CAN Connected To FlexRay

Dipl.-Ing.(FH) Marc Steuerer
Prof. Dr.-Ing. Alfred Höß

Fachhochschule Amberg-Weiden
Kaiser-Wilhelm-Ring 23
92224 Amberg
m.steuerer@fh-amberg-weiden.de

Michael Wächter
Berthold Ernstberger

Siemens VDO
Osterhofer Straße 10
93055 Regensburg

Abstract

The number of electronic control units in an automobile is increasing permanently, thus the data traffic is increasing as well and large amounts of data have to be exchanged between ECUs (electronic control unit) and other components in a vehicle. At present, most automobile manufactures are using the CAN bus system, but the performance of this bus systems has reached its limits.

The bottom line is that a new bus system with higher performance has to be established. Therefore, the FlexRay (FR) bus was developed. It represents a deterministic, fault-tolerant and high-speed bus system. But most electronic parts in a vehicle do not support this new FlexRay bus system. In a first step, the FlexRay bus system can be used as a backbone for data transfer. So it is necessary to convert CAN data into the FlexRay format and vice versa.

Among others, the German research project AUTOSAFE is concerned with this topic. The project aims at the further improvement of road traffic safety. For this purpose it is essential to transmit a lot of time and safety critical data within the system. The FlexRay bus is predestinated for a system with such requirements.

This paper describes the basic idea of the conversion process which transfers the data from one bus protocol into the other. This procedure occurs in a Gateway which contains one FlexRay and three high-speed CAN ports.

1 Introduction

Before starting the discussion on data conversion, it is recommended to consider some essential items. These are the experimental setup and a rough overview about the differences between the FlexRay and the CAN protocol.

The test setup of the system is shown in figure 1. As mentioned we are using a Gateway for the conversion of the CAN and FlexRay data. This Gateway brings three CAN channels together to one FlexRay channel. Main parts of the Gateway are a HC12 processor with three included CAN busses and the MFR4200 FlexRay controller.

The CAN and the FlexRay nodes are simulated by three standard PCs. The FlexRay PC contains a FlexRay card with two channels. The first CAN PC contains a CAN card with two channels and the other an USB CAN module with one channel.

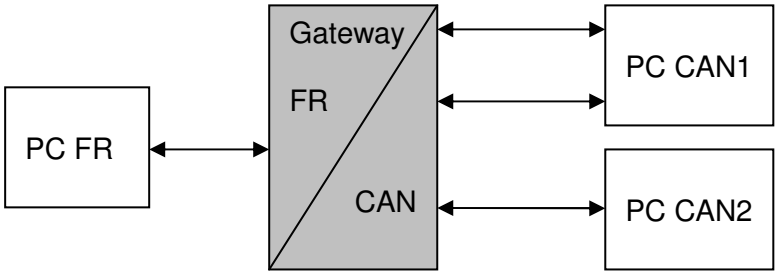

Figure 1: Experimental setup

The next paragraphs give an overview about the FlexRay and the CAN protocol. Figure 2 shows a simplified standard CAN message. A detailed description of the CAN protocol can be found in [1]. To illustrate the principle of converting messages the paper considers only the standard CAN messages.

Figure 2: Standard CAN Message

The important parts of this CAN message are the identifier (ID), the control field (CTRL) and the data field (DATA). The control field contains among other information the data length code and the remote transmission request bit. The other fields in the frame are: start of frame (SOF), cyclic redundancy check code (CRC), acknowledge bit (ACK) and the end of frame (EOF).

The FlexRay frame is quite different to the CAN frame and a little bit more complex. Figure 3 shows a simplified FlexRay frame. More information about the FlexRay specification is given in [2]. Important for the application are the identifier (ID), the payload length (LEN) and the payload segment (DATA). The payload segment is comparable with the data field of the CAN message. Further each cycle contains a counter (CNT) which indicates the current cycle.

Figure 3: Simplified FlexRay Message

Unlike the CAN frame the FlexRay SOF contains some more information (see [2]). A FlexRay cycle is composed of a defined number of such FlexRay frames. This cycle again is divided in a static segment, dynamic segment, a symbol window and a network idle time (NIT). The FlexRay frames in the static and the dynamic segment respectively are also known as slots. This leads to the structure shown in figure 4. A brief overview about the FlexRay bus is given in [3].

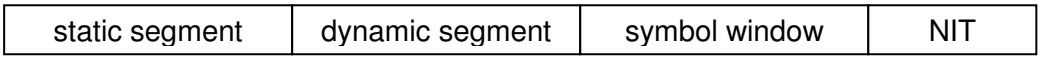

Figure 4: FlexRay communication cycle

2 Data Conversion

This section is concerned with the technique of data conversion. The most important question is; How can we put the CAN message in a FlexRay slot? The solution in this matter is to put all relevant information of the CAN message into the payload segment of the FlexRay frame. The result of this idea is shown in figure 5. It illustrates a FlexRay payload segment which contains all necessary CAN information. This segment is built of six cells.

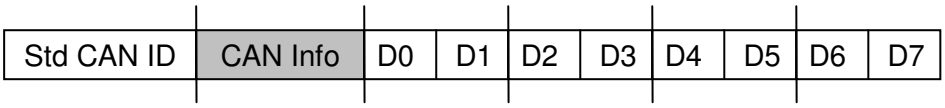

Figure 5: CAN message in a FlexRay payload segment

Each single cell of the FlexRay payload segment has a length of two bytes. That leads to a payload segment with a length of twelve bytes (six words corresponding to six cells). The standard CAN identifier goes into the first cell of the FlexRay payload segment. If an extended CAN message is desired, another cell has to be added. CAN Info structure contains special CAN information, see figure 6. In the last four cells the actual CAN data are placed, denominated as D0 to D7.

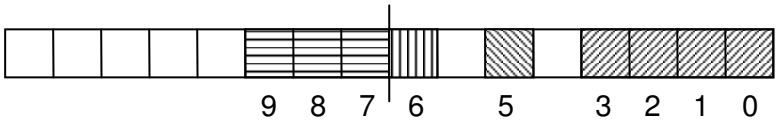

Figure 6: CAN info structure

Like every other cell the CAN info field has a length of two byte. The subsequent list shows the meaning of each bit in the CAN info structure.

- Bit 0 – 3: DLC, length of the CAN message.
- Bit 5: RTR, remote transmission request.
- Bit 7: EXT, does the message have an extended identifier?
- Bit 7 – 9: The incoming/outgoing Gateway CAN port.

The length of the FlexRay payload segment is therewith defined. Next desired parameters are the numbers of send and receive slots of the FlexRay cycle. With these three parameters we are able to appoint the cycle length. This length again is important to calculate the numbers of possible CAN messages within a FlexRay cycle.

The storage of the used FlexRay hardware is delimited. It is just possible to store the data of at most 59 slots. Two of these slots are already reserved for the synchronisation of the FlexRay clusters. Hence, it is mandatory to think about the number of CAN messages which need to be sent and received.

In our case we defined 45 send and 10 receive slots. Such asymmetrical distribution is feasible as we have more CAN data to send as to receive. The definition of send and receipt is dependent on the point of view. The definition used in this paper shows figure 7.

Altogether we need 55 FlexRay slots with six words payload each and two slots for the synchronisation. Therewith we are able to calculate the cycle time. But this calculation is more complex because the user has to mind some other settings. The detailed calculation of the configuration parameters is listed in [2]. The bottom line is that we have a cycle time of 2000µs.

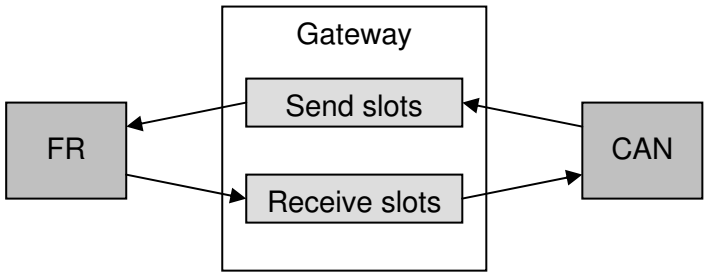

Figure 7: Slots consideration

As the FlexRay cycle now is defined, we will consider the CAN messages. Important information is the transmission time of the CAN messages. The shorter this time the more CAN messages can be transmitted in a defined time span. Time span in our application is the length of a FlexRay cycle. So it is necessary to calculate how many CAN messages can be transmitted in this time.

As bit duration we took 2µs, because we used a high-speed CAN with 500kBit/second. As average message duration we assumed the following values.

- CAN message with one data byte => 54 Bit => 108 µs
- CAN message with four data bytes => 78 Bit => 156 µs
- CAN message with eight data bytes => 120 Bit => 240 µs

For the calculation of the transferable CAN messages we consider just three different cases. These three different cases are defined as shown in table 1. The direction declares whether the Gateway receives (RX) or transmits (TX) the CAN message (see figure 7).

Cases	**Direction**	**Data length**	d_{rcv}	d_{trs}
Case 1	CAN TX CAN RX	8 bytes 8 bytes	240µs	240µs
Case 2	CAN TX CAN RX	4 bytes 4 bytes	156µs	156µs
Case 3	CAN TX CAN RX	8 bytes 1 bytes	108µs	240µs

Table 1: Transmission cases

The computation of the maximum numbers of CAN messages (n_{rcv}) that can be received at the Gateway and the transmit duration (d_{can}) happens with the following equations. It uses the parameters FlexRay cycle time (t_{FR}), the number of receive slots (S_{RX}) and number of transmit slots (S_{TX}), the number of CAN ports (n_{ports}), the received (d_{rcv}) as well as transmitted (d_{trs}) CAN messages.

$$d_{can} = S_{RX} \cdot d_{trs}$$

$$n_{rcv} = \frac{t_{FR}}{d_{rcv}} \cdot n_{ports}$$

The calculation of the three cases leads to the results shown in table 2. The grey columns show the results of the formulas above. Normally, the busload never rises above around 60%. Anyway, in these calculations underlie a busload of 100%. Result d_{can} rises in case one and case three above the given FlexRay cycle length. Also n_{rcv} is in case three higher than the allowed value. A 60% busload fulfils the desired requirements.

Case	t_{FR}	S_{RX}	d_{can}	S_{TX}	n_{rcv}
1	2000µs	10	2400µs	45	25
2	2000µs	10	1560µs	45	39
3	2000µs	10	2400µs	45	56

Table 2: Transmission calculation

It has to be kept in mind that these calculations do not include the operating time for the conversion of the data packets. On this account the conversion algorithm has to be very fast. The system should not waste time for polling the CAN and FlexRay ports. Therefore each incoming CAN and FlexRay channel at the Gateway is using an own interrupt service routine (ISR) for the interception and the conversion of the messages. Figure 8 gives a simplified overview about the conversion processes in the Gateway.

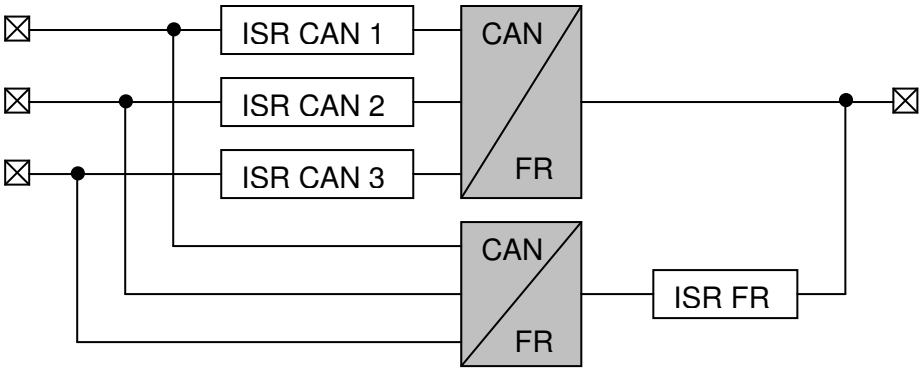

Figure 8: Conversion process

To minimise the conversion time, the Gateway works just with the effectively needed data. The system copies just the required data, e.g. if a CAN message contains four data bytes the system copies just these four bytes. Each copy process needs a lot of time, therefore it is essential to reduce the amount of copy procedures to a minimum. In the next section this coherence is obvious.

3 Operating time consideration

For the evaluation of the system it is necessary to consider the operating time the data need for their way through the Gateway. Therefore, it is needful to calculate the expected operating time and compare this with the measured values.

The calculation of these values happened already in the section above. For the measurement of the operating time we used a digital output pin of the HC12 processor in the Gateway and an oscilloscope. In this manner it is possible to measure the operation time exactly and the processor is not additionally stressed.

The regarded values are the conversion times of the data from CAN into FlexRay as well as from FlexRay into CAN. Before we take a look at the conversion it has to be noted that the conversion time is build of the time the processor needs to read the data from the hardware buffer, to convert the data and to write the data back into the hardware buffer. Most time is needed to write the data to the hardware buffer or read from it. In comparison to this, the actual conversion process takes just marginal time.

First considered value is the CAN to FlexRay conversion time (t_{cov}). The measure points are set at the CAN input and the FlexRay output (see figure 8). So we can measure the time a CAN message need from the input to the output. Table 3 shows the time in relation to the CAN data length.

Data length	t_{cov}
8 bytes	73µs
4 bytes	65µs
1 byte	60µs

Table 3: CAN to FlexRay

Next interesting value is the elapsed time for the conversion from FlexRay to CAN. The measurement points are set at the FlexRay input and at one CAN Output. The time to convert and transfer the data is the same for all three CAN ports.

In this case it does not really matter whether we send eight bytes or one byte. The length of the FlexRay frame is the same in all cases. Just the converting time varies a bit. But the differences are marginal. So the result is just one value which is the average of all possibilities – 78µs.

The meaning of these results needs to be interpreted. First, the measured values have to be compared with the calculated ones. Since we have three CAN ports it is necessary to calculate the conversion time (t_{cov}) for the usage with all three ports. Table 4 combines all values and shows their comparison. It would be optimal, if the value $t_{cov} \cdot n_{ports}$ is always lower than the time (d_{rcv}) a message needs to transfer over the bus.

Message	d_{rcv}	t_{cov}	$t_{cov} \cdot n_{ports}$
8 bytes CAN->FR	240µs	73µs	219µs
4 bytes CAN->FR	156µs	65µs	195µs
1 byte CAN->FR	108µs	60µs	180µs
FR->CAN		78µs	234µs

Table 4: Comparison

For the evaluation we have to regard two different cases which result from table 4. First case $t_{cov} \cdot n_{ports} \leq d_{rcv}$: We do not get problems with the conversion and the transfer of the CAN data. It takes longer to transmit these messages over the bus than to convert. Therewith the conversion algorithm and the processor are fast enough.

Second case $t_{cov} \cdot n_{ports} > d_{rcv}$: We have to keep in mind that this case will lead to problems for very high busloads. Normally, the busload does not exceed around 60%. This problem is illustrated in figure 9.

The scenario shall be explained in more detail by following example: We use a CAN message with a length of four bytes. For the calculation of the busload (B) in percent the following term can be used:

$$B = \frac{n_{bits}}{s \cdot T} \cdot 100$$

Parameters in this formula are the number of bits (n_{bits}), the speed of the CAN bus (s) and the periodic time (T) of the transmission. In the example we assumed a busload of 65% and that leads to a periodic time (T) of 240µs. As a result we have $T > t_{cov} \cdot n_{ports}$, from which follows that conversion algorithm and processor are fast enough (see figure 9).

Figure 9 shows the load on the three CAN busses and an example of the converting process inside the Gateway. The order of the conversion in figure 9 is by pure chance. It can be concluded that there remains an idle time after the conversion of all three CAN busses. The idle time is $t_{cov} \cdot n_{ports} - T$, in our case it is about 45µs. The same calculation can be applied for all other kinds of CAN message lengths.

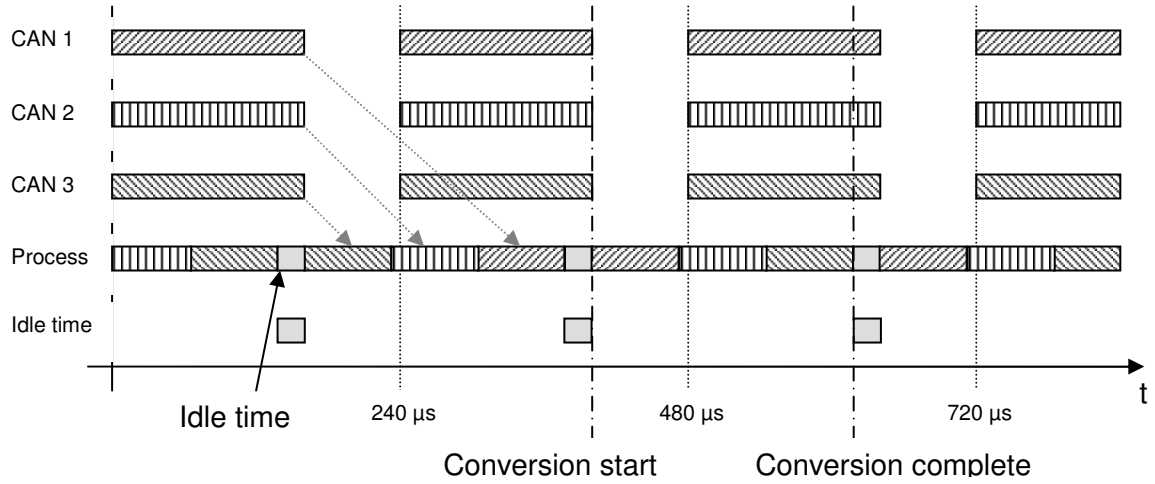

Figure 9: Example

We do not consider the way of the data from FlexRay to CAN because the results are comparable to the conversion of an eight byte CAN message (see table 4). So, the results are almost the same.

4 Conclusion

This paper introduced a basic data conversion concept having regard to the time critical transmission of the data. In a safety critical system it is mandatory that no data get lost to exclude serious consequences.

As discussed in the sections above it is possible to convert the CAN data into FlexRay data without losing any information, assuming the bus load on the CAN bus system does not exceed a certain limit.

The introduced system in this paper contains three high-speed CAN bus ports and a single FlexRay channel. With this system it is possible to convert messages if the busload on each

CAN channel does not exceed around 65%. The FlexRay bus speed does not matter in this consideration, because it is fast enough to transfer the desired CAN messages.

This busload limit varies if the number of CAN channels or the CAN bus speeds change. Further, the conversion time is heavily dependent on the used hardware (FlexRay and CAN controller as well as the microcontroller). Another important point discussed in this paper is the dependency of the FlexRay cycle length from the desired data to transmit. The longer the FlexRay cycle the more data can be transmitted.

The usage of a processor with a higher performance then the HC12 will optimise the conversion process significant. Further the integrated CAN controllers contain three transmit and five receive buffers for CAN messages. Also another FlexRay controller can be used. The MFR4200 contains just 59 message buffers each with up to 32 bytes of data. For this application the hardware configuration is satisfactory. But for more complex applications (e.g. more CAN ports or more FlexRay channels) hardware with higher performance is required.

5 Acknowledgment

This work was supported by the German Research Ministry (BMBF). The authors want to thank the Nanoelectronic unit of the BMBF for granting AUTOSAFE under funding no. 01M3076.

6 References

[1] Robert Bosch GmbH, Technical Report, CAN Specification, 1991

[2] FlexRay Consortium, Technical Report, FlexRay Communications System Protocol Specification, 2005

[3] Dr. Christopher Temple, Conference, Protocol Overview, 2003

[4] Marc Steuerer, Unpublished, Daten-Konvertierung im FlexXCon, 2006

[5] Freescale, Manual, FlexRay Communication Controllers

[6] Freescale, Manual, MC9S12DT256 Device User Guide