



BM-PCIE-X8L4

Technical Specification

8-channel CAN FD + 4-channel LIN PCIe interface card

Multi-network acquisition, parallel endurance tests, node simulation, and HIL benches.

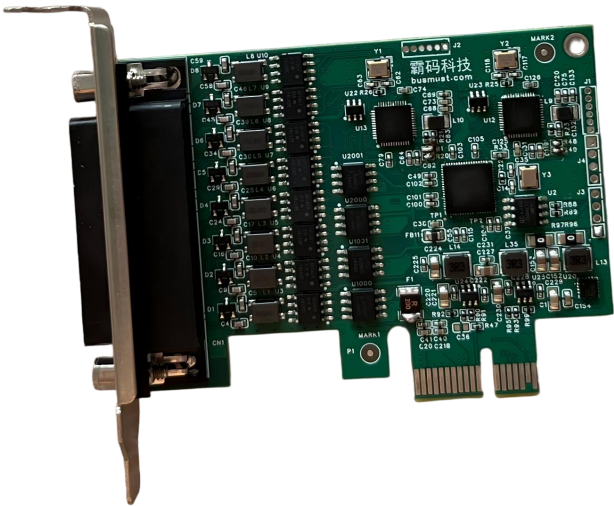


Figure 1. BM-PCIE-X8L4 card with a full-height or low-profile bracket.

Model	BM-PCIE-X8L4
Revision / date	Rev E (English) / 2026-10-13
Key specifications	8 CAN FD + 4 LIN; CAN FD data phase up to 12 Mbps
Installation	Full-height / low-profile bracket; single-slot width
Systems / interfaces	Windows / Linux; BMAPI SDK, BMCLI, BUSMASTER

Contents

2 Overview and hardware	3 CAN / CAN FD	4 LIN and saved tasks
5 Installation and expansion	6 DB25 pinout	7 Software and platforms
8 BMAPI SDK	9 BMCLI	10 Open-source tools
11 Applications and resources		

01 Overview and Hardware Specifications

BM-PCIE-X8L4 provides 8 independent CAN / CAN FD channels and 4 LIN channels through a host PCIe slot. All bus connections use one DB25 female connector. Host software configures and manages each channel. The card is powered directly from the PCIe edge connector.

Parameter	Specification
CAN / CAN FD	8 independently configurable channels; concurrent operation
LIN	4 channels; master, slave, and bus monitoring
Host interface	Standard PCIe 2.1 x1 edge connector
External connector	DB25 female; pinout on page 6
Card body dimensions	78.2 × 59.6 × 20 mm; dimension scope on page 5
Bracket	Standard full-height or low-profile PCIe bracket
Slot occupancy	Single-slot width
Power supply	PCIe slot; no external card power supply required
Power consumption	< 1.5 W
Weight	< 100 g
Operating temperature	-40 to +85 °C
Storage temperature	-40 to +105 °C
Electrical isolation	None; PCIe and bus interfaces share ground
Receive timestamp	Hardware timestamp with 1 µs recording resolution
Timed transmission	Up to 64 hardware transmit tasks per channel; 1 ms minimum period and timing resolution
Protection	Bus ESD protection; PCIe power overcurrent protection
Saved operation	Primarily CAN; settings and small replay sequences persist and start automatically; approximately 250 KB
Online logging	Log files are saved by the connected host application
Hardware frame routing	Not supported
On-card standalone file logging	Not supported
Serial port	No serial port on this card

This specification covers the BM-PCIE-X8L4 hardware and software interfaces. Bracket choice changes the chassis mounting arrangement; channel counts and electrical specifications remain the same.

02 CAN / CAN FD Features and Performance

Eight independently configurable channels support concurrent transmission and reception, hardware filtering, timed transmission, and accelerated ISO-TP flow control.

Functional specifications

Item	Specification
Protocols	CAN 2.0; CAN FD (ISO / Bosch)
Frame formats	Standard and extended identifiers; CAN FD bit rate switching (BRS)
Bit rate range	5 kbps to 12 Mbps; 12 Mbps is the maximum CAN FD data-phase bit rate
Payload	Classical CAN: 0 to 8 bytes; CAN FD: up to 64 bytes
Operating modes	Normal, Classical CAN, listen-only, internal loopback, and bus loopback
Termination	120 Ω per channel; software-controlled enable / disable
Receive filtering	Hardware receive filters configured through the SDK
ISO-TP	Hardware-accelerated flow control for host diagnostic and transport applications

Performance specifications

Item	Specification
Receive throughput*	> 26,000 fps per channel
Transmit throughput*	> 26,000 fps per channel
Receive timestamp	Hardware timestamp; 1 μ s recording resolution
Hardware timed transmission	Up to 64 tasks per channel; 1 ms minimum period and timing resolution

* Throughput conditions: 8-byte CAN FD frames, 1 Mbps arbitration phase and 12 Mbps data phase. Receive and transmit rates are measured separately. Actual throughput varies with frame format, bus parameters, and host load.

03 LIN and Saved Hardware Tasks

Item	Specification
Protocol / channels	4 channels; LIN 1.3 to LIN 2.x
Bit rate / payload	1,200 to 19,200 bps; 0 to 8 bytes
Checksum	Classic / Enhanced; generation, validation, and custom checksum
Master / slave	Dynamically switchable; bus monitoring in either role
Slave bit rate	Dynamic automatic measurement
Errors / filters	Response timeout detection; LIN ID receive filtering
Pull-up	1 kΩ per channel; software-controlled enable / disable
LIN supply	Onboard 12 V for transceivers; no auxiliary 12 V pin on DB25
Original LIN throughput specification	Receive: full bus load; transmit: > 50 fps, depending on frame length and bit rate

Master and slave operations

Operation	Master	Slave
Monitor	Receive bus frames	Receive bus frames
Master write	Send header and payload once or periodically	Cannot initiate a master write
Master read	Send header and wait for slave response; report timeout	Cannot initiate a master read
Slave response	Receive the slave response	Preconfigure payload and respond to a matching header

The master enables the 1 kΩ pull-up according to the network configuration. Slaves select the pull-up setting to suit the existing network. ID filters do not suppress master requests initiated by the card itself.

Saved operation and log storage

Saved operation primarily covers CAN. Bit rates, timed transmit tasks, and a small replay sequence survive power loss and start automatically after power-up without a host application. Hardware replay capacity is approximately 250 KB; PCIe slot power is required.

The card does not provide full standalone file recording and replay. Use BMCLI or other host software for longer online recordings and replay; log files are stored on the host.

04 Dimensions, Installation, and Expansion

The compact card occupies one slot. Select a full-height or low-profile bracket to match the chassis expansion opening. Both bracket types use the same card and DB25 interface.

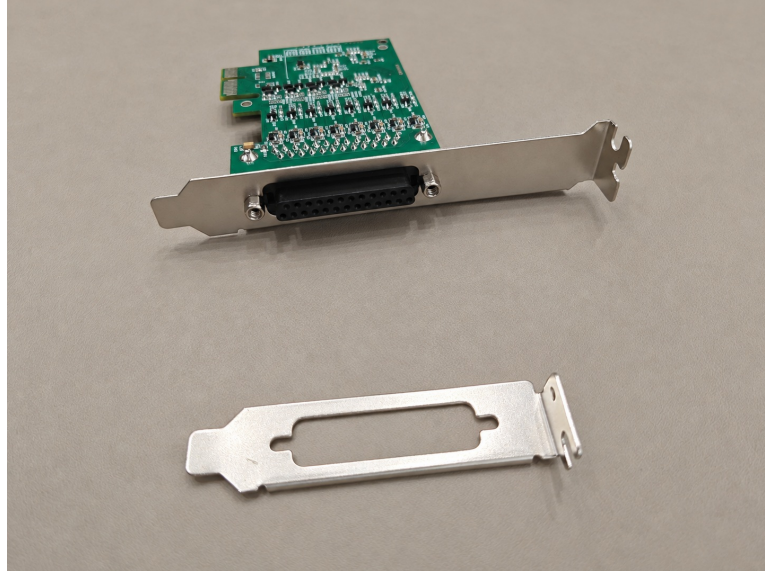


Figure 2. Card fitted with a full-height bracket, alongside the low-profile bracket.

Dimension	Value / scope
Body length × height × occupied width	78.2 × 59.6 × 20 mm
Dimension scope	59.6 mm includes the edge connector; 20 mm is the thickness-direction width including bracket width. DB25 projection is excluded.
Bracket height	Full-height / low-profile; vertical bracket height is different from card body height

PCIe connection and installation

The PCIe 2.1 x1 interface fits compatible standard x1 / x4 / x8 / x16 slots and negotiates with newer-generation slots. Power off and disconnect the host supply before installation or removal. Insert the card, secure the bracket, and install the system software and driver after startup.

A standard full-height or low-profile bracket is provided according to configuration. Specialized industrial-PC or open-carrier mounting supports and other mechanical parts are supplied by the user.

Multi-card expansion

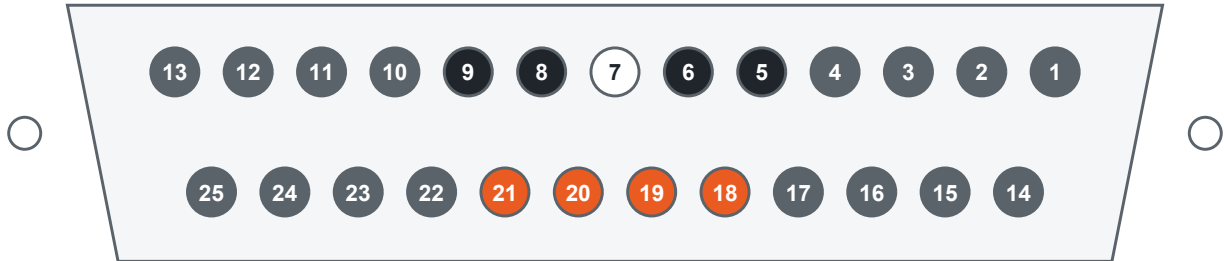
Install multiple cards in available PCIe slots or use an appropriate PCIe Switch expansion platform. Each X8L4 adds 8 CAN FD and 4 LIN interfaces. Card count depends on available slots, power, mounting, and host processing capacity.

Software capacity: BUSMASTER supports up to 32 channels; BMCLI supports up to 64 channels; BMAPI SDK has no fixed channel limit.

05 DB25 Pinout and Wiring

DB25 female connector, shown from the mating face; use a DB25 male accessory. The male wiring side may be viewed from the opposite direction. Identify connections by pin number.

DB25 female on the card - front view of the mating face



Left to right: upper row 13 to 1; lower row 25 to 14

Pin	Signal	Pin	Signal
1	CAN1_L	14	CAN1_H
2	CAN2_L	15	CAN2_H
3	CAN3_L	16	CAN3_H
4	CAN4_L	17	CAN4_H
5	GND	18	LIN1
6	GND	19	LIN2
7	PPS (reserved; not enabled)	20	LIN3
8	GND	21	LIN4
9	GND	22	CAN5_H
10	CAN5_L	23	CAN6_H
11	CAN6_L	24	CAN7_H
12	CAN7_L	25	CAN8_H
13	CAN8_L	-	-

Wiring

Connect each CAN channel to its matching CAN_H, CAN_L, and common reference ground. The eight channels are separate buses and must not be interconnected. Fit 120 Ω termination at each end of a CAN network; enable internal termination when the card is an endpoint.

LIN1 / LIN2 / LIN3 / LIN4 use pins 18 / 19 / 20 / 21. Pins 5 / 6 / 8 / 9 are common GND, paired with LIN1 / LIN2 / LIN3 / LIN4 respectively in the original definition.

Pin 7 retains the reserved PPS assignment and is not enabled. LIN transceivers use onboard 12 V; DB25 does not provide auxiliary power output.

06 Software Ecosystem and Platforms

Use BMAPI SDK to integrate the card into your application, BMCLI for command-line automation, or BUSMASTER for graphical debugging. Linux users can also use SocketCAN and python-can standard interfaces.

Environment	Support
Windows	x86 / x64; Windows 10 / 11 are common environments
Linux / Ubuntu	Select x86 / x64 / AArch64 libraries and dependencies from the SDK
Desktop PC / workstation	Standard PCIe slot and a matching full-height or low-profile opening
Industrial / vehicle PC	Standard PCIe expansion; multi-card acquisition and bench integration
ARM carrier / edge computing	Carrier with a standard PCIe slot and matching Linux / AArch64 software

ARM connection examples include the NVIDIA Jetson AGX Orin developer kit with a standard PCIe slot and the RK3588 LubanCat 5-BTB core module with a PCIe carrier. These illustrate connection options without limiting host brands. Mini PCIe, M.2, or FFC interfaces require suitable adapters, power, and mounting.

Software interfaces

Software / interface	Function / environment
BMAPI SDK	Windows / Linux application development; devices, channels, configuration, messaging, timestamps, filters, timed transmission, and ISO-TP
BMCLI	Windows / Linux command-line acquisition, log replay, node simulation, signal processing, and automation
BUSMASTER	Windows GUI for CAN / CAN FD / LIN, DBC / LDF, plots, and logs. Basic cyclic tasks use hardware if test patterns and transmit-count control are disabled and capacity is available; otherwise host scheduling is used
SocketCAN / can-utils	Standard Linux CAN / CAN FD network interface through bmsocketcan
python-can / python-lin	python-can bmcan for CAN / CAN FD; BMAPI or python-lin for LIN

Development environments

Examples and interface documentation cover C / C++, Python, Qt, C#, VB.NET, and LabVIEW. Qt is an application framework; LabVIEW uses VI integration. Each environment uses a BMAPI dynamic library matching its process architecture.

Install the matching runtime, driver, and access permissions according to the software documentation. SocketCAN and python-can cover CAN / CAN FD. LIN uses the appropriate BMAPI / python-lin interface.

07 BMAPI SDK Application Development

The SDK supplies headers, Windows / Linux runtime libraries, drivers, API documentation, and source examples for multi-channel acquisition, bench control, and production-test applications.

Function	API / capability
Initialize / enumerate	BM_Init, BM_Enumerate; map physical wiring using device serial numbers and channel information
Open / configure	BM_OpenEx; operating mode, bit rates, sample points, termination, and filters
CAN / LIN messaging	BM_Read / BM_Write; dedicated CAN and LIN helpers
Batch / asynchronous receive	BM_ReadMultiple; notification events and BM_WaitForNotifications
Hardware timed transmission	BM_SetTxTasks; period, payload, and transmit count
ISO-TP transport	BM_ReadIsotp / BM_WriteIsotp; UDS and other higher-layer applications
Save / replay	BM_SaveConfig, BM_SetReplay; device configuration storage and hardware replay
Resource cleanup	Stop tasks and close channels with BM_Close; call BM_UnInit after all channels are closed

Source examples and documentation

Example / document	Purpose
bmap_i_transmit_only / bmap_i_receive_only	Basic CAN FD transmit / receive
bmap_i_dualthread_txrx / multichannel_rx_cpp	Dual-thread messaging / parallel multi-channel receive
multichannel_isotp_txrx_cpp / bmap_i_lin_txrx	Multi-channel ISO-TP / LIN master-slave
can_analyzer_labview	LabVIEW VI integration
SDK API documentation / development guide	Parameters, return values, call sequences, and common questions

08 BMCLI Command-Line Tool

BMCLI is the BUSMUST command-line tool for Windows / Linux. Scripts and test executors can call it to automate multi-channel acquisition, node simulation, and hardware-in-the-loop benches.

Capabilities

Capability	Use
Multi-channel acquisition	Select multiple channels, record logs, organize file segments, and monitor acquisition status
Signals / databases	Decode or encode signals with DBC / LDF for transmission and observation
Node simulation	DBC-based cyclic frames, changing signals, and application-layer fault rules such as dropped frames
Recording / replay / HIL	Replay host logs with timing and channel mapping; acquire current DUT responses
Timed transmission / diagnostics	Hardware transmit tasks and ISO-TP / UDS operations
LIN communication	Use X8L4 LIN channels for master-slave communication and LDF operations
Modbus integration	Access Modbus devices through host serial ports or networking
Scripts / AI tools	Commands can be invoked by scripts, Claude Code, Codex, and other tools

Execution

Item	Implementation
Node simulation	Host-scheduled cyclic frames and application-layer fault rules
Hardware periodic transmission	txtask configures hardware-timed tasks on the card
Replay	Host log replay; small hardware sequence replay is configured separately
Platforms	Windows / Linux with matching BMAPI runtime libraries

BMCLI node simulation operates at frame and signal level; it does not provide physical-layer fault injection on the card.

Modbus is a BMCLI software capability using host serial or network interfaces. This card has no serial port.

Documentation: busmust.com/en/docs/tutorials/bmcli/

09 Open-Source Toolchain

BUSMUST provides a SocketCAN driver and Python interface projects for standard Linux CAN tools and Python applications. Source code, interface documentation, and examples are available from the company public repositories.

Standard interfaces and tools

Project / tool	Function / environment
bmsocketcan / SocketCAN	Standard Linux CAN / CAN FD network interfaces; independent channel configuration and software-controlled termination
can-utils	cansend for transmission; candump for receive and recording; Classical CAN and CAN FD
Linux network tools	ip link configures CAN parameters and shows status; ifconfig shows interface status
python-can / bmcan	CAN / CAN FD via BMAPI; Bus, Message, Notifier, and periodic transmission
python-lin	LIN via BMAPI; Python master-slave communication and messaging

SocketCAN and python-can cover CAN / CAN FD. Use python-lin or BMAPI for LIN. Build the Linux driver for the host kernel; Python interfaces require a matching BMAPI runtime.

python-can supports hardware-timed and software-periodic transmission. The implementation depends on the adapter version and task parameters. Examples include `send_one.py`, `send_one_canfd.py`, `receive_all.py`, and `periodic.py`.

BUSMUST company repositories

GitHub: github.com/busmust

Gitee: gitee.com/busmust

Project repositories

SocketCAN driver

GitHub: github.com/busmust/bmsocketcan

Gitee: gitee.com/busmust/bmsocketcan

python-can - BUSMUST maintained fork

GitHub: github.com/busmust/python-can

Gitee: gitee.com/busmust/python-can

python-lin

GitHub: github.com/busmust/python-lin

Gitee: gitee.com/busmust/python-lin

10 Applications, Wiring, and Resources

Application	Role of the card
Road / vehicle acquisition	Adds multi-network CAN FD and LIN interfaces to an existing host for vehicle communications and test-condition logging
Parallel production / endurance tests	Multiple cards cover multiple stations, monitoring communications, cyclic traffic, and DUT status
Node / residual bus simulation	Communicates with real DUTs; SDK or BMCLI simulates missing network nodes
HIL benches	Replays field data as open-loop stimuli and acquires current DUT responses; host test systems integrate closed-loop models and external I/O

DB25 connection options

Use a DB25 male accessory with the card DB25 female connector. A 25-core breakout cable connects to custom harnesses or terminal blocks; verify conductor labels against pin numbers. A solderless screw-terminal adapter simplifies bench rewiring. Both follow the pinout on page 6 and leave electrical specifications unchanged.

Installation and wiring checks

Check channel mapping, CAN_H / CAN_L, and GND. Set termination for the bus topology. For CAN FD, match arbitration and data-phase parameters. For LIN, match master/slave role, bit rate, checksum, and pull-up. Confirm basic messaging before long-duration acquisition.

High-speed CAN FD wiring

Above 2 Mbps data phase, the original specification recommends high-quality, low-capacitance twisted-pair wiring totaling no more than approximately 1.5 m, suitable shielding, and correct termination. Set sample points for the network and configure TDC on the peer controller. Validate the actual link for its wiring and node conditions.

Product and development resources

[Product page](#) · [Software downloads](#) · [BMAPI development guide](#)

[BMCLI tutorials and reference](#) · [BUSMUST GitHub](#) · [BUSMUST Gitee](#)

Website: busmust.com · Technical support: support@busmust.com

Revision E: corrected software channel capacities, 1 ms minimum period, CAN startup tasks, and approximately 250 KB persistent hardware replay.