the steps below to create a usable railtest firmware. Note that railtest is a standalone program that does not need a bootloader and overwrites NVM so the DSK is typically lost when railtest is flashed into a device.

1. Plug in a Silabs devkit or select one in the My Products pane of the Launcher perspective (xG23-RB4210)
   1. This will automatically set the pinouts to match ZRAD – primarily for the UART
2. Create the RAILtest project – RAIL-SoC RAILtest
3. Double click on the .slcp file
4. Click on the Software Components tab
5. Enter NVM3 in the Search window
6. Scroll down to: Services->Command Line Interface->Extensions->CLI:Storage in NVM3
7. Install it and be certain to name the component “inst0”
8. Build the project and flash it to the DUT
9. Open a serial port to the USB port
   1. or using SSv5 right click on the WSTK and select Launch Console
   2. Then click on the Serial 2 tab, then press ENTER in the window at the bottom
10. getversion will return the current version of railtest
11. Can also enter HELP for a list of all railtest commands

Below are specific details on the important railtest commands for Z-Wave.

|  |  |  |
| --- | --- | --- |
| Railtest command | Results | Comments |
| rx 0 | {{(rx)}{Rx:Disabled}{Idle:Enabled}{Time:240676874}} | Turns off the radio – required before changing radio configuration |
| setzwavemode 1 3 | {{(setzwavemode)}{ZWAVE:Enabled} {Promiscuous:Enabled}{BeamDetect:Enabled}{PromiscuousBeam:Disabled}} | Configures Railtest for Z-Wave mode |
| setzwaveregion 1 | {{(setzwaveregion)}{ZWaveRegion:US-United States} {ZWaveRegionIndex:1}} | US mesh  Ch0=916.0 100kbps GFSK  Ch1=908.40 40kbps FSK  Ch2=908.42 9.6kbps FSK |
| setzwaveregion 11 | {{(setzwaveregion)} {ZWaveRegion:USLR1-United States, Long Range 1}{ZWaveRegionIndex:11}} | US Long Range Channel A (DSSS)  Ch 3=912.0 100kbps OQPSK |
| setzwaveregion 12 | {{(setzwaveregion)} {ZWaveRegion:USLR2-United States, Long Range 2}{ZWaveRegionIndex:12}} | US Long Range Channel B (DSSS)  Ch3=920.0 100kbps OQPSK |
| Setchannel 0-3 | {{(setchannel)}{channel:3}} | Set the frequency based on region |
| Setpower xxx [raw] | {{(setpower)}{powerLevel:240}{power:210}} | Set transmit power  XXX = deci-dBm (140=14.0dBm)  RAW changes to “raw power level” instead of deci-dBm |
| settxtone 0 or 1 | {(settxtone)}{Stream:Enabled}{None:Disabled} {StreamMode:Tone}{Time:1251689899}} | Carrier On (1) or Off (0) |
| settxstream 0 or 1 | {{(settxstream)}{Stream:Enabled}{None:Disabled} {StreamMode:PN9}{Time:3481955496}} | PN9 modulated On (1) or Off (0) |
|  |  |  |
| clearscript 1 |  | Removes the script in FLASH |
| enterscript 1 |  | Begin capturing script commands and store to FLASH  The script is automatically run on powerup |
| endscript |  | End of the script |
| printscript |  | Print the script |
| runscript |  | Runs the script for testing purposes |

Sample commands to for a 908.40 MHz carrier that turns on when powered up:

clearscript 1

enterscript 1

rx 0

setzwavemode 1 3

setzwaveregion 1

setchannel 0

settxtone 1

endscript

For Z-Wave Mesh the maximum transmit power must be determined using the procedure in the [Z-Wave 700: Tx Power Calibration](https://community.silabs.com/s/article/z-wave-700-tx-power-calibration-adjustment-in-railtest-and-application-fw-x) or in this [KBA](https://community.silabs.com/s/article/z-wave-700-how-to-adjust-tx-power-on-slave-devices?language=en_US). Z-Wave Long Range uses dynamic power and the regulations allow up to +30dBm of transmit power but the Z-Wave chips can transmit at a maximum of +20dBm. Thus, ZWLR max power typically does not need to be set and can remain at the default of 200 (+20dBm).

# Board Status:

Rev A: 2 boards scrapped – USB chip difficult to solder due to pads under the IC.

Rev B:

* 1: ZG23A, SE=2.2.4, CTune=0x87
  + 13013-20757-03685-13998-02429-19075-02920-54669
* 2: ZG23A, SE=2.2.4, CTune=0x89, SerialAPI (no DSK)
* 3: ZG23B, SE=2.2.4, Ctune=0x98 – **shipped to David Zima** 4/19/2024 with Railtest
* 4: ZG23B, SE=2.2.4, Ctune=0x95 – works ok as an ED – Has a CR123A battery holder & reset button
  + 16512-58082-50459-58629-17292-55749-28421-30309
  + CP2102 isn’t connecting via USB
  + Suspect a short under the USB connector on one of the data pins
* 5: ZG23B, SE=2.2.4, Ctune=0x99
  + 54817-13409-17164-25837-27750-16443-46416-26908
* 6-12: bare boards

Rev C:

* 1: ZG23B, SE 2.2.4, CTune=0x94
  + 12041-60201-44764-43614-63900-49745-32509-52297
* 2: ZG23B, SE 2.2.4, Ctune=0x98
  + 31350-48981-20595-45123-33910-07500-29843-63102
* 3: Bare

# Reference Documents

1. EFR32ZG23 Datasheet
2. EFR32xG23 reference manual – details of all peripherals & registers
3. ZRAD Github repository
   1. See docs directory for this document and the Z-Reach datasheet
4. [CP2102N](https://www.silabs.com/documents/public/data-sheets/cp2102n-datasheet.pdf) Datasheet

# Issues:

1. Get GeoLocCC working
2. Code scripts to capture GeoLocCC and plot on a map
   1. The challenge is ZWLR must use S2Auth so I can NOT use raw SerialAPI as encryption is hard
   2. Can the SerialAPI do the encryption for me?
   3. Can I get Unify to send/receive GeoLoc? The PCC DLL? Z-Wave JS?
3. Measure RF range in ZW, ZWLR, EU and several antennas
4. RF review from Silabs – emailed to Mark 4/5/2024
5. 10 units built – 8 DONE – need 2 more…
6. Schematic & PCB layout – DONE
7. RF Range Testing
   1. 5 min video of results
8. White Paper

# Document History

|  |  |  |
| --- | --- | --- |
| DATE | Who | Description |
| 6/12/2024 | ER | Initial release |
|  |  |  |
|  |  |  |
|  |  |  |

# Journal

Details of the development and timeline are described here in reverse chronological order.

## 2024-08-05 – GeoLoc CC

See the Ubuntu laptop for more details. One interesting point is that making the temporary floating point variables on the ZG23 to be Double instead of just float didn’t seem to make any difference. I keep getting a power of 2 off somewhere. The longitude should be 70 but I keep getting 140. AHA! The problem is I was shifting the MSB by 23 instead of 24! I am now getting Longitude correctly into Z-WaveJS! Got the first Heat Map! ZwaveJS throws and unhandled error as soon as there is no ACK so I asked Dominic how to ignore those.

## 2024-07-30 – GeoLoc CC

Sidetracked on a personal project for the last several weeks but back to GeoLoc now. Sent Dominic more questions and hopefully have a call with him as I cannot figure out how/where to run the script to send GeoLoc commands.

The other option is if the SerialAPI can handle encryption for me. I’d much rather write a few lines of Python to talk to the SerialAPI and use the PCC to join devices. But I need something to handle encryption. SerialAPI can do encryption but ONLY for the end-device libraries, not as a controller. Bummer.

## 2024-07-15 – GeoLoc CC debug

Back to debugging GeoLoc CC to generate heat maps of the RF range. We also have plans to do range testing via skydiving, rocketry, and motorcycles in addition to just general neighborhood testing.

Domenic emailed me some basic instructions on setting up Z-Wave JS-UI for testing GeoLocCC. I would use the raw SerialAPI but we need to test using ZWLR which always uses Authenticated Security which is good because then we are only testing with 100% clean packets (1 corrupted bit and you cannot decrypt the message). But the SerialAPI doesn’t handle security S2 for me and it would be easier in general to use Z-Wave JS (I think). Initially I tried to get it working on my laptop but so much of Z-Wave JS relies on Linux commands (specifically in talking to the hardware) that I felt it would be easier to build a Linux laptop and use it for debug. I had an old Dell laptop just waiting for this purpose. See the ReadMe.txt file on the laptop for details on installing various software to get the system working.

I was able to have Z-Wave JS UI running in a docker, able to connect to the Controller, add a node via SmartStart. But it doesn’t retain the settings. See the ReadMe.txt for more details.

## 2024-06-11 – Trace Debug

Retesting Rev B using trace works perfectly. A 9.514MHz square wave comes out the TRACECLK pin. The only other option is that the JT2Mini board has a short on it? I don’t measure a short using a DMM. I only have 2 JT2MINIs and one has been modified for rev B. With the rev B Jt2Mini I do get the traceclk out on the wrong pins. So it’s looking like it’s the JT2Mini board. Yup – the 10 pin header on the JT2Mini had 2 pins that didn’t quite make connection down to the pads. Touch soldered and now the TRACECLK shows up on the test point and the scope but J-Trace still doesn’t see the it. So is the 20 pin connector not soldered well? Yes – that board for whatever reason must’ve had some oil or dirt on the pads and as a result didn’t make good solder joints. Fixed the solder joints. Trace is working well on Board C2. Thus, I can close out that issue. Trace complains the buffer is overflowing. See this [CASE](https://community.silabs.com/s/question/0D58Y0000AWmBmUSQV/trace-overflow-detected-using-segger-jtrace-on-zg23?language=en_US) for a discussion of the STALLENA bit. Setting STALLENA in MAIN works fine. Not sure why setting is via the Ozone script doesn’t stick. I filed this [case](https://forum.segger.com/index.php/Thread/9495-J-Trace-unable-to-set-STALLENA-on-ARMCM33-from-Silabs-EFR32ZG23-in-JLinkScript-b/) on Segger.

Back to GeoLocCC debug and documentation. GeoLoc is replying to a GET with a REPORT. But I need a script to pull the data out and dump a .csv which can then be plotted on a map. Updated the BOM by adding individual TABs for each build type. Released ZRAD as a public repo since the hardware is done!

## 2024-06-10 – Trace debug

Checking trace debug has been fixed. Unfortunately, my first go at it with board 2 doesn’t have the trace clock coming out. The project Ozone was using was in a tight reset loop. I need to rebuild it using the current project. Still not getting anything on traceclk or TD1. TD2 is toggling. GPIO.TRACEROUTEPEN is set to 0x0E which enables the traceclk, TD0 and TD1. PA3,4,5 and Pushpull. Thought maybe the pins where shorted on board #2 so I switch to #1 but get the same result. Traceclk is low, TD1 is high, TD2 is toggling. Wiring looks fine in KiCAD. Not shorted to GND. When I use Ozone in the debugger to set traceenb to 1 the debugger fails to read 4 bytes and then has to be restarted which is why I suspected the pin might be shorted.

The TCLK and TD1 labels on J3 are backwards. The wires were corrected but the silkscreen labels were not.

## 2024-06-07 – Documentation and testing of Rev C

Updated documentation in preparation for release. Tested the 2 new Rev C PCBs.

* Board 1: Ctune=0x6a=908.4434, 0x98=908.418, 0x94=908.420
* Board 2: Ctune=0x6a=908.448, 0x98=908.420

## 2024-06-06 – Rev C PCB assembly

Built 2 Rev C boards but I’ll need to order more components to build more as I’m out of a number of things.

## 2024-05-28 – GeoLocCC

Debugging converting the NMEA sentence into binary numbers to return via the GeoLocCC. Printf does not normally support %f and has to be enabled via settings->C/C++ Build->GNU ARM C Linker-> C Library-> check the box Printf float.

Ram usage : 53056 / 65536 B ( 80.96 %)

Flash usage : 479264 / 491520 B ( 97.51 %)

With the printf float disabled:

Ram usage : 53056 / 65536 B ( 80.96 %)

Flash usage : 474128 / 491520 B ( 96.46 %)

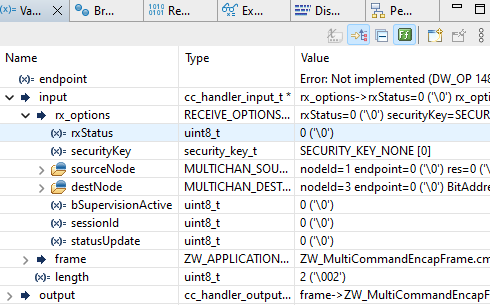
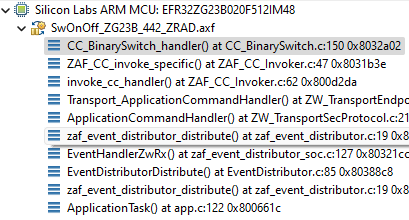
Which is 5K just to be able to print floats! But it’s not needed except to print so leave it off except when debugging.

Seem to have it basically working. Needs more testing and some trials!

## 2024-05-23 – GeoLocCC

Tried a few random ideas – commented out the INT32\_t in ZW\_classcmd.h and binary switch starts working again! WTF! There are no int32s at all in ZW\_classcmd.h file. Why would int32\_t make the code crash? The entire file only has UINT8\_T. Geo GET also returns a report so for whatever reason you can only have UINT8\_T in ZW\_classcmd.h! CO had no idea why but figures something must parse it and assumes everything is bytes.

## 2024-05-22 – GeoLocCC and I2C

The suspicion is that I2CSPM is what is blowing up Z-Wave probably because it is blocking instead of being interrupt driven which means the CPU is stuck in polling loops for long periods of time. There are several ways to go about checking this: 1) setup trace and see how long it’s sitting in poll loops (already know it’s a LONG time), 2) remove I2CSPM from one of the existing projects and get the code to compile and run, 3) make a fresh install without GeoLoc, add I2CSPM and see if Basic Set crashes, 4) rewrite the I2C interface to be interrupt driven instead of polled (I suspect I’ll eventually have do this), 5) wire up the UART instead of I2C (hand soldered unreliable connection – disadvantage of not having a standardized connector). #2 is easiest to start with – just comment out the code that runs the I2C. Basic works but geoloc get has a 0xD5 in RxStatus which is bogus. Try #3: Created SwOnOff\_ZG23B\_ZRAD\_GeoNoI2C- This project gets the 0xD5 in RxStatus so it is something in GeoLocCC. Must be an incorrect pointer to a type of some sort. Have to compare this with other CCs. Both basic and binary switch CC use REGISTER\_CC\_V5. Binary switch is also getting 0xD5 and thus is not responding to a Binary Switch GET. Let’s try the generic project. The generic project gets an rxStatus of 0 which is what is expected and thus returns a binary switch report.  This is the good project when a breakpoint is set right at the start of CC\_Binary Switch Handler.

Net result is that the problem is in the creation of the GeoLocCC and not the I2C driver. What the problem is though is a mystery. Clearly the rx\_options is filled with garbage in the broken version. But why would that break it? Is there a hardcoded limit somewhere we are exceeding? The input structure is in the AppStackBuffer (0x900 bytes) according to the MAP file.

I set a watchpoint on the address and it breaks in Transport\_ApplicationCommandHandler while pulled the value from rxOpt so I set another watchpoint there. Seems that portasm.c modifies address 2000BCDx which is where the data is coming from?

## 2024-05-20 – GeoLocCC

Ongoing effort to get Geographic Location CC to work. Working in the SwOnOff\_ZG23B\_442\_GeoCC project on Board B4. I keep getting RADIO\_BOARD\_EFR32ZG23=1 in the compiler preprocessor settings. I don’t see where SSv5 keeps pulling it from. I can’t factory reset the board because pressing the learn button immediately goes into the default handler. This is because PB1 is not defined. I had to use the source file (radio\_no\_board.c) to fix it and arbitrarily assign it to PC03. If it’s not assigned, then we get an Assert and go to the default handler. Confusing because the PB starts from 1 but the pin numbers start from 0. That got the push button to work and able to Factory Reset the board, join the network and PCC is able to control the green LED via basic on/off.

Commented out the code for zaf\_event\_distributor\_app\_zw\_rx which was a hack anyway. Need to find the source of the corruption thru the proper handler. Send a GeoLoc GET and I get this out of the UART:

BuildTxAckHeader: pFrame->frame.headerLength = 0D

BuildTxSingleHeader: pFrame->frame.headerLength = 0C

BuildTxAckHeader: pFrame->frame.headerLength = 0D

But nothing relative to getting into the GeoLoc code. Note the above is for the 1st time it’s sent which it has to resync the S2 Nonce. Once synced, only get one line with the 0x0D length. No idea where this DPRINT comes from as searching for the strings doesn’t find anything. The call to DPRINTF is from ReceiveHandler() in ZW\_transport.c in the library – set a breakpoint inside the library code for DebugPrintf to find it. If that’s in the library code, isn’t it always going to be executed and thus slow the code down? Can the compiler remove the library code if DEBUGPRINT is not defined?

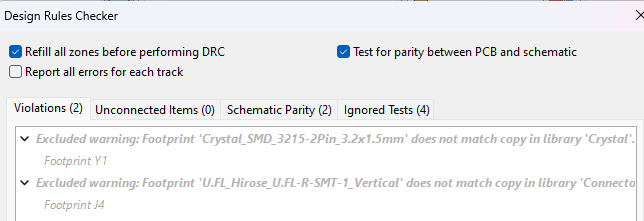
Basic Set and Indicator CC also don’t work at all once GeoCC is added. Commenting out the GeoLoc registration doesn’t fix anything. Something else is breaking the code. So, it’s a binary search to add things 1 at a time to figure out what’s causing the SDK to go into the weeds. Could it be that one of the other interrupts is causing the failure?

Create a new SwOnOff\_442\_GeoCC\_dbg where I’ll incrementally add things to find the cause. Followed the readme.md for ZRAD to setup the board with SwOnOff which is working non-secure. I can send a basic set to turn the LED on/off. Press the button to send Learn and DEBUGPRINTs are coming out uart.

Next step is to install GeoLocCC but I’m going to do it with GPS\_ENABLED not defined and then comment out almost everything except a few printfs. No changes to ZW\_classcmd.h. That works. I can send a GeoLoc GET and I get a printout from the DPRINT. So the next step would be to get it to respond by adding more stuff in. I’ll add the I2C last as maybe that’s what’s blowing it all up? I2C is blocking so maybe the CPU loses data or queues overflow? Trace might also help there.

## 2024-05-16 – Rev C PCB

Sergue at zwave.me confirmed that his zniffer is able to decode the 9 bit UART mode of the PTI module. Wired that to the CP2012 using the RX pin for the CP2015. This is just an option and at the moment is not supported but gives future possibilities. I ordered 3 Rev C PCBs.

Rev C schematic is ERC clean. The PCB has 2 excluded warnings due to footprint modifications which are OK. The footprints were slightly modified to fix other minor DRC warnings.

I want to build a small, cheap version of the board – 2 layer with PCB antenna 4”x2” which Zima says he’s had good luck with. Call it ZRAD-S for “small”. Prove out if the PCB ground plane makes a huge difference or not. Same schematic, different PCB.

## 2024-05-13 – Rev C PCB

Fixing a few errors in the Rev B PCB and adding a 2nd UART port via USB with a CP2105 as an option. Andrew Hayden mentioned they had gotten this to work. This would enable a zniffer at the same time as the SerialAPI which would provide super handy remote debugging capabilities. See Issue #5 on the repo for more details. Battery holder is backwards, the TCLK and TD1 pins are reversed and rename things to be ZRAD. Added more documentation.

Filled out a project via Screaming Circuits to see how much they would charge for a turnkey project of 25 boards - $185 per board with a 1 week turnaround without the EFR32ZG23 or the LDO (they could quote them – or we could choose another LDO). With a longer 31 day turnaround the price is down to $152 per board. They use Sunstone for PCBs which are quite expensive though even thru OSHPark the 4 layer PCB is expensive - $80 for 15 day turn or $112 for 5 days. SC Custom process has prices under $50 each. This is a complete turnkey production but doing any significant numbers are cheaper in volume (1K or more). Typically, there is a big price break at the 10K/yr production levels.

## 2024-04-26 – Railtest

David Zima is doing some RF chamber testing to determine if ZRAD can meet FCC at +20dBm TX power. He needs the Railtest commands to do it which were added to the Regulatory section above.

## 2024-04-23 – RF Range Testing

Today I will be testing mesh Z-Wave in US and EU regions at the Scammell Bridge in Dover NH. The first problem is checking that the devices are in EU region. The firmware tends to write the MFG\_ZWAVE\_COUNTRY\_FREQ and then gets stuck there even though the firmware is set to some other region. Downloading the firmware doesn’t seem to set it unless you first erase it via commander then overwrite it. The EU Zniffer firmware in 4.4.1. and 4.4.2 doesn’t appear to work on a thunderboard. Try a DevKit?

Using my N9912 SA I confirmed that board #5 is using EU frequencies. The zniffer on a TBZ however does not appear to be. However board #5 is always blasting out at 868! Even the standard SwOnOff 442 binary file turns on the TX 100%. Interesting. Board #5 was joined to the PCC with a ZWLR NodeID. So I completely wiped the DUT and started again initially with the standard SW On Off binary. OK, now it is working normally. The key is a full wipe including the \_REGION. The full wipe removes the CTUNE which is in the UD page? So it has to be reprogrammed along with the DSK.

I plugged in a devkit 548 and eventually go it into EU frequencies. Unlocked debug was not enough to clear the \_REGION! I had to use the command below! Taking hours to just get devkits into EU working order. Ugh. Very difficult to tell what’s going on with the Agilent SA.

The PCC is able to switch from US\_LR to EU via the SerialAPI (and back to US\_LR). I built an EU zniffer and programmed a DK with a ZGM230 on it with it as a dedicated EU Zniffer. My TBZ zniffer is back to a US\_LR zniffer. DK 548 has an EU SwitchOnOff demo in it.

Too late to take the equipment to the bridge for long range testing. But a quick walk thru my neighborhood and it works everywhere even around a hill. 160m thru several buildings or maybe around them?

## 2024-04-22 – RF Range Testing

The effort to add GeoLocCC has failed – see below. Using the ERTT to determine range. See the RFRangeTest.docx in the Test folder for details. Planning and some preliminary testing at the bridge in Dover NH.

Was hoping to be able to use the EK2705 chip antenna as a test but it seems it is no longer supported in SSv5.

Built an EU ZRAD but it does not talk EU because the MFG\_ZWAVE\_COUNTRY\_FREQ is set to 9. To fix it: commander flash --tokengroup znet --token MFG\_ZWAVE\_COUNTRY\_FREQ:0xFF

## 2024-04-20 – GeoLocCC

Transport\_ApplicaitonCommandHandler in ZW\_TransportEndpoint.c receives the rxStatus of 229 (0xE5). This comes from EventHandlerZwRx in zaf\_event\_distributor\_soc.c which is where the FreeRTOS queue data is pulled from.

I was hoping that with a ZG23B the code would not still have this bug. I have no idea how to get around this now as the bug is somewhere in the SDK it would seem. Has something overflowed a stack? Obviously, something is corrupt. Need to hack something to be able to do range testing next week so for now I’ll just use the zaf\_event\_distributor\_app\_zw\_rx() and put that in app.c. I can queue the REPORT but the ZAF doesn’t end up sending it I suspect because other bits in the TxOptions have invalid values and as a result the frame is not sent. Looks like the type for TxOptions\_t is not the right one. This code is so squirrely – everything is renamed, repackaged, copied, swizzled, its nearly impossible to follow. The main problem is there is no API!

The ZAF\_Transmit() function is the one that posts into the transmit queue QueueNotifyingSendToBack. Right now after queuing the REPORT the chip reboots obviously because an invalid pointer is passed thru the queue.

I have no idea how to fix this or even debug it. I emailed CO to see if he had any ideas.

The alternative is to code a simple app that sends a command to the hub and if it replies then record the GPS coordinates at that point. Then I don’t need to add a new command class. But time is short to make that happen. Might be easy enough to just light the green LED when a secure frame is exchanged, the blue LED when the secure frame fails but a NOP works, and red when nothing is getting thru. This is more obvious than the ERTT blinking an LED.

We’ll have to rely on the plain old ERTT for the summit unless CO can find a solution.

Oops – the battery holder footprint has + and – reversed. Added this to the Issue for Rev B.

## 2024-04-19 – GeoLocCC

Back to GeoLocCC since the ZG23B boards will require customized firmware just for B boards. I only have maybe 1 or 2 more ZG23A parts. I might have to order some. The other option is to rebuild GeoLoc for B boards since I’m unable to go any further with the current project since the SDK is giving me garbage.

1. Start with the wizard again with SwOnOff but with the ZG23B connected via Wstk2 mini simplicity header and board #5
2. Configure RF Region for ZWLR
3. SwOnOff project fails to build due to APP\_BUTTON\_A and LED not being defined. This is caused by the compiler setting the Preprocessor symbol RADIO\_BOARD\_EFR32ZG23=1. Note that RADIO\_NO\_BOARD is also set. Delete the RADIO\_BOARD\_EFR32ZG23 symbol and the project builds and runs (sitting in the WFI instruction).
4. Install DebugPrint as described in the ZRAD ReadMe.md
5. Configure LEDs/Buttons
6. Join the DUT to a controller – basic On/Off toggles the green LED
7. Install I2CSPM for QWIIC interface
   1. Basic set still works
   2. I had to remove the RADIO\_BOARD\_EFR32ZG23 again
8. Follow the instructions in the Geographic Location ReadMe.md
   1. Edit ZW\_ClassCmd.h file and soft link the files
   2. Had to remove RADIO\_BOARD\_EFR32ZG23 again
9. The SDK is still giving me garbage – the rxStatus has an 0xE5 in it which the code then thinks it is a multicast frame so it’s dropped.

I ordered 2 more GPS boards – 1 XA1110 and a slightly more expensive one to see if it has more accuracy. Sparkfun has a board that’s $179.95 that they claim is accurate to within 24mm! I also ordered 2 more Tag-Connect debug cables as they seem to disappear! I also ordered more antennas to have pairs of the same one. Unfortunately, the 868 recommended antenna is backordered and not expected until July! Must do EU with the less expensive antenna which were in stock.

Tested and CTUNEd board #3. Shipped it and a bare board to David Zima for testing.

## 2024-04-18 – ZG23B debug

The 3 new boards I built yesterday don’t run the firmware but get stuck in some sort of reset loop. I rewired an ETM\_ZWAVE adaptor board to fix the Trace debugging wiring mistake on the ZRAD (easier to rewire a 2 layer PCB than the 4 layer ZRAD board). Trace is working fine but it’s not easy to find the failure as the trace overflows. The task switch goes to a RAILINT which then unpacks a long list of stacked functions so I suspect there is a stack overflow somewhere. Single stepping with Trace tends to cause a hardfault accessing peripherals for some reason.

I get:

OnTraceStart() end - Took 5.22ms

LTRACE (Time since start: 0.112 578, Thread=ETM4): Trace overflow detected. Trace packets may have been lost.

For more information on how to avoid overflows see the following article: https://wiki.segger.com/J-Trace\_overflow\_error

Debug.Halt();

From Ozone with Trace enabled. Seems like the CPU might not have STALLENA enabled? I went thru this before with Segger and tried to get an answer from the Silabs CPU group but they had no idea. The Wiki page suggests: TRACE\_SetEnableStalling = 1 but it’s not a command. I recall that when Trace is enabled this is set by default (maybe?).

This [Community](https://community.silabs.com/s/question/0D58Y0000AWmBmUSQV/trace-overflow-detected-using-segger-jtrace-on-zg23?language=en_US) page asks the question is the STALLENA feature available in HW? The only other option is to prescale the CPU clock by 2 to slow it down to reduce the amount of trace data. Another option is to lower the CPU clock. The xG23 RM shows that the ETM is from SYSCLK but the CPU is based on HCLK so it should be possible to lower the clock speed by powers of 2.

Reduce the CPU clock speed and then trace does not overflow. But the clock speed isn’t sticky as the CPU overwrites the clock register on occasion. Add these lines to the JLINK script:

U32 CMU\_SYSCLKCTRL;

CMU\_SYSCLKCTRL = CMU\_BASE + 0x070;

JLINK\_MEM\_WriteU32(CMU\_SYSCLKCTRL, 0x03001);

JLINK\_SYS\_Report1("CMU\_SYSCLKCTRL: ", JLINK\_MEM\_ReadU32(CMU\_SYSCLKCTRL));

But it does seem to prevent overrunning trace.

The problem remains that it seems like the stack is overflowing somewhere.



The task switches to a huge stack of RAIL interrupts it seems. .text.RAILINT\_6433f1eb5ac17efd72f74a695e677e55 which all return but you would think that would cause stack overflows. This function is executed during startup inside of ZW\_RadioPhyInit(). When I break at function e55 I get different trace results:



Note we are inside ZW\_RadioPhyInit and everything is OK. No task switch and crazy stack. Trying to step thru the code from here ends up getting back thru reset at 0x0800170 which is the bootloader first instruction after the vector table. I suspect we’re actually underflowing the stack which is why Ozone thinks the stack is so deep.

Unfortunately, Trace isn’t providing me much help here as something is completely messed up. Well the good thing is that running the A code on a B devkit does exactly the same thing! So it seems code compiled for an A part will NOT run on a B part! I need to make a copy of the project and change the target to a B part. This also means I can use a devkit with trace with all 4 pins. Maybe that will give me more info? I also don’t have time to debug this!

I built SwOnOff using a B devkit. Then changed the target board to Custom Board and the part to an EFR32ZG23A and that project runs on a B part just fine. So, it’s not the fact that it’s an A build on a B part at least for the off-the-shelf SwOnOff. I enabled debugprint and set to US\_LR were the only customizations. The same code runs OK on my new boards so it’s not something special about the board it would seem. Nope, it works because the B is still listed as a symbol. So, switching the “target” in the Properties isn’t a valid way of switching the target as it doesn’t change the target (which is also a SSv5 bug). I used the .SLCP file to change the target. Then the project doesn’t compile because all the GPIOs and EUSART settings were lost. Ugh.

One more try of running A firmware on a B part. Start with a RB4202D in a WSTK (ZG23B). In My Products of the Launcher perspective select an EFR32ZG23A010F512IM48 part. Select the GSDK 4.4.2 SDK. New->NewProjectWizard, Next, Select Z-Wave then the SwitchOnOff project and CREATE it and name it SwOnOff\_ZG23A\_441\_runsonB. Click on Build – it should build OK. Click on the bug icon to download it and run it. It will run to main, but it then gets stuck in a reset loop forever. Click on Pause and the code should be sitting at the WFI instruction in EM1 waiting for something to do. Instead the code is often in a RAILINT doing a RequestIRCalibration() or waiting for the BURTC to sync or one of the other longer startup routines. If you then enable DEBUGPRINT (which requires some effort to enable the USART and assign the Ios) you will see it prints out just a few lines and then resets over and over again. I entered this [case](https://community.silabs.com/s/question/0D5Vm000008YhPxKAK/code-compiled-for-zg23a-does-not-run-on-a-zg23b). Seems when you create a project it loads a RADIO\_BOARD\_EFR32ZG21=1 as a Setting in Project Properties which might be what is blocking LED/Buttons.

## 2024-04-17 – GeoLocCC & 3 more boards

Call with Christian Salmony-Olsen to get Geographic Location CC V2 installed in a sample project.

* [How to implement a new command class](https://docs.silabs.com/z-wave/7.21.1/zwave-api/md-content-how-to-implement-a-new-command-class) on docs.silabs.com
  + Is a bit light on details but is a good start
* Linker Magic is the trick for adding CC to the NIF. Linux does this sort of thing which is where the concept came from
* All CO could suggest is look at what is coming in from the queue and there might be a bug in the SDK
* I did share GeoCC with CO and he will try implementing in the Trident stack

Join the GeoCC version classic without any keys. Still doesn’t turn the LED on via Basic. But the DPRINTs do say the command is supported. Sending a GeoLoc works! So at least I can get the GeoLoc debugged then figure out what the problem is with security later. ZWLR requires security so it will have to work eventually. But it doesn’t return the report. CO stated I just had to make the length non-zero and ZAF would send it. The reason it doesn’t send is because the rxStatus is 0xE5 which fails the is\_multicast function RECEIVE\_STATUS\_TYPE\_BROAD (0x04) so my code drops the frame. Obviously the rxStatus has garbage in it as E5 is an invalid value. Tracing that back to the xQueueZwRx and it is giving me 0xE5 so garbage coming from the SDK. I emailed CO asking what I do next.

Meanwhile I built 3 more boards. Board #5 removed about ½ the solder under the ZG23 EP so it would be less likely to lift seemed to help. But board 3 and 4 have the ZG23 down tight to the PCB even with all the solder paste. On all 3 boards I scraped off the excess paste to have just the thickness of the mylar and pressed the ZG23 down firmly. Much less solder shorts in general. None on the CP2102. Boards 3-5 cycle thru eResetReason 7 quickly – a few hundred ms so its not the watchdog. They do print out messages. Board #3-5 have ZG23**B**. They have SE firmware 2.1.7 – Updated to 2.2.4 but that made no difference.

I tried to single step to find what’s causing the reset but it’s something after the init routines when FreeRTOS starts up. So I tried to setup trace but the J-Trace is not showing any target power. Loose connection. Now have power but trace isn’t capturing. The TRACECLK is only 2.3MHz which is way lower than it should be. That’s because D2 is the traceclk since it’s at 10mhz. The board has the tracelk and D2 backwards. So, no trace until Rev C. I entered an Issue to fix that. The PCB is hard to rework since the trace layer is on an inner layer and not the bottom. I could rework an ETM adapter board but even that is pretty hard to do. I may go there if nothing else works.

## 2024-04-16 – GeoLocCC

After installing the Geographic Location CC, a Basic SET no longer works nor does the binary switch set! This is likely due to the NULLs used in declaring GeoLocCC support for Basic CC. But that shouldn’t impact the binary switch set but it does. Seems like the geoloc handler isn’t being called at all.

Apparently by adding GeoLocCC I need to add a function called zaf\_event\_distributor\_zw\_rx(SZwaveReceivePackage RxPkg) which there is a weak version that does nothing. This function is here as an extra call to anything not handled by the command class handler in the next paragraph.

I expect the ZAF\_CP\_CommandPublish to handle the GeoLocCC since it is a subscribed CC. But it is quite difficult to step thru this code. So switch to Ozone. EventHandlerZwRx() is the main function for handling the ZW Rx queue. Made a copy of ZAF\_CmdPublisher.c to enable debugprints in it. I commented out the registration of GeoLocCC and turned off the timer to get the CC to work. There are still 3 CCs registered even though GeoLoc no longer shows up in the NIF. The CC and Cmd are 00 for all 3. Basic and binary switch don’t work. Something has blown up the CC registration. But that code is super hard to follow. Had to create yet another SwOnOff to figure out what these are supposed to have in them. This involved redoing the LEDs, USART, DEBUGPRINT which takes upwards of an hour to plug back in each time. The DPRINTs in the temporary project also list the CmdClass and Cmd as 0. And yet somehow it is able to process the correct binary switch command class. Aha, the subscriber list is not for the CCs but instead Application, Security, Multicast as line 40 of ZAF\_Common\_helper.c describes. The key is the call to Transport\_ApplicationCommandHandler() which in turn calls invoke\_cc\_handler() which then finally goes thru the list of command classes. The list is from ZAF\_type.h using extern const CC\_handler\_map\_latest\_t. \_cc\_handlers\_v3 are identified in the .map file so this is created at compile time. The SwOnOffTmp is able to control the LEDs via a SwOnOff or Basic so it’s CCs are working fine. So why isn’t the version that I put the GeoLocCC into? The GPIOs are still wired up it seems. Never gets to invoke\_cc\_handler for some reason. Gets to ApplicationCommandHandler(). Eventually gets to TransportCmdClassSupported and for some reason fails searching thru the secure list (the device is joined S2Auth). I send the command from the PCC securely but the securityKey is security\_key\_none? WTF. The whole RxPayload directly out of the xQueueReceive is all zeroes so something is blown up? Well the TMP version (which works) also gets all zeroes? So how does it get past the security check? It’s the 2nd time thru that is all filled in. I think the first time thru just strips off the security stuff. Sometimes it’s the 2nd pass? In the GeoLoc version the 2nd pass how various non-zero values but security is still None. The first time the EZwaveReceiveType is 128, the 2nd is 132 but iLength is 0 and key is None. The temp project gets type=128 with Key=Auth and source/dest and other fields filled in, then type=132 key=None. Type=132 is StayAwake. The StayAwake can be ignored since this is an AOS. Seems like the Geo project is broken inside the SDK as the queue is delivering garbage. My choice is to start over again but I could use the zaf\_event\_distributor\_zw\_rx to hack in the GeoCC and not try to insert it via other code? This was a solid ½ day of debugging that got me nowhere. How does Silabs expect people to add code to their sample projects?

I emailed Christian to see if he can help tomorrow. I will need to build more boards tomorrow and do some preliminary testing and cannot spend much more time on getting GeoCC working.

## 2024-04-15 – I2C & GeoLocCC

Followed the instructions for install I2CSPM which I had to manually fix the sl\_i2cspm\_QWIIC\_config.h file but it was easy enough. Next step is to install GeoLocCC by following the instructions in the ReadMe.md.

## 2024-04-12 – GeoLocCC – rename project from ZReach to ZRAD

ZWA Marketing didn’t like the name ZReach so they came up with ZRAD for Z-Wave Reference Application Design. Thus, the Repo was renamed to ZRAD as were several files but not all yet.

One of the main purposes of ZRAD is to give an example of a best-in-class RF range device. Thus, we need to measure the range in real world locations. Start with testing at an open-field site. Open-field sites are hard to find in New England as we have lots of hills, trees and buildings. There are few sites with more than 1 mile of open space.

Plan for RF testing:

* Location – Merrimack River in Haverhill MA – 1+mi LOS down river with adjacent sidewalk but there are trees between the road and the river
  + Initially head west toward bridge – if connection remains strong
  + Head east down river – note the buildings in the way at the beginning
* Geographic Location CC loaded into a SwOnOff ZRAD
  + Read out the exact GPS location while the DUT is moving
  + Controller will GET the GPS location and store it with Tx power and stats
    - S2 Security encrypted = zero bits corrupted
    - SDK Retries are OK – no application retries
    - Quantitative threshold for communication vs. subjective (led blinking)
    - Store .csv file which can then be plotted on a map in excel
  + GPS sensor is the SparkFun XA1110 with a small antenna
    - Ensure the Blue LED is blinking indicating it has satellite lock
    - Should not be a problem since testing is outdoors in open space
* All trials are joined with S2 Authenticated requiring 100% perfect data or frame cannot be decrypted
* Devices without GeoLoc CC are done using ERTT sending Basic On/Off to blink LED
  + Location data is recorded manually and threshold is subjective
  + Seems to be a way to share via Google maps? Apple maps will not give GPS coords
  + Turn on location sharing and share with a google account then they can see it on google maps
* Trials:
  + ZRAD Controller ZWLR antenna #2 + ZRAD SwOnOff antenna #2
  + ZRAD Controller antenna #2 + ZRAD SwOnOff antenna #2 – classic ZW instead of ZWLR
  + ZRAD Controller EU +14dBm + ZRAD SwOnOff antenna #2 EU
  + Same as above but with antenna #1
  + Same as above but with poor antenna on ZRAD SwOnOff
  + TBZ Controller ZWLR + ZRAD SwOnOff antenna #2
  + ZG28 devkit ZWLR (chip antenna) + ZRAD SwOnOff antenna #2
  + UZB + ZRAD SwOnOff antenna #2
  + ZRAD Controller EU antenna #1 + off-the-shelf EU device
  + ZRAD Controller ZWLR antenna #1 + off-the-shelf ZWLR device (homeseer PS100) PCB antenna
  + ZRAD Controller ZWLR antenna #1 + other ZWLR device

Ongoing work with GeoLocCC. Adding GeoLocCC to my SwOnOff\_ZG23\_441\_GeoCC project. The previous GeoCC project was for a ZG230B so there are a number of challenges with that though it does seem to work on the ZRAD board. Renaming projects seems to blow up SSv5. So I started a new project using 442. Struggling just getting DEBUGPRINT installed. I configure the USART pins but doesn’t seem to change the config files. Click on SOURCE then manually edit the USART pins to connect them and then it compiles OK. The LED and button pins also had to be manually edited.

## 2024-04-11 – I2C Debug – set ctune

The spaces in the NMEA are from i2c\_read=32 which should never happen. Tried to set a [watchpoint](https://community.silabs.com/s/article/simplicity-studio-v4-watchpoints?language=en_US) on i2c\_read>31 but it doesn’t work. Ozone is also not able to set a watchpoint expression. Ozone can set breakpoints on a write or a read of a variable but not on an expression so I suspect the issue is the HW debugging logic doesn’t support it.

The problem was comparing with <= instead of <. Found this by inspection. Trace probably could have found it quickly but I haven’t set that up yet. Fetching the NMEA sentence reliable now.

+....NMEA=$GNGGA,133700.000,4310.242586,N,07052.281637,W,1,9,1.11,42.117,M,-32.898,M,,\*4A

+.........NMEA=$GNGGA,133701.000,4310.242974,N,07052.281336,W,1,9,1.11,42.052,M,-32.898,M,,\*4E

+.........NMEA=$GNGGA,133702.000,4310.242672,N,07052.281168,W,1,9,1.11,41.993,M,-32.898,M,,\*4A

+..........NMEA=$GNGGA,133703.000,4310.242547,N,07052.281174,W,1,8,1.18,41.942,M,-32.898,M,,\*47

+....................NMEA=$GNGGA,133704.000,4310.241100,N,07052.280788,W,1,8,1.18,42.034,M,-32.898,M,,\*4B

+.........NMEA=$GNGGA,133705.000,4310.239380,N,07052.279957,W,1,7,1.27,42.113,M,-32.898,M,,\*42

+.........NMEA=$GNGGA,133706.000,4310.239013,N,07052.279833,W,1,7,1.27,42.342,M,-32.898,M,,\*4D

+..........NMEA=$GNGGA,133707.000,4310.239051,N,07052.280373,W,1,6,1.34,42.311,M,-32.898,M,,\*46

The + is the timer which is set to 933ms. Each . is another I2C buffer of 32 bytes. Once the NMEA is found, then the sentence is printed.

Need to set CTUNE for the 2 boards built so far.

1. Download RailTest into the DUT
2. Rx 0
3. Setzwavemode 1 3
4. Setzwaveregion 1
5. Setchannel 2
6. Settxtone 1 – turns on the radio carrier
7. Getctune – record the CTUNEXIANA value in hex
8. Use TinySA Ultra with start=908.3 stop=908.6
9. Measure the peak which the TinySA should display – goal is to be within 1ppm of 908.420MHz (1000Hz)
10. If the peak is high, set ctune to be a higher value, if low, try a lower value
11. Settxtone 0
12. Rx 0
13. Setctune XXX
14. Settxtone
15. Measure – go back to step 11 until within 1000Hz

Board 1 started at 0x6a which measured at 908.4376. 0x78=908.4289, 0x83=908.4226, 0x89=908.4192 which is close enough, 0x87=908.4202.

RailTest does NOT program the CTUNE value in the NVM. It just sets it in RAM. Execute the following command to set it permanently (in NVM) – commander ctune set –-value <ctunevalue> -d EFR32ZG23

Always do a ctune get to check the value was programmed in the Token.

Board 2: 0x6a=908.4379, 0x87=908.4212, 0x89=908.4199.

Board 3: 0x6a=908.4453, 0x99=908.4192, 0x98-908.4199.

Board 4: 0x6a=908.4433, 0x98=908.4187, 0x94=908.4209, 0x96=908.4195, 0x95=908.4202

Board 5: 0x6a=908.4459, 0x99=908.4202

SDK 4.4.2 released yesterday but does not appear to have the lockup fix in it according to the release notes.

## 2024-04-10 – I2C debug

The sparkfun XA1110 GPS interface is simply terrible. You have to read out a 255 byte buffer then parse thru it to try to find the NMEA sentence you want and piece it back together. The Arduino example code breaks up the I2C transfers into 32 byte chunks so we can do the same. Seems like you just keep reading data out like it’s a UART until you get an entire frame of 0x0A which will indicate there is no more data? I just love debugging by trial and error! After many hours I finally have the NMEA sentence capturing. I need to start a timer to fetch the GPS value every few seconds. Use the ZAF software Timers as described [here](https://community.silabs.com/s/article/z-wave-700-how-to-use-software-timers?language=en_US) and [here](https://www.silabs.com/documents/public/user-guides/INS14259.pdf). I am getting the NMEA sentence every 1.3s but sometimes there is a space in it which there should not be. I do get buffers of all 0x0a which means we are out of data. The spaces do seem to be coming from the GPS module. Do I need to remove them to match the checksum or not? The spaces are in bad places just random locations in the middle of a number. Is this a bug in the GPS module or my code?

## 2024-04-09 – I2C QWIIC debug

Back to I2C debug via QWIIC connector to a GPS module so we can use GeoLocCC to pull the exact location of the DUT and measure the range. While the ERTT is easy, determining the point where the LED stops blinking is subjective. Often the blinking will stop, the person stops walking, the LED starts again or vice versa. Is that the edge of the RF range or not? You can walk another 50 meters and the blinking can come and go. With GPS coordinates we can also capture the TX power and background RSSI.

The project I am working with is: SwOnOff\_ZG23A\_441. This is a temporary one to get the I2C working. As is typical with all Silabs APIs, the I2CSPM looks simple but has these enormous structures inside with little to no documentation. Thus, I have to spend a lot of time tracing thru the code to figure out what to fill out and how it works. Almost quicker to write my own! I2CSPM\_Transfer sits in the While loop calling I2C\_Transfer waiting for the transfer to complete. This is a blocking call and since each I2C bit is 3us this can add up to upwards of a millisecond for a large transfer. But it is easy to use and we have spare CPU cycles. Wandering thru the code for Multilevel Sensor sample app – wow! So much complexity for such a simple task. Very hard to follow. It sends a sensor report when the button is pressed which is the only event. Unclear how a sensor GET is handled. Somehow some giant structure is initialized with a function pointer I assume. That’s way more complicated than we need for now. Can I just use I2CSPM\_Transfer? The I2CSPM\_Transfer is always returning -1 from all addresses. But is it doing anything on the I2C bus? I’ll have to hook up my scope. SCL and SDA are toggling but it always does the same addr of 0. Because I wasn’t changing it in the code! I can probe all addresses and it finds 0x10. The I2C address of the SparkFun XA1110 GPS breakout board is supposed to be 0x10 (7 bit addr). The GTOP\_NEMA\_over\_I2C\_ software guide explains how to get the NMEA sentence over I2C. I can see the transfers on the PicoScope which probe just the address and return a 0 when the GPS ACKs the addr. The I2CSPM then sends the data byte which is zero for now. It appears you just read the NMEA sentence every 500ms and one of the 2 will be a complete sentence. The other might be partial while it is being filled out. Data of the packet will be 0x0A if the data is invalid. Either the end, the beginning, or the entire buffer will be 0x0A. Merge the packets together while discarding the 0x0A to get a complete sentence.

Enabled DEBUGPRINT. When I press the LEARN button the code goes into Board\_DefaultHandler because the button handler is not defined which takes an assert. I did define PB1\_GPIO=PC05 via Z-Wave Target Boards component. Board.h has settings for PB1 which is the one I am using and defined in the Z-Wave Target Boards. Set a breakpoint in Board\_IsButtonAvailable. Run thru the init, then when pressed the breakpoint fires. Btn=BOARD\_BUTTON\_PB1[0]. This is APP\_BUTTON\_A but I need it to be APP\_BUTTON\_LEARN\_RESET which is BOARD\_BUTTON\_PB2. I filed this [case](https://community.silabs.com/s/question/0D5Vm000007oaXmKAI/how-do-i-use-the-zwave-target-boards-component-for-my-own-pcb) to get an answer. I enabled PB2 as with PC05 and just made PC03 on the header PB1. That got the Learn button to work. Blinks the blue LED while in Learn mode.

## 2024-04-04 – Rev B PCB Debug

### Antenna Smith Charts

David Zima mentioned that some antennas may assume a ground plane but other may not. Thus, I used one of the Rev A boards, cut the trace to the SMA, soldered another SMA to the bottom of the board which then can be screwed into my VNA to generate a Smith chart for each specific antenna below. I used my Agilent N9921A FieldFox spectrum analyzer in VNA mode. Calibrated the open connector (or the connector with the GND plane) to zero out the connector, then screwed the antenna into the N9912A and had it plot the smith chart from 850 to 950MHz. Used the Log Mag chart to find the minimum and set the marker there.

SETUP:

Eleven antennas were tested of various sizes and cost:

The setup used an Agilent N9912A Vector Network Analyzer. Two passes were made with each antenna, one with a GND plane and one without. The GND plane used is a Rev A PCB with an SMA connector soldered to the bottom of the board and the RF trace on the PCB was cut.

The setup shown here has the GND PCB attached. The QuickCal calibration of the N9912A was used to zero out the effect of the connectors and ground plane prior to testing.

|  |  |  |
| --- | --- | --- |
| **Part Number** | **With GND Plane** | **Without GND Plane** |
| **[TI.92.2113](https://www.digikey.com/en/products/detail/taoglas-limited/TI-92-2113/11197416)**  $6.20 – gain 1.21dBi – 198mm  Comments:  Looks like the one shipped with Silabs DevKits  Relatively insensitive to hand nearby unless touching.  VSWR under 1.5 |  |  |
| [**TS.89.4113**](https://www.digikey.com/en/products/detail/taoglas-limited/TS-89-4113/15284364)  $40.56 – gain 3.5dBi  Comments:  Directional antenna with 3.5/2.7dBi gain in 1 direction – Tomahawk shaped.  Not a candidate for Z-Reach but interesting for comparison purposes.  Not sensitive to nearby hand until touching | Basically, the same since the antenna is on a short cable so the GND plane is not close enough to impact the response. | Adding the ground plane made little difference |
| [**ANT-8/9-MMG1-SMA-1**](https://www.digikey.com/en/products/detail/te-connectivity-linx/ANT-8-9-MMG1-SMA-1/16592298)  $5.66 – gain 2dBi – 83mm  Comments:  Cable to a small magnetic mount that will stick to iron.  Insensitive to hand unless almost touching the antenna.  Advantage here is the PCB could be quite small but include a metal disk to stick it to.  VSWR is under 2 (min of 1.5) for much of the range  Wide frequency range so the same antenna is OK for EU and US |  |  |
| [**ANT-916-CW-RH-SMA**](https://www.digikey.com/en/products/detail/te-connectivity-linx/ANT-916-CW-RH-SMA/1962849)  $8.89 – gain -1.3dBi – 51mm  Comments:  Very Small.  Hand closer than 1” to make a significant difference in smith chart  Price is the same for 1 or 10,000. |  |  |
| [**FW.95.B.SMA.M**](https://www.digikey.com/en/products/detail/taoglas-limited/FW-95-B-SMA-M/6362786)  $9.13 – gain 2.7 dBi – 226mm  Comments:  With GND VSWR is 1.1  Sensitive to hand 1” away  Slim and stiff but quite long |  |  |
| [**DELTA12C/X/SMAM/S/S/17**](https://www.digikey.com/en/products/detail/siretta-ltd/DELTA12C-X-SMAM-S-S-17/6096292)  $9.65 – gain 3dBi – 130mm  Comments:  Classic Rubber Ducky  Terrible VSWR barely below 3  Very little sensitivity to a hand but it’s not good in the first place |  |  |
| [**JCG402LR-2**](https://www.digikey.com/en/products/detail/jc-antenna/JCG402LR-2/15814458)  $2.42 – gain 2dBi – 110mm tiltable  Comments:  Relatively insensitive to hand nearby until touching the base. The tip was not sensitive implying there’s no wire up there. Touching close to the base brings the impedance down right to 50 ohms. Thus it should be possible to tune the antenna to the desired band.  Lowest cost antenna.  Better without the ground plane! |  |  |
| [**2600130083**](https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/2600130083/11680415)  $14.88 – gain 0.9dBi – 179mm  Comments:  Covers 868 thru 928.  Highly impacted by a hand nearby without a GND, less so with GND.  VSWR above 2. |  |  |
| **868 antennas:** |  |  |
| [**ANT-868-CW-HW-SMA**](https://www.digikey.com/en/products/detail/te-connectivity-linx/ANT-868-CW-HW-SMA/5592340)  $8.74 – gain 0dBi – 135mm  Comments:  One of the better antennas  VSWR well below 2 for a wide frequency range  Hand has little impact more than 1” away  There is an identical antenna at 916MHz - ANT-916-CW-HW-SMA |  |  |
| [**FW.86.B.SMA.M**](https://www.digikey.com/en/products/detail/taoglas-limited/FW-86-B-SMA-M/6362785)  $9.13 – gain 2.8dBi  240mm  Comments:  Thin fixed (similar to the 915 one above)  Hand has little impact until less than 1” away  Needs the ground plane to have VSWR below 2.  One of the larger antennas |  |  |
| [**FLEXI-SMA-868**](https://www.digikey.com/en/products/detail/rf-solutions/FLEXI-SMA-868/5845738)  $5.12  Comments:  Short Rubber Ducky  One of the cheaper options, it’s also not very good based on the smith charts. Especially with a large GND plane. |  |  |

### Conclusion:

The Smith chart analysis shows that all antennas are NOT the same. Higher prices do not necessarily mean a better antenna. The cheapest antenna has some of the best results. This initial analysis will be followed up with field tests of the three recommended antennas below. After field testing, the chosen antenna will have the PCB/BOM optimized for maximum RF range while also aligning with the cost budget. This may involve making the PCB smaller and/or altering the matching and filtering components.

* Recommended antenna is the same one shipped with the Silabs DevKit

* + [TI.92.2113](https://www.taoglas.com/datasheets/TI.92.2113.pdf) - [$6.29](https://www.digikey.com/en/products/detail/taoglas-limited/TI-92-2113/11197416?s=N4IgTCBcDaICoEkB0BOMSwEZMGYAEIAugL5A) @1k – +1.21dBi – 198mm – 915MHz
  + [**TI.85.2113**](https://www.digikey.com/en/products/detail/taoglas-limited/TI-85-2113/11197203) is the 868 version
  + Reasonable cost, good response, not overly sensitive to nearby objects
  + GND plane didn’t make that much of a difference
  + The PCB size could be made smaller to lower cost
  + BOM would need to be tuned for this antenna
* Smaller low-cost alternative can be a good choice - bendable
  + [JCG402LR-2](https://www.digikey.com/en/products/detail/jc-antenna/JCG402LR-2/15814458) - [$2.54@1K](mailto:$2.54@1K) - +2dBi – 110mm – 915MHz
  + [**JCG402LR-1**](https://www.digikey.com/en/products/detail/jc-antenna/JCG402LR-1/15814463) is the 868 version
  + Without the GND plane has VSWR of 1.1 thus could make PCB smaller
* Alternative option is the antenna with a cable and magnetic mount
  + [ANT-8/9-MMG1-SMA-1](https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6146/ANT-89-MMG1-SMA-1.pdf) – $[5.66@1K](mailto:5.66@1K) - +2dBi – 82mm – both 915&868
  + Could significantly reduce the size of the PCB since the GND plane isn’t needed
    - Lower cost PCB
    - Lower cost enclosure
  + PCB would need specific tuning to achieve better results
  + Ship a metal disk for mounting and to improve RF range?

Where to test the RF Range? Testing at the beach is terrible due to sand and sun. Much better if the range can be down a river with a paved sidewalk alongside and the head end is in the shade. The River Rest Park in Haverhill MA has a 1+ mi view down the Merrimack river though there are trees in the way at various points. Follow the Buttonwoods Riverside Park. Need to bring a bike – walking is too slow to be able to test multiple antennas.

## 2024-04-03 – Rev B PCB debug

Still trying to get the SwOnOff project to work. Tried again but this time I just waited for several minutes between each step to see if it’s just SSv5 being so incredibly slow. To start off, I built just the SwOnOff and only changed the region to US\_LR and it sits in all sorts of odd loops in RAILINT and BURTC. Board #1 does not have a QR code so the code hasn’t finished initialization. It does get to Main().

Using Ozone to manually set GPIOs to check out the LED. Green works, Red and blue are backwards but work. I’ll just change the documentation. PC05 the momentary switch for Learn Mode works. I’ll check out I2C later. Right now I need to connect Basic On/Off with turning on the LEDs so we can run the ERTT.

Nothing in the .SLCP to configure binary switch CC. Looking thru the code in app.c is a p\_switches structure which apparently needs something to connect to. P\_switches comes from cc\_binary\_switch\_get\_config(). This comes from SwitchOnOff\_hw.c which again comes from p\_switch. How is this then threaded to a GPIO? SWITCH\_STATUS\_LED is from APP\_LED\_A which is BOARD\_LED1 which is simply an enum of 0 (in board.h). The board\_getLedPort uses m\_led\_info[led] to get the gpioport. This in turn uses LED\_INFO(LED1). But in this project m\_led\_info is never assigned a value. Create a project using a TBZ and see if it’s assigned somewhere. AHA! The key is the LED\_INFO is a macro that expands LED1 into several other #defines with LED1 as part of the name. Talk about obfuscation! The file radio\_no\_board.h says to Use Configuration Wizard in Context Menu to set the various LED values. But WHERE in the Wizard? Z-Wave Target Board has the configuration of the LEDs and buttons. According to the ReadMe.md, BTN0=toggle state of the switch, BTN1=learn mode, LED0=state of switch LED1=indicator CC. Thus, set BTN1=PC05 and LED1=PA00 (green) and LED2=PA10 (blue) (active low). The LEARN button and the blue LED comes on when pressed! And sending an On/Off turns on/off the green led! So the code is much more tightly integrated into SSv5 now. However, following the code is nearly impossible! This is enough to begin some range testing using the ERTT.

Next step is to get the I2C QWIIC connector checked out and connected to a GPS to enable tracking an end device to get quantitative range measurements. There is a Peripheral Driver I2CSPM in the wizard but it is polled and blocking which is not great. Should be interrupt driven. What does the TBZ multi-level sensor do? RHT sensor is enabled in Platform/Board/Board Control. The Si70xx driver is installed. The I2CSPM is enabled and configured. The Reference Clock frequency is set to 0 (does that matter or does 0 select the HFXO?) – no documentation of course so who knows. There is a short doc on docs.silabs.com but doesn’t explain the configuration in the SLC. The GPIOs are configured. Back to the SwOnOff project – install the I2CSPM. Selected Fast Mode (400K) and assigned the SCL and SDA pins. The project compiles and the I2C pins show up in the PinTool. The I2C peripheral is initialized in sl\_driver\_init so it should be working. The only interface is via the I2CSPM\_Transfer function in sl\_i2cspm. I did verify the power and gnd connections on the connector so the PCB is a go.

## 2024-04-02 – Rev B PCB debug

Built the OTA bootloader which fortunately just builds using the defaults. Created a SwOnOff project to quickly check out the LED and QWIIC connectors. Once those are checked out, I can order more boards. First step is to get DEBUGPRINT working via a custom board. With the DevKit, the USART is used for debugprint but with a custom board the project fails to build because of “undefined reference to sl\_iostream\_write”. Click on the .SLCP file, search for IO STREAM, select Services/IO Stream which then opens a popup with a variety of choices – use IO Stream:USART. That didn’t work and I cannot get any other USART to configure as SSv5 says Selected Modules are “none” with no other options. So, I deleted the project and started again but this time I enabled Iostream before trying to compile.

This time, when I Installed Z-Wave Debug Print a popup asked for which IO Stream I wanted to install so I selected EUSART. But seems like nothing happened. Closed SSv5 and reopened and debug print is not enabled so something did not save. Closed SSv5 again and opened it but this time I installed IO Stream USART first then enabled debugprint. That seems to have worked. The project compiles. Not exactly sure how that worked and it’s terrible that I have to basically randomly try various things to find one that works. All because the Silabs environment is way too heavily weighted towards their devkits and NOT customer boards!

The SwOnOff project does not boot and gets stuck in BOOTLOADER\_TEST\_FAULT\_HANDLING for some reason? I get different results each time I run. It’s not in a WFI which is where it usually is. Using Ozone I still have no idea what’s going on. The BURTC is running but seems like it’s in some sort of loop.

## 2024-04-01 – Rev B PCB debug

How to build the bootloader for Z-Wave for a custom PCB instead of a devkit? The problem with the devkits is they all use hi-security “B” parts which most users don’t need or want. The mid-security “A” parts are cheaper and firmware written for an A part will run a B part but not the reverse. Thus, it is important to compile the firmware for the A part. Downloading the devkit pre-compiled sample apps gets stuck in a loop presumably waiting for a high security feature. Using the Silabs search button didn’t find any usable articles. However, using google to search for “z-Wave 800 series build Z-Wave bootloader” bring you to: <https://community.silabs.com/s/article/Creating-OTA-bootloader-for-Z-Wave-800-series>. This KB is a bit light on the topic but at least gives some guidance. But we need an OTW version for the serial API. Add “OTW” to the google search and it pops up: <https://community.silabs.com/s/article/z-wave-700-otw-of-controller> .

This last article says to look here for the pre-compiled bootloaders: \demo-applications\protocol\z-wave\Apps\bin\

I filed a “feedback” on the website that the binary files are not in this directory.

I have a blog article in the works on how to build the bootloader. I was able to build the bootloader once I configured the Bootloader UART Driver component. The Pintool shows the Rx/Tx pins correctly.

[UG489](https://cn.silabs.com/documents/public/user-guides/ug489-gecko-bootloader-user-guide-gsdk-4.pdf) Bootloader User Guide is the main reference document but it does not explain the key requirements for Z-Wave. The document does not mention Z-Wave at all.

Next step is to build the SerialAPI in the same method – build it using a devkit and copy the necessary options into the custom board project. Built it and it chooses the standard UART pins by default so the project just builds without needing any configuration. Connect the PCC to it and it works and can join a node!

Board 1 debug: Ohm meter shows zero ohms between D2 pins 2 and 3 which are the 2 USB data lines. Short was under the USB-C connector. Touch soldered that to open the short and now the board shows up as a CP210x driver COM port.

Articles on how to implement [I2C](https://community.silabs.com/s/article/z-wave-700-how-to-implement-i2c-into-the-z-wave-embedded-framework?language=en_US) and [SPI](https://community.silabs.com/s/article/z-wave-700-how-to-implement-spi-into-the-z-wave-embedded-framework?language=en_US).

Next step is to get a SwitchOnOff into board 1 and then see how far can they reach. Need to quickly test out the LED and QWIIC connectors then order 9 more PCBs if they are all OK. Maybe quickly check the antenna tune and have to ctune both boards.

## 2024-03-29 – Rev B PCB assemble and debug

Three boards arrived today and I built 2. Still forgot to order 7.2pF caps but I had 7.5 which are close enough for now. Everything fits and the new refdes line up with the BOM.

The ZG23 soldered well but had a few shorts and both boards had 1 side slightly raised which in turn had several opens. For prototype builds I think removing some of the solder paste of the EP would make the chip not stand up. Maybe also push it down harder to make sure all 4 sides will quickly apply surface tension to hold it down. The footprint is a standard footprint so I assume for volume production the amount of paste would be more consistent and the IC would be applied with more pressure.

Board 2 connects via USB! The CP210x driver loads. Board #1 comes up as an unknown device. Touch soldered the USB connector which looks fine now – had some shorts originally and looked OK before I touched them up again. Device Manager reports the Device Descriptor Request Failed.

For now, I’ll focus on board #2 and come back to board #1 later.

The demo files are stored at: C:\SiliconLabs\SimplicityStudio\v5\offline\com.silabs.sdk.stack.super\_4.4.1\protocol\z-wave\demos. I downloaded the BRD4204 version of the SerialAPI solution but PCC does not recognize it. I can’t debug it either as all I have is the binary which is in some sort of loop and not at the EM1 WFI so the firmware is stuck somewhere.

But this has an A part and there may be other special things on the PCB that don’t match my board (I did install a 32KHz crystal). I need to build a bootloader and a SAPI using the actual part which will take some time to figure out how to configure both the bootloader and SAPI.

The SE firmware is 2.1.2 but GDK 4.4.1 has 2.2.4 so I upgraded the SE. Now it gets to main but immediately ends up in HardFault\_handler. I switched to Ozone and it ends up at the WFI like it’s supposed to. WTF. And now so is SSv5. WTF. And it now has a QR code. Now to enable debugprint. Select it, then wait at least 1 minute. Then install it. Wait another minute or more. Select IO Stream: EUSART(0). Wait another minute. Click on Install. Wait several minutes. Doesn’t seem to do anything. Nothing asking me for which GPIOs to use. Ask it to build an has nothing to do so nothing has changed as far as SSv5 is concerned. Uncomment DEBUGPRINT and attempt to build fails because it needs sl\_iostream\_write. Give up on DEBUGPRINT. I don’t really need it but it is often handy.

## 2024-03-18 – Rev A PCB debug – Rev B update

What are the possible reasons the CP2102 doesn’t connect to a PC? The most likely problem is that D- isn’t connected or is shorted to something as pin 5 is one of the recessed corner pins. First thing to view is to see if reset is being held low during power-on. Then look at the D+/D- pins during power on and see if they toggle. Not much else I can debug as the CP2102 is fully encapsulated – no oscillator or anything else coming out. Configuration is done via USB. Check that pin 13 (WAKEUP\_N) is high which it should be. Should see the TX/RX uart pins toggle as well by the ZG23 sending power on commands. On power up I see the USB pins toggling probably sending a wakeup sequence. I also see the ZG23 sending the wakeup commands on the TX. Reset quickly follows powerup so I think that is fine. I posted this [question](https://community.silabs.com/s/question/0D5Vm000005tCdsKAE/cp2102-unresponsive) to the community page. I searched the community but didn’t find anything similar.

I might as well fix all the issues and release Rev B as I can’t see what else might be wrong. Either the CP2102 connects, or it doesn’t – no way to debug it if it fails.

Rev B issues are in github:

1. The refdes are not ordered relative to their position on the PCB. I forgot to run the reorder after placement so the refdeses are the random order they were placed on the schematic.
2. C11 and C12 are 220pF caps but have a 0603 footprint instead of 0402. The 0402 can be installed in the 0603.  
   Fixed in Schematic, not PCB.
3. The color LED is the wrong pinout in the schematic - the LED needs to be a common anode, not cathode. The footprint has the wrong pinout for the selected LED - QLSP14RGB\_B (7 cents) which is a good LED. Might have to make a custom footprint and symbol to fix it.

* A LED I've used before is: CLV1A-FKB-CJ1M1F1BB7R4S3CT ($0.38) Though it does have 3.2V forward drop which won't work if on battery power where the voltage might be only 2.4V.

1. The infeed track width to be 50 ohms should be .44mm instead of .3. Move L1 to the right just enough to get the flood fill to fill in between L1 and L2. Ideally squeeze a via in there. The traces out of the ZG23 have to start at .2mm (same size as the pad) but should widen immediately to .44mm.
2. Switch from the CP2102N QFN20 to the QFN24 package. The QFN20 has 4 pins that are below and recessed slightly within the package. The QFN20 is a standard pin arrangement so the solder joints can be visually inspected and reworked as needed.
3. The QWIIC connector footprint has thru-holes but the part is SMT.
4. Label the signals on the Tag-Connect
5. Add an optional reset switch (normally not mounted). It's just super handy during debug to be able to reset the chip by pressing a button.

Rev B upgrades the KiCAD database to V8.0.0. This required updating a number of symbols and footprints which are much more standardized in V7 – specifically pin 1 markers are in most footprints.

Somehow pins 29 and 30 are backwards in Rev A? I corrected this in Rev B. TD1 and TCLK.

The U.FL connector for whatever reason has a DRC error in it that the solder mask bridges 2 nets. I made it a little smaller. As a result, there is an exclusion that the footprint doesn’t match the one the library.

The 32KHz crystal also has an exclusion because I edited it again as the silk screen lines are too close to the pad.

QWIIC connector – easier to just modify the one I already have to be SMT than try to import one. Well, turns out in KiCAD V8 there already is a JST connector library and it has the correct footprint!

DRC Report: The 2 symbols were slightly fixed to prevent errors. The extra footprints are the fudicials.

\*\* Drc report for ZRAD.kicad\_pcb \*\*

\*\* Created on 2024-03-18T15:06:45-0400 \*\*

\*\* Found 2 DRC violations \*\*

[lib\_footprint\_mismatch]: Footprint 'Crystal\_SMD\_3215-2Pin\_3.2x1.5mm' does not match copy in library 'Crystal'.

Local override; warning (excluded)

@(207.0000 mm, 114.0000 mm): Footprint Y1

[lib\_footprint\_mismatch]: Footprint 'U.FL\_Hirose\_U.FL-R-SMT-1\_Vertical' does not match copy in library 'Connector\_Coaxial'.

Local override; warning (excluded)

@(203.0000 mm, 97.2000 mm): Footprint J4

\*\* Found 0 unconnected pads \*\*

\*\* Found 2 Footprint errors \*\*

[extra\_footprint]: Extra footprint

Local override; warning (excluded)

@(174.4000 mm, 74.6000 mm): Footprint Fid1

[extra\_footprint]: Extra footprint

Local override; warning (excluded)

@(223.0000 mm, 145.2000 mm): Footprint Fid2

\*\* End of Report \*\*

Release board for fab via OSHPark and ordered a new stencil. Updated BOM with new RefDes.

## 2024-03-15 – Rev A PCB debug

Solder blobs on boards 1 & 2: Cleared them without too much trouble using soldering iron and a solder sucker. The USB connector was the most problematic but seems OK.

Plugging in the Minisimplicity header and powering the board via the WSTK SSv5 is able to detect the part and pull the info from it. Since it’s an A part, it seems none of the precompiled Demos are available so I have to create a SerialAPI and bootloader. Or is simply that this is a custom board so SSv5 doesn’t assume the GPIOs are where they are on a devkit? Naturally the bootloader project won’t compile because there’s no EUSART assigned. This is because all of Silabs projects are very centered on their devkits and if you don’t use one of those then you are 100% on your own with no help at all.

I built a BL and SAPI using the ZGM230 DK2603 board as the target. Flashed the binaries into the board and it powers up and has configured a DSK so it must be doing something. But it sits waiting for the BURTC to sync which it never will because I don’t have the 32KHz oscillator installed. But since I built the project using a devkit board it assumes it is setup. Ugh. Pintool doesn’t show anything in the SAPI project (or the BL). I can’t easily copy that to an empty SAPI project. I then create a blank SAPI project then try to figure out what else needs to be configured. The version via the devkit installs Platform->board->starter kit->BRD2603A which in turn enables a ton of stuff but not in a way I can then copy parts I need in a blank project. I created another SAPI project and this time it compiled! WTF? There are no VCOM or EUSARTs in the .slcp file? How is the UART being wired thru? How would we change it? It is in the pintool! Not sure how. It does appear to be running. I want to see if it’s coming out the UART. PC Controller does not find it thru the WSTK. Config/serial\_api\_config.h uses USART0 PA08 Tx and PA09 Rx.

USB is not working – the PC doesn’t detect a plugin in either orientation. Power is working. +5V VBUS comes in and the LDO makes 3.3V just fine. But nothing shows up in Device Manager on my PC. Using SSv5 Console it doesn’t connect. Not sure what that means. Is Tx/Rx backwards? The CP2102 is a QFN20 but the 4 corner pins are BELOW the package and not visible at the edge. Would be much better to use the QFN24 instead which is slightly larger but all the pins come to the edge of the package and none are recessed. They are the same price on Digikey and we have to room on the board. USB D+ pin is on the corner of the package. I would have thought the CP2102 will show up as a USB device when plugged into a computer without any interaction via the UART. I could wire an FTDI to the resistors to get something to work and check more things out. But that isn’t a good test of the RF range since so many things are hacked.

Board #1 & #2 are recognized via SWD and Commander: EFR32ZG23A020F512GM48

Ordered the missing components from the BOM. Not needed immediately. I got a bunch of antennas in the mail today.

## 2024-03-14 – Rev A PCB build

I built 3 boards. Board #3 has been cut down to roughly the minimum size to compare with the full size boards to measure the RF range difference with a smaller ground plane. Basically answering the question – does the additional cost for PCB real estate significantly improve the RF range? Overall size of the reference design isn’t that much of a concern but cost is an important factor though range is the priority.

I forgot to order 1.3nH inductors and 7.2pF caps. I had 1.2nH and 6.8pF which are close enough. I’ll order the correct ones for the next build. The QWIIC connector I ordered is SMT but the PCB is TH so need to pick a different connector. I didn’t order the 32khz crystals but those are not needed at this point. The QWIIC connectors appear to all be SMT so I’ll change the PCB and keep these components.

The solder paste seemed a bit thick and resulted in several shorts. I ordered the .003” thick mylar stencil. The only other option is a .005” so I am using the thinner one. My solder paste is a little thick as the basement is 55F so maybe warming the solder paste up will help is spread more evenly.

## 2024-03-13 – Antennas

I have a box of various antennas. However, they are not labeled and I don’t have the part numbers nor the specs. Some of them might be 2.4GHz, 868, 915. I really can’t tell. So, I ordered some and will keep the part number with the antenna so we know for sure which one is which. Digikey has 15,000 antennas! Filtering that down to SMA in the 900/868MHz bands with an SMA connector and in stock gets us down to 43. I ordered the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Part Number | Price  @1k | Len mm | Comments |
| [**TI.92.2113**](https://www.digikey.com/en/products/detail/taoglas-limited/TI-92-2113/11197416) | $6.20 | 198 | Looks like the one shipped with Silabs DevKits |
| [**TS.89.4113**](https://www.digikey.com/en/products/detail/taoglas-limited/TS-89-4113/15284364) | $40.56 | 118 | Directional antenna with 3.5/2.7dBi gain in 1 direction |
| [**ANT-8/9-MMG1-SMA-1**](https://www.digikey.com/en/products/detail/te-connectivity-linx/ANT-8-9-MMG1-SMA-1/16592298) | $5.66 | 83 | Cable to a mount |
| [**FW.95.B.SMA.M**](https://www.digikey.com/en/products/detail/taoglas-limited/FW-95-B-SMA-M/6362786) | $9.13 | 226 | Thin fixed |
| [**DELTA12C/X/SMAM/S/S/17**](https://www.digikey.com/en/products/detail/siretta-ltd/DELTA12C-X-SMAM-S-S-17/6096292) | $9.65 | 130 | Classic Rubber Ducky |
| [**JCG402LR-2**](https://www.digikey.com/en/products/detail/jc-antenna/JCG402LR-2/15814458) | $2.42 | 110 | Tiltable – Low cost! |
| [**2600130083**](https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/2600130083/11680415) | $14.88 | 179 | Covers 868 thru 928 |
| 868 antennas: |  |  |  |
| [**ANT-868-CW-HW-SMA**](https://www.digikey.com/en/products/detail/te-connectivity-linx/ANT-868-CW-HW-SMA/5592340) | $8.74 | 135 |  |
| [**FW.86.B.SMA.M**](https://www.digikey.com/en/products/detail/taoglas-limited/FW-86-B-SMA-M/6362785) | $9.13 | 240 | Thin fixed (similar to the 915 one above) |
| [**FLEXI-SMA-868**](https://www.digikey.com/en/products/detail/rf-solutions/FLEXI-SMA-868/5845738) | $5.12 | 73 | Short Rubber Ducky |

## 2024-03-03 – Component selection & ordering

Exported the BOM out of KiCAD and used it to manually create the BOM. Ordered enough components for 10+ boards. I already have 15 ZG23s from previous projects and samples. Entered an Issue on github that a couple of the caps are 0603 but should be 0402. The 0402 will still fit but better to have the proper footprint. I expect to spin the PCB at least once more.

The plan is to take one of the 3 boards coming in now and cut it to a much smaller size and compare the range. No sense in making the board bigger than it has to be but we are looking for maximum range. If the full size board is at least 3 dBm better than definitely keep the full size board.

KiCAD has been upgraded to version 8.0.0 which came out a week ago. I had to upgrade to checkin the ZG23 component as all libraries were upgraded to V8. The ZG23 will now be included in the KiCAD default libraries. My PR [4289](https://gitlab.com/kicad/libraries/kicad-symbols/-/merge_requests/4289) has already been merged in. I assume it’ll be included in the next tagged release.



## 2024-02-23 – PCB release to fab

There are 3 excluded DRC violations:

1. The Y2 Crystal footprint was slightly modified to clear the silkscreen error in the footprint
2. The 2 Fiducial marks are not in the schematic

These “warnings” can be ignored.

The PCB is 120mm square which I am certain is overkill. But we can cut a board to be smaller and see the net impact. Might also make a different proto board with a smaller PCB. The corners have 10mm radiuses to make a clean point for the enclosure to latch onto the PCB and make it easier to separate the boards when built as arrays.

Three PCBs and a stencil ordered. Expected in about a week. These 3 boards are just for initial testing to make sure the design doesn’t have some major problem like an incorrect pinout. One board will be for tuning the antenna(s), one is a proto and I expect to cut one to be much smaller to see if the large ground plane makes a difference. Might also want to make a 2 layer PCB version which is cheaper. I’ll order components on Monday.

## 2024-02-22 – PCB layout

DRC clean but in need of cleanup and more documentation.

## 2024-02-21 – Schematic capture

Schematic capture completed and ERC is clean.

Footprints to be picked out:

* USB-C receptacle – GCT [USB4110](https://www.digikey.com/en/products/detail/gct/USB4110-GF-A/10384547)-GF-A is SMT with 2 small mounting pins less than a buck
* [39MHz](https://abracon.com/Resonators/abm8.pdf) crystal – Crystal:Crystal\_SMD\_3225-4Pin\_3.2x2.5mm
* 32KHz crystal – Crystal:Crystal\_SMD\_3215-2Pin\_3.2x1.5mm - SC32S-7PF20PPM
* RGB LED – LED\_SMD:LED\_WS2812B-2020\_PLCC4\_2.0x2.0mm - [QLSP14RGB\_B](https://www.digikey.com/en/products/detail/quelighting-corp/QLSP14RGB-B/15848703)
  + Might need to make a custom footprint
* The RF RCs are 0402 which are cheap and still OK to hand solder. 0201s are tough to handle.
* Other RCs are 0603 and large caps are 0805 which makes them easy to handle and cheap.

PCB board created and a few components placed. Need to adjust the DRC rules to match [OSH Park](https://docs.oshpark.com/design-tools/kicad/kicad-design-rules/).

## 2024-02-20 – Schematic capture

Connected the USB UART to the same pins as the debug header but thru 100 ohm resistors. I could have put the USB UART to a different set of pins but then the firmware could NOT use the standard SerialAPI and would have to be customized with each release. Didn’t think that was a good idea. The series resistors are to provide some isolation when BOTH USB and the debugger are connected. The main issue is that during production, RailTest has to be downloaded and then commanded via the debug connector to the UART to Tx ON for crystal calibration. But since USB is also connected, it is also driving the UART pins. Simply adding some series resistors will provide sufficient isolation for the short time during production. Even during debug it’s not a big deal. During normal operation the resistors will help with EMI. UART speed is only 115.2K baud so the resistors will have no impact on that. The other option is to use a switch but that would be overkill and just 1 more place for things to fail and increase cost.

USB pins require ESD protection as described here: <https://www.digikey.com/en/articles/why-usb-type-c-circuit-protection-is-vital> but this is overkill IMHO. There are many triple TVS diodes but they do not seem to have a common pinout. Prices for these triple diodes are around 30 cents. Only need USB2 level ESD as we do not need to support USB3 which has tighter restrictions on the capacitance of the TVS diode. This [article](https://www.electronicproducts.com/how-to-protect-usb-type-c-connectors-from-esd-and-overtemperature/) has a good description of requirements. Want to have something with lots of 2nd sources and not be locked into a single part. KiCAD doesn’t have any diodes in 0402, mostly SOD-xxx. The CP2102 recommends SP0503BAHTG which is in the KiCAD library and digikey has 114K of them so it is easier to use this part.

The Tag-Connect debug header VAEM is tied directly to the 3.3V from the LDO. This is normally not a problem but could result in both the WSTK and the LDO driving the 3.3V pins. Since they are both at the same voltage it usually doesn’t matter. The WSTK should normally NOT drive the VAEM pin (slide the switch to USB or battery). The only time you would drive the VAEM pin is if you are debugging a battery powered device and want to measure the power which would be done using the AEM in SSv5. But in that case, the LDO would not be installed. The only question is during programming will the LDO short 3V3 to GND? That would require the USB to also be plugged in or powered via a test point which is acceptable but makes things more difficult. However, a functional test of the USB would require the cable being plugged in so maybe it is OK.

There is no QWIIC connector symbol or footprint in KiCad. I copied the ones from the sparkfun repo but I didn’t want to import the entire thing so I created project specific libraries which will also hold some other project specific stuff like the Z-Wave logo and things.

## 2024-02-16 – Project Start

Wrote the datasheet. Started Schematic capture and Theory of Operation (ToO) development.

Lots of LDOs to choose from. Diodes Inc [**AP2125K-3.3TRG1**](https://www.digikey.com/en/products/detail/diodes-incorporated/AP2125K-3-3TRG1/4470777) is 10 cents and will be the initial LDO to use (Digikey has 45K in stock). The TI [**TPS7A0333PDBVR**](https://www.digikey.com/en/products/detail/texas-instruments/TPS7A0333PDBVR/12165108) LDO I’ve used before is 33 cents but does have better Iq, temperature range and Vdropmax but the Diodes Inc is fine.

For the USB-C interface, I used this example: <https://hackaday.com/2023/08/07/all-about-usb-c-example-circuits/>. The classic question is what to connect the shield of the connector to? The Shield is the outside layer of the cable thus it can act as an unintentional antenna and then fail FCC. The example wires it to GND but that is likely to radiate more than leaving it unconnected. Nice discussion [here](https://electrical.codidact.com/posts/279876#:~:text=%E2%88%920-,Connect%20the%20shield%20directly%20to%20ground%20plane,to%20the%20PCB%20ground%20plane.) which mostly recommends connecting SHIELD to GND. There are 2 purposes for this connection – EMI and ESD. For ESD it makes sense to have a solid connection to have the entire board remain at the same potential which might shoot up thousands of volts in an ESD event. But I am more concerned about EMI which in an abundance of caution I put a small ferrite bead leading to the shield to reduce any EMI. Can replace this with a zero ohm resistor later, or leave it open or insert a capacitor or even just short the pads out. I’m not too worried about ESD as ZRAD is pretty much plugged in once and left there for the duration.

## 2024-02-07 – Create Repository

Created the Repo, this file and started the hardware design in KiCAD.