

CBUS - A universal layout control system

[Documentation](#)

[Schema files](#)

[Index to other public Wiki pages](#)

Introduction

CBUS is a Layout Control System running on the [W²CAN](#) (Controller Area Network). A description of the system can be found on our public webpages. [cbus.php](#) and [cbus2.php](#)



Identifiers

1. **Nodes** are the basic unit of CBUS, and each is usually implemented on one PCB. Node-ids are 16-bit, and are assigned by the user.
2. **Event** messages are sent by 'Producer' nodes and received by 'Consumer' nodes to cause some action. Event-numbers are assigned by the user, and are one of two kinds:
 - Short-event: 16-bit device#
 - Long-event: 32-bit, made up of the concatenation of a node-id and a 16-bit event#.
 - They are distinguished by the opcode of the message. Each is further qualified as an 'On' or 'Off' event by that opcode.

Message Formats

General format is 8 bytes: { opcode, optional data }

Where the opcode informs the receiving node what to do, using the data as necessary.

There are many opcodes, including:

- On-Events and Off-Events
- Train control, including programming
- Node configuration

Long Event format is: { opcode, [node-id(2).event#(2)] },

Where the concatenation [node-id.event#] is considered to be one 32-bit event.

Short Event format is: { opcode, node-id(2), device#(2) },

Where node-id and device# are independent. The device# is considered a 'short-event' or device#.

More than one node can send this event, e.g. throttles, and it will have the same effect.

By convention, device#s 1-9999 denote 'action' events; 10001-19999 the matching denote sensor events;

and device# 10000 the 'Start-of-Day' (SoD) event.

The reason for this partition is that CABs can send short events up to 9999 and it would not be advisable for these to clash with sensor events by mistake.

Tools:

- **FCU**: Windows based FLiM Configuration Utility for configuration of nodes and events.
- **JMRI**: Java Model Railroad Interface for configuration, monitoring and operating.
- **ROCRAIL**: A complete package for layout and loco operation.
- **SSI** (Solid State Interlocker) from **GPPSOFT**: A complete layout control system following British signalling and control practice.

Implementation Notes

- CBUS operates over CAN at 125kbps.
- CAN is bidirectional and has built-in error correction and message re-send.
- CBUS CAN frames have an 11-bit header and an 8-byte data-part.
 - The data-part carries the CBUS message.
 - The header must be unique, and this is ensured by including the 1 byte CAN-ID assigned to the sending node.
- The CAN-ID is retained by the node, moving it to a new layout may cause a CAN-ID conflict.
- In SLiM mode, the node-id is set by switches, and the CAN-ID is set equal to the low-byte of the node-id.
- In FLiM mode, if the node does not have a CAN-ID, one is automatically obtained by self-enumeration: the node asks all other active nodes for their CAN-ID, and then assigns itself one that is not in use.
 - NB: New modules *must* be introduced to the bus one at a time.
- The CAN-ID may be re-assigned: manually by a double push of the node's pushbutton; or by using the FCU.
- CBUS uses 29-bit header CAN messages for bootloading.
- A complete description of CBUS including the full specification and implementation notes is contained in the 'Developer's Guide' which can be downloaded from [developer_6b.pdf](#)

See also the 'documentation' link at the top of this page.

From:
https://www.merg.org.uk/merg_wiki/ - Knowledgebase

Permanent link:
https://www.merg.org.uk/merg_wiki/doku.php?id=public:cbuspublic:start&rev=1579706993

Last update: **2020/01/22 15:29**

