Advanced FullCAN Architecture New AFCAN Macro with Diagnosis Support

Wolfgang Wiewesiek, NEC Electronics (Europe) GmbH

Since 1996 NEC provides CAN interfaces. The DCAN is a single channel interface and the FCAN offers up to 5 independent channels with a focus on supporting gateway applications. These are well-established macros used in all areas of the automotive field. The successor macro AFCAN (advanced FullCAN) is designated to replace all existing CAN macros of NEC on all CPU platforms in order to provide a single architecture for the customer. It will help reduce the cost for maintaining many different S/W drivers. This has been recognised as a strong request from car manufacturers. Along with this strategic concept, NEC introduces new technical features while keeping the highly accepted features of the current FCAN. The frequent tasks like receiving and transmitting messages are now supported by separate "History Lists" that speed up these processes and provide to keep track on the sequence of messages without any CPU burden. Further the AFCAN features an "Automatic Block Transfer" mode with programmable delay in between those transmitted messages. The AFCAN carries 32 buffers per channel (available since b/o 2003). First products with 48 buffers and 16 buffers per channel are scheduled for e/o 2003. A derivative of the 48-buffer version features an integrated listen only channel in order to support monitoring of other CAN channels for diagnostic purposes.

Introduction

With the AFCAN (Advanced FCAN) NEC offers the third generation of its CAN macros. To understand the motivation of NEC to create a new type of a CAN macro, a short glance at the history of the developments of CAN macros needs to be taken.

In 1993 NEC started to develop CAN macros. The first macro, the DCAN, was mainly used in automotive applications at that time. This single channel macro provided 16 receive and two transmit message buffer. From performance point of view it could be regarded as a FullCAN type of implementation already. At least the reception path equipped with two BasicCAN objects was able to fulfil the requirements requested by dashboard applications at that time. Nowadays the DCAN is found on 8-bit and 32-bit cores with low and medium communication requirements. However the ongoing trend to more message objects and the support of more than one network made a new CAN macro inevitable. Also the quite rudimentary hardware functions of the DCAN provided to the CAN driver software were not sufficient anymore to meet the demands for quick service and thus short run-time of the driver routines.

Therefore, in 1999 NEC presented its first multi-channel macro, the FCAN. It features up to 64 message buffers, which can be freely assigned to one of the up to five CAN-channels. In comparison to the DCAN, this macro truly provides FullCAN performance for reception and transmission path. As well the FCAN offers several features that eases the driver software like i.e. the bit set/clear function for all control registers. It facilitates easy bit manipulation when using C without being forced to use bit fields. The FCAN also offers search functions for the host CPU. Thus the CPU does not have to browse the entire message buffer RAM for finding i.e. the last received message.

Many features of the FCAN belong still to the list of requirements for today's applications in the automotive field. However some new aspects have recently been recognised, which necessitate a new step of development. First of all, there is a new standardisation process ongoing. The 'Hardware Initiative Software' (HIS) as promoted by several players in the automotive field, requires standardised communication drivers throughout all platforms. Thus an approach to offer different CAN macros for each CPU core does not meet anymore the scenario put in place in the next years.

As well new demands to provide enhanced diagnosis support in communication systems comprising several CAN-bus systems with different bus speeds have been identified.

The AFCAN ('Advanced FCAN') macro meets those new challenges.

AFCAN Architecture

The concept of the AFCAN architecture follows the target to provide identical behaviour of the communication link independently of the host processor, or the amount of message buffers.

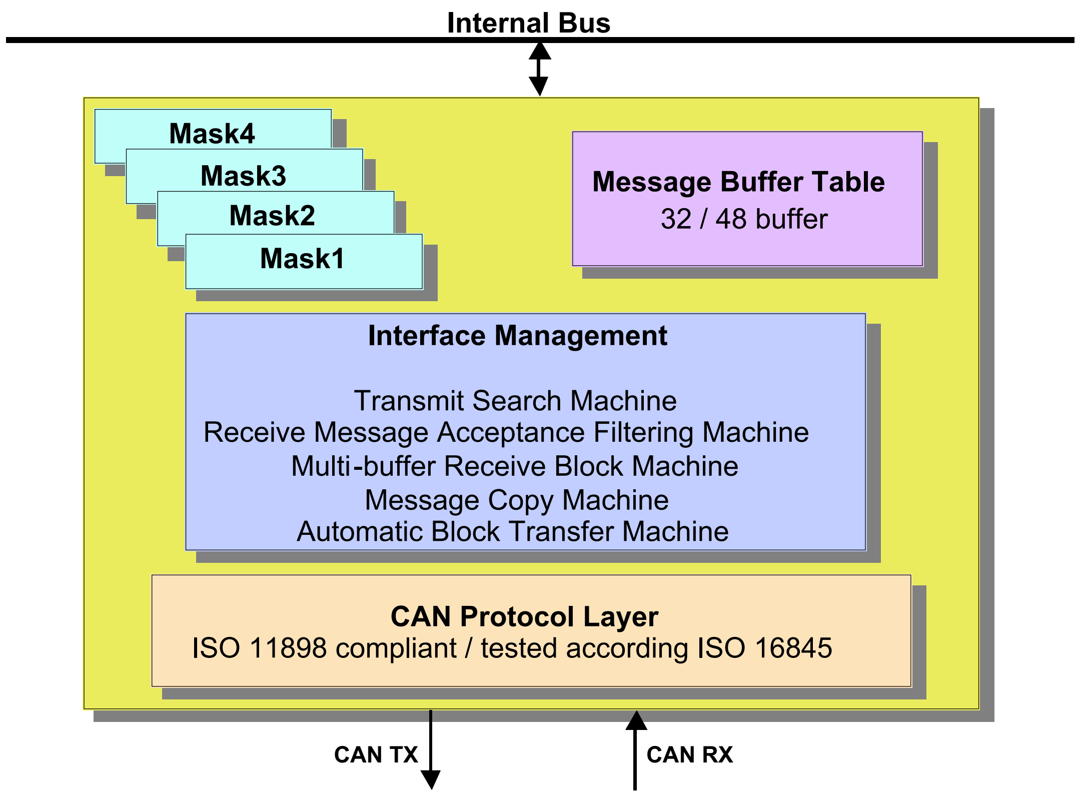
The user can use the same C source code of the driver on an 8-bit CPU and on a 32-bit core because the interface of the CAN macro to the host CPU, the NEC Peripheral Bus (NPB), is common to all devices of NEC.

The amount of message buffers for a particular instance of an AFCAN macro is merely a parameter that needs to be set up during compile time.

The demand to have identical behaviour even when the number of supported CAN-channels vary, led to the conclusion that a new CAN-interface can only be a single channel macro. Any interdependency of one CAN-bus to another one can be eliminated that way. Thus the previous concept of the FCAN, a multi-channel design, was revised. The overall architecture of the AFCAN shown below can easily be expanded to the required number of CAN-bus systems.

A single AFCAN instance can be scaled to provide 16, 32, or 48 message buffers. For low-end applications, typically just asking for a slow 8-bit core, 16 message buffer are regarded as sufficient. Still the AFCAN provides the full performance. The bus speed of 1 MBPS is even achieved as long as the host system runs at 8 MHz.

Medium and high-end communication requirements are fulfilled with the 32-buffer



version as it is seen on 32-bit cores. In those applications a second or any other number of instances of the macro can be added.

For gateway applications, the 32-buffer version may not always satisfy the demands. Thus NEC provides a 48-buffer version, which needs 16 MHz operating frequency to achieve correct operation at maximum bus speed. For lower bus speeds, i.e. 500 KBPS, the source clock can be reduced to 8 MHz as for the other AFCAN versions.

The features described below are common to all AFCAN versions.

Features

Transmit History List

The sequence of transmitted message can be tracked by the application without remarkable efforts. The AFCAN offers a 'Transmit History List' (THL) for that purpose. It is a record of seven, most recently transmitted messages. The driver software just needs to read a single register were the pointer of the oldest transmission is stored. This THL-register provides a pointer to the respective message buffer. With every read access to the THL the AFCAN will update the pointer to the next entry (if any). The advantage of this method is that there is no need for complicated evaluation of time stamps by the host CPU to recover the exact order of transmissions.

Receive History List

The analogue method is available for the reception path. The 'Receive History List' (RHL) even comprises a record of the last 23 receptions. Intentionally the reception path is equipped with more than three times the capacity as the transmit path. The number of received messages is and will also be in future substantially higher than transmitted messages. The benefit for the application is that it does not have to react that quickly on received messages but still does not loose their sequence. This can be of advantage when i.e. parameter downloads are mixed with control messages. The host processor will always be able to find out at which time the control message interleaved the parameter data in order to apply i.e. different processing algorithms to the following data at the correct point in time.

Multi-Buffer Receive Block

A Multi-Buffer Receive Block (MBRB) is built by a variable number of RX-message buffers grouped by the same configuration (identifier, mask). The AFCAN will store messages into this block without overwriting messages the CPU has not read yet. Even if the same identifier was received, the AFCAN stores the message at the next higher buffer number. The Data New (DN) flag of each message buffer indicates the service status. Similar to a FIFO, the interrupt generation can be invoked when the MBRB is full. Also other states, half-full or 7/8-full, can be configured as the assignment for interrupt generation can be done individually for each message buffer.

The CPU clears the Data-New (DN) flag of a buffer when it has been read. Then this buffer becomes available again to receive a new message. The AFCAN will always store the next message in the buffer with the lowest buffer number. Thus the sequence of buffer numbers may not always reflect the occurrence of the messages on the bus anymore. This may be confusing at first glance, but thanks to the 'Receive History List'-feature, the host processor has no problem to follow the correct sequence of the receptions. Even when the CPU does not use interrupts on MBRB and polls the contents in cyclic intervals, the sequence of received messages is safely retrieved with RHL function.

The RHL-function is a valuable new feature. The MBRB in conjunction with the RHL-function even enhances the performance of a similar function that the FCAN provides. The host processor of an AFCAN has to master almost no administrative task to use this feature – a clear benefit for the software driver.

Automatic Block Transfer Mode

Typical application scenarios are i.e. parameter downloads or transferring programming data without instantaneously occupying all bandwidth. The Automatic Block Transfer (ABT) mode provides a handy method to encounter these scenarios without asking the host processor to monitor the busload.

When the user invokes the ABT-mode, the transmissions from the lower eight buffers are automatically handled by the AFCAN. A single request for transmission will sent all message objects out of that group. The TX-objects in the ABT-group are sent in ascending order of the buffer number, i.e. irrespectively of any identifier priority among them. However these messages will not necessarily be sent back to back as a programmable delay becomes effective before the next transmission is launched.

This programmable delay timing between TX-messages of the ABT-group allows system design to limit the peak load of these messages a priori. The delay time can easily be calculated as the unit is automatically adjusted to “bit time”. If for example a program download shall use only 50% busload, the user just needs to know the average message length of the TX-messages. Assuming 100 bit times for the typical TX-message delivers also 100 for the value to be written into the delay register. The delay time is started anytime a transmission from the ABT-group finishes. Of course this delay requirement is only valid for the lower eight TX-objects in the message buffer area. When the application has assigned additional TX-objects elsewhere, these TX-objects are sent as soon as possible. Note that the current message object from the ABT-group will still have to arbitrate against other TX-objects in the AFCAN also flagged for transmission. This way other TX-communication can be interleaved to the ABT-mode even when the delay timing is set to a minimum.

Options for Diagnosis Support

In multi-channel environments there is often the necessity to monitor the exchanged data of all CAN-bus systems. For cost reasons or mechanical obstacles, only a subset of all CAN-channels, typically only one channel, is connectable to a diagnosis plug from where an external node can access the data. Thus a multi-channel node (gateway) needs to perform the transfer from a particular source CAN-bus to the diagnosis bus. A small outlook on the pros and cons of the potential or existing solutions shall explain the challenges in this field.

Some current realisations simply assign this task to the host processor of the gateway. The advantage of a very flexible copy process is given here. Even reformatting of the data received from the monitored bus can be issued before it is sent to the diagnosis CAN-bus. However, there are quite a few disadvantages that need to be taken into account. In first place, the gateway suffers a real-time infringement during the diagnosis mode because the host processor has to handle all messages. As a consequence, no guaranteed latency time for the monitored messages can be given anymore. An even more severe limitation is linked to that method as only messages normally received on the monitored CAN-channel of the gateway are transferred to the diagnosis bus. Other messages remain invisible to diagnostic system/node.

Other implementations supporting diagnosis on multiple CAN-bus systems use relays. The clear advantage to access the monitored bus in real time unfortunately is linked to quite high costs. Each CAN-bus that shall be monitored requires one relay including the control circuitry. Additionally more wire harness becomes necessary. Besides those burdens, only a CAN-bus that has the same bus speed than the diagnosis bus can be monitored.

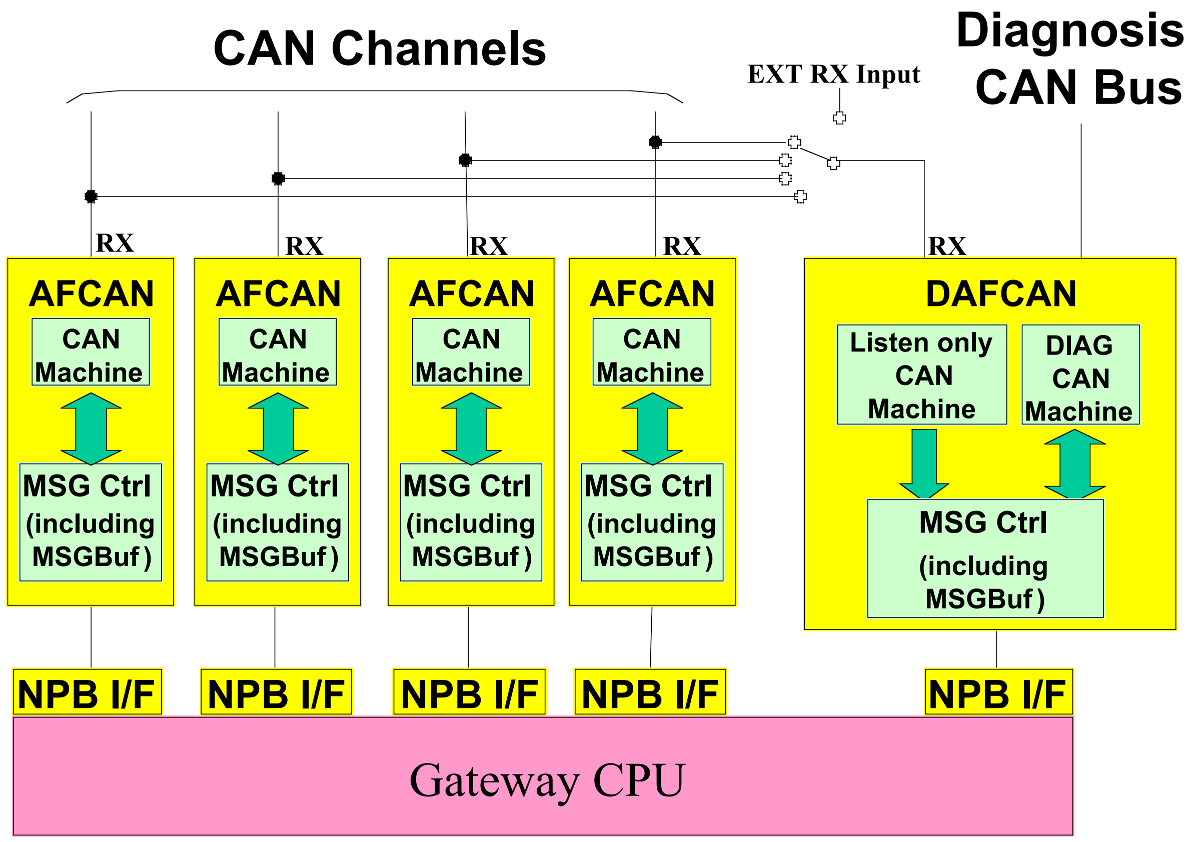
A third solution that NEC offered with the FCAN is a dedicated ‘bridge’-processor. This approach avoids already many of the previously named disadvantages. It is flexible, it can handle different bus speed on monitor versus diagnosis CAN-bus, and it does not increase the load of the host processor. However, this concept does require a common message buffer memory for all CAN-channels of the gateway. To avoid long lasting search algorithms, a specific contents addressable memory (CAM) is mandatory then. These CAM are subject to be re-engineered anytime the manufacturing process changes, which counteracts the target to provide one solution for all platforms. Besides that a standalone 'bridge'-processor is highly oversized if the purpose is just to provide messages from one CAN-bus to a dedicated diagnosis CAN-bus.

Diagnosis with DAFCAN

The DAFCAN follows a new idea to meet the requirements for diagnosis in multi-channel environments, for which NEC recently submitted a patent. The key element is the addition of a second 'listen-only' protocol core to an existing AFCAN. This second protocol core, the RXONLY-channel of the DAFCAN, picks up the signal from the other CAN-channels at their RX-pin, one at a time. The host processor selects the monitor channel before invoking the diagnosis session. The example below shows a 5-channel gateway with four ordinary AFCAN macros and one DAFCAN that is connected to the diagnosis CAN-bus (DIAG). Each CAN-interface provides 48 message buffers. That concept will be used in a new device that in short term is applied at major car manufactures.

Once the RXONLY-channel of the DAFCAN receives a valid message, i.e. this is any message that has been exchanged on the monitored CAN-bus, it stores it into the message buffer memory of the DAFCAN. The newly invented 'Mirror Mode' will take care to send the received message to the diagnosis CAN-bus automatically as soon as the bus becomes available (idle).

The benefit of the architecture is that a CAM becomes obsolete. It would be anyway a challenge to design such a RAM for a capacity of 240 message buffers. The architecture fulfils all the other requirements that a diagnosis asks for: no



additional CPU load for the gateway processor, no additional, external components needed, all valid messages are recognised, and predictable, short latency for all monitored messages.

Further the adaptation of a different baud rate between monitored CAN-bus and diagnosis bus is automatically solved by the listen-only protocol-core. Whenever a particular channel shall be monitored the gateway processor sets up the baud rate of the source channel in the RXONLY channel of the DAFCAN in advance. During the diagnosis session the Mirror Mode automatically provides the time gear between source and destination CAN-bus.

Mirror Mode

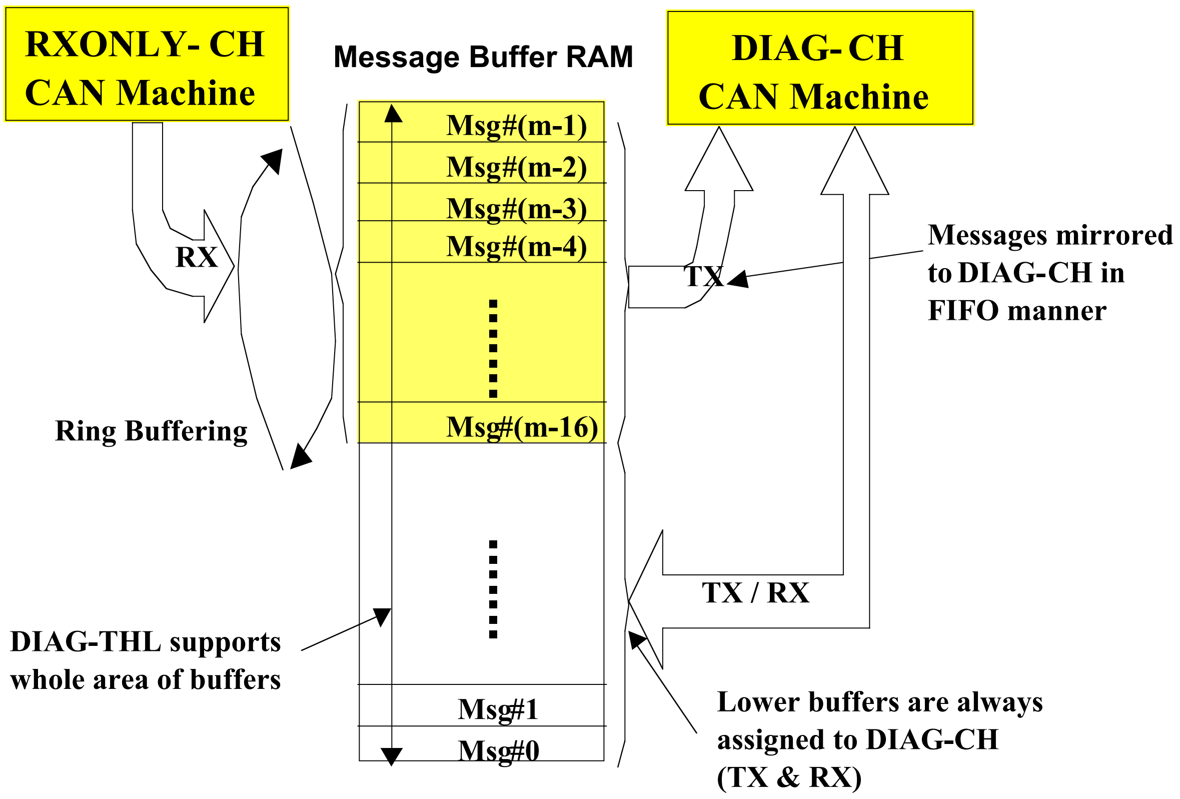
The name 'Mirror Mode' for the transfer of messages between a monitored bus and the diagnosis CAN-bus was chosen because any valid message exchanged on the monitored CAN-bus is submitted to the diagnosis system. Thus, the external analysis equipment will obtain a true image of the data exchange. Normally, only those messages received by the respective CAN-channel as pre-defined by its configuration for the normal operation are transferred.

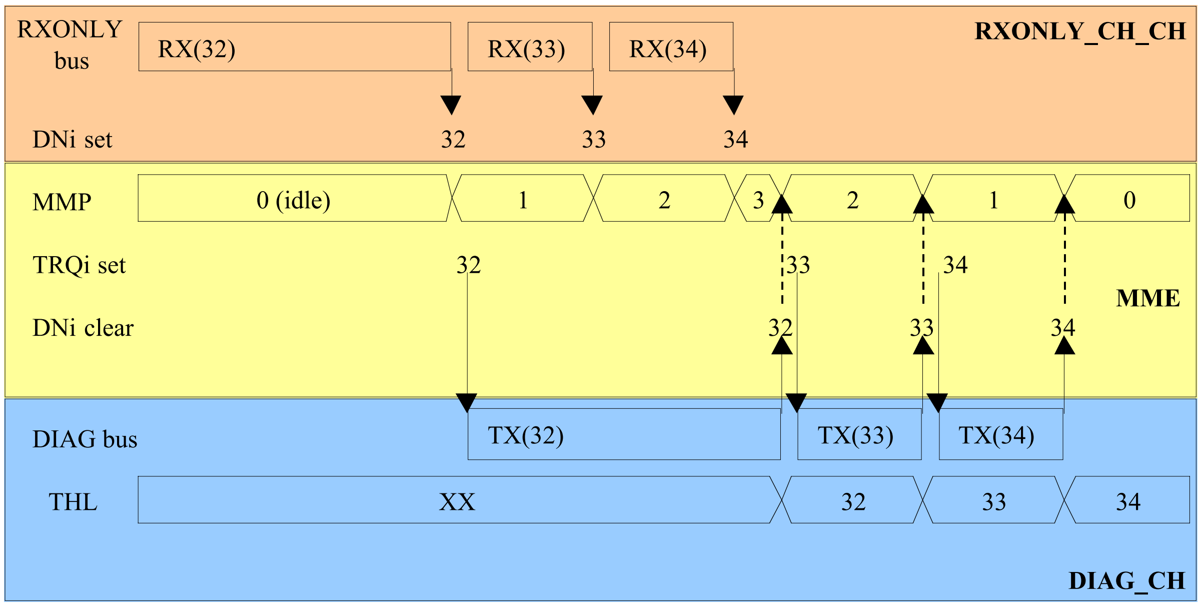
Once the Mirror Mode is started, it operates independently of the host processor. On the one hand the latency of monitored messages is predictable then, and on the other hand the diagnosis session does not impose any real time infringement to the gateway processor - two items, which can be of decisive matter when trouble shooting an unexpected behaviour.

Implementation Details

The DAFCAN provides 48 message buffers. When no diagnosis session is in place, these message objects can be used like in a regular AFCAN macro.

During Mirror mode the upper 16 buffer in the message buffer area are assigned to the listen-only channel that is connected to the monitored CAN-bus. The figure below sketches the kind of operations applied for each part of the message buffer RAM. The 'regular' CAN-channel (DIAG-CH) performs its reception operations only on the lower 32 buffers. However its transmit operations are still executed through out





Mirror Mode without Application Communication on DIAG_CH

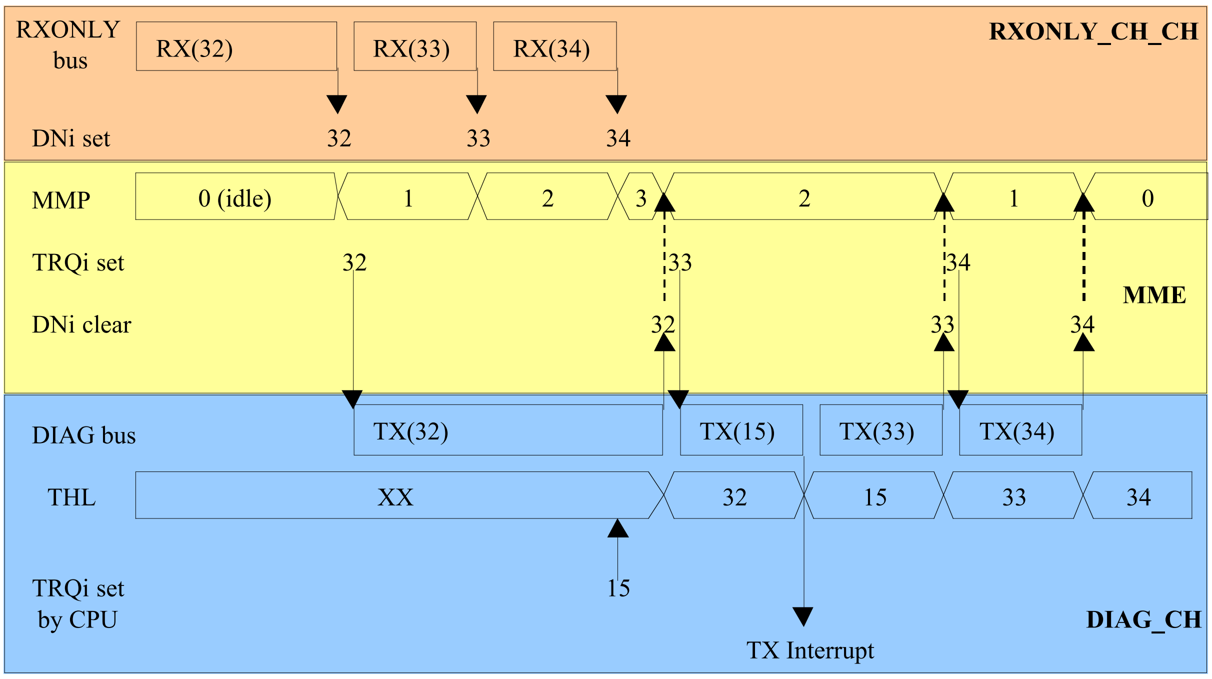
the complete CAN RAM. Thus any newly provided transmission request for any of the upper 16 message objects is captured automatically and respectively sent on the diagnosis bus.

The RXONLY-channel stores any valid message into the upper 16 message buffers starting at buffer #32 and incrementing the buffer number for each new reception. There is no masking applied at acceptance filtering. Neither there is any set-up for identifiers in the message buffers necessary. After reception of a message from the monitored channel, a dedicated state machine, the mirror mode engine (MME), automatically sets the transmission request (TRQ) if no other TRQ is pending in the upper 16 buffers.

The sequence of received message from the monitored CAN-bus is maintained on the diagnosis bus by this algorithm. The figure above illustrates an example where three messages are received on the RXONLY-channel (top bar). The MME (middle bar) recognises each of them incrementing an internal counter (MMP). However the transmission request (TRQi) for the particular message buffer i is not set before the previous message was sent (DNi = 0). Thus the messages can not overtake each other. The latency of the first message received on the RXONLY-channel is minimised. The DIAG-channel instantaneously transmits the monitored message.

Further the Mirror Mode provides enough depth of its storage capabilities when long messages are followed by short messages. This even masters situations when the diagnosis bus is busy transferring messages of other nodes or the gateway itself. Although this may be not a kind of operation of choice during a diagnosis session, the Mirror Mode can tolerate to loose arbitration on the diagnosis bus for a few messages without loosing a monitored message.

The timing diagram on the next page illustrates a communication of the gateway processor while at the same time a diagnosis session is active. It demonstrates that even during diagnosis session normal application messages can be sent or received. The example assumes that the identifiers match the message buffer number. The TX-message of the host processor (TX(15)) wins the internal arbitration against the monitored message in buffer RX(33). The transmission of monitored messages is put on hold at that time and resumes after the ‘application’ message was successfully transmitted.



Mirror Mode with interleaved Application Communication on DIAG_CH

Summary

The AFCAN provides a CAN interface to be used on any future device of NEC. The AFCAN eliminates any adaptations of driver software by the customer when changing the processor platform. Thus it meets requirements by the newly emerged HIS-standardisation efforts as its best. With its scalable message buffer memory the AFCAN covers low-, middle-, and high-end applications on all CPU-cores.

The DAFCAN offers an enhanced support for the demand of diagnosis in multi-channel environments. The patented Mirror Mode provides predictable latency times of monitored messages without causing any real time changes on the gateway node.

Wiewesiek, Wolfgang

NEC Electronics Europe GmbH

Kanzlerstrasse 2

Phone:++ 49 211 6503-566

Fax: -6566

E-mail: wiewesiekw@ee.nec.de

Website: www.ee.nec.de