CAN–Ethernet Bridge
Ethernet Interface

13 August 2021

Brisbane, Australia

http://www.prohelion.com

TABLE OF CONTENTS

1 INTRODUCTION

This document describes the Ethernet interface to the Prohelion CAN-Ethernet bridge device. This information is primarily aimed at advanced users who wish to communicate with the bridge using custom or 3rd party software.

The bridge supports bi-directional CAN-Ethernet bridging using both UDP and TCP protocols. UDP is the default protocol, and should be used for normal broadcast CAN packets. TCP mode should be used when a reliable point-to-point connection is required between the bridge and a single network device, for example when exchanging a stream of data via the CAN-Ethernet network. The bridge can only support a single TCP connection at any time.

2 CAN-UDP BRIDGING

By default, CAN packets that are received by the bridge will be broadcast on the UDP group address/port, and data contained in UDP datagrams received by the bridge from this group address/port will be broadcast on the CAN network.

The UDP group address is **239.255.60.60**, and the port is **4876**. This port is assigned by the IANA and is not configurable.

The UDP packet structure is shown in figure 1. Unless otherwise noted, all values are transmitted in big-endian format.

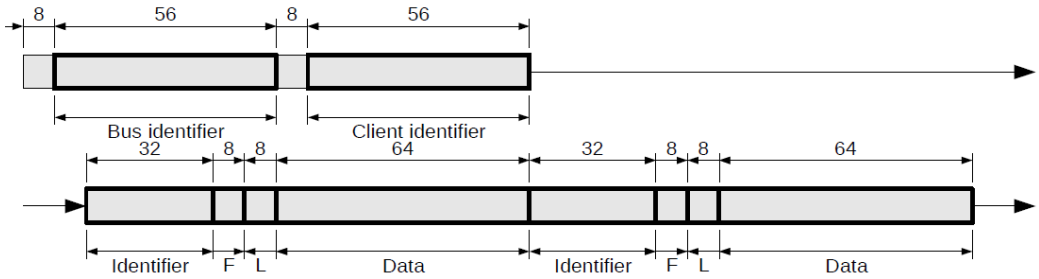

Figure 1: UDP packet structure

The *bus identifier* is a 56-bit value as shown in figure 2. There are presently two protocol versions designated by the *bus identifier*. Version 1 identifiers allow 4 bits for the *bus number* (range 0-15), while Version 2 allows 16 bit bus numbers (range 0-65535).

In protocol version 1, the first 52 bits encode the protocol version, and should read **0x5472697469756** in all packets. The least significant 4 bits represent the bus number that the packet was transmitted on.

In protocol version 2, the first 40 bits encode the protocol version and should read **0x547269FDD6** in all packets, with the least significant 16 bits representing the bus number. The bus number can be used to create separate virtual networks on the same Ethernet network (see section 7).

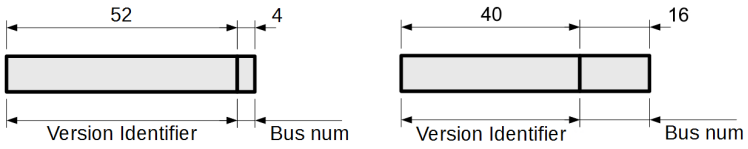

Figure 2: Bus identifier structure – Version 1 (left) and Version 2 (right)

The *client identifier* is a 56-bit value that uniquely identifies the sender of the datagram. Each device on the virtual CAN network should have a different client identifier; the CAN-Ethernet bridges use the MAC address of their Ethernet interface as their client id. This is also the recommended setting for other devices to ensure uniqueness.

The *identifier* represents the identifier of the packet on the physical CAN network. The identifier is contained in the low 11 bits (29 in extended mode).

The *flags* byte contains the bitfield shown in figure 3.

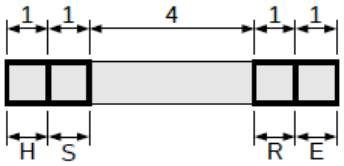

Figure 3: Flags bitfield

- **H – Heartbeat / query response packet** Indicates that this datagram contains a message from the bridge itself, rather than a bridged CAN packet. This will either be a bridge heartbeat packet, or a packet containing a response to a query request.
- **S – Settings packet** Indicates that this datagram contains a setting for the bridge itself, and should not be bridged on to the physical CAN network.
- **R – RTR packet** Indicates that the data contained in this datagram should be sent as an RTR packet on the physical CAN network.
- **E – Extended id packet** Indicates that this packet should be sent with an extended CAN identifier.

The *length* byte indicates the length of the packet data, in bytes. This length should not exceed eight bytes.

The *data* section contains the data contained in the physical CAN packet. If the length byte indicates less than 8 bytes of data, the remaining bytes of the data section should be zero.

Multiple CAN packets are able to be bundled into a single UDP datagram as shown in figure 1, this circumvents the need to repeatedly send the same bus and client identifier along with each CAN packet. It is important to note that unlike CAN-TCP bridging of multiple CAN packets outlined in section 6, each new UDP datagram must start with the bus and client identifier.

3 CAN-TCP BRIDGING

When a higher level of reliability is required in the connection between a device and the CAN-Ethernet bridge, CAN packets can be communicated over a TCP/IP connection. This connection should be made to the bridge on the same remote port as UDP communication occurs: **4876**. Data should then be sent to the bridge according to the structure shown in figure 4.

By setting the fwd identifier and fwd range fields (see below), it is possible to have the bridge use this TCP connection to forward on certain packets, rather than having them multicast via UDP.

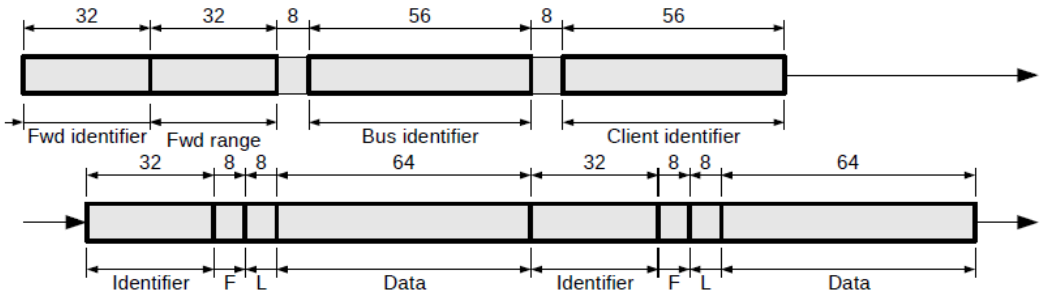

Figure 4: TCP stream structure

The fwd identifier represents the lowest CAN identifier that should be bridged from the physical CAN network and forwarded on via the TCP connection.

The fwd range represents the size of the range of CAN identifiers that should be forwarded via TCP. That is, the bridge will forward via TCP any packet with

fwd identifier <= packet CAN identifier < (fwd identifier + fwd range)

The remaining fields in figure 4 are identical to those in figure 1.

In effect, the TCP data stream consists of an initial TCP datagram with the forwarding information, followed by bus and client identifiers that are only sent once for the entire TCP stream. This initial information is followed by any number of CAN packets consisting of identifiers, flags, lengths and data which can be a part of the initial TCP datagram or any number of independent TCP datagrams afterwards.

Just like with UDP bridging in section 4, multiple CAN packets are able to be bundled into a single TCP datagram, the only difference is that after the initial TCP datagram successive transmissions need only contain CAN packets.

4 BRIDGE HEARTBEAT

In order to identify the presence of a CAN-Ethernet bridge on a network, the bridge sends periodic heartbeat datagrams over UDP. These packets are identified by an active **H** bit in the *flags* bitfield, and an identifier of **0x000**. The contents of the data field of a heartbeat

datagram is shown in figure

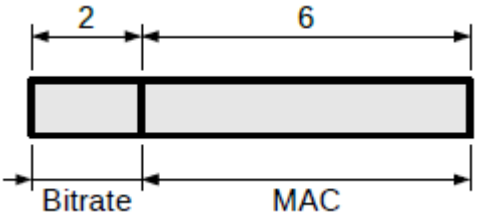

Figure 5: Heartbeat data

The *bitrate* field is an unsigned 16-bit integer showing the current CAN bitrate of the bridge in kbps.

The MAC field is a 48-byte value representing the MAC address of the bridge's Ethernet interface.

Note that the heartbeat packets are always multicast via UDP, regardless of the identifier forwarding settings used in any existing TCP connection.

5 BRIDGE CONFIGURATION

The bus number and CAN bitrate used by a bridge are user settable. To set a parameter on a specific bridge, open a TCP connection and transmit a single packet with

- the **S** bit set in the *flags* bitfield,
- the *length* field set to 3 when setting bitrate, 8 when setting bus number,
- the upper two bits of the data field set to **10b**,
- the remaining bits of the upper data byte set to **0x05** when setting the bitrate, or **0x0E** when setting the bus number, and
- the following data bytes set to the desired parameter value, stored as an unsigned integer. When setting the bus number, the full 7-byte bus identifier must be sent, consisting of either

  (a) Protocol v1: the 52 bit protocol version with 4 bit bus number, or

  (b) Protocol v2: the 40 bit protocol version with 16 bit bus number

, as shown in Figure 2. When setting the bitrate, the new rate in kbps should be sent as a two byte integer.

6 VERSION QUERYING

The hardware and firmware version of a given bridge can be queried via the Ethernet interface. To query a given bridge, open a TCP connection and transmit a single packet with

- the **S** bit set in the *flags* bitfield,
- the *length* field set to 8,

- the upper two bits of the data field set to **00b**,
- the remaining bits of the upper data byte set to **0x16**, and
- the remaining data bytes set to zero.

Upon receipt of this packet, the bridge will respond with a UDP packet with

- the **H** bit set in the *flags* bitfield,
- the *length* field set to 8,
- the upper two bits of the data field set to **01b**,
- the remaining bits of the upper data byte set to **0x16**, and
- the remaining data bytes containing version information as shown in figure 6, with the *HW* field containing the hardware version, the *FW* field the firmware version multiplied by ten, and the *build num* field the firmware build number. All fields are little-ending unsigned integers.

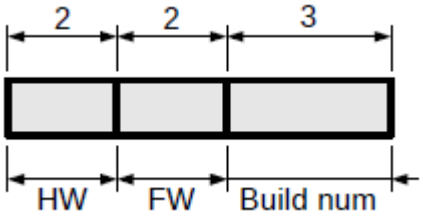

Figure 6: Version information

7 REVISION RECORD

REV	DATE	CHANGE
1	13 August 2021	Document creation (AJP)