

# 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.

# Document information

<b>Title</b>	EVK-M9DR		
<b>Subtitle</b>	Evaluation kit		
<b>Document type</b>	User guide		
<b>Document number</b>	UBX-21049360		
<b>Revision and date</b>	R02	17-May-2023	
<b>Disclosure Restriction</b>	C1-Public		

This document applies to the following products:

<b>Product name</b>	<b>Type number</b>	<b>Firmware version</b>	<b>IN/PCN reference</b>
EVK-M9DR	EVK-M9DR-0-00	MDR 2.10	N/A
EVK-M9DR	EVK-M9DR-0-01	MDR 2.16	UBX-23006159

u-blox or third parties may hold intellectual property rights in the products, names, logos and designs included in this document. Copying, reproduction, modification or disclosure to third parties of this document or any part thereof is only permitted with the express written permission of u-blox.

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](http://www.u-blox.com).

Copyright © u-blox AG.

# Contents

<b>Document information</b> .....	<b>2</b>
<b>Contents</b> .....	<b>3</b>
<b>1 Introduction</b> .....	<b>5</b>
1.1 Highlights .....	5
1.2 Kit includes .....	5
1.3 System requirements .....	5
1.4 Evaluation steps.....	5
<b>2 Device description</b> .....	<b>6</b>
2.1 USB.....	6
2.2 UART .....	6
2.3 Antenna.....	6
2.4 14-pin front connector .....	6
2.5 10-pin rear connector .....	6
2.6 Reset and safe boot buttons .....	6
2.7 I2C/SPI slide switch .....	7
2.8 LED.....	7
2.9 Backup power supply.....	7
<b>3 Getting started</b> .....	<b>8</b>
3.1 Installation.....	8
3.1.1 Mounting the device.....	8
3.1.2 Mounting the antenna .....	9
3.1.3 Connecting the cables .....	9
3.1.4 Configuring the receiver (optional) .....	9
3.2 Calibrating the receiver.....	9
3.3 Testing the receiver.....	10
3.4 Analyzing the log files .....	10
<b>4 Advanced setup</b> .....	<b>11</b>
4.1 Non-automotive applications .....	11
4.1.1 Configuring the dynamic model.....	11
4.1.2 Configuring the IMU alignment .....	11
4.2 ADR setup.....	11
4.2.1 Providing odometer input.....	11
4.2.2 Configuring the device for ADR.....	12
<b>5 Configurable CAN interface</b> .....	<b>13</b>
5.1 Valid configurations .....	13
5.2 Configuring the interface .....	13
5.3 C100 MSG.....	13
5.3.1 Configuration parameters .....	14
5.4 Configuration process.....	14
5.4.1 Connections .....	14

5.4.2 RealTerm .....	14
5.5 Updating the MCU firmware.....	16
<b>Appendix .....</b>	<b>17</b>
<b>A CAN termination .....</b>	<b>17</b>
<b>B CAN configuration examples .....</b>	<b>18</b>
<b>B.1 Wheel tick configurations.....</b>	<b>18</b>
<b>B.1.1 Two rear-wheel ticks and direction.....</b>	<b>18</b>
<b>B.1.2 Single tick and direction .....</b>	<b>19</b>
<b>B.2 Speed configurations .....</b>	<b>19</b>
<b>B.2.1 Two rear wheels and direction .....</b>	<b>19</b>
<b>B.2.2 Single speed.....</b>	<b>20</b>
<b>B.2.3 Signed speed .....</b>	<b>21</b>
<b>B.2.4 Offset speed .....</b>	<b>21</b>
<b>C Step-by-step example .....</b>	<b>22</b>
<b>D Schematic .....</b>	<b>25</b>
<b>Related documents .....</b>	<b>31</b>
<b>Revision history .....</b>	<b>31</b>
<b>Contact.....</b>	<b>32</b>

# 1 Introduction

EVK-M9DR can be used to test and evaluate all u-blox products based on the u-blox M9 single-band GNSS dead reckoning technologies. The evaluation kit (EVK) is equipped with the NEO-M9V module pre-flashed with the MDR 2.xx firmware, and allows out-of-the-box performance and feature evaluation of the following products:

- NEO-M9V
- UBX-M9340-KB

In addition, EVK-M9DR can be re-flashed with the ADR 5.xx firmware for performance and feature evaluation of the following products:

- NEO-M9L
- UBX-M9140-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.

## 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 or CR2032 coin cell battery
- Wake-on-Motion feature

## 1.2 Kit includes

- application board with an enclosure
- 1-meter USB-C cable
- 1.8-meter micro-USB cable
- active L1 GNSS antenna with a 3 m cable

## 1.3 System requirements

- 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 the 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 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

 To ensure proper function of the device, the slide switch must remain in the I2C position. The use of SPI interface is deprecated and no longer recommended.


For further information, contact u-blox technical support.

## 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 is 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 the backup voltage. In addition, a battery holder for a CR2032 coin cell is available on the PCB (from EVK-M9DR-0-01).

 **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.

## 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. Follow the steps in this chapter will help to minimize errors leading to most common issues in performance.

For additional information on installing non-automotive applications, see chapter 4.

### 3.1 Installation

#### 3.1.1 Mounting the device

To mount the EVK,

1. Choose a place in the vehicle that remains stable during operation and does not encounter excessive vibrations. A good location for the EVK installation is in the trunk, close to the center of the rear axle of the vehicle. Do not attach the EVK to any moving parts of the vehicle's interior, such as the headrest or rear-view mirror.
2. Utilize appropriate hardware to firmly attach the EVK to the chosen location.
3. Evaluate the success of the EVK installation by checking the warning flags.

Starting from MDR 2.16 firmware, a warning flag called “noisyMeas” is raised in UBX-ESF-STATUS message if the IMU on the module experiences excessive vibrations due to improper installation.

In the u-center GNSS evaluation software, this warning flag is labeled as “High Noise-level” and it is triggered in UBX-ESF-STATUS message (Figure 1) for installations that lead to excessive vibrations.

Sensor	Status	Time	Freq	Faults
Gyroscope Z	CALIBRATED	FB	53	
Single Tick	CALIBRATED	FB	10	
Gyroscope X	CALIBRATED	FB	53	
Gyroscope Y	CALIBRATED	FB	53	⚠ HIGH NOISE-LEVEL
Accelerometer X	CALIBRATED	FB	53	
Accelerometer Y	CALIBRATED	FB	53	⚠ HIGH NOISE-LEVEL
Accelerometer Z	CALIBRATED	FB	53	⚠ HIGH NOISE-LEVEL

Figure 1: High noise-level fault in UBX-ESF-STATUS messages

4. If the “High Noise-level” warning flag is raised, the dead reckoning functionality may not work properly. If the flag is raised, try to determine the cause of the vibration. Check the mounting of the EVK and ensure that it is installed firmly. If necessary, move the EVK to a more stable location.

If the “High Noise-level” warning flag continues to appear even after verifying and adjusting the EVK installation, further steps may be required. To rectify this situation, advanced configurations



can be applied to the receiver. For detailed information on advanced configurations, contact u-blox technical support.

### 3.1.2 Mounting the antenna

Place the GNSS antenna provided with the EVK 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.


### 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 the receiver configuration.

## 3.2 Calibrating the receiver


Before the receiver can operate in the dead reckoning 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.

To calibrate the receiver,

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-M9DR 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 NEO-M9V Integration manual [2] for more information about the sensor calibration.

### 3.3 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 2). This step is optional, but necessary for investigation of issues.
5. Start recording with the record button (Figure 2).
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 2). The log file will be saved automatically.



Figure 2: u-center logging controls

### 3.4 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 controls (Figure 3) 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 [6] for more information about its features.

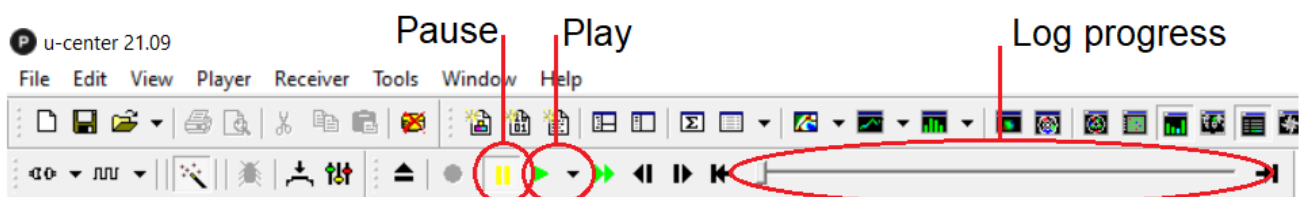



Figure 3: u-center log controls

## 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 USB-C cable provided with the kit might not be sufficiently long when evaluating use cases for non-automotive applications. Obtain 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, disable the auto-alignment and manually configure the correct alignment angles:


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.

 Note that the IMU reference frame depicted on top of the EVK-M9DR is aligned with the one of the NEO-M9V but differs from the NEO-M9L. For details, see section IMU-mount alignment in the Integration Manuals [2] [4].

## 4.2 ADR setup

To evaluate the performance and features of the ADR 5.xx firmware used by NEO-M9L and UBX-M9140-KA-DR, or to evaluate the ADR operating mode of the MDR 2.xx firmware used by NEO-M9V and UBX-M9340-KB, follow these instructions in addition to section 3.1.

 Make sure you have the desired firmware flashed on your EVK-M9DR. Firmware can be downloaded from the corresponding product pages on the u-blox website. Refer to the Firmware Update section in the u-center User guide [6] for more details on the update procedure.

### 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 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 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]) or the NEO-M9L documentation ([4], [5]) 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 the 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, **disable the wheel tick pin**. Set the value for key ID CFG-SFODO-USE\_WT\_PIN to 0. See chapter 5 for instructions on configuring 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. This 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.

## 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. For more information, see [7].

### 5.3 C100 MSG

The C100 MSG browser-based tool (see Figure 4) generates C100 MCU configuration messages for the configurable CAN feature. It can run entirely locally, without an internet connection.

C100 MSG v1.0

2

CAN bitrate: 500 kbps | CAN message ID: 0x123 | Data length code: 8 | Cycle time: 0 ms

Unit	Source	Startbit	Length	Endianness	Sign	Factor	Offset	Min	Max
Wheeltick	RR	0	8	Little-endian	Unsigned	1	0	0	255

1

CONFIG SET

CONFIG CLEAR | CONFIG GET | FW GET

3

4


Generated message: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xff 0x00 0xe8 0x03 0x34 0x00 0x79 0x74

Decode

Figure 4: C100 MSG tool

The numbers in the list below refer to Figure 4:

- **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

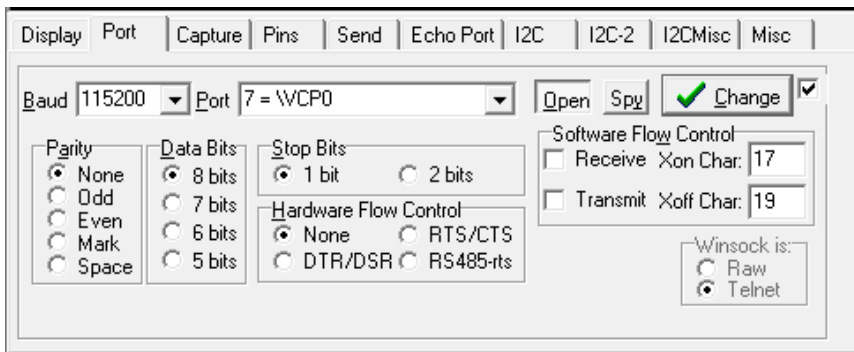
Follow these steps to configure the CAN interface:

### 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

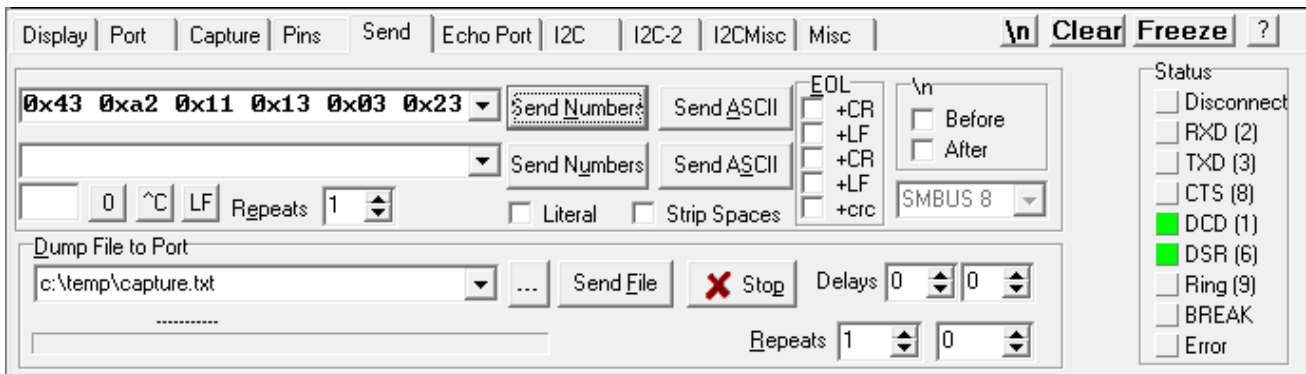
1. Select the port associated with the UART connection in the **Port** tab.
2. Set baud rate to 115200.
3. Apply changes by selecting the **Change** button. See the figure below.



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

```
C100: Default config loaded\r\n
C100: Bitrate (kbps): 500\r\n
C100: num CAN Configs found: 0\r\n
C100: Startup complete: 3\r\n
C100: MCU firmware version: \r\n
C100: C100 v1.0\r\n
```

4. Set 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 is displayed when the configuration has been accepted:


```
C100: Set configuration: \r\n
C100: CFG flashed! \r\n
C100: OK 17\r\n
```

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 is displayed and can be used to validate the configurations.

```

C100: Get configuration: CRLF
C100: Bitrate (kbps): 500 CRLF
C100: num CAN Configs found: 2 CRLF
C100: Config 1 CRLF
C100:   canMsgId 0x123 CRLF
C100:   dlc 8 CRLF
C100:   cycleline 0 CRLF
C100:   startBit 0 CRLF
C100:   length 8 CRLF
C100:   offset 0 CRLF
C100:   factor 1000 CRLF
C100:   minVal 0 CRLF
C100:   maxVal 255 CRLF
C100:   msgType 2 CRLF
C100:   source 3 CRLF
C100:   unit 1 CRLF
C100:   sign 0 CRLF
C100:   endian 0 CRLF
C100: Config 2 CRLF
C100:   canMsgId 0x123 CRLF
C100:   dlc 8 CRLF
C100:   cycleLine 0 CRLF
C100:   startBit 8 CRLF
C100:   length 8 CRLF
C100:   offset 0 CRLF
C100:   factor 1000 CRLF
C100:   minVal 0 CRLF
C100:   maxVal 255 CRLF
C100:   msgType 2 CRLF
C100:   source 2 CRLF
C100:   unit 1 CRLF
C100:   sign 0 CRLF
C100:   endian 0 CRLF
C100: OK 16 CRLF
  
```

 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 [8], and
- USB debug adapter for 8-bit MCUs [9]

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 5. The jumper is included by default. If the termination needs to be removed, open the enclosure and remove the jumper.

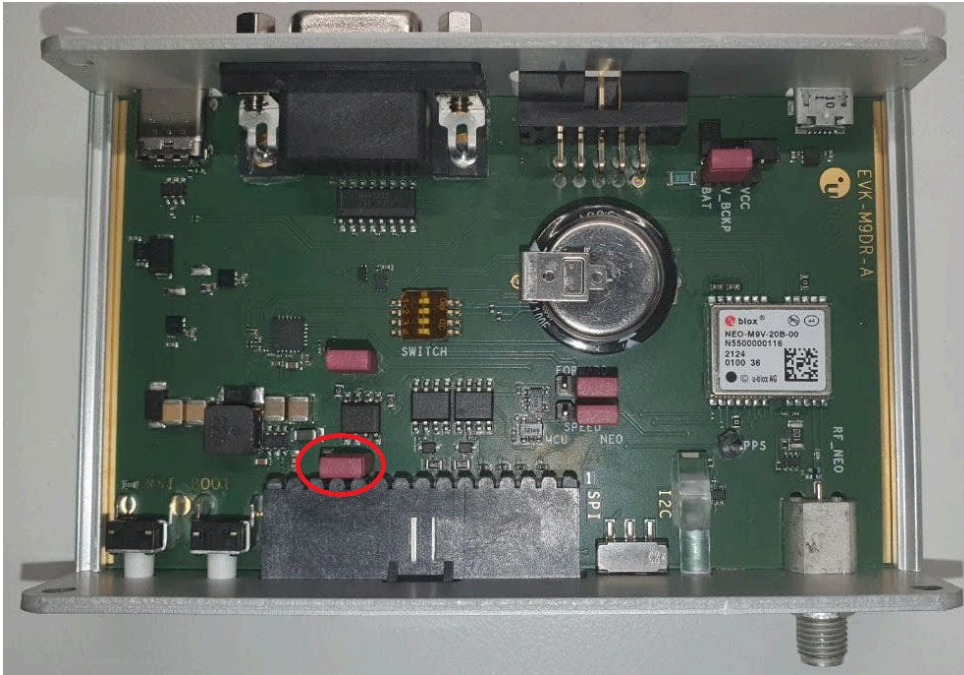


Figure 5: Jumper (circled)

## 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 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 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 2.16)
4. Update the receiver if necessary (**Tools > Firmware Update**).

### Configuring the receiver

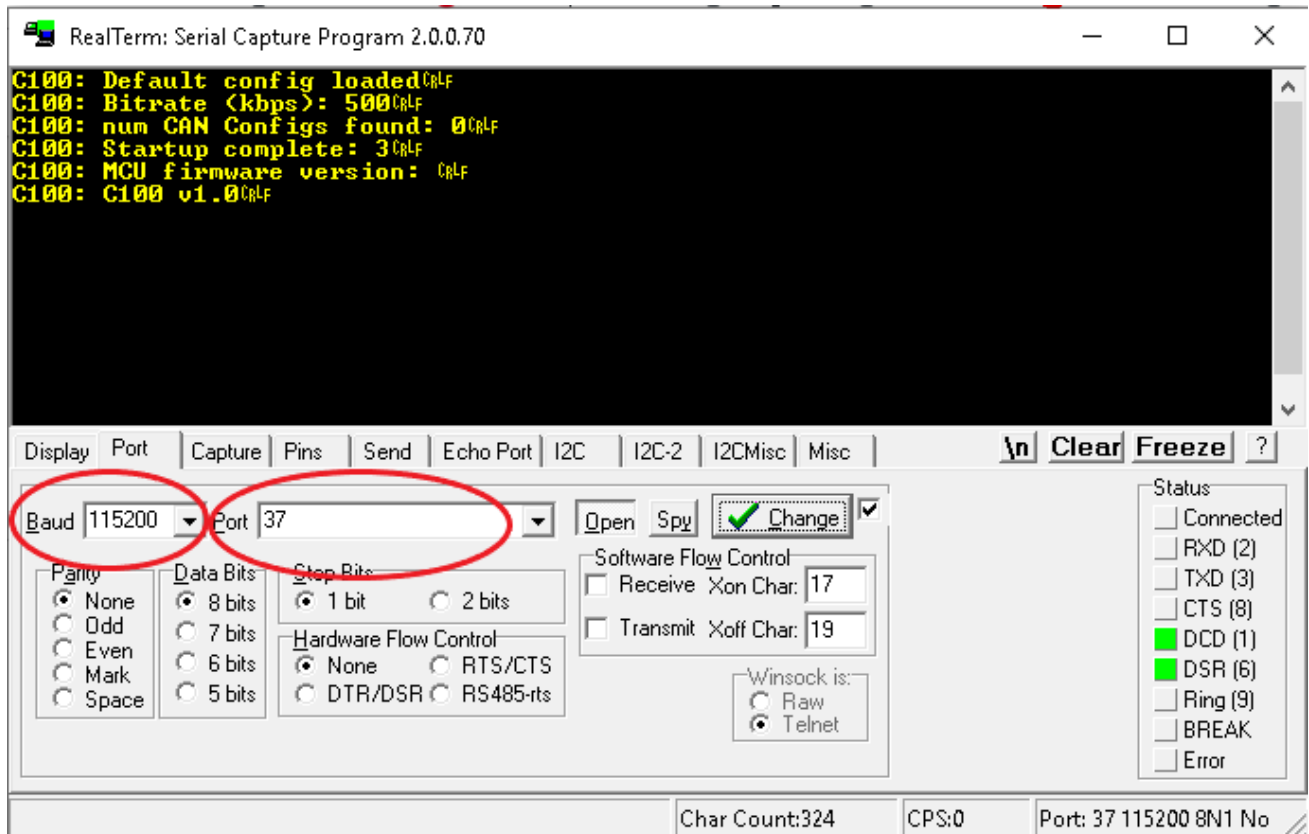
Receiver configuration can be set with UBX-CFG-VALSET message and the appropriate configuration keys.

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

 **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

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:

C100 MSG v1.0

CAN bitrate: 500 kbps | CAN message ID: 0x123 | Data length code: 8 | Cycle time: 0 ms

Unit	Source	Startbit	Length	Endianness	Sign	Factor	Offset	Min	Max
Wheeltick	RR	40	16	Big-endian	Unsigned	1	0	0	65535

Buttons: CONFIG SET, CONFIG CLEAR, CONFIG GET, FW GET

Generated message: 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

Decode

### Rear-left wheel tick:

C100 MSG v1.0

CAN bitrate: 500 kbps | CAN message ID: 0x123 | Data length code: 8 | Cycle time: 0 ms

Unit	Source	Startbit	Length	Endianness	Sign	Factor	Offset	Min	Max
Wheeltick	RL	56	16	Big-endian	Unsigned	1	0	0	65535

Buttons: CONFIG SET, CONFIG CLEAR, CONFIG GET, FW GET

Generated message: 0x43 0xa2 0x11 0x13 0x03 0x23 0x01 0x00 0x00 0x08 0x00 0x38 0x10 0x00 0x00 0x00 0x00 0xff 0xff 0xe8 0x03 0x24 0x01 0xa9 0xa8

Decode

### Direction:

C100 MSG v1.0

CAN bitrate: 500 kbps | CAN message ID: 0x123 | Data length code: 8 | Cycle time: 0 ms

Unit	Source	Startbit	Length	Endianness	Sign	Factor	Offset	Min	Max
Direction	Direction	8	2	Big-endian	Unsigned	1	0	0	3

Buttons: CONFIG SET, CONFIG CLEAR, CONFIG GET, FW GET

Generated message: 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

Decode

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 0xa8
- **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.



RealTerm: Serial Capture Program 2.0.0.70

```

C100: Set configuration: CRLF
C100: CFG flashed! CRLF
C100: OK 17 CRLF
C100: Set configuration: CRLF
C100: CFG flashed! CRLF
C100: OK 12 CRLF
C100: Set configuration: CRLF
C100: CFG flashed! CRLF
C100: OK 12 CRLF
C100: Get configuration: CRLF
C100: Bitrate (kbps): 500 CRLF
C100: num CAN Configs found: 3 CRLF
C100: Config 1 CRLF
C100:   canMsgId 0x123 CRLF
C100:   dlc 8 CRLF
C100:   cycleTime 0 CRLF
C100:   startBit 40 CRLF
C100:   length 16 CRLF
C100:   offset 0 CRLF
C100:   factor 1000 CRLF
C100:   minVal 0 CRLF
C100:   maxVal 65535 CRLF
C100:   msgType 1 CRLF
C100:   source 3 CRLF
C100:   unit 0 CRLF
C100:   sign 0 CRLF
C100:   endian 1 CRLF
C100: Config 2 CRLF
C100:   canMsgId 0x123 CRLF
C100:   dlc 8 CRLF
C100:   cycleTime 0 CRLF
C100:   startBit 56 CRLF
C100:   length 16 CRLF
C100:   offset 0 CRLF
C100:   factor 1000 CRLF
C100:   minVal 0 CRLF
C100:   maxVal 65535 CRLF
C100:   msgType 1 CRLF
C100:   source 2 CRLF
C100:   unit 0 CRLF
C100:   sign 0 CRLF
C100:   endian 1 CRLF
C100: Config 3 CRLF
C100:   canMsgId 0x123 CRLF
C100:   dlc 8 CRLF
C100:   cycleTime 0 CRLF
C100:   startBit 8 CRLF
C100:   length 2 CRLF
C100:   offset 0 CRLF
C100:   factor 1000 CRLF
C100:   minVal 0 CRLF
C100:   maxVal 3 CRLF
C100:   msgType 3 CRLF
C100:   source 5 CRLF
C100:   unit 3 CRLF
C100:   sign 0 CRLF
C100:   endian 1 CRLF
C100: OK 16 CRLF
    
```

Rear right wheeltick  
Rear left wheeltick  
Direction

GET CONFIG output

Display | Port | Capture | Pins | Send | Echo Port | I2C | I2C-2 | I2CMisc | Misc |   ?

0xe8 0x03 0x5f 0x01 0xab 0x01 | Send Numbers | Send ASCII | EOL: +CR, +LF, +CR, +LF, +crc |  Before,  After | Status: Disconnect, RXD (2), TXD (3), CTS (8), DCD (1), DSR (6), Ring (9), BREAK, Error

0x43 0xa2 0x10 0x00 0x10 0x20 | Send Numbers | Send ASCII |  Literal,  Strip Spaces | SMBUS 8 | Delays: 0, 0 | Repeats: 1, 0

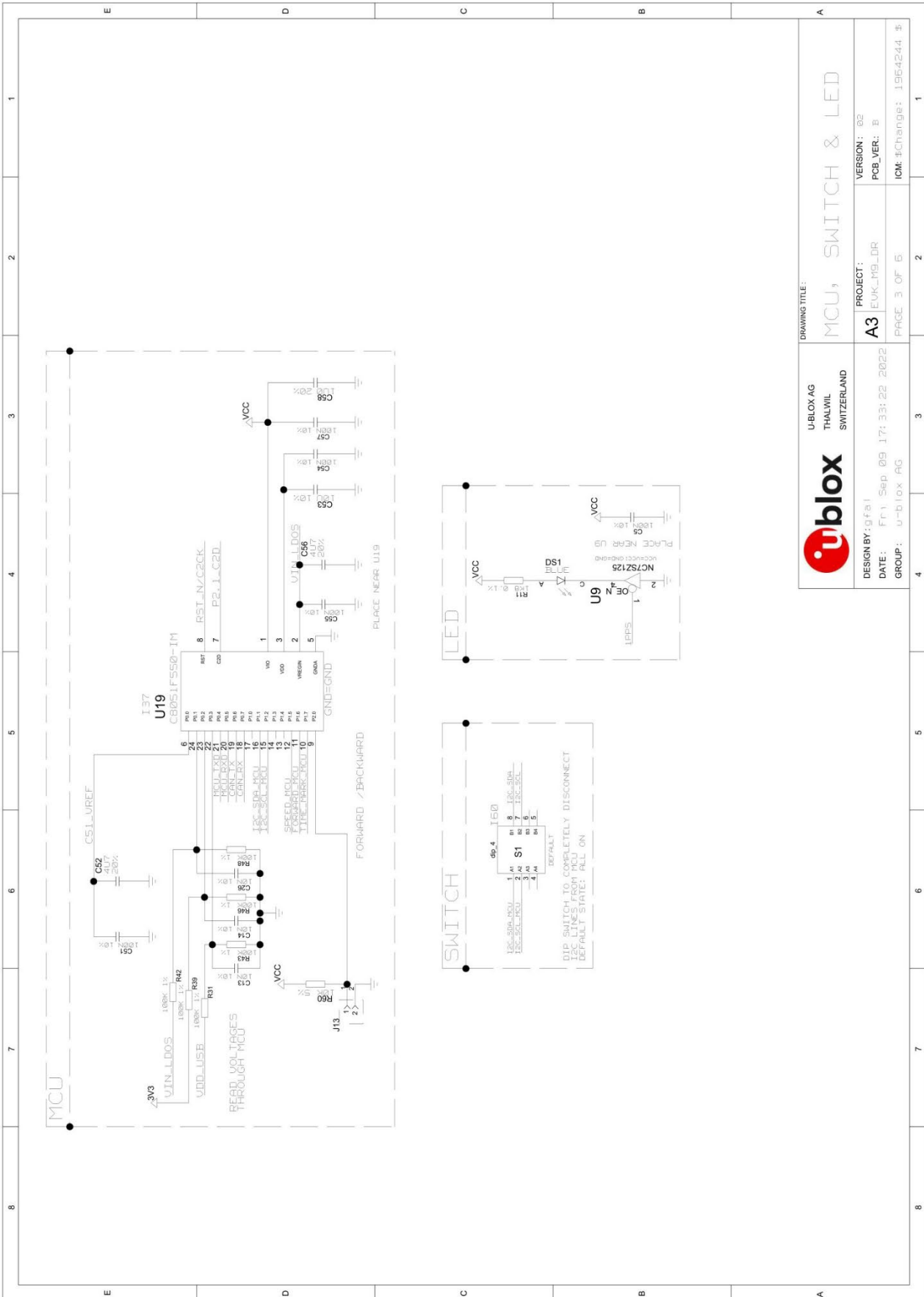
Dump File to Port: c:\temp\capture.txt | Send File | Stop | Char Count:2406 | CPS:0 | Port: 37 115200 8N1 No

## D Schematic

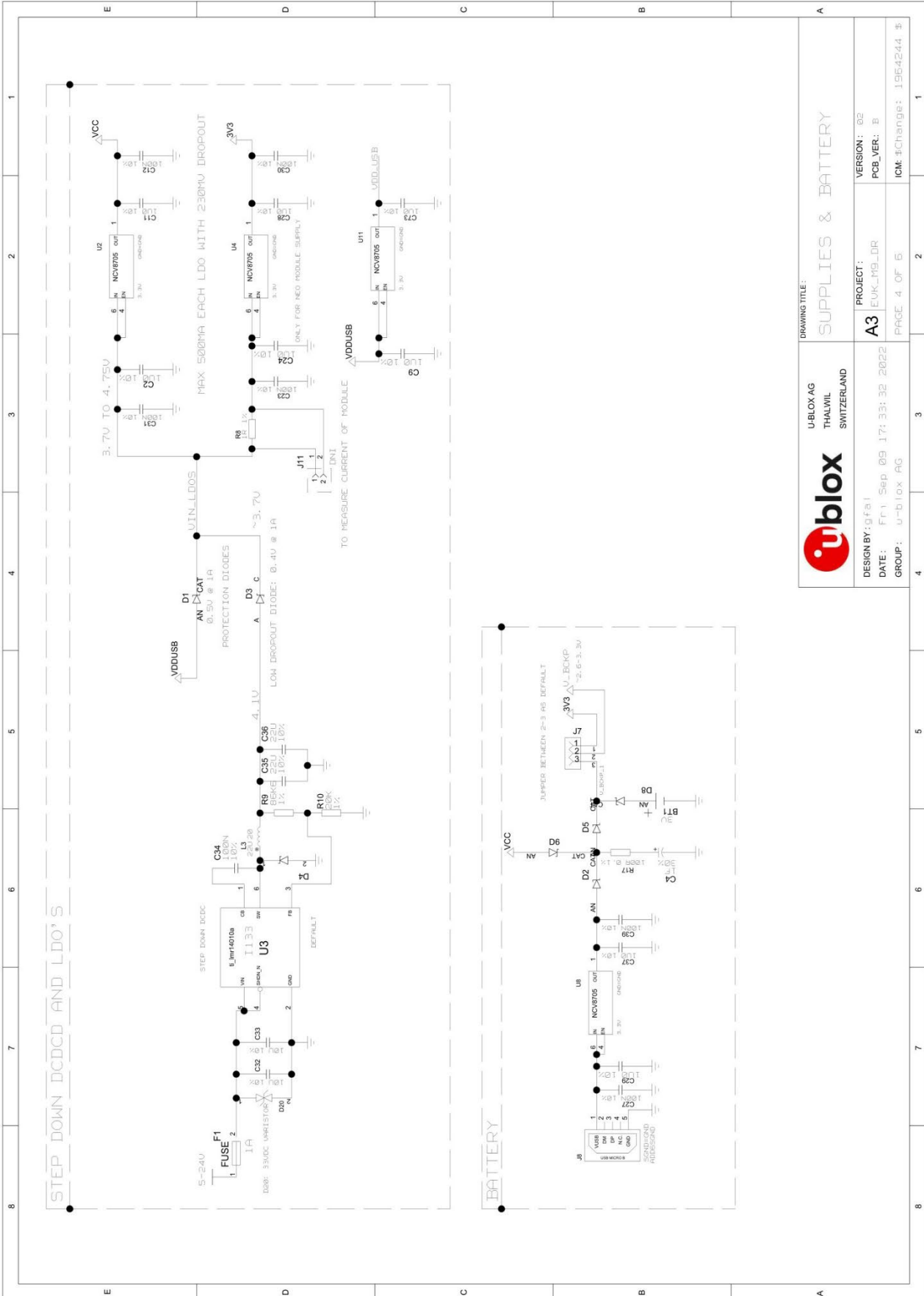
The following pages include the complete schematic for the EVK-M9DR board.



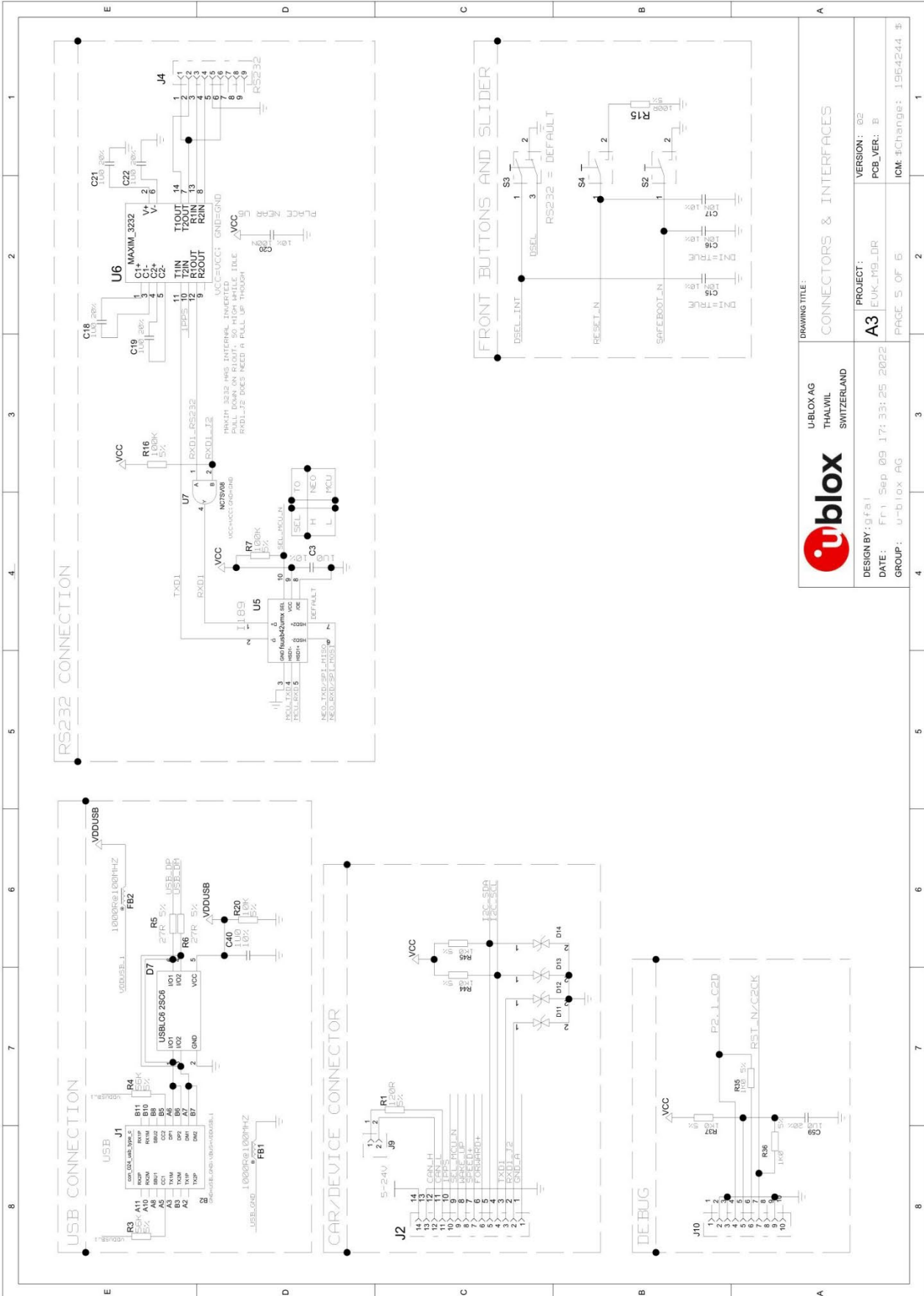
UBLOX AG THALWIL SWITZERLAND		DRAWING TITLE: NEO MODULE	
DESIGN BY: gFal DATE: Fri, Sep 09 17:33:27 2022 GROUP: u-blox AG	PROJECT: A3 EVK-M9_DR PAGE 2 OF 5	VERSION: 02 PCB_VER: B ICM: #Change: 1964244_#	



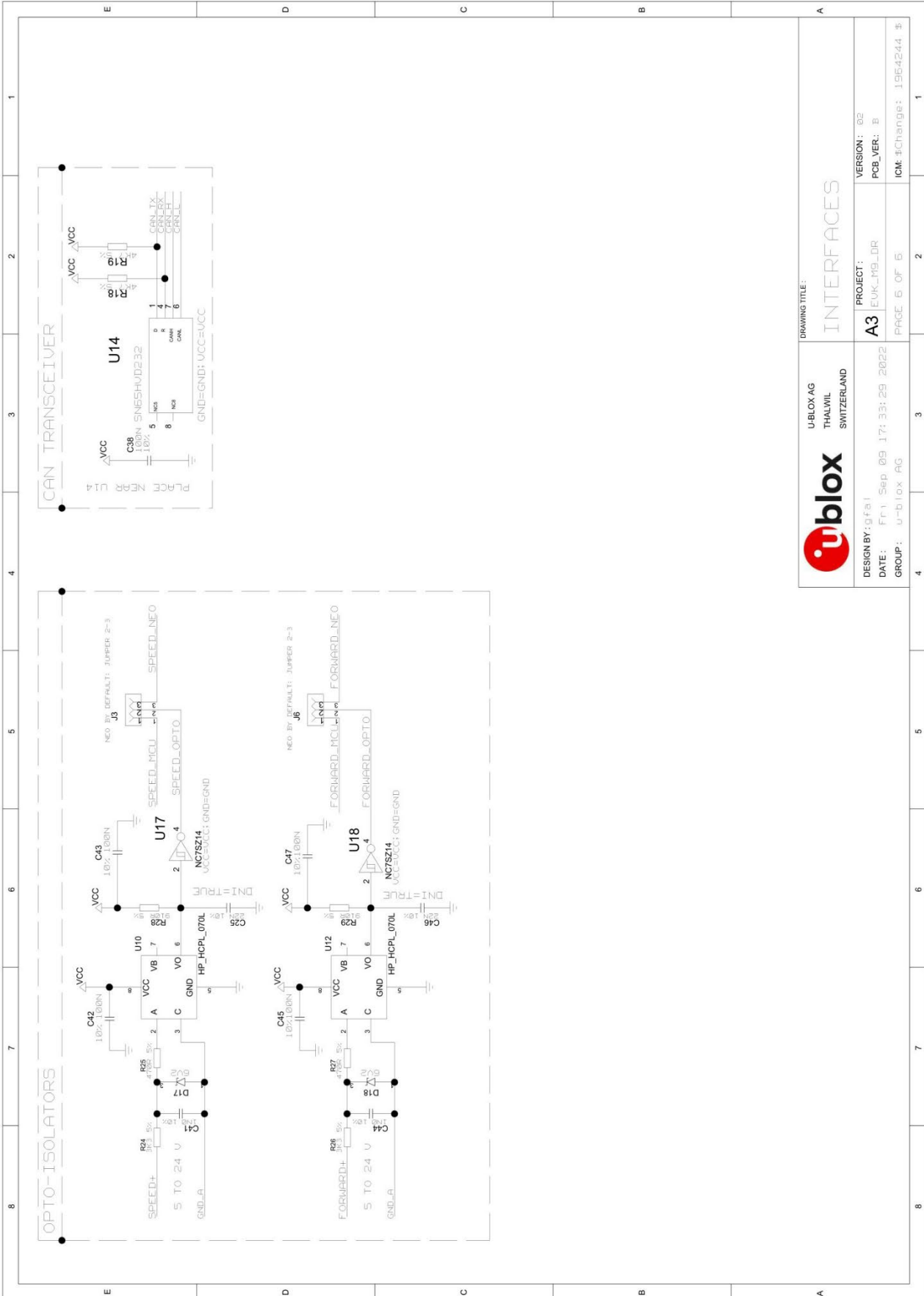
DRAWING TITLE:		MCU, SWITCH & LED	
U-BLOX AG THALWIL SWITZERLAND		VERSION: 02	
DESIGN BY: g.fai		PROJECT: A3	
DATE: Fri Sep 09 17:33:22 2022		EVK-M9_DR	
GROUP: u-blox AG		PAGE 3 OF 5	
		ICM: tChange: 19E4244_5	



		U-BLOX AG THALWIL SWITZERLAND	
DRAWING TITLE: SUPPLIES & BATTERY			
DESIGN BY: gfa1	PROJECT: A3	VERSION: 02	
DATE: Fri Sep 09 17:33:32 2022	EVK_M9_DR	PCB_VER: B	
GROUP: u-blox AG	PAGE 4 OF 6	ICM: tChange: 19E4244_5	




		U-BLOX AG THALWIL SWITZERLAND	
DRAWING TITLE: CONNECTORS & INTERFACES			
DESIGN BY: g.fai	PROJECT: A3	VERSION: 02	PCB_VER: B
DATE: Fri Sep 09 17:33:25 2022	EVK-M9_DR		
GROUP: u-blox AG	PAGE 5 OF 6	ICM: tChange: 1364244_5	



U-BLOX AG THALWIL SWITZERLAND		DRAWING TITLE:	
		INTERFACES	
DESIGN BY: g.fai	PROJECT:	VERSION: 02	1
DATE: Fri Sep 09 17:33:29 2022	A3 EVK_M9_DR	PCB_VER: B	
GROUP: ublox AG	PAGE 6 OF 6	ICM: tChangelet: 1964244_5	

## Related documents

- [1] NEO-M9V Datasheet, [UBX-2102978](#)
- [2] NEO-M9V Integration Manual, [UBX-21029776](#)
- [3] M9 MDR 2.16 Interface description, [UBX-22037308](#)
- [4] NEO-M9L Integration Manual, [UBX-20048485](#)
- [5] M9 ADR 5.15 Interface description, [UBX-22037101](#)
- [6] u-center user guide, [UBX-13005250](#)
- [7] RealTerm Serial Terminal, <https://realterm.sourceforge.io/>
- [8] Silicon Labs 8-bit Microcontroller Software, <https://www.silabs.com/products/development-tools/software/8-bit-8051-microcontroller-software>
- [9] 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](http://www.u-blox.com)).

## Revision history

Revision	Date	Name	Status / Comments
R01	03-Feb-2022	jilm	Initial release
R02	17-May-2023	Njaf, sdon	Added EVK-M9DR-0-01 with MDR 2.16. Updated schematic with battery holder (BT1). Added details about the "High Noise-level" warning flag in section Mounting the device Added note on different IMU orientation between NEO-M9V and NEO-M9L.

## Contact

Address: u-blox AG  
Zürcherstrasse 68  
8800 Thalwil  
Switzerland

For further support and contact information, visit us at [www.u-blox.com/support](http://www.u-blox.com/support).