XYPHROLABS

GPIBEE

USB & Ethernet to GPIB Adapter

User Manual

Firmware Version V1.6.2608290003

Engineered and made in Germany

www.GPIBee.com

Table of Contents

1. Introduction 4
2. Key Features 6
3. Typical Applications 7
4. Interfaces 8
   - 4.1 Interface Overview 8
   - 4.2 Interface Capability Summary 8
   - 4.3 Ethernet Interface 9
   - 4.4 USB Interface 9
5. Power Supply 11
6. Indicators and Controls 12
   - 6.1 LED Indicators 12
   - 6.2 Button Functions 12
7. Startup Behavior 14
8. Connecting to GPIBEE 15
   - 8.1 Web Interface Access 15
     - IP Address Assignment 15
     - Discovery 15
     - Opening the Web Interface 15
   - 8.2 VXI-11 15
     - VISA Resource String Format 16
     - Instance Names and GPIB Address Mapping 16
     - Recommended VISA Implementations 16
     - Interactive VISA Tools 17
     - Abort Channel 17
     - Low-Level GPIB Access (doCommands) 18
   - 8.3 USBTMC 20
     - AutoID and Resource Naming 20
     - Multi-Device Access via USBTMC 21
   - 8.4 Linux GPIB 21
9. Firmware Update Procedure 22
   - 9.1 Firmware Version Syntax 22
10. Internal SCPI Command Reference 23
    - 10.1 General Characteristics 23
    - 10.2 Identification Commands 24
    - 10.3 System Commands 24
    - 10.4 Ethernet Configuration Commands 25
    - 10.5 USB Configuration Commands 30
    - 10.6 VXI-11 Configuration Commands 39
    - 10.7 GPIB Commands 41
11. ++ Protocol Reference 46

11.1 ++ Command Reference 46

12. Technical Specifications 48

12.1 Electrical and Communication Specifications 48
12.2 Measured Transfer Performance 48
12.3 Software Compatibility 49
12.4 Physical Characteristics and Environment 49
12.5 Package Contents 50

Acknowledgements 51

13. Open Source Acknowledgements 52

14. Release Information 53

15. Disclaimer and Contact Information 57

General Disclaimer 57
Intended Use 57
Trademarks 57
Liability Limitation 57
Technical Changes 57
Contact Information 58

1. Introduction

GPIBEE is a compact, robust, multi-interface GPIB controller designed for laboratory, industrial, and automated test environments. Its solid aluminum housing provides mechanical durability, while a carefully implemented firmware delivers reliable, well-tested operation across all supported interfaces. It combines Ethernet, USB, and UART communication with full GPIB bus control, discovery mechanisms, a built-in web interface, and an internal SCPI command system. On startup, the device automatically selects between Ethernet and USB connectivity based on link detection. The active USB protocol mode (Network, USBTMC, UART++, or Linux GPIB) is configured by the user and does not change automatically.

GPIBEE supports the full spectrum of GPIB-based instruments — from the latest modern laboratory equipment to vintage measurement devices dating back to the 1970s. GPIB (also known as HPIB, the Hewlett-Packard Interface Bus, and standardized as IEEE-488.1 and IEEE-488.2) remains widely deployed across both current and classic test equipment, and **GPIBEE** is designed to work reliably with all of them. It offers flexible connectivity, robust protocol handling, and a powerful configuration interface accessible through Ethernet, USB, or SCPI.

GPIBEE is intentionally designed to remain isolated from external network resources. It can operate without a gateway, does not issue DNS queries, and never initiates connections to outside servers.

Product Summary	At a Glance
GPIBEE is a compact, multi-protocol GPIB controller for laboratory, industrial, and automated test environments. It integrates Ethernet and USB connectivity with full IEEE-488 bus control, automatic discovery, an integrated web interface, and a built-in SCPI command system. By relying on VISA-supported standard protocols such as VXI-11 and USBTMC, **GPIBEE** fits seamlessly into modern measurement workflows without requiring proprietary drivers.	• USB and Ethernet connectivity • Power over Ethernet or USB power supply • IEEE-488/GPIB instrument interface • Non-proprietary VISA-compatible software workflows • Bench, rack, and ATE deployment • Multi-protocol, multiple GPIB devices, multi-OS support • Modern and legacy GPIB instrument support

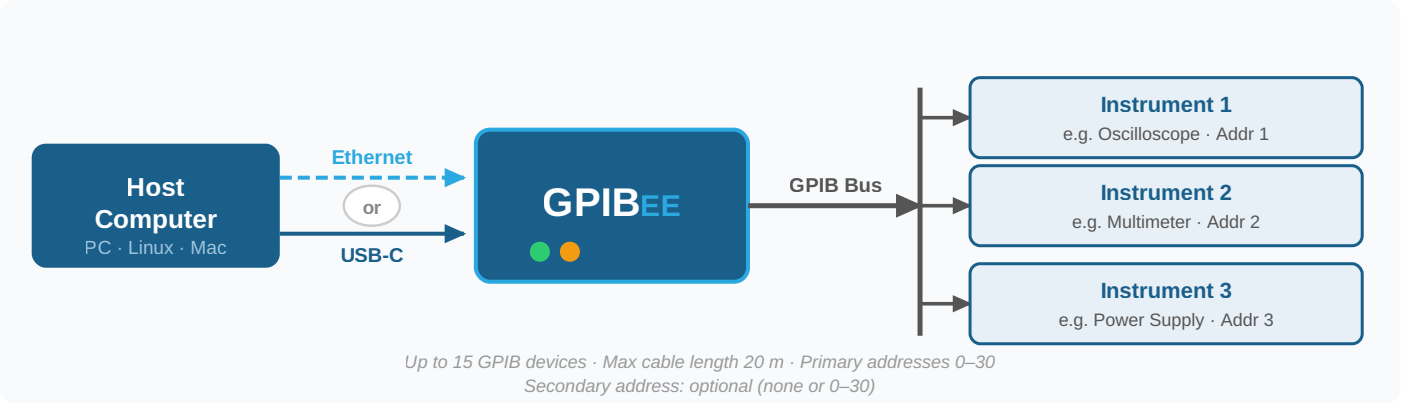

Figure 1 — System topology. The host computer connects via Ethernet or USB-C to GPIBee, which controls up to 15 GPIB instruments on the bus.

2. Key Features

Category	Key Features
GPIB Control	Full GPIB controller implementation with SRQ handling, serial poll support, transparent data pass-through, non-limited read/write length, standards-compliant timeout and lock handling, and full primary/secondary addressing.
Bus Configuration	Adjustable T1 talker delay, configurable GPIB controller address, configurable read termination using EOI or ASCII character, and support for up to 15 GPIB instruments on one bus.
Connectivity	USB and Ethernet host connectivity with automatic interface selection, Ethernet priority over USB, and automatic USB fallback when Ethernet is not connected.
Networking & Discovery	DHCP client with configurable timeout, integrated DHCP server for USB network mode, embedded webserver for configuration and status access, and discovery via mDNS, SSDP, VXI-11, and LXI.
USB Modes	Selectable USB modes: Network, USBTMC, UART ++, and Linux GPIB; supports USB High-Speed 480 Mbit/s and Full-Speed 12 Mbit/s operation with optional forced Full-Speed.
Software Integration	Designed for VISA-based environments such as LabVIEW, Python/PyVISA, MATLAB, and automated test software using non-proprietary standard protocols; includes automatic USBTMC IDN-based serial number generation.
Deployment & Configuration	Front-panel indicators for interface activity and adapter status; solid aluminum case with compact form factor; configuration via web interface, internal SCPI, or network settings.
Firmware Maintenance	Field-upgradeable firmware with active maintenance for ongoing improvements, compatibility updates, and long-term product support.

3. Typical Applications

Application Area	Use Case
Automated Test Equipment	Remote control of legacy GPIB instruments from modern test controllers. Characterization and EMC/I test automation.
Research & Development	Bench automation for oscilloscopes, power supplies, meters, and analyzers.
Manufacturing Test	Instrument control in production validation and end-of-line testing.
Service & Calibration	Portable adapter for field diagnostics and calibration workflows.
Education	Modern computer connectivity for established laboratory equipment.

4. Interfaces

4.1 Interface Overview

Interface	Protocol / Mode	Supported Access Methods
Ethernet	VXI-11	VISA-compatible LAN instrument access; multiple simultaneous connections; VXI-11 locking and discovery; SRQ service requests.
	++ Protocol	TCP port 1234; identical to USB UART ++ mode.
	Web Interface	Browser-based configuration, status, and GPIB console.
USB	Network Mode (VXI-11)	Enumerates as USB Ethernet adapter; access via web interface, VXI-11, and ++ protocol on TCP port 1234.
	USBTMC	Exposes a GPIB device as a USBTMC instrument; compatible with NI VISA, R&S VISA, Keysight VISA, PyVISA, Linux kernel driver.
	UART ++	Virtual serial port; ++ protocol; no baud rate configuration required.
	Linux GPIB	Compatible with Linux GPIB package; use board_type xyphro_ugc.
GPIB	IEEE-488 Bus	Controller with full primary/secondary addressing, SRQ, serial poll, and support for up to 15 instruments.

4.2 Interface Capability Summary

The following table summarizes the capabilities available across each interface and protocol combination.

Interface	Protocol	Multi-device	Subaddr.	SRQ	Clear	Trigger	Serial Poll	GoToLocal	Adj. Ctrl Addr
Ethernet	VXI-11	✓	✓	✓	✓	✓	✓	✓	✓
	++	✓	✓	✓	✓	✓	✓	✓	✓
USB	Network (VXI-11)	✓	✓	✓	✓	✓	✓	✓	✓
	USBTMC	⚠	✓	✓	✓	✓	✓	✓	✓
	UART ++	✓	✓	✓	✓	✓	✓	✓	✓

Interface	Protocol	Multi-device	Subaddr.	SRQ	Clear	Trigger	Serial Poll	GoToLocal	Adj. Ctrl Addr
	Linux GPIB	✓	✓	✓	✓	✓	✓	✓	✓

Note: USBTMC multi-device access requires switching the active GPIB address via the internal SCPI command SYST:COMM:USBTMC:ADDR. Only one device is active at a time in this mode. For full simultaneous multi-device access, use VXI-11 (USB Network or Ethernet mode) or Linux GPIB.

4.3 Ethernet Interface

When connected to a network, **GPIBEE** operates as a LAN-based GPIB controller. Two protocols are available: VXI-11 (supporting multiple simultaneous connections, VXI-11 locking, VXI-11 discovery, VISA compatibility, and SRQ service requests) and the ++ Protocol (identical to USB UART ++ mode, on TCP port 1234).

The Ethernet interface supports DHCP with fallback to a static IP, static addressing without DHCP, mDNS, SSDP, and full GPIB subaddressing. Network parameters such as hostname, IP address, subnet mask, and timing are fully configurable.

Supported Ethernet link modes:

- 10/100 Mbit/s
- Full and half duplex
- Autonegotiation
- Auto-MDI/MDI-X

4.4 USB Interface

GPIBEE provides four selectable USB modes. Only one mode is active at a time. USB supports both High-Speed (480 Mbit/s) and Full-Speed (12 Mbit/s) operation; forcing 12 Mbit/s Full-Speed is possible via configuration.

USB Mode	Description	Typical Use
USB Network Mode	Enumerates as a USB Ethernet adapter (USB CDC NCM class). The integrated DHCP server assigns an IP address to the host. Accessible via web interface, VXI-11, and ++ protocol on TCP port 1234.	VXI-11 over USB without a network infrastructure. Quick settings access.

USB Mode	Description	Typical Use
USBTMC Mode	Exposes a selected GPIB device as a USBTMC instrument. Device address is auto-detected or manually set. Supports AutoID (IDN-based serial number generation).	Direct VISA-style USB instrument access.
Linux GPIB Mode	Provides a USB interface compatible with the Linux GPIB package, operating as a GPIB controller using the xyphro_ugc board type.	Native Linux GPIB workflows and existing Linux-based test setups.
USB UART ++ Mode	Enumerates as a virtual serial port and supports the ++ protocol over UART. Baud rate and parity are irrelevant for this virtual UART.	Simple scripting, interactive communication, and Prologix-compatible workflows.

5. Power Supply

GPIBEE can be powered via USB or Power over Ethernet (PoE). Power source selection is independent of the active communication interface.

Scenario	Power Source
USB only	USB bus power supplies the device
Ethernet with PoE	PoE switch or injector (IEEE 802.3af Mode A and Mode B)
Ethernet without PoE	USB provides +5 V power (USB cable must be connected)

Parameter	Specification
Power sources	USB bus power or Power over Ethernet (IEEE 802.3af Mode A and Mode B)
USB current consumption	Typical idle: approximately 80 mA; maximum: 150 mA
PoE power consumption	Approximately 1.5 W typical

6. Indicators and Controls

6.1 LED Indicators

- **Green LED:** Ethernet operation mode active. Pulses while data is being transferred.
- **Orange LED:** USB operation mode active. In USBTMC mode, flashes while waiting for a GPIB instrument to be connected. Pulses while data is transferred.
- Both LEDs blink twice on power-up or when switching from Network to USB mode.
- Both LEDs flash alternately when in firmware update mode.

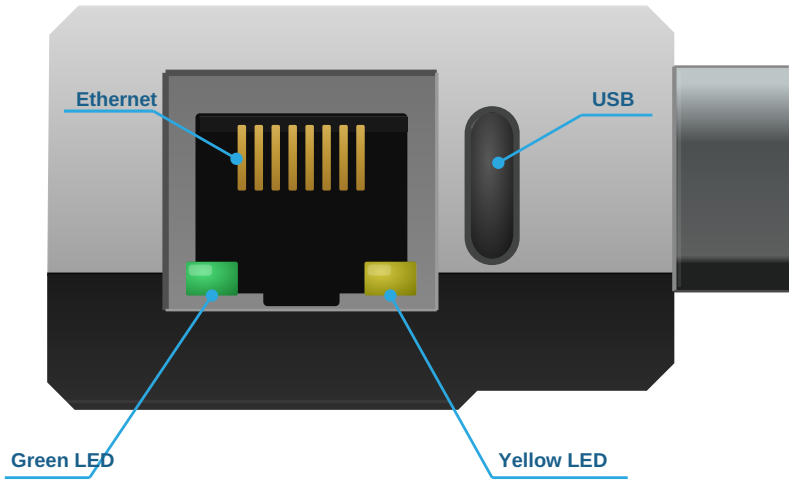

Figure 2 — LED indicator location (connector side view). The green LED indicates Ethernet/network mode; the orange LED indicates USB mode.

6.2 Button Functions

A button on the bottom side of **GPIBEE** provides the following functions:

Action	Function
Press 5 times within 2 seconds	Factory Reset: Device blinks both LEDs 5 times quickly to confirm reset. Power-cycle the device afterwards to activate the new settings.
Hold for 5 seconds	Enter Firmware Update Mode: Both LEDs flash alternately. **GPIBEE** appears as a USB mass-storage device.
Hold button, plug in USB, release	Alternative Firmware Update Entry: Hold button with no cables connected, plug in USB, then release.

Action	Function
Press once, wait ~3 seconds	Fast USB Network Mode: Temporarily enters USB network mode for easy access to the web interface without using the internal SCPI interface. Press again to return to the previous operation mode.

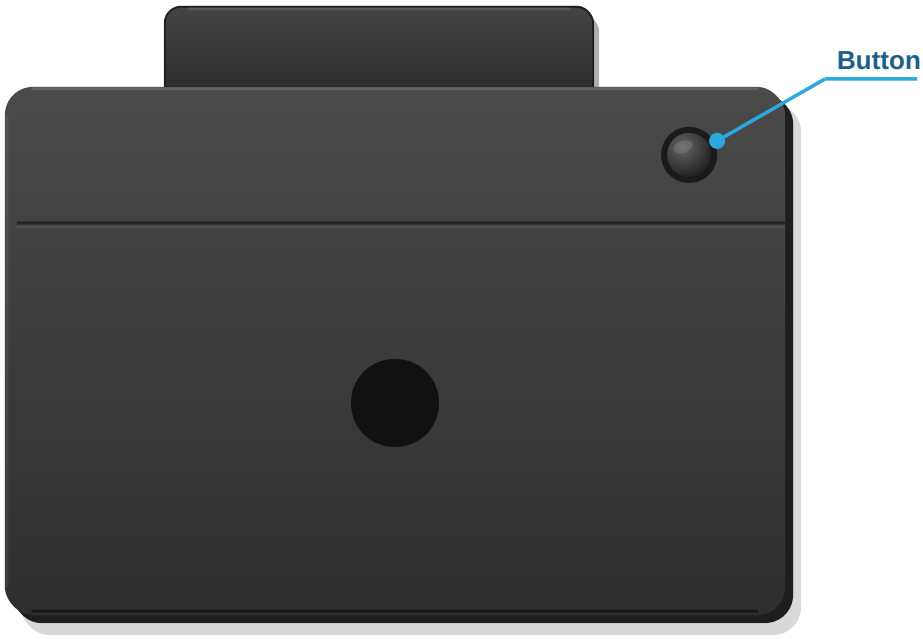

Figure 3 — Button location (backside view, GPIB connector side facing up). Button is located in the upper-right area.

7. Startup Behavior

After power-up, **GPIBEE** checks for Ethernet connectivity and waits for a configurable time (default: 4 seconds). If no network is detected, it switches to USB mode and enumerates.

When entering USB mode, behavior depends on the selected USB mode:

- **USBTMC mode:** Waits for a powered GPIB device before enumerating.
- **USB Network mode:** Enumerates immediately as a network adapter (USB CDC NCM class).
- **USB UART ++ mode:** Enumerates immediately as a virtual serial port.
- **Linux GPIB mode:** Enumerates immediately, regardless of whether a GPIB device is connected.

When the device is in USB mode, plugging in an Ethernet cable will cause it to switch back to Ethernet operation.

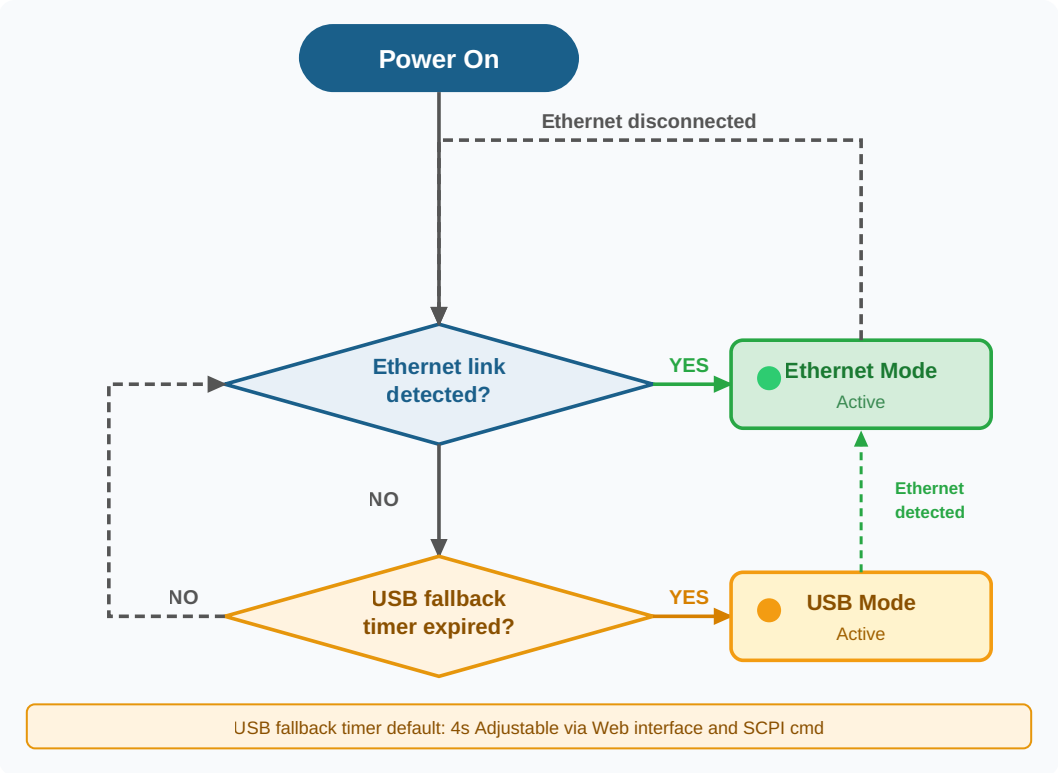

Figure 4 — Startup and interface selection flowchart.

8. Connecting to GPIBEE

8.1 Web Interface Access

IP Address Assignment

Depending on the interface used, **GPIBEE** appears under different IP addresses:

Mode	IP Address
USB Network Mode	Default device IP: `192.168.3.2` (configurable via web interface or internal SCPI).
Ethernet Mode — with DHCP	**GPIBEE** receives a DHCP address from the router.
Ethernet Mode — without DHCP	Default fallback IP: `192.168.1.2` | Default subnet: `255.255.255.0` | Default gateway: `0.0.0.0` (no gateway). All values are configurable.

Discovery

GPIBEE announces itself via the following discovery protocols (applies to both Ethernet and USB Network mode):

- **SSDP** — visible in Windows Network Explorer and similar tools
- **mDNS** — multicast DNS service advertisement
- **LXI Discovery** — LXI-compatible instrument discovery

Most VISA implementations will automatically detect **GPIBEE** . You can always connect directly using the device IP address as an alternative.

Opening the Web Interface

Enter the device IP address in any web browser. Examples:

```
http://192.168.3.2      # USB Network Mode default
http://192.168.1.2      # Ethernet fallback (no DHCP)
http://<DHCP-address>   # Ethernet with DHCP
```

8.2 VXI-11

GPIBEE supports VXI-11 over both USB Network mode and Ethernet mode. VXI-11 discovery is implemented via VXI-11 discovery, LXI discovery, and mDNS service advertisement of the internal SCPI interface (inst0). Most VISA implementations will automatically detect **GPIBEE**.

VISA Resource String Format

```
TCPIP0::<IP>::<instancename>::INSTR

Example:
TCPIP0::192.168.3.2::inst0::INSTR
```

Instance Names and GPIB Address Mapping

The instance name selects which GPIB device to access:

Instance Name	Meaning
`inst0` or `gpib0` or `hpib0`	Internal SCPI interface (**GPIB**EE itself). Use to read firmware version, configure settings, and control **GPIB**EE.
`inst0,22` or `gpib0,22` or `hpib0,22`	GPIB device at primary address 22.
`inst0,22,4` or `gpib0,22,4` or `hpib0,22,4`	GPIB device at primary address 22, secondary address 4.

The allowed primary address range is 0 to 30. The currently configured GPIB controller address (default: 0) cannot be accessed as an instrument. Supported secondary address ranges are 0–30 and 96–126 (96–126 mapped to 0–30 for compatibility).

An instance name must begin with one of the prefixes `inst` , `gpib` or `hpib` , immediately followed by `0` . No other prefixes are accepted, and the only permitted separator between the interface, the primary address and the secondary address is a comma. Names that do not follow this scheme are rejected. Some VISA implementations (e.g. R&S VISA) only accept the inst prefix, so `inst0` is recommended for maximum compatibility.

> **Changed in V1.5:** Earlier firmware versions accepted any alphabetic prefix and allowed an underscore or dash instead of a comma. Both are no longer permitted, as required for VXI-11 compliance. Resource strings using a non-standard prefix or separator must be updated.

Recommended VISA Implementations

- R&S VISA (lightweight, robust)
- NI VISA (widely used, comprehensive)
- Keysight VISA

> **Note:** Several issues were identified in pyvisa-py during development, especially around lock handling, abort channel support, and SRQ handling. Most basic operations work, but advanced VXI-11 features may not behave as expected. Significant improvements have been made to pyvisa-py since then, so if you run into any of these problems, first make sure you are using an up-to-date version of pyvisa-py before investigating further.

Interactive VISA Tools

Most VISA packages include a diagnostic tool: R&S VISA Tester, NI MAX, or Keysight Connection Expert. These tools are useful for auto-discovering VXI-11 devices, interactive SCPI communication, and initial bring-up and troubleshooting.

Abort Channel

GPIB transactions can take a long time to finish. A read from an instrument that never becomes a talker, or a write to a device that never releases its handshake, only ends once the configured timeout expires — which may be several seconds or more. VXI-11 defines a per-operation timeout for exactly this case, and **GPIBEE** implements it properly: every operation returns on its own once the timeout requested by the client has elapsed. Reliable timeout behavior therefore does not depend on the abort channel.

For the situations where waiting for the timeout is not acceptable, VXI-11 additionally defines the *abort channel* — a second, asynchronous TCP connection that runs in parallel to the normal (synchronous) core channel.

Because the core channel is blocked while an operation is running, a client cannot use it to cancel that operation. The abort channel gives the client a way in: it sends a `device_abort` call referring to the link whose operation should be terminated. Aborting does not close the link, does not drop the session, and does not require the instrument to respond.

GPIBEE implements the abort channel fully. When an abort is received, the GPIB transaction in progress is stopped at the next possible point and the blocked call on the core channel returns immediately with VXI-11 error 23 (`abort`). The link stays usable afterwards, so the next operation can be started without re-opening the connection.

The abort call is normally issued from a second thread, since the thread performing the transfer is blocked. Typical uses are a responsive “Stop” button in an application, cutting short a read into an unresponsive instrument when a long timeout is deliberately configured, and shutting an application down cleanly without waiting out a pending operation.

> **Note:** The abort channel is optional, not a prerequisite for correct operation. Applications that simply configure a suitable VISA timeout do not need it — it only helps in boundary cases where an operation must end earlier than the timeout it was started with. Also note that not every VISA implementation exposes the abort channel to the user, and some open it only on demand. If aborts appear to have no effect, verify with a raw VXI-11 client that the abort channel is actually being opened by the client library.

Low-Level GPIB Access (doCommands)

Beyond reading and writing device data, VXI-11 defines a generic operation called `device_docmd` for interface-specific, low-level transactions. VXI-11.2 — the TCP/IP to IEEE 488.1 interface specification — standardizes the set of these commands for GPIB. They are commonly referred to as *doCommands*.

doCommands provide direct access to the GPIB bus itself: sending arbitrary command (ATN true) byte sequences, reading the state of the bus lines, and driving ATN, REN and IFC. Most VISA implementations do not expose these operations for a TCPIP INSTR resource, so they enable things that are otherwise out of reach — for example addressing several instruments as listeners and issuing a single Group Execute Trigger so that they all trigger at the same moment.

GPIBEE implements the VXI-11.2 doCommand set as follows, plus one vendor-specific extension for GPIB parallel poll:

doCommand	cmd value	Function	Implementation Status
Send Command	`0x020000`	Sends a user-specified sequence of GPIB command bytes (ATN asserted) onto the bus.	Fully implemented.
Bus Status	`0x020001`	Reads individual bus lines and controller status flags — ATN/REN state, SRQ, NDAC, controller address, and so on.	Implemented with limitations — see the sub-query table below.
ATN Control	`0x020002`	Sets the state of the ATN line.	Fully implemented.
REN Control	`0x020003`	Sets the state of the REN line.	Fully implemented.
Pass Control	`0x020004`	Passes controller-in-charge status to another GPIB controller on the bus.	Not implemented. Returns VXI-11 error 8 (`OPERATION_NOT_SUPPORTED`). Reserved for a future firmware version.

doCommand	cmd value	Function	Implementation Status
Bus Address	`0x02000A`	Sets the GPIB primary address of the controller itself.	Fully implemented.
IFC Control	`0x020010`	Asserts an interface clear.	Fully implemented — a complete interface clear sequence is executed.
Parallel Poll	`0x021000`	Executes an IEEE 488.1 parallel poll and returns the 1-byte poll response. Not part of VXI-11.2 Table B.1 — a **GPIBEE** -specific extension, using a cmd value inside the same reserved range Table B.1's own commands occupy. Requires no data_in; data_out is always exactly 1 byte.	Fully implemented — a real parallel poll is executed on the bus and the actual response byte is returned.

The *Bus Status* doCommand takes a selector that chooses which item is reported. The behavior of each selector is:

Selector	Query	Behavior
1	Remote	Reports the actual state of the REN line. Implemented as intended.
2	SRQ	Reports the actual state of the SRQ line. Implemented as intended.
3	NDAC	Reports the actual state of the NDAC line. Implemented as intended.
4	System Controller	Always returns 1. **GPIBEE** is always the system controller.
5	Controller in Charge	Approximated from the bus line state: returns 1 if no bus line is driven (all lines idle/high), otherwise 0. Since control is never handed over (Pass Control is not implemented), **GPIBEE** is in practice always the controller in charge. This is a provisional implementation and may be replaced by proper state tracking in a future firmware version.
6	Talker	Always returns 0.
7	Listener	Always returns 0.

Selector	Query	Behavior
8	Bus Address	Returns the currently configured GPIB controller address of **GPIBEE** (default: 0).

> **Important:** VXI-11.2 Rule B.5.2 requires that `device_docmd` is only offered on a link identifier that is *not* associated with a device. doCommands are therefore accepted only on links opened with an interface instance name — `inst0` , `gpib0` or `hpib0` . On any instance name that specifies a GPIB address (for example `inst0,22`) doCommands are rejected. Note that this is the same link used for the internal SCPI interface, so one connection can serve both purposes.

As an example, the following command byte sequence sent through *Send Command* addresses the instruments at primary addresses 5 and 7 as listeners and triggers both of them simultaneously:

```
0x3F   UNL   Unlisten – clear all current listeners
0x25   LAD 5 Address device at primary address 5 as listener
0x27   LAD 7 Address device at primary address 7 as listener
0x08   GET   Group Execute Trigger
```

> **Note:** doCommands bypass the bus state that a VISA implementation keeps track of internally. Driving ATN, REN or IFC, or re-addressing the bus while another link has a transfer in progress, can disturb that transfer. Use them deliberately, and preferably while no other operation is running.

Since most VISA implementations do not expose `device_docmd` for TCPIP resources, a raw VXI-11 client library that implements the VXI-11.2 command set (for example `python-vxi11`) is usually the most practical way to use these operations.

8.3 USBTMC

GPIBEE implements a USBTMC interface similar to UsbGPIB-V2, providing a 1:1 connection between the PC and a single GPIB device. USBTMC devices are automatically detected by NI VISA, R&S VISA, Keysight VISA, PyVISA, and the Linux kernel USBTMC driver.

AutoID and Resource Naming

The USBTMC resource name can be configured via the web interface or internal SCPI commands. **GPIBEE** includes an optional AutoID mechanism that automatically queries the connected GPIB device using *IDN?, embeds the instrument identity into the USBTMC resource name, and ensures the VISA resource name reflects the actual instrument. This is extremely helpful when swapping instruments. Only use AutoID with instruments that support the *IDN? query.

Multi-Device Access via USBTMC

The USBTMC standard does not support multiple logical channels. To allow multi-device access, **GPIBEE** provides an internal SCPI command (SYST:COMM:USBTMC:ADDR) to switch the active GPIB address. Only one device is active at a time and the user must select the correct device before sending commands.

For full multi-device capability with independent logical channels, use USB Network Mode with VXI-11, Ethernet VXI-11, or Linux GPIB. This is the recommended primary connection method for most users when controlling multiple instruments.

8.4 Linux GPIB

Linux GPIB provides an API compatible with NI-488.2 without vendor lock-in and can be controlled from C, Python, and many other programming languages.

Install the Linux GPIB package from https://linux-gpib.sourceforge.io/ and add the following to /etc/gpib.conf (removing any other interface declarations):

```
interface {
        minor = 0
        board_type = "xyphro_ugc"
        name = "gpib0"
        pad = 0
        master = yes
}
```

After editing the configuration, call gpib_config to activate the new settings. Use the ibtest application for initial interactive tests with connected instruments.

9. Firmware Update Procedure

1. Enter Bootloader mode (see Section 6.2 — Button Functions: hold button 5 seconds, or hold during USB connect). The same procedure can be used to upgrade to a newer firmware version or downgrade to an older one.
2. GPIBee appears as a USB mass-storage device (similar to a USB flash drive). Both LEDs flash alternately to indicate firmware update mode.
3. Copy the firmware file `GPIBee.GFW` to the drive.
4. Wait approximately 10 seconds until the file is copied. The device restarts automatically once programming completes successfully.

Firmware files can be downloaded from the product firmware repository: github.com/xyphro/GPIBee (check releases for the latest and previous versions).

The programmed firmware version can be checked at any time by:

- Opening `INFO.TXT` while in bootloader mode.
- Web interface (Home section).
- Querying the internal SCPI interface: `*IDN?`

9.1 Firmware Version Syntax

The firmware version consists of three parts: V1.2.2604151622

Part	Meaning
1	Major firmware version
2	Minor firmware version
2604151622	Build version (auto-incremented each compilation)

10. Internal SCPI Command Reference

10.1 General Characteristics

The internal SCPI command interface provides programmatic access to all internal settings. Most settings can also be changed via the built-in web interface — accessible by navigating to the device IP address in any browser — but the SCPI interface allows dynamic configuration from within controlling measurement software.

Access the internal SCPI interface depending on protocol:

- **VXI-11** (over USB or Ethernet): Open instance name `inst0`.
- **++ Protocol** mode: Access GPIB primary address 31 (`++addr 31`).
- **USBTMC** : Requires a custom control-in transfer to enter vendor mode (see below).
- **Linux GPIB** : The internal SCPI interface cannot be accessed. Use the web server or button shortcut to change settings.
- **Button shortcut (non-network mode)** : When operating in USB UART++, USB USBTMC, or Linux GPIB mode, press the button once and wait 3 seconds — the device temporarily switches to USB Network mode, making the web interface accessible. Press the button again to return to the previous mode.

USBTMC vendor mode entry (enter internal SCPI via PyVISA):

```
# Enter internal SCPI mode:
dev.control_in(0xa1, 180, 0, 0, 1)   # bmRequest=0xA1, bRequest=180, wValue=0

# Exit internal SCPI mode:
dev.control_in(0xa1, 180, 1, 0, 1)   # wValue=1
# Or send SCPI command: SYST:COMM:USBTMC:EXITV
```

General command rules:

- Commands are case-insensitive.
- Commands may be abbreviated: uppercase letters are mandatory, lowercase letters may be omitted. For example, `SYST:REB` instead of `SYSTEM:REBOOT`.
- Commands may be terminated with no terminator, <CR>, <LF>, or <CR><LF>. Query responses are sent without termination characters.
- Most settings modify non-volatile memory. A reboot or power cycle is recommended after changes.

10.2 Identification Commands

*IDN?

Returns the device identification string as a comma-separated list of four fields: manufacturer name, instrument name, device serial number, and firmware version. This is a query-only command.

QUERY:

```
*IDN?
```

Reply: `XYPHROLABS,GPIBee,LJN9K,V1.2.2604151253`

> **Note:** Response field breakdown: field 1 (XYPHROLABS) = manufacturer; field 2 (**GPIB**EE) = product name; field 3 (LJN9K) = 5-character unique alphanumeric device serial number (e.g. LJN9K); field 4 (V1.2.2604151253) = firmware version with format V{major}.{minor}.{YYMMDDHHMM} where the last part is a build timestamp.

10.3 System Commands

SYSTem:REBoot

Reboots the **GPIB**EE firmware. Equivalent to a power cycle. The device restarts very quickly but may take several seconds to become accessible again due to network identification delay or the configured USB fallback timeout. The current connection is terminated — a new connection must be established after rebooting.

SET:

```
SYST:REB
```

SYSTem:PRESet

Resets all non-volatile settings to factory defaults. Equivalent to pressing the front-panel button 5 times quickly. No automatic reboot is performed after this command, giving you the opportunity to change additional settings before rebooting. Apply the reset settings by issuing SYSTem:REBoot or cycling power.

SET:

```
SYST:PRES
```

10.4 Ethernet Configuration Commands

SYSTem:COMMunicate:LAN:IPADdress

Sets the Ethernet interface IP address used when DHCP is disabled or has timed out. The address must be in dotted-decimal notation enclosed in double quotes. Loopback (127.x.x.x) and multicast (239.x.x.x, 240.x.x.x) addresses are not accepted and cause a fallback to the default IP. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:IPAD "192.168.1.150"
```

QUERY:

```
SYST:COMM:LAN:IPAD?
```

Reply: `"192.168.1.2"`

→ *Configured static IP address (not necessarily the active IP while DHCP is enabled)*

SYSTem:COMMunicate:LAN:SUBNet

Sets the subnet mask for the Ethernet interface. Masks smaller than /29 (255.255.255.248) are not accepted. Non-contiguous or otherwise invalid masks (e.g. 255.0.255.0) are also rejected. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:SUBNET "255.255.255.0"
```

QUERY:

```
SYST:COMM:LAN:SUBNET?
```

Reply: `"255.255.255.0"`

→ *Currently configured subnet mask (/24 in this example, supporting up to 254 hosts on the subnet)*

SYSTem:COMMunicate:LAN:GateWay

Sets the default gateway address for the Ethernet interface. A value of 0.0.0.0 (factory default) disables gateway routing; traffic is limited to the local subnet only. Loopback and multicast addresses are not accepted. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:GW "192.168.1.1"
```

QUERY:

```
SYST:COMM:LAN:GW?
```

Reply: `"0.0.0.0"`
→ *No gateway configured (default); device cannot route traffic beyond the local subnet*

Reply: `"192.168.1.1"`
→ *Gateway is set; device can reach remote networks through this router*

SYSTem:COMMunicate:LAN:DHCP

Enables or disables the DHCP client on the Ethernet interface. When enabled, **GPIBEE** attempts to obtain its IP address automatically from a DHCP server on the network. If the server does not respond within the configured timeout, the static IP settings are used as fallback. Accepts ON/OFF or 1/0.

SET:

```
SYST:COMM:LAN:DHCP ON
```

→ *Enable DHCP client (auto IP assignment)*

```
SYST:COMM:LAN:DHCP OFF
```

→ *Disable DHCP, use static IP settings*

QUERY:

```
SYST:COMM:LAN:DHCP?
```

Reply: `ON`
→ *DHCP client is active; IP address is being obtained from the network DHCP server*

Reply: `OFF`
→ *DHCP is disabled; the configured static IP address, subnet mask, and gateway are used*

SYSTem:COMMunicate:LAN:DHCP:TIMEout

Sets the DHCP timeout in seconds (range: 1–15). If no DHCP response is received within this period during boot, **GPIBEE** falls back to the configured static IP settings. Shorter values speed up boot time on networks without a DHCP server. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:DHCP:TIME 5
```

→ *Fall back to static IP after 5 s without a DHCP response*

QUERY:

```
SYST:COMM:LAN:DHCP:TIME?
```

Reply: `4`

→ ***GPIBEE** waits 4 seconds for a DHCP response; if none received, it switches to the configured static IP*

SYSTem:COMMunicate:LAN:MDNS

Enables or disables mDNS (Multicast DNS / Bonjour) service discovery. When enabled, the device can be reached by hostname on the local network (e.g. gpibee-XXXXX.local) without knowing its IP address. Affects both the Ethernet interface and USB Network mode. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:MDNS ON
```

→ *Enable hostname-based discovery*

```
SYST:COMM:LAN:MDNS OFF
```

→ *Disable mDNS (use IP address to connect)*

QUERY:

```
SYST:COMM:LAN:MDNS?
```

Reply: `ON`

→ *mDNS is active; device responds to hostname lookups on the local network (e.g. gpibee-LJN9K.local)*

Reply: `OFF`

→ *mDNS is disabled; device must be addressed by its IP address directly*

SYSTem:COMMunicate:LAN:SSDP

Enables or disables SSDP (Simple Service Discovery Protocol) network discovery. When active, the device announces itself via SSDP, making it visible in Windows Network Explorer and compatible with Windows Plug-and-Play network discovery. Affects both Ethernet and USB Network mode. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:SSDP ON
```

→ *Visible to network scanners*

```
SYST:COMM:LAN:SSDP OFF
```

→ *Hidden from SSDP-based discovery*

QUERY:

```
SYST:COMM:LAN:SSDP?
```

Reply: `ON`
→ *SSDP discovery is active; the device announces itself on the network*

Reply: `OFF`
→ *SSDP is disabled; the device is not visible to SSDP-based network scanners*

SYSTem:COMMunicate:LAN:HOSTname

Sets the hostname template. The {UID} placeholder is automatically substituted at runtime with the device's 5-character unique serial number, creating a unique per-device hostname. The string is automatically sanitized to remove illegal hostname characters. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:HOST GPIBee-{UID}
```

→ *Resolves at runtime to e.g. GPIBEE-LJN9K*

QUERY:

```
SYST:COMM:LAN:HOST?
```

Reply: `"GPIBee-{UID}"`
→ *The stored hostname template; {UID} is substituted with the actual serial number at runtime, not stored here*

SYSTem:COMMunicate:LAN:PPprot

Enables or disables the ++ protocol TCP server on port 1234. When enabled, instruments can be controlled using the ++ ASCII protocol over any TCP/IP connection, in both Ethernet and USB Network mode. Disable to close port 1234 if the ++ protocol is not needed. Takes effect after a reboot.

SET:

```
SYST:COMM:LAN:PP ON
```

→ *++ protocol server listening on TCP port 1234*

```
SYST:COMM:LAN:PP OFF
```

→ *TCP port 1234 closed*

QUERY:

```
SYST:COMM:LAN:PP?
```

Reply: `ON`
→ *The ++ protocol server is active and listening on TCP port 1234*

Reply: `OFF`
→ *The ++ protocol server is disabled; TCP port 1234 is not accessible*

10.5 USB Configuration Commands

SYSTem:COMMunicate:USB:PROTocol

Selects the USB interface protocol mode. Each mode presents a different USB device class to the host operating system. Takes effect after a reboot.

SET:

```
SYST:COMM:USB:PROT USBTMC
```

→ *USB Test & Measurement class (instrument visible as VISA resource)*

```
SYST:COMM:USB:PROT NET
```

→ *USB CDC NCM Ethernet (VXI-11 and ++ protocol over USB, accessible as VISA resource)*

```
SYST:COMM:USB:PROT UART
```

→ *USB CDC serial port with ++ protocol*

```
SYST:COMM:USB:PROT LINUXGPIB
```

→ *Native Linux GPIB USB driver mode*

QUERY:

```
SYST:COMM:USB:PROT?
```

Reply: `USBTMC`
→ *USBTMC mode is active; instrument appears as a VISA USBTMC resource to the host*

Reply: `NET`
→ *USB Network mode is active; VXI-11 and ++ protocol are accessible via USB*

Reply: `UART`
→ *UART/serial mode is active; ++ protocol is available via a virtual COM port*

Reply: `LINUXGPIB`
→ *Linux GPIB native driver mode is active*

SYSTem:COMMunicate:USB:FALLback

Sets the USB fallback timeout in seconds. **GPIBEE** normally prefers the Ethernet interface. If no Ethernet link is detected within this period after power-on, the device activates the configured USB mode instead. A value of 0 causes immediate USB activation without waiting for Ethernet.

SET:

```
SYST:COMM:USB:FALL 5
```

→ *Switch to USB after 5 s without an Ethernet link*

QUERY:

```
SYST:COMM:USB:FALL?
```

Reply: `4`

→ ***GPIBEE** waits 4 seconds for an Ethernet link at boot before activating USB mode*

SYSTem:COMMunicate:USB:FORCEfullspeed

Forces USB Full-Speed (12 Mbit/s) mode. By default, **GPIBEE** negotiates USB High-Speed (480 Mbit/s) when the host supports it. Enable this setting only if compatibility problems occur with specific host controllers or USB hubs that cannot handle High-Speed reliably. Takes effect after a reboot.

SET:

```
SYST:COMM:USB:FORCE OFF
```

→ *Auto: High-Speed (480 Mbit/s) when supported by host*

```
SYST:COMM:USB:FORCE ON
```

→ *Lock to Full-Speed (12 Mbit/s)*

QUERY:

```
SYST:COMM:USB:FORCE?
```

Reply: `OFF`

→ *High-Speed negotiation is active (default); device runs at 480 Mbit/s if the host supports it*

Reply: `ON`

→ *USB Full-Speed is forced at 12 Mbit/s regardless of host capabilities*

SYSTem:COMMunicate:USBNET:IPADdress

Sets the device IP address used in USB Network mode. The subnet mask is fixed at /29 (255.255.255.248), providing 6 usable addresses per segment. The address is automatically corrected to avoid network (.0, .8, ...) and broadcast (.7, .15, ...) addresses. The host PC side of the USB network link is automatically assigned the adjacent IP on the /29 subnet.

SET:

```
SYST:COMM:USBNET:IPAD "192.168.5.2"
```

QUERY:

```
SYST:COMM:USBNET:IPAD?
```

Reply: `"192.168.3.2"`

→ ***GPIBEE*** *USB network interface IP; the host PC is automatically assigned the adjacent address on the /29 (255.255.255.248) subnet*

SYSTem:COMMunicate:USBTMC:NAME

Sets the USBTMC serial number template. This string becomes the USB serial number descriptor that VISA implementations use when building the USBTMC VISA resource string (e.g. USB0::0x16D0::0x1466::<serial>::INSTR). Template substitution keys are evaluated at USBTMC startup. The provided text is automatically sanitised to produce a legal serial number string. Always enclose the parameter in double quotes (e.g. SYST:COMM:USBTMC:NAME "GPIBee-{UID}"). Substitution keys such as {UID}, {ADDR}, and {IDN0}–{IDN3} are resolved at USBTMC startup.

SET:

```
SYST:COMM:USBTMC:NAME "GPIBee-{ADDR}-{UID}"
```

QUERY:

```
SYST:COMM:USBTMC:NAME?
```

Reply: `GPIBee-{ADDR}-{UID}`

→ *The stored template; resolved at USBTMC startup to produce e.g.* ***GPIBEE*** *-5-LJN9K as the USB serial number*

SUBSTITUTION KEYS:

Key	Description	Example Value
`{UID}`	5-character unique serial number of the GPIBee device.	`LJN9K`
`{ADDR}`	GPIB address of the connected instrument. Substituted with a single number (primary only) or two comma-separated numbers (primary,secondary) depending on the detected instrument address.	`5 or 5,96`
`{PADDR}`	Primary GPIB address only. Similar to {ADDR} but always a single number regardless of secondary addressing.	`5`
`{IDN}`	Full *IDN? response from the connected instrument. Replaced with an empty string if the instrument does not respond to *IDN?.	`Rohde&Schwarz,CMU200,123456,V5.22`
`{IDN0}`	Manufacturer name — first field of the *IDN? response.	`Rohde&Schwarz`

Key	Description	Example Value
`{IDN1}`	Instrument model name — second field of the *IDN? response.	`CMU200`
`{IDN2}`	Instrument serial number — third field of the *IDN? response.	`123456`
`{IDN3}`	Instrument firmware version — fourth field of the *IDN? response.	`V5.22`

Length limiting: Any key can be prefixed with a length specifier to cap the substituted text. For example, `{:5IDN}` limits the substitution to 5 characters. Any number from 0 to 126 is valid.

Note: **GPIBEE** only issues a *IDN? query at USBTMC startup when at least one of the keys {IDN}, {IDN0}, {IDN1}, {IDN2}, or {IDN3} is present in the template. When none of these keys appear, no *IDN? query is made.

`SYSTem:COMMunicate:USBTMC:DELAY`

Sets the USBTMC startup delay in seconds. This pause occurs before **GPIBEE** initiates the GPIB bus scan and IDN query at USBTMC startup. Use this when the connected instrument needs extra time to power up and become responsive to GPIB commands.

SET:

```
SYST:COMM:USBTMC:DELAY 2
```

→ *Wait 2 s before initiating the USBTMC startup bus scan*

QUERY:

```
SYST:COMM:USBTMC:DELAY?
```

Reply: `4`

→ ***GPIBEE** waits 4 seconds after USB connection before starting the bus scan at USBTMC startup*

SYSTem:COMMunicate:USBTMC:EXITVendormode

Exits the internal SCPI vendor mode and returns the USB interface to normal GPIB device passthrough operation. This command is used when the internal SCPI interface was entered via a USB control transfer (see Section 10.1). Alternatively, vendor mode can be exited with the corresponding USB control transfer (wValue=1).

SET:

```
SYST:COMM:USBTMC:EXITV
```

SYSTem:COMMunicate:USBTMC:ADDRess

Selects the current GPIB target address for USBTMC passthrough. Accepts one number (primary address only) or two comma-separated numbers (primary and secondary). This takes effect immediately, allowing the addressed instrument to be switched without reconnecting. Secondary address values 0–30 map to GPIB secondary addresses 96–126.

SET:

```
SYST:COMM:USBTMC:ADDR 5
```

→ *Address GPIB primary address 5*

```
SYST:COMM:USBTMC:ADDR 1,0
```

→ *Primary address 1, secondary address 0 (= GPIB secondary address 96)*

QUERY:

```
SYST:COMM:USBTMC:ADDR?
```

Reply: `5`

→ *Primary address 5 is currently targeted*

Reply: `1,0`

→ *Primary address 1 with secondary address 0 (GPIB secondary address 96) is currently targeted*

SYSTem:COMMunicate:USBTMC:AUTOidRETry

Sets the maximum duration in seconds for which **GPIBEE** retries the *IDN? query after instrument power-on when the AutoID feature is active. **GPIBEE** waits 1 second between each attempt. A value of 0 (default) means the query is attempted exactly once with no retry. Use a non-zero value for instruments that need extra time to initialise before responding to GPIB commands. Maximum: 600 (10 minutes).

SET:

```
SYST:COMM:USBTMC:AUTORET 5
```

→ *Retry *IDN? for up to 5 s (duration, not attempt count)*

QUERY:

```
SYST:COMM:USBTMC:AUTORET?
```

Reply: `0`

→ **IDN? is attempted once with no retry on failure (default)*

Reply: `5`

→ **IDN? is retried for up to 5 seconds; **GPIBEE** makes one attempt per second*

SYSTem:COMMunicate:USBTMC:suppressIDN

Controls whether *IDN? write commands from the VISA host are silently discarded before being forwarded to the GPIB instrument. Some VISA frameworks automatically send *IDN? queries during device discovery, which can cause issues with instruments that do not support this command. When suppression is ON, such queries are blocked at **GPIBEE** and never appear on the GPIB bus.

SET:

```
SYST:COMM:USBTMC:IDN ON
```

→ *Suppress *IDN? writes (instrument never receives them)*

```
SYST:COMM:USBTMC:IDN OFF
```

→ *Allow *IDN? writes through normally (default)*

QUERY:

```
SYST:COMM:USBTMC:IDN?
```

Reply: `OFF`

→ **IDN? writes are forwarded to the GPIB instrument without filtering (default)*

Reply: `ON`

→ **IDN? writes from the VISA host are suppressed; the instrument on the GPIB bus never receives them*

> **Note:** The preferred approach is to disable automatic *IDN? discovery in the VISA software configuration (Keysight VISA: see instrument discovery settings; NI-VISA: disable auto-detection in NI-MAX options) rather than suppressing at **GPIBEE**.

SYSTem:COMMunicate:USBTMC:SCANSTARTaddr

Defines the first primary address checked during the USBTMC startup scan. **GPIBEE** always scans the full primary and secondary GPIB address range; this setting only controls the starting point of the scan. With start address 5, the scan order is 5, 6, ..., 30, 1, 2, 3, 4. Default: 1. Valid range: 0–30. The controller address is always skipped automatically.

SET:

```
SYST:COMM:USBTMC:SCANSTARTaddr 5
```

→ *Begin scan at address 5, wrapping around to 1–4 at the end*

QUERY:

```
SYST:COMM:USBTMC:SCANSTARTaddr?
```

Reply: `1`

→ *Startup scan begins at primary address 1 (default scan order: 1, 2, ..., 30)*

Reply: `5`

→ *Startup scan begins at primary address 5 (scan order: 5, 6, ..., 30, 1, 2, 3, 4)*

10.6 VXI-11 Configuration Commands

SYSTem:COMMunicate:VXI11:PRESENCEcheck

Enables or disables the GPIB device presence check performed during VXI-11 link creation. When enabled, **GPIBEE** probes the GPIB bus to verify the requested device responds before acknowledging the VXI-11 link. Disabling the check allows link creation to succeed immediately without a bus probe, which is useful when opening multiple simultaneous VXI-11 connections under heavy GPIB traffic.

SET:

```
SYST:COMM:VXI11:PRESENCE ON
```

→ *Verify instrument is present during connect*

```
SYST:COMM:VXI11:PRESENCE OFF
```

→ *Skip presence check (faster under high traffic)*

QUERY:

```
SYST:COMM:VXI11:PRESENCE?
```

Reply: `ON`

→ *Presence check is enabled; **GPIBEE** probes the GPIB bus during link creation to confirm the instrument is present*

Reply: `OFF`

→ *Presence check is disabled; link creation is immediate regardless of whether the instrument is on the bus*

SYSTem:COMMunicate:VXI11:MINimumIOTimeOut

Sets the minimum I/O timeout in milliseconds enforced for any VXI-11 device I/O operation. If a VISA library requests a timeout shorter than this value, **GPIBEE** clamps it to this minimum. This compensates for VISA implementations (e.g. PyVISA-py) that may set unreasonably short timeouts. Avoid setting this value too high, as it adds latency when a genuine timeout occurs.

SET:

```
SYST:COMM:VXI11:MINIOTO 10
```

→ *Guarantee at least 10 ms for every I/O operation*

QUERY:

```
SYST:COMM:VXI11:MINIOTO?
```

Reply: `10`

→ *All VXI-11 I/O operations are guaranteed at least 10 ms; shorter timeouts requested by VISA are silently extended to this value*

SYSTem:COMMunicate:VXI11:MAXimumIOTimeOut

Sets the maximum I/O timeout in milliseconds allowed for any VXI-11 device I/O operation. The VXI-11 specification permits 32-bit timeout values, theoretically allowing timeouts up to ~71,000 minutes. This setting caps any incoming timeout to the specified limit, preventing accidentally very long waits if a VISA application specifies an unreasonably large value.

SET:

```
SYST:COMM:VXI11:MAXIOTO 60000
```

→ *Cap all I/O timeouts at 60 s (60,000 ms)*

QUERY:

```
SYST:COMM:VXI11:MAXIOTO?
```

Reply: `60000`

→ *All VXI-11 I/O timeouts are capped at 60,000 ms (60 s); higher values requested by VISA are silently clamped to this limit*

SYSTem:COMMunicate:VXI11:SRQAUTOpoll

Enables or disables automatic Status Byte polling after a GPIB Service Request (SRQ). When an SRQ occurs, **GPIBEE** cannot by itself tell which device on the bus raised it, because the SRQ line used to signal a service request is shared among all GPIB devices on the bus. With this setting OFF (default), the SRQ is signaled to every enabled VXI-11 interrupt channel, so a client with multiple links open must serial-poll each device itself to find the source. With this setting ON, **GPIBEE** performs the Status Byte read on the device's behalf to determine which one raised the SRQ, and signals the interrupt only to the VXI-11 link for that device.

SET:

```
SYST:COMM:VXI11:SRQAUTO ON
```

→ *Identify the SRQ source automatically and notify only its link*

```
SYST:COMM:VXI11:SRQAUTO OFF
```

→ *Signal every enabled interrupt channel on SRQ (default)*

QUERY:

```
SYST:COMM:VXI11:SRQAUTO?
```

Reply: `ON`

→ *Automatic Status Byte polling is enabled; **GPIBEE** reads the Status Byte after an SRQ and signals only the interrupt channel of the device that raised it*

Reply: `OFF`

→ *Automatic Status Byte polling is disabled; an SRQ is signaled to all enabled interrupt channels, regardless of which device raised it*

10.7 GPIB Commands

SYSTem:COMMunicate:GPIB:PRESENT?

Checks whether a GPIB device is present at the specified address by attempting to address it on the bus. Accepts a primary address, or a primary and secondary address separated by a comma. This is a query-only command; the question mark is part of the command name.

QUERY:

```
SYST:COMM:GPIB:PRESENT? 5
```

Reply: `1`
→ *A GPIB device acknowledged at primary address 5 — instrument is present on the bus*

Reply: `0`
→ *No GPIB device detected at primary address 5*

```
SYST:COMM:GPIB:PRESENT? 5,3
```

Reply: `1`
→ *A GPIB device acknowledged at primary address 5, secondary address 3 (GPIB secondary address 99)*

SYSTem:COMMunicate:GPIB:SCAN?

Scans the GPIB bus for all connected devices and returns a comma-separated list of their addresses. An optional parameter controls whether secondary addresses are included in the scan. Secondary address entries are formatted as "primary:secondary". This is a query-only command — ensure that your VISA timeout is long enough to accommodate the scan duration before issuing it.

QUERY:

```
SYST:COMM:GPIB:SCAN?
```

Reply: `4,22,28`
→ *Three devices found at primary addresses 4, 22, and 28 (primary-only scan, ~120 ms)*

```
SYST:COMM:GPIB:SCAN? 1
```

Reply: `"1:0","1:1",22`
→ *Two secondary-addressed devices (primary 1/secondary 0 and primary 1/secondary 1) plus a device at primary address 22 (full scan, ~2.5 s)*

> **Note:** Primary-only scan (parameter 0 or omitted) takes approximately 120 ms. Full scan including secondary addresses (parameter 1) takes approximately 2.5 seconds. Set your VISA read timeout accordingly — the reply is only sent after the scan is complete.

SYSTem:COMMunicate:GPIB:T1TIME

Sets the T1 talker delay in nanoseconds. T1 is the settling time between **GPIB**EE placing a data byte on the DIO1–DIO8 lines and asserting the DAV (Data Valid) line low to signal that the data is stable. The setting is stored in non-volatile memory and also applied immediately without requiring a reboot. Valid range: 350–2000 ns; out-of-range values are clamped to the nearest limit.

SET:

```
SYST:COMM:GPIB:T1TIME 350
```

→ *Minimum T1 delay — highest possible data rate*

```
SYST:COMM:GPIB:T1TIME 1100
```

→ *Typical value for longer cable runs*

```
SYST:COMM:GPIB:T1TIME 2000
```

→ *Maximum T1 delay — maximum compatibility with old instruments*

QUERY:

```
SYST:COMM:GPIB:T1TIME?
```

Reply: `500`

→ *T1 talker delay is currently 500 ns*

> **Note:** Increase T1 if data transfer errors occur with older instruments or long GPIB cable runs. Higher T1 values reduce maximum GPIB bus throughput. The factory default is 500 ns, which is a safe, universally compatible value suitable for most instruments and cable configurations.

SYSTem:COMMunicate:GPIB:CTRLADdress

Sets the primary GPIB address used by **GPIBEE** as the bus controller. The default controller address is 0. Changing it frees address 0 for use by an instrument. The new address takes effect immediately without requiring a reboot. Valid range: 0–30.

SET:

```
SYST:COMM:GPIB:CTRLAD 0
```

→ *Default controller address*

```
SYST:COMM:GPIB:CTRLAD 2
```

→ *Move controller to address 2, freeing address 0 for an instrument*

QUERY:

```
SYST:COMM:GPIB:CTRLAD?
```

Reply: `0`

→ ***GPIBEE** is operating as GPIB controller at address 0 (factory default)*

Reply: `22`

→ ***GPIBEE** is operating as controller at address 22; all other primary addresses are available for instruments*

SYSTem:COMMunicate:GPIB:SendCoMmanD

Sends any GPIB command with ATN asserted directly onto the bus, by specifying its decimal command code as a comma-separated list — up to 16 values, each 0–255, sent in the order given (e.g. UNL, UNT, LAD/TAD addresses, PPC/PPE, and so on). If more than 16 are provided, only the first 16 are sent. This is a write-only command — there is no query form. It will seldom be needed, but is provided to allow any low-level GPIB bus command that **GPIBEE** does not otherwise expose directly. Equivalent to the VXI-11.2 device_docmd Send Command (`0x020000`) — see section 8.2 .

> **Note:** Like the VXI-11.2 doCommands (see section 8.2), this bypasses the bus state a VISA implementation keeps track of internally. Driving ATN or re-addressing the bus while another transfer is in progress can disturb that transfer — use this deliberately, and preferably while no other operation is running.

SYSTem:COMMunicate:GPIB:PPOLLENable

Configures a GPIB device to respond during a parallel poll. Internally builds and sends the same kind of IEEE-488.1 command sequence as `SYSTem:COMMunicate:GPIB:SendCoMmanD` above (Unlisten, address the device, Parallel Poll Configure, Parallel Poll Enable). Parameters: primary address (0–30, mandatory), optionally a secondary address (0–30) for devices that need one to be selected, followed by the DIO line (1–8) the device should respond on and its sense (0 or 1 — 1 means the device pulls the line low while requesting service, 0 is inverted). This is a write-only command; there is no query form.

SET:

```
SYST:COMM:GPIB:PPOLLEN 4,1,1
```

→ *Device at primary address 4 (no secondary address) responds on DIO1, sense=1*

```
SYST:COMM:GPIB:PPOLLEN 4,3,1,1
```

→ *Device at primary address 4, secondary address 3, responds on DIO1, sense=1*

Note: Whether a secondary address is included is inferred from how many numbers are given: exactly 3 (primary, DIO, sense) addresses a device without a secondary address; exactly 4 (primary, secondary, DIO, sense) addresses one with a secondary address. Any other parameter count, or an out-of-range value, is silently ignored — no error is reported.

SYSTem:COMMunicate:GPIB:PPOLLDISable

Disables parallel poll response on a GPIB device previously configured with `SYSTem:COMMunicate:GPIB:PPOLLENable` — the device stops driving its DIO line during a parallel poll. Parameters: primary address (0–30, mandatory), optionally a secondary address (0–30). This is a write-only command; there is no query form.

SET:

```
SYST:COMM:GPIB:PPOLLDIS 4
```

→ *Disables parallel poll response on the device at primary address 4 (no secondary address)*

```
SYST:COMM:GPIB:PPOLLDIS 4,3
```

→ *Disables parallel poll response on the device at primary address 4, secondary address 3*

Note: Whether a secondary address is included is inferred from how many numbers are given: 1 (primary only) addresses a device without a secondary address; 2 (primary, secondary) addresses one with a secondary address. Any other parameter count, or an out-of-range value, is silently ignored — no error is reported.

SYSTem:COMMunicate:GPIB:PPOLL?

Executes an IEEE 488.1 parallel poll and returns the result as a single decimal number (0–255). Takes no parameters. This is a query-only command; the question mark is part of the command name. Equivalent to the vendor-specific VXI-11.2 device_docmd Parallel Poll extension — see section 8.2 .

QUERY:

```
SYST:COMM:GPIB:PPOLL?
```

Reply: `136`

→ *Parallel poll result byte 136 (0x88)*

11. ++ Protocol Reference

The ++ protocol is a simple, text-based control protocol that allows access to multiple GPIB instruments through a single interface. It is compatible with the Prologix GPIB adapter command set, making migration from Prologix-based setups straightforward. **GPIBEE** exposes the ++ protocol in two ways:

- **Over UART** , when USB UART++ protocol mode is selected.
- **Over Ethernet** or USB Network mode, on TCP port 1234. Ensure the ++ protocol is enabled (default: on).

Connect using any terminal program (to the virtual COM port or TCP/IP port 1234). Commands sent to **GPIBEE** must be terminated with CR, LF, or CRLF. All commands start with the ++ characters. Any data not starting with ++ is transmitted to the selected GPIB device address.

Data transmitted to the GPIB device must be escaped:

- `+` (ASCII 43) → `ESC +` (byte sequence 27, 43)
- `CR` (ASCII 13) → `ESC CR` (byte sequence 27, 13)
- `LF` (ASCII 10) → `ESC LF` (byte sequence 27, 10)

Data read from the GPIB device is transmitted fully transparent without modification.

Example session:

```
++addr 5    # Select GPIB device at primary address 5
++auto 1    # Enable auto read-after-write
++read      # Manually trigger a GPIB read (when auto is off)
```

Note: For reading, it is recommended to use the eoi parameter if the instrument asserts EOI, or to use a termination character (e.g. ASCII 10 or 13) to ensure a quick response without timeout delays.

11.1 ++ Command Reference

Command	Parameters	Description
`++addr`	[<primary> [<secondary>]]	Set or query target GPIB address. Primary: 0–31 (31 = internal SCPI interface). Secondary: 96–129. Default: primary=1, no secondary.
`++auto`	[<automode>]	Turn on/off automatic read-after-write. 1 = auto-read with EOI, 0 = disabled (default).
`++clr`	None	Execute a selective device clear (SDC) to the currently selected GPIB address.

Command	Parameters	Description
`++eoi`	[<eoi_enabled>]	Enable or disable EOI generation on writes. 1 = enabled (default), 0 = disabled.
`++eos`	[<eos_mode>]	Select termination characters appended on writes. 0 = CR+LF, 1 = CR, 2 = LF, 3 = None (default).
`++eot_enable`	[<eot_enable>]	Enable addition of an ASCII character when reads terminate with EOI. 1 = enable, 0 = disabled (default).
`++eot_char`	[<char>]	Set the ASCII character used for EOT function. Allowed range: 0–255.
`++ifc`	None	Assert Interface Clear (IFC) for a short time.
`++loc`	None	Execute Go To Local (GTL) to the selected GPIB address. Returns the instrument to local/front-panel control.
`++mode`	[<mode>]	Select controller or device mode. 1 = controller (default), 0 = device (currently not supported).
`++read`	[termination]	Read from the current GPIB device. eoi = read until EOI received; 0–255 = read until character with given value.
`++read_tmo_ms`	[timeout_ms]	Set or query the read timeout in milliseconds (inter-byte timeout, not total transfer). Valid range: 1–3000. Default: 500.
`++savecfg`	[<save_enabled>]	Select or query if configuration is automatically saved on changes. 1 = enabled, 0 = disabled (default).
`++spoll`	[<primary> [<secondary>]]	Execute a serial poll to the current device or a specified one.
`++srq`	None	Read the SRQ line state.
`++trg`	[addr1 [addr2 ...]]	Trigger the current device or a group of up to 15 GPIB addresses.
`++ver`	None	Read the firmware version.
`++rst`	None	Reset the device (reboot). The connection must be re-established afterwards.
`++help`	None	Return help information about all commands.

12. Technical Specifications

12.1 Electrical and Communication Specifications

Parameter	Specification
Host Interfaces	USB Type-C with High-Speed 480 Mbit/s and Full-Speed 12 Mbit/s; 10/100 Mbit/s Ethernet
Instrument Interface	IEEE-488 / GPIB 24-pin connector
GPIB Compatibility	IEEE-488.1 / IEEE-488.2 compatible operation
Controller Role	GPIB controller for instrument communication
Max GPIB Instruments	Up to 15 GPIB instruments on one bus
Supported Software	VISA-compatible applications, Python/PyVISA, MATLAB, LabVIEW, NI-VISA, Keysight IO Libraries, Linux GPIB, and custom TCP/serial automation scripts
Power Input	USB bus power (max 150 mA) or PoE IEEE 802.3af Mode A/B (~1.5 W)
Configuration	Embedded webserver, internal SCPI interface, or USB utility
Firmware	Field-upgradeable and actively maintained

12.2 Measured Transfer Performance

The values below are approximate real-world transfer rates measured with actual GPIB measurement equipment. Effective transfer speed depends on instrument response behavior, GPIB handshaking, command framing, protocol overhead, driver implementation, and host software latency.

Mode / Protocol	Approx. Transfer Rate	Notes
GPIB bus burst transfer	~1 MByte/s burst	Short bus-level burst only; sustained application throughput is lower due to protocol overhead and instrument turnaround.
VXI-11 mode	~500 KByte/s	Typical practical throughput for VISA-compatible LAN access over Ethernet or USB Network mode.
++ Protocol mode	~500 KByte/s	Typical practical throughput for command-based ++ protocol over Ethernet TCP or USB UART mode.

Mode / Protocol	Approx. Transfer Rate	Notes
USBTMC mode	~700 KByte/s	Typical practical throughput exposing a selected GPIB instrument as a USBTMC device.
Linux GPIB mode	~700 KByte/s	Typical practical throughput in Linux GPIB setups using USB Linux GPIB mode.

12.3 Software Compatibility

Software / Environment	Windows	Linux	macOS	Notes
Python / PyVISA	Yes	Yes	Yes	Recommended for cross-platform scripting via VXI-11, USBTMC, TCP socket, or vendor VISA backends.
MATLAB Instrument Control Toolbox	Yes	Yes	Yes	Suitable for VISA, TCP/IP, and automated measurement workflows.
LabVIEW	Yes	Limited	Limited	Windows provides the broadest driver and toolkit support.
NI-VISA	Yes	Yes	Limited	Supports TCP/IP and USB workflows.
Keysight IO Libraries Suite	Yes	Yes	Not typical	Useful for VISA, LAN, USB, and instrument discovery workflows.
Linux GPIB tools	No	Yes	No	Use with USB Linux GPIB mode and the xyphro_ugc board type.
Custom C/C++/.NET applications	Yes	Yes	Yes	Via VISA libraries, TCP sockets, or serial/USB communication.

12.4 Physical Characteristics and Environment

Parameter	Specification
Enclosure	Solid aluminum case, compact industrial form factor suitable for laboratory and production environments
Dimensions (without connector)	60 mm × 37 mm × 23 mm
Dimensions (including GPIB connector)	60 mm × 44 mm × 24 mm

Parameter	Specification
Weight	Approximately 160 g
GPIB Connector	IEEE-488 24-pin connector
USB Connector	USB Type-C standard connector
Ethernet Connector	RJ45 standard Ethernet jack
Operating Temperature	0 °C to 55 °C
Operating Relative Humidity	10% to 90%, non-condensing
Storage Temperature	-20 °C to 70 °C
Storage Relative Humidity	5% to 95%, non-condensing
Cooling	Passive cooling, no fan required

12.5 Package Contents

Item	Description
GPIBEE	USB and Ethernet to GPIB adapter
USB cable	2 m USB-A to USB-C cable
Box size	160 mm × 85 mm × 26 mm
Total weight (packaged)	~150 g

Acknowledgements

The development of GPIBee was greatly supported by a group of dedicated beta testers who provided invaluable feedback, bug reports, and real-world testing across a wide range of instruments and environments. Their patience and enthusiasm have been essential in shaping this product into what it is today. A heartfelt thank you to all beta testers — your contributions are truly appreciated.

13. Open Source Acknowledgements

GPIBEE builds on a small number of excellent open-source software components. Their contributions are gratefully acknowledged.

Software	Description and License
scpi-parser	A lightweight SCPI command parsing library by Jan Breuer (github.com/j123b567/scpi-parser). **GPIBEE** uses it to interpret all internal SCPI commands. *License: BSD 2-Clause License. Copyright (c) 2012–2018, Jan Breuer. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the original copyright notice and disclaimer are retained.*
mongoose	A robust and easy-to-integrate networking stack that provides all network functionality, including the embedded web interface (mongoose.ws). *License: Used under an official commercial license from Cesanta.*

All other software modules — including the complete GPIB implementation, the USB stack and class drivers, and the VXI-11 server — are fully custom developments created from scratch for **GPIBEE**.

No AI-generated code: All firmware and software for **GPIBEE** were written by hand. No AI-generated source code was used in the development of this product.

14. Release Information

Version	Changes
V1.6	• New major feature: **GPIBEE Benches** (beta) - build a web interface panel for any instrument in the browser: readouts, lamps, switches, buttons and value entry, stored on the GPIBEE itself. No PC software and no programming. • New web interface page: Connections — lists every active VXI-11 session and the ++ protocol connection (if any), showing client address, byte counters, and (for VXI-11) its open links and interrupt-channel status. Individual connections can be force-closed from this page, which auto-refreshes while it is open. • New feature: vendor-specific GPIB Parallel Poll doCommand extension (device_docmd cmd `0x021000`) — executes a parallel poll on the bus and returns the 1-byte result. See section 8.2. • New internal SCPI command `SYSTem:COMMunicate:GPIB:SendCoMmanD` — write-only, sends up to 16 raw GPIB command bytes (comma-separated, 0–255 each) directly onto the bus. See section 10.7. • New internal SCPI commands `SYSTem:COMMunicate:GPIB:PPOLLENable` and `SYSTem:COMMunicate:GPIB:PPOLLDISable` — write-only, configure or disable a GPIB device's parallel poll response (DIO line and sense, with optional secondary addressing). See section 10.7. • New internal SCPI command `SYSTem:COMMunicate:GPIB:PPOLL?` — query-only, executes a parallel poll and returns the result (0–255). See section 10.7. • The GPIB web interface now allows users to set the maximum read byte count.

Version	Changes
V1.5	• Improved SRQ handling — the VXI-11 interrupt channel is now established at SRQ enablement. • GPIBee did not enumerate on some PCs when USBTMC or Linux GPIB mode was selected. • Much faster SSDP discovery enabled over USB NCM and Ethernet — adapters now appears nearly immediately in the Windows network environment when connected. • Web interface GPIB Scan: When clicking a search result table entry entry the secondary address was sometimes not taken over to the command panel secondary address field. • VXI-11 SRQ: Pyvisa-py interrupt signalling did not work. • Fixed issues of SRQ signalling and SRQ related connection handling which caused issues on NI Visa run on macOS. • New SCPI command and web interface option to enable/disable automatic SRQ source detection, so only the interrupt channel of the device that raised the request is notified — "SYSTem:COMMunicate:VXI11:SRQAUTOpoll" (default off). • Creating a new VXI-11 connection with no lock requested on an already linked connection (VXI-11 rule B.6.6) resulted into a link lock error. • A complete VXI-11 conformance test suite was developed, covering 104 rules of the VXI-11 specification — all of which now pass. Earlier firmware already worked well with the common VISA implementations; **GPIB**EE is now genuinely VXI-11 compliant, with correct error code reporting, rejection of parameters the standard does not allow, and support for multiple links and sessions. VISA implementations use a single link per session, but multiple links now work exactly as the standard intends. • Instance names now require one of the prefixes inst, gpib or hpib, followed by 0 — see section 8.2 . This is part of VXI-11 compliance. • The asynchronous VXI-11 channel is now fully functional: VXI-11 operations can be terminated before they complete by issuing an abort call. • Implemented the VXI-11.2 low-level bus commands (device_docmd): Send Command, Bus Status, ATN Control, REN Control, Bus Address and IFC Control. Pass Control is not supported. See section 8.2. • Fixed the handling of fragmented ONC/RPC records in VXI-11. TCP/IP fragmentation was, as in previous versions, unaffected. • In USB network mode the network stack was not serviced while a longer command was executing — for example a read running into a timeout. This caused no malfunction, but was not a clean way to handle it.

Version	Changes
V1.4	• New feature: Linux GPIB USB operation mode (use xyphro_ugc as board_type). • New Linux GPIB feature: Enabled SRQ interrupt transfer functionality. • Enhancement: Removed restriction to private IP address ranges. • Enhancement: Added the ability to specify a gateway. • USBTMC: Added *IDN? write suppression option for non-SCPI-compliant instruments. • USBTMC: AutoID retry counter now acts as a retry interval in seconds (max 600 s). • Feature: Configurable GPIB controller address via web interface and SCPI. • Feature: VXI-11.2 compliant instance naming (gpib0, inst0, hpib0 prefixes). • Bugfix: AutoID retry counter change also affected USB fallback time. • Bugfix: Service request signalling was disabled in previous versions. • Removed debug side-panel from web interface. • Reworked and improved this manual. • Improved GPIB read turnover. • Relaxed GPIB T1 time during scanning and USBTMC AutoID process for better compatibility. • Majorly reworked the documentation • GPIB Panel: The Log is now infinite in length and more consistant in formating and added a "clear log" button. • Added a GPIB line state monitor to GPIB panel. • Fixed VXI-11 SRQ handling. SRQ now supports concurrent sessions and does an automatic serial poll to identify which instrument generated an SRQ and notify the correct link. • Added a GPIB Scan results table with resource name generation, one-click address selection, clipboard copy to the GPIB devices panel. • Made the GPIB Log window also resizeable in horizontal direction. • Added a flashing LED indicator to the GPIB line state monitor panel which indicates whenever the state is updated. • Removed static comments in webinterface and added them as tooltip. • Panels are now flowing into multiple columns when browser window width is increased. • FIX: When Linux GPIB mode was active both LEDs were off.

Version	Changes
V1.3	• Improved GPIB write timing with closed-loop line-state method. • Fixed: GPIBee became unresponsive when creating a new VXI-11 link during an active GPIB transfer. • New SCPI command and web interface option to enable/disable GPIB presence check during link creation. • Fixed: SSDP payload contained generic info instead of device/firmware version. • USB network NCM mode: USB serial number now includes actual GPIBee device serial number. • Fixed: TCPIP data sent from GPIBee contained bogus 14 extra bytes after payload. • Fixed: Timeout handling on VXI-11 writes and reads was incorrect with wrapping. • New feature: LXI identification XML file delivery (/lxi/identification). • Feature: VXI-11 IO timeout limits (min and max). • Feature: V4 program version GETADDR portmapper support. • Fixed: Various VXI-11 lock handling issues. • Fixed: Windows 10 USB CDC NCM enumeration (added BOS descriptor). • Feature: Template string max-length specifiers (e.g. {:5IDN}). • Feature: {IDN0}–{IDN3} USBTMC serial number template keys. • Feature: Fast temporary USB Network mode via single button press. • USBTMC: Added scan start address setting, AutoID retry, and startup auto-disconnect when no device present. • USBTMC: Extended serial number template to 126 characters. • Implemented T1 time setting for compatibility with long cable runs and old instruments.
V1.2	• Integrated a manual into the web interface. • Integrated USB CDC NCM network interface over USB (VXI-11 over USB). • Added USB and network ++ protocol mode. • Resorted settings page items. • Adjusted internal SCPI commands.
V1.1	• Replaced LWIP software stack with Mongoose stack. • Added GPIB address setting to USBTMC operation mode via internal SCPI commands.
V1.0	• First version released for beta testing.

15. Disclaimer and Contact Information

General Disclaimer

The information in this manual is provided "as is" and without any warranties, whether express or implied. We assume no responsibility for errors or omissions, nor for any incidental or consequential damages arising from the use of the product or the information contained herein.

Intended Use

This device is intended for laboratory, industrial, and engineering environments where GPIB-controlled test and measurement equipment is used. It must be operated only by qualified personnel familiar with electrical safety and instrumentation interfaces.

This device is not designed or certified for use in safety-critical applications. Use in life-support, medical, or aviation systems is at the user's own risk and responsibility.

Trademarks

- R&S® is a registered trademark of Rohde & Schwarz GmbH & Co. KG.
- NI™ and National Instruments™ are trademarks of National Instruments Corporation.
- Keysight™ is a trademark of Keysight Technologies, Inc.
- All other trademarks are the property of their respective owners.

Liability Limitation

We are under no circumstances liable for:

- Loss of data
- Loss of production
- Equipment damage caused by incorrect wiring or configuration
- Indirect, incidental, or consequential damages

Liability is limited to the extent permitted by applicable law.

Technical Changes

Specifications, firmware, and documentation may change without notice as part of continuous product improvement. Specifications and performance values may vary depending on configuration, connected instruments, host system, software stack, and operating conditions.

Contact Information

No software is entirely free of defects, and **GPIBEE** is surely no exception. If you encounter crashes, unexpected behavior, or anything that feels off, we genuinely appreciate hearing about it. Your feedback is highly valued and helps to improve the product.

Contact	Information
Product Owner	Gossner Electronics and Embedded Systems GmbH
Support Email	Support@GPIBee.com
Origin	Engineered and made in Germany