

**Intelligent Name Card  
with RFID technology**

**By**

**Jan Östling**

**Royal Institute of Technology, Stockholm**

**Master's Thesis in Electrical Engineering**

**Stockholm, 2007-11-11**

**Examiner: Professor Li-Rong Zheng**

## **Abstract**

The purpose of this master thesis was to develop an intelligent name card with use of RFID technology, wireless transmission of information.

While the master thesis I studied different RFID standards to find an appropriate standard. I chose the ISO 15693 standard for it's simplicity and it's cheap, unfortunately it not have the highest memory capacity of the different standards.

Next part of the thesis was to program the graphical user interface, and together with a RFID reader be able to read and write to the transponder. Unfortunately I have not got the serial interface (RS-232) to work, instead I have simulate the Reader with the command shell (DOS-window).

When develop the GUI I chose to use java because to make the program platform independent.

The last part was to propose the hardware, it basically consist of three parts: a micro controller, a transceiver chip and an antenna. The microcontroller is used for interface and buffer between the host computer and the transceiver chip. The transceiver chip converts the data signals to radio signals and controls the transmission. The antenna can be printed on the circuit board.

The RFID tags on the market today have not enough memory to store pictures.

## Sammanfattning

Avsikten med detta examensarbete var att utveckla ett intelligent visitkort med användande av RFID teknologi, trådlös överföring av information.

Under examensarbetet har jag studerat olika RFID standarder för hitta en lämplig standard. Jag har valt att använda ISO 15693 för att den är enkel och billig, dock har den ej den högsta minneskapaciten av dom på marknaden förekommande standarderna.

Nästa del av examensarbetet har varit att programmera ett grafiskt användar gränssnitt, och tillsammans med en färdig RFID läsare från Texas instruments, kunna läsa och skriva RFID tranpondern. Dock har jag inte lyckats att få serieporten att fungera, utan jag har fått simulera RFID läsaren genom att DOS fönstret.

Vid programmeringen har jag valt att använda java för att göra programmet plattformsoberoende.

Sista delen var att ge förslag för hårdvaran, den består i princip av tre delar: a microcontroller, ett transceiver chip och en antenn. Microcontrollern används som interface och buffert mellan huvuddatorn och transceiver chipet. Transceiver chipet konverterar datasignalerna till radiosignaler. Antennen kan tryckas direkt på mönsterkortet.

Dagens RFID-tags har för liten minneskapacitet för att kunna lagra bilder.

## Table of content

Chapter		page
1	Objective	5
2	Technology overview	5
3	Choice of system	6
4	Description of the standard	7
5	Vcard	14
6	Hardware	15
7	Software	15
8	Communication between host computer and transponder	28
9	Layout	33
10	Conclusion and future work	36
11	References	37
Appendix A	Example of testing the software with simulation of the TI-reader module	

# 1 objective

The work topic is to develop “Intelligent Name Card”. This is to demonstrate how to use RFID technology for future production of intelligent paper such as a smart name card.

- Overview of RFID technology
- Summary of challenges for RFID in smart label and paper applications
- Demonstrate an intelligent name card with a RFID development board (need to program the tag, develop some user interfaces)
- Propose/design to miniaturize the card reader (so that it could be plug in a PDA or a laptop/mobile phone).

# 2 Technology overview

It exist several types of RFID systems. RFID means radio frequency identification, wireless transfer of information. It uses a magnetic or a electromagnetic field to transfer information. The systems can be divide in passive or active, with passive means that the transponder uses the transmitted field to provide the electronic. Active transponder has it's own battery. The transponders can also be divided into range where they are achievabled from a few millimeters to over 15 meters. Another distinction is writable or readable, some transponders gets a unique serialnumber when it's fabricated. Another difference its how much information to stored in the memory from 1 byte up 50 kbytes. Another difference is memory card or a processor card, processor card has it's own processor and with that better security.

The different transmission frequencies are classified into three different ranges, LF (30-300 Khz), HF/RF (3-30 Mhz) and UHF (300 Mhz-3 Ghz). Figure1 show the RFID system

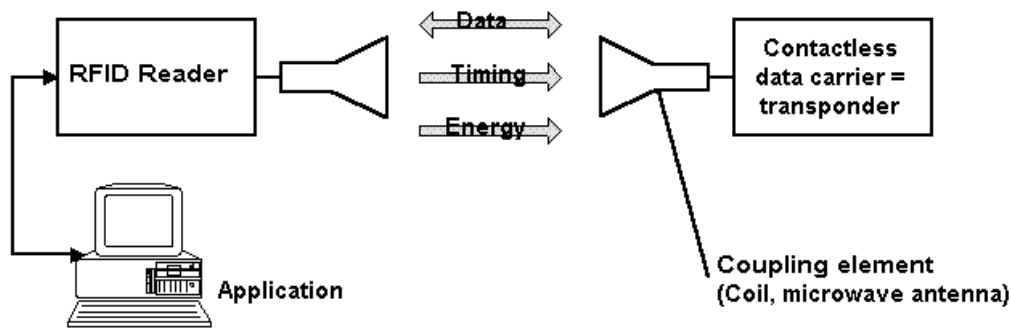


Figure 1. system overview [4]

## 2.1 Transponder Construction Formats

One format is disk (coin) format, a round transponder with diameter from a few millimeters up 10 cm. The transponder are often in a molded housing.

Glass transponder have been developed so it can be injected under the skin of an animal for identification purpose. The glass tubes is just 12-30 mm long and 2-3 mm in diameter.

The term smart label refers to a paper-thin transponder format. In transponders of this format coil is applied to a plastic foil of just 0.1 mm thickness by screen printing or etching. This foil is often laminated with paper and coated with adhesive, and its used to be stucked to goods of all types. Or the paper-thin transponder can be laminated between PVC foils and in the size of ID-1 format (size of credit cards and telephone cards, 85 mm x 54 mm x 0.76 mm). Figure 2 show a picture of transponders.



Figure 2 Example of some transponders [4]

### **3 choice of system**

The different systems available on the market have sufficient memory capacity, most of them only a few bytes. Three different standards have better memory capacity ISO 15693, ISO 14443 A and ISO 14443 B. The different is that ISO 14443 A and B are processor cards and ISO 15693 are a memory card and thus simpler and cheaper

ISO 15693 has a highest memory capacity of 8 Kbytes, but no manufacturer have no capacity that high. Instead ISO 14443 often have a capacity of 10 - 20 kbytes and 2 kbit for ISO 15693. All three have a frequency of 15.93 MHz.

I have chose ISO 15693 because of it's simplicity and it was easier to obtain information about the standardization despite the memory capacity, the memory in ISO 14443 has not enough memory to store pictures.

### **4 Description of the system**

#### **power supply**

The RFID system is inductive conected system, i.e. It uses a spool one large surface as antenna. It's passive system that means that the energy for the power supply for the transponder is emitted from the reader. A small piece of the emitted field penetrates the antenna coil. A voltage is generated in the antennacoil thru inductans. That voltage is rectified and supplys the transponder with energy. Figure 3 shows how the transmission of power works.

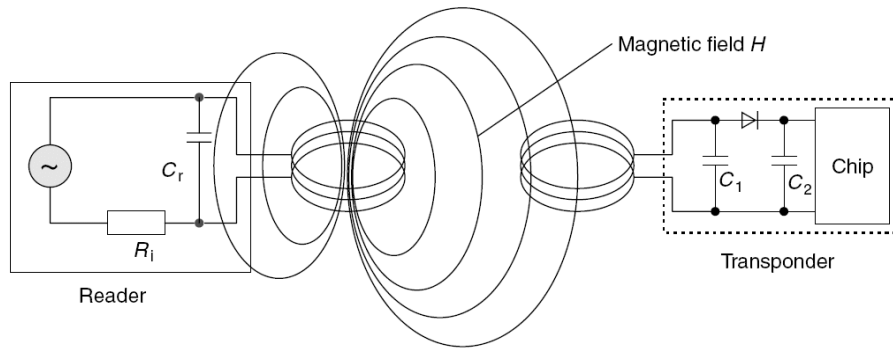


Figure 3. Energy transmission [4]

The Antenna coil on transponder and the condensator C1 is a resonant circuit and it's tuned to the transmission frequency on the reader.

### Data transfer transponder ->reader

If a transponder in resonance is placed in the magnetic field from the reader antenna the transponder draw energy from the magneticfield [4]. The resulted feedback on the reader antenna can be represented as a transformed impedance on the reader. If switching a load resistor on and of, the impedance change and with that the voltage on the reader antenna. If the additional load resistor in the transponder is switched on and off at a very high frequency  $f_s$  then two spectral lines are created at a distance of  $\pm f_s$  around the transmission frequency of the reader  $f_{reader}$  and these can easily be detected. Load modulation with sub carrier creates two modulation sidebands at the readers antenna at the distance of the sub carrier frequency around the operating frequency  $f_{reader}$ . These modulation sidebands can be separated from the significantly stronger signal of the reader by bandpass filtering on one of two frequencies  $f_{reader} \pm f_s$ . Once it has been amplified, the sub carrier signal is now very simple to demodulate. Figure 4 shows the data transmission.



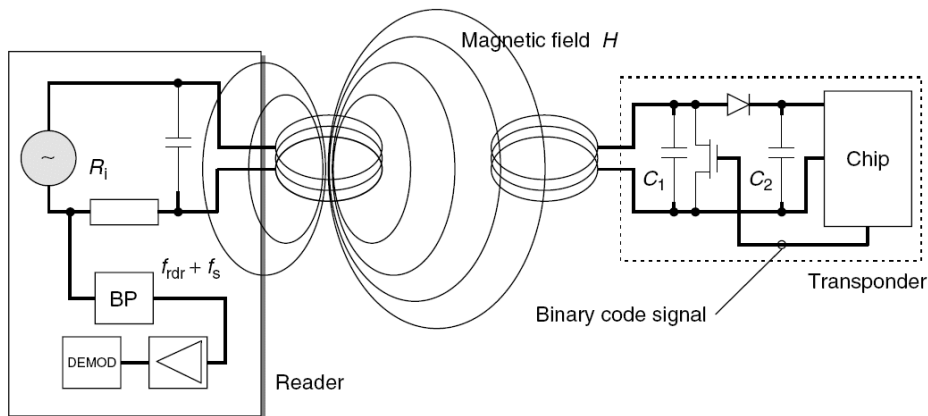


Figure 4 data transmission transponder -> reader [4]

### Data transfer reader->transponder

Datatransfer between reader and the transponder is ASK and it's easy to detect.

The power that needs is  $115 \text{ mA/m} < H < 7.5 \text{ mA/m}$ . Both 10% ASK and 100% ASK-modulation is used for data transfer between the transceiver and transponder. Independent of which modulation index is used, two different codes are used, 1 of 4 and 1 of 256. The frequency as used is 15.56 MHz, two different data rates, the range is up to two meters, 256 different blocks with four bytes in each block. The various operations to do with the transponder are read and write one or several blocks, lock different blocks from writing. It's also possible to store information in which branch the transponder is in, shops, tickets and more. To secure the data transfer a block of 16 bits CRC is used.

### Radio interface

#### Code reader to transponder

1 of 255 PPM -pulse position method [5], the value of the digit to be sent is unambiguously determined in the area 0 – 255 dependent of the time position of the pulse, viewed in Figure 5. 8 bits (one byte) sends in one moment. One byte with a

transmission time of 4.833 ms is divided in 512 timeslots, each timeslot is then 9.44  $\mu$ s it gives a transmission rate at 1.65 kbit/s. Each transmission starts with a start of frame SOF, it consist of two modulation pulses on each 9.44  $\mu$ s separated with a time slot on 56.65  $\mu$ s ( $6 \times 9.44$ ) this is showed in Figure 6.

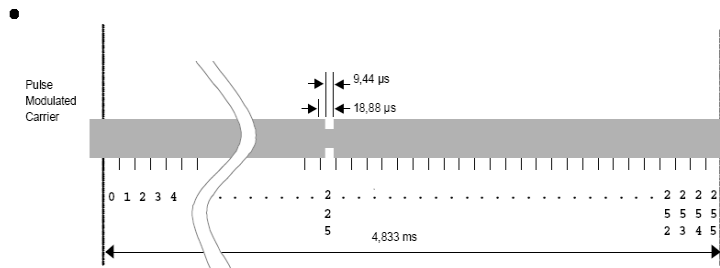


Figure 5. 1 out of 256 coding mode [5].

The transmission is ended with a end of frame EOF, it consist of one singel modulation pulse on 9.44  $\mu$ s, it's on an even time slot to be separated from the code viewed in Figure 7.

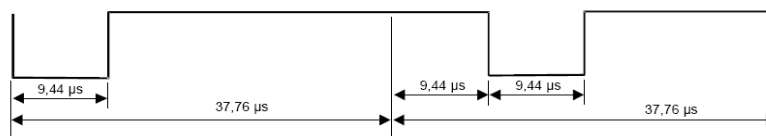


Figure 6. start of frame in 1 out of 256 [5].

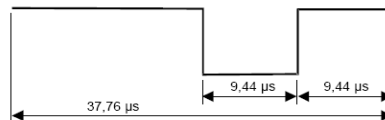


Figure 7. end of frame for booth methods [5].

### 1 of 4 code

The another code who are used id 1 of 4 [5], even in this code the value is determined on the position of the modulation pulse, the value of the digit that's transmitted is between 0 – 4. In this case a time slot is 9.44  $\mu$ s and two bits takes 75,52  $\mu$ s to be transfered, the transmission rate in this case is 26.48 kbit/s, se Figure 8 for the 4 values. The start of frame is viewed in Figure 9.

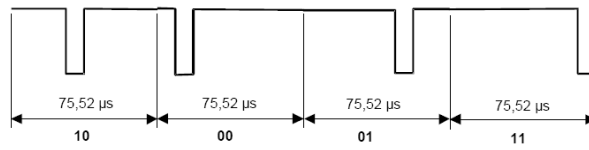


Figure 8. 1 out 4 coding example [5].

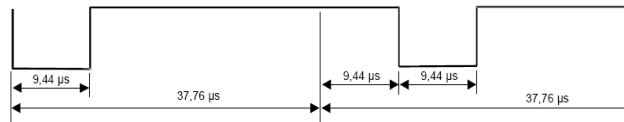


Figure 9. Start of frame 1 out of 4 code [5].

End of file is similar as in 1 of 256 coding, Figure 7.

### Data transmission transponder to reader

Load modulation with a modulated carrier is used for transmission between the transponder and reader [5]. The ohmic or capacitive modulation resistor is turned on and off with the subcarrier frequency, the subcarrier itself is modulated with manchester coding either with ASK or FSK, long distance ASK => 6.62 Kbit/s and FSK => 6.62/6.82 Kbit/s. Fast mode ASK = 26,48 kbit/s. Figure 10 shows a logic 1 and logic 0 when one subcarrier is used and Figure 11 shows a logic 0 when two subcarriers are used. Figure 12 shows a start of frame when a single subcarrier is used and end of frame is show in Figure 13 when two subcarriers is used.

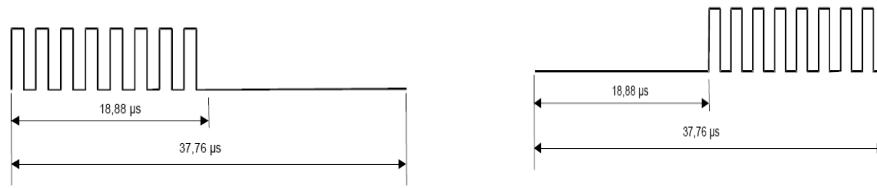


Figure 10. logic 1 and 0 when using one subcarrier [5].

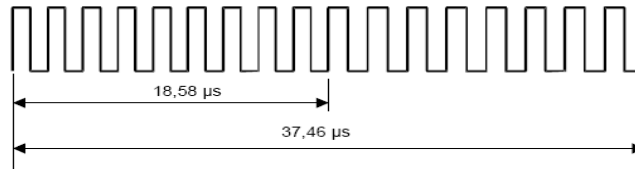


Figure 11. logic 0 when using two subcarriers [5].

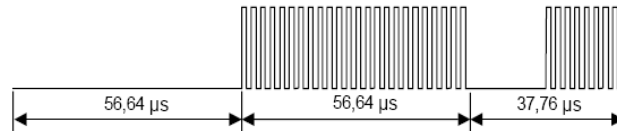


Figure 12. start of frame when using one subcarriers [5].



Figure 13. End of frame when using two subcarriers [5].

The memory of the ISO 15693 is maximal 256 block x 256 bit => 8kbytes, but the transponder from Texas instrument only have a size of 2048 bits.

The structure gives it possibility to increase the memory.

The protocol consist first a request from the reader and then a response from the transponder [6][9]. Each request from the reader consist of following fields flag, command, code, mandatory and/or option parameter field, application data fields and CRC

Each response consist of flag, command code, Mandatory and/or optional parameter field, Application data fields and CRC [6][9].

Each transponder has its own serial number of 64 bits. It can be used to address each transponder uniquely and individually. It also uses when several transponders are in the range of the reader to do an anticollision-loop.

AFI Application Family Identifier.

In this field it is possible to put information in which branch where they are used e.g. transportation, finance, medicine, gaming.

Data check

The transmitted data is checked by a 16-bit cyclic redundancy check (CRC). Both when transmitting from transponder to reader and reader to transponder using CRC.

Request Flag

Request flag is 8 bits long and contains for example information about which transfer method and which data rate.

Response Flag

Contains only an error message if an error has occurred, what kind of error is present in the error field.

Command code

The only mandatory command for the standard is inventory and stay quiet. But a lot of manufacturers use the optional standard and it is also possible to use some own commands and it is also possible for new commands in the future. The used command here is inventory, stay quiet, read multiple blocks and write multiple blocks.

Write multiple blocks starts with a

- SOF

then

- flag 8 bits
- code for multiple blocks 24 in hex
- possible 64-bit UID
- the place where the first place of data should be placed in the memory

- number of blocks to write
- the data, 16 bitars CRC
- EOF

Read multiple blocks

- SOF,
- Flag 8 bitar,
- possible UID,
- first block to read,
- number of blocks,
- 16 bits CRC and
- EOF

The transponder reponse if ok with:

- SOF,
- Flag,
- data,
- CRC
- EOF

## **Command for the TI-reader module**

The command for the for TI-reader contain some extra header code and some code removed[10]. The SOF, EOF and 16-bit CRC are removed, the CRC is calculated by reader module, instead a XOR-checksum must be calculated. Each field and the description of the field is is shown in figure 14. For example in The Command field 60<sub>hex</sub> Set the reader to the ISO-15693 format.

### Request Packet Format

Standard reader Request Packet Format (See Section 3.1)										
Header	Packet Length		Node Address		Command Flag	Command	ISO Command Data		Checksum	
							Config. Byte	Data		
'01 <sub>hex</sub> '	LSB	MSB	LSB	MSB	Flags	'60 <sub>hex</sub> '	XX <sub>hex</sub>	Data	Byte 1	Byte 2
1 byte	2 bytes		2 bytes		1 byte	1 byte	Byte 0	bytes 1 - n	2 bytes	
							1 byte	n bytes		

### Request Packet Description

Field	Length	Description
Header	1 byte	Defines the start of the packet (01 <sub>hex</sub> ).
Packet Length	2 bytes	Defines the length of the packet, including checksum.
Node Address	2 bytes	Defines the Node address of the reader.
Command Flags	1 byte	Defines how a command will be executed.
Command	1 byte	Defines the command for the reader to execute (60 <sub>hex</sub> for ISO 15693-3 commands)
Data	0 - n bytes	Defines the data required by the reader for a command.
Checksum	2 bytes	Byte 1 is an XOR checksum of all elements from the header to the last byte

Fig 14 Request packet for the TI-reader

Header	Packet Length		Node Address		Response Flags	Command	ISO Response Data	Checksum	
							Data bytes 0 - 'm'		
'01 <sub>hex</sub> '	LSB	MSB	LSB	MSB	Flags	'60 <sub>hex</sub> '		Byte 1	Byte 2
1 byte	2 bytes		2 bytes		1 byte	1 byte	'm' bytes	2 bytes	

Fig 15 Respond packet from the reader

The Respond of the packet is shown in fig 15.

### Read the transponder

It's not possible to read all the 64 blocks at the same time, thus it's done twice, first block 0 to 31 and then the block 32-63. To read the first 32 blocks the following code should be send:

01 14 00 00 00 00 96 17 03 35 00 31 65 190

Where the byte 10=35 is read multiple block. Byte 11=00 is the start block and byte 11=31 is number of blocks that shall be read.

The respond either the data from the transponder or some error message.

If an error occur the ISO Response data field contains of one byte with the errorcode. Error code is shown in fig 16.

<b>01<sub>hex</sub></b>	<b>Transponder not found</b>
<b>02<sub>hex</sub></b>	<b>Command not supported</b>
<b>04<sub>hex</sub></b>	<b>Invalid flags</b>

Fig 16 Errormessage

If the transmission is correct the ISO Response data field contains first a flag byte then transponder data. The first 32 blocks and each contain 4 bytes.

### **Write to the transponder**

The Texas Instrument reader module can only write a single block. For example write block 0 with the value 01 00 00 00.

01 17 00 00 00 00 96 17 67 33 00 01 00 00 00 02 253

Where the byte 10=33 is write single block. Byte 11=00 is the blocknumber and byte 11-14=01 00 00 00 is the data to the transponder. If the operation is correct the ISO response data field contain the flag 00. The error is according figX

## **5 Vcard**

Vcard is automate the exchange of personal information there is on traditional businesscards. Vcard are used in application as webbrowsers and here smartcards. Vcard contain information as in ordinary businesscards as name, adresse, company, phone, logotype even a picture can be added. Vcard support for another information as sound, for example to pronounce the name of the card owner correct.

Vcard contain a number of predefined types [3], vcard must start with the type Begin:VCARD and it must be ended with the type End:VCARD. Identification types is used for identification and separation the differnt parts in the vcardformat. A property is the definition of a individually attribute Propertyname [ ';' propertyparameters ] ':' propertyvalue. Propertyname must be the first text on a line. Propertyparameter is delimited from propertyvalue with a semicolon, propertparameters and propertyvalue is separated with a colon. The vcard use only ASCII characters.



TEL;Home:+1-23489-1234.

An example for vcard.

```
BEGIN:VCARD
```

```
VERSION:2.1
```

```
FN: Stefan Andersson
```

```
N:Andersson ; Stefan ; Bengt; Bengan; Engineer
```

```
ADR: ;; Mainstreet 55 ; The City ;; Zip 44 123;Sweden
```

```
END:VCARD
```

## 6 Hardware

Very simple hardware only three major components RFID transceiver chip, interface chip and the antenna. For the transceiver chip i chosen Texas Instruments transceiver chip. It's cheap chip, both for 14443 and 15693 standards. For the interface i chose an AVR 8015 microcontroller, it's a cheap microcontroller with an UART interface. The antenna is printed on the card and tuned into frequency 15.56 MHz.

## 7 Software

### Graphical user interface GUI

The goal was to do the electronic business card gui based. The software is developed in java to be platform independent. Figure 17 shows the graphical user interface.

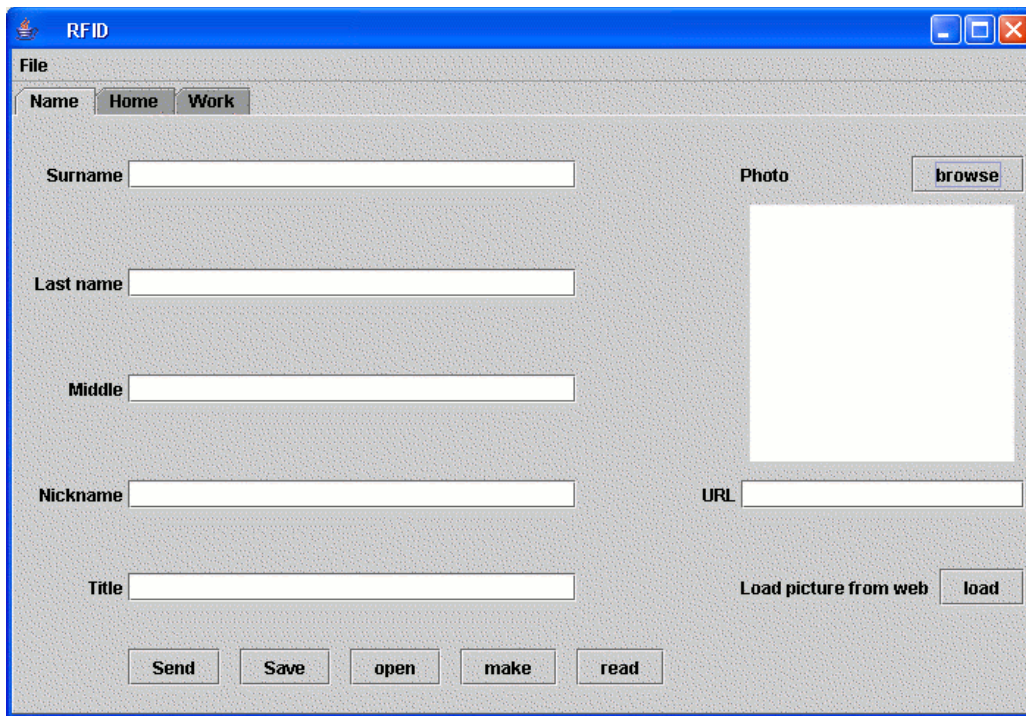


Figure 17. the grafical user interface

The different pages have information about the person, personal information and business information. Even photograph and logotype, but not with this transponder because the insufficient memory capacity only 8 kbytes, but the standard gives it possibility for more memory in the future. The software is made to demonstrate with photo and logotype.

### Buttons

#### Read

The button read is for read the RFID transponder. The information is sorted and placed in the window.

#### Save

The button save store the information from the window on the harddrive. The information is stored in Vcardformat so it can be sent away and read by other Vcard readers.

#### Open

The open button, with this button you open a Vcard card from the harddrive and information is placed in the window.

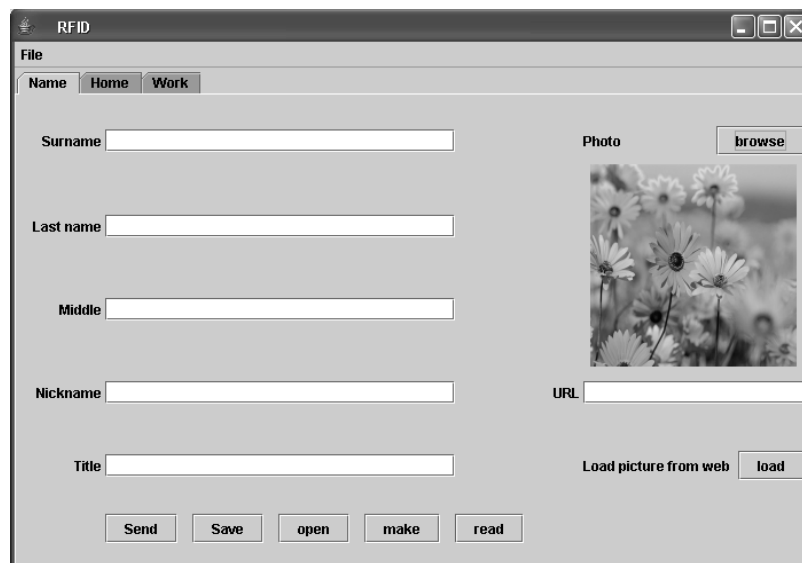
Make

With the make button code are added to be in a Vcard format.

Send

With this button the data is sent to the reader to be stored in the transponder.

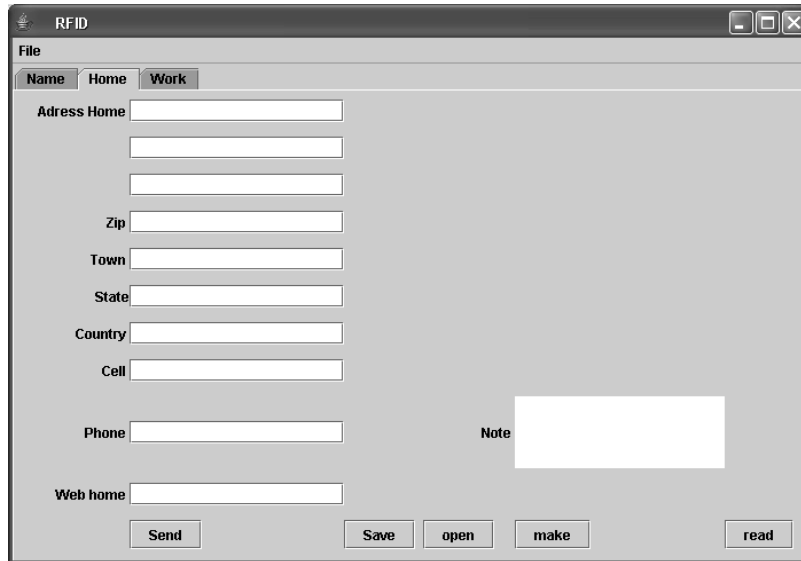
### The window



The screenshot shows a window titled "RFID" with a standard Windows-style title bar. Below the title bar is a menu bar with "File" selected. Under "File", there are three sub-menus: "Name", "Home", and "Work". The main area of the window contains a form with several text input fields and buttons. On the left side, there are five text input fields labeled "Surname", "Last name", "Middle", "Nickname", and "Title". On the right side, there is a "Photo" section with a "browse" button and a small image of daisies. Below the photo is a "URL" text input field and a "Load picture from web" button with a "load" sub-button. At the bottom of the window, there is a row of five buttons: "Send", "Save", "open", "make", and "read".

Figure 18. the first page

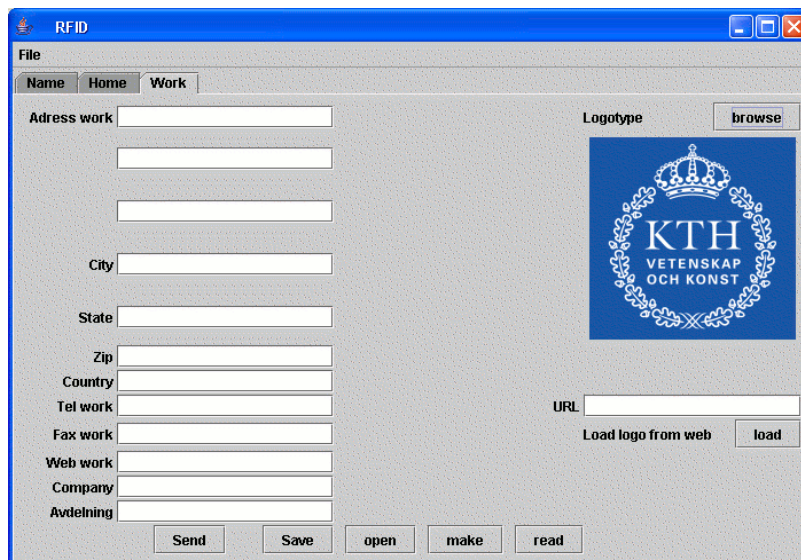
On the first page Fig 18 there is personal information name, nickname, title and it's possibility add a picture or download a picture from internet.



The screenshot shows a window titled 'RFID' with a 'File' menu. The 'Home' tab is selected. The form includes the following fields: 'Address Home' (three stacked text boxes), 'Zip', 'Town', 'State', 'Country', 'Cell', 'Phone', and 'Web home'. A 'Note' text area is located to the right of the 'Phone' field. At the bottom, there are five buttons: 'Send', 'Save', 'open', 'make', and 'read'.

Figure 19. the second page

On the second page Figure 19 there is space for address, phone number, webaddress and a place to put some notes in.



The screenshot shows the same 'RFID' window with the 'Work' tab selected. The form includes the following fields: 'Address work' (three stacked text boxes), 'City', 'State', 'Zip', 'Country', 'Tel work', 'Fax work', 'Web work', 'Company', and 'Avdelning'. On the right side, there is a 'Logotype' section with a 'browse' button and a 'Load logo from web' section with a 'load' button. At the bottom, there are five buttons: 'Send', 'Save', 'open', 'make', and 'read'.

Figure 20. the third page

Figure 20 shows the third page, this is the business page there is space for address to work, phone to work, and place for a logotype, from the transponder or from a webaddress on the transponder.

## Software structure

The software consist of three parts, one part for drawing the window on the screen. One part for events, for example if a button is pressed or when text is typed. And the last part for manage information, send and receive information on the serialport. The software are made with the help of the books [2] and [7].

To create the graphical interface java graphical standard classes javax.swing and java.awt. To create freestanding windows i used the class JFrame, it's creates a freestanding window with margins and titleline. My own class TPE is a subclass to JFrame. To TPE a JPanel toppanel is added, the class JPanel is used to create complicated graphical interfaces, for example when a window contain several small window. Then an object of the class JPanel is added to the class, it's a child component to the window. In JPanel it is possible to put other graphical components.

```
Jpanel toppanel= new JPanel
```

Then what kind of strategy for placing the components. I chose BorderLayout, it's added to toppanel with.

```
Toppanel.setLayout( new BorderLayout()
```

I the main window is four cards with flaps added, it's possible to chose window when clicking on some flap. The window is created with the constructor.

```
tabbedpane= new Jtabbedpane
```

To add a card to a JtabbedPane component the method addTab is used.

It's four methods *tabbedPane.addTab(panel)* panel = panel1-4

Panel mention wich component on each card. The components in TabbedPane is created by call of the methods createpage1-createpage4.

TabbedPane is added to topPanel with.

*TopPanel.add( tabbedPane, BorderLayout.CENTER )*

Where BorderLayout.CENTER is the type of layoutmanager for placing the components. BorderLayout.CENTER means it's placed in the middle of the window. Then it's decided what kind of layout manager for panels.

*Panel.setLayout(gridBag)* panel=panel1-panel4

Where gridBag is the Layout manager as used. How the four different panels i tabbedPane look like is decided in the methods createpage1 - createpage4.

### **Method Createpage1**

The different components is added one by one, you must also mention what kind of property a component should have. The property is for example where the components are placed, how large it's, which color. To mention the property of a component a particular class called GridBagConstraints is used. The methods buildConstraints creates [1] a new object of the class GridBagConstraints with several property.

*BuildConstraints(Constraints,a,b,c,d,e,f)*

- a mention in which row it starts
- b mention in which column it starts
- c mention how wide it is
- d mention it's height
- e and f is mention how extra space should be divide between the panes.

When it's decided how the panes look like something should be added. To show a textfield, a JLabel is created to show text on the screen and the orientation. Then the given property should be connect with the component namnLabel, it's done with.

```
Gridbag.setCoinstraints(namLabel,Constraints)
```

After that the component is added to the window.

```
Panel1.add(namLabel)
```

Next to the textfield it's a field for text input and output, it's done as earlier with buildconstraints, for it to be placed beside the textfield a=1 instead of a=0 i.e the collomn beside. Then

```
Consrains.fill = GridbagConstraints.Horizontal
```

How it should fill the pane. Then the property mentioned should be connect to a component, in this case a component for input and output of text, a Jtextfield. This component is defined earlier, because it's in used in several places in the program. It's defined earlier as:

```
JtextField krummelurer= new TextField(15)  
gridbag.setConsrains(krummelurer,Constraints)
```

Then the whole page is created in the same way.

Then the other pages is created in the same way.

On each page there is buttons for example send and save pages. They also are created in the same way

```
buildConstraints(constraints,0,6,1,1,40,100); constraints.fill=  
GridBagConstraints.NONE; constraints.anchor=GridBagConstraints.WEST  
gridbag.setConstraints(send1,constraints); panel1.add(send1)
```

Where the button `send1` is defined before because it's in use in several places in the program. To make something happening when the button is pressed, there is a listener for every class. They are connected to happening:

```
send1.addActionListener(new HandHant());
```

## **Showing pictures**

To show pictures or icons the class `ImageIcon` is used. A picture of the `ImageIcon` class internal contain an object of the class `Image`, which describes an image. The class `Image` can handle the image format GIF, JPEG and PNG. There is also possible to show motion pictures.

First the picture is downloaded with.

```
Image bilden = toolkit.getDefaultToolkit().getImage(u)
```

Then a `JLabel` is created with an `ImageIcon` object.

```
JLabel l = new JLabel ( new ImageIcon(bilden))
```

The output on the screen is done with:

```
drawImage()
```

To download and show a picture from an URL it's the same principal. Just add the package `java.net` and create an `URL` object from an web address. This constructor is used.

```
URL u = new URL()
```



## Events

In java different events is describe with eventclasses. When an event occur an object is created of that class that describe the event. After that a call to the method in that listener that is defined to listened to that type of events. As parameter to the method in the listener is the created event object. To capture an event should an listener be created. This listenerinterface have the same name as the listenerclass, but with listener instead of event, in this case ActionListener. A listener must be registred, that's done with the method add.listener for the component that the listener should be added to. The different events that's occur is save, open, make, send and browsephoto and browselogo. These are implemented in the class Handhant.

```
class HandHant implements ActionListener{ public void  
actionPerformed(ActionEvent event) if(event.getSource() == send1){  
skickaivag(); else if(event.getSource() == spar1) sparafil();
```

Etc. for each button. When the button is pressed on the first page, then event.getSource()=send1 occur and method skicka() is called and data sends to the RFID transponder, when the button save is pressed then the method sparafil() is called and information is saved.

## Method make()

The method make() take the input text and add Vcard information, to make it agree to the vcard standard.

The method make() first take text from the input textfield and place it into a string with getText().

```
nv1=family_name.getText()  
nv2=homephone.getText()
```

Then shall all strings put together to one only string and Vcard information shall be added. All obtains strings with getText(): nv1,nv2..... is send to the method

laggihop(), it take all the string in the same time. There is number of consant strings in the method laggihop().

```
String a2="BEGIN:VCARD\nVERSION:2.1\n"; String b1="N:";
```

```
String c1="FN:";
```

i.e

This Vcard information shall be put together with respective string. For example a person name, it shall have the Vcard index N:. It starts to check if: lastname =c, firstname = a, middlename = b and title = e is greater than 0.

```
if(c.length()>0 || a.length()>0 || b.length()>0 || e.length()>0)
```

Is't greater than 0, e.g ther is text in some of the textfields, then Vcard information is added.

```
A1=a1+b1;
```

There a1 from the beginning contain BEGIN:VCARD\nVERSION:2.1 and b1=N:. Then check if c >0 i.e there is a lastname, if there is, it adds to the string a1, if not only a ; is added. Then it's checked if there is a surname, is't? If yes it's added, if not a ; is added. On the same way it's doing with middlename and title. The last thing to do is to add a newline sign /n. On similar way this is doing to the other input textfields and every thing is added together in one string. The string can look like:

```
BEGIN:VCARD\nVERSION:2.1\nFN: Stefan Andersson\nN:Andersson ;Stefan  
; Bengt; Bengan; Engineer\nADR: ;; Mainstreet 55 ; The City ;; Zip 44123;  
Sweden\nEND:VCARD
```

This code is similar than the vcard format except from the new line characters /n. That string then can be saved or be send to the transponder.

### **Method open()**

The event that occur when button open is pressed. It open a file with the method oppnafil() as return a textstring DEF. DEF is a textstring in Vcard format. The string shall be sorted and show on the screen. A method is used for search the Vcard index

and then show it on the screen. To search for example for the name, the method `namnNN(To_screen)`, it search for the index N: and then it search for : and ;. The String `To_screen` is whole Vcard textstring.

```
public void namn_NN(String To_screen)
    { array_of_index[2]=To_screen.indexOf("N:",array_of_index[1]);
      array_of_index[3]=To_screen.indexOf("N;",(array_of_index[2]+2));
      array_of_index[4]=To_screen.indexOf(":",array_of_index[3]);
      array_of_indexsemi[2]=To_screen.indexOf(";",array_of_indexsemi[1]+1);
```

The for N: or N; i placed in the vector `To_screen[]`, each textfield is separated with a ;. These are placed in the vector `array_of_indexsemi[]`, the whole textfield in use are ended with a /n, that is placed in `array_of_index [5]`. Then text is showed on the screen with the function and the right position `position.setText ()`.

Lastname

```
family_name.setText(To_screen
    .substring((array_index[3]+1),array_of_indexsemi[0]))
```

Given name

```
fornamn.setText(To_screen.substring((array_of_index
    semi[0]+1),ajaarray_of_indexsemi[1]))
```

Nickname

```
nick_name.setText(To_screen.substring((array_of_indexsemi[1]+1),array_of
    index [5]))
```

The other string is done in the same way.

### **Method Read**

The method `read()` read the transponder and show the text on the screen. The method should send a request to the reader thru the serialport, but in this case it uses the command prompt. That should be send is according to 15693 standard is start of

frame SOF, flag, command code, mandatory parameter, application data fields, CRC and end of frame EOF. The TI-reader uses some extra header and don't use the 16-bit CRC, instead a XOR checksum must be calculated (see sect. 4 Command for the TI-reader module ).

It's not possible to read all the 64 blocks at the same time, thus it's done twice, first block 0 to 31 and then the block 32-63. To read the first 32 blocks the following code should be send:

```
01 14 00 00 00 00 96 17 03 35 00 31 65 190
```

In this case it's done with a *System.out.println(read\_req\_array[i])* command.

The respond is either the data from the transponder or some error message.

The respond is read to the program is done with a class of the java BufferedReader:

```
BufferedReader myIn2= new BufferedReader(new  
InputStreamReader(System.in)
```

Where (System.in) is input from the Dos-window, it should be change when use of a RS-232 interface. While the data is reading a XOR-checksum is calculated to be compared with the two last bytes. When the reading is completed and the checksum is ok, the Header is removed and it convert the Acsii-code to letters. Then the read information should placed on the screen the same methods as in the method Open() is called to place the information on screen, for example:

```
namn_NN(sum_read_ascii_string)  
namn.nickname (sum_read_ascii_string)
```

## **Send()**

The method Send() should send Vcard data to the transponder thru the serialport and the reader ( in this case only to the DOS-window). The Vcard data that have been created with the method Make() is first converted to ascii-code, then the remaining field should be added according to the 15693 -standard and the TI-reader module code (see sect. 4 Command for the TI-reader module) SOF, flag, comand code, checksum. Because the TI-reader only can send one block 4 bytes at a time, it's repeated 64 times. The input to the function is the string ABC created by the function Make(). After the convertto Asii it's divided into blocks. The header information is added, then it's write to the outport in this case the DOS-window with *System.out.println(sendarray[i])*. At the sime time the XOR-checksum is calculated

and then the checksum writes to last to byte. After each transmission of a block, it wait for the respond from the TI- module. If the transmission is ok, it's continue with the next block.

### **Exceptional events**

When an execution error come up in the software. i.e. an error come up when the program is running, for example a try to a file when doesn't exist, this is called exceptional event. In java there is a mechanism to handle this situations, when an error occurred in a method an exceptional event is generated. This exceptional event is by definite rules sent to other methods where them are captured and adequate measure are follow.

When generating an exceptional event

To signal simpler errors so-called run-time errors, for example index outside of a field. This error is signal with objects of subclass to the class `java.lang.exception`. With error occurred when input and output, for example when trying to opening a file that doesn't exist. This error is signaling with objects of subclass to the class `java.io.exception`. When an exceptional event occurred the normal executions is terminated and control of the software is transferred to the part of the software that capture events. To make software to capture exceptional events, in the declaration add `throw`.

*Public static double read( ) throws IOException*

When an exceptional event has been generated thus it should be handled it can be done with *try*-sentence.

```
while(true)
    try{
        code
        :
    }
    catch (NumberFormatException ne) {
        code
```

:  
}

## 8 Communication between host computer and transponder

Between the host computer there basic three things. A micro controller, a RFID circuit and an antenna.

The micro controllers task is to communicate with the host computer and RFID-circuit. The micro computer have a serialport, in this case a RS-232 interface. An USB-interface should be better, not just it more up to date also because it's possible to have power supply through the serial-port. The microcontroller communicates with the RFID-circuit with three wire-interface. The microcontroller shoul also acts as buffer on both ways and error handling.

The S6700 Multi Protocol Transceiver IC provides the receive/transmit functions required to communicate with the three types of transponders that operate in the 13.56 MHz ISM band [8]. A transmit encoder converts the transmitted data stream into the selected protocol; protocol selection is done in the header of the transmitted data string. The transmitter can provide up to 200 mW of RF power to a matched 50 Ohm load with a 5 V power supply. Higher output power can be obtained by an external amplifier. The receive decoder converts the signals from the RF receiver into a simple data string. The protocol uses a simple three wire seriallink between the transceiver ic and the ucontrolller to transmit data and set up data.

The antenna could be printed to the the board.

## Transceiver-chip

For each communication, the remote controller must send a command to perform an appropriate sequence [8]. A typical command is structured as follows: S1, eight bits command, data, ES1, se figure 21.

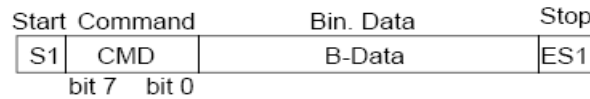


Figure 21. Data to the transceiver chip [8].



Figure 22. start of frame, a one, a zero and the end of frame [8].

The command byte is set to choose between several functions, for example if it's ISO 15693 eller ISO 14443, modulations dept, 1 of 4 eller 1 of 256, high or low baudrate. For example bit0-bit7 = 011000110.

Because the micro controller cannot control the timing of sending data to the TAG.

The transceiver ic must store the data from the micro controller. The capacity of storage being limited, management of the buffer must be implemented. The buffer is implemented as a 16 bit FIFO. In figure 23 shows an example of transmission and when buffer is full.

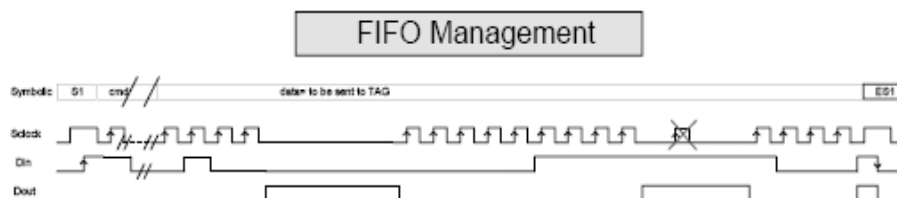


Figure 23. viewing when the buffer is full [8].

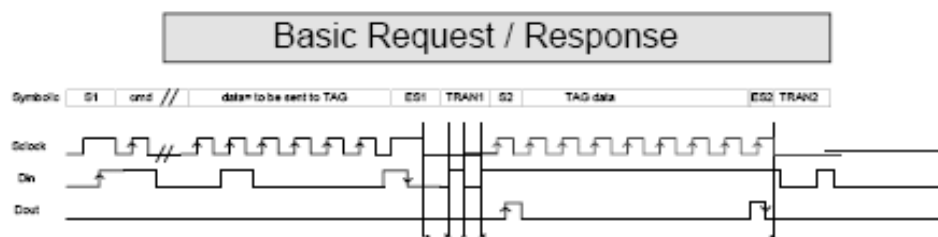


Figure 24. request and response [8].

A typical request and response -procedure are shown in figure 24.

Definition TRAN1.

During Transient 1 (TRAN1), the controller gives control of the Sclock line to the transceiver chip: Din =0

- Time a: The bit ES1 is finished.
- Time b: The controller raises Din, either to prepare a control mode change for the Sclock line or to prepare an end of transmission, ES1.
- Time c: Din is falling. The controller definitely indicates that it will give the Sclock line control to the Transceiver chip. Sclock =0 and both the controller and the transceiver chip are outputs.
- Time d: Din rises showing that the controller leaves the control of the bus until DIN falls to ask the control of Sclock back. At time d, Sclock is still equal to 0 but the pin Sclock of the controller is an input and the pin Sclock of the Transceiver IC is an output.

When the Transceiver chip has control of Sclock, it will send a S2 that corresponds to a start of frame sent by the transponder, the data (7 bits in Figure 21) and an ES2 that corresponds to the end of frame of the transponder.

Definition TRAN2:

During Transient 2, the controller regains control of Sclock: DIN =1

The controller indicates its intention to regain control over Sclock by setting Din=0 and initiate a change by making a pulse on Din. It is during this pulse that the line Sclock will change direction.

### **Communication between host-computer and the transponder**



To communicate with the RFID-circuit and the host-computer an Atmel 8515. It's cheap 8-bits microcontroller with UART. It have 512 Bytes memory, that means that information sends to the transponder must be divided.

The microcontroller is in idle when start, it waits for a start of frame. The data that sends is start of frame, number of bytes, configuration data, data and checksum. The start of frame is 8 bits 10011001. Configuration-data is information how the transponder-circuit acts. Checksum is calculated with xor checksum.

The micro-controller reads the number of bytes as being send and a byte with configurations-data, these are placed, in an each register. Then all the data are stored to the memory and the checksum is calculated, when all data is send the transfered checksum is compared with the calculated checksum. If everything is okay then the microcontroller should communicate with transponder-chip.

### **Communication with transponder-chip**

Each transmitted data bit is latched rising clock, the value of the bit must the same when the clock  $S_{clock}$  is high, Pin 8 on portA is used for clock. When data is transmitted from the microcontroller, the microcontroller controls the clock [1]. The beginning of the transmission shall have a start of frame, it's a low to high transition on the data-input when the clock holds high. The completion EOF is a high to low transition when clock holds high.

For the start of frame, the clock (portA, pin1) to one, then the program jumps to the subroutine *timer* there it waits, then it jumps back to the main program and set the data bit to one, then it jumps back to timer and waits again, then a jump back again to the mainprogram sets both the data and clock bit to zero.

### **Data transmission to the transceiver-chip**

A byte is fetched from the memory and are stored in memory register, then write to the output (portB pin 0), then the program jumps to the subroutine clock, it handle the clock and waits, then it jumps back to the main program and shifts all the bits in the register one bit to right and the new value is put to the output, then the same thing is done again seven times to write all the eight bits.

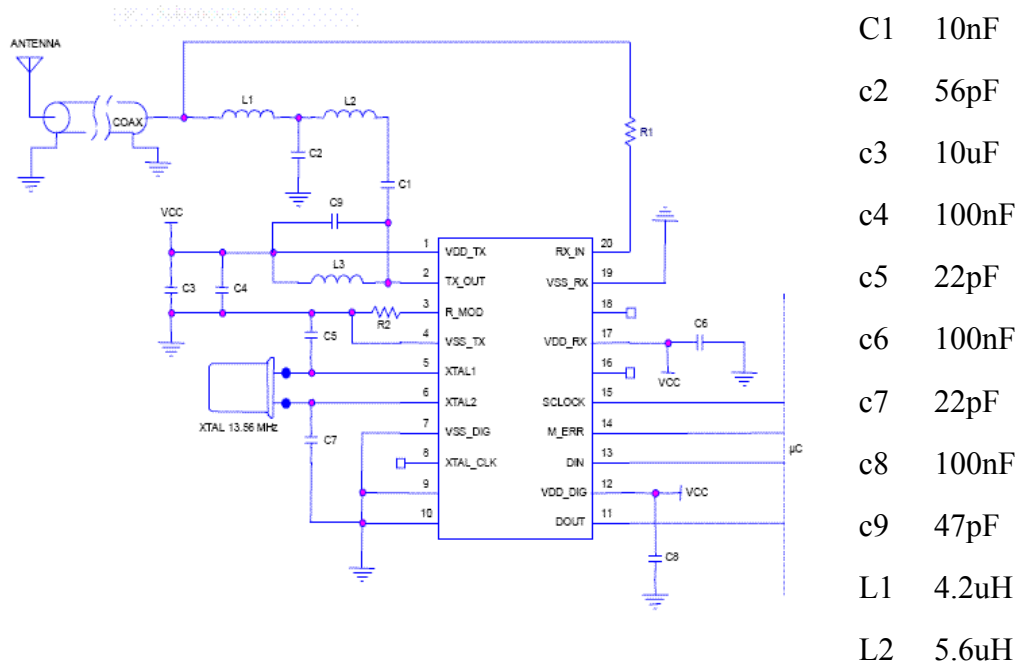
While the transmission is going on the output, the Dout (portA, pin0) is checked if it is zero, if it is one the buffer in the transceiver-chip is full and the program jumps to the interrupt routine. When the buffer is empty i.e. Dout is zero the program jumps back from the interrupt and the transmission continues. When all the bits in a byte have been sent the program jumps back and fetches a new byte from the memory and does the same thing again until the memory is empty. When the memory is empty the microcontroller sends an EOF, it's a low to high transition on Din when clock holds high.

### **Write from Transceiver-chip to micro-controller**

It checks if Dout is rising while Sclock = 1, i.e. A start of frame. When the write micro-controller to transceiver-chip procedure is finished the micro-controller leaves the control of the transponder-chip. The data is received in serial, thus is stored in a register, this register is then shifted one bit to the left. After 8 bits then the byte is stored in the memory. After it's checked if an end of frame is present, i.e. Dout is falling while Sclock = 1, if not the program jumps back and reads a new byte. When an EOF is present, the microcontroller shall write to host-computer. First a SOF, here 11001100 sends, concurrent a XOR checksum is done. After that number of bytes to send are calculated and sent. Then stored data are fetched from memory and sent. Last the calculated checksum sends. Then the program jumps back to beginning.

## **9 Layout**

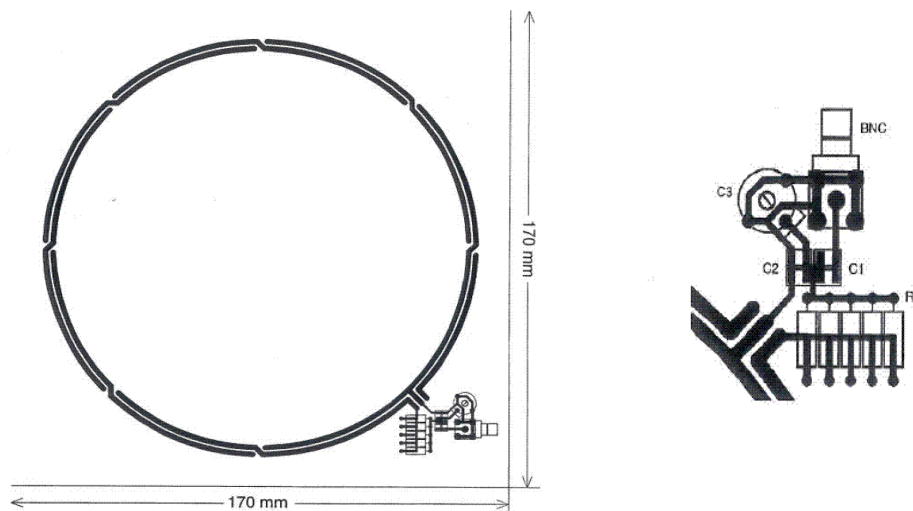
An application schematic which has been optimized to drive a 50 ohm antenna. At 5 V, this circuit will output typically 200 mW RF power when a suitable matched 50 ohm antenna is connected [8]. In Figure 25, a proposed layout for the transceiver chip is shown.



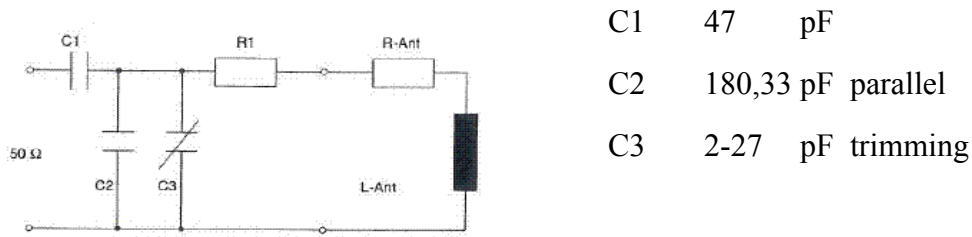
L3                                    1.2uH                                    R1    2.2kohm  
R2                                    12ohm

Figure 25. The Texas instrument transceiver chip and components [8].

An Example of an antenna design, see figure 26, with the antenna is printed on the circuit board [4].



R1    5x4.7 ohm parallel



- C1 47 pF
- C2 180,33 pF parallel
- C3 2-27 pF trimming

Figure 26. the antenna with components to match 50 ohm [4].

The pin configuration for the Atmel AVR 8085 microcontroller [1], shown in figure 27,

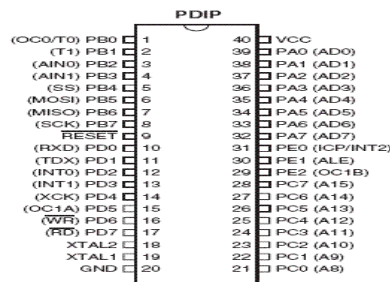


Figure 27. pin configuration of a Atmel AVR 8085 [1].

Pin 10 (RXD) receiver pin and pin 11 (TDX) is transmitter pin for the rs-232 interface. Pin 12 (INT0) is connected to the M\_error on the transceiver-chip. Pin 39 (PA0) is connected to Din on the transceiver chip. Pin 32 (PA7) is connected to Sclck on the transceiver chip. Pin 1 (PB0) is connected to the Dout on transceiver chip.

## **Conclusion and future work**

### **Conclusion**

The topic was to produce proposal to an electronic name card. It's done but the memory capacity was the big obstacle, probably in the future memory size in RFID transponder should increase, as starting point I have make the system with asset of a larger memory. I have created a graphical user interface, to handle the information on the transponder. But not to communicate with serialinterface. To store the information the format vcard is used, it's a spread format. Then I have propose how the hardware can be designed, it's with a micro controller, a transceiver chip and an antenna.

The most important thing is that the memory is to small. If you use that for the standard maximum memory size it's only possibility to store text. To store pictures, a picture is about 0.5 Mbyte, but if serialize it's about 2 Mbyte, about 5 Mbyte of memory size should be appropriate for an electronic name card.

On the other hand on and all are connected to the internet all the time and everywhere. So it's possible to use a memory with only 2048 bits with a few memory fields, for example: name, company name and web address and that the receiver of the card can add some information for example when he receive the card or just only the web address. Then the RFID- system connects to web server where all the information are.

### **Future work**

First is the RS-232 interface, other things to be added is a anti collision routine so more than one transponder can be in the area of the reader at the same time. Another thing that can be added is some security functions provided in the vcard format.

Another interface should be implemented, when the RS-232 almost have disappeared from today products.

If a larger memory, it's possibility to add more vcard functions, for example sound who pronounce the owner of the card correctly or add a map to a business location or some other place.

If transponder is developed just contain a address to website, a web server could be developed that contain information for the name card.

## References

[1] Atmel, Avr 8085 User manual.

[http://www.atmel.com/dyn/resources/prod\\_documents/doc2512.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2512.pdf)

[2] Cadenhead, Rogers. 2003. Lär dig java 2 på tre veckor. ISBN: 9163607344. Pagina förlags AB.

[3] Dawson Holst. September 1998. vCARD Directory profile.  
<http://www.ietf.org/rfc/rfc2426.txt>

[4] Finkenzeller, Klaus. 2004. RFID Handbook. ISBN 0-470-84402-7. Munich/FRG

[5] ISO/IEC 15693-2. 1999. Radio frequency power and signal interface.  
<http://www.wg8.gipp.com/sd1.html#15693>

[6] ISO/IEC 15693-3. 1999. Anticollision and Transmission protocols.  
<http://www.wg8.gipp.com/sd1.html#15693>

[7] Skansholm, Jan. 2002. Java direkt med Swing. ISBN 91-44-0228-X. Lund: Studentlitteratur.

[8] Texas Instruments. 2002. S6700 Multi-Protocol -Transceiver IC – Reference Guide.

<http://www.ti-rfid.com>

[9] Texas Instruments. Development kit, user guide.

<http://www.ti-rfid.com>

[10] Texas Instruments. Reference Guide, S6350 Midrange Reader Module,

RI-STU-TRDC-02

<http://www.ti-rfid.com>

## Appendix A

### A.1 Example of testing the software with simulation of the TI-reader module

If the following text is simulated that it's stored on tag and it will be read.

```
BEGIN:VCARDVERSION:2.1/nN:Oestling;jan;;Student/nFN:jan Oestling/n
TITLE:Student/nORG:Kth;Campus Kista/nTEL;WORK;VOICE:08-7906000/n
ADR;WORK:Kista;Sweden;;;16440;;/nURL;WORK:www.it.kth.se/nEND:VCARD
```

When read button is pressed. A request to the DOS-window ( RS-232) And the following code sends.

```
1 14 0 0 0 0 96 17 3 35 0 31 65 190
```

The input shall be, but a little bit modified, the software is done to read half the tag, 32 blocks at a time, here 24 blocks and the second field is 106 instead of 138.

```
1 106 0 0 0 0 96 0 66 69 71 73 78 58 86 67 65 82 68 10 86 69 82 83 73 79 78
58 50 46 49 10 78 58 79 101 115 116 108 105 110 103 59 106 97 110 59 59
83 116 117 100 101 110 116 10 70 78 58 106 97 110 32 79 101 115 116 108
105 110 103 32 10 84 73 84 76 69 58 83 116 117 100 101 110 116 10 79 82
71 58 75 116 104 59 67 109 112 117 115 32 75 105 67 188
```

The program send a request to get the rest of the data.

```
1 14 0 0 0 0 96 17 3 35 32 31 97 158
```

The input shall be:

```
1 106 0 0 0 0 96 0 115 116 97 10 84 69 76 59 87 79 82 75 59 86 79 73 67 69
58 48 56 45 55 57 48 54 48 48 48 10 65 68 82 59 87 79 82 75 58 75 105 115
116 97 59 83 119 101 100 101 110 59 59 59 59 49 54 52 52 48 59 59 10 85 82
76 59 87 79 82 75 58 119 119 119 46 105 116 46 107 116 104 46 115 101 10
69 78 68 58 86 67 65 82 68 0 40 215
```

And the result is shown in the Grafical User Interface.