iCC 1995

2nd international CAN Conference

in London (United Kingdom)

Sponsored by

Motorola Semiconductor
National Semiconductor
Philips Semiconductors

Organized by

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

Implementing Diagnostic Systems for CAN Networks

Andy Renshaw, GenRad Ltd.

Abstract

The ever-increasing complexity of today's electronic products creates a continuing demand for more sophisticated diagnostic systems, whilst the need for a simple user interface remains unchanged.
A popular approach to system diagnostics involves the use of diagnostic fault trees. These can provide the product designer with a simple mechanism for the testing of operational parameters and a clear graphical display of the paths within the diagnostic sequence.
This paper discusses the mechanisms that have been implemented to allow the use of fault-tree diagnostics on CAN bus systems.
Also discussed are the additional capabilities that the diagnostic system can provide on the CAN network, and other ways of implementing diagnostic systems on existing software platforms.

Introduction

Often it may appear at the start of a product's development cycle, that the product will always be easily repaired with a little knowledge of the product, the relevant circuit / wiring diagrams and some standard electronic tools (e.g. an oscilloscope and a CAN bus analyzer). However once a CAN bus and one or more microprocessors have been integrated into a product, a new world of opportunity is opened up in terms of its capabilities and future developments. Clearly as time goes on, the future releases or upgrades of the product are likely become more complex as new features are designed into it, and consequently repairs to the product become more involved and more training is required for the repair engineer. Also, repair manuals need producing and maintaining.
A versatile electronic diagnostic system can empower engineers, who do not have an in-depth understanding of a complex product's operation, to diagnose problems quickly and make repairs, without the need for specialist tools and bulky repair manuals. It performs the necessary analysis in the background and hides the complexity of CAN bus information from the engineer, allowing him to focus on the higher levels of product operation.

What form does the diagnostic system take ?

The electronic diagnostic system is an expert system. It is computer based and capable of loading and running diagnostic application programs. This allows the product designer to consider all possible causes of faults within the product at design time, and then to produce a comprehensive diagnostic application which will isolate each fault down to the lowest replaceable unit within the product.
The capability of loading diagnostic applications allows updates to be made in the field to track new product versions. The diagnostic system need not be restricted to providing fault-finding functions, there are many more facilities that it can provide. These are described later.

How is the diagnostic application designed ?

A popular approach to the design of the diagnostic application program is the use of diagnostic fault trees. The logical approach to finding an unknown fault in a system is to carry out a series of tests. The selection of each new test based on the result of the previous test, until sufficient information is known to isolate the fault completely. Fault-trees are a means of taking such an approach and implementing it in diagrammatic form.

An example of a diagnostic fault tree is shown in Figure 1.

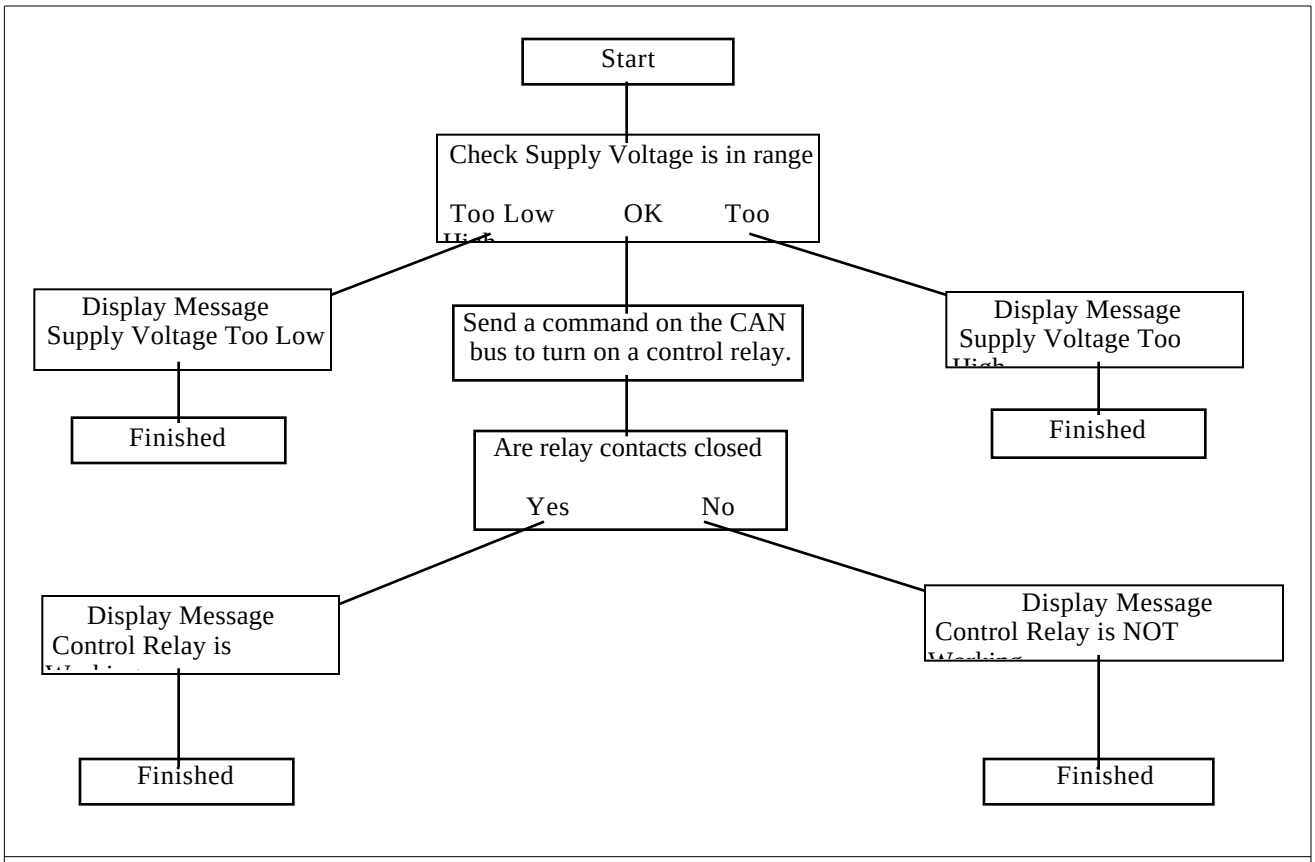

Figure 1 An example of a diagnostic fault tree.

The fault tree can contain the following :-

- Automated measurements / extraction of information from the product, e.g. accessing a parameter on the CAN bus.
- Control operations where the diagnostic system drives the product into a different state to allow further diagnosis. For example commands on the CAN bus can be used to produce stimuli for analogue measurements.
- Instructions to the engineer, telling him to reconfigure the product or the wiring setup in some way.
- Input of information by the engineer via alpha-numeric entry or selection from menus.
- Output of information to the engineer describing the nature of the fault and what corrective action to take.

Software tools are available which allow diagnostic fault trees to be created and edited on-screen. Large diagnostic applications can be designed using several fault trees with links between them.

What is involved in converting fault trees into diagnostic application programs ?

Each of the nodes on the fault tree represents an operation within the diagnostic application program, which can be converted directly into code in the application program.

Within a diagnostic authoring environment, such as the GenRad's GRADE, the diagnostic fault tree can be parsed (to check the validity of the tree e.g. syntax checks) and automatically converted into an application program. This provides the following advantages :-

- Traditional programming skills are not required to produce the diagnostic application program.
- There is a significant time saving in not having to code the application program from the fault tree.
- There is a single source for the diagnostic application, eliminating the dangers of multiple source maintenance.
- The application is self documenting as the fault trees can be printed as design records.

How does the application program communicate with the CAN bus ?

Within the diagnostic system various processes run to manage parts of the diagnostic systems operation, for example :-
- Screen output.
- Keyboard / touch-screen input.
- Engine for running the application program.
- Measurement Subsystem for analogue measurements.
- Serial Communication Subsystem for CAN bus communications or other serial communications.

Figure 2 shows a data flow diagram which gives an indication of the data flow between the diagnostic application program and the various components of the diagnostic system.

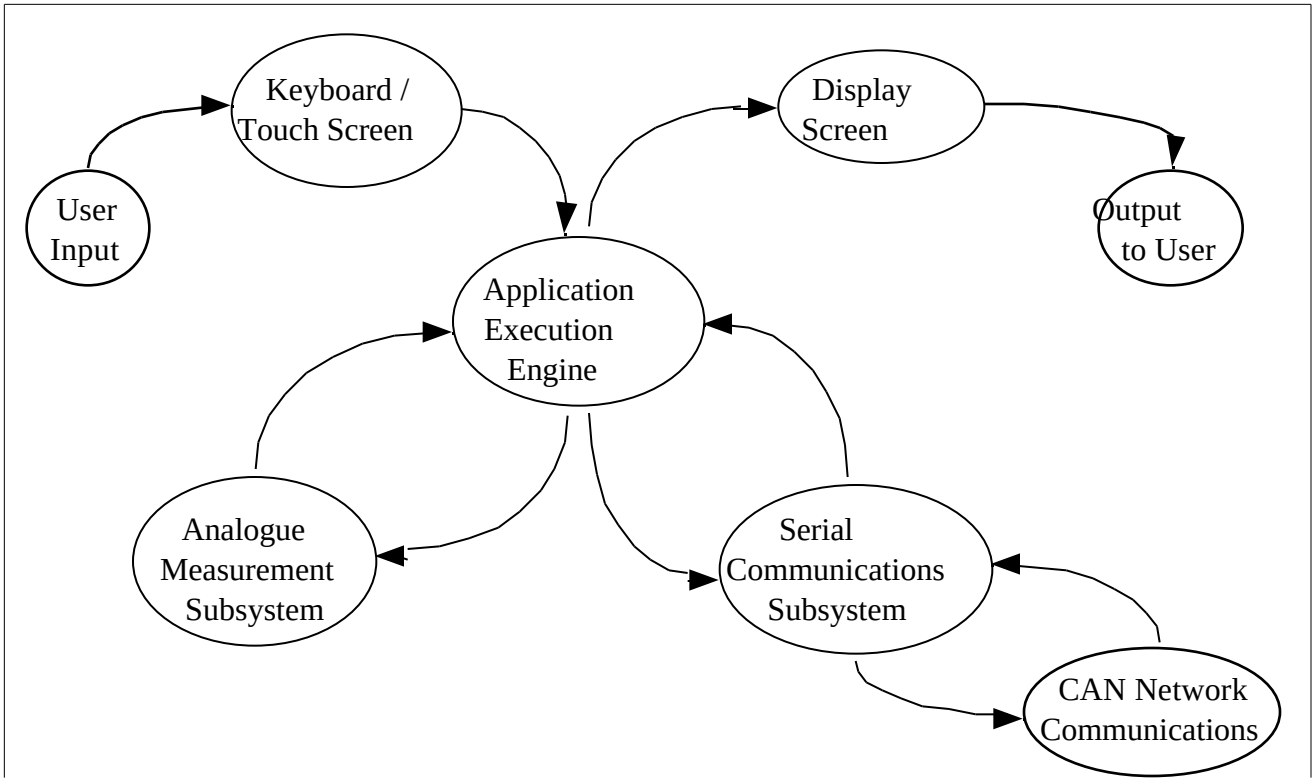

Figure 2 Data Flow in the Diagnostic Environment

Why use a separate serial communications subsystem ?

It is important that applications be insulated from the lower level aspects of protocol management for a variety of reasons :-

- The application program, which is usually written in a high level language, is not capable of executing quickly enough to handle the low level aspects of serial protocols. This is especially true of CAN given its maximum bus frequency of 1 Megabits / second.
- Application programs normally only need the result data and are not interested in the mechanics of actually obtaining that data.
- It is necessary to separate out the tools that view and manipulate data from the transport mechanisms that obtain the data. If this is not done specific tools arise for each protocol implementation with a different look and feel, and maintenance and complexity of the applications programs is increased.
- In some applications separate microprocessors could be dedicated to serial communications management and application program execution to enhance the processing capabilities of the serial communications subsystem.

Hence it is an advantage to have a standard interface between the application program and the serial communications subsystem, regardless of the bus protocol being used.

How is the interface between the application program and the serial communications subsystem achieved ?

To achieve a standard interface, a simple set of mechanisms can be implemented within the serial communications subsystem. Some of these mechanisms are described below :-

1) Channel control mechanisms - Connect and Disconnect.
   All communications between the application program and the serial driver code module will be done through a channel. The sequence of events is :-
   - Open a channel. This does not necessarily perform any communications on the CAN bus. It obtains resources for communications and configures the CAN hardware for the appropriate protocol (e.g. baud-rate and message time-outs etc.).
   - Data transfer i.e. obtaining / sending information on the CAN bus.
   - Close the channel. The resources allocated to the channel are released and the hardware in the diagnostic system is made available for other protocols to use.

2) Data access - Setting Parameters and Getting Parameters.
   In an abstract model of serial modules (controllers using nodes on the CAN bus) each module is considered to have a number of parameters. These parameters can contain information such as speed values, voltages, elapsed time, event count etc. The parameters can be identified by an integer identifier value. Actuators can be given a parameter identifier to allow them to be controlled by the Set-Parameter mechanism.

3) Fault code management - Getting Fault Codes and Clearing Fault Codes.
   A common feature of many protocols is that they support the notion of fault codes. These are simple numbers which can be interpreted to give information about the state of the module and its attached components.

4) Bus Monitoring features - Start / Stop Bus Monitoring, Set Filters, Get Next Message.
   If given message types are present on the CAN bus during normal operation, these mechanisms allows the diagnostic application to check for their presence within a sequence of messages and examine their content where required.

5) Diagnostic or self-test routine execution.
   Self-test routines can be written into the product and executed on receiving a command on the CAN bus from the diagnostic system. The result of the test is returned on the CAN bus and returned to the application program.

The above mechanisms are generally sufficient to support the implementation of fault-tree based diagnostic application programs and provide a good framework for the design of the serial protocol driver module.

If a sufficiently object-oriented approach is taken to writing the serial communications subsystem then the code modules that handle each protocol implementation on the serial (CAN) bus can just be ‘plugged in’ or left out as required. Also

What other features can the diagnostic system provide ?

As it was earlier indicated, the diagnostic system need not be restricted to providing fault-finding functions. Here are a few ideas as to some other applications of the diagnostic system :-

1) Uploading history information from products out in the field. This information can be transferred to a central database and used for statistical analysis of a products reliability.

2) Downloading product configuration information in the field to update product software versions in non-volatile memory,

3) Download language information into the product at manufacture time.

4) Downloading different levels of functionality relating to the cost of the product.

5) In-factory testing of products for validation purposes.

6) Performance checking of a live system (e.g. emissions in cars).

7) In-system programming allows the manufacturer to hold generic modules in stock and program them for the specific need after having assembled the product.

8) Product optimization, in actual usage (re-calibration).

9) Provide a means of remotely controlling the product outside its normal range of use.

10) Injection of signals into the product to stimulate responses.

11) The diagnostic system can also be used during product development for configuration and testing purposes.

Conclusion

CAN based networks represent a real opportunity in terms of enhanced product capabilities. However consideration should always be given to the effects of the added complexity that comes with them.

Whilst there are many factors that influence the quality of products as they are released into the market, it is inevitable that a proportion of those sold will develop faults. The knowledge that diagnostic systems are available for product repair increases the perceived quality of the product to the customer, and the experience of an efficient repair contributes to customer satisfaction.

It is strongly recommended that a diagnostic strategy be formulated at the start of the development cycle as this ensures that the product can actually be tested when faults develop (i.e. it encourages design-for-test). Also a diagnostic strategy encourages a more logical approach to product design and makes the design of the diagnostic application more straightforward.