CBUS - A universal layout control system

Documentation

Introduction

When first introduced, CBUS was the product of over 4 years development by MERG members Gil Fuchs and Mike Bolton.
During that time CBUS went through many stages of evolution and refinement including some major changes in direction until at version 4 in 2007 it was felt sufficiently developed to formally introduce the scheme, not just for MERG members but the world at large. Since its introduction developments have continued with contributions from a much wider team of MERG members.

The intention was to develop a system for comprehensive layout control based on a general purpose Layout Control Bus (LCB). The fundamental tenet was that of 'simplicity' without sacrificing 'universality'– a difficult juggling trick. It had to be affordable and easy to install and set up by non-technical users. It also had to cover the range from small, simple home layouts to the largest and most complex club layout imaginable.

So what are the functions of a layout control system. You can divide these into two basic categories.

1. Control of devices (outputs)
2. Detection of 'states' (inputs)

Examples of (1) are changing turnouts (points), signals, power to block sections, turntables, level crossing gates, layout lighting, setting routes, controlling the speed and direction of locomotives (by DCC or analogue DC) and any other electrical or electro-mechanical devices that may be on a layout. Examples of (2) are control panel switches, block occupancy detectors, bar code or RFID readers, turnout direction sensors, turntable position and 'RailCom'™ track detectors.

At the basic level, we wanted our system to both look and operate like a conventional 'hard wired' system having a control panel with switches to operate turnouts and simple route setting. It also had to allow use as a 'CAB bus' for DCC systems so handsets (CABs) could be connected to a DCC command station using the same wiring. At the other extreme, it had to allow full computer control, using multiple computers if necessary, and a fully automated layout with many thousands of inputs and outputs.

So far, CBUS has been referred to as a 'system'. The CBUS system can be regarded in two parts.

1. The hardware
2. The messages

The two are not completely independent as the style and frequency of the messages is determined by the hardware capability. However, they will be described separately.

Hardware requirements of the BUS

The choice of CAN.

The CAN bus (Controller Area Network) was developed by the Robert Bosch company in the 1980s for use in motor vehicles but has since been applied to many other types of machinery including aircraft and medical scanners to name just two. It became an open international standard as ISO 11519 in 1994 and a higher speed version as ISO 11898 in 2003. It is now used in virtually all modern motor vehicles so there is a wide applications base and ‘off the shelf’ components are readily available. When we looked at CAN, it seemed pretty much the ideal for a LCB. It was intended for relatively infrequent transmission of small amounts of data between devices for control purposes where response times and safety were paramount. Unlike the more familiar ‘Ethernet’, it was not intended for shipping large amounts of data between computers. Another advantage of CAN was that it was already in use for a LCB by Zimo, so there was proof it worked in a model railway environment. The data rate chosen for CBUS is 125Kbps. This is one of the defined CAN rates which go up to 1 Mbps but there is a trade off against cable length. 125Kbps allows lengths of up to 500 metres, good enough for even most garden layouts. The wire should be a twisted pair but doesn’t need to be screened. Only the ‘standard’ CAN frame is used.

The messaging scheme.

After much debate, we settled on the ‘producer-consumer’ model at least for layout control. For those used to the idea of sending specific messages from A to B – a ‘source-destination’ scheme, this is a very different concept although widely used for industrial control systems. Imagine changing a switch on a control panel. This creates an ‘event’. A frame is sent on to the bus which contains no source address, no destination address, no information, just an ‘event’ number. The node sending an event is called a ‘producer’. All nodes capable of processing an event are ‘consumers’. Every consumer on the layout receives the event. What the consumer does with the event will depend on information already in the consumer. In effect, a consumer has to be taught what to do with any event it has to act on and to ignore events that are not relevant. This is an extremely powerful and very flexible arrangement. Producers can send many events. Many consumers can act on an event and in different ways. Consumers can also act in the same way for different events. CBUS is a ‘one to many’ scheme which means that a specific event number will be restricted to one producer only. The above description may seem rather abstract. Here is a recognisable example. You want a push button (PB) on a control panel to set a route and the corresponding signals. You have a producer node on the control panel. You have a number of consumer nodes that drive turnouts and signals. Let’s say the PB creates event number 1. Turnout controller A has been taught that event 1 means set turnout 1 to normal, turnout 2 to reverse. Turnout controller B has been taught that event 1 means set turnout 3 to reverse, turnout 4 to reverse and turnout 5 to normal. Signal drivers C and D have been taught that event 1 means set the various signals or aspects for that route. Pressing the one PB sets the route and all the signals in one go. Obviously, a different PB could send event 2. The various consumers would know to act on event 2 in a different way to event 1 so set a different route. With this P-C model, you have effectively transferred some of the decision making from the producer to the consumer. It still allows for computer control or assistance, such as interlocking, if the PC or other interlocking system is placed between the original producer and the final consumer. Now, the switch event is recognised by the PC (as a consumer) and when a decision has been made, the PC sends another event (as a producer) to the final layout consumers. Items like block occupancy detectors are producers of events so a PC can know if a block is occupied. While full PC operation is easily accomplished, it is equally easy to configure quite complex layouts without any ‘programming’ at all. The latter will be described in our small layout implementation model (SLiM).

As a result of experience with 'real' layouts, we have since introduced an alternative to the P/C scheme, called 'device addressing' or 'short events'. While the message format remains the same,

layout 'devices' are given numbers by the user. The range is a two byte number or 1 to 65535. This allows a number to be associated with an actual layout device such as a turnout or switch rather than an abstract 4 byte value set by the node (as in the P / C scheme). An advantage of this method is where a CAB sends messages to turnouts by turnout number and any CAB can activate the same turnout, route, signal etc. Also layout sensors have numbers so the control panels, PCs etc. can know where the message came from. In the jargon. this gives a 'many to many' capability which fits well with computerised schemes like JMRI and, conceptually, is easier to understand. It does need a way of teaching the 'device number' though and for this we have a comprehensive configuration utility (the FCU) which runs in a Windows environment and connects to the layout via our CANUSB4 interface or wirelessly via our CANPiWi module.