ESCAN – An Open Source, High Bandwidth, Event Scheduled Controller Area Network

A. Williams, C. Quigley, R. McLaughlin, Warwick Control

Event Scheduled CAN (ESCAN) is an open source, scheduling protocol for CAN. The aims of the protocol are discussed, including the ability to optimise the available bandwidth over CAN and enable maximum bus loading as well as providing a worst case determinism for message reception.

These advantages include a simple to implement basic protocol stack, no specialist hardware requirements needed to support the protocol other than a TTCAN compliant CAN controller (this is so that the retransmission of CAN frames can be disabled). The protocol also uses a low amount of CPU and memory overhead to transmit its schedule control data resulting in high potential bus loading at all CAN bit rates.

In this paper, the protocol itself will be introduced along with a brief comparison against other scheduling protocols in the literature. Preliminary experimental performance data for ESCAN is collected and compared with a TTCAN Level 1 implementation showing benefits. Finally an analysis will be conducted on the effect of using ESCAN as a scheduling layer for the CANopen protocol.

Introduction

ESCAN is an abbreviation for Event Scheduled Controller Area Network. It is a higher layer protocol that sits on top of the CAN protocol, allowing the simple scheduling of CAN messages. As the name suggests the scheduling is based on events instead of the usual time base used by several other protocols such as TTCAN. These events that facilitate the core operation of the scheduling are the CAN message events on the bus. Using message events as the foundation of a schedule, instead of a global time base, makes the implementation of a CAN schedule many times simpler and easier to develop and operate. This paper introduces the ESCAN protocol's main concepts, compares the performance of the protocol with a TTCAN Level 1 implementation on the same family of microcontrollers and then carries out an analysis on how ESCAN could be applied to the CANopen protocol.

Determinism and CAN

CAN was developed by Robert Bosch GmbH in the 1990s [6] and has become the most prominent open standard network protocol across the world's automotive industry. One problem that CAN faced was that it is inherently non-deterministic and designers tended to rely on the protocol itself to schedule the message transmission effectively. However, this approach

created a number of problems during systems integration such as non-deterministic message delays and bus loading. It is a well-known feature of CSMA-CD (Carrier Sense, Multiple Access, Collision Detection) type networks such as CAN that they can operate with reasonably predictable message latency up to about 40% bus loading. An excursion beyond this leads to great variability in the latency for lower priority messages.

A solution to the bus loading problem of the non-deterministic approach was developed in Volvo [5] that could determine the worst case message latency by assigning higher priority (or periodicity) signals to higher priority CAN messages. Lower priority (or periodicity signals such as those that are event triggered) signals are assigned to lower priority CAN messages. The result of this was that the CAN bus was able to run at a higher loading, with the worst case latency known at design time of CAN messages and therefore design trade-offs were able to be made to ensure acceptable latency. This is a *Pseudo-Deterministic* approach since the determinism is defined as a worst case latency and is the process that has been implemented in the Volcano tools [4]. The key principles of this CAN design process are based on a Publisher-Subscriber model [3]. This methodology improves the procedure of traditional CAN bus development using a *Non-Deterministic* methodology, since it allows the OEM to deal

much better with multiple system suppliers.

Pseudo-Deterministic approaches for CAN and work by Kopetz on the Time Triggered Protocol (TTP) [2] has led the way to the *improved Deterministic* approach to design for time triggered protocols. Time triggered protocols are deterministic in their nature and their associated design process deals with the systems integration issues much better as they allow the designer to easily conceptualise the communications timing and map signals and messages across different buses. They are also good for dealing with the problem of multiple suppliers in the same way as *Pseudo-Deterministic* approaches.

The CAN protocol was extended under ISO-11898 part 4 to Time Triggered CAN (TTCAN) to address the determinism problems of CAN and therefore provide a *Deterministic* approach for CAN systems. The main characteristic of TTCAN is that bus access is controlled via a Time Division Multiplexed Access (TDMA) like method using a regularly repeating cycle of time called the *Basic Cycle*. The *Basic Cycle* is divided into a fixed number of time windows (i.e. fixed at design time) which can be a mixture of any one of four types; *Reference Message*, *Exclusive Window*, *Arbitration Window* and *Free Window*. This is sent by the time master control unit (global time master) and controls the timing of the Basic Cycle [7]. It is a technology that is currently available for commercial automotive applications such as private networks and effectively doubles the usable bandwidth of CAN [1].

Introduction ESCAN

The Event Scheduled CAN (ESCAN) protocol is a scheduling protocol for CAN messages. The protocol is inherently simple, allowing a reliable schedule to be developed in the minimum amount of time, achieving high efficiency and reliable operation. The network is formed by a higher-layer protocol that sits on top of the CAN network specified in the ISO 11898 standard. [6] It is a way of controlling CAN messages on the CAN network through scheduling, in order to achieve maximum efficiency and determinism. As the protocol sits on top of the CAN network the physical connections, interfaces, messages, etc., remain as specified in the above mentioned ISO
standard. The terminology and names introduced in this chapter relate to the properties and functions in relation to the ESCAN protocol and its functionality, independent to the CAN protocol. The ESCAN protocol was introduced and described in detail in 2011 [8]. In this section the main concepts of ESCAN are briefly described.

Advantages of ESCAN

The ESCAN protocol has many advantages for embedded system developers over existing protocols such as TTCAN;

- Simple and easy to implement protocol with a basic software stack
- No dedicated hardware required i.e. scheduling is implemented in software.
- Reliable, with the facilitation of adding redundancy when the option redundant master is used
- Plug and play of extra nodes

- Excellent expandability of the network, without any rework to current nodes on the network. Schedule size can be altered during operation, and other parameters can be tuned to improve efficiency.
- Low bus overhead for schedule control data.
- High bus loads possible at all CAN bitrates

ESCAN network and nodes

An ESCAN network consists of two types of nodes; the Event Schedule Master (ESM) and an Event Schedule Participant (ESP) (shown in Figure 1). There must be at least one ESM and one ESP on a network. The protocol does not place any limitations on the number of ESPs on the network – this is governed by the physical properties of the CAN bus and node interfaces, as it is in the ISO 11898 CAN network [6].

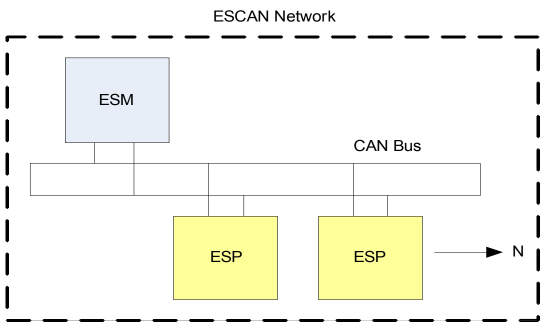

Figure 1 - A basic ESCAN network

The ESM is a dedicated node on the network for controlling and monitoring the schedule and does not perform any other function. It transmits two types of messages onto the network; a Reference Message (RM) and a Blank Message (BM) (see ESCAN Messages).

The ESP is a regular node on the network, performing its designed tasks and communicating information to other ESPs over the network. ESPs transmit their messages within allocated schedule slots and use received messages from other ESPs on the network (if required). ESPs only transmit one type of message – Data Messages (see the Data Messages (DM) section for more information on these messages).

ESCAN messages

There are three types of messages on an ESCAN network; Reference Messages (RM), Blank Messages (BM), and Data Messages (DM). All three types of messages are the traditional CAN frames specified in the ISO 11898 specification, but have special meaning within the context of the higher layer protocol. All types of messages can use either 11- or 29-bit CAN Identifiers depending on the application, however it is recommended that 11-bit IDs are used where possible to increase efficiency. Both ESM and ESP nodes receive all types of messages on the network in order to process and communicate using the schedule. An explanation of the schedule is provided in the ESCAN Schedule section, and detailed descriptions of the message types are in the following subsections.

Reference Message (RM)

The Reference Message is transmitted by the ESM and marks the beginning of a Row Cycle (see the ESCAN Schedule section). The CAN ID of an RM is reserved by the protocol and can be either 0x0 or 0x1, depending on whether a redundant ESM is implemented (see the Redundant ESM for Critical Applications section below). The RM has a fixed DLC of 1 byte containing the current row number within the schedule. The row numbering system in ESCAN begins with 0 and goes up to 255 i.e. 256 rows. This row number data byte enables ESPs to synchronise with the schedule. A diagram of the RM is shown in Figure 2 below.

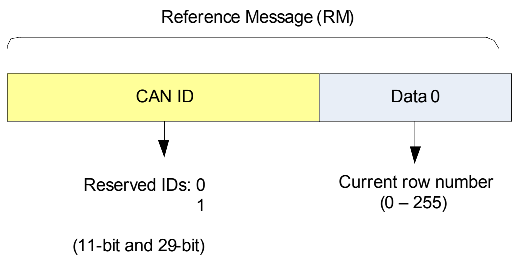

Figure 2 - Reference Message format

Blank Message (BM)

The Blank Message is transmitted by the ESM and is used to ensure the continuation of the ESCAN network when either free slots are implemented, or in the event of an ESP failure. The CAN ID of the BM is reserved in the ESCAN protocol and is either 0x7FE for 11-bit CAN or 0x3FFFFFFE for 29-bit CAN i.e. one higher than the lowest priority ID available. A BM contains no information for the other nodes and therefore has a DLC of 0 at all times. A diagram of a DM is shown in Figure 3 below.

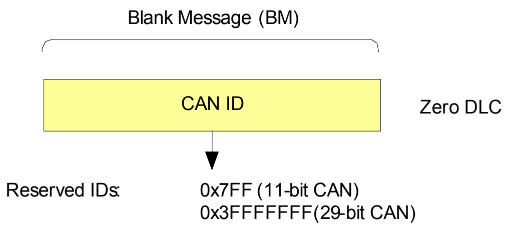

Figure 3 - Blank Message format

Data messages (DM)

Data messages are transmitted by ESPs only and are the messages used to communicate data from one ESP node to another. These are the normal frames transmitted on non-scheduled CAN networks. The only limitations on these messages are that they do not use the reserved CAN IDs of the RM and BM, and they are not remote frames (remote frames are forbidden on an ESCAN network). Apart from these they can be of any length (DLC) and any ID format (11- or 29-bit).

ESCAN schedule

All ESPs communicate to other ESPs by transmitting messages at particular times set out by the Global Schedule Matrix (GSM), a basic example of which is shown in Figure 4

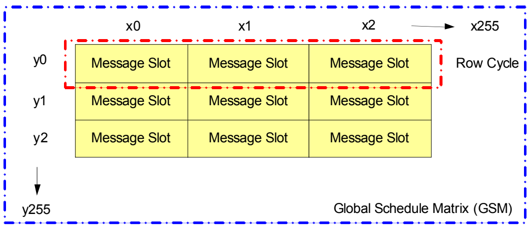

Figure 4 - The basic ESCAN Global Schedule Matrix

The GSM can be viewed as an X by Y matrix, where X is the number of columns and Y is the number of rows in the matrix. There can be any number of columns between 2 and 256 inclusive, and any number of rows between 1 and 256 inclusive. The numbering system within ESCAN always begins at 0 for both rows and columns. Each cell within the GSM represents a transmission slot for one of the three types of

CAN messages. Each row of transmission slots is called a Row Cycle. The transmissions in the first column of the GSM are always filled by Reference Messages from the ESM. The rest of the slots in the GSM are either filled with Blank Messages from the ESM or Data Messages from the ESPs.

Each DM slot is dedicated to a single message i.e. multiple messages do not share a slot. A message can transmit in any of the non-RM slots. Slots that are not occupied by an RM or a DM are filled in with Blank Messages from the ESM. The purpose of filling unused slots with BMs is so that there is always some form of message event in each slot. A slot without a message event would cause the bus to stop, and for this reason BMs are mandatory.

The ESM

The ESM is a dedicated node on the network. Its sole tasks are to transmit the RMs at the beginning of each Row Cycle, and to transmit BMs in slots which are not occupied by Data Messages from ESPs. Its responsibilities are to ensure that every slot within the GSM is filled by a message at all times. The ESM is the first node to transmit on the network. All ESPs wait for the ESM to transmit its first RM before they begin in order to synchronise to the schedule transmitted by the ESM. It was mentioned in the previous chapter that the ESM must fill all empty slots within the GSM with Blank Messages. The ESM does not need to know where the empty slots are in order to transmit its BMs. Instead the ESM waits for a period of time after the previous message in the schedule, known as the ESP-Response Gap (ESP-RG – see the section ESP-Response Gap), for any ESP to transmit a DM into the current slot. If after this period a DM has not been transmitted the ESM transmits a BM onto the network to fill the slot.

ESP-response gap (ESP-RP)

ESPs monitor the current position within the schedule by keeping track of the row number within the Reference Message and the number of Message Events received in that row since the Reference Message. In order for the schedule to work a time gap is introduced by the ESM in order to allow an ESP to respond with a message before a Blank Message is transmitted – this time gap is called the ESP-Response Gap (ESP-RG). The length of this gap is dependant solely on the slowest processor on the network.

There is no recommended ESP-RG value – it must be determined by the developer. There are two ways to calculate the optimum ESP-RG value; by calculating the response time for each ESP on the network using a simulator or other hardware setup; or by configuring the network using a large value for the ESP-RG and using an bus analyser to determine the actual ESP-RG of the ESPs on the bus. The ESP-RG can then be lowered to the optimum value during network operation through the ESM in order to achieve the best performance (see the Dynamic aspects of the Schedule section for more information on the dynamic aspects of ESCAN).

The ESP

The ESP transmits its Data Messages in specific slots within the schedule. The ESP's primary responsibility when it comes to the schedule is to keep track of the current slot within the schedule and to transmit its messages in the assigned slots when the time arises.

Receiving and transmitting

In order for the ESPs to work with the ESCAN schedule they are required to receive every message on the bus. The ESP has two variables in order to use the schedule; a Current Row variable, and a Current Column variable.

Synchronisation to the network

When an ESP starts up it is required to synchronise itself to the network schedule. Synchronisation is done through the reception of a RM from the ESM. The ESP waits until it receives a RM on the CAN bus. When the first RM is received the ESP sets the current row variable (according to the RMs data byte) and the current column is set to 1. The ESP is then in a position to communicate using the schedule. Should an ESP be disconnected from the network and reconnected again it is important that the synchronisation procedure is performed again, otherwise it will attempt to communicate with incorrect schedule information. This method means that an ESP can be connected to the network at any time, and is able to start communicating at the soonest possible moment.

Dynamic schedule negotiation

Higher layer protocols such as CANopen generally consist of a mixture of periodic frames (in the case of CANopen – Process Data Objects are generally transmitted periodically) and unscheduled bursts of data that are sent infrequently, for example configuring a CANopen node using Service Data Objects.

A scheduled approach to transmitting data works well for period data but it is wasteful to allocate slots in a schedule for frames that will only be used occasionally.

The ESCAN protocol extends the concept of a schedule tables and allows the participants in a network to dynamically expand and shrink a schedule table.

Acquiring a new slot

In normal operation the ESM will wait for the ESP-RG timeout to expire and then transmit a Blank Message in order to maintain the schedule matrix. A Blank Message is transmitted with a low priority CAN frame (that can be defined by the network designer) with a DLC of 0. The ESM keeps track of how many times a Blank Message has had to be transmitted in to a particular slot and after a configurable number of consecutive Blank Messages have been transmitted (default is 5) then the ESM will begin to transmit Empty Messages in that cell of the matrix. An Empty Message has the same format as a Blank Message, however a lower priority CAN identifier is used.

Once a slot has been marked as Empty it is now available for another ESP to use for transmitting messages. The communications stack on the ESP will keep track of where the empty slot are located and when an immediate frame transmission is requested the next available empty slot will be allocated to the ESP. The ESP can then use this slot to transmit any message it chooses with any identifier, depending on what the application on the ESP is currently doing. In the case of a CANopen node, the new slot can be used for SDO transfer between a Master and a Slave.

Releasing a slot

Once an ESP no longer needs additional slots in the schedule it can stop transmitting a CAN frame in that position. Once again, the ESM will begin transmitting Blank Messages in that slot to ensure that the network remains synchronised and the ESPs can still count the column index.

After the configured number of consecutive frames has been transmitted the ESM will flag the slot as blank and available for other ESPs to use.

Extending the schedule

Since the schedule can be built up dynamically with ESPs requesting and releasing slots as they need them the schedule could quickly fill up from its initial size. The ESM therefore always transmits a Reference Message at the end of the schedule table and immediately transmits a single Empty Message.

If no ESP is waiting for a slot then the ESM will then transmit the Reference Message with a row index of 0 and restart the schedule.

If an ESP is waiting for a vacant slot for transmission then it can use this Empty Message. The ESM will then flag this as a valid row and continue to transmit a Reference Message for it. It will also fill the remaining slots of the row with Empty Messages.

Reducing the schedule

To reduce the schedule the ESM monitors the last row of the current schedule table. Once a row consists of Empty Messages in its entirety then the row is deleted from the schedule and is no longer transmitted. The row count is reduced by one and the placeholder Reference Message transmitted at the start to indicate a blank row.

A comparison between Escan and Ttcan implementations

In order to compare the performance characteristics of the ESCAN protocol with the nearest equivalent higher layer protocol - TTCAN Level 1 (software implementation) - an Atmel CC01 processor was configured with the protocol stack for ESCAN and a basic 10 column x 10 row schedule table created to allow the CPU performance characteristics to be compared

against results obtained from analysis conducted with TTCAN by Quigley et al [1].

The experimental setup consisted of a single ESM node and an ESP configured to transmit a data frame in every slot of the schedule matrix. The frames transmitted are static and there is no application specific processing being done in addition to the ESP and ESM protocol stacks.

Bit-rate (kbit/s)	**Master CPU Loading (%)**	**Slave CPU Loading (%)**	**Max. Bus loading (%)**
50	3.6	1.4	96.5
100	4.9	2.7	95.5
125	5.9	3.3	94.5
250	8.9	6.4	90.9
500	14.4	11.9	84.9
1000	23.2	20.8	74.3

Table 1 - ESCAN Controllers @ 20MHz

Performance comparison of master nodes

The current implementation of the ESM protocol stack compares exceptionally well with a TTCAN Time Master node, with a significant reduction in the CPU requirements necessary to run the master task.

The CPU requirements for the ESM compares favourably in the way that it scales with an increased bit rate. At 1Mbit a TTCAN Master is at 80% CPU utilisation leaving only 20% available for the user application, an ESM node running at 1Mbit requires just under 25% of the available CPU cycles leaving 75% available for the application that is running alongside the protocol stack.

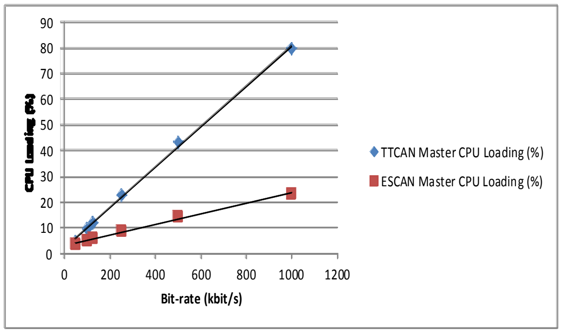

Figure 5 - Comparison of TTCAN and ESCAN Master CPU Performance

The memory requirements for the protocol Master (ESP) implementation used in this comparison are fixed for all schedule sizes as the usage is not dependant on the size (ROM 1284 bytes, RAM 31 bytes).

Performance comparison of slave nodes

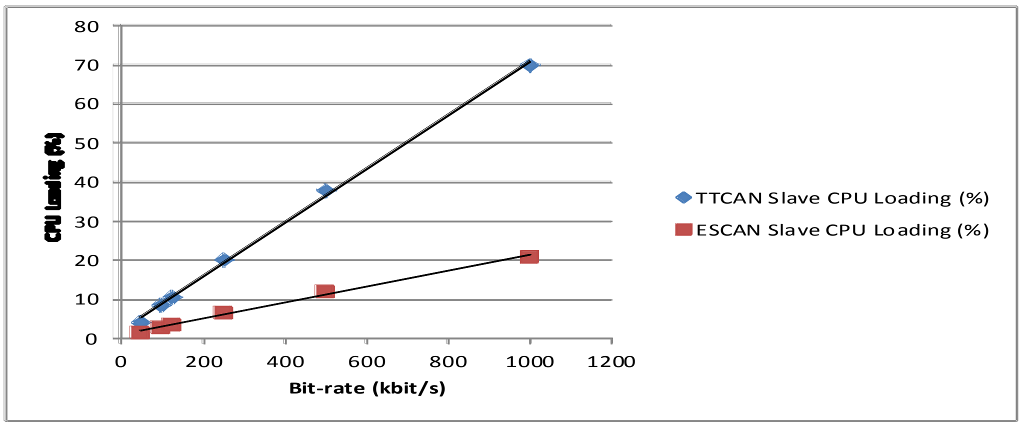

Figure 6 - Comparison of TTCAN and ESCAN Slave CPU Performance

The memory requirements for the Slave protocol implementation used is detailed in

Table 2, this highlights the difference in memory requirements for a basic implementation with a single 8 byte data frame.

Rows (Cycles)	**User Columns (Windows)**			
	1	**2**	**4**	**8**
1	25 *(58)*	26 *(63)*	28 *(73)*	32 *(93)*
2	26 *(64)*	28 *(72)*	32 *(88)*	40 *(120)*
4	28 *(76)*	32 *(90)*	40 *(118)*	56 *(174)*
8	32 *(100)*	40 *(126)*	56 *(178)*	88 *(282)*

Table 2 - ESP Memory Usage in bytes - TTCAN values provided for comparison in brackets and taken from a previous study [1].

As the schedule matrix is increased in size, the only additional memory required is for an internal message index. There would be additional memory requirements introduced by adding more distinct CAN messages however this is at the application layer and is not a consequence of the scheduling protocol.

Comparison of bus loading

The maximum bus loading achieved with the current implementation of the protocol stack is detailed in Figure 7, the synchronisation mechanisms used by the protocol mean that there is no CAN identifier arbitration needed on the bus as all frames are transmitted on the schedule rather than contesting for access to the bus. This results in a 14 – 18% improvement in the achievable bus loading across all bit rates. The reduction in bus loading found at the higher bit rates is due to the fixed processing time and fixed response gap periods becoming proportionally more significant.

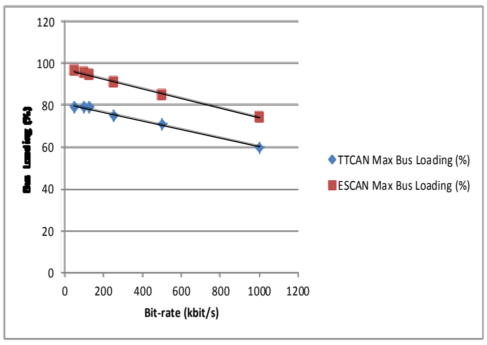

Figure 7 - Comparison of TTCAN and ESCAN Max Bus Loading

Conclusion and further work

The ESCAN scheduling protocol provides a marked improvement in memory utilisation, CPU loading and higher bus loading over the TTCAN level 1 implementation in [1]. Existing higher layer protocols such as J1939 and CANopen could be implemented on top of an ESCAN implementation to take advantage of the higher bus loading capabilities available, these protocols would particularly benefit from the scheduling since the timing parameters

for protocol features such as PDOs in CANopen are generally fixed and can be positioned in to the schedule matrix at fixed intervals where they are guaranteed to be allocated bus access.

The reduction in memory and CPU requirements results in additional resources being freed up for more complex application or alternatively would allow the selection of a lower cost microcontroller part with fewer available resources or permitting the microcontroller to be run at a lower clock

frequency, reducing the power requirements and temperature of the system.

Further work

The current design for dynamic scheduling does not take the timing requirements on the existing messages in to account. For example when the schedule table is extended a message that is currently transmitted every 100ms could end up being transmitted every 150ms as the schedule is expanded and takes longer to cycle around back to the same point.

A mechanism needs to be defined that can maintain these timing constraints and ensure that critical messages are sent at their expected interval.

Analysis of the potential for schedule fragmentation needs to be conducted as well as researching the effect these additional scheduling techniques could have on the stack size and performance of the network given that the primary motivation for ESCAN is a high performance network.

References

1. Quigley, C, Pope B, Finney J, McLaughlin R (2005); “An Automotive Specification of a Time Triggered CAN Implementation: Doubling CAN's usable data throughput”, SAE World Congress paper 2005-01-1539.
2. Kopetz H and Thurner T (1998); “TTP – A New Approach to Solving the Interoperability and Problem of Independently Developed ECUs”, In SAE International Congress and Exposition, Detroit, MI, USA.. SAE Technical Paper 981107, Feb. 1998.
3. Navet N, Song Y, Simonot-Lion F, AND Wilwert C (2005); “Trends in Automotive Communication Systems”, Proceedings of the IEEE, Vol. 93, No. 6, June 2005.
4. Rajnak A and Ramnefors M (2002); “The Volcano Communication Concept”, SAE paper no. 2002-21-0056
5. Tindell K and Burns A (1994); “Guaranteeing message latencies on CAN”, Proceedings of the 1st International CAN Conference.
6. ISO11898-Parts 1-4: (2003) Road Vehicles – Controller Area Network (CAN).

7. Hartwich F, Müller B, Führer T, Hugel R (2000); “CAN Network with Time Triggered Communication”; Proceedings 7th International CAN Conference; 2000; Amsterdam.

8. Williams A., Quigley C., Finney J. (2011); “ESCAN – An Open Source, High Bandwidth, Event Scheduled Controller Area Network”, SAE World Congress paper 2011-01-1041.

Anthony Williams
Technical Director
Warwick Control Technologies
Unit 8 Ladbroke Park
Millers Road
Warwick
CV34 5AN, United Kingdom
anthony@warwickcontrol.com
www.warwick.com