Double-CAN Controller as Bridge for Different CAN Networks

Jens Eltze
NEC Electronics (Europe) GmbH, Düsseldorf

Abstract

Due to conceptional reasons car manufacturers now implement not only one, but two (and even more!) CAN networks within each car. Such network structures require a transfer of data between the networks. This paper gives a brief introduction how such operations can be supported by special microcontroller modules.

As transfer mechanisms for data between the CAN network subsystems are not well established yet, but under deep and controversial discussion, the aim of this paper is not to give a detailed technical description of the discussed solutions. It tries to describe general aspects and strategies for such transfers.

Introduction

While in some cars the manufacturers now begin to implement a very first CAN network with only a few nodes, other cars see a fast increasing number of modules which are connected via CAN. As the maximum load of bus systems is limited, it is necessary and useful to split the whole network in several subsystems. Certainly some messages and information available on one bus are also required on the other a vice versa, so a data exchange between the several bus subsystems must be realised.

Transfer of Messages

Using CAN for communication in automotive systems, a protocol is chosen which is not address, but message related /1/. The message identifier defines the bus priority of each message.

Messages which should be available on several CAN subsystems unfortunately can not be copied “as is” between these subsystems. One reason therefor is that message identifiers are selected in such a way that the individual requirements, e.g. message priorities, of each subsystem are met.

Another important aspect is that messages are combined/packed to reduce the busload. As only parts of the data, sent within one message on a bus system, need to be transferred to the other network, a selection of data is necessary.

Last not least the bus speed, and also the message frequency, of the different bus systems is different. That means that a message available on a high-speed bus should not be copied to the low speed bus each time it appears, because otherwise the load of the low-speed bus would increase drastically. So filter mechanisms (either by time or by appearance) must be available.

Strategies to Transfer Data and Messages

To transfer data between different CAN bus systems there are several solutions possible. Any realistic, that means efficient (performance, flexibility, prices) solution is located between the following two strategies: transfer and filtering done completely by CPU, or transfer and filtering done completely by additional logic. The following figure illustrates the different strategies.

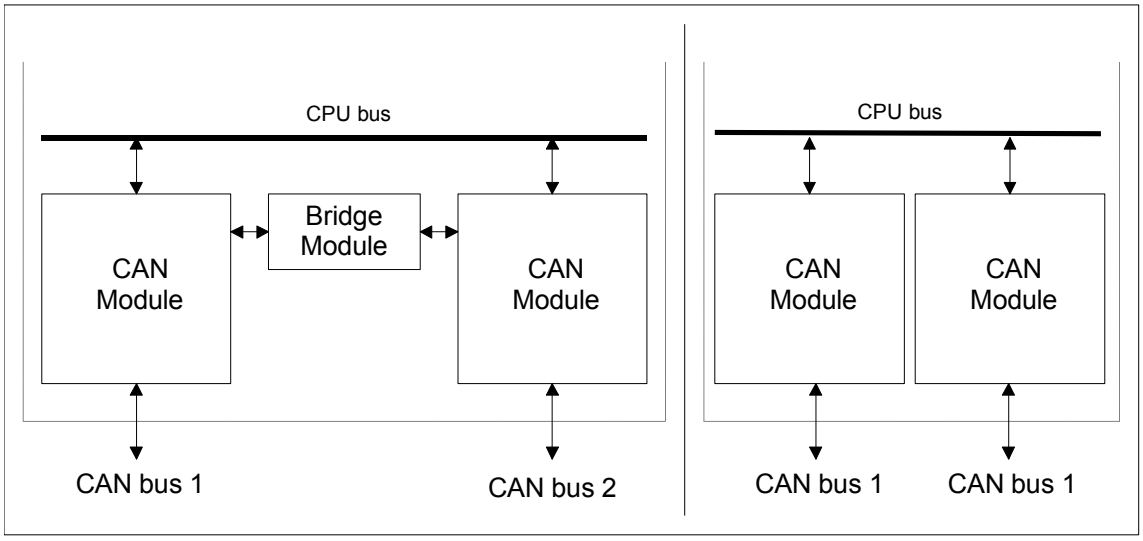

Figure 1: Strategies to transfer data between different CAN bus systems

In the personal opinion of the author the "perfect" solution is a bridge module, as shown in the figure above on the left side, which has the same characteristics as a normal CPU core, that means that it is programmable and that it allows manipulation of data. Such an additional "core" would result in a high impact on costs, so it is not reasonable today, but probably within a few years (remember the incredible development speed and the increasing performance with smaller chip sizes).

The other possibility, shown on the right side of the figure above, used the CPU to do the filtering and transfer of data. This means that the functions have to be executed by software.

While the solution described first allows a nearly “independent” run of the CPU, the other solution requires remarkable activities by the CPU. Especially the CPU load caused by interrupts when receiving messages via the CAN bus has strong influence on the reduction of CPU performance.

As pricing is a major criterion for the selection of components in the automotive area, the selected system has to be chosen carefully. Spending a separate microcontroller just for handling the bridging functions seems to result in non-acceptable costs. Also a new node has to be adapted to the CAN network, which influences the total number of network messages even due to the network management functions.

So actually the bridge function is realised with a microcontroller which is already available for other purposes. To avoid overhead, a location/module is chosen which has to get information from both CAN busses. Looking at actual CAN network architectures the dashboard is such a location, as it needs data from the engine bus system as well as from the body area.

Double-CAN as Bridge

As mentioned above, a separate “core” for the bridge functionality is not competitive in price at the moment. So NEC decided to develop a microcontroller, which allows handling of several CAN networks while serving other applications as well. Therefor NEC designed the new Full-CAN macro and the related device in such a way that is allows the most flexibility for future adaptation and integration.

Structure of CPU core

For the first devices NEC's own V850 RISC core[/2/] is selected. The following figure shows the structure of that CPU:

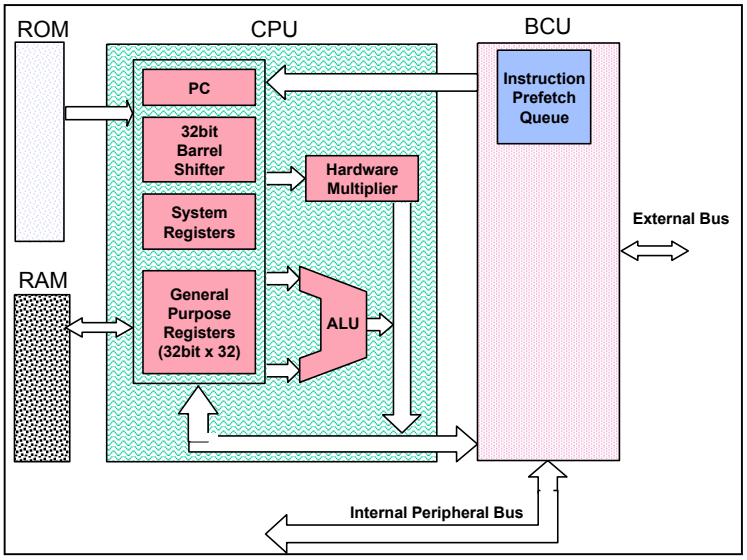

Figure 2: V850 CPU architecture

The V850 CPU has an internal 32bit structure, which shows excellent performance at low power consumption. Due to the mixed 16/32bit-instruction set, the code size is comparable to a standard 16bit CISC core, and also the low gate count of the V850 core competes with a 16bit device. Another aspect to mention is the short interrupt response time of V850 cores. If a system is chosen where the transfer is supported by additional hardware, the interrupt response time certainly has no major influence on bridging the data. But if a system needs CPU support for transferring data, it is very important to provide enough CPU performance for the main application. Due to these features a double Full-CAN device based on this core offers the power to handle two Full-CAN interfaces while serving the main application as well.

Structure of Full-CAN macro

Although a very powerful in-house CAN module is already available (DCAN [/3/]), NEC decided to design a new Full-CAN module. Reason therefor is that NEC's DCAN module is ideal for microcontroller which are connected to one CAN node, as it has Full-CAN capabilities on receive side, while transmission of messages is like a Basic-CAN. But for a device that should handle several CAN bus systems, Full-CAN behaviour seems to be necessary on receive side as well as on transmit side.

The basic structure of the Full-CAN module is given in the following figure:

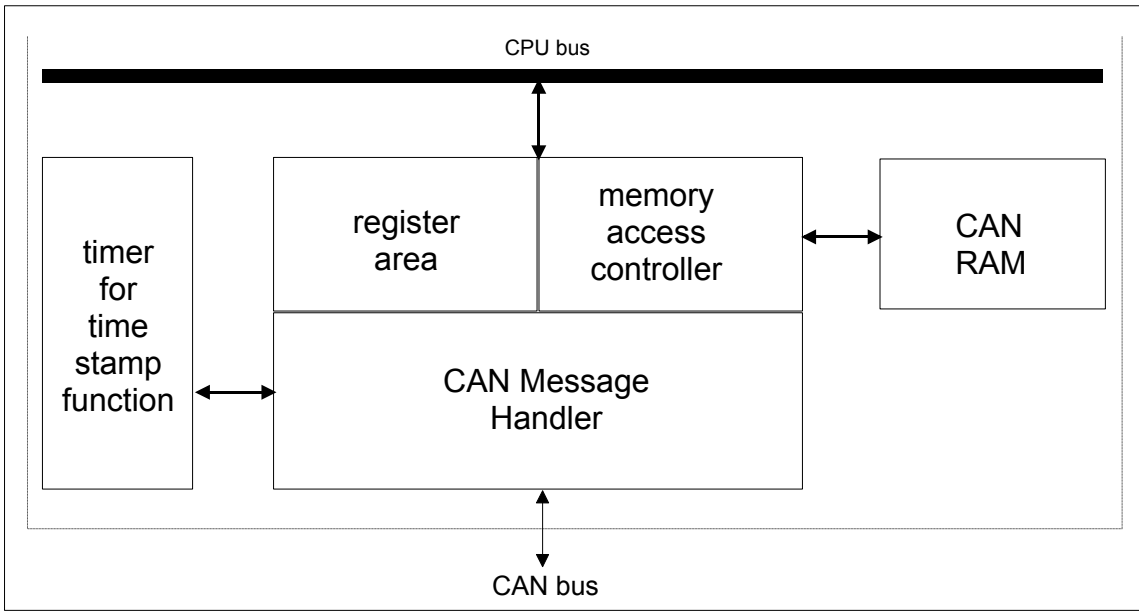

Figure 3: Structure of Full-CAN module

The CAN bus messages and their identifiers are stored in a special RAM area (CAN RAM) which can be accessed either by the CPU or by the CAN Message Handler (CMH). The Memory Access Controller (MAC) controls the access to the RAM. Data consistency – a major topic for CAN designs – is assured by special features of the MAC and by specially designed access mechanisms of the CMH. For easy access to important status information a register area is added to the module.

As known, NEC's µPD72005 /4/ introduced the time stamp functionality to the CAN world a few years ago, which is a base for the global time system support /5/. Time stamping is now well established in the CAN network systems.

Certainly the new Full-CAN module also handles time stamping of messages. Therefor a separate timer is linked to the CAN module. Whenever a new received message is stored in the CAN RAM, a time tag is automatically added. As for each message in the CAN RAM 16bytes are reserved, though the standard message data could be stored in 14 or less bytes, the remaining 2 bytes are used to store the time indicator (see Figure 4).

The allocation of messages in the CAN RAM is given in the next figure (example for a CAN configuration with 32 message buffers).

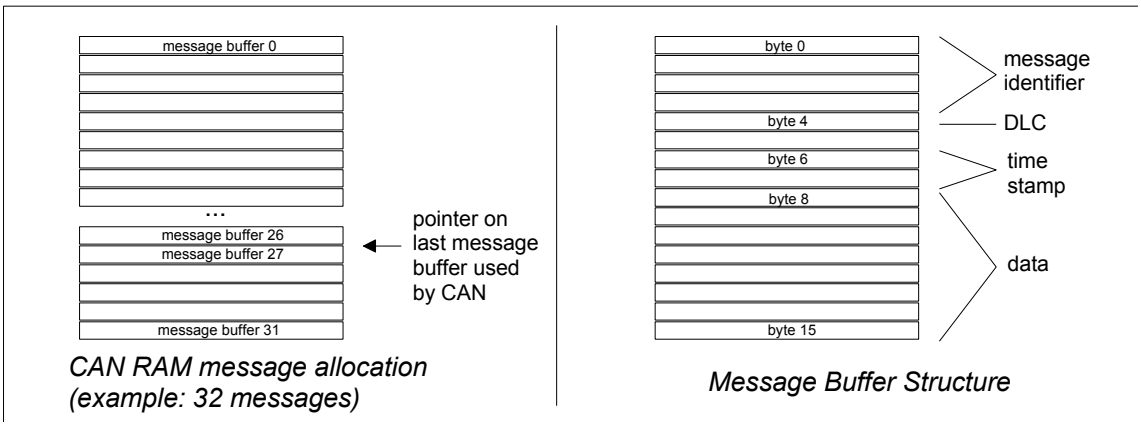

Figure 4: CAN RAM – message allocation and message buffer structure

The message area is not split into separate areas for receive and transmit messages. This allows a better handling of messages (by the user software) for adding new messages to the CAN during operation. On the other hand transmission and reception of data requires higher efforts, as all the message buffers have to be scanned.

To reduce the access to the CAN RAM by the Full-CAN module the scan operations for transmission and reception are kind of synchronised. This allows the CPU as well as additional modules, for instance a bridge, to do operations on the CAN RAM without speed reduction. Also an implementation of a high number of Full-CAN messages is possible. Nonetheless Basic-CAN functionality is also available.

"Open" Design for Full-CAN macro

As shown and described in the section above, the RAM area where CAN messages and data are stored, is accessed via a memory access controller. The memory access controller is designed in such a way that it also includes an interface for additional hardware. While for a device with a single Full-CAN this interface could be used for instance to handle network management functions, for a double Full-CAN device a module for the bridging support could be added.

For critical applications where high CPU performance is needed, such a bridge module can reduce the CPU performance required for the CAN handling. For other applications, where normally a 16bit CPU is sufficient, the V850 in combination with the Full-CAN – strategy has enough power to handle both "tasks" without the help of additional hardware. So a cost efficient solution is possible.

Summary

Due to the availability of a powerful CPU core (V850) in combination with the "open" Full-CAN design, NEC's new automotive controller series with 2 Full-CAN interfaces will allow a flexible adaptation of the microcontroller to the applications. Especially the "open" design will allow to implement additional hardware for the transfer of messages between the CAN network subsystems with minimised CPU overhead.

Literature:

/1/ Robert Bosch GmbH: CAN Specification, Version 2.0, September 1991
/2/ NEC: V850 Family User's Manual - Architecture
/3/ data sheet µPD78094x
/4/ data sheet µPD72005
/5/ K. Turski: A Global Time System for CAN Networks, 1st international CAN conference, 1994