

# FlexRay–MilCAN Bridging

D.Summers  
University of Sussex

Dr P.Charchalakis  
University of Sussex

Dr E.Stipidis  
University of Sussex

Dr F.H.Ali  
University of Sussex

**Abstract**—In a modern vehicle network, safety-critical systems must co-exist with current deterministic networks such as MilCAN. Additionally, it may be advantageous to use MilCAN to monitor the performance of a safety-critical system such as FlexRay.

Following a brief overview of both MilCAN and FlexRay technologies, the progress of an ongoing investigation into bridging the two standards is presented. The current solution is acceptable under test conditions, but unlikely to be suited to real-world use. Future plans extending the investigation on to new and more suitable hardware are then briefly discussed.

## I. THE MILCAN–A PROTOCOL

MilCAN–A is an open standard interface[1] implemented in devices destined for use in military vehicles. It defines a deterministic communications protocol as an additional layer on top of the popular Controller Area Network standard[2], allowing messages to be exchanged between devices with a guaranteed maximum latency. This guaranteed latency of delivery means that MilCAN-compliant devices can safely be used in real-time or deterministic systems, since high priority messages will be delivered within their latency contract even in cases of high bus load.

The underlying network operates at a speed of either 250 or 1000 kbps, dependant on application and bus length, and supports 29-bit message identifiers, allowing MilCAN devices to be interoperated with SAE J1939 networks[3]. The protocols are bus-compatible, in that they can be run on the same bus by different groups of devices without interfering, but communication between MilCAN–A and J1939 must be facilitated through a bridge.

The protocol achieves low-latency performance using a periodically repeating transmission schedule (the MilCAN cycle), synchronised by a master node (defined as the responsive node with the lowest source address). The schedule is divided into 1024 Primary Time Units, and frames scheduled to be transmitted in that PTU are queued for transmission at the relevant node. Message priority is determined in a two-stage process: first, the highest priority message in the transmission queue at each node is placed on to the bus, then CAN arbitration is applied to the message identifier to determine bus priority.

The Controller Area Network standard performs message arbitration on a bitwise basis, comparing the frame identifiers

of all frames on the bus and preferring recessive bits over dominant ones in each place. This has the result that lower-valued frame identifiers take priority. Controllers transmitting frames that lose priority automatically cease transmission but continue to receive, leaving the “winning” frame alone and avoiding bus collisions.

MilCAN frames use a custom message identifier format, which includes indication of message primary type and sub-type and a priority level. Message types indicate the semantic content of the frame, typically being drawn from a standardised table of values including Navigation, Lighting and Diagnostics, as well as automotive and other types. Since the network is message-addressed rather than node-addressed, the message identifier table also includes a “physically addressed” value that allows a message to be marked as coming from or destined for a specific node. Priority levels range from zero to seven, each defining a maximum delivery latency from “one communications cycle” to “whenever the bus is free”. Since message transmission takes place in order of priority, high priority messages will meet their delivery contract, even if the bus is heavily loaded. It is worth noting that message type and message priority are entirely separate: any device may transmit at any permitted priority (0 is reserved for network control messages), depending on the importance of the message function in question. Since MilCAN is based on CAN, frames may have a payload of length varying between zero and eight octets.

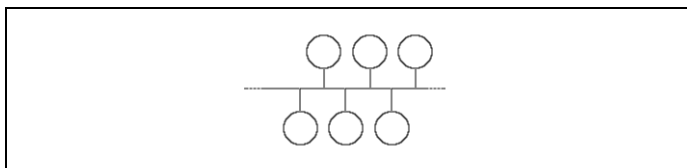
Adding the MilCAN layer on top of CAN adds a degree of determinism to the network, as well as a higher percentage utilisation since frame transmission occurs at scheduled intervals. This means that transmission can be predicted, to a degree, and that so long as the schedule is designed intelligently, each PTU can be fully loaded with frames and every PTU in the cycle can be occupied. MilCAN-based networks are found in a variety of current military systems, including the BAE Systems Land Systems (Alvis-Vickers) Challenger 2 tank, and the BAE Systems Bofors LEMUR fire-control computer and CV90AD-TD combat vehicle - air defence[4].

## II. THE FLEXRAY PROTOCOL

FlexRay is a new communications protocol, designed by the FlexRay consortium<sup>1</sup> as a safety-critical automotive X-by-wire system more flexible than its contemporaries, such as TTP. Bus data rates range up to a maximum of 10 Mbps, and each individual message may be up to 256 octets in length.

The FlexRay system specification[5] lays out most details of both hardware and software, but leaves the choice of some specifics to the implementor. Systems should use a dual-channel bus for redundant communication, although a given node need not necessarily be connected to both. In particular, the physical layer is left unspecified: ribbon-cable, twisted-pair cable or fibre-optics can all be used as required, although the development kits available at present concentrate on ribbon-cable for reasons of simplicity. The network should ideally be set up as a connected graph of active stars to maintain ideal EMI characteristics, but a Bus configuration is acceptable for small networks.

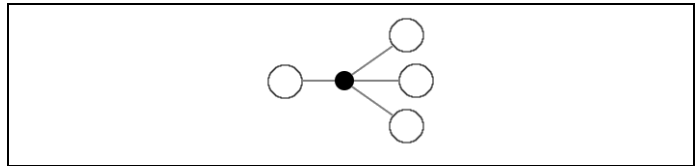
The active-star topology is a major feature of the FlexRay standard, particularly since the great majority of communications networks use a bus topology, so a moment will be taken to explain the concept. The bus topology (Figure 1, in which every node is connected to a single backbone via a series of taps) is used by many networks, embedded and commercial, including LIN, CAN and SPI. The active star topology (Figure 2) is somewhat rarer, being found in FlexRay and switched Ethernet as well as, to a lesser extent, USB. Instead of being connected to a common bus, every node has a direct connection to a star coupler which then forwards a received message to every connected node excepting the one from which it originated. This design tends to result in much better EMI performance, since any noise induced will be present only on one segment as opposed to the entire bus.



**Fig. 1:** Bus Topology

The software specification is a little more complete, defining clear structures for the communications cycle and data frames themselves. The system is based around a repeating TDMA, much like MilCAN, which is in turn divided into 2048 slots. Each slot is a whole number of macroticks in length, the macrotick being an agreed length of time that is constant for all nodes. Each slot is then divided into four sections: static segment, dynamic segment, symbol window and network idle time.

<sup>1</sup>A group of interested companies including BMW, Bosch, DaimlerChrysler, Freescale, General Motors, Philips and Volkswagen. (<http://www.flexray.org>)



**Fig. 2:** Star Topology

The static segment is a simple TDMA, with a static message schedule that is defined at compile-time for all nodes. If message transmission takes place only in this segment, and the segment TDMA is well organised, it is possible to achieve very high utilisation. The dynamic segment is the source of much of the flexibility in FlexRay, since message transmission here can be undertaken on an as-needed basis. There is no static TDMA, so frames carrying non-urgent or high-volume data can be transferred as needed on a loose priority basis. Additionally, it is possible to increase the available bandwidth by desynchronising the buses and sending different data on each during the dynamic segment, so long as the buses are re-synchronised before the start of the next slot. The length of both segments can be set during cluster setup, and either may be set to zero if that segment will not be required in this application. Finally, the symbol window is used for network management (one symbol to indicate the network is performing according to expectations, a different symbol if a problem has occurred) and the network idle time gives each node time to perform local administration and configuration tasks when needed.

FlexRay frames are semantically simpler than MilCAN frames, but have a larger payload segment. A single frame may carry up to 254 octets of data, but the header is somewhat basic, containing a frame ID, a set of flags, a data length code and a CRC. This simplicity means that FlexRay networks are very general and can easily be adapted to new configurations. Although the lack of a set of suggested type identifiers (such as is present in MilCAN) means that disparate systems cannot easily share information, the fact that FlexRay clusters are customised from scratch for their application means that this should not be a problem.

When considered against MilCAN, FlexRay has several advantages. First, FlexRay is fault-tolerant, in several senses. One of the two bus channels can be severed without affecting system performance, so long as no transmissions are scheduled to run solely on the compromised channel. A node can fail in a variety of network-hostile modes, and the low-level bus guardian built into the controller automatically locks out the transmitting component before the garbage data is dumped onto the bus. If a node falls silent, the nodes expecting to receive messages from it will log the problem (if logging is supported on the relevant architecture), but will continue to operate unless specifically programmed to halt on reception failure. Essentially, the system tends to fall back to a reduced-functionality state without compromising the entire network

when damaged. MilCAN is also fault-tolerant to some extent, but only in that it can continue operations when some nodes on the bus are non-responsive.

Second, FlexRay is capable of a much higher data rate, since its bus clock rate is an order of magnitude greater than the maximum possible rate for the CAN bus upon which MilCAN is based. Finally, it is run-time flexible in that the static segment is used to carry expected or safety-critical data, while the dynamic segment can be kept available to service unexpected or high-bandwidth low-priority transmissions.

However, it is worth noting that FlexRay is still a developing technology: there are very few manufacturers producing controllers and development kits that can form part of a FlexRay cluster, and building a FlexRay controller from first principles is essentially too great a challenge for a research group to undertake. Those kits that are available tend to concentrate on the static segment: at the time of writing, only Decomsys produce software capable of scheduling data transmission in the dynamic segment, and their support is not quite mature. In contrast, CAN development kits are plentiful and available at a wide range of prices and, if one does not suit a group's needs, it is entirely possible to construct one. Once a CAN controller exists, the MilCAN stack can then be established in software without a great deal of effort.

In summary, FlexRay shows great promise for the future. While MilCAN is currently cheaper and more convenient for commercial usage, this does not necessarily mean that it can be used in place of FlexRay, since it does not possess the required safety-critical features.

### III. THE FLEXRAY-MILCAN BRIDGE

#### A. Message bridging concepts

In any networking system, the ability to connect network segments together, or to translate messages from one protocol to another is important, as it allows devices designed to different standards to share information. The component which performs this connection and/or translation, whether implemented in hardware or software, is called a bridge.

A simple bridge would transfer every message on one network to the other: however, this may result in high bus traffic, much of which is not required by any device on the destination network. Instead, a more useful solution is the *intelligent* bridge, which filters the incoming messages and can be configured to forward everything, or only frames matching a set of criteria. A bridge that translates MilCAN to FlexRay and vice-versa is particularly useful, since MilCAN is an established vetronic protocol and is found in the control networks of many vehicles.

If a vehicle is to be upgraded with components that use

FlexRay to communicate, or if a safety-critical FlexRay automotive network needs to be connected with MilCAN-based subsystems, a bridge can be used to connect the networks together. This is typically much more economical than replacing the entire control network and upgrading all devices to the new protocol, since FlexRay devices are currently more expensive than equivalent MilCAN units.

An investigation was undertaken into FlexRay-MilCAN bridging, using commercially-available development kits. Although the investigation centred more on tunnelling than bridging (a MilCAN message was transported over a FlexRay backbone and re-emitted on the other side), an intelligent bridge is the eventual goal of this research.

#### B. Hardware set-up

In order to study FlexRay-MilCAN bridging, a custom network was built (figure 3), consisting of two MilCAN networks and a FlexRay backbone.

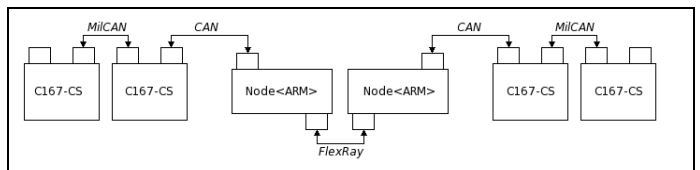


Fig. 3: FlexRay-MilCAN bridging test cluster

The MilCAN networks in the cluster were based on pairs of PHYTEC C167-CS development kits. These boards are based around the Infineon C167: a system-on-chip microcontroller with an integrated dual-CAN controller, dual RS-232 and several ADCs and PWMs, as well as a decent amount of digital I/O. A MilCAN stock library [6] was used for the MilCAN operation.

The FlexRay network used as the backbone in this cluster was built from a pair of DECOMSYS Node<ARM> development kits, which are fully-featured development kits based around an ARM microprocessor clocked at 166MHz. The Node<ARM> runs a Linux with the RTAI extension as an OS and, in contrast to the simplicity of the C167-CS, is essentially a highly specialised general-purpose computer. As well as the modular FlexRay controller board, the Node<ARM> supports LIN, RS-232, CAN and a large amount of analog and digital I/O. The network had the standard 10Mbps bus speed and was carrying no other traffic.

#### C. Functional Description

Although the Node<ARM> has an integrated CAN controller, it was not used. While the MilCAN stack is built on top of a generic CAN network, it requires access to high-resolution timers in order to maintain bus synchronisation,

and such timers are in short supply on the Node<ARM> devices. Although the RTAI system could successfully run a MilCAN stack, it remains to be tested whether doing so would interfere with the operation of the FlexRay stack. For this reason, running a MilCAN stack on the Node<ARM> was deemed non-viable at this time. Instead, a standard CAN connection was used to transfer messages between the Node<ARM> and the MilCAN network: since a MilCAN frame is a semantic reinterpretation of a CAN frame, this is not greatly complicated.

#### D. Results

Basic performance testing was done using a laptop running the Vector CANoe analysis package, injecting a single frame into the MilCAN network on one side and receiving the frame on the other side. Given the unloaded state of the network, all message reception during this test should be considered a best-case result. In all test cases, injection and reception times differed by between 3.4 and 3.8 milliseconds (figure 4).

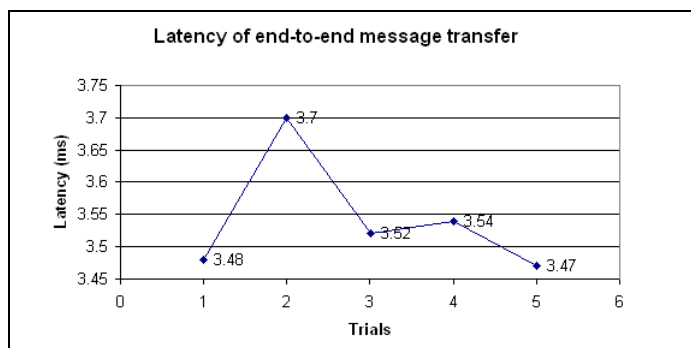


Fig. 4: Graph of average latency over multiple trials

The above investigation shows what appears to be a viable FlexRay–MilCAN bridge. Unfortunately, although the system is capable of rapid bridging in this simplified test case, the situation is likely to change when the system comes under heavy load. This situation could be partially resolved by operating the MilCAN networks at 250kbps and the CAN networks at full speed (1Mbps), but this may not be an acceptable solution in production systems.

#### IV. FURTHER WORK

The research group has recently acquired a new piece of hardware from DECOMSYS: a Node<Renesas>. As the name suggests, the development kit is based around a Renesas M32C/85 processor, and presents the same external interface as the Node<ARM>. However, although this development kit is only clocked at 24MHz, in comparison to the Node<ARM>’s 166MHz, there is no operating system overhead: the entire kit is purely interrupt-driven and remains in a simple “bare-metal” state during normal operation. It is worth noting that one of the example applications shipped with the Node<Renesas> is a FlexRay–RS-232 bridge, suggesting that bridging applications may have been a factor in its design.

Over the next three months, we will continue our bridging research on both platforms with the intention of producing a more reliable, scalable bridge with a similar level of performance. This will entail integrating a MilCAN stack into the RTAI OS on the Node<ARM>, as well as investigating the capabilities of the Node<Renesas> in this respect.

#### REFERENCES

- [1] “MilCAN A, Application Layer Specification”, Qinetiq (available on request from <http://www.milcan.org>)
- [2] Robert Bosch GmbH, CAN information website <http://www.semiconductors.bosch.de/en/20/can/index.asp>
- [3] milcan.org, Introduction to the standard <http://www.milcan.org/FAQ/faq.html>
- [4] milcan.org, Projects using MilCAN <http://www.milcan.org/Projects/projects.html>
- [5] FlexRay Consortium, FlexRay Protocol Specification, version 2.0 (available on request from <http://www.flexray.com>), 2004
- [6] “Integrated Vetric Systems – Intelligent bridging of vehicle networks over high speed backbones”, Periklis Charchalakis (Doctoral Thesis), University of Sussex, 2005