

CAN XL hardware driver infrastructure in Linux

CAN XL (extended data-field length) protocol controllers and CAN SIC (signal improvement capability) XL transceivers offer new features and also introduce new restrictions that have to be handled by the API (application programming interface).

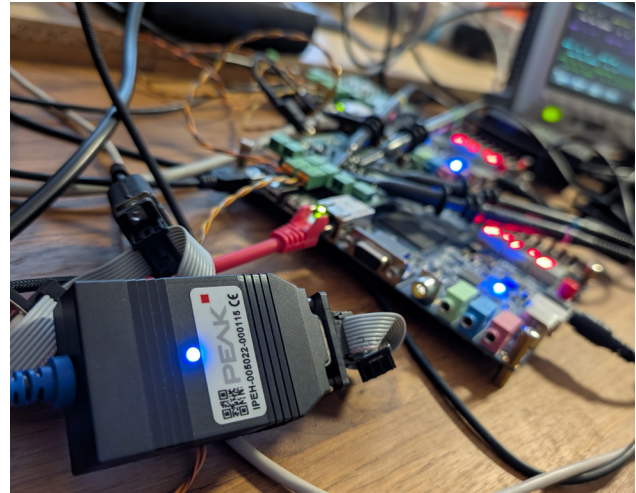
New elements like the acceptance field (AF), the service data unit type (SDT) and virtual CAN network identifiers (VCID) have to be filled and also evaluated by the application software. On the other hand, new configuration options for the XL data-phase bit rate (up to 20 Mbit/s) and the CAN transceiver mode switch (TMS) have to be provided by the CAN driver software. Both extensions for the application programmer and the driver configuration should be designed in a way that users can re-use their knowledge from CAN CC (classic) and CAN FD (flexible data rate) setups.

The basic CAN XL support for Linux already has been introduced in Linux 6.1 in December 2022, which provided the CAN XL API with all relevant data structures and virtual CAN drivers for CAN CC/FD/XL traffic. This enabled the development of the CAN XL support for tools to read/write/generate/dump CAN XL traffic (can-utils/WireShark) as well as some proof-of-concept implementations of CiA specifications like CiA 611-1 (service data unit types), CiA 611-2 (multi-PDU protocol), and CiA 613-3 (LLC frame fragmentation).

With the availability of ‚real‘ CAN XL protocol controllers the question of how to design and realize the CAN XL specific driver configuration options raised in importance.

On the Linux side, a very first PoC (proof-of-concept) setup by NXP started with a Terasic DE1-SOC FPGA board with Ubuntu 16.04 and three Bosch XCANB IP-cores already in 2022. Based on this work two boards were upgraded to Ubuntu 22.04.4 LTS and equipped to run with the latest Linux Mainline kernel version available. Although such a Linux 6.9 environment has already been successfully performed in a CiA 613-3 integration test on a CAN CiA plug-fest in May 2024, the CAN XL controller configurations (e.g., bit rates) still had to be set at compile time.

The work on the Linux integration of the CAN XL driver configuration became a boost with the release of the PCAN-USB XL CAN adapter in April 2025. Having a



(Source: Dr. Oliver Hartkopp)

second Windows and Linux capable CAN XL hardware for interoperability testing provided a user's view on the usability and configuration for CAN XL on a different operating system.

Especially the BitrateManager by Peak, which is used to configure the CAN CC/FD/XL bit rate settings with calculations for the transmitter delay compensations, the error signaling and the PWM calculations in the case of the CAN XL transceiver mode switching, was very helpful to embrace the real word effects of the released ISO 11898-1:2024 standard.

Since Linux 6.19 (released 2026-02-08) the driver infrastructure to handle CAN XL has been officially integrated. This feature upgrade also applies to the ip tool from the iproute2 package, which is used to configure the CAN node (CAN controller) settings. With `ip link help can`, the supported configuration options can be displayed (see Figure 1). The orange marked settings represent the CAN CC options, green the CAN FD and blue the CAN XL options. To enable the CAN FD support for capable nodes, the `fd on` option has

Table 1: Mode features

Mode	CAN CC	CAN FD	CAN XL	Error signalling	TMS	Remark
fd off/xl off	yes	no	no	(enabled)	(off)	CC-only mode
fd on/xl off	yes	yes	no	(enabled)	(off)	CC/FD mixed mode
fd on/xl on	yes	yes	yes	(enabled)	(off)	mixed mode
fd off/xl on	no	no	yes	(disabled)	optional	XL-only mode

NOTE The values in brackets are settings that are automatically derived from the FD/XL modes. Only in the CAN XL-only mode, the transceiver mode setting (TMS) can be enabled as an option (`tms on`), if a CAN SIC XL transceiver with Fast Mode support is attached to the CAN XL controller.

Example 1: CC/FD/XL mixed-mode configuration

```
# ip link set can0 type can bitrate 1000000 dbitrate 2000000 fd on xbitrate 4000000 xl on
# ip -details link show can0
7: can0: <NOARP> mtu 2060 qdisc noop state DOWN mode DEFAULT group default qlen 10
  link/can promiscuity 0 allmulti 0 minmtu 76 maxmtu 2060
  can <FD,TDC-AUTO,XL,XL-TDC-AUTO> state STOPPED restart-ms 0
  bitrate 1000000 sample-point 0.750
  tq 6 prop-seg 59 phase-seg1 60 phase-seg2 40 sjw 20 brp 1
  dummy_can CC: tseg1 2..256 tseg2 2..128 sjw 1..128 brp 1..512 brp_inc 1
  dbitrate 2000000 dsample-point 0.750
  dtq 6 dprop-seg 29 dphase-seg1 30 dphase-seg2 20 dsjw 10 dbrp 1
  tdc0 60 tdcf 0
  dummy_can FD: dtseg1 2..256 dtseg2 2..128 dsjw 1..128 dbrp 1..512 dbrp_inc 1
  tdc0 0..127 tdcf 0..127
  xbitrate 4000000 xsample-point 0.750
  xtq 6 xprop-seg 14 xphase-seg1 15 xphase-seg2 10 xsjw 5 xbrp 1
  xtdc0 30 xtDCF 0
  dummy_can XL: xtseg1 2..256 xtseg2 2..128 xsjw 1..128 xbrp 1..512 xbrp_inc 1
  xtdc0 0..127 xtDCF 0..127
  pwms 1..8 pwml 2..24 pwmo 0..16
clock 160000000 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
tso_max_size 65536 tso_max_segs 65535 gro_max_size 65536 gso_ipv4_max_size 65536
gro_ipv4_max_size 65536
```

Example 2: XL-only mode configuration with TMS enabled

```
# ip link set can0 type can bitrate 500000 fd off xbitrate 16000000 xl on tms on
# ip -details link show can0
7: can0: <NOARP> mtu 2060 qdisc noop state DOWN mode DEFAULT group default qlen 10
  link/can promiscuity 0 allmulti 0 minmtu 76 maxmtu 2060
  can <XL,XL-TMS> state STOPPED restart-ms 0
  bitrate 500000 sample-point 0.875
  tq 12 prop-seg 69 phase-seg1 70 phase-seg2 20 sjw 10 brp 2
  dummy_can CC: tseg1 2..256 tseg2 2..128 sjw 1..128 brp 1..512 brp_inc 1
  dummy_can FD: dtseg1 2..256 dtseg2 2..128 dsjw 1..128 dbrp 1..512 dbrp_inc 1
  tdc0 0..127 tdcf 0..127
  xbitrate 16000000 xsample-point 0.600
  xtq 6 xprop-seg 2 xphase-seg1 3 xphase-seg2 4 xsjw 2 xbrp 1
  pwms 3 pwml 7 pwmo 0
  dummy_can XL: xtseg1 2..256 xtseg2 2..128 xsjw 1..128 xbrp 1..512 xbrp_inc 1
  xtdc0 0..127 xtDCF 0..127
  pwms 1..8 pwml 2..24 pwmo 0..16
clock 160000000 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
tso_max_size 65536 tso_max_segs 65535 gro_max_size 65536 gso_ipv4_max_size 65536
gro_ipv4_max_size 65536
```

```
Usage: ip link set DEVICE type can
[ bitrate BITRATE [ sample-point SAMPLE-POINT ] ]
[ tq TQ prop-seg PROP_SEG phase-seg1 PHASE-SEG1
  phase-seg2 PHASE-SEG2 [ sjw SJW ] ]
[ dbitrate BITRATE [ dsample-point SAMPLE-POINT ] ]
[ dtq TQ dprop-seg PROP_SEG dphase-seg1 PHASE-SEG1
  dphase-seg2 PHASE-SEG2 [ dsjw SJW ] ]
[ tdcv TDCV tdc0 TDC0 tdcf TDCF ]
[ xbitrate BITRATE [ xsample-point SAMPLE-POINT ] ]
[ xtq TQ xprop-seg PROP_SEG xphase-seg1 PHASE-SEG1
  xphase-seg2 PHASE-SEG2 [ xsjw SJW ] ]
[ xtDCV TDCV xtDC0 TDC0 xtDCF TDCF pwms PWMS pwml PWML pwmo PWMO ]

[ loopback { on | off } ]
[ listen-only { on | off } ]
[ triple-sampling { on | off } ]
[ one-shot { on | off } ]
[ berr-reporting { on | off } ]
[ fd { on | off } ]
[ fd-non-iso { on | off } ]
[ presume-ack { on | off } ]
[ cc-len8-dlc { on | off } ]
[ tdc-mode { auto | manual | off } ]
[ restricted { on | off } ]
[ xl { on | off } ]
[ xtDCV TDCV xtDC0 TDC0 xtDCF TDCF pwms PWMS pwml PWML pwmo PWMO ]
[ tms { on | off } ]

[ restart-ms TIME-MS ]
[ restart ]

[ termination { 0..65535 } ]

Where:
BITRATE      := { NUMBER in bps }
SAMPLE-POINT := { 0.000..0.999 }
TQ           := { NUMBER in ns }
PROP-SEG    := { NUMBER in tq }
PHASE-SEG1  := { NUMBER in tq }
PHASE-SEG2  := { NUMBER in tq }
SJW         := { NUMBER in tq }
TDCV        := { NUMBER in mtq }
TDC0        := { NUMBER in mtq }
TDCF        := { NUMBER in mtq }
PWMS        := { NUMBER in mtq }
PWML        := { NUMBER in mtq }
PWMO        := { NUMBER in mtq }
RESTART-MS  := { 0 | NUMBER in ms }

Units:
bps         := bit per second
ms          := millisecond
mtq         := minimum time quanta
ns          := nanosecond
tq          := time quanta
```

Figure 1: ip link help can (Source: Dr. Oliver Hartkopp)

to be set. The same applies to `xl on` for CAN XL capable nodes. As CAN XL controllers offer different modes, the possible settings for `fd` and `xl` result in specific features (see Table 1).

To test the CAN configuration API, a new CAN software device `dummy_can` has been integrated in Linux 6.19, which can be loaded with `modprobe dummy_can` and creates a CAN CC/FD/XL capable device (usually `can0`).

The examples in the grey text box give a glance on how to configure CAN XL capable CAN controllers with calculated default values for TDC and PWM settings.

Together with the CAN XL driver infrastructure the CAN RAW sockets have received a new feature that instantly rejects CAN frame types, that are not supported by the selected CAN interface. E.g., writing a CAN CC frame to a CAN interface in XL-only mode will directly result in a write error (`-EINVAL`). The same applies to CAN interfaces that cannot be written to (listen-only / restricted modes), which will then return an `-EACCESS` error code.

Author

Dr. Oliver Hartkopp (Linux CAN maintainer)

