CHALMERS

Development and Implementation of Test System for Electronic Road Toll Transponders

– Utveckling och Implementering av Test System för Elektroniska Vägtulls-transpondrar

Master of Science Thesis

ANDREAS ANDERSSON

Department of Signals and Systems
Division of Communication Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2009

Report No. EX 019/2009

Development and implementation of Test System for Electronic Road Toll Transponders
Andreas Andersson, 2007

Report nr: EX019/2009

Kapsch TrafficCom AB

Department of Signals and Systems
Communication Systems Group
Chalmers University of Technology
Göteborg, Sweden

Abstract

Within the automatic road toll systems exists several operators that today use the technique Dedicated Short Range Communication (DSRC) under Road Transport and Traffic Telematics (RTTT). To make it possible to use other operators Road Side Equipment (RSE) to communicate with other On Board Units (OBU), elaborate standards have been invented for the communication between road side equipments and OBU's by the institute ETSI. In order to accomplish the standard some test must be done to verify the requirements.

This is the report of thesis in Master of Science in Electrical Engineering where the aim is to investigate the procedure and evaluating test cases for OBU's. The object of this thesis was to examine the behaviour of the OBU partly when it was subject for corrupt frames and partly when it was subject for the timing demands. Furthermore this report evaluates the test cases that have been performed.

The thesis has been conducted at Kapsch TrafficCom AB in Jönköping, one of the market leading companies in electronic road toll systems. Kapsch TrafficCom AB is for many people known as Combitech Traffic Systems AB but they are since the year 2000 owned by the Austrian company Kapsch AG.

The outturn of this project was satisfying for the test cases controlling the reaction when the OBU was subject for invalid frames. The OBU behaved according to the expectations that are specified from the standard institute.

This is a public copy of the report. Parts of it have been removed at the discretion
of Kapsch TrafficCom AB.

Keywords
DSRC, EFC, ETSI, Interoperability, RTTT, TTCN

Table of Contents

List of Figures

Tables

Acknowledgements

There are several persons I would like to thank who have contributed to make this master's thesis possible. Without you this report would not have been what it is today. First I would like to thank my examiner at Chalmers, Erik Ström. At Kapsch I would like to thank my supervisor Staffan Bråndedal how has contributed with valuable knowledge and guidance, Jonny Mårtensson standing by with his wealth of knowledge within the DSRC technology and the functionality of the hardware. Many thanks at Kapsch also go to Erik Pettersson for valuable input to the project and together with Carl-Olov Carlsson for help with reviewing the report.

Special thanks to my family who helped and encourage me in good as well as in hard times through all the years, a special thought goes to my wonderful Mother. My thanks also go to all my great friends for their support.

Andreas Andersson
Jönköping, 18 August 2007

1 Introduction

In this chapter the focus will be on the background, purpose and the scope to give a clear understanding of the project. It will give a broad introduction to the subject of electronic road toll collection.

1.1 Background

Today, traffic jams in urban environments growing into a larger problem. Not only the time spent in endless queues but also the affect on the environment is a big issue. At many places world wide the introduction of automatic road toll systems have improved the environmental situation and reduced the lines of cars.

The demand of automatic road toll systems has during the last few years increased, partly because of regulation of traffic flow, partly because of private companies' requirements to finance parts of roads, bridges and tunnels that they have built without full economic compensation from government. Before the automatic systems, the toll systems usually collected the fees at toll plazas where every vehicle had to stop and pay before entering the road that was subject to charge. Those old toll plazas are more and more replaced by automatic toll systems that enable electronic registration and payment and possibilities of electronic enforcement for a complete free flow system. In other words, there are many places that are in need of automatic road toll systems to improve for motorists.

The reason why standards have been developed in automatic road toll systems is because different companies' products shall be interoperable with each other. European Committee for Standardization (CEN) is the organisation which handles the European standards. Three European companies have urged the question to establish a standard for automatic road toll systems where Kapsch is one of them. It is essential that all RSE have such design so they can handle all OBU, including versions which meet the minimum requirements.

1.2 Purpose

To be able to use equipment between different operators there is a demand of elaborated standards. These standards are organized by CEN, however they are created in consultation with concerned companies. It is immensely significant to know that new automatic road toll equipment follows standards in order for interoperability between systems. To verify the equipment observes the standard, tests which are gathered and agreed need to be fulfilled. A group with members from the concerned companies has gathered the test cases they consider a transponder has to manage to be considered as interoperable. The tests have the function of controlling the behaviour of the equipment when it is object of different frames. It must send correct frames as replies to both incorrect received frames and correct received frames.

Most of these gathered tests are already implemented and verified with an existing test system but some tests are not verified because of short-comings in the test equipment. The test cases that already have been compiled have mostly controlled the part of the frame that is built up by the software. Since there is no option to control the hardware, the test cases built on manipulating the part of a frame controlled by the hardware have to be done in another way and not in the same way as the other cases. Also the tests controlling the timing of frames can not be done with the existing system because of the high demand of time accuracy.

1.3 Project Outline

The outline of this project is divided into several parts, where the outcome is a finalised prototype and completion of the thesis. The parts are:

- Study company literature and other literature to gain knowledge about how the Electronic Fee Collection (EFC) systems work.
- Study the system the company uses today to run test cases and to understand the limits in this test system to try to find another solution for the test cases not executable in the present system. Also in this step, gathering knowledge about what kind of tests needs to be done to get an idea how to solve them.
- Implement a solution based on the gained knowledge to run the test cases that were not executable in present test system.
- Analyze the outcome from the implemented test cases to see if the behaviour of the transponder under test is as expected.
- Confirm that other companies' transponders operate and how they behave when they are in subject of the test cases.
- Document the results and the process in a written report for use by Kapsch TrafficCom, examiner at university and other interested.

Figure 1.1: Kapsch TrafficCom AB

2 DSRC Technology Overview

This chapter describes the DSRC technique and the organisations involved for the automatic road toll systems used in many places in Europe and countries around the world. This part also explains the different parts in a frame used in the communication between the RSE and OBU.

2.1 Organisations Handling the Standards

There are three organisations specially involved in the standards adopted by Kapsch in their EFC system. These three organisations are the European Telecommunications Standard Institute, European Committee for Standardisation and Swedish Standard Institute. Several other standardisation organs contributing in one way to have a functional EFC system can also be mentioned but to make it comprehensible only the below organisation are mentioned.

2.1.1 European Telecommunications Standards Institute (ETSI)

ETSI is an independent, non-profit standardisation organ for telecommunication, radio communication and other related topics. The members of the organisation are either full members and established in the geographical area of Europe as defined by the European Conference of Postal and Telecommunications Administrations (CEPT) or they are committed to ETSI work but without presence in the CEPT. Members can be national standards organizations (e.g. SIS), network operators (e.g. TeliaSonera) and manufactures (e.g. Kapsch TrafficCom AB, Ericsson AB) and they are totally more than 650 members from 59 countries.

CEPT was originally founded by the big monopoly postal and telecommunication companies of 19 countries but today there are several members. ETSI was founded in 1988 by CEPT to handle the standardisation issues of telecommunication and is today located in Sophia Antipolis (France). Apart from the DSRC standardization, ETSI also has been successful in standardizing GSM, DECT, DVB and many more technologies. Standards from CEN are usually not including associated measurement procedures for verification of the requirements but instead ETSI supplies test methods for conformity. Still today CEPT uses ETSI to handle the standardisation issues of telecommunication. [1]

2.1.2 European Committee for Standardisation (CEN)

CEN was created in 1961 and working among other things with technical standards for interoperability between systems. To have a standard finalized takes several steps and lot of negotiations between the concerned parties. The standards are at first drafts so called prENV, after the final vote for a new standard the standard becomes binding for all members and it is a European standard EN. When the standard also is set in Sweden the standard name also gets the prefix SS-EN. [2] The EFC equipment developed by Kapsch is compliant with several CEN standards for example the three layers in the context of RTTT.

2.1.3 Swedish Standard Institute (SIS)

SIS is a comprehensive institute for standards in Sweden. It is an impartial association with representatives from industry, authorities and civil society. They look out for Swedish interests in their cooperation with CEN and ISO and it is also SIS that sets Swedish standards. [3]

2.2 The Link

The link between the road side equipment and the on board unit in this EFC system is built upon DSRC. DSRC use three layers from the common Open Systems Interconnection model (OSI-model). The reason of using only tree layers in this kind of real time systems are because they are used in a small service area and under severe real time constraints. These layers in DSRC are;

- 1 Physical Layer
- 2 Data Link Layer
- 7 Application Layer

2.2.1 Physical Layer

The transceiver is the device mounted at toll plazas and it contains the antenna and logic to communicate by microwaves with transponders. The transceiver is a part of the RSE which for example also may contain back office system and enforcement system. The transponder is the device the end user has mounted in the vehicle to communicate with the transceivers at the toll plazas. If the transponder is capable to perform more than sending and receiving DSRC commands the correct name is On Board Unit (OBU). Other features for an OBU can for example be handling of a smart card or buttons and display to handle the OBU. In this project a transponder is the complete device of the OBU.

DSRC at 5,8 GHz uses four allowed carrier frequencies: 5,7975 GHz, 5,8025 GHz, 5,8075 GHz and 5,8125GHz. [6] The reason of several carrier frequencies is to minimize risk of interference at for example toll plazas consisting of several lanes. The space between two transceivers using the same carrier frequencies can be used by transceivers with the others carrier frequencies to optimize the toll plaza area.

The transponder uses the energy in transceiver carrier signal to reflect the energy and add its data combined with a sub-carrier frequency. Transponders using this technology where they are only using its internal power source for the transponder chip and uses the received signal energy for reflecting its own signal are called semi-passive. The technology of reusing the received signal energy is also called backscatter.

The sub-carrier frequency can either be 1,5 MHz or 2,0 MHz. The transceiver decides which of the two sub-carriers the transponder is allowed to transmit with so it is important the transponder can handle both sub-carrier frequencies. The sub-carrier makes it possible for the transceiver to distinguish between sent data and received data.

Data is coded in different ways depending if the data is transmitted from the transceiver or from the transponder. When describing data transmitted from the transceiver it is also defined as downlink and describing data transmitted from the transponder it is defined as uplink. In downlink frames the bits are FM0 coded. In a FM0 coded zero bit a transition occurs in the middle of a bit interval and in a FM0 coded one bit no transition occurs in the bit interval. It always occurs a transition at the beginning of a bit interval for both zeroes and ones. The method of mapping the uplink signal to a physical signal is using Non Return to Zero, Inverted (NRZI). Here a bit zero will cause a transition but a transmitted one bit will cause no change in the level between two bits. The NRZI method makes use of bit stuffing allowing the sending and receiving clock to be synchronized. [9]

Bit rate and modulation is also controlled of the physical layer. The bit rate for downlink is 500 kBit/s and for uplink 250 kBit/s. [7] The modulation for downlink is Amplitude Modulation (AM) where the information is in the amplitude of the carrier. AM transmitters and receivers have the advantage of being of a simple construction. For uplink Binary Phase Shift Keying (BPSK) is used. Of the different PSK modulations Binary PSK is the simplest one where the phase is shifted 180° in a transition. It is the most insensitive PSK modulation but the drawback is that BPSK only can modulate 1 bit per symbol. [10]

2.2.2 Data Link Layer

The Data Link Layer is divided into two sub-layers, Medium Access Control (MAC) and Logical Link Control (LLC). The MAC sub-layer handles the rules for channel access control and it inspects all received frames to verify their validity. A frame is valid if it fulfils following requirements:

- The frame is correctly delimited by start and end flag
- The frame contains (after removal of stuff bits) a number of bits equivalent to an integer number of octets
- The frame contains a valid link address field with appropriate Link Identification (LID)
- The frame contains a correct MAC control field
- The frame does not exceed the maximum number of octets
- The frame contains a valid Frame Check Sequence (FCS) field.

Otherwise if not all the conditions are fulfilled the frame shall be discarded.

If a frame pass all the criterions and the frame contains an Link Protocol Data Unit (LPDU), the following parts will be removed from the frame, start and end flag, the FCS, the link address field and the MAC control field before passing on the LPDU and content of link address field to the LLC sub-layer.

The LLC sub-layer uses the services from MAC sub-layer to provide services to the layer above, normally to the network layer but in the DSRC case to the Application layer.

The LLC layer performs following functions:

- Initiation of control signal interchange
- Organisation of data flow
- Interpretation of received command PDU’s and generation of a proper Protocol Data Unit (PDU) response
- Processing error control and error recovery functions in the LLC sub-layer

There are two types of LLC operations, connectionless or connection oriented. In the connectionless operation there is no guarantee the data unit will be correct delivered. To increase the certainty of the delivery when using connectionless operations there is an option to have acknowledged service so the delivery is guaranteed [8].

2.2.3 Application Layer

Application Layer forms and attends the part of frame called Application Protocol Data Unit (APDU). The application layer is split into three kernels; Transfer Kernel, Initialisation Kernel and Broadcast Kernel [9]. As the kernels names suggest the T-Kernel handles transfer, I-Kernel handles initialisation and B-Kernel handles broadcast issues. More specific the T-Kernel use services of the LLC sub-layer and together with the other kernels it performs commands such as GET, SET, ACTION, EVENT-REPORT and INITIALISATION that are more described in Global Specification for Short Range Communication (GSS) [5].

The I-Kernel implements the initialisation of the communication between the transponder and the RSE by exchanging information regarding which profiles and applications it can administrate. The I-Kernel of the RSE manages the transmission of the Beacon Service Table (BST) and the transponders I-Kernel evaluates the BST if it has passed the underlying layers, which means it is a new BST. With a correct BST the transponder I-Kernel choose a random LID to encapsulate in Private Window Request (PrWRq).

In the OBU and RSE it is the B-Kernel that handles the gathering and distribution of application information. The services are either to broadcast information to the peer application or to retrieve a given file with the broadcast data specified in the broadcast data service primitive.

2.3 DSRC Encapsulation

DSRC uses encapsulation to build a complete frame. Each frame consists of a number of segments that is encapsulated in every proper frame then there are a number of segments that varying dependent of the purpose of the sent frame. The header segments that are used in every frame have the function to declare where the frame starts and stops with flags. The header also indicates with a Frame Check Sequence (FCS) whether a frame has been corrupt because of bit error and a preamble for helping the receiver to get synchronized with the bit stream. The preamble, start and stop flag are static but the value in FCS and the other fields changes depending on the application data. In figure 2.1 shows an example of the structure of a DSRC frame.

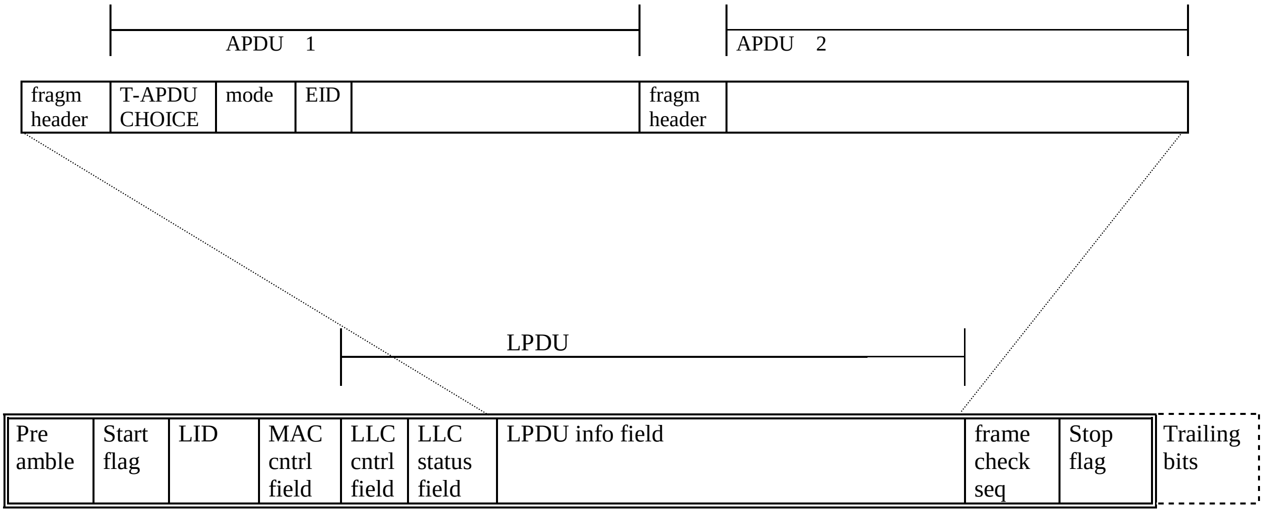


Figure 2.1: Frame format

2.4 Preamble Reference Bit Sequence

There are 2 types of preambles, one type in frames sent by the transponder and one type in frames sent from the transceiver. The preamble consists at the downlink of a sequence of 16 bits ±1 bit at the beginning of each frame. The transponder shall in other words be capable of handling a preamble between 15 to 17 bits. The preamble is set to ones and coded according to FM0 which means that for binary one a transition occurs at the beginning of every bit cell. This results in a sequence of alternating ones and zeros where each pulse has duration of 2 µs.

The reason of using a preamble is to synchronize the transponder with the downstream so the transponder can determine when bit boundaries occur. This procedure, also called self-clocking must be followed by a known bit sequence because the transponder does not know when it obtained synchronization.

The self-clocking procedure is also applied to the uplink but in a slightly different way. On the uplink the preamble consist of two parts, first 32 µs – 36 µs of unmodulated subcarrier, then eight “0” bits coded as NRZI.

2.5 Trailing Bits

The road side equipment is permitted to transmit 8 bits after end flag but since the on board unit is not required to accept those bits and none of the test cases involves these bits so no trailing bits are sent from transceiver in this project.

2.6 Frame Check Sequence

To discover if any bit error occurred on the link between the transmitter and the receiver a Cyclic Random Check (CRC) is used. The calculated value of CRC, its one complement is added as the FCS which is mandatory for all frames for downlink as well as for uplink frames. CRC is calculated to everything between the start flag and end flag, meaning the fields LID, MAC, LLC and LPDU are included. It is calculated through polynomial division to get the reminder as a 2 octet value which is used in the transmission. FCS field is the only part of the frame sending its MSB first and LSB last. The generator polynomial looks different depending which type of CRC calculation is applied. In DSRC the CRC – CCITT is used and it has the generator polynomial, $X^{16}+ X^{12}+X^5+1$ with initialisation value 0xFFFF. When the polynomial is translated to binary it becomes 10001000000100001 where each index in the binary number represents the exponent in the polynomial.

When computing a CRC it is through several divisions with generator polynomial, in this case $X^{16} + X^{12} + X^5 + 1$. It is the reminder that is added as the checksum. When the receiver control the checksum by the same computation the reminder now shall be zero otherwise a bit error occurred. There is one weakness with CRC computation, it can only detect errors, not correct them. To get an error correcting sequence the calculations will be more intricate and the overhead bits will increase. It is always a trade off which error detection/correction method to be used in signal communication.

An example of a CRC calculation over the made-up data 10111, is supposed to be transmitted. Data 10111 is translated to the polynomial $X^4+X^2+X^1+1$. As generator polynomial X^3+X^1 is used. In figure 2.2 the calculation is showed.

$$
\begin{array}{r l}
 & X^4 + X^1 + 1 \\
X^3 + X^1 \big) & \overline{X^7 + X^5 + X^4 + X^3} \\
 & \underline{X^7 + X^5 \quad \downarrow \quad} \downarrow \\
 & \qquad\quad X^4 \quad \downarrow \\
 & \qquad\quad \underline{X^4 \quad \downarrow + X^2} \\
 & \qquad\qquad\quad X^3 + X^2 \\
 & \qquad\qquad\quad \underline{X^3 \qquad + X^1} \\
 & \qquad\qquad\qquad\quad X^2 + X^1 = 110
\end{array}
$$

Figure 2.2: Example of CRC Calculation

The calculated quotient is concatenated together with data and the final frame to transmit for this example is, $10111_{\text{data}}\,110_{\text{CRC.}}$

CRC – CCITT is a common implementation for disk-controllers. The advantage of using CRC-16-CCITT is that the calculation can with ease be implemented both in hardware and software and still has good error detection strength.

2.7 Flag Bits

In every frame there is a start flag after the preamble and an end flag as the last octet in frame. Both the start and the end flag are built the same way and consist of one octet according to 0111 1110$_2$. These are the only times six binary ones in a row occurs in a frame, elsewhere it is prevented by bit stuffing which means after five ones in a row, an extra zero is added so no invalid flag can occur in the frame. The receiver inspect the sixth bit after five following ones, if the sixth bit is a one and the seventh bit is a zero it is interpret as flag but if the seventh bit also is a one, the frame is considered as invalid and is discarded.

2.8 Link Identification

The LID gives information whether the frame is of broadcast type, table 2.1, or private type, table 2.2. If the frame is private the LID contains four octets of information, randomly chosen, that makes it unique. The Broadcast LID is one octet and all bits are set to ones.

1	1	1	1	1	1	1	1
MSB							LSB

Table 2.1: Broadcast LID

The four octets of the Private LID are randomly chosen by the transponder except for the LSB's in each octet. It is the transponder that creates the private LID and transmits it the first time in Private Window Request (PrWRq). After that, the conversations between the road side equipment and the transponder uses the private LID to make sure only access a specific transponder. In table 2.2 bits marked with '*x*' represents a random produced bit and the last bit in each octet tells whether the following octet is a part of the LID or not. Zero as LSB meaning next octet is part of LID and one as LSB meaning end of LID.

x	x	x	x	x	x	x	0	octet #1
x	x	x	x	x	x	x	0	octet #2
x	x	x	x	x	x	x	0	octet #3
x	x	x	x	x	x	x	1	octet #4
MSB							LSB	

Table 2.2: Private LID

The LID can also be of multicast type and this type is reserved for test purposes and for private use. Multicast LID consists of one octet where the LSB indicates it is the last octet in the LID field. If the other bits in the octet are zeros the LID is for test purpose and it is for private use if the other seven bits forms the value in the reserved range from 120 to 126.

2.9 Medium Access Control Field

The MAC control field contains information about the content in the frame, and the type of frame and how the communication shall proceed. The bits in the MAC control field can be expressed as:

- L – specify whether an LLC part is contained in the frame (to allow also for empty LPDU frames)
- D – indicates the transmission direction: downlink or uplink
- A and R – allocation for private downlink and request for private uplink
- C/R – indicates if Command or Response
- S – MAC sequence bit (only downlink)

A MAC control field is one octet and there is a distinction made between downlink and uplink MAC control fields. Figure 2.3 shows how the bits are placed in the octet. The three LSB are reserved and shall be set to 0.

L	D	A_D R_U	C/R	S_D	-	-	-
MSB							LSB

Figure 2.3: MAC control field on downlink (D) and uplink (U)

2.10 Other Fields of the DSRC Frame

A frame can also include several other fields but those are of less importance for this project and will be mentioned concise.

LLC control field and LLC status field is one octet each. LLC control field indicates the type of command or response that is set from MAC control field. LLC status field is included in every acknowledge frame response and indicates success or failure of the processing command. The LPDU info field contains the data for service primitives. A service primitive can be an action, get, set, event report or initialisation; all of them are either request or response.

3 Initialisation Phase

Every road side transceiver is continuously transmitting frames with Beacon Service Table (BST) waiting for a transponder to enter the communication zone and respond with a private window request (PrWRq). The initialisation also contains the Private Window Allocation (PrWA) and the Vehicle Service Table (VST). All these type of frames will be further described in the report below. An overview of the communication between road side equipment and the on board equipment is illustrated in figure 3.1.

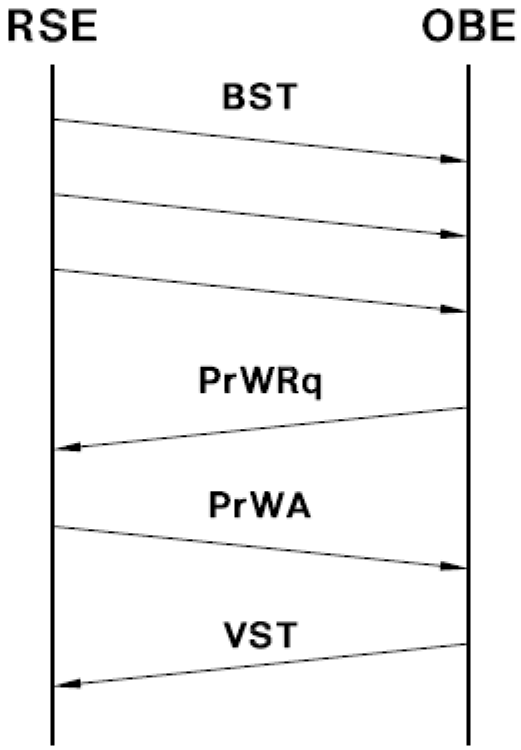


Figure 3.1: Initialisation of communication

It is in the initialisation phase most of this project's test cases are carried out. Therefore it is of importance to describe this phase for the understanding of the test cases. The content of the different frames described in this chapter are available in Appendix A; Content of Frames.

3.1 Wakeup Process

Since a transponder is in sleep mode most of the time to save battery power. The first BST reaching the transponder will wakeup the transponder from sleep mode. It does not have to be a BST frame to wakeup a transponder but any frame with a length of 11 octets shall be sufficient. The time from a wakeup frame to the transponder will be able to handle next BST shall be less than 5 ms. If it has not been exposed for frequencies of 5,8 GHz under last 100 ms it will return to sleep mode.

3.2 Evaluation of BST

Every BST contains information about which applications the road side equipment can handle and an ID telling which road side equipment has sent the BST. When a transponder receives a BST it checks the BeaconID and compares it with the last received BeaconID. If the BeaconID is the same as the last one, the transponder also checks the time elapsed between the BST's. If the time exceeds 255 s it is considered to be a new BST and the initialisation will continue otherwise the BST is discarded as an old BST where a transaction already has taken place and now will prevent a motorist to be charged twice and to save battery power. Before the transponder continue with communication it also check the profile and application list to see if the BST matching any of the profiles and applications in the transponder.

3.3 LID Creation

When a BST fulfil all conditions the transponder can choose from three allocated windows to send its PrWRq, see fig 3.2. It is by random which of the three windows, with equal chance, the transponder will transmit its PrWRq-frame in.

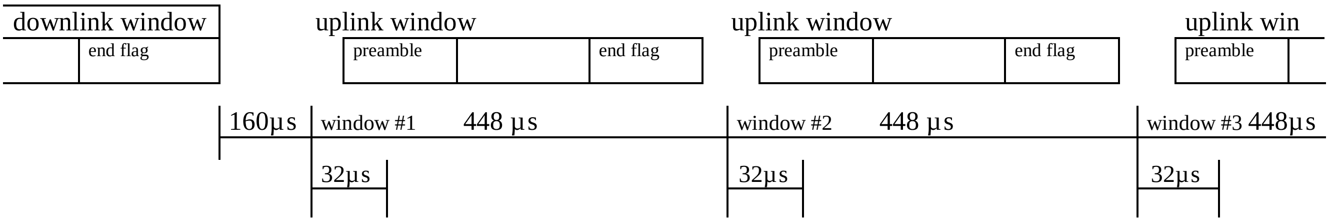


Figure 3.2: Public uplink window timing

The window allocation technique lowers the risk of two or several transponders to try to reply at the same time to one BST which cause interference. It is therefore of great importance the timing requirements are fulfilled to prevent overlapping. If two transponders reply in the same window allocation, no one will receive a Private Window Allocation (PrWA), instead the transceiver send another BST and both transponders will recalculate a new random value of which window it shall select and retransmit its request. Hopefully this time the allocation window will diverge and both will receive a PrWA each. For the transponder to know which PrWA is intended it uses a private Link Identification (LID). The road side equipment response to the on board unit with the same LID as the on board unit created at the PrWRq. In this way it is possible to address downlink frames to specific transponders.

As mentioned in chapter 2.8 the LID can either be of broadcast type or as in the case with PrWRq a private LID. The broadcast LID is used to address all transponders in the communication zone simultaneously and therefore only used in downlink frames such as BST and other public commands. Each of the four octets of the private LID has a least significant bit indicating if the octet is the last one or not. One as LSB indicates the next octet is part of the LID and a zero as LSB indicates it is the last octet of the LID. The other bits in a private LID are selected by means of a random choice. The probability for two transponders to creating the same private LID is 2^{-28}. The same LID is used through out the communication between the road side equipment and the on board unit. A new LID is created first if the transponder receives a BST containing a different Beacon ID saying it is another road side equipment. The transponder will also create a new LID if the time in the BST has exceeded 255 seconds since the previous BST. Remark, it is in the BST the time is contained and there is no clock in the transponder keeping time. The only way to see if the time is exceeded is by comparing the time from last BST with the new BST.

The good reason of using the PrWRq – PrWA procedure instead of just sending the private LID first in the VST is to be able to have short allocation times. The PrWRq and PrWA frames are much smaller than a VST resulting in a more effective retransmission time of BST.

3.4 Verify Application

When the private window procedure is successfully completed the initialisation is over and the communication can continue by the transponder transmitting a Vehicle Service Table (VST). Depending on what applications the road side equipment uses and transmits in its BST, the VST assembles the ApplicationEntityIDs which exists in the ApplicationList of the transponder and which are included in the ApplicationList of the BST. The VST provided by the OBU is used by the road side equipment to select the first EFC transaction protocol it is willing and capable to support. In one OBU several protocols can be supported. Different OBU protocols will send different VST's to the road side equipment. These VST's, although different in content, have the same structure. This structure consists of a header, a trailer and one or more data blocks in between.

3.5 Transaction and Release

After the road side equipment received a VST and got knowledge if the transponder will be able to accomplish the task the road side equipment want to do, the actual toll performance can take place.

In this project no toll service is performed instead there is an ECHO transmitted from the road side equipment after it received a VST frame. An ECHO command is a frame with a purpose to just make the transponder to send an ECHO frame back without any other computation. The transponder answers the ECHO by sending its own ECHO. It is the last frame sent in one of the test cases in this project. ECHO's with dummy data can be used for tracking the OBU until it has left the communication zone. An option to use tracking is to send a release frame which causes the transponder to power down. A release command will also make the LID invalid and communication with the same RSE will be possible first after 255 seconds. When using tracking it is not always the transponder receives a release command. In this instance the OBU keeps the transaction context (LID, random numbers, etc.) during the T-wait timer's duration of 255s

4 System Description

Before new transponder models are going to be released on the market they have to be tested to verify they follow the interoperability standard for DSRC. Kapsch TrafficCom has made many of the tests but not all of them because of limits in the existing test setup.

The task for this project was to find a solution of the non completed test cases. The reason why not all tests have been completed is because some tests require high resolution of timing measurements and some tests require manipulation of frames on bit level to simulate bit errors. These 2 demands from the test cases can not be achieved with the present test setup.

Except of being capable to handle a high timing resolution and having a high degree of frame manipulation on bit level, Kapsch TrafficCom also had a desire that the complementary test setup should be able to handle other companies' transponders.

The problems to overcome in this project can be summarised to:

- A system that should be generic to handle all kinds of DSRC transponders.
- Must have the ability of processing data in real-time.
- Having a high degree of control on bit level.

How these issues are solved is described in the following subchapters. Also other possible solutions are described and the reason why they where rejected as the final solution.

Further, this chapter describes both the existing test system and the new developed system for this project. To develop a test case system that is able to manage the difficulties that today's system could not handle there are several issues first to solve before starting up and minimize risking ending up in a dead end and having to start over. The major challenge was to find a real-time solution to overcome the severe time constraints involved in some test cases.

Another major challenge was to find a method to receive and process frames transmitted from the transponder in a way not involving the solutions used today which not have well enough real-time performance. All these issues are more or less hardware questions. The software regarding the implementation of the test cases involved programming issues but most of all to make sure the test cases were completely covered and all definitions and technical terms were understood and taken into account.

4.1 Former & Existing System

The earlier tests for conformance testing of the DSRC-link in a RTTT environment were performed with help of a program tool from Telelogic called Tree and Tabular Combined Notation version 2 (TTCN-2). Later versions use the same acronym but with a different meaning, Testing and Test Control Notation. The TTCN-2 program runs in a PC that is connected to a transceiver via a transparent link controller. Figure 4.1 shows the set up of the former test system.

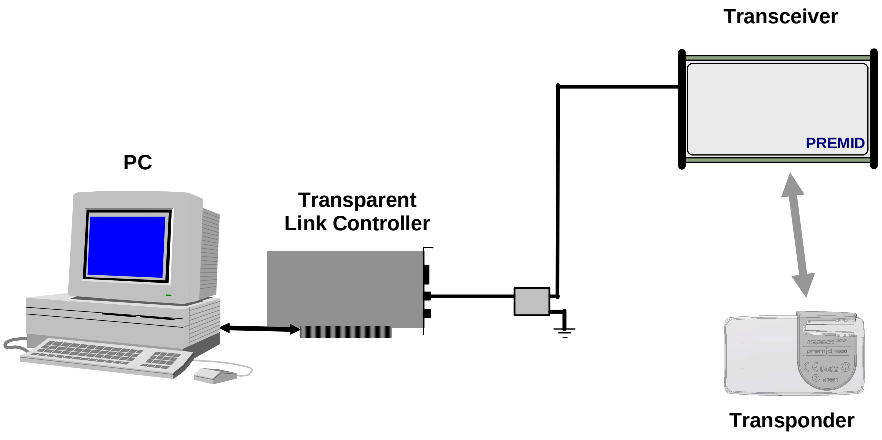


Figure 4.1: Former Existing system overview

TTCN is specially designed for the specification of tests of communication systems.[5] This test system generates executable *C* code over the ETSI test cases but to be able to communicate with a transceiver, an adaptation layer has been developed called Generic Compiler Interpreter (GCI) adaptation layer. The GCI layer has several functions. It initializes the Link Controller card (LiC-card) and converts downlink data from the TTCN environment to a format accepted by the LiC board drive and in the other direction it takes uplink data from LiC-card and transform to TTCN-format. Between the GCI layer and the LiC board there is a specially designed LiC driver that transmits data fully transparent except for flags and checksum. This is the part where this thesis project becomes useful. After the LiC board the data is transmitted by the transceiver to the transponder. Data are handled the same way in uplink but in the opposite order before data reaches TTCN runtime and a verdict can be decided. In this software tool are following parts assembled, LID, MAC, LLC and LPDU. It is in the transparent link controller the flags are added and the FCS is calculated. Since it is not possible to manipulate those parts of the frame, the test cases involving bit errors in flags or FCS will not be feasible with the former existing system. The figure below illustrates the TTCN system from implemented test case to the unit under test.

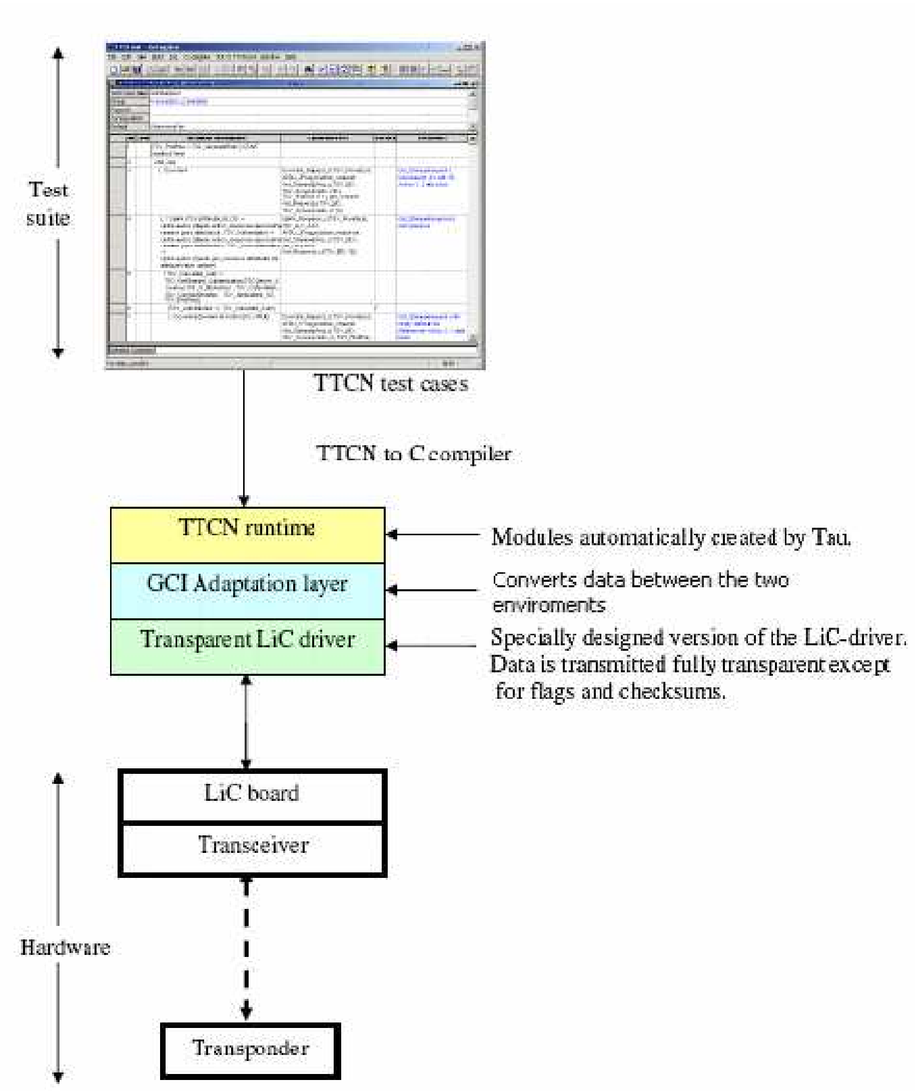

Test suite
TTCN test cases
TTCN to C compiler
TTCN runtime
GCI Adaptation layer
Transparent LiC driver
Modules automatically created by Tau.
Converts data between the two enviroments
Specially designed version of the LiC-driver. Data is transmitted fully transparent except for flags and checksums.
LiC board
Transceiver
Hardware
Transponder

Figure 4.2: TTCN chart

This test system performs tests to the three layers, Link Layer, Application Layer and MAC Layer. In every layer the tests are divided into two groups, valid test cases and invalid test cases. The valid test cases group the tests controlling how the transponder behaves when it is object of correct frames. It is the opposite with the group of invalid test cases that use frames containing errors to find out how the transponder behaves. By a command controlled interface the user choose which test to run, there are also some features in the program enabling log file, resending and to change output power level.

One great advantage of using agreed test behaviour is when having several suppliers of units to build one complete system. To be sure each unit under test (UUT) is compatible with each other, they have to be tested against each other:

UUT 1	↔	UUT 2
UUT 1	↔	UUT 3
UUT 1	↔	UUT 4
UUT 2	↔	UUT 3
UUT 2	↔	UUT 4
UUT 3	↔	UUT 4

For each new unit the number of tests increases dramatically. The number of tests can be calculated with the formula below where "*n*" is number of units.

$$Number\ of\ tests = \frac{n(n-1)}{2}$$, where n > 1

If instead each supplier use a test tool that every vendor have to pass the tests will be fewer. Only one extra test must be added for every new unit.

UUT 1	↔	Test Suite
UUT 2	↔	Test Suite
UUT 3	↔	Test Suite
UUT 4	↔	Test Suite

4.2 Possible Solutions

Before going into detail of how the selected system is working it is of interest to understand and get an overview of other possible solutions and their drawbacks, which are explaining why instead the implemented system was chosen.

Always when there is an existing solution that needs to be complemented there is a desire to reuse parts in the existing system to prevent "re-inventing the wheel again". For this project, the two main objectives, to have a high timing resolution and being able to control the transmitted frames on bit level, caused a complete new test setup. The existing test setup runs on a PC-platform using Windows as OS. This solution will not be able to handle the high timing demands. The existing system also uses a table transceiver TS3254/00A with a link controller card to connect to the PC. This solution is not able to handle incorrect frames since the link controller card only allows frames with correct Frame Check Sequences.

The choice of design is based on the facts of the limits in the present system and also based on what to achieve in the test cases. There are some basic elements that must be used in one way or the other for this project. These elements can be split down to:

- A transmitter to send DSRC frames.
- A receiver to receive DRSC frames.
- The logic to handle the communication involved in the test cases.
- A way to present the results for the tester.

All these elements must be able to work together to get a final test setup and they must fulfil the criteria of performance regarding time resolution and manipulations of bits. Since the regular PC using Windows XP as OS is out of the question because of its lack in time resolution, other OS were considered.

At the start of this project no idea existed how to solve the critical timing constraints. It was clear early in the project that no well-known operating system where able to give fair real-time performance. Windows XP has too many uncontrolled interrupts, Linux has resolution down to tenth of milliseconds. Also operating system as QNX known for its excellent real-time condition can not conduce to a solution since the time resolution with a few microseconds is not good enough in this project. Not having an operating system as base made the problem more intricate and the solution had to be found elsewhere. It is still necessary to have the logic to handle the communication so a MCU is a good replacement for the PC. Any other candidates where not considered as replacement for the handling of the communication since a MCU also will manage the possibility to make bit errors.

One way to transmit the signal is by a signal generator where modulation and frequency can be adjusted. There is one big disadvantage with a signal generator, it is only able to transmit signals but in this project receiving a signal, transmitted from the transponder is of a great importance. One solution to receive a signal is to have a transponder with an output from the transponder's transmitter and connect it physically to the MCU. This is not a desirable solution when interference has to be made to the transponder to complete a test case. The test system is more applicable if it can be used with only using DSRC to communicate with transponders. With a solution where no interference has to be done to the transponder it is also possible to test other companies' transponders where it is not possible to open the casing.

4.3 New & Supplementary Test System

This project's test system uses two programs to be able to execute the test cases of this project. The test system is arrayed according to picture 4.3 below with a PC connected to a Micro Control Unit (MCU) where the MCU communicate with the transceiver.

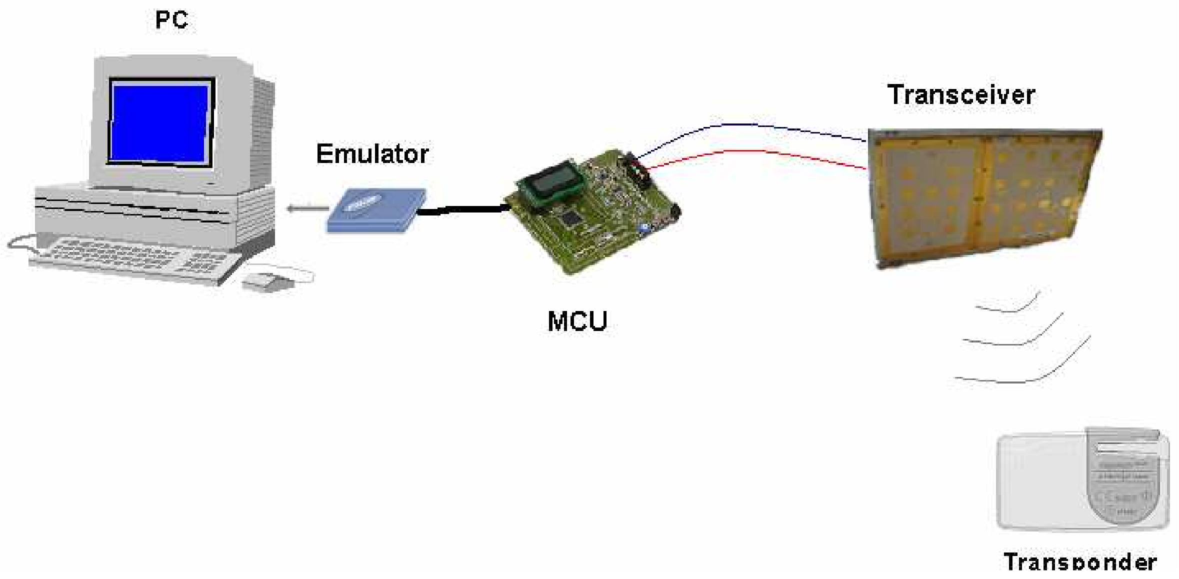


Figure 4.3: Project Test Set-up

4.4 Software

There are two developed programs in this project, *ETSI_Frame* and *ETSI_Test*. The first program, *ETSI_Frame* is a standalone program needed to get BST frames according to the ETSI test cases. The second program, *ETSI_Test* is the program running on the MCU to control the communication between the test equipment and the transponder and decide the verdict of the implemented test cases by using the BST frames created in *ETSI_Frame*.

For the development of the two programs the software, High-performance Embedded Workshop (HEW) and Visual Studio has been used. Both are development programs of *C++* executable code.

4.4.1 ETSI_Frame

This program generates a bit frame from the file "*ttcndata.txt*" which consists of data in form of a BST in hexadecimal, transported from the TTCN environment or written by the tester. In the program *ETSI_Frame* the users have the options to introduce errors according to the ETSI test cases to the frame before it is exported to the file "*rawdata.txt*". The program has some important subroutines in order to create a complete BST frame according to the chosen test case. The first subroutine is to display for the tester the different alternatives, showing the possible test cases to run. By choosing one of the alternatives "MAC_OBU_BI" or "MAC_OBU_BV" in the menu, depending if the BST frame should be subject for an invalid test case or a valid test case from the ETSI standard. The options will be followed by another menu where the tester chooses which of the test case in respective category to be executed. Test cases in the invalid test group make changes to the BST frame according to the ETSI test standard, however test cases in the valid test group only add a number to the end of the frame so when the frame is imported to "*ETSI_Test*" it knows which of the test cases to be implemented.

After a test case is selected, the file containing the TTCN created BST is read by the program. This BST is in hexadecimal format but to be able to make bit manipulations the BST is converted to binary values.

When creating the frames and introducing bit errors in the frame it is important that the invalid test cases not give rise to incorrect flags or abort sequences unless it is specified in the test case. An incorrect flag and abort sequence is if the random bit errors are at such positions in the frames that they creates five or more ones in a row in the frame. A control is made for those test cases involving random bit errors and if there is an incorrect flag or abort sequence because of the random bit errors the primary value will be restored and a new try is made until a frame is created with the present bit errors but without any invalid flags and abort sequences.

4.4.2 ETSI_Test

To get the transceiver to send the correct frame it needs to be fed by bits in appropriate order and with correct duration. The program *ETSI_Test* together with the MCU M16C29 handles this part. *ETSI_Test* also is establishing the communication between the transponder and the transceiver by receiving requests and to answer them accurately.

When the program starts it sends two BST's after switch 1 has been pressed on the MCU board. The BST is read from the text file "*rawdata.txt*" which contains the current BST that the program ETSI_Frame has created. The first BST is to wakeup the transponder and with the second one the transponder will take action. Every second time the program runs (without any reset) it sends the two identical BST's, first one for wakeup. It alternates between two BST's frames between each run because after successful connection with sent and received echo to the transponder, the transponder needs a BST from another "location" or a timeout must exceed 255 seconds. Since there is no clock built in that increases the time, the program sends a BST with another "location" every second time. After a BST is sent the program waits for a response from the transponder in form of a PrWRq. The PrWRq contains information regarding the LID that is important for a further communication with the transponder.

The MCU program has to remove stuff bits in the received frames between the start and stop flag in order to get a frame only consisting of transponder data. Depending which test case being executed and were in the communication phase the received frame is, it will be managed in different ways. Either it is just a control that the transponder transmits a reply or the received frame has information needed for continuous communication as the PrWRq has. Next step after having received the PrWRq is to get the LID and put it into the PrWA. The same LID is used through out the communication until another BST is transmitted and consequently another LID will be created by the transponder. In the frame of the PrWA the Private LID has to be concatenated together with the MAC Control Field and the Frame Check Sequence to create a complete frame.

Frames are received by waiting for the clock signal from the transceiver to become high. The high level on the transceiver's clock signal indicates when to read the transceiver's output signal which contains the frame data. A new bit is read every time the clock signal goes high levelled. This continues until the second flag has been read and indicates it is the last octet in the frame or if the received frame is of known length like the PrWRq the program stops reading from the data signal when all bits have arrived. There is no need to sample the incoming signal more than once per bit to get a satisfying result. The appended transceiver clock signal is very synchronous with the data bit stream so sampling once per bit gives 100 percent accuracy. See Appendix A, figure A.3 for an explaining flowchart of the receiver routine.

The most time critical operation is the transmitting function. It is of importance this function is implemented in a way making the bits appearing at the output with duration of 500 kbit/s FM0 coded signal. If the duration diverts more than ±100 ppm of 500 kbit/s the signal will not be recognised by the transponder. In Appendix A, figure A.3 the Send routine is illustrated as a flowchart.

The performance of the timing requirements is done by a clock in the MCU. The clock starts when the end of the second BST is transmitted. The clock is initiated to start at 0xFFFF and then count down by one for each clock cycle in the 20 MHz processor. The clock will time out and restart between the PrWRq is received and the PrWA is sent because the calculation and concatenation of both PrWA and ECHO are made before transmitting PrWA. This calculation consumes enough time to restart the timer. The time out is not a problem since there is no test case involving time measurement with a start time before PrWRq and a stop time after PrWA and therefore is no time calculation involving a time out. The timing interval is calculated by taking the difference between the time at test case start and the time of the test case end. The delay caused by the transceiver is removed in the calculation of the test cases affected of the delay before a final verdict is made. See Appendix A, figure A.5 for an explaining flowchart.

How deep in communication each run reaches is depending on which test case the tester wants to execute. Not all test cases use all commands in the initialisation. Actually most of the test cases have their verdict with only having the transceiver to transmit the BST and then evaluate the reaction of the transponder. The test case which goes furthest is one where transceiver and transponder sends and receives BST – PrWRq, PrWA – VST and ECHO - ECHO.

The result of the test is presented on a LCD (Liquid Crystal Display) with the text ”*VERDICT PASSED*” if test was successful or “*VERDICT FAILED*” if test was unsuccessful. The result can also be “*VERDICT NONE*” if a test case not will be fully executed for example if the transponder does not reply to a frame in one of the valid test cases. After final verdict, tester can run the test program again by pushing switch 1 on the MCU after each run.

A simplified flowchart of these both programs are described in Appendix A; Flowchart.

4.4.3 High-performance Embedded Workshop

The program HEW with a graphical user interface is the development environment for debugging and creations of executable code for Renesas micro control units. The program includes debugging and compilers for C/C++. It also works as the users link between PC and micro controller to download the assembled code.

4.4.4 Visual Studio

The program “*ETSI_Frame*” is developed in Microsoft’s software development product Visual Studio that allows creations of stand alone programs.

4.5 Hardware

In this project a PC, a transceiver, a microcontroller M16C29, an oscilloscope and a transponder is used to accomplish the implementation of the test cases. Also a frequency generator and a signal generator were first used to achieve and verify up to milestones in the project. The oscilloscope, frequency generator and the signal generator are not described any further in the report.

4.5.1 Personal Computer (PC)

The test application “*ETSI_Frame*” is implemented on a PC running under OS Windows XP. The PC needs a USB port to connect the micro computer and a serial port to connect to the transceiver.

4.5.2 M16C29

By using a micro control unit giving a good control over interrupts the real-time performance problem can be overcome if the processor of the MCU has high enough clock frequency. Renesas is providing a micro controller with a processor speed of 20 MHz and the other requirements as timers and I/O-ports are satisfied. Some efforts had to be done to sort out how Renesas MCU works. With enough knowledge to build a small test program using an output on the MCU to send alternating logical ones and zeros, a confirmation could be done that theory match practical case to obtain a bit duration of 2 µs. With this very important information the work could continue of using the MCU.

An oscilloscope was connected to the output when a signal was generated from the MCU. By correcting in the program’s output signal function with assembler instruction “NOP” (No Operation) a very accurate signal could be achieved. The NOP instruction only consumes clock cycles and does not affect any registers, status flags or memory, therefore is NOP suitable for timing tasks. The idea of the system is that the MCU also is able to receive a signal arose from the transponder. In the test phase of the MCU input, a frequency generator was used. The frequency generator was set to square pulse with duration of approximately 2 µs. No high accuracy of the pulses had to be made since the objective of this test was to confirm the ability of the MCU to receive a signal.

The micro control unit from Renesas is a 16-bit single-chip microcomputer with 1M bytes of address space. To be able to program the micro controller it is connected to a USB port at the PC through Renesas emulator E8. The verdict of the test cases is represented on the LCD, a 2-line by 8-character display (with a KS0066 controller IC). Three user pushbuttons are placed to the target board where one of them are used for execute the test case. The CPU has a shortest instruction time of 50 ns which corresponds to 20 MHz. It is important that the CPU is fast enough to manage the high bit rate of 500 kBits/s in this project. Other features of high value for this project are an accurate clock and timer function. The clock circuit with highest precision use the main clock which is 20 MHz and therefore has a precision down to 50 ns. There are eight 16-bit timers divided into two timer groups, one with five timers and one with three timers. Only one 16-bit timer is necessary to achieve the timing interval, since none of the test cases exceed 3276 µs that the timer at most can be before it restarts. The MCU is equipped with several I/O-ports, two of them are used as in-ports and one of them are used as out-port. The two in-ports are signals from the transceiver, one is the 4 µs clock signal and the second is the data signal. From the micro controller, one output signal with data is connected to the transceiver. The output signal has an amplitude of 5 V.

4.5.3 Transceiver

In the final test solution there is a modified transceiver controlling the transmitting and receiving of the signals. Before the transceiver was brought into use, a signal generator handled the transmitting of signals to a transponder with a soldered output of the replied signals from the transponder.

With the signal generator and the modified transponder with an output of the transmitting signal connected to an oscilloscope, confirmation could be done that with the MCU connected to the signal generator a DSRC signal was possible to transmit and the transponder replied. A way to receive a transmitted signal from any transponder is to utilise one of Kapsch TrafficCom’s transceivers. The transceivers produced for customers, house no inputs for external signals or external outputs for received signals. Kapsch TrafficCom though, access a few transceivers for laboratory research with ability to connect an extern source and send received signals from an output.

TS 3251/00A is a single lane transceiver and a part of the PREMID TS3200/00 road side system. The transceiver is connected to a Link Controller Card (LiC). The LiC card is installed in the PC and handles all messages. It is operating at 5,8 GHz for use in applications of the standardised DSRC microwave link. The transceiver used for this project is specialised in the way that it has options to get I/O signals to/from the transceiver. The input signal to the transceiver is a FM0-coded signal with a bit rate of 500 kBits/s and amplitude of 5 Volt. The output signals are a clock signal and data signal. Both output signals have an amplitude of approximately 5 Volt. The Data signal has a bit rate of 250 kBits/s and clock signal has a period time of 4 µs. Only frames meeting DSRC layer 7 requirements will be passed to the signal output. Therefore no action has to be taken by the MCU concerning noise and filtering of input signals.

The received frame from the transceiver is without preamble. When transmitting there is a slightly delay of outgoing signals but it is only tens of a nanosecond so it is negligible. The received signal delay is larger and has to be taken into account in the time measurement test cases. There is 75 µs delay of received signals to the transceiver before it reaches the output and can be clocked into the MCU. The largest part of the delay will be due to the fact the transceiver does not let through the preamble consisting of 64 to 68 µs and the preamble shall be included in the test cases. The other part of the delay, remaining 11 µs originate from the logic in the transceiver. At incoming transmissions the transceiver is first looking for a correct preamble from the transponder consisting of both parts. First part where 32 µs – 36 µs of unmodulated subcarrier and second part where 8 bits of NRZI coded “0” bits. With a correct preamble the transceiver starts looking for the start flag. With these conditions fulfilled the transceiver start to let through the sequent bits including start flag. It is these procedures and the removing of the NRZI-coding that causes the delay. The measured delay is displayed in Appendix, Screen Shots.

To control the transceiver a software program called TRX User Terminal is used to communicate with the transceiver through the serial port in the PC. The TRX User Terminal setups and initiate the transceiver when it starts up. In this program the user can control downlink parameters such as carrier frequency and output power. It is also possible to view error-codes from the transceiver in this program.

4.5.4 Transponder

For the development of the test system a TS3203/10A transponder has mostly been used. It is the transponder without tamper detection but with the analog Application Specific Integrated Circuit (ASIC) called Ella 1.1 and the digital ASIC called Alex 1.0. For the actual tests the transponder TS3203/10B has been evaluated. The difference between the two transponders is in the chipset where the transponder with version B has the upgraded Ella 2.1 and Alex 1.1. Any transponder working under RTTT and DSRC shall be suitable for the tests. To verify this, a number of different transponders also have been used from other companies.

5 Test Cases

All test cases are parted from the tree layers MAC, LLC and Application Layer. Each layer is divided into two categories, one treats the cases that controls that the OBU can handles the requirements stated in the standard. The other category manages the test cases that controls that the OBU behaves according to the standard at reception of an incorrect frame. The test cases discussed in this project are carried out from the MAC Layer and the following subheadings are the names of the test cases from the ETSI test specification. This chapter explains how each test case shall be performed and how the test cases are implemented.

5.1 Valid Test Cases

The Valid Test Cases use correct frames to confirm the correctness of the OBU. The OBU is in sleep mode when these test cases start, meaning the transponder need a wakeup frame before it is able to receive further data. In test cases involving timing evaluation, the preamble removed by the transceiver is considered to be 16 bits and therefore 64 µs are added to the start time in the timing result.

5.1.1 TC/MAC/OBU/BV/02

This test case is to verify that the OBU can receive a downlink frame within 32 µs after the end of the last transmitted frame from the same OBU. This test uses a delay starting after last bit of the VST is received from the OBU and stops before first bit of the echo frame is send from the RSE. The delay is supposed to allow exactly 32 µs to elapse. Since there is a delay in the transceiver of received frames of 11 µs the additional delay is 21 µs so the total time between end of VST to start of transmitting ECHO-frame is 32 µs, the delay to transmit a frame is negligible. The final verdict is passed if the transceiver receives an ECHO from the OBU.

5.1.2 TC/MAC/OBU/BV/03

Verify the OBU sends a PrWRq in one of the three uplink windows. T3, T4b and T5 are time definitions stated in European standard EN 12795 as 160 µs, 32 µs and 448 µs respectively. The allowed time interval is stated in table 5.1.

Public Uplink Window Number	Start Time Ts	End Time Te
1	T3<Ts<T3+T4b	Te<T3+T5
2	T3+T5<Ts<T3+T5+T4b	Te<T3+2*T5
3	T3+2*T5<Ts<T3+2*T5+T4b	Te<T3+3*T5

Table 5.1: Uplink windows

With the time values inserted the table will look like:

Public Uplink Window Number	Start Time Ts	End Time Te
1	160µs <Ts<192µs	Te<608µs
2	608µs <Ts<640µs	Te<1056µs
3	1056µs <Ts<1088µs	Te<1505µs

Table 5.2: Uplink windows with values inserted

The timer starts after the last bit in the BST is transmitted from the RSE. Then the time is read both when the first bit in PrWRq arrives and when the last bit in PrWRq arrives. The difference between the end of the BST to the start of the PrWRq gives Ts (Start Time) and the difference between the end of the BST to the end of the PrWRq gives Te (End Time). Those values shall be in the range given in table 5.2. Because of the delay and the absence of preamble in the received signal from transceiver this has been taken into account when calculating Ts and Te. For Ts the time of an estimated preamble of 64 µs is added together with the transceiver delay and for Te only the transceiver delay is added.

5.1.3 TC/MAC/OBU/BV/04

Verify that the OBU can receive a downlink frame that arrives immediately after another downlink frame. No timing is necessary in this test case since it is only a control of the OBU is able to receive a BST transmitted 0µs after a downlink frame with a private LID has been sent. This case is treated with a different frame containing both a BST with a private LID and an ordinary BST in the same array to manage the time condition. The text file containing this double BST is named "*BSTPrLIDBST.txt*". With this solution the first bit of the preamble in the ordinary BST will be transmitted without delay after the last bit in end flag of the BST with private LID. The test case will pass if a PrWRq is sent from the OBU.

5.1.4 TC/MAC/OBU/BV/05

Verify the OBU follows timing requirements to start transmitting and end transmitting a VST after it has received a PrWA. To calculate the allowed start and end time, the used equation includes the following time coefficients T3 and T4a that are stated in EN 12795 as 160 µs and 320 µs. N3 is the maximum number of octets in a frame in the downlink window. N3 is 128 octets, U8 is the bit rate of the uplink which is 250 kbit/s and U13a is a preamble time length of 32 µs to 36 µs plus another 32 µs so totally 64µs to 68 µs. The timer starts after the last bit in the PrWA is transmitted then the time is read both when first bit of the VST arrives and when the last bit of the VST arrives. The difference between the end of the PrWA and the start of the VST is Ts and the difference between the end of the PrWA and the end of the VST is Te. Also in this test case as for BV/03, the delay and absence of preamble in the signal from transceiver to the MCU is taken into account when calculating Ts and Te. The time values shall be within the range in table 5.3.

	Start Time Ts	End Time Te
Profile numbers	T3<Ts<T3+T4a	Te<T3+T4a+U13a+(8*N3+”Stuff Bits”)/U8
Values inserted	160µs<Ts<480µs	Te<(MIN)4640µs Te<(MAX)4744µs

Table 5.3: VST timing constraints

Depending if there are stuff bits or not and if it is a preamble of 16 or 17 bits, the value Te can be up to 4744 µs. The value of number of stuff bits is calculated from 126 octets since octets for start and stop flag never undergoes bit stuffing. At most there can be 20% stuff bits added to a frame between start and stop flag. With 126 octets there will be 25 extra bits to take in consideration when calculating Te. Since no calculation is done for the number of stuff bits of the arrived frame in the implementation, Te(MIN) is chosen. If the OBU is able to send a VST within Te(MIN) it will also be able to transmit before Te(MAX) exceeds. The test case passes if the VST follows the timing constraints.

5.1.5 TC/MAC/OBU/BV/08

Verify the OBU by uniform distribution chose randomly one of the public uplink windows. The timer works as in TC/MAC/OBU/BV/02 where the timer starts at end of the BST and reads the time both at the start and the end of the PrWRq sent from OBU. Ts and Te are calculated as in TC/MAC/OBU/BV/02 where the difference between the time at end of the BST and the time at the start of the PrWRq is calculated as Ts. The time Te is the difference between the end of the BST and the end of the PrWRq. Those values shall be within the criteria stated in table 5.2. If the PrWRq’s are sent in the allocated windows and it is at least 30% in each window the test pass. The tester is allowed to increase the number of runs to get the result to converge to at least 30% allocations for each window.

5.2 *Invalid Test Cases*

These test cases send manipulated frames to confirm the correctness of the OBU. All these test cases have their errors in the BST except for the test case number 7 which have a faulty PrWA. The test cases pass if the transponder does not reply to a corrupt frame. The OBU is in sleep mode when these test cases start.

5.2.1 TC/MAC/OBU/BI/01

Introduce a double bit error in between the end of the start flag and the start of the Frame Check Sequence. The double bit error is created by random procedure where two places in the frame in the mentioned area change value. The two error-bits do not necessarily have to being neighbouring but they are not allowed to give rise to any faulty flags or abort sequences.

5.2.2 TC/MAC/OBU/BI/02

Introduce a double bit error in the Frame Check Sequence by changing the value in a random process of two bits before transmission. The two error-bits do not necessarily have to being neighbouring but they are not allowed to give rise to any faulty flags or abort sequences.

5.2.3 TC/MAC/OBU/BI/03

Introduce a block of 15 consecutive bit errors to simulate a burst error somewhere between the start flag and the stop flag. The block with 15 bit errors in a row starts at a random position in the frame. If the random start number is less than 15 bits from the stop flag the remaining of the bits are changed prior the random start number. This is to prevent the stop flag from getting any bit errors. The block with bit errors must be placed in such a place it will not produce any flags or abort sequences.

5.2.4 TC/MAC/OBU/BI/04

Modify the BST so the Beacon ID contains a sequence of seven consecutive ones. The Beacon ID shall not undergo bit stuffing but the FCS shall be correct regarding the seven ones in the Beacon ID. Seven ones in a row are interpreted as an abort sequence. This test case uses a special BST with a Beacon ID according to the above specifications. It is not specified if the abort sequence shall be in manufacturer ID, individual ID or in both as the seven consecutive ones are placed at a random start index in Beacon ID.

5.2.5 TC/MAC/OBU/BI/05

Modify BST so the stop flag is replaced with an abort sequence of seven consecutive ones. The stop flag is not subject for bit stuffing. Instead of a sequence of six ones in a row, the LSB in the end flag is changed to a logical one.

5.2.6 TC/MAC/OBU/BI/07

After having received the PrWRq, the RSE shall send a PrWA with a private LID = LID1 but it will be consisting of 5 octets. In the PrWA the private LID is extended to 5 octets instead of usually 4 octets. The new manipulated LID will look like in figure 5.4 where the grey marked bits shows the added bits to the LID. The bits marked with "x" are the ones sent from the OBU in the PrWRq.

x	x	x	x	x	x	x	0	octet #1
x	x	x	x	x	x	x	0	octet #2
x	x	x	x	x	x	x	0	octet #3
x	x	x	x	x	x	x	0	octet #4
y	y	y	y	y	y	y	1	octet #5
MSB							LSB	

Table 5.4: Extend private LID

The LSB in octet 4 is changed from one to zero since it is now with a zero indicating that it is not the last octet in the LID field. Instead the LSB in octet 5 is set to one and the other bits are chosen by random. The test case passes if the transponder does not reply with a VST to the PrWA.

5.2.7 TC/MAC/OBU/BI/16

Create a signal so the BST is blocked over 15 bits between the start flag and the stop flag. To represent a blocked signal a low level is set as input to the transceiver over the 15 bits. This is not to be confused with a FM0 represented "0" that contains a transition in the middle and between every bit. The test case passes if the transponder rejects the frame and does not respond with a PrWRq.

5.2.8 TC/MAC/OBU/BI/17

Create a signal so the BST is blocked over the stop flag. The blocked part of the frame is represented as in TC/MAC/OBU/BI/16. The test case passes if the transponder rejects the frame and not responds with a PrWRq.

5.2.9 TC/MAC/OBU/BI/18

Create a signal so the BST is blocked over the start flag. The blocked part of the frame is represented as in TC/MAC/OBU/BI/16. The test case passes if the transponder rejects the frame and not responds with a PrWRq.

6 Outcome of Project Test Cases

The results of the test cases are probably the most interesting part of the report for the EFC developers. After all test cases have been implemented and executed a final verdict has been possible to detect. This part includes the result of Kapsch new transponder TS3203/10B but also the result from three other companies' transponders.

6.1 Kapsch TS3203/10B

(This section is intentionally left blank)

6.2 Other Companies' Transponder

Within the scope of this project, includes tests of competitor companies' transponders. Three transponders from three different companies were used to give a good comparison of the ETSI tests to Kapsch transponder TS3203/10B.

(This section is intentionally left blank)

6.3 Verification of Test System

To verify that the test setup gives correct results, all test cases have been verified by using an oscilloscope. The oscilloscope was connected between the MCU and the transceiver to record the signals in-between them. This means the frames created by the MCU were recorded before it reached the transceiver and the frames received to the transceiver from the transponder were recorded when they were sent from the transceiver to the MCU. In this way it was possible to confirm both the timing and the bit accuracy of the sent and received frames. The readouts from the oscilloscope showing the bit sequences together with debugging of the developed programs verified the correctness of the received and transmitted frames.

To verify the used transceiver only amplitude modulate, another measurement was made. After a frame is sent from the MCU to the transceiver it is assumed the transceiver only amplitude modulate the signal without changing bit duration or adding or removing bits in the frame. To verify the correctness of the transceiver, a simple test was performed where the received signal of the transponder where recorded and compared with the transmitted signal of the MCU. The result confirmed the assumed behaviour of the transceiver's handling of sending frames to the transponder.

7 Conclusions

Conformance testing is not an every day job instead the tests are usually done in the end of the development or just before the releasing of a new product. With the program developed in this project it is possible with only minor changes to test how the transponders behaves when they are in subject of all kinds of incorrect frames and to control the timing of frames during the initialisation phase.

It is a disadvantage of having two test set ups instead of being able to just carry out all tests at one test station. Since the test are not executed in the daily work rather when new products are going to be released and verified the issue of using two test set ups is not a great exertion.

During this project knowledge brought from university have been utilised but also a lot new knowledge have been gained. Both technical knowledge about how the DSRC system works and how to program a micro controller have been studied. Also practical elements such as estimate time requirements for different steps in the project have been useful.

This project shows, not only it is possible to run test cases, but it is also possible to send and receive frames built according to DSRC standard using a MCU and a transceiver. With the technique of connecting the programmable MCU with the transceiver it is possible to construct frames in a completely manually way and transmit them over the DSRC link. Kapsch TrafficCom can now make tests to transponders and they have a method for the test cases not covered by the former test set up but covered in this project.

(This section is intentionally left blank)

8 Future Work

Still there can be improvements to make to the software. Mostly it is regarding to make it more user-friendly. This was a compromise of the time being available and between flexibility and the ease of use of the program. All test cases for this project are fully implemented but additional tests can be added to improve the functionality. For instance a test controlling there is a valid DSRC transponder within the antenna range before the test case start. This can be done by having a test verifying a PrWRq was received after transceiver had sent the BST before continuing the test case. This is especially important for the test cases involving corrupt frames. Also options of retransmitting frames for the test cases using valid frames to control the transponder would improve functionality.

Add a choice for tester to see supplementary information stored in a file to be viewed after a test case been executed is also a possible improvement. In that way to see more specifically what went wrong in test cases were final verdict is failed or to gain knowledge about the transponder behaviour. These are easy tasks to implement since a lot of the information is already handled of the program, just not stored.

This project does not cover all of the test cases which are not covered by TTCN runtime environment. Some of them are easy implemented with this new testing tool especially those controlling the DSRC initialisation but also test cases where communication further than initialisation is involved can be implemented as long the test case does not involve heavy calculations like access credentials.

Abbreviations and Acronyms

AM	Amplitude Modulated
APDU	Application Protocol Data Unit
ASIC	Application Specific Integrated Circuit
BPSK	Binary Phase Shift Keying
BST	Beacon Service Table
CEN	European Committee for Standardization
CRC	Cyclic Random Check
DSRC	Dedicated Short Range Communication
EID	Element ID
ETSI	European Telecommunications Standard Institute
FCS	Frame Check Sequence
GCI	Generic Compiler Interpreter
GSS	Global Specification for Short range communication
HEW	High-performance Embedded Workshop
LCD	Liquid Crystal Display
LiC	Link Controller
LID	Link ID
LLC	Logic Link Control
LPDU	Link Protocol Data Unit
MAC	Medium Access Control
MCU	Micro Control Unit
NRZI	Non Return to Zero Inverted
OBU	On Board Unit – Same as transponder
OSI	Open Systems Interconnection model
PDU	Protocol Data Unit
PrWa	Private Window Allocation
PrWRq	Private Window Request
RSE	Road Side Equipment
RTTT	Road Transport and Traffic Telematics
TDMA	Time Division Multiple Access
Te	End Time
TRP	Transponder
Ts	Start Time
TTCN	Testing and Test Control Notation
UUT	Unit Under Test
VST	Vehicle Service Table

Reference

Internet References

[1] *The European Telecommunications Standards Institute* (2007)
http://www.etsi.org/about_etsi/30_minutes/home.htm (2007-05-10)

[2] *European Committee for standardization* (2007)
http://www.cen.eu/cenorm/aboutus/index.asp (2007-02-08)

[3] *Swedish Standard Institute* (2007)
http://www.sis.se/DesktopDefault.aspx?tabId=21&menuItemID=12
(2007-01-09)

[4] *Testing and Test Control Notation* (2007)
http://www.ttcn-3.org/WhatisT3.htm (2007-01-08)

Report and Book References

[5] *"GSS Global Specification for Short Range Communication*" Version 3.2

[6] EN 12253:2004 E *Road transport and traffic telematics – DSRC Physical Layer using microwave at 5,8 GHz*

[7] EN 12795:2003 E *Road transport and traffic telematics – DSRC Data Link Layer: Medium access and logical link control*

[8] EN 12834:2003 E *Road transport and traffic telematics – DSRC Application Layer*

[9] John G. Proakis, Masoud Salehi, "*Communication Systems Engineering*" Second Edition, Prentice Hall, United States of America, 2002

[10] Theodore S. Rappaport, "*Wireless Communication Principles and Practice*" Second Edition, Prentice Hall, United States of America, 2002

Appendix

Content of Frames

BST

octets	MSB LSB	Description
#1	0111 1110	start flag
#2	1111 1111	Broadcast LID (1 octet)
#3	1010 0000	MAC control field
#4	0000 0011	LLC control field
#5	1xxx x001	fragmentation header (xxx x = PDU number, ≠ 0000 and ≠ 0001)
#6	1000	T-APDU: Initialisation-Request (1st element in CHOICE codes with 0!)
	0	no nonmandApplications
	000	BeaconID.manufacturerid = 1
#7	0000 0000	
#8	0000 1	
	001	BeaconID.individualid = 0x1234567 = 19088743
#9	0010 0011	
#10	0100 0101	
#11	0110 0111	
#12	0011 0010	time = 8514720001 = 0x32C06E81
#13	1100 0000	
#14	0110 1110	
#15	1000 0001	
#16	0000 0001	profile = 1
#17	0000 0001	number of mand. app. = 1
#18	0	no eid
	0	no parameter
	00 0001	aid = 1
#19	0000 0000	profileList, empty
#20	xxxx xxxx	FCS (2 octets)
#21	xxxx xxxx	
#22	0111 1110	end flag

Table A.1: Beacon Service Table Description

PrWRq

octets	MSB LSB	Description
#1	0111 1110	start flag
#2	xxxx xxx0	private LID (4 octets)
#3	xxxx xxx0	
#4	xxxx xxx0	
#5	xxxx xxx1	
#6	0110 0000	MAC control field
#7	xxxx xxxx	FCS (2 octets)
#8	xxxx xxxx	
#9	0111 1110	end flag

Table A.2: Private Window Request Description

PrWA

octets	MSB LSB	Description
#1	0111 1110	start flag
#2	xxxx xxx1	private LID (4 octets)
#3	xxxx xxx1	
#4	xxxx xxx1	
#5	xxxx xxx0	
#6	0010 0000	MAC control field
#7	xxxx xxxx	FCS (2 octets)
#8	xxxx xxxx	
#9	0111 1110	end flag

Table A.3: Private Window Allocation Description

VST

octets	MSB LSB	Description
#1	0111 1110	Start flag
#2	xxxx xxx0	Private LID (4 octets)
#3	xxxx xxx0	
#4	xxxx xxx0	
#5	xxxx xxx1	
#6	1100 0000	MAC control field
#7	0000 0011	LLC control field
#8	1001 0001	Fragmentation header
#9	1001 0000	T-APDU: initialisation.response
#10	0000 000p	Profile (shall match to Profile used in BST)
#11	0000 0001	Number of supported applications = 1
#12	1	eid present
	1	parameter present
	00 0001	aid = 1 for EFC-applications
#13	eeee eeee	eid for addressing the application element
#14	0000 0010	parameter is type OCTET STRING
#15	0001 0000	length of parameter = 16 octets (or 12 optional)
#16	1000 0111	EFC-Context Mark
#17	0000 0000	
#18	0000 0001	
#19	0000 0000	
#20	0000 0001	
#21	0000 0001	
#22	0000 0010	Container.type = OCTET STRING
#23	0000 0010	Container.length = 2 octets
#24	xxxx xxxx	OBUGroupID
#25	xxxx xxxx	
#26	0000 0010	Container.type = OCTET STRING (optional)
#27	0000 0100	Container.length = 2 octets
#28	nnnn nnnn	RndOBU(optional)
#29	nnnn nnnn	
#30	nnnn nnnn	
#31	nnnn nnnn	
#32	1	OBUStatus present
	xxx xxxx	EquipmentClass
#33	xxxx xxxx	
#34	mmmm mmmm	manufacturerID
#35	mmmm mmmm	
#36	X	OBUStatus.absenceOfICC
	X	OBUStatus.unrecognisedICC
	X	OBUStatus.batterieFailure
	X	OBUStatus.peripheralInterfaceError
	X	OBUStatus.illegalTamperingOfTheOBU
	000	last state before sleeping state was reached: 000 = BLOCKED; 001 = WAIT 010 = INIT; 011 = READY 100 = DATA reserved for future use
#37	xxxx xxxx	reserved for private use
#38	xxxx xxxx	FCS (2 octets)
#39	xxxx xxxx	
#40	0111 1110	end flag

Table A.4: Vehicle Service Table Description

ECHO Request

octets	MSB LSB	Description
#1	0111 1110	Start Flag
#2 #3 #4 #5	xxxx xxx0 xxxx xxx0 xxxx xxx0 xxxx xxx1	private LID (4 octets)
#6	1010 0000	MAC control field
#7	0110 0111	LLC control field

#8
#9
#10
#11
#12
#13
	1001 0001
	0000 0101
	0000 0000
	0000 1111
	0000 0010
	0000 0000
		LPDU (6 octets)
		T-APDU
		EID
		Action Type 0x0F = ECHO
		Container
		Length of Data String

#14 #15	xxxx xxxx xxxx xxxx	FCS (2 octets)
#16	0111 1110	End Flag

Table A.5: ECHO Request Description

Flowcharts

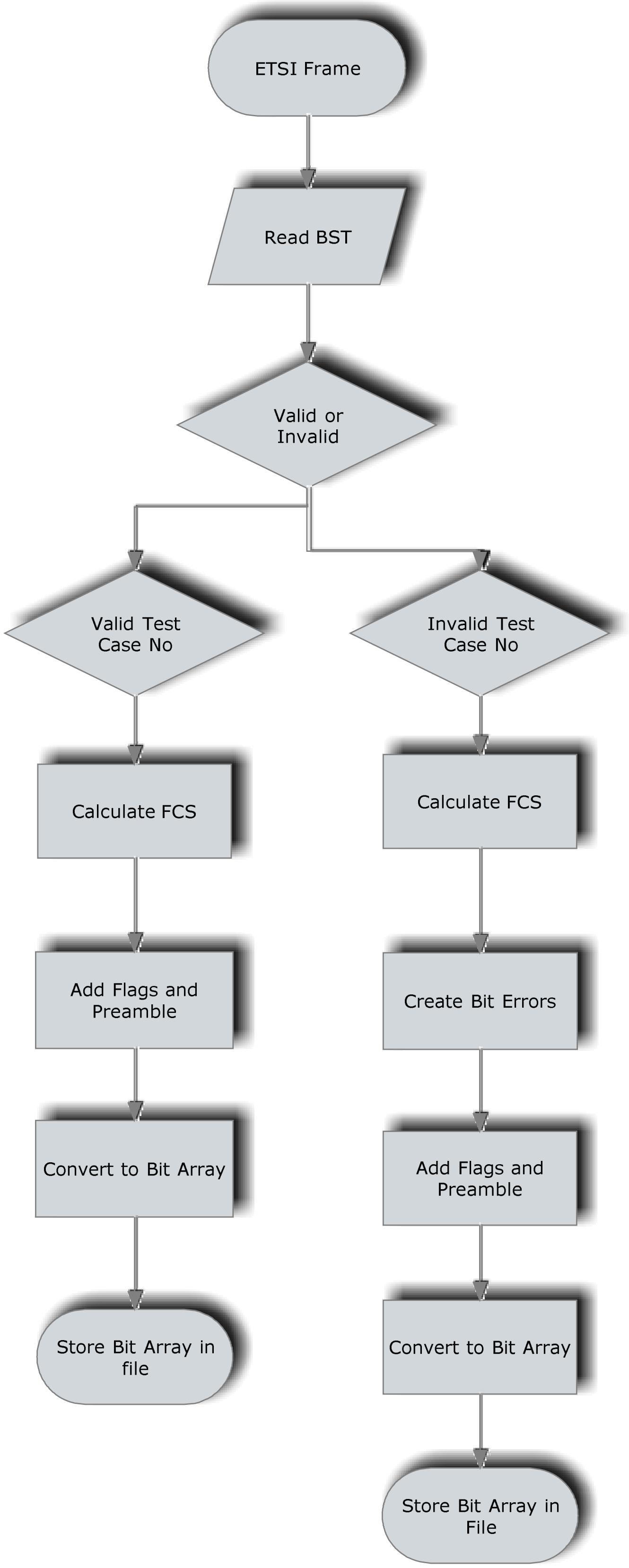


Figure A.1: Simplified flowchart over ETSI_Frame

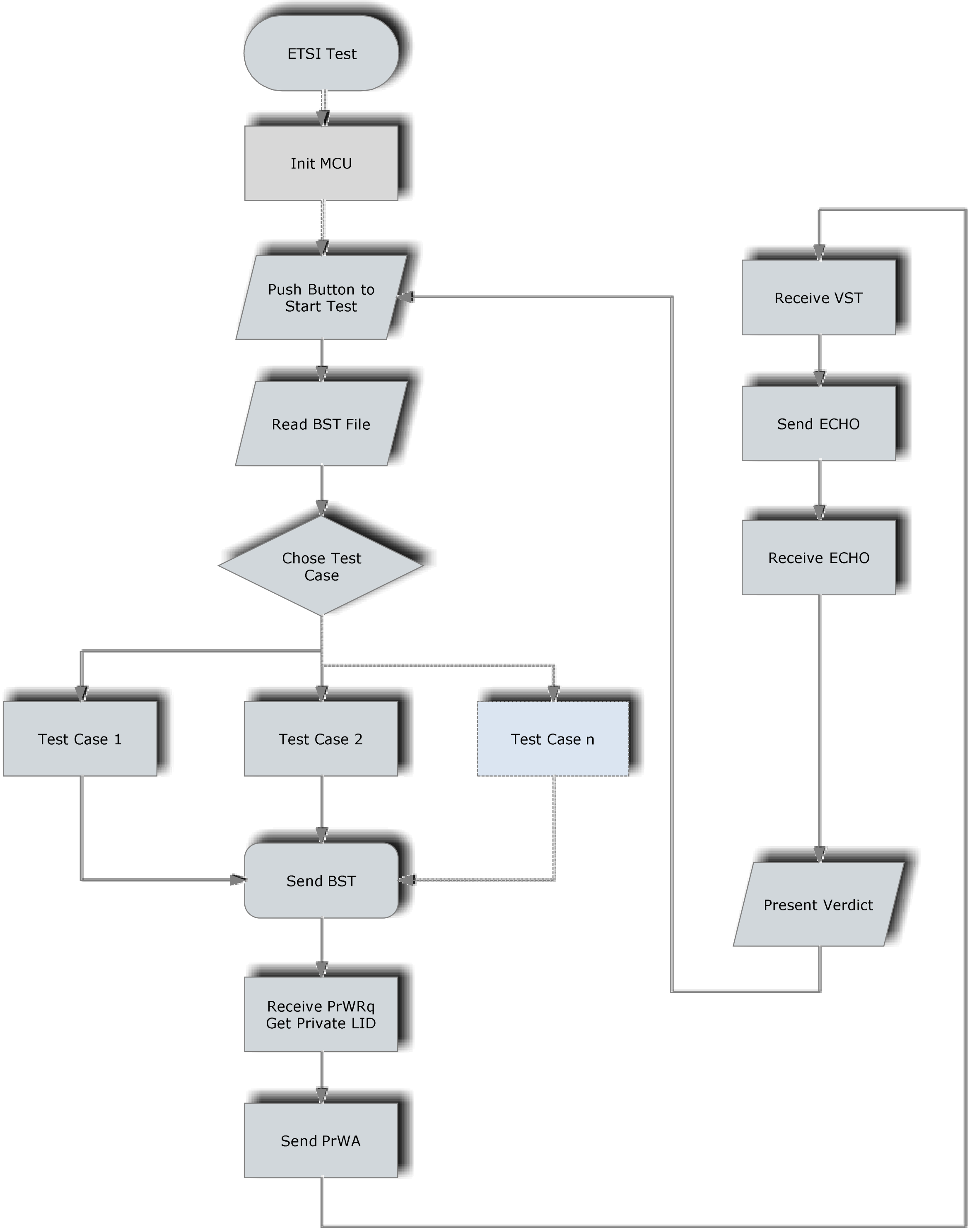


Figure A.2: Simplified flowchart over ETSI_Test

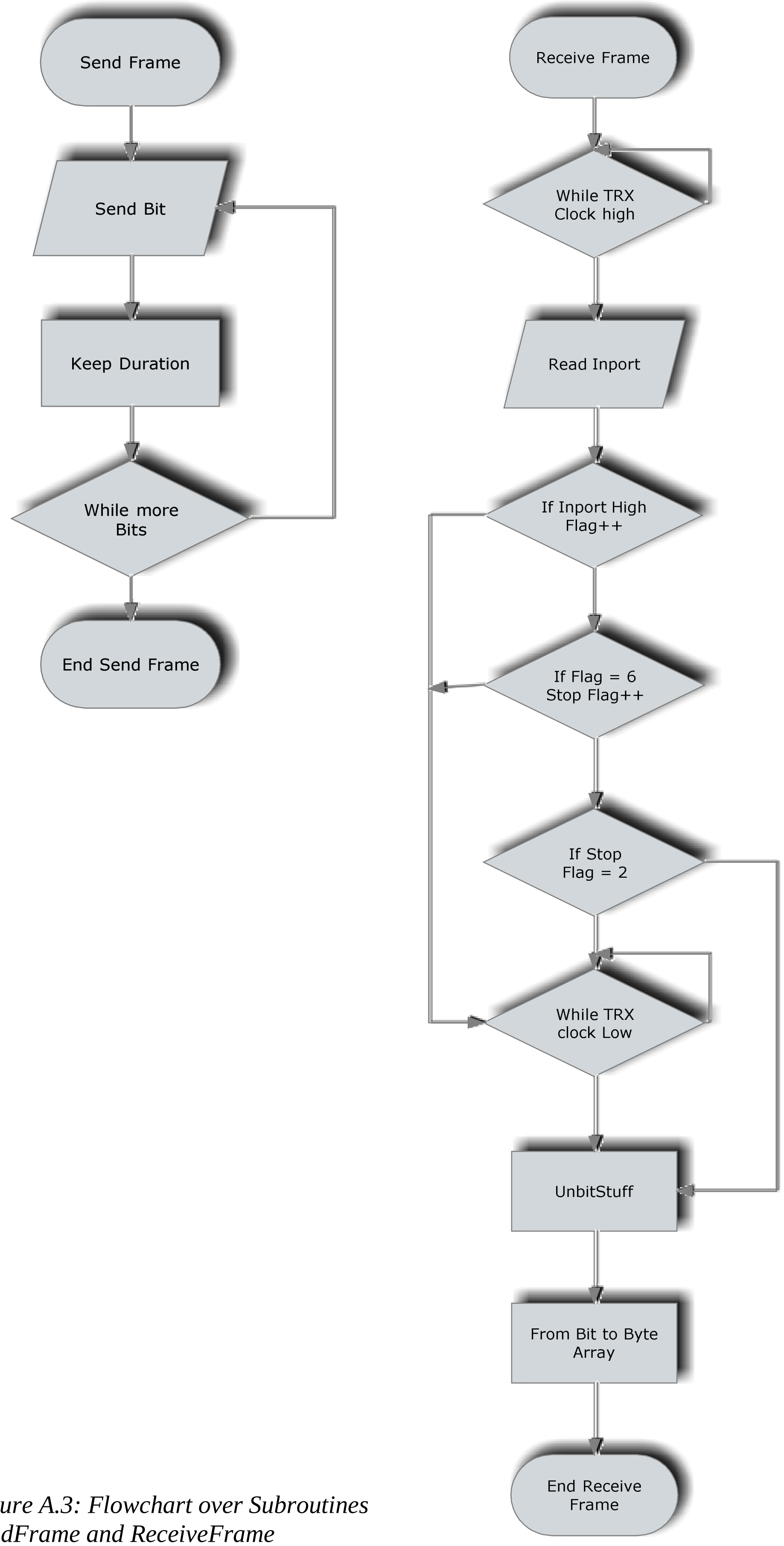


Figure A.3: Flowchart over Subroutines SendFrame and ReceiveFrame

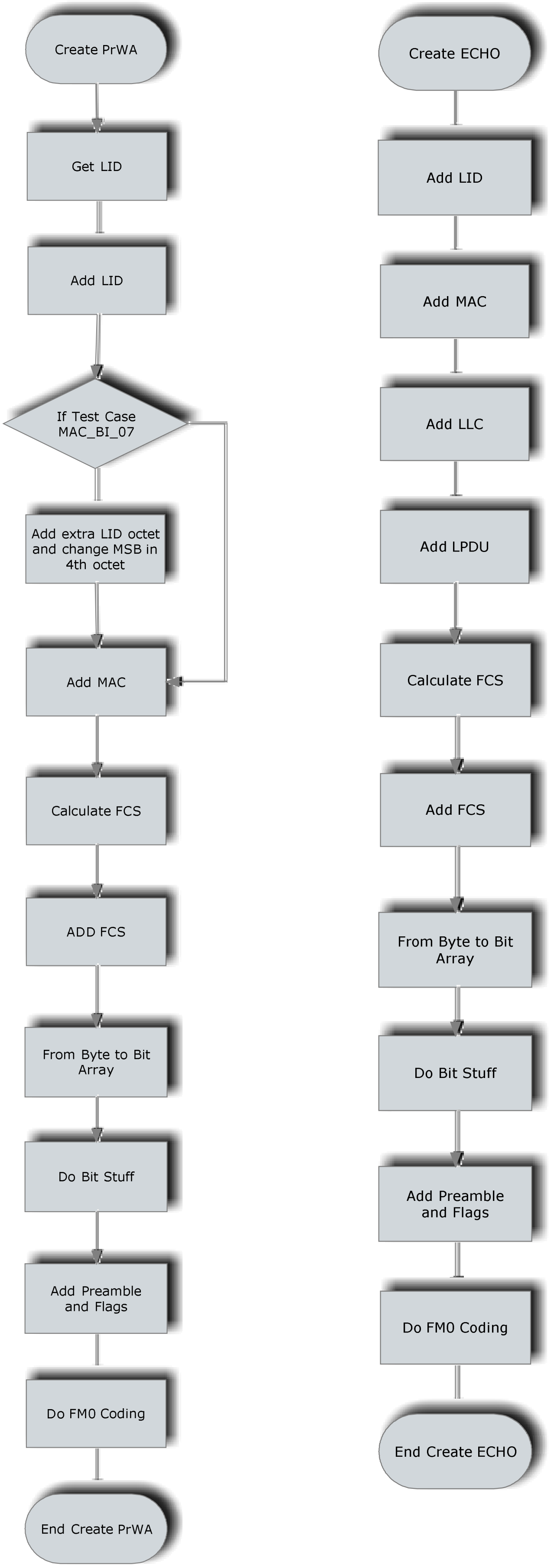


Figure A.4: Flowchart over Subroutines Create PrWA and Create ECHO

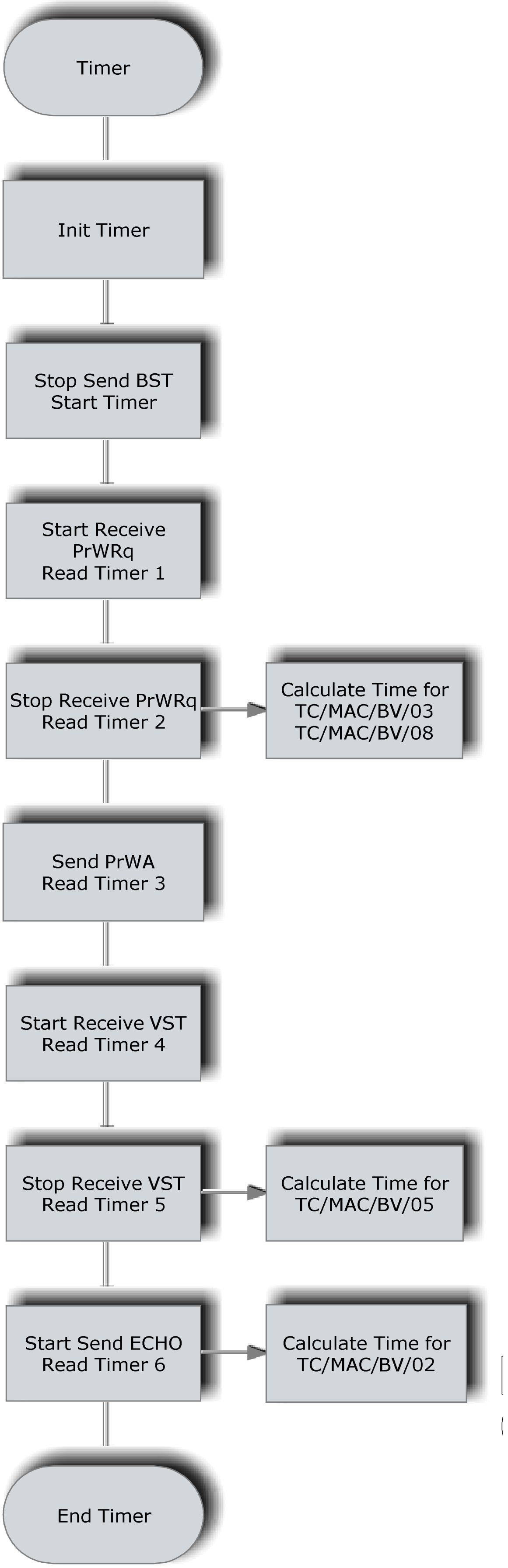
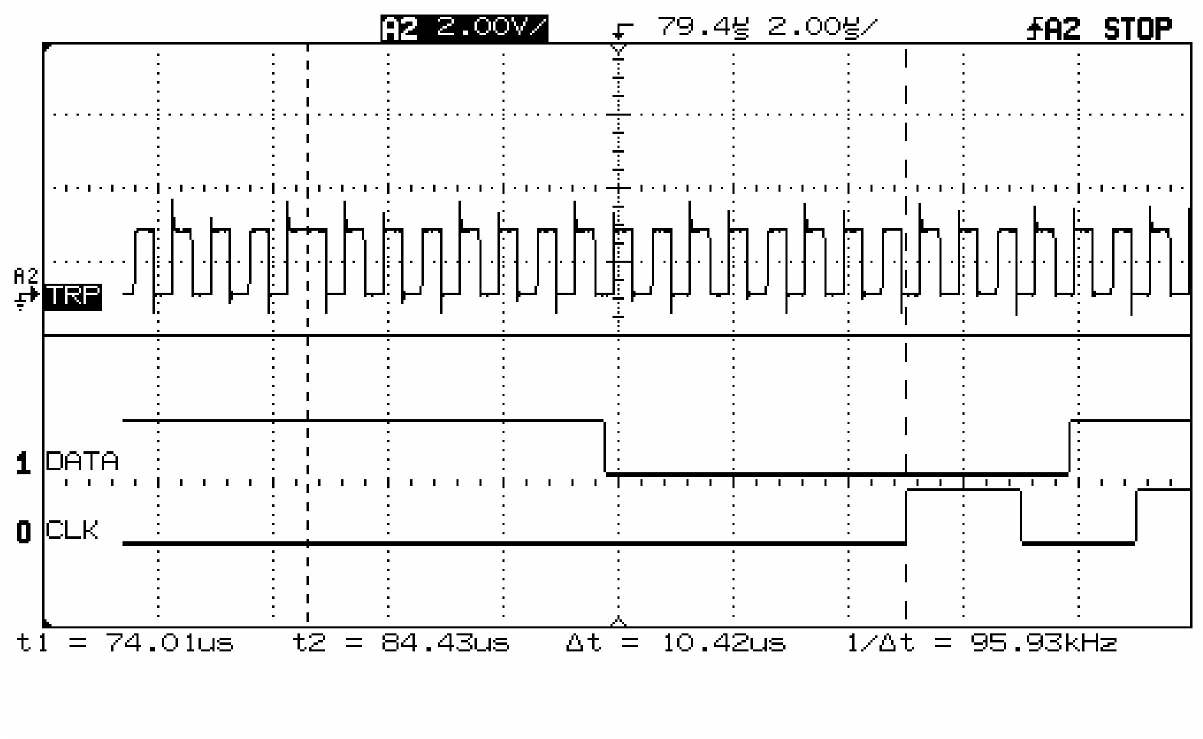


Figure A.5: Flowchart over Timer Function

Screenshot

The figure illustrates the delay in the transceiver of received uplinks. It is Δt showing the delay between the transponder starts to transmit and the transceiver starts to send the bits to the MCU. The upper square pulses display the transmitting signal from the transponder and where a new bit starts where t1 is placed. The middle graph signed DATA shows the data bits received in the transceiver and the lowest graph is the clock signal sent along with data signal to the micro controller. t2 is the time where the bit is trigged into the micro controller.

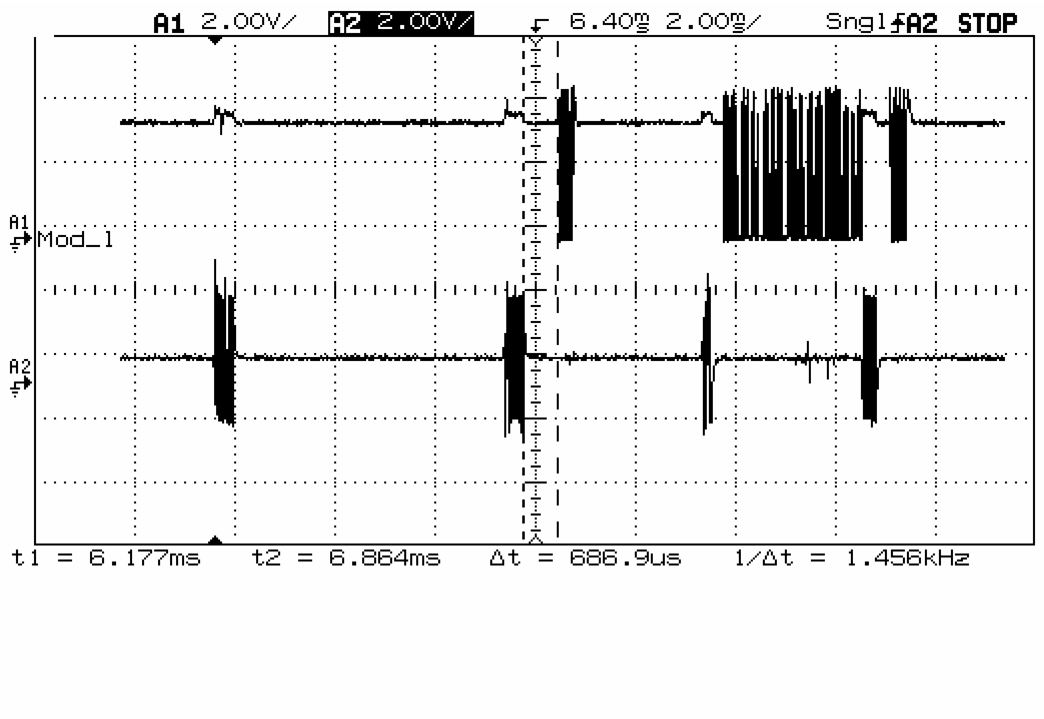


A2 2.00V/ 79.4µs 2.00µs/ A2 STOP

t1 = 74.01us t2 = 84.43us Δt = 10.42us 1/Δt = 95.93kHz

Figure A.6: Screenshot from Oscilloscope showing the delay in the transceiver

The figure below shows the communication between the RSE and the OBU where the lower signal is the signal sent from MCU to transceiver for transmitting to OBU. The upper signal is the received signal from the transponder after the transceiver sends it to the MCU. In order from left to right the signals are; wakeup BST, second BST, PrWRq, PrWA, VST, ECHO Request, ECHO Response. From the figure the time between the second BST and the PrWRq indicates the transponder replied in the second allocated window.



A1 2.00V/ A2 2.00V/ 6.40ms 2.00ms/ Sngl A2 STOP

t1 = 6.177ms t2 = 6.864ms Δt = 686.9us 1/Δt = 1.456kHz

Figure A.7: Screenshot from Oscilloscope showing the DSRC initialisation