CAN-based distributed real-time control in hybrid electric vehicles

Renji V Chacko, Dr. Z V Lakaparampil, Chandrasekar.V, Sigi C Joseph

Centre for Development of Advanced Computing, Thiruvananthapuram

A Scientific Society of the Ministry of Communication and Information Technology, Govt. of India

The implementation of an efficient real time networking with *Controller Area Network* (CAN) for distributed real time control in *Hybrid Electric Vehicle* (HEV) is presented. HEVs are the present potential choice for environmental friendly public transportation systems. Their safety and functionality can be improved and many value added features can be easily incorporated with CAN based distributed control. In HEV different electrical systems are functionally interconnected, requiring exchange of information accurately in real time within defined communication latency. The CAN controller reduces communications burden on the host CPU, thus allows to run its algorithms for better real time power train control. The differential physical layer improves data exchange integrity under EMI from power switching systems in this application. The implementation methodologies for TI *TMS320F2406* Digital Signal Processor and *PIC18F4480* Micro controller based hardware with inbuilt CAN controllers are highlighted. An efficient mailbox filter configuration and message distribution scheme in different control modules in HEV as well as state machine for minimum boot-up process for *CANopen* protocol in master and slave nodes are detailed.

I Introduction

The implementation of an efficient real time networking with high level of data accuracy, low communication latency, configuration flexibility etc, is vital part in distributed real time system design and *Controller Area Network* (CAN) protocol supports this in a great deal. The CAN protocol has found wide acceptance in automotive in-vehicle applications due to its high performance, low cost and configuration flexibility. Though it was developed for use in passenger cars, it is widely used in other automotive sectors and also by manufactures of a wide range of products. The *Hybrid Electric Vehicle* (HEV) is one of the applications that demand the distributed real time control [8][9].

Due to environmental and fuel resource concerns, there is a growing demand for *Electric Vehicles* (EVs). The EV powered solely from the onboard battery is an ideal zero emission vehicle, but can run only for a limited range on a single charge. This limitation is overcome in a HEV by the introduction of a down sized IC Engine (ICE) for the average power demand and supplemented by the Battery for transient requirements. As HEVs combine the good features of both ICE and EVs, the emissions are reduced and fuel efficiency is increased due to the optimal operation of the ICE. When compared to conventional ICE powered vehicles, HEVs eliminate the need for clutch and variable gear. During vehicle deceleration energy can be recovered to the battery with regenerative braking control extending the range and increasing the brake liner life [9].

Two implementations namely *series HEV* and *parallel HEV* are possible based on the configuration of the two power paths[8]. This paper explains the control and

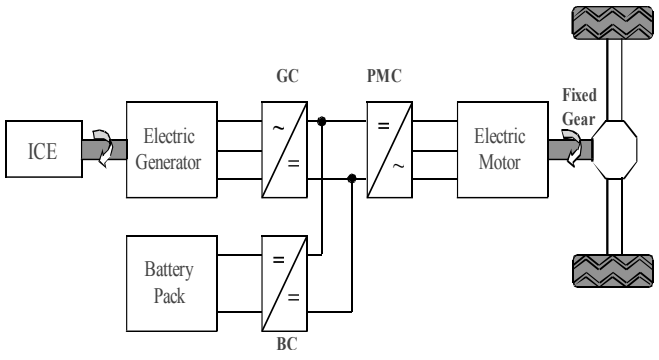


Figure 1: Functional block diagram of series HEV

communication schemes for a series HEV in distributed control architecture with CAN for a Three-wheeler application. This development work was carried out under the auspices of Ministry of Communication and Information Technology, Govt. of India.

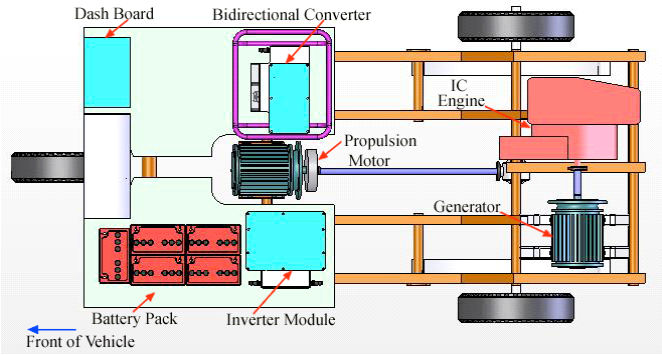


Figure 2: Distribution of series HEV modules in three wheeler

2 Series HEV control blocks

The series HEV functional blocks are shown in Fig1. In this scheme, the ICE power is converted to dc electric power with the help of an Induction Generator and *Generator Controller* (GC). The Battery is linked to the dc bus through a *Bidirectional Converter* (BC) which controls the power flow to and from the Battery. The power is fed to an Induction Motor through a Propulsion Motor Controller (PMC), which drives the wheels through a fixed gear to match the vehicle speed and torque. The ICE operation is optimized by running at narrower speed range and slow application of the load since it is less dependent on the changing vehicle power demand.

In a typical *Three wheeler HEV* implementation the control blocks and sub systems are distributed in the vehicle as shown in the Fig 2. This layout is formulated considering the load distribution and vehicle stability. As the control blocks are functionally unique they are implemented in separate hardware modules though PMC and GC are combined in the *Inverter Module* for better electrical packaging. A *Dash Board* (DB) module provides power flow control and user interfaces. The AC system based propulsion and generation schemes are developed considering the advantages over DC systems for this automotive application.

2.1 Control requirements

The functional requirements of the subsystems, which are basically power electronic modules are summarized in table I. Advanced propulsion technology and development of power electronics assumes significant importance in HEV control. These include AC drives with real time torque controller for propulsion and power controller for generation, power flow manager, compact and rugged induction motors, low maintenance battery and its monitoring system etc. Digital Signal Processor (DSP) based real time controllers play a key role in effective system operation. It provides efficient implementation of advanced AC drive control algorithms for the propulsion and generation. TI DSPs integrate many peripherals like ADCs, encoder interface, PWM generators, etc that are required for power electronics application [11]. More I/O optimized Microchip's PIC micro controllers are ideal for modules like DB, where more user interface are required rather than real time control [12]. Further these processors have got support for serial communication interface required in this kind of distributed control which are detailed in the next section.

Table I Functional requirements of HEV modules

Module	Functional Requirements
ICE	Supply average traction demand
Induction Generator	As starter motor during ICE starting As a generator during normal operation
GC	Two quadrant AC drive (motoring, generator)
Battery	Supply transient demand and absorb regenerative energy
BC	Bidirectional dc-dc converter to match battery voltage and dc voltage
PMC	4 quadrant AC drive
Induction Motor	Meet traction demand in forward and reverse As a generator for regeneration control
DB	Supervisory control and user interface

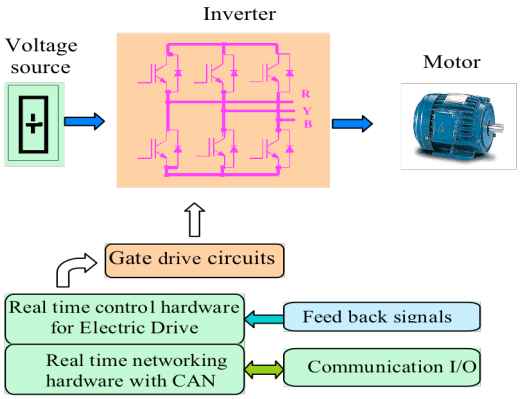


Figure 3: Components of a AC drive controller module

2.2 Data transfer requirements in series HEV

To estimate the data transfer and control requirements, deeper knowledge of power electronics system functionality and operational modes with respect to vehicle operation are essential.

The Voltage Source Inverter (VSI) scheme for AC drive control in GC and PMC, is functionally detailed in the Fig 3. The Inverter produces variable frequency ac voltage from the dc voltage source and vice versa. The control block requires high performance algorithms for effective AC drive control [10].

In this three wheeler implementation the propulsion motor requires variable frequency inputs in the range 0 to 200Hz to control the vehicle speed from 0 to 50kmph. The AC drive control algorithm based on Space Vector Control (SVC) and PWM technique for sensorless operation requires sampling time of the order of 200 microseconds for effective implementation in this frequency range. The feedback input requirement in the AC drive algorithm are motor currents, dc voltage and frequency reference. For effective torque response the motor current has to be sensed locally at every sampling period. Hence its interface is provided inside the Inverter module. The other control inputs like frequency reference, logic controls etc can be interfaced from other control modules through serial communication at a slower rate. Similarly the variables like actual motor speed, propulsion power requirement, inverter status can be sent to other control modules. Thus the control block has to perform two tasks namely the real time control and the real time communication thereby having significance for local and global control parameters.

In addition to the local control, the control algorithms implemented in the subsystems require real time parametric data transfer to take care the various modes of operation in series HEV as detailed in the table II.

During start up the ICE is cranked by operating induction generator as motor through the BC and GC. The heavy power requirement during acceleration is met with both Battery and ICE source. In steady cruising mode the propulsion demand is met from ICE alone and depending on battery condition ICE also charges the battery [5]. During deceleration the brake energy is regenerated and stored into the Battery through BC. The rate of regeneration is controlled monitoring the

TABLE II Power flow in Series HEV modes of operation

Mode of Operation	Power Sources	Power Destination	Controller in action
ICE start up	Battery	Induction Generator	BC + GC
Vehicle Acceleration	Battery + ICE	Propulsion Motor	BC+ GC + PMC
Steady cruising	ICE	Propulsion Motor (+ Battery)	GC + PMC (+BC)
Regenerative braking Vehicle Deceleration	Propulsion Motor	Battery	PMC + BC

TABLE III Data exchange requirement

Parametric Data Exchange	Source	Destination
Battery Voltage(V_{bat})	BC	PMC,GC,DB
DC link Voltage (V_{dc})	BC	PMC, GC,DB
ICE speed (N_{ice})	GC	DB
Acceleration Pedal reading (N_{ref})	DB	PMC
Propulsion motor speed (N_{pmc})	PMC	DB
Battery state of charge (B_{soc})	BC	DB
Propulsion Controller Power (P_{pmc})	PMC	DB
Generator Power (P_{gc})	GC	DB, BC
Generator Power reference (P_{ref})	PMC	GC
Probe Data (ProDat1,…ProDat9)	PMC,GC, BC	DB
Logical Data Exchange(*)		
Status of Controllers	PMC, GC,BC	DB
ICE start command (GC_{st})	DB	GC
PMC start/stop command (PMC_{st})	DB	PMC
Emergency OFF	DB	PMC,GC, BC
Forward/Reverse command	DB	PMC

(*) Logical data for each controller is in bit level and is combined into a single status byte for data transfer.

battery state of charge. Table II shows different functional modules that are involved in performing each mode of operation and the required parametric data exchange estimated in this HEV scheme are listed in table III.

2.3 Controller Area Networking in HEV

The implementation of an efficient real time networking with high level of data accuracy, low communication latency, configuration flexibility etc, is vital part in distributed real time system design and Controller Area Network (CAN) protocol supports this in a great deal. The CAN protocol specifies versatile message identifiers that can be mapped to specific control information categories with predefined priority [1]. The following features are worth mentioning in real time application.

Message Routing: The content of a message is named by an IDENTIFIER. The IDENTIFIER does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide by MESSAGE FILTERING whether the data is to be accepted. The non-destructive arbitration technique guarantees that messages are sent in order of priority and that no messages are lost.

Multicast: As a consequence of the concept of MESSAGE FILTERING any number of nodes can receive and simultaneously act upon the same message.

Message/Data Consistency: Within a CAN network it is guaranteed that a message is simultaneously accepted either by all nodes or by no node. If a receiver detects an error in the last bit that it cares about (the last but one bit of the EOF) it will send an error frame. This will corrupt the last bit that the transmitter cares (the last bit of the EOF) and will retransmit the message. Thus data consistency of a system is achieved by the concepts of multicast and error handling.

Fault Confinement: CAN nodes are able to distinguish short disturbances from permanent failures. Defective nodes are switched off. Thus the network can still be effective with the healthy nodes.

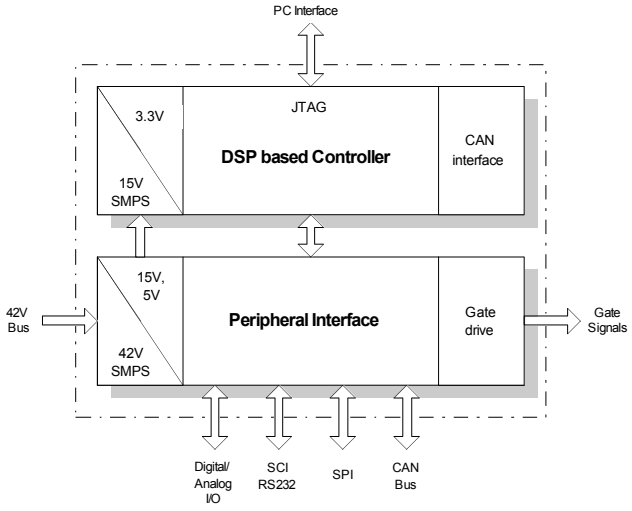


Figure 4: Control hardware block diagram

Robustness: CAN will operate in extremely harsh environments and the extensive error checking mechanisms ensure that any transmission errors are detected. The data exchange integrity under EMI from high power switching systems in applications like HEV is very important and the physical layer with differential busses take care these issues. The use of NRZ encoding ensures compact messages with a minimum number of transitions and high resilience to external disturbance.

Thus with prioritized communication of message, non-destructive bit-wise arbitration with effective error detection signaling, CAN efficiently supports distributed real-time control with a very high level of security in automotive applications.

2.4 Control hardware implementation

The digital control hardware design take care the real time control task as well as support for high speed CAN interface (ISO11898) [2]. TMS 320LF2406A DSP for drive control and PIC based control for I/O intensive are found to be an ideal choices. The power distribution in control hardware, which complains to 42V Power Net standard and communication software based on CiA standards has facilitated easy addition of CAN based modules for added features. The control hardware structure shown in Fig 4 is implemented for better performance in most power electronic systems.

Similar digital controller hardware is used for propulsion control, generator control and other subsystems with appropriate software modules to meet the specific application.

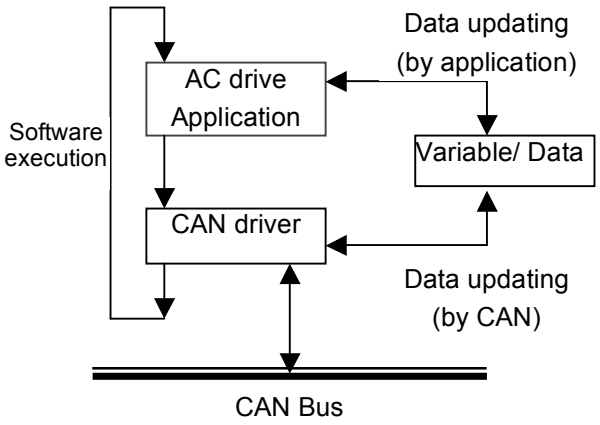


Figure 5: Software structure for real time control and communication

2.5 Control software structure

As described in the system requirements, the two main tasks to be implemented in software are, the real time control and the real time networking. As shown in Fig 5, a simpler scheduler based on timer interrupt is considered for implementation.

- *Real Time Control task*

Depending on the type of sub module, the control software implemented are,

- AC Drive control for Propulsion motor
- AC Drive control for Generator
- Control for Battery charging
- Control for power flow management
- Control Algorithms for relay logic

- *Real Time Networking task*

The networking module implements the algorithms for CAN interface conforming to CiA standards. With CAN baud rate of 1Mbps, real time data transfer between the various modules can be achieved with in the required rate for this application. The network level synchronization can be also be easily implemented by the application protocol support. The coming section details the implementation of the application layer protocol for the data transfer requirements estimated for HEV control [6].

3 CANopen in series HEV

The distributed control architecture involving the use of the CAN-bus envisaged for the HEV system is typically of the "master/slave monitoring and control" type. Every subsystem is monitored or controlled from one central node (the master). The CANopen application protocol is ideal in this type of environment [4]. The types of devices to be controlled are AC drive systems, dc-dc converters and similar power electronic systems. Many of the network control tasks in HEV basically consist of monitoring and supervisory control of the sub modules through parametric and logic data. Hence for simpler software implementation, the control module are treated as I/O modules instead of the usual convention of drives and motion control for the device profile [5] [7]. It can also be observed that master-slave as well as slave-slave data transfer are required.

TABLE V PDO mapping for TxPDO1(SYNC)

Control Node	PDO format		
PMC	P_{pmc} (*)	N_{pmc} (*)	PMC status (**)
GC	P_{gc} (*)	N_{ice} (*)	GC status (**)
BC	V_{bac} (*)	V_{dc} (**)	BC status (**)
DB	N_{ref} (*)	P_{ref} (*)	DB status (**)
(**) One byte data (*) Two byte data			

3.1 CANopen requirements

The following general requirements for *CANopen* based control applications in HEV can be distinguished:

The *CANopen* Predefined Connection Set can be used, although for some nodes it might be necessary to allow slave to slave communication through Process Data Object (PDO) linking.

Node guarding not necessary since node data is read out periodically and missing responses can serve to detect fatal errors in the node communication. However if any internal device errors have to be detected the use of the *Emergency Object* is required.

Identifier distribution by means of SDO is sufficient; *CAL* DBT services and thus the *extended boot-up* capability are not required (here default id distribution is assumed).

Setting of the Node-ID is to be done using jumpers or switches.

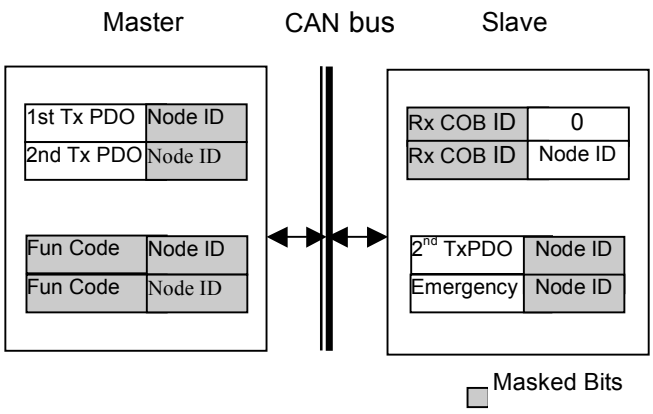


Figure 6: Filter configuration for TMS320F2406 ECAN module

3.2 PDO linking

The CANopen predefined connection set supports only master-slave Communications. In HEV control envisaged, slave–slave communication is also to be implemented. The CANopen standard defines fixed COB-ID's (default identifier) for the first 4 PDOs depending on the node number. Communication between slave nodes is only possible via a CANopen master when using these default identifiers. This, however, will result in an increased CAN bus load since data exchange between two slave nodes requires sending the message from the first slave to the master first and from there to the second slave. CANopen offers the possibility to adjust the CAN identifier for a given communication object [3]. For example, the CAN identifier for a TPDO can also be assigned to a RPDO. With this, it is possible to establish direct communication between two slave nodes without a master node.

This assignment of CAN identifiers for PDOs is also called PDO linking. The distribution of CANopen services is listed in the table IV.

The inbuilt CAN controller in the TMS320LF2406A and PIC18F4480 has message mail boxes and filters which support basic CAN data communication[11] [12]. The two mask registers available are configured to optimize the data reception in the controller itself with filter configuration as shown in Fig 6. One filter with COB-ID masked is associated with the first two receive mail boxes. This facilitates reception of message intended for the Node as well as broadcast messages (Node id 0) respectively in the mail boxes. Another filter with Node ID masked, associated with other two mailboxes facilitate the reception of PDO messages and emergency messages from slaves also.

TABLE VI PDO mapping for RxPDO1

Control Node	PDO format		
PMC	$PMC_{st}^{(*)}$	$V_{bat}^{(**)}$	$N_{ref}^{(**)}$
GC	$GC_{st}^{(*)}$	$V_{bat}^{(**)}$	
BC	$BC_{st}^{(*)}$	$P_{ref}^{(**)}$	

TABLE VII PDO mapping for TxPDO2(RTR)

Control Node	PDO format		
PMC	$ProDat1^{(**)}$	$ProDat2^{(**)}$	$ProDat3^{(**)}$
GC	$ProDat4^{(**)}$	$ProDat5^{(**)}$	$ProDat6^{(**)}$
BC	$ProDat7^{(**)}$	$ProDat8^{(**)}$	$ProDat9^{(**)}$

TABLE VIII PDO mapping for RxPDO2

Control Node	PDO format		
PMC	$P_{ref}^{(**)}$		
GC	$P_{gc}^{(**)}$		
BC	$V_{bat}^{(**)}$	$V_{dc}^{(**)}$	

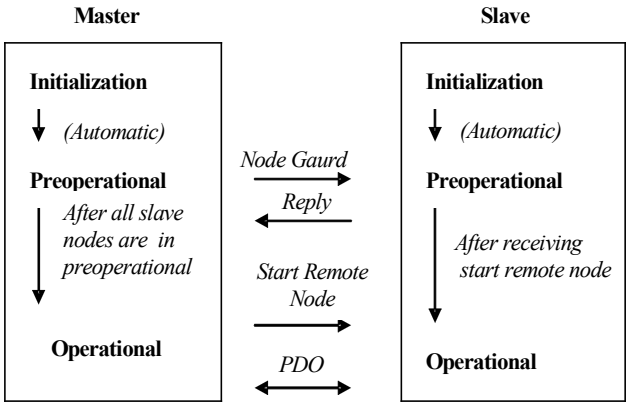


Figure 7: State machine implementation

TABLE IV CANOpen services and objects

CAN-Open Objects	Function
TxPDO1	Slave-Master Periodic transmission
TxPDO2	Slave- Master RTR transmission
RxPDO1	Master-Slave Periodic transmission
RxPDO2	Slave-Slave Periodic transmission
NMT	State machine control
SYNC	Synchronization of periodic transmissions
NMT Node Guard	Check Slave status
Emergency	Error/Abnormal condition
SDO1	Configuration

3.3 State machine implementation

The state machine implementation for minimum boot-up process for master, slave nodes and its corresponding CANopen services are highlighted in Fig 7. Further, the operations of master and slave nodes during power up are to be guaranteed across the network as the nodes may not be powered simultaneously.

Master *power ON first:* As per the above algorithm master gets into preoperational state and send Node guarding message to slaves and since no slave is powered up, there will not be any acknowledge for the message frame and the master node increments error count for no ACK, but it will be limited to 127 (error number will not be increased above 127 for ACK error). However it continues to send the node guard message. Once the slave gets powered up, it assert the ACK field for the node guard message and the operations continue normally.

Slave power ON first: The slave will get into preoperational state and wait for message from master . Once the master gets powered up the operations continue normally.

3.4 Sampling requirement for communication

The data distribution in one communication sampling is as shown in Fig 8. At 1Mbps CAN baud rate, a sampling rate in the order of 2-5 milisec across the network is possible in this configuration with each module data transfer taking approximately 100 micosec

The master issues SYNC messages at every sampling period. The slaves respond with the corresponding TxPDO for periodic data. The master then sends periodic data for each node through RxPDO. If more data are required from any slave (mainly debug information during developments and system testing) it will be requested with an RTR and slave respond with corresponding TxPDO. The slave to slave data will be transmitted through RxPDO2 periodically.

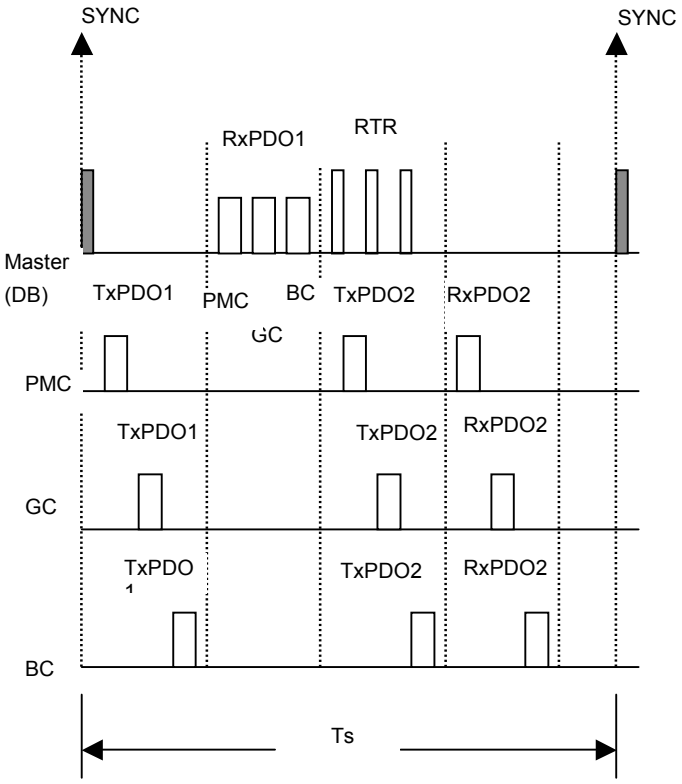


Figure 8: Data distribution in communication sampling

3.5 Object dictionary (OD)

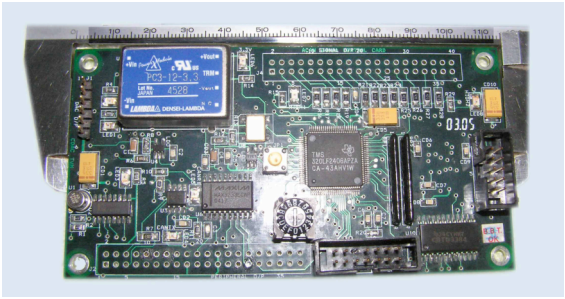
The object dictionary of a CANopen system has three parts namely a *communication*, a *device-profile* and a *device-specific* (manufacturer-specific). The OD for this application is based on the CiA device profile for I/O modules DSP-401[5].

The device profile description for various modules as implemented in master is detailed in table IV. The corresponding PDO distribution is modified from the specification, to get an optimum data transfer between the HEV modules as shown in table [V- VIII].

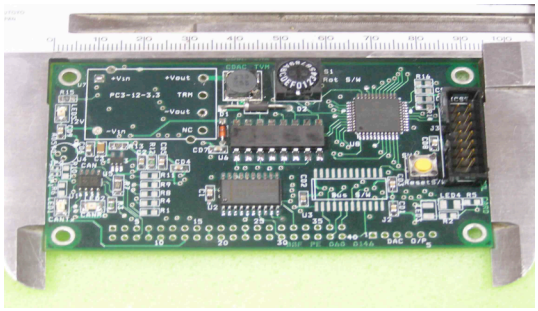
4 CAN based embedded controller

Two types of digital controllers are developed for the HEV modules. For power electronics control the design is based on TMS320LF2406A low voltage fixed point DSP from TI. A CPLD based circuit for dead band generation and fault interlock for gate control signals are incorporated. This controller is used in PMC, GC and BC.

For controller based on Microchip's PIC18LF4480, provision for more I/O interface and relay logic control is provided, which is used in DB. Multilayer PCB with on board SMPS is considered for minimum EMI. The two controllers are as shown in the Fig 9. All low voltage digital and analog signals from the controller are interfaced externally through Peripheral Interface PCB.



TI DSP based Controller



Microchip PIC based controller

Figure 9: Control hardware

TABLE IV Device profile description

Index	Object code	Name	Data type	Attributes
Dash Board				
6000h	VAR	Acceleration Pedal reading	INTEGER16	rw
Propulsion Motor Controller				
6010h	VAR	PMC Status	UNSIGNED8	rw
6011h	VAR	Commands for PMC	UNSIGNED8	rw
6012h	VAR	DC link voltage	INTEGER16	rw
6013h	VAR	Propulsion Motor Speed	INTEGER16	rw
6014h	VAR	Propulsion controller power	INTEGER16	rw
Generator Controller				
6020h	VAR	GC Status	UNSIGNED8	rw
6021h	VAR	Commands for GC	UNSIGNED8	rw
6022h	VAR	ICE speed	INTEGER16	rw
6023h	VAR	Generator controller power	INTEGER16	rw
6024h	VAR	Battery voltage	INTEGER16	rw
Battery Charger				
6030h	VAR	BC Status	UNSIGNED8	rw
6031h	VAR	Commands for BC	UNSIGNED8	rw
6032h	VAR	Battery state of charge	INTEGER16	rw
6033h	VAR	Battery parameters	INTEGER16	rw

5 Conclusion

The implementation methodology for CAN based systems for real time control application is detailed for series HEV environment. The real time control requirements are evaluated for Three wheeler application and configurations of the systems for CANopen (master and slave) protocol structure is developed. The rate of reliable data transfer determines the performance of the vehicle. This data transfer is achieved through effective error detection and signaling mechanism with CAN and thereby the vehicle safety and functionality are improved. The reliable performance of the communication system under high level switching interference from power converter PWM signals is noticeable. Further, the CAN network facilitated in the vehicle performance evaluations like acceleration, maximum speed, range, gradient fuel consumption etc., with a data acquisition module interfaced in the network. This support the CAN feature that any node can be added to the network without affecting the real time performance of the network as long as the node is a consumer. Matching performance with the conventional ICE driven vehicle and better fuel efficiency are achieved.

HEVs are a potential choice for environmental friendly public transportation systems and many value added features can be easily incorporated with CAN based distributed control.

References

[1] Robert Bosch GmbH, CAN Specification 2.0 Part B, September 1991

[2] CiA/DS 102, CAN Physical Layer for Industrial Applications, April 1994

[3] CAN-in-Automation, CANopen, CAL-based Communication Profile for Industrial Systems, CiA DS-301, Version 4.0, June 16 1999.

[4] H.Boterenbrood, CANopen - high-level protocol for CAN-bus, Version 3.0, NIKHEF internal documentation, March 20, 2000

[5] CAN-in-Automation CAN open Device Profile for I/O Modules CiA DSP- 401 vertion 1.4 Dec1996.

[6] CAN-in-Automation CAL,CAN Application layer for Industrial Applications CiA draft standard DS-201 to DS 207 Vertion 1.1 Feb 1996.

[7] CAN-in-Automation CAN open Device Profile for Drives and motion control CiA DSP- 402 vertion 1.0 july 2002

[8] Ali Emadi, Kaushik Rajashekara, Sheldon S. Williamson, Srdjan M.Lukic, " Topological Overview of Hybrid Electric and Fuel Cell Vehicular Power System Architectures and Configurations," IEEE Transactions on Vehicular Technology, Vol. 54, NO. 3, May 2005

[9] Z.V.Lakaparampil, K.A.Fathima,Gautam Poddar, Renji.V.Chacko, B.Sreekumari,V.K.Neelakandhan, "Simulation of HEV with typical driving cycle for a crowded city application application."9th European Conference on Power Electronics and Applications, Graz, Austria, August 2001

[10] Werner Leonhard, "Control of Electric Drives," Third edition, Springer 2001.

[11] TMS 320C2xx User's Guide, Texas Instruments Inc. 2000.

[12] PIC 18F44xx User's Guide, Microchip Inc. 2004.