An introduction to CBUS

CBUS is a non-proprietary Layout Control Bus (LCB) for model rail use which has been developed over many years. It first went Public in 2007. It is open for anyone to use, including for Commercial purposes, under a CC BY-ND license. It is run and controlled by the CBUS Consortium which currently consists of the 5 named copyright holders. It is independent of any manufacturers or other organisations. What follows is a brief history of its development.

History

There has long been a demand (request) for a standardised LCB. Manufacturers have developed their own proprietary schemes which are not interchangeable. Discussions between active modellers took place on the DIY-MetaDCC email list moderated by Chuck Card commencing at least by 2004. Many views were expressed and there was no general consensus as to the transport or structure which was the most suitable. Different modellers had significantly different requirements e.g. DCC cab busses, accessory control busses, computer control of layouts, compatibility with existing busses etc.

A separate email group (the MRRLCB group - June 2005) then formed to concentrate on developing a single scheme with the intention of submitting this to the NMRA (National Model Railroad Association) when it had come to a successful fruition. Once again there were substantial, and possibly irreconcilable views on the purpose, protocols and methodology but all based on reasoned arguments from 'real' MR perspectives. However, one almost universal agreement was to use the industry standard CAN bus as the transport mechanism. The debate was how best to use CAN for a MR LCB.

While the debate was rather long, it was extremely useful as it allowed all possibilities and problems to be considered by experts with real MR experience and the use of such a LCB in real situations.

The group eventually divided into two philosophies. One was to make maximum use of the capabilities within the CAN protocol and available CAN hardware including the extensive filtering and masking available to differentiate messages and reduce the load on the individual CAN nodes (see later for definition of node). This group advocated using the extended CAN frames with 29 bit headers. The second group advocated just using CAN as the transport and keeping all the message content in the data segment. This made the scheme essentially transport independent and did not rely on any capabilities within the CAN node for filtering. While this simplified the structure, it did not make full use of the CAN device capabilities and left more processing to the subsequent node firmware. However, it did allow for the use of the standard (11 bit header) CAN frame with a possible small increase in data throughput. This latter became CBUS.

The reason for this background is to emphasise that many possibilities were fully debated by a group of active and experienced modellers with real problems to solve and the pros and cons have all been considered. Most group members are also professional electronics or network / software engineers .

At no stage was development linked to any manufacturer or organisation / society.

The transport mechanism

The platform for CBUS is based on the CAN protocol. It is a true CD/MA system (collision detection/ multiple access) so there is no 'master' controller and is not a 'master – slave' configuration. CAN is a bus system developed by the Robert Bosch company, initially for automobile use. It has become an industry standard (ISO 11898) with a wide range of hardware and firmware / software support. (For a good introduction see the Microchip application note AN713.) CAN was developed for the secure and error free transmission of small amounts of time and safety critical data between multiple devices within a closed system like a motor vehicle or aircraft. It seemed ideal for a MR bus and had already been adopted by Zimo and more recently by ESU for just such a purpose. Full information on the CAN bus is available widely elsewhere. No attempt will be made to describe it here.

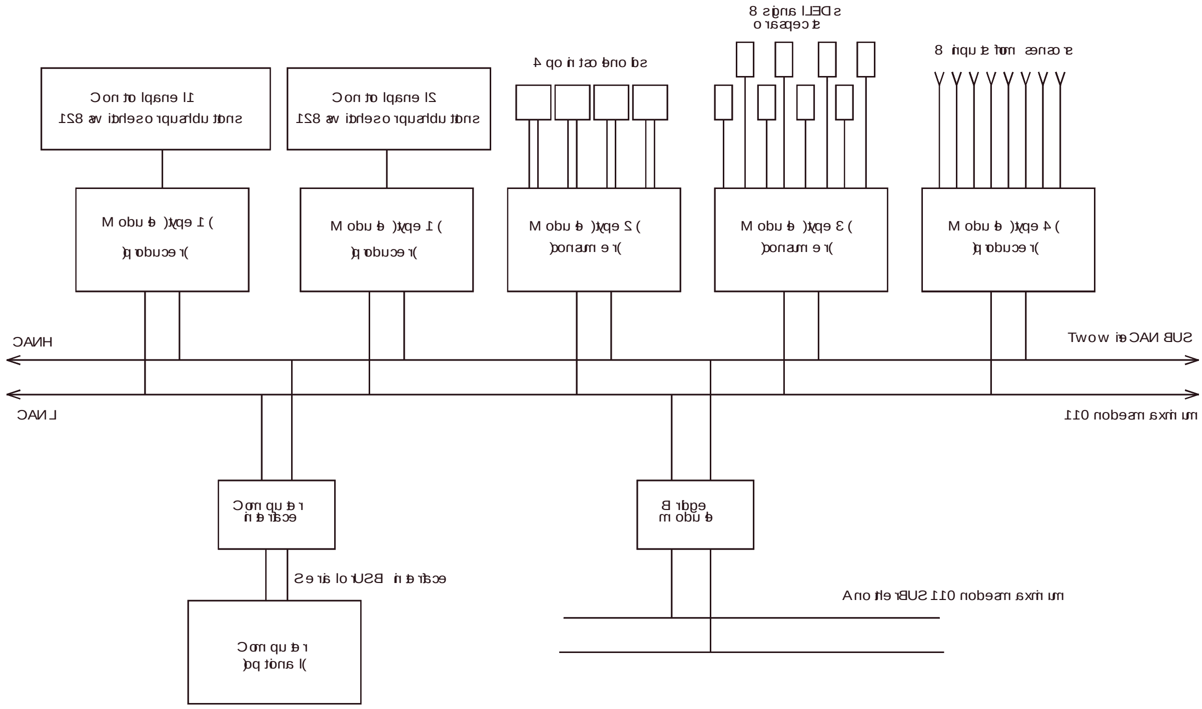
We have decided to use the standard CAN frame with its 11 bit arbitration (header) field. Also a CAN bit rate of 125KB has been adopted for two reasons. It allows sufficiently long bus cable lengths (500m) and Microchip recommend it as the maximum rate for which ceramic resonators can be used as the clock source, so reducing cost over Xtals. However, the CAN bus and associated devices have a maximum rate of 1MB.

CAN uses a non-destructive bitwise arbitration scheme on the header field. Where two nodes attempt to send at once, the node with the lower number in the header will win access. This allows for prioritisation of messages but sets the requirement that every node needs a unique header, referred to as the node ID.

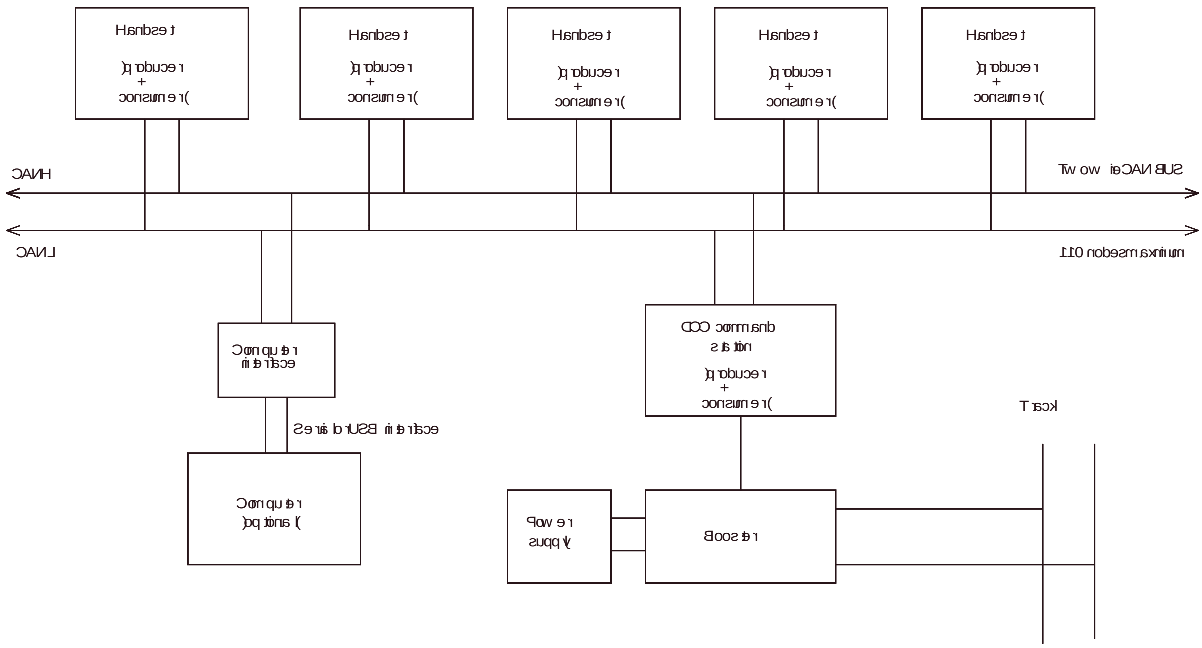
Each CAN frame has a header and between 0 and 8 data bytes. The CBUS scheme uses the CAN header purely for arbitration and prioritisation. All the layout specific messaging is in the data segment. However, as the CAN hardware contains its own powerful error correction mechanism (a 15 bit CRC check), there is no need for any error detection in the message itself. Any CAN frame with an error is rejected.

Note. With CBUS, the CAN filtering is not used. All valid CAN frames are transferred into the receive buffer(s) of every node. There is a direct parallel here with DCC packets and decoders.

While ISO 11898 doesn't specify the 'wiring' required, all CAN systems use a differential pair of wires for signalling. These require a load resistor of nominally 60 ohms between them, often a resistor of 120 ohms at each 'end' of the network, but CAN is not a 'transmission line' scheme so matching is not required and, in practice, the load can be anywhere. As modules connected to the CAN also will require power, the result is a four wire network where two wires (twisted paid for interference limitation) carry the information and two wires carry the power. This is particularly useful for modular layouts as inter-board connections are only ever 4 wires.



CBUS Basic arrangement for layout control



CBUS Basic arrangement for CCD 'bus'.

Methodology.

Node enumeration mechanism.

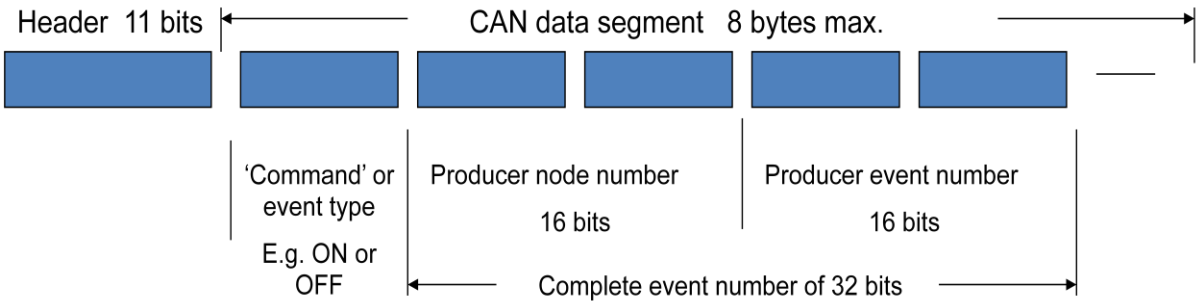
A node is defined as the hardware unit that attaches directly to the CAN bus. It is responsible for all the CAN associated activity and the receipt and transmission of messages which are contained in accessible 'buffers'. In practice, the hardware will contain a microcontroller which also performs the message translation into actions but this will be 'application specific'.

The requirement for CAN is that each node has a unique ID number. The hardware drive capability of available CAN interface ICs limits the number of active nodes on any one bus to around 128 (110 for the 82C250 or the MCP2562).

Messages

A CBUS message is from 1 to 8 bytes long. (It is possible to have long messages contained in multiple frames).

Each CBUS message starts with a 'command' byte or OpCode. The top three bits of this byte define the number of bytes in the message. There is no separate 'byte count'.



This gives a possibility of 32 commands for each number of subsequent bytes. As the message will arrive at every node, the firmware will decode this command byte first. Depending on their application, a node may only act on messages of certain lengths.

There is provision for an 'extended' or two byte command giving an extra 256 commands per data length and also a 'streaming' protocol for multi-message frames.

Basis of CBUS message structure.

The Producer / Consumer model

One of our 'starting' requirements was for a LCB that needed no 'configuration utility' that ran on a PC. This led to the 'Producer / Consumer' model where messages were 'events' and modules consisted of either producers or consumers. Under the strict P/C protocol, an 'event' is simply a number issued by a producer when an input changes. This bore no relationship to any actual hardware and works well provided every event has a different number. A consumer is taught what to do when it receives that number. For practical reasons, the 4 available bytes of the CBUS message were split into two. The first two are the Node Number of the producer and the second two are the inputs to that producer. Provided the NN is unique, the second pair can relate to (and be fixed by) which input changes. For example, if the NN is 16 and the input that changes is number 3, then the produced event will be

<00> <10><00><03>. (Numbers are in HEX)

This arrangement makes setting the NN with switches easy and the 'input' number can be fixed in the code of the module. It will be up to the user to ensure uniqueness of the NN.

A consumer is taught what to do by putting it into a 'learn' state and then sending that event. There will be switches on the consumer that need to be set to direct that event to a desired output and to set the specific action (on, off etc). Having been taught the event, the module will be taken out of learn mode and next time it sees that event it will act accordingly. With this simple scheme, a consumer doesn't have a node number. Such a 4 byte value has been called a 'long' event. It satisfies the requirement for not needing a configuration tool and will work with just physical switches (or equivalent) on a module.

The Device Addressed model.

As the complexity of the modules increased, it soon became impractical to 'set up' a module using physical switches. The original P/C model served its purpose but was a 'one to many' scheme so while adding more consumers was simple, adding producers was difficult. This was apparent when we wanted multiple CABs which were, in effect, generic producers. Also when we wanted several control panels acting on a common turnout or route. We required a 'many to many' scheme now. This led to an alternative which was the Device Addressed method.

At the heart of this model is the concept of a 'device'. This is the link between the message and the real world. A device is represented by a 16 bit (two byte) number and relates to a physical entity with properties. While CBUS deliberately does not attempt to define what these devices are or can be, examples are points (turnouts) and their associated drivers, switches on a control panel, block occupancy detectors, locos on a DCC system etc. It is also acceptable for these devices to be virtual devices within a PC such as icons on a 'glass panel' or software routines that perform a function. The only requirement is that devices have unique numbers and associated properties. It is up to the user

to allocate device numbers to actual devices. These are referred to as DNs. Associated with each event are event variables (EVs) which set actions for the module and device where necessary. There is a direct parallel here with setting a DCC loco 'long address' and setting / reading loco CVs 'on the main'.

In reality, devices will be attached to or through a specific CAN node – actual hardware. Hence the devices have to be associated with this hardware. For this purpose, a node is given a node number (NN). Again a 16 bit number but entered when the node is in setup mode. This way, the DNs can be programmed into a specific node (or nodes). The scheme allows more than one node to drive the same 'device' and devices to be moved between nodes. An example here would be a crossover or fiddle yard where there was a point driver node at each end but the point pair was considered as one device. Similarly a point and signal change could occur as one device operation but with different drivers. In this mode, only the device number matters. This has a 16bit range. However, a producer module will still send its allocated NN in the first two bytes of a message but this becomes for diagnostics only. (Which node sent it). As only two bytes mattered, they were called 'short' events.

The programming of the NN in setup mode corresponds to setting the DCC loco short address on a programming track. This only needs to be done once as all subsequent operations can reference the node by its number over the bus.

Associated with the NN are node variables (NVs) . These are where a node requires telling what to do, independent of the individual taught events. Node variables would only be used where there is a need to set specific node parameters, possibly like Baud rate for a serial connection to a PC. NN and NVs are not part of the general LCB messaging process.

Notes:

It is possible to have long and short messages at the same time. Whether a message is long or short is determined by the OpCode. OpCode, OPC or Command byte are terms used interchangeably when describing CBUS.

There are separate documents defining the 'protocol' or 'specification' and a Developer's Guide which gives technical details suited to manufacturers or CBUS users who wish to develop their own modules, systems or software. Also available is a software Configuration Tool (the FCU).

The CBUS specification is freely available under a Creative Commons CC BY-ND v4.0 license so can be used by anyone, including for commercial purposes but may not be changed. The details of the license are at:

https://creativecommons.org/licenses/

Mike Bolton (21st Sept. 2021)