EVK-F9DR

Evaluation kit

User guide

Abstract

This document describes the structure and use of the EVK-F9DR evaluation kit and provides information for evaluating and testing the automotive u-blox F9 multi-band dead reckoning technology.

UBX-23007618 - R01
C1-Public

Document information

Title	**EVK-F9DR**	
Subtitle	Evaluation kit	
Document type	User guide	
Document number	UBX-23007618	
Revision and date	R01	13-Nov-2023
Disclosure restriction	C1-Public	

This document applies to the following products:

Product name	Type number	Firmware version	PCN reference
EVK-F9DR	EVK-F9DR-0-00	LAP 1.30	N/A

u-blox or third parties may hold intellectual property rights in the products, names, logos, and designs included in this document. Copying, reproduction, or modification of this document or any part thereof is only permitted with the express written permission of u-blox. Disclosure to third parties is permitted for clearly public documents only.
The information contained herein is provided “as is” and u-blox assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability, and fitness for a particular purpose of the information. This document may be revised by u-blox at any time without notice. For the most recent documents, visit www.u-blox.com.
Copyright © u-blox AG.

Contents

1 Introduction

The EVK-F9DR can be used to test and evaluate all u-blox products based on the u-blox F9 multi-band GNSS dead reckoning technology. The evaluation kit (EVK) is equipped with the ZED-F9K module pre-flashed with the LAP 1.30 firmware, and allows out-of-the-box performance and feature evaluation of the following products:

- ZED-F9K
- UBX-F9940-KA-DR

When re-flashed with DBD 1.30 firmware, the module may also be used to evaluate the following products:

- ZED-F9L
- UBX-F9140-KA-DR

The built-in USB interface provides both power supply and a high-speed communication interface. The EVK is compact and provides a flexible and user-friendly interface between the GNSS module and test vehicles. Furthermore, it can be used with a notebook or PC running the GUI-driven u-center application, making it the perfect companion through all stages of evaluation and design-in phases of projects.

☞ EVK-F9DR is also equipped with a NEO-D9S L-band corrections receiver pre-flashed with PMP 1.04 firmware.

1.1 Highlights

- Multi-constellation, multi-band GNSS
- Automotive dead reckoning (ADR)
- Real-time kinematic (RTK)
- Configurable CAN interface
- Dedicated pins for wheel tick and direction inputs
- USB, UART, RS-232 connections
- Battery-backed RAM (BBR) through micro-USB or CR2032 coin cell battery
- Wake-on-Motion feature

1.2 Kit contents

- EVK-F9DR unit
- ANN-MB1 L1/L5 high-precision GNSS antenna with a 5-meter cable
- 1-meter USB-C cable
- 1.8-meter micro-USB cable
- PointPerfect promotion card

1.3 System requirements

- PC with Windows operating system
- u-center GNSS evaluation software
- Odometer input from a vehicle (for ADR only)

2 Device description

2.1 USB

A USB-C connector is featured for data communication and power supply. USB drivers are installed automatically through Windows update.

2.2 UART

The unit includes an RS-232 port which can be dynamically connected to the UART of the ZED-F9K, the NEO-D9S, or the onboard MCU. Selection of the UART connection is controlled by the SEL_NEO_N and SEL_MCU_N pins on the front connector.

SEL_NEO_N	SEL_MCU_N	Connected component
LOW	OPEN	NEO-D9S
OPEN	LOW	MCU
OPEN	OPEN	ZED-F9K
LOW	LOW	None

The selected UART interface is also available via the RxD and TxD pins on the front connector. The pins are at TTL voltage levels.

Do not use flow control with the RS-232 port.

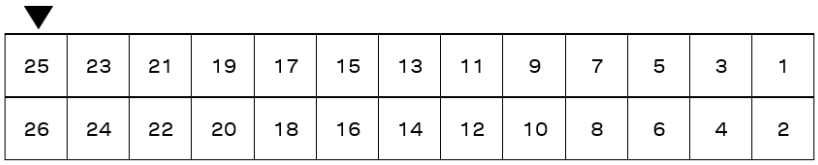
2.3 Antenna

There are two female SMA connectors on the unit. The one on the front (RF_IN) is used by the ZED-F9K module, and the one on the back (RF_AUX) is used by the NEO-D9S module.

The kit includes a u-blox ANN-MB1 L1/L5 active multi-band GNSS antenna with a 5-meter cable to be used by the ZED-F9K module. A suitable antenna for the NEO-D9S is sold separately.

2.4 26-pin front connector

The connector and its signals are described in the table below. The pins are numbered right to left, with the odd numbers on the top row and the even numbers on the bottom row.



Pin no.	Pin name	I/O	Level	Description
26	VIN 5-24V	I	5 - 24 V	Main voltage input – can be used in place of the USB connector
25	GND			Common ground for casing, power and serial interfaces
24	GND			Common ground for casing, power and serial interfaces
23	CAN_H	I		CAN high signal (ISO 11898-2)
22	VIN_LDOS	O		LDO input voltage measurement point
21	CAN_L	I		CAN low signal (ISO 11898-2)
20	CUR_ZED	O		ZED-F9K current measurement point
19	SEL_NEO_N	I		Pull-down signal for enabling UART communication with the NEO-D9S
18	CUR_NEO_D9	O		NEO-D9S current measurement point

17	SEL_MCU_N	I		Pull-down signal for enabling UART communication with the MCU
16	TIME_MARK_J2	I		Time mark input
15	WOM	O		Wake-on-motion signal output
14	GND_A			Ground for speed and direction inputs
13	WT	I	5 – 24 V	Wheel tick pulse input
12	GND_A			Ground for speed and direction inputs
11	FWD	I	5 – 24 V	Direction of travel input
10	V_BCKP_MEAS	O		Backup voltage measurement point
9	SDA / CS	I/O	3.3 V	I2C SDA / SPI CS signal
8	V_BCKP_1	I	3.3 V	Backup voltage extra input
7	SCL / SCK	I/O	3.3 V	I2C SCL / SPI SCK signal
6	GND	I		Common ground for casing, power and serial interfaces
5	TxD / MISO	I/O	3.3 V	UART TxD / SPI SDO (MISO) signal
4	VIN_LDOS	O		LDO input voltage measurement point
3	RxD / MOSI	I/O	3.3 V	UART RxD / SPI SDI (MOSI) signal
2	GND			Common ground for casing, power and serial interfaces
1	TIMEPULSE	O		Time pulse signal from the ZED-F9K (TP1)

2.5 10-pin rear connector

The 10-pin rear connector is used for updating the MCU firmware. See section 5.5 for more information.

2.6 Reset and safe boot buttons

The reset button on the front panel resets the unit.

The safe boot button is used to set the unit in the safe boot mode. In this mode, the receiver executes only the minimal functionality, such as updating new firmware into the flash. **USB communication is disabled** while in the safe boot mode.

To set the receiver in the safe boot mode:

- Press and hold the BOOT button.
- Press the RST button.
- Release the RST button.
- Release the BOOT button.

To use UART in the safe boot mode, a training sequence needs to be sent to the receiver. The training sequence is a transmission of two bytes (0x55 0x55) at the baud rate of 9600. Wait for at least 100 milliseconds before the interface is ready to accept commands.

2.7 I2C/SPI slide switch

The I2C/SPI slide switch should be kept in the I2C position to keep the I2C and UART interfaces enabled. The onboard MCU uses I2C to communicate with the receiver, so disabling I2C also disables the CAN interface.

For further information, contact u-blox technical support.

2.8 LED

On the front panel of the EVK unit, there is a single blue LED with the following functionality:

LED	Description
Solid blue	The device is powered on with no GNSS fix.
Flashing blue	The LED flashes one pulse per second during a GNSS fix. The time pulse signal is configurable, see the Interface description [2] for details.

2.9 Backup power supply

The back of the unit has a micro-USB connector for providing backup voltage for the receiver. See the ZED-F9K Integration manual [1] or the ZED-F9L Integration Manual [3] for more information about backup voltage. In addition, a battery holder for a CR2032 coin cell is available on the PCB.

⚠ CAUTION! Risk of data loss. If the backup power is interrupted, the receiver may lose all its calibration parameters. If you are using a power bank for backup power, ensure that the power supply is not interrupted due to low current intake.

2.10 NEO-D9S module

The device includes a NEO-D9S module for providing L-band corrections to the ZED-F9K module. The modules are connected via each module's UART2 port. The NEO-D9S can be connected to in u-center through either USB of UART. When connecting through UART, the NEO_UART_SEL pin must be pulled down to enable communication with the module.

For more information on how to use the NEO-D9S, refer to the C101-D9S application board User guide [4] and dedicated Application Note [6].

3 Getting started

This chapter works as a simple step-by-step guide for setting up the device and using it for evaluation in an automotive application using u-blox sensor fusion technology. The simplest setup uses Untethered Dead Reckoning (UDR) and relies solely on the ZED-F9K module.

For optimal sensor fusion performance, odometer input from the vehicle should be provided to the receiver to enable Automotive Dead Reckoning (ADR). For best overall performance, the setup should be extended to also include Real-time Kinematic (RTK) functionality, which is described in chapter 4.

The basic evaluation process consists of four simple steps: installation, calibration, testing and analysis. Follow the steps in this chapter to minimize errors leading to most common issues in performance.

3.1 Installation

The installation step consists of attaching the device and antenna to the test vehicle.

3.1.1 Mounting the device

Find a suitable location in the vehicle where the EVK can be properly attached. A good location for the EVK is in the trunk, close to the center of the rear axle of the vehicle.

Utilize appropriate hardware to firmly attach the EVK to the vehicle. The EVK must not be able to move relative to the vehicle's frame or encounter excessive vibrations. Do not attach the EVK to any moving parts of the vehicle's interior, such as a headrest or the rear-view mirror.

⚠ Sensor fusion performance will be impaired by changes in the orientation of the device.

3.1.2 Mounting the antenna

Place the provided GNSS antenna in a location with an unobstructed view of the sky, for example on the roof of the vehicle. For best performance, ensure that the antenna has contact to a ground plane which is at least 100–150 mm in diameter.

⚠ If the antenna is placed at a significant distance from the EVK, a position offset can be introduced which might affect the accuracy of the navigation solution. To compensate for the position offset, advanced configurations can be applied. Contact u-blox support for more information on advanced configurations.

3.1.3 Connecting the cables

1. Connect the GNSS antenna to the RF connector on the front panel of the device.
2. Connect the device to a PC via USB.

3.1.4 Configuring the receiver (optional)

The default configuration of the LAP firmware is usable for basic automotive applications. A custom configuration can be applied using u-center:

1. Open u-center.
2. Select the device with **Receiver > Connection > COMXX.**
3. Open the Messages View with **View > Messages View.**
4. Select the UBX-CFG-VALSET message.
5. Select the configuration item(s) with *Group* and *Key name.*
6. Modify the values and send the message to modify the configuration.

☞ Refer to the ZED-F9K documentation ([1], [2]) for a full description of the receiver configuration.

3.2 ADR setup (optional)

To evaluate the ADR performance and features of the LAP firmware, include the following steps in the setup.

3.2.1 Providing odometer input

ADR requires odometer input from the vehicle, that is, wheel ticks or speed, and direction. The following options are available for supplying the odometer input to the receiver:

- Hardware interface: wheel tick and direction pins.
- Software interface: UBX-ESF-MEAS messages.
- CAN interface: CAN_H and CAN_L pins.

Only one of the options above may be used at a time. Make the following connections based on the selected option:

A. If using the hardware interface, connect the Wheel Tick and FWD pins to the corresponding pins of the outputting sensor

B. If using the software interface, connect a serial interface (USB/UART) to the data provider

C. If using the CAN interface, connect the CAN high and CAN low signals of the CAN bus to the CAN_H and CAN_L pins in the front connector.

Refer to the ZED-F9K documentation ([1], [2]) for more information about providing odometer data. If using the configurable CAN interface, refer to chapter 0.

3.2.2 Configuring the device for ADR

The receiver can be configured with the UBX-CFG-VALSET messages. Consult the ZED-F9K documentation ([1], [2]) for more information about the configuration.

Configure the odometer sensor input depending on the used sensor:

A. If the wheel tick and direction pins on the front connector are used, enable the use of the wheel tick pin by setting the value for key ID CFG-SFODO-USE_WT_PIN to 1.

B. If using the CAN interface or the software interface, **disable the wheel tick pin**. Set the value for key ID CFG-SFODO-USE_WT_PIN to 0. See chapter 0 for instructions on configuring the CAN interface.

3.3 Calibrating the receiver

Before the receiver can operate in sensor fusion mode, it needs to gather calibration information from the movements of the vehicle. Although the calibration process is eventually completed during normal driving, it can be considerably accelerated by doing a calibration drive prior to the actual testing.

Follow these steps to perform the accelerated calibration procedure:

1. Drive to an open area, such as a parking lot, with good GNSS signal conditions.
2. With the car stationary, power on the EVK and wait for a valid 3D GNSS fix.
3. Remain stationary until the IMU status in ESF-STATUS shows "INITIALIZED".
4. Drive a figure-of-eight pattern until the alignment status in ESF-ALG shows "COARSE".
   The progress of the calibration can be monitored in u-center with the UBX-ESF-ALG and UBX-ESF-STATUS messages.
5. Drive straight at a min speed of 40 km/h until the INS status in the ESF-STATUS shows "INITIALIZED".

Once the calibration is at a sufficient level, the receiver starts using the sensors in navigation and the fusion filter status in the ESF-STATUS shows "FUSION". The receiver keeps continuously calibrating the sensors in the background to improve the quality of the solution.

For optimal performance, repeat the figure-of-eight pattern until the ESF-ALG shows "FINE". Before the actual testing, continue driving curves and straight segments until all sensors in the ESF-STATUS report "CALIBRATED".

☞ Refer to the ZED-F9K Integration manual [1] for more information about the sensor calibration.

3.4 Testing the receiver

The device is now ready for actual test drives. For replaying and analyzing the test drives afterwards, record the data into log files with u-center.

To collect a proper log file with sufficient information:

1. Open u-center.
2. Select the device with **Receiver > Connection > COMXX.**
3. Enable UBX messages according to what needs to be monitored.
4. Enable debug messages with the debug message button (Figure 1). This step is optional, but necessary for investigation of issues.
5. Start recording with the record button (Figure 1).
6. When prompted to poll the receiver configuration, select the correct receiver generation, and click "Yes".
7. Perform the test drive.
8. To stop recording, click the eject button (Figure 1). The log file will be saved automatically.

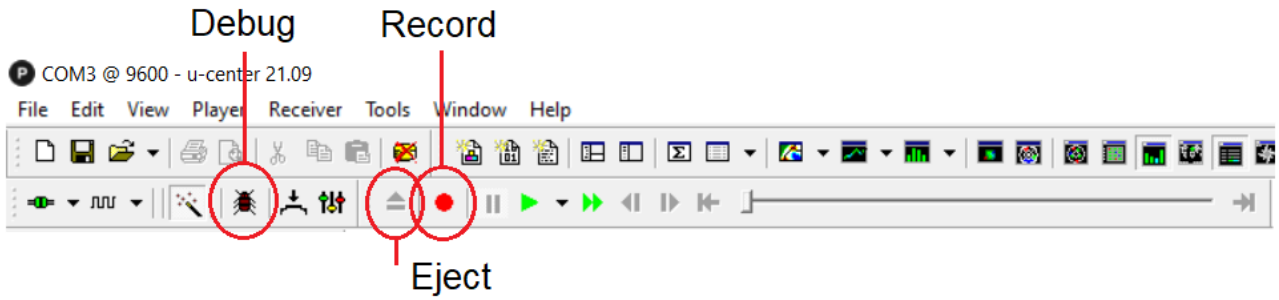


Figure 1: u-center log recording controls

3.5 Analyzing the log files

After collecting data over the test drives, u-center can be used to replay and analyze the logs in several ways:

- Checking the general receiver status
- Monitoring data in individual messages
- Using the chart view to monitor certain parameters over time

To replay a log,

1. Open u-center.
2. Open a log using **File > Open...**
3. Use the log playback controls (Figure 2) to play, pause and move the current time in the log file.
4. Open different views from the **View** menu.

☞ Refer to the u-center User guide [7] for more information about its features.

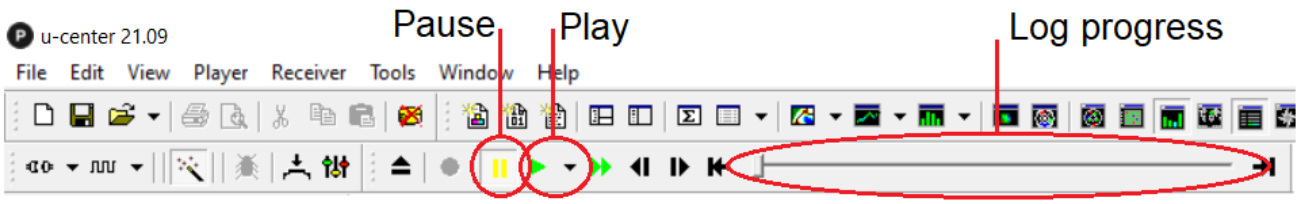


Figure 2: u-center log playback controls

4 RTK setup

To achieve accurate RTK performance, the receiver needs a constant stream of correction data, which can be obtained from correction data providers via NTRIP or MQTT protocols. Application software is needed to fetch the data from the provider's server and send it to the receiver through serial ports. This chapter explains how to use NTRIP and MQTT client on u-center to monitor the RTK status of the receiver.

4.1 Setting up NTRIP client in u-center

There are paid and free NTRIP services. RTK2go is a free community NTRIP service that hobbyists and early prototyping can use, as it provides correction data streams from other users. However, for commercial or production-grade applications, it is recommended to use more reliable commercial NTRIP services.

Start using the u-center NTRIP client with the following steps:

1. Open u-center and connect to the receiver via **Receiver > Connection.**
2. Open the NTRIP client settings from **Receiver > NTRIP Client.**
3. Fill in the NTRIP caster settings fields.
4. To fetch the available mount points from the service, click the **Update source table** button.
5. Select the correct mount point from the dropdown menu.
6. Press **OK** to start the NTRIP client.

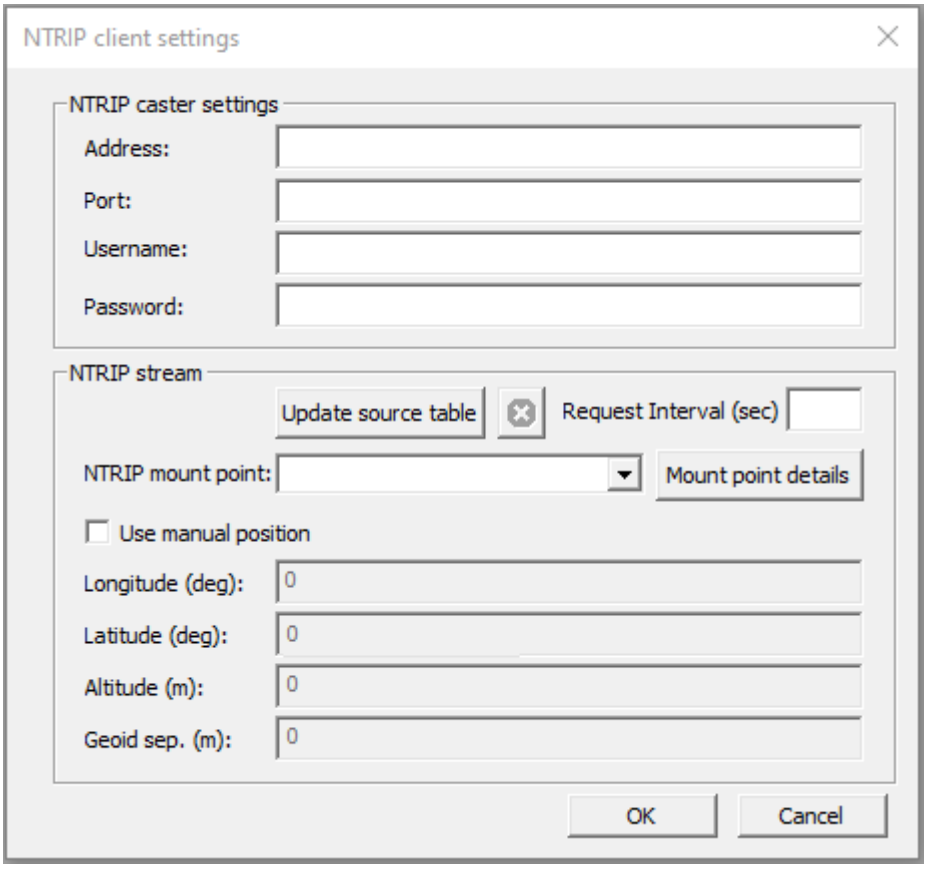


Figure 3: u-center NTRIP client

The status bar at the bottom of the u-center window shows the service status for monitoring and debugging. The connection symbol turns green when authentication is successful, and the service is connected.

Figure 4: u-center status bar

To see more information about the NTRIP client's operation, click the connection symbol. This will open the NTRIP Log window.

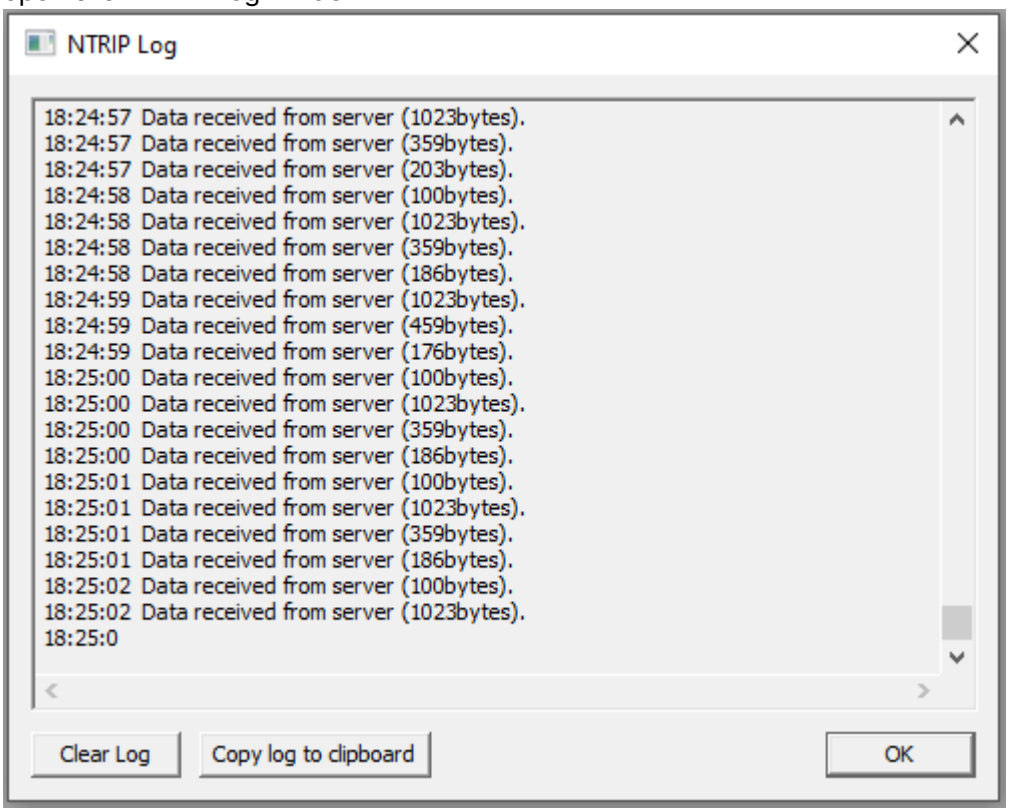


Figure 5: NTRIP client log

4.2 Setting up MQTT client in u-center

An OASIS standard messaging protocol for the Internet of Things (IoT), MQTT provides a reliable, robust, and secure messaging protocol for the IoT devices and applications. It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. (Source: https://mqtt.org/)

PointPerfect is a GNSS augmentation service that utilizes the industry standard SPARTN messaging format to enable high accuracy with high precision GNSS receivers. The SPARTN format facilitates the transfer of GNSS correction data with exceptional efficiency.

MQTT serves as the fundamental transport mechanism for various components of the service including authentication, ancillary services (such as AssistNow), service key delivery, and the service itself. To support this, u-center incorporates an MQTT client.

Start using the u-center MQTT client with the following steps:

1. Open u-center and connect to the receiver via **Receiver > Connection**.
2. Open the MQTT client settings from **Receiver > MQTT Client.**
3. In the Thingstream portal, download the u-center config file associated with your device and region of operation and add it as the “JSON config file”. For more information on how to sign in to Thingstream to get the u-center configuration file, visit PointPerfect getting started guide

4. Check the box to “Subscribe to key topic”.
5. Check the box to “Subscribe to AssistNow topic”.
6. Check the box to “Subscribe to data topic”.
7. Select the data topic using the drop-down menu suitable for the region of operation.
8. Press **OK** to start the MQTT client.

The status bar at the bottom of the u-center window provides information on the status of the service for monitoring and debugging purposes. When the authentication is successful and the service is connected, the connection symbol turns green.

Figure 6: Bottom Status bar of u-center

To view more details of the MQTT client’s operation, click the connection symbol. The MQTT Log window is displayed.

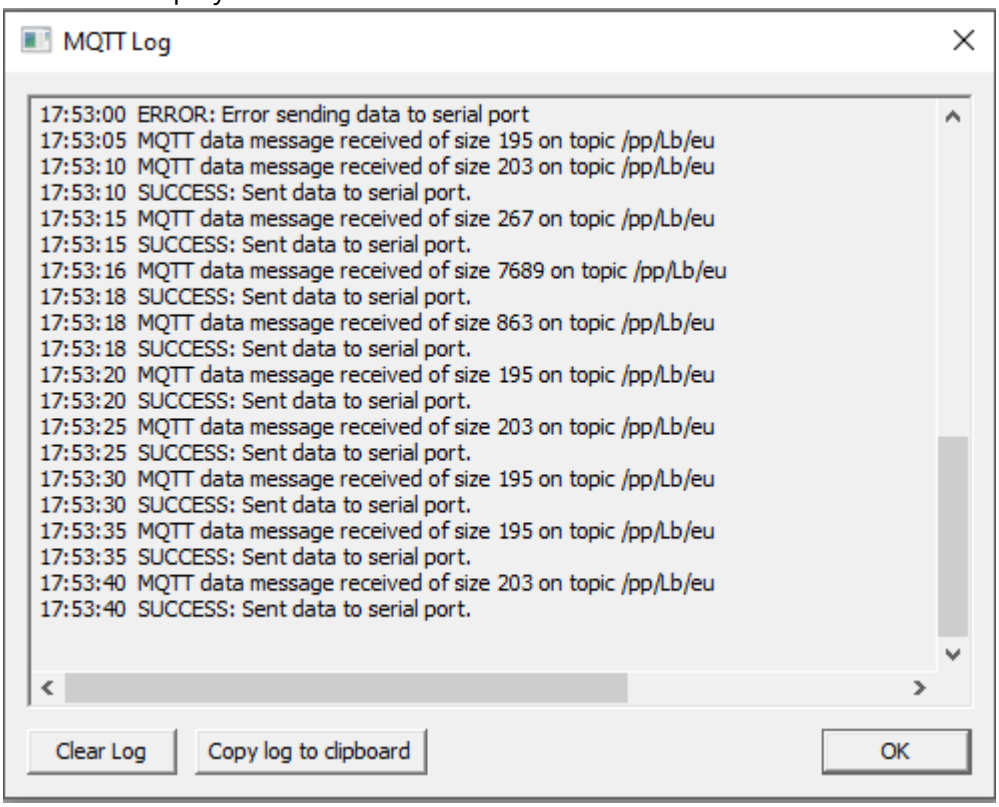


Figure 7: MQTT client log

4.3 Monitoring RTK status

Make sure the receiver is getting correction data by monitoring the RTK status in u-center (see Figure 8):

- In the general information view (**View > Docking Windows > Data**), the RTK status is shown in the Fix status field as “Float” or “Fixed” if RTK is used.
- Alternatively, in the UBX-NAV-PVT message view, the RTK status is shown in the Carrier Range Status field as “Float” or “Fixed” if RTK is used.

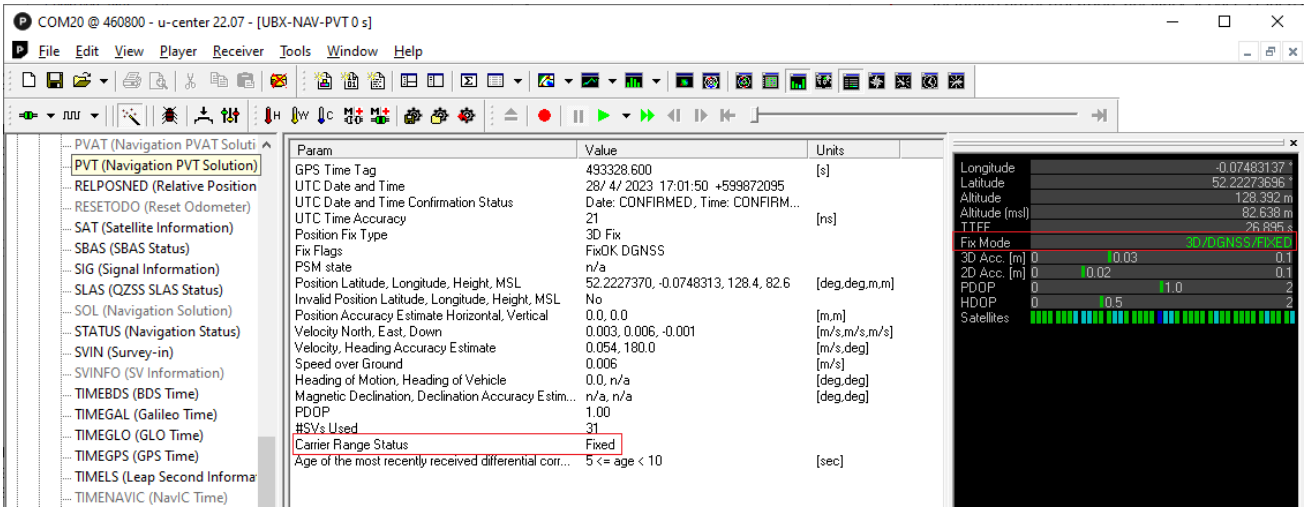


Figure 8: RTK status in u-center

In open sky scenarios, if the receiver achieves a fixed state in less than two minutes, it indicates that the receiver, antenna, and correction service are compatible and functioning properly.

5 Configurable CAN interface

☞ This chapter only applies to the ADR operating mode and can be ignored for UDR.

The device has a configurable high-speed CAN (ISO 11898-2) interface. The on-board MCU converts the configured CAN messages into UBX-ESF-MEAS messages which are sent to the receiver via I2C.

5.1 Valid configurations

The CAN interface supports the following configurations:

- Single tick from VRP + direction
- Wheel ticks from both rear wheels + direction
- Speed from VRP + direction
- Speed from both rear wheels + direction

See appendix B for example configurations.

5.2 Configuring the interface

Communication with the MCU can be established via UART. Connect the front connector pin SEL_MCU_N to ground to enable the MCU communication.

The MCU UART runs at baud rate of 115200.

The following messages are supported:

- CONFIG GET – Reports the current CAN configuration.
  - Hex string: 0x43 0xa2 0x10 0x00 0x10 0x20
- CONFIG CLEAR – Deletes the current CAN configuration.
  - Hex string: 0x43 0xa2 0x12 0x00 0x12 0x24
- CONFIG SET – Sends a configuration for one data field.
  - Hex string: generate with the tool

Sending the commands to the MCU can be done through a terminal program. We recommend using RealTerm [8].

5.3 Configuration message tool

The C100 MSG browser-based tool (Figure 9) generates C100 MCU configuration messages for the configurable CAN feature. The tool can be downloaded from the u-blox website [9]. It can run entirely locally without an internet connection.

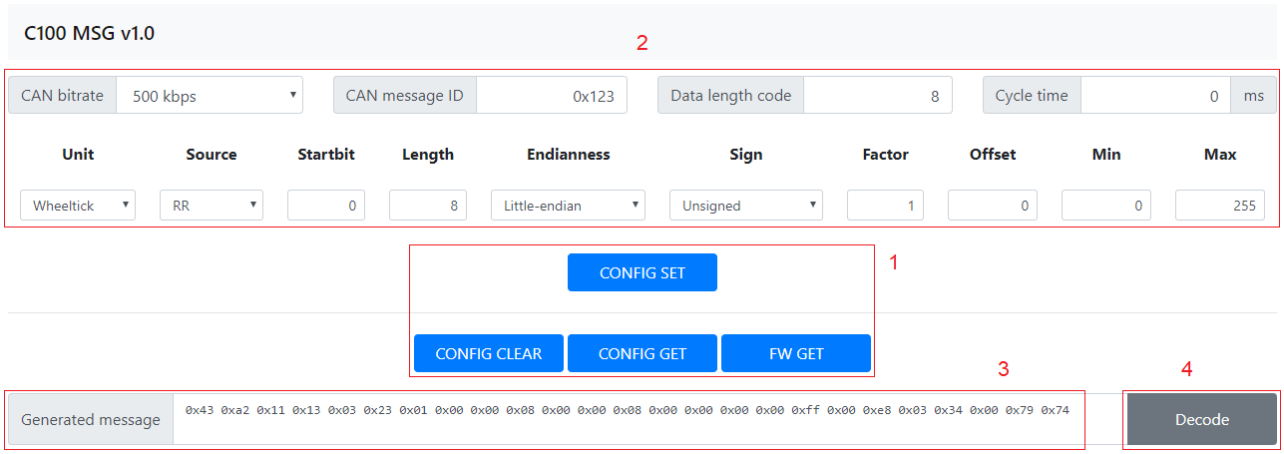


Figure 9: C100 MSG tool

The numbers in the list below refer to Figure 9:

- **1:** Select the blue buttons in the middle to generate messages.
- **2:** Fill these fields for CONFIG SET messages.
- **3:** The generated message is displayed in the text field at the bottom. It is automatically copied to the clipboard.
- **4:** Use the decode button to parse the contents of a message pasted in the text field (3).

☞ Ensure that the version number of the tool matches the MCU firmware version. Compatibility between versions is not guaranteed.

5.3.1 Configuration parameters

The following fields are required to generate a CONFIG SET message:

- **CAN bitrate**: bitrate of the CAN bus
- **CAN message ID**: ID of the message containing the wanted data
- **Data length code**: number of bytes in the CAN message
- **Cycle time**: time between consecutive messages
- **Unit**: the unit of measurement for the data
- **Source**: rear-left, rear-right wheel, etc.
- **Startbit**: index of the LSB of the value field within the CAN message
- **Length**: the bit-length of the value field
- **Endianness**: Big-endian (Motorola) or Little-endian (Intel)
- **Sign**: value is signed or unsigned
- **Factor**: scaling factor representing the value of one bit in the selected unit
- **Offset**: positive offset which shifts the zero point of the raw value
- Min/Forward:
  - Wheel tick and speed – sets the minimum value. Values smaller than this are discarded.
  - Direction – represents the value indicating forward movement
- Max/Backward:
  - Wheel tick and speed – sets the maximum value. Values greater than this are discarded.
  - Direction – represents the value indicating backward movement

5.4 Configuration process

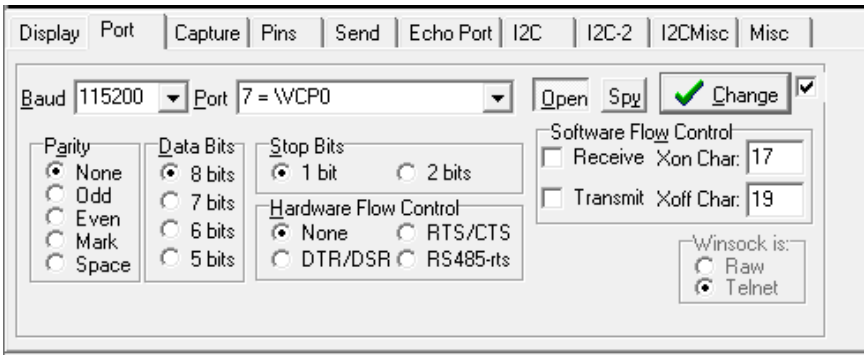
Follow these steps to configure the CAN interface:

5.4.1 Connections

1. Connect the pin SEL_MCU_N to the GND pin.
2. Connect a PC to the MCU via RS-232 cable or the front connector UART pins.

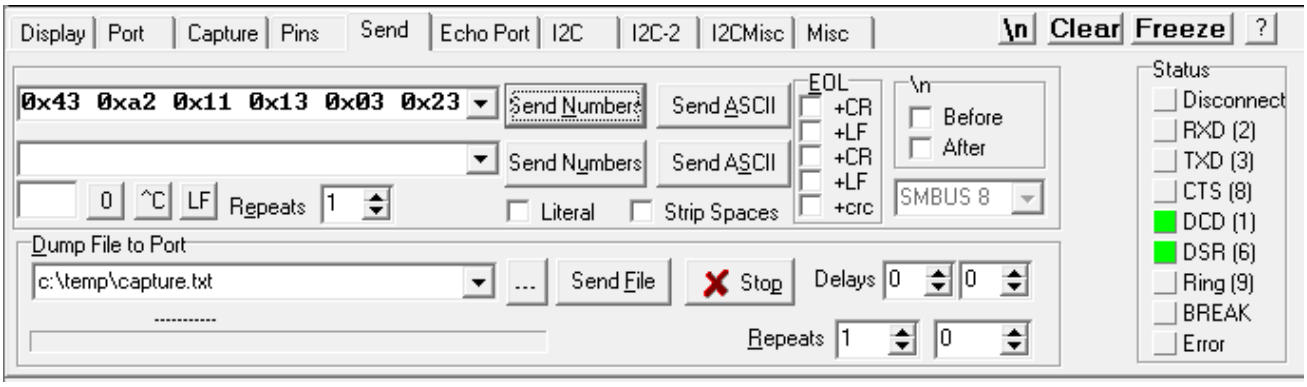
5.4.2 RealTerm

3. Select the port associated with the UART connection in the **Port** tab.
4. Set baud rate to 115200.
5. Apply changes by selecting the **Change** button. See the figure below.



Power on the device. The following startup message is displayed in the terminal window:

6. Set up the configurable CAN feature:
   6.1. Open the RealTerm **Send** tab.
   6.2. Generate CONFIG SET message(s) in the MSG tool.
   6.3. Copy and paste a CONFIG SET message into the text field.
   6.4. Send the message by selecting the **Send Numbers** button.

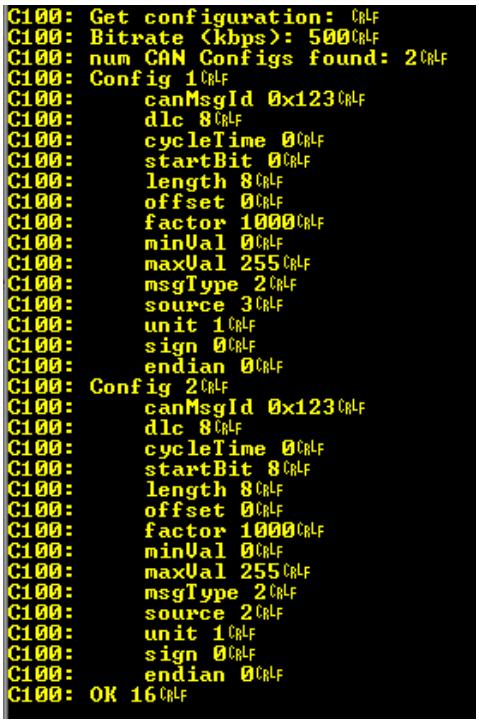


The following dialog is displayed when the configuration has been accepted:

When all configuration messages have been sent:

   6.5. Generate a CONFIG GET message.

6.6. Send the CONFIG GET message.
6.7. A dialog similar to the one shown below is displayed and can be used to validate the configurations.



☞ A configuration entry can be overwritten by sending a new CONFIG SET message with the same unit and source.

☞ All configuration entries can be deleted with the CONFIG CLEAR message.

5.5 Updating the MCU firmware

New MCU firmware and corresponding tool versions may be released e.g. to support new features or to increase the performance of the application. To update the firmware, the following equipment is required:

- Silicon Labs IDE or Flash Programming Utilities software [10], and
- USB debug adapter for 8-bit MCUs [11].

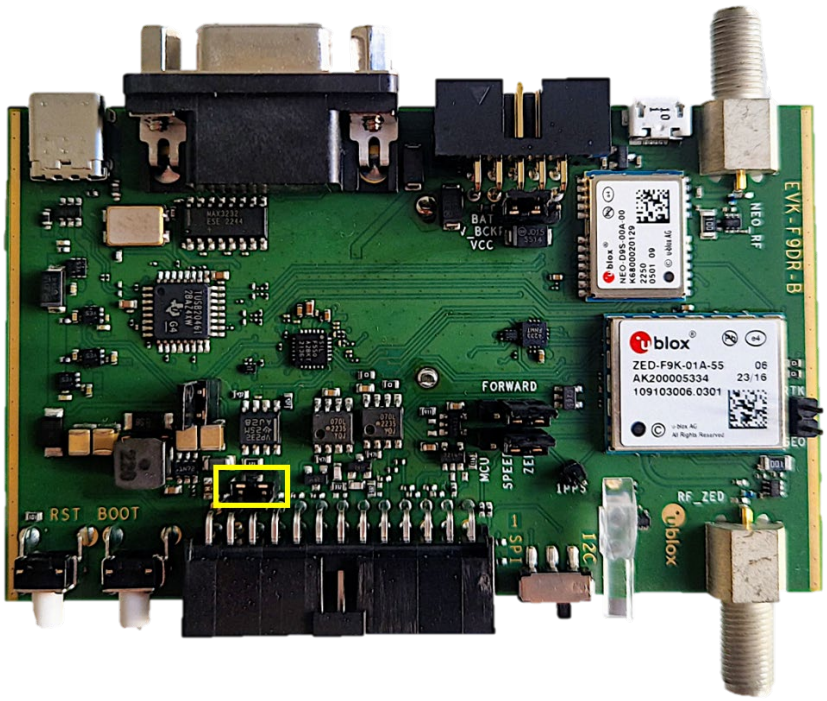
To flash the new firmware:

1. Power up the device.
2. Connect the debugger to the 10-pin rear connector.
3. If using the Silicon Labs IDE:
   a. Select **Debug > Connect** to connect the Debugger to the MCU.
   b. Select **Debug > Download** *object file* and input the correct file to the opened window.
   c. Select **Download** to start the flashing process.
4. If using Flash Programming Utilities, follow the instructions accompanying the software.
5. After the device is flashed, disconnect the debugger and reboot the device.
6. Confirm that the firmware version string matches by either checking what the MCU outputs during bootup, or by sending a FW GET command.

Appendix

A CAN termination

The CAN bus is terminated by including the jumper marked in the image below. The jumper is included by default. If the termination needs to be removed, open the enclosure and remove the jumper.



B CAN configuration examples

This appendix contains example CAN configurations. Each example uses the following settings for the CAN bus:

- CAN bitrate: 500 kbps
- CAN message ID: 0x123
- DLC: 8
- Cycle time: 0 ms

The example messages are compatible with firmware C100 v1.0.

B.1 Wheel tick configurations

B.1.1 Two rear-wheel ticks and direction

This configuration uses wheel ticks from two rear wheels and a separate direction signal. The configuration entries are described in the tables below.

Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
40	16	big-endian	unsigned	1	0	0	65535	tick	RR
56	16	big-endian	unsigned	1	0	0	65535	tick	RL
8	2	big-endian	unsigned	1	0	0	3	direction	direction

byte/bit	7	6	5	4	3	2	1	0
0								
1							msb	lsb
2								
3								
4	msb							
5								lsb
6	msb							
7								lsb

The following CONFIG SET messages are generated for this configuration:

- **RR**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x28 0x10 0x00 0x00 0x00 0x00 0xff 0xff 0xe8 0x03 0x34 0x01 0xa9 0xa8
- **RL**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x38 0x10 0x00 0x00 0x00 0x00 0xff 0xff 0xe8 0x03 0x24 0x01 0xa9 0x48
- **dir**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x08 0x02 0x00 0x00 0x00 0x00 0x03 0x00 0xe8 0x03 0x5f 0x01 0xab 0x01

B.1.2 Single tick and direction

This configuration uses single-tick data and a separate direction signal. The configuration entries are described in the tables below.

Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
32	16	big-endian	unsigned	1	0	0	65535	tick	combined
8	2	big-endian	unsigned	1	0	0	3	direction	direction

byte/bit	7	6	5	4	3	2	1	0
0								
1							msb	lsb
2								
3	msb							
4								lsb
5								
6								
7								

The following CONFIG SET messages are generated for this configuration:

- **tick**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x20 0x10 0x00 0x00 0x00 0x00 0xff 0xff 0xe8 0x03 0x44 0x01 0xb1 0x68
- **dir**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x08 0x02 0x00 0x00 0x00 0x00 0x03 0x00 0xe8 0x03 0x5f 0x01 0xab 0x01

B.2 Speed configurations

B.2.1 Two rear wheels and direction

This configuration uses speed from two rear wheels and a separate direction signal. The configuration entries are described in the tables below.

Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
52	12	big-endian	unsigned	0.1	0	0	409.6	km/h	RR
56	12	big-endian	unsigned	0.1	0	0	409.6	km/h	RL
8	2	big-endian	unsigned	1	0	0	3	direction	direction

byte/bit	7	6	5	4	3	2	1	0
0								
1							msb	lsb
2								
3								
4								
5	msb							
6				lsb	msb			
7								lsb

The following CONFIG SET messages are generated for this configuration:

- **RR**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x34 0x0c 0x00 0x00 0x00 0x00 0x00 0x10 0x64 0x00 0x39 0x01 0x41 0x58
- **RL**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x38 0x0c 0x00 0x00 0x00 0x00 0x00 0x10 0x64 0x00 0x29 0x01 0x35 0x68
- **dir**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x08 0x02 0x00 0x00 0x00 0x00 0x03 0x00 0xe8 0x03 0x5f 0x01 0xab 0x01

B.2.2 Single speed

This configuration uses a single-speed signal and a separate direction signal. The configuration entries are described in the tables below.

Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
24	8	little-endian	unsigned	1	0	0	255	mph	combined
8	2	little-endian	unsigned	1	0	0	3	direction	direction

byte/bit	7	6	5	4	3	2	1	0
0								
1							msb	lsb
2								
3	msb							lsb
4								
5								
6								
7								

The following CONFIG SET messages are generated for this configuration:

- **speed**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x18 0x08 0x00 0x00 0x00 0x00 0xff 0x00 0xe8 0x03 0x4a 0x00 0xa7 0xc0
- **dir**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x08 0x02 0x00 0x00 0x00 0x00 0x03 0x00 0xe8 0x03 0x5f 0x00 0xaa 0x00

B.2.3 Signed speed

This configuration uses a signed speed signal from both rear wheels. The configuration entries are described in the tables below.

Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
36	16	big-endian	signed	0.01	0	-327.68	327.67	km/h	RR
52	16	big-endian	signed	0.01	0	-327.68	327.67	km/h	RL

byte/bit	7	6	5	4	3	2	1	0
0								
1								
2					msb			
3								
4				lsb	msb			
5								
6				lsb				
7								

The following CONFIG SET messages are generated for this configuration:

- **RR**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x24 0x10 0x00 0x00 0x00 0x80 0xff 0x7f 0x0a 0x00 0x39 0x03 0xcb 0x03
- **RL**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x34 0x10 0x00 0x00 0x00 0x80 0xff 0x7f 0x0a 0x00 0x29 0x03 0xcb 0xa3

B.2.4 Offset speed

This configuration uses an offset speed signal from both rear wheels. The configuration entries are described in the tables below.

Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
16	16	little-endian	unsigned	0.01	50	-50	605.35	mph	RR
32	16	little-endian	unsigned	0.01	50	-50	605.35	mph	RL

byte/bit	7	6	5	4	3	2	1	0
0								
1								
2								lsb
3	msb							
4								lsb
5	msb							
6								
7								

The following CONFIG SET messages are generated for this configuration:

- **RR**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x10 0x10 0x88 0x13 0x78 0xec 0x77 0xec 0x0a 0x00 0x3a 0x00 0x19 0xb2
- **RL**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x20 0x10 0x88 0x13 0x78 0xec 0x77 0xec 0x0a 0x00 0x2a 0x00 0x19 0x52

C Step-by-step example

This step-by-step guide uses the example from section B.1.1.

Assumptions:

- User is familiar with u-center.
- USB is used for powering the device and for the u-center interface.
- Odometer sensor measurements are provided from the vehicle CAN bus via CAN_H and CAN_L pins on the front connector.
- UART RS-232 connector is used for the configurable CAN.
- RealTerm is used as the PC terminal application for the configurable CAN.

Connecting the device

1. Connect a cable between SEL_MCU_N and ground. This selects the MCU UART.
2. Connect the UART cable to the PC.
3. Connect the USB cable to the PC. Check that the blue light on the front panel is on.

Checking u-center

4. Open u-center.
5. Connect to the receiver with **Receiver > Connection > COMxx**
6. Verify that the connection is established. Poll UBX-MON-VER, and check that the FWVER is correct (LAP 1.30)
7. Update the receiver if necessary (**Tools** > **Firmware Update**).

Configuring the receiver

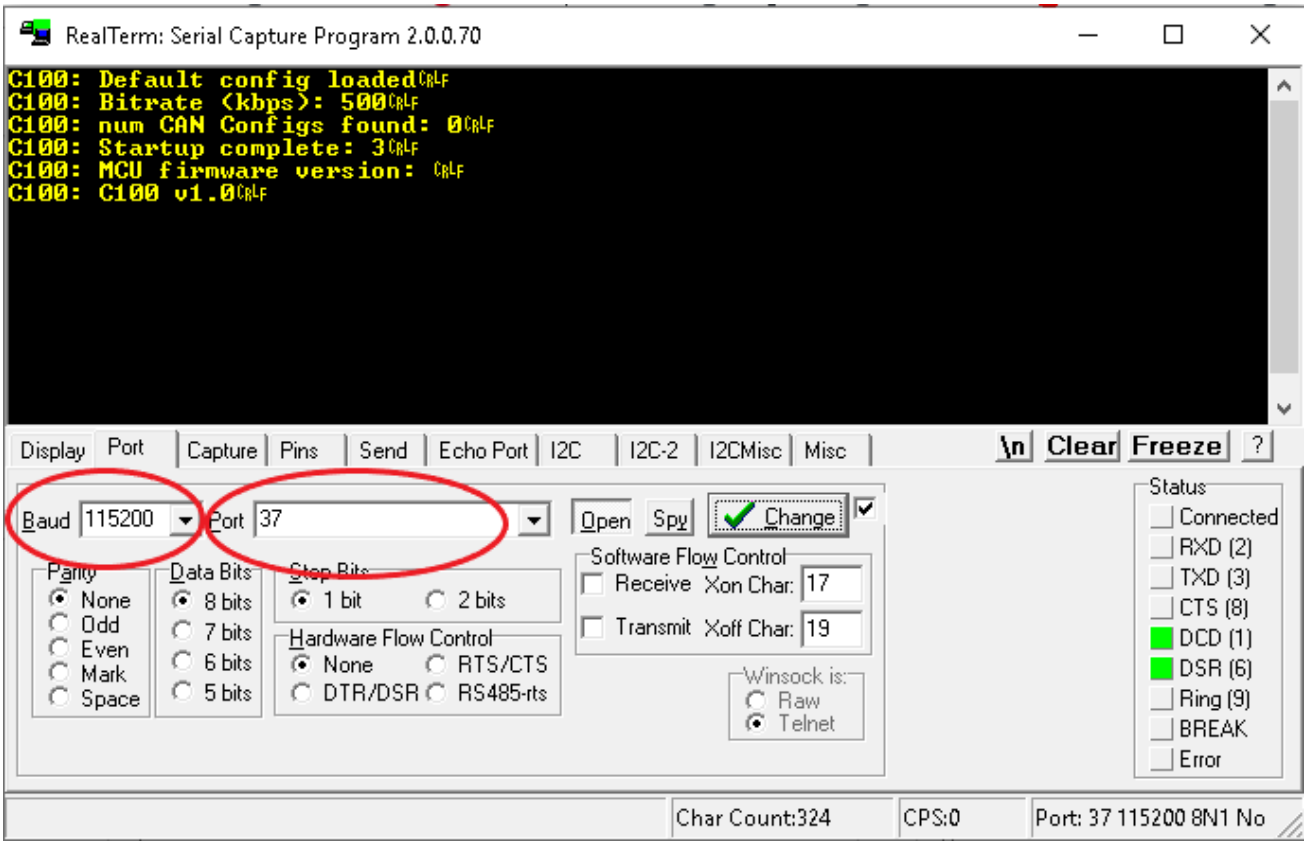
Receiver configuration can be set with UBX-CFG-VALSET message and the appropriate configuration keys.

1. Disable output messages on I2C (MCU is connected to I2C):
   - CFG-I2COUTPROT-UBX = false
   - CFG-I2COUTPROT-NMEA = false
2. Enable automatic alignment:
   - CFG-SFIMU-AUTO_MNTALG_ENA = true
3. (Optional) Enable priority navigation mode (10 Hz):
   - CFG-RATE-NAV_PRIO = 10
   - CFG-UART1-BAUDRATE = 115200
4. (Optional) Enable debug messages with the following commands based on the used serial protocol:
   a. On USB: B5 62 06 8A 3B 00 00 07 00 00 3E 02 91 20 01 3F 01 91 20 01 39 02 91 20 01 28 06 91 20 01 7A 02 91 20 01 0D 01 91 20 01 08 01 91 20 01 09 00 91 20 01 18 00 91 20 01 16 02 91 20 01 2F 02 91 20 01 5E 73
   b. On UART: B5 62 06 8A 3B 00 00 07 00 00 3C 02 91 20 01 3D 01 91 20 01 37 02 91 20 01 26 06 91 20 01 78 02 91 20 01 0B 01 91 20 01 06 01 91 20 01 07 00 91 20 01 16 00 91 20 01 14 02 91 20 01 2D 02 91 20 01 48 DF

⚠ CAUTION! Risk of data loss. Insufficient baud rate may result in data loss. Increase the UART1 baud rate when either the priority navigation mode or debug messages are enabled.

Configuring the CAN interface in RealTerm

5. Open RealTerm.
6. Select the **Port** tab.
7. Select the PC port corresponding to the MCU UART.
8. Set baud rate to 115200.
9. Restart the EVK.
10. MCU startup dialog should appear in the terminal.



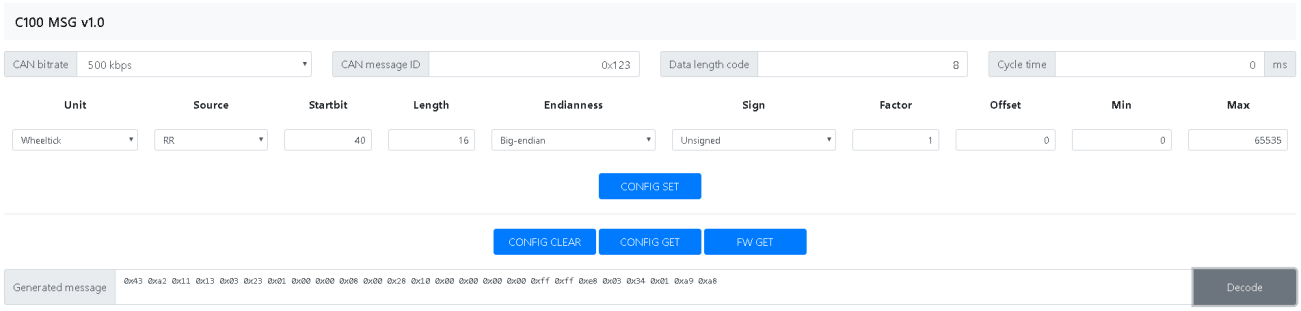
Generating the CONFIG SET strings with the MSG tool

From section B.1.1:

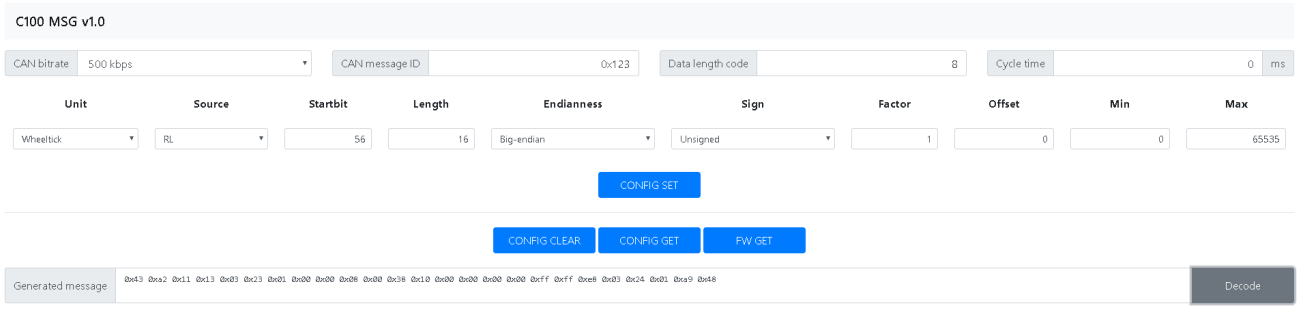
Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
40	16	big-endian	unsigned	1	0	0	65535	tick	RR
56	16	big-endian	unsigned	1	0	0	65535	tick	RL
8	2	big-endian	unsigned	1	0	0	3	direction	direction

11. Use the MSG tool to generate the CONFIG SET messages.

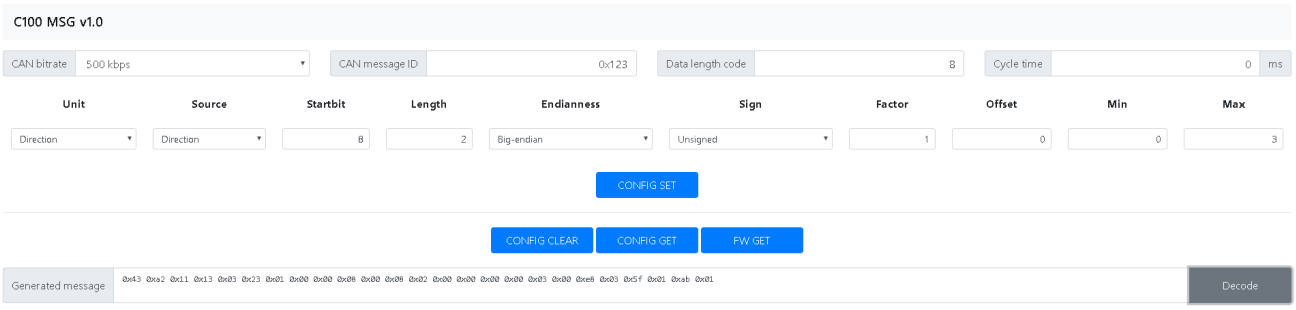
Rear-right wheel tick:



Rear-left wheel tick:



Direction:



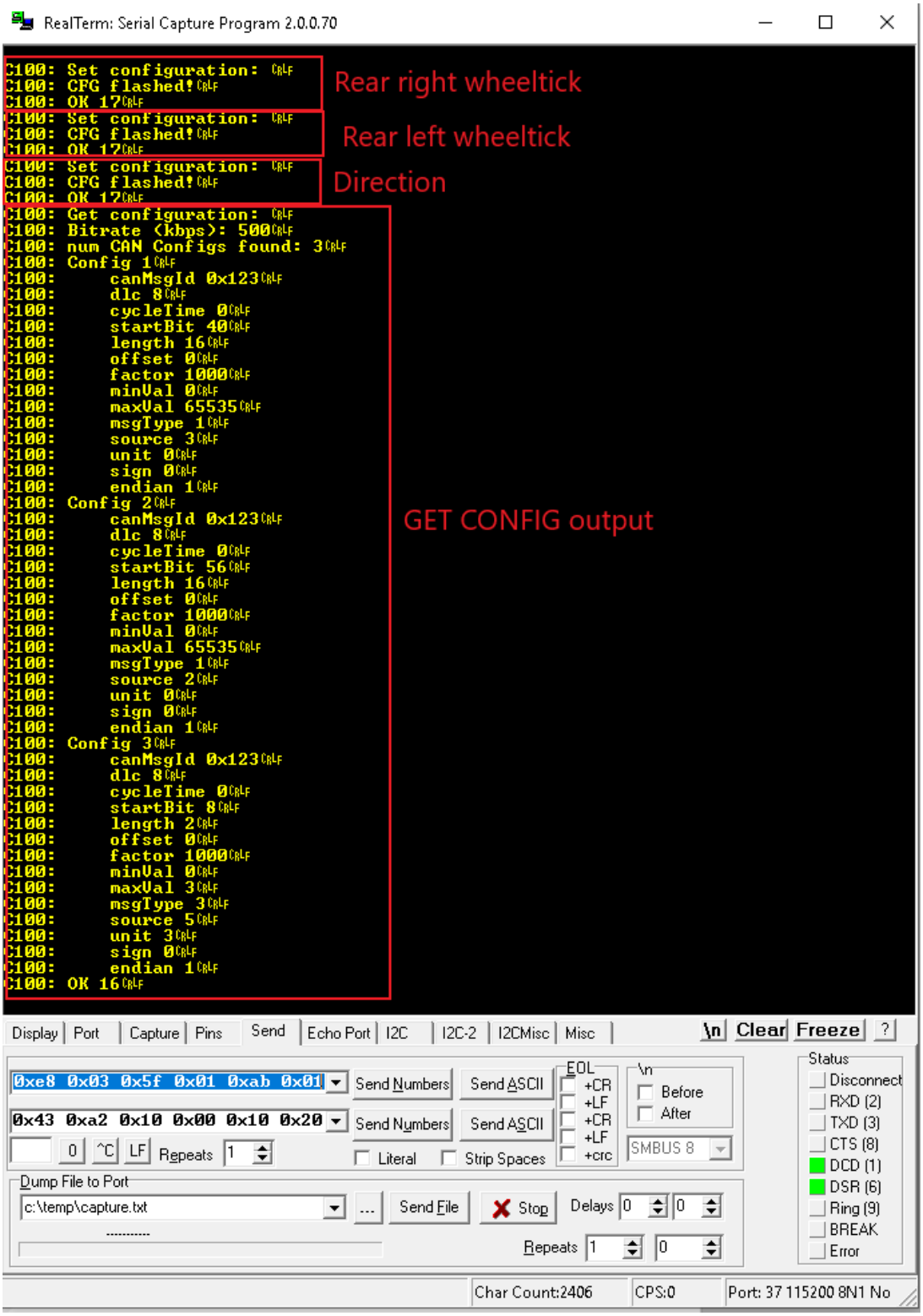
The following CONFIG SET messages are generated for this configuration:

- **RR**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x28 0x10 0x00 0x00 0x00 0x00 0xff 0xff 0xe8 0x03 0x34 0x01 0xa9 0xa8
- **RL**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x38 0x10 0x00 0x00 0x00 0x00 0xff 0xff 0xe8 0x03 0x24 0x01 0xa9 0x48
- **dir**: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x08 0x02 0x00 0x00 0x00 0x00 0x03 0x00 0xe8 0x03 0x5f 0x01 0xab 0x01

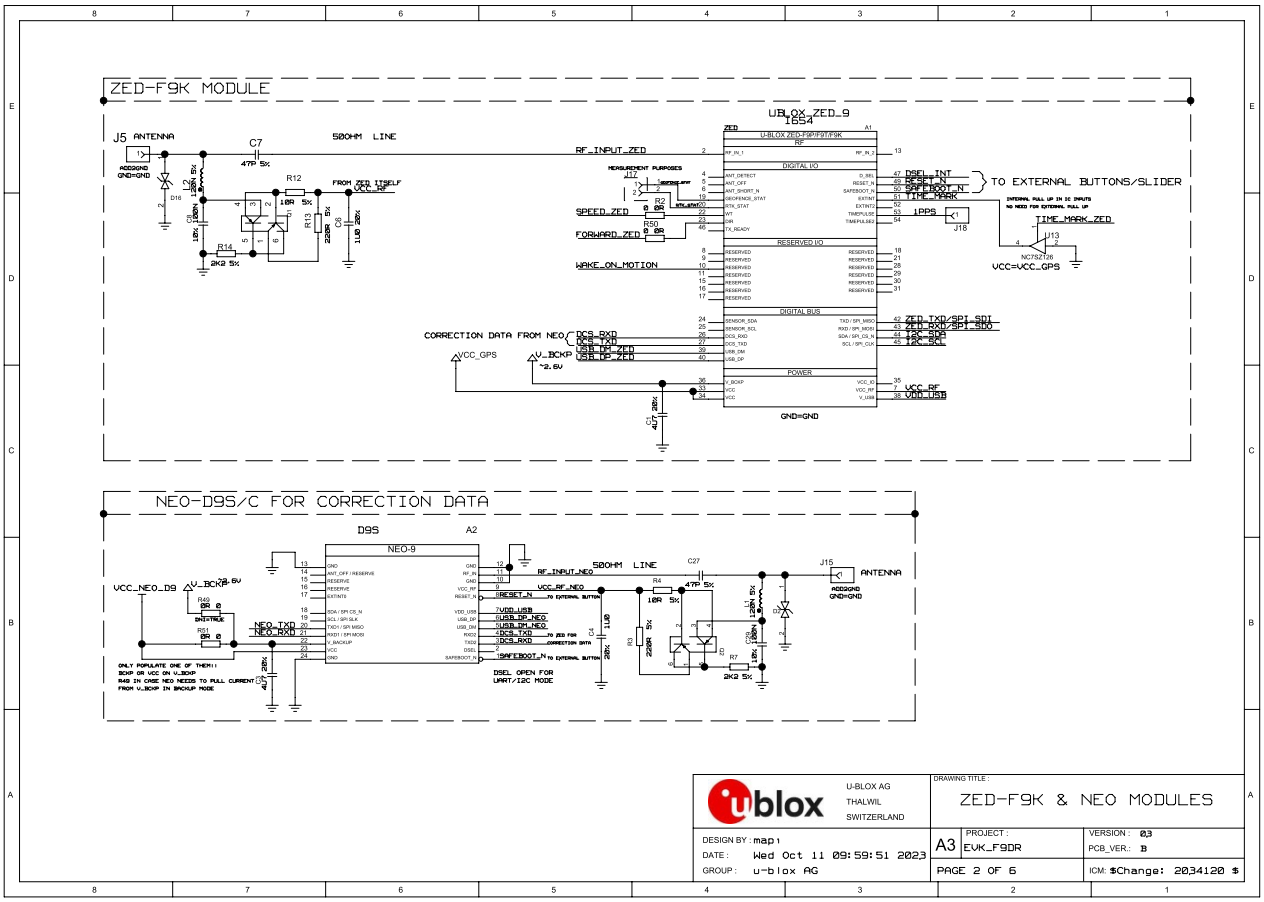
Sending CONFIG SET strings to MCU:

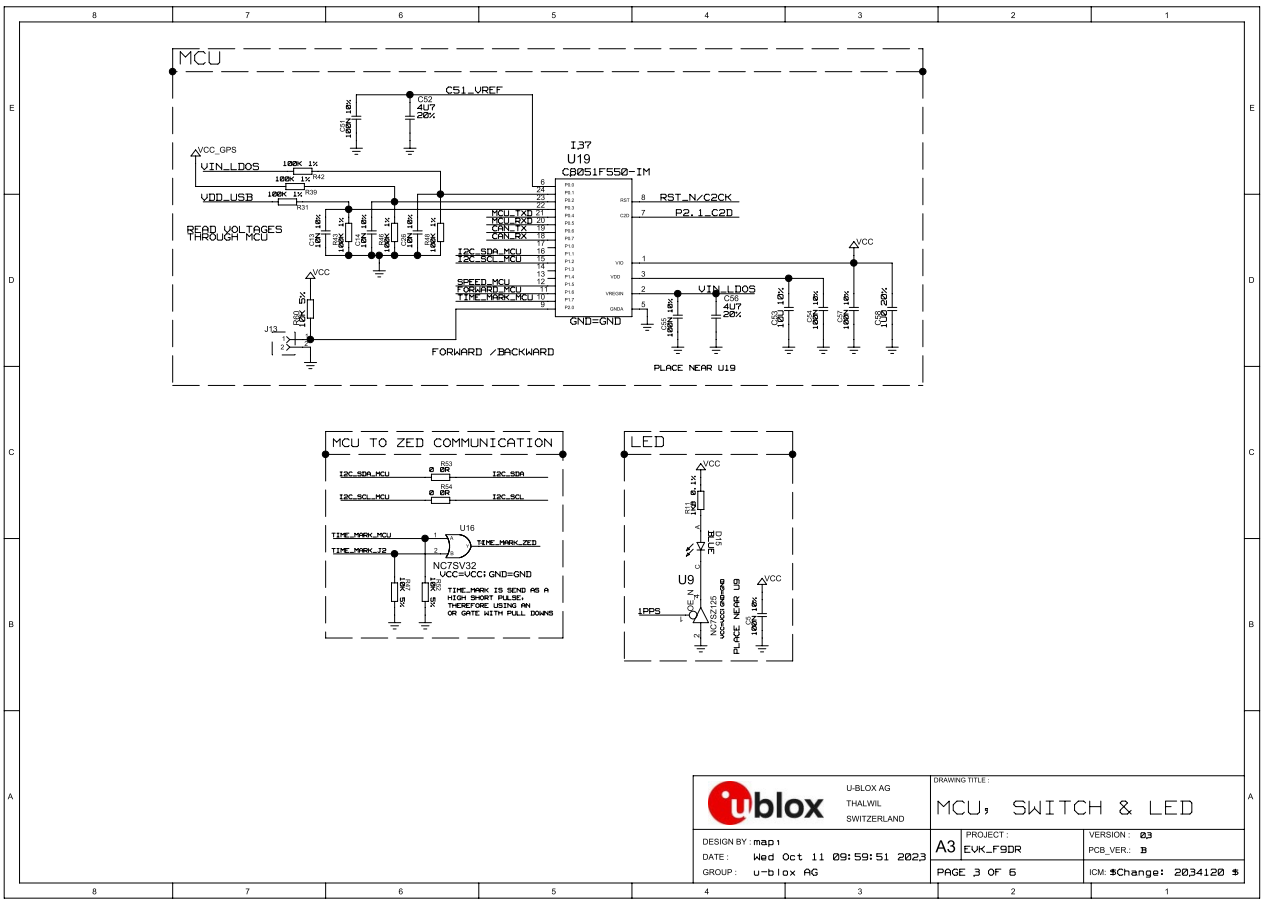
12. Open RealTerm.
13. Select the **Send** tab.
14. Copy and paste the rear-right wheel tick CONFIG SET string to the RealTerm text box.
15. Select the **Send Numbers** button.

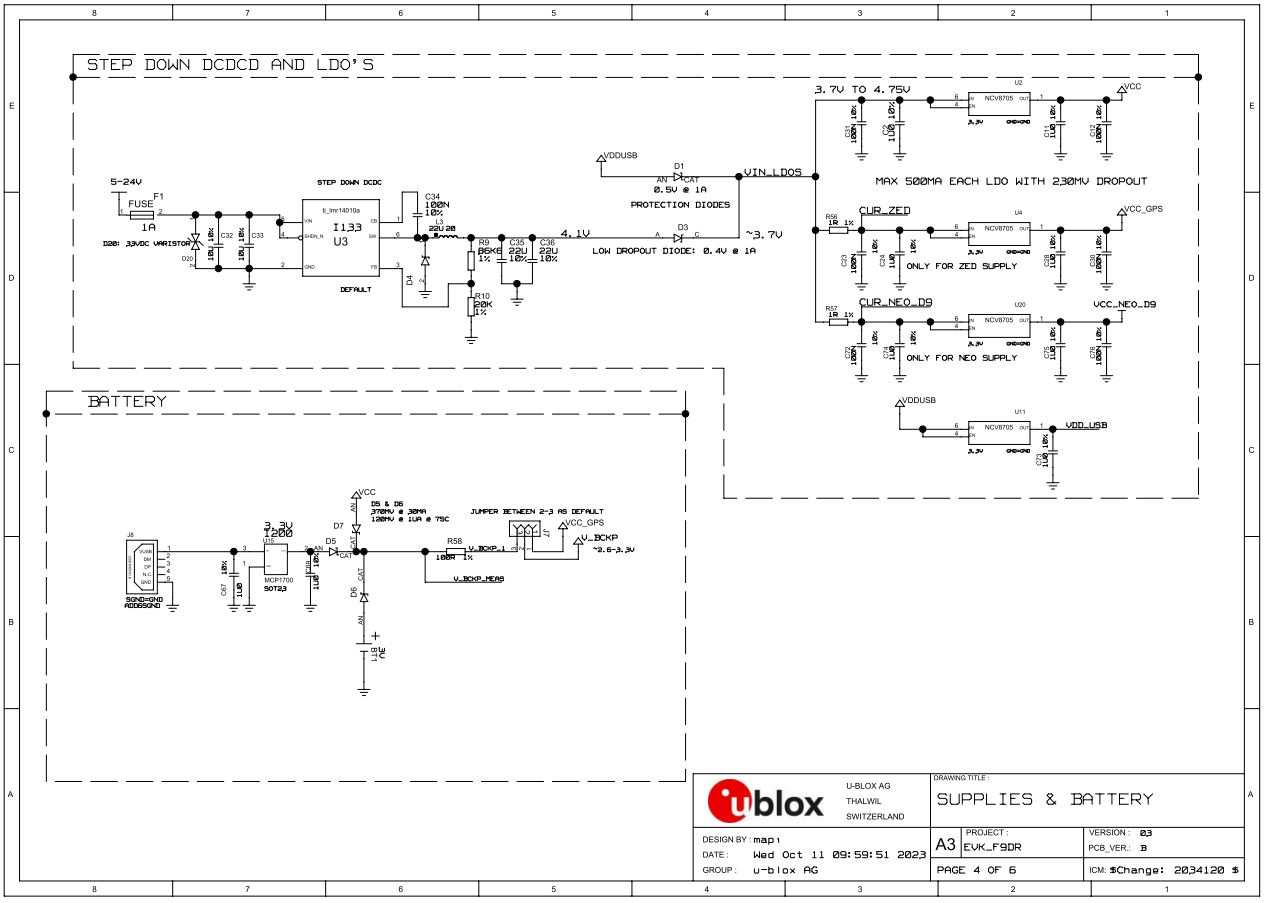
Verify configurations with CONFIG GET string, 0x43 0xa2 0x10 0x00 0x10 0x20.

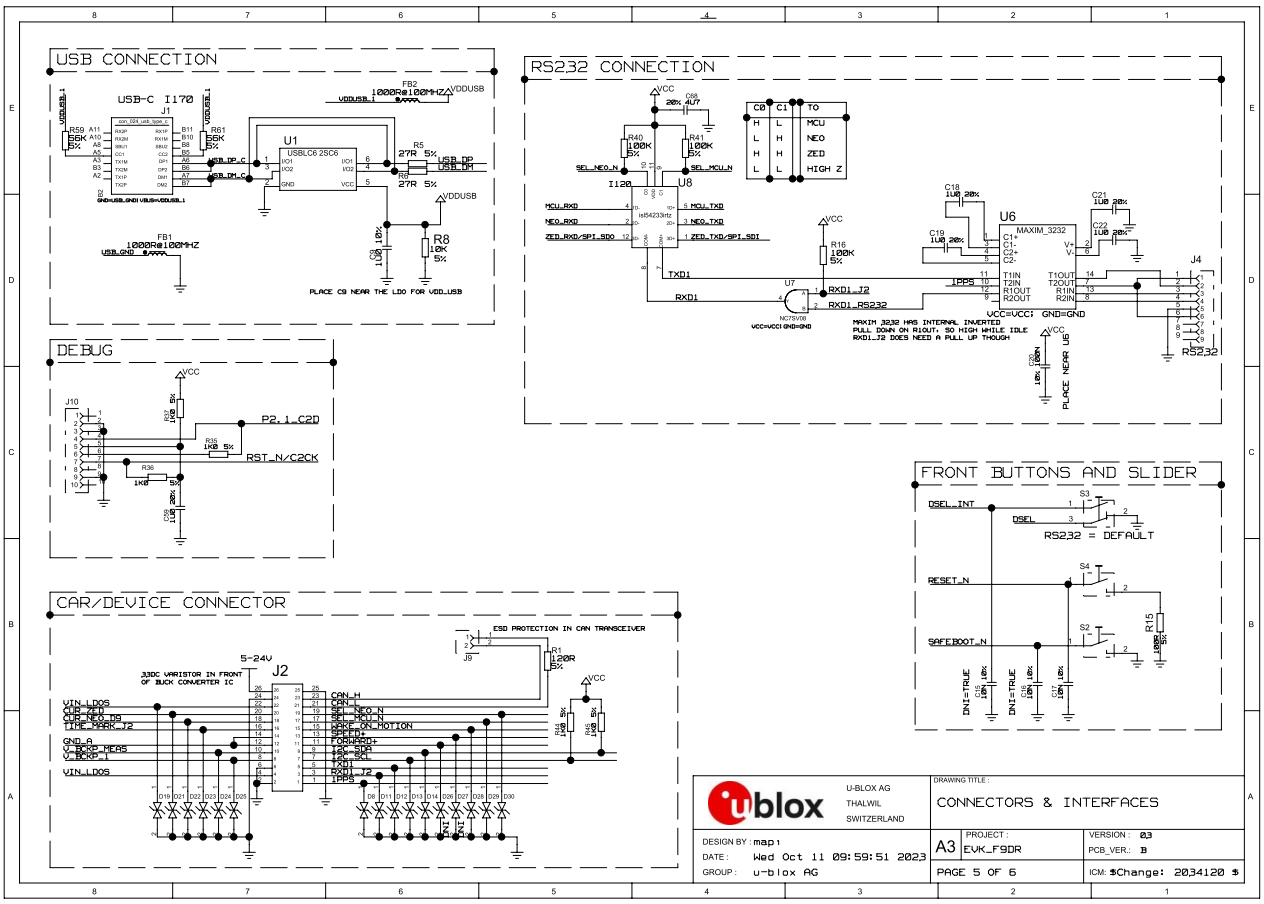

RealTerm: Serial Capture Program 2.0.0.70
C100: Set configuration:
C100: CFG flashed!
C100: OK 17
C100: Set configuration:
C100: CFG flashed!
C100: OK 12
C100: Set configuration:
C100: CFG flashed!
C100: OK 12
Rear right wheeltick
Rear left wheeltick
Direction
C100: Get configuration:
C100: Bitrate (kbps): 500
C100: num CAN Configs found: 3
C100: Config 1
C100: canMsgId 0x123
C100: dlc 8
C100: cycleTime 0
C100: startBit 40
C100: length 16
C100: offset 0
C100: factor 1000
C100: minVal 0
C100: maxVal 65535
C100: msgType 1
C100: source 3
C100: unit 0
C100: sign 0
C100: endian 1
C100: Config 2
C100: canMsgId 0x123
C100: dlc 8
C100: cycleTime 0
C100: startBit 56
C100: length 16
C100: offset 0
C100: factor 1000
C100: minVal 0
C100: maxVal 65535
C100: msgType 1
C100: source 2
C100: unit 0
C100: sign 0
C100: endian 1
C100: Config 3
C100: canMsgId 0x123
C100: dlc 8
C100: cycleTime 0
C100: startBit 8
C100: length 2
C100: offset 0
C100: factor 1000
C100: minVal 0
C100: maxVal 3
C100: msgType 3
C100: source 5
C100: unit 3
C100: sign 0
C100: endian 1
C100: OK 16
GET CONFIG output
Display
Port
Capture
Pins
Send
Echo Port
I2C
I2C-2
I2CMisc
Misc
\n
Clear
Freeze
?
0xe8 0x03 0x5f 0x01 0xab 0x01
Send Numbers
Send ASCII
0x43 0xa2 0x10 0x00 0x10 0x20
Send Numbers
Send ASCII
EOL
+CR
+LF
+CR
+LF
+crc
\n
Before
After
SMBUS 8
Repeats
Literal
Strip Spaces
Dump File to Port
c:\temp\capture.txt
Send File
Stop
Delays
Repeats
Status
Disconnect
RXD (2)
TXD (3)
CTS (8)
DCD (1)
DSR (6)
Ring (9)
BREAK
Error
Char Count:2406
CPS:0
Port: 37 115200 8N1 No

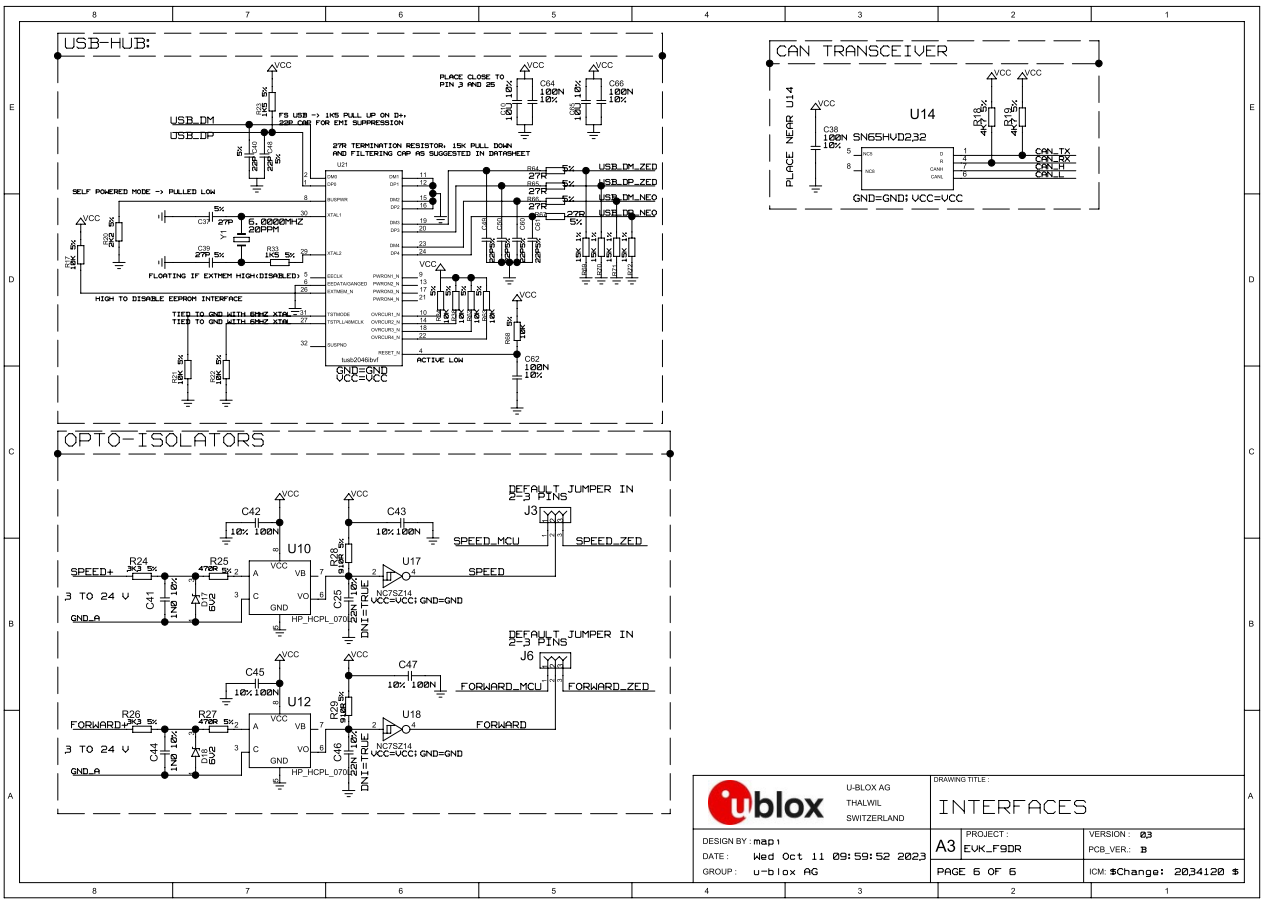
D Schematic


ZED-F9K MODULE
NEO-D9S/C FOR CORRECTION DATA
U-BLOX AG
THALWIL
SWITZERLAND
ZED-F9K & NEO MODULES
EVK_F9DR
PAGE 2 OF 6


MCU
C51_VREF
VCC_GPS
VIN_LDOS
VDD_USB
READ VOLTAGES THROUGH MCU
I37
U19
CB051F550-IM
RST_N/C2CK
P2.1_C2D
MCU_TXD
MCU_RXD
CAN_TX
CAN_RX
I2C_SDA_MCU
I2C_SCL_MCU
SPEED_MCU
FORWARD_MCU
TIME_MARK_MCU
GND=GND
VCC
J13
FORWARD /BACKWARD
PLACE NEAR U19
MCU TO ZED COMMUNICATION
I2C_SDA_MCU
I2C_SCL_MCU
I2C_SDA
I2C_SCL
TIME_MARK_MCU
TIME_MARK_ZED
U16
NC7SV32
VCC=VCC; GND=GND
TIME_MARK IS SEND AS A HIGH SHORT PULSE. THEREFORE USING AN OR GATE WITH PULL DOWNS
LED
U9
1PPS
PLACE NEAR U9
U-BLOX AG
THALWIL
SWITZERLAND
DRAWING TITLE :
MCU, SWITCH & LED
DESIGN BY : map1
DATE : Wed Oct 11 09:59:51 2023
GROUP : u-blox AG
A3
PROJECT : EVK_F9DR
VERSION : 03
PCB_VER.: B
PAGE 3 OF 6
ICM: $Change: 20341120 $


STEP DOWN DCDCD AND LDO'S
5-24V
FUSE
1A
D20: 33VDC VARISTOR
STEP DOWN DCDC
I133
U3
DEFAULT
VDDUSB
D1
0.5V @ 1A
PROTECTION DIODES
VIN_LDOS
D3
LOW DROPOUT DIODE: 0.4V @ 1A
~3.7V
4.1V
3.7V TO 4.75V
MAX 500MA EACH LDO WITH 230MV DROPOUT
CUR_ZED
ONLY FOR ZED SUPPLY
CUR_NEO_D9
ONLY FOR NEO SUPPLY
VCC_GPS
VCC_NEO_D9
VDD_USB
BATTERY
D5 & D6
370MV @ 30MA
120MV @ 1UA @ 75C
JUMPER BETWEEN 2-3 AS DEFAULT
V_BCKP
~2.6-3.3V
V_BCKP_MEAS
U-BLOX AG
THALWIL
SWITZERLAND
DRAWING TITLE :
SUPPLIES & BATTERY
DESIGN BY : map1
DATE : Wed Oct 11 09:59:51 2023
GROUP : u-blox AG
A3
PROJECT : EVK_F9DR
VERSION : 03
PCB_VER.: B
PAGE 4 OF 6
ICM: $Change: 20341120 $


USB CONNECTION
RS232 CONNECTION
DEBUG
FRONT BUTTONS AND SLIDER
CAR/DEVICE CONNECTOR
RS232 = DEFAULT
CONNECTORS & INTERFACES
U-BLOX AG
THALWIL
SWITZERLAND
DESIGN BY : map1
DATE : Wed Oct 11 09:59:51 2023
GROUP : u-blox AG
PROJECT : EVK_F9DR
PAGE 5 OF 6
VERSION : 03
PCB_VER.: B


USB-HUB:
CAN TRANSCEIVER
OPTO-ISOLATORS
INTERFACES
U-BLOX AG
THALWIL
SWITZERLAND
DESIGN BY : map1
DATE : Wed Oct 11 09:59:52 2023
GROUP : u-blox AG
PROJECT : EVK_F9DR
PAGE 6 OF 6
VERSION : 03
PCB_VER.: B

Related documentation

[1] ZED-F9K Integration manual, UBX-20046189
[2] u-blox F9 LAP 1.30 Interface Description, UBX-22005157
[3] ZED-F9L Integration manual, UBX-23006075, NDA required
[4] u-blox F9 DBD 1.30 Interface Description, UBX-23006359, NDA required
[5] C101-D9C User guide, UBX-21009111
[6] NEO-D9S and ZED-F9 configuration, Application Note, UBX-22008160
[7] u-center user guide, UBX-13005250
[8] RealTerm Serial Terminal, https://realterm.sourceforge.io
[9] C100 MSG tool, https://content.u-blox.com/sites/default/files/2022-03/C100-MSG-v1.0.zip
[10] Silicon Labs 8-bit Microcontroller Software, https://www.silabs.com/products/development-tools/software/8-bit-8051-microcontroller-software
[11] Silicon Labs 8-bit USB Debug Adapter, https://www.silabs.com/development-tools/mcu/8-bit/8-bit-usb-debug-adapter

☞ For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

Revision history

Revision	Date	Comments
R01	13-Nov-2023	Initial release

Contact

u-blox AG

Address: Zürcherstrasse 68
8800 Thalwil
Switzerland

For further support and contact information, visit us at www.u-blox.com/support.