eBLOCKS®

LEARN·DESIGN·BUILD

CAN Bus Communications

EB744-80-10

MATRIX

www.matrixmultimedia.com

EB744
CAN Solution
Course Notes

Contents

Introduction

These Teacher's notes are designed to introduce you to the concepts required to understand CAN networks and also to provide practical exercises with which to develop your skills as well as those of your students.

The course is structured into a number of sections that first take you through the basics of CAN and then into intermediate topics, such as messages and sending data. The course also deals with some more advanced topics, including the use of masks and filters. Examples and suggested work is provided as a basis for developing demonstrations and practical activities for your students.

These notes provide a framework for teaching CAN to students. How you use them for teaching is up to you. If you are teaching automotive students who do not need to know how to program you can simply make use of the downloadable example programs.

This course is carried out using Flowcode, a graphical programming language, which is shipped with the CAN solution and supplied on CD ROM. The Flowcode CAN component is designed to allow students to learn about CAN without getting bogged down with the problems of programming in C or a lower level language.

When teaching automotive students about CAN we do not envisage a great deal of programming will take place in Flowcode. However we suggest that the supervisor should have some Flowcode experience for debugging purposes. This can be quickly and easily acquired.

More advanced students will want to use Flowcode extensively. There are a number of tutorial files and resources on the Flowcode CD ROM that students can go through to help them understand how Flowcode works. Students will find that they can make rapid progress using Flowcode's graphical interface.

This course is designed for use with two levels of student:

1. Firstly for use with automotive technicians at Level 3 to gain an appreciation of CAN technology and the equipment used in fault finding CAN systems, and how that fault finding takes place. These technicians are expected to download and review programs made in flow charts, but are not expected to carry out any programming tasks.

2. Secondly for more advanced students at Level 4 to gain an understanding of CAN technology and to allow them to construct networks which communicate in CAN and higher level protocols. These students are expected to develop their own CAN networks using flowcharts with CAN macros provided. The extensive use of flow charts will allow students to quickly and easily understand CAN protocols and communication, avoiding the need to become involved with the processes of lower level CAN bus software construction.

1 Demonstrations, Worked Examples and Exercises

There are three kinds of exercises found in the Teacher's notes; *demonstrations*, *worked examples* and *examples*. We will briefly describe what we mean by each of these.

All the programs supplied on the CAN Solution CD require Flowcode V4.1 or higher to be installed on the host PC. A compatible version of Flowcode is supplied with this solution. Please refer to the Flowcode CD for installation instructions.

1.1 Demonstrations

Demonstrations are provided that can be used with technicians to see a CAN system in action.
All programs will be provided so that they can be programmed into the nodes. No programming is required, but the Flowcode flowcharts are available to students to show how they work. The demonstrations are best used in conjunction with CANKing to allow students to see the messaging in action, and to note the effect different programs have on the network traffic.

Demonstrations can be used to teach the fundamentals of CAN to students who are not required to understand and check CAN systems but will not need programming skills.

1.2 Worked examples

A worked example is provided for the first basic example to allow students to be eased into both CAN and Flowcode. The emphasis here is on getting students 'up and running' with a simple CAN system in Flowcode that can then be used as a base of experience for later examples.

1.3 Exercises

Further exercises are provided, along with a set of accompanying notes. The notes give information on setting up various aspects of the solution and can form the basis for handouts.

1.4 Further work

Questions to ponder and suggestions for further work are given with each exercise. This further work can be used as the basis of differentiated student activities, thus meeting awarding body requirements in situations where the CAN solution is being used in conjunction with a formally assessed course.

2 Basic CAN Networking

2.1 Overview

This section is designed to get you up and running with a CAN Network as fast as possible. You will be introduced to messaging and how to send receive a simple signal that can be acted upon. The sample applications will introduce you to several basic CAN features, and will serve as a starting point for further study.

2.2 What is CAN?

CAN – Controller Area Network is a serial network protocol. By which we mean it is a pre-defined way to communicate between different parts of a system. Each part needs to speak the same language, and use a common set of signals and message structures in order to be able to understand the messages and in turn to be understood. CAN is one such system.

Other systems, such as RS-232 are often point to point systems where one device will talk directly to another. A limitation of this system is that you may need to run several different connections to speak to different parts of the system, or one part may need to talk to another but may only be able to do so via an intermediary.

CAN offers a simple solution to this problem. It sends the message to all parts of the system, and lets each part (or *node*) decide for itself if the message is for it or not. Built in *error checking* and responses to messages help prevent lost messages, or jammed systems where the system *hangs* whilst waiting for a response to come in. Also messages can be responded to by multiple devices or even none at all, making it easier to construct a system that only reacts wherever and whenever it needs to.

CAN has various inbuilt systems for *error detection*, and ways to prevent all the nodes trying to talk at the same time. But you never see this, it's handled by the CAN chip behind the scenes. All you need to do is decide what messages to send and receive.

Another benefit of the CAN system is that if you wish to add another part to the system it can often be as simple as programming it to respond to the appropriate messages and wiring it into the network. You don't need to connect it up in any specific place or sequence so you can slot the new node in wherever you want, or wherever the physical system design requires it to be.

CAN was originally designed by Bosch for the Automotive industry, evolving from a need to communicate between the various ECU's (Electronic Control Units) on luxury cars. Since then CAN has grown to become a popular network system, particularly in embedded systems. CAN is used on vehicles such as cars, boats, planes, trucks, and in many other areas of industry. CAN's high speed and robust nature make it particularly suited for industrial or high speed applications.

2.3 Nodes and Networks

CAN systems comprise of two or more nodes connected in a *network* as shown in Fig. 2.1.

The *network* is the data highway to which the nodes are connected. Unlike many other systems where the connections run from device to device the CAN network is free standing. Each node feeds off the network, but does not block it, or prevent the other nodes from receiving the message.

The *nodes* can send messages onto the network, and can listen to the network to see if there are any messages on it. The node can then check the message to see if it should respond to the message, or if it should ignore it.

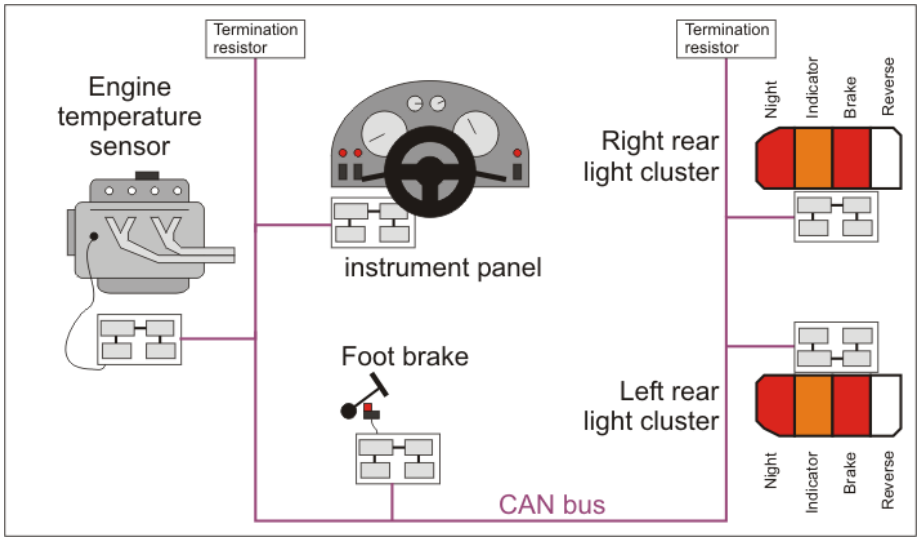

Figure 2.1 *Typical automotive CAN bus arrangement*

Nodes are independent of each other, and can be as simple or as complex as the system designer wishes them to be. A node could be a single light, or a whole dashboard. By virtue of the nodes being independent they can be added, removed or modified without needing to change any other part of the CAN network.

For instance two models of a car may have different dashboards, one a deluxe model with extra features not found on the basic model. The dashboards are both sent the exact same messages by the CAN network; it is what messages they accept, and what they do with them that makes them different. The CAN network does not care if a signal becomes a single light, or a strip of LED's. It does not even care if the 'Fancy RPM display' message is ignored when the basic dashboard is fitted. The CAN network simple puts the messages onto the network, and leaves the nodes to decide whether to use them or not. As there is no change to the messages sent, the CAN network does not need changing to accept the different dashboards so either dashboard can be slotted into the network.

2.4 The physical layer

The CAN specification does not specify the physical signal transportation layer, only the message format. By doing this CAN allows system designers to implement a *physical layer* appropriate to the system rather than having to adapt the system to match a preset physical layer.

This is a very important issue as it means that the way CAN is implemented in the physical layer can and often will differ from system to system. The physical layer for a plant wide heavy industry CAN system is likely to differ in many ways from one built into a luxury car. A CAN system such as the Matrix Multimedia CAN board is essentially our implementation of CAN using our own Flowcode component and our own CAN board to drive the physical layer. Some parts of the overall system, such as the format of the message sent are an integral part of CAN as defined by the CAN specification. Others such as the RX and TX buffers (discussed later) are part of our implementation of a physical layer for CAN.

2.5 Messages

Messages are the beating heart of the CAN system. Without them it's just a load of redundant wires and circuit boards. Each message consists of an identifier (the Message ID that will become important later on) and a stream of data. Actually, there's more – acknowledgment bits, checksums, transmission

details etc. but these are dealt with automatically by the CAN component. All you need to worry about is the ID and the data.

The Message ID's are used to help differentiate messages. A node could accept certain messages, and skip others depending on their ID values. This allows complex interrelated systems to be designed easily where multiple nodes can respond to the same message as easily as a single node can pick out a message that only it will respond to.

Messages can contain data or be completely empty. For many simple signals such as a brake light the act of sending a message may well be enough – a signal is sent and is reacted to. For others e.g. RPM, or temperature readings, data of some sort is required and can be passed along with the message. If data is sent it does not even need to be looked at. An aircraft RPM sensor reading could be used by one node to display the actual RPM, but on another node to simply activate an 'engine running' warning light without even looking at the data.

2.6 Higher Level Protocols

CAN is a *message system*. It is not responsible for the contents of the message. CAN does not care if data is expected, or if the incorrect amount of data is sent. It is not responsible for ensuring that the message is being sent to the appropriate node in the system, only that it is correctly sent to a node. CAN only cares that it is a correctly formed CAN message.

However, we do care about the data. We care about *which node it gets sent to*. We care about these things enough to create *higher level protocols* to deal with these kinds of issues. These protocols sit on top of CAN and help control the flow of messages and data on the network. Higher Level Protocols, or HLPs, are used in CAN systems to perform functions such as system startup procedures, error checking, connection and status monitoring and other administrative tasks.

Using an HLP may involve a CAN node having to send messages asking to be able to speak to another node, and requiring messages to be sent back agreeing to the communication before the real communication can begin. Such systems may seem like a major overhead when you are learning to send CAN messages, but once you understand CAN messages then your mind will automatically start to look for ways to error check and monitor the system. HLPs are the result of this natural progression.

HLPs are the glue that helps keep the system ticking over nicely. For this reason large scale systems will most likely use a HLP. One problem with HLPs is the amount of them – over forty already. Which HLP you use would most likely depend on the company you work for, or the products you deal with.

HLPs are beyond the scope of this course. The diversity makes it difficult to deal with them in detail. And the size and complexity of code needed for a HLP is too much for most basic microcontroller systems to handle. However, if you wish to look into HLPs and how they are used you can find documentation on various CAN HLPs, such as CANOpen, on the Kvaser site www.kvaser.com .

3 CAN training solution

This course is based around a CAN training solution that is set up as a four node network. This provides a *digital input node* (switches), a *digital output node* (lights) and an analogue input node (sensors) together with a monitoring/control node (the dashboard). These four nodes should help you gain an understanding of CAN network tasks within, but not limited to, an automotive context. Not all nodes are required for every task, and for some tasks you may need to reconfigure some of the nodes (e.g. two 'Lights' nodes). However for general training, and to teach the principles the four node network is ideal. A fifth connection point is available, which is used in conjunction with the Kvaser CAN Analyzer to monitor network traffic.

Figure 3.1 *The CAN training solution*

The CAN training solution consists of two backplane panels with two nodes on each panel. The E-blocks boards on each panel are all connected to the same power system which has been set up so that a single PSU attached to one of the programmer boards on a backplane will power all the boards on that backplane including the other programmer board. Only one USB cable is needed as each node requires programming separately. The USB cable can be connected to any of the nodes for programming. However, a second USB port on the PC is needed for the CAN Analyzer.

Important note: Information presented here is correct at the time this document was produced. Please check the Matrix Multimedia web site http://www.matrixmultimedia.com for the latest E-blocks documentation.