Data logging GUI Application for Flexray data logger

Nitin Koparde[1], Supriya Shanbhag[2], Sonal Suryavanshi[3], Veeresh Ambe[4]

[1, 2, 3]*Department of Electronics & Communication Engineering, KLS Gogte Institute of Technology, Belagavi, India*
[4]*Vayavya Labs Pvt. Ltd., Belagavi. India*

---***---

Abstract - *Work presented in this paper deals with the design and development of a Graphical user interface for Flexray data logger, such that it will perform real-time data logging and Flexray communication. This application is developed based on C# .NET technology, which uses a serial interface for flexray communication with the FPGA development board. Hence FPGA Board in this project has Flexray Controller and UART controller IP core in it and follows the GUI frame format for data transactions of the flexray frame. The GUI application provides real-time, low cost, and accuracy for flexray data logging, also it can be utilized for testing of flexray drivers and flexray controllers.*

***Keywords: Flexray, GUI, Datalogger, Flexray Header, CRC-11* Calculation, Flexray frame, Flexray data logging.**

1. INTRODUCATION

Flexray protocol is an automotive protocol developed by FlexRay consortium. FlexRay protocol is broadly utilized in automotive industries for its high speed and reliability than CAN bus protocol. It also provides fault tolerance and time deterministic applications. Flexray data logging is a process of recording flexray data and monitoring its events. Flexray is a main data communication protocol in some luxury vehicles, which is responsible for applications like Brake-by-wire and steer-by-wire (SBW). To check and test flexray operations data logger was introduced, that was done by oscilloscope device in the past. But oscilloscope has less accuracy and time consumption is more.

The current work was focused on the design and development of the GUI application for flexray Data logging. This provides a better accuracy rate and less time consumption than an oscilloscope.

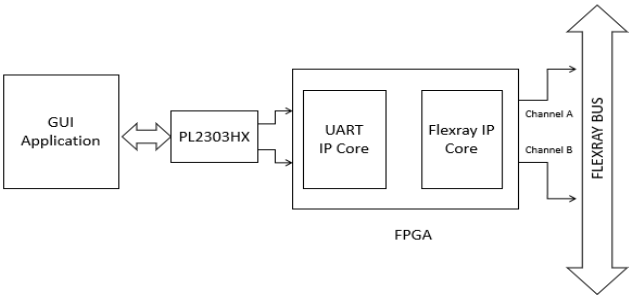


Fig. 1. The connection between the GUI and FPGA board.

In this project, FPGA board ZYNQ7020 has been used as a flexray data logger for hardware-level operations. It implements the UART controller and flexray Controller on it. PL2303HX device has been used for bridge connectivity between PC and FPGA board.

It is a USB to UART converter, which supports the programmable baud rate from 12Mbps to 75bps. In this project, the selected baud rate will be 6Mbps for optimum and stable performance. Transmitted data from the GUI is captured with PulseView application, with the help of a logic analyzer tool, and a real-time flexray data log is obtained in the trace window of the GUI.

2. Related Works

Martin Patak in "Flexray controller" [1] has given a brief explanation about Flexray protocol and its controller operations and created his IP core for it and then implemented it on the FPGA development board, that is interfaced with NIOS II bus system and is also known as a 32-bit parallel interface. He created his command formats for the Controller host interface of flexray that works very efficiently, but operating with the NIOS II system as a host is time-consuming and not user friendly.

Wu Yan, Wang Lifang, Liao Chenglin, Li Fang Implemented "In-vehicle bus Flexray monitoring system" [2] in which they designed flexray bus data acquisition and calibration system with high-speed USB interface for flexray communication, and also explained briefly about flexray hardware and software implementations. Then they developed the flexray bus monitoring system, which works accurately with real-time and reliable.

Ahmad Shukri Fazil Rahman, Samila Mat Zali, Shamshul Bahar Yaakob, Abdul Rahim Abdul Razak, Mohd Zamri Hasan, Siti Hajar Mohd Tahar in "Conceptual implementation of a data logger with a graphical user interface data extraction program" [3] they explained the implementation of a data logger with a graphical user interface with data extraction program. Hence, they designed a GUI for data extraction. It based on a microcontroller. They used two sensors that are a temperature sensor and a humidity sensor. Its data has efficiently recorded with the EEPROM module, and the GUI achieves data transmission and transformation. For real-time data logging, they implemented the RTC clock and SD card facility to record data as well and so that it enhances the efficiency and accuracy of the data logger. And here, it is used only for serial data logging.

3. METHODOLOGY

3.1 Graphical user interface

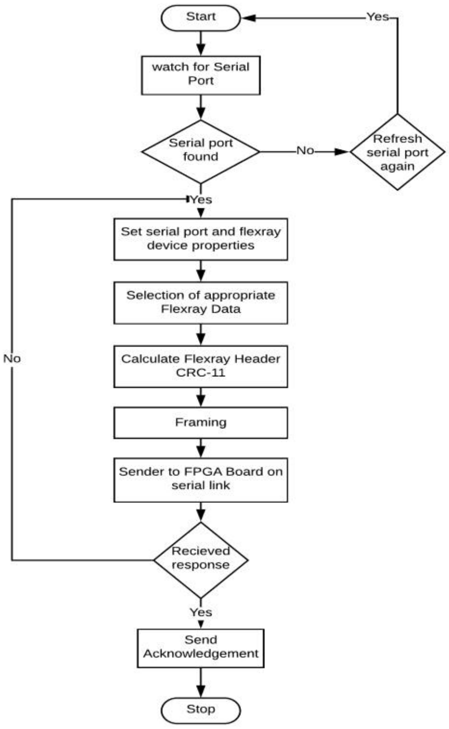

Start
Yes
watch for Serial Port
Serial port found
No
Refresh serial port again
Yes
Set serial port and flexray device properties
Selection of appropriate Flexray Data
No
Calculate Flexray Header CRC-11
Framing
Sender to FPGA Board on serial link
Recieved response
Yes
Send Acknowledgement
Stop

Fig. 2. GUI flow chart for Transmission

Figure 2 represents the working flow chart of GUI transmission. The first user needs to look for a serial port. If serial port not found, then the user needs to hit the refresh button later try again. If it's found, then the user needs to set serial port properties, then hit the connect button to open serial port. After that user needs to set flexray properties like channel selection as A or B, baud rate as 2.5Mbps,5MPbs or 10Mbps, and node type as integration mode, leading mode, or both respectively, then appropriate flexray parameters were preconfigured for optimum performance in the GUI.

At last, the user needs to select appropriate flexray data for transmission like frame type, cycle controls, null bit, startup bit, sync bit, slot id, payload length, and payload data, respectively. And then flexray header CRC-11 is calculated and displayed in GUI, the calculation process is further explained in this paper. And next framing process starts data is framed according to frame format. That is also shown below in this paper. Then the user needs to hit the send button to start transmitting frames at once to the FPGA board. If FPGA receives the frame, then it will send acknowledgment, or else the user needs to resend it again.

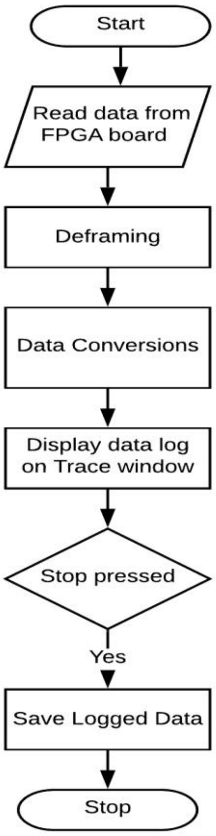


Fig. 3. GUI flow chart for Reception

Figure 3 represents the working of GUI Reception. After the transmission of flexray frames to the FPGA board, it will process the frames further then starts communicate with external flexray devices via a flexray bus. Then it starts receiving frames from it. Then those frames will be sent to the GUI through a serial link. First, GUI will read the data and then starts the de framing process by recognizing them with the start of frame and end of frame indicators, for this project ASCII 'R' will be chosen to indicate receiving frame and ASCII' 'S' has decided to show status frame. After de framing process, the data conversions process will start, their data is converted to a user-readable form in the GUI, and next in the GUI data log starts displaying with time stamps. If the stop button pressed in the GUI, then it will terminate the data logging process, or else it keeps data logging till the limit of the hardware side. And the user can also have the feature to store the logged data. The GUI development is done by using Microsoft visual studio community edition 2019 IDE tool and using C# .NET technology framework 4.8.

3.2 Flexray header CRC 11 calculation

Header CRC 11 is essential for flexray operations as only if header CRC 11 of GUI application matches with flexray controllers CRC 11 in the FPGA, then Only further communication will happen with FPGA board. Hence, it is calculated based on the calculation algorithm of flexray specification version 2.1. Figure 4 represents the flow chart for flexray header CRC-11 calculation, according to it, header CRC is 11 bits in length, the initial value is 1Ah, and CRC polynomial is 385h.

The Following equation gives CRC-11 polynomial: -

$$x^{11} + x^9 + x^8 + x^7 + x^2 + 1 = (x + 1)\ (x^5 + x^3 + 1)\ \ (x^5 + x^4 + x^3 + x + 1).$$

CRC data has formed by sync bit, Startup bit, frame id, and payload length. Then vNextBit (Data is left-shifted by bit index and concatenated with 0x01h) is calculated by applying for-loop from 19 to 0th-bit indexes, respectively.

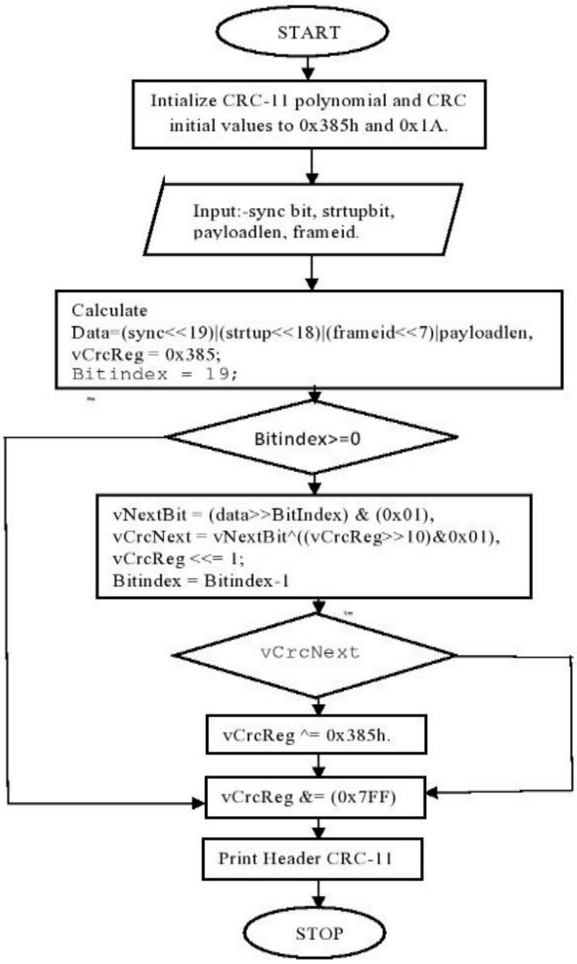

START
Intialize CRC-11 polynomial and CRC initial values to 0x385h and 0x1A.
Input:-sync bit, strtupbit, payloadlen, frameid.
Calculate
Data=(sync<<19)|(strtup<<18)|(frameid<<7)|payloadlen,
vCrcReg = 0x385;
Bitindex = 19;
Bitindex>=0
vNextBit = (data>>BitIndex) & (0x01),
vCrcNext = vNextBit^((vCrcReg>>10)&0x01),
vCrcReg <<= 1;
Bitindex = Bitindex-1
vCrcNext
vCrcReg ^= 0x385h.
vCrcReg &= (0x7FF)
Print Header CRC-11
STOP

Fig. 4. Flow chart for Header CRC-11 calculation.

Later in the same loop vCrcNext (vNextBit is XORed with (vCrcReg and concatenated with 0x01h)) is calculated. And if vCrcNext is "1",Then vCrcReg is to be XORed with 0x385h. Or else exit the loop and then vCrcReg is logically & with 0x7FFh for masking to get 11bit Flexray Header CRC.

3.3 Frame format

Frame format shown below is for data transmitted from the GUI application to FPGA Board.

Table -1: Configuration frame for each parameter.

C	SADR	SWRITEDATA	'/'

1. 'C' - Command for configuration register (1 Byte).
2. SADR - Command for Flexray core (1 Byte).

3. SWRITEDATA - Data for the command (4 Bytes). That is the Flexray parameter, each of 4 bytes.
4. '/' - Delimiter (1 Byte).
5. Total (Bytes)- 7bytes

Table -2: Startup frame.

S	SADR	SWRITEDATA	'/'

1. 'S' - Command for Startup (1 Byte).
2. SADR - Command for Flexray core (1 Byte).
3. SWRITEDATA - Data for the command (4 Bytes). It contains information on the Flexray device configuration.
4. '/' - Delimiter (1 Byte).
5. Total (Bytes)- 7bytes

Table -3: Data frame.

T	DATA	'/'

1. 'T' - Command for DATA frame (1 Byte).
2. DATA - Cycle Control + Header + Payload (5+20+1Bytes)
3. '/' - Delimiter (1 Byte).
4. Total (Bytes) - 28bytes.

Table -4: End of the frame.

E	'/'

1. 'E' - Command for End of the frame (1 Byte).
2. '/'- Delimiter (1 Byte).
3. Total (Bytes)- 2bytes

Table -5: Total Frame.

Config. frame	Startup frame	Data frame	End frame

Table -6: Stop-frame.

U	'/'

1. 'U' – Command for stop data logging process.
2. '/' - Delimiter (1 Byte).
3. Total (Bytes)- 2bytes

Stop-frame is used to terminate the data logging process.

3.4 New features of GUI application

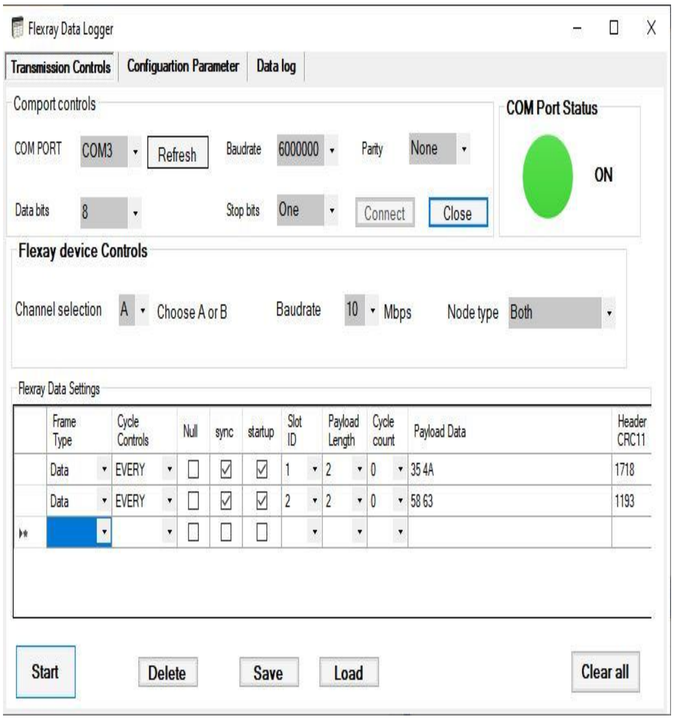
The save and Load feature: -this feature gives the user to store configured flexray data in the form of .txt file and reload it whenever the user wants those configurations. In the trace window save button is used to store the logged data.

Frame format controls: - this feature gives flexibility to GUI application, such that users can change any of those values inside the GUI application without changing the backend code.

Compatibility and Portability: - this GUI application works on the Windows platform, and it is portable, and it works for any FPGA development board that has a flexray controller IP core and UART controller IP Core implemented in it and follows GUI Frame Format.

4. RESULTS

GUI transmission has captured with the help of the PulseView software, and signals were represented in ASCII format to understand better as compare with frame formats.

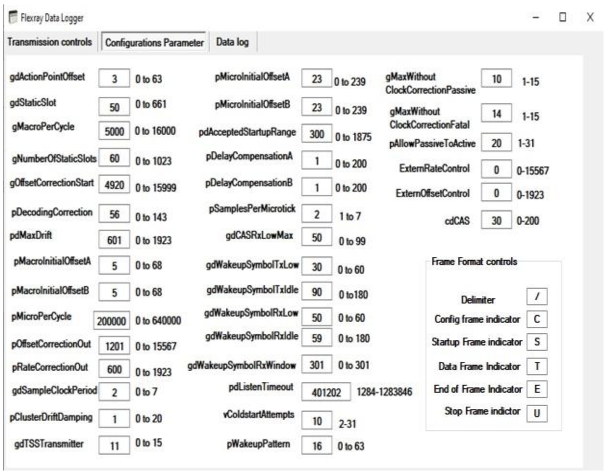


COM PORT COM3
Baudrate 6000000
Parity None
Data bits 8
Stop bits One
ON
Channel selection A Choose A or B
Baudrate 10 Mbps
Node type Both

Frame Type	Cycle Controls	Null	sync	startup	Slot ID	Payload Length	Cycle count	Payload Data	Header CRC11
Data	EVERY	□	☑	☑	1	2	0	35 4A	1718
Data	EVERY	□	☑	☑	2	2	0	58 63	1193
		□	□	□					

Fig. 5. Data selected GUI

Figure 5 represents the Data inserted for transmission to the FPGA device. Here two frames were entered to transmit, and header CRC 11 calculated as shown in Figure 6.it is in decimal format.



Flexray Data Logger

Transmission controls | Configurations Parameter | Data log

gdActionPointOffset 3 0 to 63
gdStaticSlot 50 0 to 661
gMacroPerCycle 5000 0 to 16000
gNumberOfStaticSlots 60 0 to 1023
gOffsetCorrectionStart 4920 0 to 15999
pDecodingCorrection 56 0 to 143
pdMaxDrift 601 0 to 1923
pMacroInitialOffsetA 5 0 to 68
pMacroInitialOffsetB 5 0 to 68
pMicroPerCycle 200000 0 to 640000
pOffsetCorrectionOut 1201 0 to 15567
pRateCorrectionOut 600 0 to 1923
gdSampleClockPeriod 2 0 to 7
pClusterDriftDamping 1 0 to 20
gdTSSTransmitter 11 0 to 15

pMicroInitialOffsetA 23 0 to 239
pMicroInitialOffsetB 23 0 to 239
pdAcceptedStartupRange 300 0 to 1875
pDelayCompensationA 1 0 to 200
pDelayCompensationB 1 0 to 200
pSamplesPerMicrotick 2 1 to 7
gdCASRxLowMax 50 0 to 99
gdWakeupSymbolTxLow 30 0 to 60
gdWakeupSymbolTxIdle 90 0 to180
gdWakeupSymbolRxLow 50 0 to 60
gdWakeupSymbolRxIdle 59 0 to 180
gdWakeupSymbolRxWindow 301 0 to 301
pdListenTimeout 401202 1284-1283846
vColdstartAttempts 10 2-31
pWakeupPattern 16 0 to 63

gMaxWithout ClockCorrectionPassive 10 1-15
gMaxWithout ClockCorrectionFatal 14 1-15
pAllowPassiveToActive 20 1-31
ExternRateControl 0 0-15567
ExternOffsetControl 0 0-1923
cdCAS 30 0-200

Frame Format controls
Delimiter /
Config frame indicator C
Startup Frame indicator S
Data Frame Indicator T
End of Frame Indicator E
Stop Frame indictor U

Fig. 6. FlexRay parameter configured in GUI

Figure 6 represents the FlexRay parameter that was configured for optimum performance.

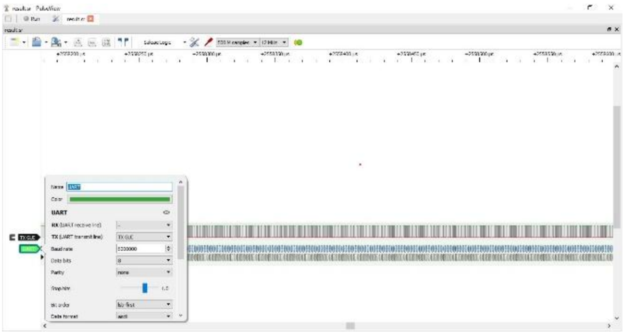


Fig. 7. Capturing transmitted frames at 6Mbps.

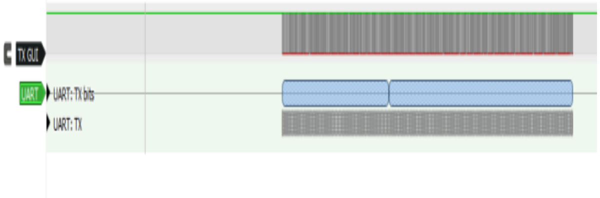


Fig. 8. Total frames generated by the GUI

Figure 8 represents the byte stream generated by the GUI application, and that downloaded into the FPGA device.

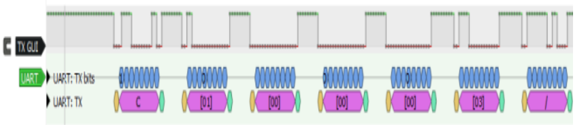


Fig. 9. Configuration frame from starting

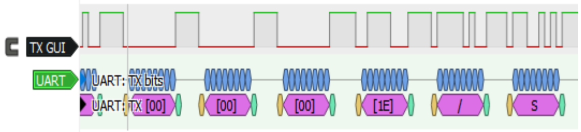


Fig. 10. Configuration frame end.

Figures 9 and 10 represent the signals of Flexray parameters; there are a total of 36 parameters in this project.

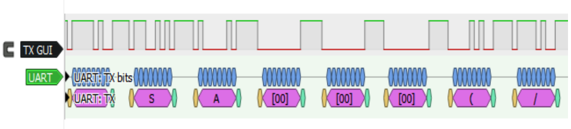


Fig. 11. Startup frame.

Figure 11 represents the Startup frame which has information of Flexray device controls,

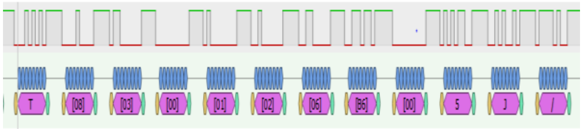


Fig. 12. First flexray message data sent

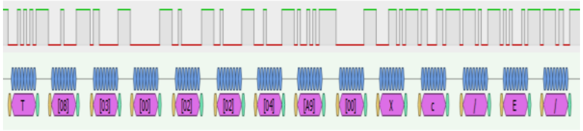


Fig. 13. Last flexray message data sent with End of frames indication.

Figures 12 and 13 have represented flexray data signals as they had selected in the flexray data setup group box, which gives flexray data information to the flexray controller.

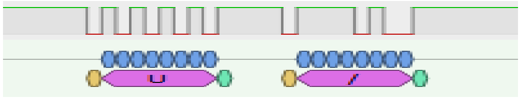
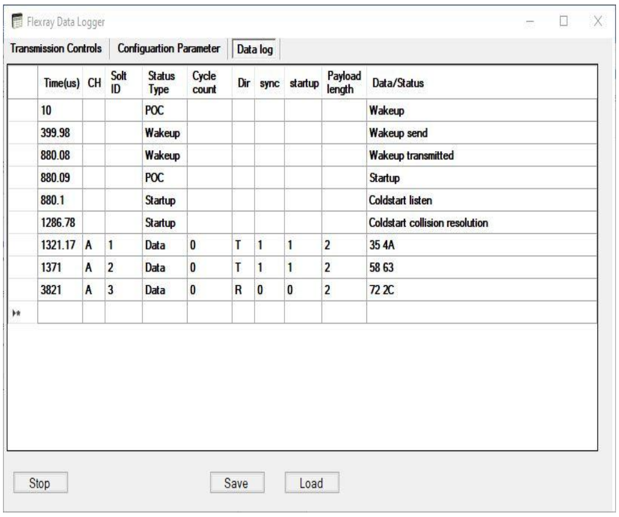


Fig. 14. Stop-frame.

The stop-frame has generated when the stop button has pressed and pressed whenever after the data logging process to stop the process of data logging.



Time(us)	CH	Solt ID	Status Type	Cycle count	Dir	sync	startup	Payload length	Data/Status
10			POC						Wakeup
399.98			Wakeup						Wakeup send
880.08			Wakeup						Wakeup transmitted
880.09			POC						Startup
880.1			Startup						Coldstart listen
1286.78			Startup						Coldstart collision resolution
1321.17	A	1	Data	0	T	1	1	2	35 4A
1371	A	2	Data	0	T	1	1	2	58 63
3821	A	3	Data	0	R	0	0	2	72 2C

Fig. 15. Trace window of GUI application

Figure 15 represents the logged data of flexray with their respective statuses generated from the flexray controller (FPGA). And this process can be stopped by pressing the stop button.

5. CONCLUSIONS AND FUTURE WORK

Previous works carried on Flexray data logger explains [2] flexray data logging with complex USB interface. And also explains about Flexray operations and its functionality [1], where the NIOS II system was used as a Host which does limited host activities. Apart from that, an old method of data logging with an oscilloscope consumes more time and can cause human error too.

To overcome the complex USB interface and non-user-friendly NIOS II system, the GUI application for FlexRay data logger is designed and developed. It performs multiple tasks with a simple, user-friendly design through a serial interface.

The GUI application presented can be used for testing flexray drivers and flexray controllers. In future work, it can be extended to support cross-platform for GUI, add support to the CAN bus and LIN protocols, and also can be able to work in the dynamic segment of the flexray frame.

REFERENCES

[1] FlexRay Consortium FlexRay Protocol Specification V2. 1 Rev. A, 2005.

[2] Wu Yan, Wang Lifang, Liao Chenglin, Li Fang, "In-Vehicle Flexray Bus Monitoring System." Key Laboratory of Power Electronics &Electric Drives, Institute of Electrical Engineering, Chinese Academy of Sciences, Beijing, 100190, China - 2014.

[3] Martin Patak, "Flexray controller," Prague, 2012.

[4] Ahmad Shukri Fazil Rahman, Samila Mat Zali, Shamshul Bahar Yaakob, Abdul Rahim Abdul Razak, Mohd Zamri Hasan, Siti Hajar Mohd Tahar," Conceptual implementation of a data logger with a graphical user interface data extraction program" Pusat Pengajian Kejuruteraan Sistem Elektrik, Universiti Malaysia Perlis, Kampus Alam UniMAP Pauh Putra, Perlis, Malaysia.