C099-F9P Setup as Base and Rover

Table of Contents

W. S. Herrick, 2022

Introduction

This document details how to build a Base/Rover Real Time Kinetic (RTK) Global Navigation Satellite Service (GNSS) survey pair for under $1100. The Rover is capable of geolocating points, relative to the Base, to centimeter accuracy in near real time. Ublox ™ Zed-F9P L1L2 GNSS chip sets

deliver RTCM (Real Time Correction Measures) from the Base to the Rover. Once the stationary Base has an accurate fix on its location, it can fix the position of the mobile Rover quickly. In this rural case, if the Base location is precise and accurate, post-fix data corrections are usually not needed. GNSS post-fixing logged raw data is also available to refine the Base location. The use of 1W 915MHz telemetry radios allows 500m-40km range between the Base and Rover (>10km loses a bit of accuracy). Commercial Base/Rover gear is expensive, this DIY pair costs about $1100.

The software configuration of the Base and Rover's C099-F9P boards is likely the most challenging aspect of this effort. Some familiarity with GNSS RTK terminology is needed to tailor the devices to a given task, like logging raw data to post-fix a location, assigning a Base location, or Surveying one in.

Goals

* Self Locating or Fixed Base sending RTCM data to Rover, accurate to <3 cm.
* SiK 915 MHz radio telemetry link, Base to Rover.
* Rover running Acenter and/or Qfield on Android or U-Center on Windows via Bluetooth as a UI and logger.
* 1 second epoch or less, all bands and birds.
* GNSS RINEX and UBX data streams can be recorded from the Base or Rover with u-center (Windows) or ACenter (Android). Convert logged UBX to RINEX with RTKConv, Net_Diff, Teqc, etc. GNSS services from many providers can handle RINEX for post fixed accuracy, some can handle UBX. See Appendix: Post-Fixing Location Accuracy.
* Other GPS applications to support the C099-F9P. The Android app, Qfield GIS (see Appendix) syncs with the PC software QGIS, for example.

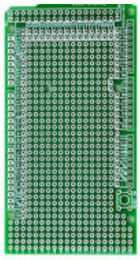
This document details how to construct a rugged field ready Base and Rover survey pair. While assembling the components isn't hard, this project does demand some basic skills with a drill, a saw to cut 1/4" plexiglass, thread tapping (optional), simple hand wiring and soldering. If you use a different enclosure or power supply for the circuitry, the sentence above won't necessarily apply. This design is a bit heavier than it might be, the Aluminum Conduit Bodies weigh a bit more than PVC. A PVC conduit body might also accommodate the C099-F9P & RFD900X antennas internally. As well, the Rover can easily adapt to a standard Surveyor's pole, the monopod used here was just convenient.

Hardware

* Windows Tablet/Laptop with USB, or Linux with Wine & USB, (eg: PineBook Pro w/ Twister OS 2.01) see Appendix: Linux Based Tools).
* Android tablet/phone with Bluetooth to run "Acenter" or Lefebure & optionally, Qfield GIS.
* Two Ublox ™ C099-F9P board kits, includes antennas.
* Two RFD900X SiK 1W telemetry radios with antennas and cables, often sold as a bundle.
* USB UART cable (USB to TTY serial TxRx), 3.3V capable (see RFD900X bundle purchase above).
* 4D MagLite™ flashlight & batteries.

* 6” x 8” 1/4” plexiglass cut to make a clear covers for the conduit bodies.
* Three 1.5” PVC NPT Male plugs, available in the plumbing section of most hardware stores.
* One 1.5” PVC short male threaded coupling (a “close nipple” in plumbing parlance).
* Two 7” metal cutting discs, used as fenders to protect the ground planes and antennas.
* Optional: sheet metal (any conductor) for larger (6”) circular ground planes.
* Tripod and Monopole, with matching lock nuts for attaching to Conduit Bodies.

Power Supply and Radio Mount Protoboard Parts



(See **Protoboard Fit to C099-F9P** and photos below)
Search the web or a local electronics store and buy:
* 2 x Prototype PCB for Arduino Mega 2560 R3 Shield Board DIY
* 4 x LM2596S DC-DC 3A Buck Adjustable Step-down Power Supply Converter

* 2 x 0.28in Digital LED Display Module DC2.5-30V Voltmeter
* 5 x Pairs Male + Female 40 Pin 2.54mm Single Row Straight Pin Header Strip - USA
* 14 x 3mm x 25mm flat head machine screw with nuts for radio and voltmeter. Must fit the 3 RFD900X mounting holes. Cut lengths of plastic tubing to make standoffs and washers for the radio.

* 4 washers for the 3mm machine screws (above).
*6V 12Ah UPS-Replacement-Battery $22

Order two C099-F9P from Digi-Key @ $249/per.

www.digikey.com/Ublox/C099-F9P-2/9817935 (Still in stock 04/06/2024)

Order RFDesign 900x bundle: two radios, antennas, cables etc. $240

https://www.ebay.com/itm/RFD900X-Telemetry-Bundle/353289271954

Weatherproof enclosures: Hub & 1.5” Conduit Body type T, qty two. $35

This set came without screws, you might do better:
Crouse-Hinds T59 1-1/2" Condulet Conduit Body, Box of 2, New | eBay
Myers FH-5 Raintight Flanged Hub, 1-1/2", New | eBay

The example here uses an Aluminum “1.5 inch Conduit Body Tee” with a 1/4” plexiglass cover as a rugged case to enclose all the electronics. The conduit body fixtures are common components used in house wiring, available at many hardware stores. The 1.5” bodies are spacious and weatherproof, anything smaller will be hard to fit. The “flanged hub” offers a flat mounting plane for the antenna. The clear lid makes the LEDs and voltmeter visible. The lid has drilled holes for the C099-F9P Odin-W2 antenna and the two RFD900X antennas.

Tripod & Monopod for Base and Rover stations $100

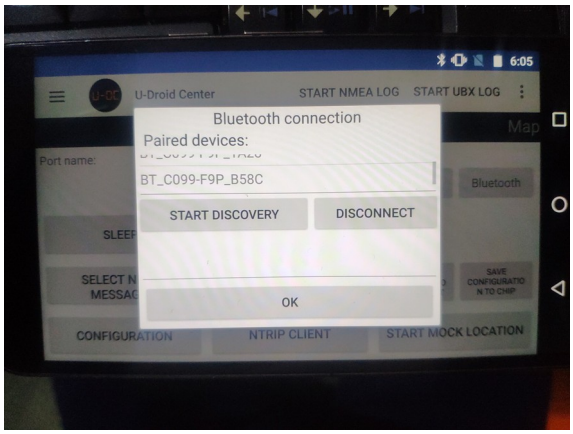
Find a tripod and survey pole/monopod you like. $150 ought to cover their cost (I already had a surveyor’s tripod and camera monopod). Get a lock nut for each to

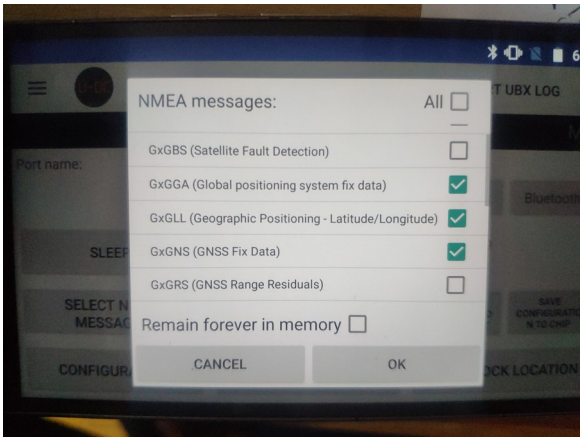
attach the conduit body. Low profile nuts are best. A full height 5/8” nut will need halving and insulation to not short out the C099-F9P mounted inside the Conduit Body.

Software: Windows, Linux and Android Are All Supported, More or Less

Download Ublox “u-center”, “s-center” and Ardupilot’s “Mission Planner”

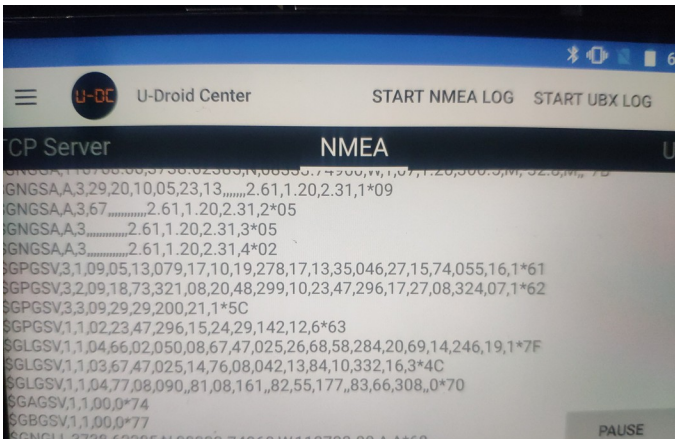
https://www.Ublox.com/en/product/u-center
u-center User guide – u-center_Userguide_(UBX-13005250).pdf
s-center user guide - s-center_UserGuide_(UBX-16012261)_C1-Public.pdf
s-center 5.2.1.exe





Purchase and Install ACenter, the U-Droid Center app for Android:

NB: You can skip Android, U-Droid ACenter and Qfield entirely if you want, & just use Ublox u-center. U-Droid is limited, but it offers UBX logging and “Mock Locations” on Android 7 & 11. Qfield is an Android app. Ublox u-center is a far better tool, but it requires Windows or Linux. U-Droid ACenter is an app from Google Play Store that offers some u-center like logging, reporting, setup functions and a TCP server bridged to the C099-F9P, all on a cell phone. The U-Droid c. $12 app is bound to the Play Store account and can be loaded on multiple devices. There are other apps that work for mock locations and QField (e.g.: Lefebure), but only U-Droid ACenter will log UBX from Android.



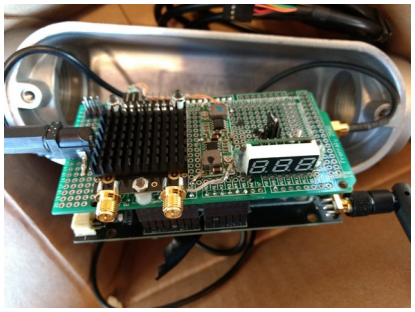
Narrative

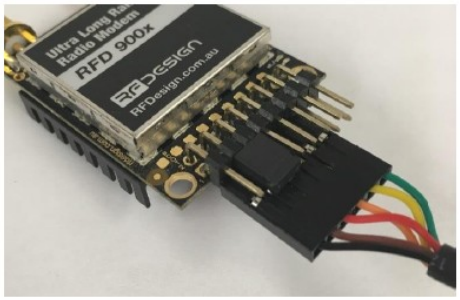
Be aware that the C099-F9P receiver is observing radio emissions from a constellation of satellites in motion. Each satellite may offer a different flavor of GNSS signal on different frequencies and move in or out of view over time. Location values from a wider spanned constellation of satellites are more

accurate than a clump (reported as PDOP). You may record the satellite observations at many intervals (the "epoch" rate), typically between 5x/ second to once every 10. The C099-F9P can manage an 80 Hz epoch for high velocity travel. Foot travel for an RTK pair only requires 1 Hz. The epoch rate impacts the telemetry, as more frequent observations makes more data to be transmitted over the radios. Radio range will diminish with higher data rates. As well, for long stationary or low velocity Rovers, frequent intervals may not improve the accuracy of your measures. The takeaway here is that clear skies to the south with few Line of Sight obstructions from base to satellite, one to ten second long epochs, and picking a time when there are a lot of visible widely spaced satellites makes for faster and more accurate fixes. Obstructions or radio frequency reflections (aka "multi-path" issues) are trouble.

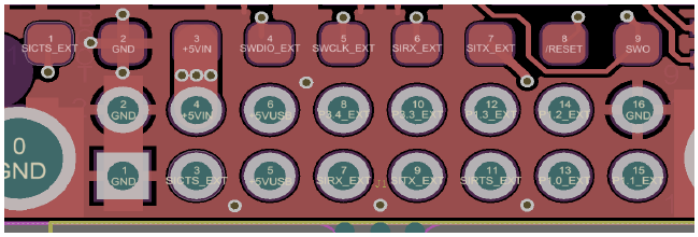
Assembling the hardware is pretty straightforward. Once assembled, the Base and Rover need to be configured and that is not so straightforward. Configuring the C099-F9P Base and Rover pair takes some knowledge of the ZED-F9P chip set's many features and options. Setting the Base for a known monument needs a different configuration than setting one to self locate with a Survey In, for example. Assigning Fixed Positions or Survey In values require attention to the scale and formats. Expect to Read The Finest Manuals as indicated below. Some familiarity with the standard GNSS information models like NMEA, RTCM or UBX will make much of the configuration immediately sensible. The full details are in the manuals listed in the Appendix: **Ublox C099-F9P Resources and Links** below.

Introduction to the Hardware and Power Supply





RFD900X pin outs

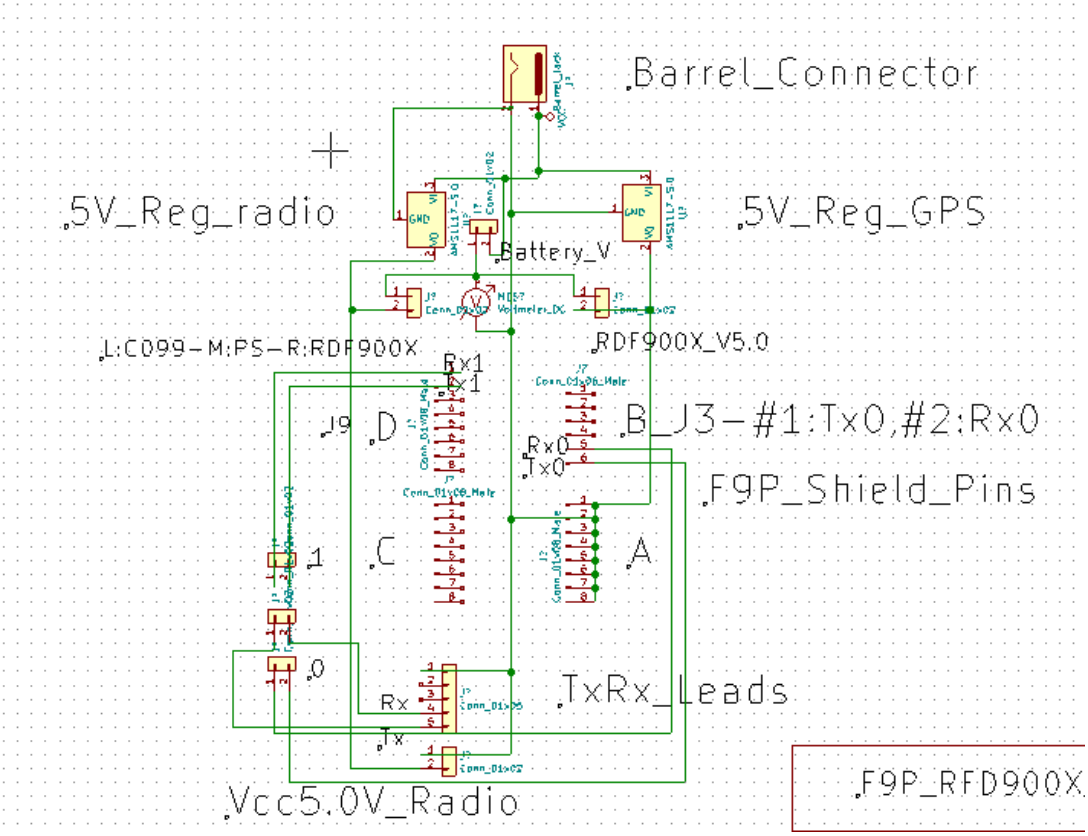


Pin #	Name	Direction	Description	Max Voltage
1	GND	-	Ground	0V
2	GND	-	Ground	0V
3	CTS	Either	Clear to send	3.3V
4	Vcc	-	Power supply	5V
5	Vusb	-	Power supply from USB	5V
6	Vusb	-	Power supply from USB	5V
7	RX	Input	UART Data In	3.3V
8	GPIO5/P3.4	Either	Digital I/O	3.3V
9	TX	Output	UART Data Out	3.3V
10	GPIO4/P3.3	Either	Digital I/O	3.3V
11	RTS	Either	Request to send	3.3V
12	GPIO3/P1.3	Either	Digital I/O	3.3V
13	GPIO0/P1.0	Either	Digital I/O	3.3V
14	GPIO2/P1.2	Either	Digital I/O	3.3V
15	GPIO1/P1.1	Either	Digital I/O, PPM I/O	3.3V
16	GND	-	Ground	0V

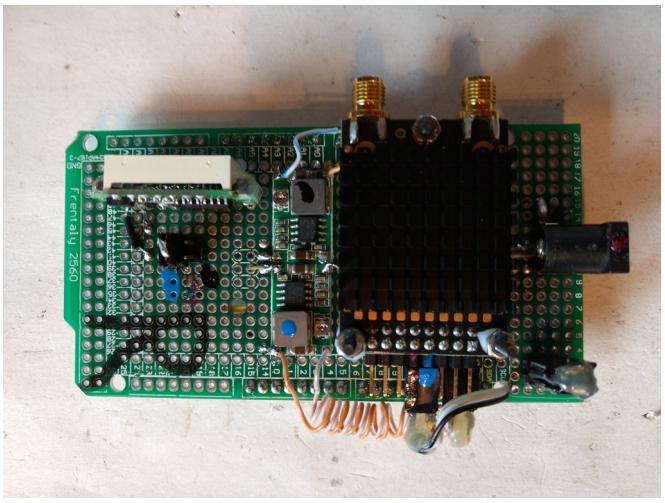
Protoboard Vreg:F9P:RFD900X Shield Schematic

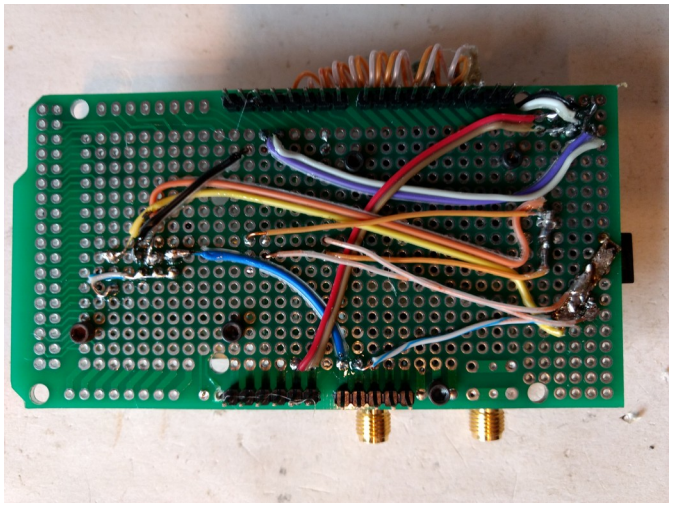
Building the Power Supply and Radio Mount Board

A soldering iron, flux, solder, a small twist drill, a couple of feet of small gauge wire and the listed parts listed are needed to construct the Power Supply and Radio Mounting boards. You will need two identical boards, though they may be jumpered differently. You can make the jumpers from a pair of female headers if need be.

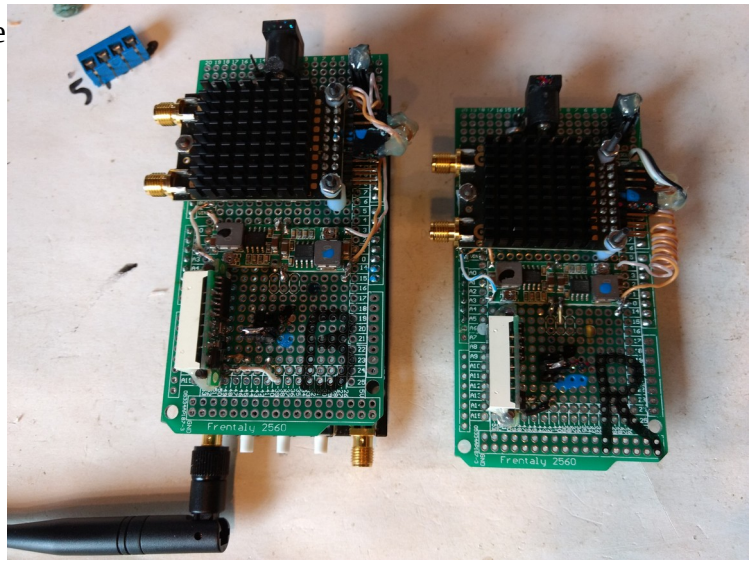


See the photographs below. Shown in the left image, there are two power supplies driven by the 6-15v DC barrel connector (Right hand end, mid board), in the center of board, with blue and black paint dots, the power supplies are pre-fabricated boards with 4 leads, +- ,in and out. The RFD900x radio mounts between the barrel connector and the power supplies. Leftmost is the voltmeter display and the 3x2 jumper pins that select which power supply to display. The UART1|2 jumper is rightmost, bottom.





The right hand image shows the hand wired underside. A printed circuit would be better.
Use a “Mega2560” prototyping board (a protoboard) to assemble the power supplies and mount the RFD900X. Wire the components as shown the schematic above. NB: The barrel connector ground lead is under the connector (see photo above, right)- solder it through the protoboard holes from below and leave a good solder pad on the bottom of the board to wire to. The barrel connector as shown here is centered at the edge, it is better to shift it down 0.1” to 0.2” towards the jumper block, closer to the C099-F9P USB port. The centered location shown here makes a tight fit in the conduit body.



Drill the protoboard to fit the machine screw standoffs for the RFD900X. Bolt it to the protoboard using plastic tubing standoffs below the radio circuit board (between the protoboard and radio board) and above the radio board on the two bolts near the radio’s bottom pinouts to prevent any shorts between the pins and the bolts. Also drill out two holes to mount a pair of machine bolts that hold up the voltmeter display as shown in the photographs above.

Set the default voltmeter jumper to report the RFD900X voltage (it is the most touchy about high or low V). Move the jumpers to report the adjustable voltage regulators to set them up or test them. Restore the default when done. The procedure is detailed below, see **Initial Setup, Bench Testing**.

Plexiglass Cover Plates, Assembling the Base and Rover Components

Use the 1.5” Conduit Body as a template and trace the outline of the open face onto the 1/4” plexiglass. Do both Conduit Bodies. Use a fine toothed saw with a narrow blade to cut the outlined sheet. A bandsaw, saber saw, hacksaw, fret saw or the like will do a clean job. If the plexiglass has a protective paper sheet layer, remove it after cutting to site the location of the holes needed. Place the cut cover onto the Conduit Body and mark the plexiglass for the Conduit Body screws. Gently drill the holes, slightly oversize. Aggressive drilling or dull drill bits can crack the plexiglass.

Assemble the C099-F9P and protoboard power supply as shown in the pictures above. Temporarily feed the antenna cable and USB cable through the 1.5” ports in the Conduit Body, using the right hand port on the Base and the center port on the Rover, and temporarily affix the cables to the C099-F9P. Do not yet attach the ODIN-W2 Bluetooth antenna or the RFD900X antennas (you may want to add the ODIN-W2 antenna briefly to better gauge the position of the hole needed in the plexiglass). With care, gently fit the boards into the Conduit Body without straining the cables so that the RDF900X antenna connectors and voltmeter are facing out towards the plexiglass. Fit the cut plexiglass cover over the Conduit Body opening and align the drilled holes, fit the screws, and loosely attach the cover. Mark the exact position of both RDF900X antenna connectors. Mark the location for the ODIN-W2 antenna hole. Remove the screws and free the plexiglass lid. Measure your antennas width at their base and drill 1/16” over that width. Gently drill the plexiglass for all three antennas.

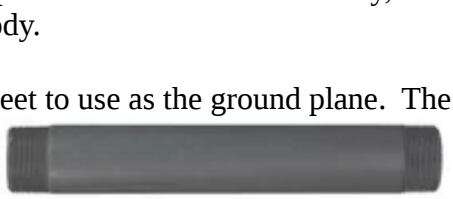
Attach all the antennas and gently fit the cover to the Conduit Body. If needed, oversize the holes more until the antennas and lid fit without strain. Duplicate the above for the second, Rover, Conduit Body. When done, remove the covers, and gently extract the C099-F9P and protoboard from the Conduit Bodies. Remove all cables so only the empty Conduit Bodies remain. The Rover uses a 4D (or larger) Maglite flashlight body attached to the conduit body as a portable power supply. The Base uses a free standing 6V 12Ah UPS Replacement battery.

Next drill the mounting holes in the Conduit Bodies to attach the tripod (Base) and monopod/survey pole (Rover). These holes are specific to the tripod and monopod, so check what fits your gear and adapt as needed. Carefully center the holes to lie directly under the antenna's center. Drill the Base station's hole centered directly under the Conduit Body's middle 1.5" tap. The Maglite 4D flashlight's bottom cap is drilled dead center.

In this example, the Base's Conduit Body needs a 5/8" NC (11 TPI) tapped hole (or over sized hole) and a nut to attach to the surveyor's tripod. A 3/8" NC (16 TPI) tapped thread (or over sized hole) and a nut in the base of the 4D Maglite flashlight will mount the Rover Conduit Body and power supply to its monopod. The 5/8" nut may need to be cut to half height and insulated to not contact the underside of the C099-F9P board (a well stocked hardware store may sell low profile nuts as well).

Screw one of the 12" long 1.5" PVC pipe sections into the center port of the Base Conduit Body, screw the other 12" PVC pipe into the end port on the Rover Conduit Body.

Use the C099-F9P kit's ground plane, or cut a 6" circular metal sheet to use as the ground plane. The 6" diameter is arguably better than the factory supplied 4" ground plane. This example uses copper, but any circle of conductive metal will serve. Use the C099-F9P kit's antenna as a template to mark the antenna's mounting holes needed to center and bolt the antenna to the ground plane, fender disk and "flanged end" plate. Size the drill to the C099-F9P's antenna's pair of holes. Carefully center the antenna on the ground plane and mark the drill hole locations. Drill the ground plane holes.



Next center one 7" metal cutting ceramic "fender" disc under the ground plane and sight through the just drilled holes. Punch through the fender disc with a nail or such to make the holes in the fender disk. Tightly stretch a double layer of good electricians tape around the rim of the fender plate to make the fender rim and edge less abrasive. Now center the ground plane over the 1.5" hole in the "flanged end" plate and as above, sight through the ground plane holes to mark the flange plate's holes. Drill the "flanged end" plate. Verify that all the holes line up.

Place the C099-F9P antenna cable coil onto the 12" pipe and screw the "flanged end" plate to the free end of the 1.5" PCV pipe. Lay the fender disk, ground plane, and antenna on top and gently bolt the set together through all the holes drilled and punched as detailed above. Free enough antenna cable from the coil to thread the connector end through the Conduit Body to the C099-F9P

board. Free enough at the antenna end to relieve any cable strain to the ground plane, fender disc and "flanged end" plate. This completes the antenna, ground plane, fender, "flanged end" plate and 1.5" x 12" pipe construction.

Attach the 4D Maglite Rover Power Supply

NB: the hardware draws enough power to drain batteries fairly quickly. Rechargeable 1.5v D cells cost about $10 per ($40 total). Search the web for "Lithium Ion Rechargeable Battery 1.5V D Cell".

Remove the following from 4D Maglite flashlight: the front/top outer ring that holds the the clear lamp cover, the reflector (the reflector and clear cover can get tossed), the lamp bulb collar, and the bulb lamp. The bulb will be drilled and soldered to wire the power supply.

Break out the glass part of the lamp bulb. Dismantle the bulb carefully and break away the glass and filament without deforming the lamp base. Carefully drill out the guts of the lamp base. Thread a power lead wire through the drilled hole from above so the wire proceeds up to the protoboard's barrel connector. The wire leads should be a flexible woven core wire, about 6" long. Solder the power lead to the center of the bulb base. Leave a healthy bead of solder at the end of the wire on the bottom of the bulbs metal base to assure good connection with the Maglite's batteries. Solder the ground lead to the inside face or edge of the drilled out lamp base so it doesn't impede the collar fitting the lamp base to the Maglite. Insert the modified lamp base, wires up to meet the protoboard's barrel connector, into the Maglite and screw down the collar. Use a voltmeter to verify that 6V DC is seen on the power lead when the Maglite has batteries and is switched on. The Maglite base cap where the batteries go in to the Maglite needs to be tight, but still free enough to unscrew readily for fresh batteries.

The 1.5" threaded pipe section that attaches the Maglite front ring to the conduit body needs to run out of thread just inside the Maglite front ring-the unthreaded (fatter) part is what holds the ring to the Conduit Body. Trim the threaded section that extends beyond the Maglite front ring and threads into the Conduit Body to about 3/4" of thread. Trim the inner unthreaded section to where it does not interfere with reassembling the Maglite but leaves enough of the unthreaded section to hold the Maglite's front ring tightly to the Conduit Body. This will likely take some artful trimming. If the front ring and threaded coupling are not a tight reliable static fit, epoxy glue the Maglite front ring's interior junction where the pipe coupling and ring meet. Be tidy, stray epoxy can clot the fine machine threads on the Maglite or the 1.5" NPT threads on the pipe.

Screw the short 1.5" threaded coupling through the Maglite's front ring, the flashlight piece that held the plastic lamp cover plate onto the Maglite. Screw the coupling through the Maglite front ring far enough that the 1.5" coupling can screw at least ¾ " into the Rover Conduit Body's bottom 1.5" port. As described above, you may need to trim the remaining section of the coupling that extends back into the Maglite if it interferes with rethreading the Maglite body back onto the front ring.

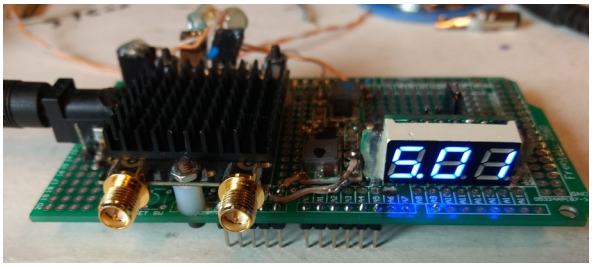
The finished effort should provide the Maglite front ring with enough 1.5" coupling to screw into the conduit body and with no interior coupling extending into the flashlight body far enough to impede screwing the ring back to the flashlight body, and with a tight/glued connection between the coupling to the ring.

Drill a 7/16” hole into two of the 1.5” PCV Male plugs to admit the antenna and USB cables. Screw the plugs into the conduit bodies. Fit the Base plug into the right hand port, and the Rover plug into the center port.
Assemble the monopod, tripod, conduit bodies, 12” pipes, flanged end plates, 7” fenders, ground plane and antennas. Tighten the fittings and bolts well, especially the tripod to conduit body bolt, and the monopod to Maglite base bolt. This finishes the construction of the Base and Rover.

Hardware Construction Completed. Next: Initial Setup, Bench Testing

Set up a bench layout to connect and exercise the naked C099-F9P boards before trying to install them in the working enclosures. The C099-F9P board runs on 5-12V DC. **The RFD900X will only run on a stable 5.0V DC. <u>Do Not provide >5.2V DC to the RFD900X.</u>** Both need at least 1A each to run. Attach the antennas to both the RFD900X and the C099-F9P <u>before</u> powering them up. You can use USB power for the first, u-center/laptop setup step on the C099-F9P. The requisite leads that connect the two Tx/Rx signals both run at 3.3v, and do not need any voltage leveling to safely interconnect these two devices.

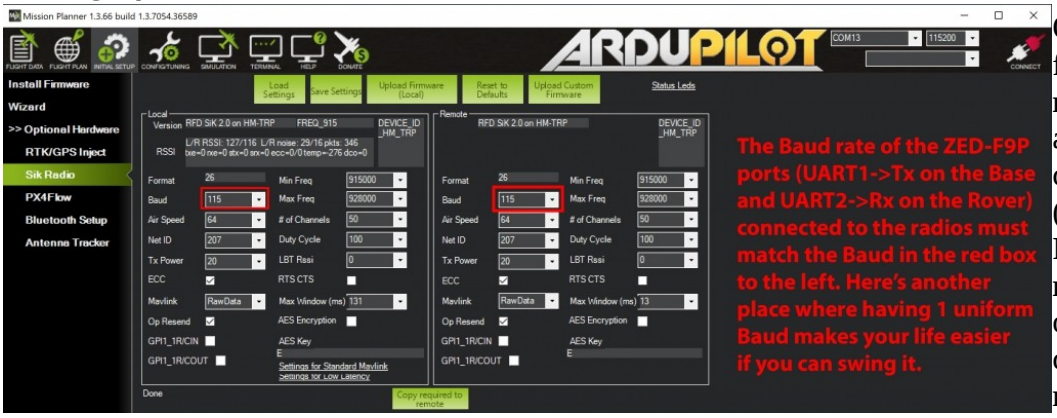
With the C099-F9P and the power leads to the RFD900X disconnected, plug a 6-12V DC power source into the barrel connector to power the voltage regulators. Test the wiring and adjust both voltage regulators to 5.00V DC. Power down when done.



Brownouts can modify flash and RAM on the C099-F9P requiring you to reconfigure the board.

On the Base station, set the jumpers to select which C099-F9P UART drives the SiK RFD900X radio telemetry RTCM stream to the Rover’s radio. Next, set the jumper on the Rover C099-F9P to choose which UART drives the Rover’s radio. **NB: the protoboard UART jumper settings must conform to the F9P message output ports as setup in u-center. The port used here is UART1. <u>NB: the C099-F9P’s OE4 jumper must also be set to enable the Zed-F9P UART1</u>** (see “Instructions to set C099-F9P’s ODIN-W2 Mbed to start Bluetooth on boot”, below).

Setting up the RFD900X radios



Once all checks out, first attach the Base radio’s power lead and the USB UART cable to the laptop (see RFD900X Pinouts, above), and run Mission Planner on the laptop to configure the Base radio.

The RFD900X radios are easily set up with “Mission Planner”. Setup examples are available: https://ardupilot.org/copter/docs/common-configuring-a-telemetry-radio-using-mission-planner.html

Run Mission Planner to Setup SiK RFD900X Radios.

* Connect FTDI USB UART cable to Base RFD900X
* Power up C099 board, watch Voltmeter, make sure radio has stable 5.0v
* Run Mission Planner on laptop connected to RDF900X w/ FTDI UART cable
and select the "**Config/Additional Hardware/SiK**" option.
* Set the Baud to 115k, port to whatever port the FTDI UART/USB cable is assigned (EG: com6).
Click "Load settings" and wait a moment while the RFD900X is polled for the current configuration then apply the following values:

Air 250k	**MinFreq 915**
NetID 25	**MaxFreq 928**
Tx 30	**50**
ECC [check]	**Duty 100**
MAVLink [RAW]	**rssi 0**
OP [check]	**ms 200**

When done, click "Save Settings" and repeat the above for the Rover’s RFD900X. The two radios must have a common baud rate, airspeed, channel count, frequency range, Net ID, Tx Power and ECC values, and if used, a common AES crypto key so each board can decrypt the data stream. When done with both radios, power them down and quit Mission Planner.

Run u-center on Base and Rover, Backup Factory Configuration, update Zed-F9P

Install u-center, s-center (or your preferred serial console) and Mission Planner on the internet connected Windows PC (see **Software**, above), then reboot.

Windows may take a couple of minutes to fetch the FTDI driver for the USB connections. Once the drivers are installed, connect the C099-F9P’s USB cable to the machine running u-center. The windows device manager should now show new USB ports (your port #s may differ than those shown below):
* C099 application board, ODIN-W2 (COM9)
* C099 application board, ZED-F9P (COM8)
* Ublox GNSS Receiver (COM7)
* USB Serial Port (COM6)

Launch u-center and backup the factory configuration in case you need it. Choose the “C099 application board, ZED-F9P ”(Com8) port and set the baud to 480600. Next run “**/Tools/Receiver Configuration/Save**“ and save the factory configuration to a file you can find later if you need it, for example: **E:\ublox\C099_F9P_factory.cfg**

Next update the Zed-F9P firmware, the procedure is well documented on the web and in the u-center documentation. **NB: reflashing the firmware is also a good way to clear out hard to diagnose misconfigurations, reflash liberally.** The ODIN software seems OK as is. Quit u-center when done.

Setup ODIN-W2 Mbed to offer Bluetooth link at boot.

Ublox embed their ODIN-W2 2.4GHz Wifi/Bluetooth hardware on the C099-F9P board. The Odin-W2 is bridged to the ZED-F9P chip over an I2C link at 400 MHz. Hence, u-center addresses IO to "I2C" not "Bluetooth" or "ODIN-W2".

Note Well that in u-center, the C099-F9P ODIN-WiFi/Bluetooth is called the "I2C" port, it is not labeled Bluetooth or WiFi.

U-center keeps configuration settings in a logical stack of layers (Flash, RAM, etc). Those settings determine what information is routed to the available IO ports, like UART1 or I2C. The ODIN-W2 itself is not controlled by u-center, it is setup as detailed next.

UBlox says all C099-F9P sold in the USA are set to ODIN-W2 Mbed by default (vs: UConnect).

This section only applies to ODIN-W2 Mbed devices, not to ODIN-W2 UConnect.
For more information, see the Ublox C099-F9P Mbed manual, section 6. Ublox's s-center offers a command line terminal to the C099-F9P ODIN-W2. However, any terminal program (eg: PuTTY, minicom, etc) should work.

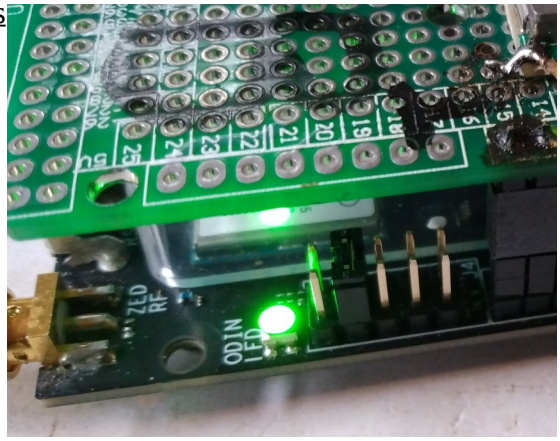
Attach the C099-F9P USB to the PC and launch the communications software. Select the C099-F9P ODIN-W2 port. Set the port's Baud Rate to 460800 8N1 with no flow control (no RTS/CTS).

The user controls the ODIN-W2 Mbed through a command line interface with syntax:
/<function_name>/run <argument 1> <argument 2>
Viz: **/help/run** from the ODIN-W2 Mbed command line for online information.

NB: Some commands require being set in a specific order. e.g.: The **mem_store** command should precede others. It is generally good to pause some seconds after asserting each command.

Instructions to set C099-F9P's ODIN-W2 Mbed to start Bluetooth on boot

On both the Base and Rover, send the following commands in this order to the ODIN-W2:
/mem_store/run bt 1
pause 5 seconds
/bt_visible/run
pause 5 seconds then power cycle the C099-F9.



Place a jumper as shown here on the OE4 pin pair to bridge the Zed-F9P's UART1 to the RFD900x radio Tx/Rx.
This jumper is very poorly documented, it is also essential.

The Mbed ODIN-W2 Bluetooth/I2C service should now wake up on boot, and messages configured with u-connect to be sent out the I2C layer will be proffered. The Android tablet/phone's Bluetooth scan should now offer the C099-F9P for pairing. Make the Bluetooth pair by scanning for Bluetooth devices from the Android tablet/phone and choose the **BT_C099-F9Pxxxxx** device.

Pair the Android Tablet/Phone to the C099-F9P, Enable "Mock Location" and Run ACenter (U-Droid)

N.B: "Mock Location" allows location aware Android applications to use the C099-F9P. Not all Android phones support mock locations. You will need to check your phone for compatibility. Many surveying and tracking applications are available. *You may find other GNSS apps, e.g.:* ***lefebure,*** *that work as well as ACenter as a mock location provider.* Turn on Location Services. *U*se the "Developer Option to set the "Mock Location" (see web) to your chosen GNSS app (Acenter, Lefebure, etc). In ACenter you can pair to either the Base and Rover C099-F9P ODIN-W2 Mbed devices and select the device you want to log, mock or view in ACenter's Main Menu Bluetooth option (scroll through the available devices to find the C099-F9P you want, Base or Rover). Once you select the Bluetooth device in ACenter, after a few seconds, a confirmation popup will report a successful connection or failure. Only one device can connect to the base or rover by bluetooth at a time.

NB: On Android versions above 7, based on personal observation, ACenter v1.8 will timeout and stop logging unless it is the foreground application. To make a proper PPP UBX based survey, launch ACenter, choose the bluetooth connection to the Base station, and start UBX logging. Starting the mock location is optional. Waiting until the Base has a fixed location will give better results for the mocked position, should you want to assign a point in a Qfield layer and or geolocate a photo with Solocator. Using the photo option in ACenter often gets the photo, but kills Acenter and requires restarting the ACenter app (this might be fixed in ACenter v 1.8) . Leave ACenter running in the foreground after any forays to Qfield or such, or the logging will quit after a few minutes.

This completes the ODIN-W2 Mbed Bluetooth and ACenter setup.

Basic Software Setup Completed

This completes the basic software setup: both C099-F9P devices have bench tested u-center via USB connection, both Mbed ODIN-W2 devices have been set to start Bluetooth on boot and both RFD900X SiK radios have been configured and wired. The next section details how to use u-center to set the two C099-F9P devices to act as a Base and Rover.

Configuring "Survey In" or "Fixed" Mode Base , Then Rover With u-center.exe

The u-blox u-center.exe application runs under Windows or Linux using Wine (see Appendix). The C099-F9P is a "Generation 9" u-blox product that is a departure from earlier u-blox designs. Few instructions or methods for older u-blox products will precisely apply to setting up the C099-F9P, though the general mechanism is the same. The C099-F9P uses a new method in u-center.exe for configuration: the Generation 9 Advanced Configuration View. U-center's Generation 9 Advanced Configuration View will read and write values by message class and name, ie: **cfg-uart2-baudrate**.

Configuration values are assigned to memory "layers". In general, use **u-center.exe** to write the configuration to the RAM and Flash layers. Flash is persistent across boots and power cycles, RAM is not. The BBR layer is Battery Backed up RAM, it loses data when the battery fails. Including BBR with a battery as a layer can make resetting the C099-F9P more difficult. However, using a battery will speed up the Rover's convergence time. Include BBR along with RAM and Flash only if you've added a battery to the CO99-F9P board.

When using **u-center.exe** to configure the C099-F9P, as a rule, save the configuration to file before sending it to the C099-F9P. Assert all changes from the factory default in one configuration and save that setup to file before applying the changes. After saving the setup to file, but before applying the settings, always click the red u-center "Reset to Default" button and wait 5 seconds. The icon shows a small gear with a red star. That red button is usually the rightmost icon in the second tier Main Menu bar, though one can move the menu bars, so it may be located elsewhere.

Once reset to default, apply the settings by clicking the "Send" button (see u-center screenshot slides below, Step 6). If your configuration is not ideal, make another pass by re-loading the setup from file. Remove any offending lines, add any corrections, then save the file again, and after again clicking the "Reset to Default" button and the brief delay, send the repaired configuration to the C099-F9P.

The u-blox provided C099-F9P configuration settings for an RTK Base and Rover survey pair are listed below, with edits to assert baud rate and make the I2C (bluetooth) messaging the same as USB. The Base/Rover pair detailed in this document use SiK radios to link the Base and Rover RTCM data, and that differs from the u-blox CO99-F9P factory default. Use the configuration detailed below for both the Base and the Rover to enable the SiK radio link. NB: you will need a different Base file to acquire NMEA data for a PPP (Precise Point) post-fix measurement, these examples use UBX messages to make the RINEX data set, not NMEA. For PPP measurements UBX RAWX & SFRBX messages are logged and then converted to RINEX and subsequently processed. Udroid Acenter or ublox u-center can log UBX messages. See the Appendix "Further Processing to Post Fix Location Accuracy" for more instructions.

Real Time Correction Messages (RTCM) will not be generated until the Base has a fixed location. RTCM values from the Base only begin to transmit after the Base has either been assigned and acquired a Fixed base location or a Survey In step has completed and the Base location determined within the error limits set in the Survey. See **Section 3.1 RTCM corrections** in the ZED-F9P Integration Manual (link in Appendix below) for a complete list of RTCM messages.

Status LEDs

The C099-F9P board provides three LEDs near the USB port to show the device status. The RTK status LED provides an indication of the state of the ZED-F9P module RTK-STAT pin.
• At start-up the LED is off.
• RTK Float: When a valid stream of RTCM messages is being received and utilized, but no RTK fixed mode has been achieved, the yellow LED flashes.
• RTK fixed: the yellow LED remains lit without flashing.
• The blue time pulse LED will flash at the default 1 Hz rate when the time solution is valid.
NB: If you do not see the illuminated Yellow LED, you are not getting RTCM data to the Rover.

The settings listed below are U-blox published settings to configure the C099-F9P as a Base and Rover edited to conform to the 115200 baud SiK radio link on UART1, setting the I2C messaging the same as the USB, provide UBX_RAWX and SFRBX records to make RINEX files for post processing, and a 5 minute & 3 meter survey in.

Base Setup+115k baud radio+3 minute 4 meter Survey In

F9P Base config

```
# Config changes format version 1.0
# created by u-center version 21.09 at 11:31:12 on Monday, 27 Dec 2021
[del]
[set]
  RAM CFG-UART1INPROT-NMEA 0 # write value 0 to item id 0x10730002 in layer 0
Flash CFG-UART1INPROT-NMEA 0 # write value 0 to item id 0x10730002 in layer 2
  RAM CFG-UART1INPROT-RTCM3X 0 # write value 0 to item id 0x10730004 in layer 0
Flash CFG-UART1INPROT-RTCM3X 0 # write value 0 to item id 0x10730004 in layer 2
  RAM CFG-UART1OUTPROT-UBX 0 # write value 0 to item id 0x10740001 in layer 0
Flash CFG-UART1OUTPROT-UBX 0 # write value 0 to item id 0x10740001 in layer 2
  RAM CFG-UART1OUTPROT-NMEA 0 # write value 0 to item id 0x10740002 in layer 0
Flash CFG-UART1OUTPROT-NMEA 0 # write value 0 to item id 0x10740002 in layer 2
  RAM CFG-UART1OUTPROT-RTCM3X 1 # write value 1 to item id 0x10740004 in layer 0
Flash CFG-UART1OUTPROT-RTCM3X 1 # write value 1 to item id 0x10740004 in layer 2
Flash CFG-UART1INPROT-UBX  0 # write value 0 to item id 0x10730001 in layer 2
  RAM CFG-UART1INPROT-UBX  0 # write value 0 to item id 0x10730001 in layer 0
  RAM CFG-MSGOUT-RTCM_3X_TYPE1005_UART1 0x1  # write value 1  0x1 to item id 0x209102be i
Flash CFG-MSGOUT-RTCM_3X_TYPE1005_UART1 0x1  # write value 1  0x1 to item id 0x209102be i
  RAM CFG-MSGOUT-RTCM_3X_TYPE1074_UART1 0x1  # write value 1  0x1 to item id 0x2091035f i
Flash CFG-MSGOUT-RTCM_3X_TYPE1074_UART1 0x1  # write value 1  0x1 to item id 0x2091035f i
  RAM CFG-MSGOUT-RTCM_3X_TYPE1084_UART1 0x1  # write value 1  0x1 to item id 0x20910364 i
Flash CFG-MSGOUT-RTCM_3X_TYPE1084_UART1 0x1  # write value 1  0x1 to item id 0x20910364 i
  RAM CFG-MSGOUT-RTCM_3X_TYPE1124_UART1 0x1  # write value 1  0x1 to item id 0x2091036e i
Flash CFG-MSGOUT-RTCM_3X_TYPE1124_UART1 0x1  # write value 1  0x1 to item id 0x2091036e i
  RAM CFG-MSGOUT-RTCM_3X_TYPE1230_UART1 0x5  # write value 5  0x5 to item id 0x20910304 i
Flash CFG-MSGOUT-RTCM_3X_TYPE1230_UART1 0x5  # write value 5  0x5 to item id 0x20910304 i
  RAM CFG-MSGOUT-RTCM_3X_TYPE1005_USB 0x1  # write value 1  0x1 to item id 0x209102c0 in
Flash CFG-MSGOUT-RTCM_3X_TYPE1005_USB 0x1  # write value 1  0x1 to item id 0x209102c0 in
  RAM CFG-MSGOUT-RTCM_3X_TYPE1074_USB 0x1  # write value 1  0x1 to item id 0x20910361 in
Flash CFG-MSGOUT-RTCM_3X_TYPE1074_USB 0x1  # write value 1  0x1 to item id 0x20910361 in
  RAM CFG-MSGOUT-RTCM_3X_TYPE1084_USB 0x1  # write value 1  0x1 to item id 0x20910366 in
Flash CFG-MSGOUT-RTCM_3X_TYPE1084_USB 0x1  # write value 1  0x1 to item id 0x20910366 in
  RAM CFG-MSGOUT-RTCM_3X_TYPE1124_USB 0x1  # write value 1  0x1 to item id 0x20910370 in
Flash CFG-MSGOUT-RTCM_3X_TYPE1124_USB 0x1  # write value 1  0x1 to item id 0x20910370 in
  RAM CFG-MSGOUT-RTCM_3X_TYPE1230_USB 0x5  # write value 5  0x5 to item id 0x20910306 in
Flash CFG-MSGOUT-RTCM_3X_TYPE1230_USB 0x5  # write value 5  0x5 to item id 0x20910306 in
  RAM CFG-MSGOUT-RTCM_3X_TYPE1094_UART1 0x1  # write value 1  0x1 to item id 0x20910369 i
Flash CFG-MSGOUT-RTCM_3X_TYPE1094_UART1 0x1  # write value 1  0x1 to item id 0x20910369 i
  RAM CFG-MSGOUT-RTCM_3X_TYPE1094_USB 0x1  # write value 1  0x1 to item id 0x2091036b in
Flash CFG-MSGOUT-RTCM_3X_TYPE1094_USB 0x1  # write value 1  0x1 to item id 0x2091036b in
  RAM CFG-MSGOUT-UBX_NAV_PVT_USB 0x1  # write value 1  0x1 to item id 0x20910009 in layer
Flash CFG-MSGOUT-UBX_NAV_PVT_USB 0x1  # write value 1  0x1 to item id 0x20910009 in layer
  RAM CFG-MSGOUT-UBX_NAV_SVIN_USB 0x1  # write value 1  0x1 to item id 0x2091008b in laye
Flash CFG-MSGOUT-UBX_NAV_SVIN_USB 0x1  # write value 1  0x1 to item id 0x2091008b in laye
  RAM CFG-UART1-BAUDRATE   0x1c200   # write value 115200  0x1c200 to item id 0x40520001 in layer 0
Flash CFG-UART1-BAUDRATE   0x1c200   # write value 115200  0x1c200 to item id 0x40520001 in layer 2
  RAM CFG-TMODE-MODE       1 # write value 1 - SURVEY_IN to item id 0x20030001 in layer 0
Flash CFG-TMODE-MODE       1 # write value 1 - SURVEY_IN to item id 0x20030001 in layer 2
  RAM CFG-TMODE-SVIN_MIN_DUR 0xb4 # write value 180  0xb4 to item id 0x40030010 in layer 0
Flash CFG-TMODE-SVIN_MIN_DUR 0xb4 # write value 180  0xb4 to item id 0x40030010 in layer 2
  RAM CFG-TMODE-SVIN_ACC_LIMIT 0x9c40 # write value 40000  0x9c40 to item id 0x40030011 in layer 0
Flash CFG-TMODE-SVIN_ACC_LIMIT 0x9c40 # write value 40000  0x9c40to item id 0x40030011 in layer 2
```

Configuration group CFG-NAVSPG

U-blox receivers can be tuned to different velocity models to adapt the navigation engine to the expected application. The settings can be changed on the fly without performing a power cycle or reset. CFG-NAVSPG-DYNMODEL is the configuration item used to select the dynamic model.

• Portable: Applications with low acceleration, e.g.: portable devices like an RTK Rover.

• Stationary: The Base. Used in stationary applications, e.g.: computing RTCM data for a Rover.

C099-F9P Base: Survey-in

If the Base station is not placed in a known location, the Base will need to run a “Survey-In”. Survey-in is a procedure that estimates the receiver position by building a weighted mean of all valid 3D position solutions. Place the Base in a location with a clear southern sky view to run a Survey In. Two major parameters are required when configuring:
• A minimum observation time defines the minimum observation time independent of the actual number of fixes used for the position estimate. Values can range from one day for high accuracy requirements to a few minutes for coarse position determination.
• A given PDOP accuracy to define a limit on the spread of positions that contribute to the calculated mean. The circle of error is given in 0.1mm measures, ie: 40,000 is a 4 meter range.

Survey-in ends when both requirements are successfully met.

The receiver begins operation in time mode and can output a base position message if configured. The survey-in status can be queried using the UBX-NAV-SVIN message.

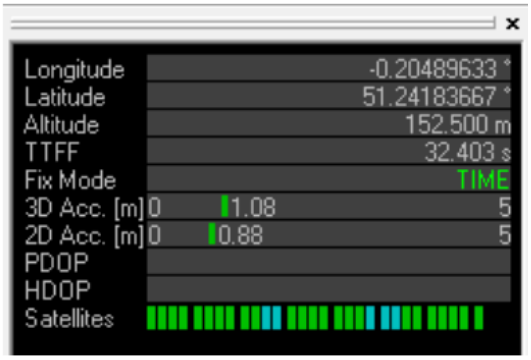
To configure a base station into survey-in mode (CFG-TMODE-MODE=SURVEY_IN), the following items are required (you can adjust the minutes and accuracy)**:**

CFG-TMODE-MODE
Receiver mode: **survey-in**

CFG-TMODE-SVIN_MIN_DUR
Survey-in minimum duration in seconds: **180**

CFG-TMODE-SVIN_ACC_LIMIT
Survey-in position accuracy limit in one tenth millimeters: 4**0000**

The accuracy setting is applied to the PDOP readings, not the 2/3D accuracy.



u-center data view in TIME mode.

You may need to relocate the base antenna or change the required accuracy and/or survey-in time. The survey-in mode requires reasonable settings based on the environment and achievable accuracy in the base location. 40000 (0.1 mm x 40000 = 4 m) for estimated accuracy and survey-in time of 180 seconds is a sensible starting point. In good satellite visibility, the base is unlikely to achieve an accuracy better than 1 m. In occluded or multi-path conditions, the time to achieve a specified accuracy can take longer.

Use the u-center "Packet Console View" to verify message output. Once surveyed in correctly, it will indicate a TIME solution mode in the u-center Data view.

F9P Assign Base to a Fixed Position

The Zed-F9P can fail to enter Fixed mode readily in some cases. The configuration messages must be presented in the right order. Prior settings can interfere and may be resolved by reflashing the configuration and /or the firmware without restoring any configuration. Starting the base in survey-in

then asserting fixed mode after the F9P has begun to self-locate may also work when a direct start in fixed mode fails.

The receiver position coordinates are entered manually. Errors in the base station position translate into rover position errors (use an accurate benchmark for the base). The base station position accuracy must match or exceed the desired rover absolute position accuracy.

To configure Fixed mode (CFG-TMODE-MODE=FIXED), the following items are relevant:

```
Configuration item Description
CFG-TMODE-MODE Receiver mode fixed
CFG-TMODE-POS_TYPE Determines whether the ARP position is given in ECEF or LAT/LON/HEIGHT
CFG-TMODE-ECEF_XECEF X coordinate of the ARP position,cm
CFG-TMODE-ECEF_YECEF Y coordinate of the ARP position,cm
CFG-TMODE-ECEF_Z ECEF Z coordinate of the ARP position,cm
CFG-TMODE-LAT Latitude of the ARP position
CFG-TMODE-LON Longitude of the ARP position
CFG-TMODE-HEIGHT Height of the ARP position
CFG-TMODE-ECEF_X_HP High-precision ECEF X coordinate of the ARP position
CFG-TMODE-ECEF_Y_HP High-precision ECEF Y coordinate of the ARP position
CFG-TMODE-ECEF_Z_HP High-precision ECEF Z coordinate of the ARP position
CFG-TMODE-LAT_HP High-precision latitude of the ARP position
CFG-TMODE-LON_HP High-precision longitude of the ARP position
CFG-TMODE-HEIGHT_HP High-precision height of the ARP position
CFG-TMODE-FIXED_POS_ACC Fixed position 3D accuracy estimate
```

NB: assert the TMODE configuration values in the order presented in the u-blox ZED-F9P Integration Manual, also shown above. After the receiver is configured to fixed mode, select the position format to use: either LLH or ECEF with optional high precision (0.1mm) coordinates added to the standard cm value.

For example, with CFG-TMODE-POS_TYPE=ECEF, the base antenna position can be entered to cm precision using CFG-TMODE-ECEF_X, CFG-TMODE-ECEF_Y, CFGTMODE-ECEF_Z.
For high precision (0.1mm) coordinates only enter the 0.1mm (-99..99) to append to the cm values, they sum to make the exact high precision locus. Use CFG-TMODE-ECEF_X_HP, CFG-TMODE-ECEF_Y_HP, CFG-TMODE-ECEF_Z_HP. The same applies with corresponding coordinates used with CFG-TMODE-POS_TYPE=LLH. Read the ZED-F9P Integration Manual “High Precision” details, eg: for the “CFG-TMODE-ECEF_X” and“ CFG-TMODE-ECEF_X_HP” on how to correctly enter the values.

This confusing tidbit is from the ZED-F9P Integration Manual: The "3D accuracy" estimate in "Fixed Position" and the "Position accuracy limit" in "Survey-in" will affect the rover absolute position accuracy. Note that the availability of the position accuracy does not mitigate the error in the rover position, but only accounts for it when calculating the resulting positioning accuracy.

Rover Setup

Per Ublox Zed-F9P Integration Manual (see Appendix): the C099-F9P Rover with factory default settings will enter RTK float mode as soon as it receives an input stream of RTCM correction messages (the Rover’s RTK LED flashes yellow). Once the rover has resolved the carrier phase ambiguities, it

will enter RTK fixed mode (LED stays yellow). When in this mode, the relative position accuracy between base and rover can be expected to be correct to the cm-level.

Per p.14 of the ZED-F9P Integration Manual (see Appendix):
The RTCM 3x stream must contain the "GLONASS code-phase biases" message (type 1230) or the "Receiver and Antenna Description" message (type 1033) otherwise the GLONASS ambiguities can only be estimated as float, even in RTK fixed mode. If using a ZED-F9P as a base,"GLONASS code-phase biases" message (type 1230) must be enabled to be output. For a ZED-F9P being used as a rover, if no RTCM 1230 message is received or if the reference receiver type reported in RTCM 1033 message is unknown to the ZED-F9P receiver, **the ZED-F9P rover will only be able to estimate GLONASS ambiguities as float**, even in RTK fixed mode. This will result in degraded performance, especially in challenging environments.

F9P Rover config + 115k baud radio + HP Pos ECEF + NMEA Datum

```
# Config changes format version 1.0
# created by u-center version 18.11 at 11:16:51 on Tuesday, 27 Nov 2018
[del]
[set]
  RAM CFG-UART1INPROT-UBX  1   # write value 1   to item id 10730001 in layer 0
Flash CFG-UART1INPROT-UBX  1   # write value 1   to item id 10730001 in layer 2
  RAM CFG-UART1INPROT-NMEA 0   # write value 0   to item id 10730002 in layer 0
Flash CFG-UART1INPROT-NMEA 0   # write value 0   to item id 10730002 in layer 2
  RAM CFG-UART1INPROT-RTCM3X 1   # write value 1   to item id 10730004 in layer 0
Flash CFG-UART1INPROT-RTCM3X 1   # write value 1   to item id 10730004 in layer 2
  RAM CFG-UART1OUTPROT-UBX 1   # write value 1   to item id 10740001 in layer 0
Flash CFG-UART1OUTPROT-UBX 1   # write value 1   to item id 10740001 in layer 2
  RAM CFG-UART1OUTPROT-NMEA 1   # write value 1   to item id 10740002 in layer 0
Flash CFG-UART1OUTPROT-NMEA 1   # write value 1   to item id 10740002 in layer 2
  RAM CFG-UART1OUTPROT-RTCM3X 0   # write value 0   to item id 10740004 in layer 0
Flash CFG-UART1OUTPROT-RTCM3X 0   # write value 0   to item id 10740004 in layer 2
  RAM CFG-USBINPROT-UBX    1   # write value 1   to item id 10770001 in layer 0
Flash CFG-USBINPROT-UBX    1   # write value 1   to item id 10770001 in layer 2
  RAM CFG-USBINPROT-NMEA   1   # write value 1   to item id 10770002 in layer 0
Flash CFG-USBINPROT-NMEA   1   # write value 1   to item id 10770002 in layer 2
  RAM CFG-USBINPROT-RTCM3X 1   # write value 1   to item id 10770004 in layer 0
Flash CFG-USBINPROT-RTCM3X 1   # write value 1   to item id 10770004 in layer 2
  RAM CFG-USBOUTPROT-UBX   1   # write value 1   to item id 10780001 in layer 0
Flash CFG-USBOUTPROT-UBX   1   # write value 1   to item id 10780001 in layer 2
#
  RAM CFG-USBOUTPROT-NMEA  1   # write value 1   to item id 10780002 in layer 2
Flash CFG-USBOUTPROT-NMEA  1   # write value 1   to item id 10780002 in layer 2
  RAM CFG-USBOUTPROT-RTCM3X 0   # write value 0   to item id 10780004 in layer 0
Flash CFG-USBOUTPROT-RTCM3X 0   # write value 0   to item id 10780004 in layer 0
#
  RAM CFG-UART1-BAUDRATE   0x1c200 # write value 115200 0x1c200 to item id 0x40520001 in
Flash CFG-UART1-BAUDRATE   0x1c200   # write value 115200  0x1c200  to item id 0x40520001 in
#Flash CFG-USBOUTPROT-RTCM3X 0   # write value 0   to item id 10780004 in layer 2
##
Flash CFG-MSGOUT-NMEA_ID_DTM_USB 0xa # write value 10  0xa   to item id 0x209100a6 in layer
  RAM CFG-MSGOUT-NMEA_ID_DTM_USB 0xa # write value 10  0xa   to item id 0x209100a6 in layer
Flash CFG-MSGOUT-UBX_NAV_HPPOSECEF_USB 0x1 # write value 10  0xa   to item id 0x2091002e in
  RAM CFG-MSGOUT-UBX_NAV_HPPOSECEF_USB 0x1 # write value 10  0xa   to item id 0x2091002e in
##
#Flash CFG-MSGOUT-NMEA_ID_DTM_I2C 0xa # write value 10  0xa   to item id 0x209100a6 in layer
#Flash CFG-MSGOUT-NMEA_ID_GLL_I2C 0xa # write value 10  0xa   to item id 0x209100c9 in layer
#Flash CFG-MSGOUT-UBX_NAV_HPPOSECEF_I2C 0x1 # write value 10  0xa   to item id 0x2091002e in
#Flash CFG-MSGOUT-UBX_NAV_HPPOSLLH_I2C 0x1  # write value 10  0xa   to item id 0x20910033 in
#Flash CFG-MSGOUT-UBX_NAV_SVINFO_I2C 0x1 # write value 10  0xa   to item id 0x2091000b in la
```

```
#Flash CFG-MSGOUT-UBX_NAV_SVIN_I2C 0x1 # write value 10  0xa   to item id 0x20910088 in laye
#Flash CFG-MSGOUT-UBX_RXM_RAWX_I2C 0x1 # write value 1  0x1    to item id 0x209102a4 in laye
#Flash CFG-MSGOUT-UBX_RXM_RTCM_I2C 0xa # write value 10  0xa   to item id 0x20910268 in laye
#  RAM CFG-MSGOUT-NMEA_ID_DTM_I2C 0xa # write value 10  0xa   to item id 0x209100a6 in layer
#  RAM CFG-MSGOUT-NMEA_ID_GLL_I2C 0xa # write value 10  0xa   to item id 0x209100c9 in layer
#  RAM CFG-MSGOUT-UBX_NAV_HPPOSECEF_I2C 0xa # write value 10  0xa   to item id 0x2091002e in
#  RAM CFG-MSGOUT-UBX_NAV_HPPOSLLH_I2C 0xa # write value 10  0xa   to item id 0x20910033 in
#  RAM CFG-MSGOUT-UBX_NAV_SVINFO_I2C 0xa # write value 10  0xa   to item id 0x2091000b in la
#  RAM CFG-MSGOUT-UBX_NAV_SVIN_I2C 0xa # write value 10  0xa   to item id 0x20910088 in laye
#  RAM CFG-MSGOUT-UBX_RXM_RAWX_I2C 0x1 # write value 1  0x1    to item id 0x209102a4 in laye
#  RAM CFG-MSGOUT-UBX_RXM_RTCM_I2C 0xa # write value 10  0xa   to item id 0x20910268 in laye
```

The image below shows ACenter on an android device reporting the Rover status during an initial test. The Base has a Time (float) fix and is sending RTCM data to the Rover, refining the Rover's position to 7 cm. Once the Base achieves a fixed location, the Rover position relative to the Base can be accurate to less than three centimeters.

Screenshots and u-center Examples

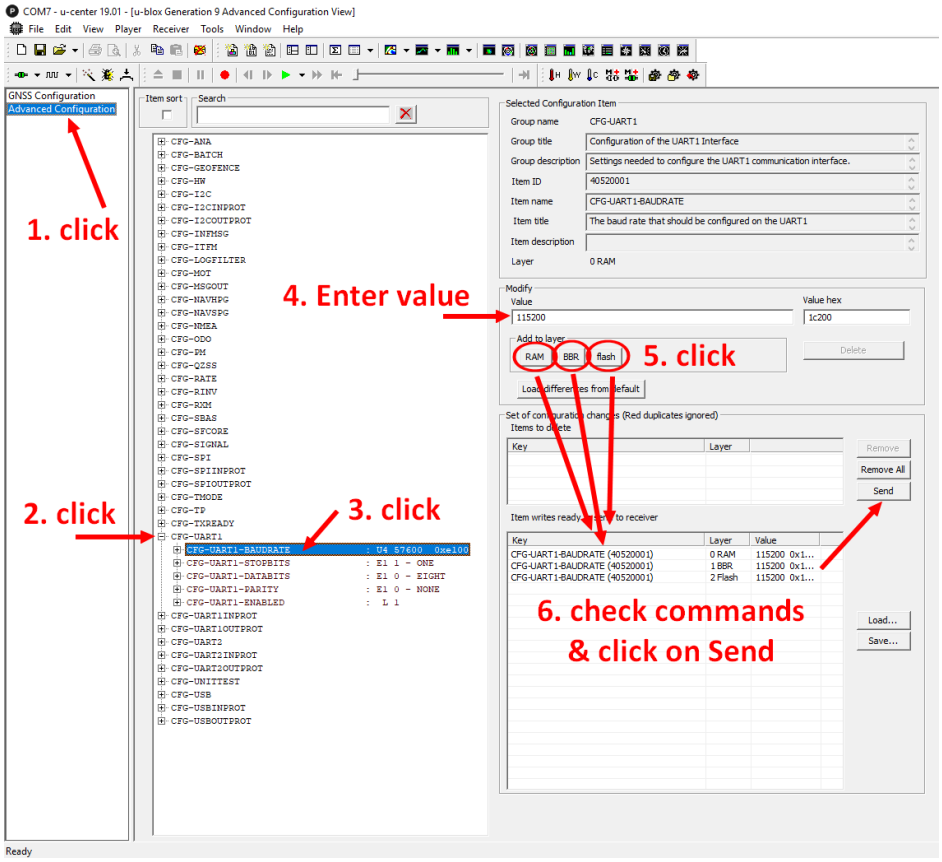
The following slides are examples showing u-center's Generation 9 Advanced Configuration View.

The next six slides are from Drotect's: Configuring the C099-F9P: DroTek Tutorials

You may use alternate sequences and methods, like the Android app acenter's TCP server to drive flash/RAM hexadecimal Flash/RAM data directly. The u-center Advanced Gen 9 configuration is a good start for most users.

Lower right: Time Pulse (TP) and RTK Fixed (RTK) LEDs showing a successful RTCM fix sent from the Base station to the Rover via RDF900x radio links on UART1.

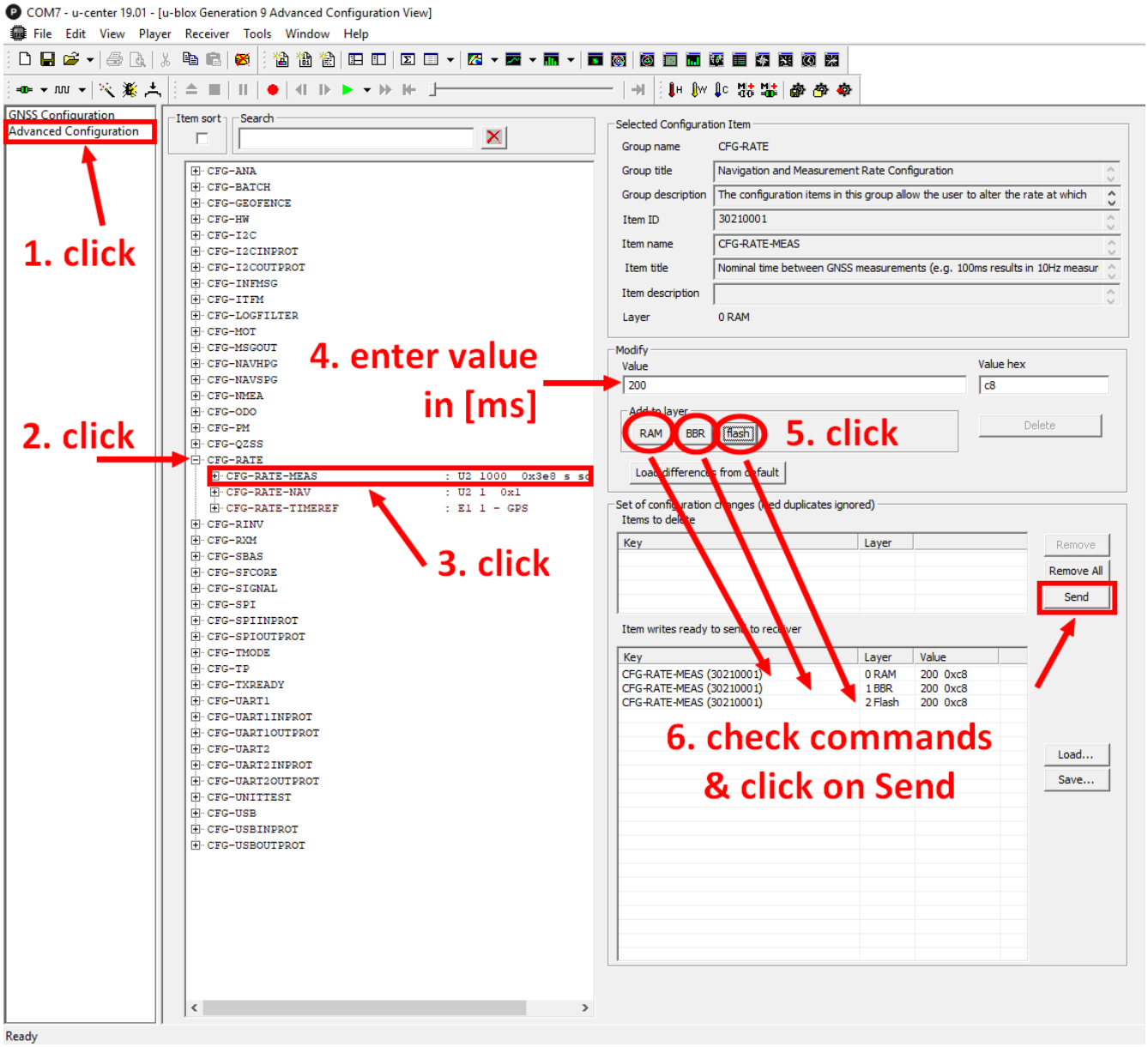
Set Baud Rates on C099-F9P UART1



The DroTek slide above shows the basic workflow in u-center.exe's Generation 9 Advanced Configuration View. The C099-F9P is attached by Bluetooth Serial or USB cable and the COM port and baud rate assigned by Windows. Each class of configuration message can be assigned a value. The configuration value can be stored in several "layers". The pertinent layers for the C099-F9P are Flash and RAM. Ignore the example's BBR values unless you have added the optional battery to the C009-F9P.

All the assignments should be saved to a file. Collect all the changes to a single file. Reset the C099-F9P to default, then send the changes to the C099-F9P. If changes need to be made to the configuration, load the settings from file, remove offending lines, add the changes, save the file, reset to default then send the configuration to the C099-F9P.

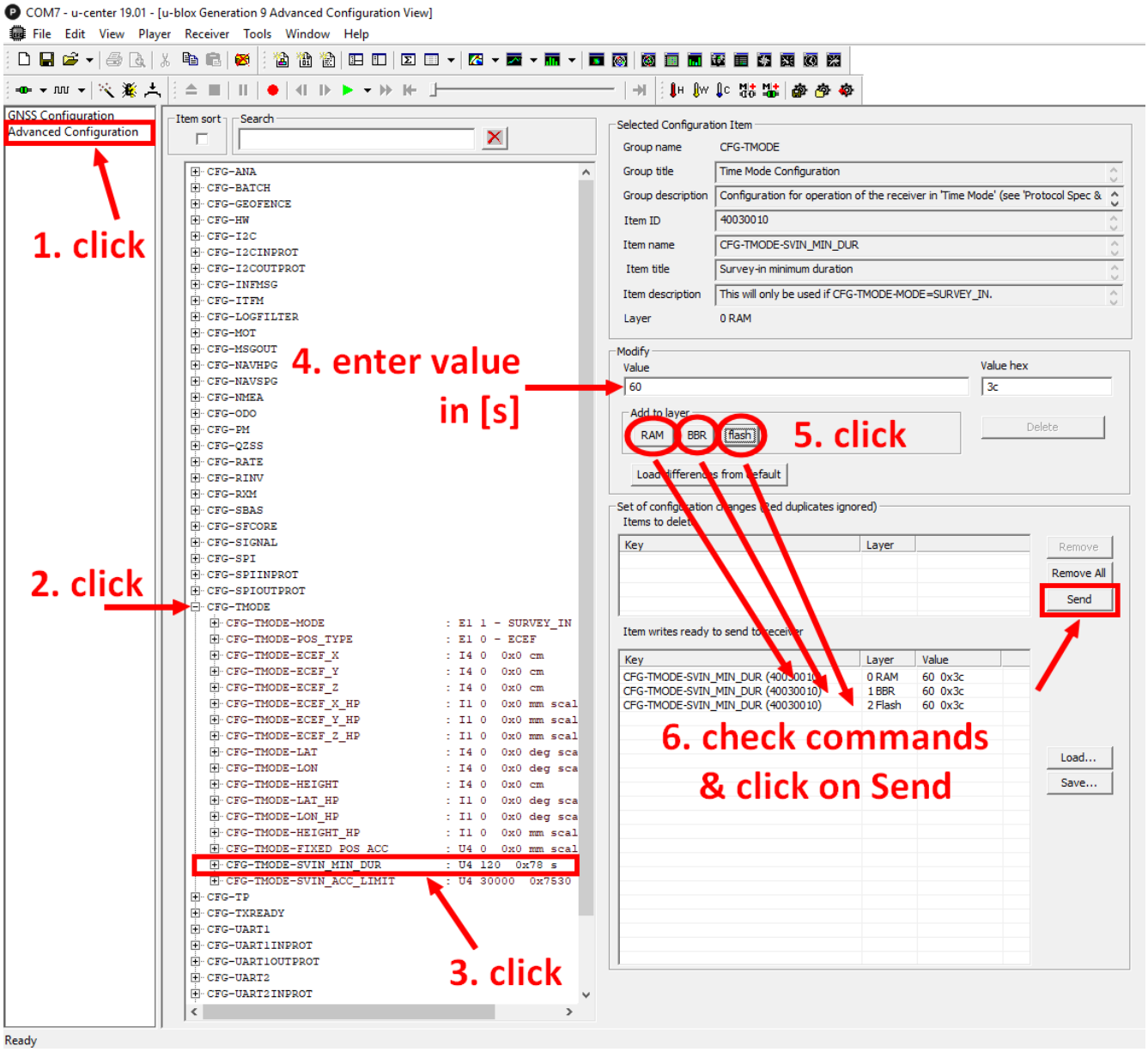
Set the Epoch Rate



Click on the small '+' on the **CFG-RATE** line, click on **CFG-RATE-MEAS** and enter a valid update rate value in milliseconds (ex : a 1s refresh rate = 1000ms)

Set Survey-in time

Click on the small '+' on the **CFG-TPMODE** line and then click on **CFG-TPMODE-SVIN_MIN_DUR.** Enter a valid position value in seconds (ex : a 60s value = 60)



Set position accuracy

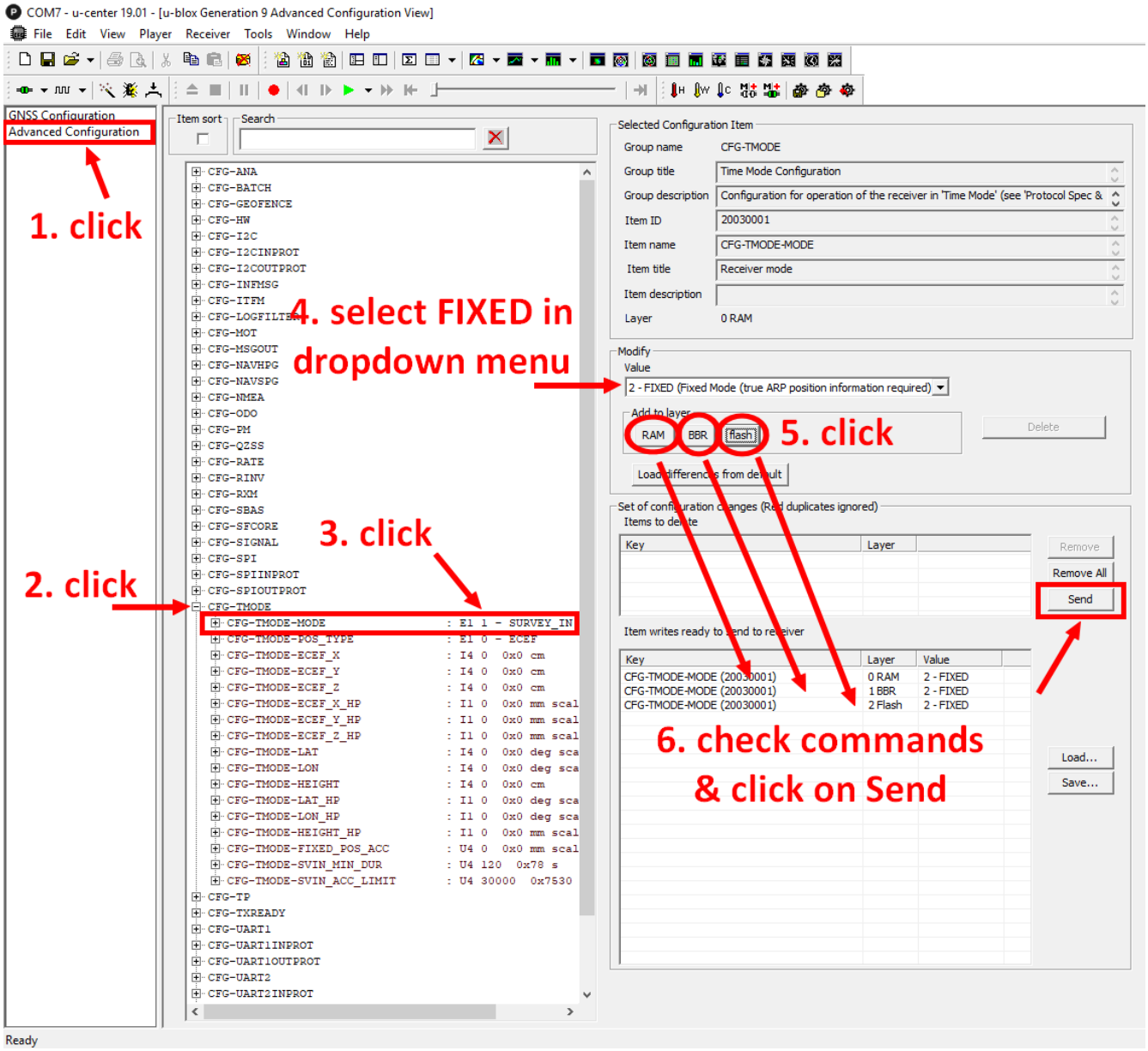
Click on the small '+' on the **CFG-TPMODE** line, click on **CFG-TPMODE-SVIN_ACC_LIMIT.** Now enter a valid position value in one tenth millimeter scale (1m = 10000).

You can monitor the status of the survey-in status using the NAV-SVIN message. The receiver will output messages upon configuration setting, however RTCM 1005 will only be output once the survey-in is completed, or the fixed coordinates are entered for the base antenna.

Base: Fixed Position

Refer to the Fixed Base section above for all the requisite messages. The following example shows how to configure the Base at a known WGS84 LLH location. Configure all the messages shown in the Fixed Base section above using the example procedure shown below.

Set Fixed Base Location



Once the receiver is set in fixed mode, select the position format to use: either LLH or ECEF, with optional high precision (0.1 mm) coordinates (vs the standard cm value). This example uses LLH. To assign the WGS84 LLH values:

Click on the small '+' on the CFG-TPMODE line, click on CFG-TPMODE-POS_TYPE, select "1-LLH" in the dropdown menu and Ram/Flash then Send.

Next click on the small '+' on the CFG-TPMODE line, click on CFG-TPMODE-LAT and enter a decimal degree value as a 1E-7 numeral (35.9635687°= 359635687), choose Ram/Flash then Send. The High Precision (_HP) values can be assigned too in the range (-99..99) as the 8 & 9th digits of the decimal degrees, and 0.1mm measures of the height (altitude+ARP).

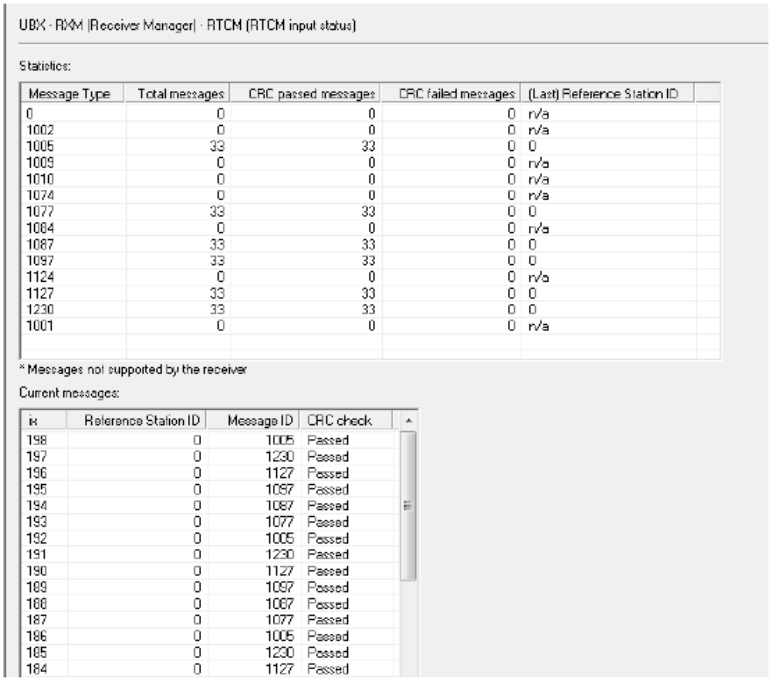
And lastly, click on the small '+' on the CFG-TPMODE line, click on CFG-TPMODE-HEIGHT. Enter the sum of the WGS84 height (altitude) and the Antenna (ARP) height above the ground in cm (eg: 200m => 20,000).

This video covers the same operation:
Setup a ZED-F9P Base in Permanent Location

Rover: Configuration With u-center

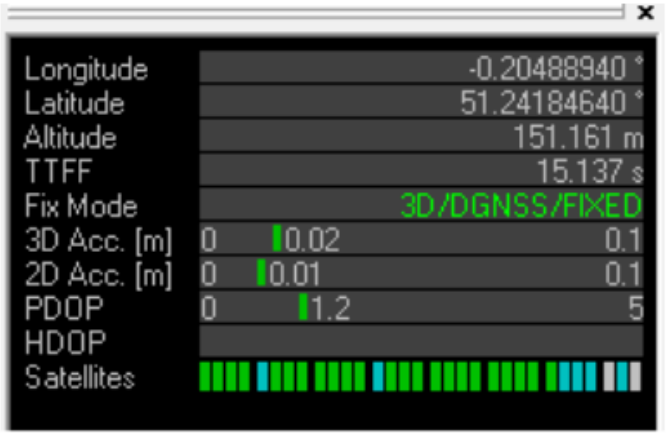
These slides show a working Rover as it receives RTCM data.

To ensure all the required RTCM messages, including most importantly RTCM 1005 or 4072.0, are being received regularly, examine the UBX-RXM-RTCM output in u-center. See below.

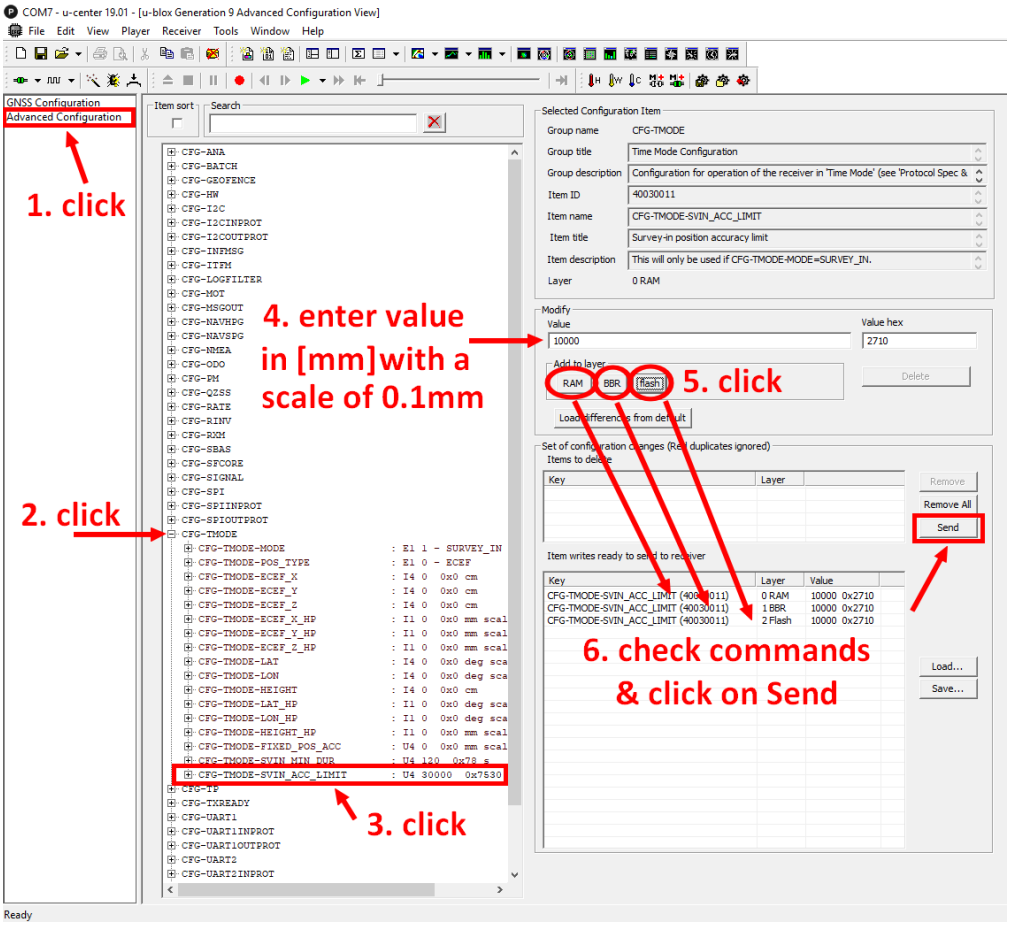


Rover: u-center UBX-RXM-RTCM view.

Once the rover has started to receive valid RTCM messages, it will transition through 3D Fix to 3D/ DGNSS to Float, and, ultimately, into Fixed mode. This will occur when it is receiving all required RTCM messages, including RTCM 1005 or 4072.0, under sufficient signal conditions.
See below:



Rover: u-center data view with RTK Fixed



U-center can compose configuration files and save them to file. The files can submit message and configuration changes to a C09-F9P without u-center by simply sending them out the correct com port in binary file.

This completes the construction, initial setup, and RTK configuration of the Base and Rover. For preliminary testing, a rough “Survey In” will fix the Base location and initiate the RTCM data transfer to the Rover.

Proving Accuracy by Field Test

Ideally, use a set of known valid monuments/points to validate the Base/Rover pair. If available, a known monument pair will expedite a field test. Old monuments may be less valid: plate tectonics, subsidence, fault slips, and the like can shift a monument’s position.

Next best but much slower, make two “Survey In” points. Use u-center to configure the Base to report the UBX Raw messages. The C099-F9P can be further configured to send Raw data for post-fixing a location’s position (CFG-MSGOUT-UBX_RXM_RAWX_I2C, etc). The requisite messages are detailed in the ZED-F9P Interface Description linked below. Collect raw messages in UBX format for some hours (2,12,or 24 are common durations) to log files with ACenter or u-center. Translate the Raw files to RINEX as needed (viz: RTKLibexplorer on GitHub). Depending on the GNSS service (see Appendix), expect some hours to weeks delay before the service has their best data ready to post-fix yours. Once both points have been post-fixed and refined, use a second run with the GNSS post-fixed locii as a Fixed Location Base and prove the Rover with the second post-fixed GNSS point.

Finis

This completes the construction, bench testing, configuration and validation of the C099-F9P Base and Rover survey pair. Explore ublox documents and the myriad GNSS tutorials for more examples showing the use of the ublox ZED-F9P, ODIN-W2, and C099-F9P gear. Mock locations with Android offer a rich body of GNSS related tools for surveying and GIS. It is easy to offer an internet NTRIP caster. High velocity UAV Rover machines in the air or on the ground can acquire RTCMs from the Base by reconfiguring the C099-F9P’s epoch and dynamic model. The ublox ZED-F9P is very flexible.

Appendix

Further Processing to Post Fix Location Accuracy

Ublox raw GNSS observations logged as a UBX file of measures can be refined to a more accurate position by interpolation with observations from other base stations, typically run by governments or survey equipment vendors.

You can also use Net_Diff (see below) or RTKLibExplorer (the 2020+ revisions are needed for L2 observations) on the laptop to make the differential post-fix corrections yourself using data sets provided by one of the websites listed below. RTKLIB Explorer's convbin and rtkconv will convert raw UBX into RINEX .obs/.nav etc.

The ublox u-center and the ACenter Android application can log both NMEA and UBX data. Not all sites directly support UBX. Trimble inc., Canada, the EU and the US all offer websites to make differential corrections. Canada has a good reputation for their service.

The UBX-RXM-RAWX and UBX-RXM-SFRBX messages are used to post-process measurements. UBX-RXM-SFRBX are the messages needed to generate a Navigation file. UBX-NAV-PVT, NAV-SAT and NAV-SIG report the internal status to u-center.

Real time NTRIP caster correction services are offered on the internet. Those can immediately refine the F9P's Base location, though with less accuracy than the best raw data post-fix by interpolation available weeks after the observations. The baseline distance from the NTRIP station to the Base is generally suggested to be under 6 miles. Both ACenter and u-center can attach an NTRIP caster service to offer real time correction. **If using an NTRIP virtual reference service, the C099-F9P must output the NMEA GGA message to return to the NTRIP caster.** Without this, the NTRIP caster will not provide correction information.

NMEA messages are enabled by default on the UART1, I2C, SPI and USB interface.

A "PPP" model can use UBX-RXM-RAWX logged messages from both the Base and Rover to post-fix from a single survey pass (starting with a rough estimate of the Base location, vs known Base location solutions detailed above) with a "kinematic" solution, it is not documented further here.

Web Based GNSS Post-Processing Services

Trimble CenterPoint RTX Post-Processing Service

https://www.nrcan.gc.ca/maps-tools-publications/maps/tools-applications/10925

OPUS: the Online Positioning User Service, process your GNSS data in the National Spatial Reference System

CORS Data Products- National Geodetic Survey

GPS/GNSS Post-Processing Tools | Software | UNAVCO

Laptop Based GNSS Post-Processing Software

GitHub - YizeZhang/Net_Diff at 20200421

A guide to use Net_Diff.pdf

https://github.com/rtklibexplorer

Web Gleaned Notes About the F9P and Post-Processing

An F9P processed on the PPP application of RNCAN using 2 hours (RINEX from UBX) data resulted in a difference of 2cm N and 1cm E against those obtained with a surveying Trimble R8. F9P internal solution had a 99.2% fix rate, while RTKLIB Explorer solutions with RTKNAVI for the F9P solution had a 96.4% fix rate. A four hour set of clear sky UBX messages resolved to a 1.1cm by 0.9 cm solution.

The raw observation and navigation messages are UBX-RXM-RAWX/ UBX-RXM-SFRBX, and solution position messages are NMEA-GNGGA. The NMEA-GNGGA messages from the F9P default to a resolution of 1e-7 degrees of latitude and longitude which works out to roughly 1 cm. Increase the resolution of the GNGGA message by setting a bit in the UBX-CFG-NMEA message. Change the base station position to "RTCM Antenna Position" to take advantage of the F9 base station RTCM 1005 base location messages

Very Useful Websites and Tutorials

These sites offer examples of the hardware used in Base/Rover RTK GPS surveying:
The Taming of the Ublox ZED-F9P – Deep South Robotics
Adding a radio link – rtklibexplorer
SiK Telemetry Radio — Copter documentation
How to Build an RTK Base and Rover – Sky Horse Tech
sparkfun tutorial: how-to-build-a-diy-gnss-reference-station
Configuring the C099-F9P: DroTek Tutorials

RFDesign RFD900X Data Sheet, Setup Manual, Software & Firmware

http://files.rfdesign.com.au/Files/documents/RFD900x%20DataSheet%20V1.1.pdf
http://files.rfdesign.com.au/Files/documents/RFD900x%20Peer-to-peer%20User%20Manual.pdf
https://files.rfdesign.com.au/tools/
https://files.rfdesign.com.au/firmware/

Ublox C099-F9P Resources and Links

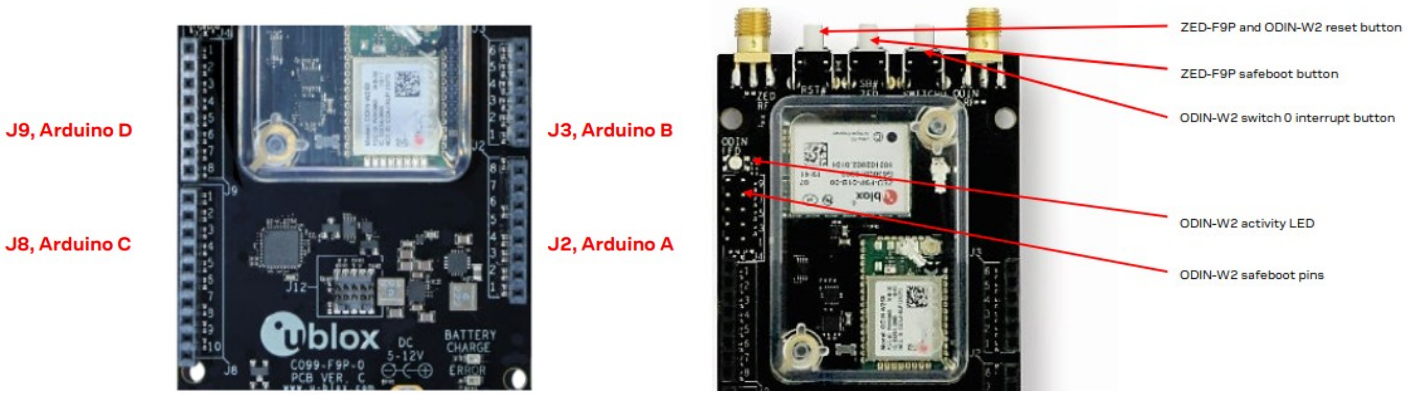
u-blox zed-f9p-module documentation resources
Ublox C099-F9P-AppBoard-Mbed-OS3-FW_UserGuide
u-blox ZED-F9P_InterfaceDescription
u-blox Message Protocols (similar to above)
ublox-C099_F9P quick configuration of Base and Rover
U-Center Manual
researchgate: ZED-F9P vs Trimble
https://en.wikipedia.org/wiki/NMEA_0183
/UBX_F9_100_HPG_113_ZED_F9P.7e6e899c5597acddf2f5f2f70fdf5fbe.bin

Communication interfaces

ZED-F9P provides UART1, UART2, SPI, I2C (Bluetooth/WiFi) and USB interfaces for communication with other devices. The interfaces are set up via the configuration interface which is described in the ZED-F9P interface description [2]. Each protocol can be enabled on several ports at the same time (multi-port capability) with individual settings (e.g. baud rate, message rates, etc.) for each port. Furthermore, more than one protocol can be enabled on a single port at the same time (multi-protocol capability). As a practical matter, the C099-F9P will send and receive data on the ports you assign in u-center. The messages sent and received need to be tailored to your use case. RTCM data must be sent and received on ports set up for RTCM transfers. As well, Raw data used for RINEX/UBX post fixing requires similar configuration to be sent to Acenter/u-center devices for logging and subsequent processing.

Port #
0 0x0000 I2C
1 0x0001 UART1
2 0x0102 UART2
3 0x0003 USB
4 0x0004 SPI

C099-F9P Headers



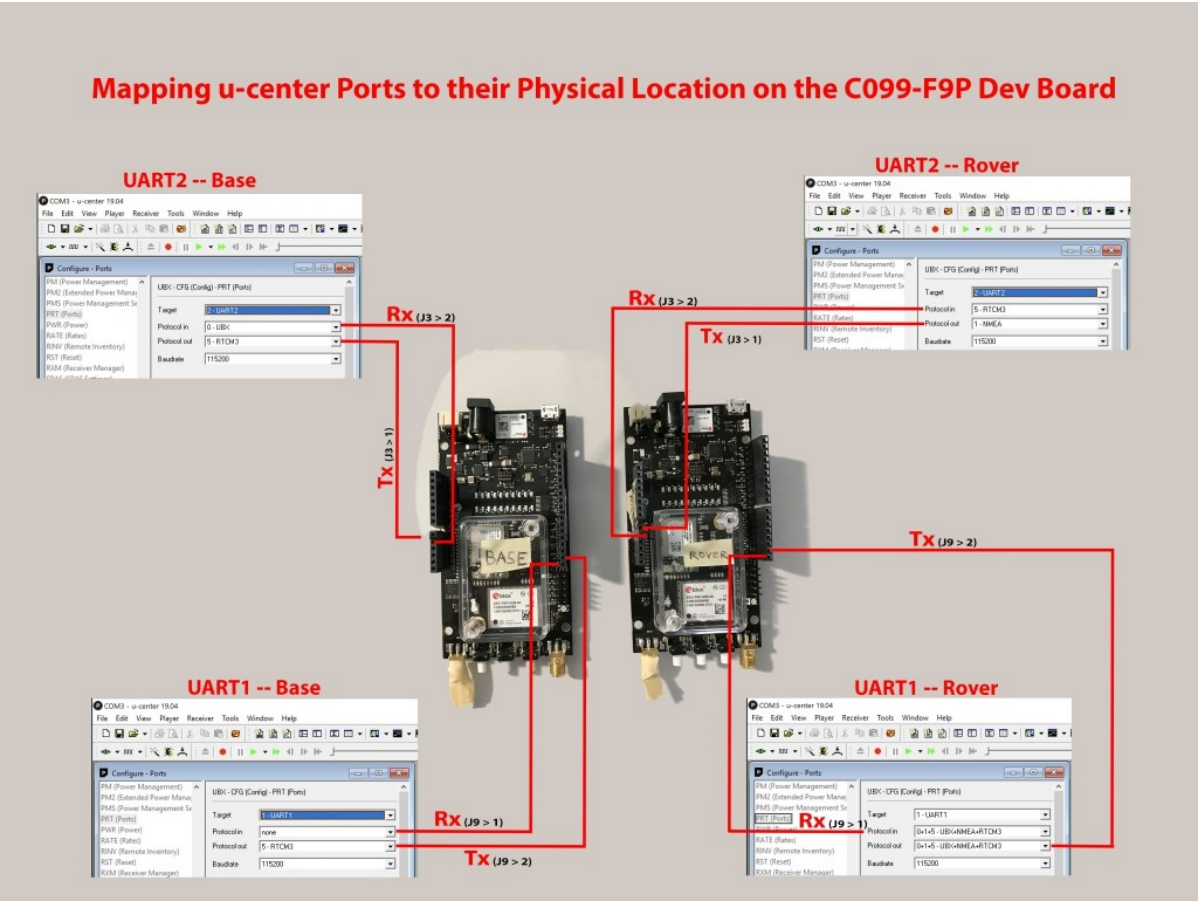
Images:
https://www.Ublox.com/sites/default/files/C099-F9P-AppBoard-Mbed-OS3-FW_UserGuide_%28UBX-18063024%29.pdf

UART interfaces

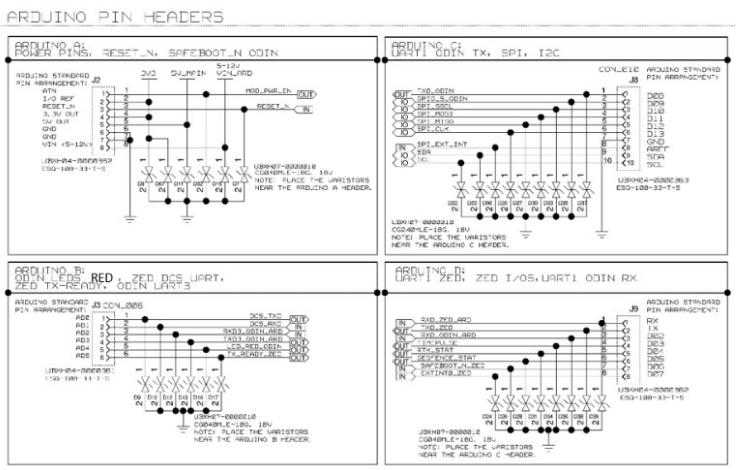
ZED-F9P includes 2 UART ports.

UART1 can be used as a host interface. It supports a configurable baud rate and protocol selection.

UART2 is available as an optional stand-alone RTCM input interface. **It cannot not be used as a host interface.**



C099-F9P UART Pinouts: shows UART1 & UART2 pin locations



The UART serial ports consist of an RX and a TX line. Neither handshaking signals nor hardware flow control signals are available. These serial ports operate in asynchronous mode. The baud rates can be configured individually for each serial port. However, there is no support for setting different baud rates for reception and transmission.

The UART RX interface will be disabled when more than 100 frame errors are detected during a one-second period. This can happen if the wrong baud rate is used or the UART RX pin is grounded. An error message appears when the UART RX interface is re-enabled at the end of the one-second period.

NB: The 5x2 pin set below the ODIN LED routes the the ODIN WiFi/Bluetooth to the Zed-F9P. It is very poorly documented (see the ODIN MBED docs). The second pin pair below the ODIN LED

(OE4) bridges the Zed-F9P UART1 to the Tx/Rx pin pair used here (aka: the Arduino UNO pins). **Set the jumper to enable the RFD900x radio's Tx/Rx connection to the Zed-F9P.**

The default baud rate is 38400 baud. To prevent buffering problems it is recommended not to run at a lower baud rate than the default. The standard rate suggested here for the UARTs is 115200 baud, the USB & I2C rates are 460800 baud.

Linux Based Tools and HowTos

u-center running under wine

The u-center software can be used under linux with **wine**, the windows emulator. Download the u-center software from http://www.u-blox.com/en/evaluation-tools-a-software/u-center/u-center.html and start the downloaded exe file using wine. This will install u-center on your Linux machine.

USB Serial port in wine

To let wine know about the three serial ports linux recognized on the u-Blox GNSS receiver, create symbolic links. Use dmesg to show the ports added when the C099-F9P USB port is connected.

For example: for a u-Blox C099-F9P connected to **/dev/ttyUSB0**, then the command to link this serial port to the **com5** port in wine is:

ln -s /dev/ttyUSB0 ~/.wine/dosdevices/com5

Link all three new C099-F9P USB ports to ~/.wine/dos_devices/comX as shown above (eg: com5, com6 ,com7). U-center will now run as usual and open serial ports to a connected C099-F9P.

Bluetooth serial port in wine

Bluetooth on the C099-F9p is driven by the ZED-F9p's I2C port. In order to fully implement U-center via Bluetooth, additional messages vs the defaults need to be enabled. See the Base Station Advanced View text file shown above in the Base Setup section for a list of the requisite messages. Without those additional messages, U-Center via Bluetooth (I2C) will fail to properly manage and report the ZED-F9P status. Survey-In or Fixed Base modes, for example, may not ever report the final Time status and the base may never begin to report RTCM corrections to the Rover.

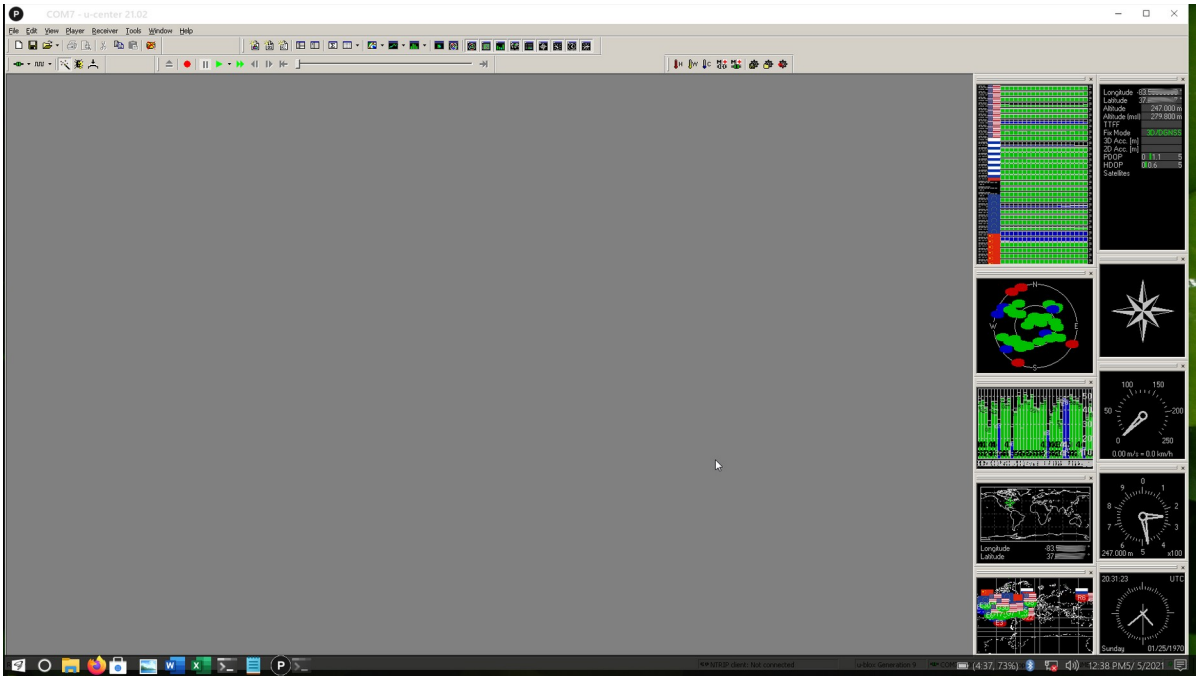
NB: the last step , **rfcomm bind**, needs to be run everytime the Zed-F9P needs to connect, do all the other steps just once. Replace the example device addresses shown below with your C099-F9P's.

*** Construct /etc/bluetooth/rfcomm.conf**
Add these lines to the file(assert your own device #'s):

```
rfcomm0 {
  bind no ;
#rover
  device 60:09:C3:2A:C2:31;
```

```
#base
# device 60:09:C3:29:B2:C4;
  comment "UBlox Rover";
}

rfcomm1 {
  bind no;
#rover
#  device 60:09:C3:2A:C2:31;
#base
 device 60:09:C3:29:B2:C4;
  comment "UBlox Base";
}


* Restart bluetooth
systemctl restart bluetooth

* One Time-Create the symbolic link from rover /dev/rfcomm0 to
COM8, base rfcomm1 to COM9 (for wine/u-center).
rfcomm bind rfcomm0 60:09:C3:2A:1A:28
rfcomm bind rfcomm1 60:09:C3:29:B5:8C

ln -s /dev/rfcomm0 /home/wh/.wine/dosdevices/com8
ln -s /dev/rfcomm1 /home/wh/.wine/dosdevices/com9

* Each session, as root, run:
rfcomm release rfcomm0 60:09:C3:2A:1A:28
rfcomm release rfcomm1 60:09:C3:29:B5:8C
rfcomm bind rfcomm0 60:09:C3:2A:1A:28
rfcomm bind rfcomm1 60:09:C3:29:B5:8C
```

Next run **wine u-center.exe** The applications look and feel under wine should be identical to Windows.



PineBook Pro & Twister with the W10 theme running u-center with a C099-F9P.

PineBookPro: Running u-center.exe on a $220 6 core aarch64 linux laptop

The PineBook Pro is a $220 (as of 05/01/2021) 14” laptop. This example uses “Twister OS” v2.01 (https://twisteros.com/twisterarmbian.html) from a 32GB micro SD card as instructed from the PineBook Pro website (https://wiki.pine64.org/wiki/ROCKPro64_Software_Release#Twister_OS) . NB: after writing the Twister .img file to the SD Card, you must make a one-time edit to its **/boot/armbianEnv.txt** file and replace the dtb name with **rk3399-pinebook-pro.dtb** before booting. This configures the boot loader. The Twister boot step is slow. Create a new user for your sessions, but do not remove the pi user. Login in with your new ID, From the desktop menu options under **Games**, run the **UPDATE BOX86** once before running **Wine Desktop** and installing u-center. After setting the USB ports for wine as detailed above, u-center runs without complaint. Twister is a Debian Linux (Armbian), and will readily support Linux applications, like the Quantum Geographic Information System’s QGIS.

Integrating the C099-F9P with QGIS and QFIELD

For the Rover, the C099-F9P provides precise “mock” location measures to Android 7 & 11 applications via ACenter. Qfield is an Android Rover application built to acquire data for the QGIS mapping program. QGIS and Qfield are both open source and free. Install the “Qfield Sync” plugin in QGIS to construct and transfer the data set for Qfield.

Broadly, a base image and editable vector layers are sufficient to use in Qfield for basic surveying and feature sampling. Qfield can construct lines between features, and guide the Rover in the field to that line, for example. One can make far more sophisticated multi-featured, multi-layered QGIS/Qfield projects. Adding accurate latitude/longitude points is relatively simple.

You will likely need to simplify a complex data and layered QGIS project for Qfield, as the tablet/phone is faster with a small data set.

Clone the project to a working directory. Launch QGIS and open the new working project and remove all externally (net data) sourced layers. Create or duplicate a vector layer to hold the GPS Lat/Long points to be acquired from the C099-F9P Rover. Add any new layers you may want to collect features for. Add line or polygon layers to field capture tracks or tracts.

Invoke the Qfield Sync plugin, and set the preferences for the Qfield project. Set the import and export directories to suit. Exclude the base (bottom visible layer) map or photo from the import list, instead set it in the "Add Base Map" dialog last. Set the new layers created above to be editable offline. Save the Qfield preferences.

In QGIS, reselect the Qfield Sync plugin and opt to run the "Create Qfield Project". Zoom to set the extent of the base map and layers that will be sent to Qfield and the Rover. If there are configuration errors reported at this step, return to the Preferences dialog and repair them. When good, assert the zoom extent as before and OK the run.

Once completed, copy the exported Qfield project directory just created to the Android phone/tablet. Use **SimpleSSHD** (Dropbear) on the Android device and **scp** on the u-center workstation as shown in the **Copy the Files to the Tablet/Phone** section above. Change to the Qfield export directory and replace the ***.cfg** with just ***** to copy all the files in the export directory to the tablet/phone.

Run Qfield on the Android device, and "Open Local Project", navigate to the Download Directory used by Dropbear, and select the Qfield project by name. The QGIS project is now available on the android phone/tablet. Launch Qfield and select the imported project. The tutorials linked below demonstrate how to use Qfield to acquire points and features.

Once you have acquired the features from a Rover tour, send the data back to the workstation. Copy the same files to the workstation to a new directory with **scp**. From QGIS, use the Qfield Sync import step, or simply open the new file to view the Rover data.

QGIS Installation Files
QField APK (In Assets pulldown link)
https://www.line-45.com/industries/conservation
https://www.line-45.com/labs/QGIS-QField.pdf

GPSD, UBXTOOL, et al

Linux offers a suite of tools to support the ZED-F9P. Use only **gpsd** versions above V3.19. The more recent, the better. The ZED-F9P has a very different internal configuration model, and older versions of **gpsd** cannot support it. Debian tends to lag, so use either Sid or building from source rather than installing a stable .deb is the better approach for the time being (20210101). NB: Android tablet/phones can support a Linux installation with limitations (eg: https://f-droid.org/en/packages/tech.ula/). The routines below may work under Android(Userland) and Linux.
gnu source releases gpsd: download and build the most recent.

gpsd ppp-howto: A good discussion of what is PPP
Gnu gpsd suite: a service daemon that monitors one or more GPS'
ubxtool manual pages: gpsd-clients, ubxtool
gpsd and ubxtool examples

Use an Android tablet/phone to configure C099-F9P

This procedure is touchy, it is better to use the standard tools. For the bleeding edge folks, this method may have some appeal, but most will prefer u-blox u-center, as above, and should just skip this section.

The u-center Users Guide section 8.4 GNSS configuration details how to Read/Write binary configuration files to and from the C099-F9P when it is attached by USB cable to u-center. Those procedures make a full configuration, not individual layers or messages. The next section offers a means to configure smaller changes to the C099-F9P over Bluetooth using ACenter and Android in the field. The configurations themselves are still made with u-center, but not immediately applied to the C099-F9P. Instead ,they are captured to file as hex bytes and those hex files are converted to binary. The binary files are copied to the Android tablet/phone running ACenter where they can be loaded on the C099-F9P in the field.

Android is evolving and the security model for files varies by release version. The console routine, Termux, works around many file permission issues. However, for some versions of Android, you may have to search the web to resolve file permission issues. ACenter (aka U-Droid Center) also requires the “Developer Option” to be set (the methods vary by Android version).

Software on an Android Tablet/Phone to Install u-center Configuration Hex Files.

First install the Console and Secure Copy applications on the android tablet/phone:
https://f-droid.org/en/packages/com.termux
https://f-droid.org/en/packages/org.galexander.sshd/

On the tablet/phone launch Termux and while connected to the internet, install **nc** and **mc**.
pkg install nc
pkg install mc (optional)

Use **CTRL-d** to exit the Termux console. **mc** is a file manager and editor that can ease the construction and use of C099-F9P configuration files. **nc** is NetCat, a venerable *nix tool for communicating with data streams.

Make the bash script below on the wine & u-center machine. It is a linux bash script and will not work under Windows or Android. Save the script to a working directory where you can create and write files, and set file modes. Run the script to construct the binary configuration files from the u-center Generation 9 Advanced Configuration View hex files you’ll copy, see below.

*Copy the text below, starting with the **#!/bin/bash** line and ending with the **# End of f2z.sh script.** Save the text to a linux file named **f2z.sh** and then invoke **chmod 766 f2z.sh***

f2z.sh

```
#!/bin/bash
#*********************************************
```

```
# Create ublox binary message files.
#
# This routine makes binary configuration strings for a
# ublox C099-F9P board from hex bytes presented in ublox u-center.
#
# The configuration strings are made in u-center as described in
# the manual. However, rather than send them to an attached C099-F9P,
# use the mouse to copy hex bytes from bottom rightmost window in
# U-Center Gen9 Advanced View, the UBX-CFG-VALSET Message Hex Codes.
# Paste the Flash layer hex bytes to a file (eg: f.hex), and
# the Ram layer bytes to a second file (eg: r.hex). Once Flash and
# RAM hex bytes are copied, run this script as
#
#    /bin/bash f2z.sh f.hex r.hex
#
# Two binary files are produced, zf.cfg and zr.cfg
#
# Copy those files to the Android tablet/phone. Start ACenter,
# make a bluetooth pair with the C099-F9P, and launch ACenter's TCP server.
# Start a Termux console on the tablet/phone paired to the C099-F9P,
# change to the directory where you copied the files. Invoke:
#
#    nc  127.0.0.1 11000 < zf.cfg
#  wait some seconds, then
#    nc  127.0.0.1 11000 < zr.cfg
#
# where 127.0.0.1 is the IP given by ACenter's TCP Server
# and 11000 is the port # given by ACenter's TCP Server
#
#**********************************************
#
F4Z="zf.cfg"
R4Z="zr.cfg"
LOGFILE="z.log";
#
echo " " > ${LOGFILE}
echo "BOF $1" | tee -a ${LOGFILE}
echo "#!/bin/bash" > t.sh
echo -n "echo -ne " >> t.sh
echo -ne "\"\\x" >> t.sh
sed 's: :\\x:g' <$1 >> t.sh
echo -e "\" > $F4Z"  >> t.sh
cat t.sh | tee -a ${LOGFILE}
echo "EOF $1" | tee -a ${LOGFILE}
chmod 777 ./t.sh
./t.sh
#
echo "BOF $2" | tee -a ${LOGFILE}
echo "#!/bin/bash" > t.sh
echo -n "echo -ne " >> t.sh
echo -ne "\"\\x" >> t.sh
sed 's: :\\x:g' <$2 >>t.sh
echo -e "\" > $R4Z"  >> t.sh
cat t.sh | tee -a ${LOGFILE}
echo "EOF $2" | tee -a ${LOGFILE}
```

```
chmod 777 ./t.sh
./t.sh
rm ./t.sh
#
echo
echo "Now copy $F4Z & $R4Z to tablet/phone, run ACenter TCP Server"
echo "From Android Termux console invoke”
echo “nc  127.0.0.1 11000 < $F4Z "
echo "nc  127.0.0.1 11000 < $R4Z "
#
# End of f2z.sh script
```

Generate the Configuration Files with u-center.

Use the standard u-center Generation 9 Advanced Configuration procedures on the workstation/laptop to construct and record a system of configuration messages to a file. Instead of sending the configuration to the C099-F9P immediately, use the mouse to select and copy hex bytes from the bottom rightmost window in U-Center Generation 9 Advanced View, the **UBX-CFG-VALSET Message Hex Codes** window. Paste the hex bytes to a file. Do each layer, Flash and RAM, individually.

Make two files, one for the Flash configuration (**f.hex**) and the other for the RAM (**r.hex**). Save both files in the same directory as **f2z.sh** (see above). Avoid all stray characters in the cut & paste when you copy the hex bytes from u-center, include no extra blanks or lines. Use a plain text editor (no formatting, wrapping, etc) so the sole contents of **f.hex** and **r.hex** are the hexadecimal characters copied from the U-Center Generation 9 Advanced View UBX-CFG-VALSET Message Hex Codes window.

Invoke:
/bin/bash f2z.sh f.hex r.hex
to make the two binary files **`zf.cfg`** and **`zr.cfg`**

Copy the Files to the Tablet/Phone

Create a WiFi or wired network link from the tablet/phone to the linux workstation with the two binary files **`zf.cfg`** and **`zr.cfg`** made above. Attach the tablet/phone to the same network. Use any file manager with a wired connection to copy the ***.cfg** files. To copy the pair of binary **cfg** files to the table/phone via WiFi, first start the **SimpleSSHD** (DropBear) application on the android tablet/phone, it will report the IP address you need next. Now return to the directory on the u-center workstation with the **zf.cfg** & **zr.cfg** files and from there invoke:

scp -P 2222 *.cfg 192.168.01.100:/sdcard/.

Replace the **192.168.01.100** with the IP address shown by **SimpleSSHD**.
Replace **/sdcard/.** with the destination directory on the tablet/phone (keep the suffixed dot).
If you’ve changed the default port in **SimpleSSHD,** adjust the **2222** above to the new port.
The ***.cfg** is promiscuous, and will copy all **.cfg** files in the current directory.

As soon as the **scp** command on the u-center workstation is entered, the **SimpleSSHD** (DropBear) application on the tablet/phone will immediately generate and display a one time password and you will be prompted by **scp** on the workstation for that password. Type in the password on the u-center

workstation to make **scp** copy the files to the destination directory on the tablet/phone. Stop **SimpleSSHD** on the tablet/phone when you have finished copying the files.

Once you have the u-center ZED-F9P configuration files **zf.cfg** & **zr.cfg** on the tablet/phone, you can apply them in the field as detailed below. Keep as many configuration files as you like, in individual directories.

Configure the C099-F9P in the Field From the Tablet/Phone

The android tablet/phone with ACenter is the primary user interface to the Rover in the field. Acenter can offer a TCP Server bridged to the C099-F9P for the binary configuration files **zf.cfg** & **zr.cfg**

ACenter can reset the C099-F9P to default before you apply the configuration. Wait some seconds after the reset, then recheck the Bluetooth link to assure the C099-F9P is connected and running before starting ACenter's TCP Server, switching to a console, and copying the binary files.

With both a Termux Console Terminal shell and ACenter running on the Android tablet/phone, and the Bluetooth link to the C099-F9P Base or Rover selected in ACenter, start the TCP server in ACenter. Note the address and port offered there (typically 127.0.0.1 when there is no Wifi or Cell net connections). Swap focus to the Termux Console command line. First copy the **zf.cfg** file to the C099-F9P. Use the given address and port from the TCP server, shown in this example as IP: 127.0.0.1, Port: 11000. Wait some seconds between the **zf.cfg** and **zr.cfg** runs.

nc 127.0.0.1 11000 < zf.cfg
nc 127.0.0.1 11000 < zr.cfg

If the tablet/phone has disconnected from the Wifi/wired network, you may instead use an Android shell script like **c2z.sh** shown below:

```
#!/system/bin/sh
nc 127.0.0.1 11000 < zf.cfg
sleep 5
nc 127.0.0.1 11000 < zr.cfg
```

Invoke **c2z.sh** from the tablet/phone's console as: **/system/bin/sh c2z.sh** First create **c2z.sh** by copying the four lines above to a file named "c2z.sh" on the tablet/phone, and from a console:
chmod 766 c2z.sh

NB: Some configuration messages invoke a reply from the ZED-F9P that slows down the information exchange. If the above method fails, either use u-center or break your configuration messages into smaller groups, and send them sequentially.

Geodesy, very briefly

This section offers a nutshell review of the field of geodesy, modeling the earth for mapping.

When documenting survey data or when giving coordinates to others, always cite the datum and projection of the data. The datum defines the shape, size, and position of the best ellipsoidal approximation of the true surface of the Earth, either locally or globally. A datum is described by four

parameters: the equatorial radius, the polar radius, a measure of flattening (vs round), and the ellipse's centroid's distance to the center of the Earth. According to Appendix B of the ZED-F9P Integration Manual, when using RTK, the Rover outputs positions in the datum of the reference Base station.

Per https://www.fws.gov/r7/nwr/Realty/data/LandMappers/Public/Help/HTML/Datumsprojectionsandcoordinatesy.html, the parameters for several datums that are commonly used in the United States:

Datum	NAD27	NAD83	WGS84
Ellipsoid name	Clarke 1866	GRS 1980	WGS 1984
Semi-major axis (meters)	6,378,**206.400**	6,378,**137.000**	6,378,**137.000**
Semi-minor axis (meters)	6,356,**583.800**	6,356,**752.314**	6,356,**752.314**
Inverse flattening	0.00339**008**	0.00335**281**	0.00335**281**
Eccentricity squared	0.006**76866**	0.006**69438**	0.006**69438**
Datum type	Local (N. America)	Local (N. America)	Global

For Example:

Datum	NAD27	NAD83	WGS84
Longitude	134° 21' **49.21**"	134° 21' **55.59**"	134° 21' **55.66**"
Latitude	58° 22' **3.30**"	58° 22' **2.13**"	58° 22' **2.15**"

All three coordinates above identify the same physical point on the Earth's surface, but the seconds value varies.

With Mercator projection, directions are correct while other qualities such as area are distorted. The Mollweide projection gives accurate areas but directions are distorted.

Ublox ZED-F9P Integration Manual Appendix B: Fitting Rover Observations to Maps

RTCM ITRF geodetic models

Real time kinematic (RTK) is a differential system where the rover uses the reference datum of the reference station (the Base's datum). The International Terrestrial Reference Frame (ITRF, see Down to Earth, below) must be obtained from the reference network and then used to transform the rover position output to match the required reference frame. The rover does not output the position in the local (it's own) receiver WGS84 (based on ITRF2008) datum, it matches the (Base) reference receiver reference frame. The user application needs to do the transformation for use in a mapping application if it does not use the same reference frame. An offset can occur if this is not done.

The ITRF reference frame years are listed below:
- ITRF94
- ITRF96
- ITRF97
- ITRF2000

• ITRF2005
• ITRF2008
• ITRF2014

The user needs to transform the frame of the reference receiver (or base) into the required system being used by their application and mapping system. If comparing the rover position with a reference system, the same RTCM stream should be used to ensure that the reference and the rover output the same position. If the reference system is post-processed, its output must be transformed to the same ITRF and datum being used by the rover.

For example, when viewing RTK-accurate positions that could be in any ITRF transform reference frame (the base's receiver reference frame) on Google Earth: The heights on Google Earth refer to EGM96 and are, therefore, Geoidal heights. The lat/long are referred to the WGS 84 (ITRF2008) ellipsoid. The result is a visible inaccuracy, and the RTK receivers' position output must be transformed to the Google Earth transform system before it can be used. There may also be local map tile inaccuracies.

Down to Earth: the Gravimetric Centroid Datums

Viz also: https://en.wikipedia.org/wiki/World_Geodetic_System
The world geodetic system called WGS 84 is the reference system used by the Global Positioning System (GPS). It is geocentric and globally consistent within ±1 m. Current geodetic realizations, ITRFs, of the International Terrestrial Reference System (ITRS) geocentric reference system family as maintained by the International Earth Rotation Service (IERS) are geocentric and internally consistent at the few-cm level while still being meter-level consistent with WGS 84.

https://en.wikipedia.org/wiki/International_Terrestrial_Reference_System
ITRF solutions

```
The ITRF realizations developed from the ITRS since 1992 include:
Name code   Epoch       EPSG  Notes

ITRF92      1988.0      4914  First realization of the ITRS
...
ITRF2000    1997.0      4919  First solution that combines unconstrained space
geodesy solutions without a tectonic plate motion model[

ITRF2005    2000.0      4896  Constructed with input data under the form of time
series of station positions and Earth Orientation Parameters

ITRF2008    2005.0      5332  Includes tropospheric modeling and improved solution
methods.

ITRF2014    2010.0      7789  Generated with an enhanced modeling of nonlinear
station motions
```

Geodetic coordinate systems and ellipsoids

In practice, the relevant organizations choose to keep their respective frames very close to the International Terrestrial Reference Frame (ITRF), defined and managed by the International Earth Rotation and Reference Systems Service (IERS). However, because the Earth's tectonic plates and

even parts of the Earth's core move, new versions of ITRF are defined every few years, generally with changes of the order of a few millimeters. Consequently, the major GNSS occasionally decide that they need to update their reference frames to be better aligned with the latest ITRF.

So, for example, GPS switched to WGS84 (G1150) in GPS week 1150 (early 2002) based on ITRF2000, while GLONASS switched from PZ90.02 to PZ90.11 at the end of 2013, based on ITRF2008. The net effect of this is that all the major GNSS use almost the same reference frames, but there are some small (generally sub-centimeter) differences between them, and these differences occasionally change.

In order to produce positions that can be shown on a map, it is necessary to translate between raw coordinates (for example, x, y, z) and a position relative to the Earth's surface (for example, latitude, longitude and altitude), and that requires defining the form of ellipsoid that best matches the shape of the Earth.

Historically, many different ellipsoid definitions have been used for maps, many of which predate the existence of GNSS and show quite significant differences, leading to discrepancies of as much as 100 meters in places.

Fortunately, most digital maps now use the WGS84 ellipsoid, which is distinct from the WGS84 coordinate system, but defined by the same body. **However, for RTK position accuracies now in the centimeter-level, the ITRF year for the WGS84 datum used by the mapping system must be known in order to transform the RTK rover position into the correct reference frame.**

The ZED-F9P stores the EGM96 geoid model with limited resolution, leading to degraded accuracy of the reported mean sea level height and geoid separation. If the user application needs higher geoid separation accuracy, it is required to apply its own adjustment to the ellipsoidal height output from the ZED-F9P.

A Short History of the Datum: the ordered tuple describing the earth as an ellipsoid

The analytic geometry and associated physics use of the phrases “translation”, “rotation”, and “transformation” differs from the meanings used in the mapping and geodesy realms. https://en.wikipedia.org/wiki/Geographic_coordinate_conversion ...”In geodesy, geographic coordinate *conversion* is defined as translation among different coordinate formats or map projections all referenced to the same geodetic datum. A geographic coordinate *transformation* is a translation among different geodetic datums. “

https://en.wikipedia.org/wiki/Geodetic_system
Since the rise of the global positioning system (GPS), the ellipsoid and datum WGS 84 that it uses has supplanted most others in many applications. The WGS 84 is intended for global use, unlike most earlier datums.
…
Datum conversion is the process of converting the coordinates of a point from one datum system to another. Because the survey networks upon which datums were traditionally based are irregular, and the error in early surveys is not evenly distributed, datum conversion cannot be performed using a simple parametric function. For example, converting from NAD27 to NAD83 is performed using

NADCON (later improved as HARN), a raster grid covering North America, with the value of each cell being the average adjustment distance for that area in latitude and longitude. Datum conversion may frequently be accompanied by a change of map projection.
…

The North American Datum of 1983 (NAD 83) is "The horizontal control datum for the United States, Canada, Mexico, and Central America, based on a geocentric origin and the Geodetic Reference System 1980 (GRS80). "This datum, designated as NAD 83 ...is based on the adjustment of 250,000 points including 600 satellite Doppler stations which constrain the system to a geocentric origin." NAD83 may be considered a local referencing system.

WGS 84 is the World Geodetic System of 1984. It is the reference frame used by the U.S. Department of Defense (DoD) and is defined by the National Geospatial-Intelligence Agency (NGA) (formerly the Defense Mapping Agency, then the National Imagery and Mapping Agency). WGS 84 is used by DoD for all its mapping, charting, surveying, and navigation needs, including its GPS "broadcast" and "precise" orbits. WGS 84 was defined in January 1987 using Doppler satellite surveying techniques. It was used as the reference frame for broadcast GPS Ephemerides (orbits) beginning January 23, 1987. At 0000 GMT January 2, 1994, WGS 84 was upgraded in accuracy using GPS measurements. The formal name then became WGS 84 (G730), since the upgrade date coincided with the start of GPS Week 730. It became the reference frame for broadcast orbits on June 28, 1994. At 0000 GMT September 30, 1996 (the start of GPS Week 873), WGS 84 was redefined again and was more closely aligned with International Earth Rotation Service (IERS) frame ITRF 94. It was then formally called WGS 84 (G873). WGS 84 (G873) was adopted as the reference frame for broadcast orbits on January 29, 1997.[7] Another update brought it to WGS84(G1674).

The WGS 84 datum, within two meters of the NAD83 datum used in North America, is the only world referencing system in place today. WGS 84 is the default standard datum for coordinates stored in recreational and commercial GPS units.

Users of GPS are cautioned that they must always check the datum of the maps they are using. To correctly enter, display, and to store map related map coordinates, the datum of the map must be entered into the GPS map datum field.

Coordinate Reference Nomenclature

https://en.wikipedia.org/wiki/ECEF: earth-centered, earth-fixed (ECEF) represents positions as X, Y, and Z coordinates. The origin is defined as earth’s gravitational centroid.

Most geographic information systems (GIS) and GIS libraries use EPSG Geodetic Parameter Dataset codes (see below) as Spatial Reference System Identifiers (SRIDs) and EPSG definition data for identifying coordinate reference systems, projections, and performing transformations between these systems, while some also support SRIDs issued by other organizations (such as Esri).

https://en.wikipedia.org/wiki/EPSG
The European Petroleum Survey Group (EPSG) Geodetic Parameter Dataset (also EPSG registry) is a public registry of geodetic datums, spatial reference systems, Earth ellipsoids, coordinate transformations and related units of measurement. Each entity is assigned an EPSG code between

1024-32767, along with a standard machine-readable well-known text (WKT) representation. The dataset is actively maintained by the IOGP Geomatics Committee.(https://epsg.org/home.html)

A good overview: https://www.oc.nps.edu/oc2902w/coord/coord.pdf

1 degree of latitude is 111.3km at the equator.
The change of a fraction of a minute:1 degree = 60 minutes
1 minute = 1 degree/60 = 111.32km / 60 = 1.855km
1 minute = 1855m
0.1min = 185.5m
0.01min = 18.55m
0.001min = 1.855m
0.0001min = .1855m = 185.5mm
0.00001min = 0.0185m = 18.55mm = 1.855cm

Additional precision: using the UBX protocol can output up to 8 digits of precision in dd.dddddddd format which is 1.11mm of precision

https://geodesy.noaa.gov/TOOLS/Htdp/Htdp.shtml
HTDP is a utility that allows users to transform positional coordinates across time and between spatial reference frames.
To compare across dates: https://geodesy.noaa.gov/TOOLS/Htdp/Htdp_displacement.html

The 2022 standard being implemented to replace NAD83:
https://cdn.ymaws.com/www.mnsurveyor.com/resource/resmgr/2019-annual-meeting/Dave_Zenk_01_New_Datums_Repl.pdf

Example of a Statutory State Plane reconciling with the upcoming NATRF 2022 specification.
https://geodesy.ky.gov/assets/pdfs/KySPCS-SSD_Draft.pdf

Snippets From The Kentucky State Plane Coordinate System Standards and Specifications Document July 1, 2020 (ignore freely, this only applies to Kentucky post 2021)
Zone 3: NATRF 2022 Kentucky LDP Zone 3 – South East Zone Name: **NATRF 2022** Kentucky LDP Zone 3 – South East (KYLDP3SE-2022) Projection Type: Oblique Mercator (OM) Central Scale Factor (k0): 1.000035 (35 ppm) Central Meridian: 276° 45’ E (276.75° E) Central Parallel: 37° 27’ N (37.45° N) Projection Axis Skew Azimuth: 60° False Northing: 150,000 meters False Easting: 1,810,000 meters Included Counties: Bell, Breathitt, Clay, Floyd, Harlan, Jackson, Johnson, Knott, Knox, Laurel, Lee, Leslie, Letcher, Magoffin, Martin, Owsley, Perry, Pike, Rockcastle, Whitley, Wolfe

========
The WGS84 (G1674) frame is identical to the International Terrestrial Reference Frame of year 2008 (ITRF2008). Thus, except for very accurate geophysical calculations
(crustal dynamics; tectonic strain determination, etc.), the WGS84 (G1674) is supposed to be aligned with the ITRF2008. In another words, both have the same origin, orientation, and scale.

https://www.ngs.noaa.gov/CORS/Articles/SolerWGS84(G1674)-to-NAD83(2011).pdf

Let's assume now that we want to transform coordinates between the WGS84 (G1674) ≡ ITRF2008 and NAD83 (2011). The first thing that we need to keep in mind is that, in theory, the WGS84 (G1674) or ITRF2008 coordinates are the ones initially computed by the manufacturer's software. However, they are originally obtained at the time of the observation, that is, the epoch corresponding to the mid-point of the observation window during the GPS data were collected. Consequently, we cannot compare coordinates from a set of observations taken today to results we determined one year ago. These two sets of coordinates must be reduced to a common epoch before the comparison is made.
...
In order to do this final step, the 14 transformation parameters (three shifts, three rotations, one scale and their variations with time) between the two frames are needed. The rigorous transformation is somewhat involved [Soler and Snay, 2004], thus, the easiest approach is to use the software developed at NGS which is interactively available through the Internet. Its name is Horizontal Time Dependant Positioning (HTDP) and can be located at the following URL:
http://www.ngs.noaa.gov/TOOLS/Htdp/Htdp.shtml

Replacing the three existing NAD 83 reference frames will be four plate-fixed terrestrial reference frames. The tectonic plate for each frame may be inferred from their names, which are:

- North American Terrestrial Reference Frame of 2022 (NATRF2022)

SPCS2022 will replace the State Plane Coordinate System of 1983 (SPCS 83), and policies associated with SPCS 83 will be superseded. Convert NAD83 or NAD27 geodetic positions (latitude and longitude) to State Plane Coordinates (SPC) HTDP WGS84(ITRF2008) to NAD83(2011) / NAVD 88 / SPCS83 Kentucky South zone. The NAD 83 realization used for SPCS zones depends on the tectonic plate where it is located:

- *North America and Caribbean plates:* NAD 83 (2011)

https://geodesy.noaa.gov/NCAT/
https://geodesy.noaa.gov/TOOLS/spc.shtml

Troubleshooting

* When things are working well, expect the Base and Rover to take about 1 minute to first fix, and about 3 minutes to get 1-4 meters precision. Set the Base to Tmode Survey In with 180 seconds time, and 40000 tenth millimeter PDOP (4 meters). Make those values larger for poorer observation conditions. Once the survey is complete, the Base has met the Time and PDOP accuracy values. Run u-center on the Base station and expect to see "Time" reported by u-center's "data" display, and to soon see the Rover's RTK LED flash yellow, indicating RTK Float. If the Rover hasn't gained enough observations, this will take longer. The RTK LED should become a constant yellow after a couple of minutes, indicating the Rover is now "Fixed" by the Base RTCM messages. This is success.

* The ODIN LED shows green when the ODIN radio has initialized properly. Ignore the ODIN LED flashing red- it simply means that the Bluetooth connection to the C099-F9P has ended by a standard disconnect. This happens any time that u-center, Acenter Udroid, or any other Bluetooth device has been connected and then disconnected.

* Bench test the Base/Rover pair. The antennas need a good clear view to the satellites.

* An oscilliscope can prove the Tx/Rx connections, expect a 115200 Baud 3.3v square wave. An alternative is a UART connection and a terminal program- expect to see binary data when RTCM messages are sent.

* Use u-center's "Messages" menu to inspect the **ubx comm** values to observe RTCMs, Tx, Rx data counts and history.

* If either of the OE4 jumpers (Base or Rover) are missing, no Rx data will be observed on the Rover. The Base will report Tx being sent, but it never reaches the RFD900x radio until the OE4 jumper is in place.

* Reflash the Zed-F9P firmware to flush possible misconfigurations.

* The u-center "set to default" (cfg-cfg message) seems to not always clear values set to Flash memory, reflashing the firmware will clear those.

* Use the ublox github sourced Base and Rover advanced configuration text files as a test set to validate the RTK link. Add new messages incrementally to those files and test each revision. Do not use your own configuration files to troubleshoot unless you are experienced and confident that the files are good.

* While the documentation details port protocols, most are preset and do not need assertion in the configuration files. Try the github configuration files and use u-center to check what messages are sent, you may not need to add to the default configuration.

* The UBXCFG-MSGOUT-UBX_RXM_RAWXX & CFG-MSGOUT-UBX_RXM_SFRBX are needed to log for UBX post-fix corrections.

Revision History

12/29/2021:
* Added details about OE4 jumper
* Added images of RTK Float/Fixed LED
* Added Troubleshooting details.
* Revised the example Base and Rover advanced configuration text files.

02/13/2022:
* Added Udroid ACenter v 1.8 results for mock location in Android > v7.
* Added chmod and rm to f2z.sh script

03/25/2022:
* Added requiring Udroid ACenter to remain in the foreground to not time out UBX logging.
* Added suggestion about Base station benchmarks needing to lie flush with the ground to avoid being disturbed by animals.

04/06/2024:
* Added the detail that some Android phones do not support mock locations (even though they proffer the feature in the Developer Options).
* Refined the reference to the RTKLIB toolset to explicitly list the https://github.com/rtklibexplorer/RTKLIB toolset. It makes a better RINEX file than the older releases.
* Checked that Digikey still sells the C099-F9p. U-Blox quit making them, so the supply is limited.