621 Technologies Inc.

CAN-to-BT Gateway

Part No:

Reference Manual

CAN-to-BT Gateway

Table of Contents

Introduction

Innovative CAN over Bluetooth LE Gateway with OpenSYDE support

This device is an advanced **gateway** that bridges the communication between **Controller Area Network (CAN)** systems and Bluetooth Low Energy (LE) devices, enabling wireless communication and monitoring. It provides seamless integration of the **CAN bus** protocol with Bluetooth LE, allowing remote interaction with CAN networks without the need for a physical connection.

Supports 2 CAN Busses

The gateway can handle two **CAN busses**, making it highly flexible and scalable. The ability to manage two separate CAN busses is especially useful in systems that use multiple CAN networks (for example, in automotive or industrial applications where different subsystems might have their own dedicated CAN networks).

Monitoring and Producing CAN Messages

The gateway allows users to both **monitor** and **produce CAN messages**:

- **Monitoring**: It can capture CAN messages, letting users to monitor the real-time communication between ECUs (Electronic Control Units) over the CAN network and provide them for displaying on the mobile device screen. This is crucial for troubleshooting, diagnostics, and performance monitoring.
- **Producing**: The gateway can also generate and send CAN messages. This is useful for testing, simulation, or injecting specific data into the network to simulate different scenarios or control behavior in ECUs remotely.

Monitoring j1939 TP (Transport Protocol) Messages

j1939 is a higher-layer protocol built on top of the CAN bus, widely used in heavy-duty vehicles like trucks, buses, and construction machinery. The **j1939 Transport Protocol (TP)** allows large messages to be split into smaller chunks, sent over the CAN bus, and then reassembled.

This gateway can monitor **j1939 TP messages**, meaning it can capture and interpret these segmented messages, which are often critical in vehicle and industrial diagnostics. Monitoring j1939 TP messages helps engineers track and debug the transmission of large data payloads across the network, ensuring the communication is functioning as expected.

Identifying and Flashing OpenSYDE ECUs

OpenSYDE is an open-source, integrated development toolchain from STW designed for designing, developing, and managing control systems for mobile machines

The gateway has the capability to **identify** ECUs that are part of the OpenSYDE ecosystem and **flash** (update) their firmware over the air via Bluetooth LE. This is a powerful feature for remote ECU maintenance or software updates. It makes maintenance more efficient by eliminating the need for a physical connection to the CAN bus.

1. Hardware 1B0

1.1. Electrical Characteristics

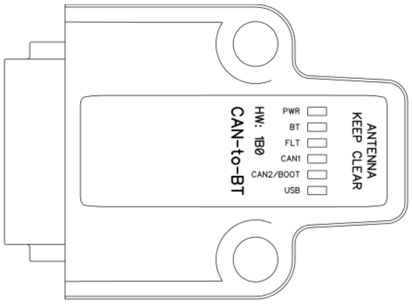
Parameter		Value	Unit
Vin	Power Supply Voltage	9 … 24 (42 max)	VDC

1.2. Connection

Mating Connector Type: Deutsch DT06-12S

	Pin		
Gnd	1	12	Vin
CAN1-L	2	11	CAN1_H
CAN2-L	3	10	CAN2-H
USB-Gnd	4	9	USB-ID
USB-Shield	5	8	USB-Vbus
USB-D-	6	7	USB-D+

1.3. Indication



LED	Colour	Description
PWR	Green	On Device is powered
BT	Blue	Flashing Bluetooth in **Advertising** mode On Bluetooth is connected
FLT	Red	Flashing Fault condition occurred On
CAN1	Green	Flashing CAN1 interface is active and communicates
CAN2/Boot	Green	Flashing CAN2 interface is active and communicates On When connected through USB indicates that the Gateway is in the bootloader mode
USB	Orange	Flashing USB interface is connected and communicates

2. Bluetooth

2.1. Specification

Bluetooth LE 5.1

2.2. Advertising

Device Name: CAN-to-BT ****************[1]

[1] **************** - the Gateway serial number

Advertised Service: 8031ffff-15a7-4d77-af22-c2926b4b2eef

2.3. GATT Server

2.3.1. *Device Information Service (DIS)*

Bluetooth SIG service

2.3.2. *Hardware Configuration and Status Service*

UUID: 80310**0100**-15a7-4d77-af22-c2926b4b2eef

2.3.2.1. CAN Bus Interface Characteristics:

2.3.2.1.1. CANx BUS Speed (bit/s)

UUIDs:

CAN Bus Interface	Characteristic	UUID
1	CAN1 BUS Speed	80310**0110**-15a7-4d77-af22-c2926b4b2eef
2	CAN2 BUS Speed	80310**0118**-15a7-4d77-af22-c2926b4b2eef

Direct Read/Write:

Size: 3 bytes

Format: Unsigned Integer

Byte:	0	1	2
Value:	MSB		LSB

Automatic Speed Detection:

Size: 1 byte

Format: Unsigned Integer

Properties:

Type	Description	Supported Values[2]	Default Value
Read	CAN Bus Speed	1000000 800000 500000 250000 125000 50000 20000 10000 0 – Start detection[3] 1 – Abort detection[3]	250000
Write[1]			
Notify[3]		0 – Detection in progress Bit-Rate value – detection complete	

[1] Writing any value other than supported will be ignored and result in an error response

[2] List of all supported values can be received by reading **List of supported CAN BUS Speeds** characteristic. Check section 2.3.2.1.4 for details.

[3] Valid only in automatic detection mode. Refer to the section 2.3.2.2 for details

2.3.2.1.2. CANx BUS Interface State

UUIDs:

CAN Bus Interface	Characteristic	UUID
1	CAN1 BUS State	8031**0111**-15a7-4d77-af22-c2926b4b2eef
2	CAN2 BUS State	8031**0119**-15a7-4d77-af22-c2926b4b2eef

Size: 1 byte
Format: Integer

Properties:

Type	Description	Supported Values	Default Value
Read	CAN Bus Interface State	0 – Error Active 1 – Error Warning 2 – Error Passive 3 – Bus-Off 4 – Stopped	4

2.3.2.1.3. Active CAN BUS Interface

UUID: 8031**0120**-15a7-4d77-af22-c2926b4b2eef
Size: 1 byte
Format: Integer

Properties:

Type	Description	Supported Values	Default Value
Read	Active CAN BUS Interface ID	0 – None 1 – CAN1 2 – CAN2	0
Write[1]			

[1] Writing any value other than the specified will be ignored and result in an error response

2.3.2.1.4. List of supported CAN BUS Speeds (bit/s)

UUID: 8031**0121**-15a7-4d77-af22-c2926b4b2eef
Size: 3*n bytes, where n is the number of values in a list
Format: Byte Array

Byte:	0	1	2	3	4	5	…	3*(n-1)+0	3*(n-1)+1	3*(n-1)+2
Value:	Value 1			Value 2				Value n		
	MSB		LSB	MSB		LSB		MSB		LSB

Properties:

Type	Description	Values
Read	Supported CAN BUS Speeds	See format section above

2.3.2.2. Description

The Gateway has two CAN2.0 bus interfaces, which can be individually configured. However, only one interface can be active at a time[1]. The supported bit-rate values for each interface can be accessed via reading the **List of supported CAN BUS Speeds** characteristic. These values can then be written to the **CANx BUS Speed** characteristic to set the speed of the corresponding interface at any time.

Additionally, the Gateway can automatically detect the bus bit-rate when both interfaces are inactive (the **Active CAN BUS Interface** characteristic being set to 0).

To automatically detect the bus bitrate:

- Write byte value 0 into **CANx BUS Speed** characteristic of the corresponding interface
- If the write operation is successful, the notification with byte value 0 will be sent indicating that the process started
- If detection is successful the new value will be stored into characteristic and the notification with the new value will be sent. Otherwise the bit-rate value will not change.

The operation can be aborted by writing a byte value of 1 to the **CANx BUS Speed** characteristic of the corresponding interface, in which case the bitrate value will remain unchanged.

[1] We anticipate supporting both busses simultaneously in the future

2.3.2.3. Sensors Characteristics:

2.3.2.3.1. Supply Voltage (mV)

UUID: 8031**0180**-15a7-4d77-af22-c2926b4b2eef
Size: 2 bytes
Format: Integer

Byte:	0	1
Value:	MSB	LSB

Properties:

Type	Description	Values	
Read	Supply Voltage	Vin < -2.0V -2.0V ≤ Vin ≤ 32.5V Vin > 32.5V	0xF000 (under-voltage) -2000…32500 mV 0xF001 (over-voltage) 0xF00F (hardware fault)

2.3.2.3.2. Internal Temperature (°C)

UUID: 8031**0181**-15a7-4d77-af22-c2926b4b2eef
Size: 2 bytes
Format: Integer (0.01°C/Bit)

Byte:	0	1
Value:	MSB	LSB

Properties:

Type	Description	Values
Read	Internal Temperature	-50.0°C … 200°C

2.3.3. *ECU Information Service*

UUID: 80310**0200**-15a7-4d77-af22-c2926b4b2eef

2.3.3.1. Characteristics:

2.3.3.1.1. Control/Status

UUID: 80310**0201**-15a7-4d77-af22-c2926b4b2eef
Size: 1 byte
Format: Integer

Properties:

Type	Description	Supported Values	Default Value
Read	Service status	See Table 1	-1
Notify			
Write[1]	Request Information	0 – Submit request 1 – Reset ECU	

[1] Writing any value other than the specified will be ignored and result in an error response

Reading this characteristic returns the current status of the service.
It also can report the status of the request upon its completion by sending a notification.
Writing ***0*** to this characteristic:

- will be ignored and will cause an error response if service is busy
- otherwise, will initiate a new request to the ECU which node id is specified in the **ECU Node Identifier** characteristic

Writing ***1*** to this characteristic will initiate a reset request to the ECU

2.3.3.1.2. ECU Node Identifier

UUID: 80310**0202**-15a7-4d77-af22-c2926b4b2eef
Size: 1 byte
Format: Unsigned Integer

Properties:

Type	Description	Supported Values	Default Value
Read	Node Identifier	0…126	0
Write[2,3]			

[2] Writing any value outside of the specified range will be ignored and result in an error response

[3] Writing to this characteristic will be ignored and will result in an error response if the unit is busy

The node ID of the ECU for which device information is requested.

2.3.3.1.3. ECU Serial Number

UUID: 8031**0203**-15a7-4d77-af22-c2926b4b2eef
Size: 6 bytes
Format: Byte Array

Properties:

Type	Description	Supported Values	Default Value
Read	Serial Number		No Data

The value is updated after the request is processed

2.3.3.1.4. ECU Name

UUID: 8031**0204**-15a7-4d77-af22-c2926b4b2eef
Size: max 17 bytes
Format: Text

Properties:

Type	Description	Supported Values	Default Value
Read	Name		No Data

The value is updated after the request is processed

2.3.3.1.5. ECU Supplier HW number (Article Number)

UUID: 8031**0205**-15a7-4d77-af22-c2926b4b2eef
Size: 4 bytes
Format: Unsigned Integer

Byte:	0	1	2	3
Value:	MSB			LSB

Properties:

Type	Description	Supported Values	Default Value
Read	Supplier HW number (Article Number)		No Data

The value is updated after the request is processed

2.3.3.1.6. ECU Supplier HW version (Article Version)

UUID: 8031**0206**-15a7-4d77-af22-c2926b4b2eef
Size: 2 bytes
Format: Char Array

Properties:

Type	Description	Supported Values	Default Value
Read	Supplier HW version (Article Version)		No Data

The value is updated after the request is processed

2.3.3.1.7. Protocol Version

UUID: 8031**0207**-15a7-4d77-af22-c2926b4b2eef
Size: 3 bytes
Format: Byte Array

Byte:	0	1	2
Value:	Major	Minor	Patch

Properties:

Type	Description	Supported Values	Default Value
Read	Protocol Version		No Data

The value is updated after the request is processed

2.3.3.1.8. Flashloader software version

UUID: 8031**0208**-15a7-4d77-af22-c2926b4b2eef
Size: 3 bytes
Format: Byte Array

Byte:	0	1	2
Value:	Major	Minor	Patch

Properties:

Type	Description	Supported Values	Default Value
Read	Flashloader software version		N0 Data

The value is updated after the request is processed

2.3.3.1.9. Flashloader protocol version

UUID: 8031**0209**-15a7-4d77-af22-c2926b4b2eef
Size: 3 bytes
Format: Byte Array

Byte:	0	1	2
Value:	Major	Minor	Patch

Properties:

Type	Description	Supported Values	Default Value
Read	Flashloader software version		No Data

The value is updated after the request is processed

2.3.3.1.10. Flash count

UUID: 8031**020a**-15a7-4d77-af22-c2926b4b2eef
Size: 4 bytes
Format: Unsigned Integer

Byte:	0	1	2	3
Value:	MSB			LSB

Properties:

Type	Description	Supported Values	Default Value
Read	Flash count		No Data

The value is updated after the request is processed

2.3.3.2. Description

This service is used to identify ECUs connected to the CAN bus. It retrieves the device information from the ECU with ID specified in the **ECU Node Identifier** characteristic. The request is initiated by writing ***0*** to the **Control/Status** characteristic. If the target ECU is online and responds to the request the received values are accessible by reading the corresponding characteristics. If the ECU is offline or is not openSYDE compatible the information characteristics return ***No Data***. While the request is in progress the status characteristic returns ***Busy*** value. After request has been completed the status changes to ***Good***, ***Flash Loader Activation Fault*** or ***Device Info Access Fault*** depending on the result. The notification with the updated status can also be sent to the client.

2.3.4. *ECU Programming Initialization Service*

UUID: 80310**0210**-15a7-4d77-af22-c2926b4b2eef

2.3.4.1. Characteristics:

2.3.4.1.1. Control/Status

UUID: 80310**0211**-15a7-4d77-af22-c2926b4b2eef
Size: 1 byte
Format: Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Service status	See Table 1	-1
Notify			
Write[1]	Submit information	0 – Submit request 1 – Reset ECU	

[1] Writing any value other than the specified will be ignored and result in an error response

Reading this characteristic returns the current status of the service.
It also can report the status of the request upon its completion by sending a notification.
Writing ***0*** to this characteristic:

- if service is busy it will be ignored and result in an error response
- otherwise, it submits information about the program area (size, address and time stamp, stored in the corresponding characteristics) to the ECU which Node ID is specified in the **ECU Node Identifier** characteristic

Writing ***1*** to this characteristic will initiate a reset request to the ECU

2.3.4.1.2. ECU Node Identifier

UUID: 80310**0212**-15a7-4d77-af22-c2926b4b2eef
Size: 1 byte
Format: Unsigned Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Node Identifier	0…126	0
Write[2,3,4]			

[2] Writing any value outside of the specified range will be ignored and result in an error response
[3] Writing to this characteristic will be ignored and will result in an error response if the unit is busy
[4] Writing to this characteristic will reset the value of **Control/Status** characteristic to ***Undefined***

The node ID of the ECU to be programmed.

2.3.4.1.3. Program Area Size

UUID: 8031**0213**-15a7-4d77-af22-c2926b4b2eef
Size: 4 bytes
Format: Unsigned Integer

Byte:	0	1	2	3
Value:	MSB			LSB

Properties:

Type	Description	Supported Values	Default Value
Read Write[1,2,3]	The Program Area Size	0…4294967295	No Data

[1] Writing any value outside of the specified range will be ignored and result in an error response
[2] Writing to this characteristic will be ignored and will result in an error response if the unit is busy
[3] Writing to this characteristic will reset the value of **Control/Status** characteristic to ***Undefined***

The size of the program area data to be flashed to the ECU.

2.3.4.1.4. Program Area Address

UUID: 8031**0214**-15a7-4d77-af22-c2926b4b2eef
Size: 4 bytes
Format: Unsigned Integer

Byte:	0	1	2	3
Value:	MSB			LSB

Properties:

Type	Description	Supported Values	Default Value
Read Write[4,5,6]	The Program Area Address	0…4294967295	No Data

[4] Writing any value outside of the specified range will be ignored and result in an error response
[5] Writing to this characteristic will be ignored and will result in an error response if the unit is busy
[6] Writing to this characteristic will reset the value of **Control/Status** characteristic to ***Undefined***

The address in the ECU memory area where the program data should be located.

2.3.4.1.5. Program Time Stamp

UUID: 8031**0215**-15a7-4d77-af22-c2926b4b2eef
Size: 6 bytes
Format: Byte Array

Byte:	0	1	2	3	4	5
Value:	Year	Month	Day	Hour	Minute	Second
	0…99	1…12	1…31	0…23	0…59	0…59

Properties:

Type	Description	Supported Values	**Default Value**
Read	The Program Time Stamp	See format section above	No Data
Write[1,2,3]			

[1] If the value of any field is outside of its specified range the writing will be ignored and result in an error response

[2] Writing to this characteristic will be ignored and will result in an error response if the unit is busy

[3] Writing to this characteristic will reset the value of **Control/Status** characteristic to ***Undefined***

2.3.4.2. Description

This service attempts to activate programming mode on the target ECU, which is identified by the node ID in the **ECU Node Identifier** characteristic. All of the following characteristics — **ECU Node Identifier**, **Program Area Size**, **Program Area Address**, and **Program Time Stamp** — must be written prior to submitting the request, in any order. The request is initiated by writing ***0*** to the **Control/Status** characteristic. While the request is in progress the status characteristic returns ***Busy*** value. After request has been completed, the status changes to ***Good***, ***Flash Loader Activation Fault*** or ***Flash Download Request Fault*** depending on the result.

Successfully switching the ECU into programming mode initializes the **ECU Program Flashing Service** allowing transfer of the program to the ECU.

2.3.5. *ECU Program Flashing Service*

UUID: 8031**0220**-15a7-4d77-af22-c2926b4b2eef

2.3.5.1. Characteristics:

2.3.5.1.1. Control/Status

UUID: 8031**0221**-15a7-4d77-af22-c2926b4b2eef
Size: 1 byte
Format: Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Service status	See Table 1	-1
Notify			
Write[1]	Submit information	0 – Submit request 1 – Reset ECU	

[1] Writing any value other than the specified will be ignored and result in an error response

Reading this characteristic returns the current status of the service.
It also can report the status of the request upon its completion by sending a notification.
Writing ***0*** to this characteristic:

- will be ignored and will cause an error response if service is busy
- otherwise, it transmits data packet stored in **Packet Data** characteristic to the ECU

Writing ***1*** to this characteristic will initiate a reset request to the ECU

2.3.5.1.2. Packet Sequence Number

UUID: 8031**0222**-15a7-4d77-af22-c2926b4b2eef
Size: 2 bytes
Format: Unsigned Integer

Byte:	0	1
Value:	MSB	LSB

Properties:

Type	Description	Supported Values	Default Value
Read	Packet Sequence Number	0…65535	0

The packet number that the service is expecting from the client. This characteristic is read-only. It automatically increments after the packet is successfully received which means that the checksum from **Packet Checksum** matches the calculated one.
If transmission is unsuccessful this value remains the same and the packet needs to be retransmitted.

2.3.5.1.3. Packet Checksum

UUID: 8031**0223**-15a7-4d77-af22-c2926b4b2eef
Size: 2 bytes
Format: Unsigned Integer

Byte:	0	1
Value:	MSB	LSB

Properties:

Type	Description	Supported Values
Write[1]	The Packet Checksum	0…65535

[1] Writing any value outside of the specified range will be ignored and result in an error response

The packet checksum is calculated using the 16 bit version of Fletcher's algorithm. Check out *https://en.wikipedia.org/wiki/Fletcher's_checksum* website for more information.

2.3.5.1.4. Packet Data

UUID: 8031**0224**-15a7-4d77-af22-c2926b4b2eef
Size: 1…500 bytes
Format: Byte Array

Properties:

Type	Description	Supported Values
Write	The Packet Data	

The segment of program data.

2.3.5.2. Description

This service provides the functionality for transferring the program code to the ECU.

After successfully switching ECU into the programming mode the Packet Sequence Number will be reset to 0. The data is sent in packets of up to 500 bytes. The packet preparation involves writing payload data into **Packet Data** characteristic and the data checksum in the **Packet Checksum** characteristic in any order. The packet transmission is initiated by writing ***0*** to the **Control/Status** characteristic. While the request is in progress the status characteristic returns ***Busy*** value. If the transmission completed successfully the **Packet Sequence Counter** increments and the status is changed to ***Good***. Otherwise, the **Packet Sequence Counter** remains the same and the status returns the possible reason of the fault. The status remains valid only while all the other characteristics retain the values used during transmission. If any of characteristic is changed the status will change to ***Undefined***. If the **Packet Sequence Counter** reaches its maximum possible value of 65535 it will roll over to 0 at the next increment.

Status codes:

Undefined	Current set of values is not processed
Good	The transmission completed successfully
Checksum Error	A discrepancy between the **Packet Checksum** characteristic value and the data checksum detected

When all program code is transferred to the ECU (the number of transferred bytes is equal to the value of **Program Size** characteristic) the ECU resets and the status returns ***Transaction Complete*** value.

2.3.6. ECU Flash Block Info Service

UUID:	8031**0230**-15a7-4d77-af22-c2926b4b2eef

2.3.6.1. Characteristics:

2.3.6.1.1. Control/Status

UUID:	8031**0231**-15a7-4d77-af22-c2926b4b2eef
Size:	1 byte
Format:	Integer
Properties:	

Type	Description	Supported Values	Default Value
Read	Service status	See Table 1	-1
Notify			
Write[1]	Submit information	0 (Submit request) 1 (Reset ECU)	

[1] Writing any value other than the specified will be ignored and result in an error response

Reading this characteristic returns the current status of the service.
It also can report the status of the request upon its completion by sending a notification.
Writing ***0*** to this characteristic:

- will be ignored and will cause an error response if service is busy
- will initiate a new request to the ECU which node id is specified in the **ECU Node Identifier** characteristic

Writing ***1*** to this characteristic will initiate a reset request to the ECU

2.3.6.1.2. ECU Node Identifier

UUID:	8031**0232**-15a7-4d77-af22-c2926b4b2eef
Size:	1 byte
Format:	Unsigned Integer
Properties:	

Type	Description	Supported Values	Default Value
Read	Node Identifier	0…126	0
Write[2,3]			

[2] Writing any value other than the specified will be ignored and result in an error response

[3] Writing to this characteristic will be ignored and will result in an error response if the unit is busy

The node ID of the ECU for which device information is requested.

2.3.6.1.3. ECU Flash Block Identifier

UUID: 80310**0233**-15a7-4d77-af22-c2926b4b2eef
Size: 1 byte
Format: Unsigned Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Flash Block Identifier	0…255	No Data
Write[1]			

[1] Writing any value other than the specified will be ignored and result in an error response

The ID of the ECU Flash Block for which the information is being required.

2.3.6.1.4. ECU Flash Block Start Address

UUID: 80310**0234**-15a7-4d77-af22-c2926b4b2eef
Size: 4 bytes
Format: Unsigned Integer

Byte:	0	1	2	3
Value:	MSB			LSB

Properties:

Type	Description	Supported Values	Default Value
Read	The Flash Block Start Address	0…4294967295	No Data

The value is updated after the request is processed.

2.3.6.1.5. ECU Flash Block End Address

UUID: 80310**0235**-15a7-4d77-af22-c2926b4b2eef
Size: 4 bytes
Format: Unsigned Integer

Byte:	0	1	2	3
Value:	MSB			LSB

Properties:

Type	Description	Supported Values	Default Value
Read	The Flash Block End Address	0…4294967295	No Data

The value is updated after the request is processed.

2.3.6.1.6. ECU Flash Block Signature Valid

UUID:	8031**0236**-15a7-4d77-af22-c2926b4b2eef
Size:	1 byte
Format:	Unsigned Integer

Properties:

Type	Description	Supported Values	
Read	The Flash Block Signature Valid flag	0 – Invalid 1 – Valid	No Data

The value is updated after the request is processed.

2.3.6.1.7. ECU Flash Block Application Version

UUID:	8031**0237**-15a7-4d77-af22-c2926b4b2eef
Size:	max 16 bytes
Format:	Text

Properties:

Type	Description	Supported Values	Default Value
Read	The Flash Block Application Version		No Data

The value is updated after the request is processed.

2.3.6.1.8. ECU Flash Block Application Build Date

UUID:	8031**0238**-15a7-4d77-af22-c2926b4b2eef
Size:	11 bytes
Format:	Text (Mmm dd yyyy)

Properties:

Type	Description	Supported Values	Default Value
Read	The Flash Block Application Build Date		No Data

The value is updated after the request is processed.

2.3.6.1.9. ECU Flash Block Application Build Time

UUID: 8031**0239**-15a7-4d77-af22-c2926b4b2eef
Size: 8 bytes
Format: Text (hh:mm:ss)

Properties:

Type	Description	Supported Values	Default Value
Read	The Flash Block Application Build Time		No Data

The value is updated after the request is processed.

2.3.6.1.10. ECU Flash Block Application Name

UUID: 8031**023a**-15a7-4d77-af22-c2926b4b2eef
Size: max 32 bytes
Format: Text

Properties:

Type	Description	Supported Values	Default Value
Read	The Flash Block Application Name		No Data

The value is updated after the request is processed.

2.3.6.1.11. ECU Flash Block Additional Info

UUID: 8031**023b**-15a7-4d77-af22-c2926b4b2eef
Size: max 255 bytes
Format: Byte Array

Properties:

Type	Description	Supported Values	Default Value
Read	The Flash Block Additional Info		No Data

The value is updated after the request is processed.

2.3.6.2. Description

This service allows access to information about the software currently present in the ECU Flash memory.

This information is organized in blocks. In order to access any block information, the ECU Node Identifier and Flash Block Identifier values must be provided followed by writing 0 to the Control/Status characteristic. If the request is completed successfully, the status characteristic will change to ***Good***. Otherwise, it will indicate the possible reason for the fault.

To obtain information from all blocks, the request should be made for each block ID, beginning from 0. Once all blocks are read, the status characteristic will return ***Flash Block ID Out of Range*** value.

2.3.7. CAN Bus Monitor Service

UUID: 8031**0300**-15a7-4d77-af22-c2926b4b2eef

This service consists of number of similar sets of characteristics called Message boxes. Each Message Box is intended for processing a particular CAN identifier and has the following structure:

- Operation Mode
- Status
- Rx Filter
- Packet
- Frame Count

2.3.7.1. UUID assignment:

Message Box #	Characteristic	UUID
1	Operation Mode	8031**0309**-15a7-4d77-af22-c2926b4b2eef
	Status	8031**030a**-15a7-4d77-af22-c2926b4b2eef
	Rx Filter	8031**030b**-15a7-4d77-af22-c2926b4b2eef
	Packet	8031**030c**-15a7-4d77-af22-c2926b4b2eef
	Frame Count	8031**030d**-15a7-4d77-af22-c2926b4b2eef
2	Operation Mode	8031**0311**-15a7-4d77-af22-c2926b4b2eef
	Status	8031**0312**-15a7-4d77-af22-c2926b4b2eef
	Rx Filter	8031**0313**-15a7-4d77-af22-c2926b4b2eef
	Packet	8031**0314**-15a7-4d77-af22-c2926b4b2eef
	Frame Count	8031**0315**-15a7-4d77-af22-c2926b4b2eef
3	Operation Mode	8031**0319**-15a7-4d77-af22-c2926b4b2eef
	Status	8031**031a**-15a7-4d77-af22-c2926b4b2eef
	Rx Filter	8031**031b**-15a7-4d77-af22-c2926b4b2eef
	Packet	8031**031c**-15a7-4d77-af22-c2926b4b2eef
	Frame Count	8031**031d**-15a7-4d77-af22-c2926b4b2eef
4	Operation Mode	8031**0321**-15a7-4d77-af22-c2926b4b2eef
	Status	8031**0322**-15a7-4d77-af22-c2926b4b2eef
	Rx Filter	8031**0323**-15a7-4d77-af22-c2926b4b2eef
	Packet	8031**0324**-15a7-4d77-af22-c2926b4b2eef
	Frame Count	8031**0325**-15a7-4d77-af22-c2926b4b2eef

Note: Future releases are expected to increase the number of message boxes to 8.
Additional functionality to be developed

2.3.7.2. Characteristics:

2.3.7.2.1. Message box Operation Mode

UUID: See section 2.3.7.1
Size: 1 byte
Format: Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Operation Mode	0 – Inactive 1 – Send 2 – Receive	0
Write[1]			

[1] Writing any value other than supported will be ignored and result in an error response

2.3.7.2.2. Message box Status

UUID: See section 2.3.7.1
Size: 1 byte
Format: Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Status Value	TBD	0

2.3.7.2.3. Message box Rx Filter

UUID: See section 2.3.7.1
Sizes: 3 bytes (IDE flag = 0)
5 bytes (IDE flag = 1)
Format: Byte Array

Standard (11-bit) CAN identifier

Byte:	0									1	2
	Flags									**ID**	
	Bit	7	6	5	4	3	2	1	0	MSB	LSB
Value:	Flag	IDE	RTR							0x0000 … 0x07FF	

Extended (29-bit) CAN identifier

Byte:	0									1	…	4
	Flags									**ID**		
	Bit	7	6	5	4	3	2	1	0	MSB	…	LSB
Value:	Flag	IDE	RTR							0x00000000 … 0x1CFFFFFF		

Flag	**Value**
IDE	0 – Standard ID 1 – Extended ID
RTR	0 – Data Frame ~~1 - Remote Transmission Request~~

This revision of software does not support filtering of Remote Transmission Request so setting RTR flag is ignored and it is always read as 0.

Properties:

Type	Description	Supported Values	Default Value
Read	Rx Filter	See format section above	No Data
Write			

This characteristic contains the CAN message ID filter which is used when the Message box is in the receiving mode. It supports Standard (11-bit) and Extended (29-bit) identifiers. The IDE flag (bit 7) specifies the type of identifier. The RTR flag (bit 6) selects between filtering Data Packets or Remote Transmission Requests.

2.3.7.2.4. Message box Packet

UUID: See section 2.3.7.1
Sizes: 8 … 16 bytes (IDE flag = 0) depending on the data length
10 … 18 bytes (IDE flag = 1) depending on the data length
Format: Byte Array

Standard (11-bit) CAN identifier

Byte:	0									1	2	3	4 … 11	12	…	15
	Flags									**ID**		**Data Length**	**Data**	**Cycle Time (mS)**		
	Bit	7	6	5	4	3	2	1	0	MSB	LSB	0…8	0 to 8 bytes	MSB		LSB
Value:	Flag	IDE	RTR							0x0000 … 0x07FF				0…4294967295		

Extended (29-bit) CAN identifier

Byte:	0									1	…	4	5	6 … 13	14	…	17
	Flags									**ID**			**Data Length**	**Data**	**Cycle Time (mS)**		
	Bit	7	6	5	4	3	2	1	0	MSB	…	LSB	0…8	0 to 8 bytes	MSB	…	LSB
Value:	Flag	IDE	RTR							0x00000000 … 0x1CFFFFFF					0…4294967295		

Flag	Value
IDE	0 – Standard ID 1 – Extended ID
RTR	0 – Data Frame ~~1 – Remote Transmission Request~~

This revision of software does not support filtering of Remote Transmission Request so setting RTR flag is ignored and it is always read as 0.

Properties:

Type	Description	Supported Values	Default Value
Read	**Send Mode:** Transmitting packet **Receive Mode:** Last Received packet	See format section above	No Data
Write	**Send Mode:** Transmitting packet **Receive Mode:** *Not Allowed*		
Notify	**Send Mode:** Transmitting packet **Receive Mode:** Last Received packet		

2.3.7.2.5. Frame Counter

UUID: See section 2.3.7.1

Read:
Size: 4 bytes
Format: Unsigned Integer

Byte:	0	1	2	3
Value:	MSB			LSB

Write:
Size: 1 byte
Format: Unsigned Integer

Properties:

Type	Description	Supported Values	Default Value
Read	The Frame Counter	0…4294967295	0
Write[1]		0	

[1] Writing any value other than the specified will be ignored and result in an error response

The value of this characteristic increments with each transmission or reception of a message in the corresponding message box. It can be reset by writing a byte value 0.

2.3.7.3. Description

Each message box can be set to one of the following modes:

Inactive mode:
The message box is idle. The **Message box Rx Filter** and **Message box Packet** characteristics can be configured with initial values before switching to one of the active modes described below.

Send mode:
In order to select this mode, the **Message box Packet** characteristic must be initialized. The outgoing message will be based on the value of this characteristic. The “Cycle Time” field of the packet will be used as a message retransmission period. If this value is 0 the only one message will be transmitted and the **Operation Mode** characteristic will be switched to the ***Inactive*** mode. The **Frame Counter** will increment each time the message is sent. Also notification from the **Message box Packet** characteristic will be sent if the client device is subscribed.
Both **Message box Rx Filter** and **Message box Packet** characteristics may be updated without switching to the ***Inactive*** mode. The **Message box Rx Filter** characteristic value is ignored in this mode. If the **Message box Packet** characteristic is updated the new message will be scheduled for transmission at the next event.
Multiple message boxes can be configured to transmit the message with the identical CAN ID.

Receive mode:

In order to select this mode, the **Message Box Rx Filter** characteristic must be initialized. Multiple message boxes cannot be configured to receive messages with identical CAN IDs.

When a message matching the filter is received, it will be placed in the **Message Box Packet** characteristic, and a notification will be sent if the client device is subscribed. The **Frame Counter** will increment each time the message is sent.

Modifying the **Message Box Rx Filter** or **Message box Packet** characteristic is not permitted in this mode.

2.3.8.j1939 TP Monitor Service

UUID: 8031**0380**-15a7-4d77-af22-c2926b4b2eef

This service consists of number of similar sets of characteristics called Message boxes. The Message Box is designed for processing one specific CAN identifier and has the following structure:

- Operation Mode
- Control/Status
- Rx Filter Source Address
- Rx Filter Destination Address
- Rx Message Data

2.3.8.1. UUID assignment:

Message Box #	Characteristic	UUID
1	Operation Mode	8031**0389**-15a7-4d77-af22-c2926b4b2eef
	Control/Status	8031**038a**-15a7-4d77-af22-c2926b4b2eef
	Rx Filter - Source Address	8031**038b**-15a7-4d77-af22-c2926b4b2eef
	Rx Filter - Destination Address	8031**038c**-15a7-4d77-af22-c2926b4b2eef
	Rx Message Data	8031**038d**-15a7-4d77-af22-c2926b4b2eef

2.3.8.2. Characteristics:

2.3.8.2.1. Message box Operation Mode

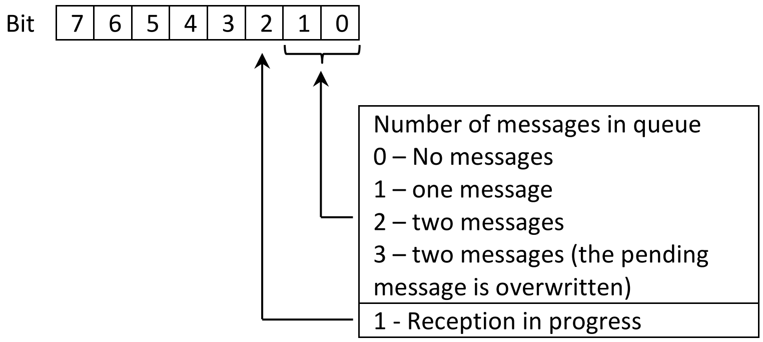
UUID: See section 2.3.8.1
Size: 1 byte
Format: Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Operation Mode	0 – Inactive 2 – Receive	0
Write[1]			

[1] Writing any value other than supported will have no effect and will produce an error response

2.3.8.2.2. Message box Control/Status

UUID: See section 2.3.8.1
Size: 1 byte
Format: Integer



Number of messages in queue
0 – No messages
1 – one message
2 – two messages
3 – two messages (the pending message is overwritten)

1 - Reception in progress

Properties:

Type	Description	Supported Values	Default Value
Read	Control Value	See format section above	0
Write		0 – Discard current message 1 – Reset current message fragment counter	

2.3.8.2.3. Rx Filter - Source Address

UUID: See section 2.3.8.1
Size: 1 byte
Format: Unsigned Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Filter value for the Source Address of receiving j1939 TP Message	0…255	0
Write[1]			

[1] Writing to this characteristic will be ignored and will cause an error response if the unit is not in ***Inactive*** mode

2.3.8.2.4. Rx Filter - Destination Address

UUID: See section 2.3.8.1
Size: 1 byte
Format: Unsigned Integer
Properties:

Type	Description	Supported Values	Default Value
Read	Filter value for the Destination Address of receiving j1939 TP Message	0…255	255
Write[1]			

[1] Writing to this characteristic will be ignored and will cause an error response if the unit is not in ***Inactive*** mode

2.3.8.2.5. Rx Message Data

UUID: See section 2.3.8.1
Size: Varies depending on the fragment type and the data length. The maximum size is 258 bytes
Format: Byte Array

Fragment #0 (Header - Message Info)

Byte:	0	1	2	3	…	6	7	8	9
Value:	**Fragment #**	**Checksum**[2]		**Timestamp (mS)**			**Message Size (bytes)**		**Status**
		MSB	LSB	MSB		LSB	MSB	LSB	See table below
	0	0x0000 … 0xFFFF		0…4294967295			0x0000 … 0xFFFF		

[2] Checksum is calculated over data block (bytes 3 … fragment length) using 16 bit version of Fletcher's algorithm

Value	Status	Description
0	Good	Message received successfully
1	Out of sequence fault	Message reception was disrupted and its reassembly failed due to lost or corrupted data packet
2	Data size mismatch fault	Number of data bytes received differs from the number specified in the TP.CM
3	Timeout	Message reception was canceled because the consecutive frame was not received within 750ms.

Fragment #1…7 (Message Data)

Byte:	0	1	2	3	…	258 (max)
Value:	**Fragment #**	**Checksum**[3]		**Data**		
		MSB	LSB			
	1…7	0x0000 … 0xFFFF				

[3] Checksum is calculated over data block (bytes 3 … fragment length) using 16 bit version of Fletcher's algorithm

Properties:

Type	Description	Supported Values	Default Value
Read	Active Message	See format section above	No Data

2.3.8.3. Description

This service provides the functionality for receiving of j1939 TP multi-packet messages.

In order to configure and activate a message box the following steps should be performed:

- Ensure that the message box is inactive (**Operation Mode** characteristic reads ***0***)
- Configure the receive filter by writing Source and Destination addresses values into corresponding characteristics

Switch Message Box to Receive Mode (write ***2*** into **Operation Mode** characteristic) to activate the message box

- Each received multi-packet message is added to the message queue, which can store the data of up to 2 messages (with the queue depth is 2).

  The first received message becomes an ***active*** (available for reading from **Message Data** characteristic) and the message counter in the **Control/Status** characteristic increments from 0 to 1. If another message received while the first (active) message is not consumed it gets placed in a queue and becomes a ***pending*** message and the message counter in the **Control/Status** characteristic increments from 1 to 2. In case when another message is received while the queue is full (contains 2 messages) the ***pending*** message gets overwritten and the message counter in the **Control/Status** characteristic gets value 3.

Receive process

A reception session starts when the j1939 TP.CM packet with ID matching the message box filter values and with control byte (byte 0) containing BAM value is received. The j1939 priority of the received TP.CM packet is recorded for use in filtering of subsequent TP.DT packets. The ***Reception in progress*** flag (bit 3) in the **Control/Status** characteristic changes to ***1*** indicating reception in progress. The message box will continue receiving j1939 TP.DT packets until all the message packets/bytes are received. At a completion the ***Reception in progress*** flag (bit 3) in the **Control/Status** characteristic changes to ***0*** and the counter of the messages in the queue increments. If the queue is full the new message overwrites the pending message in the queue and the message counter gets value 3 indicating the fact. If, during reception session, the received j1939 TP.DT packet has an unexpected packet number (byte 0) the session aborts and the ***Packet out of sequence fault*** flag is set in the **Control/Status** characteristic.

Read Message

The earliest received (active) message can be retrieved by reading from the **Rx Message Data** characteristic. The message data is stored in series of fragments of a maximum size of 255 bytes. This format is used because the maximum length of the j1939 TP message exceeds the size limit of the BLE GATT characteristic. Each fragment consists of a sequence number, checksum and a data field.

There are 2 types of message fragments:

- Header Fragment. It provides information about the message such as timestamp and the length of the message data. It always has a sequence number 0.
- Data Fragments. These fragments provide the message data in a sequence of blocks. They have numbers from 1 to 7.

When a new message became active, the fragment number counter is set to 0. Each reading from a **Rx Message Data** characteristic causes fragment number incrementing by 1. When the last fragment is read the fragment sequence number resets to 0. This allows rereading the message in case of any issue. The fragment sequence counter could also be forced to reset to 0 by writing ***1*** into the **Control/Status** characteristic.

When all fragments are read and the whole message is reconstructed the active message can be discarded by writing ***0*** into the **Control/Status** characteristic. If there is a pending message in a queue it becomes active and available for reading. If there is no messages in the queue, reading from a **Rx Message Data** characteristic will return ***No Data*** value.

Table 1. Status Codes:

Code	Description
-1	Undefined
0	Busy
1	Good
2	Transaction Complete
3	ECU Inaccessible
4	Access Denied
5	Value Out of Range
6	No Active CAN Bus
7	Checksum Error
8	Flash Loader Activation Fault
9	Device Info Access Fault
10	Flash Download Request Fault
11	Flash Data Transfer Fault
12	Data Transfer Exit Fault
13	System Reset Fault
14	Flash Block ID Out of Range
15	Flash Block Info Access Fault
	--- TBD ---

3. Using the Gateway

Use DIS service to identify the connected Gateway revision and supported functionality.

This Gateway allows flashing one program area at a time. If the firmware file has multiple areas the flashing process needs to be repeated for each area.

3.1 ECU Flashing procedure (Gateway software rev. 1A0)

- Connect Gateway to the CAN bus. The **Power On** LED turns on, the **BT Status** LED starts flashing indicating that the Gateway is in the advertising mode
- On the client device look for and connect to the Bluetooth device with the attributes (name, service, etc.) specified in section 2.2
- After the Bluetooth connection is established the **BT Status** LED on Gateway stops flashing and turns on
- Configure target CAN bus Interface by writing the bit rate and activate it
- Write the target ECU Node ID
- Check the target ECU information if needed
- Submit the firmware area address, length and firmware timestamp to activate ECU program mode
- Start writing firmware area data using blocks of 500 bytes or smaller. The status of operation is returned through the write response code. If the data block is accepted the Gateway returns success status. Otherwise it returns the error response code and the data is ignored. In this case the block should be resent until it is accepted
- Continue writing the data until all firmware area data is sent
- When all data is sent the ECU will restart automatically

4. Updating the firmware

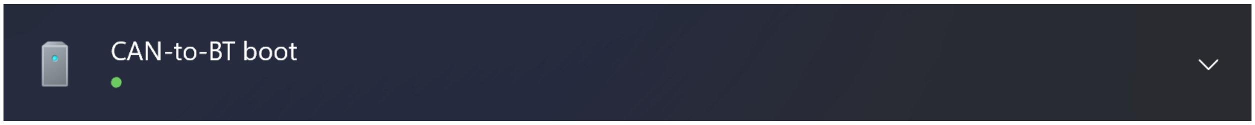
The Gateway has a built-in recovery mode that allows recovering or updating the firmware via the USB port. To update the firmware, make sure you have the following:

- Terminal software utility such as **AuTerm** or **mcumgr-client** installed on your computer
- Deutsch-to-USB interface cable

4.1 Entering Recovery mode

To activate recovery mode, disconnect the Gateway from the main power and connect it to a computer using the Deutsch-to-USB cable.

The new USB device named "CAN-to-BT boot" should appear in a list of devices.



CAN-to-BT boot

When the Gateway enters firmware recovery mode, the blue **BT** LED will stay off and green **CAN2/Boot** LED will turn on to indicate activation.

4.2 Flashing procedure

Prerequisites

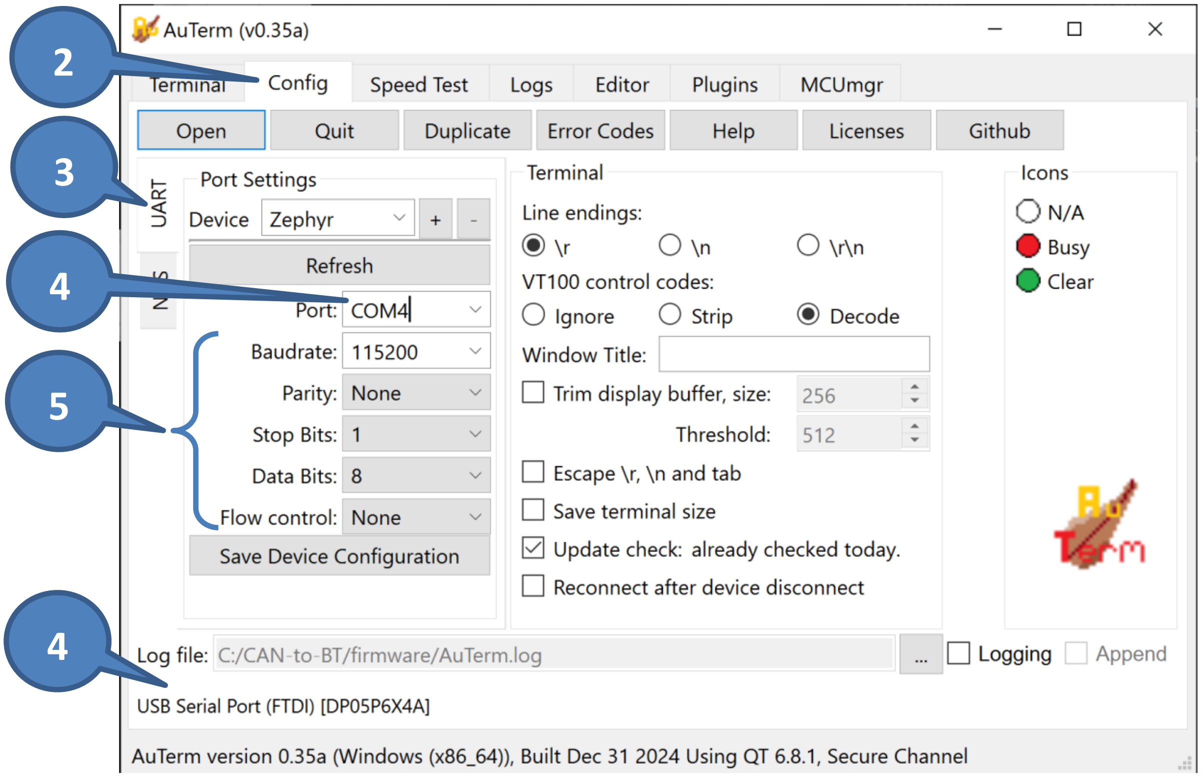
- Correct firmware binary file is saved on the computer.
  The firmware file name follows the format:

  <project name>_<hardware revision>_<firmware revision>.bin

- The flashing software tool is installed
- The Gateway is switched to recovery mode, as described in section 4.1

4.2.1 Using AuTerm

1. Launch AuTerm application
2. Navigate to the **Config** tab
3. Ensure the **UART** Port tab is selected
4. In the **Port Settings**, select the COM port assigned to the Gateway (e.g., COM4). If the port is not shown in the drop down list, ensure that the Gateway is connected and in the Recovery mode, then press **Refresh** and check again. Once the correct port is selected, the Port Type should automatically update to ***"USB Serial Port (FTDI) [DP05P6X4]"***.
5. Verify that the remaining **Port Settings** match the configuration shown in the image below:



Port Settings

Device: Zephyr

Port: COM4

Baudrate: 115200

Parity: None

Stop Bits: 1

Data Bits: 8

Flow control: None

Terminal

Line endings: ● \r ○ \n ○ \r\n

VT100 control codes: ○ Ignore ○ Strip ● Decode

Window Title:

□ Trim display buffer, size: 256

Threshold: 512

□ Escape \r, \n and tab

□ Save terminal size

☑ Update check: already checked today.

□ Reconnect after device disconnect

Icons

○ N/A

🔴 Busy

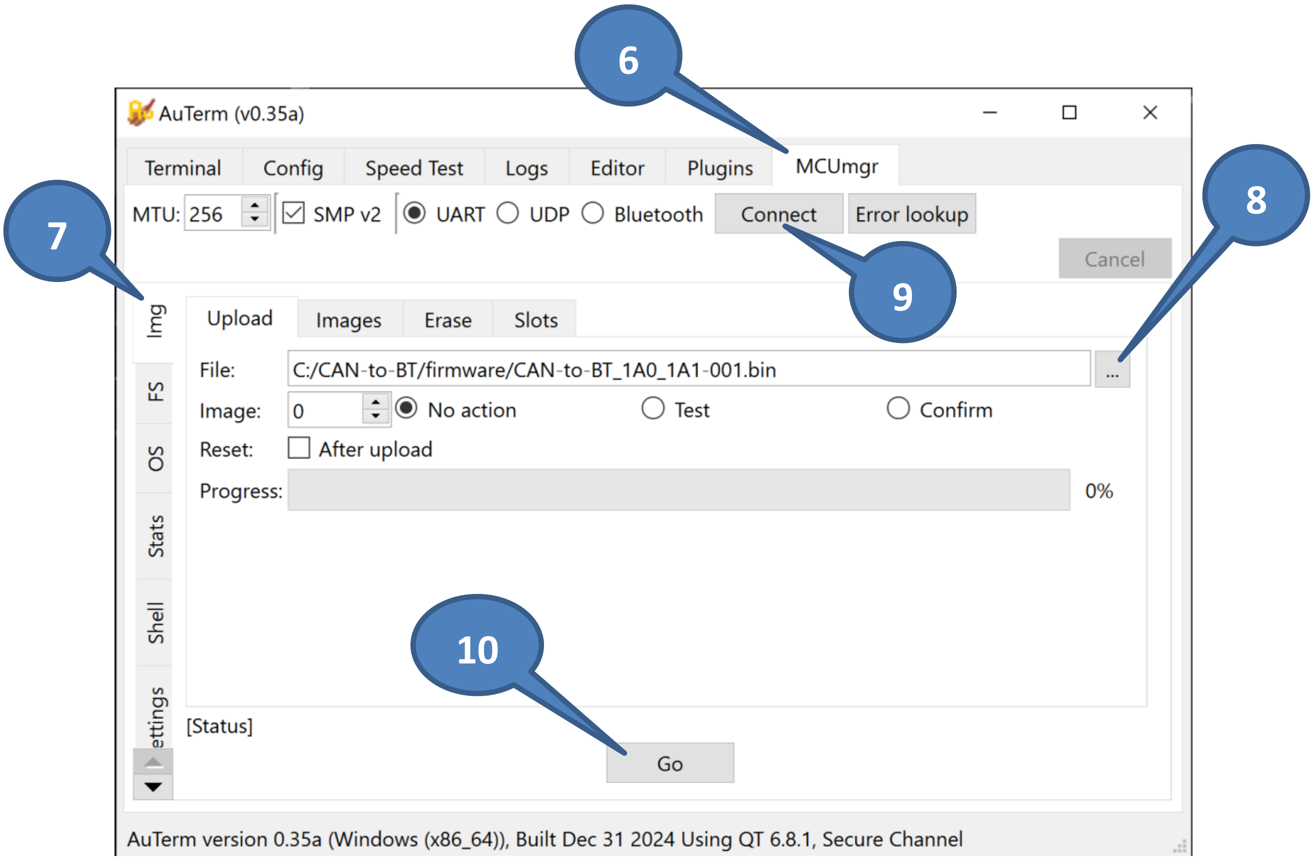
🟢 Clear

Log file: C:/CAN-to-BT/firmware/AuTerm.log □ Logging □ Append

USB Serial Port (FTDI) [DP05P6X4A]

AuTerm version 0.35a (Windows (x86_64)), Built Dec 31 2024 Using QT 6.8.1, Secure Channel

6. Navigate to the **MCUmgr** tab
7. Ensure the **Img** tab is selected
8. Select the firmware file to be uploaded
9. Click the **Connect** button to establish a connection with the Gateway
10. Press **Go** button to start flashing process



AuTerm (v0.35a)

MTU: 256 ☑ SMP v2 ● UART ○ UDP ○ Bluetooth

File: C:/CAN-to-BT/firmware/CAN-to-BT_1A0_1A1-001.bin

Image: 0 ● No action ○ Test ○ Confirm

Reset: □ After upload

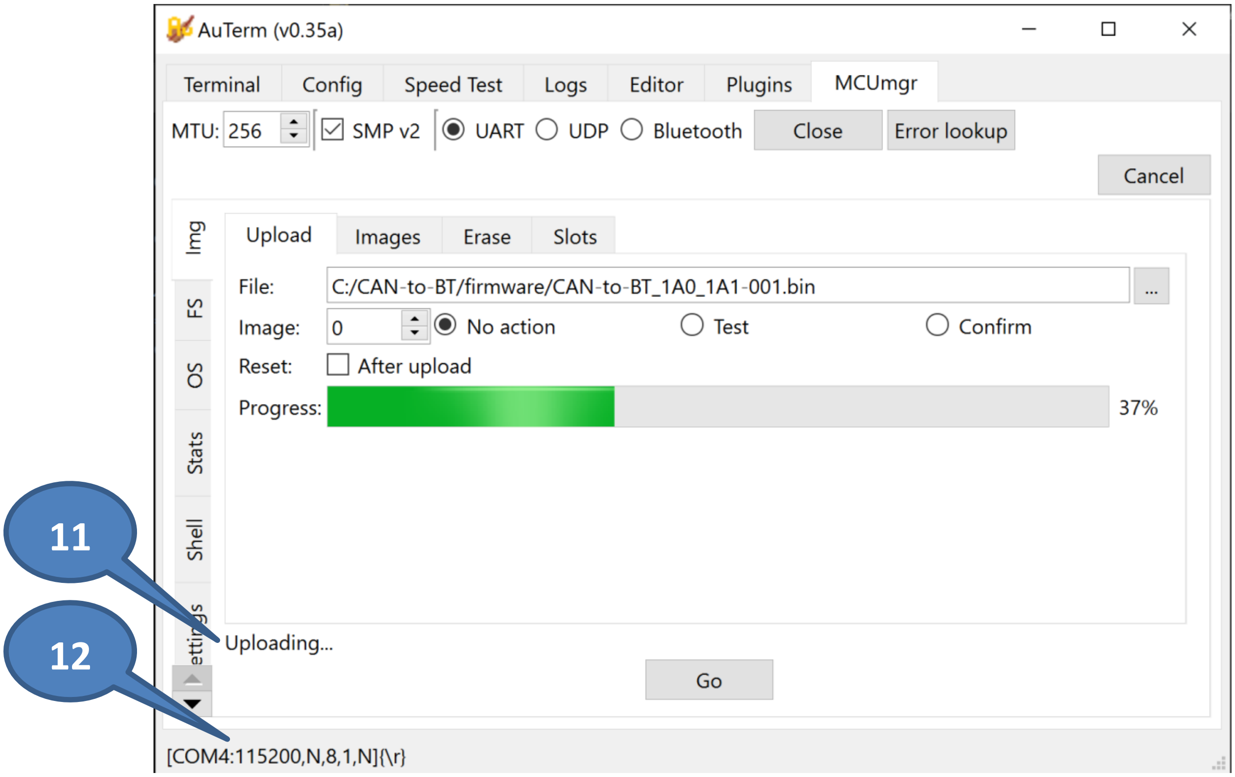
Progress: 0%

[Status]

AuTerm version 0.35a (Windows (x86_64)), Built Dec 31 2024 Using QT 6.8.1, Secure Channel

During the firmware update

11. The **Status** will change to “***Uploading…***”
12. The serial connection attributes will appear in a status bar



MTU: 256 ☑ SMP v2 ● UART ○ UDP ○ Bluetooth

File: C:/CAN-to-BT/firmware/CAN-to-BT_1A0_1A1-001.bin

Image: 0 ● No action ○ Test ○ Confirm

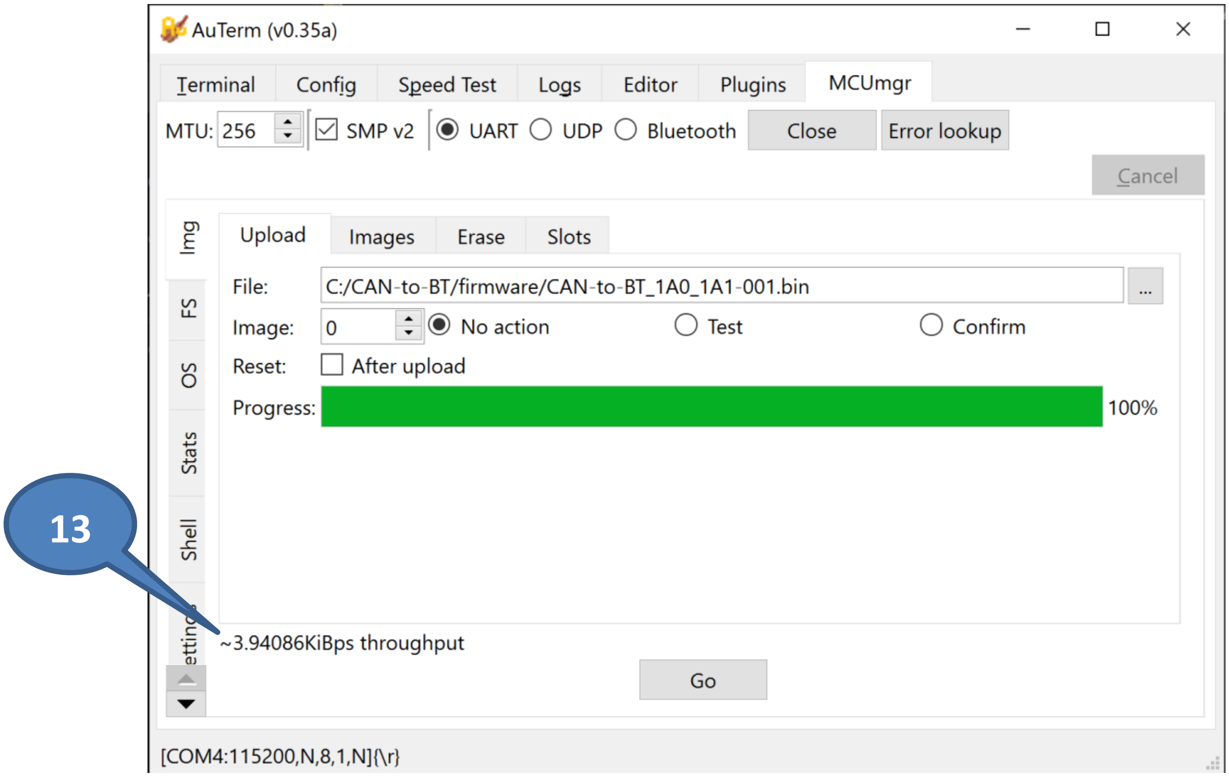
Reset: □ After upload

Progress: 37%

Uploading...

[COM4:115200,N,8,1,N]{\r}

13. Once the flashing process is complete, the status will display the transmission rate



MTU: 256 ☑ SMP v2 ● UART ○ UDP ○ Bluetooth

File: C:/CAN-to-BT/firmware/CAN-to-BT_1A0_1A1-001.bin

Image: 0 ● No action ○ Test ○ Confirm

Reset: □ After upload

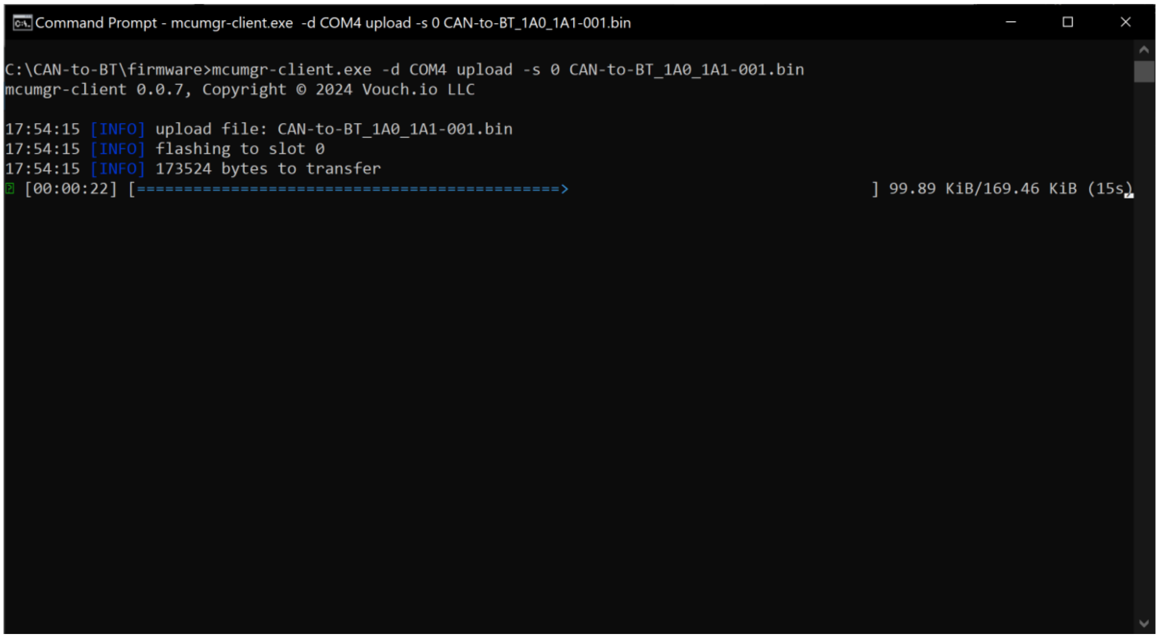
Progress: 🟩 100%

~3.94086KiBps throughput

[COM4:115200,N,8,1,N]{\r}

4.2.2 Using mcumgr-client (Windows)

1. Check the Assigned COM Port
   - Open Device Manager
   - Expand the Ports (COM & LPT) section
   - Note the COM port assigned to the Gateway (e.g., COM4)
2. Open Command Prompt
   - Press **Win + R**, type ***cmd***, and press **Enter** to open the Command Prompt
3. Navigate to the Application Directory
   - Use the ***cd*** command to navigate to the folder where the flashing tool is located
     Example:
     cd C:\Users\YourName\Downloads\FlasherTool
4. Run the Flashing Command
   - Execute the console application with the required arguments
     Example:
     mcumgr-client.exe –d COM4 upload –s 0 CAN-to-BT_1A0_1A1-001.bin


Command Prompt - mcumgr-client.exe -d COM4 upload -s 0 CAN-to-BT_1A0_1A1-001.bin

```
C:\CAN-to-BT\firmware>mcumgr-client.exe -d COM4 upload -s 0 CAN-to-BT_1A0_1A1-001.bin
mcumgr-client 0.0.7, Copyright © 2024 Vouch.io LLC

17:54:15 [INFO] upload file: CAN-to-BT_1A0_1A1-001.bin
17:54:15 [INFO] flashing to slot 0
17:54:15 [INFO] 173524 bytes to transfer
  [00:00:22] [==============================================>                                 ] 99.89 KiB/169.46 KiB (15s)
```

Note:

This document supersedes all previously released versions. It may contain technical inaccuracies or typographical errors and is subject to change without notice.