EVK-M9DR

Evaluation kit

User guide

Abstract

This document describes the structure and use of the EVK-M9DR evaluation kit and provides information for evaluating and testing the u-blox M9 single-band GNSS multi-mode dead reckoning technology.

UBX-21049360 - R01
C1-Public

Document information

Title	**EVK-M9DR**	
Subtitle	Evaluation kit	
Document type	User guide	
Document number	UBX-21049360	
Revision and date	R01	3-Feb-2022
Disclosure Restriction	C1-Public	

This document applies to the following products:

Product name	Type number	Firmware version	PCN reference
EVK-M9DR	EVK-M9DR-0-00	MDR 2.10	N/A

Contents

1 Introduction

EVK-M9DR can be used to test and evaluate the u-blox M9 single-band GNSS dead reckoning technologies. The device is equipped with the NEO-M9V module, and allows performance and feature evaluation of the following products when used in conjunction with the provided active antenna:

- NEO-M9V
- NEO-M9L
- UBX-M9140-KA-DR
- UBX-M9340-KB

The built-in USB interface provides both power supply and a high-speed communication interface. The device 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.

1.1 Highlights

- Multi-constellation GNSS
- Multi-mode dead reckoning (MDR)
- Configurable CAN interface
- Dedicated pins for wheel tick and direction inputs
- USB, UART, RS-232 connections
- Battery-backed RAM (BBR) through micro-USB
- Wake-on-Motion feature

1.2 Kit includes

- Application board with enclosure
- A 1-meter USB-C cable
- A 1.8-meter micro-USB cable
- Active L1 GNSS antenna with a 3 m cable

1.3 System requirements

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

1.4 Evaluation steps

Experience the performance of the u-blox NEO-M9L and NEO-M9V modules in four simple steps:

1. Set up
2. Calibrate
3. Test
4. Analyze

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 receiver or the onboard MCU. Selection of the UART connection is controlled by the SEL_MCU_N pin on the front connector: when the pin is low, the MCU is selected.

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

Flow control should **not** be used with the RS-232 port.

2.3 Antenna

The kit includes a u-blox active GNSS antenna with a 3-meter cable. There is a female SMA connector (RF IN) available on the front side of the unit for connecting the antenna.

2.4 14-pin front connector

The connector and its signals are described in the table below.

Pin no.	Pin name	I/O	Level	Description
14	VIN 5-24V	I	5 - 24 V	Power input – can be used in place of USB
13	GND			Common ground pin for case-work, power and serial interface connections
12	CAN_H	I		Connect to the vehicle CAN high wire (ISO 11898-2)
11	CAN_L	I		Connect to the vehicle CAN low wire (ISO 11898-2)
10	TIMEPULSE	O	-	Time pulse signal output
9	SEL_MCU_N	I	-	Pull-down signal for enabling UART communication with the MCU
8	WoM	O		Wake-on-motion signal output
7	Wheel Tick	I	5 - 24 V	Wheel tick pulse input
6	FWD	I	5 - 24 V	Direction of travel input
5	SDA			Reserved
4	SCL			Reserved
3	TxD	I/O	3.3 V	UART TxD
2	RxD	I/O	3.3 V	UART RxD
1	GND_A			Ground for wheel tick and direction signals

⚠ Leave the reserved pins open.

2.5 10-pin rear connector

This 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 safe boot mode. In this mode the receiver executes only the minimal functionality, such as updating new firmware into the SQI flash. **USB communication is disabled** while in safe boot mode.

To set the receiver in 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 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 switch must be kept at the I2C position to ensure correct operation of the device!

⚠ Contact u-blox technical support for assistance if required.

2.8 LED

On the front panel of the unit, a single blue LED may be configured to follow the receiver time pulse signal. If there is no GNSS fix, the LED will be lit without flashing.

2.9 Backup power supply

The back side of the unit has a micro-USB connector for providing backup voltage for the receiver. See the NEO-M9V Integration manual [1] for more information about backup voltage.

⚠ If a power bank is used for backup power, ensure that the power supply is not interrupted due to low current intake.

3 Getting started

This chapter works as a simple step-by-step guide for successfully setting up the device and using it for evaluation in a basic automotive application using untethered dead reckoning (UDR) technology.

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

☞ For other application types, some modifications to the installation step may be necessary. See chapter 4 for more information.

3.1 Installation

3.1.1 Mounting the device

Firmly attach the device to the vehicle to avoid any movement and vibration relative to the vehicle. The device must not be attached to any moving part of the interior of the vehicle, for example, a headrest or a rear-view mirror. A good location is in the trunk, close to the center of the rear axle of the vehicle.

⚠ Dead reckoning performance can be seriously 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, the roof of the vehicle. For best performance, ensure that the antenna has contact to a ground plane with a minimum of 100–150 mm diameter.

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 MDR 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 NEO-M9V documentation ([2], [3]) for more information about receiver configuration.

3.2 Calibration

Before the receiver can operate in dead reckoning mode, it needs to gather calibration information from the movements of the vehicle. Although the calibration process will complete eventually during normal driving, it can be accelerated by doing a calibration drive prior to actual testing. The calibration drive may speed up the evaluation process considerably.

The calibration drive must be done in an open area, such as a parking lot, under good GNSS signal conditions. Drive the vehicle to such a place and do the following steps. The progress of calibration can be monitored in u-center with the UBX-ESF-ALG and UBX-ESF-STATUS messages.

1. With the car stationary, power on the EVK-M9DR and wait for a valid 3D GNSS fix.
2. Remain stationary until IMU status in ESF-STATUS shows "INITIALIZED".
3. Drive a figure-of-eight pattern until the alignment status in ESF-ALG shows "COARSE".
4. Drive straight at a minimum speed of 40 km/h until INS status in 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 ESF-STATUS will show "FUSION".

The receiver will continue calibrating the sensors in the background continuously to improve the quality of the solution. For optimal performance, it is recommended to repeat step 3 until ESF-ALG shows "FINE" and to drive curves and straight segments until all sensors in ESF-STATUS report "CALIBRATED" before actual testing.

☞ Refer to the NEO-M9V Integration manual [2] for more information about sensor calibration.

3.3 Testing

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, do the following steps:

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.

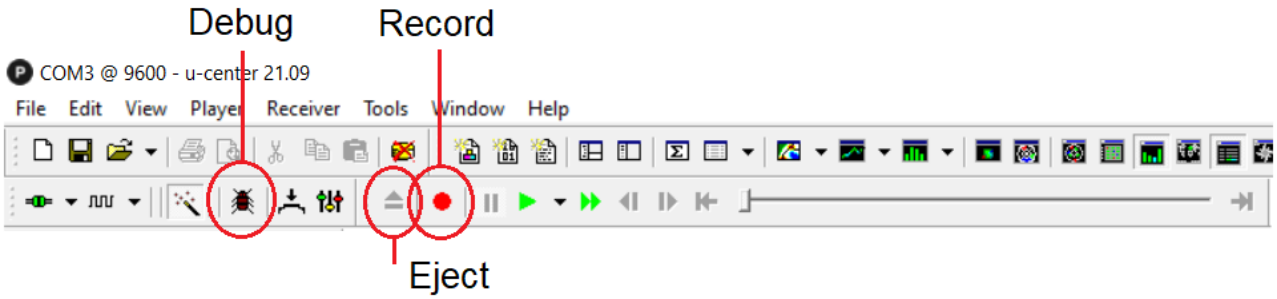


Figure 1: u-center logging controls

3.4 Analysis

After collecting data over 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, do the following:

1. Open u-center.

2. Open a log using **File > Open...**
3. Use the log 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 [4] for more information about its features.

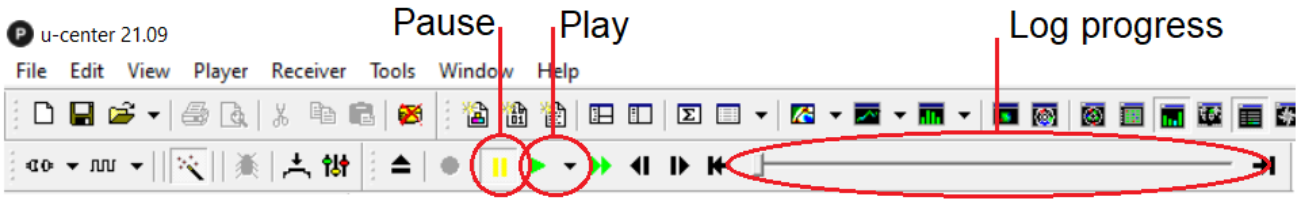


Figure 2: u-center log controls

4 Advanced setup

4.1 Non-automotive applications

To use the device in a two-wheel vehicle, some extra steps are required. Follow these instructions in addition to section 3.1.

☞ The UBX-C cable provided together with the kit might not be sufficiently long when evaluating these use cases. Get a cable with appropriate length before starting the test.

4.1.1 Configuring the dynamic model

There are two dynamic models for two-wheel vehicles: motorbike and e-scooter. Select the model which better suits the application and configure the receiver accordingly:

- For motorbike applications, set `CFG-NAVSPG-DYNMODEL = 10`.
- For e-scooter applications, set `CFG-NAVSPG-DYNMODEL = 12`.

☞ Refer to the NEO-M9V Integration manual [2] for more information about different dynamic models.

4.1.2 Configuring the IMU alignment

When using the motorbike or e-scooter dynamic model, auto-alignment must be disabled, and the correct alignment angles have to be configured manually:

1. Disable auto-alignment by setting `CFG-SFIMU-AUTO_MNTALG_ENA = 0`
2. Set the roll angle with `CFG-SFIMU-IMU_MNTALG_ROLL`.
3. Set the pitch angle with `CFG-SFIMU-IMU_MNTALG_PITCH`.
4. Set the yaw angle with `CFG-SFIMU-IMU_MNTALG_YAW`.

☞ Refer to the NEO-M9V Integration manual [2] for determining the alignment angles.

4.2 ADR setup

Follow these instructions in addition to section 3.1 to set up the device for testing ADR.

4.2.1 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 odometer input to the receiver:

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

Only one of the options above may be used at one 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.

See the NEO-M9V documentation ([2], [3]) for more information about options 1 and 2. For option 3, refer to chapter 5.

4.2.2 Configuring the device for ADR

The receiver can be configured with UBX-CFG-VALSET messages. Consult the NEO-M9V documentation ([2], [3]) 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, **the wheel tick pin must be disabled**. Set the value for key ID CFG-SFODO-USE_WT_PIN to 0. See chapter 5 for instructions on how to configure the CAN interface.

☞ It is highly recommended to verify that the configuration is correct and to perform system sanity checks.

☞ If the GNSS antenna is placed at a significant distance from the receiver, position offsets can be introduced which might affect the accuracy of the navigation solution. In order to compensate for the position offset, advanced configurations can be applied. Contact u-blox support for more information on advanced configurations.

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 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. For more information, see [5].

5.3 C100 MSG

The C100 MSG browser-based tool (see Figure 3) generates C100 MCU configuration messages for the configurable CAN feature. It can run entirely locally, without an internet connection.

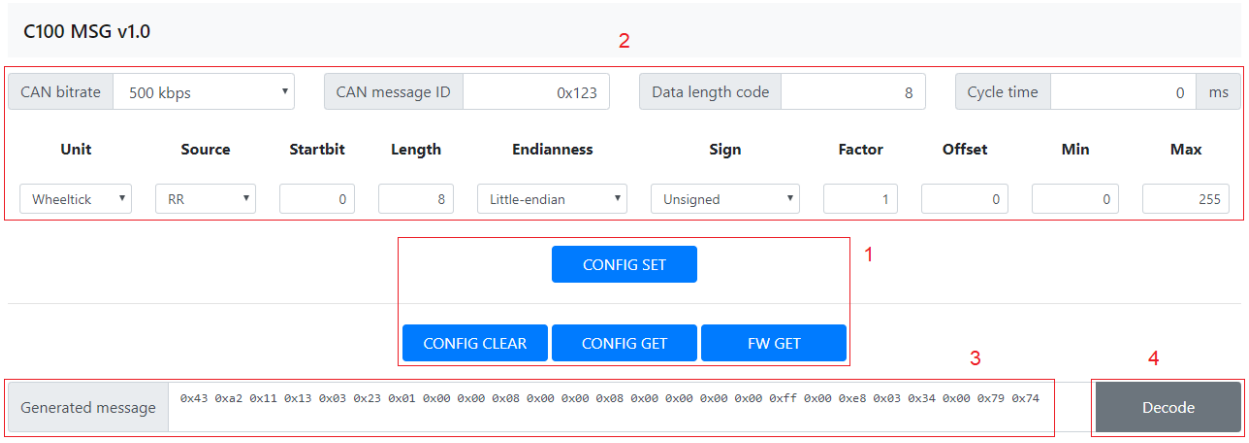


Figure 3: C100 MSG tool

The numbers in the list below refer to Figure 3:

- **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

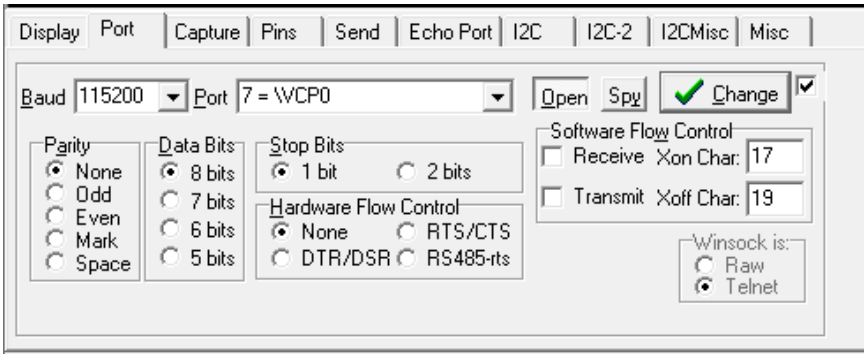
Follow these steps to configure the CAN interface:

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.

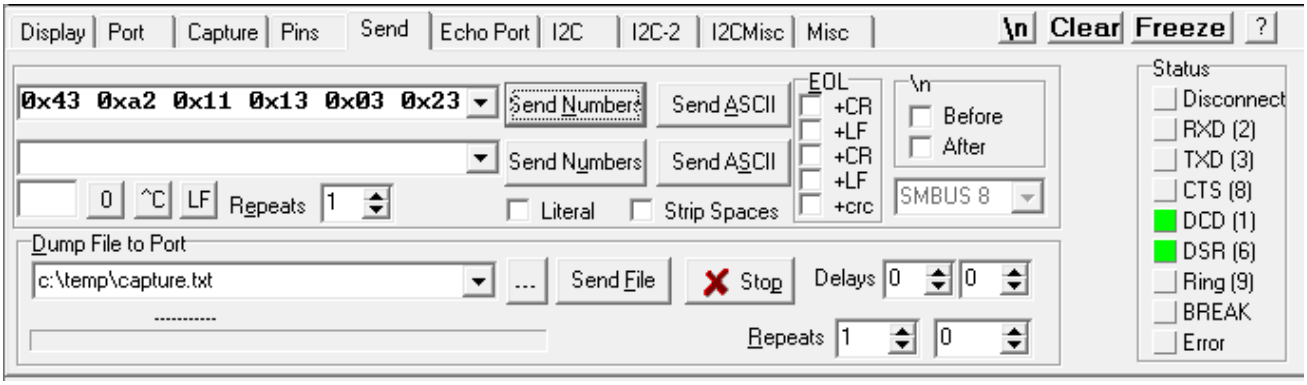
5.4.2 RealTerm

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



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

4. Setting up the configurable CAN feature:
   4.1. Open the RealTerm **Send** tab.
   4.2. Generate CONFIG SET message(s) in the MSG tool.
   4.3. Copy and paste a CONFIG SET message into the text field.
   4.4. Send the message by selecting the **Send Numbers** button.



The following dialog should be displayed when a configuration has been accepted:

When all configuration messages have been sent:

   4.5. Generate a CONFIG GET message.
   4.6. Send the CONFIG GET message.
   4.7. A dialog similar to the one shown below should be displayed and can be used to validate the configurations.

```
C100: Get configuration: CRLF
C100: Bitrate (kbps): 500CRLF
C100: num CAN Configs found: 2CRLF
C100: Config 1CRLF
C100:     canMsgId 0x123CRLF
C100:     dlc 8CRLF
C100:     cycleTime 0CRLF
C100:     startBit 0CRLF
C100:     length 8CRLF
C100:     offset 0CRLF
C100:     factor 1000CRLF
C100:     minVal 0CRLF
C100:     maxVal 255CRLF
C100:     msgType 2CRLF
C100:     source 3CRLF
C100:     unit 1CRLF
C100:     sign 0CRLF
C100:     endian 0CRLF
C100: Config 2CRLF
C100:     canMsgId 0x123CRLF
C100:     dlc 8CRLF
C100:     cycleTime 0CRLF
C100:     startBit 8CRLF
C100:     length 8CRLF
C100:     offset 0CRLF
C100:     factor 1000CRLF
C100:     minVal 0CRLF
C100:     maxVal 255CRLF
C100:     msgType 2CRLF
C100:     source 2CRLF
C100:     unit 1CRLF
C100:     sign 0CRLF
C100:     endian 0CRLF
C100: OK 16CRLF
```

☞ 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 [6], and
- USB debug adapter for 8-bit MCUs [7]

Follow these steps 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 circled in Figure 4. The jumper is included by default. If the termination needs to be removed, open the enclosure and remove the jumper.

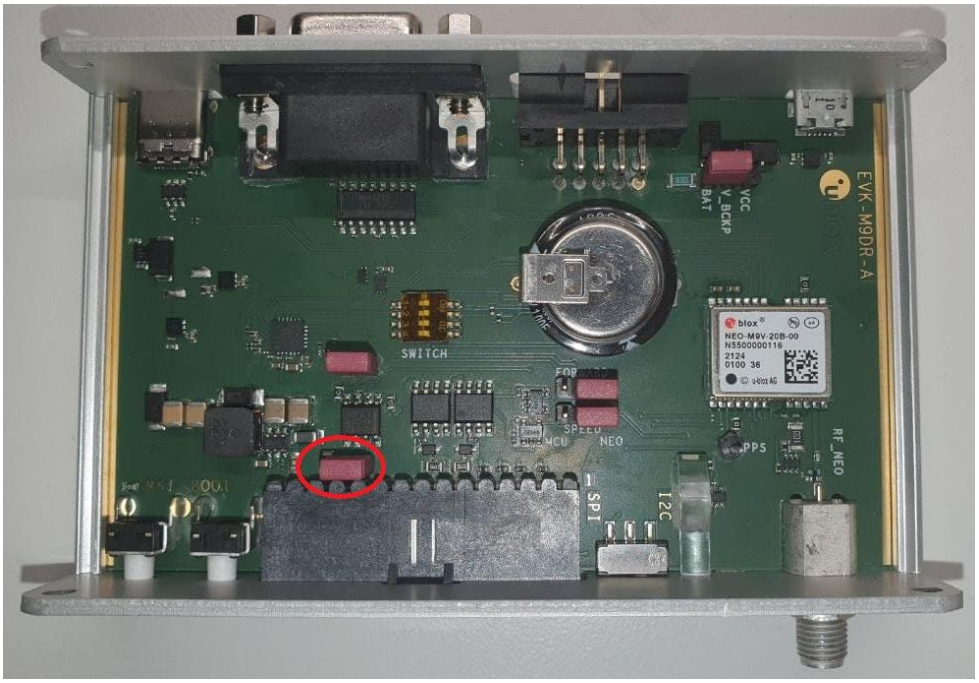


Figure 4: Jumper (circled)

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 will be used for powering the device and for the u-center interface.
- Odometer sensor measurements will be provided from the vehicle CAN bus via CAN_H and CAN_L pins on the front connector.
- UART RS-232 connector will be 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 will select the MCU UART.
2. Connect UART cable to PC.
3. Connect USB cable to PC. Check that the blue light on the front panel is active.

Checking u-center

1. Open u-center.
2. Connect to the receiver:
   - **Receiver > Connection > COMxx**
3. Verify that the connection is established. Poll UBX-MON-VER, and check that the FWVER is correct (MDR 2.10)
4. Update the receiver if necessary (**Tools** > **Firmware Update**).

Configuring the receiver

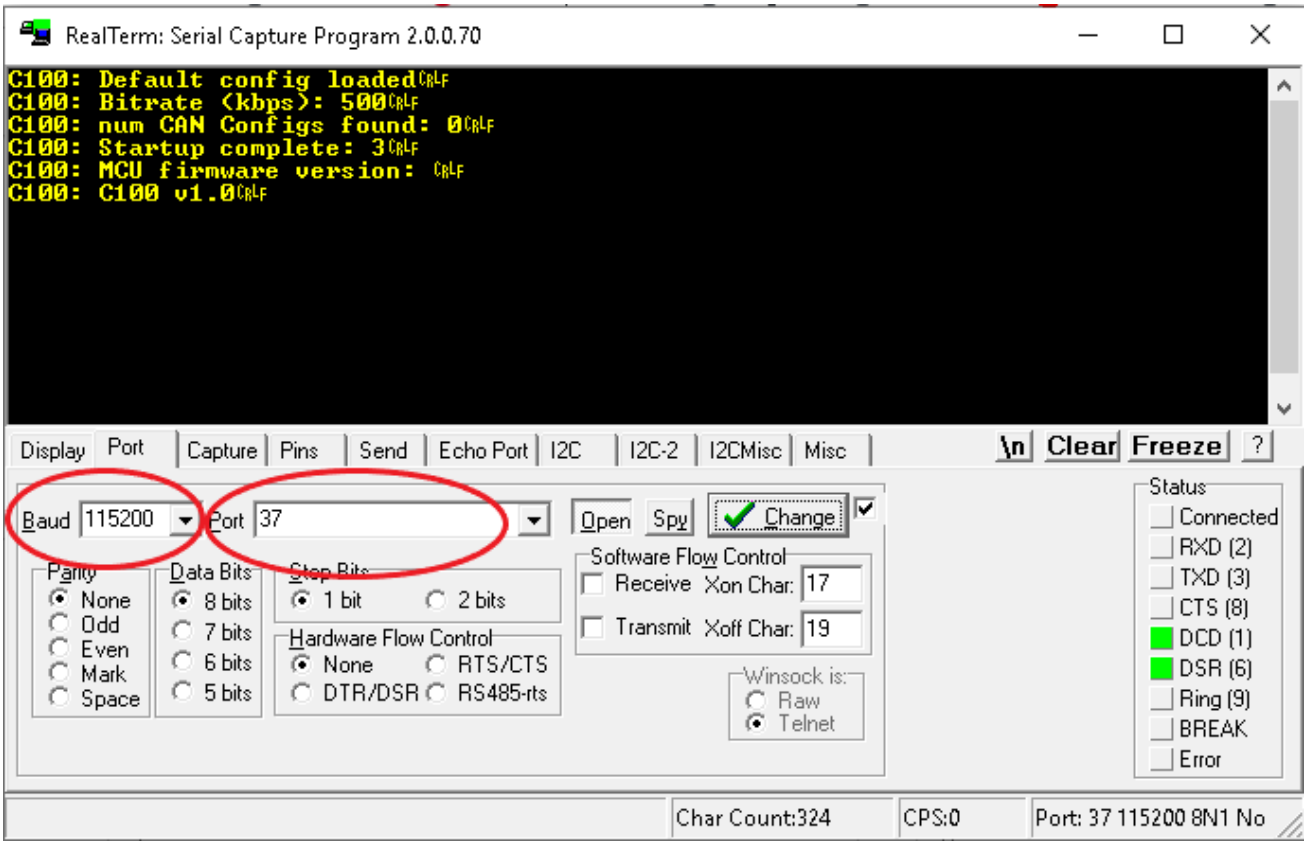
Receiver configuration can be set with UBX-CFG-VALSET message and the appropriate configuration keys.

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

⚠ The UART1 baud rate will need to be increased when the priority navigation mode is enabled, or if debug messages are enabled.

Configuring the CAN interface in RealTerm

1. Open RealTerm.
2. Select the **Port** tab.
3. Select the PC port corresponding to the MCU UART.
4. Set baud rate to 115200.
5. Restart the EVK.
6. MCU startup dialog should appear in the terminal.



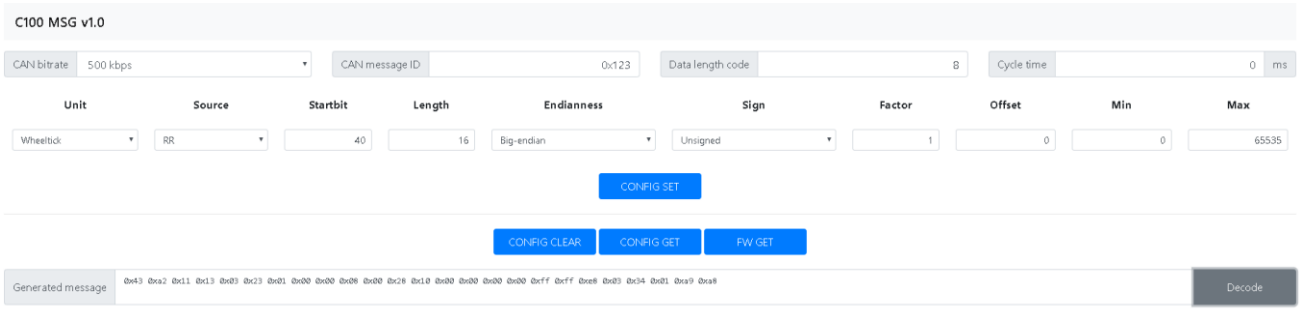
Generating the CONFIG SET strings with the MSG tool

From section B.1.1:

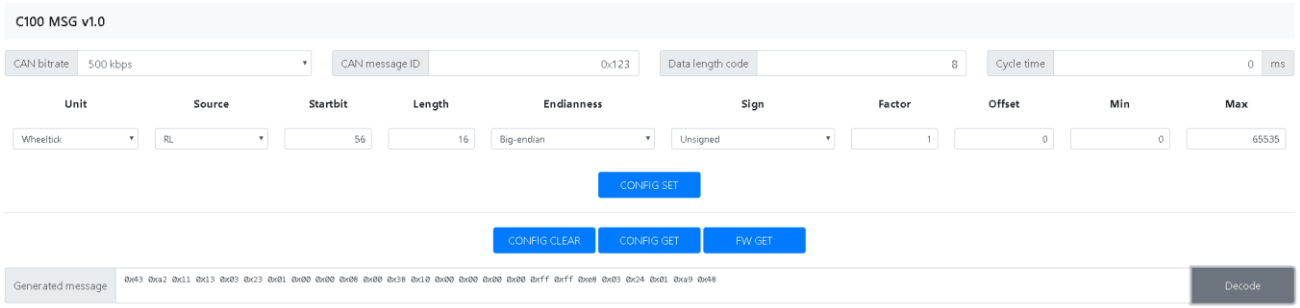
Startbit	Length	Byte order	Value type	Factor	Offset	Min	Max	Unit	Source
40	16	big-endian	unsigned	1	0	0	65535	tick	RR
56	16	big-endian	unsigned	1	0	0	65535	tick	RL
8	2	big-endian	unsigned	1	0	0	3	direction	direction

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

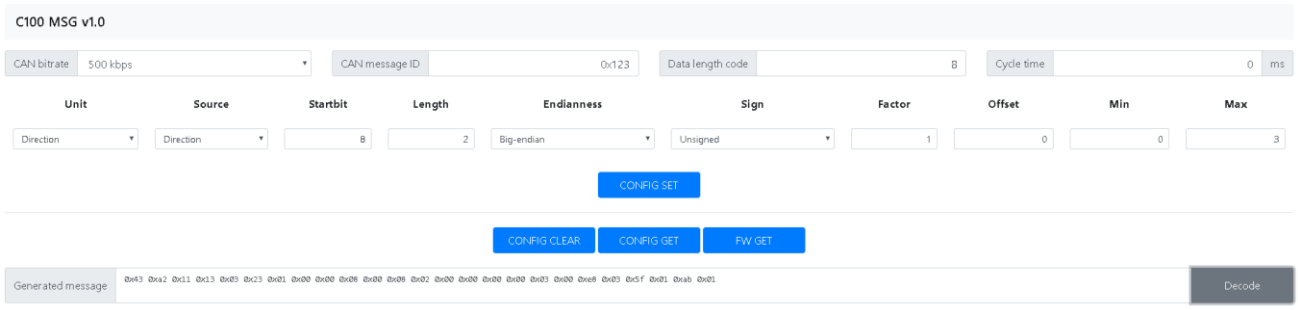
Rear-right wheel tick:



Rear-left wheel tick:



Direction:



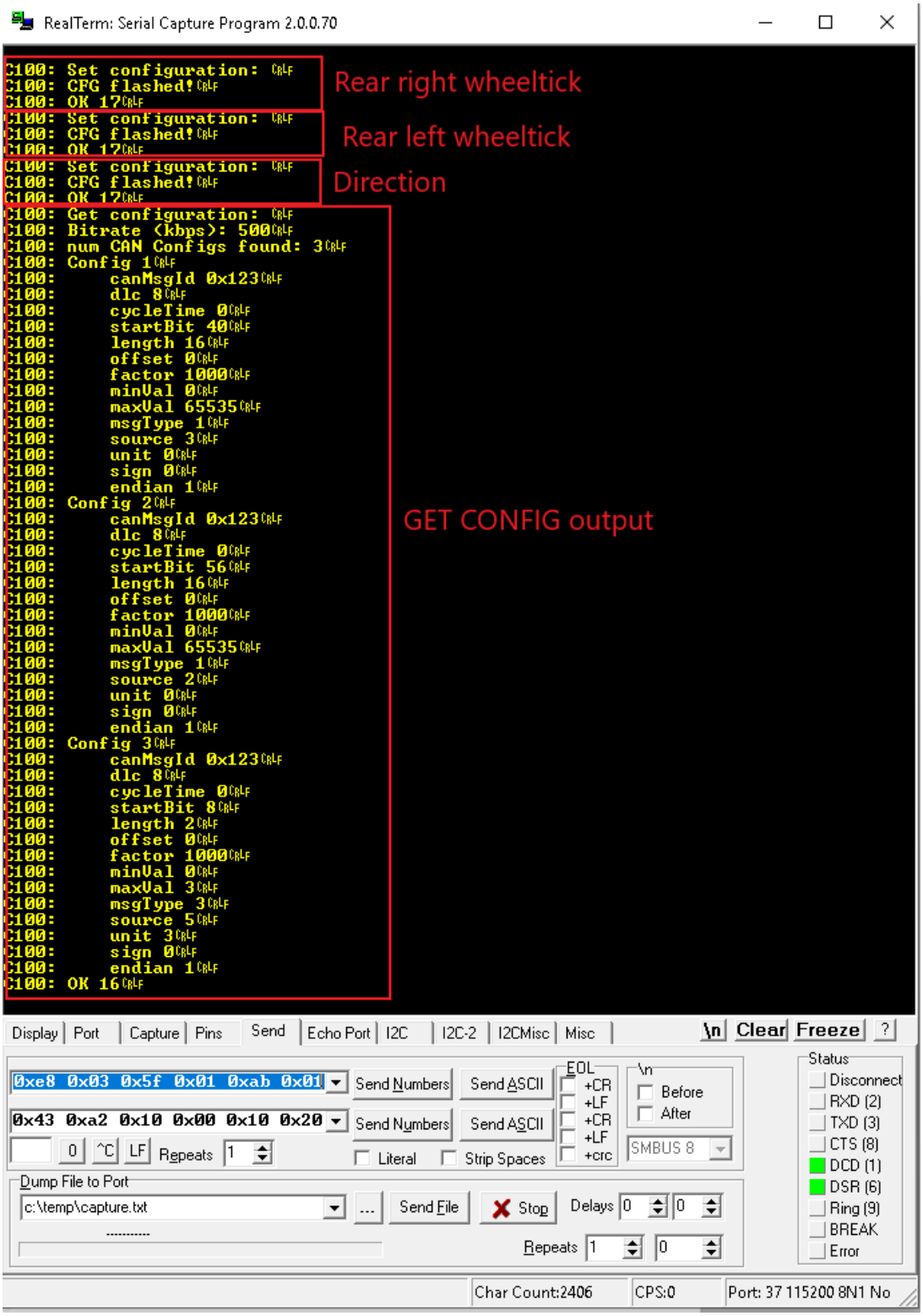
The following CONFIG SET messages are generated for this configuration:

- **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:

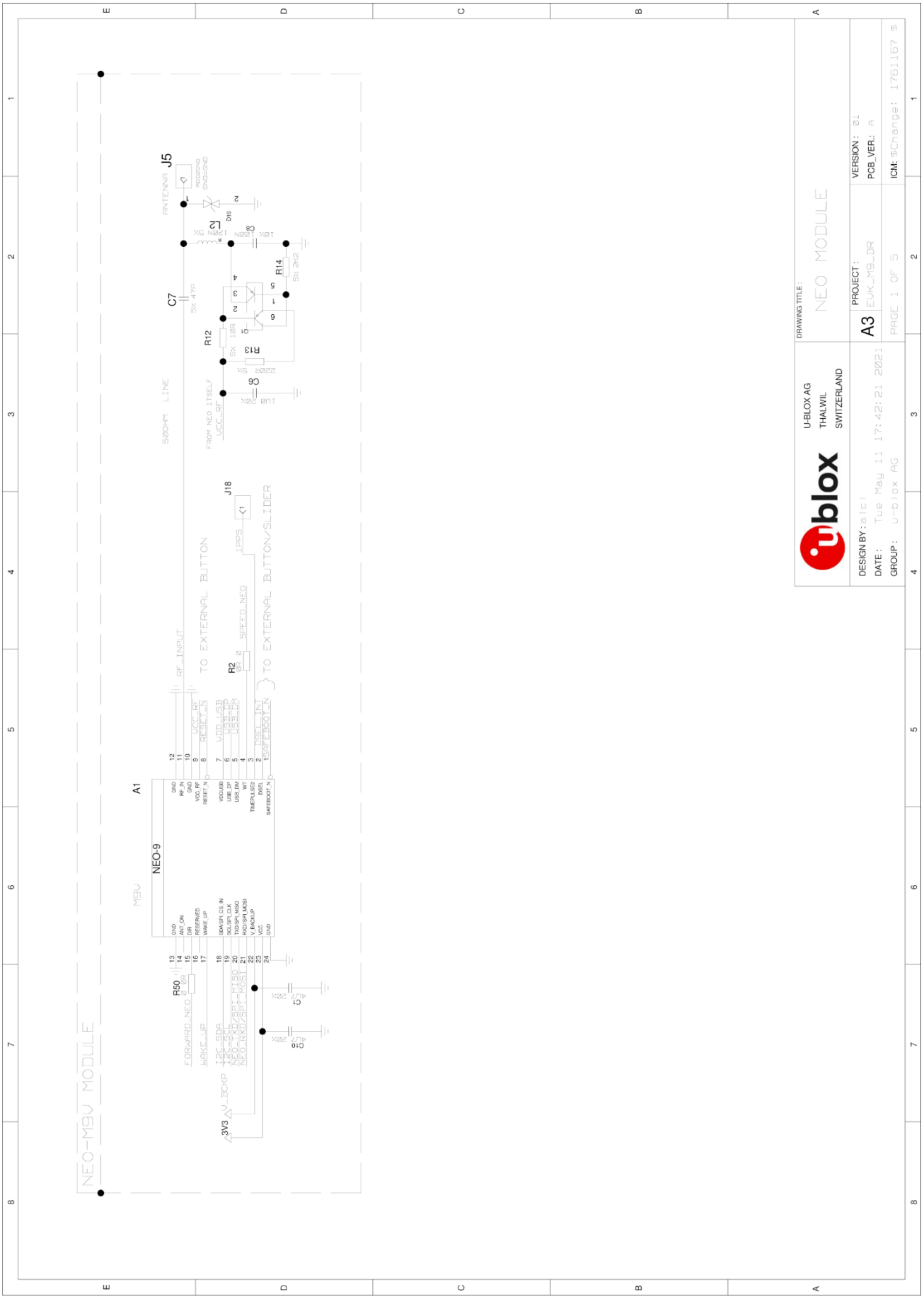
8. Open RealTerm.
9. Select the **Send** tab.
10. Copy and paste the rear-right wheel tick CONFIG SET string to the RealTerm text box.
11. Select the **Send Numbers** button.

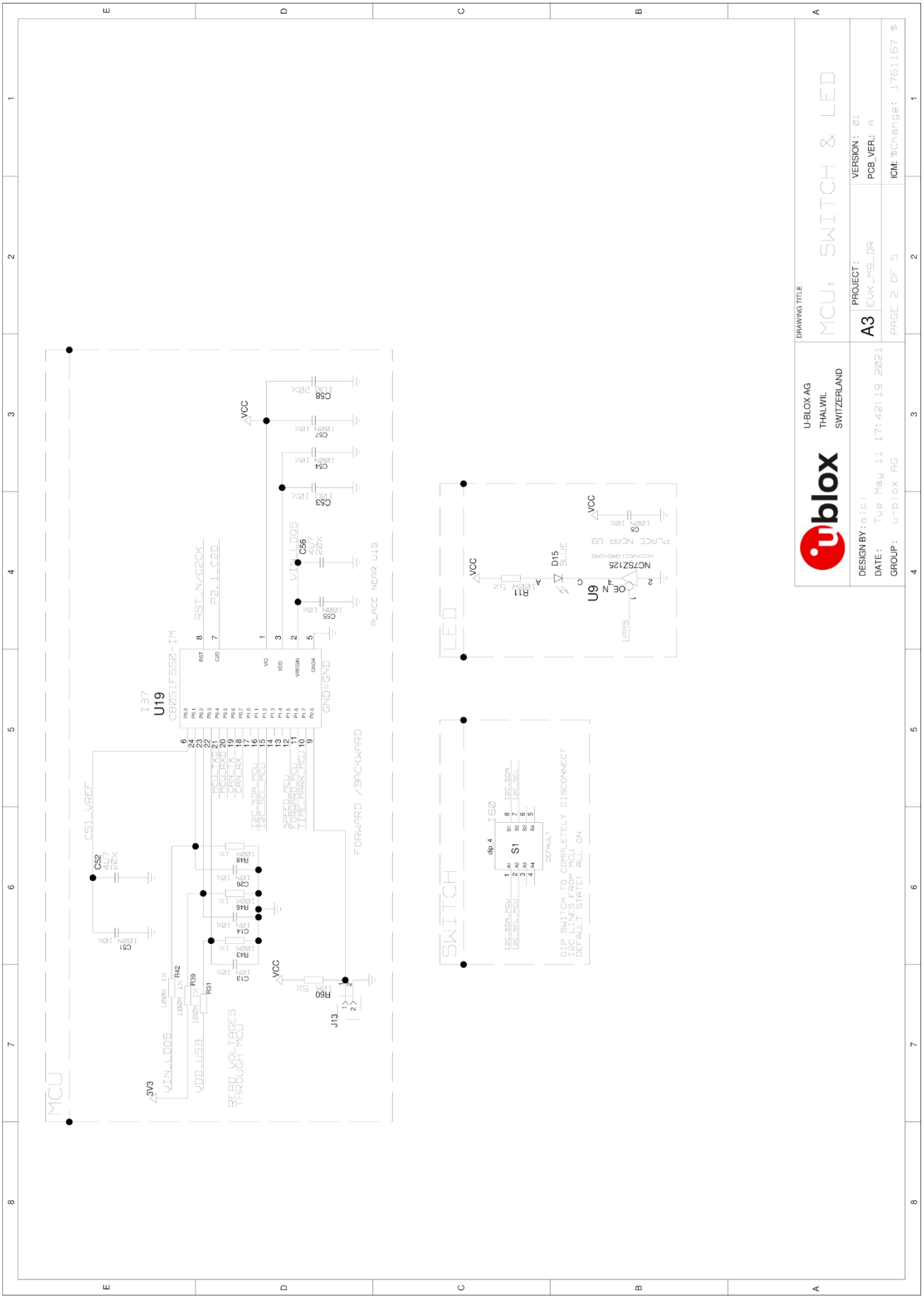
Verify configurations with CONFIG GET string, 0x43 0xa2 0x10 0x00 0x10 0x20.



D Schematic

The following pages include the complete schematic for the EVK-M9DR board.


NEO-M9V MODULE
ANTENNA
J5
50OHM LINE
RF_INPUT
TO EXTERNAL BUTTON
TO EXTERNAL BUTTON/SLIDER
SPEED_NEO
FORWARD_NEO
WAKE_UP
I2C_SDA
NEO_TXD/SPI_MISO
NEO_RXD/SPI_MOSI
V_BCKP
3V3
M9V
NEO-9
A1
J18
R2
R50
C5
C10
C7
R12
R13
R14
C6
D16
L3
DRAWING TITLE:
NEO MODULE
U-BLOX AG
THALWIL
SWITZERLAND
A3
PROJECT:
EVK_M9_DR
VERSION: 01
PCB_VER: A
PAGE 1 OF 5
ICM: #change: 1761167 #
DESIGN BY: alcl
DATE: Tue May 11 17:42:21 2021
GROUP: u-blox AG


MCU, SWITCH & LED
U-BLOX AG
THALWIL
SWITCHERLAND
A3
PROJECT: EVK_M9_DR
VERSION: 01
PCB_VER: A
PAGE 2 OF 5
ICM: #change: 1761167
DESIGN BY: alcl
DATE: Tue May 11 17:42:19 2021
GROUP: u-blox AG
MCU
SWITCH
LED
U19
C8051F550-IM
U9
NC7SZ125
D15
BLUE
S1
VCC
3V3
FORWARD / BACKWARD
PLACE NEAR U19
PLACE NEAR U9
GND-GND
DIP SWITCH TO COMPLETELY DISCONNECT I2C LINES FROM MCU
DEFAULT STATE: ALL ON


STEP DOWN DCDCD AND LDO'S
BATTERY
SUPPLIES & BATTERY
U-BLOX AG
THALWIL
SWITZERLAND
DRAWING TITLE:
PROJECT: EVK_M9_DR
VERSION: 01
PCB_VER: A
PAGE 3 OF 5
ICM: #Change: 1761167 5
DESIGN BY: alcl
DATE: Tue May 11 17:42:22 2021
GROUP: u-blox AG
A3
VCC
3V3
VDDUSB
VIN_LDOS
~3.7V
~4.1V
3.7V TO 4.75V
MAX 500MA EACH LDO WITH 230MV DROPOUT
ONLY FOR NEO MODULE SUPPLY
TO MEASURE CURRENT OF MODULE
PROTECTION DIODES
LOW DROPOUT DIODE: 0.4V @ 1A
JUMPER BETWEEN 2-3 AS DEFAULT
STEP DOWN DCDC
U3
F1
FUSE
5-24V
1A
U2
U4
U11
U8
NCV8705
D1
D2
D3
D4
D6
J7
J8
J11
R8
R9
R10
R17
C34
C35
C36
C37
C39
L3


RS232 CONNECTION
FRONT BUTTONS AND SLIDER
USB CONNECTION
CAR/DEVICE CONNECTOR
DEBUG
CONNECTORS & INTERFACES
U-BLOX AG
THALWIL
SWITZERLAND
A3
PROJECT : EVK_M9_DR
VERSION : 01
PCB_VER.: A
PAGE 4 OF 5
ICM: $change: 1761167 $
DESIGN BY: alci
DATE : Tue May 11 17:42:20 2021
GROUP : u-blox AG
MAXIM_3232
USBLC6 2SC6
RS232
USB
J1
J2
J4
J9
J10
U5
U6
U7
D7
S2
S3
S4


CAN TRANSCEIVER
U14
SN65HVD232
DIGITAL ISOLATORS (OPTO SIMULATOR)
U17
U18
U10
U12
NC7SZ14
SPEED_NEO
FORWARD_NEO
SPEED_MCU
FORWARD_MCU
SPEED_OPTO
FORWARD_OPTO
INTERFACES
U-BLOX AG
THALWIL
SWITZERLAND
PROJECT : EVK_M9_DR
A3
PAGE 5 OF 5

Related documents

[1] NEO-M9V Datasheet, UBX-2102978
[2] NEO-M9V Integration Manual, UBX-21029776
[3] M9 MDR 2.10 Interface description, UBX-21036678
[4] u-center user guide, UBX-13005250
[5] RealTerm Serial Terminal, https://realterm.sourceforge.io/
[6] Silicon Labs 8-bit Microcontroller Software, https://www.silabs.com/products/development-tools/software/8-bit-8051-microcontroller-software
[7] Silicon Labs 8-bit USB Debug Adapter, https://www.silabs.com/development-tools/mcu/8-bit/8-bit-usb-debug-adapter

☞ For regular updates to u-blox documentation and to receive product change notifications, register on our homepage (www.u-blox.com).

Revision history

Revision	Date	Name	Status / Comments
R01	03-Feb-2022	jilm	Initial release

Contact

For complete contact information, visit us at www.u-blox.com.

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone: +1 703 483 3180
E-mail: info_us@u-blox.com

Regional Office West Coast:

Phone: +1 408 573 3640
E-mail: info_us@u-blox.com

Technical Support:

Phone: +1 703 483 3185
E-mail: support_us@u-blox.com

Headquarters
Europe, Middle East, Africa

u-blox AG

Phone: +41 44 722 74 44
E-mail: info@u-blox.com
Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone: +65 6734 3811
E-mail: info_ap@u-blox.com
Support: support_ap@u-blox.com

Regional Office Australia:

Phone: +61 2 8448 2016
E-mail: info_anz@u-blox.com
Support: support_ap@u-blox.com

Regional Office China (Beijing):

Phone: +86 10 68 133 545
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Chongqing):

Phone: +86 23 6815 1588
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shanghai):

Phone: +86 21 6090 4832
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone: +86 755 8627 1083
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office India:

Phone: +91 80 405 092 00
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

Regional Office Japan (Osaka):

Phone: +81 6 6941 3660
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone: +81 3 5775 3850
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Korea:

Phone: +82 2 542 0861
E-mail: info_kr@u-blox.com
Support: support_kr@u-blox.com

Regional Office Taiwan:

Phone: +886 2 2657 1090
E-mail: info_tw@u-blox.com
Support: support_tw@u-blox.com