PREPRINT

J. Sobotka, J. Novák: FlexRay ECU mission critical parameters measurement, Measurement. Vol. 100, March 2017, pp. 213-222.
https://doi.org/10.1016/j.measurement.2016.12.051

FlexRay ECU mission critical parameters measurement

Jan Sobotka [1], Jiří Novák [1]

[1] Czech Technical University in Prague, Faculty of Electrical Engineering, Technická 2, 166 27 Prague 6, Czech Republic

jan.sobotka@fel.cvut.cz, jnovak@fel.cvut.cz

Abstract

Network operation of a FlexRay ECUs (Electronic Control Units) in passenger cars is influenced by the significant number of parameters that have to be written into the ECU FlexRay controller. To keep the FlexRay network robust, the correct parameter values must be set in all ECUs of the FlexRay communication cluster. This is not a trivial task since particular ECUs are supplied by different manufacturers and any manufacturer can change some parameter either by mistake or even intentionally. Effect of such a change is generally unpredictable and can often be observed under specific operational conditions only. The most serious effect is a global FlexRay network failure which usually leads to the fatal vehicle malfunction. Hence it was necessary to develop, implement and validate new dedicated measurement methods enabling evaluation of actual values of the most critical FlexRay parameters at the OSI (Open Systems Interconnection) data-link layer and thus the ECUs individual acceptances testing for system integrator verification purposes. As the mass production FlexRay controllers are not applicable due to a lack of test specific features, deployment of these methods is enabled by utilization of unique FPGA-based FlexRay controller implementation. Proposed measurement methods are focused on parameters specifying the FlexRay wakeup protocol, FlexRay startup procedure, and the FlexRay synchronization mechanism. Each measurement method is described in details including its limits and prerequisites. All the developed methods were validated by experiments on real FlexRay networks and results are included in the paper. Two different types of FlexRay controller core (Freescale and Bosch E-Ray) were used in EUTs (ECU under test) to eliminate the risk of measurement method dependence on a specific controller implementation.

Keywords

FlexRay, ECU, parameter, synchronization, startup, testing

1. Introduction

Passenger car manufacturers play a role of a system integrator today, as a substantial part of vehicle subsystems is supplied by their contractors. This is especially true for vehicle electronics, where particular ECUs are supplied by different manufacturers. Nevertheless, the ECUs have to collaborate seamlessly together. ECUs collaboration takes place through vehicle communication network technologies like CAN, LIN or FlexRay, expected network functionality is thus vital for reliable and safe vehicle operation.

In the case of FlexRay ECU, communication behaviour is affected by tens of parameters that must be set according to the vehicle manufacturer specification for the FlexRay cluster robustness and reliability. The vehicle manufacturer may not rely on ECU manufacturer declaration of conformity (in terms of correct parameter values) and has to measure the actual parametrization instead. This approach is common for CAN and LIN networks, where the number of critical parameters is lower, and measurement methods and instruments are widely available. As far as we know, such measurement methods are not available for the FlexRay technology at all.

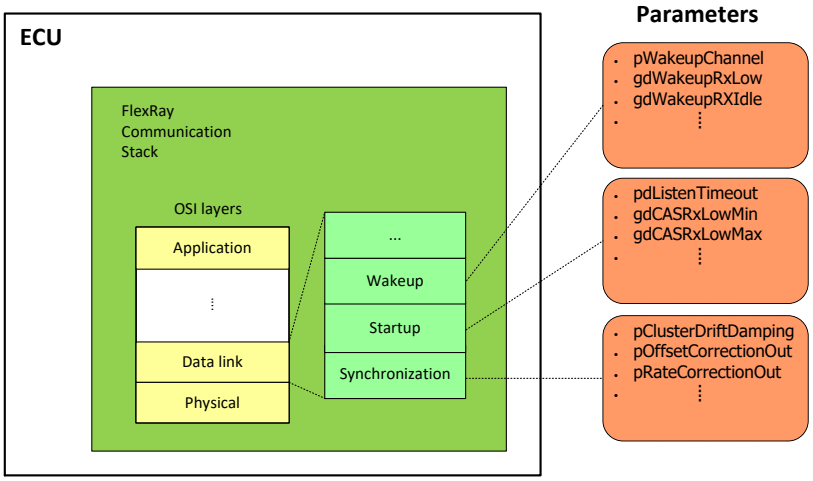

Figure 1: Scope and objectives of the paper

Figure 1 defines the scope and objectives of the paper. FlexRay communication stack within an ECU complies with the OSI model. Data-link layer entities and protocols are implemented by the FlexRay communication controller – the chip provided by the semiconductor manufacturer. Here it is important to emphasize that the paper does not focus on the chip implementation testing and validation; it is expected to work in accordance with the standard. However, the data-link layer behaviour is also significantly influenced by parametrization of its protocols and entities. Values of particular parameters are set by the ECU firmware, and thus it is the responsibility of the ECU manufacturer to set the required parameter values. ECU is a black (or at best a grey) box from the car manufacturer point of view. He knows what parameter values should be set (he has specified them), but he has no access to the ECU firmware to verify them. Car manufacturer, therefore, needs measurement methods and instruments providing for measuring these parameter values without having access to internal ECU data.

The paper proposes a set of measurement methods that are capable to reveal actual parametrization of FlexRay controller in particular ECU. It thus allows for the car manufacturer an independent evaluation

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

of component conformity, which is crucial for the vehicle functionality within the wide operating conditions. The methods are based on idea that the actual parameter values can be extracted from the bus communication by observation of the reaction to the appropriate generated stimuli. As neither the standard FlexRay controllers nor the available FlexRay analysis and simulation tools are able to generate all necessary stimuli, dedicated instrument was designed and implemented [1]. As usual in practice, the time dedicated for measurement is limited [2] and all described methods are thus designed to respect this fact.

The methods described in this paper are not intended for the mass production line measurement. Instead, they are envisioned for usage during ECU development and deployment in a pre-production vehicle integration and evaluation phase. Production phase firmware and ECU parametrization have to stay unchanged or, in the case of necessary changes, the measurements should be passed again.

The paper is organized into following sections. First, a brief necessary overview of FlexRay protocol relevant parts is provided. Next section describes related work, followed by motivation for this research. Major part, describing the measurement methods, is divided into three sections corresponding to the FlexRay wakeup, startup and synchronization processes. Measurement of wakeup and startup parameters is easier in comparison with measurement of time synchronization parameters. Despite this fact, the corresponding methods are included to provide comprehensive survey. Last sections deal with analysis of measurement accuracy, implementation details, and with results of measurements on real systems.

2. Description of relevant parts of FlexRay communication system

Description of FlexRay Communication System can be found in book [3] or directly in ISO protocol specification [4], which is the primary information source. Following paragraphs are focused on relevant principles only. Shortly, FlexRay is high speed (up to 10 Mbit/s) communication technology primarily intended for automotive applications. Both time triggered and event triggered communication is possible and redundant physical layer topology can be used for increased reliability.

Wakeup procedure serves to drive network from low power mode (e.g. after a car is unlocked) [5] to normal operation mode. The procedure starts with ECUs in either standby or sleep mode, where at least bus drivers are powered. The process is controlled at application layer by the ECU firmware. An ECU initiating the cluster wakeup starts sending a wakeup pattern to the bus on one of redundant channels. Another ECU, which recognizes the wakeup pattern, wakes up and continues the process by sending the wakeup pattern to the second FlexRay channel (if it is present). After the wakeup is finished and all the ECUs are woken up, they continue with a startup phase. Below the methods for evaluation of proper wakeup pattern transmission and recognition are provided.

Communication startup phase is intended to initialize the communication cycle and clock synchronization. Start of the FlexRay network depends on a network type. FlexRay specification distinguishes three network types: TT-L, TT-D and TT-E according to clock synchronization principle. In TT-L network the nodes are synchronized using a single clock master. TT-D cluster uses distributed clock synchronization mechanism which will be discussed in next paragraph. Last TT-E type uses an external time gateway. This work is focused on TT-D FlexRay network type, which is the most widespread. For the startup process, the network is divided into coldstart and noncoldstart nodes (ECUs). Only the coldstart

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

ones are active during the communication startup. First coldstart node, which does detect no bus communication, becomes a leading coldstart node. This node immediately sends CAS (Collision Avoidance Symbol) and then starts sending startup frames (normal frames with a startup flag set) according to the TDMA communication schedule. Following coldstart node initializes its clock synchronization and joins communication in communication cycle No. 4. Finally the noncoldstart nodes join the communication. Below the methods for evaluation of proper ECU startup parameter values are provided.

FlexRay normal operation phase is based on the modified TDMA (Time Division Multiple Access) communication cycle structure, as depicted in Figure 2. The individual ECUs' clocks are synchronized using distributed clock synchronization algorithm. ECUs labelled as synchronization nodes are sending frames with sync flag set within the static segment slots. Other (non-synchronization) ECUs only correct theirs clock according to the synchronization nodes. All nodes measure an arrival time of synchronization frames locally. Based on the arrival time deviations the clock corrections are calculated using a Fault-tolerant midpoint (FTM) algorithm [6] and applied in particular nodes.

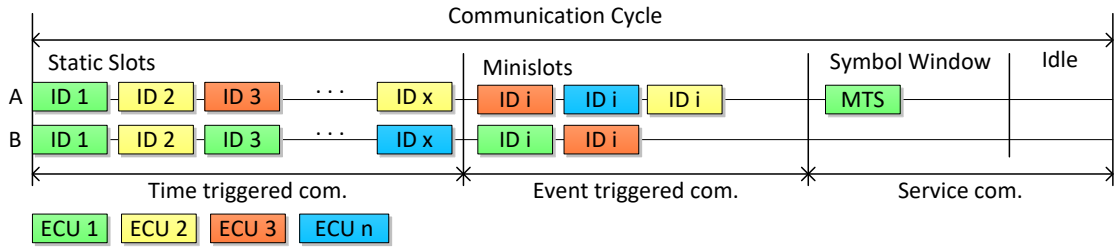

Figure 2: FlexRay communication overview

Behaviour of clock synchronization process is controlled by several parameters. Below the methods for measurement of their values are provided.

3. Related work

Growing complexity of automotive embedded systems requires new measurement and validation techniques [7]. Similar research focused on other vehicle distributed systems (mostly the CAN) was conducted in past. The CAN interface configuration is much simpler than that for the FlexRay, since only few parameters are used, such as time quantum, length of particular bit segments and synchronization jump width. Especially one of them is critical for the ECU with CAN interface deployment – it is the sample point position within the bit time. The risk of incorrect setting and the measurement method is described in [8].

Scope of this paper is a measurement and validation of FlexRay configuration parameters affecting behaviour of individual nodes forming a communication cluster. TEODACS project [9] was the largest activity related to this work. It was targeted at general deployment and validation of FlexRay based distributed system. Under this project the method for remote measurement of offset correction was published [10]. This measurement technique is used in section 6 as a part of several here presented measurement methods. Authors of [11] and [12] have focused on extraction of FlexRay cluster global parameters. They rely on passive bus communication monitoring approach only. The limitation of this approach is inability to reveal the values of all parameters, especially the local node-specific parameters

that define node-specific behaviour within the wakeup, startup and synchronization mechanisms. These parameters apply at boundary conditions that are not usually reached during normal operation. Paper [12] supports our conviction, that the approach based on application of active stimuli is needed. This work is thus an expected extension of communication parameters extraction methods based on passive bus observation. It provides for validation of FlexRay system by means of each single FlexRay node parameters measurement as mentioned in section 1. Compared to papers [11] and [12] we use both the passive communication monitoring as well as the active stimuli to force the necessary FlexRay node response. Presented work can therefore be viewed as a complement to already published passive monitoring based methods of global cluster parameters identification.

The paper extends and validates our preliminary results published in conference proceedings [1] and [13]. Measurement algorithms are described more precisely with important details. Also some inconsistencies, e. g. imprecise *pdListenTimeout* measurement published in [1] are explained and corrected. Simulation of measurement of *pOffsetCorrectionOut* validation from [13] is supported by experimental results. Moreover, limits of offset correction measurement are derived and described by equations 9 and 10.

4. Motivation for measurement of configuration parameters

Significant FlexRay node parameter is a *pRateCorrectionOut*, which determines maximal possible rate correction value the node is allowed to apply. Let's consider the following situation. Desired value of *pRateCorrectionOut* parameter in ECU specification is 601 µT, which is maximal value for communication speed 10 Mb/s and communication cycle length 5 ms. This value allows the FlexRay controller to correct maximal permitted oscillator deviation, which is defined by standard [4] as 1500 ppm of oscillator's nominal frequency.

Let assume that in the ECU delivered from the supplier the configured *pRateCorrectionOut* value is not 601, but only 300 µT instead. Such a violation of node parameters specification would not influence ECU's behaviour under most conditions unless the local oscillator's frequency reaches deviation higher than 750 ppm from nominal value (more precisely from the cluster average value). In this moment the ECU ends communication and goes to the halt state, while it would continue operating with correct parameterization. Using a crystal oscillator natural behaviour [14], this probably happens after long time (several years) due to crystal aging in combination with high or low temperatures [15]. To detect this kind of specification breach special testing methods are needed, allowing the car manufacturer to ask the ECU supplier to fix the parametrization already in pre-production.

Wrong value of *pdListenTimout* is another example of a manufacturer specification breach. *pdListenTimout value* specifies a time spent in a coldstart listen phase. According to specification [4] the *pdListenTimout* value has to be calculated according equation (1).

$$pdListenTimeout[\mu T] = 2 \times (pMicroPerCycle[\mu T] + pRateCorrectionOut[\mu T]) \quad (1)$$

Depending on communication cycle schedule, lower as well as higher parameter values can cause problems with FlexRay cluster startup (e.g. leading coldstart node change or startup phase extension). Three possible example scenarios are shown in Figure 3. All cases consider two coldstart nodes in communication network.

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

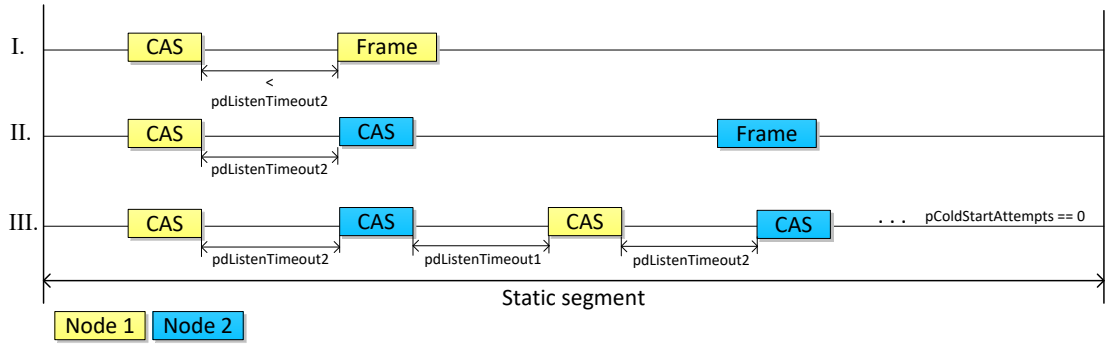

Figure 3: Effects of wrong pdListenTimeout value

The first case labelled I. shows situation when the gap between Collision Avoidance Symbol (CAS) and startup frame in static slot (actual *pdListenTimeout* of node 2) is shorter than the required *pdListenTimeout.* In this case the startup behaviour of the cluster is not seriously affected. The second scenario shows the change of leading coldstart node. Node's 2 *pdListenTimeout* timer expires before the node 1 sends its first startup frame (assigned key slot is later in static segment of communication cycle) and cluster startup is thus a bit delayed. Case III. means that *pdListenTimeout* parameters in both nodes are too small; the nodes are alternating in coldstart leading and the startup delay is significant. To prevent all these problems an evaluation of *pdListenTimeout* parameter actual value is necessary.

Examples presented above show that the problems originated in a wrong ECU parametrization may occur later (caused by components ageing), under specific operating conditions (e.g. extreme temperature), under the spare part change or under the simultaneous influence of already mentioned effects. This kind of problems is hard to reveal because of their sporadic nature. Evaluation of configuration parameter can dramatically increase confidence in FlexRay network fault free operation during the car life cycle. Apposite measurement methods for FlexRay configuration critical parameters were therefore developed and they are presented in following sections.

Wakeup parameters

Bus communication starts with a wakeup procedure, which is intended to power up and force the FlexRay POC (Protocol Operational Control) automaton of each connected node to the ready state. Responsibility for proper wakeup of a node is divided among the bus driver, host (an MCU) and FlexRay communication controller. Bus driver should be able to recognize the wakeup pattern and to wakeup other components including the host MCU and communication controller. Remaining steps are driven by the host with support of communication controller. Summary of relevant parameters is in Tab. 1.

Table 1 Wakeup parameters summary

Parameter	**Range**
pWakeupChannel	Channel A | B
gdWakeupRxLow	8 - 59 gdBit
gdWakeupRxIdle	8 - 59 gdBit

gdWakeupRxWindow	76 - 485 gdBit
gdWakeupTxIdle	45 - 180 gdBit
gdWakeupTxActive	15 - 60 gdBit
pWakeupPattern	0 – 63

Parameter *gdBit* expresses nominal bit time, for bit rate 10 Mbit/s it is equal to 100 ns.

pWakeupChannel

pWakeupChannel denotes channel where the wakeup pattern is transmitted by the node. Evaluation is simple and based on a bus observation only. If the node does not detect wakeup pattern, it sends wakeup pattern on the channel defined by *pWakeupChannel.*

gdWakeupTxIdle, gdWakeupTxActive, pWakeupPattern

These three parameters define the wakeup pattern waveform. *gdWakeupTxIdle defines* duration of bus idle state. *gdWakeupTxActive* is a time period of low bus state. Wakeup symbol consists of one *gdWakeupTxActive* followed by *gdWakeupTxIdle.* Wakeup pattern is a sequence of several wakeup symbols specified by *pWakeupPattern*. Evaluation of these parameters is possible by an oscilloscope or bus sampling with reasonable sampling period (lower than *gdBit/2* period).

gdWakeupRxLow, gdWakeupRxIdle, gdWakeupRxWindow

Complementary to wakeup pattern transmission, wakeup pattern recognition is controlled *by gdWakeupRxLow, gdWakeupRxIdle* and *gdWakeupRxWindow*. Relationship between parameters is shown in Figure 4. For the proper node wakeup it is crucial to test its ability to recognize wakeup pattern on the bus.

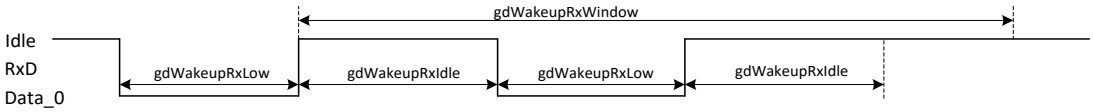

Figure 4: Wakeup pattern in context wakeup window

Testing of wakeup pattern recognition uses assumption, that if node has recognized the wakeup pattern, it does not send wakeup pattern itself. Selected wakeup pattern parameter is being decreased from the maximal permitted length until the wakeup pattern is not detected by EUT. This principle is used for *gdWakeupRxLow* and *gdWakeupRxIdle.* The third parameter is evaluated by changing right side of inequality 2. Modification of *gdWakeupRxLow* value can be used.

$$gdWakeupRxWindow \geq 2 \cdot gdWakeupRxIdle + gdWakeupRxLow \quad (2)$$

Evaluation of the wakeup parameters is straightforward, but integral part of the exhaustive validation of FlexRay controller parametrization. Similar principle is used for LIN (Local Interconnect Network) cluster wakeup testing [16]. In contrary, the LIN wakeup signal timing is fully defined by the standard.

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

5. Startup parameters

Startup is a key process intended to initialize time synchronization for whole cluster. Critical parameters influencing a startup procedure are summarized in Tab. 2. A correct setting of FlexRay startup parameters is necessary to ensure fault-free cluster startup.

Table 2 Startup parameters summary

Parameter	Range
vColdstartInhibit	True | False
pdListenTimeout	1926 - 2567692 µT
cdCASRxLowMin	29 gdBit
gdCASRxLowMax	28 - 254 gdBit
gColdStartAttempt	2 - 31

Type of node

Test of the startup related parameters has to be distinguished by type of FlexRay node. Four node types are considered in following test. They are TT-D and TT-L, both in variants of coldstart or noncoldstart. Coldstart or noncoldstart node is determined by *vColdstartInhibit* parameter. The parameter is of Boolean type. True denotes ability to start communication (coldstart node) while False defines noncoldstart node. Coldstart node starts sending startup frame, which could be detected by the tester.

gColdStartAttempt

The test is relevant for TT-D coldstart node only, because TT-L coldstart never terminates coldstart attempt (it sends two startup and synchronization frames). Evaluation of *gColdStartAttempt* is possible by counting of received startup frames according to equation 3.

$$gColdStartAttempt = \frac{N_{RSF}}{N_{FCA}} \tag{3}$$

Where N_{RSF} is a number of received Startup frames
N_{FCA} is a number of startup frames per coldstart attempt defined by standard, equals to 5

Collision avoidance symbol

Collision avoidance symbol length shall be between *cdCASRxLowMin* and *gdCASRxLowMax*, otherwise it does not have to be recognized. The same idea as for wakeup pattern parameters measurement is used. If the collision avoidance symbol is not detected, the tested node tries to send collision avoidance symbol by itself. Iterative algorithm is used, where tester starts sending the collision avoidance symbol with a little bit lower length than the minimal permitted value for *cdCASRxLowMin* discovering. Next the

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

tester continues increasing collision avoidance symbol length to *gdCASRxLowMax.* The range of recognized symbol lengths is finally evaluated.

pdListenTimeout

Measurement of *pdListenTimeout* uses a property that *pdListenTimeout* timer is restarted when the idle state is recognized on the bus. After the timeout expiration the ECU sends a collision avoidance symbol. The time interval measured between collision avoidance symbols should be corrected by *cChannelIdleDelimiter* and *cdCASActionPointOffset* values (standard defined constants) that affect bus idle recognition and collision avoidance symbol transmission. The principle is shown in Figure 5 and expressed by equation 4, where *ts_CAS* means timestamp of CAS.

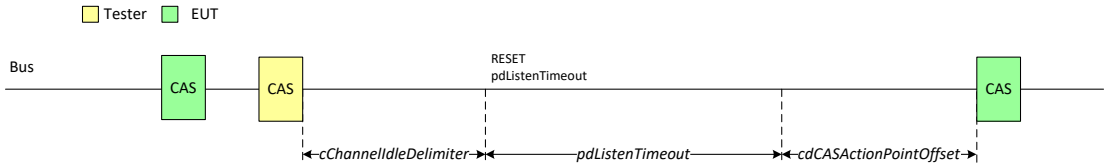

Figure 5: *pdListenTimeout* measurement

$$pdListenTimeout = ts\_CAS_{EUT} - ts\_CAS_{Tester} - cChannelIdleDelimiter - cdCASActionPointOffset \quad (4)$$

6. Evaluation of Clock Synchronization Parameters

Functionality of a FlexRay synchronization mechanism is influenced by four parameters. First parameter is local time unit microtick, which nominal value is specified by *pdMicrotick*. *pOffsetCorrectionOut* and *pRateCorrectionOut* are the maximal limit values for offset and rate part of clock correction. Rate correction is additionally reduced by *pClusterDriftDamping* parameter. Incorrect parametrization is difficult to reveal under the normal operating conditions. Example of such violation and its consequence is provided in section Motivation for evaluation of configuration parameters. Parameter names and their ranges are recapitulated in Tab. 3.

Table 3 List of measured parameters

Parameter	**Permitted range**
pdMicrotick (µT)	12.5 ns, 25 ns, 50 ns
pClusterDriftDamping	0 - 10 µT
pOffsetCorrectionOut	15 - 16082 µT
pRateCorrectionOut	3 - 3846 µT

Cycle length control and measurement

All presented methods are based on knowledge of communication cycle length (duration time). Cycle length is measured using a principle depicted in Figure 6. All incoming frames are marked by timestamp.

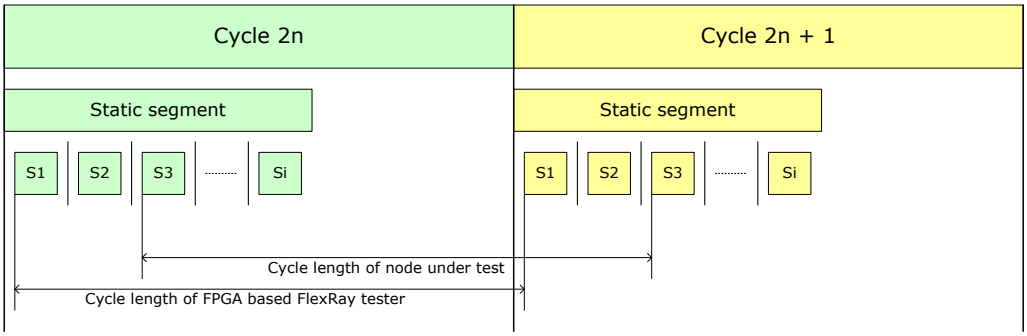

Figure 6: Cycle length measurement

EUT cycle length is synchronized by FTM algorithm. For three synchronization frames (two of them with the same cycle length are generated by the tester) two of three measured time deviations (the highest and the lowest one) are discarded. The EUT frame deviation is always zero and therefore it is either the highest or the lowest one (other two are the same). The remaining deviation value is always that generated by the tester and EUT is thus forced to follow its communication cycle length.

Offset correction measurement

Some methods presented later need to be aware of the value of the offset correction in particular communication cycle. Method for offset correction measurement published in [10] can be utilized. In our opinion, there is a mistype in formula in section III.C of [10], where the even and odd communication cycles are swapped. Offset correction is applied in odd communication cycle. To obtain offset correction with correct sign it is necessary to subtract even cycle length from odd cycle length. Presumption for proper use of the method is short term oscillator stability (constant rate correction). Magnitude of offset correction can be calculated according to the equation 5.

$$OC(2n+1) = CL(2n+1) - CL(2n) \tag{5}$$

Where *OC (2n+1)* is offset correction applied in odd communication cycle *(2n+1)*
CL (2n+1) is length of odd communication cycle
CL (2n) is length of previous even communication cycle

pdMicrotick

Usually *pdMicrotick* is directly derived from local clock source. Depending on desired communication speed, three nominal values are possible - 12.5 ns, 25 ns and 50 ns. Complete communication cycle schedule is derived from this parameter. The *pdMicrotick* parameter represents minimal possible change in the FlexRay node timing. The change is observable by a frame transmit time for frames transmitted in a static slot. Proposed method deals with the presumption that clock synchronization

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

works with microtick resolution and two FlexRay nodes are never synchronized absolutely. Cycle length is affected by components from equation 6.

$$CL(2n+1) = pMicroPerCycle + RC(2n+1) + OC(2n+1) \quad (6)$$

Where *CL (2n+1)* is length of odd communication cycle *(2n+1)*
RC (2n+1) is a Rate Correction value applied within the cycle
OC (2n+1) is an Offset Correction value applied within the cycle

A small difference between EUT and tester communication cycle lengths (denoted Δ) always exists. This small difference is being accumulated over few communication cycles until it exceeds magnitude of 1 µT (as shown in Figure 7). Within the following odd communication cycle the EUT synchronization mechanism corrects the difference by adding 1 µT to actual offset correction value. *pdMicrotick* value can thus be evaluated by precise measurement of EUT communication cycle lengths. Minimal distance in communication cycles histogram is equal to *pdMicrotick.*

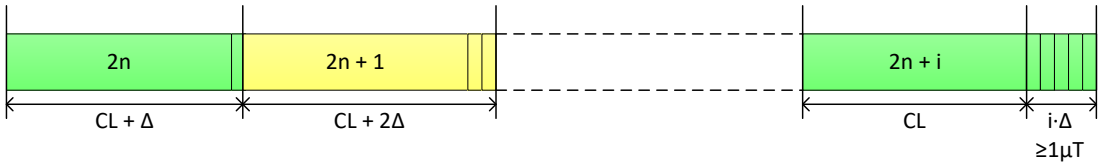

Figure 7: Cycle difference accumulation

pClusterDriftDamping

The parameter value is subtracted from the actual calculated value of rate correction. The *pClusterDriftDamping* can be interpreted as the insensitivity zone of rate correction. Clock frequency difference below this limit has to be corrected by offset correction (red squares in Figure 8). For example, if *pClusterDriftDamping* is equal to 5 µT and the clock frequency difference is higher than 5 µT per cycle, minimal applied offset correction is 10 µT each odd communication cycle.

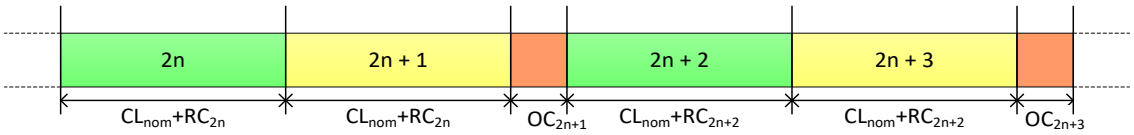

Figure 8: Offset correction affected by *pClusterDriftDamping*

For the parameter measurement it is necessary to assure that the tester and the EUT clock difference is higher than maximal permitted *pClusterDriftDamping* value, which is 10 µT. Afterwards, the value can be extracted from measured cycle lengths according to equation 7 (it does not reflect sporadic 1 µT corrections). Division by two reflects the accumulation of the parameter over two communication cycles.

$$pClusterDriftDumping = \frac{CL(2n+1) - CL(2n)}{2} \quad (7)$$

Where *CL (2n+1)* is length of odd communication cycle
CL (2n) is length of previous even communication cycle

pRateCorrectionOut

An idea for *pRateCorrectionOut* measurement is to slightly push the EUT synchronization mechanism to its limits. After the limit is reached, EUT stops sending frames. The measurement takes place in even communication cycles, since only even cycles are not affected by the offset correction. Maximal and minimal measured cycle lengths determine the value of *pRateCorrectionOut* according to equation 8. Either positive or negative Rate correction value is applied (lengthening and shortening of communication cycle). To obtain the actual value of the parameter it is necessary to divide the measured difference of cycle lengths by two.

$$pRateCorrectionOut = \frac{CL_{max} - CL_{min}}{2 \times pdMicrotick} \tag{8}$$

Where CL_{max} is maximal measured cycle length during pushing synchronization mechanism
CL_{min} is minimal measured cycle length during pushing synchronization mechanism

Figure 9 depicts this measurement step by step. First cycles are of the same length (EUT and tester are synchronized). Next, the tester starts forcing EUT to shorten its communication cycle using the principle described in section *Cycle length control and measurement*. When the lower limit is discovered, the tester starts with increasing the communication cycle length and the upper limit is discovered consequently. Missing communication is expressed by white bars in Figure 9.

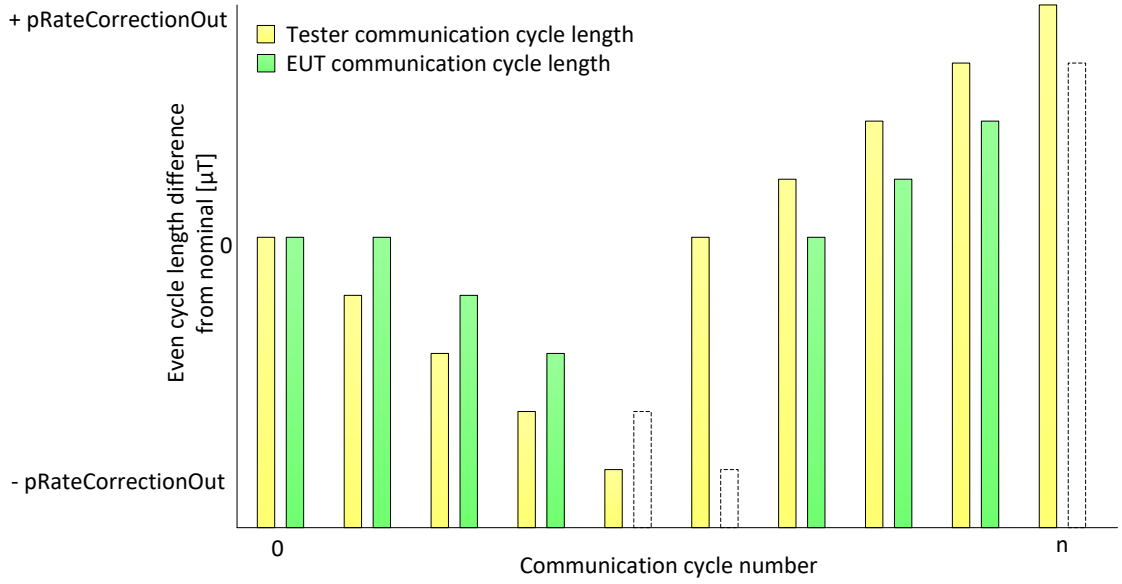

Figure 9: Rate correction affected by *pClusterDriftDamping*

pOffsetCorrectionOut

The offset correction is calculated using FTM algorithm from the deviations measured in each odd communication cycle. This value is applied at the end of the same communication cycle. Thus the offset correction value evaluation is only possible within the corresponding odd communication cycle. EUT offset correction is induced by fast shift of the tester's synchronization frames time position. This shift magnitude should be corrected by EUT offset correction, which value is measured using the method described in section *Offset correction measurement*. If the required offset correction value is higher than the *pOffsetCorrectionOut* limit, only this limit value is applied. This method is further limited by the actual duration of the static slot. The shifted test frame may not violate the static slot boundaries. Area of possible synchronization frames shifting is expressed by green shaded area in Figure 10.

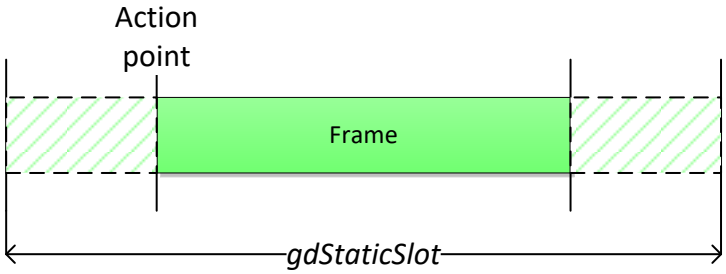

Figure 10: Offset correction measurement limitation

Maximal negative offset correction value (frame shift towards start of static slot) that can be measured is limited by *gdActionPointOffset* and is denoted by minus superscript in equation 9.

$$OC^{-}[\mu T] = gdActionPointOffset \times (gMacroPerCycle \div pMicroPerCycle) \quad (9)$$

Maximal positive offset correction value (a frame shift towards the end of static slot) that can be measured is limited by the frame length, *gdStaticSlot* and *gdActionPointOffset* and is denoted by plus superscript in equation 10.

$$\begin{aligned} OC^{+}[\mu T] &= (gdStaticSlot - gdActionPointOffset) \\ &\times (gMacroPerCycle \div pMicroPerCycle) - FrameLength \end{aligned}$$

$$\begin{aligned} FrameLength[\mu T] &= (gdTSSTransmitter + cdFSS + 80gdBit(header + trailer) \\ &+ 2 \times gPayloadLength \times 10 + cdFES) \\ &\times (cSamplesPerBit \div pSamplesPerMicrotick) \end{aligned} \quad (10)$$

Definition of all parameters used in equations above can be found in FlexRay standard [4].

Frame shifted over the limits is marked invalid by EUT internally and thus not used for synchronization. This behaviour can be used for evaluation of *gdActionPointOffset* and *gdStaticSlot* parameters.

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

7. Validation of measurement methods on real FlexRay network

Proposed methods were implemented as a part of developed FlexRay controller and tester. The controller is designed in VHDL and implemented on Altera Cyclone FPGA. The NIOS II microprocessor is used as a control element and runs testing methods implemented in C language. Two EUT types were used. The first is based on a 16 bit microprocessor MC9S12XF with integrated Freescale FlexRay controller. The second is based on 32 bit microprocessor TMS570LS31 from Texas Instruments with integrated Bosch FlexRay controller called E-Ray [17]. Both controllers are implemented according to specification version 2.1 [18]. The implementation of the FlexRay controller and design of startup testing methods was supported by the master thesis [19].

FlexRay network used for practical evaluation of presented methods was parameterized according to the Tab. 4. Parameters closely related to measurement methods are explained in corresponding paragraphs; detailed description of each parameter can be found in FlexRay Communication System Specification [4]. Testing network consisted of just two nodes (Tester and EUT) – see Figure 1.

Table 4 Test Network Configuration

Parameter	Value
gdActionPointOffset	3 MT
gdDynamicSlotIdlePhase	1
gdMinislot	40
gdStaticSlot	50
gdSymbolWindow	13 MT
gdTSSTransmitter	11 gdBit
gMacroPerCycle	5000 MT
gNumberOfMinislots	22
gNumberOfStaticSlots	60
gOffsetCorrectionStart	4920 MT
pClusterDriftDamping	1 µT
pDecodingCorrection	56 µT
pDelayCompensation[A]	1 µT
pDelayCompensation[B]	1 µT
pdListenTimeout	401202 µT
pLatestTx	21
pMacroInitialOffset[A]	5 MT
pMicroInitialOffset[A]	5 µT
pMacroInitialOffset[B]	23 MT
pMicroInitialOffset[B]	23 µT
pMicroPerCycle	200000 µT
pMicroPerMacroNom	40 µT

Measured values were compared with values configured in communication controller registers. As far as we know similar work with comparable results was not published. The results fully correspond with assumptions with respect to clock frequencies difference discussed in section *Measurement accuracy and speed*. To eliminate this issue, the validation tests are made using the tester clock frequency very close to clock frequency of the EUT (driven by arbitrary generator Tektronix AFG3102). The tester clock frequency was 80.0206 MHz for the ECU with E-Ray controller and 80.0011 MHz for the ECU with Freescale controller. Pre-set values of particular measured parameters are chosen from interval of all

possible values. Global FlexRay controller's settings are mentioned in Table 4 (with exception of measured parameters). Bus monitoring was done by Tektronix DPO4034 oscilloscope.

Table 5 Selected experimental results

Parameter	Actual Value	Measured on E-Ray	Measured on Freescale
gdCASRxLowMin	29 gdBit	29 gdBit	29 gdBit
gdCASRxLowMax	64 | 83 | 120 gdBit	64 | 83 | 120 gdBit	64 | 83 | 120 gdBit
pdListenTimeout [µT]	1926 | 401202 | 800000	1933 | 401211 | 800009	1934 | 401211 | 800010
pdMicrotick	25 ns	25 ns	25 ns
pClusterDriftDamping	1 | 3 | 10 µT	1 | 3 | 10 µT	1 | 3 | 10 µT
pRateCorrectionOut	500 | 600 | 700 µT	499| 600 | 699 µT	500 | 599 | 700 µT

Real measurement of parameters *gdCASRxLowMin* and *gdCASRxLowMax* shows that designed methods are able to identified actual values with bit level resolution precisely. Measurement of *pdListenTimeout* parameter after subtraction of CAS symbol length according to equation 4 contains an error of 10 µT or 250 ns maximally. This error is caused by combination of three factors. They are the remaining difference in tester and EUT clock frequencies, delay in receiving path of controller and tester time stamping implementation. The practical impact of this error on result usability is nevertheless negligible. Methods for *pdMicrotick* and *pClusterDriftDamping* identification work fully within expectations. Vital evaluation *pRateCorrectionOut* works according to presumptions.

Results of offset correction limits evaluation are summarized in the Table 6. Due to the limits represented by equations 9 and 10, two FlexRay network schedules were used. First schedule labelled as default setup is the setup previously mentioned in Table 4. Modified setup enables the full range measurement of permitted offset correction. Schedule of the static segment was changed to static slot length 100 MT, 30 static slots per communication cycle and action point offset 10 MT.

Table 6 pOffsetCorrectionOut measurement (all values in µT)

Set Value	Expected		Measured on E-Ray		Measured on Freescale	
pOffsetCorrectionOut	OC^+	OC^-	OC^+	OC^-	OC^+	OC^-
1201Default setup	224	120	230	130	230	128
1201 Modified setup	1201	400	1200	410	1201	410
300 Modified setup	300	300	300	300	300	300

Results correspond with presumptions with exception of static slot boundaries violation. Experiments demonstrate that the used controllers consider received frames valid approximately 10 µT before and after the defined static slot boundaries. All experimental results indicate that presented methods are able to evaluate critical parameters of a single FlexRay node.

8. Measurement accuracy and speed

Tester and EUT clock frequencies are never equal. There is always small difference between frequencies, usually few microticks per communication cycle in terms of FlexRay. Clock frequencies difference is necessary to take into account for presented parameters evaluation. It is assumed that difference of oscillator frequencies is at worst 1500 ppm from nominal value. This is the maximal permitted clock deviation according to standard [4]. For nominal 80 MHz clock frequency it is equal to deviation of up to 120 kHz. Quality of used oscillator should be subject of special testing.

Final paper at https://doi.org/10.1016/j.measurement.2016.12.051

Test objective is to reveal EUT parameters values from communication controller registers; therefore it is necessary to know EUT actual clock frequency or minimize clock differences between the tester and the EUT. Our experiments were done using the second approach. Before the measurement the clock source (Tektronix AFG3102) was set as close to the EUT actual clock frequency as possible. The tester clock frequency setting was done using observation of tester rate and offset correction actual values. Achieved clock difference was better than 1 µT per communication cycle. Explanation can be found in paragraph about *pdMicrotick* measurement. According to [14] the short term stability the measurement methods rely on is usually not the issue.

In the context of intended application measurement time is not critical. Moreover, it is short – typically in the range of few communication cycles. An overview of measurement times is stated to provide complete characteristics of the presented methods. Results are summarized in Tab. 7. Measurement time always depend on the parameter value and mostly on global network setup. Presented methods can be divided into two groups. Measurement algorithms from the first group work in a fixed number of steps. In the second group there are iterative algorithms, where the number of steps depends on parameter value and network setup. Measurement time estimation is provided in Tab. 7. in the column Approximate Time.

Table 7 Overview of measurement duration

Parameter	Range	Algorithm - number of steps	Approximate Time
pdListenTimeout [µT]	1926 – 2567692 µT	fixed	2× pdListenTimeout
cdCASRxLowMin + Max [gdBit]	29 - 254	Depends on cdCASRxLowMax	gdCASRxLowMax – cdCASRxLowMin pdListenTimeout
gColdStartAttempt	2 - 31	Depends on gColdStartAttempt	(5× pMicroPerCycle + pdListenTimeout) × gColdStartAttempt
pdMicrotick [ns]	12.5; 25; 50	Depend on clock difference	1.25 s♣
pClusterDriftDamping [µT]	0 - 10	Depend on pClusterDriftDamping	pMicroPerCycle × pClusterDriftDamping × 2
pOffsetCorrectionOut [µT]	15 - 16082	Fixed	2x pMicroPerCycle
pRateCorrectionOut [µT]	3 - 3846	Depend on pRateCorrectionOut	pRateCorrectionOut * 2 * pMicroPerCycle

♣ worst case time for experimental setup given in Tab. 4.

9. Conclusion

The aim of this work is to fill the gap in the area of FlexRay communication system available measurement and testing techniques. The paper proposes the novel complex set of methods for measurement of the data-link layer parameters of a FlexRay network node. Presented methods cover the parameters related to all three FlexRay node operational control states, i.e. the cluster wakeup, startup and normal operation. Individual measurement methods are described in detail; they were designed with respect to usability in a real world testing by means of a straightforward implementation and short execution time. Compared to the cited publications the new methods are based not only on

passive communication monitoring, but active stimuli are used as well. The validity of presented measurement methods is checked by experiments. Experiments have been conducted on real FlexRay ECUs based on two different FlexRay controller implementations (Freescale and Bosch E-Ray) to avoid implementation specific result. Commented experimental results with measurement setup and discussion of measurement accuracy are included. The experimental results prove the validity of all new methods. For *pOffsetCorrectionOut* measurement method, they simultaneously show its fundamental dependence on static slot configuration.

Acknowledgements

This work was supported by the Grant Agency of the Czech Technical University in Prague, project Model-Based Testing methods for automotive electronics systems (SGS16/171/OHK3/2T/13), and by the Technological Agency, Czech Republic, programme Centres of Competence, project # TE01020020 Josef Božek Competence Centre for Automotive Industry.

References

1. Sobotka, J. and Novák, J. "FlexRay controller with special testing capabilities"'2012 International Conference on Applied Electronics', 2012.
2. Shreejith, S., Fahmy, S. A. and Lukaseiwycz, M. "Accelerating validation of time-triggered automotive systems on FPGAs"'Field-Programmable Technology (FPT), 2013 International Conference on', 2013, pp. 4-11.
3. Obermaisser, R., ed. (2011), Time-Triggered Communication, CRC Press.
4. ISO 17458-2:2013 Road vehicles – FlexRay communications system – Part 2: Data link layer specification (January 2013)
5. Schmutzler, C., Simons, M. and Becker, J. "On demand dependent deactivation of automotive ECUs"'Design, Automation Test in Europe Conference Exhibition (DATE), 2012', 2012, pp. 69-74.
6. J. L. Welch and N. A. Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronization", Information and Computation, vol. 77, no. 1, pp. 1-36, April 1988.
7. Desogus, M., Reorda, M., Sterpone, L., Avantaggiati, V., Audisio, G. and Sabatini, M. "Validation and robustness assessment of an automotive system" 'Design and Test Symposium (IDT), 2013 8th International', 2013, pp. 1-6.
8. Novák, J.: New Measurement Method of Sample Point Position in Controller Area Network Nodes. Measurement. 2008, vol. 2008, no. 41, p. 300-306. ISSN 0263-2241.
9. Armengaud, E., Watzenig, D., Steger, C., Berger, H., Gall, H., Pfister, F. and Pistauer, M. "TEODACS : A new vision for testing dependable automotive communication systems"' Industrial Embedded Systems, 2008. SIES 2008. International Symposium on', 2008, pp. 257-260.
10. Armengaud, E. and Steininger, A. "Remote Measurement of Local Oscillator Drifts in FlexRay Networks" 'DATE: 2009 DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION, VOLS 1-3', IEEE, 345 E 47TH ST, NEW YORK, NY 10017 USA, Design, Automation and Test in Europe Conference and Exhibition, Nice, FRANCE, APR 20-24, 2009, 2009, pp. 1082-1087

11. Heinz, M., Hoss, V. and Muller-Glaser, K. "Physical Layer Extraction of FlexRay Configuration Parameters" 'Rapid System Prototyping, 2009. RSP '09. IEEE/IFIP International Symposium on', 2009, pp. 173-180.
12. ARMENGAUD, Eric; STEININGER, Andreas; HORAUER, Martin. Automatic Parameter Identi cation in FlexRay based Automotive Communication Networks. In: *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on*. IEEE, 2006. p. 897-904.
13. Sobotka, J. and Novák, J. "METHODS FOR MEASUREMENT OF FLEXRAY NODE BASIC TIMING PARAMETERS"'XX IMEKO World Congress', 2012.
14. H.-P. Company. Fundamentals of Quartz Oscillators. In HP Application Note 200-2, 1997.
15. Johnson, R., Evans, J., Jacobsen, P., Thompson, J. and Christopher, M. "The changing automotive environment: high-temperature electronics," *Electronics Packaging Manufacturing, IEEE Transactions on* (27:3), 2004, pp. 164-176.
16. ISO 17987-6:2016 Road vehicles -- Local Interconnect Network (LIN) -- Part 6: Protocol conformance test specification (October 2016)
17. "Semiconductors & Sensors." FlexRay Communication Controller IP. Robert Bosch GmbH, n.d. Web. 28 Aug. 2012. <http://www.bosch-semiconductors.de/en/ipmodules/flexray/flexray.asp>.
18. FlexRay Consortium, FlexRay Protocol Specification V2.1 Rev. A, 2005.
19. Martin Paták. Methods for testing the flexray start-up mechanism. Master's thesis, CTU in Prague, 2012.