# Transceiver design for contactless desktop reader supporting MODE1 and MODE2 of ISO/IEC 18000-3

Diploma Thesis

at

Graz University of Technology


submitted by

**Selma Školjić**

Institute for Broadband Communication
Graz University of Technology
A 8010 Graz, Austria

in cooperation with


Ao.Univ.-Prof. Dipl.-Ing. Dr. Erich Leitgeb
Dipl.-Ing. Peter Pirc (Infineon Graz)

Graz, June 2010

# Abstract

The goal of this project was to create a contactless desktop reader for ISO/IEC 18000-3 supporting MODE1 and MODE2. The reader consists of an analog front-end and a microcontroller. The microcontroller programming was the main aim of this diploma thesis.

To achieve this, initial theoretical research was conducted, emphasizing the basics of an RFID system and the magnetic field, as well as the two supporting MODEs of the ISO/IEC 18000-3. The ARM based microcontrollers are also investigated. Based on this research, different algorithms were developed and tested. Ultimately, tests were run on the analog front-end and microcontroller functioning together and results were presented.

The first part of this document is a short introduction to RFID systems in general. The parts of RFID systems are explained and the RFID systems classification is given. Also, there is a short introduction to coding and modulation techniques. The next chapter gives us an overview of the ISO/IEC 18000-3. Since MODE1 is compatible with the ISO/IEC 15693, this chapter will cover this standard, also. In next chapter, the coding techniques for both modes are explained. Decoding techniques are described next. Finally the conclusions are presented.

**Keywords:** RFID, ISO/IEC 18000-3, Vicinity, Phase Jitter Modulation, Reader

# Kurzfassung

Ziel dieses Projekts war es, einen kontaktlosen Desktop Leser für den Standard ISO/IEC 18000-3 zu entwickeln, welcher MODE1 and MODE2 unterstützen würde. Der Leser setzt sich aus einer analogen Eingangstufe und einem Microcontroller zusammen, wobei letzterer das Hauptthema dieser Diplomarbeit ist.

Um das genannte Vorhaben zu realisieren, musste zuerst der theoretische Aspekt untersucht werden. Dabei wurde beim theoretischen Zugang von den Grundlagen des RFID Systems und des Magnetfeldes bis zu den zwei unterstützten MODEs des ISO/IEC 18000-3 Standards, alles abgedect. Die ARM Microcontrollers sind, auch, untersucht geworden. Anhand dieser Untersuchungen wurden verschiedene Algorithmen entwickelt und getestet. Als Abschluss wurden Tests in Bezug auf die gemeinsame Funktionsweise der analogen Eingangstufe und des Microcontrollers durchgeführt und die Ergebnisse präsentiert.

Der erste Teil des Dokuments beinhaltet eine kurze allgemeine Einführung zu RFID Systemen, wobei einzelne Bestandteile der Systeme und die Klassifikation der Systeme angeführt werden. Des weiteren wird eine Einführung zu den Kodier- und Modulationstechniken gegeben. Das nächste Kapitel bietet einen Überblick über den Standard ISO/IEC 18000-3. Da MODE1 mit dem Standard ISO/IEC 15693 kompatibel ist, wird dieser Standard ebenfalls behandelt werden. Im darauf folgenden Kapitel werden die Kodiertechniken beider Modi erläutert. Es folgen die Dekodiertechniken und abschließend die Präsentation der Schlussbetrachtungen/Ergebnisse.

**Schlüsselwörter:** RFID, ISO/IEC 18000-3, Vicinity, Phase Jitter Modulation, Reader

# Statutory declaration

*I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.*

Graz, ..... Signature .....

# Eidesstattliche Erklärung

*Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.*

Graz, am ..... Unterschrift .....

# Acknowledgements

I would like to express my gratitude to Infineon Technologies Austria AG, for providing me the opportunity to write my diploma thesis in such a highly professional environment and in such fantastic working atmosphere. Among all my colleagues I am in particular indebted to Peter Pirc, my project manager and Christian Pontesegger for his support on the digital design. On the academic side, I would like to thank Prof. Erich Leitgeb for being my helpful mentor.

Last but not least, I would like to thank my parents for their emotional and financial support throughout my studies.

Selma Školjić

Graz, Austria, 2010

# Contents

# List of Figures

# List of Tables

# Glossary

| | |
|---|---|
| AFI | Application Family Identifier |
| APB | Advanced Peripheral Buss |
| ARM | Advanced RISC Machine |
| ASK | Amplitude Shift Keying |
| BPSK | Binary Phase Shift Keying |
| CPHA | Clock Phase |
| CPOL | Clock Polarity |
| CRC | Cyclic Redundancy Check |
| DSFID | Data Storage Format Identifier |
| EOF | End Of Frame |
| FIFO | First In, First Out |
| FSK | Frequency Shift Keying |
| FTDMA | Frequency and Time Division Multiple Access |
| HF | High Frequency |
| HSE | High Speed External |
| HSI | High Speed Internal |
| IC | Integrated Circuit |

| | |
|---|---|
| IEC | International Electrotechnical Commission |
| ISO | International Organization for Standardization |
| JTC | Joint Technical Committee |
| LSB | Last Significant Bit |
| LSE | Low Speed External |
| LSI | Low Speed Internal |
| MFM | Modified Frequency Modulation |
| MISO | Master In Slave Out |
| MOSI | Master Out Slave In |
| MSB | Most Significant Bit |
| NRZ | Non-Return to Zero |
| PJM | Phase Jitter Modulation |
| PPM | Pulse Position Modulation |
| PSK | Phase Shift Keying |
| RFU | Reserved for Future Use |
| RZ | Return to Zero |
| SOF | Start Of Frame |
| SPI | Serial Peripheral Interface |
| UID | Unique Identifier |
| VCD | Vicinity Coupling Devices |
| VICC | Vicinity Integrated Circuit Card |

# 1. Introduction

*Radio Frequency Identification* (RFID) is a technology for identifying objects using a radio frequency signal. This technique is part of automatic identification procedures (auto-ID) and at first was intended as a replacement for bar codes. But the range of application of RFID systems increased enormously and it quickly became more then just a better version of bar codes. Today, RFID systems have become a part of our every day life. They are most commonly used as anti-theft devices in shops. Further fields of application lie in e.g. ticket services for public transport, car and automotive industry, animal tracing, pharmaceuticals and medical industry, e-passports, etc. This particular type of RFID systems are more related to a chip card than to bar codes. The data is stored on an electronic carrier while the energy support and data exchange is contactless.

A typical RFID system consists of a reader (also called interrogator) and one or more transponders (tags). The transponder can be attached to any object or animal that needs to be identified. The reader emits an electromagnetic field which is used for data transmission. If the tag is *passive*, then the electromagnetic field of a reader is also used for energy supply.

Since the applications for RFID systems are getting more and more complex, the demands on tags are also growing. First of all, the tags memory capacity is growing. Automatically, the data transfer time becomes longer, making data transfer speed another very important factor in RFID systems development. The next important step in RFID systems development is reading and writing data to multiple tags within field. This diploma thesis particularly focuses on RFID systems defined in ISO/IEC 18000-3, which are generally based on a multiple tags in field communication.

The ISO/IEC 18000-3 MODE1 is compatible with ISO/IEC 15693. The tags based on this standard are also called *vicinity* tags. They are used for asset tracking, inventory tracking, document tracking, parking management and supply chain integration.

The ISO/IEC 18000-3 MODE2 covers the phase jitter modulation (PJM) technology which is 10 times faster than other HF technologies. PJM tags are designed for applications where many

tags are stacked together such as documents management, gamings, medicine, pharmaceuticals and jewelry. In documents management, for instance, it is possible to detect all documents with tags even if they are stacked together without separation, e.g. in shelves.

The aim of this diploma thesis was to create a low cost reader that would support MODE1 and MODE2 of ISO/IEC 18000-3. The analog front end of a reader is coverd by Eva-Maria Hanzl as part of her diploma thesis "Analog Front End for a Contactless Desktop Reader supporting ASK and PSK modulation techniques". This diploma thesis focuses on programming of a microcontroller embedded in a analog front-end. First, the microcontroller codes the reader command and sends it to the analog front-end, which modulates it and emits a modulated signal via antenna. After tag response on the command, the load modulated signal is demodulated by the analog demodulator. Then the microcontroller has to demodulate the subcarrier and decode the data.

The first part of this document is a short introduction to RFID systems in general (2). The parts of RFID systems reader and transponder are explained. The RFID systems classification is given based on operating frequency and range. Also, there is a short introduction to coding and modulation techniques which are most commonly used in RFID systems. The next chapter will provide an overview of the ISO/IEC 18000-3 (3). Therein, the communication signal interface between reader and transponder and other way around will be explained. In addition, the command and reply format are introduced. Since MODE1 is compatible with the ISO/IEC 15693, this chapter will cover this standard as well. In Chapter 3, the coding techniques for both modes are explained and, also, the microcontroller configuration. Decoding techniques are described in Chapter 4. Finally the conclusions are presented in Chapter 6.

# 2. Overview of RFID technology

The following chapter presents a short overview of the RFID technology. In order to properly understand the way a transceiver works, basic knowledge of RFID technology is essential. The following sections are according to [2].

## 2.1 Typical RFID components

Every RFID system consists of a minimum of two components: a *transceiver* and a *transponder*. The transponder or tag is attached to an object and can be seen as a data-carrying device. A transceiver or reader is able to read data from tag. Depending on its field of application, a reader can also write data onto a transponder.

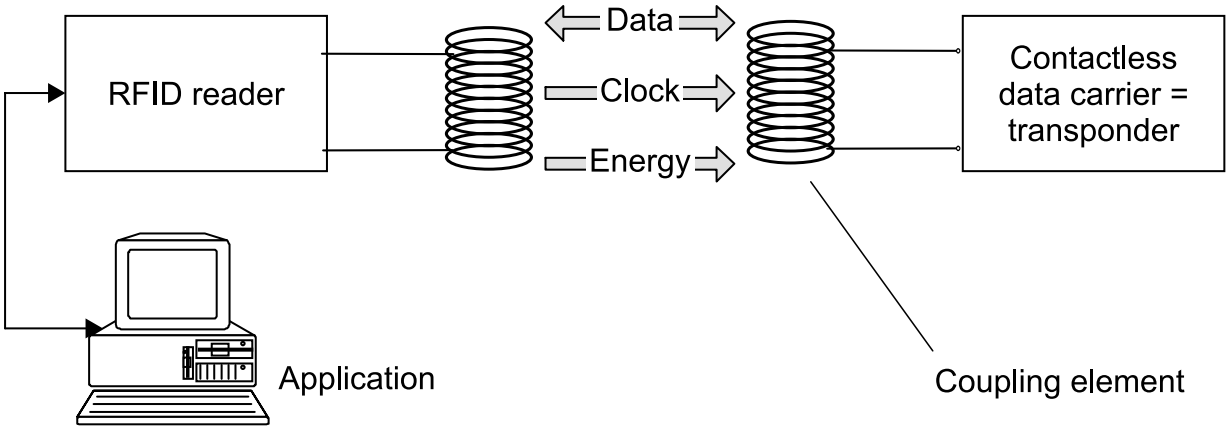


**Figure 2.1:** Basic RFID system (adapted from [2])

### 2.1.1 Transponder

There are many types and forms of transponders, but they can altogether be divided into three major groups:

**Low-end systems** have a memory size of a few bytes. With these systems, memory can only be read. The most prominent example for this system are the 1-Bit-Transponders, which are only used to detect whether an object is or is not within reader field. These systems have no anticollision algorithm.

**Mid-range systems** are more complex and have a larger memory size (up to 100 kbytes). Memory can be read and written. They have anticollision algorithms, which makes it possible to place more than one transponder in the reader field.

**High-end systems** have the largest memory size and are equipped with a microprocessor. This enables them to process more complex commands.

Depending on their power supply, mid-range and high-end systems can be classified into *active* and *passive* transponders. Active transponders have an internal power supply consisting of batteries or solar cells. Passive transponders, on the other hand, obtain their energy from the reader field. Both types have their advantages and disadvantages. An active transponder can be read at large distances, but it can not function without a battery, so the lifetime is limited. Such transponders are usually larger and more expensive. Passive transponders are significantly less expensive, much smaller and last longer. One of the major disadvantages here is, however, that passive transponders can only be read from short distance.

All of the above mentioned types of transponders come in different sizes and shapes, depending on their type of application. Some of the usual transponder shapes include: disks, cylinders, cards, keyfobs, smart labels etc.

### 2.1.2 Reader

The reader controls the communication process of RFID systems. Elements of a reader such as this include: a transmitter and receiver, a control unit and a coupling device. The reader generates an electric, magnetic or electromagnetic field, which supplies transponders with power and transmits data and commands. Besides the basic functions like power supply and data control, the reader should be able to preforming more complex tasks such as: anticollision, authentication and data encryption.

## 2.2 Classification of RFID systems

There are many ways to classify RFID systems based on their characteristics. The following subsections shall discuss only those who are most important and most relevant for this diploma thesis.

### 2.2.1 Operating frequency

As previously mentioned, the reader generates an electromagnetic field. The frequency of this field is called *operating frequency* of the RFID system. All RFID systems operate in following frequency bands:

- Low Frequency [LF] range from 30 kHz to 300 kHz
- High Frequency [HF] range from 3 MHz to 30 MHz
- Ultra High Frequency [UHF] range from 300 MHz to 3 GHz

### 2.2.2 Range

The area size that is covered by a reader where transponders can be identified is called range. There are three categories of RFID systems based on their range: close coupling systems, remote coupling systems and long range coupling systems.

Some of the main characteristics of *close coupling* systems are:

- range is less then one centimeter
- coupling is based on electric and magnetic energy
- frequency values lie between DC and 30 MHz
- As example door locking systems

*Remote coupling* systems have following characteristics:

- range is up to one meter
- coupling is mostly based on magnetic energy

- frequency values lie between 135 kHz and 13.56 MHz
- As example contactless smart cards

And finally the *long range coupling* systems can be recognized by the following characteristics:

- range can reach to a meter and beyond
- coupling is based on electromagnetic waves. They operate like conventional radio systems
- frequency values lie between 890 MHz and 930 MHz or 2.45 GHz
- Used for logistics

## 2.3 Physical fundamentals of RFID

As already mentioned in the previous section, inductive coupling is a common type of remote coupling. Some of the most common types in recent years have been various kinds of inductively coupled tags, including tags that follow the ISO/IEC 15693 standard for vicinity-coupled smart cards [5]. As the ISO/IEC 18000-3 tags also are inductively coupled tags, the next subsection will explain the basics of a magnetic field.

### 2.3.1 The magnetic field

The magnetic field is produced by all moving charges. It can be described by a *magnetic field strength* H vector, which depends on the amplitude of the current. Conductors are mostly loop-shaped. If there is more than one loop stacked together, we have a short cylindrical coil and the H is equal to:

$$H = N\frac{I}{2\pi r} \tag{2.1}$$

N represents the number of loops while r stands for radius. The maximum strength is concentrated in the center of the coil. Increasing the current or the number of the loops would also increase the strength of the magnetic field.

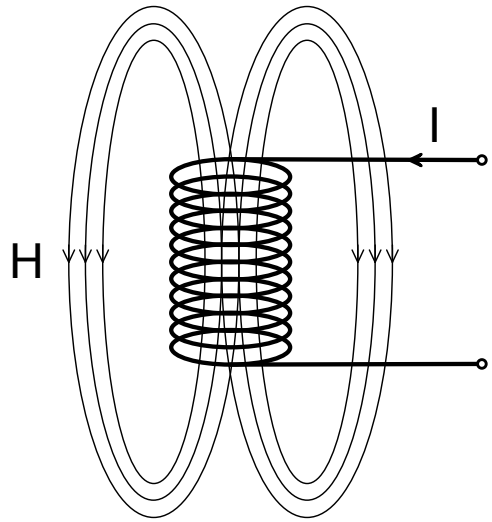


**Figure 2.2:** Magnetic field of a current-carrying cylindrical coil (adapted from [2])

The *Magnetic flux* $\Phi$ and the *magnetic flux density* B, also represent the physical measures for describing the magnetic field

$$\Phi = \int_A \mathbf{B} \cdot d\mathbf{A} \tag{2.2}$$

Formula 2.2 shows that $\Phi$ is calculated by determining the total number of streamlines of a magnetic field that pass through a specified area A. B denotes the number of flux lines by a certain area. If B is constant, the formula can be simplified as follows:

$$\Phi = |\mathbf{B}| \cdot A = B \cdot A. \tag{2.3}$$

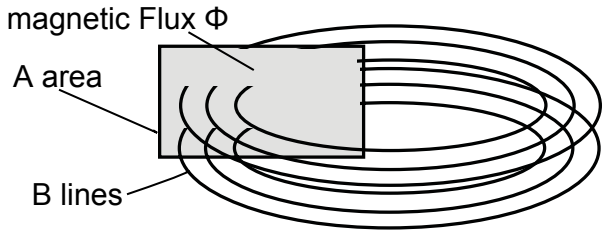


**Figure 2.3:** Relation between magnetic flux $\Phi$ and magnetic flux density B (adapted from [2])

$\Phi$ and therefore B depend on the material present in the magnetic field. The property of the material relevant for the magnetic field is called magnetic conductivity $\mu$. According to that, $\Phi$

and B can be calculated as follows:

$$\Phi = B \cdot A = \mu_0 \mu_r \mathbf{H} A = \mu \mathbf{H} A \tag{2.4}$$

$\mu_0$ ... magnetic field constant ($\mu_0 = 4\pi \cdot 10^{-7}\,{}^{\mathrm{Vs}}/_{\mathrm{Am}}$)
$\mu_r$ ... relative permeability of the material ($\mu_{r,air} = 1$)

The quotient of the magnetic flux generated and the current flowing are called *inductance* L. L is a measure for how much flux can be created by a certain current. There are two types of inductance called self-induction and mutual-induction. Self-induction happens when an electromotive force and a voltage are induced in the same circuit in which the current and flux are changing.

$$L = \frac{N\Phi}{I} = \frac{N\mu HA}{I} \tag{2.5}$$

If an electromotive force and a voltage are induced into one circuit as a result of current and flux changing in another circuit, than we are talking about mutual-induction (see Figure 2.4).

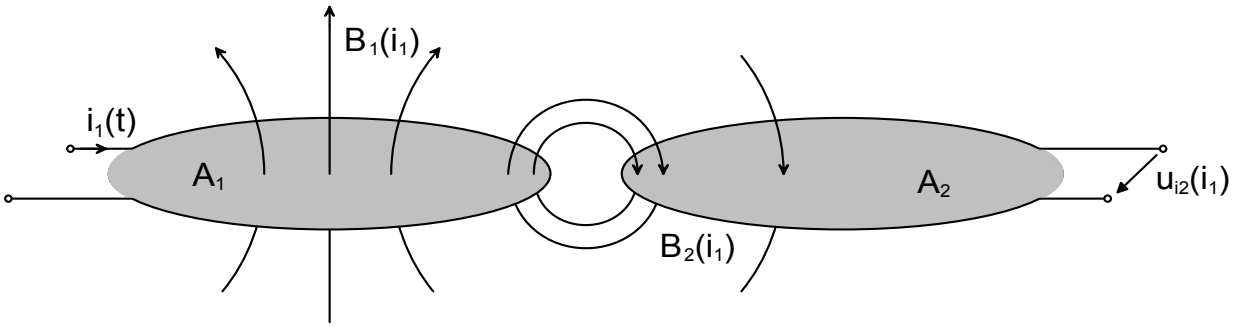


**Figure 2.4:** Mutual induction (adapted from [2])

Two conductor loops are placed next to each other. Current $I_1$ flows through loop 1 and consequently, a magnetic flux $\Phi_1(I_1)$ occurs. This loop is called *exciting*. A part of this flux penetrates the remote conductor loop 2, which is called *adjacent*, and this flux is defined as $\Phi_2(I_1)$. The relation between these two flux is called mutual-induction and is described in formula 2.6:

$$M_{21} = \frac{\Phi_2(I_1)}{I_1} \tag{2.6}$$

Actually, it is entirely irrelevant which loop is exciting and which one is adjacent, since $M_{21}$=$M_{12}$=M.

The coupling flux depends on the dimensions and geometry of both conductor loops and their position to each other. The *coupling factor k* is used to provide qualitative information about magnetic coupling, independently from geometric dimensions.

$$k = \frac{M}{\sqrt{L_1 \cdot L_2}} \tag{2.7}$$

The value of k can be between 0 (completely decoupled) and 1 (perfectly coupled).

### 2.3.2 The alternating magnetic field

While the previous subsection was a short introduction of a static magnetic field, this subsection will provide a short overview of an alternating magnetic field.

Change of magnetic flux $\Phi$ generates an electric field. Electromagnetic waves will be produced by an alternating electric, magnetic field, if vacuum is present as a medium. If there is an open conductor loop present, induction voltage $u_i$ (see formula 2.8) will be generated at the end of the loop.

$$u_i = -\frac{d\Phi(t)}{dt} \tag{2.8}$$

If there are two open conductors loops and one is generating an alternating magnetic field while the other is affected by it, energy can be transferred. This type of coupling is called transformer coupling and is used to supply passive transponders with power. Figure 2.5 depicts this principle. In exciting loop L1, current $I_1$ flows as result of voltage $u_1$. An alternating magnetic field is generated and it penetrates $L_2$. As a result, voltage $u_2$ is induced. This means that an alternating current in one loop can induce an alternating voltage in another loop within near filed. The right part of the figure shows a circuit for the transformer setup. The left part shows a real system, where resistors are present.

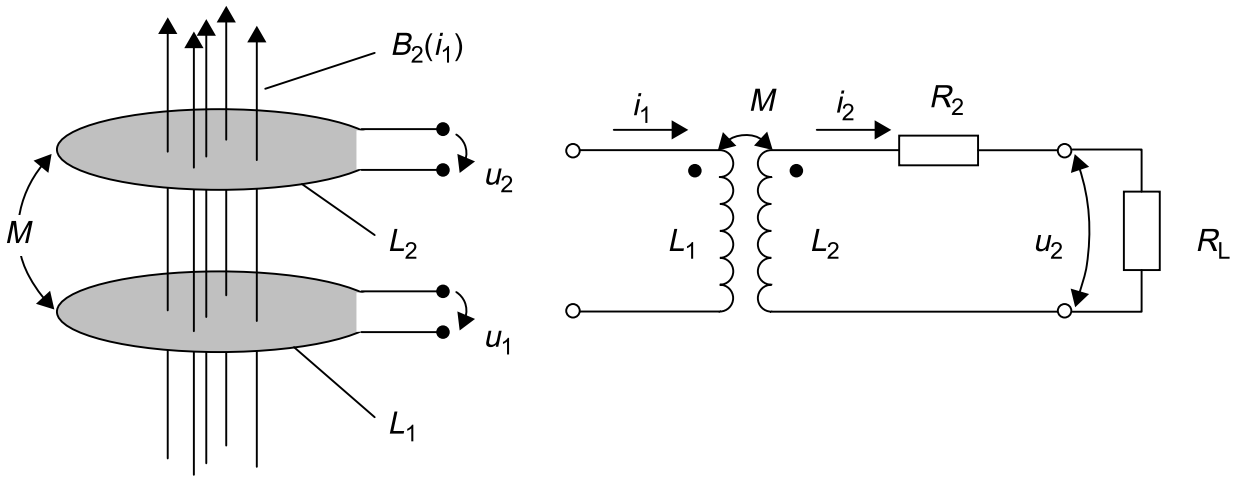


**Figure 2.5:** Transformer coupling (adapted from [2])

$R_2$ ... resistance of the loop windings
$R_L$ ... resistance applied to the loop

## 2.4 Coding and modulation

The main aim of this diploma thesis was to design coding and decoding techniques for the ISO/IEC 18000-3. Therefore, this section provides an in-depth analysis of coding techniques used in RFID and, also, a short overview on digital modulation.

Data transfer between reader and transponder can be divided into three major sections (see Figure 2.6):

- signal coding (signal processing) and the modulator (carrier circuit/analog front end) in the reader (transmitter)
- the transmission medium (channel)
- signal decoding (signal processing) and the demodulator (carrier circuit) in the transponder (receiver)

The modulator acts as an interface to the actual communication medium and often involves functions such as frequency conversion, amplification, and antenna-like transducers. The channel encoder is a discrete-input/discrete-output device whose usual purpose is seen in providing some error-correction capability for the system [21].

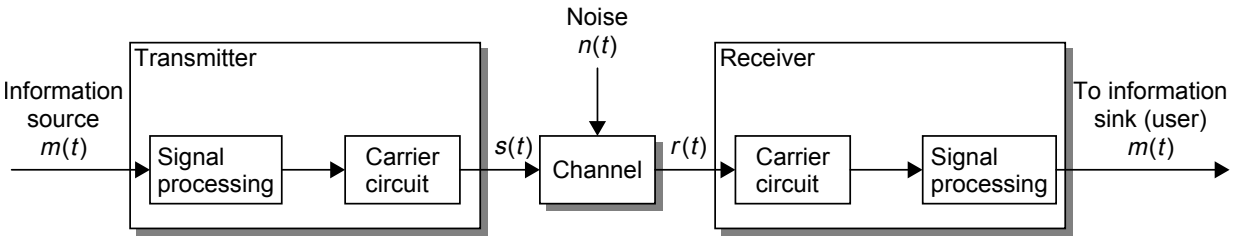


**Figure 2.6:** The communication system (adapted from [2])

### 2.4.1 Coding

The term signal coding actually stands for optimizing a signal (data baseband) according to the characteristics of the transmission channel (medium magnetic field). This procedure provides protection against interference and collision.

The following subsections will discuss some of the most commonly used line codes in RFID systems.

#### 2.4.1.1 Non-return to zero

Non-return to zero (NRZ) is the simplest line code. In this code, a binary $1$ is represented as a high signal and binary $0$ as a low signal. Usually, NRZ is used with PSK or FSK modulation. The biggest problem with this line code occurs when confronted with a large number of bits of the same value. It is very hard to detect boundaries of the received bits at the receiver. Figure 2.7 shows NRZ for binary string $01000010$.

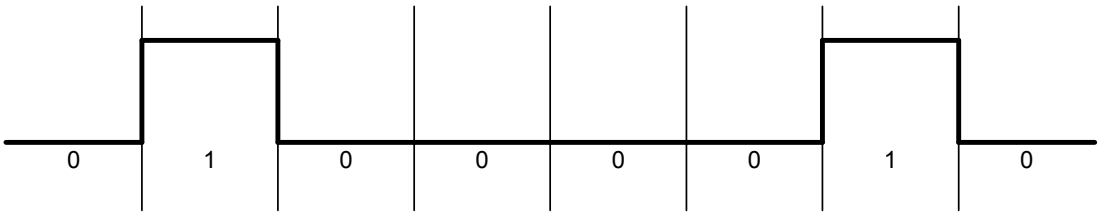


**Figure 2.7:** Non-return to zero for binary string 01000010

#### 2.4.1.2 Manchester code

The Manchester coding uses a negative transition in the middle of bit 1 and a positive transition in the middle of bit 0. The transitions of this type of coding are synchronized with the clock (see Figure 2.8). This way, sender and receiver are easy to synchronize, making the code applicable for a large series of bits carrying the same value.

The Manchester coding is used by ISO/IEC 18000-3 MODE1 for data transmission from vicinity-card to vicinity coupling device (see Section 3.1).

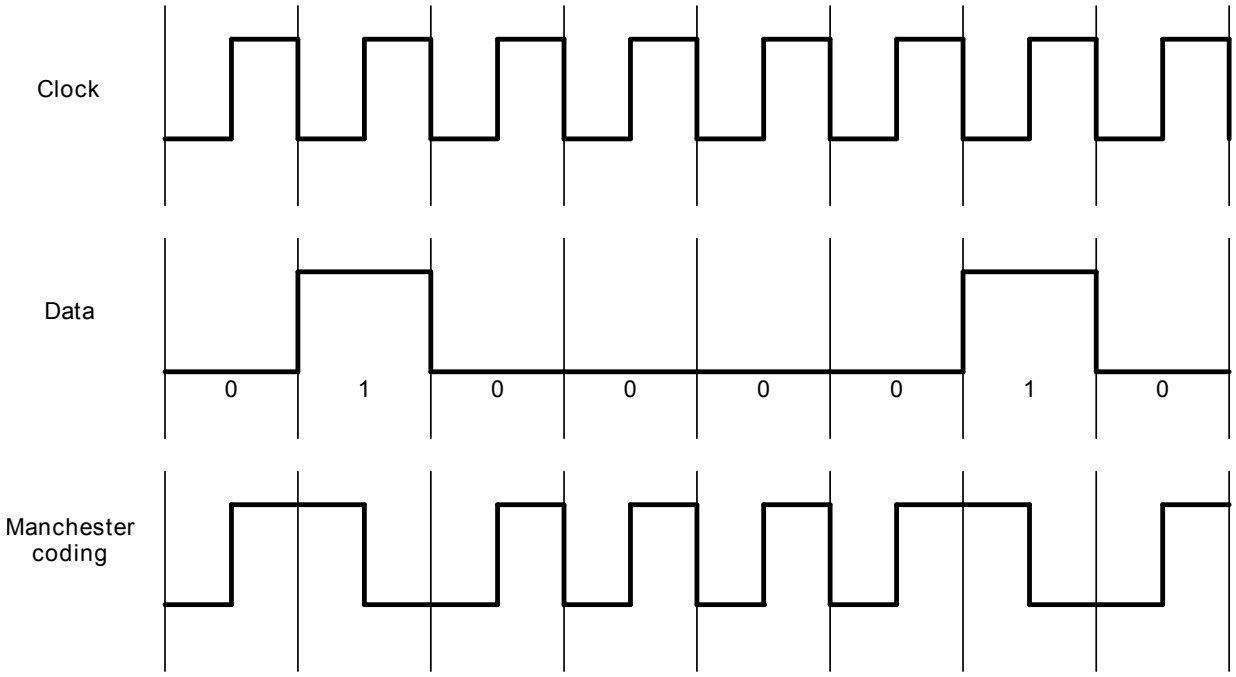


**Figure 2.8:** Manchester coding for binary string 01000010

#### 2.4.1.3 Unipolar RZ coding

RZ means return to zero. This means that at some point of a bit interval, the signal returns to a point that is either high or low. During the first half bit period, binary 1 is represented by a high signal. Binary 0 is the same as in NRZ, and therefore low. The problem of a long string of 1-es is solved, but this technique is still problematic when it comes to detecting boundaries of the received long string of 0-s. Figure 2.9 depicts unipolar RZ for binary string 11100010.

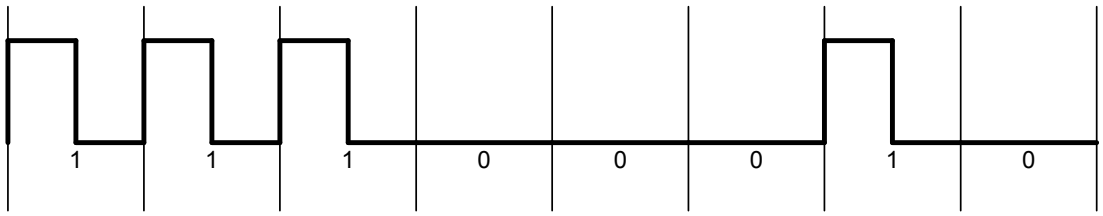


**Figure 2.9:** Unipolar RZ for binary string 11100010

#### 2.4.1.4 Miller coding

The Miller coding means that bit '1' is going to have a state change at the middle of a bit interval and a bit '0' at the beginning of a bit interval. If bit '0' follows bit '1', no state change will occur. This coding is also known as *modified frequency modulation*(MFM) and is used by ISO/IEC 18000-3 MODE2 for data transmission (see Section 3.2).

Besides the standard Miller code, there is, also, a modified Miller code where every transition is replaced by a negative pulse. This variant is often used in RFID. The pulse is very short, allowing a continuous power supply from reader to transponder during data transmission. Figure 2.10 shows an example for binary string 01000010 in both Miller coding (a) and modified Miller coding (b).

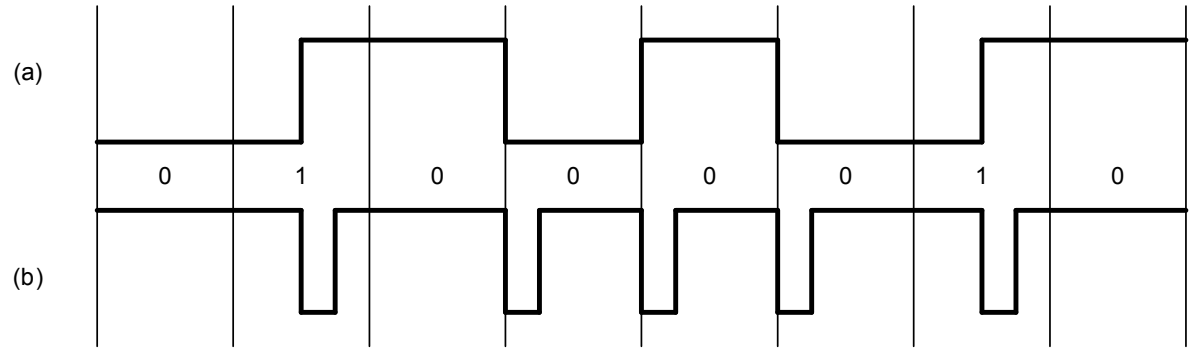


**Figure 2.10:** Miller coding (a) and Modified Miller coding (b) of binary 01000010

#### 2.4.1.5 Pulse-pause coding

Pulse-pause coding is a very simple coding type. Here, binary 1 represents a pause of duration *t* and binary 0 the pause of duration *2t*. Just like in modified Miller coding, the pulse durations here are very short. Therefore, it is possible to provide continuous power supply for the transponder. An example of pulse-pause coding is shown in figure 2.11.

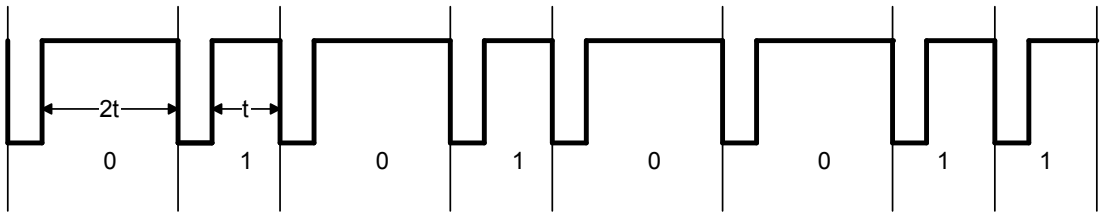


**Figure 2.11:** Pulse-pause of binary 01010011

### 2.4.2 Digital modulation

Digital modulation is the transformation of a digital signal into waveforms that are compatible with a transmission medium (channel). There are three types of modulation: amplitude shift keying (ASK), frequency shift keying (FSK) and phase shift keying (PSK).

#### 2.4.2.1 Amplitude shift keying

Amplitude shift keying is used in ISO/IEC 18000-3 MODE1. This particular type of keying sends digital data over analog carriers by changing the amplitude of a wave in time with the data stream. The amplitude of the carrier signal is switched according to the baseband signal. The analytic expression for ASK is:

$$s\left(t\right) = A\left(t\right) \cdot \cos\left(\omega_0 t + \varphi_0\right) \tag{2.9}$$

A(t) represents the baseband information and $\cos(\omega_0 t + \varphi_0)$ represents the carrier. A binary ASK can be characterized by the *modulation index m* (see formula 2.10) which is a measure for the ratio of the amplitude levels.

$$m = \frac{\hat{u}_{HI} - \hat{u}_{LO}}{\hat{u}_{HI} + \hat{u}_{LO}} \cdot 100\,\% \tag{2.10}$$

The modulation index ranges from 0, meaning no modulation, to 100%, meaning that one amplitude state is zero. MODE1 of ISO/IEC 18000-3 uses two modulation indexes, 10% and 100%.

ASK can actually be described as a multiplication of an unipolar digital baseband signal and a high-frequency, sinusoidal carrier signal as shown in figure 2.12 .

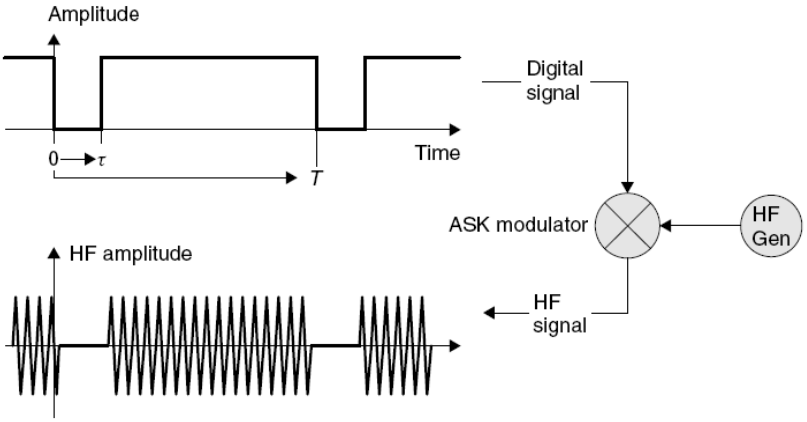


**Figure 2.12:** Generation of a 100% ASK modulation (adapted from [2])

#### 2.4.2.2 Frequency shift keying

Frequency shift keying sends changes through the frequency of the wave. The analytic expression for FSK modulation is:

$$s(t) = A \cdot \cos(\omega_0(t)\, t + \varphi_0) \tag{2.11}$$

$\omega_0(t)\, t$ represents the frequency that changes. The simplest FSK is a binary FSK, which means using a couple of discrete frequencies to transmit binary information. Figure 2.13 depicts the generation of a binary FSK.

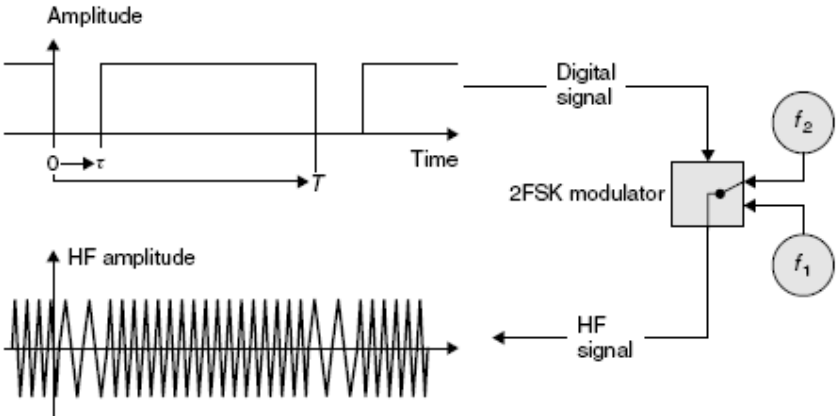


**Figure 2.13:** Generation of binary FSK modulation (adapted from [2])

#### 2.4.2.3 Phase shift keying

Phase shift keying means that the phase of a carrier signal is modulated. Mathematically, PSK can be described as follows:

$$s(t) = A \cdot \cos(\omega_0 t + \varphi_0(t)) \tag{2.12}$$

$\varphi_0(t)$ represents the phase that changes. The simplest PSK is a binary PSK (BPSK). The binary states '1' and '0' are converted into corresponding phase states, in this case $0\,^\circ$ and $180\,^\circ$. Mathematically, a binary PSK is the same as a multiplication of the carrier oscillation by $1$ and $-1$.The generation of this type of modulation is shown in figure 2.14.

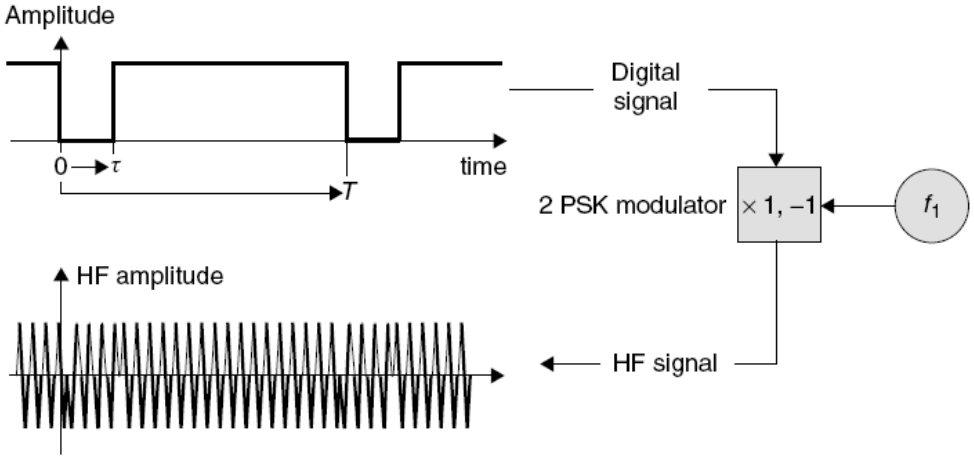


**Figure 2.14:** Generation of binary PSK modulation (adapted from [2])

Communication between the tag and the reader by ISO/IEC 18000-3 MODE2 takes place by using BPSK. Communication between reader and tag takes place by using *phase jitter modulation PJM*. PJM will be explained in further detail in chapter 3 (see section 3.2.1.1).

# 3. ISO/IEC 18000-3

The *International Organization for Standardization* (ISO) and the *International Electrotechnical Commission* (IEC) are the worlds largest publisher of international standards. The members of ISO or IEC are national bodies that participate in the development of international standards. ISO and IEC have technical committees which deal with particular fields of technical activity, and collaborate with each other due to mutual interest.

ISO/IEC JTC1 is a joint technical committee of ISO and IEC in the field of information technology. The ISO/IEC 18000 *Radio frequency identification for item management* has been developed by this committee. The main goal of this protocol is to use the same protocols for all frequencies, where possible. This way, the problems of migrating from one frequency to another would be minimized, just like the costs for software and implementation. This standard consists of 7 parts:

- Part 1: Reference architecture and definition of parameters to be standardized
- Part 2: Parameters for air interface communications below 135 kHz
- Part 3: Parameters for air interface communications at 13.56 MHz
- Part 4: Parameters for air interface communications at 2.4 GHz
- Part 5: Parameters for air interface communications at 5.8 GHz (withdrawn)
- Part 6: Parameters for air interface communications at 860 MHz to 960 MHz
- Part 7: Parameters for air interface communications at 433 MHz

This diploma thesis focuses on the ISO/IEC 18000-3. It has three MODES of operation. MODE1 is compliant with ISO/IEC 15693, which will be described in the next section 3.1. MODE2 is completely defined in section 3.2. MODE3 is a high speed interface with two options.

Option 1 is ASK based, option 2 is PJM based. Since MODE3 is not included in this diploma thesis it will not be described. Though all of the above modes are not interoperable, they are also non-interfering.

## 3.1 ISO/IEC 18000-3 MODE1

Since the ISO/IEC 18000-3 MODE1 is compatible with ISO/IEC 15693, the following subsections will cover this standard.

ISO/IEC 15693 forms a part of a series of International Standards that specify a contactless smart card. The standard consists of four parts. The first three parts are: *Part 1 - Physical characteristics*, *Part 2 - Air interface and initialization* and *Part 3 - Anticollision and transmission protocol* shall be described in the next sections.

*Part 4 - Extended command set and security features* will not be described since it is not an integral part of this diploma thesis.

### 3.1.1 ISO/IEC 15693 Part 1

This part describes the physical characteristics of a transponder. In this standard, the transponder is referred to as *Vicinity integrated circuit card* (VICC). A card of the card type ID-1 into which integrated circuit(s) and coupling means have been placed and in which communication to such integrated circuit(s) is done by inductive coupling in the vicinity of a coupling device. The reader that provides power and control for data transmission is referred to as *vicinity coupling device* (VCD).

### 3.1.2 ISO/IEC 15693 Part 2

The second part of ISO/IEC 15693 defines the power and data transfer between VICC and VCD. The VCD produces a magnetic field. The frequency of the field is $13.56\,\text{MHz} \pm 7\,\text{kHz}$ . The field strength should lie between 150 mA/m and 5 A/m. Whenever power is received by the card, it is able to respond to incoming commands from the VCD.

#### 3.1.2.1 Communications signal interface VCD to VICC

ASK is a modulation principle used in communication between the VCD and the VICC. The modulation index can be either 10% or 100% (see section 2.4.2.1). The VCD decides which one is to be used. Data coding will be implemented using pulse position mode. The VICC supports two data coding modes: *1 out of 256* and *1 out of 4*. Since this thesis is focusing on a low-cost transceiver, only the $1$ out of $4$ mode will be used.

**1 out of 256 coding mode**

In this coding technique the one byte value is represented by the position of one pause. The value of the byte is determined by the position of the pause. One pause has a length of 18.88 $\mu$s ($256/f_c$) which gives us a byte length of 4.833 ms. The next figure depicts this pulse position modulation technique for byte '20' ($(20)_h = (32)_h$) sent by the VCD to the VICC.

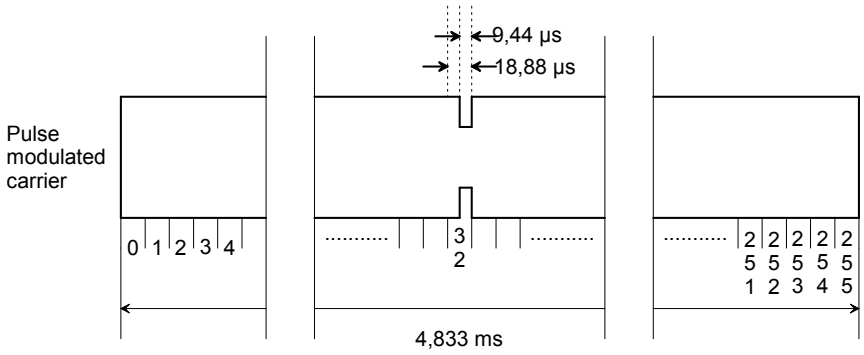


**Figure 3.1:** 1 out of 256 coding mode VCD to VICC (adapted from [7])

Data framing is used at the beginning and at the end of every command. These data frames represent code violations. They are used to make synchronization and independence of protocol easier. Figure 3.2 depicts the SOF for this coding technique.

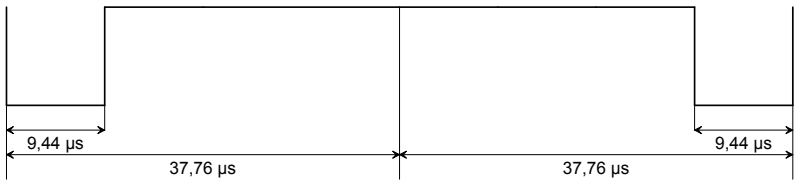


**Figure 3.2:** Start of frame of the 1 out of 256 coding mode VCD to VICC (adapted from [7])

The EOF is described in the next figure. It is the same for both coding modes.

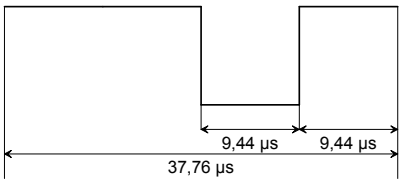


**Figure 3.3:** End of frame for either mode VCD to VICC (adapted from [7])

### 1 out of 4 coding mode

Figure 3.4 depicts the 1 out of 4 pulse position technique and coding.

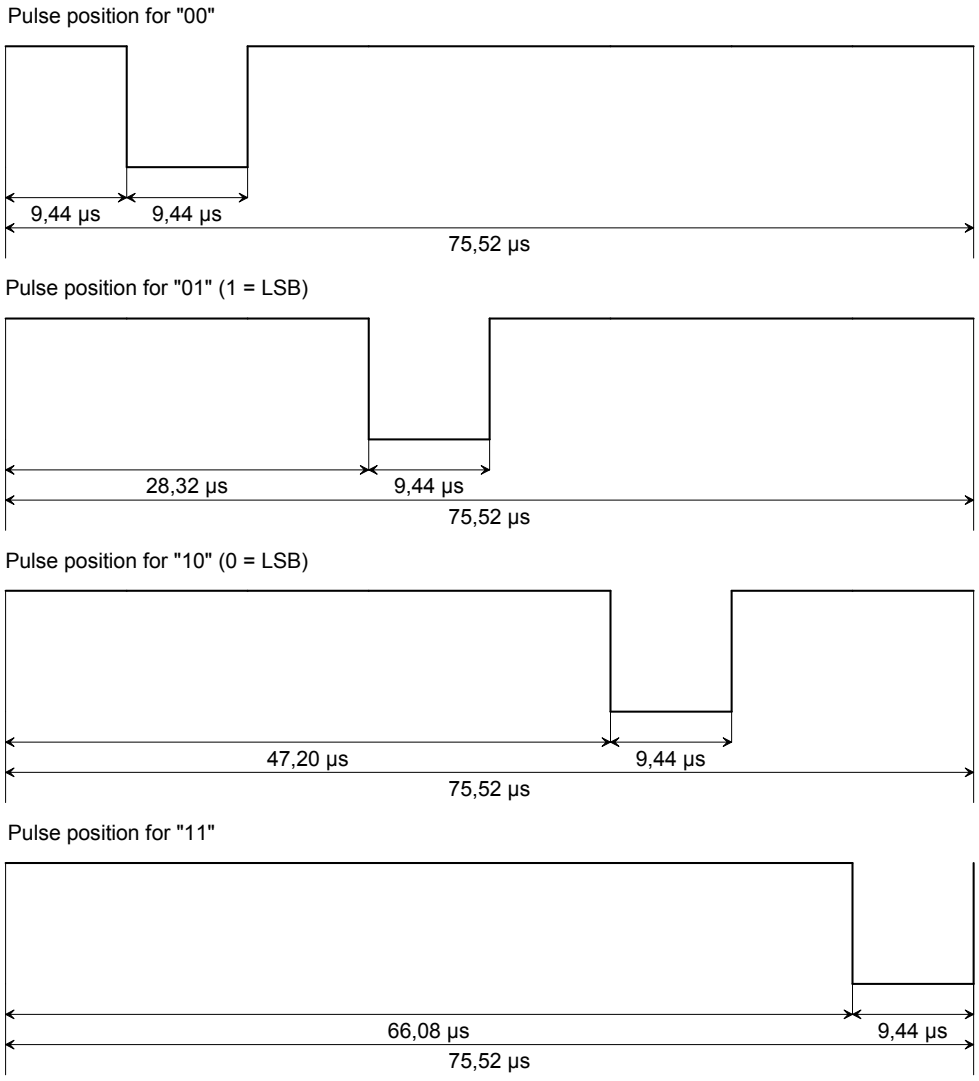


**Figure 3.4:** 1 out of 4 coding mode VCD to VICC (adapted from [7])

Two bits are coded as the position of a 9.44 $\mu$s pause in a 75.52 $\mu$s symbol time, providing a bit rate of 26.48 kbits/s. The least significant bits are being transmitted first. Figure 3.5 shows a transmission example of byte 'D2' ($(D2)_h = (11010010)_b$). First, the LSB pair '10' is being transmitted.

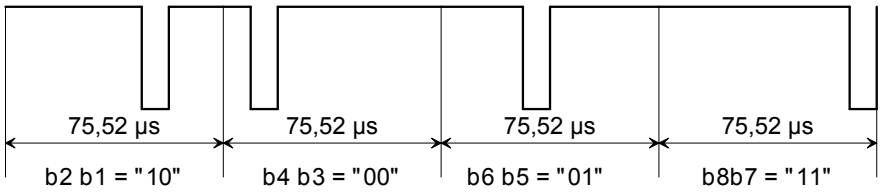


**Figure 3.5:** 1 out of 4 coding example

Figure 3.6 depicts the SOF of the 1 out of 4 data coding mode. As already mentioned, the EOF is the same as in the previous mode (see figure 3.3).

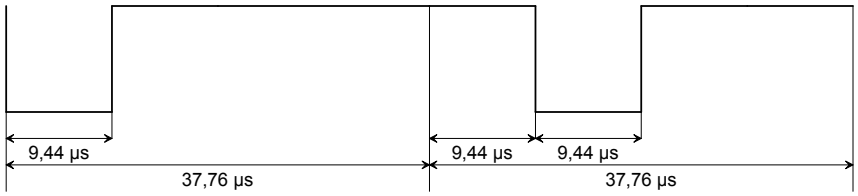


**Figure 3.6:** Start of frame of the 1 out of 4 mode VCD to VICC (adapted from [7])

#### 3.1.2.2 Communications signal interface VICC to VCD

One or two subcarriers may be used as selected by the VCD. The VICC supports both modes. In order to keep the transceiver simple, only one subcarrier will be used in this diploma thesis.

**Bit representation and coding when using one subcarrier**

The frequency of the subcarrier load modulation will be $f_c/32 = 423.75\,\mathrm{kHz}$. Data shall be encoded using Manchester coding. Logic 0 starts with 8 pulses of $f_c/32$ (423.75 kHz), followed by unmodulated time $256/f_c$ (18.88 $\mu$s). Logic 1 works the other way round, as to be seen in figures 3.7 and 3.8.

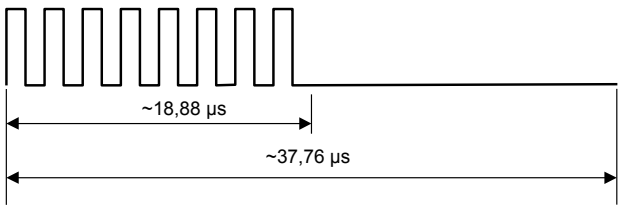


**Figure 3.7:** Logic 0 when using one subcarrier VICC to VCD [7]

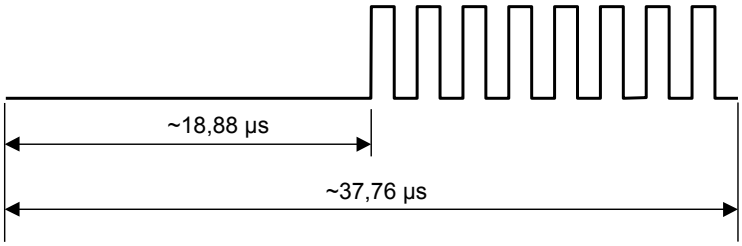


**Figure 3.8:** Logic 1 when using one subcarrier VICC to VCD [7]

Just like the commands, responses also include data framing. The data frames represent code violations. When using one subcarrier, a start of frame comprises of three parts (see figure 3.9). First, there is an unmodulated time of minimum $768/f_c$ (56.64 $\mu$s), followed by $24$ pulses of $f_c/32$ (423.75 kHz) and a logic $1$ at the end.


**Figure 3.9:** Start of frame when using one subcarrier VICC to VCD [7]

The end of a frame also comprises of three parts (see figure 3.10). At the beginning, there is a logic $0$, followed by $24$ pulses of $f_c/32$ (423.75 kHz) and after that an unmodulated time of minimum $768/f_c$ (56.64 $\mu$s).


**Figure 3.10:** End of frame when using one subcarrier VICC to VCD [7]

**Bit representation and coding when using two subcarriers**

Using two subcarriers logic $0$ starts, also, with $8$ pulses of $f_c/32$ (423.75 kHz) and is then followed by $9$ pulses of $f_c/28$ (484.28 kHz).

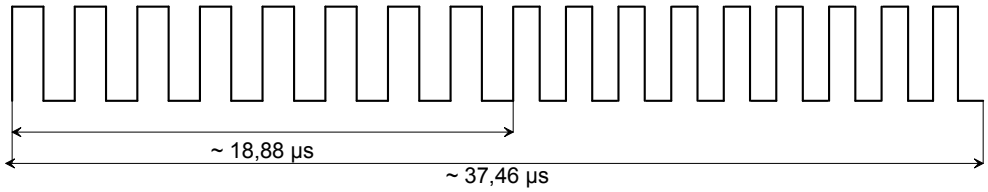


**Figure 3.11:** Logic $0$ when using two subcarriers VICC to VCD [7]

Logic $1$ starts with $9$ pulses of $f_c/28$ and ends with $8$ pulses of $f_c/32$.

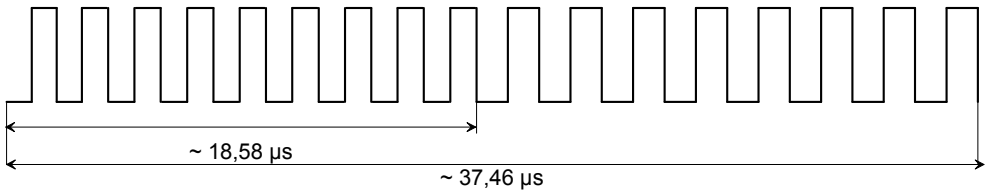


**Figure 3.12:** Logic $1$ when using two subcarriers VICC to VCD [7]

Similar as in one carrier, SOF and EOF also have tree parts. SOF starts with $27$ pulses of $f_c/32$ (423.75 kHz), followed by $24$ pulses of $f_c/32$ (423.75 kHz) and at the end, logic $1$ (see figure 3.13). Parts of EOF are: logic $0$, $24$ pulses of $f_c/32$ (423.75 kHz) and $27$ pulses of $f_c/32$ (423.75 kHz), as depicted in figure 3.14.

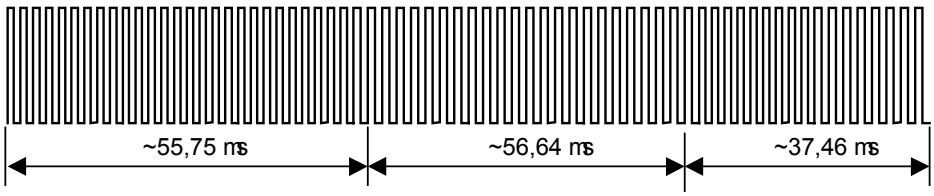


**Figure 3.13:** Start of frame when using two subcarriers VCD to VICC (adapted from [7])


**Figure 3.14:** End of frame when using two subcarriers VCD to VICC (adapted from [7])

### 3.1.3 ISO/IEC 15693 Part 3

This part describes the anticollision and transmission protocols. Furthermore, it discusses topics such as protocol, commands and other parameters required to initialize communication between a VICC and a VCD. Besides anticollision, the optional means for an easier and more efficient selection of one out of several cards based on their application criteria is described.

#### 3.1.3.1 VICC data elements

All VICCs have a *unique identifier* UID. The UID is used to address each VICC individually and is set by the IC manufacturer. The $8$ MSB bits are 'E0', followed by a 8 bits IC manufacturer code and a unique serial number of 48 bits. Another part of VICC data is the application family identifier (AFI). AFI is used to extract from all VICCs present in reader field only the VICCs that are needed for application chosen by VCD. AFI is programed and locked by VCD commands. Not all VICCs support AFI. If AFI is not supported by a VICC, it will ignore all requests that have AFI value.

The way data is structured in the VICC memory may, also, be programmed and locked by the VCD commands. This is called a data storage format identifier (DSFID). VICC memory is organized in blocks. Up to $256$ blocks can be addressed and block size can average up to $256$ bits.

#### 3.1.3.2 Request and response format

The transmission protocol defines data exchange between VCD and VICC and vice versa. It is based on the so called *reader talks first* concept which means that the VICC can only transmit after receiving a valid command from the VCD. Each VCD request and VICC response are contained in a frame (SOF and EOF). The VCD sends a request consisting of flags, command code, mandatory and optional parameter fields, application data fields and CRC. The VICC response consists of: flags, mandatory and optional parameter fields depending on command, application

data fields and CRC.

**Flags** specify which actions should be preformed by VICC (request flags) or have been preformed (response flags) and which corresponding fields are present. Using flags the VCD specifies: whether one or two subcarriers will be used, which data rate will be used and whether all or only the one VICC in field shall respond.

**The command code** is only present within a request. There are four types of commands. The *Mandatory commands* are supported by all VICCs. The command codes range from '01' to '0F'. Then there are *Optional commands* whit a code range from '20' to '9F'. The VICCs may support them, though not compulsory. The *Custom commands* are manufacturer specific functions and range from 'A0' to 'DF'. Last but not least, there are *Proprietary commands* that are used for tests and programming of system information. Their range lies between 'E0' and 'FF'.

**Mandatory and optional parameters fields** are parts of a request or a reply that depend on the command sent by the VCD. If for instance, the VCD sends an inventory command during the anticollision sequence, the command may have a *mask length* and *mask value* block. Mask value keeps the value of right received bytes and mask length has the length of the mask. Only those VICCs whose UID starts with these bytes are going to respond. AFI and DSFID fields are, also, optional parameters for requests.

**Application data** depends, also, on a command that is sent by the VCD. For example if the VCD sends a write command, it has to contain the number of blocks which will be written on and the data that needs to be written on the VICC.

**CRC** is a technique used for detecting errors in digital data transmission. A *Checksum*, meaning a particular number of check bits, is added to the message being transmitted. The receiver can determine whether or not the check bits agree with the data. Appendix A gives an example along with a description of the CRC detection method.

#### 3.1.3.3 VICC states

Depending on VCD commands, the VICC can be in four different states. Figure 3.15 shows these VICC states. The *Power-off* state is a mandatory state where the VICC can not be activated by the VCD. Another mandatory state is *Ready*.The VICC is in ready state when activated by a VCD and can process any request if the *Select-flag* is not set. *Quiet* state means that the VICC can process any request where the *Address-flag* is set and the *Inventory-flag* is not. *Select* state is the only optional state where the VICC shall process only requests with the *Select-flag* set.

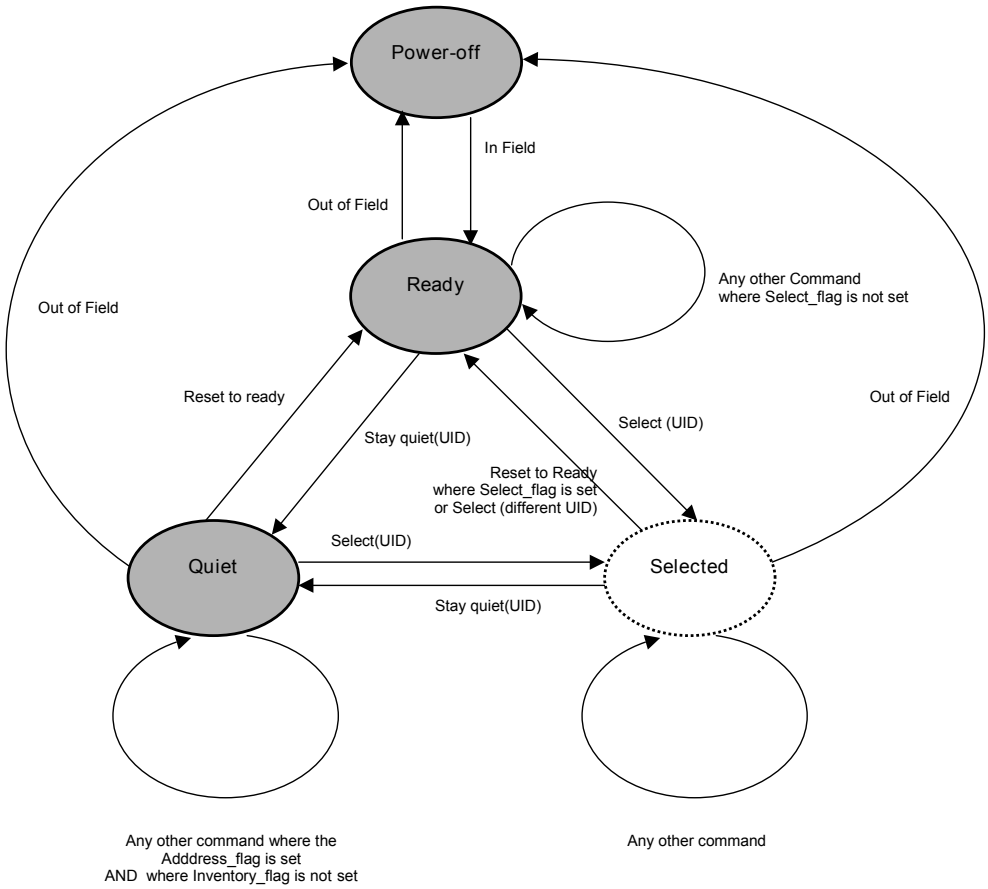


**Figure 3.15:** VICC state transition diagram [6]

#### 3.1.3.4 Anticollision

An anticollision sequence is needed whenever multiple VICCs are present within the VCD field. The VCD sends an inventory command in which it sets the number of slots. The number of slots can be either 16 or 1. The slots start immediately after EOF and VCD switch to another slot by sending an EOF. If two or more VICC sends their response during one slot, the VCD detects this and remembers collision. It then switches to another slot. If there is no VICC response in one slot, the VCD will switch to another slot. The VCD can either interrupt the anticollision sequence, or continue until the last slot, before sending request to the one VICC that was able to respond without collision.

In order to avoid simultaneous transmission of both VCD and VICC, the following timing specifications are defined:

1. *VICC waiting time before transmitting response after reception of an EOF from the VCD*

The VICC shall wait $t_1$ time after receiving EOF of the VCD command, before sending a response. If the VICC detects a carrier modulation during this time, it should reset its timer and wait for further time period $t_1$. The minimal value is 318.6 $\mu$s, the nominal value is 320.9 $\mu$s and the maximum is 323.3 $\mu$s.

2. *VICC modulation ignore time after reception of an EOF from the VCD*

   During time $t_{mit}$, VICCs shall ignore any received 10% modulation after detection of valid command EOF. The minimum value is 323.3 $\mu$s+ $t_{nrt}$. $t_{nrt}$ represents the nominal response time of a VICC.

3. *VCD waiting time before sending a subsequent request*

   When the VCD has received a VICC response, it shall wait $t_2$ time before sending a subsequent request. The minimum value here is 309.2 $\mu$s.

4. *VCD waiting time before switching to the next slot during an inventory process*

   If the VCD starts to receive one or more VICC responses, it shall wait until the reception of response is complete, then wait for $t_2$ and afterwards send EOF to switch to another slot. If the VCD receives no response, than it shall wait $t_3$ before switching to the next slot. When using 100% modulation the minimum value of $t_3$ is 323.3 $\mu$s + $t_{sof}$ (time for a VICC to transmit an SOF). If 10% is used, then $t_{3_{min}}$ is 323.3 $\mu$s + $t_{nrt}$ (nominal response time of a VICC).

The next figure depicts a possible anticollision sequence.

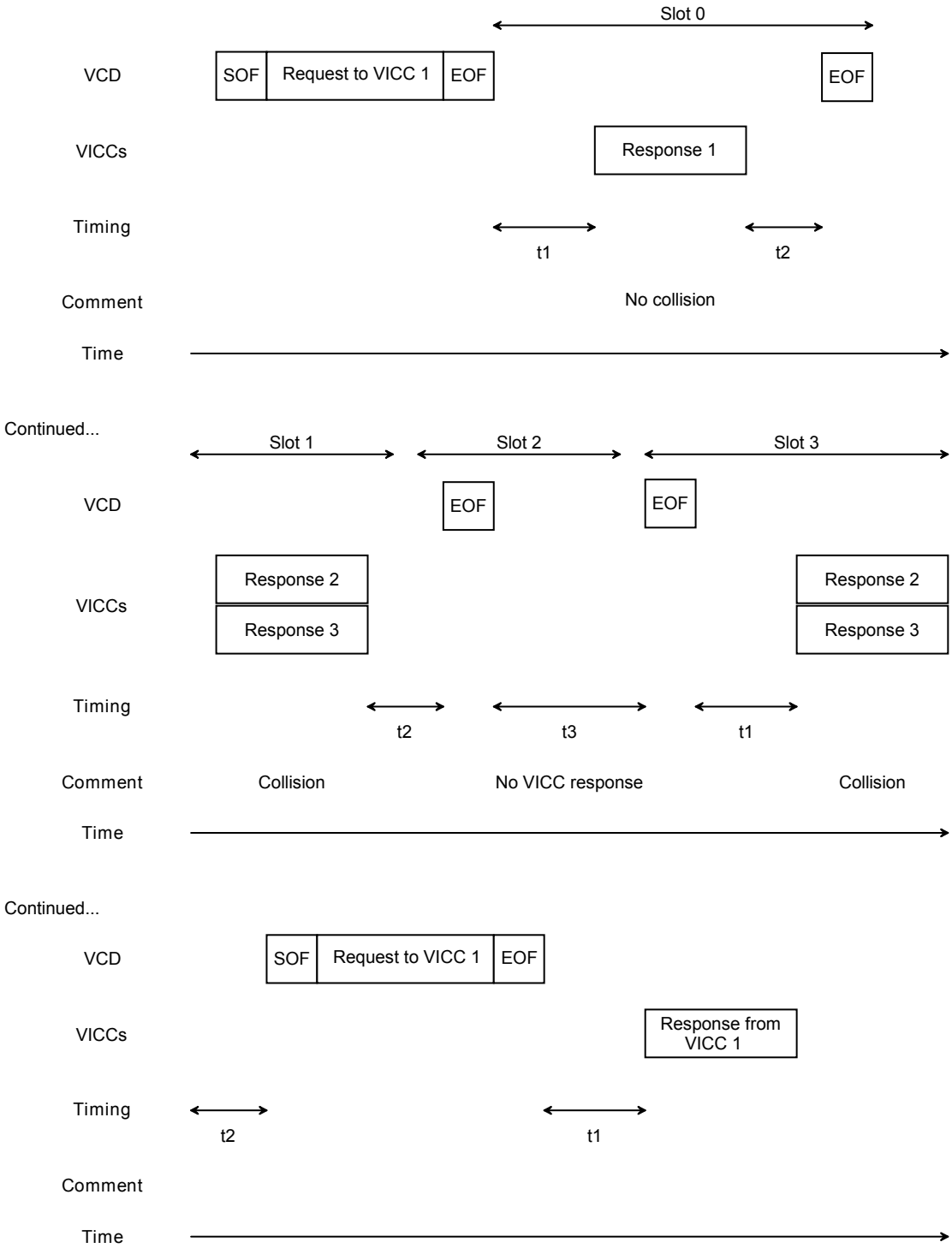


**Figure 3.16:** Description of a possible anticollision sequence in MODE1 [6]

## 3.2 ISO/IEC 18000-3 MODE2

MODE2 is not interoperable with any other MODE described in this standard. It is also not interfering with any other MODE. This system is also based on the *reader talk first* technique. The frequency of the field is 13.56 MHz $\pm$ 7 kHz.

### 3.2.1 Communications signal interface interrogator to tag

Interrogator commands are transmitted by Phase Jitter Modulation (PJM). PJM is a new modulation method and ISO/IEC 18000-3 is the only RFID systems standard that does not use amplitude modulation for communication.

#### 3.2.1.1 Phase jitter modulation

PJM can be described as variation of phase shift keying (PSK) (see section 2.4.2.3). Data transmission is done by using small phase shifts (between $\pm$ 1.0 $^\circ$ and $\pm$ 2.0 $^\circ$) of the interrogator field. The PJM signal can be considered as the sum of an unmodulated carrier signal and a data modulated quadrature carrier signal, which is attenuated and filtered [20]. Figure 3.17 depicts a PJM waveform.

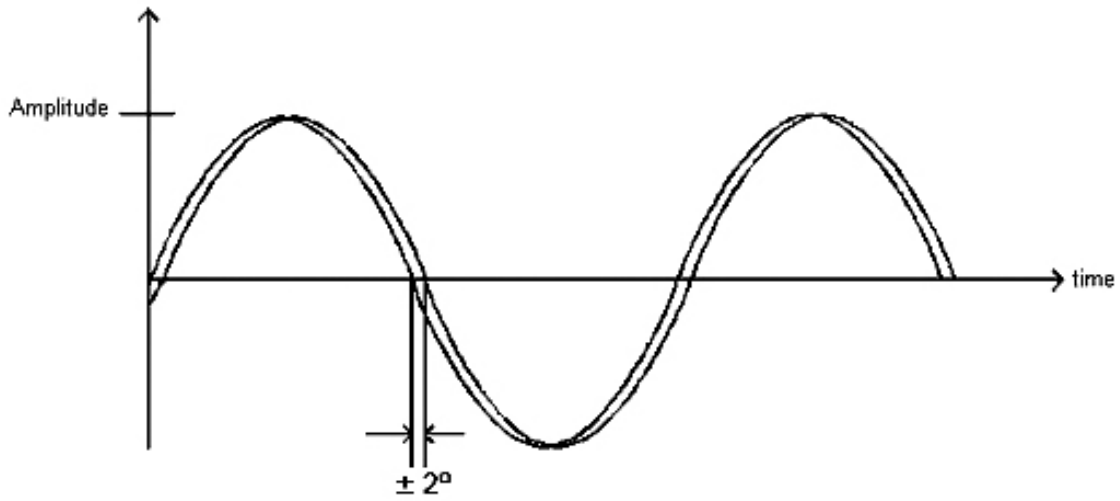


**Figure 3.17:** PJM waveform[8]

PJM has many advantages over the previously applied methods. Some of the most important

are:

- There is no reduction in power transfer to tag.
- Sideband levels are independent of data rate.
- The bandwidth of the PJM signal does not exceed the original modulated data bandwidth.

#### 3.2.1.2 Data rate and data coding

The command data rate is 423.75 kbit/s ($f_c/32$). The bit interval is 2.36 $\mu$s. The *Modified Frequency Modulation* (MFM) is used as the encoding method. MFM or Miller coding is described in section 2.4.1.4. The next figure (3.18) depicts a command MFM encoding and timing of binary 00110101.

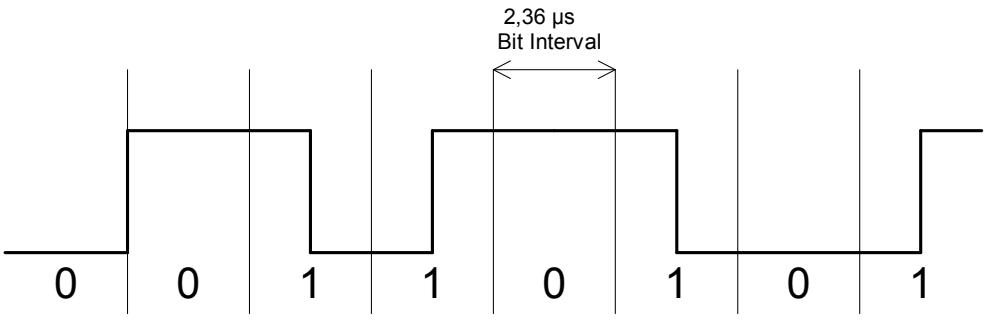


**Figure 3.18:** Command MFM encoding of binary 00110101

#### 3.2.1.3 Interrogator to tag frames

In this MODE, frames are called flags. The command flag defines the start of a command and the bit interval timings. Flags consist of three parts. The first part contains 9 bits of valid MFM data. Second comes a MFM encoding violation. The encoding violation is not present in normal data. It has three parts: a 2 bit interval (4.72 $\mu$s) , a 1.5 bit interval (3.54 $\mu$s) and at the end, another 2 bit interval. Eventually, a bit '0' defines end of a flag. Two possible flags are depicted in figure 3.19.

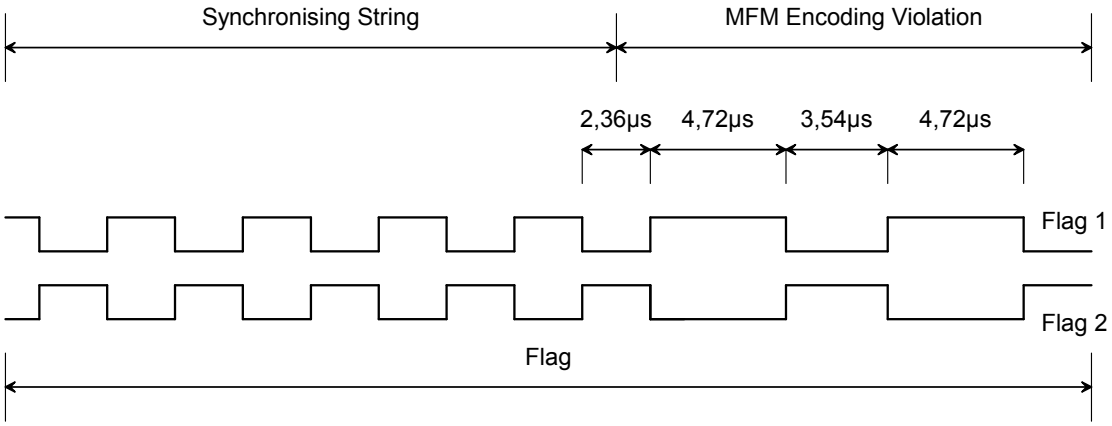


**Figure 3.19:** Two possible command flags (Interrogator to tag)[8]

### 3.2.2 Communications signal interface tag to interrogator

This MODE uses the same coding technique for replies and commands (MFM). There are 8 selectable modulated subcarriers which tags may use. Tags randomly select a channel to reply on. Table 3.1 shows possible channel frequencies and division ratios.

| Channel | Frequency [kHz] | Division ratio |
|---|---|---|
| A | 969 | 14 |
| B | 1233 | 11 |
| C | 1507 | 9 |
| D | 1808 | 7.5 |
| E | 2086 | 6.5 |
| F | 2465 | 5.5 |
| G | 2712 | 5 |
| H | 3013 | 4.5 |

**Table 3.1:** Channel frequencies and division ratios(adapted from [8])

Since the aim of this diploma thesis is to come up with a *low-cost* transceiver, the interrogator is a single channel one, meaning that tags can only reply on one channel. For this kind of interrogators, Channel G is considered to be the most suitable, although we have chosen channel B, since it is a slower one.

The modulation type is BPSK, which is also the simplest form of PSK. It uses two phases separated by $180\,°$.

#### 3.2.2.1 Data rate and data coding

The replies are MFM encoded (see section 2.4.1.4). The only difference is that the data rate is set to 105.9375 kbit/s $\approx$ 106 kbit/s and the bit interval is now 9.44 $\mu$s. An example of reply MFM encoding of binary string 00110101 is shown in figure 3.20

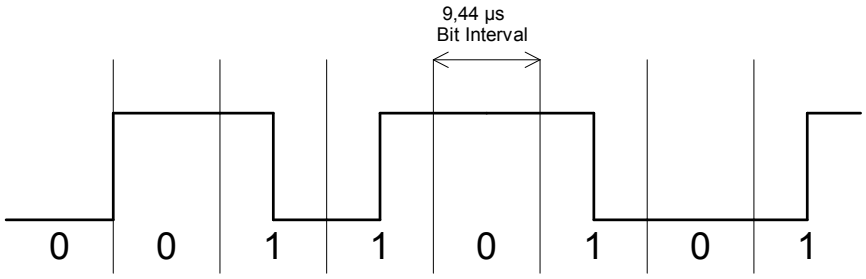


**Figure 3.20:** Reply MFM encoding of binary 00110101

#### 3.2.2.2 Tag to interrogator frames

A flag comprises of the same three parts as an interrogator to tag flag (see section 3.2.1.3), with the exception that the period of bit interval is four times longer. Figure 3.21 depicts two possible flags.

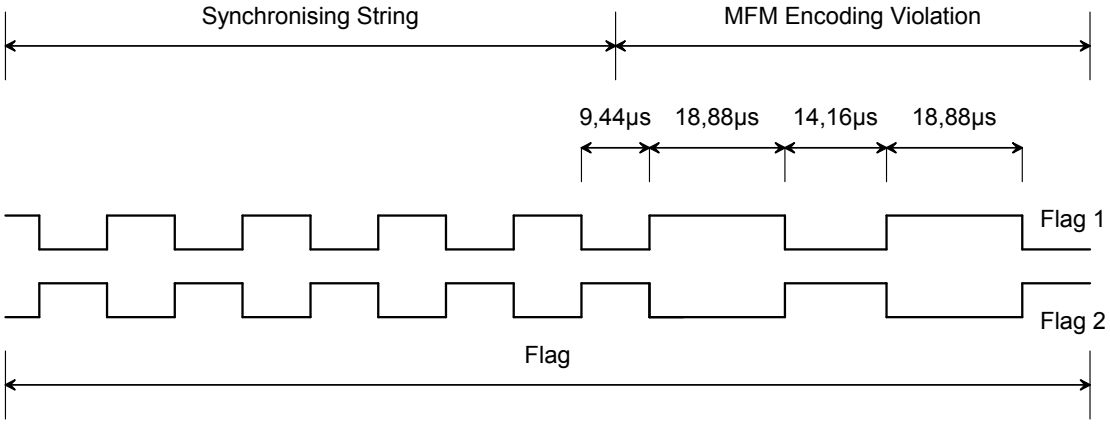


**Figure 3.21:** Two possible command flags (Tag to interrogator)[8]

### 3.2.3 Command format

The fields of a valid command are shown in table 3.2. All fields are transmitted following the *least significant bit first* principle.

| Code | Field | Bits | Comment |
|---|---|---|---|
| F | Flag | 16 | MFM violation sequence |
| Cd | Command | 16 | command field |
| Cn | Command number | 16 | command number field |
| SS | Specific identifier | 32 | used to communicate with an individual tag |
| G | Application group identifier | 16 | used to communicate with tags from the same application group |
| C | Conditional identifier | 16 | used to communicate with tags that meet a conditional test |
| PPP | Password | 48 | used to restrict writes to tag memory. It shall only be provided if required by tag. |
| R | Read address and length | 16 | 8 bit address and 8 bit length fields for memory read |
| W | Write address and length | 16 | 8 bit address and 8 bit length fields for memory write |
| Ra | Read address | 16 | 16 bit address field for memory read |
| Rl | Read length | 16 | 16 bit length field for memory read |
| Wa | Write address | 16 | 16 bit address field for memory write |
| Wl | Write length | 16 | 16 bit length field for memory write |
| D | Write data | 16 | data to be written to the tag |
| C | CRC | 16 | validation CRC |

**Table 3.2:** Command fields(adapted from [8])

Depending on the type of command the interrogator sends, different fields shall be used. Table 3.3 shows the format for *group read* and *specific write* commands.

| Command type | Start fields | Identifier fields | Address&length fields | Data | CRC |
|---|---|---|---|---|---|
| group read | F [Cd] Cn | G Ci | [R] or [Ra Rl] | | C |
| specific write | F [Cd] Cn | SS [PPP] | [W] or [Wa Wl] | D | C |

**Table 3.3:** Valid command format for *group read* and *specific write*(adapted from [8])

### 3.2.4 Tag reply format and tag states

The reply fields are shown in table 3.4. They are also transmitted based on the *least significant bit first* principle.

| Code | Field | Bits | Comment |
|---|---|---|---|
| F | Flag | 16 | MFM violation sequence |
| H | Hardcode | 16 | if tag includes hardcode then all hardcode data is sent in replies |
| T | Time stamp | 16 | is set equal to the command number included in first valid interrogator command |
| L | Lock pointer | 16 | identifier field |
| M | Manufacturing code | 16 | identifier field |
| SS | Specific identifier | 32 | identifier field |
| G | Application group identifier | 16 | identifier field |
| Ci | Conditional identifier | 16 | identifier field |
| Co | Configuration word | 16 | identifier field |
| D | Read data | 16 | data request by a valid command |
| CC | CRC | 32 | validation CRC |

**Table 3.4:** Reply fields(adapted from [8])

For the MODE2 reply, the 32 bit CRC mode is used. Appendix B gives a method explanation and an example of CRC32.

Just like VICCs, the PJM tags can also be in four different states (see section 3.22). The first state is *Power off*. Whenever a tag is in this state, it can not be activated by the interrogator. There is no field. The next state is *Active*. The tag is activated by the interrogator and it can

process any command. Once the tag has received a valid interrogator command, it switches to *Tag Reply* state. After the tag reply is complete it goes back to active state. Receiving a fully muted command brings tag into *Fully Muted* state. The tag remains in this state until receiving a new valid identifier command. Any other command will be ignored.

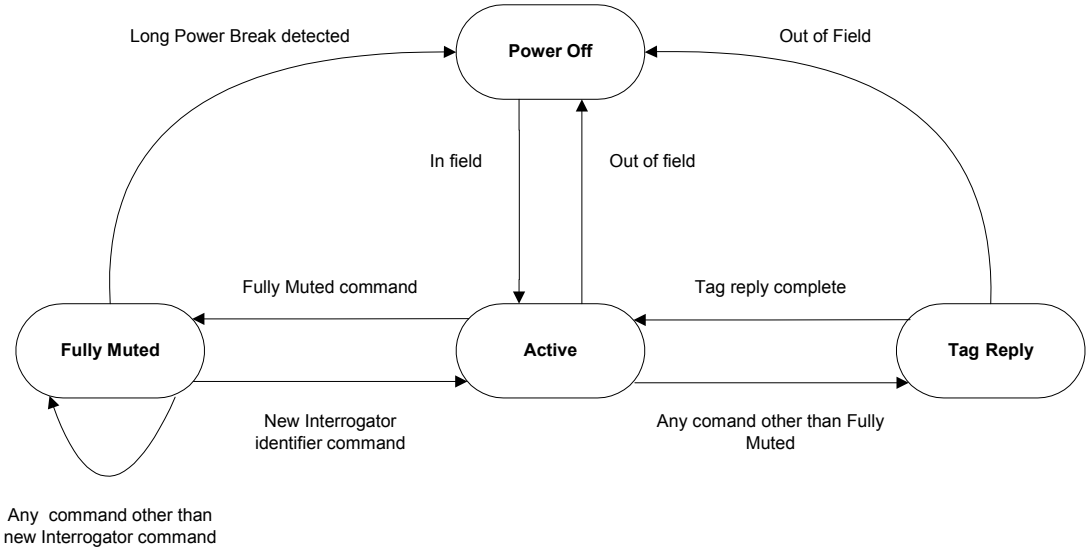


**Figure 3.22:** Tag state diagram [8]

Every tag can detect an interrogator´s field power breaks that last longer than 5 $\mu$s. Breaks between 5 $\mu$s and 10 $\mu$s are called *short power breaks*, where after the tag will initialize and wait for a command. If the break lies between 10 $\mu$s and 50 ms then the tag will exit fully muted state. This type of break is called *long power break*.

### 3.2.5 Collision management

One of the most important advantages of the ISO/IEC 18000-3 is that it allows to detect a large number of tags. To make that possible, a combination of Frequency and Time Division Multiple Access (FTDMA) is used. Tags reply to a command on a randomly selected channel. For each command, the tag will select another channel. In order to detect a large number of tags, the interrogator can temporarily mute tags, once they have been identified, so they do not reply. The next figure depicts an anticollision management sequence.

The interrogator sends a read UID command for just one channel. After receiving a tag reply, the interrogator checks CRC. If CRC is correct, the interrogator sends another command. If CRC

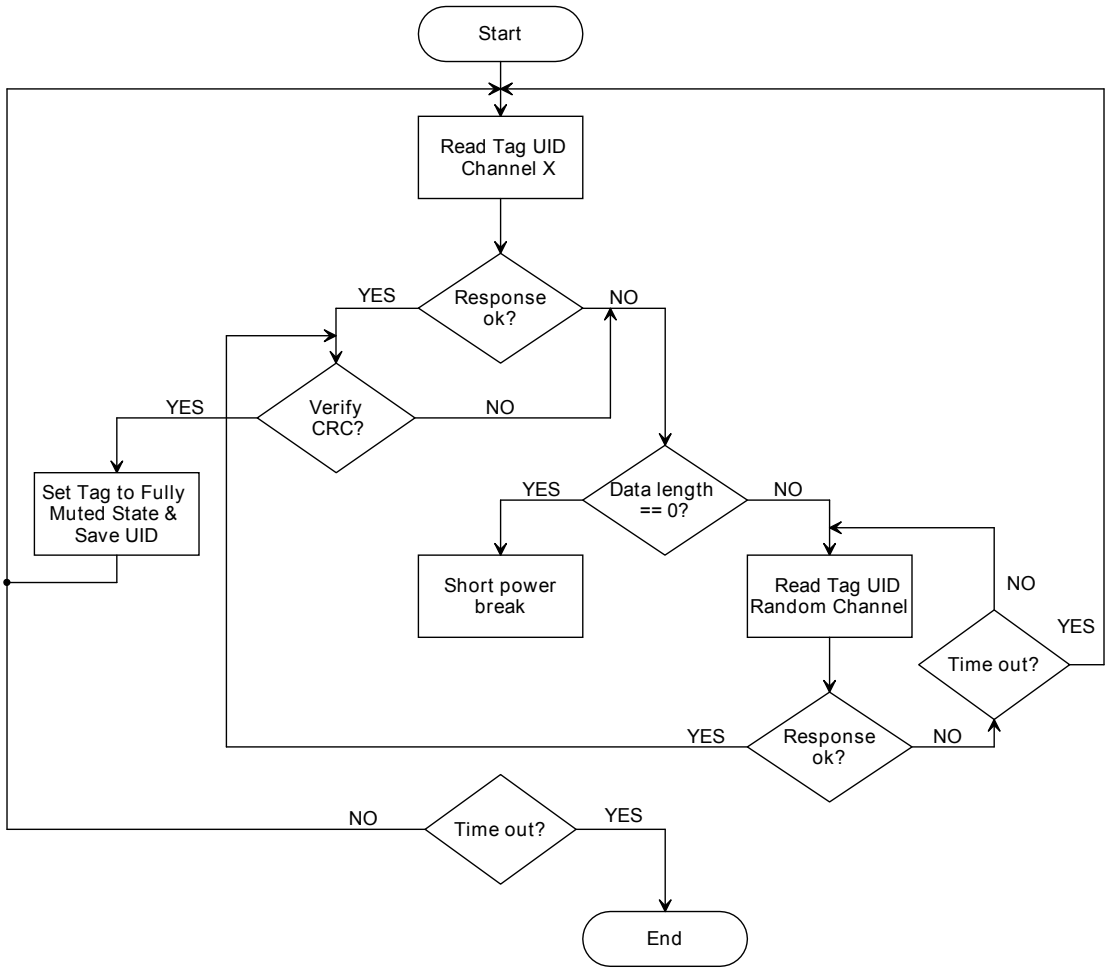


**Figure 3.23:** Anticollision management sequence in MODE2

is invalid, the data length will be checked. If there is no data (data length = 0) the interrogator makes a short power break and then sends the command again. Because of the short power break, the tags will be reinitialized, though the ones in fully muted state will not change their state. If there is data and CRC is not right, a collision has occurred. Multiply tags replied on same channel. In this case, the reader sends a read UID command for a random channel. This command will be sent for so long, until a interrogator receives a tag reply. After that, the CRC verification is run again and the same sequence starts over. The interrogator keeps repeating this sequence until all tags in field are muted and all UIDs are stored.

# 4. Coding of the coded baseband signal for ISO/IEC 18000-3

This chapter presents the practical realisation of the project. As already mentioned, the aim of this diploma thesis was to develop a contactless desktop reader supporting MODE1 and MODE2 of ISO/IEC 18000-3. In the beginning, the microcontroller of the STR71x family from ST Microelectonics was used on the reader board and Integrated Development Environment. As the decoding technique on this microcontroller did not lead to the desired results, it was replaced by a STM32Fx Cortex M3 family microcontroller. In the next sections, the coding technique for both microcontorllers will be explained.

## 4.1 MODE decision

Before a command can be sent to a tag, the modulator needs to know which MODE will be used in use. There are three possibilities: for MODE1 10% or 100% ASK and PJM for MODE2. This is done by using two output signals (PJM and ASK pins). Depending on these two signal values, the modulator detects which mode is going to be used. The next table shows pin values and their matching modulation.

| **PJM pin** | **ASK pin** | **Selected Modulation** |
|---|---|---|
| low | low | 100% ASK |
| low | high | 10% ASK |
| high | low | PJM |
| high | high | RFU |

**Table 4.1:** Signals for modulation type detection

## 4.2 Serial peripheral interface

In both cases, the buffered SPI was used. Choosing this communications interface gave us the possibility of a 8- or 16-bit transfer frame format selection, two 10-word depth (16-bit) FIFOs buffer, an internal clock prescaler and up to 18 MHz speed. In this section, the basics of the buffered SPI architecture will be explained.

The SPI allows full-duplex, synchronous, serial communication between devices. It is based on a Master-Slave protocol. A SPI system may either consist of one master and several slaves, or of a system in which devices may be either master or slave. In this case, the device operates as master.

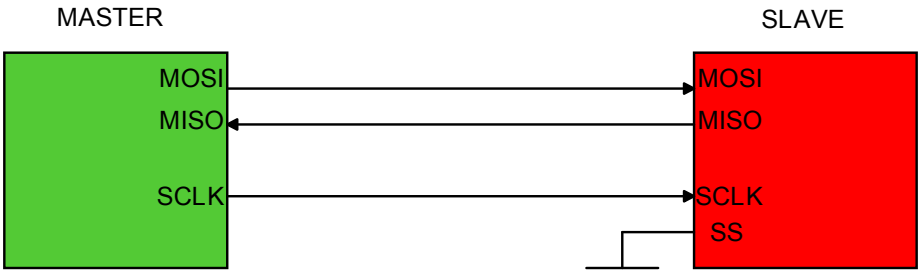


**Figure 4.1:** SPI pins

A SPI has 4 pins (see figure 4.1). The serial clock pin (*SCLK*), which is used as an output in master mode or as an input in slave mode. There are two data lines: Master Input/Slave Output (*MISO*) and Master Output/Slave Input (*MOSI*). The slave select ($\overline{SS}$) pin is used for selecting a slave device. After the SCLK is stable, $\overline{SS}$ is pulled low and kept low during data transfer. The $\overline{SS}$ on master is always high. During transfer, data is being simultaneously shifted in and out. There are two 10-word 16-bit FIFOs that can operate with 8 and 16 bit long words. The synchronization of the shifting and sampling of the data is done by SCLK. It is possible to select the clock polarity (CPOL) and clock phase (CPHA). The clock is selected by setting the clock polarity bit to active high or active low. Once the clock phase bit is cleared, the first data sample is captured on the first edge of SCLK and, if the bit is set, the data capture is on second edge.

## 4.3 Coding techniques for the STR71x microcontroller

Initially, the clock and the SPI had to be configured. This was done by using the STR71x firmware library (refer to [17]). This library consists of a collection of routines, data structures and macros that cover the features of each peripheral on a microcontroller.

### 4.3.1 Clock configuration

The STR71x microcontroller can have tree external clock oscillator. The main oscillator is used to derive the CPU and peripheral clock. On the demo board, we used a main crystal of 13.56 MHz. On the APB2 bridge, where the SPI is located, we achieved 54.24 MHz (refer to [19]). Figure 4.2 shows how this is done.

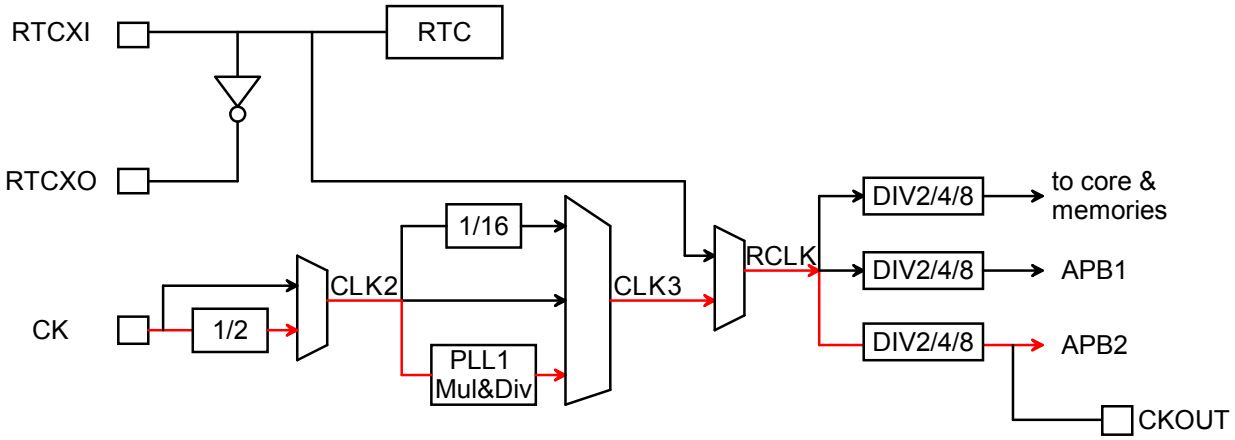


**Figure 4.2:** Clock tree of the STR71x (adapted from [10])

First, the clock *CK* is dived by two. After that, the PLL1 multiplication factor is set to 16 and the divider factor to 2. At the end, no divider factor is used for APB bridges. Mathematically, this can be described as follows:

$$CK = CLK = 13.56MHz$$
$$CLK2 = \frac{CLK}{2} = \frac{13.56MHz}{2} = 6.78MHz$$
$$CLK3 = \frac{CLK2 * 16}{2} = \frac{6.78MHz * 16}{2} = 54.24MHz$$

### 4.3.2 SPI configuration

First of all, the MOSI and SCLK I/O pins are configured as output. The $\overline{SS}$ pin is kept high. SPI is enabled and set as master. However, only the MOSI pin is used by analog part (see figure 4.3).

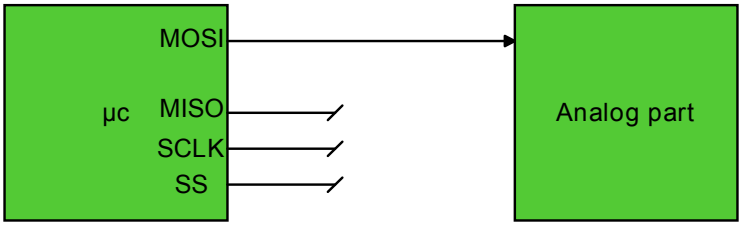


**Figure 4.3:** SPI pin usage

Since the clock is given by master, the baud rate has to be configured. This is done by using function BSPIClockDividerConfig (BSPIx, Divider factor bits) (refer to [17]). For MODE1 (vicinity card) a bit length of 9.44 $\mu$s had to be achieved (see section 3.1.2.1).The maximum divider factor bits is $256$. This way, we can average a bit length of 4.72 $\mu$s. For MODE2 (PJM), the bit length had to be 1.18 $\mu$s. Therefore the divider factor $64$ was chosen. Mathematically, this can be described as follows:

$$f = \frac{13.56MHz * 4}{256} = 0.211875MHz$$
$$t = \frac{1}{0.211875MHz} = 4.719764\mu s \quad \text{(MODE1)}$$
$$f = \frac{13.56MHz * 4}{64} = 0.8475MHz$$
$$t = \frac{1}{0.8475MHz} = 1.17994\mu s \quad \text{(MODE2)}$$

The clock polarity bit is set to $1$, while the clock phase bit is $0$. This means that SCLK idles high and that data capturing begins on the first edge of SCLK. The word length is set to $8$ bits.

### 4.3.3 MODE1 transmission

The following figure shows a flowchart of transmission for MODE1 (vicinity).

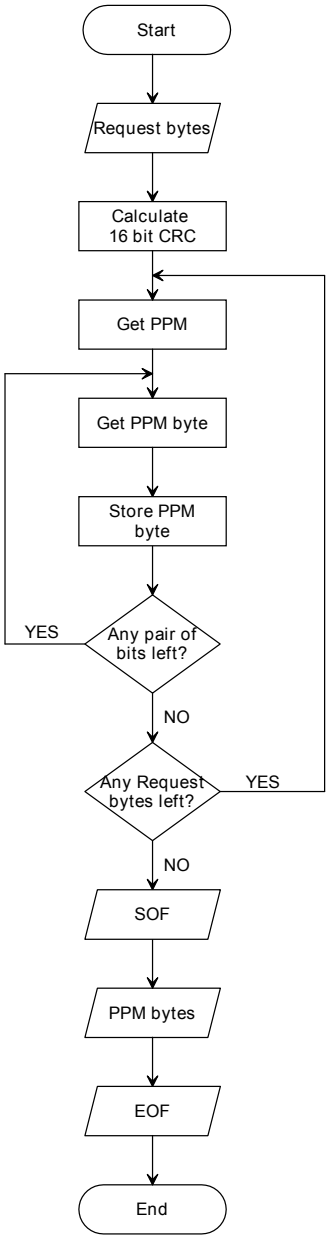


**Figure 4.4:** Flowchart MODE1 coding technique for the STR71x

Once the SPI configuration is completed, the CRC has to be calculated. With a variable *CRCmode* a 16 or a 32 bit CRC can be choose. The vicinity commands have 16 Bits CRC. After

the two CRC Bytes are calculated, they are appended to request byte stream.

MODE1 is completely compliant with ISO/IEC 15693 (see section 3.1) and for the VCD to VICC communication the 1 out of 4 pulse position modulation (PPM) was used. The *request byte* is the input value of *getPPMByte* function. This function sends a request byte to the *getPPM* function, which breaks it down into 4 pairs of bits and sends them back to the *getPPMByte* function (LSB pair of bits first) one by one. This function produces two *PPM Bytes* from each pair of bits and returns them to main function, where they get stored. The reason why two bytes are made is because with SPI clock division, the maximum time of 4.72 $\mu$ s is accomplished and the length of one bit is 9.44 $\mu$ s. The next figure shows how two PPM Bytes are made from a pair of bits (see section 3.1.2.1).

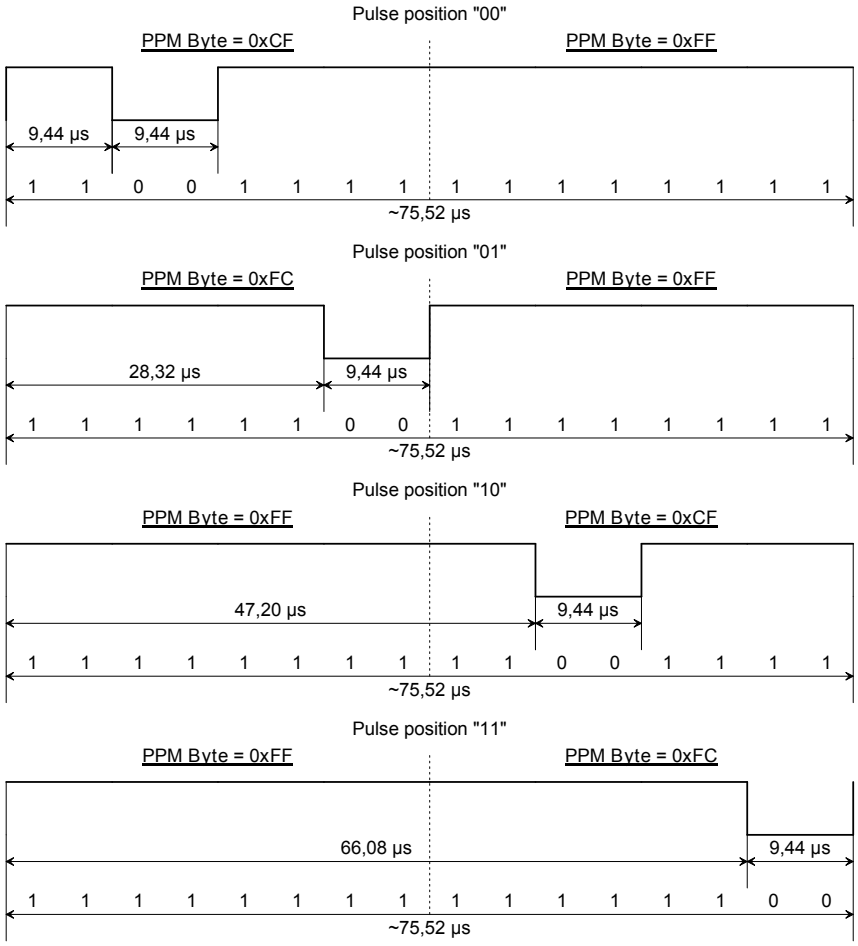


**Figure 4.5:** Making of PPM Bytes for the STR71x MODE1

After all request bytes, including two CRC Bytes, are converted to *PPM Bytes*, sending of command using SPI starts. *Start of frame*(3.1.2.1) will be sent first. Two bytes are send as SOF.

Figure (4.6) shows how these two bytes are created.

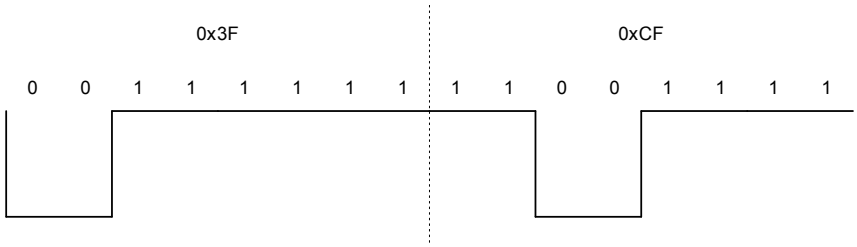


**Figure 4.6:** MODE1 Start of frame

The PPM Bytes are then sent, followed by the *End of frame*. Figure (4.7) depicts the creation of EOF bytes.

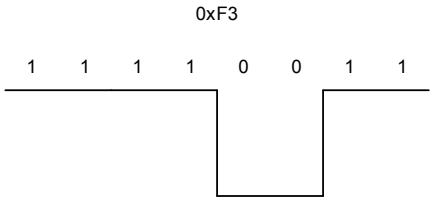


**Figure 4.7:** MODE1 End of frame

### 4.3.4 MODE2 transmission

For interrogator to tag communication the modified frequency modulation (MFM) is used. Similar as in MODE1, the first thing to do after the SPI configuration, was to calculate the CRC. For PJM commands, a 16 bit CRC was used. Two calculated CRC bytes were appended to request. Afterwards, the bytes were send to *getMFMBytes* together with the *FLAGmode* variable. This variable gives information about flag that has been used (see section 3.2). If the flag ends with high, then the data has to start with null if LSB is '0' or with one if LSB is '1' and other way round.

In order to create MFM Bytes, MFMs have to be created by the *getMFM* function. To create MFMs, first the *getBit* function has to create bits. This function returns the bits to the *getMFM* function one by one, and 2 MFMs are created from every bit. They are then sent to function *getMFMBytes*, which creates MFM Bytes from 8 MFMs. Figure 4.8 depicts the MODE2 (PJM) transmission block diagram.

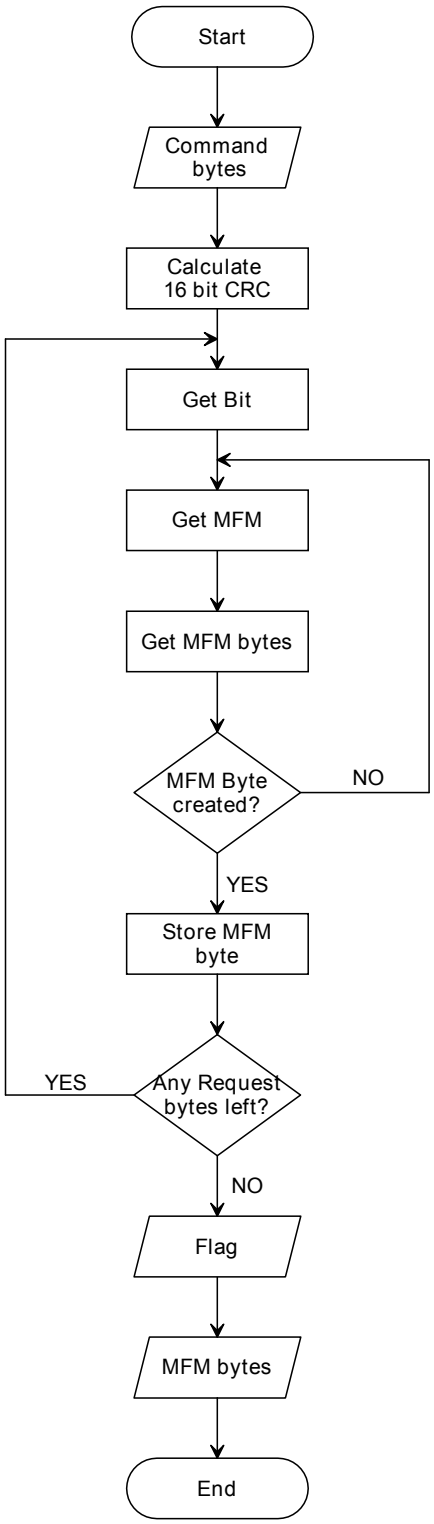


**Figure 4.8:** Flowchart MODE2 coding technique for the STR71x

Having all command bytes converted to MFM Bytes, the flag is sent first and the MFM bytes follow. The Flag is send as an array of 4 bytes. Figure 4.9 shows how these 4 bytes are determined.

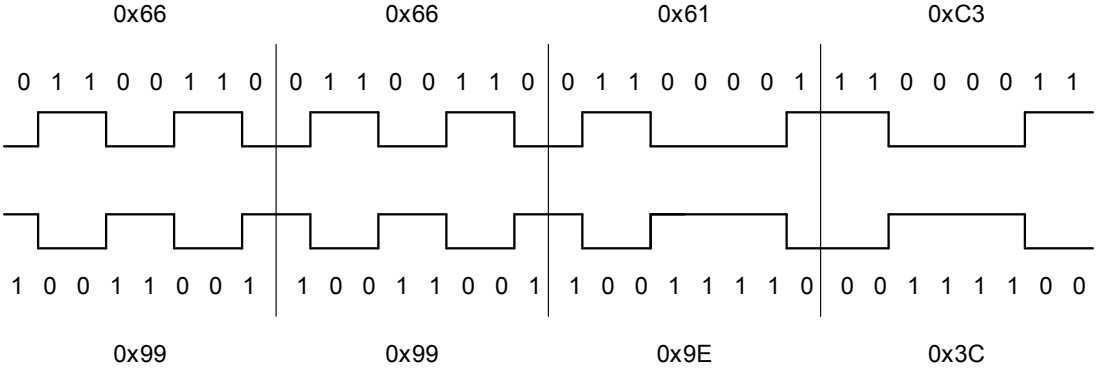


**Figure 4.9:** MODE2 flags array

## 4.4 Coding techniques for the STM32Fx microcontroller

As already mentioned, the SPI communications interfaces were, also, used for the STM32x. Since the coding technique used for the microcontroller STR71x gave us satisfying results, the same principle was also applied here.

### 4.4.1 Clock and SPI configuration

The STM32x microcontroller has two internal RC oscillators: high speed internal (HSI) and low speed internal (LSI) and, also, two external oscillators: high speed external (HSE) and low speed external (LSE) (see figure 4.10). The processor can be clocked by either HSE or HSI or from an internal phase locked loop (PPL). In our case the PPL is run by a HSE 13,56 MHz. The output frequency of PLL is then multiplied by 8 and afterwards the APB2 prescaler 8 is used to achieve 13,56 MHz by SPI1, since it lies on the APB2 bridge. The multiplication by 8 is done to enable timer 3 to preform an input capture on every rising edge needed for the decoding technique (see section 5.2). Since the maximum frequency for the STM32 is 72 MHz, choosing no prescaler for APB1 led to an overclocking of the peripherals on APB1 (108,48 MHz). During the work on this diploma thesis we have experienced no side effects of overclocking. However, some of the side effects of overclocking, could be the rise of current consumption or reduction of the operating

temperature range. The red line in next figure shows the clock configuration path for SPI and timer3.

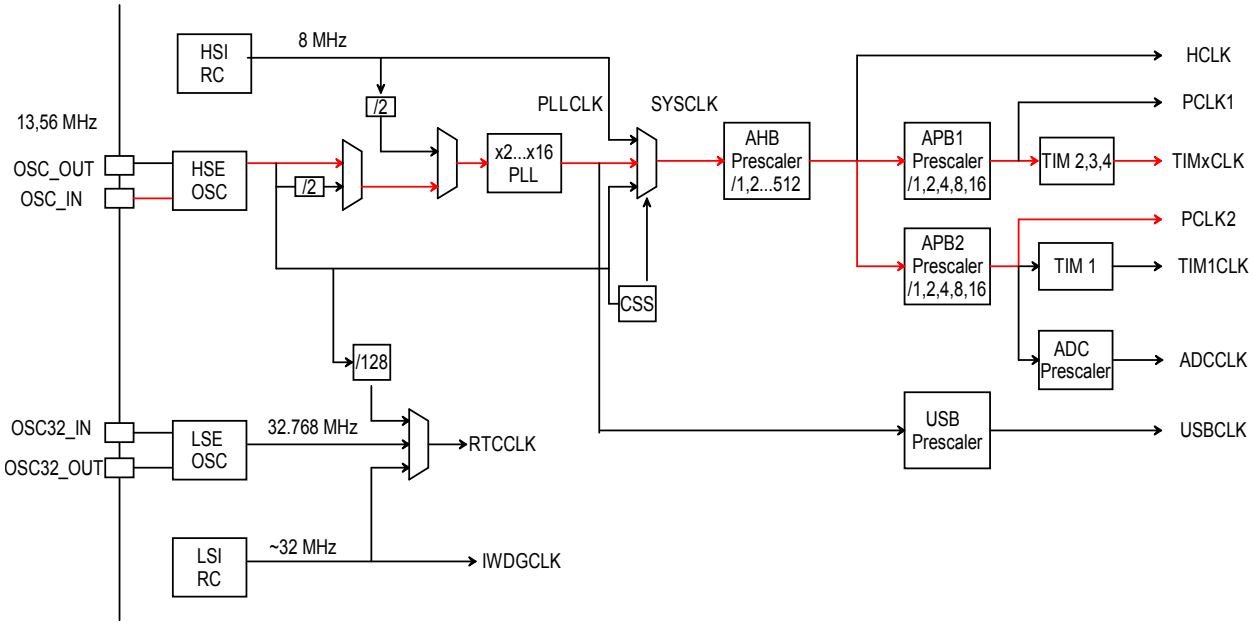


**Figure 4.10:** Clock tree of the STM32x (adapted from [11])

The SPI configuration was done exactly as in our previous case. The MOSI and SCLK pin (see figure 4.1) were configured as output.The $\overline{SS}$ pin was configured as input and kept high. SPI was enabled and set as master. The prescaler was last set. This time, it was possible to achieve exactly 9.44 $\mu$s needed for MODE1 with prescaler 128. For MODE2, prescaler 16 was used to achieve 1.18 $\mu$s. Mathematically, this can be described as follows:

$$f = \frac{13.56MHz}{128} = 0.1059375MHz$$
$$t = \frac{1}{0.1059375MHz} = 9.4395\mu s \quad \text{(MODE1)}$$
$$f = \frac{13.56MHz}{16} = 0.8475MHz$$
$$t = \frac{1}{0.8475MHz} = 1.17994\mu s \quad \text{(MODE2)}$$

### 4.4.2 MODE1 transmission

After the configuration, the *CRC* had to be calculated for all command bytes from *SOF* to *EOF*. The functions *calculateCRC* arguments were *currentCRCvalue* and *RequestByte*. The initial value of the variable *calculateCRC* was a preset value of 16 bit CRC. The particularized CRC calculation example is described in appendix A. Two calculated CRC bytes were appended to request.

After that, the request was coded using a 1 out of 4 pulse position modulation. The function *getPPMByte* took one *Request Byte* and made 4 pair of bits, returning one *PPM Byte* for each pair. The determination of a *PPM Byte* is shown in the next figure.

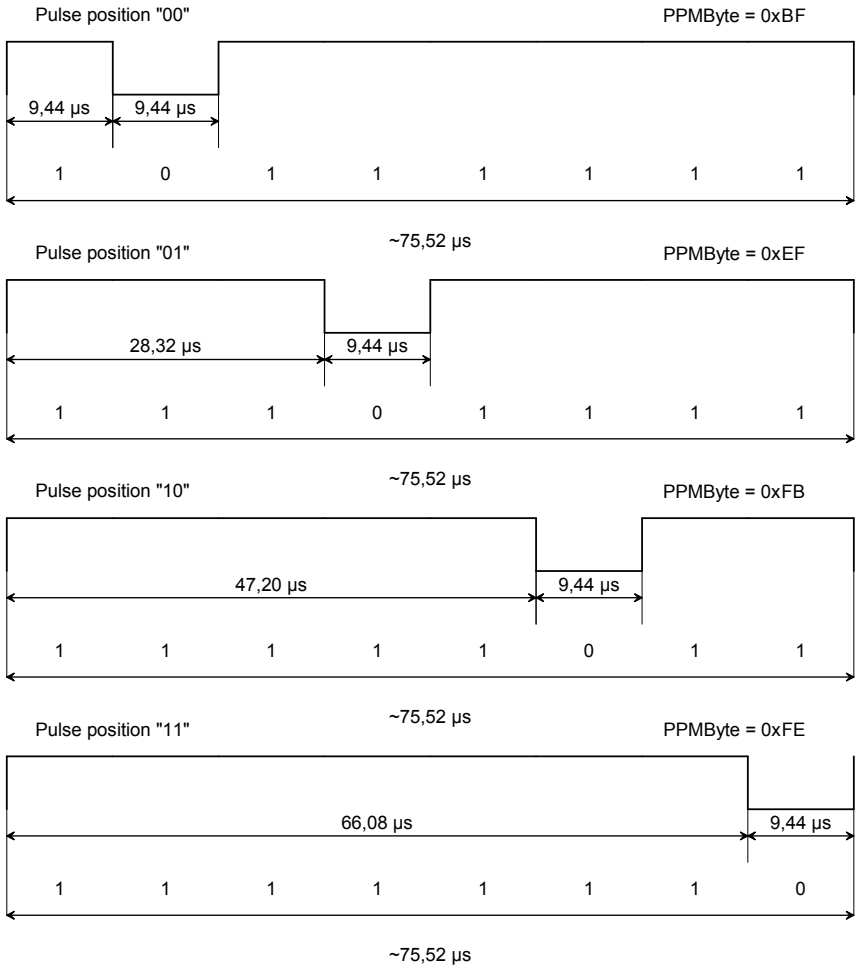


**Figure 4.11:** Example of making PPMBytes for the STM32x

After all *Request Bytes* and CRC are coded, the SPI transmission starts in the following order:

*SOF*, coded request, *EOF*. The next figure (4.12) shows a flowchart of this coding technique.

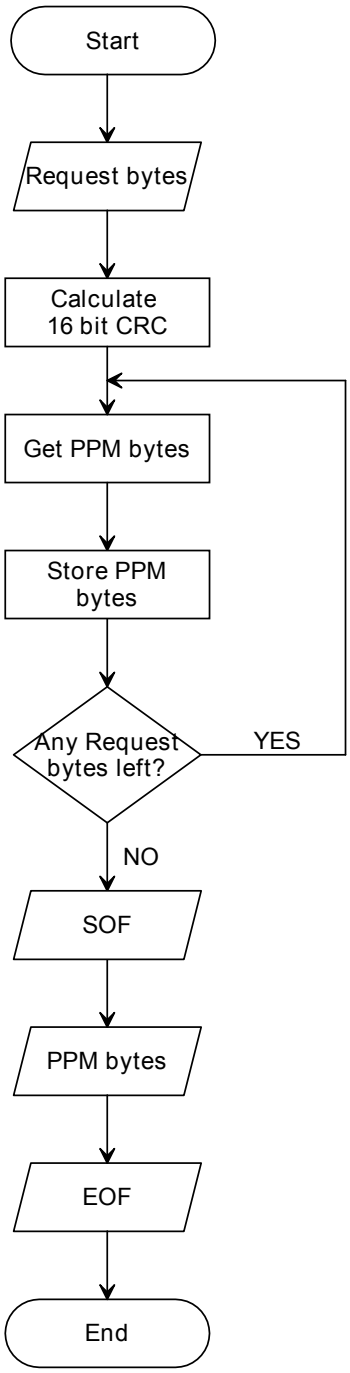


**Figure 4.12:** Flowchart MODE1 coding technique for the STM32x

### 4.4.3 MODE2 transmission

Figure 4.13 depicts the MODE2 (PJM) transmission flowchart.

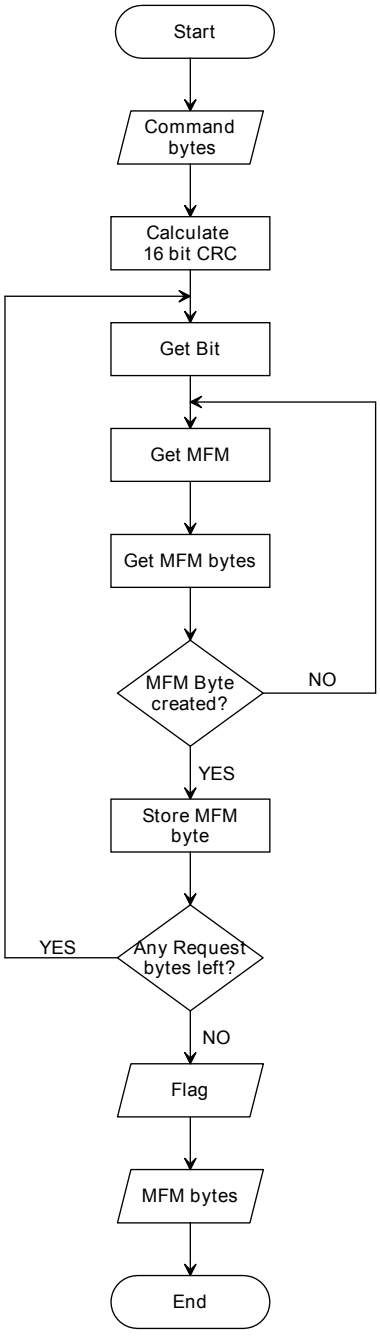


**Figure 4.13:** Flowchart MODE2 coding technique for the STM32x

For CRC calculation, the same *calculateCRC* function was used as for MODE1. And the two

*CRC bytes* were appended to command. After that, the *getMFMByte* was called. Arguments of this function are *Command Byte* and *FLAGmode*. *FLAGmode* provides an information about the end of flag (high or low). 8 *MFM* bits are required to create one *MFMByte*. This is done by using function *getMFM*. This function takes one *Command Byte* and sends it to function *getBit*. The *getBit* function creates 8 bits from one *CommandByte* and returns them one by one. *getMFM* takes one bit and creates two *MFM* bits and returns them one by one. After *getMFMBytes* receives 8 *MFM* bits, it returns a *MFMByte*. After all *CommandBytes* are converted, the *FLAG* is sent and all *MFMBytes* follow using the SPI MISO pin (see figure 4.3). For *FLAG*, the same 4 Bytes array was used as for the STR71x microcontroller (see figure 4.9).

# 5. Decoding of the coded baseband signal for ISO/IEC 18000-3

In this chapter, the decoding techniques according to ISO/IEC 18000-3 will be discussed. The data received from tag is decoded and the CRC, which is appended to the data, is additionally verified. Manchester coding is used for MODE1 (see section 3.1) and modified frequency modulation for MODE2 (see section 3.2). This decoding technique was developed for two types of microcontrollers the STR71x and the STM32Fx.

## 5.1 Decoding techniques for the STR71x microcontroller

The first attempt was to use the STR71x microcontroller. The concept was to use input capture mode of the microcontroller´s timer. In this mode, the timer count would be captured every time there is a transition (rising or falling edge) on a matching input capture pin.

### 5.1.1 MODE1 decoding

The maximum speed from detection of a transition to the execution of the next command line was about 3.4 $\mu$s. Since the modulated part of logic $0$ and logic $1$ had $8$ pulses with a period of 2.36 $\mu$s (see section 3.1.2.2) this concept was based on the idea that the microcontroller should be able to detect every second transition.

$$\frac{f_c}{32} = \frac{13.56MHz}{32} = 423.75kHz \tag{5.1}$$

$$T = \frac{1}{423.75kHz} = 2.36\mu s \tag{5.2}$$

The figure 5.1 shows flowchart for this decoding technique.

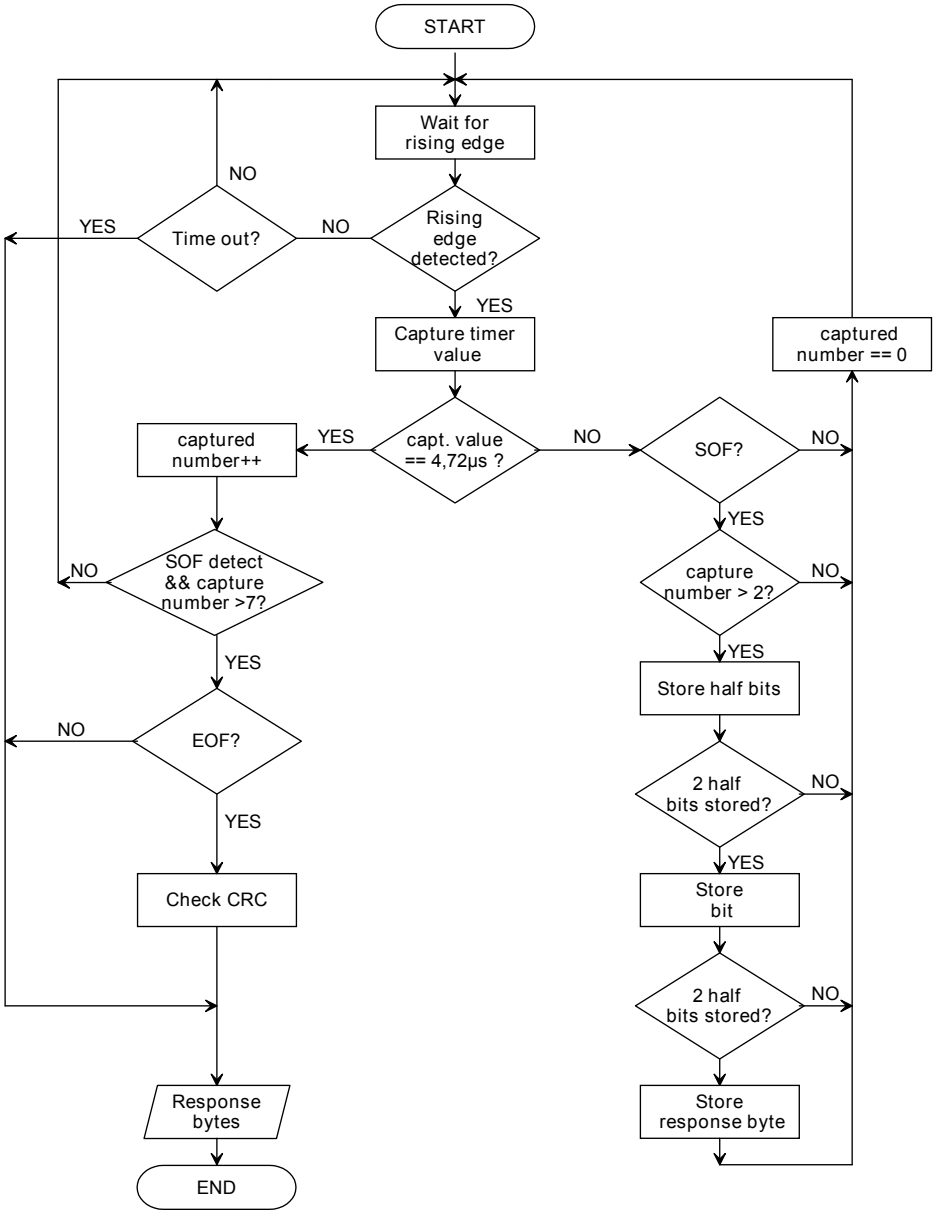


**Figure 5.1:** Flowchart MODE1 decoding technique for the STR71x

Every time a transition was detected, the counter value of timer 1 would be captured. After that the timer would be reset. This measure was taken to avoid timer overflow. Depending on the counter value, the half bits would then be created. Detecting three 4.72 $\mu$s periods in row means half bit 1 is detected. 23,6 $\mu$s is half bit 0 (see figure 5.2) and 42.48 $\mu$s is two half bits in a row. If there were seven 4.72 $\mu$s periods, two half bits 1 were detected.

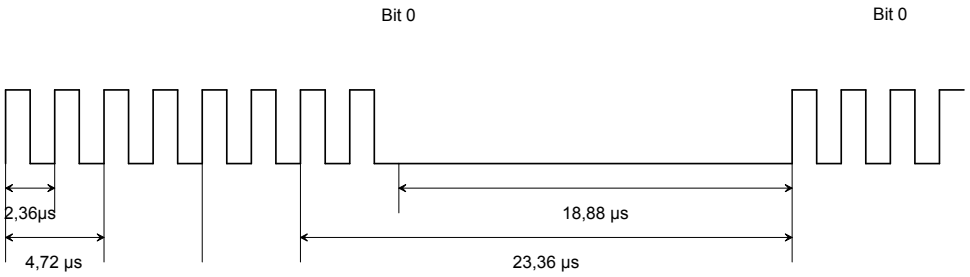


**Figure 5.2:** Example of vicinity periods of binary 1

The first thing to do was to detect SOF. This would take a waiting time of eleven periods of 4.72 $\mu$s with an additional period of 18.88 $\mu$s. Since bit one is also a part of SOF, the first four 4.72 $\mu$s periods after SOF are ignored. Until the actual bit detection starts. Every time two half bits are detected, a bit is created and after 8 bits a byte is created. EOF means eleven 4.72 $\mu$s periods in a row. After the EOF is detected, the 16 bit CRC would be verified.

In the practice, the microcontroller was not able to detect every second rising edge which lead to wrong byte detection. Since the detecting of MODE2 reply was also not satisfying it was decided to chose another microcontroller.

### 5.1.2 MODE2 decoding

In MODE2, channel B (f = 1.23 MHz) was used, which means that there were 11 BPSK per bit (see section 3.2.2). Since it was not possible to make input capture for every rising or falling edge, we planed to make a digital signal which would generated a pulse (peak) on every phase change (see figure 5.3). This should have been a part of another diploma thesis (analog part of a reader) but was never implemented because of time limitation.

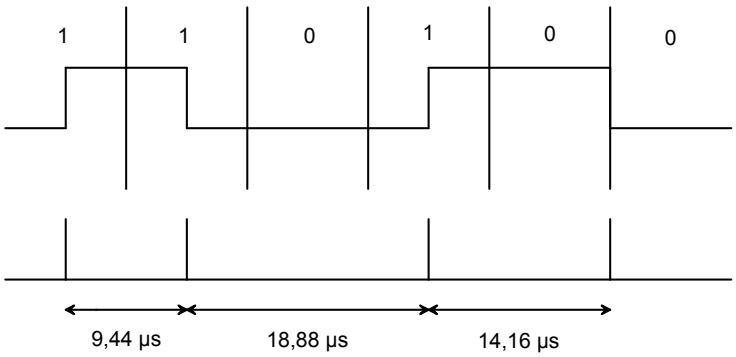


**Figure 5.3:** MODE2 possible reply sequence

Detecting every phase change would make it possible to decode tag reply. The number of equal MFMs in a row could be determined by the time elapsing between two input captures. There are three possibilities: 9.44 $\mu$s or two MFMs in a row (11 / 00), 14.16 $\mu$s or three MFMs (111/000) and 18.88 $\mu$s or four MFMs (1111/0000) (see table 5.1).

| Time between two input capture | MFM bits |
|---|---|
| 9.44 $\mu$s | 11 / 00 |
| 14.16 $\mu$s | 111 / 000 |
| 18.88 $\mu$s | 1111 / 0000 |

**Table 5.1:** MFM bits detection MODE2 for the STR71x

The first step was to detect the flag. As already mentioned in section 3.2.2.2 the flag has MFM encoding violation not present in normal data. There are $4$ state changes separated by a two bit interval, a $1,5$ bit interval and a $2$ bit interval with a bit $0$ appearing at the end. Depending on the LSB, of tag reply there are two possibilities for flag array. If LSB is $0$, the flag array would be [18.88; 14.16; 18.88; 9.44] otherwise it would be [18.88; 14.16; 18.88; 14.16].

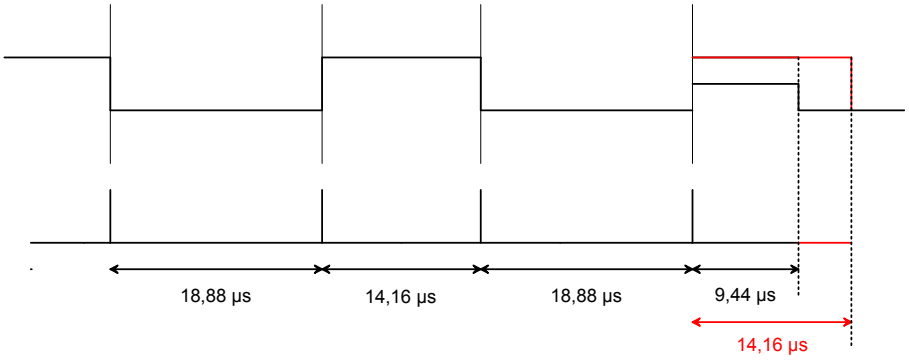


**Figure 5.4:** MODE2 reply flag

After flag detection, the decoding process starts. Input values for function *storeMFM* are the number of MFMs and a flag type (reply begins with low or high). Every time two MFMs are stored, a bit will be created. Two equal MFMs in a row mean bit $0$ otherwise it is $1$. After $16$ bits, a word is created. An end of tag reply is detected whenever waiting time for next input capture exceeds 18.88 $\mu$s. After that, a reply CRC is checked. The same function *calculateCRC* is used for the coding technique, only that this time, *CRCmode* is $32$ bits. The CRC would be calculated

for all words except for the last two. After the calculation is done, the two calculated CRC words would be compared with the last two decoded words. If they are the same, the CRC is correct. The flowchart of this decoding technique is depicted in figure 5.5.

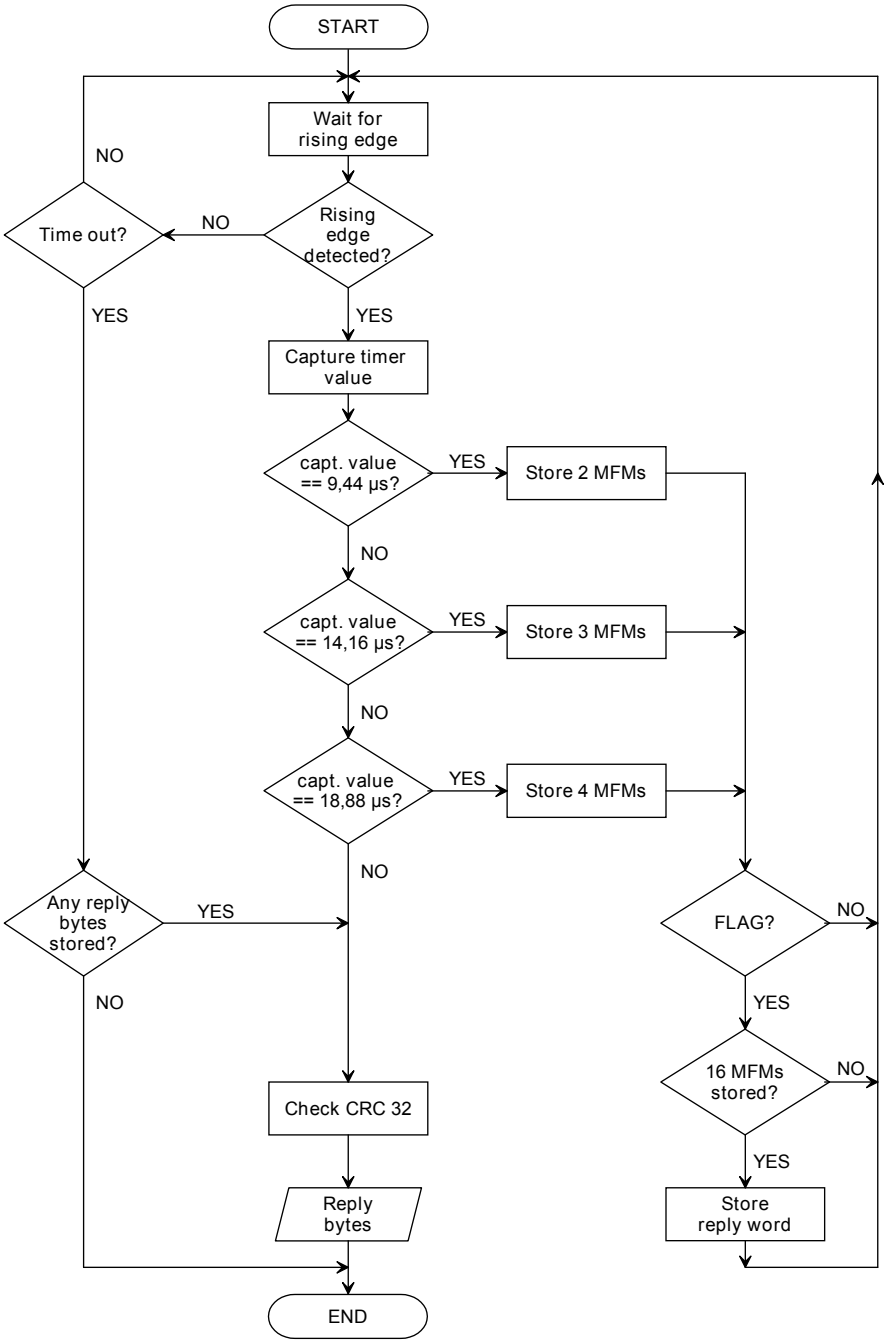


**Figure 5.5:** Flowchart MODE2 decoding technique for the STR71x

## 5.2 Decoding techniques for the STM32Fx microcontroller

Since using the STR71x has not produced the desired results for our decoding technique, another microcontroller, the STM32F103ZE, was chosen. This is a ARM-based 32-bit microcontroller with $254$ to $512$kB and up to 72 MHz frequency. With this microcontroller, the time from detection one input capture until next command line was less then 2 $\mu$s, so it was possible to create a satisfying decoding methode.

### 5.2.1 MODE1 decoding

Here, the same concept with input capture mode was used. This time it was possible to detect every rising edge of the MODE1 response, since the time from input capture detection to next command line was less than 2 $\mu$s, while the minimal time period between two rising edges was 2.36 $\mu$s (see section 3.1.2.2).

As mentioned before, every time a transition is detected (in our case a rising edge) a $16$ bit value of the counter will be latched. The time period is calculated by subtracting two consecutive captures in order to avoid timer overflow. The next figure shows three possible counter values on the example of binary string 001.

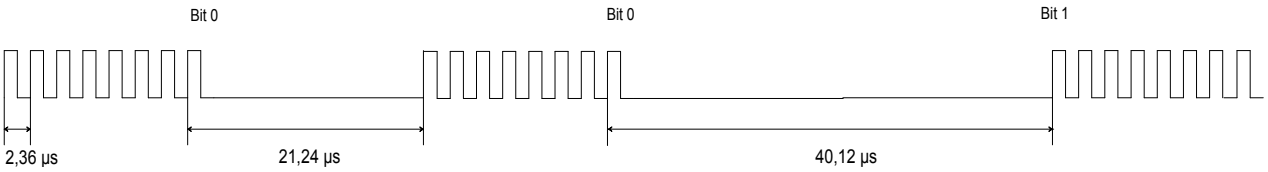


**Figure 5.6:** Example of MODE1 periods of binary 001

For instance, the calculated period for period 2.36 $\mu$s was equal to $(100)_h$.

$$
\begin{aligned}
&f_{timer} = 108.48MHz \\
&T_{timer} = \frac{1}{108.48MHz} = 9.218ns \\
&t = 100_h = 256_d \\
&256 * 9.218ns = 2.36\mu s
\end{aligned} \tag{5.3}
$$

In order to keep the reader robust, the timer counter values up to $(350)_h$ were detected as 2.36 $\mu$s. This way it was still possible to detect the modulated part of a bit even if the demodulator would make an error, or the input capture would skip one rising edge. If the captured timer counter value would be less then $(70)_h$ (less than 1 $\mu$s between two rising edges), the pulse would be ignored. Also, the intervals for the other two possible periods (21.24 $\mu$s and 40.12 $\mu$s) were chosen widely (up to $\pm$10 $\mu$s).

First of all, the *SOF* had to be detected. This would mean that 23 periods of 2.36 $\mu$s in a row are followed by unmodulated period of 21.24 $\mu$s. Since the valid MODE1 reply can not contain more then 16 pulses in a row, it was decided that everything above 18 and under 24 pulses would mean an SOF sequence. This of course, applies only for the case that other parts of *SOF* are present. In this way the *SOF* would still be detected even if rising edge would be skipped by the microcontroller or the demodulator would make an error. After this sequence is detected, the next seven 2.36 $\mu$s periods are ignored, since they are also part of the *SOF* (see figure 5.7). Every time a 2.36 $\mu$s is detected, the counter would be incremented by one. If a longer period is detected, the counter would be reseted to null. In this way, all incoming data from demodulator will be ignored until the SOF is detected.

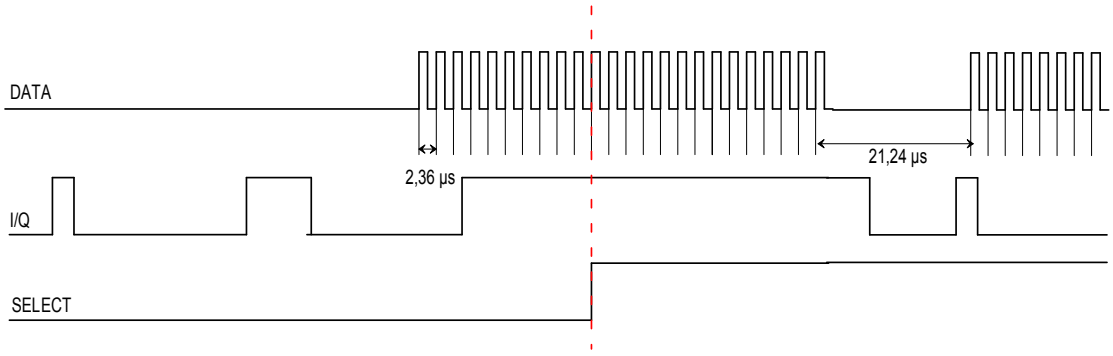


**Figure 5.7:** MODE1 SOF detecting and demodulator channel decision

During the SOF the demodulator channel decision needs to be made. In the analog part (demodulator), two signals are created (I and Q) and the better one is sent to the microcontroller. In order to avoid an I/Q switch during the response, the microcontroller controls the I/Q switching. After the first 11 pulses of the SOF, the decision is on the *I/Q* pin (high or low). The microcontroller reads this value and sends it back to the demodulator via *SELECT* pin (see figure 5.7). After receiving this value, the demodulator switches to the desired channel.

After the SOF was detected, the *Response Bytes* had to be created using function *getRespon-*

*seByte*. As long as the 2.36 $\mu$s periods are detected, the value of variable *captureNumber* is incremented by one. Once a longer period is detected, function *getResponseByte* is called with arguments *captureNumber* and a value of longer period (*longPeriod*). In this function the *half bits* are created. A seven 2.36 $\mu$s periods means half bit 1. In order to keep the reader error-tolerant, a half bit 1 was detected every time there were more than 3 and less than 8 2.36 $\mu$s periods in a row. If there were more than 8 periods and less than 16, that would mean two half bits 1 were detected. A 18.88 $\mu$s period is 0 and 40.12 $\mu$s means two 0 half bits in a row. Table 5.2 shows how half bits are detected.

| | **(3 < CN < 8) * 2.36 $\mu$s** | **(8 < CN < 16) * 2.36 $\mu$s** |
|---|---|---|
| **21.24 $\mu$s** | 1 0 | 1 1 0 |
| **40.12 $\mu$s** | 1 0 0 | 1 1 0 0 |

**Table 5.2:** Creation of half bits for MODE1 STM32x

Every time two half bits are detected, the *createResponseByte* is called. This function then creates a bit from two half bits (see section 3.1.2.2) and after eight bits are detected, it creates a *Response Byte*.

*EOF* is detected when there are more than eighteen (ideal case 23) 2.36 $\mu$s periods in a row and the unmodulated period after these pulses needs to be longer than 40,12 $\mu$s. Also, the last detected bit needs to be 0. If there are more than 16 pulses but no other parts of *EOF* the detected response bytes will be stored but the CRC will not be checked.

The *CRC value* will be calculated for all *Response Bytes* and eventually the value will be compared with the *Residue* value for a 16 bit CRC (see Appendix A). If the two values are equal, CRC is correct and MODE1 response is valid. The figure (5.8) depicts a flowchart for this decoding technique.

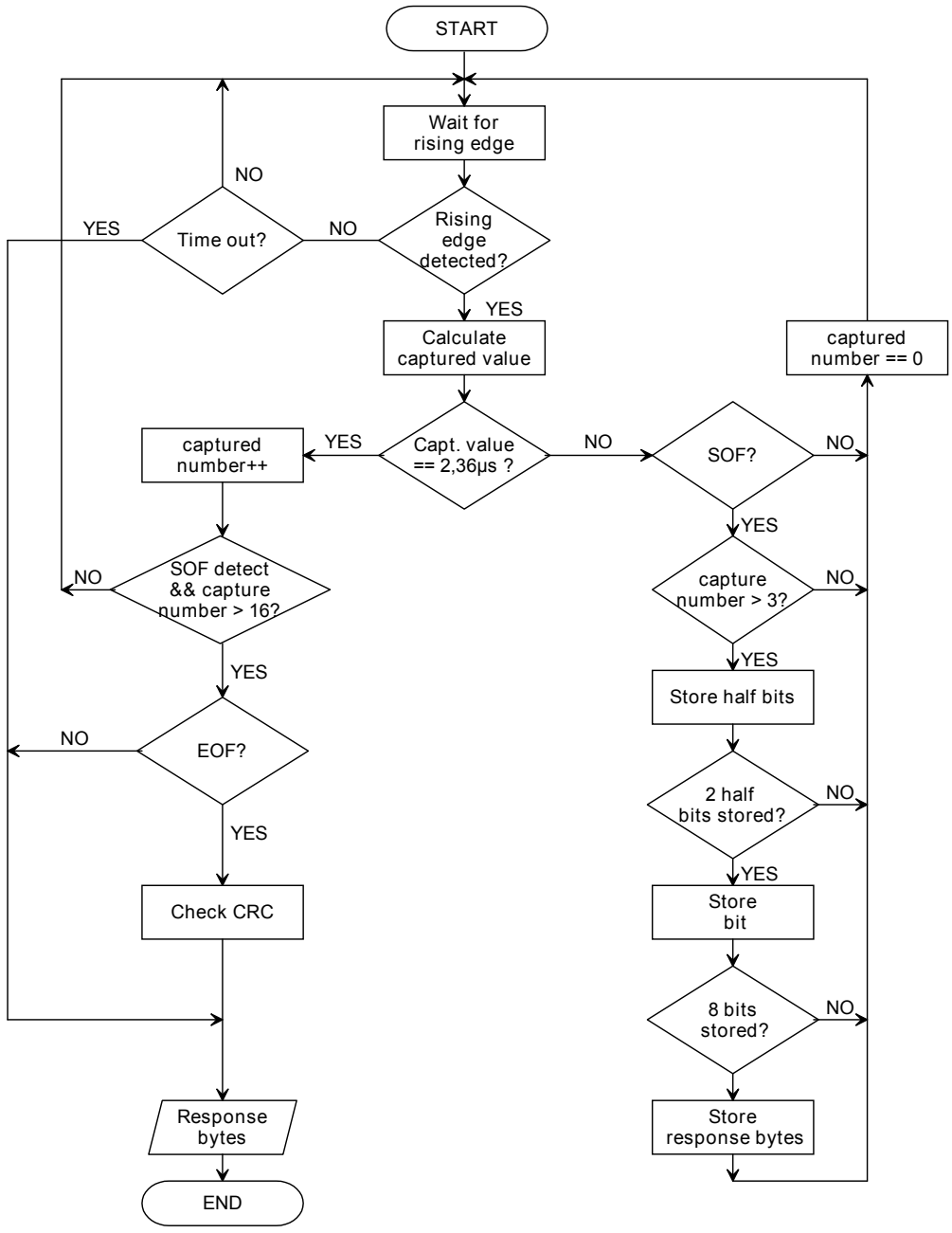


**Figure 5.8:** Flowchart MODE1 decoding technique for the STM32x

### 5.2.2 MODE2 decoding

The decoding technique for MODE2 was also based on input capture. The division factor for channel B is $11$ so the subcarrier frequency is 1.23 MHz (5.4).

$$f_{subcarrier} = \frac{13.56MHz}{11} = 1.23MHz \tag{5.4}$$

$$T = \frac{1}{1.23MHz} = 0.813\mu s \tag{5.5}$$

The period of 813 ns (5.5) and the maximum speed of input capture detection is about 2 $\mu$s so it was decided to make an input capture out of every fourth rising edge. This was done using the STM32x library (refer to [16]). Using the timer input capture prescaler, it was possible to chose when the timer value capture would be preformed. Just like in MODE1, here we also calculated the time period by subtracting two consecutive captures in order to avoid timer overflow. Since one bit actually takes a $11.64$ BPSK periods (5.6), the time elapsed between four rising edges is not always the same. Normally, this period equals 3.252 $\mu$s (5.7), but at phase change, the period would be shorter or longer. Equasion (5.8) shows how long the period would be if the bit starts with a whole period and ends with exactly $0.64$ of a period. In practice, this period can also be longer than 3.252 $\mu$s, so this detection technique is based on the fact that every time a phase change occurs, the detected period will not be 3.252 $\mu$s.

$$R = 106kbit/s \qquad \text{(Data rate)}$$

$$T_{bit} = \frac{1}{106kbit/s} = 9.44\mu s$$

$$DR = \frac{9.44\mu s}{0.813\mu s} = 11.64 \tag{5.6}$$

$$0.813\mu s * 4 = 3.252\mu s \tag{5.7}$$

$$0.813\mu s * 3 + 0.813\mu s * 0.6 = 2.9268\mu s \tag{5.8}$$

If we detected the period of 3.252 $\mu$s, the calculated period would, normally, be $(160)_h$. The period between $(150)_h$ and $(180)_h$ was considered as a period without phase change. Subsequently, every other captured value would mean a phase change.

$$
\begin{aligned}
&f_{timer} = 108.48MHz \\
&T_{timer} = \frac{1}{108.48MHz} = 9.218ns \\
&t = 160_h = 352_d \\
&352 * 9.218ns = 3.24473\mu s \approx 3.25\mu s
\end{aligned} \tag{5.9}
$$

This way, it was possible to detect how many equal MFMs there are in a row. Detecting two equal periods (≈3.252 $\mu$s) followed by a different one would mean two MFMs in a row (9.44 $\mu$s). Four equal MFMs in a row (18.88 $\mu$s) would then mean five equal periods and one different following, whereas everything in between would mean three MFMs in a row (14.16 $\mu$s). After the MFMs are detected, the bits are created the same way as described in 5.1.2.

Just like before, the first step is to detect the flag. But before flag, an I/Q decision is to be made. As mentioned in section 3.2.2.2, the first part of flag is a synchronizing string of 9 bits of valid data. After detecting two of these bits, the I/Q decision would be made. Exactly like in MODE1, the decision will be on the input pin. The microcontroller reads this value and sends it back to the analog part. After that, the demodulator switches to the desired channel.

Since there is no EOF by MODE2 the end of data would be detected if waiting time for the next input capture would exceed 5 $\mu$s. After the end of data, the 32 bit CRC is to be checked. The flowchart for this technique is shown in the figure 5.9.

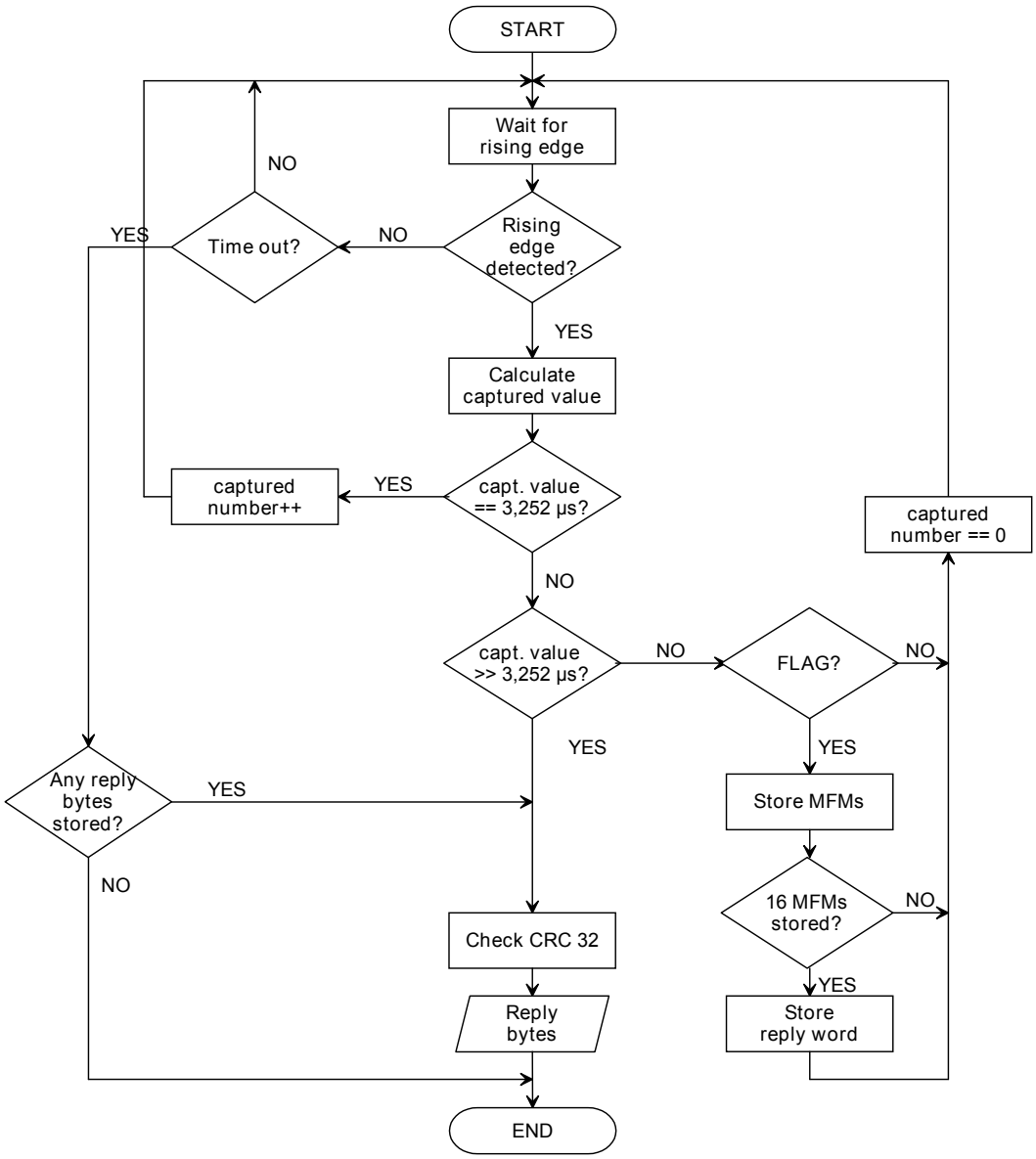


**Figure 5.9:** Flowchart MODE2 decoding technique for the STM32x

# 6. Reader implementation

As already mentioned in chapter 1 this diploma thesis deals with digital part of a contactless desktop reader. The analog front-end of a reader is a part of another diploma thesis. This chapter will present the analog front-end microcontroller interface. After that the user interface commands for this reader are introduced.

## 6.1 Interface between microcontroller and analog front-end

The interface for the communication between analog part and microcontroller is depicted in figure 6.1. The interface is composed of 6 pins.

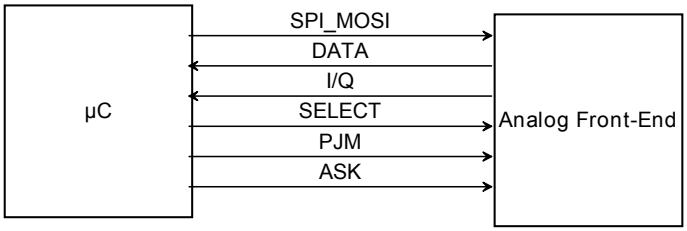


**Figure 6.1:** Interface between microcontroller and analog front-end

The *SPI MOSI* pin sends the MODE1 and MODE2 commands to modulator (see section 4.2). The *DATA* pin is used to read the tag response. The next two pins are responsible for the I/Q decision. After the demodulator has chosen the better signal it sends the decision to microcontroller via pin *I/Q*. The microcontroller reads this decision and sends it back to demodulator using pin *SELECT*. A high value means that I channel has been chosen. Channel Q has been chosen, when red value is low. The next two pins, *PJM* and *ASK*, are mode decision pins (see Section 4.1).

## 6.2 User interface commands

At the end of this diploma thesis user interface command sets were made for both MODEs. These command set combine coding and decoding techniques with the needed configuration of the microcontroller.

### 6.2.1 User interface command set for MODE1

For MODE1, the user needs to choose the appropriate ASK mode by sending the mode value first and then the command bytes. The commands are:

**Init_Vicinity** (mode) this command preforms the SPI and clock configuration (see section 4.4.1). Depending on the *mode* value (ASK 10% or ASK 100%), the PJM and ASK pins are going to be set accordingly (as described in Table 4.1).

**Calculate_CRC16** (host_len1, host_buf1) the MODE1 command bytes are stored in *host_buf1*. *Host_len1* give us the number of stored bytes. This command calculate the CRC value and appends two CRC bytes to the command bytes.

**Vicinity_Request** (host_len1, host_buf1) combines the coding and decoding technique for MODE1, as explained in the previous two chapters. The request bytes are coded and sent using SPI after the VICC response. The response will be decoded and stored in *host_buf1*. *Host_len1* keeps the number of decoded response bytes.

**Check_Vicinity_CRC16** (host_len1, host_buf1) if the EOF is detected during response decoding, this function will be called to check whether CRC is correct.

### 6.2.2 User interface command set for MODE2

The user interface command set for MODE2 is similar:

**Init_PJM** command preforms the SPI and clock configuration (see Section 4.4.1). It also sets the PJM and ASK pins (as described in table 4.1).

**Calculate_CRC16** (host_len1, host_buf1) calculates the CRC value and appends two CRC bytes to command bytes.

**PJM_Command** (mode, host_len1, host_buf1, PJM_Bytes, PJM_Bytes_len) combines the coding and decoding technique for MODE2, as explained in previous two chapters. The command bytes are stored together with CRC bytes in a 8 bit *host_buf1* array. The MODE2 tag

replies are 16 bit values that are stored in *PJM_Bytes* array. The *PJM_Bytes_len* store the number of reply bytes. With *mode* value, the user can choose how the command flag start (high or low).

**Check_PJM_CRC32** (PJM_Bytes, PJM_Bytes_len) checks the reply CRC from the end of flag.

# 7. Conclusion

The aim of this diploma thesis was to design a digital part of a low cost reader that would support MODE1 and MODE2 of the ISO/IEC 18000-3 standard. The analog front end of the reader was part of another diploma thesis, as already mentioned in 1.

In order to understand contactless RFID systems, the theoretical research was done first. At the beginning, the basics of inductively coupled systems were investigated along with their corresponding coding techniques. In addition, research was done on ARM based microcontrollers, focusing on the STR71 family first and the STM32 family second. STMLibrary was consulted for both types of microcontrollers.

After the theoretical research, the ISO/IEC 18000-3 standard was investigated. The system models were designed based on the communications signal interfaces defined by this standard. The models were programmed in program language C using Eclipse development environment.

Having selected the most promising models for both MODEs and both directions (reader to tag and tag to reader), the models were applied to program a microcontroller STRx71. For this, the STMLibrary ([17]) was used. The tests of reader to tag communications for both MODEs gave desired results. The card reader communication was tested using a generated tag replay signal. After these tests did not give satisfying results, the decision was made to switch to another type of micrrocontroler, the STM32 Cortex M3.

Since the reader to tag direction provide good results in the first tests, the same principle was used for the new microcontroler. This time, the tag to reader direction also gave wanted results. Again the STMLibrary([16]) was used for programming of the microcontroller.

The next step was to run an overall system test. The analog front-end was tested together with the microcontroller. The I/Q and the MODE decision had to be made by the software. The reader tag communication indeed gave very good results. The tag to reader direction gave, also, a satisfying results. The robustness of the reader for this type of communication could be improved. Also, the MODE 2 communication should be expanded for all eight channels as

defined by the standard. The next step could be an implementation of MODE3, so that the whole ISO/IEC 18000-3 standard could be covered by this reader. Since MODE3 also used PJM and ASK modulation, this could probably be done using the same concepts, with only a few changes.

# A. Cyclic redundancy check MODE1

CRC is technique for detecting errors in digital data transmission. It is a very reliable method and it can even be used for large numbers of data. CRC recognizes errors in transmission but it can not correct them. For ISO/IEC 18000-3 MODE1 a 16 bit CRC mode is used.

## A.1 16 bit CRC detection

The CRC is being calculated considering all bytes after the SOF up to EOF. They are appended to each request and each response before the EOF. The CRC is transmitted on the principle LSB first. The next table shows the CRC definition according to ISO/IEC 15693.

| CRC type | Length | Polynomial | Direction | Preset | Residue |
|---|---|---|---|---|---|
| ISO/IEC 13239 | 16 bits | $x^{16}$+$x^{12}$+$x^5$+1=‘8408’ | Backward | ‘FFFF’ | ‘F0B8’ |

**Table A.1:** CRC 16 definition (adapted from [6])

In order to protect data transmission from shifting errors, the attached CRC is the complement of the calculated CRC. When checking received response, two CRC bytes are also included in the re-calculation while the residue then equals ‘F0B8’.

## A.2 16 bit CRC calculation example

The calculation of a 16 CRC begins with a preset value ‘FFFF’. The XOR operation is done for the preset value and the first data byte. If the last bit of the compute value is 1 (logic AND with 1), the value would be shift right and the XOR operation would be done with generator polynomial (see A.1). If the last bit is not 1, the value would just be shift right. This is done 8 times. Then comes the next data byte and the same procedure is repeated. After all bytes are done, the compute value is inverted CRC. If we want to check the transmitted CRC value, then the CRC bytes also have to be included in the CRC calculation, if CRC is correct the checked CRC value is going to be equal to residue ‘F0B8’.

The next table shows a CRC calculation and a inventory request CRC check. The request consists of four parts: the flag ‘06’, the command code ‘01’, the mask length ‘00’ and CRC ‘CD09’. Flag ‘06’ means that we are using a single subcarrier, a high data rate and 16 slots. The mask length ‘00’ means we are not using any mask. First, the SOF is transmitted, then the request and after that the EOF.

| **Step** | **Reqest bytes** | **Calculate CRC (VCD)** | **Check CRC (VICC)** |
|---|---|---|---|
| 1 | Initialised | ‘FFFF’ | ‘FFFF’ |
| 2 | ‘06’ | ‘6AB1’ | ‘6AB1’ |
| 3 | ‘01’ | ‘B5E1’ | ‘B5E1’ |
| 4 | ‘00’ | $\sim$ ‘F632’ = **‘09CD’** | ‘F632’ |
| 5 | ‘CD’ | | ‘F8E’ |
| 6 | ‘09’ | | **‘F0B8’** |

**Table A.2:** Example of CRC16 calculation

# B. Cyclic redundancy check MODE2

In MODE2, the commands use a 16 bit CRC, the same one that is used by the ISO/IEC 15693 (see Appendix A). Replies from tag to interrogator use a 32 bit CRC.

## B.1 32 bit CRC detection

The CRC is calculated considering all bytes after the flag. They are appended to each reply. The LSB is transmitted first. The next table shows the CRC definition according to ISO/IEC 18000-3.

| Length | Polynomial | Direction | Preset | Residue |
|---|---|---|---|---|
| 32 bits | $x^{32}$+$x^{26}$+$x^{23}$+$x^{22}$+$x^{16}$+ $x^{12}$+$x^{11}$+$x^{10}$+$x^{8}$+$x^{7}$+$x^{5}$+ $x^{4}$+$x^{2}$+$x^{1}$+1 =‘EDB88320’ | Backward | ‘FFFFFFFF’ | ‘2144DF1C’ |

**Table B.1:** CRC 32 definition (adapted from [8])

Also, here is the ones complement of the calculated CRC the value attached to the message. For checking of reply, the 2 CRC bytes are also included in the re-calculation and residue equals ‘2144DF1C’.

## B.2 32 bit CRC calculation example

The calculation of a 32 CRC is similar to the one of the 16 bit CRC. It also begins with a preset value, which this time equals ‘FFFFFFFF’. The XOR operation is done for the preset value and the first data byte. If the last bit of the compute value is 1 (logic AND with 1) the value would be shift right and then the XOR operation would be done with generator polynomial (see B.1). If the last bit is not 1, the value would just be shift right. This is done 16 times. Then comes the next data byte and the same procedure is repeated. After all bytes are done, the compute value is CRC. If we want to check the transmitted CRC value, then the CRC bytes also have to be

included in CRC calculation. If no error has occurred during the transmission the checked CRC value is going to be equal to residue ‘2144DF1C’.

The next table shows a CRC check example for tag reply. First comes the time stamp ‘1234’, after that a specific identifier ‘00030002’, then the contents of the requested memory locations ‘0010’ and ‘0011’ and at the end, two CRC bytes ‘219F’ and ‘C7D5’.

| **Step** | **Reply bytes** | **Checked CRC (Interrogator)** |
|---|---|---|
| 1 | Initialised | ‘FFFFFFFF’ |
| 2 | ‘1234’ | ‘094A9040’ |
| 3 | ‘0002’ | ‘7399A576’ |
| 4 | ‘0003’ | ‘B52DBBB2’ |
| 5 | ‘0010’ | ‘815B13BD’ |
| 6 | ‘0011’ | **‘C7D5219F’** |
| 7 | **‘219F’** | ‘41D9D52A’ |
| 8 | **‘C7D5’** | **‘2144DF1C’** |

**Table B.2:** Example of CRC32 calculation (adapted from [8])

# Bibliography


[1] P. Hawrylak / M. Mickle / J. Cain. *The Internet of Things - From RFID to the Next-Generation Pervasive Networked Systems*. Auerbach Publications, Taylor & Francis Group, 2008. Chapter 1.

[2] K. Finkenzeller. *RFID Handbook*. John Wiley & Sons, 2003. Second Edition.

[3] R. Mäusl / J. Göbel. *Analoge und digitale Modulationsverfahren*. Hüthig, 2002.

[4] M. Gebhart. RFID Systems (Lecture notes), 2009. Graz University of Technology, Institute for Communication Networks and Satellite Communications.

[5] H. Bhatt / B. Glover. *RFID Essentials*. O'Reilly, January 2006.

[6] ISO/IEC. ISO/IEC 15693, Vicinity cards. Technical Report Part 3: Anticollision and transmission protocol, Committee identification: ISO/IEC JTC1/SC17/WG8, 2000-04-01. Identification cards: Contactless integrated circuit(s) cards - VicinityCards.

[7] ISO/IEC. ISO/IEC 15693, Vicinity cards. Technical Report Part 2: Air Interface and initialization, Committee identification: ISO/IEC JTC1/SC17/WG8, 2000-05-01. Identification cards: Contactless integrated circuit(s) cards - VicinityCards.

[8] ISO/IEC. ISO/IEC 18000. Technical Report Part 3: Parameters for air interface communication at 13,56 MHz-, Committee identification: ISO/IEC JTC1/SC17/WG8, 2004. Information technology- Radiofrequency identification for item management-.

[9] D. Kusternigg. Contactless Demonstration Reader Decoder Optimization for Receiving Very High Bit Rates. Master's thesis, Graz University of Technology, Institute for Electronics, 2009.

[10] T. Martin. *The insider´s guide to the ARM STR71x*. Hitex (UK) Ltd., 2006.

[11] T. Martin. *The insider´s guide to the STM32*. Hitex (UK) Ltd., 2008.

[12] J. Proakis. *Digital Communciations*. McGraw-Hill, 4th edition, 2000.

[13] V. Hunt / A.Puglia / M. Puglia. *RFID - A Guide to Radio Frequency Identification*. John Wiley & Sons, 2007.

[14] K. Kupfmüller / W. Mathis / A. Reibiger. *Theoretische Elektrotechnik*. Springer Verlag, 2006. 17. Auflage.

[15] B. Sklar. *Digital Communciations - Fundamentals and Applications*. Prentice Hall, 2nd edition, 2002.

[16] STMicroelectronics. *ARM®-based 32-bit MCU STM32F101xx and STM32F103xx firmware library*, 2007.

[17] STMicroelectronics. *STR71x firmware library*. STMicroelectronics, 2007.

[18] STMicroelectronics. *Reference manual STM32F101xx and STM32F103xx advanced ARM-based 32-bit MCUs*, 2008.

[19] STMicroelectronics. *STR71xF microcontroller family Reference manual*. STMicroelectronics, 2008.

[20] Magellan Technology. Phase Jitter Modulation (PJM) used by ISO/IEC 18000-3 Mode 2. Technical report, Magellan Technology Pty Ltd, 2007.

[21] S. Wilson. *Digital Modulation and Coding*. Prentice Hall, 1996.

[22] F. Xiong. *Digital Modulation Techniques*. Artech House, 2nd edition, 2006.