

# **CAN Control System for an Electric Vehicle**

---

**A Thesis submitted in partial fulfilment of the requirements  
for the degree of Master of Engineering (Electrical and Electronic)  
at the  
University of Canterbury  
Christchurch, New Zealand**



**By  
Abdel Azzeh  
B.Tech. (Hon.)**

**University of Canterbury  
Christchurch, New Zealand**



# ABSTRACT

---

The University of Canterbury has purchased a 1992 Toyota MR2 and used it as the platform to construct a new electric car. Similar to the common combustion engine vehicle, electric vehicles require control systems to control the operation of 12Vdc auxiliary loads, such as lights, indicators and windscreen wipers, where traditional technology results in a large number of wires in the wiring harness. Also, with the added complexity of modern vehicles, the need for integrating independent control systems together has become very important in providing safer and more efficient vehicles.

To reduce the number of wires and make it possible for different control systems to communicate, and so perform more complex tasks, a flexible and reliable control system is used. The CAN (Controller Area Network) control system is a simple two-wire differential serial bus system, which was developed by Bosch for automotive applications in the early 1980s. The power and control system within the vehicle is named the “Power Distribution Network” and it is implemented by using multiple power converters and the CAN control system.

This thesis presents the design, implementation, and test results of the CAN control system for the MR2. The 12Vdc nominal battery voltage is converted to an intermediate voltage of 48Vdc. This configuration is considered more efficient than the usual 12Vdc distribution system since smaller and lighter wires can be used to carry the same amount of power. The power distribution network operates off the 48Vdc intermediate voltage, and provides 12Vdc output to power all auxiliaries within the vehicle.

The Power Distribution Network is implemented with two major subsystems: the auxiliary power system, which consists of multiple converters to step-down voltage from the 48Vdc intermediate voltage to the 12Vdc, and the CAN control system, which is developed to control and integrate the 12Vdc auxiliary loads within the vehicle.

The prototype CAN control system is fully operational and has been tested with 12Vdc loads which are used to simulate most of the auxiliary loads in the vehicle. Experimental measurements show that the prototype is able to successfully control and maintain the network of independent nodes. This confirms that in principle the CAN control system is suitable for controlling the auxiliary loads in an electric vehicle.



# ACKNOWLEDGMENTS

---

I would like to express my gratitude to those who have helped me in the course of this thesis. First and foremost, I would like to gratefully thank my supervisor, Dr. Richard Duke for his guidance, assistance and encouragement throughout the thesis.

Thanks are also extended to the technical staff members: Mr. Philipp Hof for his assistance with the software implementation, Mr. Ron Battersby and Mr. Ken Smart – for their technical assistance in the laboratory.

Last but not least, I would like to express my deepest thanks to my mother, father, family and friends for their encouragement and support over the years of study.



## **PUBLICATION RELATED TO THIS THESIS**

---

Abdel Azzeh, Richard Duke “CAN Control System for an Electric Vehicle”, ENZCon 2005, the 12<sup>th</sup> Electronics New Zealand Conference. Manukau City, New Zealand, November 2005.



# TABLE OF CONTENTS

---

<b>ABSTRACT .....</b>	<b>I</b>
<b>ACKNOWLEDGMENTS.....</b>	<b>III</b>
<b>PUBLICATION RELATED TO THIS THESIS.....</b>	<b>V</b>
<b>TABLE OF CONTENTS .....</b>	<b>VII</b>
<b>LIST OF FIGURES.....</b>	<b>IX</b>
<b>LIST OF TABLES.....</b>	<b>XI</b>
<b>1 INTRODUCTION .....</b>	<b>1</b>
1.1 UNIVERSITY OF CANTERBURY’S ELECTRIC VEHICLE.....	2
1.2 THESIS OUTLINE.....	5
<b>2 BACKGROUND .....</b>	<b>7</b>
2.1 SUMMARY .....	12
<b>3 POWER DISTRIBUTION NETWORK .....</b>	<b>13</b>
3.1 AUXILIARY LOAD CURRENT INVESTIGATION .....	15
3.2 POWER SUPPLY SELECTION AND PLACEMENT IN THE MR2 .....	17
3.3 CAN CONTROL SYSTEM.....	20
3.4 SUMMARY .....	20
<b>4 CONTROLLER AREA NETWORK (CAN).....</b>	<b>21</b>
4.1 CAN OVERVIEW.....	22
4.2 THE CAN PROTOCOL .....	24
4.2.1 <i>CAN Message Frame Format</i> .....	25
4.3 SUMMARY .....	27
<b>5 CAN CONTROL SYSTEM DESIGN.....</b>	<b>29</b>
5.1 CAN NODES ALLOCATION .....	30
5.2 CAN IMPLEMENTATION METHOD .....	31
5.3 CAN CONTROL SYSTEM DESIGN.....	32
5.3.1 <i>CAN Network ICs Selection</i> .....	35

5.3.2	<i>Loads Description and Control Methods</i>	37
5.4	SUMMARY	41
<b>6</b>	<b>CAN CONTROL SYSTEM CONSTRUCTION</b>	<b>43</b>
6.1	HARDWARE	43
6.1.1	<i>CAN Main Controller Node</i>	45
6.1.2	<i>CAN Rear Load Node</i>	48
6.1.3	<i>CAN Wipers Load Node</i>	50
6.1.4	<i>CAN Electric Windows Load Node</i>	51
6.1.5	<i>CAN Front Load Node</i>	53
6.2	SOFTWARE	55
6.2.1	<i>Software Methodology</i>	55
6.2.2	<i>CAN Main Controller Software</i>	56
6.2.3	<i>CAN Load nodes Software</i>	58
6.3	SUMMARY	61
<b>7</b>	<b>PROTOTYPE PERFORMANCE</b>	<b>63</b>
7.1	CAN CONTROL SYSTEM TESTS	63
7.1.1	<i>Time Delays</i>	65
7.1.2	<i>Electric Windows Current Sensing</i>	68
7.1.3	<i>Missing Node Detection</i>	69
7.1.4	<i>Synchronization Problem</i>	70
7.2	SUMMARY	71
<b>8</b>	<b>FUTURE DEVELOPMENT &amp; CONCLUSION</b>	<b>73</b>
8.1	FUTURE DEVELOPMENT	73
8.2	CONCLUSION	74
<b>9</b>	<b>REFERENCES</b>	<b>77</b>
	<b>APPENDIX A</b>	<b>79</b>
	<b>APPENDIX B</b>	<b>81</b>
	<b>APPENDIX C</b>	<b>83</b>
	<b>APPENDIX D</b>	<b>85</b>
	<b>APPENDIX F</b>	<b>89</b>

# LIST OF FIGURES

---

FIGURE 1.1: 1903 KRIEGER ELECTRIC CAR .....	2
FIGURE 1.2: UNIVERSITY OF CANTERBURY'S EV1 .....	3
FIGURE 1.3: UNIVERSITY OF CANTERBURY'S EV2 .....	3
FIGURE 1.4: UNIVERSITY OF CANTERBURY'S EV3 .....	4
FIGURE 2.1: EXAMPLE HIGH-END VEHICLE [5] .....	10
FIGURE 2.2: CONVENTIONAL BRAKE SYSTEM, SIEMENS ELECTRONIC WEDGE BRAKE .....	11
FIGURE 3.1: OVERALL POWER DISTRIBUTION NETWORK DIAGRAM .....	14
FIGURE 3.2: CURRENT WAVEFORM OF 21W LIGHT BULB .....	16
FIGURE 3.3: POWER SUPPLY PLACEMENT IN MR2 [2] .....	18
FIGURE 3.4: MR2'S ELECTRICAL WIRING DIAGRAM .....	19
FIGURE 4.1: RELATION BETWEEN DATA TRANSFER RATE AND BUS LENGTH [10]. .....	22
FIGURE 4.2: ISO/OSI REFERENCE MODEL .....	23
FIGURE 4.3: STANDARD AND EXTENDED FRAMES FORMAT IN CAN [2] .....	25
FIGURE 5.1: CAN NODES DISTRIBUTION .....	30
FIGURE 5.2: POSSIBLE STRUCTURES OF THE CAN NODE .....	31
FIGURE 5.3: CAN NODES DETAILS .....	33
FIGURE 5.4: CAN DIFFERENTIAL VOLTAGE .....	34
FIGURE 5.5: HIGH AND LOW SIDE MOSFET CONTROL .....	38
FIGURE 5.6: WIPERS MOTOR UNIT BLOCK DIAGRAM [14] .....	39
FIGURE 5.7: HEADLIGHT MOTOR UNIT [14] .....	40
FIGURE 6.1: CAN MAIN CONTROLLER NODE BLOCK DIAGRAM .....	46
FIGURE 6.2: CAN MAIN CONTROLLER NODE BOARD .....	47
FIGURE 6.3: CAN REAR LOAD NODE BLOCK DIAGRAM .....	48
FIGURE 6.4: CAN REAR LOAD NODE BOARD .....	49
FIGURE 6.5: CAN WIPERS LOAD NODE BLOCK DIAGRAM .....	50
FIGURE 6.6: CAN WIPERS LOAD NODE BOARD .....	51

FIGURE 6.7: CAN ELECTRIC WINDOWS NODE BLOCK DIAGRAM.....	52
FIGURE 6.8: CAN ELECTRIC WINDOWS LOAD NODE BOARD.....	53
FIGURE 6.9: CAN FRONT LOAD NODE BOARD.....	54
FIGURE 6.10: CAN MAIN CONTROLLER SOFTWARE FLOWCHART.....	57
FIGURE 6.11: CAN LOAD NODE SOFTWARE FLOWCHART.....	60
FIGURE 7.1: PROTOTYPE PHOTOGRAPH.....	64
FIGURE 7.2: MESSAGE TRANSMISSION WAVEFORMS. TIME SCALE: 1MS/DIV.....	66
FIGURE 7.3: HEADLIGHTS TURN-ON. TIME SCALE: 100MS/DIV.....	67
FIGURE 7.4: WINDOWS CURRENT WAVEFORM. TIME SCALE: 5S/DIV.....	69
FIGURE 7.5: ERROR LED. TIME SCALE: 2MS/DIV.....	69
FIGURE 7.6: FRONT AND REAR INDICATORS TIMING MISMATCH. TIME SCALE: 100 $\mu$ S/DIV.....	71

# LIST OF TABLES

---

TABLE 3.1: AUXILIARY LOADS IN THE VEHICLE – MR2 [2].....	15
TABLE 3.2: MAXIMUM CURRENT DEMAND OF THE LOADS [2].....	16
TABLE 3.3: POWER CONSUMPTION OF THE VEHICLE LIGHTS .....	17
TABLE 7.1: TIME DELAYS FOR SWITCHING LOADS .....	68



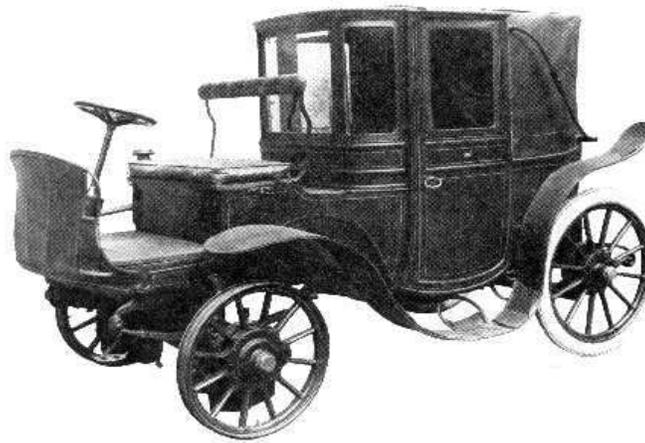
# 1 INTRODUCTION

---

Electric vehicles have a much longer history than most people realise. Electric vehicles were seen soon after Joseph Henry first introduced the first DC powered motor in 1830. A small model of an electric vehicle was built by Professor Stratingh in the Dutch town of Gröningen in 1835. The first full sized electric vehicle was built in 1834 by Thomas Davenport in the USA, followed by Moses Farmer who built the first two passenger electric vehicles in 1847. There were no rechargeable electric cells (batteries) at the time. An electric vehicle did not become a viable option until the Frenchmen Gaston Plante and Camille Faure respectively invented (1865) and improved (1881) the storage battery [1].

Figure 1.1 shows a 1903 Krieger electric car. This car is a front wheel drive electric-gasoline hybrid car and has power steering. A gasoline engine supplements the battery pack. Between 1890 and 1910, there were many hybrid electric cars and four wheel drive electric cars. Electric cars were more expensive than gasoline cars at the time, and so, the industry concentrated more on making and developing combustion engine vehicles [1].

Electric vehicles are known as Zero Tailpipe Emissions Vehicles, and they produce less pollution than petrol or LPG powered vehicles. As electric vehicles have fewer moving parts, maintenance is also minimal. With no internal combustion engine there are no oil changes, tune-ups and most of all no exhausts. Electric vehicles are also far more energy efficient than gasoline vehicles and they are very quiet [2].



**Figure 1.1: 1903 Krieger electric car [1]**

Electric vehicles have been in continual use since the 1900s in various applications. Today these quiet vehicles with no tailpipe emissions are no longer limited to golf carts. New advances in battery technology, system integration and aerodynamics, and research and development by major vehicle manufacturers have led to the production of electric vehicles that can play a practical role on city streets [2].

## **1.1 University of Canterbury's Electric Vehicle**

For almost 30 years now the Department of Electrical and Computer Engineering has maintained a roadworthy electric car. The objective was to provide a suitable laboratory test bed by which to demonstrate the drive system, control system and various other systems. The first electric vehicle (EV1) was originally developed as a mobile test bed for proving AC induction motor speed controllers, which were being developed during the early 1970s for industrial applications. The EV1, shown in Figure 1.2, was first registered to run on the road in September 1976. Although the EV1 had a limited range of only 40km at 50km/h and maximum speed of 80km/h, it demonstrated that an electric powered vehicle could be viable for operation in urban driving conditions. This vehicle is now on display at a local transport and technology museum [3].



**Figure 1.2: University of Canterbury's EV1**

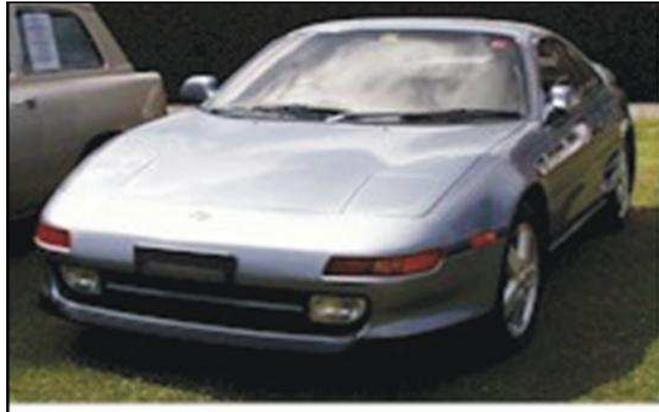
In early 1982 a second electric vehicle (EV2) was completed, in conjunction with the Mechanical Engineering department, and registered to run on the road. The vehicle was based on a 1962 Austin A40 Farina as shown in Figure 1.3. Major body alterations were undertaken to lower the car's aerodynamic drag. With its much improved drag co-efficient over the EV1, the range of the EV2 was improved to 60km on one charge at a constant 65km/h speed with a top speed of up to 80km/h. EV2 has been used on a daily basis up until April 2005 when it too was consigned to the local transport and technology museum. The only major outward signs of change from 1982, when it first went on the road, were a paint job in 1999, a microprocessor based controller to replace the original analogue controller and an IGBT inverter which replaced the original thyristor inverter [2, 3].



**Figure 1.3: University of Canterbury's EV2**

In early December 1999, the University of Canterbury purchased a Toyota MR2 for use as the next version in the electric vehicle (EV3) research programme, shown in Figure 1.4. This time the decision was made to build a higher performance vehicle that would both attract people's attention and their imagination. Initial planning was commenced on this

project by interested staff and students to investigate and modify the control system, battery system, electric drive system and mechanical structure. The MR2 was registered to run on the road in May 2006, with a range of about 25km for around town driving and a top speed of over 100km/h.



**Figure 1.4: University of Canterbury's EV3**

Similar to the common combustion engine vehicle, electric vehicles require control systems to control the operation of 12Vdc auxiliary loads, such as lights, indicators and windscreen wipers, where traditional technology results in a large number of wires in the wiring harness. Also, with the added complexity of modern vehicles, the need for integrating independent control systems together has become very important in providing safer and more efficient vehicles. An efficiency gain can be seen in the distribution of power converters around the vehicle. Power converters are used to step-down the voltage from the battery bus to 12Vdc to power the auxiliary loads within the vehicle. Ideally they should be located as close as possible to their point of use, thus minimising the length of wires in the 12Vdc supply harness, reducing their weight and the space required around the vehicle.

To reduce the number of wires and make it possible for different control systems to communicate, and so perform more complex tasks, a flexible and reliable control system is used. The CAN (Controller Area Network) control system is a simple two-wire differential serial bus system, which was developed by Bosch, for automotive applications in the early 1980s [4].

## 1.2 Thesis Outline

This thesis focuses on the development and implementation of the CAN control system for the electric vehicle, MR2 application. The CAN control system is developed to integrate and control most 12Vdc auxiliary loads and power supplies within the MR2. The main objectives are to demonstrate the operation, assess the overall performance, and identify particular benefits which might accrue when using the CAN control system. CAN has been successfully applied to gasoline automotive applications before, but rarely to electric vehicles. Electric vehicles have their own power requirements, where the whole vehicle runs on the power supplied from the batteries. The voltage is derived from a high voltage bus to drive low voltage applications, thus the CAN control system needs to conform to specific requirements.

Chapter 2 of this thesis outlines the development of in-vehicle networking in the past few decades up until the present time, in which several networks, such as LIN, MOST and CAN are jointly employed to control and integrate all independent control systems within modern vehicles.

Chapter 3 introduces the auxiliary power system and the CAN control system of the power distribution network implemented within the MR2. The overall structure of the power distribution network is discussed along with the power supplies selection and placement in the MR2 to best serve their applications. The general structure of the CAN control system and its integration within the vehicle's power distribution network is then examined.

Chapter 4 describes the basics of the CAN protocol, including its message format and properties. Then the reasons for selecting CAN are made according to the strengths and advantages of this protocol.

Chapter 5 introduces the prototype design and the plan for installing the CAN control system in the MR2. This includes investigating the scope of the CAN control system, data transfer rate, components selection, loads description and their assignment to individual CAN nodes, CAN node allocation within the vehicle and the overall CAN control system implementation method.

Chapter 6 details the hardware construction of the individual CAN nodes within the CAN control system prototype, along with their programmed software to successfully control selected 12Vdc auxiliary loads according to the CAN protocol.

Chapter 7 presents the experimental results of tests confirming the correct operation of the CAN control system. Waveforms captured during the tests are also presented in this chapter.

Finally, Chapter 8 discusses possible future developments to the CAN control system to further enhance its functionality, followed by the thesis conclusion signifying the successful operation of the CAN control system prototype, which was designed and implemented for the use in the electric vehicle, MR2 application.

## 2 BACKGROUND

---

Until recently, various control systems within a vehicle were completely independent of one another, preventing the chance of sharing information around the vehicle. Sharing information between independent control systems within a vehicle can make the vehicle safer, more reliable and more fuel efficient. Traction control, which is almost a necessity in modern vehicles, is a safety example, where four sensors are employed around the vehicle, each measuring the rotational speed of a wheel and sending the readings to a main control system. The main control system then assesses the information received from all four sensors and determines, by calculating rotational speed differences, whether or not one or more of the wheels has lost traction. If the main controller finds that one of the wheels is spinning faster than the others, it sends a message to the brake control system telling it to apply the brakes to that specific wheel. In doing so, the main controller ensures that the vehicle will always be in contact with the road, thus, providing a safer ride for all passengers onboard.

To satisfy customer demands, the past four decades have witnessed an exponential increase in the number and sophistication of electronic systems in vehicles. Today, the cost of electronic systems in luxury vehicles can amount to more than 23 percent of the total manufacturing cost. Such electronic systems range from highly important safety systems, such as airbags and traction control systems, to the more luxurious systems, for example audio and video systems [5].

This increasing amount of electronics in vehicles has raised an important issue within the automotive industry. The more electronic systems in a vehicle, the more wires are needed to transfer data and power around the vehicle. Added wiring increases vehicle weight, lowers performance, and makes adherence to reliability standards difficult. For an average well-tuned vehicle, every extra 50 kilograms of wiring increases fuel consumption

by 0.2 litres per 100 kilometres travelled. Also, complex wiring harnesses take up large amounts of vehicle volume, limiting the ability to add functionality. The wiring harness has become the single most expensive and complicated component in vehicle electrical systems [5].

The solution was soon found. By imitating computer networks, serial networks can be built within vehicles to handle all data sharing, with the least amount of wires. These networks typically consist of a number of independent nodes controlling specific tasks around the vehicle, monitored and controlled by one or more main controllers. This would replace point-to-point wires with only one cable on which data can be transferred around the vehicle. On the other hand, replacing the old 12Vdc electrical system with a higher voltage system would proportionally reduce the required current for a given delivered power. Smaller currents will use smaller and lighter-gauge cables, allowing an expected 20 percent reduction in cable bundle size [5].

A 42Vdc voltage system has been chosen to become the standard in modern combustion engine vehicles. This is a direct consequence of integrating the alternator and starter into one unit, which promises to increase vehicles' overall efficiency by an expected 20 percent, or approximately an extra 0.2 kilometres per litre. The old 12Vdc system could not provide enough torque for the integrated alternator-starter unit to start the engine, so the voltage needed to be raised. The 42Vdc system requires a 36Vdc battery and produces a maximum operating level of 50Vdc, with a maximum dynamic over voltage of 58Vdc. Engineers regard a 60Vdc limit as the safe maximum for cars; greater voltages can generate electric shocks [5]. This voltage also falls within the New Zealand standards for extra-low voltage systems, running on a maximum of 50Vac or 120Vdc, which allows maintenance on such systems to be carried out by anyone, even if they were not certified electricians.

Since the start of vehicle networking developments, many network protocols have been introduced to the market, all trying to lead and become the standard for in-vehicle networking, but unfortunately none have succeeded so far. This is due to the wide range of requirements needed by the vast number of electronic systems implemented within vehicles. Most protocols provide specific advantages over other protocols. To find only one protocol that would replace all the rest and become the standard for vehicle networking is what major automobile companies around the world are trying to achieve.

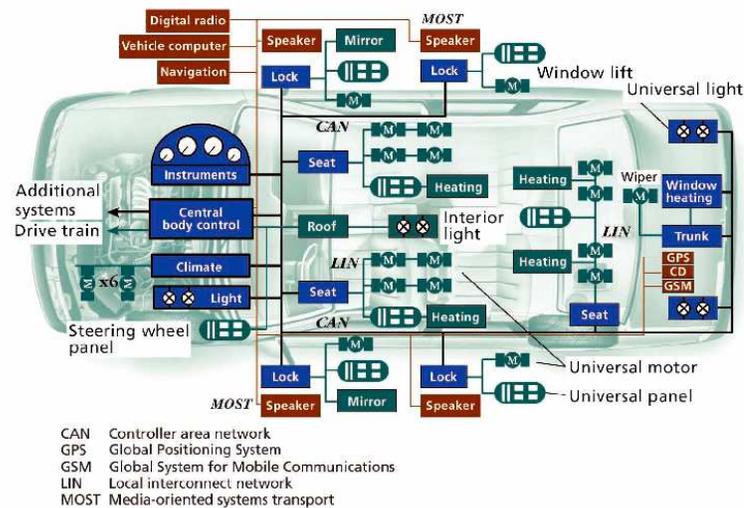
Having said that, several protocols are currently being used together to control specific tasks around vehicles. Today's vehicles typically include multiple networks, each controlling a specific part of the vehicle. Whether they are different speed networks, using the same protocol, or simply different protocols networks in a vehicle, data sharing between them is achieved via gateways. A gateway is, in essence, a buffer that enables different networks to share information via a mutual protocol and speed.

Today, the most dominant and mature of all vehicle network protocols is the CAN (Controller Area Network) protocol, which was developed by Bosch in the early 1980s for automotive applications. Since its inception, it has gained a lot of support and is now used in many more fields other than automobiles. It is now found operating lifts, controlling machine tools, monitoring engine room activity aboard ships and much more. It has gained its reputation from its simplicity, robustness and high level of reliability, with its ability to exchange short real-time messages at a maximum speed of 1Mbps [6].

Although CAN is very popular in the industry, it has one major flaw. CAN is an asynchronous protocol, and it is almost impossible to synchronize two or more nodes in the same network together, that is, each node has an independent clock. This is specifically a disadvantage when it comes to multimedia distribution networks within a vehicle, or any time dependent system for that matter, where one or more network nodes need to be synchronized together, such as speaker outputs in audio systems.

To tackle such problems and enhance existing automotive systems, other networking systems have been developed. These technologies include D2B (Domestic-Digital-dataBus), FlexRay, and MOST (Media-Oriented System Transport), all of which employ fibre media for higher speeds and EMI immunity. TT-CAN (Time-Triggered extension to CAN) which enables CAN nodes in a network to be synchronized, as well as the TTP (Time-Triggered-Protocol) series that competes with FlexRay for safety critical systems. Because these systems serve high-end applications and are relatively expensive, designers require a low-cost alternative to serve mundane tasks, such as controlling body functions from seats to sunroofs. As a result, car makers increasingly embrace LIN (Local-Interconnect Networks), which positions itself at the lowest level in the automotive-networking hierarchy [7]. An overview of the networks mentioned above can be found in [5].

An example of a modern high-end vehicle, where CAN, LIN and MOST networks are employed, is shown in Figure 2.1. CAN controls all real-time critical functions, such as engine management, ABS and traction control. LIN is used as a sub-bus of CAN, and controls all non-safety critical tasks, such as car seats, sun roofs, rain sensors and door mirrors. The optical fibre network MOST, also acting as a sub-bus of CAN, is chosen for high volume streaming to control automotive multimedia. All networks within the vehicle are controlled by a main control system, which also acts as a gateway for all networks to share data [5].



**Figure 2.1: Example high-end vehicle [5]**

With the vast expansion in automotive electronics, manufacturers are also developing technologies to replace older mechanical systems with new lighter, cheaper and more reliable electronic versions. X-by-wire is the term used to denote the conversion from old inaccurate mechanical systems to newer more sophisticated electronic systems. One of the most intriguing new concepts of X-by-wire, called Brake-by-wire, is the wedge brake system, which has been developed by the German company Siemens. The wedge brake system replaces all the tedious large parts of the conventional brake, the hydraulic system, with a more compact, more accurate and easier to maintain electronic system, as shown in Figure 2.2. A brief overview of the wedge brake operation can be found in [8].



**Figure 2.2: Conventional Brake System (left), Siemens Electronic Wedge Brake (right)**

No one knows what the future holds for automobiles, but with such fast evolving technologies in vehicle networking alongside the advancements in X-by-wire systems, vehicles are becoming safer, more fuel efficient and more spacious.

The ultimate future vehicle is one that is fully automated, safe, fuel efficient and as luxurious as possible. One would picture, that in such vehicles all a passenger/driver has to do is to sit in the vehicle and enter the destination on a touch screen, sit back and enjoy the comfortable lavish ride. With sensors monitoring everything in and around the vehicle, a GPS system, or better, providing its exact location, and a powerful main computer that coordinates and manages all networks within it, vehicles would be able to transport passengers safely and cheaply to their destinations with minimal human interaction.

In this thesis a CAN control system is designed and built specifically to control and integrate selected 12Vdc auxiliary loads in the MR2. As mentioned before, this would replace point-to-point wires by one bus cable connecting several independent nodes together and controlled by one main controller. This would decrease the number of wires used around the vehicle, make it possible to share information between different control systems and make developing and installing new modules possible for a safer and more efficient vehicle.

## 2.1 Summary

With the growing electronics industry, modern vehicles are becoming more and more complex, and the need for integrating subsystems together is becoming more important than ever before. To accommodate different system's needs and requirements, manufacturers have developed new vehicle networking protocols, such as CAN, LIN and MOST, which are currently being used jointly in vehicles. To demonstrate the ability of integrating different control systems together in a vehicle, the CAN (Controller Area Network) control system is implemented to control and integrate selected 12Vdc auxiliary loads within the MR2. Since the CAN control system is designed to control power flow to auxiliary loads, the existing power distribution network used within the vehicle has to be understood. For this reason, the power distribution network used within the vehicle to supply power to all parts of the vehicle, including the CAN control system, is detailed in the next chapter, Chapter 3.

## 3 POWER DISTRIBUTION NETWORK

---

The University of Canterbury has purchased a 1992 Toyota MR2 and used it as the platform to construct a new electric car [3]. As already discussed, electric vehicles require control systems to control and supply power to the 12Vdc auxiliary loads, such as lights, indicators and windscreen wipers. This chapter presents the design plan for the power distribution network for the MR2.

The electric car is powered by 26 series connected 12Vdc lead-acid batteries producing a nominal 312Vdc system, which supplies power to the entire vehicle [3]. To provide power to the auxiliary loads, the voltage is stepped down in two stages, as shown in Figure 3.1. The Battery Voltage Conversion Station block produces a 48Vdc intermediate distribution bus, while the Auxiliary Power System blocks produce the 12Vdc [2], which in turn supplies the individual loads controlled by the CAN Control System.

The 48Vdc intermediate distribution bus is considered more efficient than the usual 12Vdc distribution system for the following reasons. The higher voltage results in a lower current draw to deliver the same amount of power to the loads when compared with the usual 12Vdc distribution system. Thus there is a net decrease in power loss in the wires due to the lower current drawn. Second, with a 48Vdc voltage bus, the weight and space required for the wiring could be reduced since smaller and lighter wires could be used in the vehicle. While carrying on with this concept, a 312Vdc distribution system may be considered to be even more efficient, but it is not considered a practical option because of the safety issues related to distributing high voltage around the vehicle.

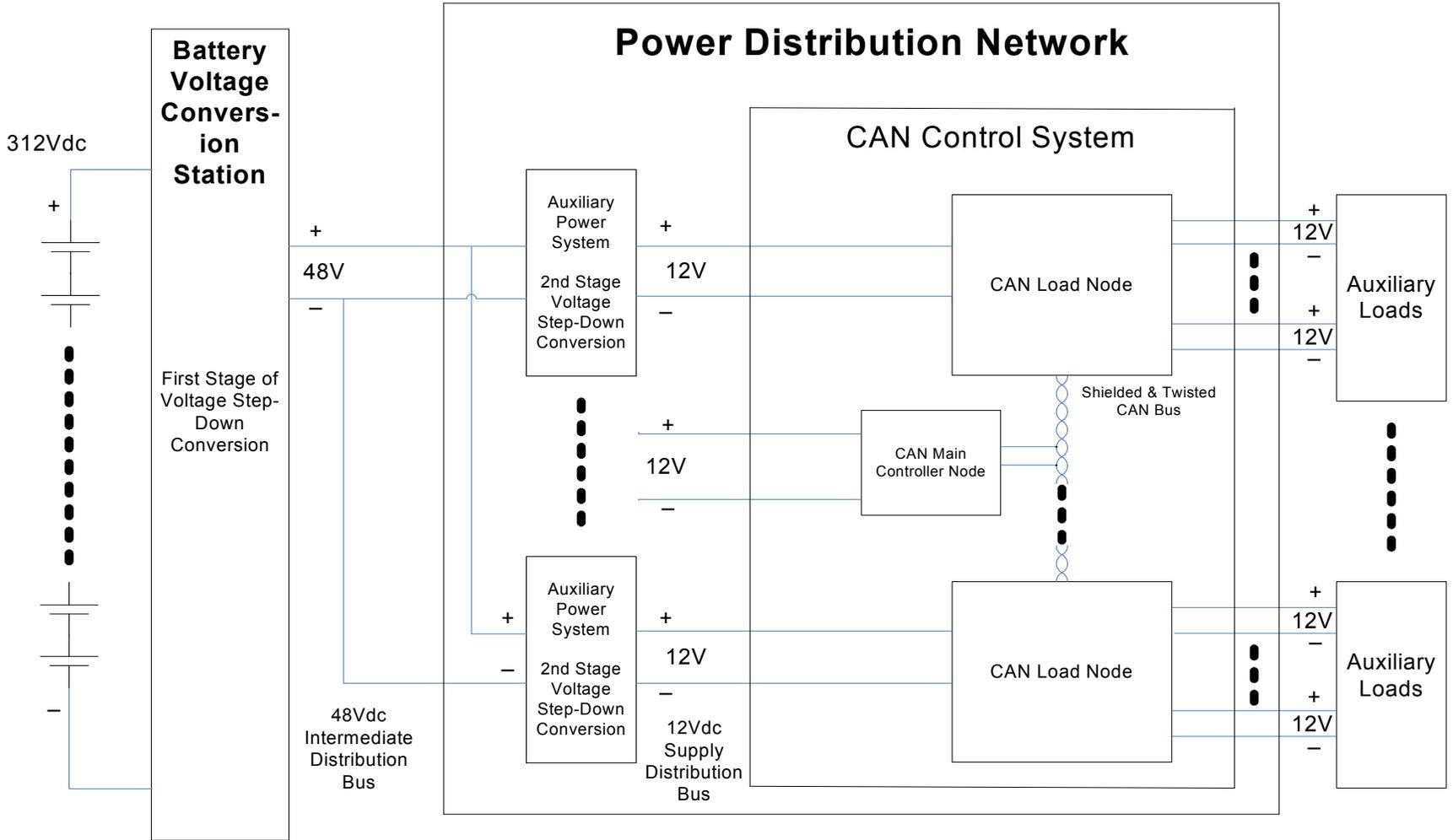


Figure 3.1: Overall Power Distribution Network diagram

### 3.1 Auxiliary Load Current Investigation

The power requirements of existing loads that are to be retained in the petrol-electric conversion and any new auxiliary loads required to be added need to be determined. Current consumption of the loads is important for an electric vehicle as it determines the power rating, number and placement of the dc-dc converters installed in the vehicle.

Auxiliary loads in the vehicle can be classified into two main categories: resistive and inductive. Lighting loads (such as headlights) typically make up the bulk of resistive loads. Other electronic equipment like the car audio system and the rear defogger are also considered as resistive loads. Inductive loads normally involve electric motors, for such functions as power steering, heater fans, electric windows and windscreen wipers. Table 3.1 summarises the existing auxiliary loads that are to be retained in the vehicle [2].

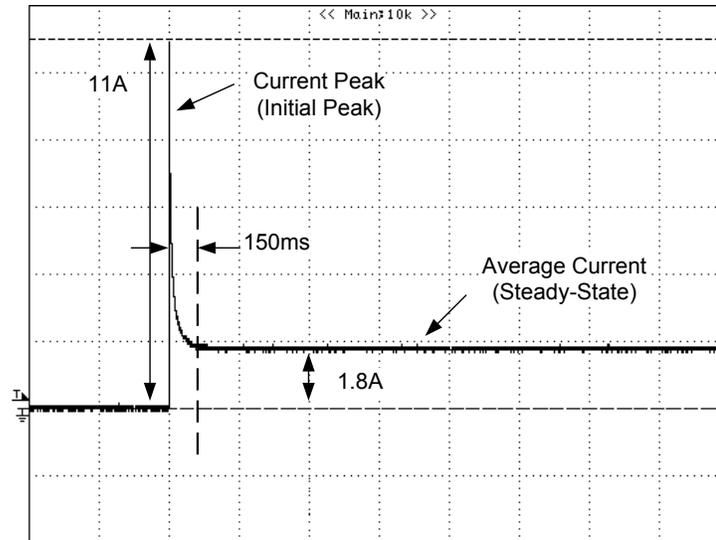
**Table 3.1: Auxiliary loads in the vehicle – MR2 [2]**

Loads	Type	Description
Headlights	Resistive	Two filaments inside each headlight for high beam and low beam.
External Lights	Resistive	Include red taillights (park and brake), fog lights (not working), reverse lights and hazard (indicator) lights.
Internal Lights	Resistive	Include door lights, reading lights and boot light.
Power Steering	Inductive	Include a motor for hydraulic pressure.
Fan Heater	Resistive	Heating elements.
	Inductive	Include air conditioning system (not working) and fans.
Other Electronics	Resistive	Include car audio system (CD/Tape/Radio player and speakers), rear defogger and combination meters.
	Inductive	Include electric windows, windscreen wipers, central locking, radio aerial winder, side mirrors and horn.

Measurements were made of the current drawn by the MR2's auxiliary loads and they are all recorded for easy reference in Appendix A.

Transient current peaks occurred in many measurements and they exist because loads are switched on and off. Figure 3.2 shows an example of the turn-on current waveform for a 21W brake light bulb. It can be seen that when the bulb is initially turned on, current rushes into it to reach a peak of around 11A, almost six times as much as its normal steady-state operating current of 1.8A. Although the current is stabilised relatively quickly down to steady-state within 150ms, turning on more than one load at the same time

could cause current overshoot to add up to reach very high values, which in turn could exceed the supply power rating and cause it to shutdown.



**Figure 3.2: Current waveform of 21W light bulb**  
(Current scale: 2A/div; Time scale: 500ms/div)

Table 3.2 shows the summarised current consumption of the auxiliary loads in the vehicle. It is seen that the headlights, other external lights and power steering consumed the major amount of power. It also shows that the magnitude of current peaks can be extremely large. Current limiting techniques to reduce the peak currents have been studied and used in the vehicle [2].

**Table 3.2: Maximum current demand of the loads [2]**

Loads	Maximum Current Consumed	
	Peak Current (Initial peak)	Average Current (Steady-State)
Headlights	27.4A	15A
External Lights	53.8A	31A
Internal Lights	10.3A	2.1A
Power Steering	49.2A	25A
Fan Heater	25.6A	11.8A
All Other Electronics (see Table 3.1 for descriptions)	42.1A	20A
Total	208.4A	104.9A

Power consumption can be further reduced by replacing all external and internal light bulbs in the vehicle, except headlights, with high intensity LEDs. Preliminary calculations are carried out to show that the current consumption of these loads can be decreased significantly if LEDs are used. For example, the current consumption of external lights could be reduced from 31A to 10A by replacing all the external lights with LEDs. Table 3.3 summarises current consumption of the external and internal light bulbs compared to using high intensity LEDs.

**Table 3.3: Power consumption of the vehicle lights before and after the replacement of high intensity LEDs [2]**

Lights	Current Required in normal light bulbs	Current Required in high intensity LEDs
External Lights (Excluded Headlights)	31A	10A
Internal Lights	2.1A	0.7A

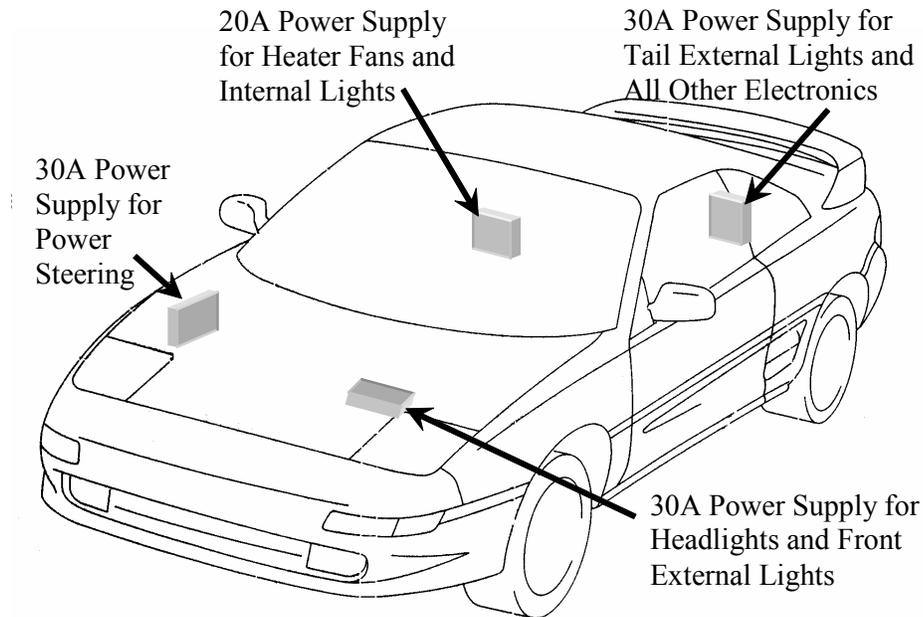
Sample multi-LED bulb replacements were trialled in some external lights, but it soon became very clear that without speciality lens covers the intensity of these lights was insufficient for safe use.

## 3.2 Power Supply Selection and Placement in the MR2

The power distribution network, shown in Figure 3.1, is designed to provide power to the various 12Vdc auxiliary loads. The power received from the 48Vdc intermediate bus could be stepped-down to 12Vdc either by using a single power supply with large power rating or by using a number of independent lower power rated application dependent power supplies. Higher conversion efficiencies are usually achieved by having well defined loads, where the variation between light and heavy loads is relatively small. A single power supply would normally be operating on light loads, thus providing low efficiency. In the MR2, a number of power supplies are distributed around the vehicle to better optimize the overall efficiency.

Using the information provided by Table 3.2, it was decided to use a distributed power supply system consisting of four dc-dc isolated step-down converters with output power ratings ranging from 150W to 500W. Figure 3.3 illustrates the proposed placement of these four power supplies in the vehicle. The philosophy in implementing this distributed

auxiliary power system is to locate the supply as close as possible to its point of use, thus minimising the length of wires in the 12Vdc supply harness.



**Figure 3.3: Power supply placement in MR2 [2]**

Figure 3.4 shows the MR2's electrical wiring diagram. By examining the diagram, it can be seen that the maximum total power that can be delivered to all 12Vdc auxiliary loads is about 1.5kW. If at anytime the threshold of any individual converter is exceeded, due to transient current peaks, the auxiliary converter may be forced to shut down. A solution for this problem will be discussed in the next section of this chapter.

Upon comparing Figure 3.3 and Figure 3.4, it can be seen that there are some differences in the converter's power ratings and the loads they are powering. The reason behind this is that Figure 3.4 shows the current electrical system in the MR2, which was developed as a first attempt in practically implementing the distributed 12Vdc power supplies to power all 12Vdc loads within the vehicle, while Figure 3.3 shows the expected future and final plan for distributing the power supplies within the MR2.



### 3.3 CAN Control System

Within any vehicle, a control system is needed to control the distributed power system. The distributed power system in the electric vehicle is controlled by the CAN Control System block, shown in Figure 3.1. The figure shows the general structure of the CAN control system and its integration within the vehicle's power distribution network.

The CAN control system consists of one CAN Main Controller node and four CAN Load nodes. All CAN nodes are connected to a twisted pair of wires, the CAN Bus, to transmit and receive messages.

The CAN control system is also responsible for ensuring that transient currents would not accumulate to reach the maximum power threshold that can be supplied by a single Auxiliary Power System block (power converter). This is accomplished by inserting, in software, time delays between switching high-demand loads, such as headlights. The CAN is chosen to replace the old factory control system, because it reduces the wiring harness weight and complexity in the vehicle. The design and construction of the CAN control system are presented in Chapter 5 and Chapter 6.

### 3.4 Summary

The 312Vdc nominal battery voltage is converted to a 48Vdc intermediate voltage for distribution around the MR2. 48Vdc was chosen as a compromise between the two extremes of either 312Vdc or 12Vdc. Current consumption investigations are made of the MR2's auxiliary loads. Transient current peaks which may occur when loads are switched in and out could load down the dc-dc auxiliary converters and cause them to shut down. The CAN control system, however, manages the power system and ensures that transient currents do not add up to exceed each converter's threshold.

The power distribution network as used in the electric vehicle application has been discussed. There are two major subsystems: the power distribution network, which converts and delivers power to the various auxiliary loads, and the CAN control system that controls the supply of power to the 12Vdc auxiliary loads. An overview of the CAN protocol is described in the next chapter, Chapter 4.

## 4 CONTROLLER AREA NETWORK (CAN)

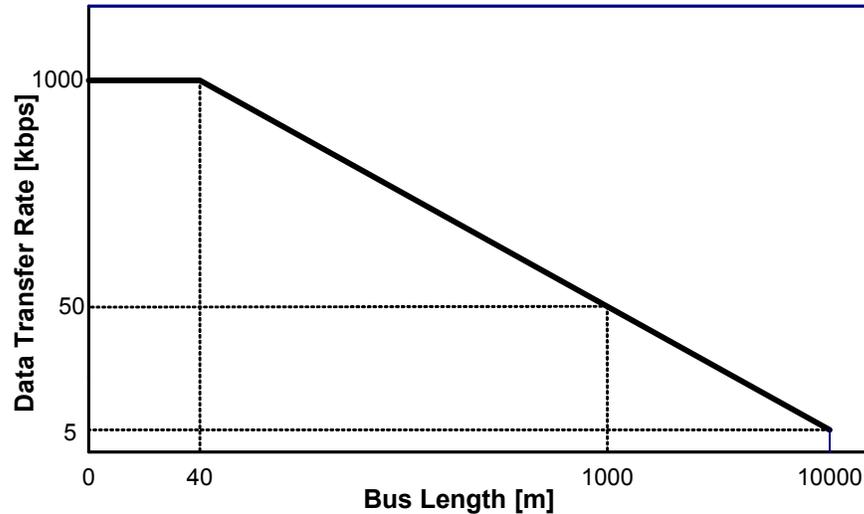
---

CAN is a well-designed serial communications protocol that supports sending and receiving short real-time control messages with a very high level of reliability. It is designed to connect several control systems together in a network, and is capable of operating at speeds of up to 1Mbps, with powerful error detection and handling schemes.

CAN was initially created by the German automotive system supplier Robert Bosch in the mid-1980s for automotive applications, as a method for enabling robust serial communication. The goal was to make automobiles more reliable, safe and fuel-efficient, while decreasing wiring harness weight and complexity. Since its inception, the CAN protocol has gained widespread popularity in industrial automation and automotive applications [9].

CAN is a multi-master differential bus system. Since the CAN protocol is message based, network nodes do not have a specific address. Instead, the address information is contained in the identifier of the transmitted messages, indicating the message content and its priority. The number of nodes may be changed dynamically without disturbing communication of the other nodes [10].

The maximum CAN bus data transfer rate of 1Mbps can be achieved with a bus length of up to 40 metres when using a twisted wire pair, which is the most common bus medium used for CAN. Other transmission mediums, such as coaxial cable or fibre optics could be used. The choice is determined by the environment that the system is going to operate in. The relation between data transfer rate and bus length, for twisted wire pair, is shown in Figure 4.1. For bus lengths longer than 40 metres the bus speed must be reduced, and at bus lengths above 1000 metres special drivers must be used [10].



**Figure 4.1: Relation between data transfer rate and bus length [10].**

The following sections of this chapter describe the basics of the CAN protocol, including its message format and properties. Then the reasons for selecting CAN are made according to the strengths and advantages of this protocol.

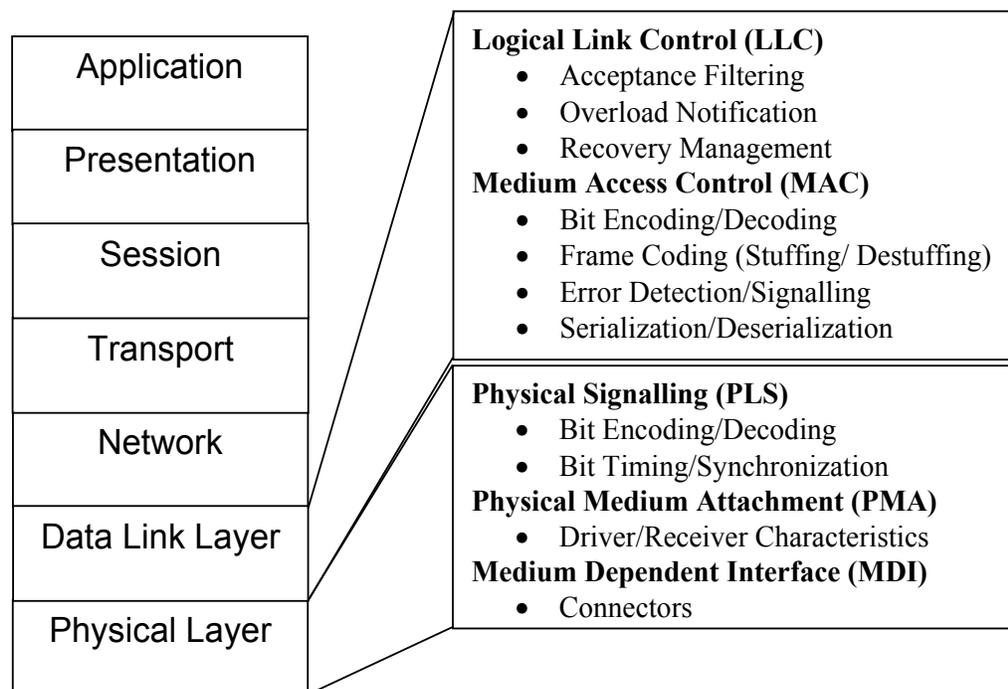
## 4.1 CAN Overview

Most network applications follow a layered approach to system implementation. This systematic approach enables interoperability between products from different manufacturers. A standard was created by the International Standards Organization (ISO) as a template to follow for this layered approach. It is called the ISO Open Systems Interconnection (OSI) Network Layering Reference Model and is shown in Figure 4.2 [9], [11]. The seven layers are defined as follows;

- The application layer is the main interface for the user(s) to interact with the application and therefore the network.
- The presentation layer takes the data provided by the Application layer and converts it into a standard format that other layers can understand.
- The Session layer establishes, maintains and ends communication with the receiving device.

- The Transport layer provides transparent transfer of data between end users, thus relieving the upper layers from any concern while providing reliable and cost-effective data transfer.
- The Network layer determines the way that the data will be sent to the recipient device. Logical protocols routing and addressing are handled here.
- In the Data Link layer, the appropriate physical protocol is assigned to the data. Also, the type of network and the packet sequencing is defined.
- The Physical layer is the actual hardware level. It defines the physical characteristics of the network such as connections, voltage levels and timing.

### ISO/OSI Reference Model



**Figure 4.2: ISO/OSI reference model [9]**

The CAN protocol uses the Data Link Layer and Physical Layer in the ISO/OSI model. The rest of the layers are left to be implemented by the system developer. For the use of CAN, two standards have been defined for the bus interface specifying a 5V differential electrical bus as the physical interface[9, 10]. The two CAN standards are:

- CAN High Speed according to ISO-11898 for bit rates between 125kbps and 1Mbps
- CAN Low Speed according to ISO-11519 for bit rates up to 125kbps

## 4.2 The CAN Protocol

The CAN communication protocol is a CSMA/CD protocol. The CSMA stands for Carrier Sense Multiple Access. What this means is that every node on the network must monitor the bus for a period of no activity before trying to send a message on the bus (Carrier Sense). Also, once this period of no activity occurs, every node on the bus has an equal opportunity to transmit a message (Multiple Access). The CD stands for Collision Detection. If two nodes on the network start transmitting at the same time, the nodes will detect the ‘collision’ and take the appropriate action. In the CAN protocol, a non-destructive bitwise arbitration method is utilized. This means that messages remain intact after arbitration is completed even if collisions are detected. All of this arbitration takes place without corruption or delay of messages [9].

As mentioned earlier, the CAN protocol is a message-based, not an address based protocol. This means that messages are not transmitted from one node to another node based on addresses. Embedded in the CAN message itself is the priority and the contents of the data being transmitted. All nodes in the system receive every message transmitted on the bus. It is up to each node in the system to decide whether the message received should be immediately discarded or kept to be processed [9].

A major benefit of this message-based protocol is that additional nodes can be added to the system without the necessity to reprogram all other nodes to recognize this addition. This new node will start receiving messages from the network and, based on the message identifier (ID), decide whether to process or discard the received information [9].

CAN nodes have the ability to determine fault conditions and transition to different modes based on the severity of problems being encountered. They also have the ability to detect short disturbances from permanent failures and modify their functionality accordingly. CAN nodes can transition from functioning like a normal node (being able to transmit and receive messages normally), to shutting down completely (bus-off) based on the severity of the errors detected. This feature is called Fault Confinement. No faulty CAN node or nodes will be able to monopolize all of the bandwidth on the network because faults will be confined to the faulty nodes and these faulty nodes will shut off before bringing the network down. The CAN protocol also provides sophisticated error detection

and error handling mechanisms such as Cyclic Redundancy Check (CRC) and high immunity against electromagnetic interference. Temporary errors are recovered. Permanent errors are followed by automatic switch-off of defective nodes. Every node in the system is informed about an error. This is very powerful because both Fault Confinement and the error detection methods guarantee bandwidth for critical system information [9, 10].

### 4.2.1 CAN Message Frame Format

The CAN protocol supports two message frame formats, the only essential difference being in the length of the identifier. The so-called CAN standard frame, also known as CAN 2.0 A, supports a length of 11 bits for the identifier, whereas the so-called CAN extended frame, also known as CAN 2.0 B, supports a length of 29 bits for the identifier. The structure of the standard and extended frames of the CAN protocol are shown in Figure 4.3 [9, 10].

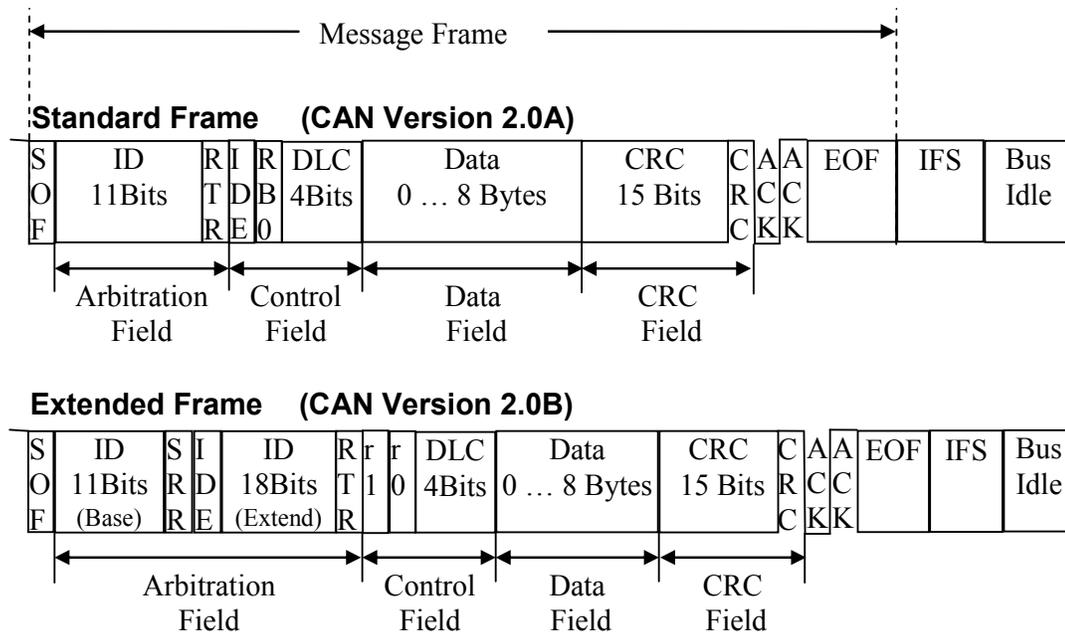


Figure 4.3: Standard and extended frames format in CAN [10]

As shown in Figure 4.3, messages in the CAN standard frame consists of seven different bit fields:

- A Start of Frame (SOF) field. This is a dominant (logic 0) bit that indicates the beginning of a message frame.
- An Arbitration field, containing an 11 bit message IDentifier (ID) and the Remote Transmission Request (RTR) bit. A dominant RTR bit indicates that the message is a Data Frame. A recessive (logic 1) value indicates that the message is a Remote Transmission Request (otherwise known as Remote Frame.) A Remote Frame is a request by one node for data from some other node on the bus. Remote Frames do not contain a Data Field.
- A Control Field containing six bits. The IDentifier Extension (IDE) bit distinguishes between the CAN standard frame and the CAN extended frame. Followed by the Reserved Bit Zero (RB0) bit, which is a reserved bit and is defined to be a dominant bit by the CAN protocol. The four bit Data Length Code (DLC) indicates the number of bytes in the “Data Field” that follows.
- A Data Field, containing from zero to eight bytes.
- The Cyclic Redundancy Check (CRC) field, which guarantees the frame’s integrity, contains a 15 bit cyclic redundancy check code and a recessive delimiter bit.
- The ACKnowledge (ACK) field, consisting of two bits. The first is the Slot bit which is transmitted as a recessive bit and is overwritten as a dominant bit by those receivers which have successfully received the message. The second bit is a recessive delimiter bit
- The End of Frame (EOF) field, consisting of seven recessive bits.

Following the end of a frame is the Intermission Frame Space (IFS) field consisting of three recessive bits. After the three bit Intermission period the bus is recognized to be available, during which other CAN nodes can begin to transmit on it [4].

As mentioned before, a message in the CAN extended frame is likely to be the same as a message in CAN standard frame format. The only difference is the length of the identifier used. The identifier is made up of the existing 11-bit identifier (so-called base identifier) and an 18-bit extension (so-called extend identifier). The distinction between CAN standard frame format and CAN extended frame format is made by using the IDE bit,

which is transmitted as dominant in case of a frame in CAN standard frame format, and transmitted as recessive in case of a frame in CAN extended frame format. As the two formats have to co-exist on one bus, the message in CAN standard frame format always has priority over the message in extended format. That is, if a message collision occurred between two messages with different formats and the same identifier/base identifier, the message with the standard format gains higher priority to transmit on the bus than the extended format message, and will be transmitted first [2].

Since extended frames have a longer identifier than standard frames, the number of possible addresses that can be obtained with extended frames is higher than that of standard frames. For this reason, extended frames are intended to be used in CAN networks comprising a high number of nodes.

### **4.3 Summary**

The CAN protocol has been optimized for systems that need to transmit and receive relatively small amounts of information reliably to any or all other nodes on a network. CSMA/CD allows every node to have an equal chance to gain access to the bus, and allows for smooth handling of collisions. Since the protocol is message-based, not address based, all nodes on the bus receive every message, regardless of whether it needs the data or not, nodes then decide whether the message received should be immediately discarded or kept to be processed. Fast, robust message transmission with fault confinement is also a big plus for CAN because faulty nodes will automatically drop off the bus not allowing any one faulty node to bring down the network. This effectively guarantees that bandwidth will always be available for critical system messages to be transmitted. With all of these benefits built into the CAN protocol and its momentum in the automotive world, CAN is ideal for application to the MR2. The CAN protocol supports two message frame formats, the standard frame format and the extended frame format. Each frame format consists of various fields and their descriptions are provided. The standard format was selected as the CAN protocol format for the MR2 application, since only five nodes are implemented in the network, eliminating the need to use extended frames. Using only standard frames within the CAN control system also gives all messages equal priority on the bus. The following chapter, Chapter 5, describes the design of the control system for the MR2 using the CAN protocol.



## 5 CAN CONTROL SYSTEM DESIGN

---

Following the decision to implement the CAN control system in the MR2, the scope of the communication network needs to be investigated. The length and width of the MR2 is 4.1 metres and 1.7 metres respectively. This means that approximately 10 metres of cable is required to run around the vehicle once. Therefore, to allow for further expansion of the communication system, a total length of 20 metres of CAN bus cable is reserved for the MR2 application. According to the standard chart in Figure 4.1 (Chapter 4) the maximum data transfer rate of 1Mbps can be achieved using a 20 metre long cable.

Selecting a bus data transfer rate is a trade-off between capacity for growth and application performance. Low bus data transfer rates can lead to missed deadlines, performance bottlenecks, and limited growth space. However, high data transfer rates can reduce maximum bus length, amplify susceptibility to noise, and increase demands on both the network interface and the application [12].

In modern vehicles it is common practice that all real-time control safety applications, such as airbags or ABS braking, operate at a very high speed, 250kbps to 1Mbps, where the benefits outweigh the costs. For other general applications within vehicles, including the control of 12Vdc auxiliary loads, the major requirement is that the network reacts relatively fast to user commands, typically around 125kbps [13, 14].

Since the 1992 MR2 does not have airbags or ABS braking, the need for very high data transfer speeds is diminished. This allows the use of a lower data transfer rate within the vehicle. A data transfer rate of 125kbps is used, as it is the common speed for controlling auxiliary loads and is easier to implement in hardware, by eliminating the need for noise filters.

## 5.1 CAN Nodes Allocation

In order to control the MR2's auxiliary loads using the CAN control system, many CAN nodes could be distributed around the vehicle. As a model system, Figure 5.1 shows a main controller (M), which acts as the interface between the user and the network, controlling four nodes (Nd1...Nd4) distributed around the vehicle. Each node handles the operation of a selected set of auxiliary loads adjacent to its location. The loads handled by individual nodes in the model system are:

- **Nd1** – Controls all rear lighting loads (brake, park, indicators, reverse and number plate lights).
- **Nd2** – Controls electric windows (driver and passenger windows).
- **Nd3** – Controls the windscreen wipers.
- **Nd4** – Controls all front lighting loads (headlights, park, and indicators) and headlight motors.

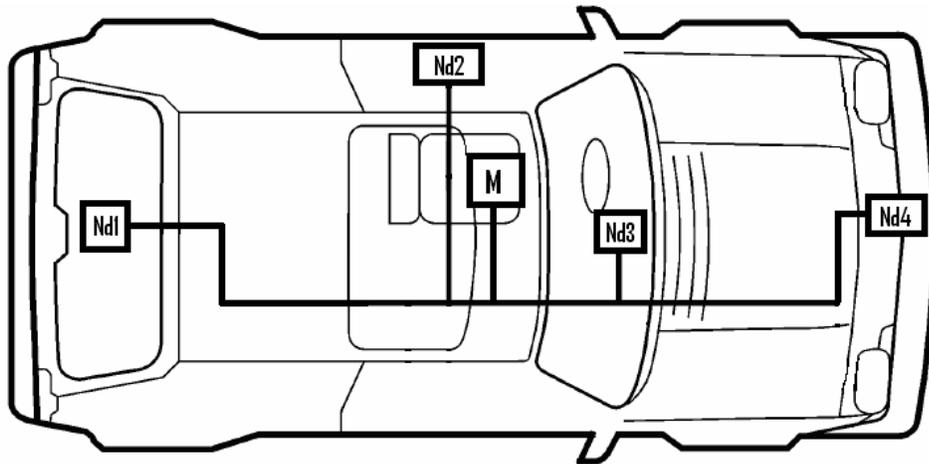


Figure 5.1: CAN nodes distribution

## 5.2 CAN Implementation Method

All CAN node implementations (M, Nd1...Nd4) have a common structure, consisting of a host micro-controller (MCU), CAN controller and CAN transceiver. There are differences, however, in the manner of integration of the above-mentioned components. The CAN nodes can either use a CAN controller to interface to a MCU, or an integrated CAN controller which already has a MCU and a CAN controller built-in as shown in Figure 5.2.

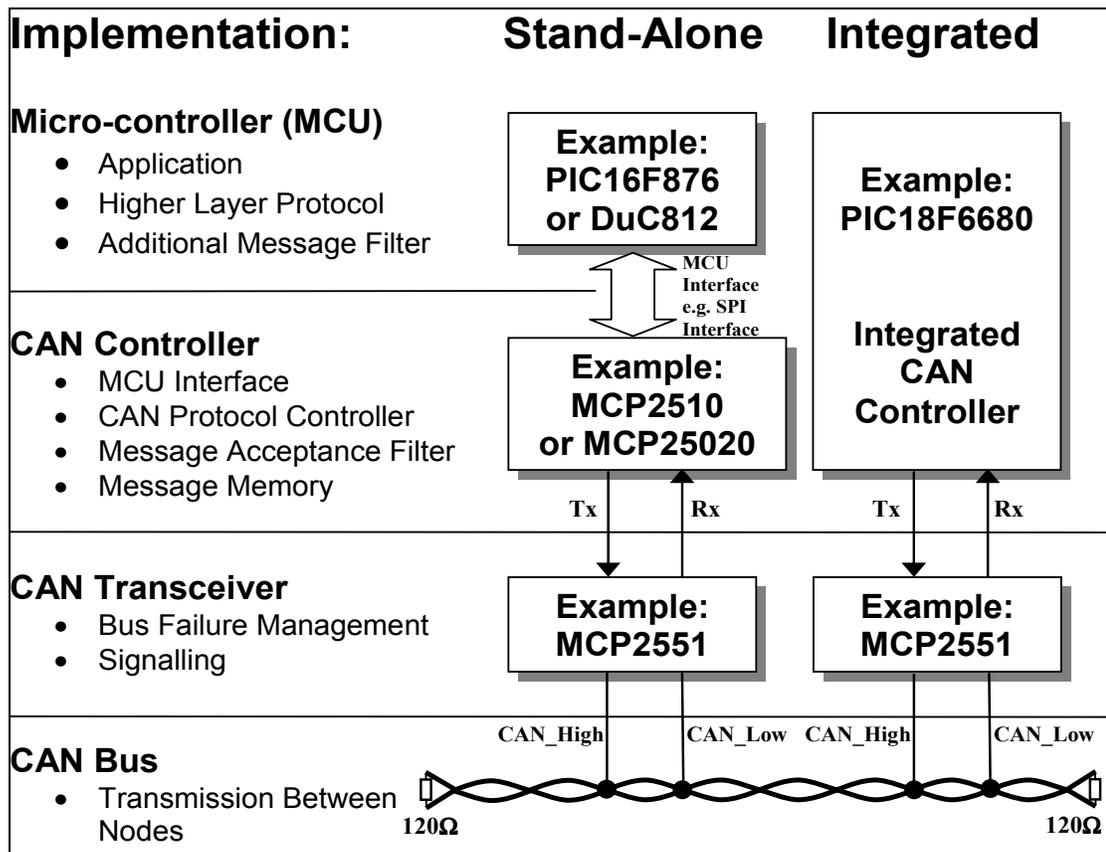


Figure 5.2: Possible structures of the CAN node

The reason for using separate MCU and CAN controller ICs (integrated circuits), Stand-Alone setup in Figure 5.2, is that the CAN controller needs to be initialised by software programs. These programs are stored in the memory of the MCU so the CAN controller can be accessed through the MCU interface, such as SPI (Serial Peripheral Interface). The CAN controller handles all the transmission and reception of CAN messages via the CAN bus. Circuits with separate CAN controller ICs are designed to

interface to different MCUs, allowing the software developed for one system (for example in C language) to be reused in another system, even if the MCU is different.

An architecture with integrated CAN controller, Integrated setup in Figure 5.2, causes a lower MCU load than separate controllers (because of shorter time required to access to the CAN peripheral) and reduces space requirements. On the other hand, software developed for the integrated CAN peripheral of one MCU may not apply to a second MCU with on-chip CAN, especially if the MCUs are supplied by different vendors. Thus, the benefits of a separate CAN controller configuration is more favourable in the CAN control system design, since software can be reused in future designs [2, 10]. In both setups, Stand-Alone and Integrated, the CAN node is coupled with the physical bus by the CAN transceiver IC, which provides differential transmit capability to the bus and differential receive capability to the CAN controller.

### **5.3 CAN Control System Design**

To implement a reliable network in the vehicle, the system should be able to switch auxiliary loads on and off as required and provide robust and accurate control. The structure of the CAN Control System block shown in Figure 3.1, Chapter 3, is detailed in Figure 5.3.

The ISO-11898, CAN High Speed, standard assumes the network wiring technology to be close to a single line structure in order to minimise reflection effects on the bus line. It requires the use of a twisted pair cable, shielded or unshielded, with 120 ohm nominal impedance, and terminated with a 120 ohm resistor at either end to suppress signal reflections along the bus.

There are not many cables in the market today that fulfil this requirement, whereas a shielded twisted pair cable with 60 ohm nominal impedance is readily available and inexpensive. So it was decided to use a 60 ohm cable for this model system rather than a 120 ohm cable as required by the standard. To properly terminate such a cable, one would expect to have 60 ohm resistors connected at its ends. This is true if the CAN transceiver IC was designed for such a setup. Unfortunately, the transceiver is designed to drive a minimum of 45 ohm bus load [15], which cannot be achieved using 60 ohm resistors at

either end, giving a total of 30 ohm bus load. To fix this, the 60 ohm termination resistors were replaced with 120 ohm resistors, as shown in Figure 5.3. This would increase the load to 60 ohms so that the transceivers can function properly but would generate reflected signals at the ends of the cable. However, for a 20 metre long cable with 125kbps data transfer rate, the signal reflections are very small and would not cause significant communication problems.

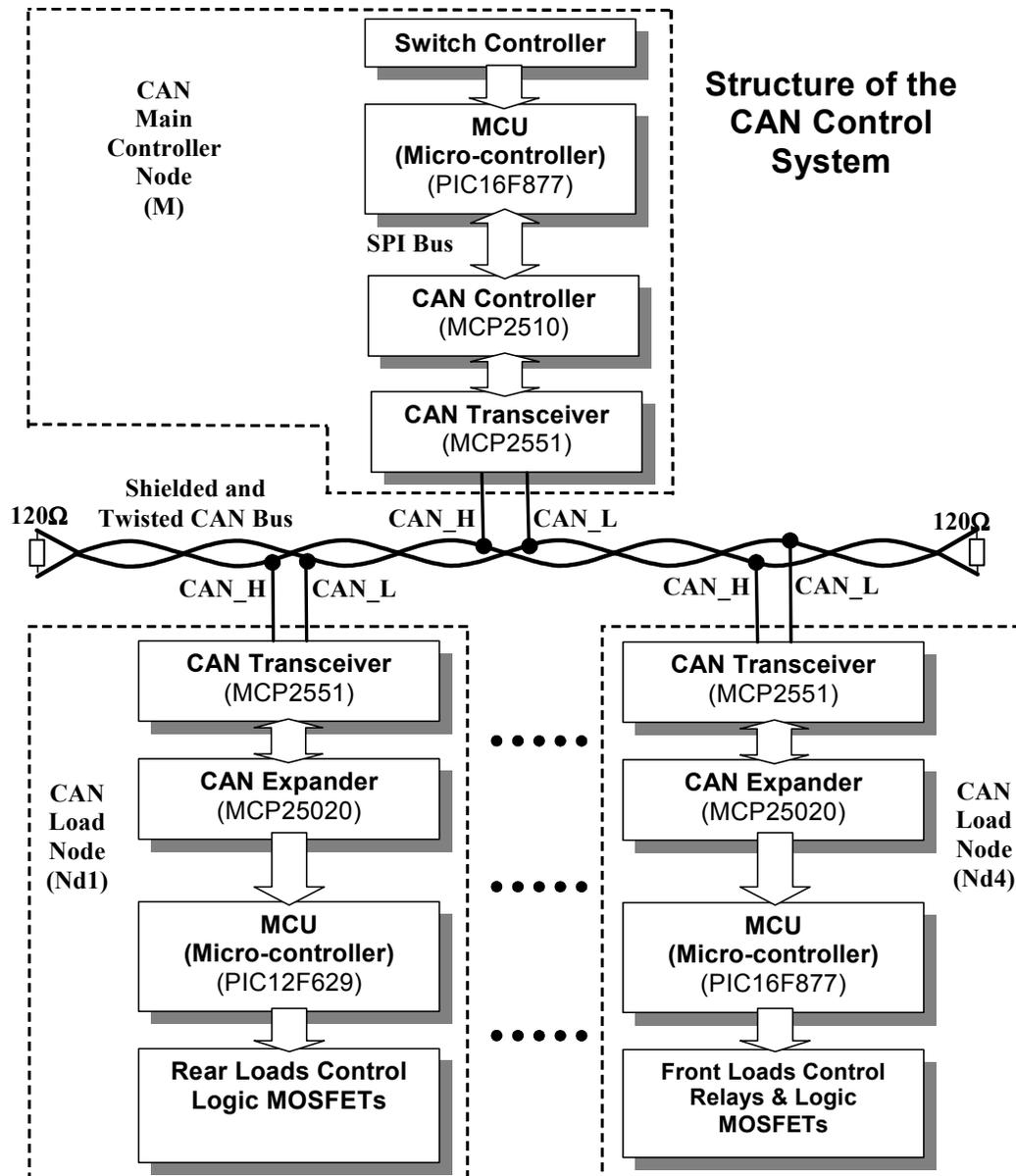
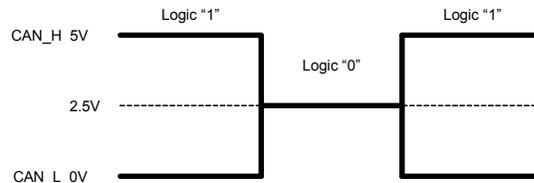


Figure 5.3: CAN nodes details

CAN High and CAN Low (CAN\_H and CAN\_L), represent the differential transmission system for the CAN bus. As shown in Figure 5.4, the differential voltage between CAN\_H and CAN\_L is what is used by the CAN protocol. Initially, when there is no data being transmitted on the bus, both CAN\_H and CAN\_L are set to 2.5Vdc. But when, for example, logic 1 is transmitted along the bus, CAN\_H voltage increases by 2.5V and CAN\_L voltage drops by 2.5V, which gives a difference of 5V, that is logic 1.



**Figure 5.4: CAN differential voltage**

An advantage of such a technique is that if a voltage spike was induced in the cable both conductors would be affected equally and the voltage difference between the two conductors would be maintained, thus providing the CAN network with a good degree of noise immunity.

Since CAN is a differential bus system, all network nodes must have a common ground. This is needed to set up the reference point for the transceivers, so that all messages sent and received are referenced to a common ground. If the node grounds were independent, then any slight ground voltage shift at any of the nodes would disturb communication within the network and cause errors in the system. This would not cause a problem in the MR2, because all auxiliary loads' power supplies have a common ground, Figure 3.4 (Chapter 3), and the twisted pair cable shield has been used to connect all node grounds together, which would also provide any noise imposed on the cable with a path back to the supply.

For the remainder of this chapter every part of the network design will be explained separately, starting with the ICs selection for the individual nodes, followed by a description of the loads and their control methods used within the network.

### 5.3.1 CAN Network ICs Selection

This section explains in detail which ICs have been used in the design and the reasoning behind their choice. It must be noted, in advance, that all ICs (MCUs, CAN controller and CAN transceiver ICs) used in the network are manufactured by Microchip Technology Inc.

#### 5.3.1.1 CAN Main Controller Node

The CAN main controller node acts as the interface between the user and the network, shown in Figure 5.3. That is, the user will only interact with the main controller which, in turn, controls the rest of the network and ensures proper network performance at all times. To be able to send and receive standard CAN messages the main controller has two ICs onboard, the PIC16F877 MCU and the MCP2510 Stand-Alone CAN controller with SPI interface.

The MCU is responsible for monitoring user commands from a set of switches resembling real life switches found in vehicles, while the CAN controller is responsible for handling all messages transferred and received via the CAN bus. When an input is detected, the MCU initiates communication with the MCP2510 via SPI, which then generates CAN messages destined to one or more of the four CAN load nodes. Also, when a message is received by the MCP2510 from the network, communication is initiated again with the MCU, so that the MCU can read and process the message.

The PIC16F877 MCU was chosen because of its large number of input/output (I/O) pins, which are needed to monitor all user inputs, while the MCP2510 CAN controller was chosen because it incorporates an SPI interface, which is supported by the MCU.

#### 5.3.1.2 CAN Load Nodes

For the CAN load nodes, the same two IC structure is used as in the main controller with one difference. As shown in Figure 5.3, all load nodes (Nd1 ... Nd4) have a CAN expander IC, MCP25020, replacing the Stand-Alone CAN Controller IC, MCP2510, used

in the main controller. As the name suggests, the MCP25020 operates as an I/O expander for the CAN system and is responsible for handling all messages transferred and received via the CAN bus. The MCP25020 was chosen for its straight-forward programmability and simplicity of use.

In the load nodes, instead of using an SPI interface between the MCU and the MCP25020, the output pins of the MCP25020 are directly connected to the input pins of the MCU. The reason for this is that the MCP25020 does not incorporate an SPI interface to enable it to communicate directly with the MCU. In this setup, the MCP25020's output pins state provides the MCU with the data needed to identify the command and take action accordingly.

As shown in Figure 5.3, the differences between the CAN load nodes are in the choice of the MCU chipset, which depends on the number of I/O pins needed to drive the controlled loads, and the circuitry that they are designed to interface to. That is, Nd1 uses an 8-pin PIC12F629 MCU that controls the rear indicators (three inputs, two outputs needed), while Nd4 uses a 40-pin PIC16F877 MCU that controls the front loads (seven inputs, eleven outputs needed), shown in Figure 5.3. To illustrate further, Nd1 needs two inputs from the MCP25020 to indicate whether or not indicator switches are turned on via the CAN bus, and two output pins to control the indicators circuitry, to turn the indicator's light bulbs on and off. Further explanation and design detail will be presented in Chapter 6.

A feature of both the MCP25020 and the MCP2510 is that they have masks and filters that, if used, make them capable of receiving messages with several addresses. This broadcast message property can be used to coordinate multiple nodes and implement more complex control functions. For example, in the MR2 this is particularly useful when sending an indicator or hazard lights message, where the message needs to be sent to both the front and rear nodes at the same time.

The MCU in every load node (local MCU) is responsible for controlling specific tasks without the involvement of the main controller. The wipers node could be taken as an example. The driver turns on the wipers and sets them at a specific speed, the main controller sends a command message to the node telling it to turn on the wipers at that speed. The node's local MCU then takes over to provide the on-time off-time of the wipers.

This added local intelligence makes the system more reliable since it decreases traffic on the bus, which in turn decreases input execution times, data transfer delay times and data collisions which can produce errors on the bus.

One may wonder why the CAN expander IC, MCP25020, was not used in the main controller instead of the CAN controller IC, MCP2510. The reason behind this decision is that, besides the unavailability of an SPI interface in the MCP25020, it only comes in one size providing a maximum of seven I/O pins, which would not be enough to handle the large number of inputs and outputs needed by the main controller, but are sufficient for the control of the load nodes. In the CAN control system various ICs were used to simplify the overall structure of the network, while demonstrating the system's flexibility in using different ICs to suit different applications.

### **5.3.2 Loads Description and Control Methods**

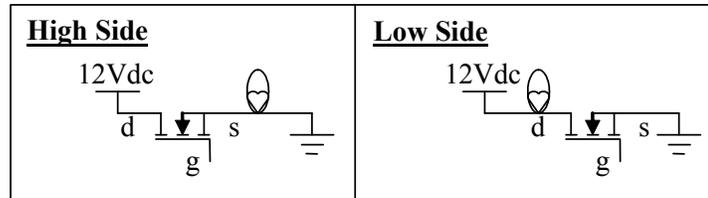
The load nodes in the system are designed to control both resistive and inductive loads. This section examines the design methods used to control the auxiliary loads in the network and explains the functionality of the loads.

#### **5.3.2.1 Resistive loads**

All resistive loads, such as light bulbs, in the network are controlled using logic MOSFETs, which can be operated directly from the MCU without the need for any extra circuitry.

To fully switch on a power MOSFET,  $V_{GS}$  (voltage between gate and source) must be higher than  $V_{DS}$  (voltage between drain and source). This means that if a MOSFET is controlled from the high side it would not work properly, while it can be properly controlled if it is driven from the low side with the source referenced to ground, Figure 5.5 shows both high and low side control setups.

Driving MOSFETs from the low side would seem to cause a safety problem, where all loads would be at 12Vdc potential even if the load was not turned on. This is not considered to be a safety problem since engineers regard a 60Vdc limit as the safe maximum for cars [5]. Worst case low side scenario is that if an accident takes place and the electrical network gets damaged, one or more of the bulbs might short to ground and light up. This too, would not cause a safety hazard for passengers.



**Figure 5.5: High and low side MOSFET control**

Although using low side drive control is considered safe, one non-critical problem has been found. For a double filament bulb connector, such as the brake or headlight bulb, there are only three wires attached to it, two inputs and one ground. This would cause a problem when using the low side setup, since grounding the only ground wire would turn on both filaments. To solve this problem, 12Vdc was applied to the ground wire and the two voltage wires were used as separate grounds, since current direction in bulbs is not important. The only problem with this setup is that double filament bulbs cannot easily be replaced with LEDs, since LEDs are polarized and reversing the current would not turn them on.

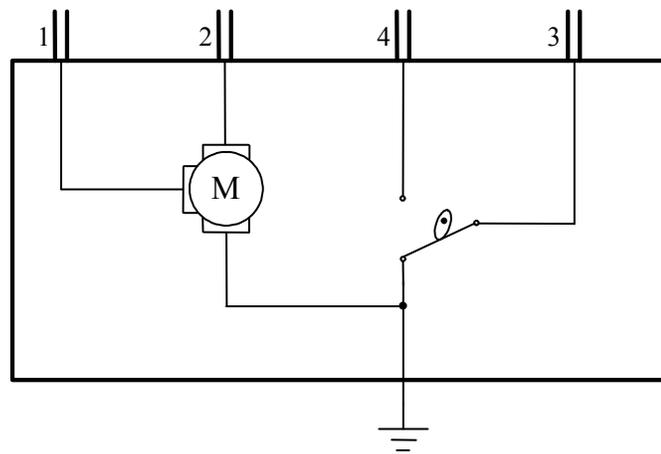
### 5.3.2.2 Inductive loads

Inductive loads, such as motors, are controlled using Single Pole Change-over (SPCO) relays, which make it easier to drive motors in both directions as required by the MR2 motor's applications.

Node Nd2, which controls the electric windows, controls two basic DC motors to raise and lower the windows. This is simply achieved by reversing the current flow direction. However, Nd2 also has current sensors to monitor the current through the

window motors and an interrupt signal is sent to the MCU if over current, caused by the window encountering an obstacle, is detected prompting the MCU to stop the motor.

On the other hand, nodes Nd3 and Nd4 control more complex wipers and headlights motor functions respectively. The wipers motor assembly is manufactured as a single unit and operates at two speeds. The single unit consists of a DC motor and a worm gear. The motor has the worm gear on its armature shaft to increase its output torque and sense when the wipers are in their park position. The block diagram of the wipers motor unit is shown in Figure 5.6.



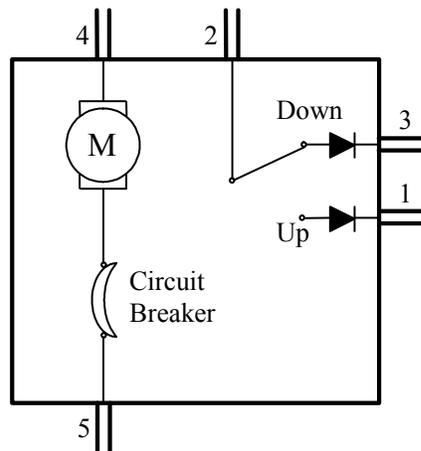
**Figure 5.6: Wipers motor unit block diagram [16]**

If the inputs of the wipers motor are controlled properly, it would rotate in one direction at two different speeds, high and low, and be able to park the wipers when turned off. The two speeds are achieved by having two windings in the motor with different resistances. The resistances generate different pole fluxes in the motor which in turn generate two different back-emfs, and hence two different speeds.

With 12Vdc always applied to pin 4 and keeping pin 3 not connected (floating), the motor can be operated at high speed by applying 12Vdc to pin 1 and keeping pin 2 floating. This would cause current to flow through the higher resistance winding of the motor to ground, rotating its armature at high speed. For low speed, 12Vdc is applied to pin 2 while pin 1 is kept floating, and through the lower resistance winding current flows, rotating the armature of the motor at low speed. While the motor is rotating, at any speed, the worm

gear rotates as well, keeping pin 3 connected to pin 4 at all times except at the park position. At the park position, pin 3 is connected to ground as shown in Figure 5.6. For this reason, parking the motor can be simply accomplished by connecting pin 3 to pin 2. This causes current to flow from pin 4 to pin 3 then to pin 2, running the motor at low speed until it reaches the park position, at which point pin 3 gets grounded and no current flows to pin 2 halting the motor at the parking position as required. Intermittent wiper control using the low speed function of the wiper motor will be discussed in Chapter 6.

Each headlight motor comes as a self-contained unit comprising a single speed motor, circuit breaker, worm gear and two diodes. Figure 5.7 shows a block diagram of the headlight motor unit.



**Figure 5.7: Headlight motor unit [16]**

The headlight motor unit's task is to raise and lower the front headlights of the MR2. The headlight motor unit is designed so that one full worm gear revolution covers the distance needed to fully raise or lower the headlight to its park position. While in operation, pin 2 has to be at 12Vdc potential, and so, to raise the front headlight, pin 4 needs to be connected to ground while pin 1 is connected to pin 5. This rotates the motor in one direction until the headlight is fully raised and then parks, with the worm gear connecting pin 2 to pin 3, Figure 5.7. To lower the front headlight pin 5 needs to be grounded while pin 3 is connected to pin 4, which rotates the motor in the opposite direction until the headlight is fully lowered and parks, this time connecting pin 2 to pin 1, positioned ready for the next raising cycle. The circuit breaker serves to protect the unit in case an obstacle gets in the way of the headlight, such as a person sitting on one of the headlights. If too much current

flows through it before reaching the parking position, the circuit breaker will heat up and open, and once it cools down it would close again until the headlight is fully raised or lowered.

## **5.4 Summary**

Various CAN hardware implementation methods were discussed and a separate CAN controller configuration was chosen for the network nodes. The main controller uses a CAN controller IC, MCP2510, while the CAN expander, MCP25020, was chosen to be used for the remaining four load nodes, to handle all CAN communication to and from the network. The CAN control system was designed to control selected auxiliary loads within the electric vehicle, comprising of one master node controlling four model load nodes providing robust and accurate control of selected auxiliary loads around the vehicle. The construction details of the CAN control system are presented in the following chapter, Chapter 6.



## **6 CAN CONTROL SYSTEM CONSTRUCTION**

---

As discussed previously in Chapter 5, the CAN control system has been designed to control selected 12Vdc auxiliary loads within the electric vehicle. To successfully control the various auxiliary loads in the electric vehicle, the CAN load node's hardware must be designed specifically to satisfy the individual needs of each load, while the software which meets the CAN protocol criteria controls each load.

The communication network has been designed to operate at a maximum data transfer rate of 125kbps, using standard frames only (Chapter 5). To operate the network at higher speeds both the hardware and the software described in this chapter would need to be modified. Also, if extended frames were required, software changes would be necessary.

This chapter serves to explain, in detail, the hardware construction of the individual CAN nodes within the network, along with their programmed software, to successfully control all auxiliary loads according to the CAN protocol.

### **6.1 Hardware**

In order to implement a reliable network in the vehicle, the system hardware should be able to switch auxiliary loads on and off as required and provide robust and accurate control.

This section describes the hardware construction of the CAN control system. Note that all CAN load nodes have the same general construction. The only differences between them are in the choice of the MCU chipset and the circuitry that they are designed to interface to.

In the design of all CAN nodes, the following basic design principles were applied;

- Double sided PCBs were used, with the ground plane on both top and bottom sides. This reduces the ground inductance within the circuit, thus minimizing self-generated ground noise and EMC emissions from the circuit.
- All ICs throughout the network have decoupling capacitors placed at their voltage input rails to compensate for their power demand voltage sag and to filter out noise. The decoupling capacitors were placed as close as possible to the ICs to minimize stray inductance produced by the PCB tracks.
- Components onboard the individual CAN nodes were placed as close as possible to each other, in order to minimize the lengths of track used, thus, minimizing stray inductances.
- All PCB components used were through-hole components, since through-hole components are cheaper and more readily available in the department than surface-mount components.
- IC sockets were used for all ICs in the network. Using them avoids damaging the individual PCBs, by de-soldering and soldering, whenever an IC needs to be replaced. This is especially useful during the prototype development phase.
- Current limiting resistors were used wherever possible to protect the circuitry from being damaged in case a fault occurs within the network.
- All logic MOSFETs used in the network were chosen to have very low  $R_{DSon}$  (Drain-to-Source on-resistance) to minimize the power loss when conducting. The MOSFETs used were IRLZ34N, from International Rectifier, and have a maximum  $R_{DSon}$  of 0.06 ohms.
- All relays used were Single Pole Change-Over (SPCO) relays, and were chosen for their low control current demand, 40mA. They are made by Finder and operate from 12Vdc with a maximum rated current of 6A.
- Calculations determined that heat sinking was not required for any of the MOSFETs, as a result of their adequate thermal characteristics (high power rating). However, as a safety precaution, two of the five nodes have heat sinks placed on their 5Vdc regulators to prevent them from overheating. This is due to the higher number of 5Vdc dependent components onboard.

As well as common design principles, some specific components were used throughout the network, and can be found on all CAN nodes within the network;

- Male DB9 connector, used to connect each node to the CAN bus.
- RJ45 connector, used to connect each node board to the Microchip MPLAB-ICD (In-Circuit Debugger) for on-board software debugging of the PIC MCUs. The MPLAB-ICD tool is used to enhance the code development and software debugging process in the node boards, and allowing the PIC MCU to be easily programmed after each node has been manufactured
- A voltage regulator, which provides 5Vdc output, from the 12Vdc input, to power all digital components in the circuit.
- Power LED, indicates the presence of power in the circuit.
- Reset button, to manually reset ICs onboard the CAN nodes.

In the remaining sections, the detailed hardware construction of each CAN node is explained separately, starting with the CAN main controller and ending with the CAN front load node. The detailed schematic diagrams of the CAN nodes can be found in Appendix C.

### **6.1.1 CAN Main Controller Node**

This section describes the CAN main controller node's hardware construction to meet the requirements needed to send and receive user dependent CAN messages.

The detailed block diagram of the main controller construction is shown in Figure 6.1. This block diagram shows the interconnections between the various parts within the main controller. Since the main controller acts as an interface between the user and the rest of the CAN control system, it incorporates a set of switches that mimic those found in a vehicle to control the various 12Vdc auxiliary loads. Single Pole Double-Throw (SPDT) toggle switches were used to replace single operation switches (on-off). Onboard the main controller SPDT switches control the brake, reverse, high beam and hazard lights as well as the wipers' washer motor load. In dual operation cases, where no two switches could be on at the same time, such as the window switches, three position Single Pole Centre-Off (SPCO) toggle switches were used. The SPCO switches control the driver's side and

passenger's side electric windows, left and right indicators and the wipers' intermittent speed control. For the rest of the functions, where multiple operation switches are needed, two four position rotary switches have been used to replicate rotary switch functions. One rotary switch controls the park and headlights, and headlight motors, while the other controls the windscreen wiper operation. In total 19 switches are required to control all 12Vdc auxiliary loads within the model network.

The PIC16F877 MCU, shown in Figure 6.1, stores the software in memory and reads the settings of the 19 switches, then initiates communication with the CAN controller IC, MCP2510, via the SPI interface to send CAN messages. In coordination with the MCP2510, the MCU is also capable of replying to messages received from another node on the CAN bus. The MCP2510, is responsible for handling all messages transferred to and received from the CAN bus.

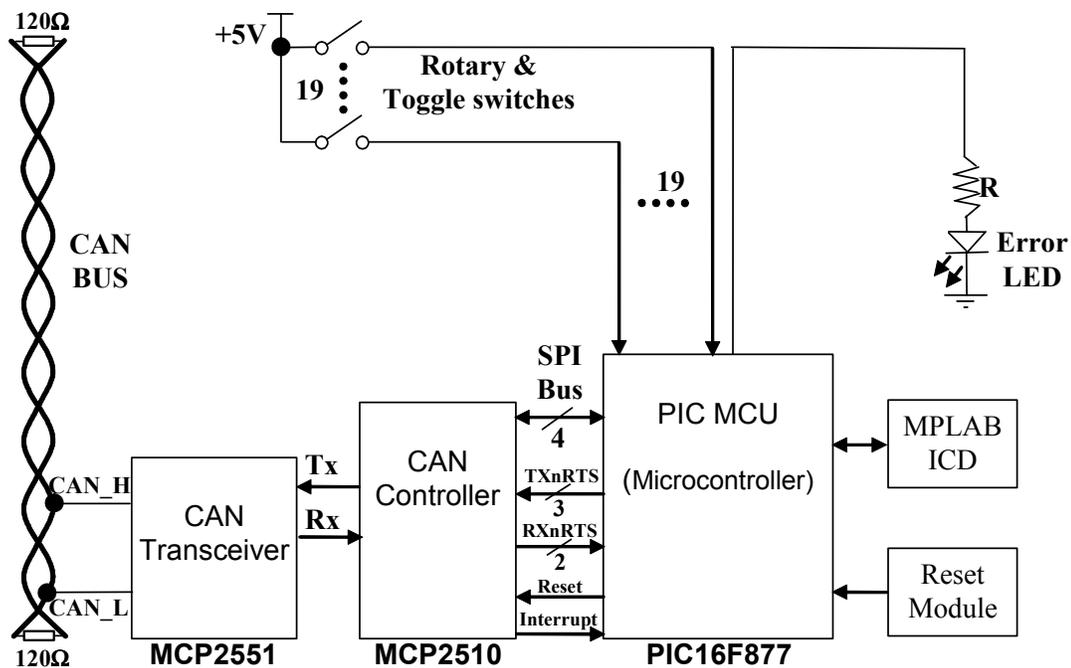
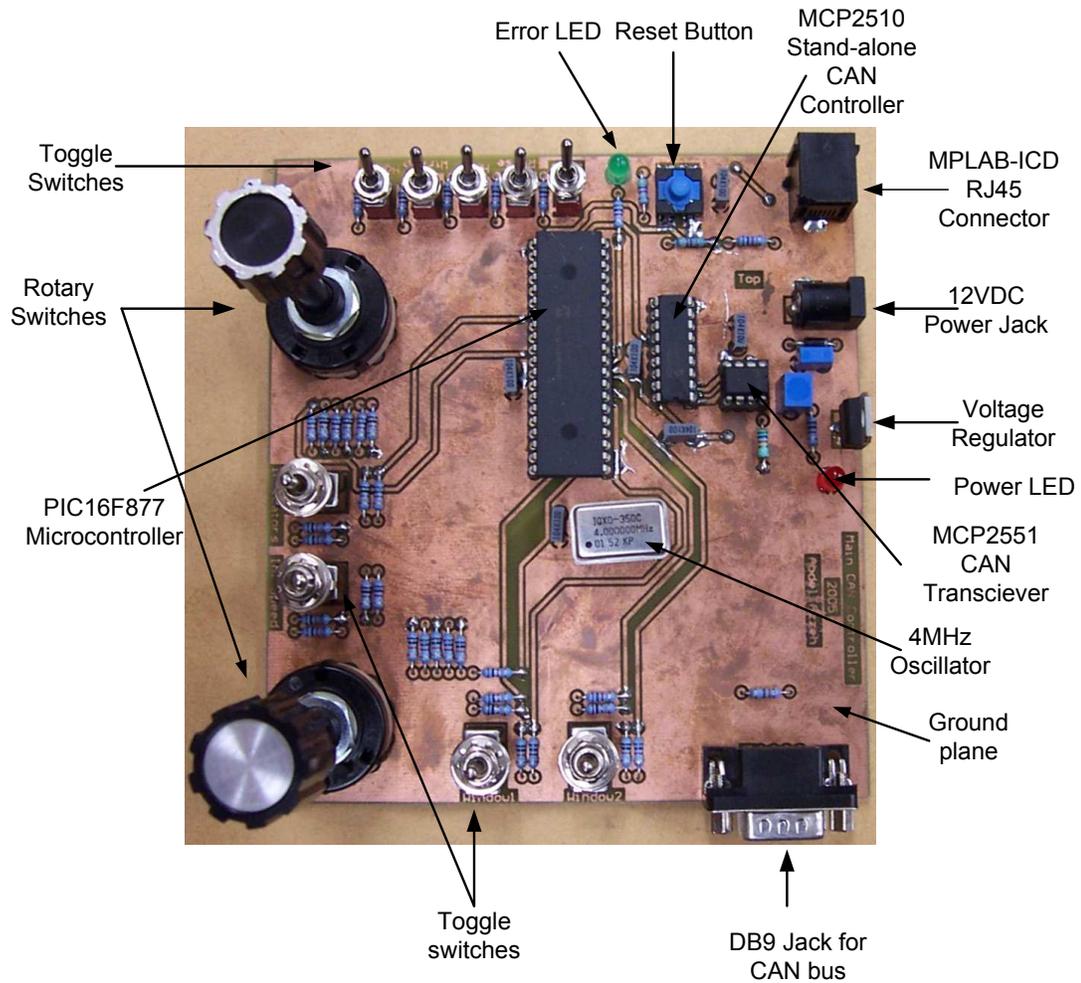


Figure 6.1: CAN main controller node block diagram

Turning on any of the 19 switches generates a command message with an identifier that matches that of the node controlling the required load. For example, toggling the brake light switch generates a message with an identifier that matches that of the rear node only. This means that the rear node will be the only one to accept and process the command of turning the brake lights on and off.



**Figure 6.2: CAN main controller node board**

Figure 6.2 shows a photograph of the completed CAN main controller node. 12Vdc is fed to the circuit through the 12Vdc power jack, which is then regulated, via the voltage regulator, to give 5Vdc that powers the rest of the circuit. A push-to-close reset button is available for manual reset of the PIC MCU. The Error LED is used to indicate that an error has occurred within the CAN control system.

### 6.1.2 CAN Rear Load Node

The CAN rear load node is responsible for controlling all rear auxiliary loads in the electric vehicle. All loads located at the rear end of the vehicle are resistive loads, light bulbs, in which brake, reverse, indicators, parking and number plate lights are included. Logic MOSFETs were used to control all rear resistive loads. Figure 6.3 shows a block diagram of the CAN rear load node.

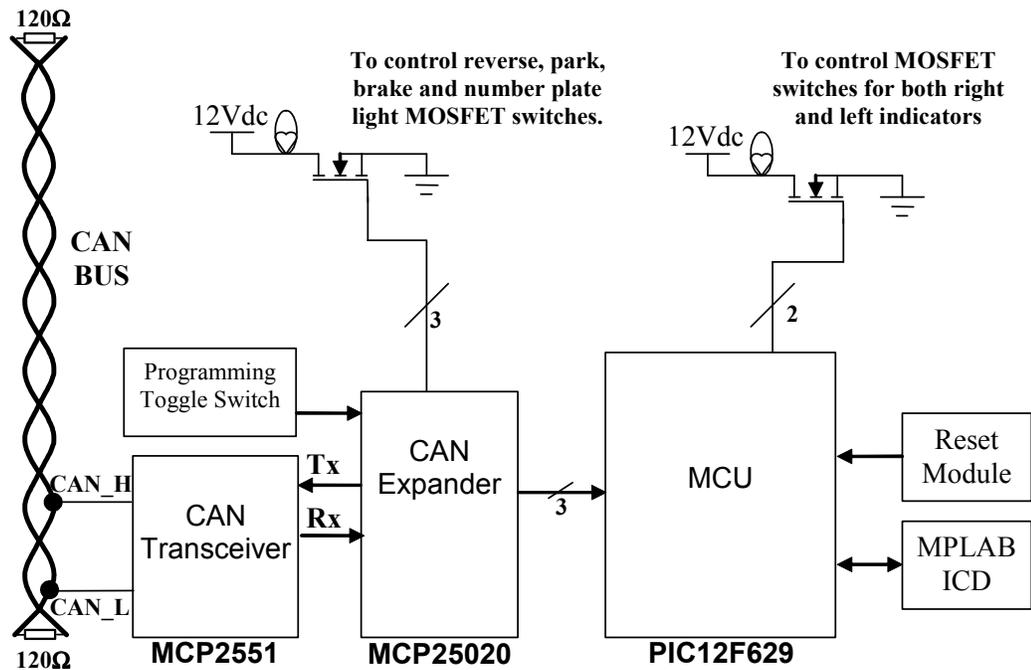


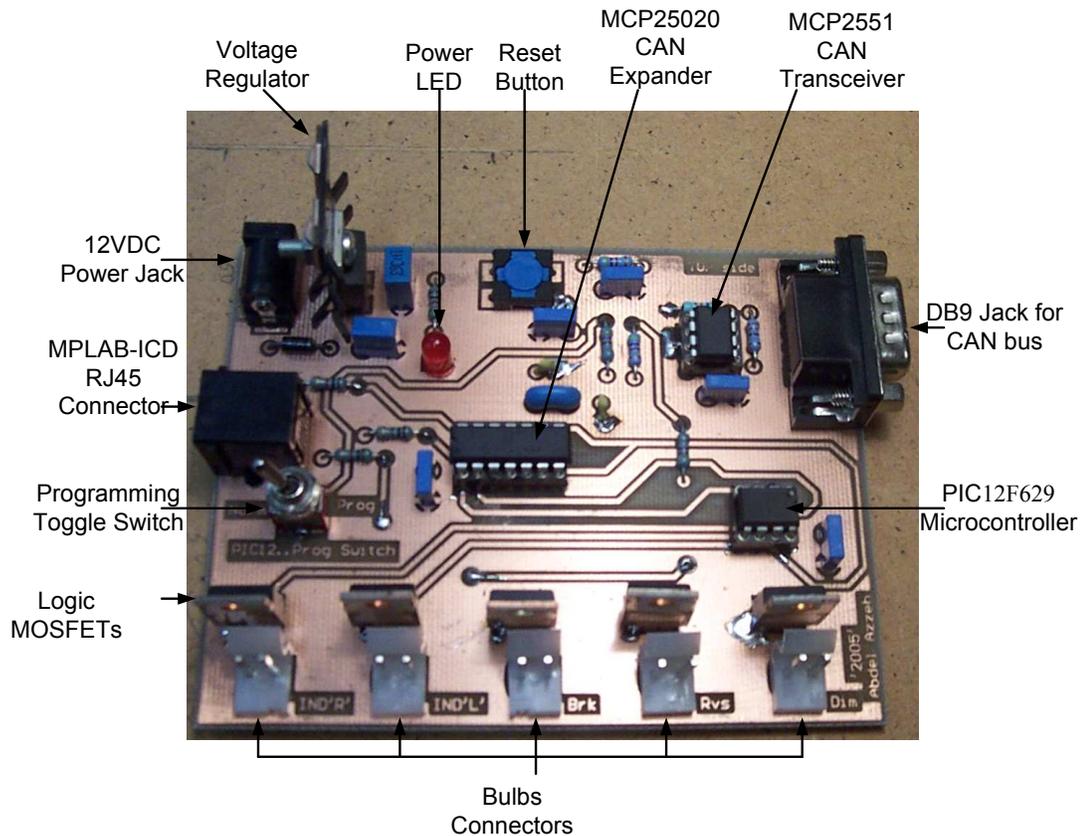
Figure 6.3: CAN rear load node block diagram

The CAN rear load node uses an 8-pin PIC12F629 MCU, since only three inputs and two outputs are required for this node. The MCU provides the on-off time, blinking function, for the rear indicators. The brake, reverse, parking and number plate lights are directly controlled from the CAN expander IC MCP25020, which operates as an I/O expander for the CAN control system and is responsible for handling all messages transferred to and received from the CAN bus.

The reason the MCU only controls the indicators, and none of the other rear loads, is because they are the only functions that require extra processing, timing and synchronization, rather than just being either on or off. An advantage of this setup is that

the time taken to turn on the brake lights will be shortened, since no extra processing time by the MCU is required. This would count as a safety measure as well, since brake lights are the most critical function of all rear node auxiliary loads and need to have the fastest response time than any other function.

In this node, the pins needed to program the MCU are also used as inputs from the MCP25020 to the MCU. For this reason, a programming toggle switch is used to keep the MCP25020 in the reset state when programming the MCU using the MPLAB-ICD. In this reset state, all the MCP25020 pins need to be held at high impedance, tristate, so that programming the MCU would not be interrupted at any time. Figure 6.4 shows a photograph of the completed CAN rear load node.



**Figure 6.4: CAN rear load node board**

### 6.1.3 CAN Wipers Load Node

This section describes the hardware construction of the CAN wipers load node. As the name suggests, this node is responsible for controlling the operation of the windscreen wipers.

The block diagram in Figure 6.5 shows the general construction of this node. The overall construction of this node is similar to the rear node described in section 6.1.2, except that in this node, instead of resistive loads, only inductive loads are controlled using both logic MOSFETs and relays. The wipers' water pump load is controlled using a MOSFET, in the same fashion as controlling a resistive load, because no current direction change is required for the water pump motor. Controlling the wipers' motor, as detailed in Chapter 5, is done by controlling a set of three relays, needed to be able to run the motor at two different speeds, and to provide the wiper park facility. Since the functions of the loads in this node are interrelated, synchronization is required, all of this node's loads are controlled by the MCU onboard, the PIC16F876. Figure 6.6 shows a photograph of the completed CAN wipers load node.

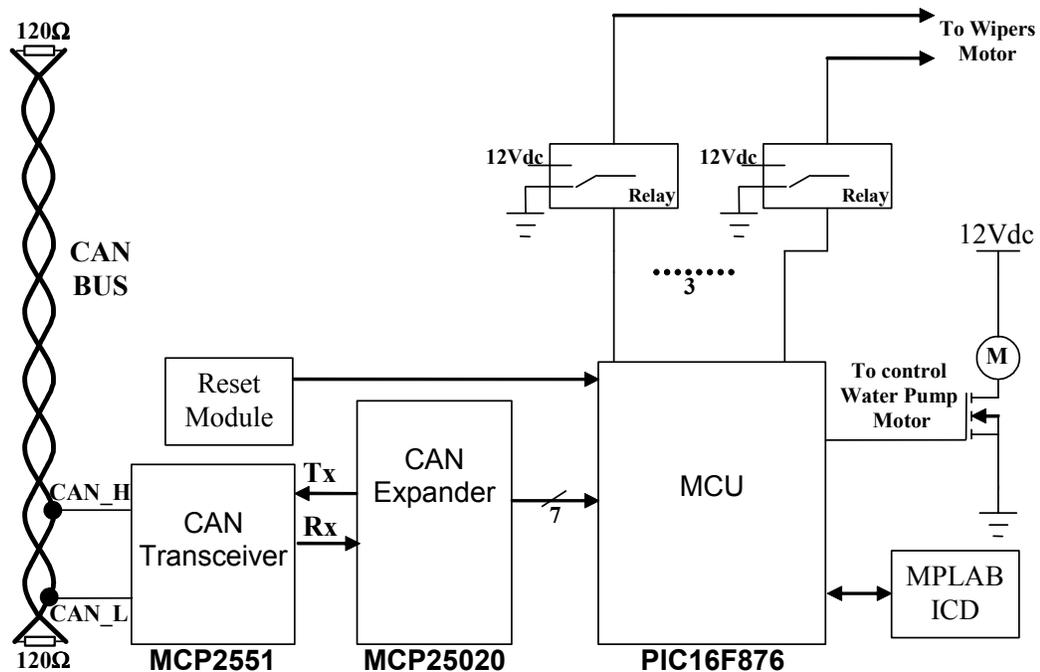
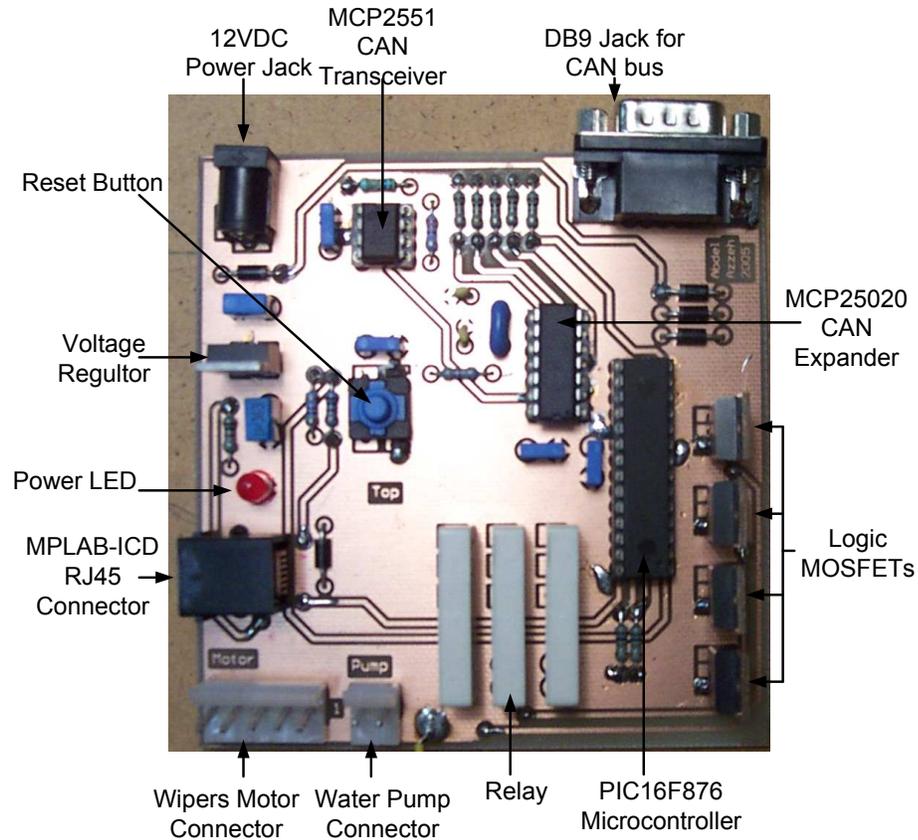


Figure 6.5: CAN wipers load node block diagram

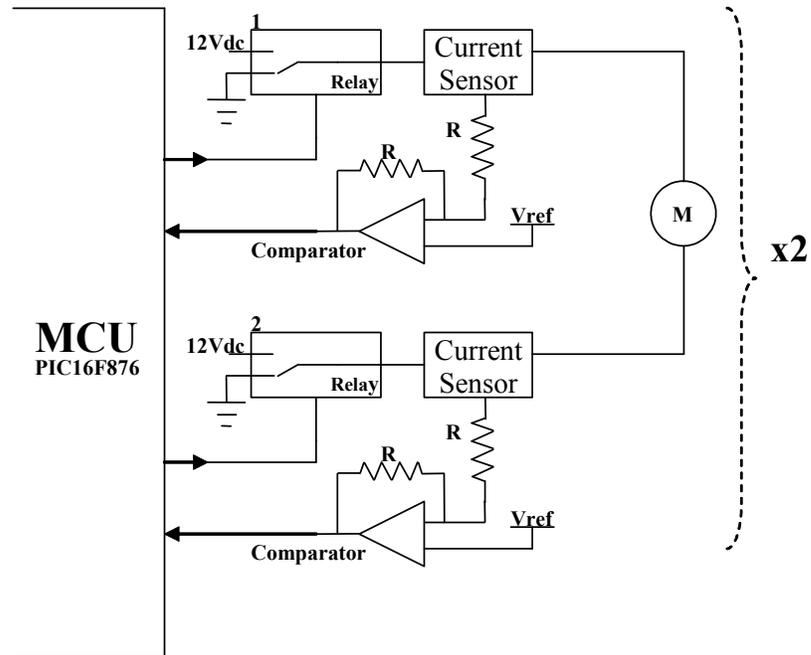


**Figure 6.6: CAN wipers load node board**

#### 6.1.4 CAN Electric Windows Load Node

The main task of the CAN electric windows load node is to control both electric windows in the MR2. Depending on the inputs to the MCU, which describes the command sent to the node from the main controller, the MCU raises or lowers any of the two windows. This node also has protective current sensors which monitor the current through the motors.

This node uses a PIC16F876 MCU to control the electric windows. Since the rest of the circuit is almost identical to that in Figure 6.5, only the connections between the MCU and the loads are shown in the block diagram in Figure 6.7.



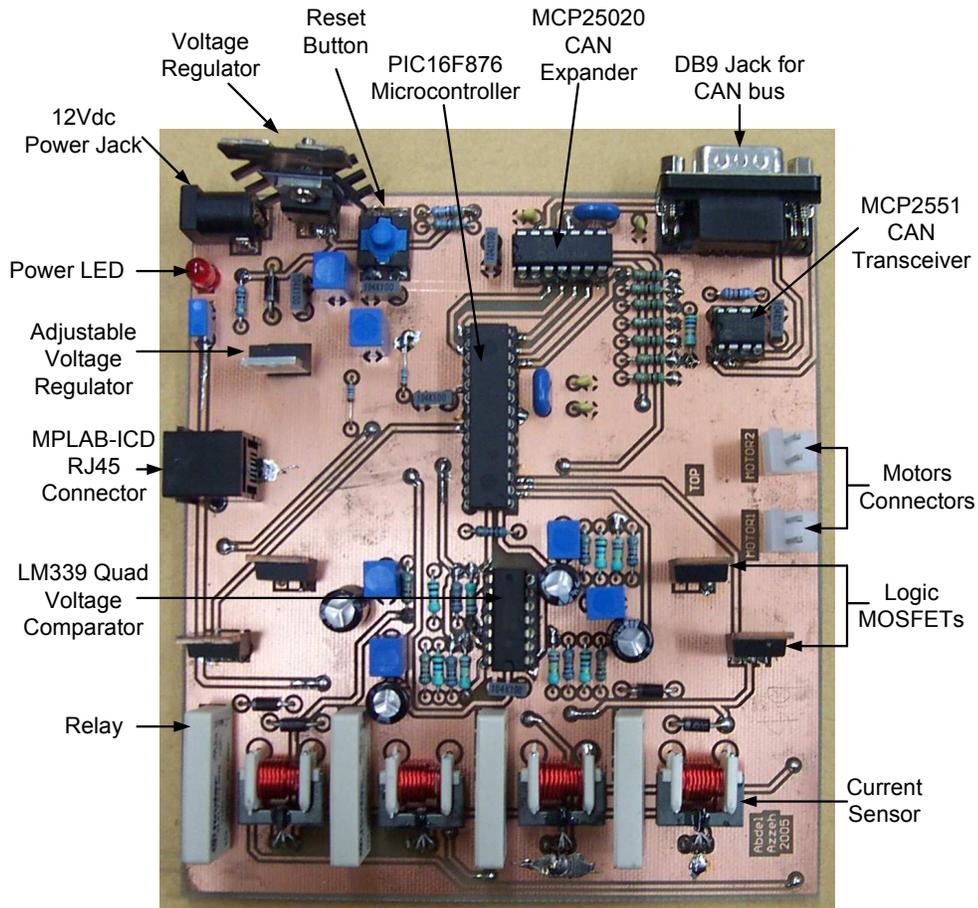
**Figure 6.7: CAN electric windows node block diagram**

From Figure 6.7, it can be seen that the relay's outputs can be connected to either ground or 12Vdc, depending on the control input from the MCU. The Hall Effect current sensors, from Raztec (NZ) Ltd., give a voltage output proportional to the current passing through them and voltage comparators, LM339, are used to compare the output voltage of the current sensors with a reference voltage  $V_{ref}$ , supplied by an adjustable voltage regulator, LM317. Two current sensors are used to control each motor because reversing the current direction across the sensor causes it to output a negative voltage (0 to -5Vdc). If a -5Vdc supply was available in the circuit, only one sensor would be sufficient to control each motor. Also, in this node relays could have been replaced with a semiconductor bridge circuit to avoid reliability issues, but were found to be more expensive.

When voltage is applied to the input of relay 1, its output pin connects to the 12Vdc rail, and while no input is applied to relay 2, its output pin would be connected to ground. This makes power flow from the 12Vdc of relay 1 to the ground provided by relay 2, to turn the motor in one direction, and vice versa.

When a current of 4A is exceeded through either current sensor, the voltage comparator output switches high and the MCU switches the motor off. This indicates that the window has been raised or lowered to its limit, or has encountered an object while being

raised and can go no further and is considered to be a safety measure. The resistors surrounding the comparators provide hysteresis for the comparators to avoid multiple switching at the comparator's output. A photograph of the completed CAN electric windows load node is shown in Figure 6.8.



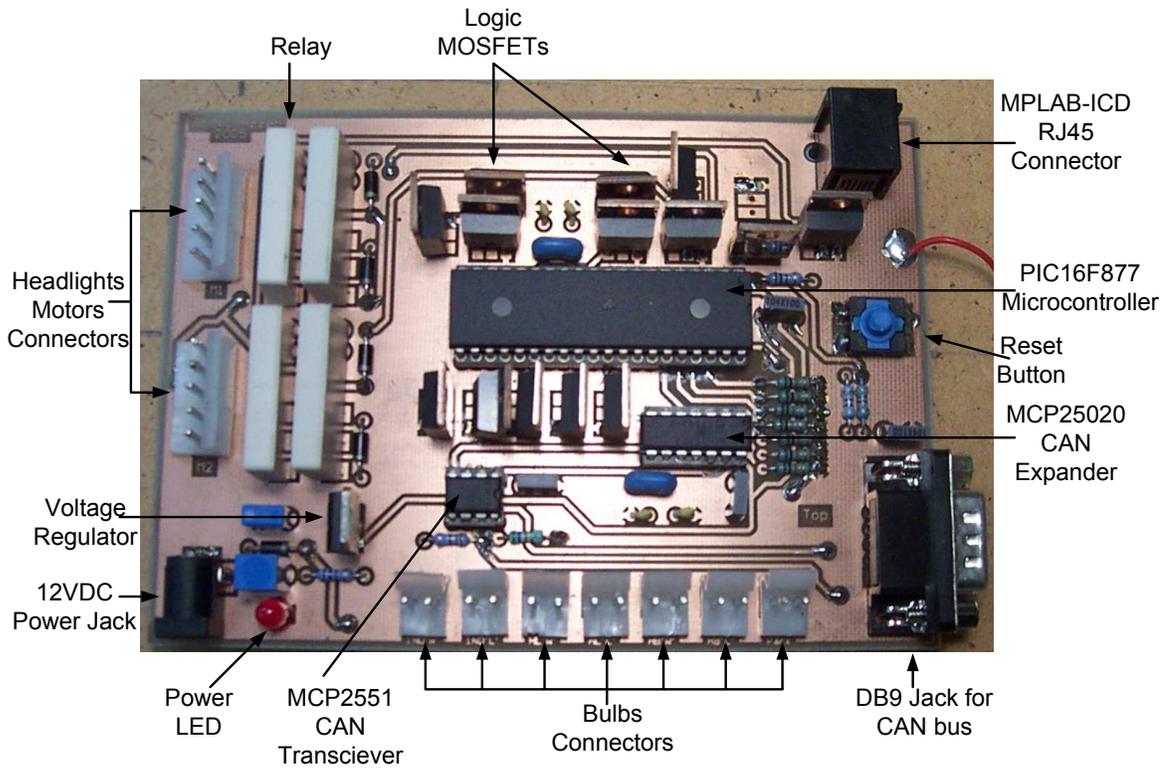
**Figure 6.8: CAN electric windows load node board**

### 6.1.5 CAN Front Load Node

This node comprises both logic MOSFETs and relays to control the front loads. It controls the front park lights, indicators, headlights and the headlight motors. Figure 6.9 shows a photograph of the completed front load node. The loads in this node are controlled in the same manner as in the previous nodes, that is the motors are controlled using relays to raise and lower the headlights, while the light bulbs are controlled using logic

MOSFETs. The MCU, PIC16F877, onboard this node coordinates the operations and timing needed to reliably control the front loads.

Since, as mentioned in Chapter 5, the headlight motors assembly already contains a circuit breaker, current sensors are not needed to protect them, as in the case of the CAN electric windows node.



**Figure 6.9: CAN front load node board**

## 6.2 Software

The software for the CAN nodes is written in C and installed onto the PIC MCUs using Microchip's MPLAB IDE. Full listings of the software are included in Appendix D.

This section describes the method used in programming the CAN control system followed by the software behavioural structure for the CAN nodes.

### 6.2.1 Software Methodology

There are a few different approaches to designing a network of independent nodes, such as the CAN control system for the MR2. The difference is primarily in the frequency of communication between the independent nodes, to determine if any of the nodes in the network were missing or have encountered a severe error and shut down. In some industrial CAN control systems, such as security systems, the nodes are setup to send messages to the main controller at predefined intervals to confirm their presence and functionality in the network. This is done so that if any node is removed or tampered with, an error would be generated triggering an alarm.

In the MR2's CAN control system it was decided that the main controller would rely on the acknowledgement messages sent back from the nodes to confirm their presence and functionality. The philosophy in implementing the CAN control system in this way is to minimize traffic on the bus, and since the vehicle is a contained environment, network nodes are not expected to be missing. If a node was disconnected for some reason or completely shut down, then when a command message is issued by the main controller to that node, the node would not respond with an acknowledgement message, which would prove that the node is missing, and an error would be issued.

If the system is to be extended to include the braking function, a CAN node would only control the brake booster, and a physical mechanical connection would still remain to ensure passenger's safety in case the CAN node fails.

As mentioned in Chapter 5, all CAN nodes have masks and filters which allow them to receive specific messages destined for them and to discard others. The main controller is capable of sending and receiving messages to and from all load nodes. The load nodes, however, are only capable of receiving command messages from the main controller, and responding with only an acknowledgment message (ACK) if the received message was successfully processed, or an error message if not. Also, the front and rear nodes have a common ID. The common ID is used to broadcast messages to both the front and rear nodes from the main controller at the same time, used for indicators, hazard lights and park lights. All CAN nodes are programmed to read and process messages with specific IDs and they are detailed in Appendix D.

## 6.2.2 CAN Main Controller Software

The main controller software flow chart is shown in Figure 6.10. On power-up the registers within the MCU are initialised. The first part of this process involves setting up all the I/O pins of the MCU as either inputs or outputs depending on their function, and configuring the SPI bus for serial communication with the MCP2510 and initializing all registers within the MCP2510, by writing the appropriate information to the proper registers.

After the initialisation process is completed, the MCU starts to read the status of the input pins, 19 switches. When the information is received the MCU checks to see if the received input status has changed from the previous setting. If the status has changed, the program enters the “message transmission” loop to generate a CAN command message according to input states and transmits it onto the CAN bus as shown in Figure 6.10. After the message gets transmitted successfully onto the bus, the MCU will then ask the MCP2510 to check if there is any message available to be received from the CAN bus, by bringing the system to the “Listen to the bus” mode. In the “Listen to the bus” mode, the system constantly monitors the CAN bus for messages to be received. At this point either an acknowledgement or an error message is expected in response to the sent message. The program enters the “message reception” loop if a positive answer is received, otherwise, a counter “i” is incremented and the message is sent again. If the node does not reply to the message after it being sent five times, it is assumed that the node is absent from the network, and the Error LED, pointed out previously in 6.1.1, is turned on and the MCU returns to reading the input pins status awaiting the next command message transmission.

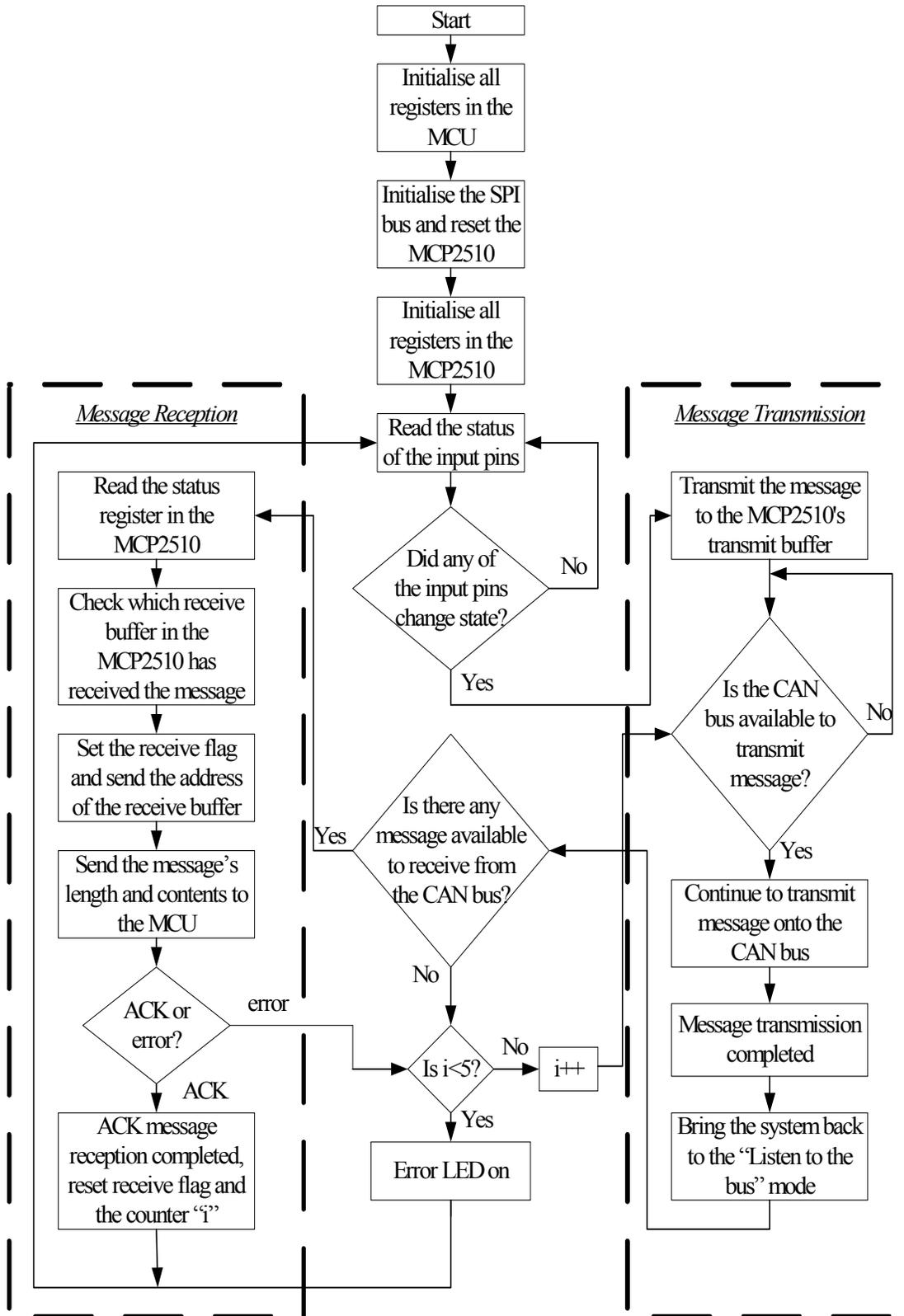


Figure 6.10: CAN main controller software flowchart

In the message reception loop, after the received command message is checked with the filter, the MCU then reads the status register to see which receive buffer has received the message. The next step of the process is to set the receive flag and send the register address information back to the MCU for message reception. After the message is received, the MCU will read the message and determine whether it is an ACK or an error message. If an error message is received and the counter is less than five, the message is sent again. Otherwise it is assumed that the node is defective and the Error LED is also turned on. If an ACK message was received, however, the receive flag and the counter “i” are reset for the next message transmission loop, triggered by a change in the input pins status.

### 6.2.3 CAN Load nodes Software

The load nodes software flowchart is shown in Figure 6.11. On power-up the registers within the MCU and the MCP25020 are initialised. The first part of this process involves setting up all the I/O pins of the MCU and the MCP25020 as either inputs or outputs depending on their function. The MCP25020 is then brought to the “Listen to the bus” mode waiting for a command message from the main controller. If a message is available on the bus, it is read and accepted if its identifier is matched to the filter, otherwise it is discarded and the system returns to monitoring the bus. Upon accepting the message, its contents are processed.

If the message is successfully processed, the appropriate output pin states are changed according to the message contents, and an ACK message with the appropriate ID is sent back to the main controller, informing it that the message has been successfully processed. On the other hand, if an error occurs while processing the message, such that the message was invalid, an error message is sent back to the main controller, which in turn resends the message. The system is then brought back to the “Listen to the bus” mode, waiting for its next command message from the main controller.

As mentioned in chapter 5, the MCP25020 output pin states provide the MCU onboard every node with the data needed to identify the command and take action accordingly. That is, the MCU onboard every node is programmed to monitor its inputs, MCP25020 outputs, continuously. On a change of inputs state, the MCU identifies the new

inputs state and processes a software function controlling a specific load, which corresponds to the new pins state. For example, from Figure 6.3, the MCP25020 receives a command message with an ID that matches its filters. The MCP25020 processes the command and sets its output pin one high, and send an ACK onto the CAN bus. The local MCU, while monitoring its inputs, detects the state change of the input pin which is directly connected to output pin one of the MCP25020. This change triggers the MCU to process a software function, controlling logic MOSFETs, that turns on the left indicator, wait for around 400ms then turn it off again and so on, generating the blinking effect. Once that same input pin changes state again, this time to low, another software function will be processed turning the indicator completely off.

The software for each load node is written to duplicate real life functions in the MR2. The wipers intermission function has 3 settings, controlled by an SPCO switch on the main controller, providing low (10 seconds intermission), medium (6 seconds intermission) and high (2 seconds intermission) speeds using the low speed function of the wiper motor. The park relay of the wipers node is always energized when the load node board is on standby. This means that if the wipers were not parked properly when the vehicle is turned off, wipers stopped somewhere along the windscreen, they would be parked when the vehicle starts again. Also, the front node is programmed to turn on the headlights before raising them, which is what was observed when testing the MR2.

As mentioned in Chapter 2, CAN is an asynchronous protocol. This means that all nodes' clocks are independent of one another, eliminating the chance of synchronizing two functions on separate nodes together. In this prototype, this characteristic does not affect any function except the indicator and hazard lights. As mentioned before, the front node controls the front indicators along with the rest of the front loads, while the rear node controls the rear indicators as part of the rear loads. Since their clocks are independent, there bound to be a mismatch between the front and rear indicators. Although the mismatch would not be noticeable when the indicators or hazard lights are first switched on, the mismatch would accumulate with time to reach a significant and visible time difference between the front and rear indicators. This complication will be looked at in detail in the next chapter, Chapter 7.

The MCU onboard each load node has the responsibility of controlling the load's hardware properly, and their software is not discussed in this chapter, but is rather well commented for easier understanding, and can be found in Appendix D.

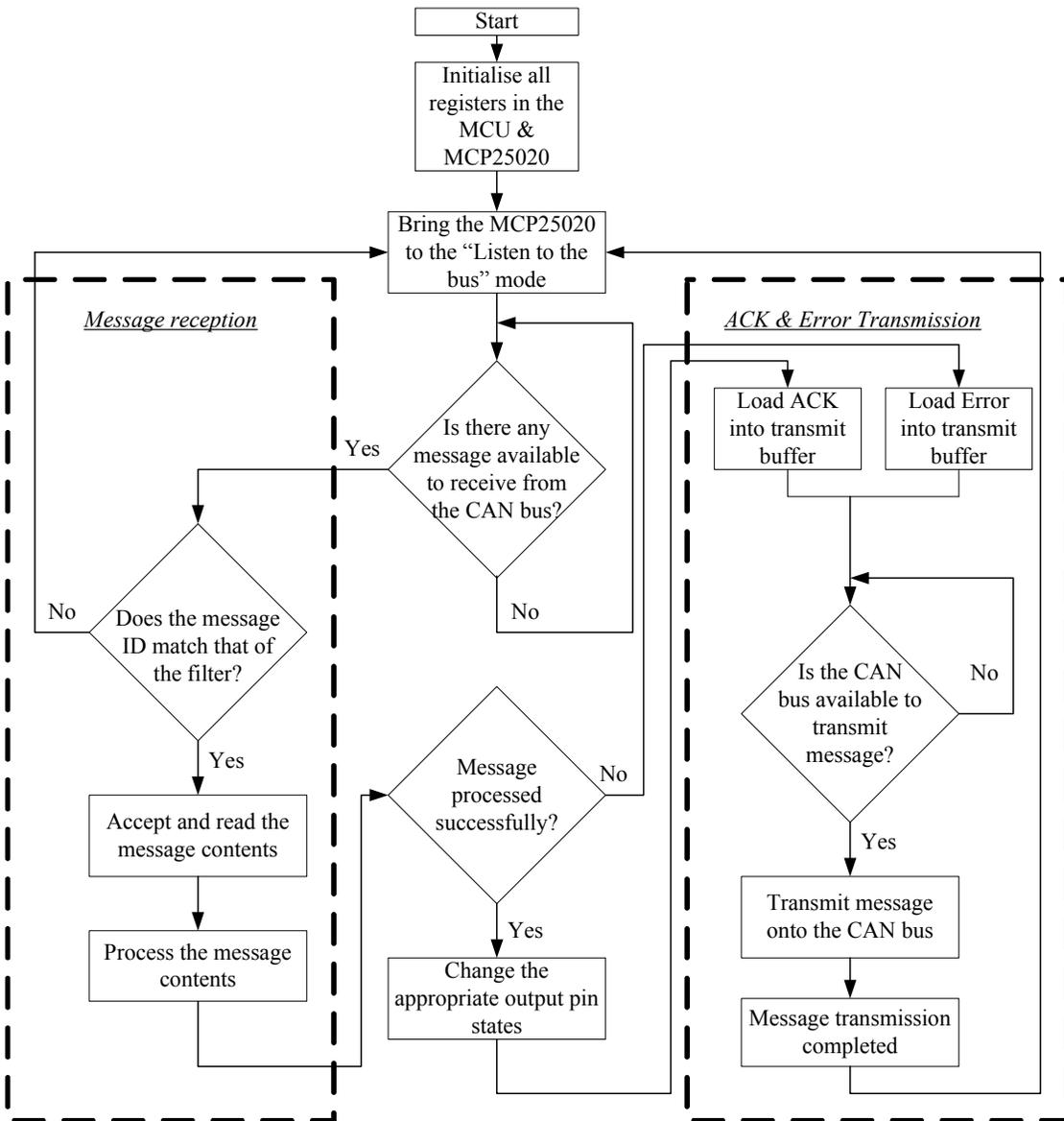


Figure 6.11: CAN load node software flowchart

## 6.3 Summary

The model CAN control system is developed and implemented to control selected 12Vdc auxiliary loads in the MR2. The system hardware is broken down into two main parts; the CAN main controller node and the CAN load nodes.

The main controller has the ability to monitor 19 switches and automatically generate CAN messages based on their status. These messages are then transmitted to the CAN load nodes via the CAN bus. The main controller is also capable of detecting the absence of any of the load nodes from the network.

The CAN load nodes have a different configuration and function to the main controller. They receive messages from the main controller, process them and inform the main controller whether the messages were processed properly or not. The only differences between the CAN load nodes are in the choice of MCU chipset and the circuitry that they are designed to interface to.

The software for the main controller and load nodes are written in C language. The software is written so that the main controller controls and monitors all load nodes within the network. Where the load nodes process the main controller's messages, and ensure proper and reliable control of the loads.

The model CAN control system boards have been developed and constructed successfully. The next step of the project is to test the network and evaluate its performance. Performance testing details are presented in the following chapter, Chapter 7.



## 7 PROTOTYPE PERFORMANCE

---

Experimental tests were carried out to evaluate the CAN control system. This chapter presents the results of tests confirming the correct operation of the model CAN control system. Waveforms, which were captured during the tests, and timing measurements are presented in the following sections of this chapter.

### 7.1 CAN Control System Tests

To demonstrate the correct operation of the model CAN control system, it must be capable of efficiently controlling and maintaining the network of five independent nodes, while each independent load node controls its designated loads accurately.

Figure 7.1 shows a photograph of the completed model CAN control system. All CAN nodes are connected to the 20 metre CAN bus, while loads have been connected to their specified CAN nodes. The figure shows all five nodes, the 20 metre CAN bus, a headlight motor, wipers motor (and light bulb resembling the washer motor) and a window motor. The total power all network nodes consume without operating any of the loads, on standby, was measured to be 2.17W. This is due to power consumed by operating some components within the network while on standby, such as the relays discussed in 6.2.3.

Six female DB9 connectors are placed along the CAN bus, one at each end and four equally spaced along the bus. Five connectors are used to connect the CAN nodes onto the network, while the sixth is used to connect a CAN development board, which connects between the network and a computer on which all traffic on the bus can be displayed. This CAN development board was used to monitor the traffic within the CAN control system throughout the development stage.

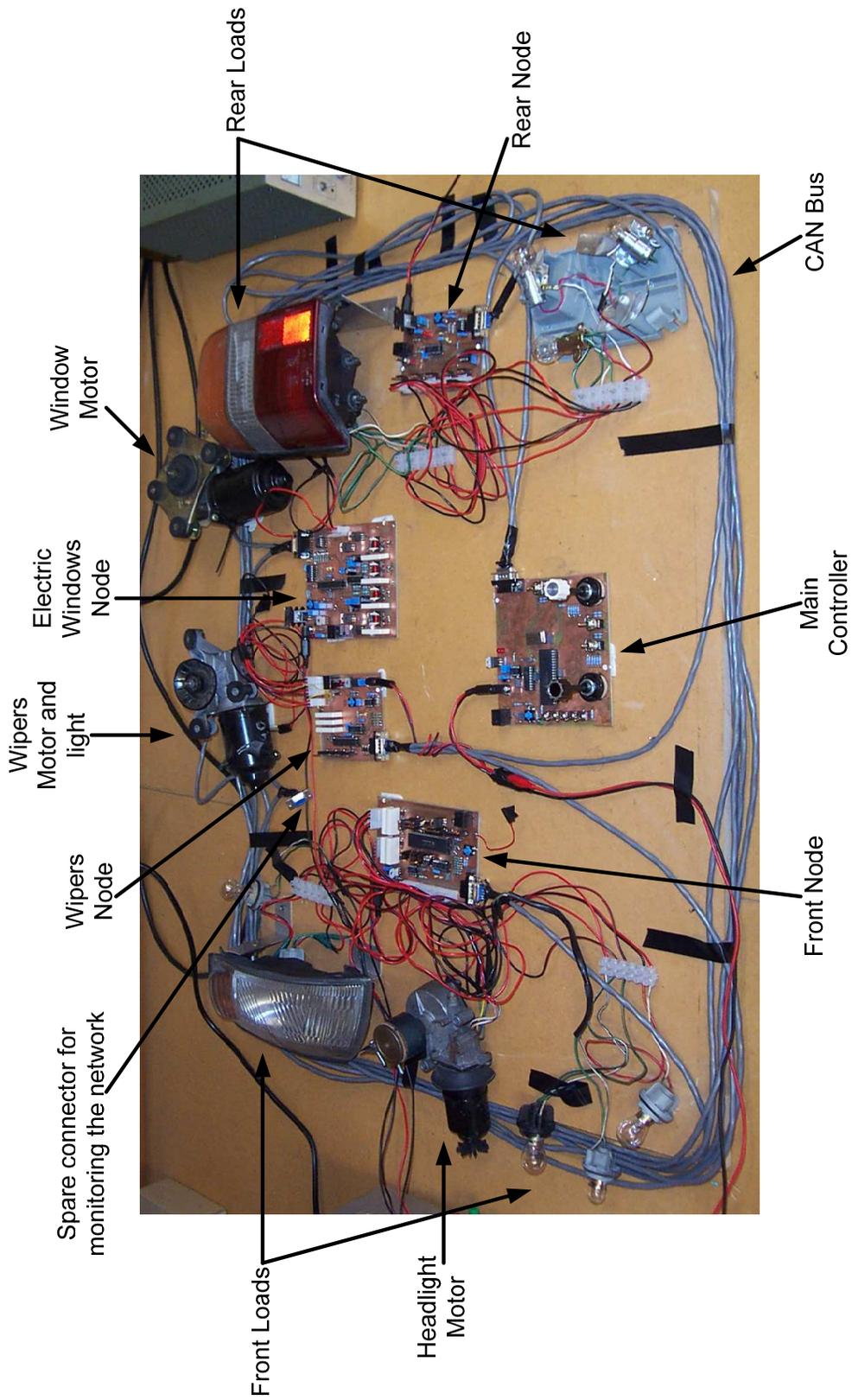


Figure 7.1: Prototype Photograph

Figure 7.1 shows the CAN front load node connected to the front loads (left) and to one end of the bus. The rear loads are connected to the rear load node which is connected to the other end of the bus. The main controller, wipers and electric windows nodes are all connected to three out of the four remaining connections available on the bus. The CAN development board is not connected onto the network, and its remaining spare DB9 connector can be seen in the figure. The layout conforms to the layout discussed earlier in Chapter 5, where the front (Nd4) and rear (Nd1) nodes are connected to the ends of the CAN bus.

Within the CAN control system there are no restrictions on where the nodes are connected along the bus, except that both the front and rear nodes have to be connected to either end of it. The reason behind this is that the two termination resistors discussed in Chapter 5 are onboard the front and rear nodes, and if the front and rear nodes were not connected to the ends of the bus, the signal would not be terminated properly causing significant communication problems.

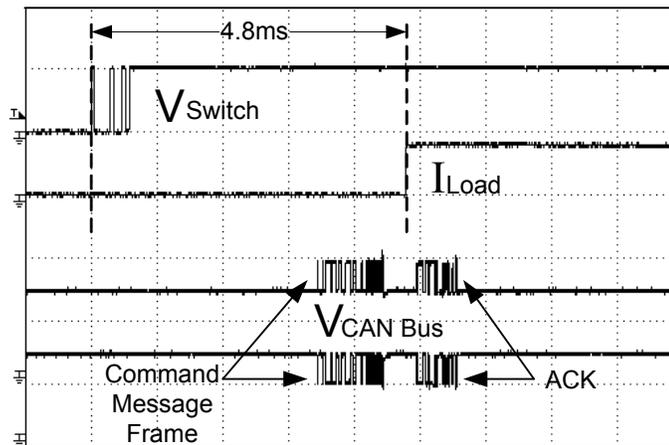
The network was powered by four independent bench power supplies resembling the four independent power supplies in the MR2, while the CAN bus is placed around the system boards to resemble the noisy environment inside the MR2 during the tests.

The remainder of this section examines the waveforms captured and timing measurements illustrating the correct operation of the CAN control system. The following sections show the CAN control system's time delays, electric windows current sensing operation, missing node detection and the synchronization problem between the front and rear load nodes.

### **7.1.1 Time Delays**

To efficiently control the 12Vdc auxiliary loads, the system has to respond relatively fast to user commands. Figure 7.2 shows the waveform captured when transmitting a command message generated from toggling a switch on the main controller to turn on the rear brake light on the rear node via the CAN bus.

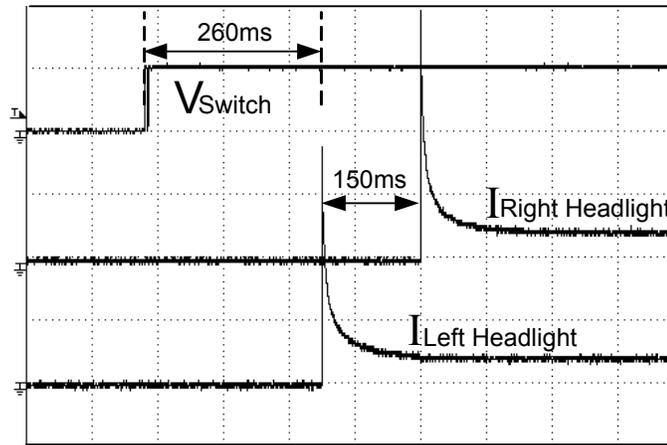
At 125kbps network speed, using 20 metre long shielded twisted pair cable, the time taken between the switch operating and the load turning on was found to range from 4.8ms to 10ms. The time delay is very insignificant when compared with a typical human reaction time, of say 0.7s at best [17]. It must be noted that all ICs in the network use 4MHz crystals to supply their clock signal. Also, the time taken by the CAN nodes to process and acknowledge a command message is variable, which is due to variable software processing time.



**Figure 7.2: Message transmission waveforms. Time scale: 1ms/div. (Top: Voltage across switch @ 5V/div; Middle: Current flow through light bulb @ 5A/div; Bottom two signals: Differential voltages across the CAN bus @ 2V/div)**

Figure 7.2 shows that a CAN command message frame is generated from the main controller upon toggling the brake switch. After the message is read and processed to turn on the brake light, an ACK message is sent back onto the CAN bus for the main controller to read and confirm proper processing of the received message.

As mentioned before in Chapter 3, time delays are to be inserted between high demand loads to ensure that current overshoot does not exceed the maximum power threshold of a single converter. To demonstrate this concept, a 150ms time delay has been inserted between the turn on of the headlights, shown in Figure 7.3.



**Figure 7.3: Headlights turn-on. Time scale: 100ms/div. (Top: Voltage across switch @ 5V/div; Middle: Current flow through right headlight @ 5A/div; Bottom: Current flow through left headlight @ 5A/div)**

For the headlights, it was found that the minimum time taken to switch them on was 100ms, while the maximum was found to be 260ms, which is not a problem in this case where it is a one-time non-critical function, unlike the brake lights. The extra delay in turning on the headlights, or any other load, is due to longer program processing time in the node's local MCU. An advantage to adding local MCUs to every CAN node is that the amount of data transferred on the bus is very low, thus minimizing the chance of error generation within the network.

Table 7.1 shows the minimum and maximum times found between toggling the designated switch on the main controller and turning on the corresponding load in the network. From the table it can be seen that the rear loads, except the indicators, controlled directly from the MCP25020 (CAN expander IC) take the least amount of time to operate, as discussed in Chapter 6, and is due to the elimination of extra processing time imposed on the rest of the loads by their local MCUs. Also, there is a significant difference between the time taken to operate the front and rear park lights, which is also due to the difference in processing time, but is not a problem since it is a one-time non-critical function with the difference not noticed by the human eye.

The table also shows that the headlights take less time to operate than the headlight motor. This verifies that the headlights would be turned on before being raised, as mentioned in Chapter 6.

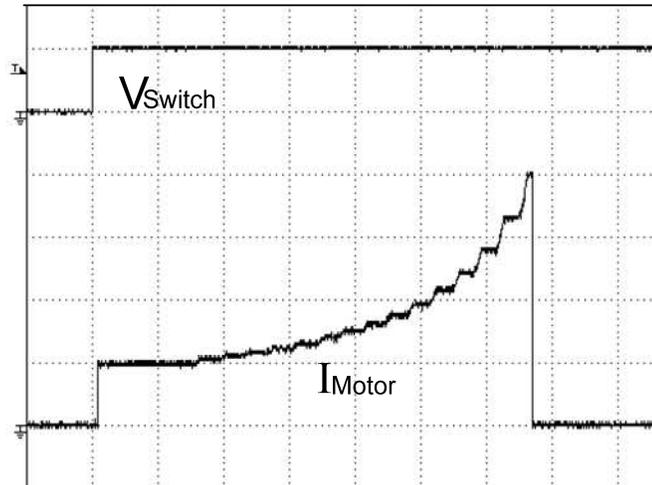
**Table 7.1: Time delays for switching loads**

<u>Load</u>	<u>Minimum Time (ms)</u>	<u>Maximum time (ms)</u>
Brake lights	4.8	10
Reverse lights	4.8	10
Rear park and number plate lights	4.8	10
Front Park lights	100	140
Indicators	80	150
Wipers motor and pump	10	80
Electric windows	100	150
Headlights	100	260
Headlight motors	120	280
High beam light	10	60

As discussed in Chapter 4, there are no priorities between the CAN nodes or messages within the model CAN control system. For that reason, the first command message issued is always the first to be processed in the network. In this model system, the bus load is minimal and the time taken to switch loads on and off is fast enough for this application, Table 7.1. But if the system is to be expanded and more options added to it, increasing time delays within the network, brake lights would need to be prioritized over all other loads since they are the most safety-crucial load within the model network.

### 7.1.2 Electric Windows Current Sensing

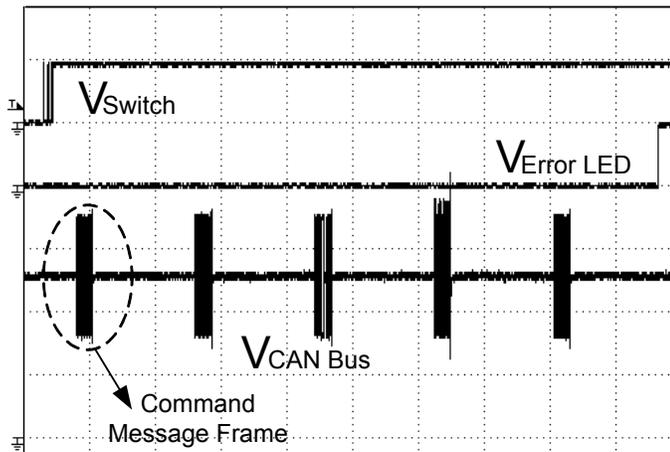
As discussed in Chapter 6, the electric windows node employs current sensors that monitor the current through the motors to ensure that the motors are stopped if an obstacle is encountered or if the windows have reached the frame. Since loading the motors' shaft manually was difficult, a high power potentiometer was used instead, to manually increase motor current and monitor the node's behaviour. Figure 7.4 shows the waveform captured as the potentiometer's resistance is dropped, increasing the current to reach around 4A. Once the current reaches 4A, the local MCU onboard the windows node stops the motor as required.



**Figure 7.4: Windows current waveform. Time scale: 5s/div. (Top: Voltage across switch @ 5V/div; Bottom: Current flow through electric motor @ 1A/div)**

### 7.1.3 Missing Node Detection

Chapter 6 considers the ability to detect a missing node in the network, by sending a message five times if no ACK was received, and then turning the Error LED on.



**Figure 7.5: Error LED. Time scale: 2ms/div. (Top: Voltage across switch @ 5V/div; Middle: Voltage across Error LED @ 5V/div; Bottom two signals: Differential voltages across the CAN bus @ 1V/div)**

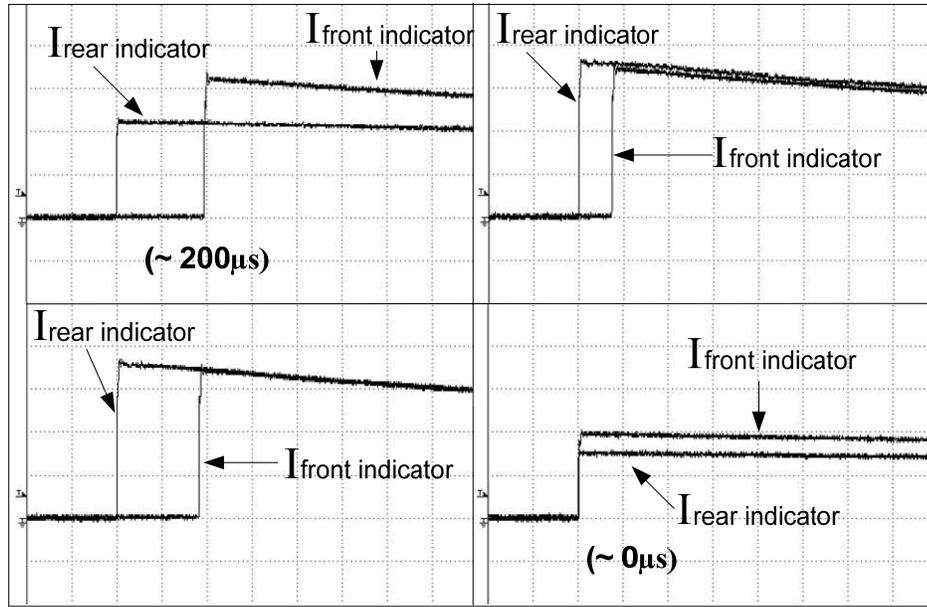
Figure 7.5 illustrates the functionality of the Error LED. The rear load node was removed from the network and the brake switch was toggled on the main controller board. It can be seen that the message was sent five times before the Error LED is turned on, indicating the absence of the CAN rear load node, shown in Figure 7.5.

#### **7.1.4 Synchronization Problem**

As discussed in Chapter 6, CAN is an asynchronous protocol, and due to the independence of node's clocks and variability in the time taken to process a command message, a mismatch between the front and rear indicators could accumulate to reach a visible time difference between the front and rear indicators.

Figure 7.6 shows four samples of pairs of waveforms generated when first switching the left indicators on. From the figure, the time mismatch between the front and rear indicators/hazard lights was found to range from 0s to 200 $\mu$ s at the most. It can also be seen that the rear indicator is always turned on prior to the front one. That is mainly caused by the software processing time difference between the front and rear local MCUs. That is, since the front node's MCU contains more software than the rear one, the front MCU takes longer to go through its software and process incoming command messages than the rear MCU, which only holds the software to control the rear indicators, detailed in Chapter 6.

Upon examining Figure 7.6, it can be seen that current amplitudes differ from one waveform to another. This has been found to be directly related to the bulb filament temperature, the cooler the filament the higher the inrush current [18, 19]. And since the waveforms were captured in different intervals, the filaments temperatures were not identical, hence the difference in current waveform amplitudes in Figure 7.6.



**Figure 7.6: Front and rear indicators timing mismatch. Time scale: 100 $\mu$ s/div.  
(Waveforms: current flow through front and rear indicator light bulbs @ 1A/div)**

## 7.2 Summary

The prototype of the CAN control system was constructed and tested successfully. Message transmission, time delays and time mismatch waveforms were captured to verify the correct operation of the system. The minimum time delay of the brake light message transmission via the CAN bus was found to be 4.8ms, with a maximum of 10ms. This is considered to be fast enough for the MR2 application, with the driver not noticing the delay. Other load delays were found to be higher than that of the brake lights due to longer processing times within the independent node's MCUs. This is not considered a problem since they are one-time non-critical functions, such as headlights.

Finally, the overall system was tested and all the 12Vdc loads that are used to simulate the selected auxiliary loads in the vehicle were switched on and off successfully. From these experimental results, it can be concluded that the CAN control system developed for controlling 12Vdc auxiliaries in the MR2 is expected to operate successfully.



## 8 FUTURE DEVELOPMENT & CONCLUSION

---

### 8.1 Future Development

A prototype CAN control system for the most recent departmental electric vehicle, a Toyota MR2 has been designed, constructed and tested. The CAN control system was constructed to control the supply of power to the 12Vdc auxiliary loads in the MR2. The prototype consists of one CAN main controller node and four CAN load nodes. All CAN nodes are connected to a twisted pair of wires, the CAN bus, on which messages are transmitted and received. The purpose of implementing the CAN control system is to demonstrate and assess the basic operation of CAN to be used in the MR2 application. The next stage of the system development is to construct new CAN load nodes to control more functions within the MR2, such as power steering, and add new systems to the CAN control system, such as an alarm system, X-by-wire systems and a LIN network as a sub-bus of CAN to control non-safety critical tasks around the vehicle.

As for the synchronization problem between the front and rear CAN load nodes, demonstrated in Chapter 7, a TT-CAN network could be built, as an extension of the already built model CAN control system, to synchronize the front and rear load nodes only, while the rest of the network remains unchanged. High intensity LEDs with speciality lens covers could be used to replace the existing external and internal light bulbs to further reduce the current demand of the loads. To be able to replace double filament bulbs with LEDs, drive the loads from the high-side rather than low-side as discussed in Chapter 5, High-Side Power MOSFET Switches (IPS511 from International Rectifier for example) can be used. High-Side Power MOSFET Switches are designed specifically for driving loads from the high-side in automotive applications. Those switches can be used to replace the MOSFETs controlling the loads in the CAN control system, so that all loads would be at 0Vdc potential until turned on, where 12Vdc would then be applied to them. Furthermore,

in the electric windows CAN node, the relays controlling the electric motors can be replaced with semiconductor based bridge circuits, example H-Bridge, to avoid reliability issues associated with relays and further improve the overall system performance.

Finally, if too many functions were added to the existing CAN control system and the time delays elevated to unacceptable levels, the data transfer rate would need to be raised to increase the overall network speed, while both the hardware and the software described in the previous chapters would need to be modified to accommodate for higher speeds. For example, if safety-critical functions, such as ABS brakes or Brake-by-wire, were added onto the CAN control system, the system should be modified to run at higher speeds and give these safety-critical functions priority over non-critical functions. This would increase the control system's reliability, which in turn ensures a safer ride for passengers onboard the vehicle.

## 8.2 Conclusion

This thesis documented the design, implementation, and test results of a CAN control system for the MR2 electric car application. It is proposed that the 312Vdc nominal battery voltage is converted to a 48Vdc intermediate voltage. The use of a 48Vdc distribution voltage is considered more efficient than the usual 12Vdc distribution system since smaller and lighter wires can be used to carry the same amount of power.

The type of loads determined the power rating, number and placement of the auxiliary dc-dc converters that form the Auxiliary Power System in the vehicle. The CAN control system controls the supply of power to the selected 12Vdc auxiliary loads.

The CAN topology is selected for communicating around the electric vehicle since the CAN protocol is optimised for systems that need to transmit and receive a relatively small amount of information reliably to any or all other nodes on the network. Since the protocol is message-based, all nodes on the bus receive every message, regardless of whether the node required the data or not. Fast robust message transmission with fault confinement is the big plus for CAN because faulty nodes will automatically drop off the bus, which does not allow any faulty node to bring down the network.

Various CAN implementation methods were discussed and the stand-alone CAN controller configuration was chosen. This is because with this configuration, multiple designs could be developed in the future and the software and hardware development is reusable. The CAN control system was designed and consists of five CAN nodes. Prototype of these boards are developed and constructed successfully.

The prototype of the CAN control system was constructed and tested successfully. To verify the correct operation of the CAN control system, the overall time delay of the message transmission via the CAN bus was found to be fast enough for the MR2 application, with the driver not noticing the delay. Finally, the overall system was tested and all the loads that are used to simulate the selected auxiliary loads in the MR2 were operated and controlled successfully. It is expected that a fully working CAN control system would fulfil the requirements of the electric vehicle and operate successfully in the MR2.



---

## 9 REFERENCES

---

- [1] F. Didik, "Electric Vehicle Companies Since 1834," [http://www.didik.com/ev\\_hist.htm](http://www.didik.com/ev_hist.htm), 2007.
- [2] W. Chen, "An Auxiliary Power Distribution Network for an Electric Vehicle," vol. Master of Engineering: University of Canterbury, 2003.
- [3] R. Duke, "Construction and performance of an electric Toyota MR2," presented at Australasian Universities Power Engineering Conference, Hobart, Australia, 2005.
- [4] R. B. Gmbh, "CAN Specification," vol. Version 2.0, 1991.
- [5] G. Leen and D. Heffernan, "Expanding Automotive Electronic Systems," *IEEE*, pp. 88-93, 2002.
- [6] D. Marsh, "Network Protocols Compete for highway supremacy," in *EDN Europe*, vol. June 2003, 2003, pp. 26-38.
- [7] D. Marsh, "LIN simplifies and standardises in-vehicle networks," in *EDN*, vol. October 2005, 2005.
- [8] S. Wilkinson, "Coming Soon: Electronic Brakes," <http://www.popsci.com/popsci/automotivetech/cd9a7178706d8010vgnvcm1000004eecbccdrerd.html>, 2006.
- [9] K. Pazul, "Controller Area Network (CAN) Basics," Microchip Technology Inc., AN713, May 1999.
- [10] P. Furmanski and Z. Stolarski, "Controller Area Network Implementation in Microwave Systems," in *Microwaves, Radar and Wireless Communications, 2002. MIKON2002. 14th international conference*, vol. 3, 2002, pp. 869-873.
- [11] Wikipedia, "OSI Model," [http://en.wikipedia.org/wiki/OSI\\_model](http://en.wikipedia.org/wiki/OSI_model), 2007.
- [12] B. P. Upender and A. G. Dean, "Variabilty of CAN Network Performance," presented at 3rd International CAN Conference, Paris, France, 1996.
- [13] Freescale Semiconductor, "High-Speed CAN," <http://www.freescale.com/webapp/sps/site/application.jsp?nodeId=012795pnQYC5xWDFBb>, 2005.
- [14] Intel Cooperation, "Introduction to In-Vehicle," <http://www.intel.com/design/auto/autolxbk.htm>, 2005.
- [15] Philips Semiconductors, "Application Note AN96116," 1996.

- [16] Toyota Motors Corporation, "Toyota MR2 Electrical Wiring Diagram," Dec., 1989.
- [17] M. Green, "Driver Reaction Time,"  
<http://www.visualexpert.com/Resources/reactiontime.html>, 2005.
- [18] Harison Toshiba Lighting, Inc. "General Lamp Information,"  
<http://www.harison-toshiba.com/lampinfo.htm>, 1999.
- [19] Gilway Technical Lamp, "Tungsten Filament Lamps,"  
<http://www.intl-lighttech.com/applications/appl-tungsten.pdf>.

## APPENDIX A

## MR2's Auxiliary Current Consumption Table

Current Usage for a Toyota MR2						
Load	Description	Current Measurement (Amps)		Time To Reach Steady State	Comments	References
		Initial Peak	Steady State			
Front Lights	Cap Rise	22A	5A, 0A	250ms, 540ms	No Head Lights Operating	
	Cap Down	21.2A	3.2A, 0A	250ms, 530ms		
	Low Beam Headlights	27.4A	8.2A	500ms	No Cap Operating	
	High Beam Headlights	27.4A	9.8A	500ms		
	Emergently Headlights On	27.4A	10.6A	600ms	Cap Rise + High Beam Headlights	
	Front Fog Lights	<i>Not working</i>				
	Hazard Lights	27.4A	8A, 0A	250ms, 400ms	6 Indicator Lights Operating	
	Dim Lights	26.4A	5A	75ms	2 Yellow Front and 4 Red Back Lights	
Back Lights	Break Lights	27.4A	7.4A	250ms	4 Red Break Lights Operating	
	Back Up Lights	<i>Can not make it work</i>				
	Right Handed Indicators	18.6A	4A, 0A	200ms, 250ms	3 Indicator Lights Operating	
	Left Handed Indicators	<i>Exactly same results obtained as above</i>				
Internal (Dome) Lights	Driver Side Reading Light	4.4A	0.64A	75ms	1 Bright yellow Light Operating Only	
	Both Reading Lights	5.32A	1.2A	75ms	2 Reading Lights Operating	
	Door Lights and Sound	4.24A	0.64A	75ms	Sound Is Based On The Key Position	
	Door and Reading Lights	8.08A	1.76A	100ms	2 Reading Lights; 2 Red Door Lights	
	Back Door Light	2.24A	0.28A	75ms	1 Bright Yellow Light Operating	
	Shift Car Key To The ACC Position	3.52A	3.52A, 2.6A, 1.84A	280ms, 2s, 6.36s	Key Shift To The ACC Position Is Required Before Operating The Heater	
Heater Fan	On Low	5.4A	3.8A	1.5s	All Been Operated From Off; No Change On The Current Value When Alternate The Directions Of Air-Blow	
	On Medium	15.6A	7.4A	1.1A		
	On High	25.6A	11.8A	750ms		
	Low to Medium to High	5.4A, 10.2A, 18A	3.8A, 7.6A, 11.2A	N/A	To Investigate The Sparks Formed; Measurements Were Done Manually	

Figure A: Current Consumption Table 1 of 2

Load	Description	Current Measurement (Amps)		Time To Reach Steady State	Comments	References
		Initial Peak	Steady State			
Power Steering	Turn On	49.2A	12A	2s	While No Movement On the Wheels	
	Turn To Right Hand Ended	69.6A	56.4A	N/A	Hard Lock On the Steering, These Were Done Manually, So No S.S. Available.	
	Turn To Left Hand Ended	Same As Results Obtained Above		N/A	Turn From Right Hand Lock To The Left Hand Lock	
Power Window	Turn From Right Hand End To The Left Hand End	66A	56.8A	N/A	Use Driver Side Automatic Window Winder For Open And Close Window	
	Open Driver Side Window	16.8A	2A, 15.6A	180ms, 2.8s	These Were Done Manually, Therefore No S.S. Available At The End.	
	Close Driver Side Window	17.4A	4.8A, 16.4A	3.65s	All Been Operated From Off; No S.S. Available But Has Average Current of 0.5A For Low & 1A For High Speed	
	Open Both Window	23.6A	4.6A	200ms	For Sparks Investigation Purpose. No Sparks Presented On the Curve	
	Close Both Window	23.6A	9.8A	220ms	No Significant Change On Current Readings While Alternate Volumes	
Window Wiper	Intermit Speed	2.22A	N/A	N/A	While The Radio Was On, No Change On The Current Readings When Insert A Tape	
	Low Speed	2.04A	N/A	N/A	Automatically Turned On While At ACC Stage, Please Refer To ACC Measurement	
	High Speed	2.16A	N/A	N/A	Lock Doors	
	From Low To High Speed	1.56A	N/A	N/A	Unlock Doors	
	From High To Low Speed	1.62A	N/A	N/A	Horn Operation	
Car Audio	Radio On With Aerial Up	5.68A	2A, 1A	1s, 5s	PWM Waveform Obtained, Average Current of 10A	
	Radio Off With Aerial Off	5.12A	1A, 0A	2s, 4.5s	Can't Make Mirror-Exacting Work	
	Tape Player	While The Radio Was On, No Change On The Current Readings When Insert A Tape				
Central Locking	CD Player	Automatically Turned On While At ACC Stage, Please Refer To ACC Measurement				
	Lock Doors	9.12A	8.24A, 0A	100ms, 1ms		
	Unlock Doors	8.88A	7.6A, 0A	150ms, 1ms		
Other Electronic	Horn Operation	10A	N/A	N/A		
	Wing Mirrors Shifting	0.436A	0.1A	100ms		
	Rear Defogger On	3.92A	3.8A	250ms		
	Air Conditioning	Air Conditioning does not work				
Combination Meters	Combination Meters	4.2A	3A	100ms	Only An Estimate Result	

Figure B: Current Consumption Table 2 of 2

## **APPENDIX B**

### **Publication**

---

Publication is an effective approach for recognition of the research among other engineers as well as industries. Throughout the course of the research, a paper was submitted to the 12<sup>th</sup> Electronics New Zealand Conference 2005 (ENZCon2005). The electronic copy of the paper is available in the thesis CD-ROM attached in Appendix F



---

# APPENDIX C

## Schematic Diagrams

---

The schematic diagrams for all CAN control system nodes are available in the thesis CD-ROM attached in Appendix F. This appendix contains the information of the files on the CD-ROM.

Protel DXP was used to draw the schematic diagrams for the CAN control system nodes. It must be noted that the schematic diagrams on the CD-ROM are available in their original format “.SchDoc” and can only be opened using Protel DXP or greater.

<b><u>Filename</u></b>	<b><u>Description</u></b>
CAN Main	Contains the CAN Main Controller Node's schematic diagram.
CAN Rear	Contains the CAN Rear Load Node's schematic diagram.
CAN Wipers	Contains the CAN Wipers Load Node's schematic diagram.
CAN Electric Windows	Contains the CAN Electric Windows Load Node's schematic diagram.
CAN Front	Contains the CAN Front Load Node's schematic diagram.



# APPENDIX D

## Software

---

This appendix contains information of the files used in programming the CAN control system's ICs explained in Chapter 6. The files are written in C language using MPLAB IDE v6.6, which is also used to program the ICs, and are available in the thesis CD-ROM attached in Appendix F.

The CAN expander IC, MCP25020, is programmed using a different software than the MPLAB IDE, called MCP250xx Programmer v1.1. The software uses a graphical interface to program the IC. For this reason, images were captured to show the configuration of each IC used in each load node and are also included in thesis CD-ROM.

In general, the nodes are programmed to only read and accept messages with the following identifiers (IDs) (represented in Hexadecimal form);

- Main Controller Node: messages with IDs = 0x511, 0x411, 0x311, 0x211
- Front Node: messages with IDs = 0x5F0 and 0x5B0
- Wipers Node: messages with IDs = 0x6F0
- Electric Windows Node: messages with ID = 0x7F0
- Rear Node: messages with ID = 0x7B0 and 0x5B0

On the other hand, the nodes are programmed to transmit messages with the following IDs;

- Main Controller Node: messages with IDs = 0x5F0, 0x5B0, 0x6F0, 0x7F0 and 0x7B0.
- Front Node: messages with IDs = 0x511
- Wipers Node: messages with IDs = 0x211
- Electric Windows Node: messages with ID = 0x311
- Rear Node: messages with ID = 0x411

By examining the above IDs, it can be seen that the main controller is capable of sending and receiving messages to and from all load nodes, while the load nodes are only capable of receiving messages from the main controller.

The tables below show the information of the files on the thesis CD-ROM;

The **CAN Main** folder;

<u>Category</u>	<u>Filename</u>	<u>Description</u>
Project	MainProj.mcw	Project file of the program for the CAN Main Controller. When this is opened using MPLAB IDE v6.6 or greater, all the necessary files for programming the PIC16F877 are automatically loaded.
Source (C Language)	Main.c	Sends CAN messages to specific CAN load nodes, via the CAN Controller IC (using SPI), in response to toggling switches onboard the CAN Main Controller board.
	mcp.c	Provides initialization and generic functions used to communicate with the CAN Controller IC, MCP2510, via SPI.
	delay.c	A delay function file, used within Main.c
Header	mcp.h	Defines the functions and variables used by Mcp.c file.
	delay.h	Defines the clock rate for the delay source file.
	pic.h	Defines the necessary bits & registers for the PIC MCU

The **CAN Rear** folder;

<u>Category</u>	<u>Filename</u>	<u>Description</u>
Project	Rear.mcw	Project file of the program for the CAN Rear Load node. When this is opened using MPLAB IDE v6.6 or greater, all the necessary files for programming the PIC12F629 are automatically loaded.
Source (C Language)	Rear.c	Controls the rear loads of the example CAN control system.
	delay.c	A delay function file, used within Rear.c
Header	pic.h	Defines the necessary bits & registers for the PIC MCU
	delay.h	Defines the clock rate for the delay source file.
Other (See MCP2502x datasheet for more info)	General.bmp	Captured image of the main page of the MCP250xx program, where CAN and GPIO registers are included.
	GPIO_Reg.bmp	Captured image of the general purpose register values for the CAN Expander IC onboard the Rear Load node.
	CAN_Reg.bmp	Captured image of the CAN register values for the CAN Expander IC onboard the Rear Load node.

The **CAN Wipers** folder;

<u>Category</u>	<u>Filename</u>	<u>Description</u>
Project	Wipers.mcw	Project file of the program for the CAN Wipers Load node. When this is opened using MPLAB IDE v6.6 or greater, all the necessary files for programming the PIC16F876 are automatically loaded.
Source (C Language)	Wipe.c	Controls the wipers of the example CAN control system.
	delay.c	A delay function file, used within wipe.c
Header	pic.h	Defines the necessary bits & registers for the PIC MCU
	delay.h	Defines the clock rate for the delay source file.
Other (See MCP2502x datasheet for more info)	CAN_Reg.bmp	Captured image of the CAN register values for the CAN Expander IC onboard the Wipers Load node.
	GPIO_Reg.bmp	Captured image of the general purpose register values for the CAN Expander IC onboard the Wipers Load node.

The **CAN Electric Windows** folder;

<u>Category</u>	<u>Filename</u>	<u>Description</u>
Project	EWindows.mcw	Project file of the program for the CAN Electric Windows Load node. When this is opened using MPLAB IDE v6.6 or greater, all the necessary files for programming the PIC16F876 are automatically loaded.
Source (C Language)	Wind.c	Controls the electric windows of the example CAN control system.
	delay.c	A delay function file, used within wind.c
Header	pic.h	Defines the necessary bits & registers for the PIC MCU
	delay.h	Defines the clock rate for the delay source file.
Other (See MCP2502x datasheet for more info)	CAN_Reg.bmp	Captured image of the CAN register values for the CAN Expander IC onboard the Windows Load node.
	GPIO_Reg.bmp	Captured image of the general purpose register values for the CAN Expander IC onboard the Windows Load node.

The **CAN Front** folder;

<b><u>Category</u></b>	<b><u>Filename</u></b>	<b><u>Description</u></b>
Project	Front.mcw	Project file of the program for the CAN Front Load node. When this is opened using MPLAB IDE v6.6 or greater, all the necessary files for programming the PIC16F877 are automatically loaded.
Source (C Language)	Front.c	Controls the front loads of the example CAN control system.
	delay.c	A delay function file, used within front.c
Header	pic.h	Defines the necessary bits & registers for the PIC MCU
	delay.h	Defines the clock rate for the delay source file.
Other (See MCP2502x datasheet for more info)	CAN_Reg.bmp	Captured image of the CAN register values for the CAN Expander IC onboard the Front Load node.
	GPIO_Reg.bmp	Captured image of the general purpose register values for the CAN Expander IC onboard the Front Load node.

# APPENDIX F

## Thesis on CD-ROM

---

The attached CD-ROM contains most of the files related to the research, including an electronic copy of this thesis. The structure of the CD-ROM contents is shown below:

