



Daniel Fidalgo Morais **Implementação da camada MAC de um sistema
DSRC de identificação.**

**Implementation of the Mac layer of an identification
DSRC system**



**Daniel Fidalgo Morais Implementação da camada MAC de um sistema
DSRC de identificação.**

**Implementation of the Mac layer of an identification
DSRC system**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. João Nuno Pimentel Matos, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Mestre Ricardo Matos Abreu, do Instituto de Telecomunicações, Pólo de Aveiro

Este trabalho foi, em parte desenvolvido no âmbito do Projecto Viave realizado no Instituto de Telecomunicações – Pólo de Aveiro, e só foi possível graças ao apoio e suporte da Brisa, Auto estradas de Portugal.

O Júri / Examination Committee

Presidente / President

Prof. Dr. Nuno Miguel Gonçalves Borges de Carvalho
Professor Associado com Agregado da Universidade de Aveiro

Vogal / Members of the board

Prof. Dr. António João Nunes Serrador (Arguente)
Departamento de Electrónica e Telecomunicações e Computadores do ISE de Lisboa

Vogal / Members of the board

Prof. Dr. João Nuno Pimentel da Silva Matos (Orientador)
Professor Associado da Universidade de Aveiro

**Agradecimentos /
Acknowledgements**

Agradeço a todos os que de forma directa ou indirecta possibilitaram a realização do trabalho. Família, amigos, colegas de trabalho e entidades participantes.

Particularizando, o meu orientador, Professor Doutor João Nuno Pimentel da Silva Matos e co-orientador, Mestre Ricardo Matos Abreu, pelo apoio prestado durante a execução do trabalho.

Palavras-chave

Camada MAC, DSRC, EFC

Resumo

Ultimamente o mundo tem vindo a assistir ao fenómeno da globalização ao nível económico-social e político. A tecnologia acompanha esta evolução, a vários níveis proporcionando maior qualidade de vida aos cidadãos, na grande quantidade de produtos e na elevada qualidade dos serviços prestados. Neste contexto é de prever a extensão do conceito de pagamento automático de portagens ou outros serviços associados ao automóvel, no espaço europeu.

Inserido num projecto de desenvolvimento dum sistema de comunicações de curto alcance (DSRC – Dedicated Short Range Communication) de acordo com as normas europeias mais recentes, neste caso a EN12795, o presente trabalho descreve uma possível implementação da camada MAC (Medium Access Control), para o referido sistema. O resultado obtido no trabalho é um conjunto de primitivas de software que implementam a camada MAC do RSE (Road Side Equipment).

Keywords

Camada MAC, DSRC, EFC

Abstract

Lately the world has witnessed the phenomenon of globalization, political, social and economical changes have happened. Technology has ridden along with this tendency at many levels, providing a great number of products and services accessible to all of us. In this context, it is predictable that this concept of automatic payment or any other vehicular service will extend in Europe.

There is a project which aims the construction of a DSRC (Dedicated Short Range Communication), according to the latest European standards. This work is part of the referred system and describes a possible implementation of the MAC layer, following the EN12795 standard. The result of this work is a set of software primitives that implement the RSE (Road Side Equipment) MAC layer.

Table of Contents

Table of Contents.....	1
List of Tables.....	3
List of Figures.....	4
1) Introduction.....	6
1.1) Overview.....	6
1.2) Definitions and Basic Principles.....	7
1.2.1) Hardware and basic communications definitions.....	7
1.2.2) General Approach.....	8
1.2.3) Technical Approach.....	10
1.3) State of the art.....	16
2) Work Description.....	18
2.1) RSE and OBE hardware and communication Issues.....	18
2.1.1) Existing Hardware.....	18
2.1.2) European Standard 12795.....	19
2.2) Microcontrollers Communication Issues.....	31
2.2.1) Hardware Connection - Serial Peripheral Interface (SPI).....	31
2.2.2) Interface Software - Communication Protocol Master/Slave.....	31
3) Developed Software.....	42
Pony Prog 2000.....	42
3.1) Software Functions between the OBE and the RSE:.....	42
3.2) Software functions between the microcontrollers:.....	49
4) Conclusions and Future Work.....	57

APPENDIX A – References.....	59
APPENDIX B - Glossary.....	61
Definitions	61
Acronyms	61
APPENDIX C – Standard IEEE 802.11p.....	64
APPENDIX D – Microcontrollers Features.....	65

List of Tables

TABLE 1 - MAC/LLC LIST OF SERVICE PRIMITIVES.....	20
TABLE 2 - LLC LIST OF SERVICE PRIMITIVES	21
TABLE 3 - LPDU FIELD FORMAT	27
TABLE 4 - FRAME TYPES BETWEEN MICROCONTROLLERS	35
TABLE 5 - ERROR SITUATIONS IN FRAMES EXCHANGE	39
TABLE 6 - APPLICATION MESSAGES	40
TABLE 7 - SPI SOFTWARE FUNCTIONS (PHYSICAL LAYER)	53

List of Figures

FIGURE 1 - (ROAD SIDE EQUIPMENT) AND OBE (ON BOARD EQUIPMENT)	7
FIGURE 2 - DOWNLINK E UPLINK DESCRIPTION.....	8
FIGURE 3 - STEP 1: RSE SENDS A PUBLIC DOWNLINK (BST BEACON SERVICE TABLE).....	9
FIGURE 4 - STEP 2: OBE SENDS A PRIVATE UPLINK REQUEST	9
FIGURE 5 - STEP 3: RSE SENDS A PRIVATE UPLINK ACKNOWLEDGE	10
FIGURE 6 - STEP 4: OBE SENDS A VST (VEHICLE).....	10
FIGURE 7 - TECHNICAL APPROACH SCHEME.....	11
FIGURE 8 - OSI (OPEN SYSTEM INTERCONNECTION) MODEL.....	13
FIGURE 9 - EXISTING RSE HARDWARE PROTOTYPE	18
FIGURE 10 - OBE HARDWARE PROTOTYPE.....	19
FIGURE 11 - MAC SERVICE PRIMITIVES.....	19
FIGURE 12 - EXAMPLE OF FRAMES EXCHANGE USING MAC SERVICE PRIMITIVES	21
FIGURE 13 - FRAME FORMAT (EN12795).....	24
FIGURE 14 - FRAME SIZE BY FIELDS (EN12795).....	24
FIGURE 15 - PRIVATE LID FORMAT	24
FIGURE 16 - BROADCAST LID FORMAT	25
FIGURE 17 - MULTICAST LID FORMAT	25
FIGURE 18 - DOWNLINK MAC CONTROL FIELD FORMAT	25
FIGURE 19 - UPLINK MAC CONTROL FIELD FORMAT	26
FIGURE 20 - LLC CONTROL FIELD	27
FIGURE 21 - ADDRESS ESTABLISHMENT	28
FIGURE 22 - UPLINK TO PUBLIC DOWNLINK TRANSITION	28
FIGURE 23 - DOWNLINK TO DOWNLINK TRANSITION	29
FIGURE 24 - DOWNLINK TO PRIVATE UPLINK TRANSITION.....	29
FIGURE 25 - DOWNLINK TO PUBLIC UPLINK TRANSITION	30
FIGURE 26 - UPLINK TO PUBLIC UPLINK TRANSITION.....	30
FIGURE 27 - SPI HARDWARE CONNECTIONS	31

FIGURE 28 - DETAILED SPI HARDWARE CONNECTIONS, BY INTERF.....	32
FIGURE 29 - DETAILED SPI CONNECTION, BETWEEN SLAVE AND MASTER	33
FIGURE 30 - SPI PROTOCOL FRAME FORMAT.....	34
FIGURE 31 - VALID SEQUENCE WHERE MASTER INITIATES COMMUNICATION	37
FIGURE 32 - VALID SEQUENCE WHERE SLAVE STARTS COMMUNICATION.....	37
FIGURE 33 - VALID SEQUENCE WITH TIMEOUT ON THE SLAVE SIDE.....	38
FIGURE 34 - VALID SEQUENCE WITH TIMEOUT ON THE MASTER SIDE	39
FIGURE 35 - BLOCK DIAGRAM OF THE RSE FOR PROGRAMMING INTERFACE.....	41
FIGURE 36 - BLOCK DIAGRAM OF THE RSE	41
FIGURE 37 - SERVICE PRIMITIVES OF THE MAC LAYER	43
FIGURE 38 - RSE STATE MACHINE.....	44
FIGURE 39 - STATE MACHINE DESCRIBING THE MAC LAYER BEHAVIOR OF THE OBE.....	45
FIGURE 40 - STATE MACHINE CONCERNING ONLY THE PUBLIC UPLINK BRANCH	46
FIGURE 41 - STATE MACHINE CONCERNING ONLY THE PRIVATE UPLINK BRANCH	47
FIGURE 42 - HARDWARE CONNECTION, TO TEST THE SOFTWARE BETWEEN THE RSE AND THE OBE.....	49
FIGURE 43 - LAYERS DIAGRAM OF THE MICROCONTROLLER'S PROTOCOL.....	50
FIGURE 44 – SOFTWARE FUNCTIONS RUNNING, WHILE MASTER IS SENDING A BYTE.....	51
FIGURE 45 - SOFTWARE FUNCTIONS RUNNING, WHILE SLAVE IS SENDING A BYTE	51
FIGURE 46 - FIELDS HANDLED BY THE LOWER MAC LAYER	53
FIGURE 47 – FIELDS HANDLED BY THE UPPER MAC LAYER	54
FIGURE 48 – FIELDS HANDLED BY THE APPLICATION LAYER.....	55
FIGURE 49 - HARDWARE CONNECTION, TO TEST THE SOFTWARE BETWEEN RSE AND OBE	56

1) Introduction

This chapter describes the main features of this project, the definitions and the basic principles. In 1.1) Overview, the motivation of this project is depicted and in the following chapters, the basic definitions are listed to provide the basics concepts to understand the whole project and this document.

1.1) Overview

Lately the world has witnessed the phenomenon of globalization, political social and economical changes have happening. Technology has ridden along with this tendency at many levels, providing a great number of products and services accessible to all of us.

In Europe, the growth of the European Union is a fact. With the purpose of unify us into a united political, monetary and legislative system. There are Committees in the field, managing all the technical issues, as the Standardization European Committee.

Nowadays, Via Verde¹, which is an EFC (Electronic Fee Collection) system, uses is a DSRC (Dedicated Short Range Communication) system, in the RTTT (Road Traffic and Transport Telematics) environment, based on LDR (Low Data Rate) transmission, with a data rate of approximately 32Kbits/s. There are also other powerful data signalling rates such as MDR (Medium Data Rate), with a data rate of approximately 256kbits/s and 512 kbits/s.

This work is part of a larger project which aims the construction of two compliant units: an OBE (On Board Equipment) and a RSE (Road Side Equipment), using MDR data signalling rate, according to the European Standard EN 12795 [1] (released by Standardization European Committee) and using the know-how of the existing technologies already in use. The purpose of this work is to improve, and probably replace, the existing technology. This allows the technological unification of several countries under the same working pattern. The advantage is the usage of this service without the need of hardware changes, i.e., the user can travel in all the countries using the same tag for electronic toll payment.

There are also other applications, such as the fuel and parking payment and also a new project the electronic licence plate, also known as e-Plate.

This work is not a typical academic research limited to laboratory proof-of-concept. It is instead the implementation of specific features according to the requests of the project's partner and customer *Brisa*. There will be, in a short term, field tests.

¹ Via verde: is the EFC (Electronic Fee Collection), used in Portugal nowadays.

1.2) Definitions and Basic Principles

The next subchapters describe the main elements denomination of the standard, Hardware and Software, and also the basic principles of the practical application regarding this work.

1.2.1) Hardware and basic communications definitions

The hardware built in this project, until no is the fixed and the mobile equipment. The following paragraphs list these equipments and also the basic type of communication between them.

- RSE: Road Side Equipment

Road Side Equipment or RSU (Road Side Unit), also known as fixed equipment, is the fixed communication facility with one or more downlink channels and, optionally, one or more uplink channels (Please see Figure 1).

- OBE: On Board Equipment

On Board Equipment or OBU (On Board Unit), also known as mobile equipment, is the mobile communication facility capable of receiving information from the fixed equipment on the downlink and, optionally, also capable of transmitting information to the fixed equipment on the uplink, (Please see Figure 1).

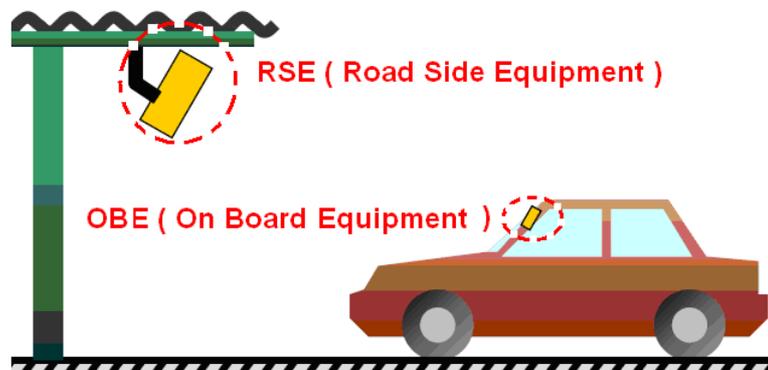


Figure 1 - (Road Side Equipment) and OBE (On Board Equipment)

- Downlink

Downlink is the communication channel on which fixed equipment transmits its information. Public and Private Downlink windows are distinguished by their information, Public if it is generic information and Private if specific user information is present, (Please see Figure 2). Using MDR, the data signalling rate is 512 Kbits/s.

- Uplink

Uplink is the communication channel on which mobile equipment transmits its information. Public if it is generic information and Private if specific user information is present, (Please see Figure 2). Using MDR, the data signalling rate is 256 Kbits/s.

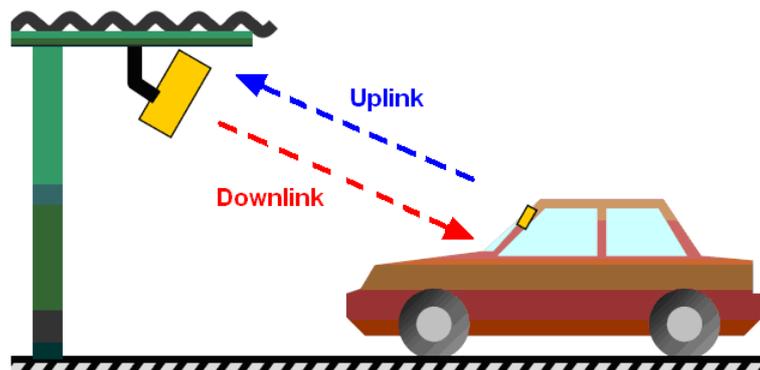


Figure 2 - Downlink e Uplink description

1.2.2) General Approach

In this chapter, the communication sequence is analysed and listed, step by step, in a simple and basic way. The timeouts and the error situations are not considered here. To study in detail these cases please refer to sub chapter 2.1.2) European Standard 12795.

Step 1) The RSE fixed on the toll, starts the communication by broadcasting, time to time, a BST (Beacon Service Table) in a **Public Downlink**, trying to obtain responses from any car in range, (Please see Figure 3). A BST is an application message providing from the RSE.

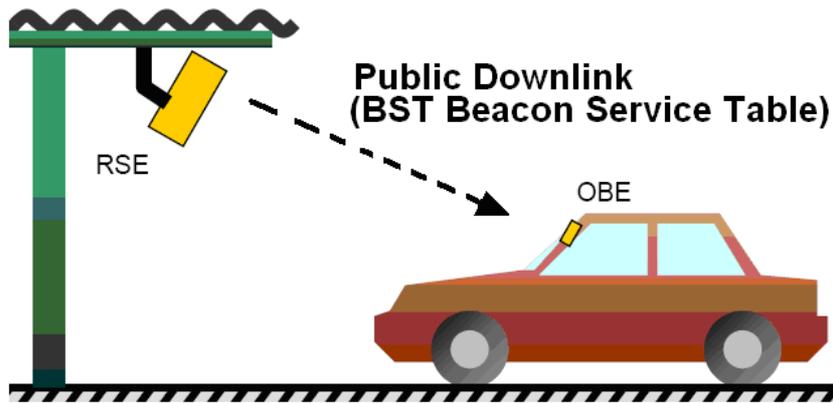


Figure 3 - Step 1: RSE sends a Public Downlink (BST Beacon Service Table)

Step 2) The OBE placed in the windshield of the car, when in range, responds with a **Private Uplink Request**, requesting a time window to send information of the vehicle's identification, (Please see Figure 4).

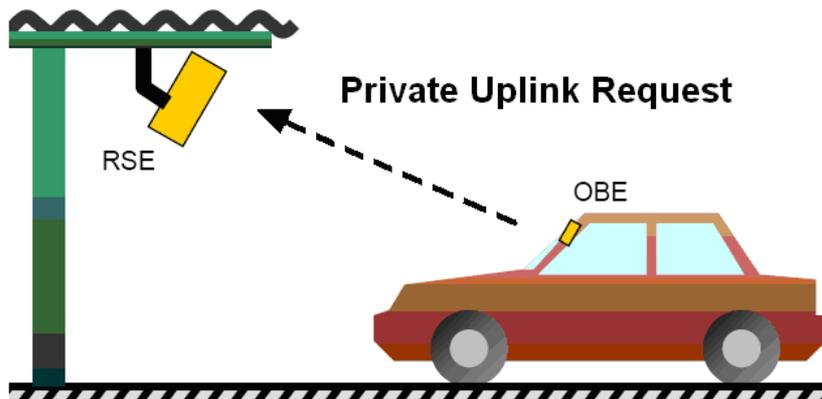


Figure 4 - Step 2: OBE sends a Private Uplink Request

Step 3) The RSE, after detecting the request, gives permission to the vehicle to transmit information, by sending a **Private Uplink Acknowledge**, (Please see Figure 5).

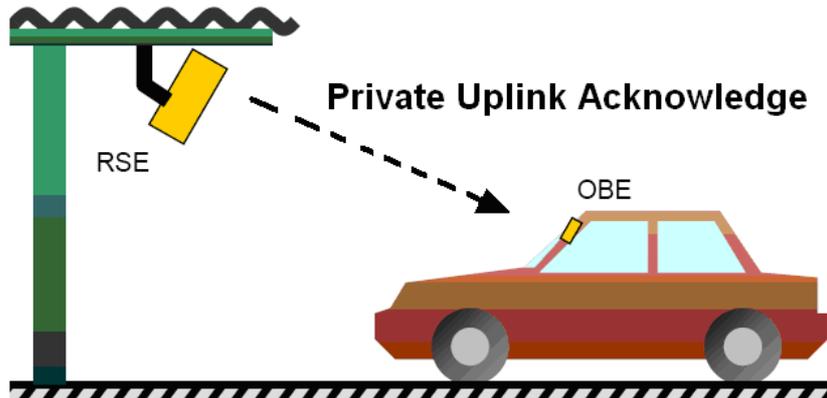


Figure 5 - Step 3: RSE sends a Private Uplink Acknowledge

Step 4) The OBE responds with a VST (Vehicle Service Table), containing the information needed to proceed the process, (Please see Figure 6). A VST is an application message providing from the OBE.

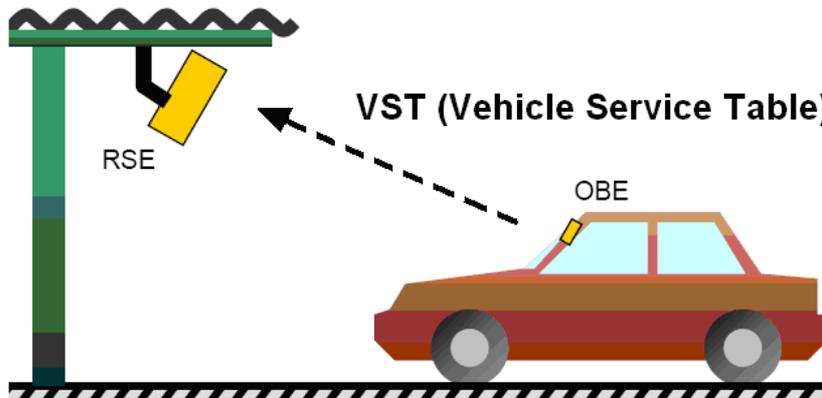


Figure 6 - Step 4: OBE sends a VST (Vehicle)

1.2.3) Technical Approach

To accomplish the entire project's specifications it is mandatory to be aware of aspects concerning many subjects. This DSRC system can be divided in two parts, please see Figure 7. The first part is the RSE and OBE wireless communication and the second part is the RSE internal microcontrollers' communication.

Regarding the first part, the main issues to study, in detail, are standard EN12795, also the features of the microcontroller and the existing software functions, of the physical layer. Standard EN 12795 imposes some time

constrains, described in 2.1.2) European Standard 12795, that need to be handled very carefully.

Concerning the second part, the studying issues are, regarding the hardware, the SPI (Serial Peripheral Interface) of the microcontroller, and regarding the software, the Communication Protocol between the microcontrollers, described in 2.2.1) Hardware Connection - Serial Peripheral Interface (SPI) and in 2.2.2) Interface Software - Communication Protocol Master/Slave, respectively.

The use of two microcontrollers, working in a master/slave mode was a request of our partner *Brisa*. The slave controls the wireless communication and all the low level issues (physical), while the master controls all the high level issues (application), and can be connected to an external network.

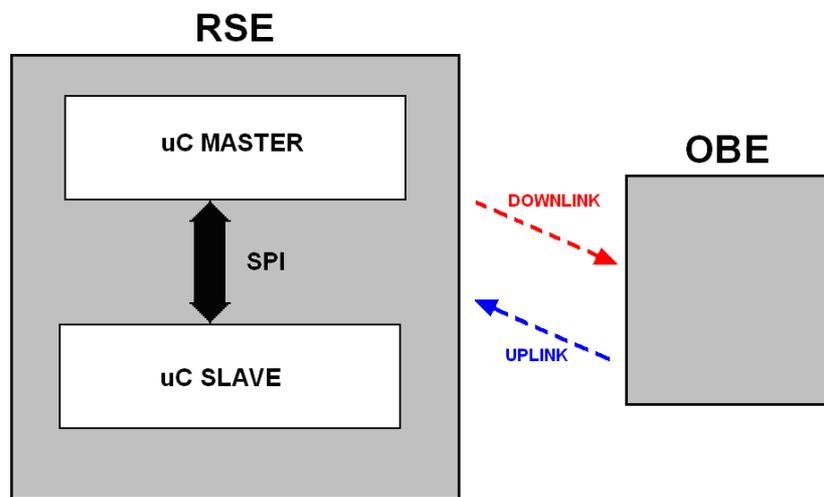


Figure 7 - Technical approach scheme

ATMEGA 128

The ATMEGA 128 is the microcontroller used in this work, as the slave. It is a 8 bit microcontroller is based on a RISC architecture, built over low power CMOS. It executes instructions, which achieve throughputs of almost 1MIPS per MHz. The advantage of these features is the power consumption optimization in relation with the processing speed. For more details please see APPENDIX D – Microcontrollers Features.

TX27 i.MX27 module

The TX27 is the microcontroller used in this work, as the Master. Our partner, in this project, *Brisa* is responsible for the firmware of this microcontroller. For more details please see APPENDIX D – Microcontrollers Features.

OSI (Open system Interconnection) model

The Open System Interconnection Reference Model (also known as OSI Reference Model or OSI Model and OSI Seven Layer Model) is a generic theoretical communications protocol, based on layers, used to project and conceive network architectures.

A layer is a set of procedures and/or connections, interacting between them, each one of them providing or executing different tasks according to the layer's features. Each layer can interact with the previous or the following layer of the hierarchy. Theoretically, the layers of the same hierarchy rank from different structures are linked by a horizontal link.

The OSI model contains seven specific layers, dealing with all the issues, from the hardware features up to the user specifications. The layers are, from the lowest level up to the highest, L1 - Physical, L2 - Data-Link, L3 - Network, L4 - Transport, L5 - Session, L6 - Presentation and L7- Application.

The advantage of this model is the organization, allowing to the model projector a simpler way of conceiving and implementing the whole protocol, by dividing it in smaller parts. The description by layers is in Figure 8.

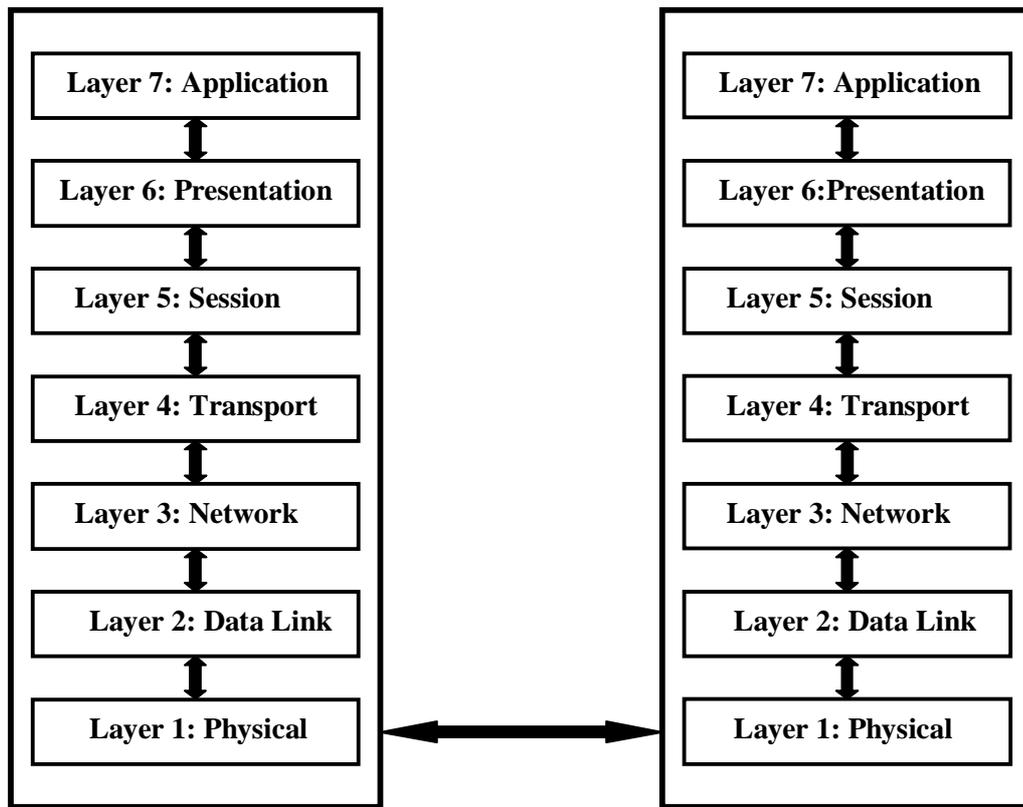


Figure 8 - OSI (Open system Interconnection) model

Layer 1: Physical Layer

The Physical Layer (L1) handles the hardware issues and the physical connections. This layer only interacts with the following one, the Data Link Layer. The connection between a device and a physical medium is established in this layer, including all the details usually depict in the devices' datasheets, such as pins, connectors and voltages. Physical layer is responsible for the interaction, sending and reception of signals with the communication medium, and also for the signal modulation.

Every communication network needs a physical layer, as happens with the well-known Ethernet, IEEE 802.11 and IEEE 802.15.4, among others.

Layer 2: Data Link Layer

The Data Link Layer (L2) is responsible for data transfer between network elements and also detection of communications errors proceeding from the lower layer L1.

In this layer data is arranged into sets of bytes, named frames, each one with a specific format according to the type of information to be transferred. Typically frames have headers including start bytes, named flags, used for frame detection; frames also contain data fields where the information to be transferred to upper or lower layers is stored; and finally control fields, used for error detection.

This layer can be divided in two sub layers, the MAC (Medium Access Control) and the LLC (Logic Link Control). MAC sub layer is the lowest one, and it handles the basic issues relating with addressing and channel control of the network elements. The LLC is the highest one, and it provides flow control mechanisms, between MAC sublayer and Network layer.

Layer 3: Network Layer

The Network Layer (L3) is responsible for routing the information through multiple networks, this includes the addressing and also provides traffic control. The addressing set is established by the network designer.

An example of a protocol working on this layer is the IP (Internet Protocol).

Layer 4: Transport Layer

The Transport Layer (L4) provides the interface between the upper and the lower layers, by dividing or gathering information, according to the situation, before sending it to the respective layer. When the information proceeds from the upper layers, it is divided into smaller blocks and then transmits them to the lower layer (L3). On the other hand, when the information proceeds from the lower layers, it is gathered into bigger blocks and then transmitted to the upper layer (L5).

Besides this feature, this layer also guarantees the flow control, by preventing the overload of information, in a certain moment, on the network.

There are some examples of protocols that work in this layer, such as TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

Layer 5: Session Layer

The Session Layer (L5) provides to the user an network interface, by granting and managing the access to the network, by the other layers. This layer is also responsible for the authentication, the configuration and the operation mode of the sessions, by controlling, starting and finishing, the connections between network elements.

An example of a protocol working on this layer is the NFS (Network File System).

Layer 6: Presentation Layer

The Presentation Layer (L6), also known as syntax layer, whose main feature is to handle the issues related with information syntax and semantics. This layer encrypts the information into known types of data representation, in order to avoid incompatibility problems.

An example of a format type used is the ASCII (American Standard Code for Information Interchange).

Layer 7: Application Layer

The Application Layer (L7) is the highest level layer and establishes an interface between the architecture and the user and it defines the data content to be transferred. This layer also manages all the issues related with identification of the communications elements, resources availability and synchronization.

There are some examples of protocols working in this layer such as FTP (File Transfer Protocol) and HTTP (Hypertext Transfer Protocol)

Note: This project is based on the OSI model, but only Physical Layer (L1), Data Link Layer (L2) and Application Layer (L7) are used. In this work, only L2 is considered. L1 was already done and L7 and the LLC of L2 are managed by our partner Brisa.

MAC (Medium Access Control) sublayer, applied to this case

MAC sublayer is fitted between the physical layer and LLC (Logical Link Control) sublayer, for both equipments the fixed RSE and the mobile OBE. The medium access control is unbalanced. The access to the physical medium is always controlled by the RSE, granting access to mobile elements.

The functioning is made in half duplex mode, that is, the communication is made in both ways but one at a time and also in asynchronous TDMA (Time Division Multiple Access)

1.3) State of the art

There are some new incoming projects that come into sight as a state-of-the-art solution to Intelligent Transportation Systems (ITS). Using this technology, the main one is the electronic license plate. There is also technology with the goal of improving road safety, with some recent and interesting progresses.

Regarding the electronic license plate, also known as e-plate [2], it is an electronically tagged number plate, in practice it is an OBE. The customers with Via Verde are already under the new law.

The Portuguese government approved a law [3] (Decree number 112/2009 of 18th of May), in which the use of the e-plate will be mandatory, this will start in 2010. The Decree, in article 19th specifies the technology and it is the DSRC systems using the frequency of 5,8Ghz, under EN 15509 EFC – Interoperability application profile, either for MDR and LDR.

Vehicular wireless communication systems are present in our roads today, specially in highways for toll collection. However, a vehicular communication system is a resource that should also be used to improve road safety. Thus, the challenge is to build a communication system that can be used for toll collection and to transmit and receive warning messages as well [4].

These new projects are another variant of the DSRC technology, but working in real time vehicular communications and networks using WAVE (Wireless Access in Vehicular Environment), allowing a new way of communication between vehicles, with some interesting features such as intelligent management and data exchange services of vehicle safety services, and Internet access.

This is accomplished by enabling each vehicle to communicate with surrounding vehicles and with fixed road side equipments. When an unexpected situation occurs, automatically a warning message is sent to the vehicles approaching the accident area and to the highway operator so it can call the emergency services.

The communication system, partially described here, is based on the standard IEEE 802.11p [5] and is a vehicular communication system that uses the 5.9GHz band for Dedicated Short Range Communications (DSRC), these systems adopt orthogonal frequency-division multiplexing (OFDM) and achieve data rates of 6–27Mbs/s. In WAVE systems, an RSU can cover a range of up to 1000m.

2) Work Description

This chapter describes all the issues analysed and handled in this project, in order to develop the MAC layer software, for both situations. The first one, is the RSE and OBE communication, please see 2.1) *RSE and OBE hardware and communication Issues*, and the second one is the RSE internal microcontrollers communication, please see 2.2) *Microcontrollers Communication Issues*.

2.1) RSE and OBE hardware and communication Issues

This sub chapter explains, in detail, the behavior of the RSE and the OBE in the presence of each other, in terms of the hardware and how they are supposed to interact.

2.1.1) Existing Hardware

The hardware for both devices is in advanced development state, with two laboratorial prototype units built as can be observed, RSE in Figure 9 and the OBE in Figure 10. Also, a first bare software layer is written and tested, covering physical frame transmission and reception, as well as some MAC details as bit stuffing/de-stuffing and Cyclic Redundancy Check (CRC) generation and checking. This software was developed for Atmel's AVR family of microcontrollers, which are present on both devices and where the MAC/LLC software also runs. For more hardware information please consult [6], [7], [8] and [9].

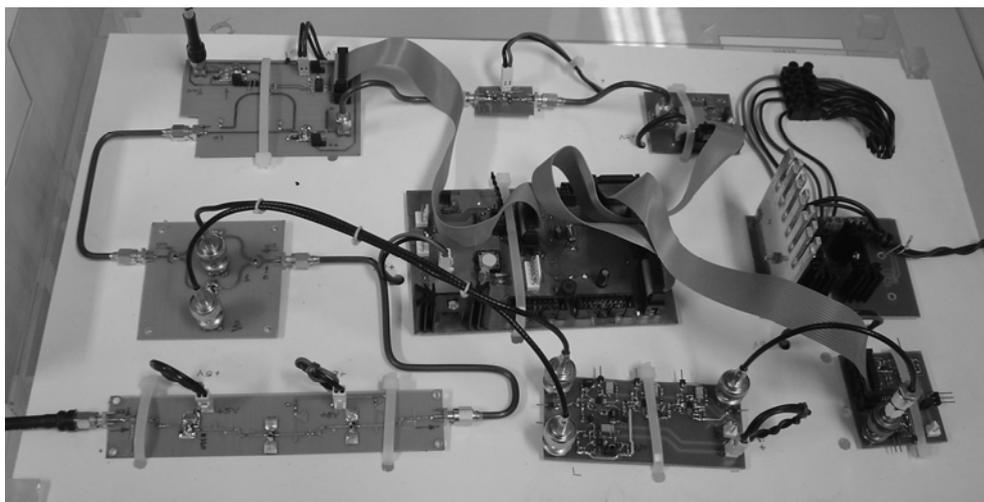


Figure 9 - Existing RSE hardware prototype

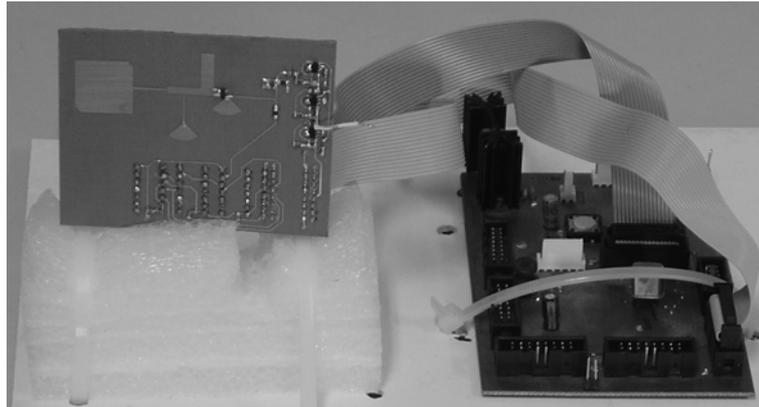


Figure 10 - OBE hardware prototype

2.1.2) European Standard 12795

This subchapter depicts the standard used in the project. Standard EN 12795 is one of a series of European Standards defining the framework of a DSRC (Dedicated Short Range Communication) in the RTTT (Road Traffic and Transport Telematics) environment.

This standard defines the Data Link Layer of the DSRC, it is subdivided in the two sub layers, the MAC (Medium Access Control) and the LLC (Logic Link Control) Sublayer. Last revision, was in March 2003.

MAC Service Primitives

There are the following primitives, MA-DATA.request and MA-DATA.indication, to communicate with LLC sublayer (Please see Figure 11 and Table 1)

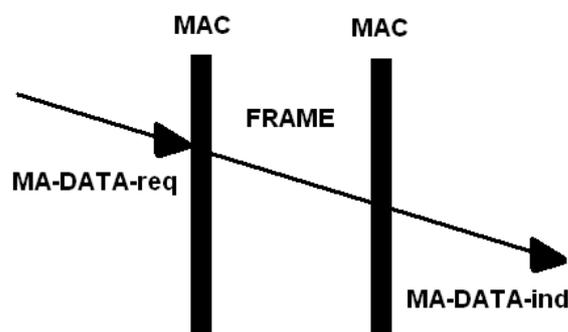


Figure 11 - MAC service primitives

- MA-DATA.request: used to receive sublayer LLC frames

- MA-DATA.indication: used to send sublayer LLC frames

MAC/LLC Service Primitives			
Fixed Equipment		Mobile Equipment	
F-MA-DATA.request	F-MA-DATA.indication	M-MA-DATA.request	M-MA-DATA.indication

Table 1 - MAC/LLC list of Service Primitives

Fixed Equipment (RSE), primitives' description:

- **F-MA-DATA.request:** This primitive is passed to the MAC sublayer, by the LLC layer, to request that an LPDU is transmitted to a mobile SAP in the first available downlink window.
- **F-MA-DATA.indication:** This primitive is passed from the LLC sublayer to the MAC sublayer to indicate the successful reception of a valid frame from a mobile SAP.

Mobile Equipment (OBE), primitives' description:

- **M-MA-DATA.request:** This primitive is passed from the MAC sublayer to the LLC sublayer to request that an LPDU is transmitted to a fixed SAP in an uplink window.
- **M-MA-DATA.indication:** This primitive is passed from the LLC sublayer to the MAC sublayer to indicate the successful reception of a valid frame from a fixed SAP.

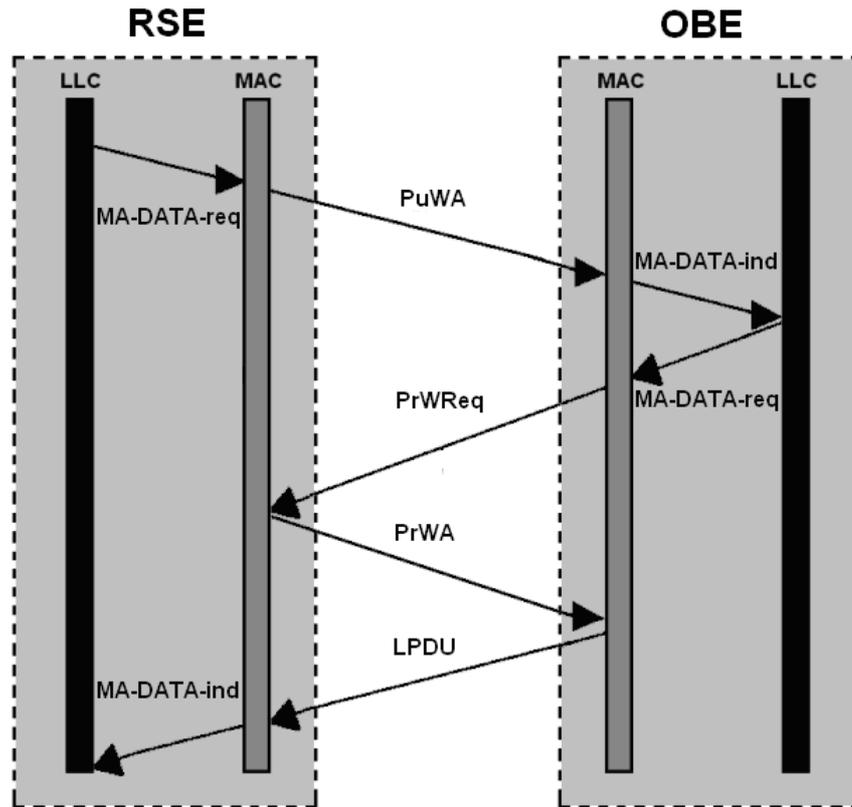


Figure 12 - Example of frames exchange using MAC service primitives

LLC (Logical Link Control) sublayer

LLC sublayer is responsible for creating and handling the PDU (Protocol Data Unit) *command e response*. There are two types of data link, the unacknowledged connectionless-mode service and the acknowledge connectionless-mode service. (Please see Table 2)

LLC Service Primitives	
Unacknowledge connectionless-mode service	Acknowledge connectionless-mode service
DL-UNIDATA.request DL-UNIDATA.indication	DL-DATA-ACK.request DL-DATA-ACK.indication DL-DATA-ACK-STATUS.indication
	DL-REPLY.request DL-REPLY.indication DL-REPLY-STATUS.indication
	DL-REPLY-UPDATE.request DL-REPLY-UPDATE-STATUS.indication

Table 2 - LLC list of Service Primitives

Unacknowledged connectionless-mode service

This service provides to the application layer the possibility of exchange link service data units (LSDUs) without the establishment of a data link connection on an unacknowledged base. In this case the data transfer can be point to point, multicast, or broadcast.

Service primitives:

- **DL-UNIDATA.request:** The primitive shall be passed from the application layer to the LLC sublayer to request transmission of a LSDU (Link Layer Service Data Unit).
- **DL-UNIDATA.indication:** The primitive shall be passed from the LLC sub layer to the application layer to indicate the arrival of a LSDU.

Acknowledged connectionless-mode service

This service provides to the application layer the possibility of exchange link service data units (LSDUs) which are acknowledged at the LLC sublayer, without the establishment of a data link connection. In this case the data unit transfer is point to point.

Data unit transmissions primitives:

- **DL-DATA-ACK.request:** The primitive shall be passed from the RSE application layer to the RSE LLC sublayer to request transmission of a LSDU to the specific OBE.
- **DL-DATA-ACK.indication:** The primitive shall be passed from the OBE LLC sublayer to the OBE application layer to indicate the arrival of a non-null and non-duplicate LSDU from the RSE
- **DL-DATA-ACK-STATUS.indication:** The primitive shall be passed from the RSE LLC sublayer to the RSE application layer to indicate the success or failure of the previous associated acknowledge connectionless-mode data unit transmission request.

Data unit exchange primitives:

- **DL-REPLY.request:** The primitive shall be passed from the RSE application layer to the RSE LLC sublayer to request transmission of a previously prepared data unit from the specified OBE (in

case of null parameter) or to exchange data units with the specified OBE.

- **DL-REPLY.indication:** The primitive shall be passed from the OBE LLC sublayer to the OBE application layer to indicate either a request of a LSDU from the RSE
- **DL-REPLY-STATUS.indication:** The primitive shall be passed from the RSE LLC sublayer to the RSE application layer to indicate the success or failure of the previous acknowledged connectionless-mode data unit exchange request and to pass data if available.

Data unit preparation reply:

- **DL-REPLY-UPDATE.request:** The primitive shall be passed from the OBE application layer to the OBE LLC sublayer to request preparation of a LSDU for future access. A subsequent DL-REPLY-UPDATE.request service primitive serves to replace the currently associated LSDU with a new one.
- **DL-REPLY-UPDATE-STATUS.indication:** The primitive shall be passed from the OBE LLC sublayer to the OBE application layer to indicate the success or failure of the previous associated data unit preparation request DL-REPLY-UPDATE.request.

Broadcast LID (please see Figure 16).

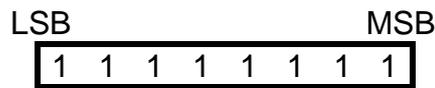


Figure 16 - Broadcast LID format

b) Multicast LID (please see Figure 17).

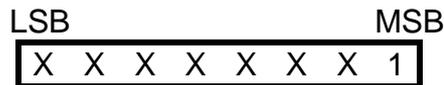


Figure 17 - Multicast LID format

Note: The issues regarding transparency and bit stuffing were handled in the physical layer.

- MAC Control Field

This field defines the MAC controlling bits. There are two different cases, according to the stream direction Downlink (please see Figure 18) and Uplink (please see Figure 19). It is used to: indicate if the frame contains an LPDU, specify the transmission direction, allocate public/private windows and also request private windows.

1) DOWNLINK



Figure 18 - Downlink MAC Control Field format

Bits detailed description:

- **L (LPDU bit):** indicates the existence or absence of the LPDU on the frame
 If L equals '1', LPDU exists on the frame.
 If L equals '0', LPDU does not exist on the frame

- **D(0):** indicates the link direction,
 If D(0) equals '0' is a downlink
 If D(0) equals '1' is an uplink

- **A:** indicates window allocation
 If A equals '1', RSU is allocating an Uplink Window
 If A equals '0', RSU is not allocating an Uplink Window

- **C/R (Command and Response bit)** identifies the LPDU as a command or a response.
If C/R equals '0' LPDU is a command
If C/R equals '1' LPDU is a response
- **S:** distinguishes the first allocation of a private uplink.
- **X** Don't care.

2) UPLINK



Figure 19 - Uplink MAC Control Field format

Bits detailed description:

- **L (LPDU bit):** indicates the existence or absence of the LPDU on the frame
If L equals '1', LPDU exists on the frame.
If L equals '0', LPDU does not exist on the frame
- **D(0):** indicates the link direction,
If D(0) equals '1' is an uplink
If D(0) equals '0' is a downlink
- **R: indicates window request**
If R equals '1', OBE is requesting a window
If R equals '0', OBE is not requesting a window
- **C/R (Command and Response bit)** identifies the LPDU as a command or a response.
If C/R equals '0' LPDU is a command
If C/R equals '1' LPDU is a response
- **X** Don't care.

- LPDU

The LPDU has from 10 up to 129 bytes and carries information regarding the OBE User and one byte for control (Please see Table 3 and Figure 20).

LLC control field	Information field
1 byte	9 a 128 bytes

Table 3 - LPDU field format

LLC control field

M	M	M	P/F	M	M	1	1
---	---	---	-----	---	---	---	---

Figure 20 - LLC control field

- M: Modifier function bits
- P/F: Poll bit (command LPDU transmission)/ Final bit (response LPDU transmission)
 - Command LPDU transmission
 - If P=1 Poll
 - If P=0 No Poll
 - Response LPDU transmission)
 - If F=1 Final
 - If F=0 No Final

- Frame Check Sequence

The Frame Check Sequence field has two bytes which are used for error detection. The error is detected, by using the generator polynomial $X^{16} + X^{12} + X^5 + 1$ (the initial value is $FFFF_{16}$). Link address, MAC control and LPDU should be included in the calculation of the Frame Check Sequence.

Address establishment

The fixed equipment (the RSE) supports an SAP (Service Access Point) to broadcast and others SAPs to communicate privately with the mobile equipments (or OBEs) within the communication range. Each OBE also needs to have at least two SAPs to establish the communication, (Please see Figure 21). It is assumed that the Broadcast SAP is always active (actually it is in sleeping mode most of the

time). The OBE responds by creating a private LID and requests a PrivateSAP. When a frame containing a private LID is received by the RSE, it is created a new corresponding SAP.

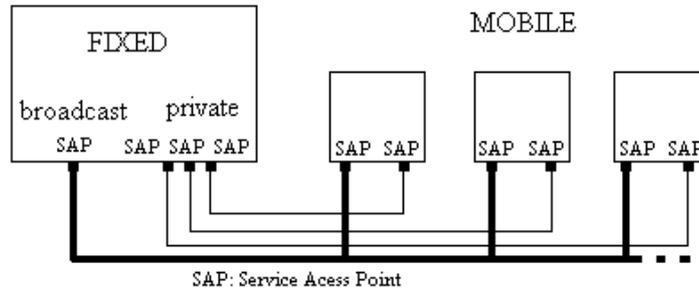


Figure 21 - Address establishment

Window Management

There are different situations of message exchange, each one with a specific time displacement. Both RSE and OBE send or receive according to the situation. Public and private Downlink/Uplink windows are distinguished by their LID, whether a Broadcast LID or a Private LID is present.

- **Uplink to Downlink transition:**

The mobile equipment transmits in an Uplink Window (please see Figure 22) and the corresponding response arrives in a Public Downlink Window. T_1 ($32\mu s$) is the uplink to downlink turn around time.

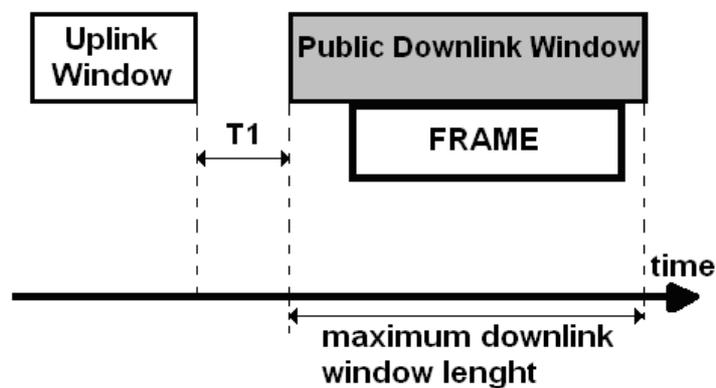


Figure 22 - Uplink to Public Downlink transition

- **Downlink to Downlink transition:**

The fixed equipment transmits a Downlink Window (please see Figure 23). If the following Window is also a downlink, there is no time gap. $T_2=0 \mu s$. (The maximum time of the downlink window is the time needed, by the RSE, to send the whole frame).

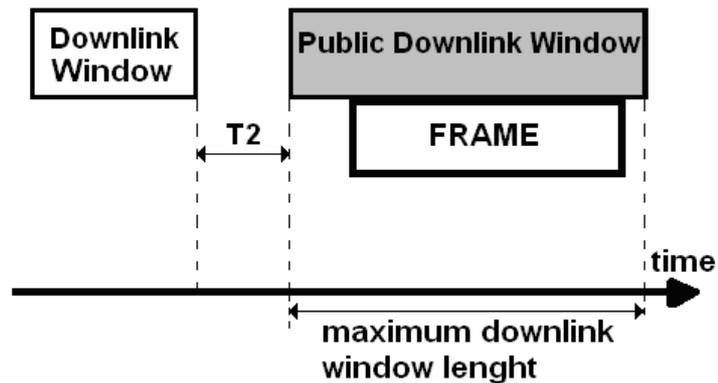


Figure 23 - Downlink to Downlink transition

- **Downlink to Uplink transition**

- **Private Mode (private LID):**

The fixed equipment transmits in a Private Downlink Window (please see Figure 24) and the corresponding response arrives in a Private Uplink Window. T_3 ($160 \mu s$) is the downlink to uplink turn around time, T_{4a} ($320 \mu s$) is the maximum time to start of transmission.

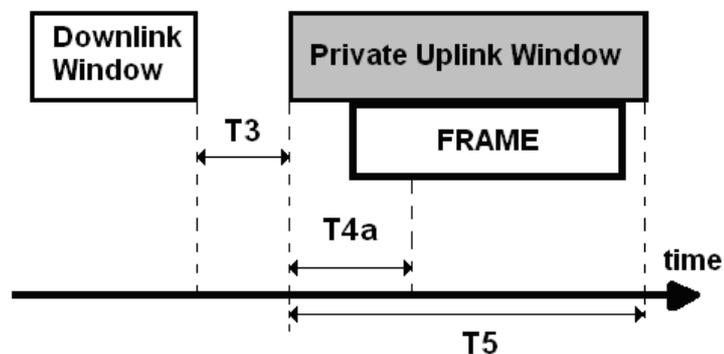


Figure 24 - Downlink to Private Uplink transition

➤ **Working in Public Mode (broadcast):**

The fixed equipment broadcasts a message in a Public Downlink Window and expects the response(s) in the following three consecutive Public Uplink Windows (please see Figure 25). T3 (160μs) is the downlink to uplink turn around time, T4b (32μs) is the maximum time to start of transmission and T5 (448μs) is the time duration of the uplink window

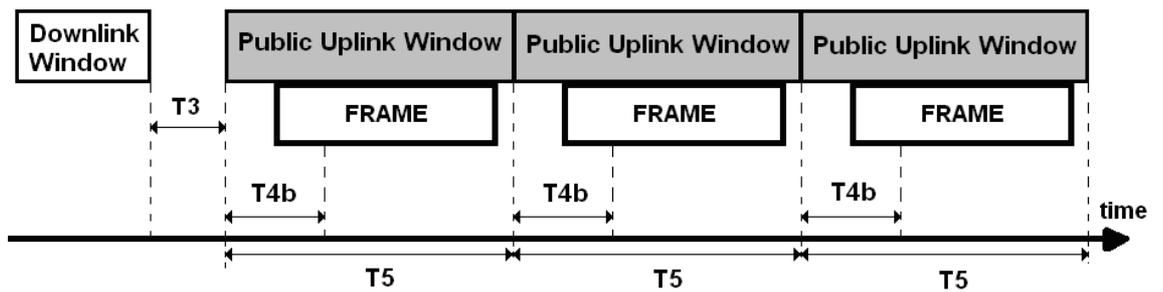


Figure 25 - Downlink to Public Uplink transition

• **Uplink to Uplink Transition**

In this case there is no time gap (Please see Figure 26).

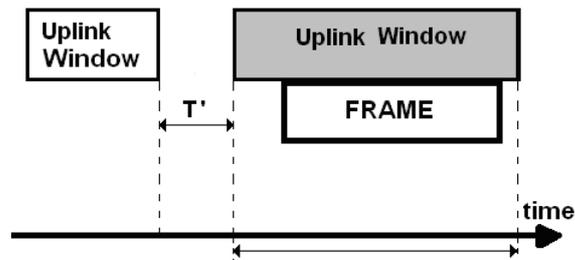


Figure 26 - Uplink to Public Uplink transition

2.2) Microcontrollers Communication Issues

This subchapter describes the communication issues between the two micro controllers present in the RSE. Both hardware and software questions are answered in the following chapters.

2.2.1) Hardware Connection - Serial Peripheral Interface (SPI)

The hardware connection between microcontrollers, Master and Slave, was established using the Serial Peripheral Protocol (SPI) protocol, available in both, (please see Figure 27).

Pins SCKL (SPI clock), SS\ (Slave Select), MOSI (Master Out Slave In) and MISO (Master In Slave Out), are standard SPI pins, while Busy and RTS (Request To Send) are stipulated for proper use of the protocol, for details please see the following paragraph *Application Interface*. This protocol is and Master/Slave unbalanced, which means that all the exchanges are controlled by the Master.

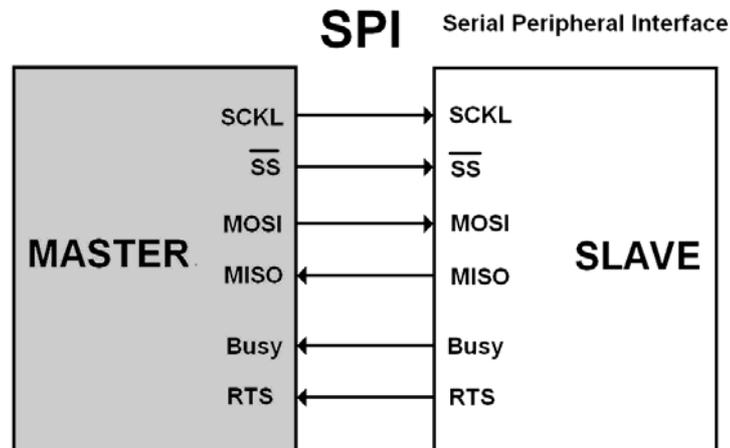


Figure 27 - SPI Hardware connections

2.2.2) Interface Software - Communication Protocol Master/Slave

This subchapter describes in detail the interconnection between Master and Slave modules. The figure below shows an overview of the existing interfaces (please see Figure 28). The next subchapters define each of the interfaces.

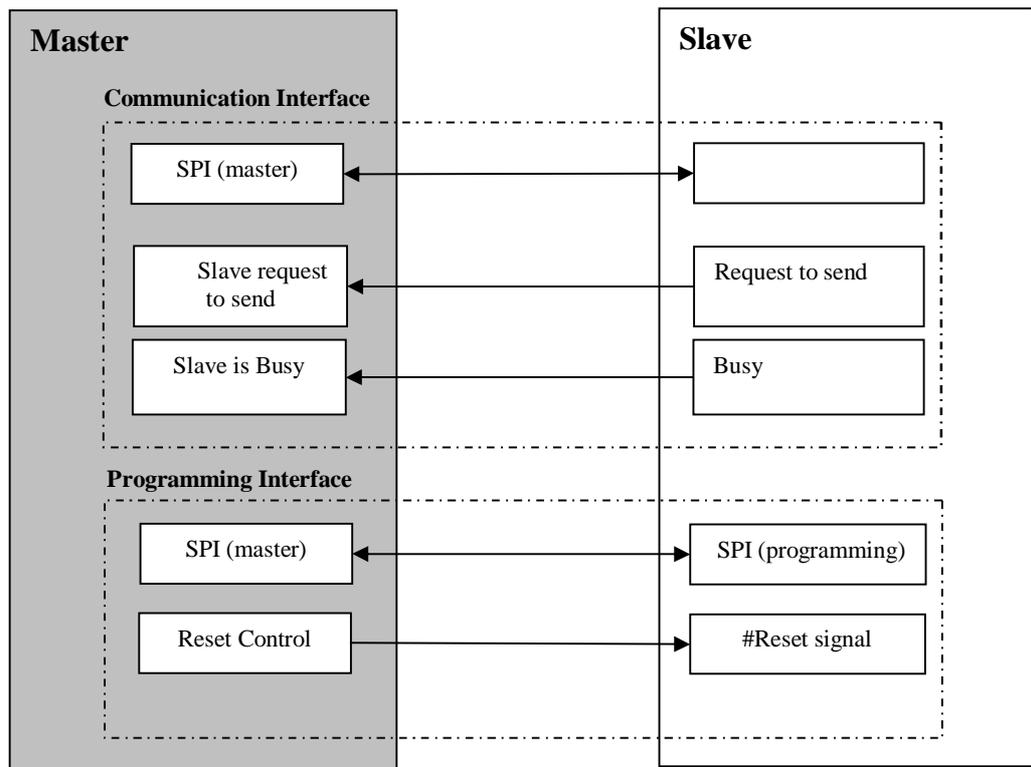


Figure 28 - Detailed SPI hardware connections, by interf

- a) Communication Interface:
- Interface used for communication application specific commands, LDR/MDR commands, etc
- b) Programming Interface:
- Interface used by the Master to program the Slave firmware

a) Application Interface

The application interface is used to exchange application information between the two microcontrollers. It is composed by the following components:

- SPI: Serial Peripheral Interface, protocol used for communication between a Microcontroller and external devices. It is a common module in microcontrollers. It is a master slave connection, and allows high speed communication.
- Request to Send: Digital signal used by the Slave to signal the Master that it has data to transmit. This signal must be used because SPI protocol is Master/Slave, and the communication must be started by Master. This signal is also used to signal the end of transmission from slave to master: RTS is active until the slave transmits the last byte. This way the master can determine the length of the message.

- **Busy:** Digital signal used by the Slave to indicate that is busy and cannot accept communication from the Master. When the master has a message to transmit, it will poll this signal for a pre-determined number of times, until it is allowed to transmit. If the maximum number of retries is exceeded, the master will do a protocol restart.

Physical Interface

The figure below shows the detailed connection between Master and Slave, (please see Figure 29).

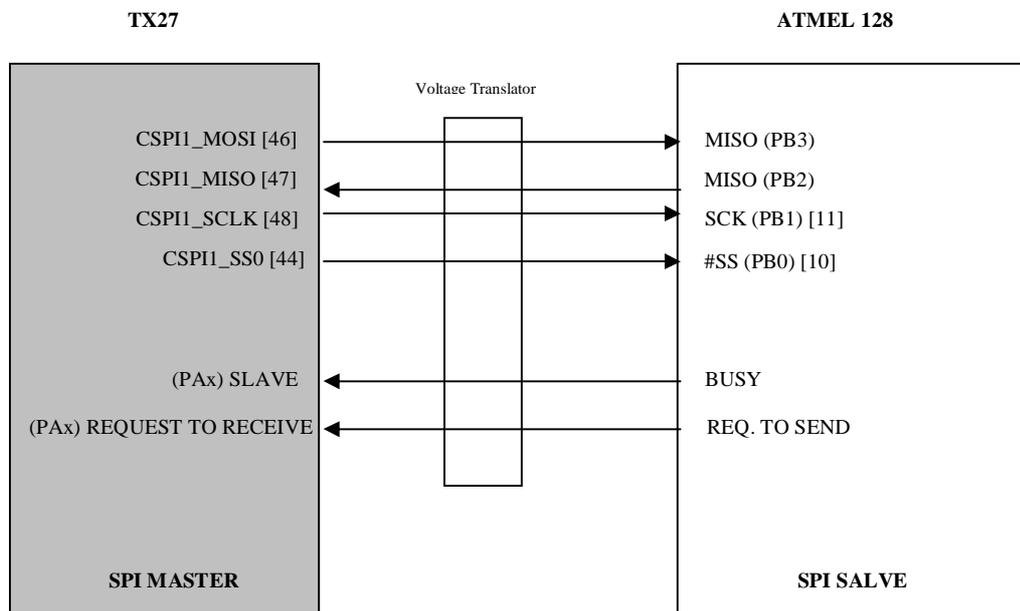


Figure 29 - Detailed SPI Connection, between Slave and Master

Due to the difference in performance, between Master and Slave, the speed of communication is limited by Slave performance. According to the slave reference manual, in SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high period should be:

- Low period: Longer than 2 CPU clock cycles.
- High period: Longer than 2 CPU clock cycles.

Considering these requirements and the clock frequency, of 16MHz => 62,5ns, SPI maximum SCK signal should be = $4 * 62,5 \text{ ns} = 250 \text{ ns}$ => 4MHz or Mbit's

SPI communication parameters:

- Less than 4 Mbit's
- Data length = 8bits
- Clock Polarity = base value of the clock is zero
- Clock Phase = data is read on clock's rising edge (low-> high transition) and data is changed on a falling edge (high-> low clock transition)

PROTOCOL SPECIFICATIONS and RULES

i) Frame Format

The structure of the frame to be exchanged between the master and slave is the following (please see Figure 30):

START	FRAME TYPE	FSN	SIZE	MESSAGE	DATA	CRC
-------	---------------	-----	------	---------	------	-----

Figure 30 - SPI protocol frame format

- **Start** (1byte) – STX ASCII Character 0x02
- **Frame Type** (4 bits) – Field that indicates the type of frame
- **FSN** (4 bits) – Frame sequence number used to keep track of last message transmitted/received. Master and slave have their own FSN, and they need to track the state of both FSNs. Each part has to maintain one FSN for outgoing messages, and another independent FSN for incoming messages. FSN = 1 to 15. After 15 it should be restarted to 1. 0 is a special value used only after startup
(The evolution of the FSN is described in the section Frame Sequence Number Rules.)
- **Size** (1byte) – Size of the field data in bytes
- **Message** (1 byte) – Field indicating the command sent from the master
- **Data** (0..240 bytes) – Data field, size is defined according command field
- **CRC** (2 bytes) - Cyclic Redundancy Check, with the dimension of 2 bytes

Frame Types

The frame type defines the kind of information handled on the frame, please see table Table 4. (Ack stands for acknowledge, and Nack for not acknowledge)

Type	Name	Description
1	Data	Frame used by master/slave to send data to the other parts
2	Ack	Acknowledge message used to confirm reception of a Data frame. In this case fields message and data have length = 0.
3	Nack	Not acknowledge message used to alert theta the message was received with errors. In this case fields message and data have length = 0 and FSN=0.

Table 4 - Frame Types between microcontrollers

CRC Calculation

The polynomial definition is: $x^{16} + x^{12} + x^5 + 1$ (CRC CCITT 16). The CRC calculation is based on all fields (including the START field), except from the CRC field itself. CRC start value is $FFFF_{16}$.

ii) Time Rules

- Timeout between retries (default 50ms)
- Number of retries in resend sequence: 3

iii) FSN (Frame Sequence Number) Rules

The following rules have to be implemented regarding the FSN manipulation:

- The FSN of a data packet is filled with the current counter value,
- The FSN of an ACK (NACK) packet is filled with the FSN of the data packet to ACK (NACK).
- On the reception of an ACK (NACK) packet, the FSN is compared to the latest data packet sent. If it is an ACK, the counter is increased on a hit.

- FSN should be used from 1 to 15
 - After power-up FSN is set to 0. This means that next frame received should be accepted and no FSN check should be done.
- Every time a protocol restart is detected (reception of a frame with FSN = 0) the application should be notified.

iv) Communication Rules

- While the slave keeps the busy signal active the master cannot send any information to the slave.
- Due to the SPI communication characteristics the slave can only send data when the master is transmitting. This means the slave should use the I/O line RequestToSend to signal the master that it has data to be transmitted. The slave should maintain the RTS signal until it transmits the last byte of the message. When the master detects the absence of this signal, it considers that the message is finished and stops transmitting the SCLK. This way, the master determines the length of the received message. To avoid *deadlock*, the master should maintain the SCLK signal up to a maximum of 500 bytes.
- Each frame received should be acked/nacked with exception for the ACK/NACK itself. This includes messages in both directions. For instance, when the slave sends a message (Ex: “**status_reply**”) to the master in result of a previous master->slave command (Ex: “**request_status**”), the master must acknowledge the received message with the correct FSN.
- If resend sequence is exhausted, i.e. maximum number of retries is sent, the application should be notified

v) Communication sequences

a) Valid sequence, without errors or timeouts. In the first situation Master starts communication, please see Figure 31, and in the second it is the Slave to start, please see Figure 32.

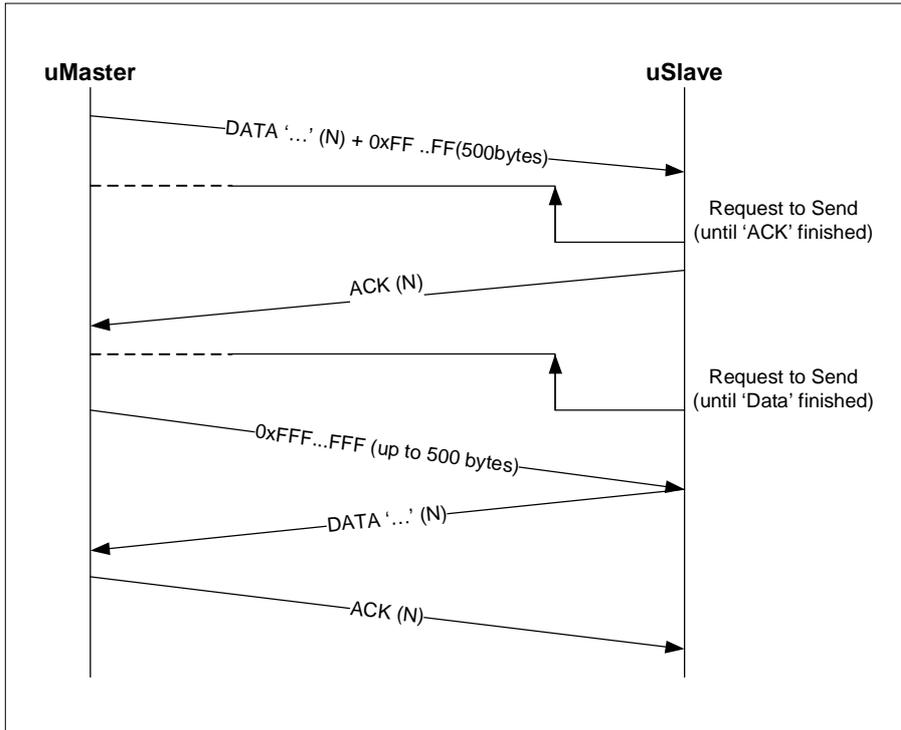


Figure 31 - Valid sequence where master initiates communication

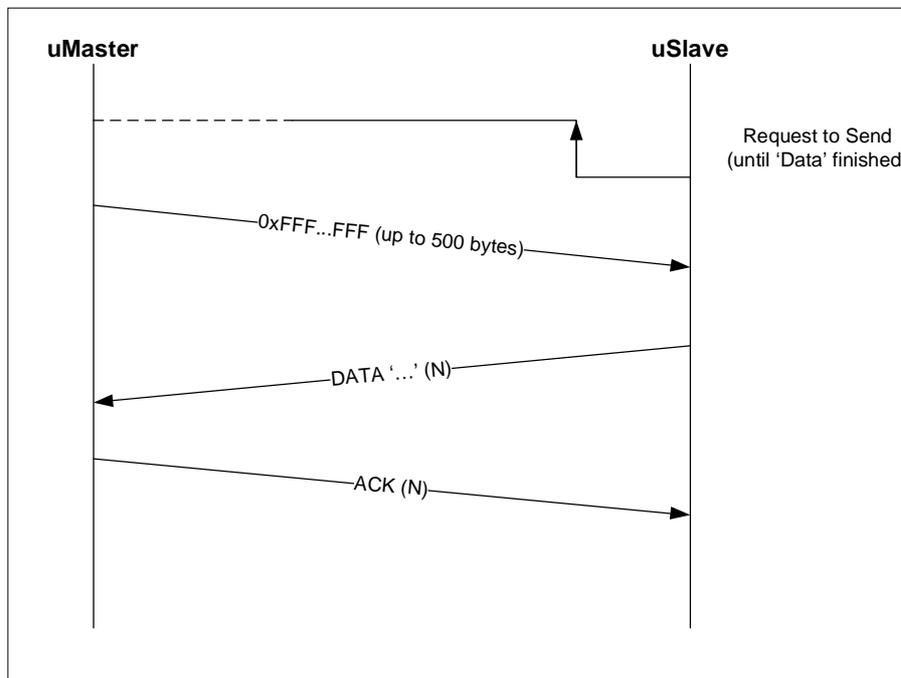


Figure 32 - Valid sequence where slave starts communication

b) Valid sequence but with timeouts occurred in the process. In the first situation Master starts communication, please see Figure 33, and in the second it is the Slave to start, please see Figure 34.

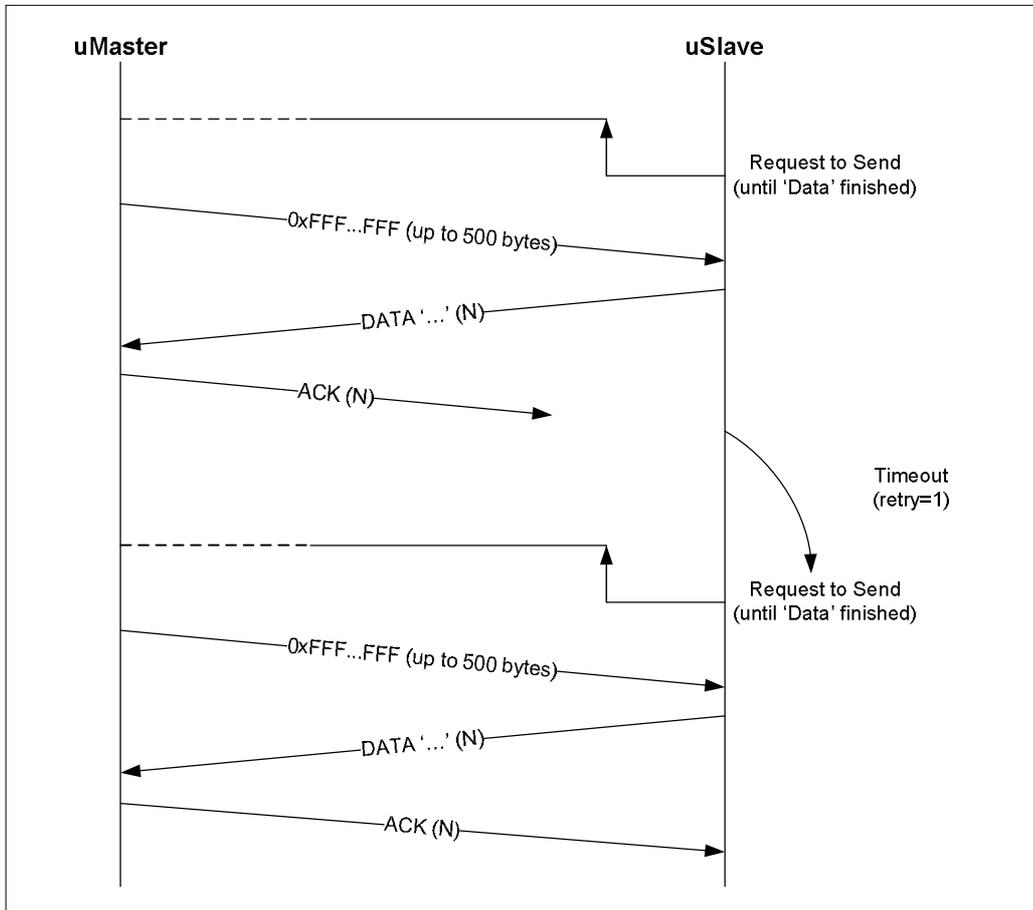


Figure 33 - Valid sequence with timeout on the slave side

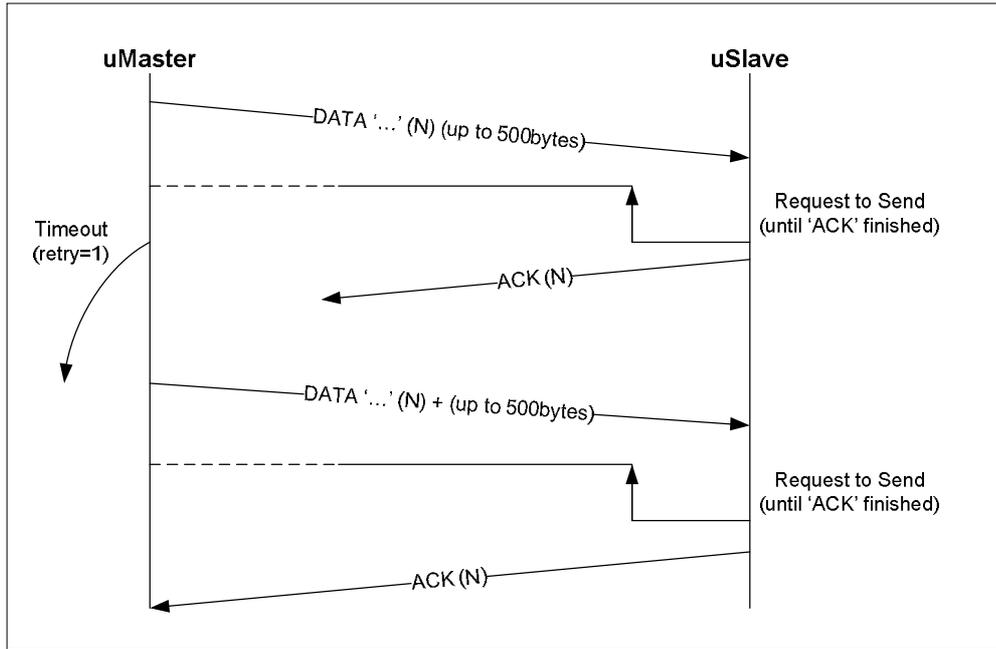


Figure 34 - Valid sequence with timeout on the master side

vi) Types of Errors

Errors can occur all along the process, for example, a missed byte at the reception, can generate a bad crc, which implies a wrong message. So, the software must be prepared for those situations. The following table lists the possible errors and how to handle the problems, in those cases. (Please see Table 5)

Error Type	Proposed Action
Invalid Frame Received (both master or slave)	Reply with NACK message.
Frame with Bad CRC received (master or slave)	Reply with NACK message.
Receiver detects command while expecting ACK or NACK	Discards message. Keeps waiting for ACK. Does not enter the resend sequence and does not send NACK.
Receiver detects FSN larger than expected	Discards. Enters the resend sequence. If resend limit is exceeded a protocol restart should be performed and the application notified.
Receiver detects duplicate FSN	Discards. Enters the resend sequence. If resend limit is exceeded a protocol restart should be performed and the application notified.

Table 5 - Error situations in frames exchange

vii) Application Messages

The list of commands exchanged, in the application layer are listed in the following table, please see Table 6.

Message	Name	Direction	Description
1	Enable	Master to Slave	Enable RF
2	Disable	Master to Slave	Disable RF
3	Configure	Master to Slave	Configure RF Channel , Power, etc.
10	Send MDR	Master to Slave	Send MDR Data
11	Received MDR	Slave to Master	Received MDR Data from the OBE
12	Send LDR	Master to Slave	Send LDR Data
13	Received LDR	Slave to Master	Received LDR Data from the OBE
20	Status Request	Master to Slave	Status request
21	Status Information/Reply	Slave to Master	Status Information/reply

Table 6 - Application messages

b) Programming Interface

This Interface allows the Master to load firmware into the Slave. The Slave is ATmega128 from Atmel, running on a 16MHz clock and powered with 5V.

For more details consult the microcontroller datasheet. This microcontroller provides hardwired support for several on-system programming modes as can be seen on the datasheet at section Memory Programming (ATmega128 datasheet page 286). The programming protocol is specified in that section.

For simplicity, it is recommend Serial Programming using the SPI interface (as seen on ATmega128 datasheet page 300). Note that, although the very same SPI hardware is used for memory programming, but MISO and MOSI signals are mapped to different pins, as shown on that page, for programming it. The SCK (clock) signal remains on the usual pin.

The programming procedure has full support in hardware and does not need extra code in the slave microcontroller.

Physical Level

The figure below represents the physical connection between the Master and the Slave in order to support the programming interface (Please see Figure 35).

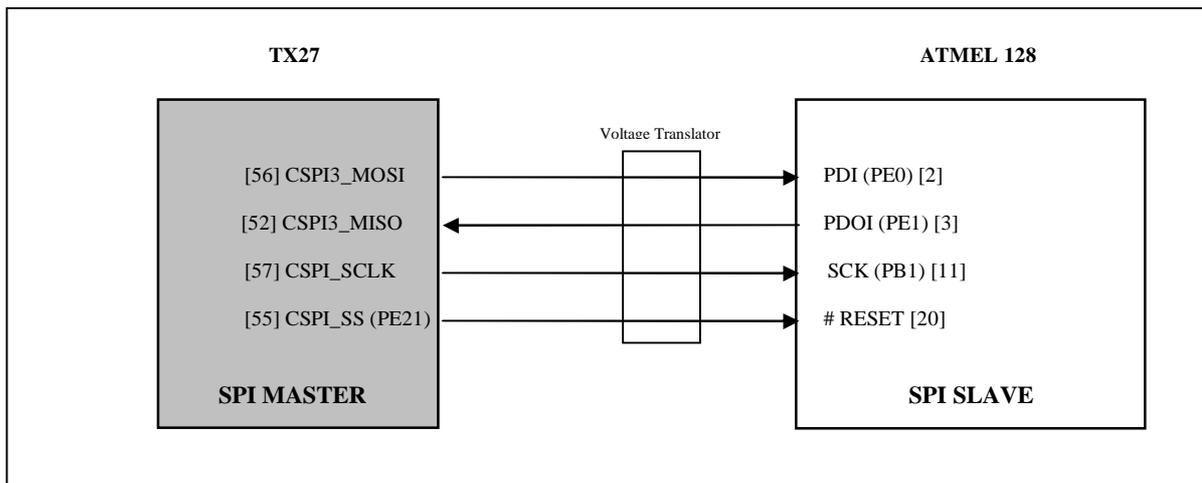


Figure 35 - Block Diagram of the RSE for programming interface

The following picture, displays the block diagram of all the components of the RSE (Please see Figure 36).

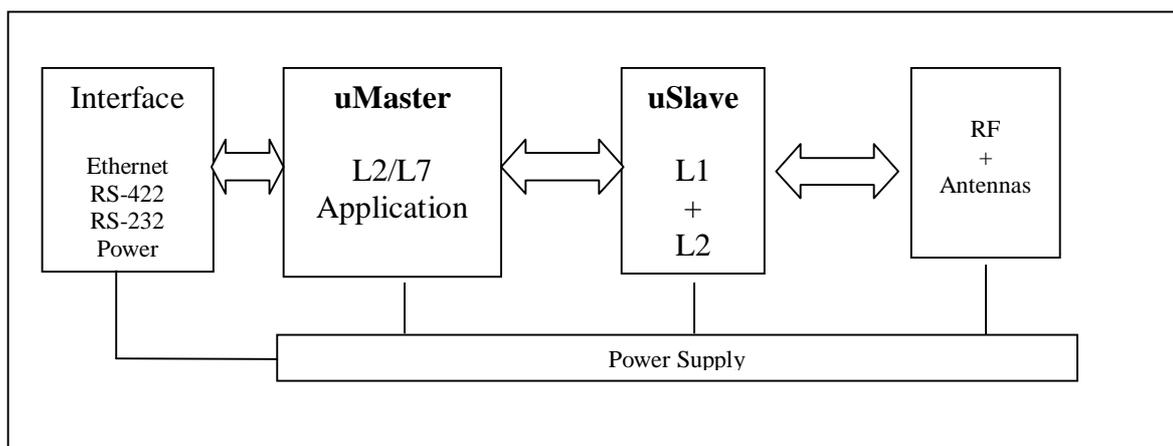


Figure 36 - Block Diagram of the RSE

3) Developed Software

The project has two specific parts, so the software was developed separately. The first part is responsible for the communication between the OBE and the RSE (please see sub chapter 3.1), and in the second part the software deals with the communication issues between the Microcontrollers. In the end, both parts will be integrated together, and able to interact with each other.

The software was developed in C language, using the editor *AvrEdit* [10], and the microcontroller loader *Pony Prog 2000* [11] and [12].

AvrEdit

[10] This windows program is not a compiler itself, but an editor / IDE to easily use the free GNU C-compiler avrgcc. It features configurable syntax colouring, a simple file browser, a makefile generator and even a small bitmap-to-array editor.

Pony Prog 2000

[11] and [12] It is a serial device programmer *software* with a user friendly GUI framework available for Windows95/98/ME/NT/2000/XP and Intel Linux. Its purpose is reading and writing every serial device. At the moment it supports I²C Bus, Microwire, SPI eeprom, the Atmel AVR and Microchip PIC micro.

3.1) Software Functions between the OBE and the RSE:

Software to simulate the RSE behavior in the interaction with the OBE was also developed. There was also the need to develop some functions of the next layer (the application layer), to test some conditions in the MAC layer.

The service primitives to communicate with the next layer LLC are: MA-DATA-req (LID,LPDU) and MA-DATA-ind(LID, LPDU), shown in Figure 37.

Some software functions, regarding the physical layer, of the OBE are already developed.

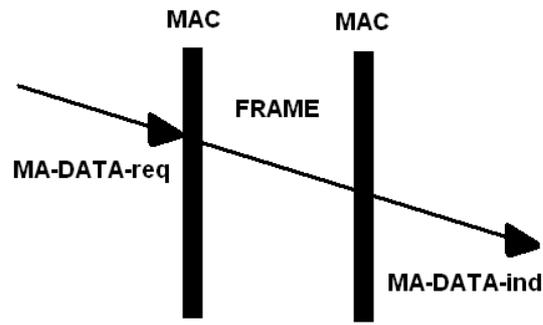


Figure 37 - Service Primitives of the MAC layer

RSE

The following state machine, please see Figure 38, describes the behaviour of the RSE, when an OBE is range.

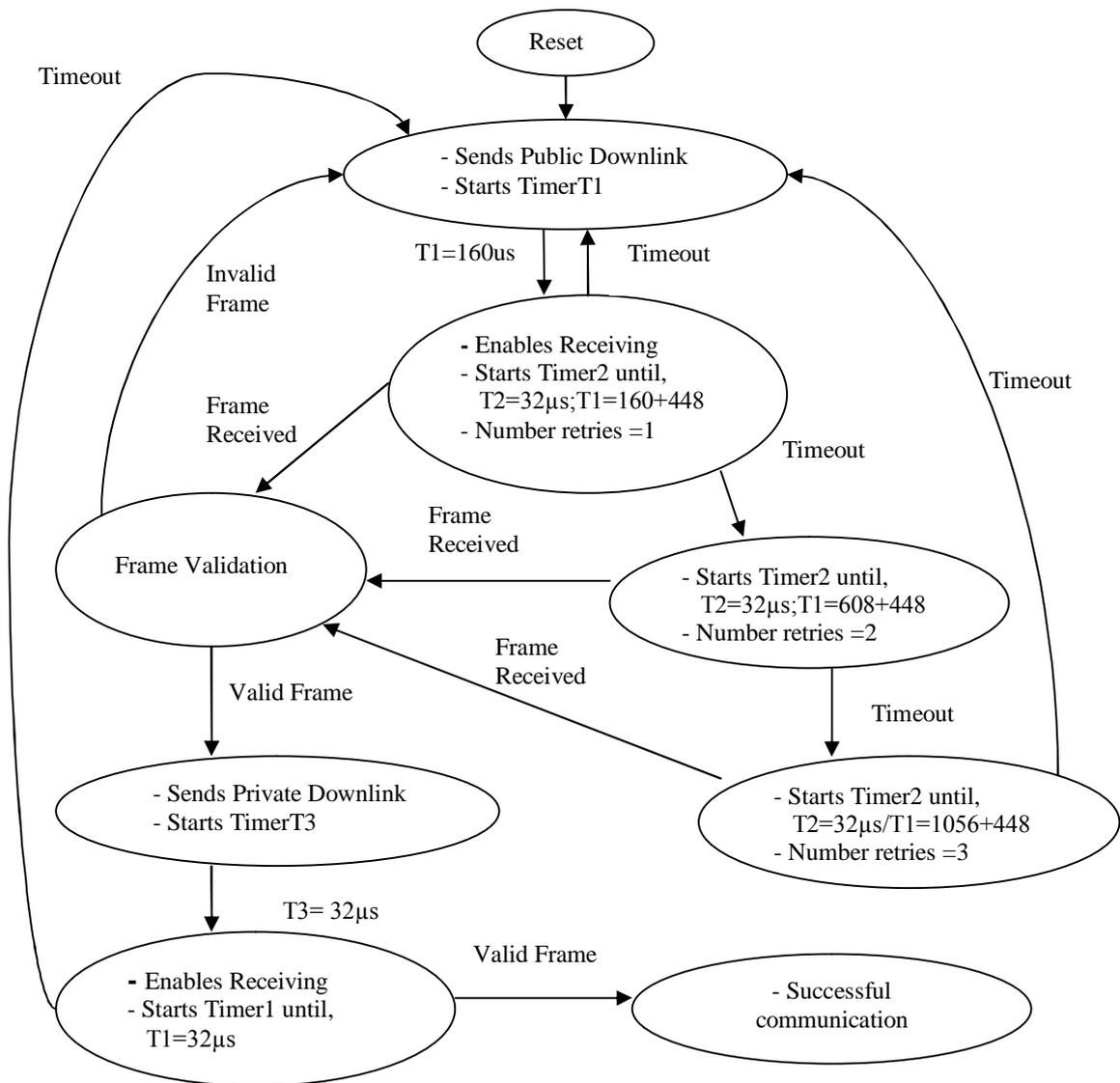


Figure 38 - RSE State machine

OBE

The software regarding the OBE is still ongoing. So, here are just presented the OBE's basic principles and the algorithm to implement the software (please see Figure 39).

The following paragraphs explain the process, state by state, and the corresponding transition conditions are indicated. The discussion is separated from Public to the Private Allocation, beginning in the point where the state machine finishes the common steps.

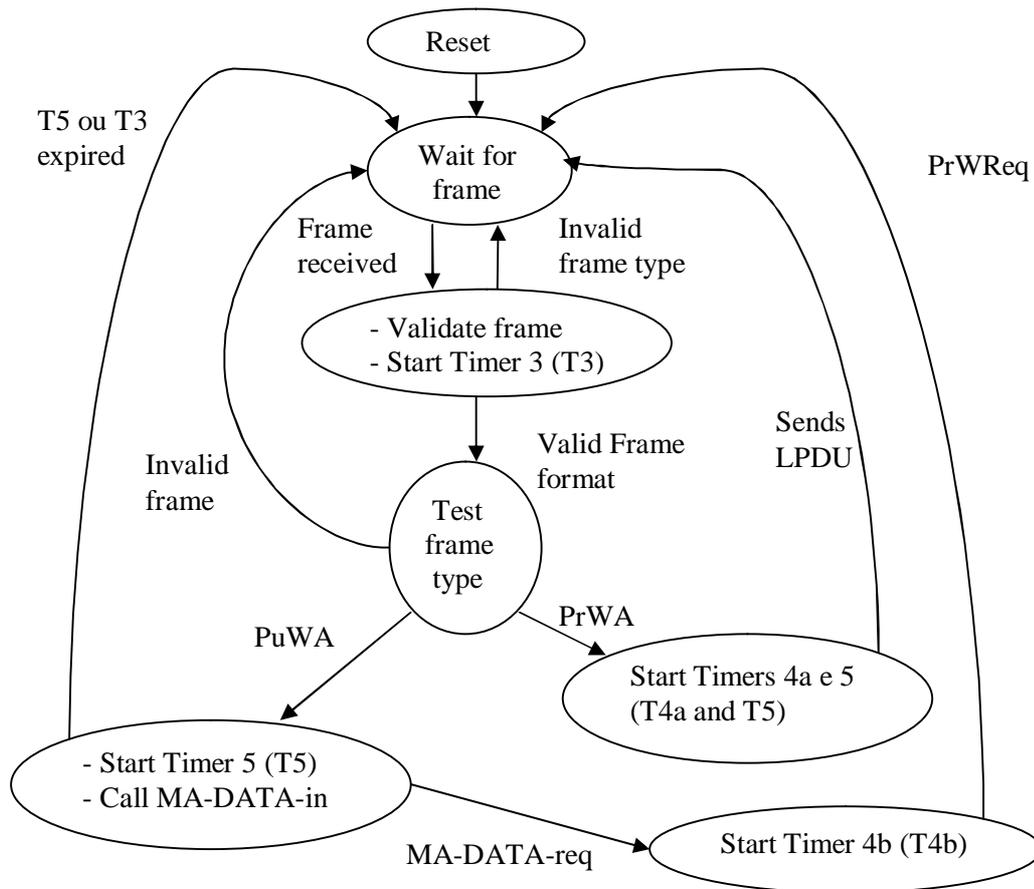


Figure 39 - State Machine describing the MAC layer behavior of the OBE

'Wait for frame' State:

In this state, the OBE is waiting for a new input proceeding from the RSE, whether it is the first or any other frame, during the communication.

Transition condition: the reception of a new frame.

'Validate Frame/ Start Timer3 (T3)' State:

In this state, the frame is validated by comparison with the pre-defined format, particularly the Cyclic Redundancy Check (CRC). Timer3 (T3) is enabled. This timer is used to control both situations (Public or a Private Uplink Window)

Transition conditions: If an error occurs during the validation, the process returns to 'Wait for frame'. If not, it advances to the following state.

'Test frame type' state:

Here the distinction is made between a PuWA (Public Window Allocation) and PrWA (Private Window Allocation)

Transition conditions: If the frame received is a PuWA, advances to 'Start Timer5 (T5)/ Call the MA-DATA-ind' state. If it receives a PrWA, it advances to 'Start Timers4a, 5 (T4a, T5)'.

Transition condition: PrWReq is sent

Private Window Allocation branch (please see Figure 41)

If in the 'Test frame type' state a PrWA is detected, the following state will be 'Start Timers4a, 5 (T4a, T5)'.

'Start Timers4a, 5 (T4a, T5) state:

Both Timer4a (T4a) and Timer5 (T5) are enabled, to proceed with information sent.

Transition condition: The pending LPDU is sent.

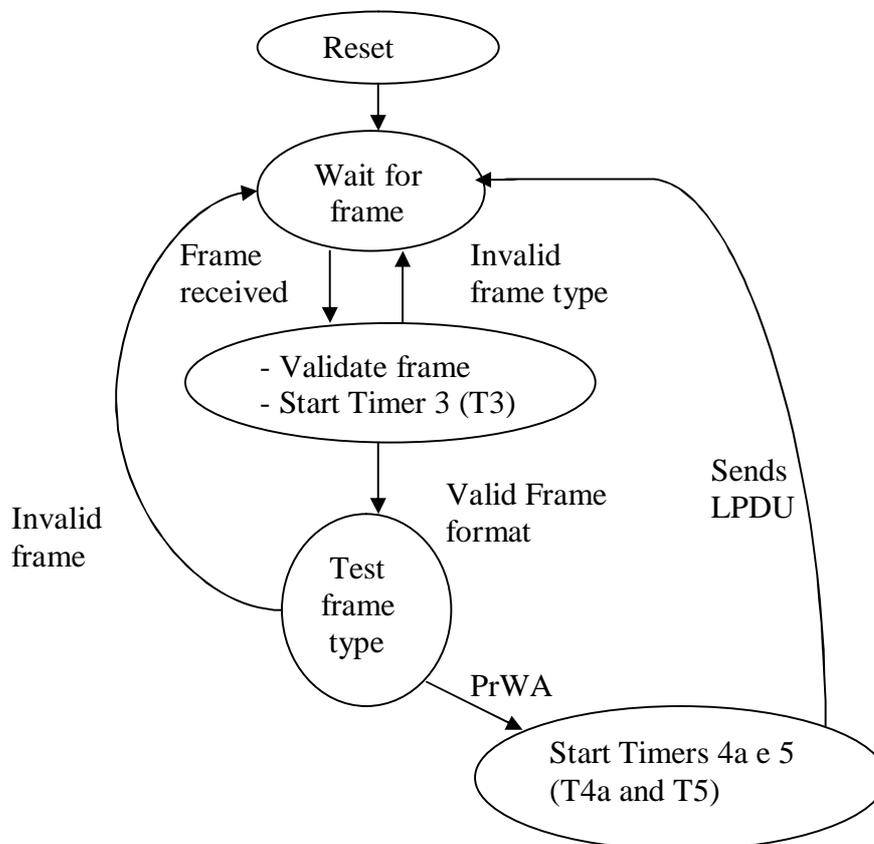


Figure 41 - State machine concerning only the Private Uplink branch

a) Developed Functions

For now, only the software concerning the RSE (send and receive) is done. The software regarding the OBE is still on going. Here, just the RSE functions are

presented.

Due to the differences of behaviour between the public and private (number of consecutive windows and time constrains), there was the need of creating different functions for the Downlink. The following functions implement the referred state machine (please see Figure 38).

- ***void public_downlink (*array_out_public)***

This function executes a public downlink, by sending the information, of parameter **array_out_public*, according to frame format (please see Figure 13), from the RSE into the OBE. In this case data is generic for all vehicles. The time constrains are described in 2.1.2) European Standard 12795. In public mode, there are three consecutive public windows. So, this function has to send the information and start the timer, to control the turn around time in order to start the receiving mode, (please see Figure 38).

- ***void private_downlink(*array_out_private)***

This function executes a private downlink, by sending the information, of parameter **array_out_private*, according to frame format, (please see Figure 13), from the RSE into the OBE. In this case data is specific for each vehicle. The time constrains are described in 2.1.2) European Standard 12795. In private mode, there is a single window. So, this function has to send the information and start the timer, to control the turn around time in order to start the receiving mode, (please see Figure 38).

- ***void uplink()***

This function executes an uplink, by receiving the information according to frame format, (please see Figure 13), from the OBE into the RSE. This function can receive generic or specific data. The time constrains are described in 2.1.2) European Standard 12795. The timer is set and if a timeout occurs, this function terminates, (please see Figure 38).

All this functions, after the testing and the quality control will be packed, according to the Service Primitives of the MAC layer.

b) Test Conditions

The test conditions are in accordance with the European Standard EN 12795. The software is tested first in well known conditions with specific values being inserted step by step. Only after all the laboratorial tests are done, the software will be proved in real conditions. The RSE is still in prototype phase, for testing the software it is used a development kit, containing a microcontroller similar to the one used in the RSE (Please see Figure 42). The times values imposed by the Standard were confirmed using an oscilloscope.

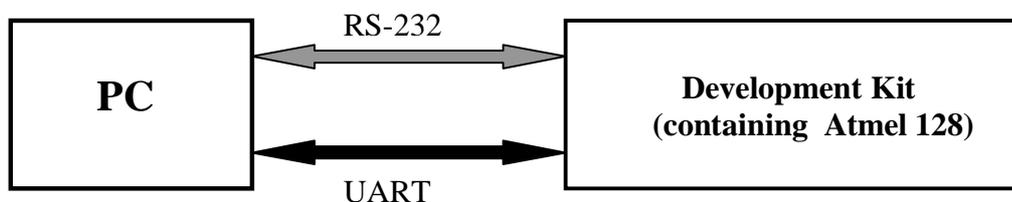


Figure 42 - Hardware connection, to test the software between the RSE and the OBE

NOTES:

1. *Brisa* is responsible for the implementation of standard 12795, from LLC (Logical Link Control) sub layer, up to all the layers above, including Application layer.

3.2) Software functions between the microcontrollers:

To establish the connections between the microcontrollers, many functions were developed to assure the correct data transfer. A virtual set of stipulated layers was created (please see Figure 43). Development was planned and tested from bottom to up, layer by layer.

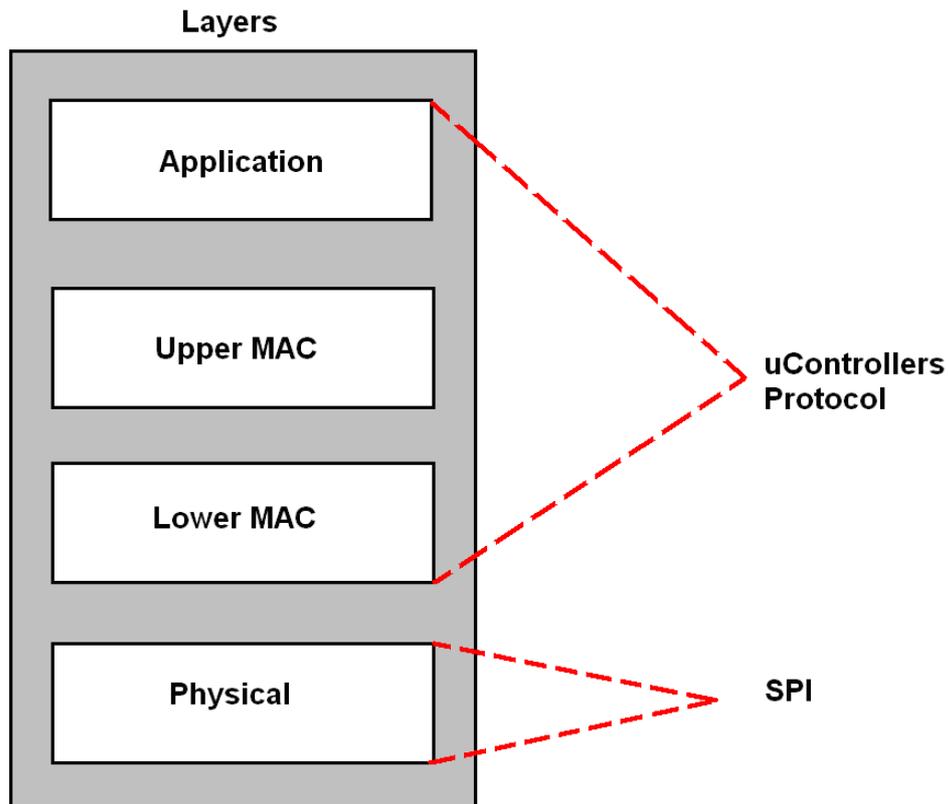


Figure 43 - Layers diagram of the microcontroller's protocol

a) Developed Functions

1) Physical layer:

The first step was to create the lowest level of communication via SPI hardware. The byte is the smallest piece of information that can be exchanged. So, functions were created, from the lowest to the highest level of complexity.

i) Byte Transfer

The basic function is the byte transfer one, in both ways from Master to Slave and vice-versa. When the Master is transmitting mode the Slave is receiving mode (please see Figure 44), the functions are listed below:

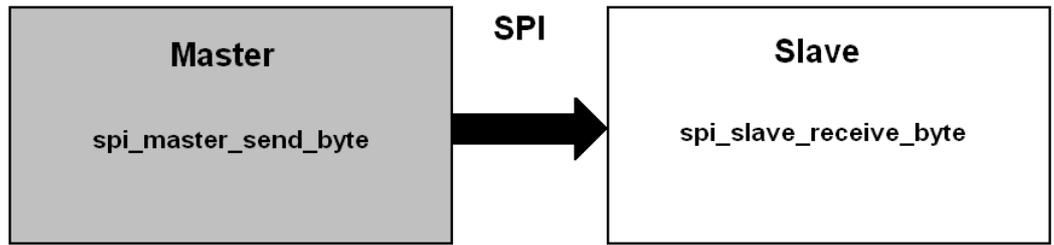


Figure 44 – Software functions running, while Master is sending a byte

- ***int spi_master_send_byte (char value);***

In this function, the Master sends a byte to Slave, via SPI hardware connection. The parameter 'value' is the content to be sent. It returns '0x00' in case of success and '0xFF' in case of error.

- ***int spi_slave_receive_byte();***

In this function, the Slave receives a byte from Master, via SPI hardware connection. There are no parameters. It returns '0x00' in case of success and '0xFF' in case of error.

When the Slave is transmitting mode the Master is receiving mode (please see Figure 45), the functions are listed below:

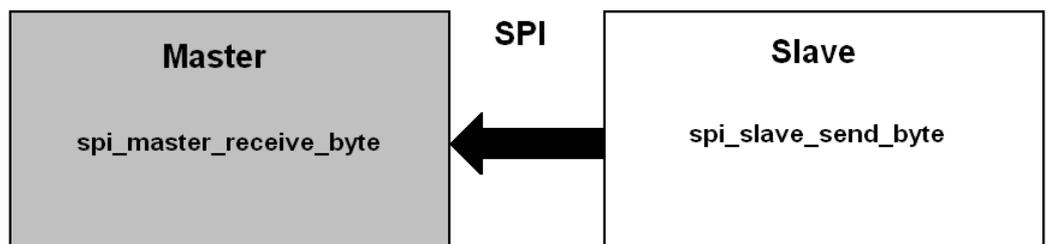


Figure 45 - Software functions running, while Slave is sending a byte

- ***int spi_slave_send_byte (char value);***

In this function, the Slave sends a byte to Master, via SPI hardware connection. The parameter 'value' is the content to be sent. It returns '0x00' in case of success and '0xFF' in case of error.

- ***int spi_master_receive_byte();***

In this function, the Master receives a byte from Slave, via SPI hardware connection. There are no parameters. It returns '0x00' in case of success and '0xFF' in case of error.

ii) String and Buffer Transfer

After the first functions were done and tested, the next step was to send and receive strings and also buffers, in both master and slave microcontrollers (strings and buffers are a sets of bytes, the difference between them is that the string has an ending character '\0'). Using the previous software functions and applying the same methodology, as in byte transfer, the functions trading buffers and strings were created. The summary of all the functions are listed below (Please see Table 7).

While the master is in sending mode, the slave has to be in receiving mode so:

- ***int spi_master_send_string(* str);*** and ***int spi_master_send_buffer(* buf);***
- ***int spi_slave_receive_string();*** and ***int spi_slave_receive_buffer();***

While the slave is in sending mode, the master has to be in receiving mode so:

- ***int spi_slave_send_string(* str);*** and ***int spi_slave_send_buffer(* buf);***
- ***int spi_master_receive_string();*** and ***int spi_master_receive_buffer();***

SPI Software functions (physical layer)		
Action Described	Master	Slave
RSE sending a byte and OBE receiving a byte	int spi_master_send_byte (char value);	int spi_slave_receive_byte();
OBE sending a byte and RSE receiving a byte	int spi_master_receive_byte();	int spi_slave_send_byte (char value);
RSE sending a string and OBE receiving a string	int spi_master_send_string (char *str);	int spi_slave_receive_string();
OBE sending a string and RSE receiving a string	int spi_master_receive_string();	int spi_slave_send_string (char *str);
RSE sending a buffer and OBE receiving a buffer	int spi_master_send_buffer (char *buf);	int spi_slave_receive_buffer();
OBE sending a buffer and RSE receiving a buffer	int spi_master_receive_buffer();	int spi_slave_send_buffer (char *buf);

Table 7 - SPI Software functions (physical layer)

After this stage, all functions were consolidated under a software primitive, named spi.c and the respective header file spi.h.

2) Lower MAC layer:

This layer handles all the validation issues related with sending and receiving frames, according to the established format. In this stage, the handled fields are the *start*, *size* and *crc*, (Please see Figure 46). The fields, analysed in this function are described one-by-one:

- **start:** it is in this function that **start** flag detection is made, searching for the value 0x02
- **size:** in this stage, the content of the field isn't analysed if it is coherent or not with the rest of the frame, only if it larger or smaller than the permitted.
- **crc:** in here crc of the received frame is calculated, ignoring the last received bytes, after this it is compared with the last two bytes, if it matches the frame is accepted for a further layer if it has a bad crc, the frame is discarded.

START	FRAME TYPE	FSN	SIZE	MESSAGE	DATA	CRC
-------	------------	-----	------	---------	------	-----

Figure 46 - Fields handled by the Lower MAC layer

The developed functions for the lower MAC layer are listed bellow:

1. *int master_send_lower_mac (* str);*
2. *int slave_send_lower_mac (*str);*
3. *int master_receive_lower_mac();*
4. *int slave_send_lower_mac();*

3) Upper MAC layer:

This layer handles responses between microcontrollers, after the reception of a frame. It responds with an ack (acknowledge) if the frame is valid and with a nack (not acknowledge) if the frame is not valid, according to protocol rules. If a frame trade occurs without errors FSN is increased. It is in here that timeouts are controlled.

In this stage, the handled fields are the frame type and fsn, (Please see Figure 47). The fields, analysed in this function are described one-by-one:

- **frame type:** it is in this function that **frame type** is checked
- **fsn:** the received fsn is compared with the expected one, and if they mismatch the frame is discarded.

START	FRAME TYPE	FSN	SIZE	MESSAGE	DATA	CRC
-------	---------------	-----	------	---------	------	-----

Figure 47 – Fields handled by the Upper MAC layer

The developed functions for the upper MAC layer are listed bellow:

1. *int master_send_upper_mac (* str);*
2. *int slave_send_upper_mac (*str);*
3. *int master_receive_upper_mac();*
4. *int slave_send_upper_mac();*

4) Application layer:

In this higher level layer, the function interprets not the format but the content of the fields. If no error occurred and the command, sent by the Master is valid, the function executes it.

In this stage, the handled fields are size, Message and Data, (Please see Figure 48). The fields, analysed in this function are described one-by-one:

- **size:** here the coherency of size is tested, if the received size isn't in accordance with the message the frame is discarded.
- **message:** this field message, contains the type of command to execute, in this stage the type is analysed and compared with the list of commands, and if there is no error, the command is executed.
- **data:** in here the content of data field is gathered according to the command to execute.

START	FRAME TYPE	FSN	SIZE	MESSAGE	DATA	CRC
-------	---------------	-----	------	---------	------	-----

Figure 48 – Fields handled by the Application layer

The developed functions for the upper MAC layer are listed below:

1. *int master_application ();*
2. *int_slave_application ();*

b) Test Conditions

The purpose of test conditions is to bring the experiment near, as close as possible, to reality. So, two development kits were used, one as a Master and other as a Slave, each one connected with a PC, and between them with the SPI connection (Please see Figure 49).

The results of these tests show that the software is robust enough to deal with the error situations depict in Table 5 .

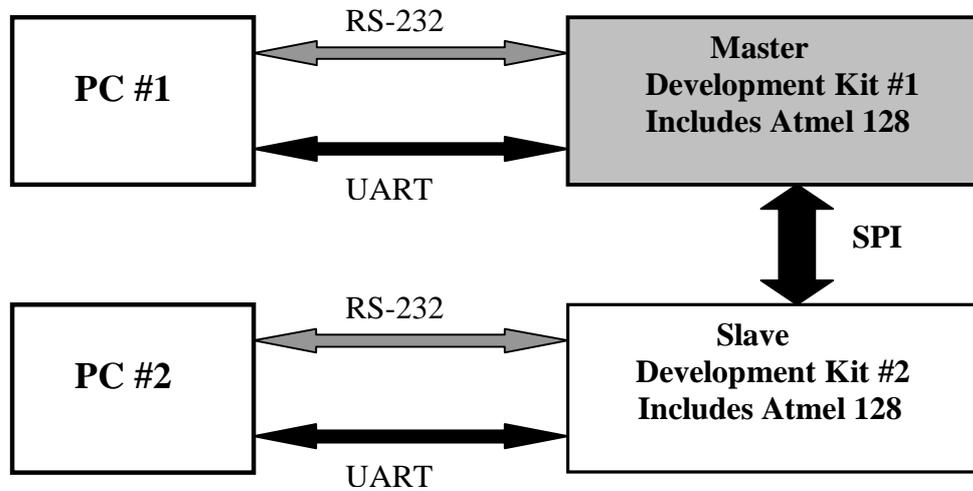


Figure 49 - Hardware connection, to test the software between RSE and OBE

NOTES:

1) All the **.c** files have the respective header file **.h** (for instance, for *file_rse_#1.c* file it was also created *file_rse_#1.h*).

2) All the files concerning the Master, in subchapter 6.2) “Functions between the microcontrollers”, are only for test purposes, because the Master is ruled by our partner *Brisa*.

3) Debugging is a hard task, because it takes a great amount of time, loading the software into the microcontrollers every a change is done.

4) Conclusions and Future Work

This chapter depicts the conclusions of this work, divided by the different aspects in study and according to the situation, and also the future work describing the possible further improvements, in order to upgrade the initial version.

CONCLUSIONS:

- **MDR versus LDR**

MDR technology, as expected, is better than LDR technology due to the higher data rate and the presence of information security protocols in MDR which are absent on LDR. The differences of baud rate, between both technologies, 32Kbits/s for the LDR and 256kbits/s or 512 kbits/s for the MDR, allows more versatility and robustness in the communication. The use of standard EN12795 as reference to establish the rules, guarantees that transfer security is highly increased.

The developed hardware for this project is able to work, sending and receiving frames, with both technologies LDR and MDR. The RSE starts communication, by sending LDR and MDR alternatively and waits for a response of the corresponding sent frame, and completes the rest of the frames exchange in the respective mode.

In terms of software, the upper layers need to be able to reconfigure the operating mode constantly, according to the technology being used at that moment.

- **RSE**

- a. **Communication between RSE and OBE software:**

Regarding this part, the presented RSE state machine complies with the demanded features and constraints required by EN 12795. Moreover, it is robust to communication errors and/or unexpected messages. Multiple error case scenarios, in communication, were considered and the software is robust enough to deal with those situations.

- b. **Communication between Microcontrollers software:**

Concerning this part, the presented RSE state machine complies with the demanded features and constraints required by the existing protocol. Multiple error case scenarios, in communication, were considered and the software is robust enough to deal with those situations.

- **OBE**

The OBE is still on going, in terms of Hardware and Software. The Hardware is in Prototype phase. There are already a few functions done, regarding the Physical Layer working.

This approach is still simple enough to be implemented in a low-cost microcontroller with typical features. This is an ongoing work which will culminate with robust service primitives for the MAC layer. The next step will be to close up the prototype and move on to the field tests.

FUTURE WORK

This work can be improved by programming some specific functions, on the microcontrollers interface, in assembly instead of C language. This will reduce the processing time and will allow the microcontroller to reduce the response times, to one another.

There is a software function that still needs to be developed and tested, integrating the OBE/RSE and the microcontrollers' software into a main function.

APPENDIX A – References

- [1] [EN12795] EN 12795:2003 Dedicated Short-Range Communication (DSRC) - DSRC Data link layer: Medium Access and Logical Link Control (review);

- [2] <http://www.identecsolutions.com/electroniclicenseplate.html>, (site visited in September 2009);

- [3] <http://www.cantinhodoemprego.com/index.php/uteis/outros/583-dispositivo-electronico-de-matricula-em-todos-os-veiculos-automoveis-chip.html> (site visited in November 2009);

- [4] <http://www.hindawi.com/journals/wcn/2009/576217.html> (site visited in September 2009);

- [5] http://en.wikipedia.org/wiki/IEEE_802.11p, (site visited in September 2009)

- [6] [EN12253] European Standard EN 12253, Comité Européen de Normalisation (CEN), Jul. 2004- Physical layer using microwave at 5.8 GHz;

- [7] [EN13372] European Standard EN 13372, Comité Européen de Normalisation (CEN), Jul 2004 - DSRC profiles for RTTT applications;

- [8] N. Almeida, R. Abreu, J. N. Matos, N. B. Carvalho, J. S. Gomes, "Low Cost Transceiver for DSRC Applications", Asia-Pacific Microwave Conference, Dec. 2006;

- [9] R. Abreu, N. Almeida, J. N. Matos, N. B. Carvalho, J. S. Gomes, "A Homodyne Low Cost Uplink Receiver for Digital Short Range Communication Systems", Vehicular Technology Conference, Spring 2007;

[10] http://www.avrfreaks.net/index.php?module=Freaks%20Tools&func=viewItem&item_id=370 (site visited in December 2009);

[11] <http://www.lancos.com/ppwin95.html>, (site visited in September 2009);

[12] <http://software.informer.com/getfree-ponyprog2000-wiki/> (site visited in December 2009);

APPENDIX B - Glossary

Definitions

- Downlink: communication channel on which fixed equipment transmits its information.
- Fixed equipment: fixed communication facility with one or more downlink channels and, optionally, one or more uplink channels.
- Link Identifier: unique address used for addressing the mobile equipment.
- Mobile equipment: mobile communication facility capable of receiving information from the fixed equipment on the downlink and, optionally, also capable of transmitting information to the fixed equipment on the uplink.
- Service access point: interface point between data link layer and application layer, that has a unique Link Identifier and that allows layers to communicate.
- Uplink: communication channel on which mobile equipment transmits its information.
- Window: period of time during which the physical medium is allocated either to the fixed or mobile equipment.

Acronyms

- ACK: acknowledge
- CAN: acknowledge command/ response
- BST: Beacon Service Table

- C/R: Command/ Response
- DSRC: Dedicated Short Range Communication
- F: Final
- FCS: Frame Check Sequence
- FE: Fixed Equipment
- HDLC: High-level Data Link Control
- IEEE: Institute of Electrical and Electronics Engineers
- ITS :Intelligent Transportation Systems
- LID: Link Identifier
- LLC: Logical Link Control
- LPDU: Link Layer Protocol Data Unit
- LSB: Least Significant Bit
- L1: Layer1 of DSRC Physical layer
- L2: Layer2 of DSRC Data link layer
- L7: Application Layer Core of DSRC
- M: Modifier function bit
- MAC: Medium Access Control
- ME: Mobile Equipment
- MSB: Most significant Bit
- NACK: Not Acknowledge
- OBE: On Board Equipment (alternative descriptor to mobile equipment)
- OBU: On Board Unit (alternative descriptor to mobile equipment)
- OSI: Open System Interconnection

- P: Poll
- PDU: Protocol Data Unit
- P/F: Poll/ Final
- R: Response
- RR: Response Request
- RSE: Road Side Equipment (alternative descriptor to fixed equipment)
- RSU: Road Side Unit (alternative descriptor to fixed equipment)
- SAP: Service Access Point
- UI: unnumbered Information

-
- V(RI): receive state variable (LLC)
 - V(SI): transmit state variable (LLC)

APPENDIX C – Standard IEEE 802.11p

IEEE 802.11p is a draft amendment to the IEEE 802.11 standard to add wireless access in vehicular environments (WAVE). It defines enhancements to 802.11 required to support Intelligent Transportation Systems (ITS) applications. This includes data exchange between high-speed vehicles and between the vehicles and the roadside infrastructure in the licensed ITS band of 5.9 GHz (5.85-5.925 GHz). IEEE 1609 is a higher layer standard on which IEEE 802.11p is based.

802.11p will be used as the groundwork for Dedicated Short Range Communications (DSRC), a U.S. Department of Transportation project based on the ISO Communications, Air-interface, Long and Medium range (CALM) architecture standard looking at vehicle-based communication networks, particularly for applications such as toll collection, vehicle safety services, and commerce transactions via cars. The ultimate vision is a nationwide network that enables communications between vehicles and roadside access points or other vehicles. This work builds on its predecessor ASTM E2213-03.

The 802.11p Task Group is still active. Per the official IEEE 802.11 Work Plan predictions the approved 802.11p amendment is scheduled to be published in November 2010.

Most recently the European Commission has allocated 5.9 GHz band for priority road safety applications and inter-vehicle, infrastructure communications. The intention is that compatibility with the USA will be ensured even if the allocation is not exactly the same; frequencies will be sufficiently close to enable the use of the same antenna and radio transmitter/receiver.

APPENDIX D – Microcontrollers Features

Atmega 128

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
 - 133 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
 - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
 - 128K Bytes of In-System Reprogrammable Flash
 - Endurance: 10,000 Write/Erase Cycles
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program
 - True Read-While-Write Operation
 - 4K Bytes EEPROM
 - Endurance: 100,000 Write/Erase Cycles
 - 4K Bytes Internal SRAM
 - Up to 64K Bytes Optional External Memory Space
 - Programming Lock for Software Security
 - SPI Interface for In-System Programming
- JTAG (IEEE std. 1149.1 Compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
 - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Two 8-bit PWM Channels
 - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
 - Output Compare Modulator
 - 8-channel, 10-bit ADC
 - 8 Single-ended Channels
 - 7 Differential Channels
 - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
 - Byte-oriented Two-wire Serial Interface
 - Dual Programmable Serial USARTs
 - Master/Slave SPI Serial Interface
 - Programmable Watchdog Timer with On-chip Oscillator

- On-chip Analog Comparator
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated RC Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
 - Software Selectable Clock Frequency
 - ATmega103 Compatibility Mode Selected by a Fuse
 - Global Pull-up Disable
- I/O and Packages
 - 53 Programmable I/O Lines
 - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
 - 2.7 - 5.5V for ATmega128L
 - 4.5 - 5.5V for ATmega128
- Speed Grades
 - 0 - 8 MHz for ATmega128L
 - 0 - 16 MHz for ATmega128

TX27 i.MX27 module

The TX27 is the first member of the new Freescale i.MX processor series from Strategic Test. TX modules are a complete computer, implemented on a board smaller than a credit card and ready to be designed into your embedded system. TX modules together with the Development Kit-5 give you the tools to achieve the fastest time-to-market for your newproduct designs. Because these products are supplied by Strategic Test, you can trust that the hardware and software is stable and backed up by truly responsive and competent technical support. All hardware and software development and manufacturing is performed in house - so that we can control the quality from start to end.

The module of the TX27 is specifically targeted at embedded applications where size, high cpu-performance and the lowest power consumption are critical factors. The TX27 is based on Freescale's new i.MX27 multi-media processor and includes mobile SDRAM and Flash memory. The integrated LCD controller allows direct connection to the screen, while the internal Ethernet controller only needs the addition of magnetics and a connector. The standard PCMCIA interface permits simple extension and integration into a target system.

Key features:

- 400 MHz CPU
- 64 to 128 MB mobile SDRAM mobile DDR-SDRAM
- 128 MB NAND Flash
- Ethernet 10/100 MAC, only need to add magnetics/connector
- USB 2.0 host / OTG
- LCD controller to 800 x 600
- MPEG-4 H.263/H.264 hardware CODEC for D1 video
- Camera interface
- OS support: Linux 2.6, WinCE 5.0 (6.0 in development)
- Operating temperature range -20°C..85°C
- Small size: 67.6 x 26 x 4.2 mm
- RoHS compliant (lead free)

The TX27 accepts an input voltage from various sources:

- 1-cell Li-Ion/Polymer (3.0V to 4.2V)
- 5.0V USB supply
- AC wall adapter
- 3.3V

The on-module regulator of the TX27 delivers up to 1A current 1.8V and 3.3V power rails that can be used to power the carrier board.

Expansion

The TX27 module is supplied with a DIMM200 connector to provide the designer with access to all of the i.MX27 signals for nearly unlimited expansion.

Ultra-Low Power Consumption

Perfect for wireless and multimedia battery-powered applications, the TX27 module is equipped with ultra-lowpower components including mobile 1.8V SDRAM's and 1.8V NAND Flash.

The i.MX27 processor

The i-MX27 is derived from the popular i.MX21 processor and based on the ARM926EJ-S core. However, the i.MX27 has a h.264 DI hardware codec for high-resolution video processing, an 110/10 Ethernet MAC, security, plug-and-play connectivity and additional power management features.

CPU

- ARM926EJ-S 400 Mhz core
- 16 KB L1 I-Cache and D-Cache
- ETM real-time debug
- Smart Speed Switch

Connectivity

- Ethernet 100/10
- USB 2.0 high-speed OTG
- USB 2.0 high-speed host

Multimedia

- LCD controller up to 800 x 600, 18 bit per pixel
- MPEG-4 H.263/H.264 hardware codec for D1 high resolution video processing
- Camera interface

OS Support

- Linux 2.6
- Windows CE 5.0 with WinCE 6.0 in development
- RedBoot bootloader