Comparison of CAN Gateway Modules for Automotive and Industrial Control Applications

Jan Taube[1,2], Florian Hartwich[1], Helmut Beikirch[2]
[1]Robert Bosch GmbH Reutlingen, [2]University of Rostock

Bus architectures with up to five independent CAN channels are used in today's automotive and industrial control systems. Caused by the rising numbers of sensors, actuators and electronic control units over the last years, modern control concepts demand devices supporting cross-linking of these channels. This interconnection is realized with a CAN gateway that connects several CAN buses between sub networks at different speeds.

Current gateway implementations are based on one of two concepts. The one concept is an application-specific multi-channel CAN controller with shared message object memory. This concept is inflexible regarding the gateway structure, especially the number of CAN channels, but it enables the transfer of messages between the networks without causing a high load on the host CPU. The other concept is a set of single channel CAN controllers served by a message handling software on the host CPU. This implementation is more flexible regarding the gateway structure, but the load on the CPU depends on the combined bus traffic of all connected CAN networks. Starting from these two solutions, a new concept has been developed, combining the advantages of a flexible structure with a low CPU load.

In this paper, the three concepts are compared and advantages/disadvantages are shown. In addition, problems in the design of gateways are discussed.

Introduction

The increased complexity of automotive and industrial networks and the need for data transparency and information exchange within the overall system led to the introduction of gateways.

Theoretically, the term gateway is not quite correctly used in automotive applications. In the literature, the term „gateway" is used for a network node of a communication network equipped for interfacing with another network that uses different protocols. It may contain devices such as protocol translation, rate converters and signal translators to provide system interoperability.

In that context, the term „bridge" is used to describe a device of a communication system that links or routes signals from one bus or network to another, to extend the distance span and the capacity of a single network. It does not modify packets or messages, it only reads them and forwards those with destinations not on the same segment of the network as the transmitter.

In automotive and industrial control applications, the term gateway is preferred even though the data is transferred between networks using the same protocol, because these gateways perform more functions than the forwarding of messages.

These functions [5] can/must be message filtering (to prevent the overload of a low-speed network when transferring messages from a high-speed network), message transfers with identifier translation, message integration (combining parts of the data of several messages into a new message), and the synchronisation of time-triggered networks (when implemented) to guarantee that the information is updated on time.

In general, the gateway functionality could be implemented in software, as long as several CAN modules are available in the ECU. But a large amount of messages would cause a high load on the CPU, leaving less performance for the ECU control applications until real-time operation can no longer be guaranteed.

Therefore dedicated gateways have to be developed with the objective of reducing the demands on the CPU performance. However, there is not one single solution fitting for all applications; a concept is required that can be easily adapted to different demands.

This paper wants to compare available gateway implementations with a new innovative structure that combines the advantages of both gateway concepts.

1 Gateway Implementations

Gateway implementations which can be found in present-day automotive and industrial applications are based on one of two concepts. The first concept is a set of discrete channel CAN controllers served by a message handling software on the host CPU. This concept is flexible regarding the number of CAN channels, but a high-performance host CPU is required to ensure real-time operation at full bus-load.

The second concept is an application specific complex channel CAN controller. This concept is inflexible regarding the gateway structure, especially the number of CAN channels. Furthermore such gateways need elaborate control mechanisms. However, this structure supports the transfer of messages to other networks without causing a high load on the host CPU.

A third concept is the new modular gateway. These gateway concepts will be described in the following text.

Discrete Channel Gateway

The most distributed gateway concept is the discrete channel gateway. This gateway consists of the components CPU, CPU Peripheral Bus and several single channel CAN modules (Figure 1).

There are different implementations of this gateway concept available, depending on the preferences of the manufacturers. The CPU may be a CISC or RISC machine. In most applications, its software controls not only the gateway function, but may also include some other tasks. Each CAN module is connected to the CPU via the CPU's peripheral bus. During the gateway operation, the CPU needs to read all necessary control informa-

tion and received message objects over the peripheral bus from one CAN module and then writes the same data over the peripheral bus to some other CAN module(s). Concurrent message transfer requests from different CAN channels are served sequentially.
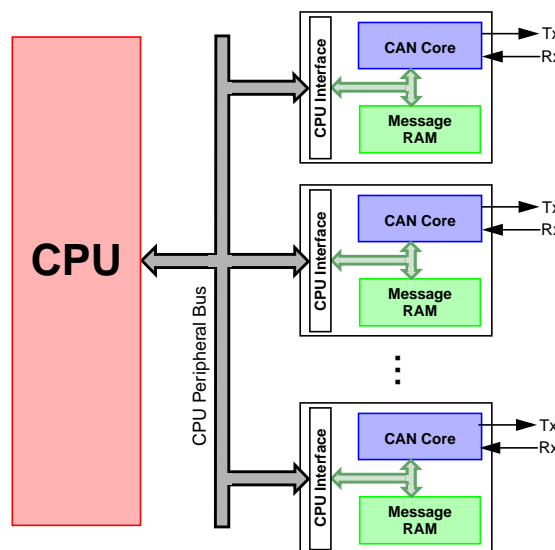

Figure 1: Structure of discrete channel gateway modules

The number of CAN channels connected to the CPU's peripheral bus can easily be adapted to actual requirements, but a rising number of CAN channels will increase the CPU load even if the CAN bus-load remains unchanged.
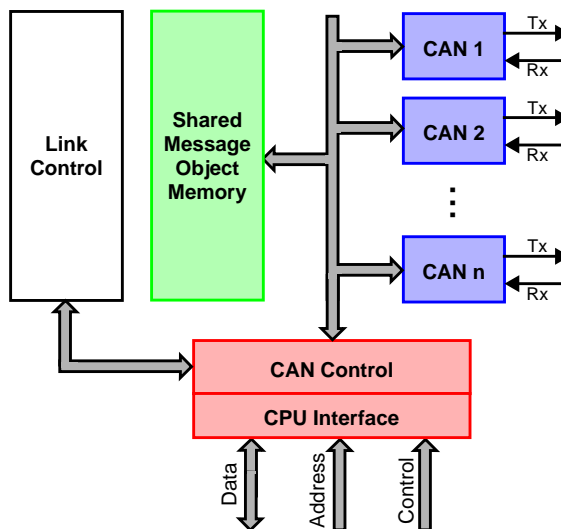
Complex Channel Gateway

Figure 2: Structure of complex channel gateway modules

The complex channel gateway, which was designed in the last years, is a concept to

reduce the demand for CPU performance. It is more elaborate than the discrete channel gateway and consists of a shared Message RAM, several CAN Cores and an internal Control Unit (CAN Control). In some implementations, a Link Control is added to provide a basic gateway functionality without causing any CPU load (Figure 2).

The system design is comparable to a single channel CAN module with two or more CAN protocol controllers. A CAN protocol controller performs the serial communication in a CAN bus system according to the CAN protocol specification. The CAN Control unit manages the data flow between the CAN protocol controllers, the Message RAM and the CPU Interface, respectively the CPU itself. It also controls the access of the different instances to the Message RAM and prevents data corruption. The Message RAM is implemented as shared memory, the messages for all CAN channels are combined in the same RAM block to reduce the need for internal data transfer and therefore to reduce the CPU load. The optional Link Control unit is configured with the fundamental routing rules. These rules are checked when a new message is received. If a rule for a message is defined, it will be performed by the CAN Control unit without any need for CPU action. Different functions may be provided, dependent on the complexity of the Link Control and CAN Control units. This can be a simple copy of the complete message, a transfer of the message data with a translated identifier, up to message integration, where data of several messages is combined into a new one.

This concept, especially when it includes a Link Control unit, reduces the CPU load significantly. Its disadvantage is that it is complicated to change the number of CAN channels. This would require major changes in the Link Control unit and in the CAN Control unit, especially in the arbitration of concurrent accesses to the shared Message RAM. Up to now, there is no complex channel gateway available which supports more than 5 CAN channels.

Modular Gateway

The modular gateway is based on a proven single channel CAN module (Figure 3) which

is expanded with a gateway interface (Figure 4). Several instances of this adapted single channel CAN module may be combined and be turned into a gateway controlled by an application specific Gateway Unit (Figure 5).
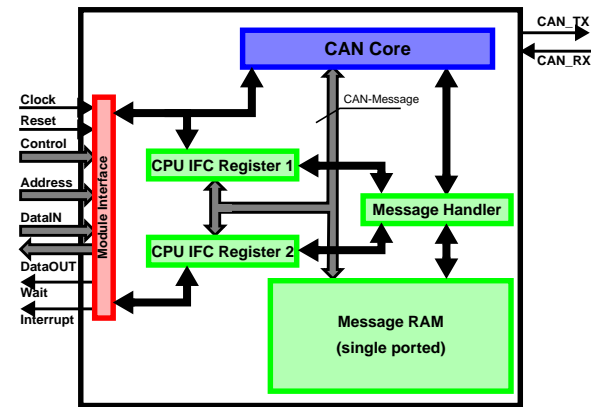

Figure 3: CAN Module w/o Gateway Interface

The single channel CAN module (in this case, Bosch's C_CAN IP, Figure 3) comprises the components CAN Core, Message Handler, Message RAM and CPU IFC Registers. The CAN Core performs the serial communication on the CAN bus. Individual messages can be (pre-)configured in the Message RAM and are managed by the Message Handler. This includes the transfer of messages between the CAN Core and Message RAM, acceptance filtering and the handling of transmission requests and interrupts. Two sets of CPU Interface Registers are used for the data transfer between the CPU's peripheral bus and the Message RAM. They consist of the complete data, header and control information, which are moved as one single word on the internal data bus.
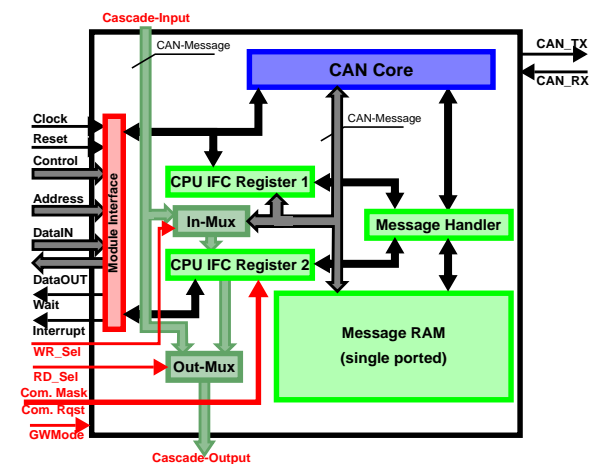

Figure 4: CAN Module with Gateway Interface

The existing single channel CAN module is expanded by several functional blocks to adapt it into a gateway cell. These functional blocks are an input multiplexer In-Mux and an output-multiplexer Out-Mux together with the necessary control signals to direct the data flow (Figure 4).

The two multiplexers give access to the internal data bus, making it possible to load the complete CPU IFC Register in parallel from a wide input port (Cascade-Input) and to export the contents of the CPU IFC Register over an equally wide output port (Cascade-Output). The Cascade-Input may also be routed directly to the Cascade-Output.

These two wide ports allow the transfer of a complete CAN message and necessary control information in one step directly from one CAN cell to other CAN cells, avoiding the bottleneck of the CPU's peripheral bus.
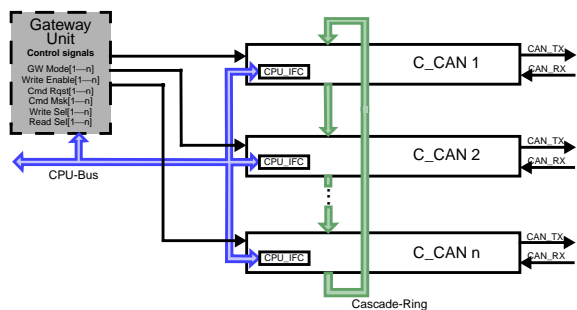

Figure 5: Structure of modular gateway module

When several instances of this adapted single channel CAN module are cascaded into a gateway module, the wide input and output ports are connected to the cascade ring bus. This allows the transfer of a complete message and control signals to all connected cells in one clock cycle. The data flow between the CAN cells is controlled by an application specific Gateway Unit that provides the control signals for the multiplexers and the information to load/store a message from/to the Message RAMs.

2 Comparison of Gateway Concepts

Semiconductor manufacturers and system designers will use different parameters when they compare the advantages of gateway modules. For semiconductor manufacturers, these parameters may be the module's gate count, its adaptability to different numbers of CAN channels, as well as the possibility of interconnection with several other protocol interfaces (e.g. FlexRay, LIN, MOST, etc.). For the system designers, these parameters may be the flexibility in programming the gateway functionality, access time for read/ transfer of messages, or the required CPU performance. In general, a compromise has to be found between these parameters, especially the system structure/module size and the needed CPU performance.

Discrete channel gateways are solutions optimized for modularity and module size. Several CAN cells can be connected to the CPU bus, only the address space has to be adapted. No semaphores or additional flags are necessary to control concurrent requests to a message buffer by several instances, because only the CPU can access the cells. This allows the interconnection of an unspecified number of CAN channels.

The simple system structure without any additional interconnecting logic reduces the module size to a minimum. The lack of hardware support for gateway functions however increases the need for CPU performance. All data transfers between two or more cells, data manipulations, and insertions need to be executed by the CPU. Especially the sequential read and write cycles for a message transfer between cells cause high CPU burdens. In summary, a high number of CPU cycles is unavoidable. This number of cycles is increased by the interrupt handling or by the polling of the connected cells (to check for receptions), and by special functions like the already named data manipulation or the combination of several messages.

Even regarding the high CPU burden, this gateway model provides the most flexibility in programming, since the gateway functionality is implemented in software. Such a gateway model meets the requirements of a wide range of gateway applications.

In automotive and industrial control applications, where the numbers of interconnected CAN cells and routed messages cause a very high CPU load, gateway models are needed that provide additional functions. One such gateway model is the complex channel gateway that implements a wide range of functions in hardware, its central component is the shared Message RAM.

The messages for all connected CAN channels are configured and stored in the same RAM block. A message that is to be received on one channel and to be transmitted on another channel will occupy only one segment of the RAM block. This feature reduces the CPU load, because it supersedes the transfer operations of the message from CAN cell(n) to the CPU and then from the CPU to CAN cell(m). Automatic transmissions of received messages on other networks can be started if a Link Control is implemented. The CAN Control Unit detects the reception of a message and checks the routing rules in the Link Control. If a rule is defined, it is executed by the CAN Control Unit. Several functions can be implemented, dependent on the complexity of the Link Control and CAN Control Units. This can be the simple copy of a message onto another network up to the merger of several messages into a new one, combined with cyclically updated transmissions.

The reduction of the CPU load is paid with less flexibility in the system design. A complicated control system is needed to transfer the messages between the CPU, the CAN cells, and the RAM, arbitrating between possibly concurrent requests by several cells to the shared message RAM, requiring flags and semaphores to assure data consistency. A priorisation of all modules is required to define which unit gets access to the RAM. When a specific application of the gateway needs new transfer functions or a different number of CAN channels, this will require a redesign of the whole gateway structure (especially link control and CAN control).

The modular gateway is a merger of discrete channel and complex gateway, combining the advantages of both concepts. The optimized structure allows a fast data transfer between several cells without causing a high CPU load [4]. If an internal state machine is provided, the CPU load can be reduced to a minimum. The transfer of messages from one Message RAM over the cascade ring to one other Message RAM takes more time then in a complex channel gateway with Link Control unit where no data transfer between two CAN cells is necessary. However, the transfer over the cascade ring takes less then two CAN bit times and the cascade ring

has another advantage: It allows the transfer of a message to one, several or all connected cells simultaneously with nearly the same effort.

The modular structure allows also a flexible programming of the gateway function. Even when the gateway function is controlled by a finite state machine, the CPU keeps full access to all functions of each CAN cell. For example, it can write or read messages and can start their transfer. Concurrent requests of the CPU and of the FSM to the same cell are solved in a deterministic way, semaphores and flags are not necessary. This maximises the flexibility of the module. If additional functions beyond a simple message transfer/copy are required (e.g. the merging of messages), special modules that implement this features can be inserted into the cascade ring.

Different applications need quite different solutions. A modular structure allows to design a new gateway by combining components of a library, speeding up the design time significantly. The size of such a module is marginally larger than that of a discrete channel gateway structure, it is increased by the Gateway Unit and the optional message manipulation functions.

The modular gateway structure is not restricted to the CAN protocol, it is possible to add several other protocol interfaces to the cascade ring. These can be different bus systems like TTCAN, FlexRay or LIN. When implementing the time-triggered variant of CAN, the gateway structure (incl. gateway control unit) can be the same. Only the concerned CAN cells have to be replaced by the time-triggered ones. A different interface structure then the cascade ring might be used for the implementation of bus systems transferring multimedia data (e.g. MOST, IEEE 1394, Bluetooth) because of different requirements regarding higher data rates on the one hand and less emphasis on transmission reliability, security, and time schedules on the other hand.

An exemplary implementation of the modular gateway structure was tested to demonstrate the functionality of the cascaded ring structure. It consists of three CAN nodes, enhanced by a data integration unit (mes-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

fastcase
Smarter legal research.