VLCB Versatile Layout Control Bus

VLCB Technical Introduction

0.1 Table of Contents

0.2 Document History

Date	Changed by	Summary of changes
15th January 2024	Ian Hogg M.5144	Initial document, split out of MNS specification
27th May 2024	Ian Hogg M.5144	Updated Bus traffic indicators for 2 LEDs to be Mode specific.

0.3 Trademarks

Where CBUS is used within the VLCB documents it should be noted that CBUS® is the registered trademark of Dr. Mike Bolton.

1 Introduction

The VLCB is a series of specifications of building blocks that define a layout control bus for the purposes of controlling a model railway. The complete specification consists of several documents and service specifications. Each service describes and specifies a protocol used to provide a service or capability to the module.

The VLCB Minimum Node Specification (MNS), describes the mandatory functionality that a module must implement to be considered a VLCB compatible module. The other VLCB service specifications describe the optional services that may be implemented to make a useful module.

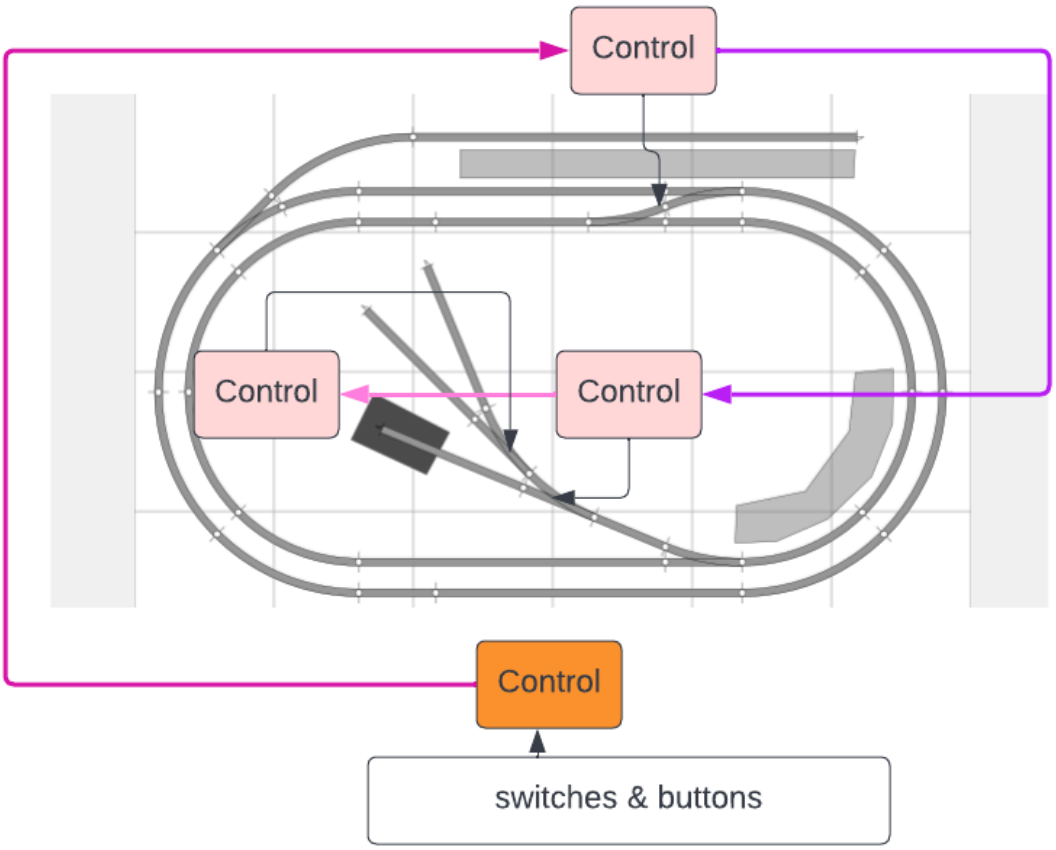
Where services are optional, or where the specifications offer a choice, then the module implementor may decide which are the most appropriate options, and the precise modalities of implementation.

The VLCB specification team has worked to make the implementation as straightforward as possible, and to eliminate ambiguities or areas of doubt. This helps both users and module developers by removing confusion, and by removing the requirement for ad-hoc custom and practice information.

1.1 What is The VLCB

1.1.1 Introduction

A Layout Control Bus (LCB) is a distributed mechanism for controlling model railways.



As can be seen the key features of a layout Control bus are:

1. Control units (points, signals, trains etc.) are distributed around the layout as opposed to being centralised as in the traditional model railway control system. The electronics used to implement these control units are called *modules*.
2. Modules typically provide control functionality such as a signal input or an output, although some modules may provide multiple functions.
3. Some form of communications “bus” is routed to each operational module to enable commands to be sent around the LCB.
4. The commands must be standardised and agreed on by each module so that modules sending commands can be understood by control modules expecting to receive such commands.
5. Typically each control node is a microprocessor based device, i.e. it is “smart“.

Hence VLCB is a conventional LCB system, its primary features are:

- It is communications transport layer agnostic, although most modules are expected to be based on CAN bus, a widely accepted communications system.
- It is designed to be simple to plug together and get operational without much technical knowledge.

- It is extendable and future proof.
- It is technically capable of meeting current model railway control requirements, but also flexible enough to cope with future extensions and technology advancement.
- It covers features to aid diagnostics and discovery of its capabilities such that the modules can not only track and report errors but can in theory work out what is being attached to the layout itself.

1.1.2 VLCB messages

VLCB modules send and receive VLCB messages on the VLCB bus/network. The messages each have an opcode which determines the purpose of the message. Messages may be for the purpose of module instruction commands or module responses or general messages; one of the most important message purposes is the indication of an event.

Please refer to the VLCB Opcode specification for a complete description of the message types and formats.

1.1.3 VLCB Module configuration

VLCB modules have a set of *parameters* which are read-only and may be queried by a management platform to determine the module type, version and some basic information.

In addition VLCB modules may support *node variables* (NVs) which are typically read-write to allow a management platform to configure the operation of the module.

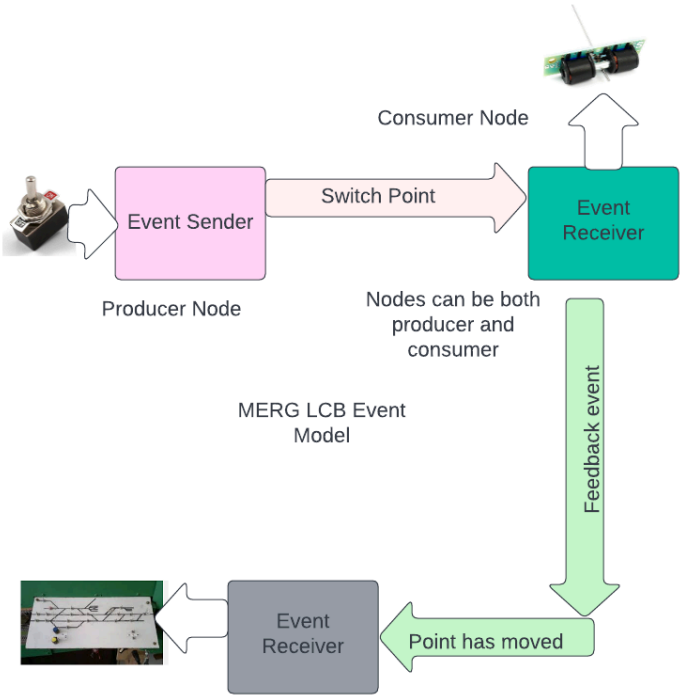
VLCB modules optionally support events which may be configured by setting *event variables* (EVs) within modules.

1.1.4 The VLCB EVENT model

Like many layout control buses, VLCB is primarily (though not exclusively) “event based”.

An Event is a simple message sent by a producer module and received by a consumer module, and this is often called a P/C event, for Producer/Consumer event.

An Event works because the consumer understands what the event means to it (and only it), even though the event may trigger different actions in other modules.



Hence, VLCB modules typically generate events to perform actions or receive events to obtain status back as to the success of these actions.

These events may be predetermined by the module (otherwise known as “default events”) or may be “learned”, by the user teaching the module what event to produce, and also teaching any other module(s) that needs to act on that event.

1.1.5 Why do we need another layout bus specification?

MERG members have developed many layout buses, some oriented at specific roles, some aimed at beginners, some for more sophisticated CMRI. Some of these are supported by kits available from the Kit Locker, whilst others are private MERG members endeavours (RoscoeBus, AT-BUs) etc.

However, the documentation of how to design modules for such LCBs is often sparse, incomplete, or scattered amongst individuals or the MERG forum. VLCB is, we believe, the first attempt to comprehensively document such a layout Control bus. The following goals are therefore indicative:

- The VLCB spec should be comprehensive enough that a designer should find within its specifications all the detail needed to implement a module (node);
- The spec would remove ambiguities that have arisen in other complex specs, which require recourse to other information or knowledgeable people – ie the specification should be completely comprehensive;

- However, the spec should be straight-forward to follow, and be relatively easy to implement.

1.1.6 What is the relationship between CBUS and this document?

VLCB is designed to be compatible with CBUS version revision 4 ver 8j, and **CBUS modules with this, or earlier, versions of CBUS firmware should remain interoperational with VLCB modules.** The team worked hard to maintain this compatibility, rejecting some proposed protocol extensions because they would impact compatibility, and prevent users from using their pre-existing CBUS modules. VLCB extensions are designed to not affect existing CBUS modules since VLCB added opcodes where necessary, but older modules can just ignore these.

The VLCB team cannot guarantee that future versions of CBUS will remain compatible with VLCB, although all reasonable effort will be made to keep compatibility between the two.

A VLCB-compliant module should still operate on the same network as CBUS modules and, excluding the new VLCB functionality, be managed by the FCU.

The VLCB specification mandates some of the information given in the CBUS Developers Guide.

1.2 Why is VLCB better?

The VLCB specification comes after much experience has been gained in MERG from both designers and users. In that regard it attempts to deal with the many issues that have arisen namely:

- Incompatibility between modules
- Need for tight definitions
- Additional diagnostics & other features
- Self testing and user testing features to verify module operation
- Software compliance tests to verify adherence to the specification

The VLCB spec has the benefit of a later arrival, and the authors, being involved in layout buses for years, have the advantage of bringing experience and expertise to ~~bear on~~ this new specification.

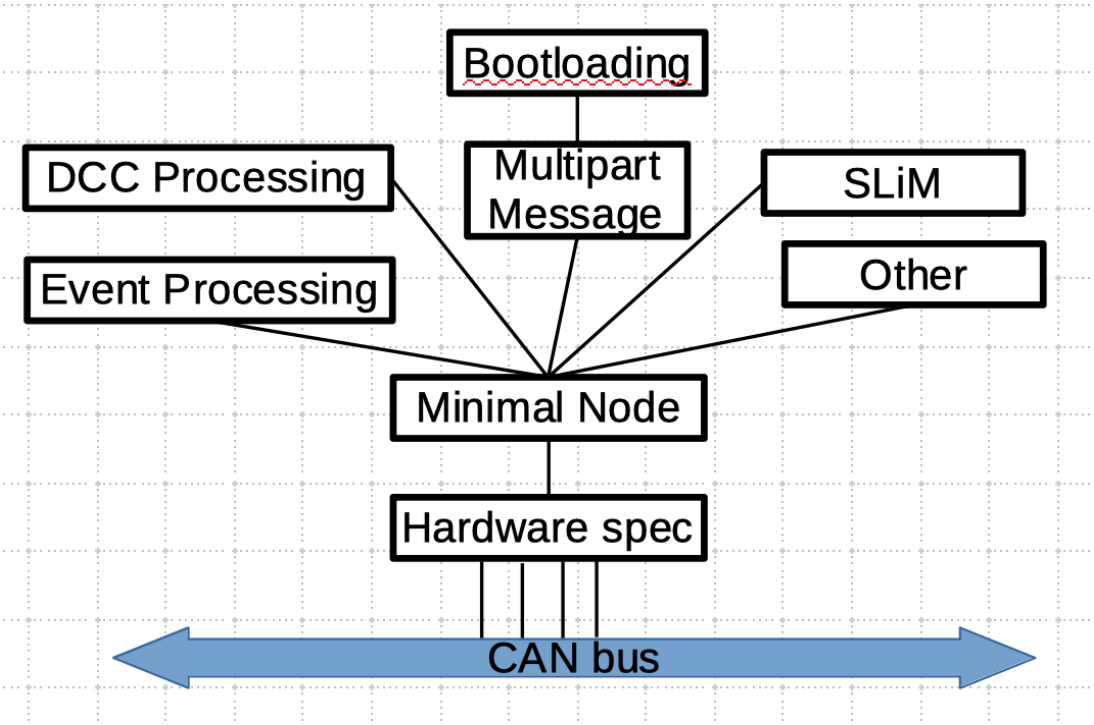
1.3 The VLCB Services Model

The VLCB specification, and its associated compliance software, defines a series of services:

Services shall:

- Have a Service Document describing the service and its protocols.

- Be assigned a Service#
- Use generic error#s, diag# when possible.
- Document error messages, and use GRSP messages to report errors.
- Extend Service Discovery when necessary, using the SD and ESD messages.
- Extend Diagnostics when necessary, using the RDGN and DGN messages.



Partial Hierarchy of Services

Some higher layer-services rely on lower services to deliver base functionality.

A series of C/C++ libraries will be developed for each service (similarly assembly-level code), which can be combined to help develop a complete module. In addition, prototypical code samples will demonstrate how the services are used. These will be made public via software repositories.

The VLCB Minimum Node Specification requires that modules support a service discovery process to allow management systems to determine the capabilities of a module.

1.3.1 Modules and their Services

A VLCB module must implement the MNS Service, since this service implements the basic functionality of a module, including handling requests for its node number, parameters, etc., heartbeat, Service Discovery and diagnostics. It does not implement a useful module, and needs other Services added to it to achieve a fully operational one.

A module must respond with the list of services it supports when responding to a service discovery request. The module must implement all parts of an included service including extensions to the Diag-service and that service's specific response-codes. It is expected that implementations of services would be as libraries which can be included within a module's project.

The table below illustrates some possible modules and their included services.

		Module				
		CANServo	CANPAN	CANINP	Universal	Wifi-CAN
Service	MNS	✓	✓	✓	✓	✓
	NV	✓	✓	✓	✓	✓
	Event Teach	✓	✓	✓	✓	
	Producer		✓	✓	✓	
	Consumer	✓	✓		✓	✓
	CoE				✓	
	Bridge				✓	✓
	CAN	✓	✓	✓	✓	✓
	Wifi					✓

Example Modules vs Services

1.3.2 Service Responsibilities

Services have to provide specific capabilities:

- Respond to RQSD with any service specific data
- Support any service specific modes
- Store and restore their data to non-volatile memory
- Respond to any VLCB messages which are handled by the service
- Provide service specific diagnostic data values

1.4 Mapping of VLCB to lower protocols

The VLCB messages may be transported over a variety of lower protocols such as CAN, BLE, TCP/IP, Serial.

The mapping of the VLCB messages onto these lower protocols ~~can be~~ are/will be described in separate protocol-specific services or in module-specific implementations.

1.5 Module Version Numbering

Module firmware versioning shall follow semantic versioning (See: https://en.wikipedia.org/wiki/Software_versioning#Semantic_versioning) with major, minor, patch subnumbers:

- The Major version number should be updated if there are changes which are not backwards compatible with previous versions. If the Major version number is updated then the Minor number and Patch number are reset to zero.
- The Minor version number is updated when there are additions to the firmware but backwards compatibility is maintained. When the Minor number is updated then the Patch number is reset to zero. Note that the Minor version is represented with a lowercase letter.
- The Patch number is updated for each build release. The Patch number should only be changed for bug fixes and there should be no changes to the VLCB interface so that the services and functionality supported by the module is defined by Module ID + Major version + Minor version. Note that the Patch number is stored as the BETA module parameter. See 16.1.1 Parameter block.

All these numbers are manually updated when making a software change. Only the developer really knows if the changes they've made affect the interface, a major change or a minor change. It is preferred that the module's build system automatically increments the Patch number whenever a change is made.

Software should be stored in an online repository which features distribution support, such as Github.

2 Operation of VLCB

Although VLCB can be transported over a point-to-point connection using serial or TCP it has been designed for operation on a broadcast bus so that a message transmitted by one module is received by all other modules.

When a message is received by a module that module must decide if the message is relevant for itself and needs to be acted upon. There are a number of different mechanisms for this:

- Configuration messages having a node number (NN), in which the NN acts as a target-address, require the receiving-module to check that the NN matches its own node number.
    - Configuration messages which request information from the module will have a response of the data requested or a GRSP indicating an error.
    - Configuration messages which write information to the module must respond with a GRSP response once the write operation is complete and the module is ready to accept more requests.

      For backwards compatibility with CBUS the module may also return a WRACK message or a CMDERR message.
- Event-messages have a NN/EN combination, and this combination must be configured within the module as an event to be acted upon. The NN within the message is not a target/destination address and does not need to match the receiving module's node number.
- Setup-messages have no NN and are only acted upon when the module is in Setup Mode. Therefore only one module shall be in Setup Mode at any time.

It should be noted that whilst Configuration messages use a target address, Event messages are a form of Publisher/Subscriber or Producer/Consumer exchange. This allows a many to many exchange of event messages.

Messages are assigned a priority based upon their opcode. The priority is detailed for each message within the VLCB Opcode specification. When the communication message transport protocol supports prioritisation then the opcode priority should be used as a guide to mapping the priority to the transport protocol specific priority.

2.1 Node Numbers

Although some specialised modules may not require a Node Number most modules must be assigned a network wide unique Node Number. Node Numbers are in the range 1~65535 although some are reserved for modules with fixed node numbers.

A factory fresh module does not have a node number (or assigned the invalid number of 0) and a module must request a number using the RQNN opcode message. The management software then assigns and sets a Node Number with the SNN message. The VLCB MNS specification covers the process and the module modes associated with Node Number assignment.

Once a module has been assigned a unique Node Number the module may then participate in normal VLCB network operation and respond to VLCB messages addressed to it and also participate in producing and consuming events.

Note that the module may also use its Node Number to create default events. If the Node Number of the module is subsequently changed the module designer must consider whether these events are updated.

2.2 Modes

Modules start in a factory fresh condition in *Uninitialised* mode. Modules in Uninitialised mode do not participate in network operations but instead wait for physical user interaction such as the user pressing a button on the module whereupon the module enters *Setup* mode. In Setup mode the module is able to have limited interaction on the VLCB bus and advertises its presence by requesting a Node Number with a RQNN message. The management software may request information about the module before assigning a Node Number with a SNN message. Upon being assigned a Node Number the mode shall enter *Normal* mode and full operations of the module shall be started.

The module may also support other services which support other modes such as Learn mode for event teaching, Event Acknowledgement mode or Bootloading mode. Please refer to service specific documentation for the modes supported by each service.

2.3 Distinguishing VLCB modules from CBUS modules

A configuration tool is likely to need to be able to distinguish between VLCB and CBUS modules. There are a few supported ways to achieve this.

2.3.1 Parameter flags

The 6th bit of the parameter flags will be set for all VLCB modules and should be clear for CBUS modules. The parameter flags are included in the PNN message from a module in response to the QNN request. The flags may also be requested using the RQNPN request.

2.3.2 Presence of MNS Service

VLCB modules must support the RQSD request to query which services are supported by the module. Since all VLCB modules must support the MNS service this may be used to query whether the module is VLCB compliant.

The configuration tool should send a RQSD(NN, 0) request to the module and if the module includes a SD(NN, ?, 0, ?) within its response messages then the module is VLCB compliant.

CBUS modules should ignore the RQSD request or return an error response.

3 SLiM/FLiM

The equivalent of CBUS' FLiM is VLCB Normal mode. Normal mode is a mandatory requirement of VLCB.

SLiM style of operation is not currently supported by VLCB however this may be added as a future optional service.

4 Power up

4.1 First module powerup

The module must initialise its hardware to a working, safe state and prepare its volatile and non-volatile memory ready for operation.

The module may undertake power-up self test operations such as required by optional services or the application. Any tests performed are module dependent.

The module shall enter Uninitialised state and its node number will default to zero after first module power up as zero is not a normal module valid number. Since there may be more than one module on the system in an Uninitialised state, a module can only enter Setup state from Uninitialised state as a result of action by direct operator interaction with the module.

4.2 Module subsequent power up

The module must initialise hardware to a working, safe state and prepare its volatile memory ready for operation.

The module may undertake automatic power-up self test operations such as input/output loopback tests. The tests should be specified in the service specific specifications.

The module should retain its state when powering down, and when next powered up return to that state, whether it be Uninitialised or Normal.

4.3 Saving/Restoring module state

This Minimum Node Service specification requires that modules save some configuration in non-volatile memory so that it is retained upon power down and restored upon power up. Consideration must be given to wear-out of non-volatile memory. Techniques such as moving the addresses which are used to save data or batching writes must be considered. See https://en.wikipedia.org/wiki/Wear_leveling. Hardware to detect power off and trigger the flushing of data to NVM before power fails entirely is another possibility.

If a technique of delayed batch write is used then pending writes must be performed sufficiently frequently to ensure that writes are completed after an operating session is finished and before power off. It is recommended that writes are flushed every few seconds.

5 Module test support

5.1 User initiated tests

A module may provide self-test facilities. Test mode may be entered by some means such as briefly holding down the module's push button during power up. The exact operation during User Test operation is module specific and therefore not covered by VLCB specification. Examples of tests would be to go through a sequence of flashing LEDs or driving outputs to particular states, inputs may be tested by reflecting the input state on any outputs.

5.2 Power on self test

Modules shall **automatically** execute power-up tests every time they are turned on. These tests shall include the power on tests as defined for each service implemented by the module and application specific tests.

.

6 Module User Interface

6.1 Options for displaying mode and errors

Mode	Display			
	Recommendation for 1 LED	**Mandatory for 2 LEDs**	**Recommendation for Terminal**	**Recommendation for Screen**
Uninitialised Mode	Flash 50% 0.5Hz	Green ON, Yellow OFF	Uninitialised Mode	UNT
Setup Mode	Flash 50% 1Hz	Green OFF, Yellow Flash 50% 1Hz	Setup Mode	SET
Normal Mode	ON	Green OFF, Yellow ON	Normal Mode	NOR
Warning for final acknowledgement for factory reset	Flash 50% 2Hz	Both LEDs flash 50% 2Hz alternately	Reset Warning	FRT
Learn Mode	ON	Green OFF, Yellow ON	Learn Mode	LRN
Boot Mode	ON	Green ON, Yellow ON	Bootloading	BOT
Message received	Off for 0.25s then ON	Uninitialised mode: Yellow 0.25s flash & Green ON Normal mode: Green 0.25s flash & yellow ON Setup mode: no change	Message received	*** RX Shown for 0.5s
Message acted upon	Off for 0.5s then ON	Uninitialised mode: Yellow 0.5s flash & Green ON Normal mode: Green 0.5s flash & yellow ON	Message processed	*** RXA Shown for 0.5s

		Setup mode: no change		
Transmit error	Off for 1s then ON	Green OFF, Yellow ON	TX error	*** ETX Shown for 5s
Receive error	Off for 1s then ON	Green OFF, Yellow ON	RX error	*** ERX Shown for 5s
Memory fault	flash 50% 2Hz	Both LEDs flash 50% 2Hz	MEMORY FAULT	*** EMM Shown continuously
Fatal error	flash 50% 2Hz	Both LEDs flash 50% 2Hz synchronised	FATAL ERROR!	*** F!! Shown continuously

Where *** continues to show mode

6.2 User Interface Options for requesting action

A facility to request that a module leaves Uninitialised mode and enters Setup mode so that it can request a node number must be provided. This is provided by a physical input device such as a push button.

Other alternate user interface options are permitted such as ASCII terminal input or touch screen. The exact handling of these alternate input methods is left to the module designer but it is recommended that there is consistency with other modules with similar input devices. It is possible that the standard will be more prescriptive in future versions.

Recommended user interface actions are listed below.

Mode	**Push button**	**Software/ Service**	**Terminal**	**On screen**
Change mode from Uninitialised to Setup.	The push button is held down for at least 4 seconds	MNS	Enter ‘s’ and RETURN	
Change mode from Normal to Setup	The push button is held down for between 1 and 2 seconds	MNS	Enter ‘s’ and RETURN	
Change mode from Normal to Uninitialised	The push button is held down for at least 4 seconds	MNS	Enter ‘u’ and RETURN	

Test Mode	If the application supports a test mode. The push button is held down at power up for between 2 and 6 seconds then released.(*) If held down for more than 30 seconds then the module shall continue normal operation.	Application	Enter ‘t’ and RETURN	
Factory reset	The push button is held down at power up for between 10 and 30 seconds then released and then, within 5 seconds, pressed again for between 2 and 4 seconds.(*)	MNS		
Force bootloader	If the module supports bootloading. The push button is held down at power up for between 0 and 2 seconds then released.(*)	Bootloader		
Reboot module	Press reset button on board.		Enter ‘*’ and RETURN.	

* The exact timings may depend upon whether a bootloader is used with the application firmware and the behaviour of the bootloader when the push button is held down.

7 Hardware requirements

While this section includes a hardware specification, most of this specification Is **not mandatory** however good practice would suggest, where practical, these optional features are included.

7.1 PCB

VLCB does not define any restrictions on PCB size, position of connectors or use of TH/SM components.

Devices **should** be mounted so that they are not prone to physical damage so consider mounting MOSFETS and regulators flush to PCB.

Consideration **must** be given to power dissipation and heat transfer. Consider using heat sinks where applicable.

7.2 12v DC module voltage

The module must be designed to work with 12v nominal DC supplies. It is recommended that some form of device is fitted to protect against overcurrent damage, such as a resettable polymeric PTC device, or simply a current limiting resistor.

It is also recommended that protection against reverse voltage is also provided, diode forward voltage drop and current capacity for the module must be considered.

7.3 CAN connector

Where a module uses the CAN service the following standard connector specified should be used.

Recommended: 3.5 mm pluggable screw terminals:

PIN#	Pin Name	Comments
1	Power IN	+12VDC power input to module
2	CANL	Connects to CAN driver chip, eg MCP2551
3	CANH	Connects to CAN driver chip, eg MCP2551
4	CAN 0V	Often electrical ground

Note: This specification does not exclude additional, **higher** voltage, power feeds for specialised purposes like servos, DCC etc. It also does not prevent alternative provision for **additional** specialised connectors, such as waterproof, anti-vandal etc.

7.4 Daughter boards

Daughter boards may be used, but must not interfere with the main board connectors or prevent buttons being used and/or indicators being seen..

7.5 Mounting Holes

The module must have suitable mounting arrangements, for the intended nature of use. VLCB does not mandate the size or position of mounting holes.

7.6 Hardware Documentation

A compliant hardware **must** come with sufficient instructions and these, as a minimum, must include:

- Pinouts and Connection diagrams,
- Sufficient detail to understand how the module is expected to be used,
- Module maximum current consumption, separate figures if more then one power feed is used,
- Maximum current capability of outputs,
- Voltage range of outputs,
- Input voltage range of inputs,
- Whether outputs are short circuit protected,

- The maximum voltage tolerable on inputs,
- Construction guide,
- Fault finding guide,

If further detailed instructions are available this **must** be referenced.

8 Conformance Test

Developers when designing and implementing a new module shall check that it meets the requirements of VLCB and shall be able to perform a VLCB conformance test. It is the intention that a Conformance Test Kit shall be available to facilitate this process.

The conformance test should be in the form of a “hardware-in-the-loop” test, using the target hardware, since there is a wide range of MCU types now available

As VLCB is neatly segmented into services, the conformance test will state which services it covers, with the intent that the test should expand to cover more services as specifications for new services are developed and completed

As services can evolve over time, the conformance test will tailor its tests to the version of the service that’s returned - skipping the test of that service if it’s an unknown version (the version of the service is expected to reflect the service specification version)

The scope for the test of each service includes the testing of the operation of each opcode supported by that service. Where possible, this would entail verifying the response for each opcode.

8.1 VLCB compliance

8.1.1 Hardware compliance

See section Hardware requirements.

8.1.2 Firmware compliance

The module must pass the VLCB conformance tests.

A compliant module **must** come with sufficient instructions and these, as a minimum, must include:

- Version number,
- Overall functionality,
- Description of the usage of any user interface,
- NV usage, if applicable,
- Event and EV usage, if applicable
- Known bugs and workarounds.

If further detailed instructions are available this **must** be referenced.

8.2 Performance Testing

The module **must** be capable of processing messages at the necessary rate to keep up with what the transport rate is sending to it. Modules with CAN at 125 kbits per second, would need to support a message approximately every 450us.

Hence to maintain such performance the module designer **must consider**:

- **Buffer incoming messages**

  This allows the module to occasionally take longer to process incoming messages. The buffer **must** be big enough to accommodate the expected worst-case delay from the message arriving in the buffer to when the message is acted upon.

- **Buffer outgoing messages**

  Because the module may experience, in high traffic situations, delays in successfully accessing the transport bus, the module **must** maintain output buffers consistent with the longest access delay expected.

  Where a large number of messages need to be sent as a result of a single request then the module must ensure that messages are sent at no more than one every 10ms.

8.2.1 Error reporting

Buffer overflows or message lost conditions are serious errors and the user **must** be informed. This means using the display hardware available either LED or another local display device. In addition if other communications interfaces are available, these interfaces **must** be capable of providing similar error indications.

8.2.2 Diagnostics support

The module must also collect or save, error flags and counters as required to report buffer error conditions via the diagnostics protocol.

This includes serious errors like running out of memory etc. For a full specification see the VLCB diagnostics specification.

8.3 Compatibility logo

Only once the module as a whole (hardware, firmware and documentation) has passed conformance testing may the module be described as VLCB compliant and use the VLCB logo.

An exception to the above is made so that the module’s PCB may use the VLCB logo upon satisfactory completion of the hardware compliance checking.

The version number of the MNS spec that the module is compliant with shall also be included.

9 Configuration Tool support

VLCB modules must be supported by a configuration tool to allow general users to be able to configure and use their system in a user-friendly way.

There are a number of potential configuration tools:

- FCU
- MMC (Module Management Console)
- FCU lite
- Module based web interface

Module developers must ensure that their module is able to be developed by the leading configuration tool at the time of compliance testing and release. It is hoped that support by some of these tools can be provided by a Module Descriptor File such as JSON or JavaScript.

10 Glossary

CAN	Controller Area Network. A standard communications bus originally defined by Bosch. Widely used in cars, industry and other electrically noisy environments.
CBUS	A set of messages for model railway control. The CBUS system was developed over 4 years by Mike Bolton and Gil Fuchs and introduced with specifications and an initial range of kits in 2007. Since then the system has been further developed by many MERG members into a very comprehensive Layout Control Bus.
EN	Event Number
FLiM	Full Layout implementation Mode
MNS	Minimum Node Specification
NN	Node Number
Parameter	Describe the capabilities of a module. Parameters are read-only and set by the module's firmware. Some parameters are dynamic and can change during module operation.
PCB	Printed Circuit Board
SLiM	Simple Layout implementation Mode