Acta Technica *62* (2017), No. 6B, 389–408

Study on the system of real-time and fault tolerance of steering by wire electric car flexray

Zhuan You[1]

Abstract. With the development of automotive electronics, more and more people use the wire control technology to overcome the defects of the traditional mechanical equipment and the hydraulic control system and to improve the handling stability, safety and comfort of the vehicles. FlexRay bus is a new advanced communication technology that aims to control the vehicle speed, and it has already been used in the concept vehicle. In order to further explore the potential of FlexRay when applied in the automotive industry, in this paper, we focus on the FlexRay communication module of the Steer-by-wire (SBW) System, and analyze and improve the complex part or deficiency of the protocol. Finally, we make the module's communication function be well enough to ensure the real time and fault tolerance function of the SBW system. Through the monitoring of communication controller and bus monitor scheduling, we could make sure that the node only get access to the bus at the correct point, thus to provide theoretical basis of achieving bus monitoring capacity for the FlexRay.

Key words. FlexRay, Steer-by-wire, Optimal scheduling, Clock synchronization, Bus monitor, Fault tolerance

.

1. Introduction

Automotive steering system is the key factor to improve the handling stability of the vehicle and to reduce the driver's controlling load and to improve the performance of the Driver-vehicle Closed-loop System. According to the steering power source, the steering system can be divided into mechanical steering system and power steering system [1]. In mechanical steering system, we take the driver's physical power as power source. Besides, we use mechanical transmission components, but due to the small output of the steering torque, the scope of application should be

[1]Department of Automobile Engineering, Huaian College of Information Technology, Huaian, Jiangsu 223003, China

limited. Power steering system, based on the mechanical steering system, it added a steering power assisting device. Therefore, it takes the driver physical strength and the power of the motor as steering power source. According to the different ways of providing power, we could divide the systems into Hydraulic Power Steering, namely HPS, Electronically Hydraulic Power Steering System, namely EHPS, Electric Power Steering, namely EPS and Steer-by-Wire, namely SBW.

The fundamental difference among Steer by wire and other steering systems lies in that it cancelled the mechanical or hydraulic connecting device between the steering wheel and the turning wheel, thus it breaks through fixed transmission ratio limit of the meshing gear, and reduces the total clearance among the inertial, friction and transmission parts of the system, and finally improves the response speed and response accuracy of the system.

Domestic research mainly focuses on algorithms related to Steer-by-Wire system, such as the algorithms of return torque and stability of the steering wheel. Domestic research doesn't form in-depth research of the communication protocol used in the system. And, FlexRay is through accurate and high speed communication network protocol to meet the requirements of SBW system when there are fault tolerance and time uncertainty of the message transmission, thus to improve the controllability, stability and safety of car handling. Therefore, the research of the SBW system's communication network is a research in a new field, and it is also crucial to improve the vehicle performance and gradually makes it more intelligent.

2. Improvement of FlexRay clock synchronization algorithm

FlexRay adopts the distributed clock synchronization mechanism. The clock synchronization of the normal nodes in the network is less affected by the error nodes (including synchronous nodes and non synchronous nodes). Therefore there is robustness in the system. The host of which the error nodes rest could base on the value of results of clock synchronization mechanism, namely, *zSyncCalcResult*status, to transfer the nodes from POC: normal active state to POC: normal active state or POC: halt state. And the detailed methods could is shown in Table 1.

Table 1. Error handling of clock synchronization algorithm

zSyncCalcResult	function	Deal with the time	POCstate
WITHIN_BOUNDS	There is no error in clock synchronization algorithm	odd-period	Maintain or come into POC: normal active state
MISSING_TERM	phase correction value or frequency correction value has not been calculated out	odd-period	greater than or equal to *zSyncCalcResult* = MISSING_TERM of odd-periodic*: gMaxWithoutClockCorrection- Fatal*, thus into POC: halt state; greater than *gMaxWithoutClockCorrectionPassive* and at the same time less than *zSyncCalcResult* = MISSING_TERM of odd-periodic*: gMaxWithoutClockCorrectionFatal*, thus into or maintain POC: normal passive state; In other cases, maintain POC: normal active state(1<= *gMaxWithoutClockCorrectio- nPassive*<= *gMaxWithoutClockCorrectionFatal*<= 15)
EXCEEDS_BOUNDS	phase correction value and frequency correction value are greater than limited value	Odd/even periods	Into POC: halt state

2.1. Phase deviation state diagram

In the description of clock synchronization state diagram, we usually adopt the transmission delay description method[9], namely use the transmission delay of the time stamp between two nodes to form the edge of the clock synchronization state diagram. However, with the development of modern network technology and the expansion of the scope of the clock synchronization, the information transmission appears to be sudden and intermittent, so it is difficult for us to accurately estimate the time stamp of the transmission delay. Due to the fact that the clock drift rate is more stable than the transmission delay of time stamp, we take the phase deviation as the basis of generating state diagram. The application of continuous time interval is more accurate than real-time interval, and it could conduct real-time and dynamic evaluation on the performance of clock synchronization. Besides, the application of continuous time interval could determine the system stability by exchanging the information of phase deviation state diagram, so as to improve the accuracy of clock synchronization in the network.

As shown in Figure 1, we assume that that the nodes *i, j, k, l* are the synchronization nodes in the FlexRay network. We establish the phase offset trend equation $y = P \times x + C$ by the least square method. In this equation, x stands for phase deviation, weight P stands for change rate of x, namely the node's clock drift rate.

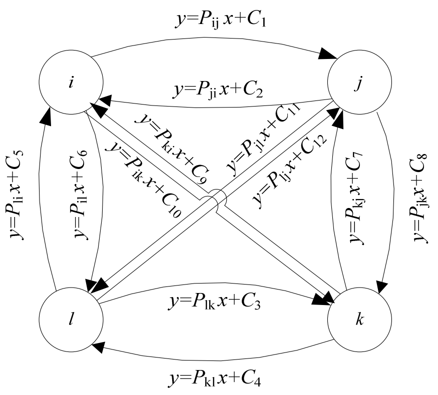

Fig. 1. Phase deviation state diagram

If the weights between synchronization nodes i and j meet the equation $P_{ij} = -P_{ji}$ (the weight P_{ij} is the j clock drift rate when i is taken as reference node), namely $|P_{ij}| = |P_{ji}|$, which means the clock drift rate of node i equals to that of node j, and the clock synchronization of the two nodes are stable, and through continuous clock correction the phase deviation can be synchronously reduced. Therefore, when there is no synchronization node error in the network, the whole phase deviation state diagram will be stable. If there is an exception in the clock synchronization nodes, then the weight values in the figure that take the node as the starting point and the end point will be changed.

2.2. The dynamic detection method synchronous node

There are at least three synchronization nodes in the FlexRay network, and each synchronization node sends or receives the synchronization frame in static segment (namely the ST segment). Under normal circumstances, if there is a synchronization node error, we should use the method of dealing with non synchronous node to deal with it, namely through judging the value and count value of the error cycle of zSyncCalcResul to make the node from being POC: normal active state to the POC: passive state or POC: halt state. If the synchronization node enters the normal POC: normal state, then it is forbidden to send the synchronization frame, but it still could receive the frame; if the synchronization node enters the POC: halt state, then we should re initialize the node. Although the synchronization error could not paralyze the clock synchronization of the entire network, it still affects the accuracy of the clock synchronization. In order to reduce this kind of influence, by exchanging the information of the phase deviation state diagram, we should conduct active detection towards the synchronous nodes and evaluate the stability of the network, thus to improving the accuracy of the clock synchronization.

Take FlexRay network in Figure 1 as an example, and table 2 shows the synchronous frame transmission time slot and the content in the load period.

Table 2. Transmission time slot and content of the synchronization frame

NODE	i	j	k	l
Synchronous frame transmission time slot	slot1	slot2	slot3	slot4
Transmitted content of the synchronization frame	Pji, Pki, Pli	P_{ij}, P_{kj}, P_{lj}	P_{ik}, P_{jk}, P_{lk}	P_{il}, P_{jl}, P_{kl}
Received content of the synchronization frame	P_{ij}, P_{ik}, P_{il}	P_{ji}, P_{jk}, P_{jl}	P_{ki}, P_{kj}, P_{kl}	P_{li}, P_{lj}, P_{lk}

For synchronous node i, after received the synchronous frames sent by other nodes, it should read related phase deviation's state diagram weights P_{ij}, P_{ik}, P_{il} from load segments, and then compare with weights Pji, Pki, Pli obtained from this node to detect the stability of the FlexRay network. If all synchronization nodes have been in a synchronous correction state, then we can take the weights P'_{ji}, P'_{ki}, P'_{li} calculated out in pre cycle to detect the clock synchronization performance of this node i.

$$\begin{aligned} Error1 &= |P_{ji}-(-P_{ij})| + |P_{ki}-(-P_{ik})| + |P_{li}-(-P_{il})| \\ &= |P_{ji}+P_{ij}| + |P_{ki}+P_{ik}| + |P_{li}+P_{il}|\,, \end{aligned}$$

is used to evaluate the clock synchronization state between node i and other nodes, and this equation is a general measure of the network stability. If the error is equal to 0, which signifies that the synchronization clock correction between the node i and other nodes maintains a synchronized stable state; if the error is greater than *limitpassive*1, which signifies that node i deviates from the clock synchronization range of the network, then the node should enter the POC: normal passive state; if the error is greater than *limithalt*1, which signifies that the normal clock synchronization between node i and other synchronization nodes can't be maintained, thus we should initialize the node. Since the number of MT in the communication cycle is constant, that is to say FlexRay could achieve synchronization when the clock synchronization layer is the maximum one. Therefore, so when the clock deviation maintains a MT, the node must enter the POC: halt state. Taking into account that *Error*1 is obtained from the absolute value, then we can define the limit values in the conditions that the nodes enter the POC: halt state and POC: normal passive state respectively as

$$limithalt1 = 2 \times pMicroPerMacNom\,, \tag{1}$$

$$limitpassive1 = 0.5 \times limithalt1\,, \tag{2}$$

From the two equations, we can know that *pMicroPerMacNom* is the number of μT in one MT. The new frequency correction value could be obtained from the FTM algorithm, namely,

$$dynratecor1 = FTM(P_{ji}, P_{ki}, P_{li})\,. \tag{3}$$

Pji, Pki, Pli are the phase deviation change rates when we take jkl as the reference nodes. Due to the face that these weight values are based on the phase deviation change rates of multiple cycles, so *dynratecor*1 can more accurately present the clock drift rate of the local nodes in successive time intervals; and the correctness of the correction of the original frequency correction values, which are based on the measured values even/odd doubly periods, is less accurate than *dynratecor*1.

$Error2 = |P_{ji}-P'_{ji}| + |P_{ki}-P'_{ki}| + |P_{li}-P'_{li}|$ is used to detect the clock synchronization performance of node i itself. Besides, it is more often used to conduct nodes' self detection, and it must satisfy the premise that the network is stable, that is to say *Error*1=0. If the error is equal to 0, which indicates that the clock synchronization performance of node i could maintain a normal state; if the node is greater than *limitpassive*2, which signifies that the clock runs abnormally, thus the node should enter the POC: normal passive state; if the node is greater than *limithalt*2, which indicates that the clock synchronization performance of node i declined obviously, in this condition we should initialize the node. Among them, *limitpassive*2 and *limithalt*2 are two threshold values of *Error*2, and their definition method is similar to *Error*1, namely

$$limithalt2 = 2 \times pMicroPerMacNom\,. \tag{4}$$

$$limitpassive2 = 0.5 \times limithalt2\,. \tag{5}$$

The new frequency correction value is

$$dynratecor2 = FTM(P_{ji}, P_{ki}, P_{li})\,. \tag{6}$$

With the increase of the number of phase deviation, the clock drift rate and synchronization status information reflected by weight value P will be more accurate. Due to the fact that the dynamic detection method of synchronous node is based on a large number of data acquisitions, the error or loss of individual data will not affect the evaluation and detection results.

2.3. Simulation and performance comparison of the software

Assume that there are four nodes in a FlexRay network, and three of which are synchronized nodes. The cold boot process should be ignored while conduct the software simulation process. Besides, we should focus on the clock synchronization conditions of Cycle8∼Cycle13–these six cycles, and each cycle should be simplified as ST segment and NIT segment. We should set the relevant protocol parameters meet the following equations gdCycle = 1000, gdMacrotick = 1 μ s, pMacroPerCycle = 1000, gdStaticSlot = 100MT.

In order to calculate the phase and frequency correction value of each node, we should write the FTM algorithm Matlab as: [offsetcorrection, ratecorrection] = FTM_algorithm (num, offsetarray, ownoffset). In the above equation, num is the number of synchronization nodes, for cSyncNodeMax = 15, the maximum value of num of is 15; offset- array is the expected arrival time of certain synchronization frame, and it takes the node's local time as reference. Through FTM_algorithm function, we could obtain phase and frequency correction value of each node among Cycle10∼Cycle11 as shown in table 3. In table 3, *pdMicrotick* is the length value of μT, and its unit is μs; and we take μT as unit of phase and frequency correction value.

Table 3. Phase and frequency correction value among Cycle10∼Cycle11

nodes	*pdMicrotick*(μs)	Phase correction value(μT)	Frequency correction value(μT)
Non synchronous node	0.1	-820	-43
synchronization node 1	0.025	429	28
synchronization node 2	0.025	-29	-3
synchronization node 3	0.025	-406	-27

In order to improve the synchronization accuracy of the network, based on the original clock synchronization algorithm of FlexRay, we introduced the dynamic detection method of synchronous nodes, namely work on the phase deviation diagram based on the measured value of Cycle8∼Cycle9. For the synchronous node 1, we should establish $Dev_2[1][56]$ and $Dev_3[1][56]$ to store the phase deviations obtained from different synchronization frames. The storage method of other synchronous nodes is similar to that of synchronization node 1, thus there is no need for us to repeat them. Take $Dev_{FrameID}[1][56]$ into account to establish the equation of phase deviation trend $y{=}P \times x{+}C$, and the phase deviation state diagram is shown in Figure 2.

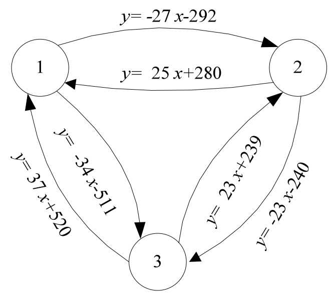

Fig. 2. Phase deviation state diagram of synchronous node

In Figure 2, if $P_{ij} = -P_{ji}$, which indicates that the clock synchronization between the synchronization node i and the synchronization node j is stably maintained. If $P_{ij} \neq -P_{ji}$, which indicates that total phase correction between the synchronization node i and the synchronization node j has not been achieved. It can be seen from the figure that the weight value that takes synchronization node 1 as the beginning point and the terminal point does not satisfy the equation $P_{ij} = -P_{ji}$, which means the synchronous between this node and other nodes could not be achieved. Besides, the clock synchronization between the synchronization node 2 and the synchronization node 3 is in steady state. Therefore, the synchronization node 1 is the node that has a poor performance in the current network, which will affect the synchronization precision of the network, so we should adjust the frequency correction value of this node.

According to the dynamic detection method steps of synchronous node, we should first determine the values of *limitpassive*1, *limithalt*1, *limitpa- ssive*2 and *limithalt*2. And based on the protocol parameters *gdCycle* $= 1000\mu$s, *pSamplesPerMicrotick* $= 2$, *gdSampleClockPeriod* $= 0.0125\mu$s, *pMicroPerMacroNom* $= 40$, *pdMicrotick* $= 0.025\mu$s, *pMicroPerCycle* $= 40000$, *gdMacrotick* $= 1\mu$s and *pMacroPerCycle* $= 1000$, we could get the results that *limitpas- sive*1 $=$ *limitpassive*2 $= 40\mu$T, *limithalt*1 $=$ *limithalt*2 $= 80\mu$T. From Figure 2 we can see that the current network is not in a stable state. Therefore, we can only take *Error*1 into consideration, and do not need to calculate the *Error*2. The test results are as shown in table 4.

Table 4. Dynamic detection results of synchronous nodes in Cycle10∼Cycle11 1

reference node	The deviation of clock drift rate (μT)	*Error*1	The original frequency correction value(μT)	*dynratecor*1 (μT)
synchronization node 2	$\lvert P_{21} + P_{12}\rvert = 2$	5 <*limitpassive*1	28	31
synchronization node 3	$\lvert P_{31} + P_{13}\rvert = 3$			

From table 4 we can see that the new frequency correction value is 31μT, in Cycle12∼Cycle13 we take this value to correct the clock of synchronization node 1, while the clock correction value of other synchronization node remains unchanged, and then we work out state graph weight values of new phase deviation diagram, as Table 5 shows.

Table 5. The clock correction results obtained by introducing the dynamic detection method on Cycle12∼Cycle13

node	*pdMicrotick*(μs)	The deviation of clock drift rate (μT)	
Non synchronous node	0.1	/	
synchronization node 1	0.025	P_{21}	26
		P_{31}	36
synchronization node 2	0.025	P_{12}	-27
		P_{32}	23
synchronization node 3	0.025	P_{13}	-35
		P_{23}	-23
*Error*1	2 <*limitpassive*1		

From table 5 and table 4 we can see that, with the continuous detection and correction going on, *Error*1 gradually decreased. If the *Error*1 is close to 0 T, which would indicates that all the synchronization nodes are in the modified state. In order to make sure that the network state being stable, we should conduct dynamic assessment towards the clock performance of the node itself. Namely we should base on the value of *Error*2 to determine the new frequency correction value, and the calculation method is the same as that of *Error*1, thus we do not present it once again. When *Error*2 is 0, which shows that the clock performance of the node is normal, and under this condition, it will not affect the stability of the network. When the *Error*2 is not 0, it indicates that the clock drift rate of the node is abnormal, under this condition we need to adjust the clock correction value. If *Error*2 is modified, and *Error*1 returned to the state of not being 0, which would indicate that the former stability of the network is only a temporary phenomenon and the true synchronization state has not been achieved among the nodes.

Table 6 shows us the comparison of the phase deviations of non synchronous nodes before and after the improvement of the synchronization algorithm. In this table, for the fact that the Cycle8∼Cycle9 has not yet started the clock correction, the contrast values of the two cycles are the same. And the phase deviation of Cycle9 is relatively high. Starting with the Cycle10, we use dynamic detection method. With the clock synchronization performance of the three synchronization nodes being continuously improved, the phase deviation value of the non synchronous nodes with different synchronization reference gradually reduces. The above shows us that the local time of the node is closer and closer to the global time, so the clock correction value of the node itself gradually becomes smaller.

Table 6. Comparison of phase deviation values among non synchronous nodes(unit:μT)

period	synchronous node reference					
	synchronous node1		synchronous node2		synchronous node3	
	before improvement	after improvement	before improvement	after improvement	before improvement	after improvement
8	-200	-200	-130	-130	-70	-70
9	-820	-820	-560	-560	-280	-280
10	-169	-165	-121	-117	-118	-115
11	-118	-113	-107	-101	-104	-99
12	-109	-103	-100	-93	-98	-91
13	-106	-97	-92	-83	-86	-78

3. The software design of FlexRay bus monitor

3.1. Interface design of hardware of bus monitor

BG has its own independent clock synchronization mechanism. This mechanism monitors the TxEN CC (Data Enable Transmit) signals sent out by CC, and through the BGE (Guardian Enable Bus) [10] signal to control the data sent by BD. When CC chooses to send the data in the wrong time slot, BG will notify the host of the error message in an interruption form. Besides, it would disconnect the connection between the node and the network, that is to say, to prohibit the BD continuing to send out data. At the same time, CC would also monitor the BGE signal through BGSM (Guardian Schedule Monitoring Bus) [10]. If the time slot schedule BG obtained functions abnormally, CC will report the error message to the host in an interruption form. Thus, we could define the signal transmission among BG and CC and BD hosts.

BG and the host can be connected through the Peripheral Interface Serial, namely SPI, to achieve configuration function of the host towards BG and send error interruption signal to the host when there is timing error in BG and CC. SPI interface uses the master-slave mode, and this mode is for the synchronous serial data transmission between CPU and peripheral low speed devices. The clock is controlled by the main device. When there is clock shift pulse, the data is transmitted bite by bite. And the high ones are sent earlier, and the lower ones are sent later. The transmission rate of full duplex communication is faster than the I2C bus, reaching up to a several Mbps. According to the connection method between BG and the host and the signal transmission definition, we can get a list of BG hardware interface as shown in table 7.

Table 7. Definition of BG interface

BG name	connecting object	BG pin	Input / output	Description
BG (A)	BD (A)	RxD_A	Input	data received by BD (A)
		BGE_A	output	signals controlled by BD (A)(enable / disable)
	CC	$\overline{TxEN\_A}$	Input	control signal of CC towards BD (A)(enable / disable)
		TBGE_A	output	control signal of BG towards BD (A)(enable / disable)
	host	SCK	Input	clock signal
		SDI	Input	configuration information of BG (A)
		SDO	output	error interrupt information
		$\overline{SCSN}$	Input	enable signal of BG (A)
BG (B)	BD (B)	RxD_B	Input	the data received of BD (B)
		BGE_B	output	signals controlled by BD (B)(enable / disable)
	CC	$\overline{TxEN\_B}$	Input	control signal of CC towards BD (B)(enable / disable)
		TBGE_B	output	control signal of BG towards BD (B)(enable / disable)
	host	SCK	Input	clock signal
		SDI	Input	configuration information of BG (B)
		SDO	output	error interrupt information
		$\overline{SCSN}$	Input	enable signal of BG (B)

3.2. *Software functions of the bus monitor*

The host commands could be divided into two parts, of which one par is the commands sent by host to the CC, and the other part is the commands that the host uses to configure BG, which can be seen in table 8. The third column in the table is the commands sent to the CC, but before the being sent out, we should firstly inform BG of these commands, so that BG could adjust the monitoring strategy in a timely manner; the fourth column in the table is the configuration command of BG.

Table 8. Host command

host command (POC state)	CC response	BG response	
		CC command	BG configuration command
ALL SLOTS (normal active, normal passive)	At the end of the cycle, CC retreat from single slot mode	Change monitoring strategy, and allow CC to send data in all stipulated time slots	/(BG does not send data)
ALLOW_COLDSTART (except default config, config, halt)	Allows CC to perform a cold start	Change monitoring strategy, and permit the deliveries in CC that are consistent with cold start monitoring strategy	/(BG can't perform the cold start cluster)
CONFIG (default config, ready)	Allow the configuration of CC	/(unless BG need to follow the state of CC)	Allow the configuration of BG
CONFIG_COMPLETE (config)	CC enters POC: ready	/	BG enters POC: ready
DEFAULT_CONFIG (halt)	Allow CC to conduct the default configuration	/	Allow BG to conduct the default configuration
FREEZE (all the POC state)	CC enters POC: halt state	Change monitoring strategy, and prohibit all the deliveries from CC	BG enters POC: halt state
HALT (normal active, normal passive)	At the end of the cycle, CC enters POC: halt state	Change monitoring strategy, and prohibit all the deliveries from CC after the cycle is ended	At the end of the cycle, BG enters POC: halt state
READY (except default config, config, halt, ready)	CC enters POC: ready state	Change monitoring strategy, and prohibit all the deliveries from CC	BG enters POC: ready state
RUN (ready)	CC begin to start cluster communication	Change monitoring strategy, and monitor the boot process of CC	BG start to join the cluster, but no cluster communication is allowed
WAKEUP (ready)	CC perform cluster awakening	Change monitoring strategy, and monitor the awakening process of CC	/(BG can't awaken the cluster)

4. The real-time and fault tolerance evaluation of steering system of FlexRay wire

4.1. The FlexRay communication module hardware

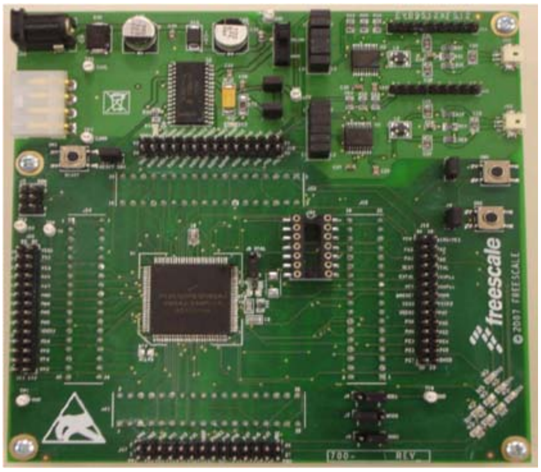

Fig. 3. EVB9S12XF512 evaluation board

In this paper, we choose the EVB9S12XF512 evaluation board of Freescale as basic ECU nodes, as shown in Figure 3. EVB9S12XF512 includes a 16-bit-microcontroller: MC9S12XF512 with FlexRay communication controller embedded in it, two FlexRay bus drivers: TJA1080 and a high speed CAN bus driver: MC33742.

(1) Micro controller MC9S12XF512

There is a XGATE co processor, 512K byte Flash and an embedded FlexRay communication controller in the micro controller MC9S12XF512. The package diagram of this micro controller is as shown in figure 4. In figure 4, XGATE is aimed to conduct fast interruption processing, thus it could reduce the load of CPU while conduct interrupt processing. Therefore, it provides a higher level of disruption, and through the sharing of some service procedures it could shorten the working time and process of CPU. XGATE, like CPU, is a programmable kernel that supports the C compiler. When the source is off, XGATE begins to run; when the interruption task is completed, it will stop all the clocks and wait for the next event, thus it could reduce power consumption. Therefore, although MC9S12XF512 is a 16 bit microcontroller, it can achieve the performance that of 32-bit microcontroller.

(2) The FlexRay bus driver

The FlexRay bus driver uses TJA1080 produced by NXP company, and it could provide data rate up to 10 Mbps. So it is suitable for working during the 14V∼42V voltage range. And the electromagnetic radiation of FlexRay bus driver is low, and the input differential mode is adopted in this bus driver. The IO terminal voltage of the bus driver can altered by itself accordingly, at the same time, there is BGE interface reserved for the bus monitor. The chip provides 4 kinds of bus states: low

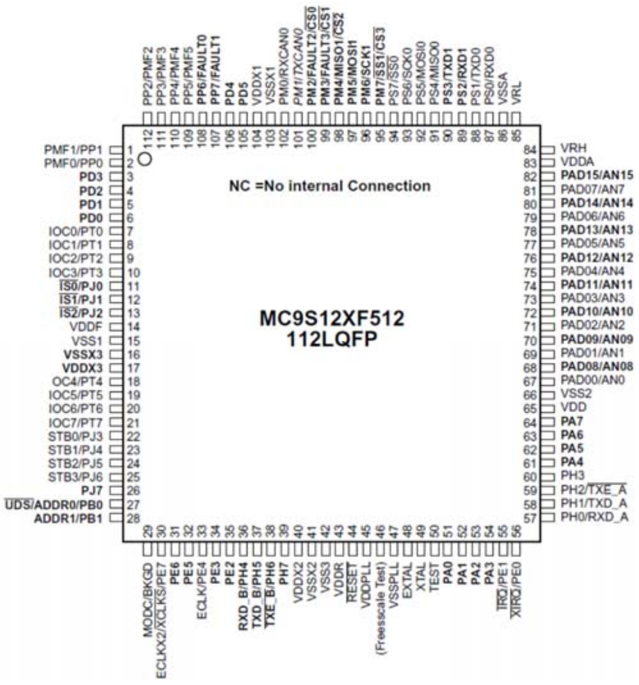

Fig. 4. The package diagram of the microcontroller MC9S12XF512

power state, idle state, data 1 state and data 0 state.

In order to achieve the voltage adaptive adjustment of the IO terminal, in the circuit, we connect drive voltage of the communication controller (namely the drive voltage of the microcontroller with the VIO pin of TJA1080. And then, no voltage conversion chip is needed to achieve the normal communication between TJA1080 and controller. The bus driver circuit that the channel B corresponds is similar to Figure 5, thus we left out this part.

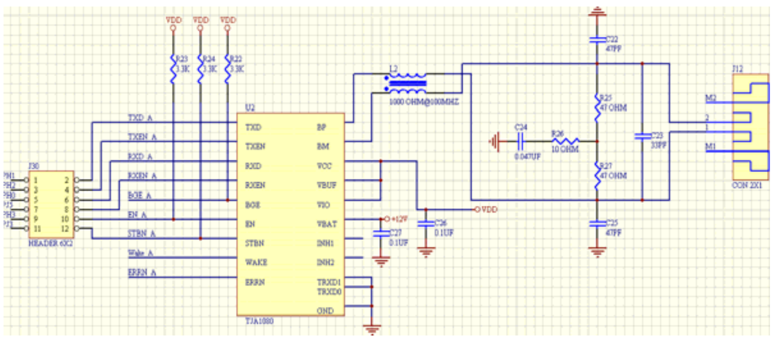

Fig. 5. FlexRay bus driver circuit diagram

4.2. Real Time and Fault Tolerance Evaluation

We integrate the steering, yaw warning and electronic stability control system to get the message scheduling scheme shown in table 6-5. The ST segment transmits a ST message containing a variable steering drive ratio and a steering wheel torque, and at the same time it starts and synchronizes this message. Due to the fact that the message in DYN segment is triggered by the event, so when the sensors are in normal state, this segment only receives yaw warning and information related to steering in electronic stability control system; if a sensor failure happens, the correct variable steering transmission ratio or steering wheel aligning torque value will be sent in the previous dynamic slots (here the "correct" value could be obtained from the sending node based on the processing of the previous fault-tolerant calculation. Because that this aspect has nothing to do with FlexRay communication, not many researches are conducted) to correct the error value sent by ST segment in the cycle. In order to ensure the real-time performance of the system, in each cycle we should pre-use the fault-tolerance algorithm to calculate the variable steering drive ratio / steering wheel torque value as backup-information, when there is a fault in the sensor, the ST segment could directly send this value.

Table 9. Message scheduling scheme

NODE		n1	n2
ST message	send	Start the synchronous frame (the load data could be transferred into variable steering ratio	Start the synchronous frame (the load data is the steering wheel torque)
DYN message	send	the adjusted variable steering ratio	the adjusted steering wheel torque
	receive	yaw warning and information related to steering in electronic stability control system(sent by yaw warning and the nodes in electronic stability control system)	

From the analysis of reference [11] and reference [12], we can know that the ideal transmission ratio is a two digits that is bigger than 0, and steering Chinese driver's preferred wheel torque range is 1.75∼4N.m. After comprehensively considering the influence of data accuracy on automobile safety and the general format of the automobile signals, we set the length of the load segment in ST message to be 16×16bits, that is to say the parameter $gPayloadLengthStatic = 16$.

DYN message includes the correct variable steering drive ratio / steering wheel torque value, yaw warning and electronic stability control system information related to steering, and its definition is as shown in table 10.

Table 10. DYN message definition

DYN message	Message state	sensor State	numerical value	significance
the modified variable ratio steering / steering wheel torque value(16×16 bits)	send	fault	obtained according to the fault tolerance algorithm	indicate rotation angle of the steering motor or self-aligning torque motor
yaw warning / information related to steering in electronic stability control system(16×16bits)	receive	normal / fault	obtained according to sensor detection value and logic algorithm	make the steering system slightly generate correction steering force or provide relevant information for the steering torque algorithm

Based on the analysis of ST message and DYN message, we could conduct the configuration of the system parameters, as shown in table 11. Here, we set the data rate allocation of the FlexRay wire steering system to be 10Mbps, communication cycle length to be 5ms, and static slot to be slot1∼ slot66. When the maximum value of the minimum time slot (minislot, shortened as (MS)) 22, the maximum value of dynamic slot is 22, and take into account the former static slot, we could get the starting number of dynamic slot to be slot67 with the maximum to be slot88.

Table 11. Configuration parameter table

system parameters		parameters in ST and DYN segments	
data rate	10Mbps	static slot number	66
the maximum value of simultaneous node	15	static slot length	50MT
the maximum length value of clock tick	1μs	AP offset	3MT
cycle length	5000MT	load length of ST segment	16*16bit
length of ST segment	3300MT	the maximum value of MS	22
length of DYN segment	880MT	length of MS	40MT
symbol window length	15MT	AP offset of MS	3MT
NIT length	805MT	the maximum value of the load length in DYN segment	16*16bit
start point of phase correction	4229MT	note: MTis macrotick for short, namely its unit is	
digit number sent by TSS	11	the biggest clock tick.	

Due to the fact that the message sent by ST segment is not much, thus we need not to adjust the node-sending sequence or increase the number of static time slots, and we could simply select the maximum value of message length to be the best length value of static time slot. Considering the fact that if the time slots of sending the synchronous frames are too concentrated, it would affects the clock synchronization precision, so we choose to send ST message containing variable steering drive ratio and steering wheel torque in slot1 and slot4.

Before we arrange the scheduling of DYN segment, we should first conduct schedulability analysis on the systematic parameters in table 12. For the received DYN message is completed with FIFO disruption, we should pre-determine the transmitting slots before making DYN scheduling table, and then arrange the received time slots behind the sent slots. And we should notice that the schedulable analysis does not include message receiving. For the convenience of calculation, we uniform the parameter units for MS (according to Table 10, the length of a MS is *gdMS* = 40MT = 40μs), and we should simplify the cycles as ST segment and DYN segment, and the length of each segment is shown as the following.

$$STs = 3300MT \text{div } gdMS = 3300MT \text{div } 40MT = 82.5MS$$

$$Tbus = \quad Tbus = STs + DYNs = 82.5MS + 22MS = 104.5MS$$

Table 12. DYN message and parameters in DYN segment

NODE	DYN message parameters			parameters in DYN segment		
	DYN message	*M*m (MS)	*LTx*m	*Tbus* (MS)	*STs* (MS)	*DYNs*(MS)
n1	DM1	2	14	104.5	82.5	22
n2	DM2	2	16			

In Table 12, the message length Mm could be from equations(9) and(10). The parameter values in the equations could be obtained from Table11 and the common protocol parameters, which include the action point phase value *gdActionPointOffset* = 3MT, the idle channel bounds digits number *cChannelIdleDelimiter* = 11*gdBit*, the maximum bite time *gdBitMax* = 0.10015μs, the minimum propagation delay *gdMinPropagationDelay* = 0μs, the maximum propagation delay *gdMaxPropagationDelay* = 2μs, the maximum length of clock tick *gdMacrotick* = 1μs, the maximum clock deviation *cClockDeviationMax* = 0.0015, transmission start sequence TSS, length of TSS *gdTSSTransmitter* = 11 *gdBit*, frame start sequence FSS, length of FSS *cdFSS* = 1*gdBit*, frame end sequence FES(Frame End Sequence), length of FES *cdFES* = 2 *gdBit*, dynamic trailing sequence DTS, length of DTS *gdDTS* = 4 *gdBit*.

$$\begin{aligned} M_m =& 2 \times gdActionPointOffset[MT]+ \\ & + round(((aFrameLength[gdBit] + cChannelDelimiter[gdBit]) \\ & \times gdBitMax[\mu s/gdBit] + gdMinPropagationDelay[\mu s] \\ & + gdMaxPropagationDelay[\mu s])/(gdMacrotick[\mu s/MT] \\ & \times (1 - cClockDeviationMax))) . \end{aligned} \tag{7}$$

$$\begin{aligned} aFrameLength[gdBit] =& gdTSSTransmitter[gdBit] + cdFSS[gdBit] \\ & + 80gdBit + aPayloadLength[2-byte-word] \times 20gdBit \\ & + cdFES[gdBit] + gdDTS[gdBit] . \end{aligned} \tag{8}$$

Among them, MT is the largest clock tick of one unit. According to Table 10, we could know 1MT $= 1\mu$s ; *gdBit* shows us that the unit is a bit of time, namely $1gdBit = 1/$ bus rate $= 1/10$M $= 100$ns$80gdBit$ in equation (9), corresponding to the first paragraph and the last paragraph of the message, includes 8 bytes. For the fact that while sending each certain bite, we need to add Byte Start Sequence(BSS), the equation is as the following

$$8 \times (8gdBit + cdBSS[gdBit]) = 8 \times (8gdBit + 2gdBit) = 80gdBit\,. \tag{9}$$

Load length value *aPayloadLength* is the word number of load segment, namely, the number of double byte word. Therefore, we need to multiply it with $20gdBit$ to get the digit number, here the 20gdBit could be obtained from equation(11).

$$2 \times (8gdBit + cdBSS[gdBit]) = 2 \times (8gdBit + 2gdBit) = 20gdBit\,. \tag{10}$$

Because both the load segment of DM1 and DM2 is 16×16bit, namely the word number $aPayloadLength = 16$, the message length is $M_{Dm1,Dm2} = 2 \times 3 + round((418 \times 0.10015 + 2)\text{div } 0.9985) = 50MT \approx 2MS$.

*LTx*m is node parameter, and its value could be determined by the message length, MS number and data rate. Besides, we could allocate this value while initializing FlexRay module. For DM1, its data in load segment is the correct and variable steering ratio, and the message length is 2MS. For the number of MS in DYN segment is 22, DM1must be sent out in front of the MS whose serial number is 20(namely slot 86)Considering that, logically speaking, only after the driver completed steering operation, could he feel the corresponding steering wheel torque. Besides, the consequences of instructions that cause steering error or delays would be more serious. Therefore, we set the parameter value of *LTxm* in DM1 to be 14 (namely slot80), and the parameter value of *LTxm* in DM 2 to be 16 (namely slot82), thus set aside enough sending room.

The corresponding transfer decision-making process is as shown in Table 13, and the table corresponds with the dynamic programming process of two-dimensional array A[n][j]. Because the DYN message is little, the n in the table directly corresponds with message name. And j stands for the dynamic slot number 1 and 2 (i.e. slot67, slot68). Because we merely arrange two time slots of DYN message delivery, we would not consider the time slots after dynamic slot 2 (namely slot68).

Table 13. Optimized dispatching decision-making process of DYN segment(unit:MS)

n	j	
	1	2
DM1	120.5	119.5
DM2	241	241

As the numbers in the last line of A array values are the same, and there is no other information of message scheduling to be used as the basis of making trade-off choices, we choose the priority solution of dynamic message shown in Table 13. In

Table 13, the meaning of n and j is the same with that of them in table 12. And Mdlm is the deadline, Rtm is the worst-case response time, and diffm is difference between the above two, namely $diff_m = Mdl_m - Rt_m$. All DYN message should complete the information transmission at the critical value point, thus while setting Mdlm, we take the service quality 11.10 (higher than the critical value of Tp's 11) as benchmarks of Tpc, namely Tp_c= $13.6ms = (13.6 \times 1000)\mu s\text{div } gdMS = 340MS$. Then we divide Tpc and Tbus exactly, and the left-out part be made up by the corresponding value of LTxm, as shown in equaiton (11). Because the complementary value is determined by the message parameter, the deadline reflects the importance of the message.

$$Mdl_m = round(Tp_c\text{div } Tbus) \times (Tbus + LTx_m \times gdMS)\,. \tag{11}$$

Now, the dead line of DM1 is Mdl_{Dm1}= $round(340\text{div } 104.5) \times (104.5 + 14) = 355.5MS$. The worst corresponding is the value of the ID frame value of dynamic time slot 2 (dynamic time slot 1 send DM2) in the second dispatching decision-making phase, namely $Rt_{Dm1} = F_{Dm1} + S_{Dm1} + M_{Dm1} = 20 + 96.5 + 2 = 118.5MS$. In the same way, the deadline of DM2 is Mdl_{Dm2}=361.5MS, and the worst corresponding time $Rt_{Dm1} = 120.5MS$.

Table 14. Priority solution of DYN message(unit:MS)

n	j	Mdlm	Rtm	$diff$m
DM1	2	355.5	118.5	237
DM2		361.5	120.5	241

$diff_{Dm2} \geq diff_{Dm1}$ indicates that the worst-case response time of DM1 is close to the deadline. In order to guarantee the schedulability of the system, we should first send DM1, and assign the smaller frame ID value to FrameIDDM1. Thus we can get the time slot arrangement in table 15, namely the corresponding dynamic DM1 timeslot 1 (namely slot67), and the corresponding dynamic DM2 2 time slot (namely slot68). And then we receive information related to steering of the yaw warning and electronic stability control systems in the dynamic slot 3 and slot 4 (i.e. slot69, slot70). Due to the fact that the message sending timing is associated with the scheduling of the node, and the two nodes belong not to wire-control steering network, so we don't compare the arrangement orders while receiving time slots.

In table 15, the message sending of ST segment is achieved according to the schedule table, while DYN segment obtains the present message scheduling by dynamic programming algorithm. Therefore, we can adjust the arrangement according to actual condition. In this table, SM stands for the Static message, namely ST message; The abbreviation of Dynamic message is DM, namely DYN message; chAB means sending or receiving data with double channels, and chA means sending or receiving data only with channel A; n3 and n4 are respectively the nodes in yaw warning system and electronic stability control system. Due to the fact that the microcontroller MC9S12XF512 does not support the dual channel communication of DYN segment, so we only use channel A to send and receive message.

Table 15. Time triggered schedule

Time slot	MB delivery	MB receive	FIFO receive
	STsegment		
slot1	n1, chAB, SM1(start and synchronous frame, and the load data is the variable steering ratio)	n2, chAB	
slot4	n2, chAB, SM2(start and synchronous frame, and the load data is the steering wheel torque)	n1, chAB	
	DYN segment		
slot67	n1, chA, DM1(the adjusted variable steering ratio)		n2, chA
slot68	n2, chA, DM2(the adjusted steering wheel torque)		n1, chA
slot69	n3, chA, DM3(information of yaw warning system)		n1 & n2, chA
slot70	n4, chA, DM4(information of electronic stability control system)		n1 & n2, chA

5. Conclusion

In this paper, we mainly study the FlexRay communication module of wire steering system, and make the communication function of the module could ensure the real-time and fault tolerance of the steer-by-wire system, so as to meet the safety requirements of steer-by-wire design in vehicles.

In terms of scheduling, we out forward two different optimization schemes. Because that, we used scheduling algorithm in the FlexRay static segment, the optimization scheme of this segment is on offline to determine optimal static parameter value and node transmission sequence, in this way to minimize the total response time of all static message. Dynamic segment of FlexRay is based on FTDMA, and this media's access mechanism has time triggering characteristic and is confirmable at the same time. Besides, this media contains the event triggering flexibility. Therefore, to optimize the dispatch of this segment, we must carry on the schedulability analysis based on the worst-case response time of the message to make the optimal scheduling process always meet the prerequisite of system scheduling. And we adopt the optimized scheduling algorithm based on dynamic programming, that is to say, through multi-stage decisions to gradually find the optimal scheduling dynamic segment arrangement. At the same time, we summarize priority solution of the dynamic message.

In the aspect of clock synchronization, we introduce the concept of phase deviation state diagram. We replace the real time intervals with continuous time intervals, and based on the more accurate clock discrepancy information to improve the network synchronization accuracy; and according to the weight changes of the state diagram, and conduct real-time and dynamic evaluation of the synchronization clock performance of the synchronization nodes and the stability of the network to make each node's local time be closer to the global time.

Acknowledgement

This work was supported by Senior Visiting Engineers Plan Funded Project of Higher Vocational Institutions in Jiangsu Province(2015FG007).

References

[1] Hanzlik: (2006) *A Case Study of Clock Synchronization in FlexRay.* Technical Report in Real-time Systems Group, May 2006.

[2] Paul Milbredt, Martin Horauer and Konrad Reif: (2007) *Wakeup and Startup of Flexray Cross Linkings.* ATZelektronik worldwide Edition, 2007(4):30-32.

[3] P. M. Szecowka, M. A. Swiderski: (2007) *On Hardware Implementation of Flexray Bus Guardian Module.* International Conference on Mixed Design of Integrated Circuits and Systems 2007 (MIXDES '07), June 2007: 309-312.

[4] Juan R. Pimentel: (2004) *An Architecture for a Safety-Critical Steer-by-Wire System*[R]. 2004 Society of Automotive Engineers, 2004-01-0714.

[5] Joachim Langenwalter, Tom Erkkinen: (2004) *Embedded Steer-by-Wire System Development.* Embedded World, February 2004:17-19.

[6] Paul Yih, J. Christian Gerdes.: *Steer-by-Wire for Vehicle State Estimation and Control.* (AVEC), (2004).

[7] Thomas F, Florian H, Robert H, et al.: (2003) *Flexray-the communication system for future control systems in vehicles.* Tech. Report Paper, SAE World Congress, 3: 1-110.

[8] Claesson V, Ekelin C, SuriN: (2003) *The event-triggered and time-triggered medium- access methods.* Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03), 2003:131-134.

[9] Doo-Hyun Kim, Eun Hwan Jo, Moon Hae Kim: (2004) *Time-triggered and message-triggered object architecture for distributed real-time multimedia services.* PCM, 2004:330-337.

[10] Roman Obermaisser, Philipp Peti, Fulvio Tagliabo: (2007) *An integreated architecture for future car generations.* Real-Time Syst, 2007:1-30.

Received May 7, 2017