



eScada

v25.7.2
Drivers

eScada.Drivers.SocketCAN

eScada.Drivers.SocketCAN

(SocketCAN is an open-source Linux CAN networking stack and driver framework)

OS availability

Linux, RaspBian

Atomic data type

Byte oriented protocol.

Documentation reference

<https://en.wikipedia.org/wiki/SocketCAN>

SocketCAN configuration is hardware- and driver-dependent.

The user is responsible for installing and configuring the appropriate driver for the selected CAN interface. Please consult the driver documentation supplied by the hardware manufacturer for all device-specific settings and limitations.

Before using the application, we recommend validating the CAN interface using can-utils (e.g., ip link, candump, cansend) to ensure the driver and communication parameters are correctly configured.

Parameters available in every section

Channel:	none	
Device:	Channel ¹	CAN Bus channel name (e.g. can0, can1)
	Bus type ²	Classic CAN, CAN-FD
	Reconnect timeout [ms]	Waiting time before a reconnection after a break-down
	Receiving Frames Interval [ms]	Timeout interval used to wait for bus frames.
Group:	none	
Tag:	BRS Flag ³	Bit Rate Switch

¹ You can specify more than one channel name by using multiple rows or separating names with comma (',') or semicolon (';')

² Classic CAN: “CAN 2.0” frames with up to 8 data bytes, fixed bitrate for the whole frame (supports 11-bit or 29-bit IDs, and remote frames via RTR).

CAN-FD: “Flexible Data-rate” CAN with up to 64 data bytes; can optionally switch to a higher bitrate in the data phase (BRS), supports 11-bit or 29-bit IDs, and does not support remote (RTR) frames.

³ BRS = Bit Rate Switch: in a CAN FD frame it indicates whether the transmitter switches to the higher data-phase bitrate after arbitration (BRS=1) or stays at the nominal bitrate for the whole frame (BRS=0)

CAN Bus frames supported for both CAN types

- Receive/Transmit standard (11 bit) frame.

Identifier in hexadecimal format (000-7FF)

- Receive/Transmit extended (29 bit) frame.

Identifier in hexadecimal format (0000000-1FFFFFFF)

CAN Bus frames supported for Classic CAN only

- Receive/Transmit standard RTR (11 bit) frame.

Identifier in hexadecimal format (000-7FF)

- Receive/Transmit extended RTR (29 bit) frame.

Identifier in hexadecimal format (0000000-1FFFFFFF)

The RTR frame, is a frame requesting the transmission of a specific CAN identifier.

Addressing

The tag address can be specified as follow:

11 bit

FS.hhh

Receive/Transmit standard (11 bit) frame

RS.hhh

Receive/Transmit standard RTR (11 bit) frame (valid only for Classic CAN frames)

29 bit

FE.hhhhhhhh

Receive/Transmit extended (29 bit) frame

RE.hhhhhhhh

Receive/Transmit extended RTR (29 bit) frame (valid only for Classic CAN frames)

CAN ID identifier

hhh

CAN ID identifier in hexadecimal format (000-7FF)

hhhhhhhh

CAN ID identifier in hexadecimal format (0000000-1FFFFFFF)

During tag declaration, only unsigned byte can be specified as data type; it is the default data type.

- Receiving frames

Receiving frames must be declared with the Read only attribute set to TRUE.

The payload of these frames is populated automatically as soon as a frame with the configured CAN ID is received.

- Sending frames

Sending frames must be declared with the Read only attribute set to FALSE.

Cycling mode

This mode is enabled by setting the Polling attribute to a value greater than 0.

Frames are transmitted periodically according to the polling interval.

Frames may also be transmitted when a value changes (if configured).

For RTR frames, the tag items are populated as soon as a frame with the corresponding CAN ID is received.

On change mode

This mode is enabled by setting the Polling attribute to 0.

Frames are transmitted only when a single item value changes.

RTR mode (valid only for Classic CAN frames)

When an RTR request is received for a given Message ID, the device transmits the corresponding frame (tag) in response.

Data packing and transmission strategy

The CAN bus uses data frames with a payload of up to 8 bytes (Classical CAN). To manage event-driven communication effectively, the user must define how the payload bytes are populated before transmission.

If the transmission modes described above do not suit your application, we recommend using macros or Lua code to customise data transmission.

The application can update the frame payload using helper functions (for example, the “SetArray” macro) or by executing custom logic in Lua (see the “BeginUpdate” example).

This allows the user to:

- decide which bytes are updated (and how values are encoded),
- control when the payload is refreshed, and
- determine when the frame is sent (periodic transmission, on-change transmission, or event-triggered transmission).

In other words, the system provides the mechanisms to write frame data and trigger transmission, while the user defines the data-packing strategy and the transmission timing.

Consecutive items

Classic CAN supports 0–8 data bytes

CAN FD supports 0–64 data bytes per frame.

Device tag information

Read time [ms] This is the time between two received frames.

Write time [ms] This is the time between two sent frames.

How to get values from frames

The values contained in CAN bus frames can be read or written using the following derived variables and functions: HexToValue, ConvertTo, SplitBits, Function, ValueToHex, SetItem, and Copy.

LUA code can also be used to manipulate frame bytes directly, providing a powerful way to extract data from a frame or to build/update a payload before transmission.

General note

A CAN interface/controller can support both Classic CAN and CAN-FD, but it operates in one mode at a time (selected at device's initialisation); switching mode requires reconfiguration/reset, so it can't run both concurrently on the same channel.

On a physical CAN network, configuration must be consistent: Classic-only nodes cannot communicate with CAN-FD frames, if CAN-FD traffic (especially with BRS) appears on the bus, Classic nodes will treat it as errors.

The same wiring can be shared only if all nodes are CAN-FD capable and you restrict traffic to Classic CAN format (no FD frames).