VLCB Versatile Layout Control Bus

Minimum Node Service Specification

Service# 1

Version 0.99B

Compatible with CBUS ® 4.0 Rev 8j

What's New:

Add new entries to the top.

- 2023-07-11 IH corrected diagnostics table in section 8.2.4 to align with list in 8.2.2
- 2023-01-05 IH merged Appendix C into section 9
- 2022-12-29 IH Changed service discovery for Service Index and number of services
- 2022-12-15 IH added section on ESD data, removed SQU
- 2022-12-10 DH added a sentence about exiting Setup mode should a Mode-message setting Setup mode be sent to another node.
- 2022-12-05 IH Removed GSTOP
- 2022-12-04 DH Added 1.3.2 Service Responsibilities
- 2022-11-14 IH Moved CAN diagnostics to CAN service spec. Reformatted remaining diagnostics.
- 2022-11-13 Added section on Modules and their Services
- 2022-11-13 New Why? Sections within introduction. Also added to diagnostics section
- 2022-10-27 IH Reordered sections. Removed SD opcode details as that is in the opcode specification.
- 2022-10-26 IH Moved the CAN info to the CAN service document and removed CAN from the document title. Also merged some sections as agreed at meeting 23rd Oct. Moved actions to meeting minutes document so they can span multiple documents.
- 2022-10-10 IH Tidied up some sections after yesterday's meeting. Moved NVs and opcodes to new documents. Moved CBUS comparison to new document.
- 2022-09-30 DPH Added this section "What's new", so others can find updated stuff more easily. Please enter a summary with a date for changes made.

Discussion points

- Compliance testing – which tool / framework.
  Conceptually a user can plug in a module and check that it complies with the VLCB Services.
  Also should be useful with or without the module in firmware development, where the module might be virtual.

- What does a node need to do if its NN is changed? Eg updating other node's events (NN:EN → NN':EN). I think this is better done by a Tool, rather than bus-protocols.
  3.2.6 Normal to Normal

- Do we need or should we have a shutdown (end of day) opcode and process? This is optional but a module's documentation must state if it is supported.

0.1 Table of Contents

0.2 Document History

Date	Changed by	Summary of changes	Service version
22nd December 2022	Ian Hogg M.5144	Initial document	1
12 April 2023	Ian Hogg M.5144	Changed name to VLCB	1
25 April 2023	Ian Hogg M.5144	Updated version number to use Patch number and added more clarity.	1
15 May 2023	Ian Hogg M.5144	Allocated bit 6 of module parameter flags to support for service discovery.	1

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.

This specification, the VLCB Minimum Node Specification (MNS), describes the mandatory and optional services, and the specifications that a module must implement to be considered a VLCB compatible module.

This specification is as prescriptive and definitive as possible, terms such as “**shall**“, “**will"** or “**must"** mean the feature described must be implemented, whereas “**may**“ or “**should"** mean the service is considered important but its inclusion is optional. In many cases implementation means incorporating the complete relevant specification of the service as detailed in a relevant VLCB document.

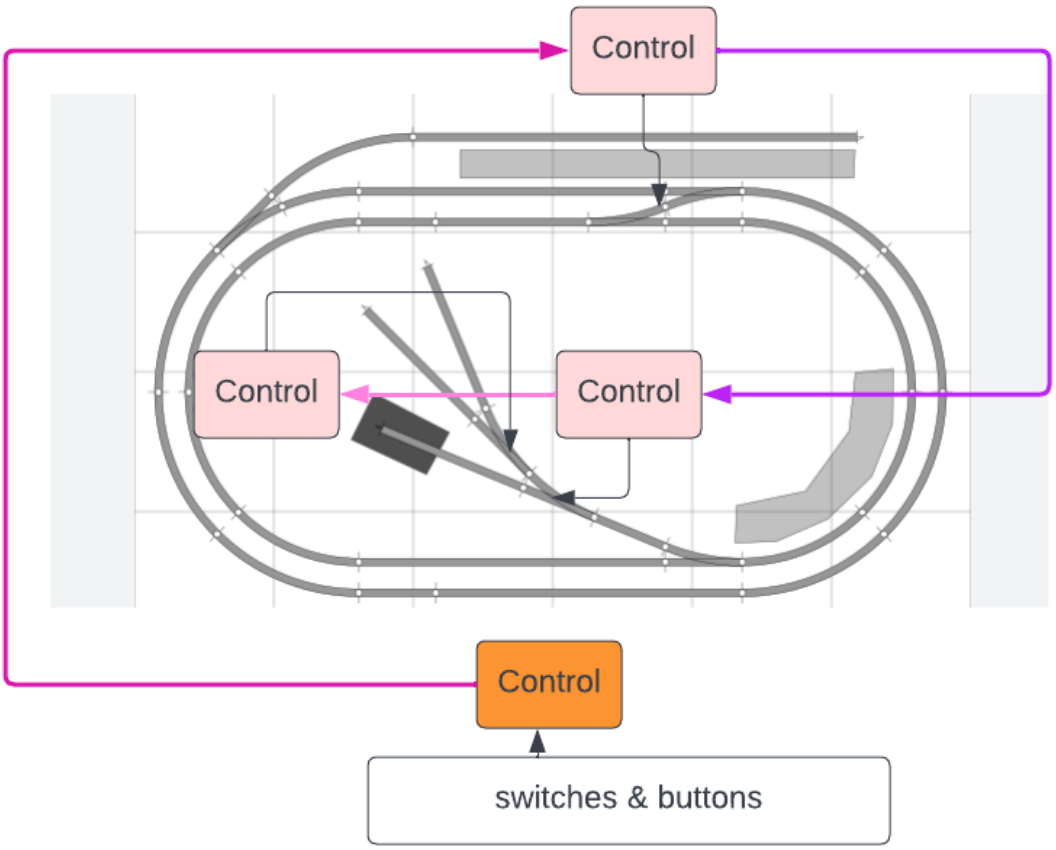
Where services are optional, or where the specifications offer a choice, then the module implementor may decide which are the most appropriate options, and the precise modalities of implementation. Where the MMS specification is precise, then that specific requirement **must** be implemented, to enable the module to conform to the VLCB MNS specifications.

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

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.
2. Some form of communications “bus” is routed to each operational module to enable commands to be sent around the LCB.
3. 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.
4. 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.
- Its 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.

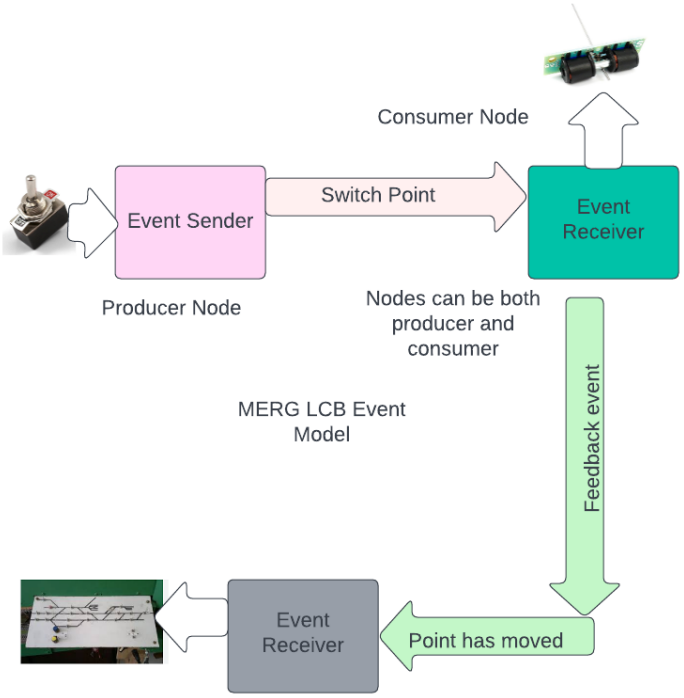
- It covers features to aid diagnostics and discovery of its capabilities such that the modules can not only track and report errors but can in theory work out what is being attached to the layout itself.

1.1.2 The VLCB EVENT model

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

Events are simple messages sent by a producer module which are received by a consumer module , this is often called the P/C event.

Events work because the consumer understands what the event means to it (and only it as an event may trigger different actions in different modules.



Hence VLCB modules typically generate events to perform actions or receive 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 simultaneously teaching every module that needs to act on that event as well.

1.1.3 Why do we need another layout bus specification

MERG many layout buses, some oriented at specific roles, some aimed at beginners, some for more sophisticated CMRI, and 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 a layout Control bus, to 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 requiring recourse to other information or knowledgeable people, ie the specification should be completely comprehensive,
- However the spec should be straightforward to follow and be relatively easy to implement.

1.1.4 What is the relationship between CBUS(r) and this document

VLCB is compatible with CBUS ® to a certain point however it contains supersets and other features that are not present in CBUS ® . That is to say a module designed to be VLCB cannot be guaranteed to be backwards compatible with older CBUS® modules as it may contain additions and features not found in legacy CBUS®.

However in general older CBUS(r) should remain interoperational with VLCB modules but no such compatibility can be guaranteed as VLCB is not responsible for what happens in CBUS(r) today or tomorrow

Hence its not possible to flatly guarantee interworking between CBUS and VLCB, but for legacy modules it is expected that nothing in the VLCB spec prevents then from working together

1.2 Why is VLCB better ?

It is a debatable point, but 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

Hence the VLCB specification is more comprehensive and should allow delivery of modules that work well together, are easier for the user to discover if there are issues and equally support for some advanced features.

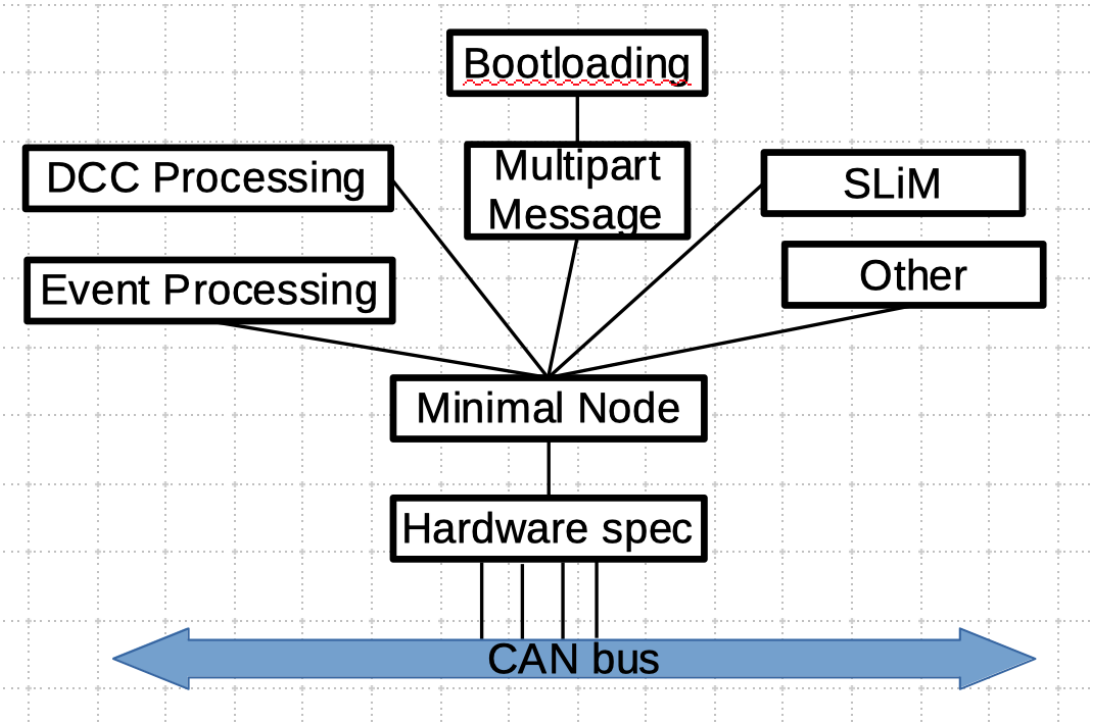
The VLCB spec has the benefit of a late arrival, and the authors, being involved in layout buses for years, have the advantage of bringing experience and expertise to bear on this new specification.

1.3 The VLCB Services Model

The VLCB specification, and its associated compliance software, specifies a series of services that together constitute the whole LCB spec.

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, some services are mandatory , some are optional.

It is hoped that 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. It is most likely that these will be made public via repositories, such as Github.

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 MNS. It needs to be a full participant on VLCB, and therefore must respond to requests for its node number, parameters, etc. The Minimal Node is not very useful, as it cannot do anything. Therefore, all useful nodes will include some of the other optional services.

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 vs Services						
Service	Profile					
	CANServo	CANPAN	CANINP	CanCan	…	Wifi-CAN
MNS	✓	✓	✓	✓	✓	✓
NV	✓	✓	✓	✓	✓	✓
Events	✓	✓	✓	✓		
Producer		✓	✓	✓		
Consumer	✓	✓		✓		✓
Bridge				✓		✓
CAN	✓	✓	✓	✓		✓

Module vs Services						
Service	Profile					
	CANServo	CANPAN	CANINP	CanCan	…	Wifi-CAN
CAN2				✓[1]		
Wifi						✓

1.3.2 Service Responsibilities

Services have to provide specific capabilities:

- Respond to RQSD with SD and ESD
  Service-dependent data is to be provided via these responses.
- Store and restore their data to non-volatile memory

1.4 CBUS™ compatibility

VLCB is designed to be compatible with CBUS(™) modules. VLCB covers a tighter definition of CBUS opcodes along with additional functionality, but it is intended to be backwards compatible with CBUS as far as possible. The specifications documents will indicate the version of CBUS(™) that it supports.

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

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

This specification is compatible with CBUS revision 4 ver 8j. Compatibility with future versions is not guaranteed.

1.5 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 is outside the scope of VLCB Minimum Node Specification and can be described in protocol specific services or in module specific implementations.

[1] It isn't possible to have multiple instances of a service so it would be necessary to have a CAN2 service for the second CAN interface or a dual-CAN service.

1.6 Module Version Numbering

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

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 character.

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 the 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 for the module's build system to automatically increment 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.
- 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. A single module should 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.

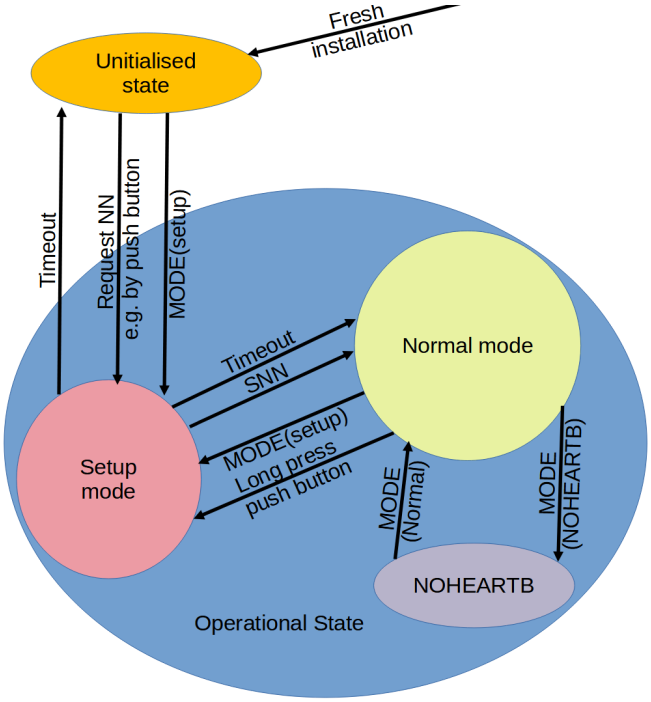
Messages are assigned a priority based upon their opcode. The priority is detailed for each message within the VLCB Opcode specification. When the communication message transport protocol supports prioritisation then the opcode priority should be used as a guide to mapping the priority to the transport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

3 Module States and Modes

An MNS module must implement the following features to support module initialisation and to allow it to participate in a VLCB.
A module will be in one of the following Modes:

- Uninitialized state – it has no node number and does not respond to VLCB messages.
- Operational state - responds to VLCB messages according to mode:
    - Setup mode – CANID has been set, awaiting NN, able to respond to commands with no NN addressing. Only a single node on the network must be in setup mode at any time.
    - Normal mode – it has a Node Number and is ready to FULLY participate with the LCB.
    - NOHEARTB mode - it has a Node Number and participates with the LCB but does not transmit the HEARTB heartbeat message on a regular basis.
    - A module may support other modes as required by included functionality such as learn mode required by event configuration feature and ENACK for event acknowledge.

The mode shall be stored in non-volatile memory so that it can be restored upon power up. Modes Uninitialised, Normal and NOHEARTB, ENACK shall be persisted. In particular Setup mode should not be persisted. Modes supported by other services are generally not persisted but it is under the governance of the service.



The diagram shows the States and Modes concerned with Minimum Node Specification, other parts of the VLCB specification are likely to introduce additional Modes such as Learn, Event acknowledgement and Bootloading.

The MODE message is used to set a node's active mode, in addition to legacy messages.

3.1 Uninitialised State

After preparing the module's hardware for operation a newly installed module will start in the Uninitialised state.

The module must indicate that it is in the Uninitialised state by showing a green LED or other means. Please see 16 Appendix B - Module User Interface for possible user interface options.

Note Uninitialised state is conceptually the same as the CBUS SLiM state.

Messages received: MODE.
Messages possibly sent: RQNN.

3.2 Operational State

Operational state is for when the module is accepting VLCB messages. There are a number of sub-modes associated with Operational state.

3.2.1 Setup Mode

The module is in the process of being assigned a node number. Only a single module on the network can be in Setup mode at a time. Therefore, should a module in Setup mode detect a Mode-message instructing another module to enter Setup-Mode, it must immediately exit Setup mode.

The module will respond to Setup opcodes, which do not require a node number.

The module must indicate that it is in Setup mode by flashing the yellow LED at 1Hz or other means. Please see 16.1 Options for displaying state and errors for possible user interface options.

The module does not have a node number whilst in Setup mode. The module should internally assign a value of zero for its node number. Any previously held node number should be released as the module enters Setup mode using NNREL.

Messages received: SNN.
Messages possibly sent: None.

3.2.2 Normal Mode

As the name suggests this is the normal operating mode of the module. The module has a node number and will respond to configuration commands containing a NN matching its own node number.

The module will also consume and produce events, if these are supported.

The module must indicate that it is in the Normal Mode by showing a yellow LED or other means. Please see 16.2 Status Indications for possible user interface options

Note: Normal Mode is conceptually the same as the CBUS FLiM Mode.

Messages received: Most.
Messages possibly sent: Most.

3.2.3 NOHEARTB Mode

The no-heartbeat mode operates in the same way as Normal mode but with the exception that Heartbeat messages are not sent. Ie sending heartbeats is the default behaviour. See Section 8.1.1 Heartbeat message for a description of the regular Heartbeat message.

3.2.4 Learn Mode

VLCB services may introduce additional modes. For example the event service introduces Learn mode. Learn Mode is not part of MNS but included here to aid understanding.

Learn mode is required when configuring a module's Event Variables. A maximum of one module on the network should be in Learn mode at any time.

Whilst in the Learn Mode the module is able to respond to event teaching requests.

Messages received: Learn-messages + MODE.
Messages possibly Learn-messages - see Learn Service Doc.

3.3 State transitions

3.3.1 Fresh Node to Uninitialised

When powered up for the first time the module shall automatically enter an Uninitialised State.

The module firmware must execute its start-up initialisation, in order to put the module into a default configuration.

3.3.2 Uninitialised to Setup

The module must be able to be instructed to transition to Setup Mode. This can be done by holding the module's push button for at least 4 seconds or by other means such as a user interface command. Please 16.2 Status Indications for possible user interface options.

Alternatively the MODE opcode can be used to change from Uninitialsed state to Setup mode.

If the module uses a CAN interface then the module shall undertake a self-enumeration to obtain a CANID.

When entering Setup Mode the module must transmit a RQNN to request a node number from the configuration software platform. In addition the node will implement a 30 secs timer, after expiration the node will terminate the node setting process, **and** ignore any late SNN or if the SNN never arrives. The node will exit the node number setting process and revert to Uninitialised state.

3.3.3 Setup Mode to Normal Mode

When the module receives a SNN CBUS message, the module shall enter Normal Mode and set its node number to the NN contained within the SNN message.

The module must then respond by transmitting a NNACK containing its new node number.

3.3.4 Normal Mode to Setup Mode

When the user holds the push button for more than 8 seconds the module shall transition from Normal Mode to Setup Mode.

The MODE opcode can also be used to transition a module from Normal Mode to Setup mode. The module will respond with a GRSP(ok) message to indicate that the mode change has been implemented.

When entering Setup Mode the module must send a RQNN message to request a new node number, the NN should be set to the module's current node number.

When entering Setup mode the module shall start a 30 seconds timer so that if a SNN is not received within 30 seconds the module shall return to its previous mode. If the module had a node number then it shall reclaim the node number by sending a NNACK message and the module shall resume using the previous node number.

3.3.5 Normal Mode to NOHEARTB Mode

The Mode message is used to put the module into NOHEARTB Mode, all Normal processes will continue, including any heartbeat-events, except for no Heartbeat messages being generated.

3.3.6 NOHEARTB Mode to Normal Mode

Using the MODE message to set Normal Mode will reactivate Heartbeat messages.

3.3.7 Normal Mode to Learn Mode

The MODE message is used to put a module into Learn Mode from Normal Mode. In addition, the legacy NNLRN message may be used to do the same.

3.3.8 Learn Mode to Normal Mode

The MODE message is used to return a module to Normal Mode from Learn Mode. In addition, the legacy NNULN message may be used to do the same.

A MODE or NNLRN message directed to a different module will also revert a module from Learn Mode to Normal Mode, to ensure only one module is in Learn Mode at any one time. .

3.4 Node Number

A Node Number shall be required for MNS modules, The table shows the allowed range of numbers that can be freely used

Node Number Ranges	
NN Range	**Use**
0	Reserved for special use by system, E.g. NN==0 is uninitialized.
1-65279 (0x0001-0xFEFF)	Available for user use
65280-65535 (0xFF00-0xFFFF)	Reserved for devices with fixed node numbers

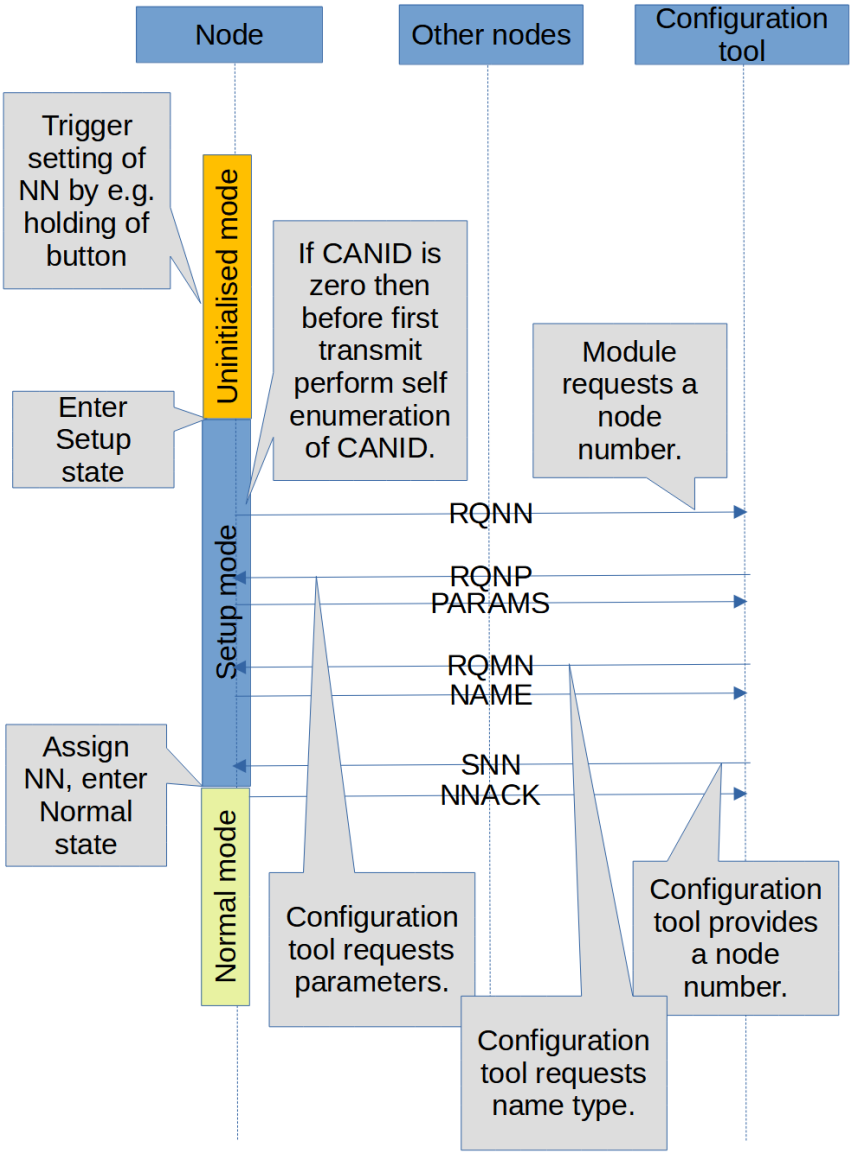
Node number Range

3.4.1 Duplicate Node Number

The VLCB does not under normal circumstances permit multiple nodes to have the same Node Number. Nodes should avoid this situation, and must implement a detection and reporting (via diagnostics) facility when it sees a response message with its own NodeNumber that it did not send.

3.4.2 Node Number Assignment

A node must facilitate the assignment of a node number by one or more mandatory methods, namely PushButton, local communication interface or software setup.



Note that, only one node **will** execute a RQNN and SNN, NNREL, NNACK sequence at one time. At the end of this sequence, Parameter 8 Bit 3 shall be set to indicate “Normal” (FLiM) mode.

4 SLiM/FLiM

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

The MMS specification does not support SLiM style of operation. This may be added as a future service.

5 Power up

5.1 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 CAN loopback tests. 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, the module will only respond to MODE opcode over the bus in order to instruct it to go to setup mode but may also react to other forms of user instruction to instruct it to enter Setup mode and obtain a node number.

5.2 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. MNS does not specify any automatic power-up tests.

When a module that is in Operational state is powered down it must restore to Operational state, Normal mode when next powered back up. A module which was in Uninitialised state must revert to Uninitialised state when powered back up.

5.3 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.

5.4 Module power up User Test

A module may provide self-test facilities. Test mode is entered by holding down the push button during power-up or by other means. 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.

The module must be powered up without holding the push button to return to standard VLCB operation.

6 OPCODE support

The list of all MNS mandatory opcodes is recorded here as commands, their parameters, the required action and the required response opcode and action. For the full description, parameter formats and possible errors please see the VLCB MNS opcode specification.

List of Opcodes that MNS must support:

Request to Module	Module's Response	Use/meaning
	RQNN	Request NN (node number). Used to obtain a (new) NN. Enter setup mode.
SNN	NNACK	Set a new NN for the module in setup mode. Exit setup mode
	NNREL	Release previous node number
QNN	PNN	Query NNs
RQNP	PARAMS	Request first 8 parameters
RQMN	NAME	Request module type name
RQNPN	PARAN	Request a single parameter
	CMDERR	Error response
	GRSP	Error response
NNRSM		Revert module to manufacturer's defaults
RDGN	DGN	Request diagnostics
	HEARTB	Heartbeat
RQSD	SD	Request service discovery
	ESD	Request service discovery
MODE	GRSP	Sets a node's Mode
SQU		
NNRST		Reset module
NNRSM		Restore module to manufacturer's settings

7 Service Discovery support

The module **will** implement the VLCB Service Discovery protocol.

A service is any feature which a specific module implements **irrespective** of whether that feature is mandatory or not. The Service Discovery lists the current feature list and its feature identification, called the Service#.

7.1 Service Discovery Request

The service Discovery is a command response system, carried out at normal CAN priority.

On receipt of opcode RQSD (request service discovery) directed at a specific node and with a service equal to zero, that node will respond with a SD message indicating the number of services followed by a SD message for each service supported by the module.

The node will respond within 2 seconds with SD for ServiceIndex zero (Minimum Node Specification). All other supported services within 5 seconds.

On receipt of a message with opcode RQSD directed at a specific node and with a non-zero ServiceIndex, that node will respond with a ESD message for that specific service if supported or GRSP(Invalid service) if the node does not support that requested service.

7.2 Service Discovery Response

There will be one SD response for each service supported by the module. The format and data of the command and response is detailed in the VLCB opcode specification.

The version of the service definition implemented by the module is also returned. This is not the version of the software implementation.

7.3 ESD Extended Discovery

In addition the module will implement the Service Discovery extended information option. This uses RQSD with a Service#, the node then responds with ESD which contains additional specific information about that feature.

The service must document the meaning of the data bytes within the ESD message.

8 Diagnostics support

A MMS module **will** implement the Type 1 and Type 2 VLCB diagnostics. Each Service can specify its own set of diagnostics, please consult individual Service specifications for the details.

Collecting diagnostics information

The module **must** be capable of collecting and in some cases non volatile storage of the necessary data to fulfil its obligations to report such diagnostics as are outlined in the this Diagnostics Specification and the Service specific specifications

8.1 Heartbeat

8.1.1 Heartbeat message

The module shall transmit a heartbeat message with the HEARTB opcode. This message shall be broadcast every 5 secs at the lowest priority. This heartbeat message will contain the node number, a sequence counter and a number of global flags indicating module status.

The payload will be

- NodeNumberHi , The high byte of the designated Node.
- NodeNumberLo, The low byte of the designated Node.
- SeqenceCnt, This is a count from 0 incrementing on each message transmitted and wrapping around to zero, It facilitates detection of missing frames.
- StatusByte1: This is a binary representation of the module error states as outlined in Section 8.3 Module Status, 0x00 Shall always represent error free, normal operation.
- StatusByte2: Reserved for future expansion, set to 0x00.

A module will NOT transmit any HEARTB message until it has been assigned a NodeNumber by any specified means , as soon thereafter as practical the HEARTB low priority diagnostic should begin.

The diagnostic message is normally enabled but may be disabled using NOHEARTB mode. Whilst in NOHEARTB mode the module otherwise operates as per Normal mode. Heartbeat messages can be re-enabled by returning the module to Normal mode.

8.1.2 Heatbeat Event

If the module supports the Produced event service then the module shall also support the ability to configure transmission of a heartbeat event. If the module supports configurable produced events then the event number (EN) may be configured otherwise the default of 0xFFFF shall be used.

This event shall be broadcast whenever the module status changes at the lowest priority.

The format of this heartbeat is that of a standard long event pair ACON/ ACOF.

An ACOF message is used to indicate normal operation of the module, a ACON message is used to indicate an abnormal operation.

The heartbeat event is optionally sent in addition to the mandatory HEARTB message.

8.2 MNS Service Specific Diagnostics

A module **must** implement the OPCODE command RDGN and its response DGN opcode. On receipt of the command the module will respond as quickly as practical, with the response packets. The format of the data etc is detailed in the Diagnostics Specification.

Type 2 Command responses should be sent at normal priority, and cannot be disabled.

The following Diagnostics are available:

8.2.1 RQDN Request Diagnostics

This RQDN message may be issued to the module by any node. RQDN takes the following parameters:

- Node NumberHi ; The high byte of the designated node thats is to respond,
- NodeNumberLo : The low byte of the designated node that is to respond,
- Service# : The service identifier,
- DiagnosticCode: see below.

The diagnostic data associated with the specified service and DiagnosticCode is returned in a DGN message. The target node will respond as soon as practical (within 15 secs) with DGN messages.

If DiagnosticCode is specified as zero then a sequence of messages for each DiagnosticCode associated with the service is returned. If the Service# is specified as zero then a sequence of DGM messages is produced for each DiagnosticCode for each Service. This may generate a large number of messages, these must be sent at a rate so that other modules are not stressed. A suggested inter-message time of 10 ms is suggested.

8.2.2 DiagnosticCode for MNS

The DiagnosticCode will indicate the type of information requested. The following DiagnosticCodes are defined by MNS. Other services may define their own diagnostics.

0x00: return a series of DGN messages for each supported DiagnosticCode data.

0x01: return module status code.

0x02: return uptime upper word.

0x03: return uptime lower word.

0x04: return memory error count.

0x05: return number of Node Number changes.

0x06: return number of received messages acted upon.

8.2.3 Diagnostics Data

On receipt the Node will respond with DGN and valid Data or GRSP (see opcodes).

The DGN response message contains two data bytes: DiagnosticVal1 and DiagnosticVal2.

Two bytes of data are always returned, 0x00 being used for no data.

The DiagnosticVal bytes may be combined to form a 16 bit counter value with DiagnosticVal1 being the higher byte and DiagnosticVal2 the lower byte.

8.2.4 MNS Diagnostics payload data return

The DiagnosticByte1 and DianosticsByte2 are defined for the MNS service and each DiagnosticCode thus:

DiagnosticCode	DiagnosticByte1	DiagnosticByte2	Description
0x01	STATUS	0x00	Facsimile of the global module status byte, see section 8.3 Module Status
0x02	UPTIME Upper Hi	UPTIME Upper Lo	Upper Word of 32bit uptime measured in seconds.
0x03	UPTIME Lower Hi	UPTIME Lower Lo	Lower Word of 32bit uptime measured in seconds.
0x04	MEMFLT	0x00	Memory fault indicator. 0x00: no fault , 0x01: Flash write fault, 0x02: Flash read Fault , 0x04: EEPROM read fault , 0x08: EEPROM Write fault , 0x10: Stack overflow, 0x20: RAM fault , 0x80: general unspecified memory error. Indicator values may be OR'ed together.

0x05	NNCNT Hi	NNCNT Lo	Number of nodeID changes. The node number was assigned or reassigned since power up by any node assignment method.
0x06	MESSACTED Hi	MESSACTED Lo	Count of messages processed by the module. The principle is this count is messages the module “acted on” in some way.

8.3 Module Status

The Heartbeat message and DiagnosticCode 1 contain a module status byte providing an overall view of the health of the module. This is represented by a count of recent errors.

The module status shall be calculated by having a 8bit counter which is incremented whenever an error diagnostic is incremented by any of the module’s services. This increment should max at 255 and not rollover to 0. The counter shall be decremented if it is above 0 every 5 seconds.

9 MNS Service Specific GRSP response codes

Codes 1~12 match those of CMDERR. Additional error codes have been added with error codes beyond those used with CMDERR, starting at 0xFF, and descending.

GRSP Generic Response		
Basic responses, matching CMDER		
Result Code	Description	Comment
0	ok	
1	Command Not Supported.	NOT_IMPLEMENTED
2	Not In Learn Mode.	
3	Not in Setup Mode.	
4	Too Many Events.	
5	No Event.	
6	Invalid Event variable index.	
7	Invalid Event.	
8	Reserved.	
9	Invalid Parameter Index.	
10	Invalid Node Variable Index.	
11	Invalid Event Variable Value.	
12	Invalid Node Variable Value.	

Additional MNS specific GRSP codes:

Code	**Service #**	**Error**
0	1	OK
252	1	Invalid Service
253	1	Invalid diagnostic code.
254	1	Unknown non-volatile memory type (NVM).

10 Testing support

10.1 Power on self test

Modules shall **automatically** execute power-up tests every time they are turned on. Please see sections 5.1 First module powerup and 5.2 Module subsequent power up regarding the power-up automatic self test facilities.

10.2 User Test

Module specific user test facilities should be provided to allow a module to be tested **by users** for functionality on the construction workbench or on the layout.

To enter User Test operation it is recommended that holding down the push button during power-up or by other means will enter Test operation. See section 5.3 Module power up User Test.

10.3 Conformance Test

Developers when designing and implementing a new module shall check that it meets the requirements of VLCB 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.

To be added. What is the System under test? Is it the hardware and firmware combined or just the firmware?

Do we need to specify test points on the PCB so that the test environment can programmatically apply voltages to the SUT? I.e. to programmatically press the Push button and monitor the LEDs?

(DPH) Initial example process.
Note there are sw packages that make this standardized.

- Plug in module
- Expect: RTR with responses
- Expect: RQNN(oldNN), record oldNN
- Send SNN(newNN), expect NNACK(newNN)
- Send RSNN(newNN, newerNN), expect NNACK(newerNN)
- Send RQSD(NN,0), expect SD(NN,F0,F1, F2, F3), remember these
- For each Service flagged:
  Send RQSD(NN,service#), expect ESD(NN,Service#,version,…)

- ….

Park testing until a chat with David Ellis

Do we have set username opcode, or extended read/write parameters etc

11 Hardware

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

11.1 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.

11.2 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.

11.3 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.

11.4 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..

11.5 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.

11.6 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.

11.7 Compatibility logo

Subject to the satisfactory completion of the hardware compliance checking, the documentation **or** PCB, but at least one, should carry the VLCB compatibility logo. The version number of the MNS spec that the module is compliant with shall also be included.

12 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
- WebFCU
- 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 configuration file such as JSON or JavaScript.

13 Firmware Documentation

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.

13.1 Compatibility logo

Subject to the satisfactory completion of the compliance testing, the documentation should carry the VLCB compatibility logo. The version number of the MNS spec that the module is compliant with shall also be included.

14 Compliance and 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 125kbits 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.

14.1 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.

14.2 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.

14.3 Compliance testing

The official compliance software solutions will be available to developers, so as to facilitate testing the module against the specification. This is the only legitimate way to verify compliance.

Upon successful compliance, the VLCB compliance logo **must** be displayed on either the pcb **or** the accompanying documentation if the module designer claims compliance. The version number of the MNS spec that the compliance is tested against must also be displayed in the same fashion.

Note that other additional methods of compliance may be added or substituted,

15 Appendix A - Module User Interface

15.1 Options for displaying state and errors

State	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
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	Green 0.25s flash, Yellow ON	Message received	*** RX Shown for 0.5s
Message acted upon	Off for 0.5s then ON	Green 0.5s flash, Yellow ON	Message processed	*** RXA Shown for 0.5s
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	FATAL ERROR!	*** F!! Shown continuously

Where *** continues to show mode

15.2 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 may be provided by the mandatory MODE message with a zero node number. Preferably a physical input device such as a push button is also provided. The module shall enter Setup mode from Uninitialized mode if the push button is held down for at least 4 seconds, see section 3.3.2 Uninitialised to Setup.

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.

16 Appendix B - Service Data

16.1 Module Parameters

See https://merg.org.uk/merg_wiki/lib/exe/fetch.php?media=cbus:cbus_-_new_parameter_structure-6.pdf for details for parameter structure.

16.1.1 Parameter block

The module parameters provide information about the module's capability, information for the bootloader and information about the physical hardware. Entries in red are deprecated and replaced by service discovery but included for backwards compatibility. Entries in blue are only required for the CBUS PIC bootloader.

Address	Param#	Name	Usage	VLCB should set these values
	0	Number of parameters		24
0x820	1	Manufacturer	module manufacturer	See ModuleId
0x821	2	minor version		The minor version ascii character x.Y.z.
0x822	3	ModuleId	Module type identifier. See cbusdefs.h	Combined with Manufacturer for a 16bit code unique to the type of module. The ID can be obtained by the module developer using the VLCB web portal.
0x823	4	No Events	Max number of events available	Max number of events supported.
0x824	5	No EV per event	Max number of Evs per event	Max no Evs per event
0x825	6	No NV	Number of Nvs	Number of Nvs
0x826	7	Major version		The major number X.y.z

0x827	8	flags	Module's capabilities and settings	See below for each bit.
0x827	8.0	Consumer	Indicates if the module is able to be configured for consumed events.	Set if the Consumer service used.
0x827	8.1	Producer	Indicates if the module is able to be configured for produced events	Set if the Producer service is used.
0x827	8.2	FLiM	Indicates if the module is in FLiM mode (1) or SLiM (0).	Set to 1. VLCB does not support SLiM.
0x827	8.3	Bootable	Indicates if FCU can use the CBUS PIC bootloading protocol	Set to 1 if the requirements of the FCU bootloading process are met. See section 16.2 Parameters for bootloader.
0x827	8.4	CoE	Indicates if the module is able to consume its own produced events.	Set to 1. VLCB mandates self consumption.
0x827	8.5	Learn	Indicates if the module is in Learn mode.	Set if currently in learn mode.
0x827	8.6	Service Discovery	Indicates if the module supports service discovery	Set to 1. VLCB mandates service discovery. CBUS modules should have this set to 0.
0x827	8.7		Currently unused.	0
0x828	9	Processor Id		Type of CPU, 15=PIC18F26K80. See cbusdefs
0x829	10	Protocol	e.g. CAN	Set to 1 if CAN service is used, 2 for Eth, 3 for MiWi

0x82A~ 0x82D	11~15	Load Address	The start address of the application code.	0x0800[2]
0x82E~ 0x831	15	Processor code	PICs support the ability to determine the processor type (DEVID)	Hex file contains 0x0. RQNPN to read processor code dynamically from hardware.
0x832	19	Manufacturer code		1 for microchip, 2 for Atmel, 3 for Arm
0x833	20	Beta version		The Patch number x.y.Z
0x834~ 0x837	21~24	reserved		0
0x838~ 0x839		Number of parameters		0x0014
0x83A~ 0x83D		Address of module name	The module name is stored as a left justified, padded with spaces 7 character ASCII string within the module address space. This is the address of the name and can be used within the hex file.	Address of module name
0x83E~ 0x83F		checksum	The 16bit addition of bytes in parameter block 0x820 ~0x84D	checksum

16.1.2 Parameters for bootloader

The FCU supports over-the-network loading of module application code. The bootloader protocol itself is independent of processor type/family however the FCU imposes requirements on the file format and file contents to be loaded.

Existing CBUS modules contain a block of code residing between 0x0000~0x07FF which provides the bootloading functionality. If the VLCB module application firmware is compatible

[2] Load Address: According to the bootloader protocol the application firmware can specify any start address however the current CBUS PIC based bootloader firmware assumes an address of 0x800.

with both the FCU requirements and the existing bootloader then the Bootable flag may be set. The BOOT service should also be included to support the BOOTM opcode.

The requirements imposed by the FCU make the assumption that the module is based on a Microchip PIC processor, although it may be possible to implement a compatible application with other processors.

16.2 ESD data bytes

The Minimum node service currently does not return any data within the ESD response, all data bytes are set to 0.

17 Glossary

CAN	Controller Area Network. A standard communications bus originally defined by Bosch. Widely used in cars, industry and other electrically noisy environments.
CANID	CAN identifier
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
EN#	Event Number index
EV#	Event Variable index
FLiM	Full Layout implementation Mode
GridConnect	A means of encoding CAN frames for transmission over an ASCII serial link. See https://www.google.com/url?q=https://www.gridconnect.com/products/can-usb-adapter-pcan-usb&sa=D&source=docs&ust=1665387273449617&usg=AOvVaw0fepa9tJq-858M5sbX66fV
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