iCC 1996

3rd international CAN Conference

in Paris (France)

Sponsored by

Motorola Semiconductor
National Semiconductor
Philips Semiconductors

Organized by

CAN in Automation (CiA)
international users and manufacturers group
Am Weichselgarten 26
D-91058 Erlangen
Phone +49-9131-69086-0
Fax +49-9131-69086-79
Email:headquarters@can-cia.de
URL: http://www.can-cia.de

CAN in Electric Vehicles

APPLICATION - ELECTRIC DUAL DRIVE SYSTEMS

(GENERIC APPROACH)

Matthew Bailey
Principal Software Engineer
Wavedriver Limited
Melbourn Science Park
Cambridge Road
Melbourn, Royston
Hertfordshire SG8 6TA
United Kingdom

matthewb@wavedriver.co.uk

Abstract

CAN is well-suited to distributed real-time control of electric passenger vehicles. This paper describes the use of CAN to provide a tightly-coupled dual drive system, with real-time control distributed over the two drives. In order to achieve this reliably, the application layer has been extended. In order to maintain the integrity of periodic variables a *Promptness* feature using producer-consumer semantics is available as well as time-out mechanisms on inter-application events. Furthermore, a *Global Data* philosophy has been adapted where all data shared between the applications are tagged with a unique ID. This has many benefits as well as making it easy to monitor the overall system state via a Bus Analyser. The above approach can be used in and extended to a wide variety of performance-critical advanced vehicle applications.

INTRODUCTION TO ELECTRIC VEHICLES AND WAVEDRIVER'S DISTRIBUTED ARCHITECTURE

There is considerable interest world-wide in the use of electric and hybrid-electric powertrains. This interest is stimulated by three main objectives:-

· A substantial reduction in urban pollution, much of which stems from vehicle tailpipe emissions.
· Improved utilisation of scarce global resources through the development of high efficiency vehicles.
· The economic and political advantages of reduced dependence on imported oil.

Electric vehicles (EVs) are mechanically straightforward. They require only a single moving part, the drive motor rotor, upstream of the transmission. Due to the excellent inherent power and torque characteristics of a modern AC system often only a simple single speed reduction transmission is required. An obvious benefit of this is greatly reduced maintenance costs.

Electronically, an AC electric drive is relatively advanced with multi-processor technology controlled by sophisticated software. The use of network topologies, such as CAN, for electric vehicles is therefore both a natural consequence of vehicle system design and easily implemented. CAN is ideal for electric vehicle communications mainly because:-

➢ It provides robust communications and is highly immune to "electrical" noise.
➢ Components are readily available and relatively cheap, compared to the overall cost of a network device, such as an electric drive.
➢ It is fast enough to permit real-time control amongst distributed applications.
➢ It is flexible and easy to use.

A typical contemporary electric vehicle consists of a number of main elements:-

· A traction battery, which provides the main energy source. A wide variety of chemistries are on offer, or under development, ranging from improved versions of Lead Acid batteries through to advanced Lithium systems currently in the trial stage. A Battery Management System (BMS) is required to monitor the modules within the battery and to make small charge adjustments to these modules to keep the battery string in balance.

· A liquid-cooled AC traction drive to convert the DC traction battery voltage into the variable-voltage variable-speed AC waveforms needed to optimally drive the traction motor. An excellent system is required to accurately control the current and magnetic flux vectors in real-time in both the normal and high-speed weak field region of operation.

· A liquid-cooled AC induction or permanent magnet traction motor.

· A traction battery charger or chargers. Various solutions are used based on single or three phase AC mains input. Broadly they divide into three main types:-

    1. Low rating (3kW) single phase on-board chargers.

    2. Higher rate (12 - 100+ kW) single or three phase off board chargers.

    3. Systems such as the Wavedriver, which re-use the AC traction drive as a medium to high rate charger. This has obvious commercial benefits together with the advantage of knowing the usage history of the traction battery, important in its management.

· A DC/DC converter to maintain the charge of the 12 or 24 Volt auxiliary battery required to power conventional vehicle auxiliaries such as the lights, windscreen wipers and in-car entertainment.

Where possible the on-board charger and DC/DC converter will also be liquid cooled.

The network architecture for an electric or hybrid vehicle, shown below, is similar to that of a conventional CAN-based vehicle with different modules on the network reflecting the different power train elements.

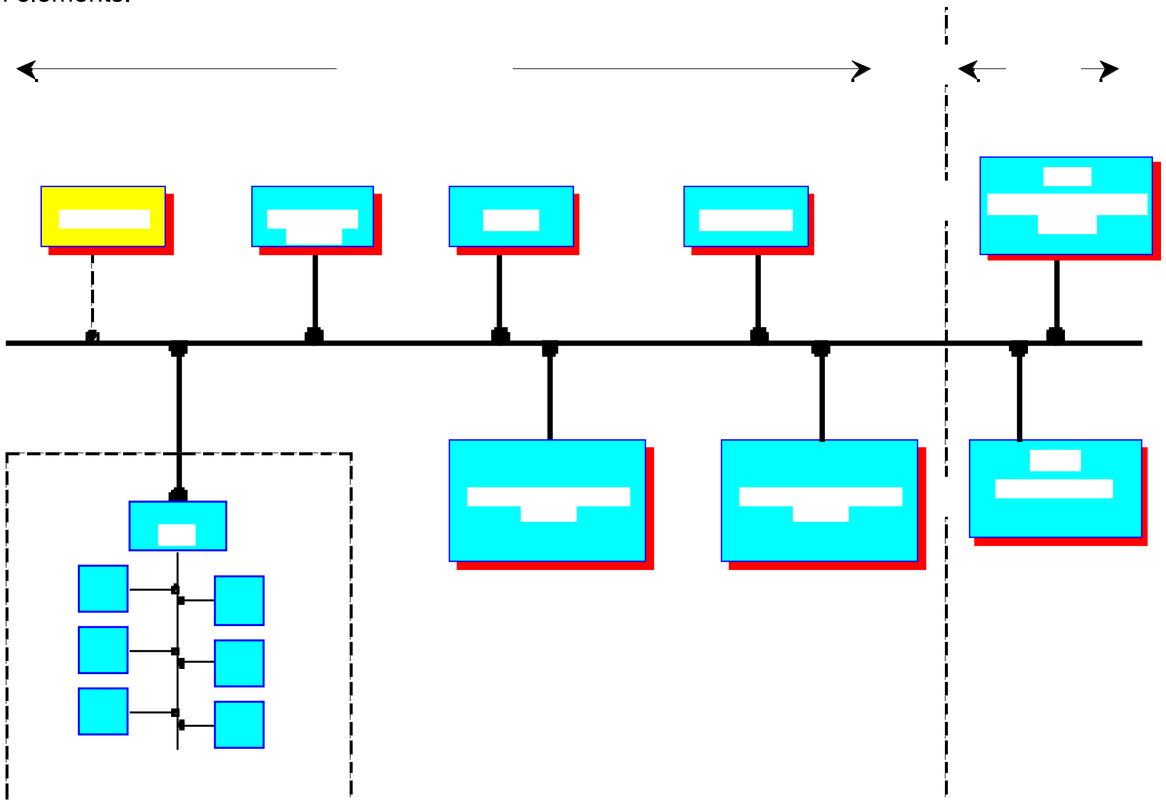

Figure 1 - Wavedriver Electric Vehicle Network Architecture

It can be seen how a pure electric vehicle can be relatively easily extended to hybrid operation if this is planned into the system design.

OUTLINE OF ELECTRIC DUAL DRIVE APPLICATION - MOTORING

Introduction and Scope

This paper discusses the implementation of an EV requiring 80 kW of power. The redundant system requires the use of a generic Dual System model, consisting of two electric drives. Depending on the vehicle configuration each drive can either be controlling a motor on an axle (i.e. front and rear drive), or be combined to drive the same axle. This paper discusses an EV configured with a drive and motor per axle.

An EV has three basic modes of operation, Motoring (driving), Park, and Charging. This paper only deals with inter-drive communication in **Motoring** mode.

The scope of the discussion only relates to interactions within the generic Dual System, i.e. between the two drives, and does not include other distributed applications which can be on an EV CAN, such as a traction battery, or vehicle I/O. However, the same principles to be discussed apply to interactions between all applications on a control network, namely "Promptness" and "Message Exchange" time-outs.

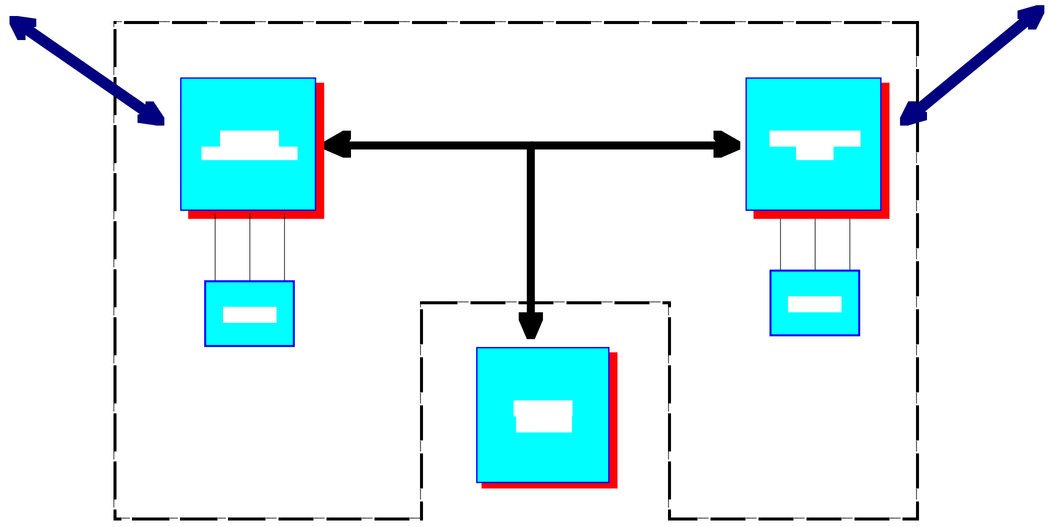

Figure 2 - Generic Dual Drive System (Speed Control) - Overview Diagram

Dual System - Brief Overview

The above diagram illustrates, in block diagram format, the structure of the distributed Dual System application. An EV requiring two or more drives may operate under synchronised speed control or torque control. In either case, similar types of information must be exchanged between drives. For this paper, an example of an EV being controlled by predictive speed control will be discussed.

The generic Dual System shown above receives external control inputs as shown. External commands allow a user (or intelligent controller) to:-

- Configure a drive - as Master or Slave.
- Modify drive parameters.
- Change the operational mode of the Dual System (via the Master).
- Control vehicle speed (via the Master).

The external Dual System environment receives diagnostic messages from each drive indicating its own status and the overall status of the Dual System.

As shown above, a high speed CAN inter-connects the Master and Slave drive. Different types of information must be exchanged for Dual System operation, these are:-

- drive commands and responses - for control of Dual System operating modes.
- statuses and diagnostics - to maintain Dual System integrity.
- predictive speed control - to control the vehicle speed.

The nature of this information causes the drives to communicate in two distinct manners, namely **confirmed events** in the form of commands/responses and **periodic** information, either fast (_30ms) for speed control or slow (_100ms) for maintaining system integrity.

Further details about inter-drive communications for Dual System operation can be found in the Appendix.

GENERIC DUAL SYSTEM APPLICATION MODEL AND GLOBAL DATA PHILOSOPHY

To aid understanding of how the drive applications interact, the generic application model for the Dual System (from the communication point of view) is shown below. (Note that the maximum amount of data for each type of information to be exchanged is designed not to exceed 7 bytes thus only CAN messages are needed).

APPLICATION MODEL

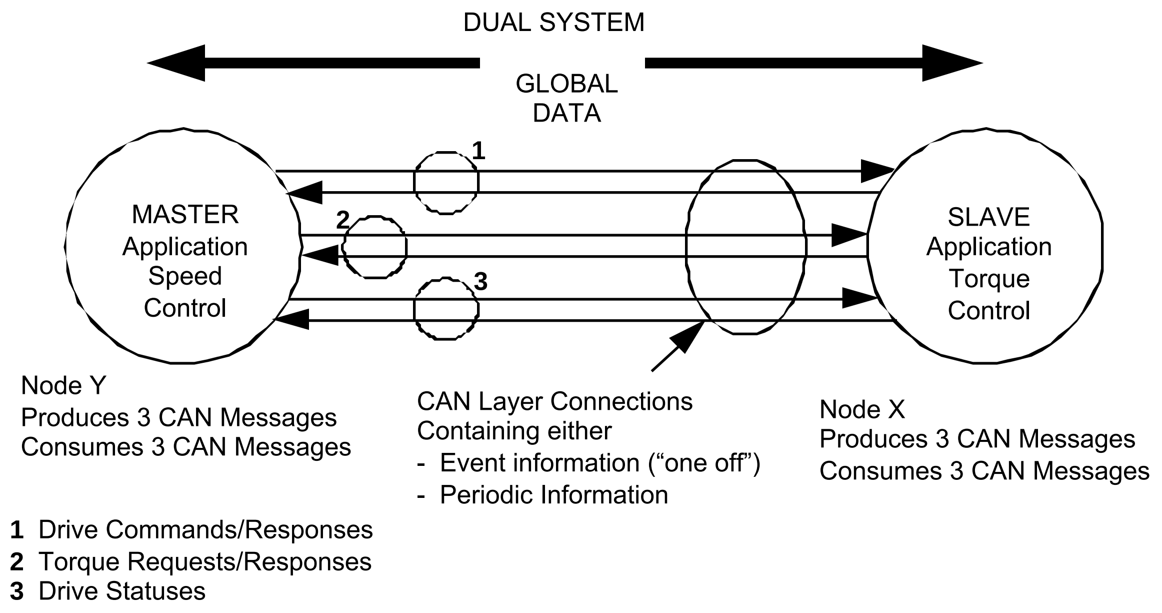

Figure 3 - Application Model

Abstracting the communications as a series of connections for transmitting/receiving different types of information provides focus on the interaction between the drive applications. As a consequence, details about the implementation of the communications are no longer needed.

The approach adopted is that of common data, referred to as Global Data, which is shared (via the CAN) by the two drive applications. The Global Data list contains items of data required by a drive application to operate within the Dual System. Each Global Data item has a unique tag enabling an application to identify information being sent or received on a connection. (Note: to maintain Dual System integrity, a drive application must check that it is receiving valid data, i.e. A Master receiving drive commands means that there are two Masters !).

The generic Dual System requires 6 different **types** of Global Data for the Master and Slave drive applications:-

- Drive Commands
- Drive Responses
- Torque Request
- Torque Response
- Master Drive Status
- Slave Drive Status

The system wide concept of Global Data has some distinct advantages such as:-

- Forces, at the system functional/design stage, a list of the system wide data needed to be exchanged between applications. This provides useful focus for system design.
- More importantly, abstracts a distributed application from the communication details (but not from the timing requirements on the data).

As shown above, each drive application requires six CAN Layer connections (three outgoing, three incoming) for the exchange of each type of Global Data. Global Data types are mapped onto the outgoing and incoming connections according to whether the host drive is configured as Master or Slave. A connection is only uni-directional and exists by the transmission and reception of a CAN Message ID. Global Data will be sent/received via one of these connections, according to its type. Due to the nature at which each type of information must be exchanged - as mentioned above, it is more robust (and simpler) to have a dedicated connection per Global Data type.

For each drive on the CAN, a Producing/Consuming Node approach has been adopted. Each node has a unique number and has a fixed list of CAN Layer Messages that it Produces (Transmits) and Consumes (Receives). The table below provides further detail.

	Master Drive	**Slave Drive**
Node ID	**1 - if Front, 2 - if Rear**	**1 - if Front, 2 - if Rear**
Global Data Type		
Drive Command	**Produce**	**Consume**
Drive Response	**Consume**	**Produce**
Torque Request	**Produce**	**Consume**
Torque Response	**Consume**	**Produce**
Master Drive Status	**Produces**	**Consume**
Slave Drive Status	**Consume**	**Produce**

Table 1 - Producer/Consumer Details

Each CAN Message ID is encoded to indicate the producing node number (either 1 or 2) and the type of application connection.

For example, Node 1 produces:

CAN ID 0x0005	/* 000000001 01 */	Node 1, Drive Command/Response connection
CAN ID 0x0009	/* 000000010 01 */	Node 1, Drive Torque Request/Response connection
CAN ID 0x000d	/* 000000011 01 */	Node 1, Master/Slave Status connection

Node 1 Consumes:

CAN ID 0x0006	/* 000000001 10 */	Node 2, Drive Command/Response connection
CAN ID 0x000a	/* 000000010 10 */	Node 2, Drive Torque Request/Response connection
CAN ID 0x000e	/* 000000011 10 */	Node 2, Master/Slave Status connection

This logical approach of encoding information into the message ID enables a drive application to identify the type of connection. However, further encoding is required to **uniquely** identify the Global Data item within a message. This is because any one of the two drives can be Master or Slave, thus the drive application on node 1 could produce drive commands or responses on the same connection - depending on its configuration.

Thus, for a drive application to recognise an item of Global Data within a message (on a connection), the tag identifying the Global Data item is sent together with the data. The tag is located in the first byte of a message. The following diagram provides and illustration:

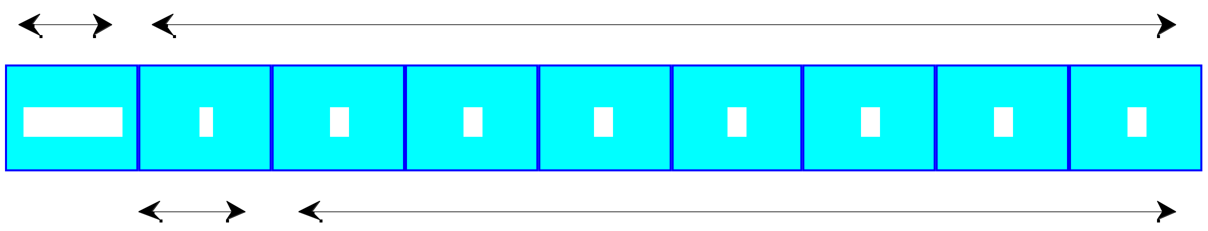

Figure 4 - Typical CAN Message Exchanged Between Drives

Note: The above approach of tagging data in the first byte is not adopted for all Wavedriver applications. It is quite common when interfacing to engine control units and traction battery applications on CAN that Global Data types are identified via fixed message IDs.

Finally, the above approach for the Application Model and Global Data provides an added benefit that a maintenance person can determine the exact status of the Dual System by monitoring inter-drive communications (via a bus analyser).

IMPORTANT APPLICATION COMMUNICATION ISSUES

Introduction

Although the CAN profile indicates if another device has consumed a message it does not contain the facility to monitor the "timings" of communications between devices (or rather applications), which is fundamental to distributed real-time control. The following two issues illustrate this point further.

Issue 1 - Message Exchange (Associated Events Initiated by Producing Node)

The system requirements specify the time for the Dual System to change into certain modes, so that it behaves in a manner acceptable to the customer.

For example, the Master has been commanded to change the Dual System state, the result of which is to firstly change the operating state of the Slave. It creates the appropriate drive command, encodes it into a CAN message and transfers the message to the Slave. The Master waits for the Slave to respond. What if the Slave never responds ? What if the Slave responds slower than is acceptable to the customer. The Master has no communication function available in the CAN profile to handle the Slave responding slowly or even not at all.

Issue 2 - Promptness (Node Consuming Periodic Information)

For the speed controller to work in a manner acceptable to the customer, it is fundamental that the exchanges of torque demands and responses circulate at 30ms intervals - with 60ms interval the acceptable worst case. Furthermore, to maintain the integrity of the Dual System so that events can be safely responded to in an acceptable time, a drive must receive the partner drive's statuses and diagnostics on a regular basis, with a worst case of 200ms.

For example, the Slave is expecting torque demands from the Master at 30ms intervals, what if the Master starts sending them at 100ms. The Slave cannot know that the speed control has degraded to an unacceptable level. Or, the Slave responds too slowly to a serious event because it is receiving Master statuses at 500ms, maybe the Master indicates that there is a fault and the Dual System must stop, meaning that the Slave must apply a zero torque to the motor. The CAN profile does not provide functionality to cater for these scenarios.

Conclusion

The way to ensure that the above scenarios are detected is to use two timing mechanisms and these will be discussed in the following section.

SOLUTIONS TO APPLICATION COMMUNICATION ISSUES

Introduction to "Promptness" and "Freshness" Terms

The Promptness functionality is not a new one. It is part of the FIP standard, see ref [1]. The general philosophy is to monitor the periodicity at which a network variable (equivalent to a CAN message), containing data required by the host application, is being received and if it is at an acceptable rate.

The maximum periodic rate that a variable can degrade to, is configurable and is determined by the nature of the data.

At this point it is useful to mention that the FIP protocol also has a notion of "Freshness" for periodic data. This status indicates how recent or "Fresh" the data content is within a variable. This status is attached to the variable. The "Freshness" mechanism is required because periodic variables are continually transmitted on the network, regardless of the "transmitting" application updating their data content. Thus, this mechanism permits the host application to check that the data within a variable has been updated ("Refreshed") by the transmitting application within an acceptable time.

However, in the case of CAN, message transfers can be fully controlled by the host application. This means that when a message is transmitted by the host, the message contents can be updated first with the latest data and then transmitted, thus when a node receives the message it knows that the data is the latest (most "Fresh") from the remote application. Thus, the notion of "Freshness" can be an inherent part of the CAN protocol.

"Promptness" and "Message Exchange" Features in a Comms Stack

It could be argued that the time-out features mentioned are purely a Data Link or Transport issue. However, Wavedriver adopts the view that these features are an application issue and has consequently decided to **abstract** the services from the type of network and put these features in the **Application Layer** associated with the **Global Data**. The advantages to this philosophy are:-

· The time-out features are **network type independent**, so these features could be used on other control networks.
· The time-out features are **Higher Layer protocol independent**, so one could use the Promptness feature with a periodic CAL reception or periodic Transport reception.
· It **Focuses** the system designer on the application issues and not on the communications, e.g. concern about Global Data exchanges and not on message exchanges.
· The application software is more **portable**, as these features are associated with the application and not with the communications of the host device.
· If a CAN message contains several types of Global Data, then **different** time-out mechanisms can be used for each Global Data item in the message - giving greater scope to the application.

Whichever approach one takes, somewhere in the communications stack these time-out features are needed. Communications in a distributed application are going to contain the exchange of "combined one off events" and "periodic" information. Below is the flexible stack model adopted by Wavedriver.

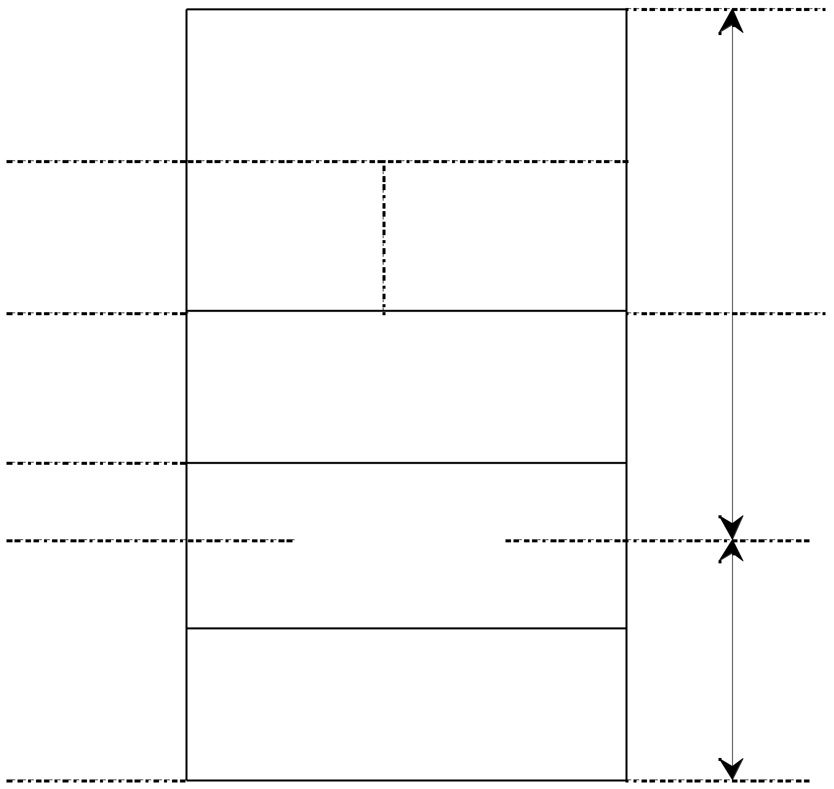

Figure 5 - Wavedriver CAN Comms Stack

The Intel 82527 and a line driver provide the Physical Layer and most of the Data Link Layer. A set of software services, located on the host processor, complete the Data Link layer. These services are for management of:

- Intel 82527 Device (initialisation, configuration, minor network management etc...)
- Message Objects (configuration, transmission services, reception services)

The Higher Layers are not mandatory and are used according to the protocols used by the remote applications that the host application is interacting with. The host application is not forced to use the Higher Layers and can select which one to use, e.g. for the generic Dual System application no Higher Layers are required, but on a Power Management network Data Logging would be required where much information must be exchanged together via the CAN, and thus a Transport Layer used (see ref [2]), or in a Heavy Vehicle application J1939 would be used. The Higher Layers interface to the set of software services in the Data Link Layer.

Example of Message Exchange Time-out

Using the same scenario as above of a message exchange where the Master commands the Slave to change its drive operating state. The Master expects the response within 1.5 seconds.

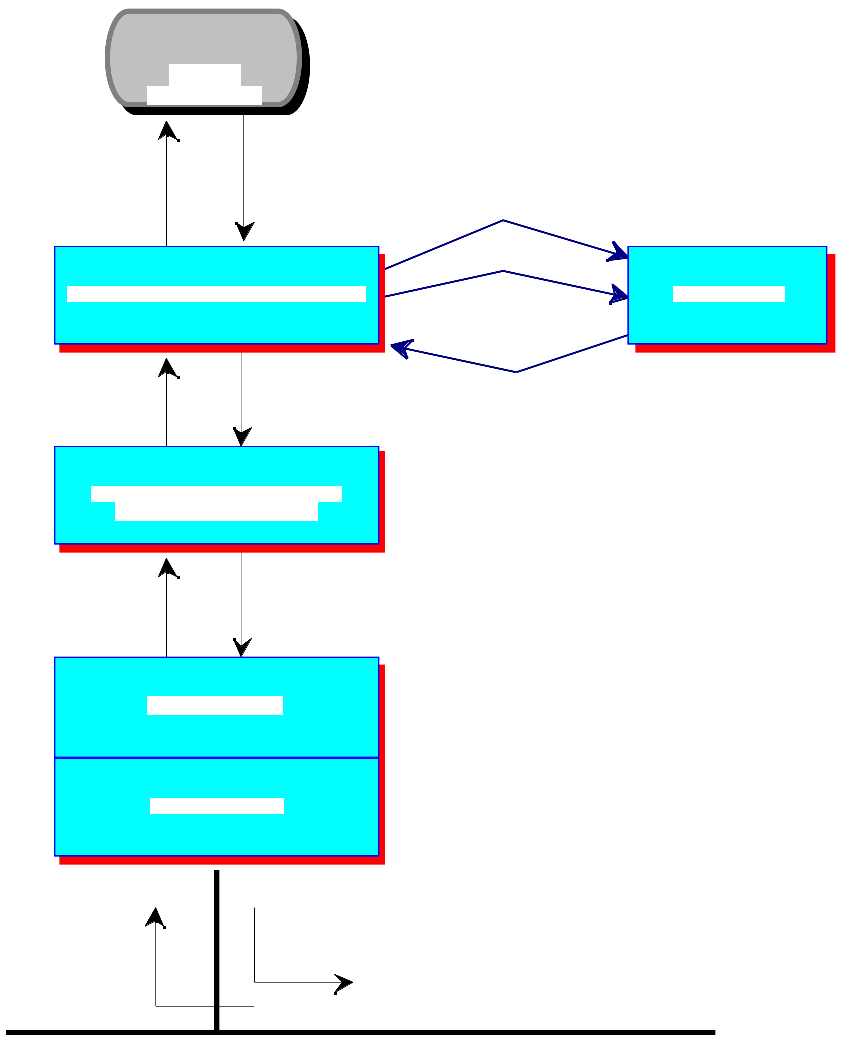

Figure 6 - Message Exchange - Drive/Command Response

The above diagram illustrates the possible outcomes of this transaction:

1. Master application creates a Global Data item of type drive command for the Slave and sends it to the "Message Exchange" time-out partition in the Application Layer.

2. A software timer is started of 1.5 seconds duration.

3. The Global Data item is encoded into a CAN message.

4. The CAN message is transmitted to the Slave.

5. The next event will be either, the Master receives a CAN message from the Slave or the software timer expires.

6. If the software timer expires then:

   - A Global Data item is created of type drive response - indicating that the Slave failed to respond in time.
   - Any late response from the Slave is ignored.

- ➢ The application may then send an unconfirmed Global Data item forcing the Slave to go to IDLE.
- ➢ The Master carries on alone.

7. If the Slave responds in time then:

   - ➢ The CAN message is decoded into Global Data.
   - ➢ The software timer is aborted (providing the Global Data item is valid, i.e. a response).
   - ➢ The Global Data is passed onto the Master application - indicating that the Slave responded in time.

Example of use of Promptness

Using the above example of the Slave expecting Torque Requests from the Master periodically at 30ms, with 60ms as the worst case.

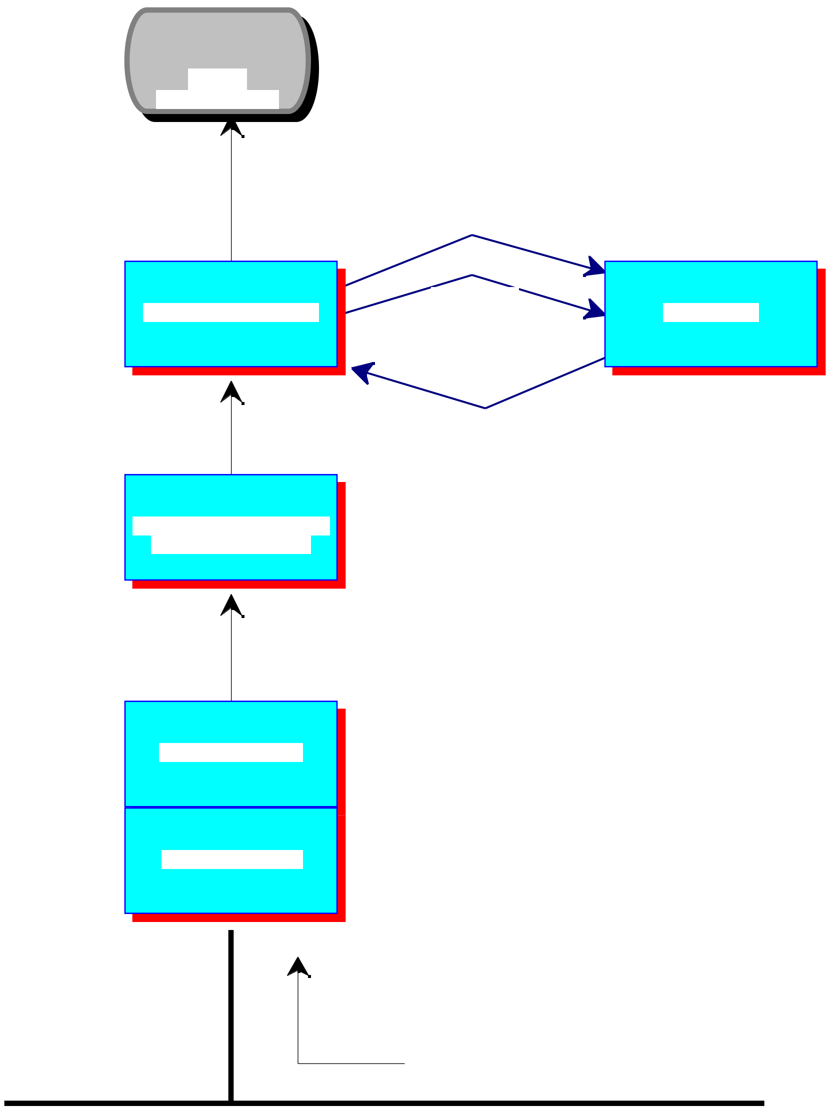

Figure 7 - Promptness - Torque Demand From Master

1. A CAN Message is received containing a torque demand.

2. This is decoded into Global Data.

3. A software timer is then started for a 60ms duration and the Global Data is passed onto the Slave Application - with an indication of no time out.

4. The next event is that either another CAN Message is received containing a torque demand or the software timer expires.

5. If another CAN Message is received and after decoding it contains a torque demand then:

   - The software timer is restarted.
   - The torque demand is passed on to the Slave application - with an indication of no time-out.
   - Repeat 4.

6. If the software timer expires then:

   - A Global Data item of type torque demand is created with an indication that a time-out has occurred.
   - This is then passed on to the Slave application to take appropriate action.
   - All future torque demands are ignored.

SUMMARY and CONCLUSIONS

This paper has discussed an Electric Dual Drive System and has shown that CAN, although providing cheap and robust communication between devices, lacks certain fundamental timing facilities needed by distributed applications for real-time control. Two types of communication scenarios illustrate these timing issues; the first is where an application sends out a confirmed request and requires a response indicating the outcome within a defined time, the second is where an application consumes periodic data which must be received above a defined rate to maintain system integrity.

Fortunately, due to the nature of the CAN profile, it is possible to "add" functionality to the CAN communications stack to meet the above timing requirements. This functionality could be placed almost anywhere in a communications stack, but is ideally abstracted and placed into the Application Layer. This approach has distinct advantages, the main ones being that the time-out features become independent of the network type, resulting in portability/re-use of the application, and are Higher Layer protocol independent thus adding necessary functionality.

The timing functionality is made available in the form of "Message Exchange" and "Promptness" services to the application. The "Message Exchange" service permits the application to know if a response to a request has been returned in a specified time. The "Promptness" service permits the application to know if periodic data is being received at an acceptable rate.

This paper also details an Application Model and Global Data approach used for inter-application communications. These approaches are useful as they abstract the interaction between the drive applications away from unnecessary communication details, but not from timing criteria. One further benefit pointed out is that, using a bus analyser monitoring inter-drive communications, it is straightforward to find out the statuses of the Dual System and its constituent elements.

APPENDIX - Details of Inter-Drive Communications for Dual System Operation

Operating Modes for Dual System and Dual System Drive

The operation of the Dual System and each drive are modelled in the form of State Transition Diagrams shown below:

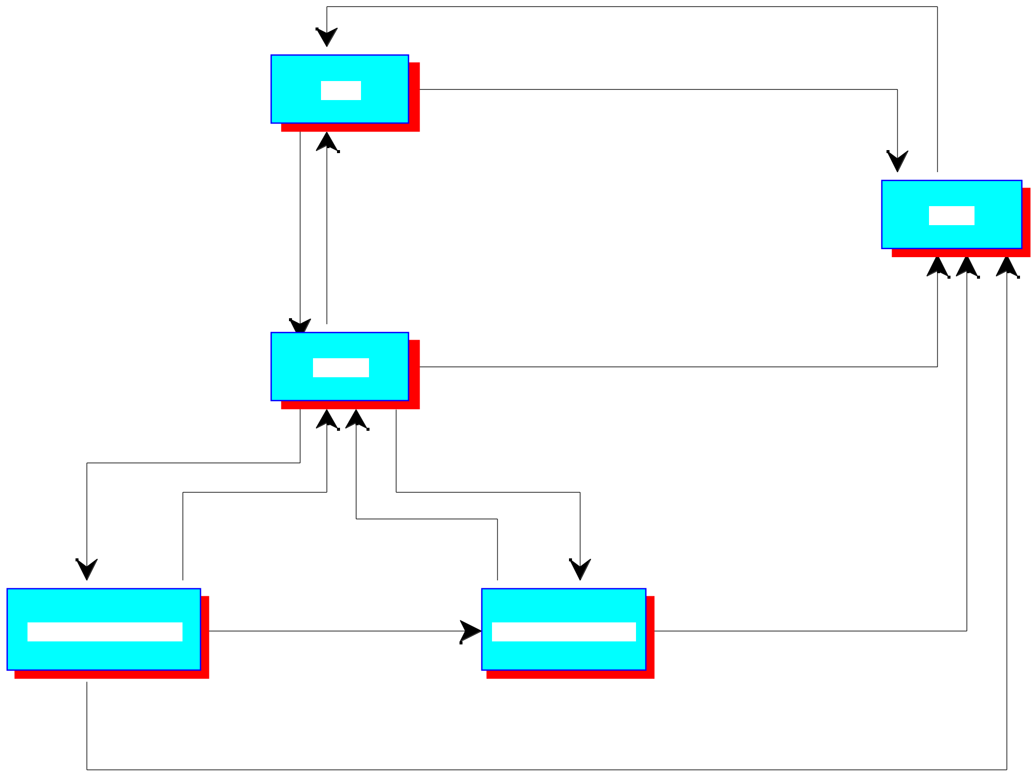

Figure A1 - Dual System Operating Modes

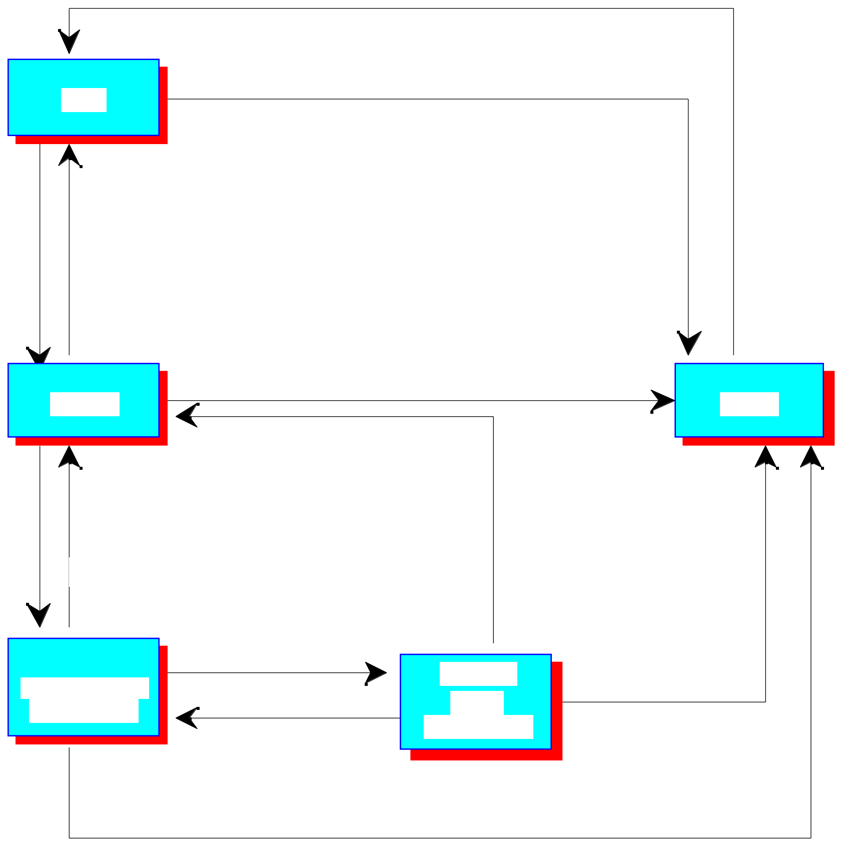

Figure A2 - Dual System Drive Operating Modes

In simple terms, the combined Operational States of the drives determine the overall Operational State of the Dual System. e.g. If both Drives are in any of the ACTIVE states then the Dual System is also in the ACTIVE Operational state, or if the Slave drive moves to the FAILED state then the Dual System moves to the ACTIVE Degraded state.

The external environment (as shown in figure 2 of the main text) controls the Dual System by sending desired operational states to the Master, who internally translates the command into desired drive operational states. A command is then sent to the Slave to "move" to the desired state, which then sends a reply confirming or not confirming the transition. After this the Master itself moves to the desired drive state, the combined drive states then determine the Dual System state.

Dual System Integrity

Whilst the Dual System is not idle its integrity must be maintained. This is done by the Master and Slave monitoring statuses and events occurring (in the Dual System) on the host as well as the partner drive. To do this each drive periodically exchanges detailed information with the other. This information is then checked by the receiver and appropriate actions taken. A brief list of the information exchanged is:

- Application heartbeat
- Dual System Operating State
- Drive Operating State
- Statuses and Diagnostics - statuses and reasons for faults, e.g. Slave - CAN failure, or other information such as auxiliary battery voltage is low.

Predictive Speed Control

The speed control function is distributed between the Master and Slave drives. The Master receives commands for the desired Dual System speed. The Master calculates the reference torques for the front and rear drives (taking into account many parameters such as current speed) which when applied will cause the vehicle to move at the desired speed. The appropriate torque demand is sent to the Slave according to it being front or rear. The Slave creates a response indicating current speed of the motor that it is controlling and confirmation of the last torque demand received. This is then sent to the speed controller on the Master.

The scheduling of the communications is such that torque demands and torque responses must circulate between the Master and Slave periodically. Typically the nominal period for communications will be in the range 20-50 ms, with performance degradation occurring if the cycle time rises above two times the nominal.

REFERENCES

[1] NORME FRANCAISE NF C 46-601 to C 46-607, "Bus FIP pour exchange d'information entre transmetteurs, actionneurs et automates", AFNOR (1990).

[2] Multiplex Technology Applications in Vehicle Electrical Systems, SAE SP-954. "Network Architecture for CAN", Jorg, Kaiser and Unruh.

MJB/njr/12329