INTELLIGENT AUTOMATION & SOFT COMPUTING, 2017
http://dx.doi.org/10.1080/10798587.2017.1302712

Automatic FIBEX Generation for Migration from CAN Message Description Format to Flexray Fibex Format

Young Hun Song[a], Suk Lee[a], Kyoung Nam Ha[b] and Kyung Chang Lee[c]

[a]School of Mechanical Engineering, Pusan National University, Busan, Korea; [b]Marine Robot Center, Korea Institute of Industrial Technology, Busan, Korea; [c]Department of Control & Instrumentation Engineering, Pukyong National University, Busan, Korea

ABSTRACT

Recently, FlexRay was developed to replace the controller area network (CAN) protocol in the chassis network systems to provide high-speed data transmission as well as hardware redundancy for safety. However, FlexRay network design is more complicated than with CAN protocol, which has been an in-vehicle network (IVN) standard for car manufacturers for decades, because the FlexRay has many parameters such as the base cycle or slot lengths. To simplify the FlexRay network design and assist vehicle network designers in configuring a FlexRay network, this paper presents an automatic field bus exchange format (FIBEX) generation method for migration from the CAN message description format such as the DBC format to the FlexRay FIBEX format. The automatic FIBEX generation method is examined by simulating a chassis networking system using a DBC benchmark tool, which demonstrates the feasibility of the system and the reduction in workload for network designers.

KEYWORDS

In-Vehicle Networking (IVN) systems; FlexRay; CAN message format; Field bus exchange format (FIBEX); Automatic FIBEX generation algorithm

1. Introduction

Consumers are placing more demands on vehicle safety and convenience, and the implementation of technologies to meet these demands has led to a growing requirements for in-vehicle network (IVN) protocols that have high-data-rate and deterministic transmission characteristics (Kim, Lee, & Lee, 2015; Rosset, Souto, Portugal, & Vaspues, 2012). To meet this demand, the FlexRay protocol was developed in 2000 by a consortium including BMW, Daimler–Chrysler, Motorola, and Philips. It is now widely used for chassis network systems, including the braking, steering, and suspension systems. To provide well-characterized transmission delays as well as priority transmission, the FlexRay protocol uses both time-division multiple access (TDMA) and flexible TDMA (FTDMA). Furthermore, FlexRay can provide data bandwidths of up to 10 Mbps, and has seen uptake by many car manufacturers (Armengaud, Steininger, & Horauer, 2008; Han, Natale, Zeng, Liu, & Dou, 2013; Jang, Park, Han, Lee, & Sunwoo, 2011; Kang, Park, & Jeong, 2013; Park & Sunwoo, 2011; Schmidt & Schmidt, 2009; Zeng, Natale, & Ghosal, 2011).

However, FlexRay network design is more complicated than the controller area network (CAN) protocol, which has been an IVN standard for car manufacturers for decades. The CAN DBC format is a standard CAN message description format (Liebezeit, Junghanns, Bonin & Serway, 2012; Vector, 2007) that provides clear definitions of various signals, including sensor and motor signals generated by electric control units (ECUs) (Cummings, 2008; Leen, Hefferman, & Dunne, 1999; Sethna, Stipidis, & Ali, 2006). Using CAN, car manufacturers are able to maintain compatibility of network designs, because DBC defines the message identification (ID), transmission period, transmit nodes, and receive nodes.

FlexRay uses the field bus exchange format (FIBEX) (Association for Standardization of Automation and Measuring Systems [ASAM], 2013; Stroop & Stolpe, 2006) to specify a platform configuration register (PCR). The PCR is a set of parameters that define a FlexRay network (or cluster), including the basic time unit, transmission cycle length and size, and number of TDMA slots. PCR includes 53 parameters that are closely interrelated, and must be carefully selected (Song, Lee, & Lee, 2013). FIBEX defines the way that a FlexRay network operates, i.e., which node uses which TDMA slot to send a given signal, and which node should listen to which slot in order to receive the signal. While a CAN network designer can simply focus on the behavior of each node, a FlexRay designer must consider the operation of the network in addition to the operation of the node. This requires an in-depth knowledge of the protocol, and has hindered widespread uptake of FlexRay. Therefore, a simplified migration technique from CAN to FlexRay is required (Mishra & Naik, 2005; Rai, Jestin, & Vitkin, 2008; Reiter, Viehl, Bringmann, & Rosenstiel, 2016; Rho, Azumi, Oyama, Sato, & Nishio, 2016; Zhao & Zhong, 2008).

In this paper, we describe an automatic FIBEX generation method for migration from the CAN DBC format to the FlexRay FIBEX format. We analyze the structure of FIBEX, DBC, and the 53 PCR parameters that must be defined in the FlexRay protocol specification. We then detail a DBC-based automatic FIBEX generation algorithm that automatically generates the PCR parameters from the DBC data. Finally, we implement the system and assess the validity of the generated FIBEX data using CAN DBC benchmarking files. The automatic FIBEX generation method not only simplifies the FlexRay network design, but also reduces working complexity in migrating from the CAN DBC format to the FlexRay FIBEX format.

CONTACT Kyung Chang Lee gclee@pknu.ac.kr

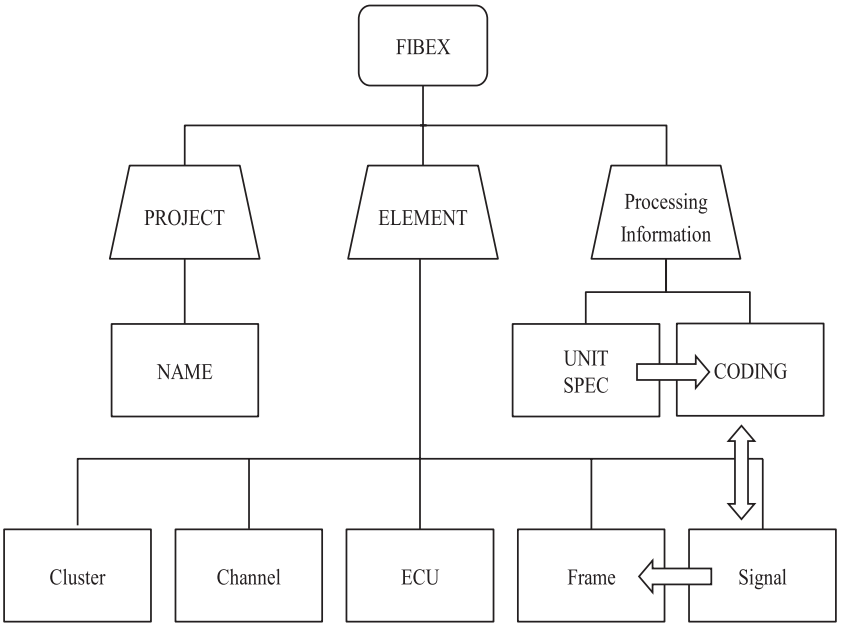


Figure 1. Structure of the FlexRay FIBEX File Format.

2. Structure of FIBEX and DBC

Figure 1 shows the structure of the FlexRay FIBEX file format. FIBEX was developed by the Association for Standardization of Automation and Measuring systems (ASAM) consortium (ASAM, 2013; Stroop et al, 2006) to facilitate exchange of data. FIBEX is used to define the FlexRay network design information, and is based on the extensible markup language (XML) (Choi et al, 2014). Furthermore, FIBEX can be expanded to support different network protocols. FIBEX uses a structure called the object model to represent and access information, and objects are classified into hierarchical groups. These structures are called trees, and the items included in the trees are called elements. In Figure 1, the designer requires a lot of information to generate a FIBEX data structure, including the cluster scheduling, transmission and reception node, and frame ID. However, the signal information within a frame is not required by the FIBEX standard.

Figure 2 shows the structure of the CAN DBC file format. The DBC file format, which is used for CAN message descriptions was developed by Vector Inc. (Liebezeit et al, 2012; Vector, 2007). CAN DBC is a vehicle standard database format developed for data exchange between vehicle ECUs. The physical CAN node can access set up information from DBC. In addition, DBC allows the designer to monitor and analyze a system using simulations, i.e., without requiring physical nodes. In Figure 2, some information, including the bit timing, nodes, and messages, must be defined for basic operation. The header defines the version, which may be left blank, as well as the symbol. The bit timing defines the CAN network bandwidth and bit timing register (BTR). The node defines the name of the nodes on the network. The message includes the ID, name, size, and signal information. The signal includes the name, start bit, size, storing method, and receive node.

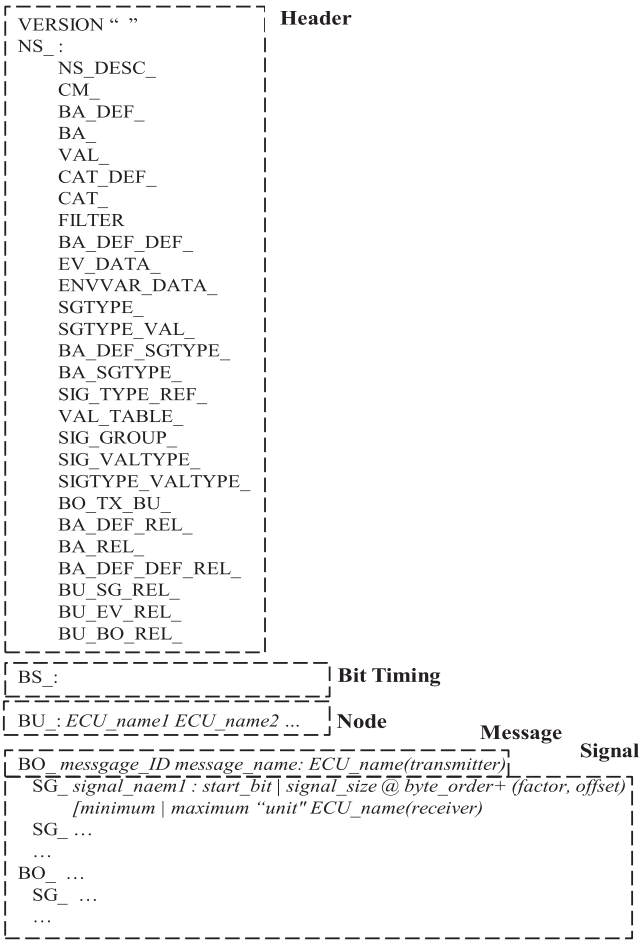


Figure 2. Structure of the CAN DBC File Format.

3. Analysis of PCR Parameters of FlexRay

To characterize the transmission delay in FlexRay, it is necessary to maintain clock synchronization between nodes. This is determined by the 53 PCR parameters, which include the cluster parameter and node parameter defined by the FlexRay standard. The network designer must understand the FlexRay protocol and configure the related parameters manually. However, this is not straightforward, and the technical issues create a barrier to upgrading the CAN networks that are widely used in existing IVNs.

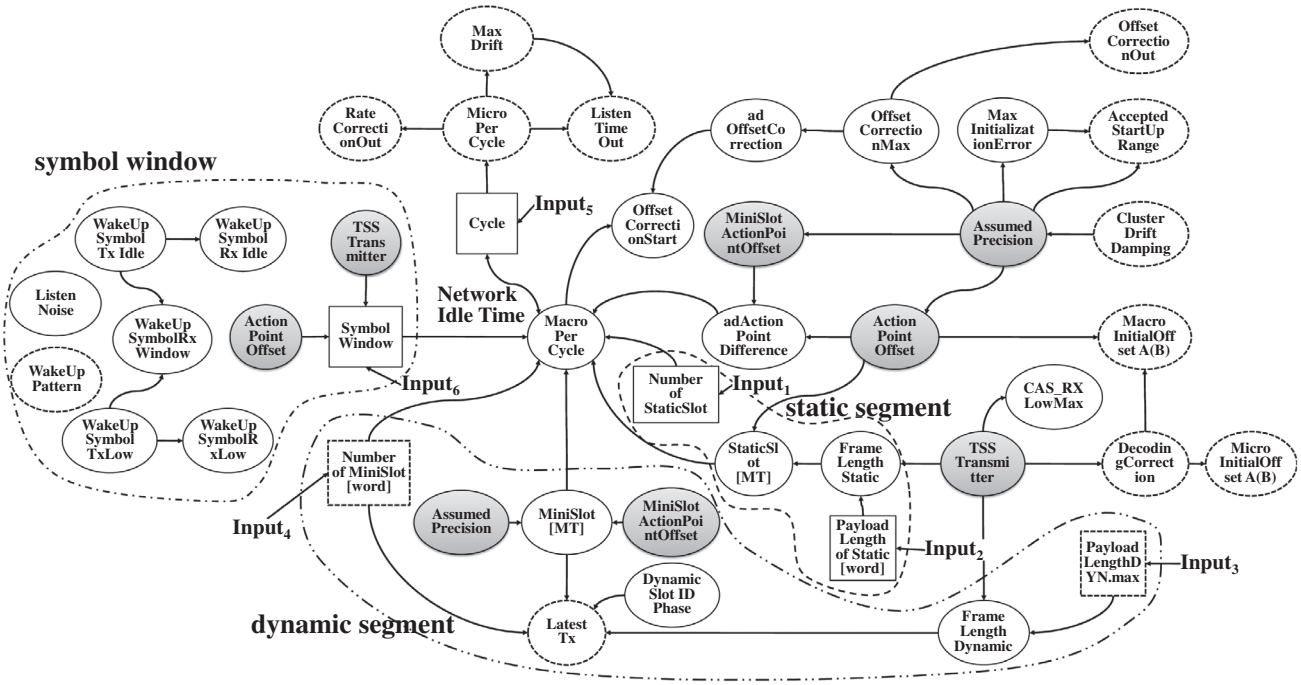


Figure 3. Schematic Diagram Showing the Platform Configuration Register.

Figure 3 shows a schematic diagram illustrating the relationship between the PCR parameters defined by the FlexRay protocol standard. These are divided into static segment parameters, dynamic segment parameters, symbol window parameters, and other parameters. In Figure 3, the parameters in the solid boxes are global cluster parameters, which have the same value as all nodes. The parameters in the dotted box are node parameters. The parameters in the circular box are calculated from a mathematical relationship, whereas those in the rectangular box are calculated and assigned by the network designer.

The communication cycle (Input5 in Figure 3) is determined by the network designer; however, it is affected by the "Macro Per Cycle", which determines the macro-tick in each cycle, and also by the "Micro Per Cycle" for selecting the "Rate Correction Offset", "Max Drift", and "Listen Time Out".

The four PCR parameters shown in the center of Figure 3, "Payload Length of Static", "Frame Length Static", "Static Slot [MT]", and "Number of Static Slot", configure the static segment. These four parameters are referred to here as the "Action Point Offset" and "TSS Transmitter" for calculating related parameters. The number of static slots (Input1) and length of the static slot (Input2) parameters are required from the designer, and the four cluster-configuration parameters must be specified in every node that is part of the cluster.

There are six parameters shown in the lower part of Figure 3 for configuring dynamic slots: "Payload Length DYN. Max", "Frame Length of Dynamic", "Latest Tx", "Dynamic Slot Idle Phase", "Number of Mini Slot", and "Mini Slot [MT]". As with the static slot, two parameters, "Assumed Precision" and "Mini Slot Action Point Offset", are used to calculate related parameters. The maximum dynamic slot data length (Input3) and number of mini-slots (Input4) are determined by the network designer. The three parameters shown in the solid box in Figure 3 are the cluster configuration parameters; the other parameters in the dotted box are the configuration parameters, which can be different in each node.

The symbol window, shown in the left of Figure 3, is composed of eight configuration parameters. The symbol window does not have any relationship with the configuration parameters. However, when "Symbol Window" is used for synchronization, the "Action Point Offset" and "TSS Transmitter" parameters are referenced. The size of the symbol window (Input6) must be provided by the designer; however, the other seven parameters are calculated using protocol-defined values.

All parameters that are not classified are used for clock synchronization, or calculated from other parameters. These configuration parameters are determined by the transmission width in the FlexRay protocol.

Table 1 lists the PCR parameters that are defined by the FlexRay protocol standard, but are not shown in Figure 3. These configuration parameters are determined by the designer, but are not related to the other parameters. In Table 1, "Max Without Clock Correction Passive", "Max Without Clock Correction Fatal", "Network Management Vector Length", "Cold Start Attempt", and "Sync Node Max" are cluster configuration parameters and must be set to the same value in every node in the cluster. Other node parameters can be set to different values. Every parameter in Table 1 must be defined. There are no effects in terms of communication or synchronization in the FlexRay protocol when we change one of these parameter values.

Table 1. Optional PCR Parameters of FlexRay.

Network management	Max Without Clock Correction Passive
	Max Without Clock Correction Fatal
	Network Management Vector Length
	Allow Halt due to Clock
	Allow Passive to Active
	Extern Offset Correction
	Extern Rate Correction
Protocol defined	Cold Start Attempt
	Sync Node Max
	Delay Compensation A(B)
User specified	Channel
	Wake Up Channel
	KeySlot Header CRC
	KeySlot ID
	KeySlot Used For Sync
	KeySlot Used For Startup
	SingleSlot Enabled

4. DBC-based Automatic FIBEX Generation Algorithm

The DBC-based automatic FIBEX generation algorithm is shown in Figure 4, and consists of three parts; a DBC parser, PCR generator, and FIBEX generator. When a CAN DBC file is imported, the DBC parser collects ECU data and message information. To collect ECU data, it searches for the ECU header 'BU_:' in the DBC file, and generates an ECU linked list for storing the ECU name, number of transmitted messages (Txmsg), and number of received messages (Rx msg). In addition, it searches for the message header 'BO_:', which defines a transmission address; if this is not found; the message is ignored to reduce the network traffic. If a transmission address is defined, the DBC parser generates a message linked list to save the CAN ID, data length, message generation cycle, and periodic features of the message (cyclic or aperiodic). If a message is transmitted from multiple ECUs, the DBC parser generates and stores multiple messages per ECU. It then checks for a signal header 'SG_:' to verify the receiving address and stores the received information as a linked list. After executing the DBC parser, we obtain a linked list, as shown in Figure 5. In this linked list, messages are allocated to the ECU using ECU and message information.

When the linked list is generated, the communication cycle of FlexRay (TC), is determined by calculating the greatest common divisor (GCD) of the cyclic message generation cycle of all messages in the PCR generation part. The time of the static slot (TSS) is calculated by multiplying the number of cyclic messages (NSS) by the maximum length of the cyclic messages (LSS). The time of the dynamic slot (TDS) is calculated by multiplying the number of aperiodic message (NDS) by the maximum length of those messages (LDS). The length of the symbol window (TSW) is fixed to a default value of 0, and the network idle time (TNIT) is calculated from the communication cycle, segment time (Tsegment), sum of TSS, TDS, and TSW. If TNIT < 0, the automatic generation algorithm is terminated, because FIBEX generation is not possible. If TNIT ≥ 0, optional parameters, including a wake-up pattern or channel selection, are generated. If necessary, it is possible to change the optional parameters manually (this is called expert mode).

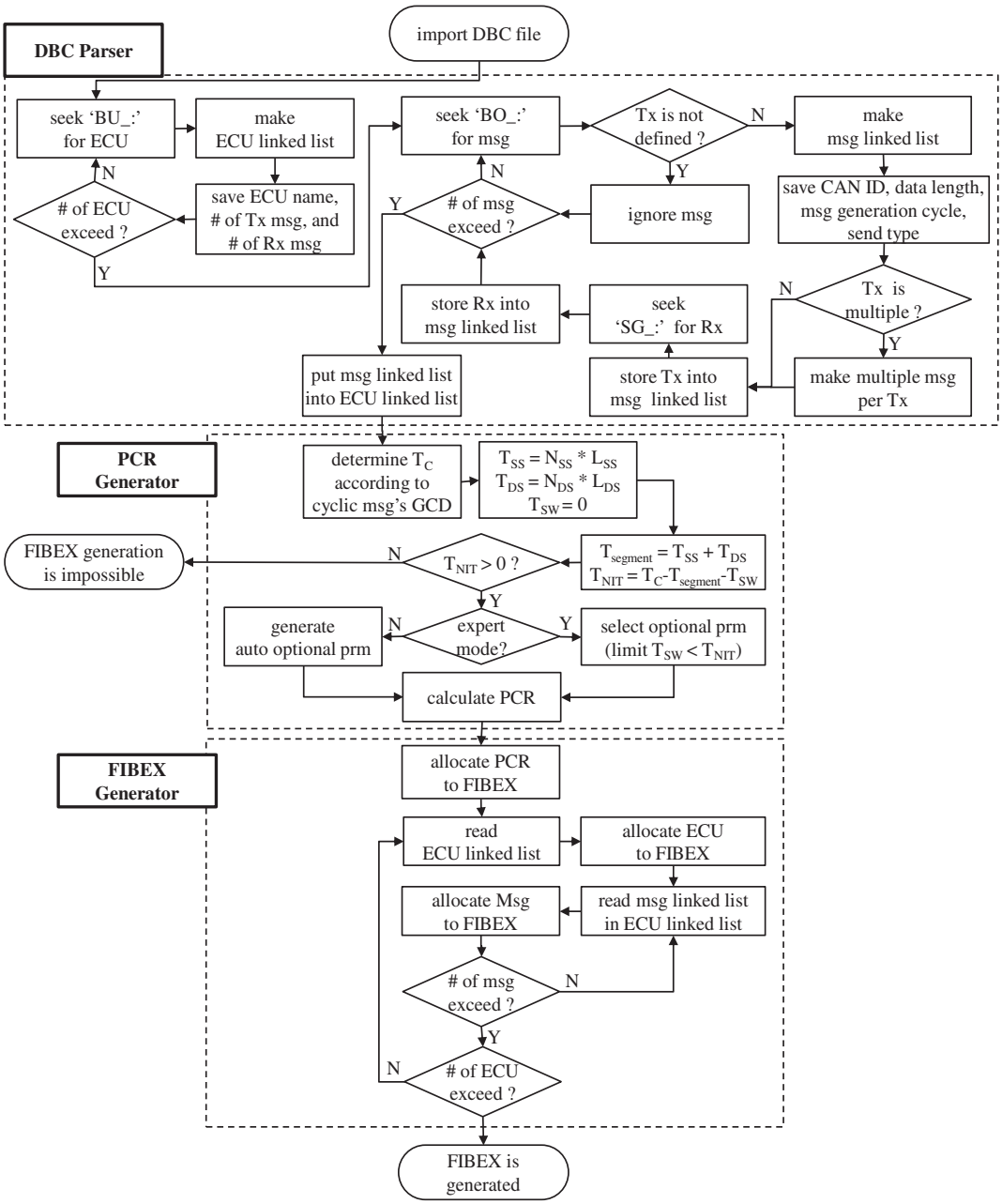


Figure 4. Flowchart for the DBC-based Automatic FIBEX Generation Algorithm.

Otherwise, the remaining PCR parameters are generated automatically from the relationships shown in Figure 3.

To generate a FIBEX file format, as shown in Figure 1, the FIBEX generation part makes objects for the PCR, allocates PCR values, and generates the cluster information describing the FlexRay network. The ECU and message information are read from the linked list, and then allocated to a FIBEX data structure.

5. Performance Evaluation of the DBC-based Automatic FIBEX Generation Algorithm

Figure 6 shows the DBC-based automatic FIBEX generation program, which was developed using Microsoft Visual Studio 2010 MFC. The program can show the imported DBC file formats and the generated PCR parameter for user inspection. The program consisted of nine steps. Firstly, a CAN DBC file is imported by clicking the 'CANdb read' button. (①). Then, the baud rate is fixed to 10 Mbps and the base cycle is automatically calculated using GCD values. (②). Also, the four parameters describing the number of static slots, number of mini-slots, static payload length, and maximum dynamic payload length are calculated based on the message information of the DBC file. (③). Secondly, the 'Segment check' button is pressed. (④). Then, additional data including the network macro per cycle, static slot length, mini-slot length, and network idle time are calculated, as well as the optional PCR parameter ('optional

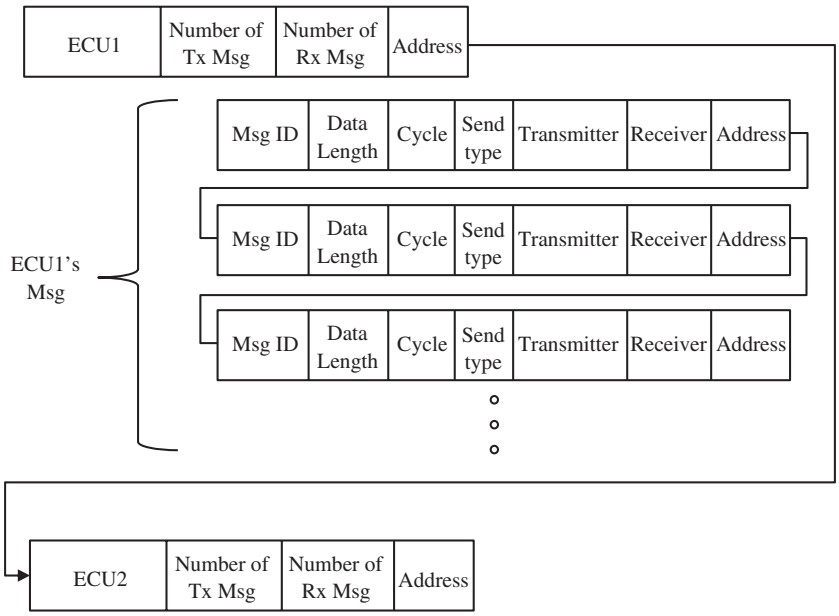


Figure 5. Allocation of Messages using a Linked List.

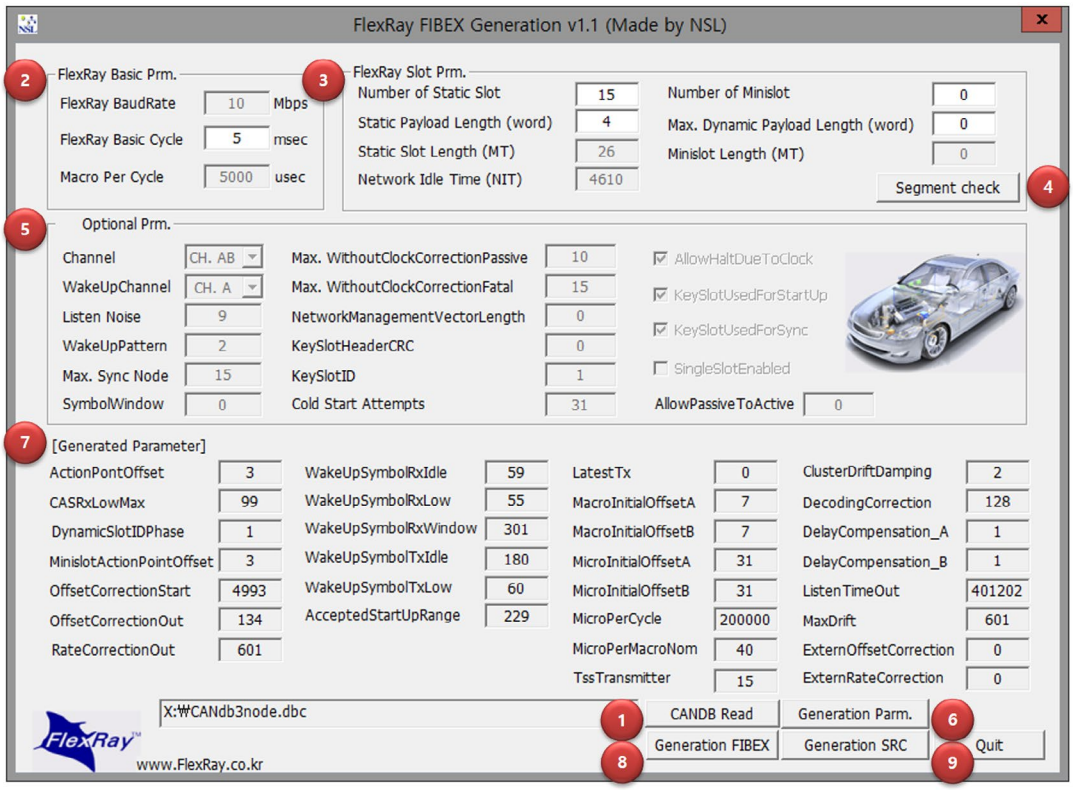
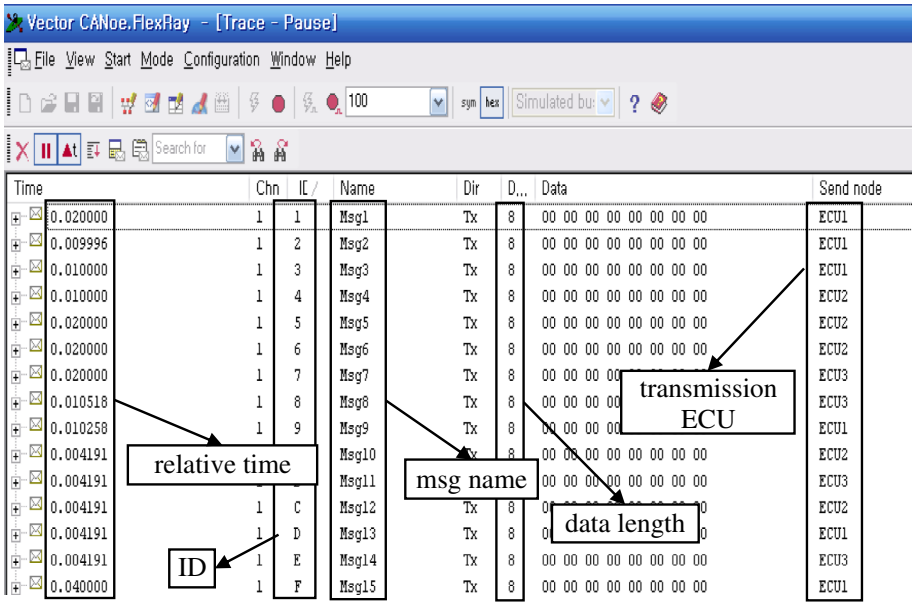
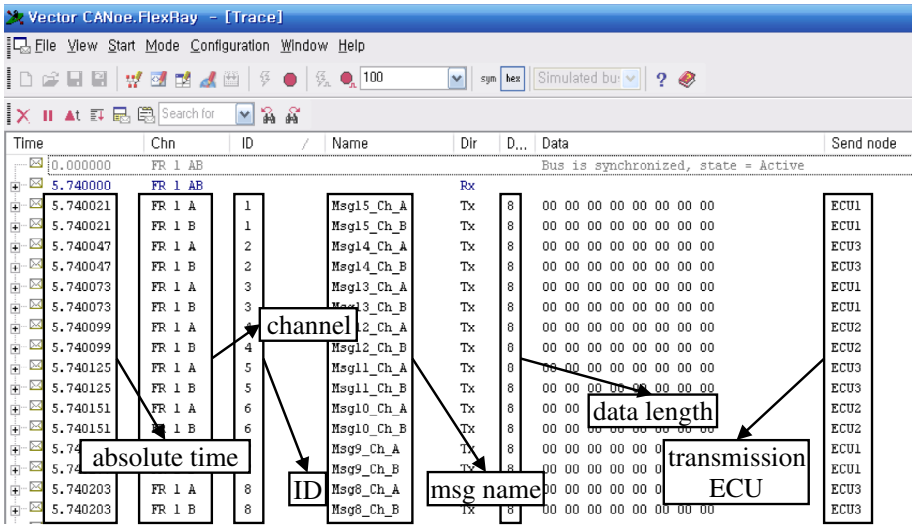


Figure 6. Automatic FIBEX Generation Program.



(a) CAN simulation model with sample DBC



(b) FlexRay simulation model with automatically generated FIBEX

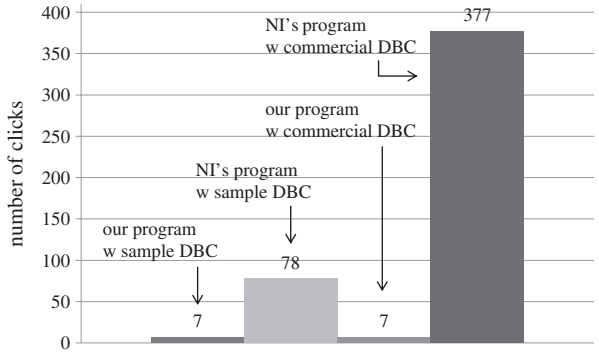
Figure 7. Simulated Data using the CANoe Simulation Model.

prm.' Box). (⑤). If the sum of the static segment, dynamic segment, and NIC is larger than the base cycle, the program shows a pop-up error message window. Thirdly, 'Generation Parm.' Button is pressed. (⑥). Then, the PCR parameters are generated automatically from the relationships shown in Figure 3. (⑦). Finally, the user acquires the FIBEX file by pressing the 'Generation FIBEX' button. (⑧). Also, user acquires the firmware-level source file for the target microcontroller by clicking the 'Generation SRC' button. (⑨).

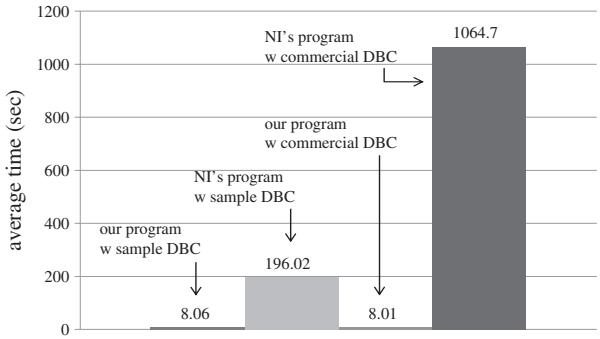
To verify the data integrity of the automatically generated FIBEX data, we created a CAN and FlexRay simulation model using a sample DBC file, shown in Table A.1. The transmission speed of the CAN DBC was set to 500 kbps, and 15 messages were allocated to three ECUs. The DBC defined the send type (cycle or aperiodic), length, cycle time, transmitter, and receiver. The CAN and FlexRay simulation model were developed using Vector's CANoe. FlexRay, which is the most widely used commercial software package for evaluating the data integrity of vehicle network systems.

Figure 7(a) shows simulated data from the CAN simulation model with the sample DBC file. We can see that 15 messages were sent and received without significant network delays. Figure 7(b) shows simulated data for the FlexRay simulation model, including the automatically generated FIBEX file from the sample DBC file. The 15 messages were transmitted and received without any error. The automatically generated FIBEX data maintained integrity, and the FIBEX generation program operated successfully.

To evaluate the commercial feasibility of the DBC-based automatic FIBEX generation program, we compared the performance of the conversion utility with that of the commercially available National Instruments (NI) Database Editor. We created a more complex DBC message description database with 34 ECUs, 38 cyclic messages, and 34 aperiodic messages, shown in Table A.2. This DBC file was defined for the sub-network of a chassis network system from a Korean automotive vendor. We automatically generated the FIBEX file from the DBC file using our method and manually created a FIBEX file



(a) number of mouse clicks required for FIBEX generation



(b) average time for FIBEX generation

Figure 8. Performance Evaluation of Our Software Application and NI's Database Editor.

using NI's Database Editor. We then checked the data integrity using simulation model made by CANoe.FlexRay, and verified that the two FIBEX files were generated successfully.

Figure 8(a) shows the number of mouse clicks required to convert the individual DBC files into FIBEX files. Our program required only seven clicks, whereas the commercial software required 78 and 377 clicks for sample and commercial DBC file described in Table A.1 and A.2. Figure 8(b) shows the average time for the conversion obtained from 10 trials. Our program required 8.06 and 8.01 seconds for the two cases described in Table A. However, NI's Database Editor required 196.02 and 1064.7 seconds. The reason for this significant difference is that the existing commercial application requires user input to design the ECU, message, and signal; however, our method uses the DBC data.

6. Summary and Conclusions

This paper presents a DBC-based automatic FIBEX generation method. The FlexRay network design is more complex than the CAN network design, and a FlexRay designer must provide details of the protocol and specify numerous inter-related parameters, whereas a CAN network designer only needs to specify the behavior of individual nodes as well as a few network-related parameters, such as the transmission data-rate. This complexity has hindered the widespread uptake of the FlexRay protocol. The conclusions derived from this research are as follows:

First, when FlexRay network systems are designed using the automotive industry's CAN message description format such as the DBC format, it is necessary to calculate the 53 platform configuration register (PCR) parameters. In this paper, five key parameters are identified among the 53 PCR parameters, and the relationships between these parameters are analyzed to compute proper values based on five key parameter values.

Second, a software application is developed to convert the DBC format into the FIBEX format. The DBC and FIBEX formats are analyzed to parse and export data, and a rudimentary logic is used to select key parameter values so that the conversion process can proceed with minimal input from the user.

Third, the results of the conversion process are verified for correctness. It is verified by the network simulation that the FlexRay network composed by the automatically generated FIBEX operates similarly compared to the CAN network operation. The application facilitates conversion, requiring far less user input. This feature is expected to assist FlexRay network designers to migrate from CAN to FlexRay.

However, this research has been limited to testing the feasibility of the DBC-based automatic FIBEX generation for simple simulation model by using CANoe.FlexRay. Hence, its performance should be evaluated using an experimental model with real ECU for chassis networking system. Besides, further research may include the optimal selection of parameter values in terms of network utilization, transmission delay, jitter, and so on.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Young Hun Song received a B.S. degree from Pukyong National University, Busan, Korea, in 2008. He is a researcher in the Korea Electrotechnology Research Institute, Changwon, Korea. His research interests include in-vehicle networking system and networked control system.

Suk Lee received a Ph.D. degree from The Pennsylvania State University, University Park, in 1990. He is a professor in the School of Mechanical Engineering, Pusan National University, Busan, Korea. His research interests are industrial network, in-vehicle network, home network, and networked control system.

Kyoung Nam Ha received a Ph.D. degree from Pusan National University, Busan, Korea, in 2010. He is a principal researcher in the Korea Institute of Industrial Technology, Busan, Korea. His research interests include marine robot, remote control system, and in-vehicle networking system.

Kyung Chang Lee received a Ph.D. degree from Pusan National University, Busan, Korea, in 2003. He is a professor in the Department of Control and Instrumentation Engineering, Pukyong National University, Busan, Korea. His research interests are embedded network system, industrial network, robotic network, in-vehicle network, home network, wireless sensor network, and networked control system.

References

Armengaud, E., Steininger, A., & Horauer, M. (2008). Towards a systematic test for embedded automotive communication systems. *IEEE Transactions on Industrial Informatics, 4*(3), 146–155.

Association for Standardization of Automation and Measuring Systems (ASAM) (2013). Field Bus Exchange Format, http://www.asam.net.

Choi, J., Choi, J., Ko, H., Bae, K., An, K. J., Kim, C. S., & Choi, J. (2014). A Smart Service Robot Middleware on Ubiquitous Network Environments. *Intelligent Automation & Soft Computing, 20*(1), 47–59.

Cummings, R. (2008). Easing the transition of system designs from CAN to FlexRay, *SAE Technical Paper*, 2008-01-0804.

Han, G., Natale, M. D., Zeng, H., Liu, X., & Dou, W. (2013). Optimizing the implementation of real-time Simulink models onto distributed automotive architectures. *Journal of Systems Architecture, 59*(10), 1115–1127.

Jang, K., Park, I., Han, J., Lee, K., & Sunwoo, M. (2011). Design framework for FlexRay network parameter optimization. *International Journal of Automotive Technology, 12*(4), 589–597.

Kang, M., Park, K., & Jeong, M. K. (2013). Frame packing for minimizing the bandwidth consumption of the FlexRay static segment. *IEEE Transactions on Industrial Electronics, 60*(9), 4001–4008.

Kim, M. H., Lee, S., & Lee, K. C. (2015). Performance evaluation of node-mapping-based Flexray-CAN gateway for in-vehicle networking system. *Intelligent Automation & Soft Computing, 21*(2), 251–263.

Leen, G., Hefferman, D., & Dunne, A. (1999). Digital networks in the automotive vehicle. *Computer & Control Engineering Journal, 10*(6), 257–266.

Liebezeit, T., Junghanns, A., Bonin, M., & Serway, R. (2012). Software-in-the-loop using virtual CAN buses: current solutions and challenges, 5th Conference Simulation and Testing for Automotive. *Electronics,* 1–11.

Mishra, P. K., & Naik, S. M. (2005). Distributed control system development for FlexRay-based systems. *SAE Technical Paper*, 2005-01-1279.

Park, I. & Sunwoo, M. (2011). FlexRay network parameter optimization method for automotive applications. *IEEE Transactions on Industrial Electronics, 58*(4), 1449–1459.

Rai, D., Jestin, T.K., & Vitkin, L. (2008). Model-based development of AUTOSAR-compliant applications: Exterior lights module case study. *SAE international journal of passenger cars-electronic and electrical systems, 1*(2008-01-0221), 84–91.

Reiter, S., Viehl, A., Bringmann, O., & Rosenstiel, W. (2016). Fault injection ecosystem for assisted safety validation of automotive systems. *High Level Design Validation and Test Workshop (HLDVT), 2016 IEEE International,* 62–69.

Rho, J., Azumi, T., Oyama, H., Sato, K., & Nishio, N. (2016). Distributed processing for automotive data stream management system on mixed single-and multi-core processors. *ACM SIGBED Review, 13*(3), 15–22.

Rosset, V., Souto, PF., Portugal, P., & Vaspues, F. (2012). Modeling the reliability of a group membership protocol for dual-scheduled time division multiple access networks. *Computer Standards & Interfaces, 34*, 281–291.

Schmidt, E.G., & Schmidt, K. (2009). Message scheduling for the FlexRay protocol: The dynamic segment. *IEEE Transactions on Vehicular Technology, 58*, 2160–2169.

Sethna, F., Stipidis, E., & Ali, F.H. (2006). What lessons can controller area networks learn from FlexRay. *IEEE Vehicle Power and Propulsion Conference,* 1–4.

Song, YH., Lee, S., & Lee, KC. (2013). *Automatic FIBEX generation from CANdb for FlexRay network*. 2013 XXIV International Conference on Information, Communication and Automation Technologies, Sarajevo, Bosnia and Herzegovina, 978–981.

Stroop, J., & Stolpe, R. (2006). *Prototyping of automotive control systems in a time-triggered environment using FlexRay*. Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design, Munich, Germany, 2332–2337.

Vector. (2007). DBC File Format Documentation.

Zeng, H., Natale, M., & Ghosal, A. (2011). Schedule optimization of time-triggered systems communication over the FlexRay static segment. *IEEE Transactions on Industrial Informatics, 7*(1), 1–17.

Zhao, Y., & Zhong, Z. (2008). Development on automotive electronic real time control software based on auto-code generation technology. *Computer Aided Engineering, 3*, 1–12.

Appendix

Table A.1. Message Set of a Sample DBC file.

transmitter	message	ID	length	send type	cycle time	receiver
ECU1	Msg1	0x01	8	Cyclic	20	ECU2
	Msg2	0x02	8	Cyclic	10	ECU2, ECU3
	Msg3	0x03	8	Cyclic	10	ECU2
	Msg9	0x09	8	Cyclic	10	ECU2
	Msg13	0x0d	8	Cyclic	5	ECU3
	Msg15	0x0f	8	Cyclic	40	ECU2
ECU2	Msg4	0x04	8	Cyclic	10	ECU1
	Msg5	0x05	8	Cyclic	20	ECU1
	Msg6	0x06	8	Cyclic	20	ECU1
	Msg10	0x0a	8	Cyclic	5	ECU3
	Msg12	0x0c	8	Cyclic	5	ECU1, ECU3
ECU3	Msg7	0x07	8	Cyclic	20	ECU1, ECU2
	Msg8	0x08	8	Cyclic	10	ECU1, ECU3
	Msg11	0x0b	8	Cyclic	5	ECU2
	Msg14	0x0e	8	Cyclic	5	ECU1

Table A.2. Message Set of a Commercial DBC File for Chassis Network System.

transmitter	message	ID	length	send type	cycle time	receiver
ECU1	Msg1	0x01	8	cyclic	20	ECU3, 5, 8, 14
	Msg2	0x02	8	cyclic	20	ECU14
ECU2	No Msg Transmit					
ECU3	Msg3	0x03	8	cyclic	20	ECU1, 5, 8, 10, 12, 13, 27, 29
	Msg68	0x68	8	aperiodic	100	ECU27
ECU4	Msg51	0x51	8	cyclic	20	ECU8, 14, 23, 31
	Msg52	0x52	8	aperiodic	100	ECU4, 13, 24
	Msg53	0x53	8	aperiodic	100	ECU8, 14, 23, 28

(*Continued*)

Table A.2. (*Continued*).

ECU5	Msg4	0x04	8	cyclic	1000	ECU8, 21, 22, 23
	Msg5	0x05	8	cyclic	1000	ECU21
	Msg6	0x06	8	aperiodic	100	ECU21
	Msg7	0x07	8	cyclic	10	ECU8
	Msg8	0x08	8	aperiodic	100	ECU10, 31
ECU6	No Msg Transmit					
ECU7	Msg9	0x09	1	cyclic	100	ECU8
ECU8	Msg11	0x11	8	cyclic	20	ECU4, 10, 12, 13, 14, 18, 31
	Msg13	0x13	8	aperiodic	100	ECU12
ECU9	Msg12	0x12	8	aperiodic	100	ECU3, 5, 7, 10, 12, 13, 14, 15, 18, 23, 27, 29, 31, 34
ECU10	Msg14	0x14	8	aperiodic	100	ECU8
ECU11	Msg16	0x16	8	aperiodic	100	ECU12
	Msg17	0x17	8	aperiodic	100	ECU12
	Msg18	0x18	8	aperiodic	100	ECU12
ECU12	Msg21	0x21	8	cyclic	10	ECU1, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 17, 19, 20, 21, 22
	Msg22	0x22	8	cyclic	10	ECU1, 3, 4, 5, 7, 8, 13, 14, 15, 17, 23, 29,34
	Msg23	0x23	8	cyclic	10	ECU19
	Msg24	0x24	8	cyclic	10	ECU8, 10, 13, 15, 28, 29
	Msg25	0x25	8	cyclic	10	ECU4, 5, 14, 29
	Msg26	0x26	8	cyclic	10	ECU8, 13, 14, 15, 19, 29
	Msg27	0x27	8	aperiodic	100	ECU11
	Msg28	0x28	8	cyclic	10	ECU6, 8, 15, 20, 29
	Msg29	0x29	8	cyclic	10	ECU29
	Msg30	0x30	8	cyclic	10	ECU29
ECU13	Msg31	0x31	8	aperiodic	100	ECU8, 14, 29
ECU14	Msg10	0x10	2	aperiodic	100	ECU26
	Msg32	0x32	8	aperiodic	100	ECU8, 12, 13
	Msg33	0x33	8	aperiodic	100	ECU1, 4, 5, 7, 13, 18, 22, 23, 27, 29, 34
	Msg34	0x34	8	cyclic	10	ECU33
	Msg35	0x35	8	aperiodic	100	ECU4
	Msg56	0x56	8	cyclic	10	ECU1, 4, 5, 8, 12, 13, 23, 27, 29, 34
	Msg57	0x57	4	cyclic	20	ECU1, 5
	Msg58	0x58	8	cyclic	20	ECU4, 12, 29
	Msg59	0x59	4	cyclic	20	ECU1
	Msg60	0x60	8	cyclic	20	ECU1, 5, 8, 10, 12, 13, 22, 23, 29
	Msg66	0x66	8	aperiodic	100	ECU20
	Msg69	0x69	8	aperiodic	100	ECU4, 7, 22
ECU15	Msg36	0x36	8	cyclic	10	ECU8, 12, 27
ECU16	No Msg Transmit					
ECU17	Msg37	0x37	8	cyclic	100	ECU12
ECU18	Msg39	0x39	8	aperiodic	100	ECU8, 18, 23
ECU19	Msg40	0x40	8	cyclic	10	ECU12
ECU20	Msg41	0x41	3	cyclic	100	ECU8, 14, 27
	Msg42	0x42	8	aperiodic	10	ECU27
	Msg67	0x67	8	aperiodic	100	ECU14
ECU21	Msg43	0x43	8	aperiodic	100	ECU5
	Msg44	0x44	8	aperiodic	100	ECU5
	Msg45	0x45	8	aperiodic	100	ECU5
ECU22	No Msg Transmit					
ECU23	Msg46	0x46	8	aperiodic	100	ECU4, 8
	Msg47	0x47	8	aperiodic	100	ECU4, 8
ECU24	Msg48	0x48	8	cyclic	10	ECU12
ECU25	Msg49	0x49	8	aperiodic	100	ECU8
ECU26	Msg50	0x50	5	cyclic	10	ECU4, 5, 7, 14, 22, 23, 34
ECU27	Msg54	0x54	8	aperiodic	100	ECU20
	Msg55	0x55	8	aperiodic	100	ECU8, 16
ECU28	No Msg Transmit					
ECU29	Msg61	0x61	8	cyclic	10	ECU1, 4, 7, 8, 10, 12, 13, 14, 22, 27, 31, 34
	Msg62	0x62	8	cyclic	10	ECU4, 8, 12, 14, 15, 31, 34
	Msg63	0x63	8	cyclic	10	ECU4, 8, 10, 12, 14
ECU30	Msg38	0x38	8	aperiodic	100	ECU1, 3, 4, 5, 6, 7, 8, 12, 14, 15, 17, 19, 20, 24, 26, 29, 32, 33
ECU31	Msg64	0x64	8	aperiodic	100	ECU12
ECU32	Msg65	0x65	2	cyclic	50	ECU8, 10, 31
ECU33	Msg70	0x70	8	cyclic	10	ECU14
	Msg71	0x71	8	cyclic	10	ECU14
	Msg72	0x72	8	aperiodic	100	ECU14
ECU34	Msg19	0x19	5	cyclic	50	ECU8, 10, 31
	Msg20	0x20	5	cyclic	20	ECU7, 8