# Entwicklung eines Demosystems für spezielle RFID-Anwendungen

MA696
Masterarbeit

Walther Pachler

Institut für Elektronik (IFE)
Technische Universität Graz
Leiter: Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Pribyl



unterstützt durch Infineon Technologies Austria AG

Begutachter:
Ass.-Prof. Dipl.-Ing. Dr.techn. Peter Söser
Betreuer: Dipl.-Ing. Günter Hofer (Infineon)

Graz, im August 2011

Never stop thinking

Diese Diplomarbeit wurde unterstützt von
Infineon Technologies Austria AG
Development Center Graz
Abteilung Contactless and RF Exploration
Leitung Dipl.-Ing. Gerald Holweg

# Kurzfassung

Diese Diplomarbeit beschäftigt sich mit einer weiterentwickelten Form des RFID Transponders. Der Transponder von Infineon unterscheidet sich nicht nur in der Größe von konventionellen Tags, sondern zusätzlich auch durch die integrierte Peripherie, wie zum Beispiel einen Shunt- und Temperatursensor. Die Größe des Chips wurde mit der Coil-on-Chip Technologie minimiert, welche, wie der Name schon vermuten lässt, auch die Antenne des Transponders am Chip integriert. Die Vorteile dieser Chips liegen neben der Größe und somit den Einsatzgebieten auch in der kostengünstigen Produktion. Ohne zusätzlich benötigte Antennen, Pads oder die generelle Weiterverarbeitung des Siliziumchips (wie z.B. Bonding), entfallen etliche Kosten. Somit kann dieser sehr kostengünstig (wenige Cent) hergestellt werden und ist theoretisch direkt nach der Siliziumproduktion einsatzfähig.

Ziel dieser Masterarbeit war es, ein UHF RFID Komplettsystem zu entwerfen, welches in der Lage ist mit dem bereits erwähnten Chip zu kommunizieren. Nach erfolgreichem Zusammenbau der eigens dafür entwickelten Komponenten (wie zum Beispiel einer kleinen UHF Antenne) soll das System stabil laufen und dem EPCglobal (Electronic Product Code) Class-1 Gen2 Standard entsprechen. Der EPCglobal-Standard definiert das Kommunikationsprotokoll von UHF RFID Systemen. Die Sensorwerte sollen außerdem periodisch abgerufen und in Form von Diagrammen dargestellt werden. Schlussendlich ergeben sich durch diese Neuentwicklung von Infineon mehrere neue Anwendungsbereiche, wobei auf manche genauer eingegangen und ein dazu passendes Demosystem gebaut wird. Die zwei resultierenden Demosysteme sind in Abbildung 1 dargestellt.


Abbildung 1: Geld- und Bienendetektion-Demosystem

Schlagwörter: UHF RFID, Reader, On-Chip-Antenne, OCA, Tag, Label

## Abstract

This master's thesis is related to a new form of RFID transponder developed by Infineon. The transponder includes multiple types of sensors, such as shunt- or temperature-sensor. By virtue of the OCA-technology which is used, the size of that transponder could be minimized to one square millimeter. As the name OCA (on-chip-antenna) suggests, the antenna is effectively a coil which is integrated on the chip. As a consequence, no additional antennas, pads and other post wafer fabrication processes are used. Therefore the biggest advantages are the low price and the new application areas.

Goal of this master's thesis was to develop a whole UHF-RFID-system, which operates with this new type of transponders. Therefore all necessary components, such as an UHF-nearfield-antenna should be used. The system should remain stable and be compliant to the EPCglobal Class-1 Gen2 standard.

Accordingly, all sensor of the transponder should be read and plotted into an arithmetic chart. A graphical interface is used to analyze the values and to operate the system. Finally, the thesis discusses new application areas and makes assumptions on some of the points that arise. Some new application areas should be figured out and solved with a demo system (such as bee-detection).



Keywords: UHF RFID, On-Chip-Antenna, OCA, bee-detection

## 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)

## 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.

..... .....

date (signature)

# Danksagung

Am Ende meines Masterstudiums Telematik entstand die vorliegende Masterarbeit in Kooperation mit der Firma Infineon Technologies. Daher gebührt vor allem mein Dank der Firma Infineon, insbesondere Dipl.-Ing. Gerald Holweg und Dipl.-Ing. Günter Hofer, die mir diese wertvolle Zeit und Industrieerfahrung ermöglicht haben.

An dieser Stelle danke ich auch meinen Betreuer seitens der Universität, Herrn Dipl.-Ing. Dr.techn Peter Söser. Durch Infineon hatte ich das Gefühl meinen Teil zu etwas Großem beizutragen.

Ganz besonders möchte ich mich bei meiner Mutter und meinen Schwestern bedanken, die mich immer, auch während des gesamten Studiums hinweg, in jeglicher Art und Weise unterstützt haben. Dazu gehört natürlich auch mein Vater, der das Lesen dieser Danksagung leider nicht mehr erleben durfte. Ich weiß wie stolz du auf mich warst, jedoch bin ich mir sicher, dass du, wo auch immer du jetzt bist, das alles mitverfolgst!

Danke auch an alle CRE-Kollegen. Ich freute mich jeden Tag in der Früh auf diese lustige, angenehme und vor allem kompetente Abteilung, in der ich so einiges dazulernen durfte.

Zuletzt danke ich noch meiner Liebe Kerstin, die mir neben meiner Familie auch in schwierigeren Zeiten sehr viel Rückhalt und Unterstützung gab.

Walther Pachler

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Kapitel 1: Einleitung

RFID (Radio Frequency Identification) revolutioniert seit Jahren die Identifikationstechnologie. Strichcodes und ähnliche Systeme gelten als veraltet und reichen in deren Kapazitäten und Einsatzmöglichkeiten meist nicht mehr aus. Ohne Sichtkontakt können auf diese Weise Objekte drahtlos detektiert und identifiziert werden.

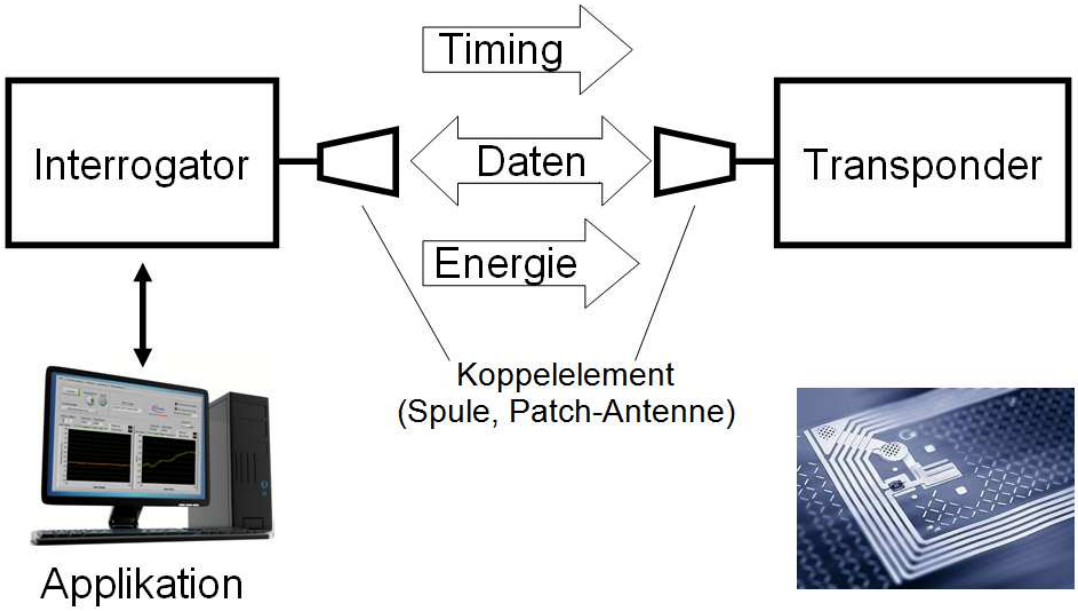


Abbildung 1.1: RFID-System

Ein konventionelles RFID-System, wie in Abbildung 1.1 dargestellt, besteht aus einem elektronischen Datenträger, dem Transponder, und einem entsprechenden Lesegerät, dem Reader, der oft auch als Interrogator bezeichnet wird. Diese zwei Komponenten kommunizieren mit einem geeigneten Koppelelement (Spule etc.) über ein elektromagnetisches Feld. Meist speichert der Transponder Identifizierungsdaten des Objekts. Die Applikation, die am Reader angeschlossen ist, verarbeitet die Daten entsprechend und stellt sie beispielsweise grafisch dar.

## 1.1 Motivation

Ziel dieser Masterarbeit ist es ein UHF RFID Gesamtsystem aufzubauen, jedoch mit einer weiterentwickelten Form des Transponders. Von konventionellen Transpondern unterscheidet sich dieser in erster Linie durch die Größe. Mit der Coil-on-Chip Technologie wurde die Größe des UHF-Transponders auf $1\,mm^2$ reduziert. Wie der Name Coil-on-Chip vermuten lässt, befindet sich hier die Antenne direkt am Chip. Zusätzlich sind im Transponder Sensoren wie Temperatur- und Shuntsensor integriert.

Es ist zu betonen, dass es sich hierbei um einen Testchip handelt. Das zu entwickelnde System soll also eine Kommunikation mit dem OCA(On-Chip-Antenna)-Transponder ermöglichen und die Funktionsweise verifizieren. Dabei sollen insbesondere die integrierten Sensoren verwendet und getestet werden. Die gemessenen Temperatur- und Shuntwerte sind anschließend graphisch auszuwerten.

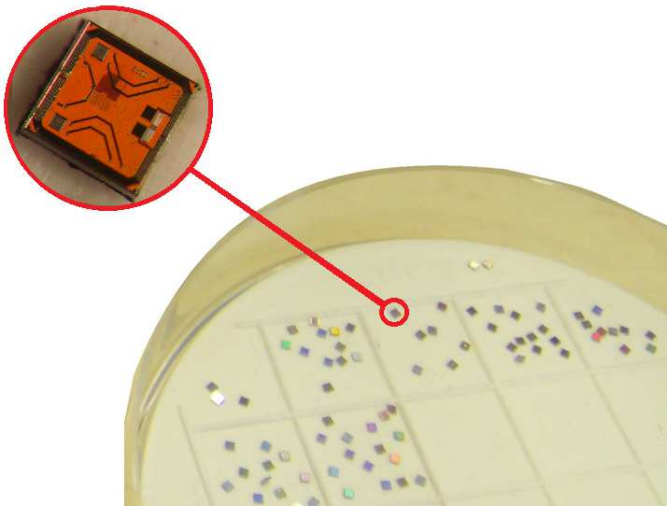

Abbildung 1.2: OCA(On-Chip-Antenna)-Transponder

Betrachtet man in Abbildung 1.2 die Größe der OCA-Transponder so erkennt man das Potential des Chips. Die Einsatzmöglichkeiten des nur $1\,mm^2$ großen Chips scheinen schier unbegrenzt. Sei es zur Detektion und Identifizierung kleinster Objekte, bis hin zum Sicherheitsmerkmal von Papier, Gutscheinen, Geld und dergleichen. Die in Abbildung 1.2 dargestellten Transponder sind nur $200\,\mu m$ dick. Das Silizium kann ohne Probleme auch auf $40\,\mu m$ zugeschliffen werden. Der Begriff "Intelligenter Staub" trifft hier auf jeden Fall zu.

> *"Die Neigung der Menschen, kleine Dinge für wichtig zu halten, hat sehr viel Großes hervorgebracht." Georg Christoph Lichtenberg (1742-99) [11]*

## 1.2 Gliederung der Arbeit

Die Arbeit beginnt mit einer kompletten Erörterung aller Teilkomponenten des zu entwickelten UHF-RFID Systems. Dabei werden in Kapitel 2 die einzelnen Hardwarekomponenten vorgestellt und deren Einsatz, Funktionsweise und Zusammenspiel mit den anderen Komponenten erklärt.

Kapitel 3 beschreibt anschließend die Systemsoftware. Die Software vom Tag bis hin zur graphischen Sensorauswertung geht über mehrere Instanzen, welche in diesem Kapitel einzeln hervorgehoben werden. Hier werden Aufbau und die dazugehörigen Schnittstellen zur benachbarten Ebene erklärt.

Ausgehend von einer Oszilloskopmessung wird in Kapitel 5 die Kommunikation auf Richtigkeit überprüft bzw. verifiziert und die ermittelten Daten mit dem EPC-Global-Standard für UHF verglichen.

Das sechste Kapitel geht näher auf die aus dem entwickelten System resultierenden Demosysteme ein. Sowohl der Aufbau als auch die Entwicklung eines Demosystems für Geldverifikation und Bienendetektion werden hier vorgestellt.

Abschließend werden dem Leser im letzten Kapitel (Kapitel 7) eine kleine Zusammenfassung und ein kurzer Ausblick der verwendeten Technologie gewährt.

# Kapitel 2: Beschreibung und Aufbau der Hardware-Systemkomponenten

In diesem Kapitel wird die gesamte verwendete Hardware des RFID Systems vorgestellt. Dabei handelt es sich zum größten Teil um den zur Verfügung gestellten USB Reader und dessen Modifikationen. Zusätzlich werden die verwendeten UHF OCA Tags bzw. Labels vorgestellt und deren Funktionsweise detailliert erörtert. Um die Kommunikation zwischen OCA Chips und dem USB Reader herzustellen, musste zuerst der im Original-Reader integrierte Dipol mit einer für diese Anwendung gebauten Nahfeldantenne ausgetauscht werden. Beide Antennen werden hier detailliert analysiert und beschrieben. Neben den ersten erfolgreichen Tests wird in diesem Kapitel auch die Hardwarekonfiguration, die zum Umprogrammieren des Readers verwendet wird, aufgezeigt.

## 2.1 Der OCA Transponder

Dieser spezielle von Infineon entwickelte Tag entspricht dem Herzstück der Diplomarbeit. Ziel ist es, den (in seiner Größe einzigartige) UHF Tag mittels Reader auszulesen und dessen Sensordaten zu verarbeiten. Dabei ist eine geeignete Speicherbehandlung mit dem dazugehörenden Timing wichtig. In den folgenden Punkten werden zunächst der OCA-Transponder und seine Fähigkeiten allgemein beschrieben. Anschließend wird auf die Sensorik, deren Auswertung und die speziellen Ausführungsformen des Chips genauer eingegangen.

Typische RFID Tags sind aufgrund der limitierten Energieversorgung über das elektromagnetische Feld in ihrer Funktionalität stark eingeschränkt. Der hier vorgestellte, von Infineon entwickelte Chip wurde daher speziell für die Low-Power Anwendung entwickelt. Er ermöglicht den Betrieb und die Energieversorgung von verschiedensten Sensoren die laut [7] auch mehrere mW benötigen dürfen.

### 2.1.1 Multifrequenz-Ausführung

Das Hauptaugenmerk des Chips liegt in der Fähigkeit, sowohl HF- als auch UHF als Arbeitsfrequenz verwenden zu können. Je nach Betrieb nützt der Chip die jeweiligen Vorteile der unterschiedlichen Frequenzen. HF profitiert von der hohen zur Verfügung stehenden Energie im elektromagnetischen Feld, wohingegen UHF eine hohe Lesedistanz gewährleistet. Sowohl der EPC HF- als auch der EPC Class 1 Gen 2 UHF-Standard sind im Chip implementiert. Eine Frequenzdetektion wählt den jeweiligen Betrieb aus. Typische HF Systeme arbeiten bei einer Frequenz von 13,56 MHz, UHF-Systeme von 865 MHz bis 965 MHz und bei 2,45 GHz.[7]

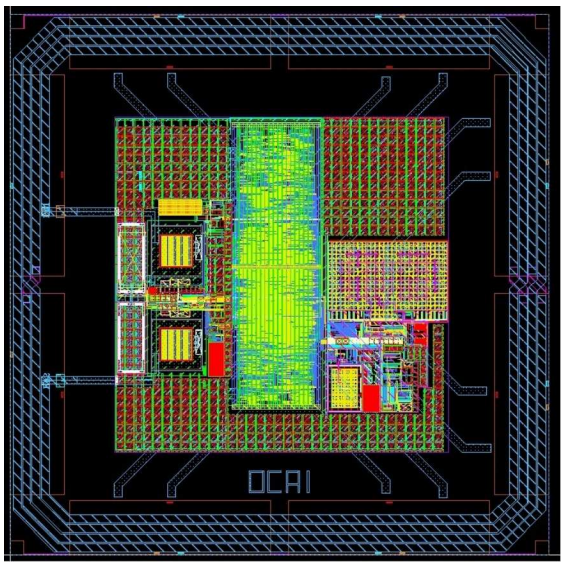


Abbildung 2.1: Layout des OCA-Transponders [7]

Für die unterschiedlichen Frequenzen bzw. Betriebsmodi werden natürlich auch unterschiedliche Antennen verwendet. [7] beschreibt ein *comprehensive system*, dessen Antenne einen kombinierten Betrieb von HF und UHF ermöglicht. Diese Variante wurde als Label ausgeführt und bereits in vielen Publikationen veröffentlicht. Diese Masterarbeit konzentriert sich jedoch auf die OCA-Ausführungen, die in Abbildung 2.1 dargestellt ist. Die On-Chip-Antennen (OCA)-Variante bietet alle im ersten Kapitel beschriebenen Vorteile. Die unterschiedlichen Ausführungsformen des OCA-Chips werden im Punkt 3.1.2 beschrieben.

### 2.1.2 Sensor- und ADC-Auswertung

Bei den im OCA Transponder implementierten Sensoren handelt es sich um einen Temperatur- und einen Shuntsensor. Die Signale beider Sensoren werden über den gleichen Analog-Digital-Converter (ADC) verarbeitet. Je nach Befehl wird der jeweilige Sensor an den ADC angeschlossen.

Ein solcher Befehl erfolgt im Normalfall über das sogenannte Special-Function-Register des Transponders. Dieses Register verfügt über erweiterte Funktionen und dient unter anderem zur Datenverarbeitung und Steuerung der I/O-Peripherie. Die für die Sensoren benötigten Signale sind ”ADC-Ready” und ”ADC-Enable”. Wie der Name der Signale vermuten lässt, steuern diese den Analog-Digital-Converter des Transponders. ”ADC-Enable” startet die Sensormessung und ”ADC-Ready” signalisiert ein fertiges Ergebnis. Da dem Transponder von Haus aus wenig Energie zur Verfügung steht verbrauchen Sensormessungen verhältnismäßig viel Strom. Daher ist eine solche Steuerung unbedingt notwendig.

Der implementierte Digitalteil des OCA-Transponders verbindet dieses Special-Function-Register mit dem ersten Register (erste zwei Wörter) der Userbank. Die Userbank ist der Teil des Speichers, auf dem der Benutzer mit den jeweiligen Befehlen einen ausgewählten Bereich lesen oder beschreiben kann. In Abbildung 2.3 ist der Bereich der Userbank, der zur Steuerung der Sensoren dient, dargestellt. Der genaue Ablauf der Sensorsteuerung ist aus dem im Transponder implementierten VHDL-Code ersichtlich. Der wichtigste Teil vom VHDL-Code für die erwähnte ADC-Steuerung wird in den folgenden Abbildungen aufgezeigt.

```
adc_data <= "111100" & adc_data_i when adc_rdy_i = '1' else
            "000000" & adc_data_i;
```

Abbildung 2.2: Ausschnitt vom Digitalteil des OCA-Tags

Der VHDL-Code in obiger Abbildung beschreibt das Beenden der Sensormessung. Setzt der Analog-Digital-Converter des OCA Transponders das Bit *adc_rdy_i* auf 1, so signalisiert er, dass der Sensorwert erfolgreich ausgelesen wurde. Somit sind die digitalen Sensordaten vorhanden und abrufbar. In diesem Fall setzt der Digitalteil des Transponders, wie aus dem Code ersichtlich, die vier MSB Bits des zweiten Registers der Userbank auf logisch high. Die ersten vier MSB Bits von *adc_data_i* also ”000000” wenn keine Daten und ”111100” wenn Daten verfügbar sind. Die Daten selbst befinden sich dann in den LSBs des Registers. Die Analogwerte der Sensoren werden vom ADC mit einer Auflösung von 10 Bit in Digitalwerte gewandelt. Wenn eine erfolgreiche Messung

durchgeführt wurde, liegt also folgendes Datenformat des zweiten Registers (2 Wörter - 16 Bit) vor: ”1111 00DD DDDD DDDD” (D=Daten). Man erkennt, dass das Register einen Maximalwert F3FFh nicht überschreiten kann. Dies kann in weiterer Folge zu einer zusätzlichen Fehlererkennung überprüft werden.

```
case sreg2 is
  when IDLE =>
    if adc_en = '1' and memRdy = '1' then
      next_sreg2    <= waitADCRDY;
      set_adc_data  <= '1';
      adc_start_mem <= '1';
    end if;

  when waitADCRDY =>
    if adc_rdy_i = '1' then
      next_sreg2    <= waitXUS;
      set_adc_data  <= '1';
      adc_start_mem <= '1';
    end if;

  when waitXUS =>
    run_timer <= '1';
    if timer_rdy = '1' and memRdy = '1' then
      next_sreg2    <= IDLE;
      set_adc_cfg   <= '1';
      adc_start_mem <= '1';
    end if;

  when others => null;
end case;
```

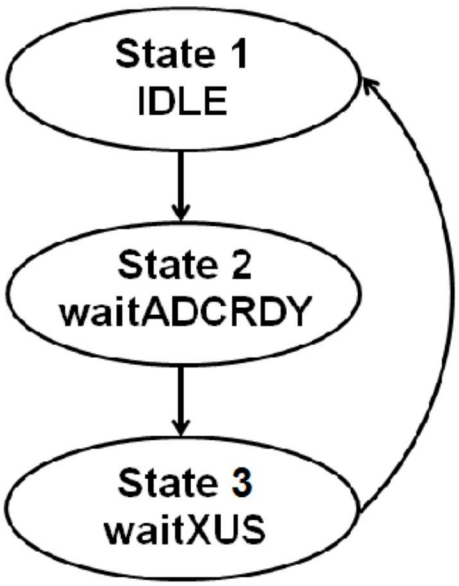


Abbildung 2.3: Ausschnitt vom Digitalteil des OCA-Tags

Beim zweiten wichtigen Digitalteil für die Sensorauswertung des Transponders handelt es sich um die Aktivierung der Messung. Hier ist ein für das Projekt wichtiges Timing vorhanden. Dies wird unter anderem im VHDL-Code der Abbildung 2.3 aufgezeigt. Startbedingung für den Analog-Digital-Converter ist das Bit *adc_en*. Dieses Bit wird z.B. durch den RFID-Reader mit einem ”Write-Befehl” auf den Speicher gesetzt. Natürlich muss auch überprüft werden, ob der Speicher fertig geschrieben und einsatzfähig ist. Wenn also *adc_en* und $MemRdy$ auf high sind, startet die Messung und das erwähnte Timing beginnt.

Nach dem Start wartet das Special-Function-Register des Transponders auf das Ergebnis des Analog-Digital-Converters, welcher die Daten wie bereits erwähnt, mit dem ”ADC-Ready”-Signal freigibt. Der ADC benötigt für die Wandlung des Sensorsignals von Analog zu Digital bis hin zum Bereitstellen der Daten ca. 500 $\mu s$. Während dieser Zeit dürfen keine

zusätzlichen Befehle vom RFID-Reader zum Transponder gesendet werden. Jegliche Art von Kommunikation würde das Feld und somit die Energieversorgung des Transponders ändern bzw. schwächen, was die Sensormessungen stark beeinflussen würde. Weiters darf logischerweise auch das Feld, das der Reader für die Energieversorgung des Transponders zur Verfügung stellt, nach dem Setzen von *adc_en* nicht abgeschaltet werden. Typische RFID-Reader stellen nicht dauerhaft Energie zur Verfügung. Sehr oft, wie auch manchmal nach dem "Write-Befehl", wird bei Readern ein sogenannter Feld-Reset durchgeführt, welcher keine Sensormessung ermöglichen würde.

Nach den 500 $\mu s$ Wartezeit auf $adc\_rdy\_i$ wird noch eine einstellbare Zeit ($waitXus$) gewartet. Dies gewährleistet den sicheren Betrieb des dazugehörigen Analogteils im Transponder. Standartmäßig sind hier 10 $\mu s$ eingestellt. Abbildung 2.3 zeigt klar den beschriebenen zeitlichen Ablauf.

Das Auslesen der Sensorwerte des Transponders mit dem RFID-Reader besitzt daher eine zusätzliche zeitliche Bedingung. Daher müssen alle oben beschriebenen Bedingungen erfüllt werden, um das Auslesen zu ermöglichen und das Ergebnis nicht zu verfälschen.

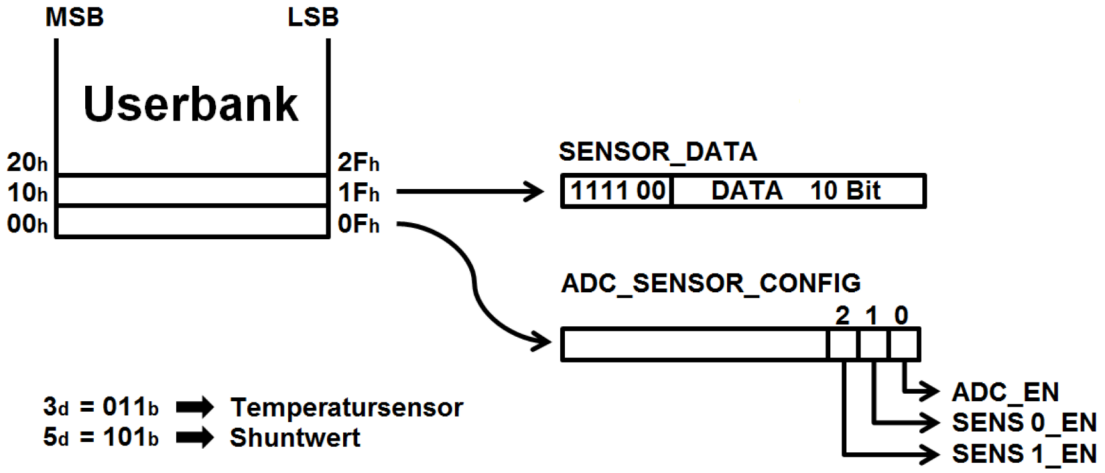


Abbildung 2.4: Memorymapping

Abbildung 2.4 stellt eine Übersicht der Sensormessung und dessen Speicherverwendung dar. $ADC\_SENSOR\_CONFIG$ (das erste Register der Userbank) wird dazu verwendet, den entsprechenden Sensor auszuwählen und die Messung zu starten. Dies ermöglicht ein Write-Befehl des RFID-Readers. Wird z.B. Dezimal 3 (=011b) auf das erste Register geschrieben, so werden die beiden Bits $ADC\_EN$ und $SENS0\_EN$ gesetzt. Beim zweiten Write-Befehl

(kleiner Bug im Digitalteil des Transponders) wird nun die Temperaturmessung (Sensor 0) gestartet. Wird der Dezimalwert 5 (= 101b) auf das erste Register der Userbank geschrieben (Setzen von $ADC\_EN$ und $SENS1\_EN$), startet die Shuntwertmessung. Das oben beschriebene Signal "ADC-Ready" des Analog-Digital-Converters setzt nach kurzer Zeit die vier MSB des Registers $SENSOR\_DATA$ und signalisiert damit, dass die Sensordaten (in den 10Bit LSB von $SENSOR\_DATA$) abrufbar sind. Diese können nun mit einem Read-Befehl des RFID-Readers ausgelesen und entsprechend ausgewertet werden. Dabei muss auf das oben beschriebene Timing geachtet werden.

## 2.2 UHF Readerantenne

Als Antenne des Lesegeräts (eng. Reader) wurde die von [3] entwickelte Nahfeldantenne zur Verfügung gestellt. In seiner Diplomarbeit beschreibt Herr Cordic die Entwicklung und den Aufbau der Antenne und weshalb diese für die Kommunikation mit dem OCA-Transponder sehr gut geeignet ist. Bei der Antenne handelt es sich um eine kleine Leiterschleife, die bei 865 MHz eine optimale Kopplung zum OCA Transponder gewährleistet. Diese Spule kann als folgendes Ersatzschaltbild betrachtet werden:

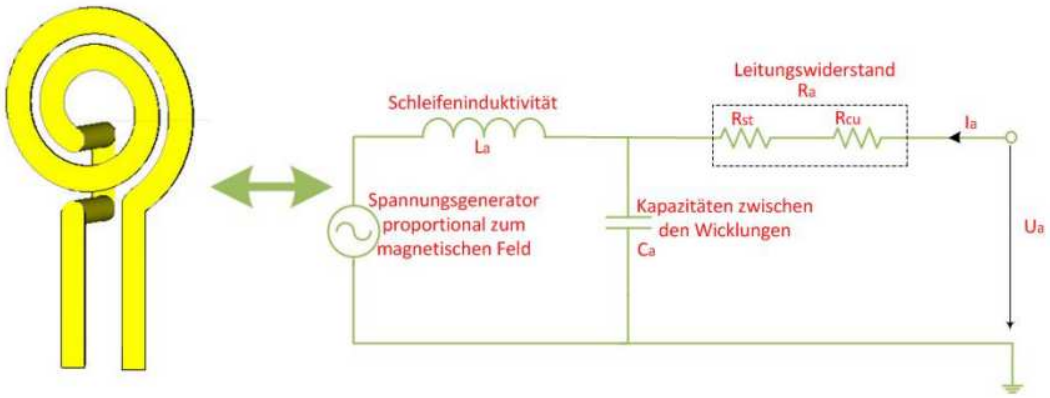


Abbildung 2.5: Nahfeldantennen und Ersatzschaltbild [3]

Die einzelnen Komponenten (ohmscher-, kapazitiver- und induktiver Anteil) ergeben sich aus dem realen Aufbau der Spule. Der Leitungswiderstand Ra setzt sich aus den Verlusten der Leitung Rcu und dem Strahlungswiderstand Rst zusammen. Wie der Name Rcu vermuten lässt, handelt es sich bei dem verwendeten Material der Leitung um Kupfer. Die hier auftretenden Verluste (Leistung Pv) werden in Wärme umgewandelt. [5]

$$P_v = |I_a|^2 \cdot R_{cu} \tag{2.2.1}$$

Der Strahlungswiderstand beschreibt hingegen die abgestrahlte Leistung:

$$P_{st} = |I_a|^2 \cdot R_{st} \tag{2.2.2}$$

Die Kapazität Ca stellt die Summe der parasitären Kapazitäten zwischen den Spulenwindungen dar. La entspricht der Spuleninduktivität.[5]

### 2.2.1 Anpassung

Um die Antenne optimal und mit einem hohen Wirkungsgrad einsetzen zu können, wird ein Anpassungsnetzwerk (engl. Matchingnetwork) verwendet. Hier wird die Eingangsimpedanz der Antenne an die Ausgangsimpedanz des Readers angepasst. Das Anpassungsnetzwerk und dessen Bauteilwerte können entweder analytisch mit Bauteil-Impedanzen und Schaltungstopologie berechnet, oder grafisch z.B. mit dem Smith-Diagramm ermittelt werden. Im Allgemeinen gibt es Pi- oder T-Anpassnetzwerke, mit denen alle Impedanzen aneinander angepasst werden können (Leistungsanpassung konjugiert komplexer Impedanzen). Für eine bestimmte Betriebsfrequenz können sowohl Pi- als auch T-Netzwerk in das jeweilige andere Netzwerke (T- oder Pi-Netzwerk) umgerechnet werden. Für die Anpassung der verwendeten Antenne kann jedoch auf einige Bedingungen der Anpassung verzichtet werden.

- Bei unserer Anwendung ist die Antenne immer eine ohmsch- induktive Last
- Phasenverschiebung des Signals am Verstärker-Ausgang zum Signal an der Antenne ist irrelevant. Die Antenne ist in unserem Fall sehr klein und viel kleiner als die Wellenlänge,
- Verluste in Kondensatoren sind vernachlässigbar; Verluste an Spulen im Anpassnetzwerk fallen hingegen ins Gewicht.

Somit ist es auch möglich, ein einfacheres Netzwerk in L-Struktur zu verwenden. [5]

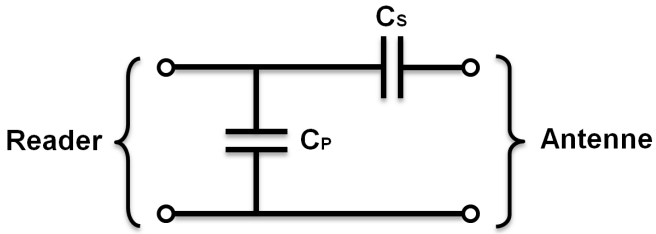


Abbildung 2.6: verwendetes Anpassnetzwerk

Cs erlaubt die kapazitive Variation entlang der Kreise des in Abbildung 2.7 dargestellten Smith-Diagramms. Hat also die induktive Antenne bei 865 MHz einen ohmschen Widerstand von 50 Ohm, kann mit dem seriellen Kondensator der Imaginärteil der Antenne kompensiert werden, ohne dabei den Realteil zu verändern. Die Antenne wäre in diesem Fall mit nur einem Kondensator auf 50 Ohm angepasst. Da dieser Fall nicht immer eintritt, kann die genaue Abstimmung der Anpassung mit dem Parallelkondensator durchgeführt werden, welcher Real- und Imaginärteil variiert.

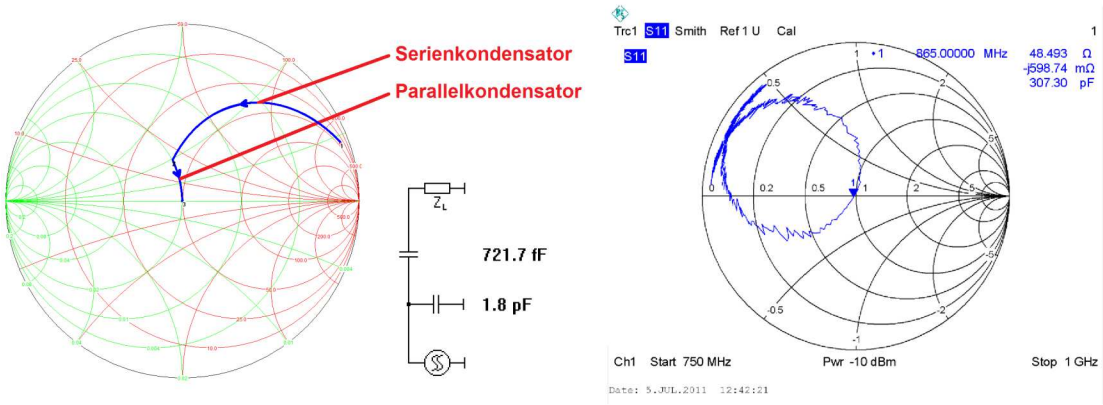


Abbildung 2.7: Anpassung und Verifikation durch Messung

Wie bereits erwähnt ist die Antenne ohmsch-induktiv. In Abbildung 2.7 ist der Anpassvorgang bei einer solchen Anfangsbedingung dargestellt. Wie die graphische Lösung vermuten lässt, ist auch eine umgekehrte Reihenfolge der verwendeten Kapazitäten möglich. Verwendet man zuerst die Parallel- und dann erst die Serienkapazität, so führt diese Variante mit kleineren Kapazitätswerten zum selben Ergebnis. Diese Topologie wird auch im Normalfall bei niedrigen Frequenzen verwendet um die Bauteilwerte gering zu halten. Um auch in unserem Fall bei einer Frequenz von 865 MHz Standard-Trimm-Kapazitäten (bis 5 pF) verwenden zu können, wurde die in Abbildung 2.6 aufgezeigte L-Struktur

verwendet.

Nach erfolgreicher Dimensionierung der Bauteilwerte, wurde das Anpassnetzwerk aufgelötet und mit Hilfe des Netzwerkanalysers und der Trimm-Kapazitäten fein abgestimmt. Das dazugehörende Messergebnis ist in obiger Abbildung im rechten Smith-Diagramm dargestellt.

### 2.2.2 Installation der UHF Nahfeldantenne

Abbildung 2.8 zeigt die bereits angepasste UHF-Nahfeldantenne und deren Installation. Die Originalantenne des Readers wurde entfernt und durch die Nahfeldantenne ersetzt und geeignet befestigt.

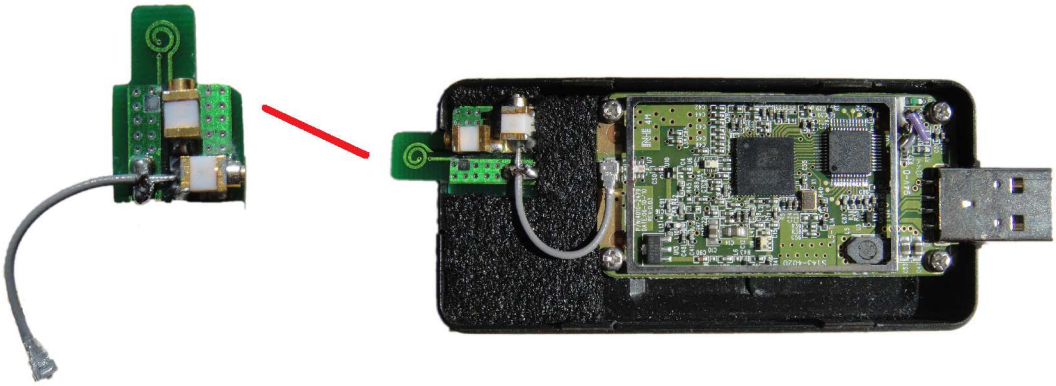

Abbildung 2.8: Readerantenne - Installation

Natürlich könnte die Nahfeldantenne auch innerhalb des Gehäuses angebracht werden. Aufgrund der geringen Lesereichweite und wegen der besseren Demonstrationsmöglichkeit des Systems wurde die Antenne jedoch sichtbar, wie in der Abbildung 2.8 dargestellt, befestigt.

## 2.3 Silabs C8051F340 Demokit

Der im Reader vorhandene Mikrocontroller C8051F340 besteht aus einer Steuereinheit des UHF Controller Chips und einem Interface vom UHF Chip zum Hostsystem (z.B. PC). Um den Mikrocontroller und dessen Möglichkeiten näher kennenzulernen, wurde auf das C8051F340 Demo Kit zurückgegriffen. Es besteht aus dem in der Abbildung 2.9

dargestellten Demo Kit Board, einem USB Debug Adapter, einem USB Kabel und einer für das Board passenden Stromversorgung. Zusätzlich ist die gesamte für Entwicklung und Debugging benötigte Software vorhanden. Bei "Silicon Labs IDE" handelt es sich zum Beispiel um die mitgelieferte Programmierumgebung. Mit dieser Software ist neben der Entwicklung der C-Programme auch die Programmierung des Mikrocontrollers möglich. Bevor der Eingriff in das Original-System des UHF-Readers RFID ME durchgeführt wurde, konnten durch das Demoboard der Programmablauf und vor allem die Kommunikation zum PC ausreichend geübt und entwickelt werden. Mehrere mitgelieferte Beispielprogramme erleichterten bzw. veranschaulichten die Funktionen des C8051.

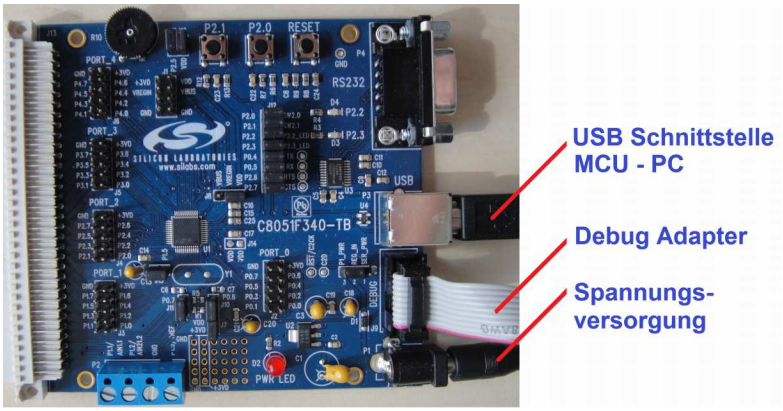


Abbildung 2.9: C8051 Demokit

## 2.4 USB Reader - RFID ME

Eine der Hauptanforderungen der Masterarbeit war es, für das UHF RFID System einen konventionellen USB Reader zu verwenden. Dieser sollte so modifiziert werden, dass die vollständige Kontrolle über das Gerät gewährleistet ist. Nur so ist das Implementieren der speziellen Befehle zur Sensorauswertung möglich. Bei dem von Infineon zur Verfügung gestellten USB Reader handelte es sich um den UHF USB Reader "RFID ME" von RF-IT-solutions. Dieser unterstützt ISO 18000-6b bzw. den EPCglobal Generation 2 Standard.

Die Grundfunktionen des Readers und dessen angebotenen Applikationen funktionierten von Anfang an einwandfrei. Mit Hilfe der mitgelieferten Software war es sogar möglich, eine zuvor definierte Internetseite mittels Tagerkennung aufzurufen. D.h. der EPC eines Tags wird auf verschiedenste Weise mit dem Internet verarbeitet. Er fungiert entweder als Weblink, Key oder sogar als Google Suchergebnis.

RFID ME ist ein zuverlässiger und leicht zu bedienender USB UHF Reader und ist somit


Abbildung 2.10: RFID ME von RF-IT-solutions

am Markt sehr beliebt. Die maximale Ausgangsleistung des Readers beträgt 20 dBm (0,1 W). Mit der integrierten Dipolantenne liegt somit die maximale Lesedistanz typischer UHF Tags bei 0.5 bis einem Meter.

RFID ME ist ein typischer RFID-Reader, dessen Funktionsblöcke folgenden Komponenten endsprechen. In Abbildung 2.10 ist das Innenleben des Readers dargestellt. Der hervorgehobene bzw. gezoomte Bereich zeigt den Mirkrokontroller C8051F340 (rechts) und den UHF Chip AS3992 (links), die über ein Port (sowohl C8051F340 als auch AS3992 verwenden Port 1) verbunden sind. Die restlichen Peripherien entsprechen der Spannungsstabilisation und den Chipbeschaltungen wie zum Beispiel der Oszillatorschaltung. Das analoge Frontend des Readers ist fast zur Gänze im UHF Chip AS3992 von *austria microsystems* implementiert. Zwischen der originalen Antenne (in Abbildung 2.10 dargestellter Dipol) und dem AS3992 sind nur mehr das Anpassungsnetzwerk für die Antenne und diverse Filter auf der Printplatte vorhanden.

### 2.4.1 UHF Chip AS3992

Der Kern des RFID ME UHF USB-Readers besteht aus dem von *austria microsystems* entwickelten UHF Reader Chip AS3992. Der AS3992 UHF Gen2 Reader Chip bietet neben einem integrierten analogen Frontend, wie bereits erwähnt, die komplette Abwicklung des ISO 180006c/b Protokolls. Der Chip ist in jedem Reichweitebereich (*short*− oder *wide* −

*range*) einsetzbar, da verschiedene dafür benötigte Programmieroptionen vorhanden sind. Generell können fast alle Reader Parameter bzw. die kompletten Reader-Konfigurationen über die entsprechenden Control-Register des Chips eingestellt werden.

Die wichtigsten Eckdaten aus dem Datenblatt [14] sind:

Der AS3992 Chip ist zu seinen Vorgängern Pin- und Firmware kompatibel. Zusätzlich verfügt der Chip gegenüber seinen Vorgängern eine verbesserte Sensitivität von maximal -86 dB, die wiederum im entsprechenden Register einstellbar ist. Die Sendereinheit des Chips erzeugt eine maximale Ausgangsleistung von 20 dBm bei 50 Ohm. Ein integrierter Spannungsregler stellt eine stabile Spannungsversorgung für das ganze Readersystem zur Verfügung. Eine Spannungregulierung zwischen 4,1 V bis 5,5 V ist nicht notwendig, und somit kann der Chip also auch ohne weiters per USB versorgt werden.

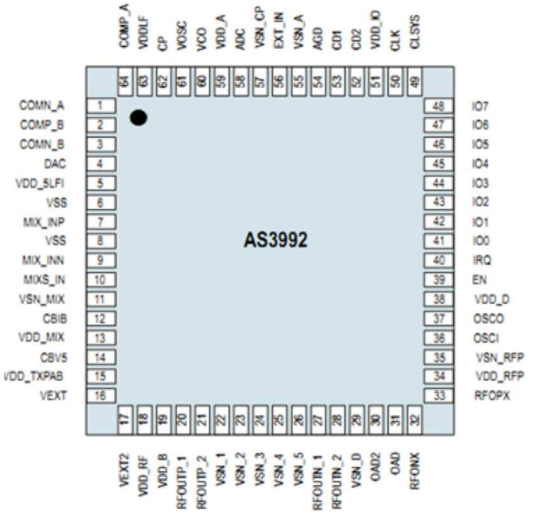


Abbildung 2.11: Pinbelegung AS3992

**Kommunikation [14]**
Als Kommunikationsinterface zwischen AS3992 und einem Host System (zum Beispiel Mikrocontroller) steht sowohl eine serielle, als auch eine parallele Ausführung eines Ports zur Auswahl. Es besteht auch die Möglichkeit zur direkten Übertragung, wobei hier Coder und Decoder ausgelassen werden und ein Zugriff des Hostsystems auf das analoge Frontend in Echtzeit gewährleistet wird.

Als einstellbare Schnittstelle stehen im wesentlichen 10 Pins zur Verfügung, die unterschiedlich konfiguriert werden können. Entweder als Parallel- oder als Serial Peripheral Interface (SPI) Bus. Bei Aktivierung (Power-Up) des Chips bzw. Übergang von Low auf High des Enable Pins (EN), überprüft der AS3992 die Zustände der zehn Interface-Pins.

Sind alle auf Low, wird der Parallel-Mode verwendet. Soll die serielle Variante als Kommunikationsinterface ausgewählt werden, müssen die unbenützten Pins IO1 auf VDD und IO0 auf GND gelegt werden. Die restlichen I/O Pins und deren genaue Funktion im jeweiligen Betrieb werden hier nicht weiter angeführt und können dem AS3992 Datenblatt von austria microsystems entnommen werden.

Eine Kommunikation besteht immer aus einer Startbedingung und darauf folgenden Adressen und Command-Wörtern. Da die Kommandos 8 Bit breit sind, eignet sich als Host System auch ein Low-cost 8 Bit Mikrocontroller, wie z.B. der C8051 von Silabs, bestens. Eine passende Stopbedingung beendet eine Übertragung bzw. Kommunikation. Der AS3992 bietet drei verschiedene Kommunikations-Moden.

- *Continuousadressmode*
- $Non - continuousadressmode$
- *Commandmode*

| Bit | Beschreibung | Bit Funktion | Adress | Command |
|---|---|---|---|---|
| 7 | Command control bit | 0=Address,1=Command | 0 | 1 |
| 6 | Read/Write | 1=Read,0=Write | R/W | Not used |
| 5 | Continuous address mode | 1=Cont,0=Non-cont mode | Cont | Not used |
| 4 | Address/Commandbit 4 | | Adr 4 | Cmd 4 |
| 3 | Address/Commandbit 3 | | Adr 3 | Cmd 3 |
| 2 | Address/Commandbit 2 | | Adr 2 | Cmd 2 |
| 1 | Address/Commandbit 1 | | Adr 1 | Cmd 1 |
| 0 | Address/Commandbit 0 | | Adr 0 | Cmd 0 |

Tabelle 2.1: Bitbedeutung von Adress und Commandword [14]

Aus den letzten zwei Spalten der Tabelle sind die unterschiedlichen Bedeutungen von Bit0 bis Bit6 des Adress- und Command Wortes ersichtlich. Bit 7 (MSB) dient als Auswahlbit und unterscheidet zwischen Adress- und Command-Wort. Nach jedem Adressbefehl kommen Daten. Hier unterscheiden sich die erwähnten Kommunikationsmöglichkeiten folgendermaßen:

*Continous adress mode*
Dieser Mode wird mit Bit 5 aktiviert (Cont = 1). Hier wird das erste Datenbyte auf die gewünschte Adresse geschrieben bzw. von der Adresse gelesen. Die Read und Write Auswahl erfolgt durch Bit 6 (siehe Tabelle). Anschließend werden bis zur Stop-Bedingung Daten mit periodisch erhöhter Adresse ausgelesen.

| Start | Adrc x | Data(x) | Data(x+1) | Data(x+2) | ... | Data(x+n) | StopCont |
|---|---|---|---|---|---|---|---|

Dieser Modus wird vor allem für das Schreiben von Control-Register Bereichen verwendet. Hierbei können in einem einzigen Stream zum Beispiel vordefinierte Werte bzw. ein Set-up der Control-Register des AS3992 von einem Host System (Mikrocontroller) gesendet werden.

*Non-continuous adress mode*
Beim $non - continuous\,adress\,Mode$ wird ein Paar bestehend aus Adresse und Datum immer einzeln behandelt, sprich nach jeder Adresse folgt das entsprechende Datum. Wird mehr als ein Byte gelesen bzw. geschrieben, sollte jedoch der continous adress mode verwendet werden.

| Start | Adr y | Data(x) | Adr y | Data(y) | ... | Adr z | Data(z) | StopSgl |
|---|---|---|---|---|---|---|---|---|

Beide Kommunikationsmöglichkeiten werden für des Schreiben oder Lesen von Konfigurationsregistern oder des FIFOs verwendet.

*Command mode*
Der *command mode* wird für die direkte Steuerung des Readers verwendet. Auf diese Weise werden Befehle wie ”initialize transmission” übertragen.

| Start | Cmd x | Cmd y | ... | StopSgl |
|---|---|---|---|---|

Es sind auch Kombinationen der unterschiedlichen Moden möglich. Nach einem non-continuous adress Mode können alle anderen Moden verwendet werden. Der Continous adress mode muss jedoch immer mit der Stop-Bedingung beendet werden. Ein Wechsel zu einem anderen Mode ist hier nicht möglich. Details sind im Datenblatt vorhanden.

### 2.4.2 Mikrocontroller C8051F340

Aus Abbildung 2.10 ist ersichtlich, dass der AS3992 UHF Chip über einen Port mit einem C8051F340 Mikrocontroller parallel verbunden ist. Der Mikrocontroller steuert im Allgemeinen den UHF Chip und regelt die Kommunikation. Da im AS3992 UHF Chip die komplette Abwicklung des ISO 180006c/b Protokolls integriert ist, stellt der C8051 im Wesentlichen das Interface zu einem Hostsystem dar. Die im Mikrocontroller implementierte USB-2.0-Schnittstelle-(full-speed) eignet sich ideal für eine solche Verbindung mit dem PC. Den Kern des Mikrocontrollers stellt eine high-speed 8051 CPU (25-48

MIPS) mit einem 64 kB großen Flash-Speicher dar. Weitere wichtige Kenndaten sind der im Chip integrierte UART, SMBU, SPI-Bus, Timer, die Zähler und PWM Generatoren. Einige der analogen Features, die im Mikrocontroller inkludiert sind, sind ein Mehrkanal 10-bit ADC, ein interner Oszillator, oder der Temperatursensor.

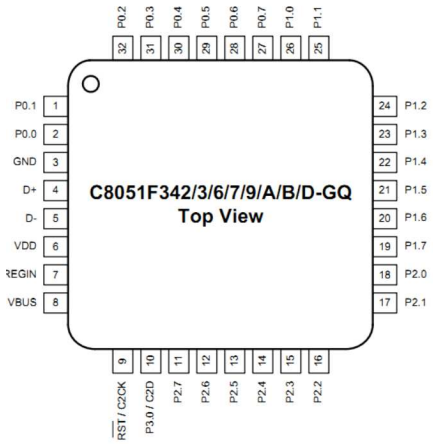


Abbildung 2.12: Pinbelegung C8051F340 [8]

### 2.4.3 Analyse und Modifizierung des analogen Frontends

Um das Gerät vom normalen Reader zu einem Reader, der OCA-Tags lesen kann, aufrüsten zu können, musste dieser mit der im Kapitel zuvor vorgestellten Antenne ausgestattet werden. Da keine spezifische Information (Datenblätter etc.) über das Gerät vorhanden war, wurde der Reader zerlegt und analysiert. In den ersten Schritten wurde das analoge Frontend des Readers genau unter die Lupe genommen. Es galt herauszufinden, welche Ausgangsimpedanz der Reader besitzt, um ihn mit der entsprechende Last zu beschalten. Die kleinen Nahfeld-Antennen konnten nicht einfach mit dem im Reader integrierten Dipol ausgetauscht werden, da sonst bei eventueller Fehlanpassung die Ausgangstufe des Readers gefährdet werden könnte.
Die Analyse des analogen Frontends und vor allem das Ausmessen des Dipols ergaben, dass die integrierte Antenne eine Impedanz von 50 Ω aufweist. Somit beträgt natürlich auch die Impedanz der Ausgangstufe 50 Ω, um eine ideale Anpassung zu gewährleisten. Wichtig war es, den Dipol nicht von der Masse bzw. den Verbindungen des restlichen Readers zu trennen, da diese die Impedanz der Antenne stark beeinflussen.

Abbildung 2.13 stellt das Smithdiagramm des originalen Readers und dessen Antenne dar. Man erkennt, dass diese, wie erwartet, im UHF Bereich sehr gut angepasst ist. Bei 865 MHz entspricht die Impedanz der Antenne 50 Ω und ist gleich der Impedanz der

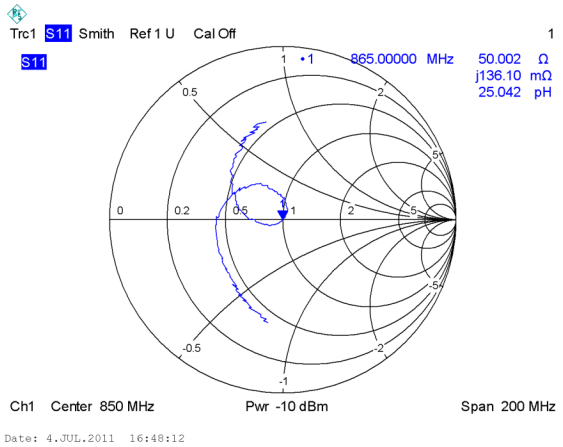


Abbildung 2.13: Smithdiagramm des integrierten Dipols

Ausgangsstufe des Readers. Nach erfolgreicher Anpassung der oben angeführten Antenne konnte diese nun ohne Bedenken ausgetauscht und getestet werden.

### 2.4.4 Erster Kommunikationstest

Zu Beginn der Arbeit wurde hauptsächlich die vorhandene bzw. mitgelieferte Software verwendet. Diese erwies sich als sehr praktisch. Mit Hilfe des USB-Readers und der kleinen UHF-Antenne konnten so die ersten erfolgreichen Kommunikationsversuche mit einem OCA-Tag aufgebaut werden. Die Software des RFID ME ermöglichte bereits jetzt das Auslesen des EPC. Nach mehreren Tests ergab sich eine maximale Lesedistanz von einem halben Zentimeter. Die Ausgangsleistung betrug hierbei immer 20 dBm. Der erste Kommunikationstest ist in Abbildung 2.14 dargestellt.

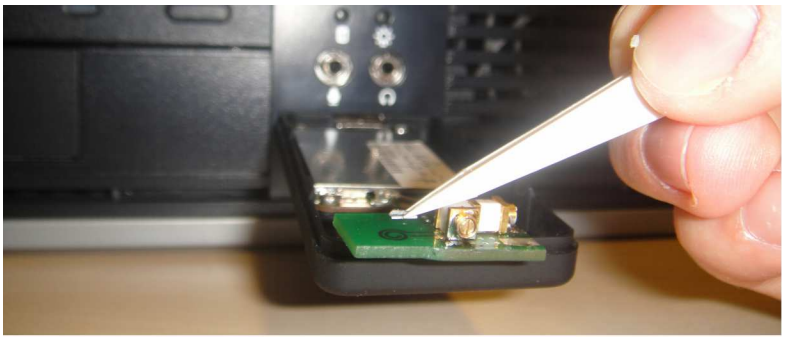

Abbildung 2.14: erster Kommunikationstest

## 2.5 ISP (In-System-Programming) Hardware Konfiguration

Um Zugriff auf den Flashspeicher des Mikrocontrollers zu erlangen, gibt es verschiedene Möglichkeiten (z.B. über Bootloader). Der für das Demoboard (Silicon Laboratories C8051F340 DK) zur Verfügung stehende USB-Debug-Adapter bietet alle Vorteile der "In-System-Programmierung". Somit ist eine direkte Kommunikation zwischen dem USB-Port eines Computers und dem angeschlossenen Mikrocontroller möglich, obwohl sich der Mikrocontroller bereits in einem System befindet und eventuell sogar arbeitet. D.h. sämtliche Peripherie (in Falle der Masterarbeit, die Peripherie des Readers) ist bereits am Mikrocontroller angeschlossen.


Abbildung 2.15: Silabs Debug Adapter

Der USB-Debug-Adapter verfügt über beide von Silicon Laboratories entwickelten Schnittstellen, JTAG und C2. Die Energieversorgung erfolgt über USB. Dadurch ist auch die Versorgung eines Mikrokontrollers und dessen Peripherie bis zu 100 mA möglich. Die Pin-Beschreibung des Verbindungskabels ist in Tabelle 3.1 aufgelistet.

| Pin Nummer | Beschreibung |
|---|---|
| 1,8 | Nicht Angeschlossen |
| 2,3,9 | **GND (Ground)** |
| 4 | **TCK (C2D)** |
| 5 | TMS |
| 6 | TDO |
| 7 | **TDI (C2CK)** |
| 10 | USB Power |

Tabelle 2.2: USB Debug Adapter Pin-Beschreibung [10]

Beim C2-Interface handelt es sich um ein eigens von Silicon Labs entwickeltes, serielles 2-Draht Kommunikationsprotokoll. Ein C8051F34x Mikrocontroller beinhaltet ein solches Debug Interface und muss daher nur an den beiden Eingängen C2D und C2CK angeschlossen werden. Diese entsprechen den Pins 4 und 7 des USB Adapters. Zusätzlich sollten beim Anschließen natürlich auch die Massen verbunden werden, um einen sicheren

Debug Betrieb zu gewährleisten.

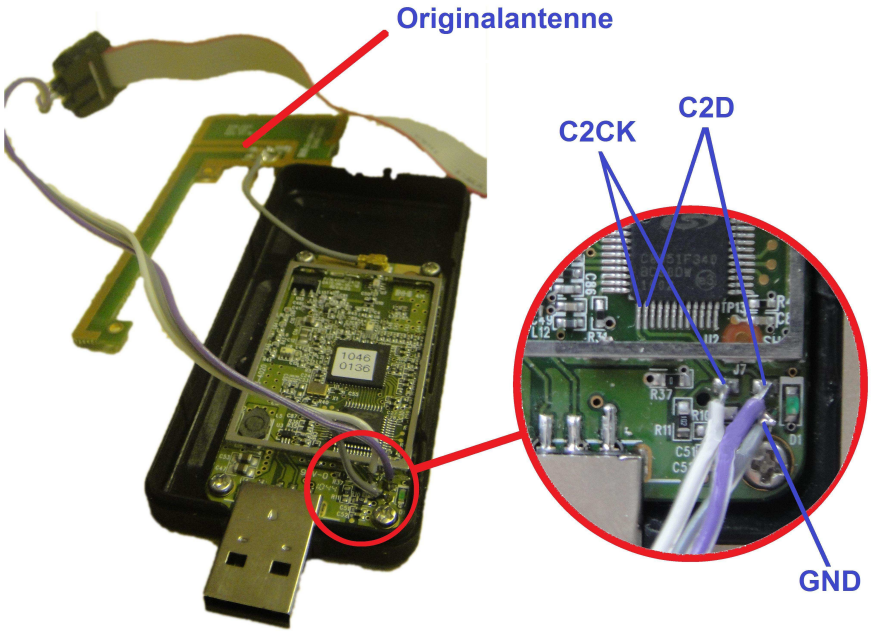


Abbildung 2.16: Debug Adapter Beschaltung

Die Abbildung 2.16 zeigt den Hardwareeingriff. Wie schon beschrieben, kann der Mikrocontroller C8051F340, obwohl er in einem System implementiert ist, durch die C2-Schnittstelle programmiert und vollständig überwacht werden. Im Falle des UHF-Readers (RFID ME) sind zufällig die benötigten Signale vom Mikrocontroller am ersten Layer der Printplatte auf größere Pads herausgeführt. Hier wurde, wie die Abbildung 2.16 zeigt, die C2-Schnittstelle des USB Adapters angeschlossen. Mit diesem Eingriff ist man nun in der Lage den Reader bzw. den Mikrocontroller umzuprogrammieren. Das auf den Mikrocontroller geschriebene Programm kann durch die C2 Schnittstelle auch Schritt für Schritt verifiziert werden.

Für den eigentlichen Programmiervorgang wurden die vom C8051F340 Demoboard zur Verfügung gestellten Tools verwendet. Dabei wurden vor allem die Programmieroberflächen SilabsIDE und *µVision* in Anspruch genommen. Bei dem dort eingestellten bzw. verwendeten C8051 Compiler handelte es sich um Keil. Mit dem Demoboard von Silicon Labs wurde auch eine Software, die ein direktes Beschreiben des Flashspeichers ermöglicht, mitgeliefert. Mit Hilfe der Software "Flash Programmer" war somit auch das Programmieren des Mikrocontrollers mit einem fertigkompilierten hex-File möglich.

# Kapitel 3: Entwicklung und Aufbau der Systemsoftware

Die Kommunikation vom eigentlichen RFID-Tag bis hin zur Graphischen Sensorauswertung verläuft bei der im Kapitel zuvor vorgestellten Hardware über mehrere Layer hinweg. Ähnlich wie beim bekannten OSI-Modell, durchlaufen in dieser Anwendung Informationen und Befehle vom Tag zur Auswertungssoftware und wieder zurück mehrere Instanzen. Jeder der in Abbildung 3.1 dargestellten Knoten verfügt über eine eigene Art von Datenverarbeitung und passt diese an die gegebenen Schnittstellen an.

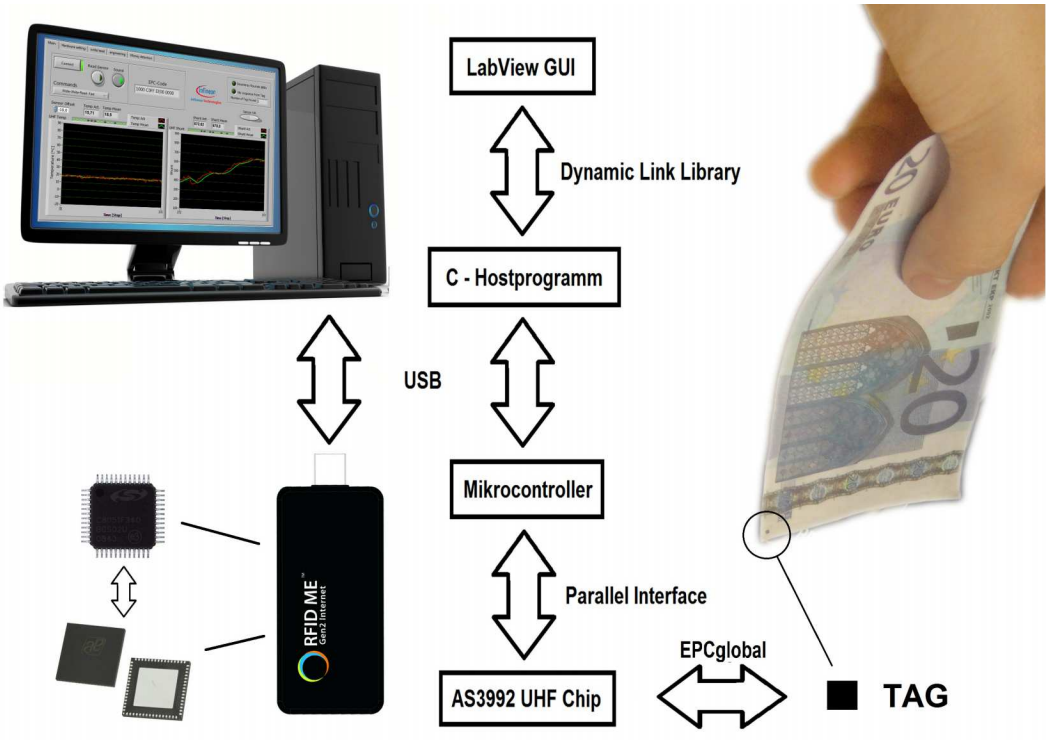


Abbildung 3.1: Systemübersicht

Je höher die Instanz, desto detaillierter die Datendarstellung und Befehlseingabe. Während der Entwicklung wurde das System immer um eine Ebene erweitert. Auf der niedrigsten Ebene befindet sich die Kommunikation zwischen RFID-Tag und UHF-Reader. Anschließend gelangen Befehle und Informationen per USB-Übertragung zum C-Hostprogramm. In den ersten Prototypen wurden hier die Daten mittels einer Konsolenanwendung ausgewertet. Die Konsolenanwendung ermöglichte auch die erste Befehlseingabe. Um die geforderte benutzerfreundliche Datenauswertung und Befehlseingabe zu ermöglichen, wurde schlussendlich das C-Hostprogramm zu einer DLL (Dynamik Link Library) umgerüstet und in das LabVIEW-Programm integriert.

## 3.1 Mikrocontroller

Das Mikrocontrollerprogramm fungiert als Schnittstelle zwischen dem UHF-Chip AS3992 des USB-RFID-Readers und dem PC mit der Auswertungssoftware. Je nach einkommendem Befehl steuert der Mikrocontroller den UHF Chip AS3992. Dabei werden die Befehle und Daten von der USB-Schnittstelle verarbeitet und dem AS3992 Chip zur Verfügung gestellt. Eine ReportID kennzeichnet den entprechenden Befehl vom C-Hostprogramm. Auch die Antwort des UHF-Chips wird vom Mikrocontroller mit einer ReportID gekennzeichnet und über USB zum Hostprogramm gesendet. So kann zwischen den Antworten unterschieden und das erfolgreiche Ausführen des Befehls überprüft werden. Der unter Abbildung 2.10 hervorgehobene Bereich zeigt, dass der Mikrocontroller mit dem UHF-Chip über einen Port verbunden ist. Bei beiden Ports des Chips handelt es sich um Port1. Somit wird der Befehl nach der Verarbeitung dem AS3992 parallel übermittelt. Die Kommunikation erfolgt wie im Punkt 3.4 beschrieben. Befehlserweiterungen bzw. Änderungen können in der *usb_commands.c* und *usb_commands.h* Datei durchgeführt werden. Hier ist auch der spezielle write-write-read Befehl implementiert.

### 3.1.1 Programmablauf

Der Ablauf des Programms wird im folgenden Flussdiagram veranschaulicht. Mit dem Start bzw. dem Anschließen des Readers über eine USB Schnittstelle wird auch die Initalisierung von USB durchgeführt. Anschließend führt der Mikrokontroller einen Funktionstest durch. Nach dem beschriebenen Startvorgang befindet sich der Mikrocontroller in einem Polling Status. Sobald ein USB Paket angekommen ist, wird usbCommands() durchgeführt. Hier wird aufgrund der ReportID die entsprechende Call-Funktion des ausgewählten Befehls aufgerufen. Nach dem Durchführen dieser Funktion ist der Mikrocontroller wieder im Pollig Status. Das Programm wurde an einen Beispielcode des UHF-Chips AS3992 angelehnt.

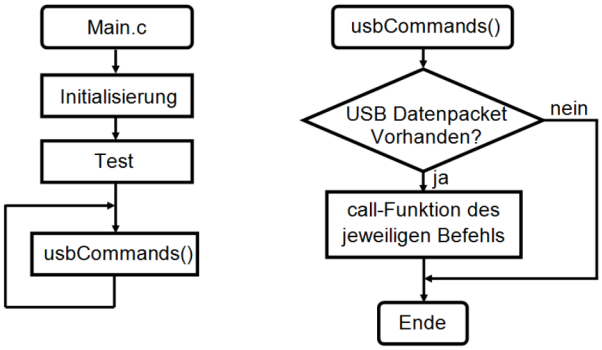


Abbildung 3.2: Mikrocontroller Flussdiagram

### 3.1.2 Befehlsformat zur darüberstehenden Instanz

Die Befehle und deren ReportID werden in Tabelle 3.1 zusammengefasst. Die Antwort der Befehle entspricht der ReportID des Befehls + 1. Sowohl Befehl als auch dessen Antwort besteht aus der in der Tabelle dargestellten Byteabfolge. Die zusätzlichen Daten werden byteweise nach ReportID und Framelength angefügt. Wie der Namen Framelength vermuten lässt, legt dieser die Länge bzw. Größe der angefügten Daten durch dessen numerischen Wert fest.

| Befehl | ReportID | FrameLength | Datenbeschreibung |
|---|---|---|---|
| Antennen Power | 24 | 3 Bytes | Energie ein/aus |
| writeRegister | 26 | 6 Bytes | Register Adresse + Daten |
| Antwort writeRegister | 27 | 3 Bytes | Schreibfehlererkennung |
| Inventory | 49 | 3 Bytes | Tag information |
| Antwort Inventory | 50 | variabel | Länge des EPC |
| write | 53 | variabel | Adresse, Bank und Daten |
| Antwort write | 54 | 4 Bytes | Anzahl der geschriebenen Words |
| read | 55 | 5 Bytes | Adresse, Bank und Anzahl |
| Antwort read | 56 | variabel | Anz. gelesenen Bytes |
| write-write-read | 73 | 13 Bytes | Sensorauswahl |
| Antwort w-w-r | 74 | 12 Bytes | Sensordaten |
| Gen2config | 89 | 14 Bytes | Gen2 Daten zum Einstellen |
| Antwort Gen2config | 90 | 14 Bytes | Gen2 Daten eingestellt |

Tabelle 3.1: uC-Befehlsformat

## 3.2 C-Hostprogramm

Sowohl beim Mikrocontrollerprogram als auch beim C-Hostprogramm handelt es sich um eine Datenkonvertierung. Hierbei wird der USB-RFID-Readers mit der Auswertungssoftware des PCs, in unserem Fall LabVIEW, verbunden. Prinzipiell könnte auch in dieser Ebene die graphische Auswertungs- und Steueroberfläche des USB Reades implementiert werden, jedoch wird hier auf die Vorteile von LabVIEW zurückgegriffen. LabVIEW stellt in diesem Fall eine leicht erweiterbare, benutzerfreundliche, graphische Eingabeoberfläche zur Verfügung, welche zusätzlich die Sensorwerte des OCA Tags auf verschiedenste Weise verarbeiten kann. Daher werden im C-Hostprogramm lediglich alle Befehle und Daten hin und vom USB Reader konvertiert und zur nächsten Instanz weitergereicht.

Die erste Schnittstelle im C-Programm stellt die Verbindung zum Mikrocontroller des USB-Readers dar. Hier wurde auf API-Funktionen zurückgegriffen, die nach Auswahl der dementsprechenden USB Geräte Klasse verwendet wurden.

Um die Daten und Signale dem LabVIEW-Programm weiterzureichen, wurde das C-Programm als DLL umgewandelt. Diese DLL kann im LabVIEW eingebunden und als Programmteil verwendet werden. Dabei wurden aufgrund möglicher Erweiterungen mehrere Ein- bzw. Ausgangsparameter definiert, die als zweite Schnittstelle des C-Programms fungieren.

### 3.2.1 USB-Kommunikation zum Reader

USB (Universal Serial Bus) ist, wie weitgehend bekannt, eine der modernsten Kommunikationsmöglichkeiten zwischen einem PC und einem externen Gerät. Nicht geläufig ist nun, dass alle USB-Geräte beim ”Startup”, d.h. wenn das Gerät angeschlossen wird, unterschiedlich konfiguriert werden. Jedes Gerät wird in eine, mit unterschiedlichen Protokollen und Funktionen, definierte Geräteklasse unterteilt. Beispiele für solche Geräteklassen werden in Tabelle 3.2 aufgezeigt. [4]

| Geräte Klasse | Beispiel |
|---|---|
| Display Device | Monitor |
| Communication Device | Modem |
| Audio Device | Lautsprecher |
| Mass storage Device | Festplatte |
| Human interface Device | Maus, Tastatur |

Tabelle 3.2: USB-Geräte Klassen

Das Human Interface Device (HID) kommuniziert mit dem PC über definierte und strukturierte Nachrichten. Diese Klasse beinhaltet vor allem, wie schon der Name HID vermuten lässt, Geräte, die mit dem Benutzer direkt interagieren. Dazu gehören zum Beispiel Tastaturen, Mäuse, einzelne Eingabegeräte wie Knöpfe oder andere Möglichkeiten zur Steuerung eines Computers. Unabhängig vom Namen und der Spezifikation sind jedoch auch nichtinteraktive Geräte enthalten, deren Datenformat mit den oben angeführten Geräten weitgehend übereinstimmt. Sensoren, Messsysteme und auch Strichcodelesegeräte fallen in diese Kategorie. Es ist ersichtlich, dass auch ein RFID-Reader dieser Geräteklasse zugeordnet werden kann. Diese Geräte verfügen zwar über kein "Human interface", stimmen jedoch mit der HID-Spezifikationen [4] überein.

Grundsätzlich unterscheiden sich die Klassen in den Datentransportanforderungen. Audioübertragungen zum Beispiel, müssen isochron, d.h. zeitgleich übertragen werden. HID Datentransport ist hierbei wesentlich einfacher und wird in folgenden Absatz beschrieben.

Man unterscheidet zwischen HID Device (Gerät) und HID Driver (z.B. ein Personal Computer). Diese zwei Komponenten sind miteinander über zwei Leitungen, den sogenannten Pipes, verbunden. Bei HID erfolgt der Datentransport ausschließlich über control- und interrupt pipe-Transfer. [4]

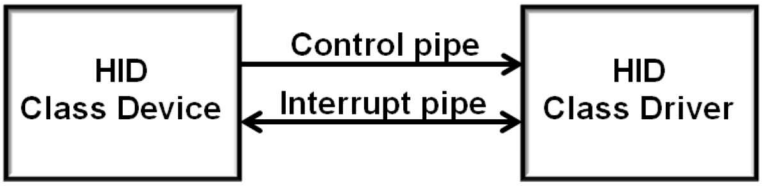


Abbildung 3.3: HID Verbindung

Aufgaben der Control-Pipe:

- Anforderung und Empfang der Klassendaten
- Senden von Daten, wenn von HID Class Driver angefordert (z.B. mit Get-Report)
- Empfangen der Daten vom Host

Aufgaben der Interrupt Pipe:

- Empfängt asynchron, d.h. ohne Anforderung, regelmäßig Daten vom Gerät
- Sendet Daten mit geringer Wartezeit zum Gerät

Information bzw. Daten werden in Form von "Reports" gesendet. Dabei handelt es sich um eine Art Datenpacket, das in einem Vorgang gesendet wird. Diese Report können bei der

HID-Klasse nun entweder über die Control Pipe oder die Interrupt Pipe gesendet werden. Interrupt Übertragungen werden von Geräten, wie z.B. einer Tastatur, verwendet, die eine garantierte und schnelle Antwort vom Computer benötigen. Schnelle, kleine Befehle vom Computer zum Gerät werden typischerweise per Controll-Übertragung durchgeführt. Bei einem USB-System, das mehrere Report-Strukturen aufweist, müssen die Reports mittels einer ReportID unterschieden werden. Kapitel 7 der HID USB Spezifikation (Device Class Definition for Human Interface Devices HID Version 1.11) beschreibt die Standard- und klassenspezifischen Abfragen, die zur USB-Übertragung zwischen HID Geräten und somit auch den UHF-RFID-Reader und PC nötig sind. [4]

Unter dem Betriebssystem Windows ist es nicht nötig einen eigenen HID-Treiber zu schreiben, da man die bereits im Betriebssystem integrierten Treiber verwenden kann. Windows stellt eine Reihe verschiedenster sogenannten API Funktionen zu Verfügung. Sie sind im Betriebssystem implementiert und gelten als Schnittstelle bzw. Verbindung zwischen Hardware und Programmierer.

Jede Programmierumgebung kann diese von Windows zur Verfügung gestellten API-Funktionen verwenden. Solange das USB-Gerät den HID-Kriterien entspricht, kann man die unter Microsofts WDK (Windows Driver Kit) vorhandenen Dokumente und Entwicklungswerkzeuge nutzen. Hier sind auch die in der Masterarbeit verwendeten Headerfiles (hidsdi.h, hidusage.h, hidpi.h) zu finden. Diese ermöglichen den Zugriff auf HID Geräte mit der Programmierumgebung Visual C++. [13]

### 3.2.2 Erweiterung zur Dynamik Link Library

Eine DLL ist, wie auch die bekannte EXE, eine ausführbare Datei. Das Dynamic Linking wird verwendet, um eine Verbindung mit einer Anwendung (in unserem Falle das C-Programm) herzustellen. Dabei bleiben die DLLs in ihren Dateien, d.h. sie werden nicht in die ausführbare Anwendung (z.B. das LabVIEW-Programm) kopiert. Das Umrüsten des C-Quellcodes einer normalen Anwendung zur DLL ist in wenigen Schritten erklärt. Wie auch beim C-Hostprogramm des Readers wurde hier die Main-Funktion durch die auszuführende DLL-Funktion ersetzt. Diese muss mit $\_\_declspec(dllexport)$ eigens exportiert und gekennzeichnet werden. Die Exportfunktion ist eine Microsoft-spezifische Erweiterung der Programmiersprache C und C++. Beim Einbinden der DLL in LabVIEW sollte auch die Headerdatei "extcode.h" angegeben werden, da diese bei komplexen DLLs von LabVIEW vorausgesetzt wird. Natürlich müssen auch die entsprechenden Kompiliereigenschaften eingestellt werden. Ein Beispiel für das Erstellen einer DLL für LabVIEW ist unter [1] zu finden.

### 3.2.3 Befehlsformat zur darüberstehenden Instanz

Das Datenformat der Befehle und deren Antwort zwischen LabVIEW und C-Program, ist folgendermaßen aufgebaut.

Herzstück des C-Programms ist eine Case-Anweisung, die je nach ReportID die unterschiedlichen, sprich ausgewählten Befehle, ausführt. Der Maximalwert der ReportID beträgt 255. Die Bedeutung der ReportID bzw. dessen gleichzusetzender Befehl wird in Tabelle 3.3 zusammengefasst. Bei positiver Ausführung des mit der ReportID ausgewählten Befehls sind zusätzlich die dementsprechenden Rückgabewerte der DLL in dieser Tabelle aufgelistet.

| ReportID | Befehl | Rückgabewert positiv | Rückgabewert negativ |
|---|---|---|---|
| >255 | error | -1 | |
| 1 | Inventory | 1 | -1 |
| 2 | Gen2config | 2 | -2 |
| 3 | Antennen Power | 3 | -3 |
| 4 | writeRegister | 4 | -4 |
| 5 | read | 5 | -5 |
| 6 | write | 6 | -6 |
| 7 | write-write-read | 7 | -7 |
| 8 | Fast WWR | 8 | -8 |
| sonst | DLL-Output = 0 | 99 | |

Tabelle 3.3: C-Befehlsformat

Den mit der ReportID ausgewählten Befehlen werden alle benötigten Daten übergeben. In deren Unterprogrammaufruf wird anschließend auf die Antwort und die Daten des Readers gewartet.

## 3.3 LabVIEW

Das entwickelte Programm GUI 1.1 dient hauptsächlich als Graphisches User Interface. Hier werden die Befehle definiert und Hardwareeinstellungen vorgenommen. Weiters werden hier die Daten ausgewertet und dargestellt. Eine der wichtigsten Anforderungen dieser Software war es, die benutzerfreundliche Bedienung des UHF-RFID-Systems zu gewährleisten. Der User soll in der Lage sein, ohne viel Aufwand das System in Betrieb zu nehmen und auf einfachste Weise entsprechende Messungen oder andere Aufgaben durchzuführen. Auf die speziellen Befehlsanforderungen und dementsprechend integrierten

Features der Software wird später noch genauer eingegangen. Schlussendlich soll auch der Aufwand zur Erweiterung und Änderung des Programms durch weitere Benutzer minimiert werden.

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) ist eine sehr mächtige, leicht verständliche grafische Programmierumgebung und wird somit den oben erwähnten Anforderungen gerecht. Daher entschied ich mich auch für diese Software von der Firma National Instruments.

"LabVIEW ist eine grafische Programmierumgebung, mit der sich anspruchsvolle Mess-, Prüf-, Steuer- und Regelsysteme entwickeln lassen. Dabei werden intuitive grafische Symbole eingesetzt und miteinander verbunden, so dass eine Art Flussdiagramm entsteht. LabVIEW kann mit hunderten von Messgeräten eingesetzt werden und bietet zahlreiche integrierte Bibliotheken für erweiterte Analyse- und Darstellungsfunktionen zur Erstellung virtueller Messgeräte. Die LabVIEW-Plattform hat sich über die Jahre zum Industriestandard entwickelt und kann mit verschiedenen Ziel- und Betriebssystemen eingesetzt werden."[9]

### 3.3.1 Programmablauf LabVIEW

LabView stellt also den obersten Layer des Datenflusses im UHF-RFID-System dar. Der Kern des Graphischen-User-Interface-Programms, in Abbildung 3.4, ist eine eigens geschriebene DLL-Datei, die als Softwareschnittstelle fungiert. Sie inkludiert den im nächsten Kapitel beschriebenen C-Source-Code, der für den eigentlichen Kommunikationsaufbau zwischen USB-Reader und PC zuständig ist.

Der DLL Funktionsblock umfasst 26 Signalleitungen die als Input oder Output fungieren. Diese gewährleisten die Kommunikation mit dem C-Code, dem 2. Layer des Datenflusses in diesem UHF-RFID-System. Die Signalleitungen können im Wesentlichen auf 10 Inputvariablen plus Header und 10 Outputvariablen plus Header aufgeteilt werden. Weiters beinhalten sie einen Fehlerrückgabewert und zwei Arrays, die sowohl als Input als auch als Output verwendet werden können.

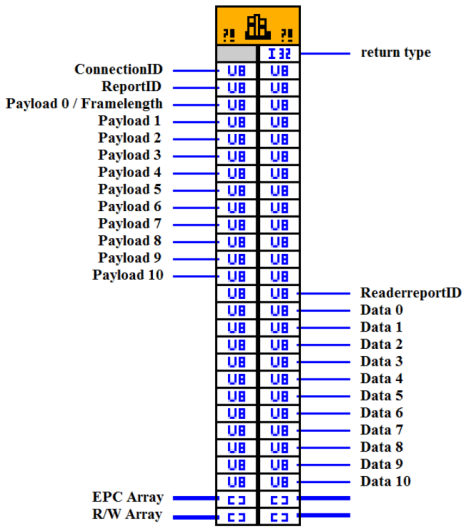


Abbildung 3.4: Call Library Function - Block

#### 3.3.1.1 Initialisierung

Die zwei erwähnten Arrays werden bei Programmaufruf initialisiert, sprich deren benötigte Speicher reserviert. Für das EPC Array wird der Maximalwert von 64 Byte, den ein Electronic Produkt Code heutzutage nicht überschreiten darf, verwendet. Das Read/Write Hilfsarray wird, wie aus Abbildung 3.5 ersichtlich, mit 255 Byte initialisiert

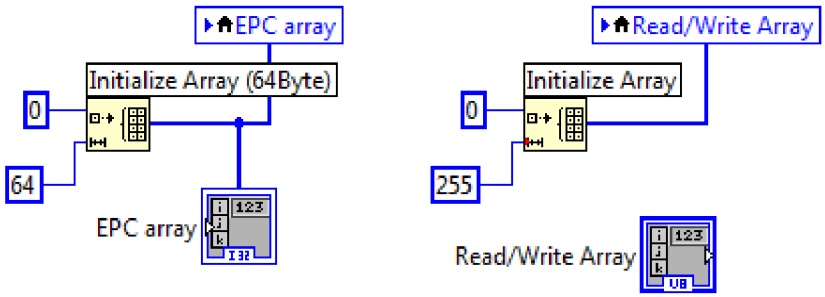


Abbildung 3.5: Speicherreservierung

#### 3.3.1.2 Verbindungsaufbau zum Reader

Nach der Initialisierung wartet das Programm auf den Verbindungsaufbau. Dieser wird im Wesentlichen nur über die ConnectionID der DLL gesteuert. Legt man, wie in Abbildung 3.6 dargestellt, auf die ConnectionID eine ”1” (im LabVIEW Programm per Button), so wird in der DLL der Initialisierungsteil von USB etc. durchgeführt. Details werden im nachfolgenden Kapitel (DLL /C-Source) genau erörtert. Ist schlussendlich die Verbindung mit der Hardware (USB-Reader) aufgebaut, wird der Rückgabewert der DLL verändert. ”2” kennzeichnet einen erfolgreichen Verbindungsaufbau, was auch in Form von einem Popup bzw. Message und der ”Connection LED” bestätigt wird.

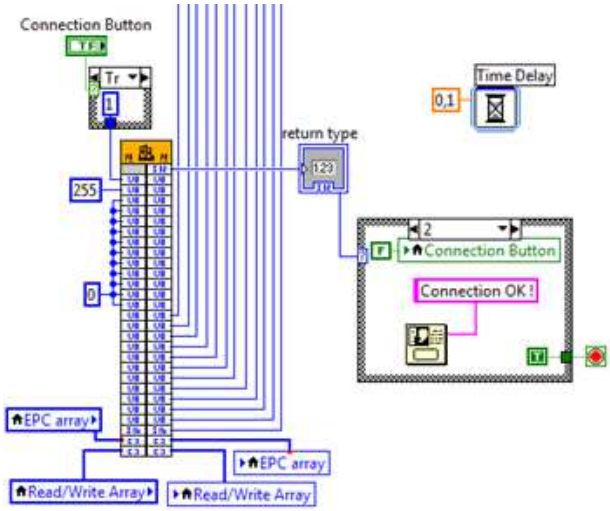


Abbildung 3.6: Aufbau der Verbindung

#### 3.3.1.3 Befehle

Die unterschiedlichen Befehle und somit unterschiedlichen Parameter, die dem Kern des LabVIEW-Programms (die Reader-DLL ) übergeben werden, sind in einem Case-Block zusammengefasst. In Abbildung 3.7 und Abbildung 3.8 sind drei verschiedene Befehle dargestellt. Jeder Befehl hat eine Nummer, eine gewisse ReportID, welche der DLL übergeben wird. So entspricht der Befehl "Inventory" der Zahl 1. Inventory benötigt keinerlei weitere Angaben, somit werden die restlichen Eingabeparameter der DLL auf 0 gesetzt (dies dient nur der Übersichtlichkeit, d.h. das Setzen der restlichen Parameter ist nicht zwingend notwendig). Nach dem Anlegen der Zahl 1 startet die DLL also Inventoryrounds und liefert bei Erfolg den Rückgabewert 50.

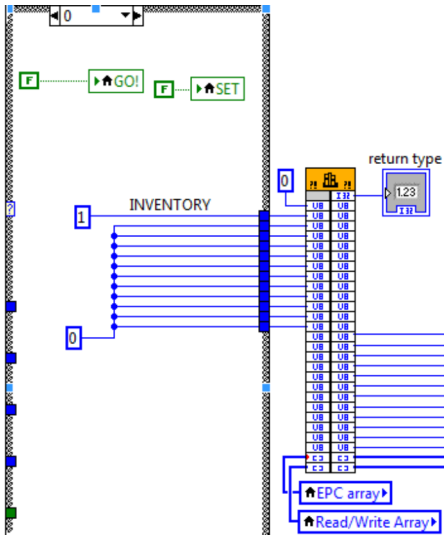


Abbildung 3.7: Inventory

Der nächste wichtige Parameter, der angegeben werden muss, ist die Framelength. Durch diese wird angegeben wieviel weitere Daten gesendet werden. So sind es beim zweiten Befehl (beim 2. Befehl "GEN2config" wird die Zahl 2 als ReportID angegeben) 13 zusätzliche Wörter. Details sind im Kapitel Reader-DLL angegeben. Die weiteren Daten sind im Allgemeinen befehlsorientierte Zusatzangaben bzw. Übergabeparameter, deren Größe je nach Befehl stark variiert. Der Befehl "Write" ist hierbei ein Paradebeispiel. Wie aus Abbildung 3.8 ersichtlich wird als Framelänge neben den Grundeinstellungen (9 Wörter) noch die eingestellte Anzahl der Wörter hinzuaddiert, die auf den RFID-Tag geschrieben werden soll. Wie auch der Befehl Inventory, wird hier der Erfolg des Befehls mit dem entsprechenden Rückgabewert der DLL verifiziert.

Es werden nicht alle Befehle des Grafischen User Interface erörtert, da sich deren Ausführung in LabVIEW kaum unterscheidet. Zusammenfassend ist nochmals zu erwähnen, dass jedem Befehl eine Nummer der sogenannten ReportID zugewiesen wird. Anschließend werden alle anderen Parameter (Anzahl je nach Befehl) der Reader-DLL übergeben.

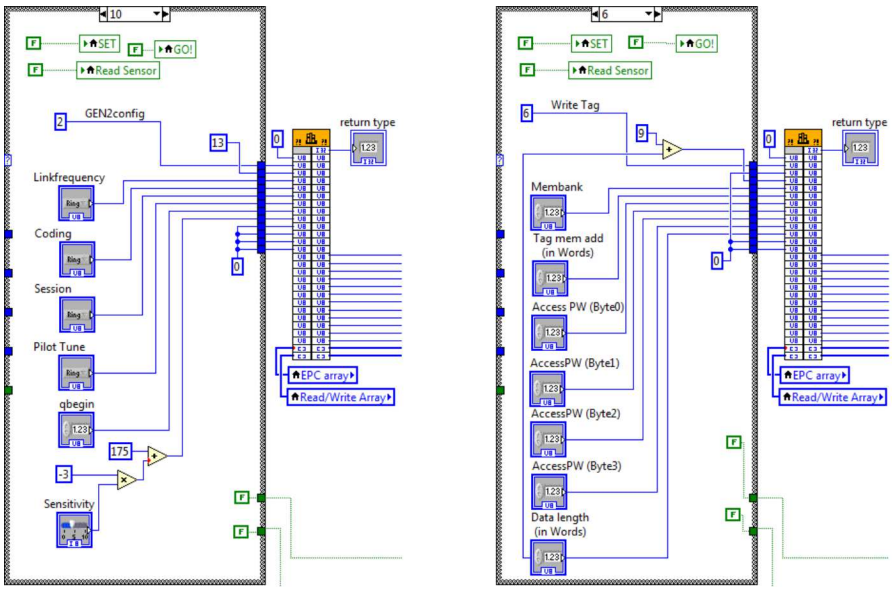


Abbildung 3.8: Hardwaresetting und Write-Befehl

#### 3.3.1.4 Ausgabe/Auswertung

Als Datenquelle wird wiederum die Reader-DLL verwendet, welche wie bereits öfters erwähnt als Schnittstelle zum C-Source fungiert. Wird ein Befehl erfolgreich ausgeführt, so liefert die DLL neben dem Rückgabewert zur Verifikation der Durchführung auch alle vom Befehl angeforderten Daten. (Daten 0 - Daten 10). Ähnlich wie die Kommunikation zwischen LabVIEW und C verhält sich auch die Kommunikation zwischen C und Reader über USB. D.h. auch der Reader antwortet mit einem Rückgabeparameter, der in Form der ReportID in C weitergegeben und somit auch in LabVIEW verfügbar wird. Sind "return type" und ReaderreportID geeignete (d.h. positive) Antworten so können auch die restlichen Daten ausgelesen und durch die zwei erwähnten Rückgabewerte entsprechend zugeordnet werden.
In Abbildung 3.9 ist eine solche Datenauswertung dargestellt. Hier wurde der oben erwähnte Hardwareeinstellungsbefehl (Zahl 2), sprich GEN2config, am Eingang angelegt, welcher anschließend ausgewertet wird. Die gelieferten Daten entsprechen der Antwort des Readers, d.h. dem aktuellen Hardwarestatus. Es ist ersichtlich, dass durch den Befehl GEN2config also nicht nur die Hardware verändert, sondern auch der Status abgerufen werden kann.

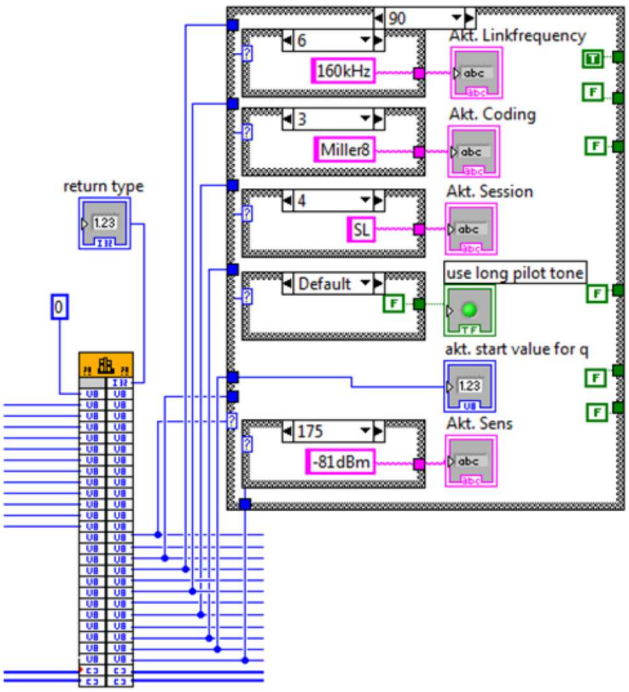


Abbildung 3.9: Auswertung

#### 3.3.1.5 Sensordatenauswertung

Die Sensordaten sind nach dem in Kapitel 2 detailliert beschriebenen Vorgang abrufbar. Nach einem erfolgreich ausgeführten Write-Write-Read-Befehl liegen also die Sensorwerte am Ausgang der DLL vor. Da es sich hierbei nur um die Rohdaten (Bits) des RFID-Tags handelt, müssen diese dementsprechend in Messwerte, wie Temperatur oder Spannung, umgewandelt werden. Sowohl Shunt- als auch Temperatursensor des Transponders arbeiten mit demselben implementierten Analog-Digital-Converter (ADC). Der ADC-Wert steht nach einer abgeschlossenen Messung (diese wurde durch den ersten Write-Befehl initialisiert) mit einer Auflösung von 10 Bit zur Verfügung.

**Temperatursensor**
Der Sensor ist direkt am im Transponder implementierten Analog-Digital-Converter angeschlossen. Die Simulationen und Tests von [18] ergeben folgende ADC-Werte bei veränderter Temperatur:

| Temperatur | ADC Digitalwert |
|---|---|
| $-40°C$ | 280 |
| $0°C$ | 367 |
| $27°C$ | 443 |
| $130°C$ | 756 |

Tabelle 3.4: ADC-Wert Simulation

Der ADC verfügt nur über einen bestimmten analogen Bereich, den er bei einer Auflösung von 10Bit ins Digitale wandeln kann (siehe Abbildung 3.4). Durch den eingestellten ADC-Betriebsmodus (0,3 V Offsetspannung) wird bei der Temperaturmessung ein Wert von 367 als Offset verwendet um einen 0-Abgleich durchzuführen. Durch die Simulation ergibt sich folgende Temperaturauflösung:

$$\frac{756 - 280}{130 + 40} = 2,8\frac{LSB}{°C}$$

D.h. es werden $2,8\,LSB$ pro $°C$ verwendet. Somit wird der ADC-Wert, der nach einem erfolgreichen Write-Write-Read ausgelesen wurde, mit $2,8$ dividiert, um die entsprechende Temperatur zu erhalten. Bevor die Messung jedoch übernommen wird, muss ein zusätzlicher, einstellbarer Sensoroffset dazu addiert werden. Dieser ergibt sich aus den natürlichen Prozessschwankungen der Produktion. Jeder hergestellte Tag besitzt diese Schwankung, die durch diese Einstellung kompensiert wird.

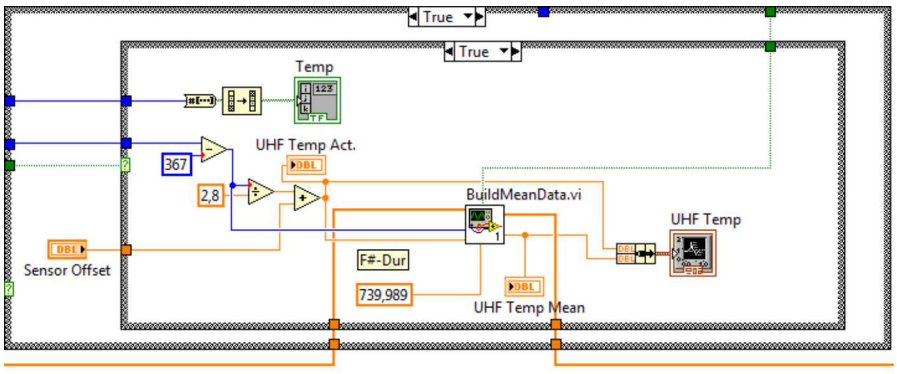


Abbildung 3.10: Temperaturauswertung

**Shuntsensor**
Wiederum wird der vorhandene ADC genutzt, um den Shuntwert mit einer Auflösung von 10Bit zu wandeln. Auch hier haben die Simulationen von [18] Richtwerte zur Auswertung ergeben.

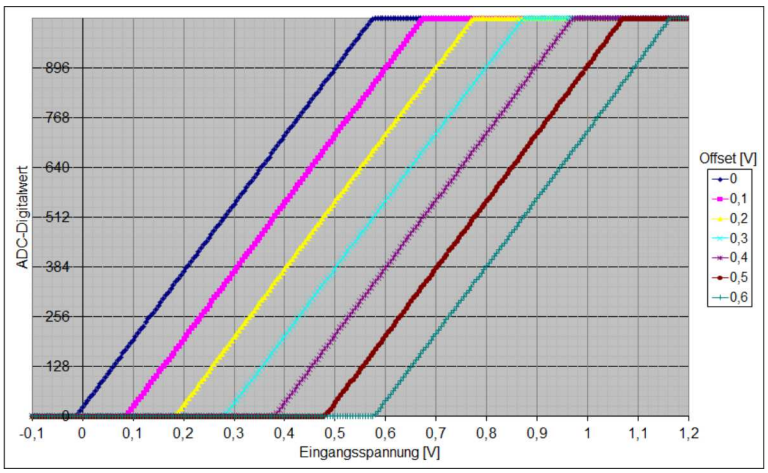


Abbildung 3.11: ADC Simulation

Abbildung 3.11 zeigt die Simulation des im Transponder implementierten Analog-Digital-Converters und dessen Arbeitsbereich. Je nach Eingangsspannung gibt dieser den entsprechenden 10Bit-Digitalwert aus. Der ADC verfügt über einen zusätzlichen Pin, an dem eine Offsetspannung angelegt werden kann, um den Arbeitsbereich zu verändern. In unserem Fall wurde im Transponder die Offsetspannung 0,3 V eingestellt. D.h. Spannungen zwischen 280 mV und 880 mV können gemessen werden. Alle anderen werden als Minimal- bzw. Maximalwert ausgegeben. Der Messbereich des ADCs umfasst also 600 mV bei einer Auflösung von 1024 Quantisierungsstufen.

$$\frac{600}{1024} = 0,586 \frac{mV}{LSB}$$

Wiederum müssen hier beide bereits beschriebenen Offsetabgleiche vorgenommen werden.

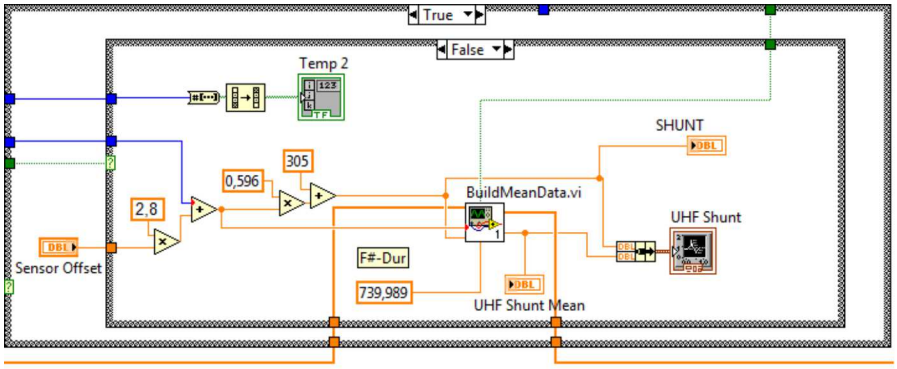


Abbildung 3.12: Shuntsensorauswertung

### 3.3.2 Graphisches User Interface GUI

Da die Funktionsweise bzw. der Programmablauf detailliert beschrieben wurde, kann nun auf die graphische Oberfläche des User Interfaces näher eingegangen werden. Hierbei wurde auf Benutzerfreundlichkeit gesetzt. Anstelle eines unübersichtlichen Fensters mit allen Elementen, wurden die Elemente und Funktionen kategorisch zusammengefasst. Die in den nächsten Abbildungen dargestellten Fenster stellen diese Kategorien dar, die mit den vorhandenen Menüreitern entsprechend ausgewählt werden können.

Das Hauptfenster des LabVIEW-Programms ist in Abbildung 3.13 dargestellt. Es umfasst neben den wichtigsten Ausgabeelementen und ein paar Einstellungen auch die Möglichkeit die Verbindung mit der Hardware herzustellen. Das Hauptfenster konzentriert sich auf den Kern der UHF-RFID-Kommunikation, dem Auslesen der Sensordaten der RFID-Tags. Im Folgenden werden die Grund-Bedienelemente des Hauptfensters dargestellt und beschrieben.

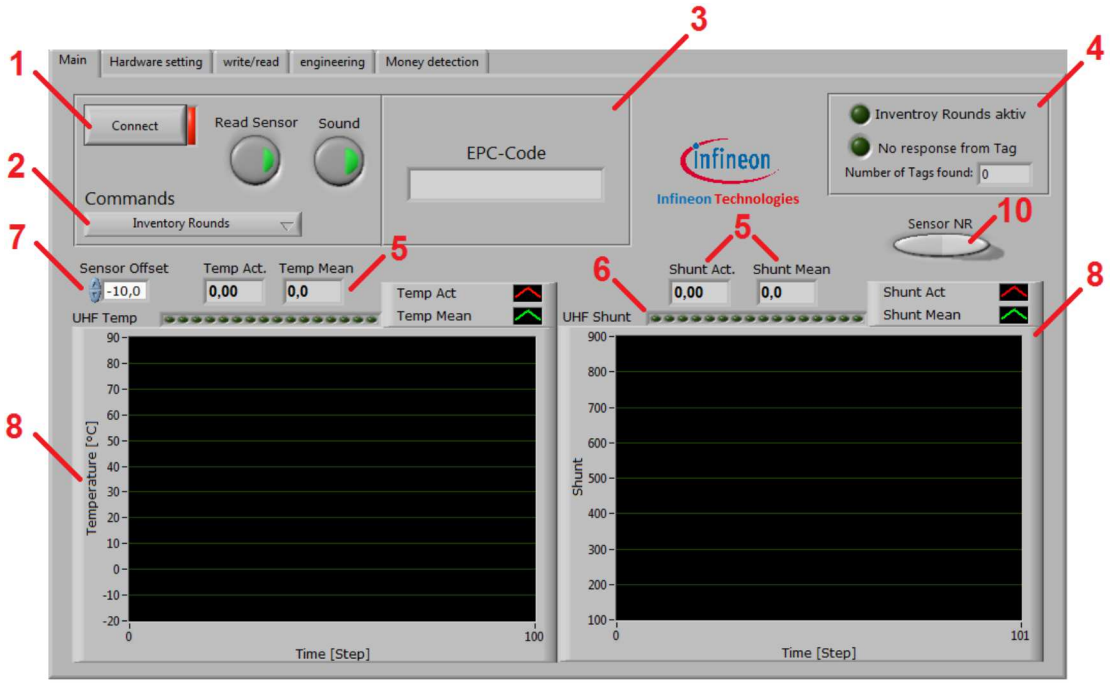


Abbildung 3.13: Hauptbildschirm

1. **Connect:** Dieser Button sollte nach Ausführung des Programms betätigt werden, um die Verbindung zum USB-Reader herzustellen. Ändert sich die Farbe der Anzeige-Led von Rot nach Grün, ist die Verbindung hergestellt und es wurden alle

nötigen Parameter (wie zum Beispiel die USB-Schnittstelle) initialisiert.

2. **Commands:** Hier wird die Programmfunktion ausgewählt. Je nach Befehlsauswahl operieren die dementsprechenden Ausgabeblöcke. Der Befehl "Inventory Rounds" zum Beispiel, sucht nur in der Umgebung vorhandene Tags und gibt deren EPC aus. "WWR" ist der elementarste und wichtigste Befehl. Dieser besteht aus der benötigten Befehlsabfolge Write-Write-Read, die das Auslesen der Sensordaten vom Tag ermöglicht. Der Befehl "WWR-Fast" kann nach "WWR" ausgeführt werden. Dieser minimiert die zeitlichen Abläufe und steigert somit auch die Lesegeschwindigkeit der Sensordaten erheblich.
3. **EPC-Ausgabe:** Dieses Ausgabeelement zeigt nach erfolgreicher Kommunikation den empfangenen Electronic Produkt Code an. Das gilt sowohl für "Inventory Rounds" als auch für die zwei "WWR"- Befehle.
4. **Inventory Round:** Dieses Feld ist mit der Auswahl des Befehls "Inventory Rounds" aktiv, was auch die oberste LED im Feld durch Leuchten bestätigt. Wenn die zweite LED leuchtet befindet sich kein RFID-Tag in der Nähe bzw. kann nicht vom Reader gelesen werden.
5. **Sensordaten:** "Shunt Act." und "Temp Act." gibt den aktuell gelesenen Sensorwert des RFID-Tags aus. Bei den zwei Werten "Shunt- und Temp Mean" handelt es sich um deren Mittelwert.
6. **Sensordaten Bitausgabe:** Hierbei handelt es sich um eine weitere Darstellungsform der aktuell ausgelesenen Sensordaten. Entwickler und Ingenieure können dadurch die einzelnen Sensorbits leichter nachvollziehen und verifizieren.
7. **Sensor Offset:** Jeder hergestellte RFID-Tag hat aufgrund von Prozessschwankungen leicht unterschiedliche ADC-Werte. Dadurch variiert auch der Shunt- und Temperaturwert zwischen Tags mit einem gewissen Offset. Mit dem Eingabeelement kann der Offset kompensiert werden.
8. **Diagramm:** Im vorhandenen Graphen werden alle gelesenen Sensorwerte und deren Mittelwerte als Funktion der Zeit dargestellt.

Im zweiten Fenster, das mit den Menüreitern ausgewählt werden kann, werden die Hardwareeinstellungen vorgenommen. Das ist bei verschiedenster Belastung des Readers (zum Beispiel unterschiedliche Antennen oder größere Tags) wichtig. Bei unterschiedlichen Anwendungen, wird empfohlen, auch die Ausgangsleistung und Sensitivität dementsprechend zu verändern. Abbildung 3.14 stellt alle vorhandenen Einstellmöglichkeiten dar. Wird ein USB-Reader das erste Mal verwendet, sollte anschließend eine der Anwendung entsprechende Hardwarekonfiguration durchgeführt werden.

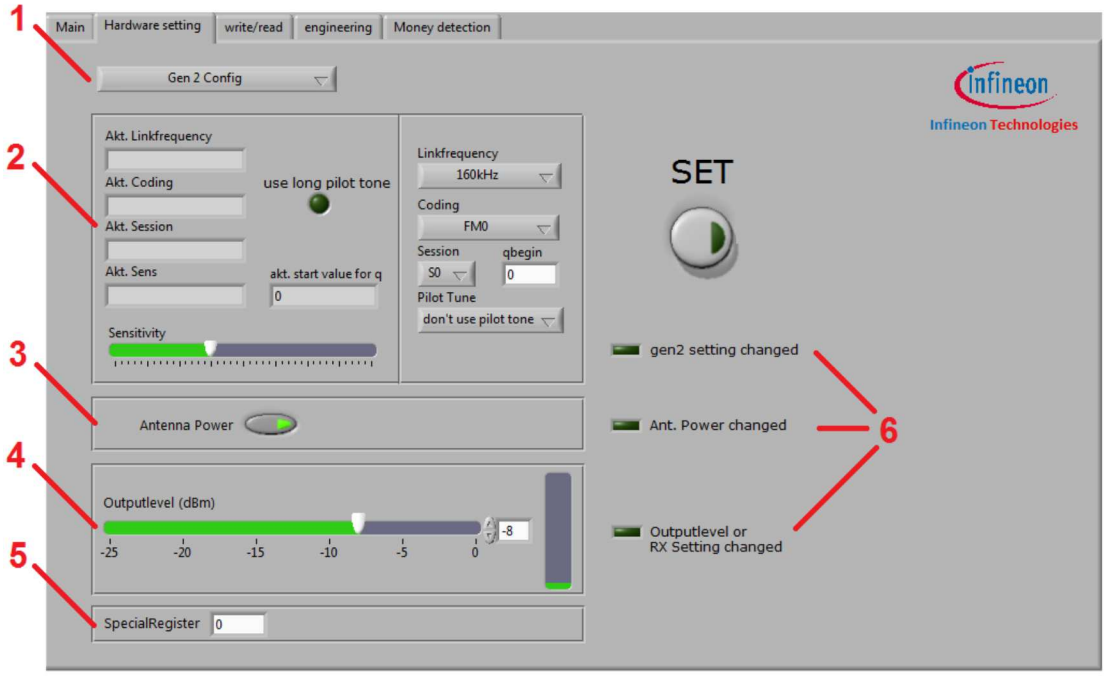


Abbildung 3.14: Hardwaresettings

1. **Befehle:** Wiederum werden hier durch das Trop-Down Menü die Befehle ausgewählt. Nach Auswahl ist das entsprechende Feld aktiv und dessen Einstellung wird mit ”SET” übernommen.

2. **Gen2 Einstellungen:** Aktuelle Gen-2 Hardwareeinstellungen werden hier angezeigt. Die Sensitivität und alle anderen einstellbaren Gen2-Optionen werden mit ”SET” übernommen. War dies erfolgreich, wird auch wiederum die Ausgabe aktualisiert.

3. **Antennen Power:** Mit diesem Button kann die Ausgangsstufe des Readers ein- bzw. ausgeschalten werden.

4. **Outputlevel:** Diese Einstellung ermöglicht das Variieren der Ausgangsleistung. Dabei wird der Maximalwert der Leistung mit dem eingestellten Wert (in dBm) geschwächt.

5. **Registersetting:** Um die Readingdistance zu maximieren, müssen zusätzlich Register des AS3992 UHF-Chips entsprechend verändert werden. Das Setzten von ”RX Special Setting” auf den Defaultwert ”0” ist empfohlen.

6. **Status-LED:** Das Leuchten der entsprechenden LED bestätigt eine erfolgreiche

Veränderung der Hardwareeinstellung.

Im dritten Fenster sind die zwei Befehle Read und Write implementiert. Sie ermöglichen, wie der Name schon sagt, Lesen und Schreiben der verschiedenen Speicherbänke. Alle dazu verwendeten Funktionen sind in diesem Fenster (Abbildung 3.15) zusammengefasst.

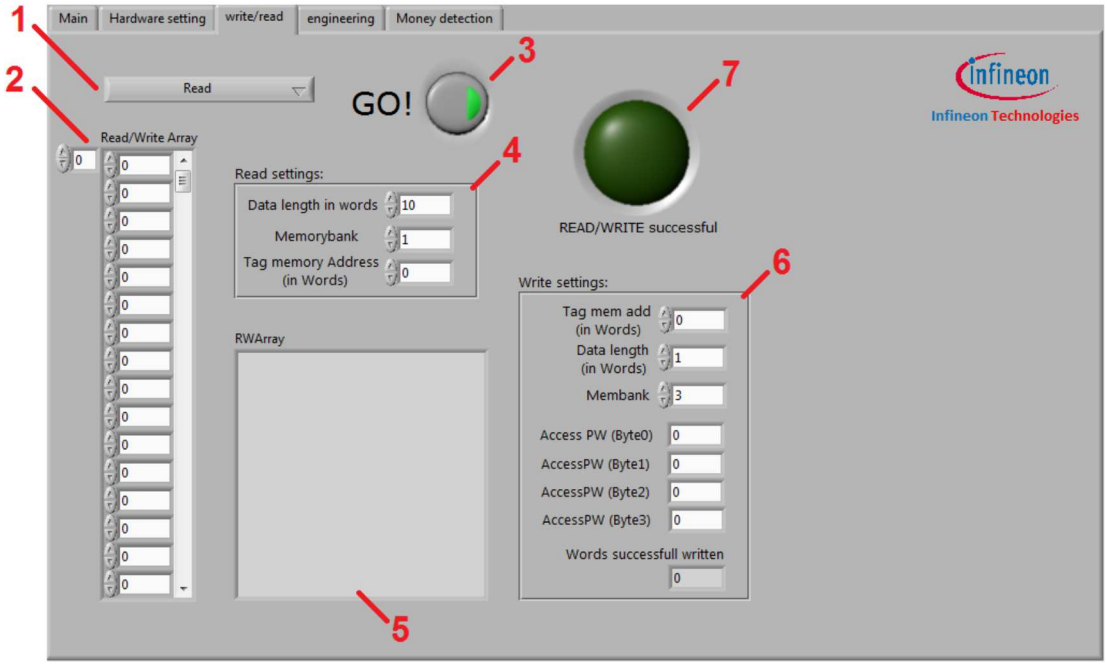


Abbildung 3.15: Read/Write Befehlsfenster

1. **Befehlsauswahl:** Hier kann lediglich zwischen Read und Write gewechselt werden.
2. **Buffer Array:** Dieses Array dient zur Dateneingabe. Nach Ausführen des Befehls ”write” wird dieser Buffer, zumindest der unter ”Write settings” eingestellte Teil, zum Tag geschrieben.
3. **GO:** Führt den ausgewählten Befehl aus
4. **Read settings:** ”Tag memory Address” entspricht einem Pointer. Ab diesen eingestellten Wert wird die entsprechende Anzahl der Wörter (Einstellung unter ”Data length in words”) gelesen. Natürlich kann auch die Speicherbank ausgewählt werden.
5. **RWArray:** Dieses Feld entspricht dem read bzw. write- Ausgabebereich.
6. **Write settings:** Wie auch unter Read-settings stehen hier die Grundeinstellungen zur Auswahl. Weiters ist noch die Eingabe eines Passwortes möglich, dass eventuell

erforderlich ist.

7. **Status-LED:** Die große LED bestätigt einen erfolgreichen Transfer.

Das Letzte Feld entspricht dem Demosystem namens ”Money detection”. Wie man zweifellos aus den Namen schon entnehmen kann, handelt es sich dabei um eine Art Geldscheinerkennung. Wird ein Tag mit dem entsprechenden EPC in Lesereichweite gebracht, so wird dieser erkannt und dessen zuvor definierter Euro-Wert als Bild ausgegeben.

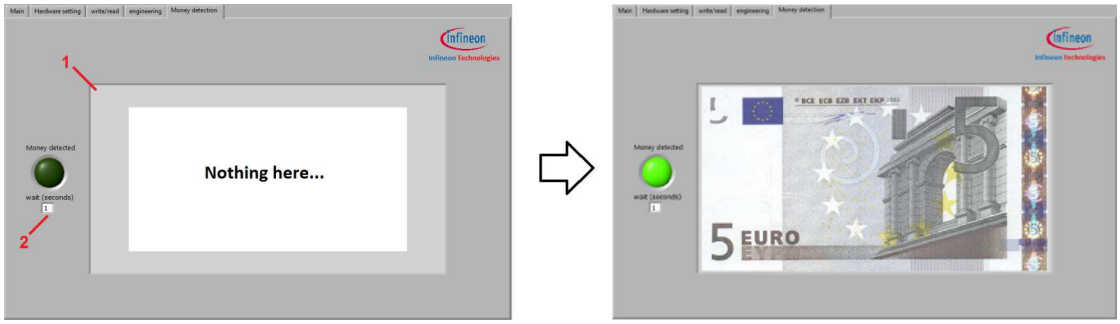


Abbildung 3.16: Demosystem ”Moneydetection”

1. **Anzeige:** Nach der Erkennung wird hier der entsprechende Euro-Wert ausgegeben.
2. **wait:** Hier kann die Zeit, wie lange das Erkennen eines Tags angezeigt wird, verändert werden. D.h. nach erfolgreicher Detektion vergeht eine gewisse Anzahl von Sekunden bis wiederum das Suchen nach Tags gestartet wird. Während dieser Zeit wird der entsprechende Euroschein angezeigt.

# Kapitel 4: Verifikation der Kommunikation

Die Kommunikation zwischen RFID-Reader und OCA-Tag erfolgt nach dem EPCglobal-Standard. Die EPCglobal-Spezifikation beschreibt alle physikalischen und logischen Anforderungen des UHF-RFID Systems.

In diesem Kapitel wird der Kommunikationsaufbau der EPCglobal-Spezifikation detailliert beschrieben und direkt mit den Messungen des entwickelten RFID-Systems verglichen. D.h. jeder einzelne Befehl und dessen Antwort bei einer Kommunikation zwischen dem OCA-Tag und dem Reader wird hier aufgezeigt und analysiert. Dabei wird auch auf den speziellen Write-Write-Read Befehl zur Sensorauswertung genauer eingegangen. Die Messung erfolgte mittels Oszilloskop, dessen Tastkopf direkt in das Feld gelegt wurde.

## 4.1 Kommunikationsablauf

Abbildung 4.16 beschreibt einen typischen Ablauf der Befehle und Antworten des RFID-Systems. Die folgenden Messungen zeigen eine solche spezifische Folge, welche nun genauer analysiert wird. Beispielsweise besteht auch ein simpler Schreibvorgang gegebenenfalls aus einer Reihe von Befehlen und deren Antworten. Ist in diesem Fall z.B. noch kein Tag ausgewählt, muss dieser zuvor mit einer $Inventory-round$ gesucht werden. Jeder Ablauf einer Kommunikation im entwickelten RFID-System stimmt mit dem von EPCglobal definierten UHF-Standard überein.

## 4.2 Inventory Round

Bei einer Inventory Round handelt es sich um den Aufbau der Kommunikation zwischen Reader und Tag. Hier wird unter Berücksichtigung der Randbedingungen ein Tag vom Reader ausgewählt und kennengelernt. Erst dann ändert sich der Zustand des Tags zu *Acknowledged* und später zu *Open*, um auf dessen Speicher zugreifen zu können. In den folgenden Bildern wird jeder Befehl der Kommunikationsabfolge einzeln betrachtet. Das Diagramm in der oberen Hälfte der Bilder stellt die gesamte Inventory Round dar. Der heraus gezoomte Bereich entspricht je einem Befehl bzw. einer Antwort und wird in der unteren Hälfte abgebildet. Auf die Zeitintervalle zwischen den Befehlen wird hier nicht

näher eingegangen. Dennoch beweist die Messung, dass diese eingehalten wurden, da sonst die Befehle ignoriert und ein Error-Code gesendet worden wäre.

### 4.2.1 Select

*Select* ist ein optionaler, erster Befehl des Kommunikationsaufbaus zwischen Reader und Transponder. Hier startet der Reader die Kommunikation und wählt, wie der Name des Befehls vermuten lässt, eine bestimmte Anzahl von Transpondern aus, die sich in Kommunikationsreichweite befinden. Die Auswahlkriterien ergeben sich aus gesetzten Bits bzw. Flags des Select-Befehls, welche die Transponder in verschieden Gruppen unterteilen. Alle Transponder im elektromagnetischen Feld führen diesen Befehl aus verhalten sich dementsprechend.

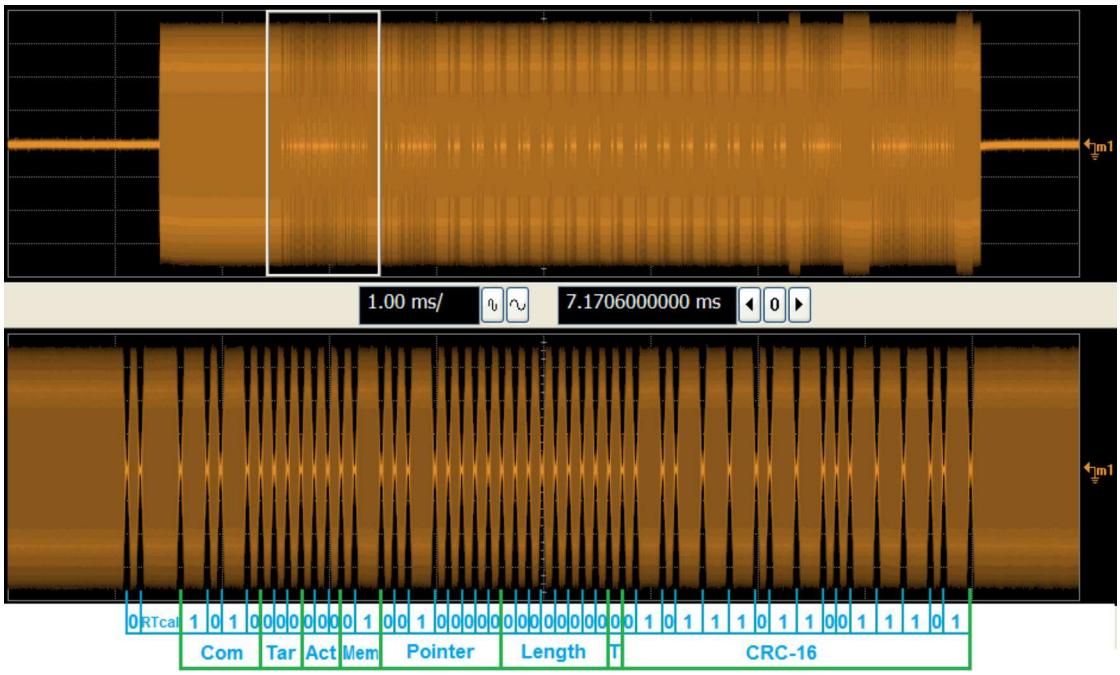


Abbildung 4.1: Select

Außer dem Befehl *Query* startet ein Kommunikationsvorgang immer mit einem sogenannten $Frame - Sync$. Wie auch Abbildung 4.2 zeigt, besteht es aus einem Delimiter, einem data-0 Symbol und dem RTcal. Der Delimiter dient als Sicherheitstrennung und leitet die Kommunikation ein. Anschließend sendet der Reader im $Frame - Sync$ ein data-0, welches die Länge des in den Grundlagen beschriebenen Tari-Symbols festlegt.

Nun wird das für die 0 und 1 Unterscheidung wichtige RTcal übertragen.

$$RTcal = 0_{length} + 1_{length} \tag{4.2.1}$$

Die Dauer beider Symbole data-0 und data-1 entsprechen also addiert der Dauer des RTcal-Symbols.

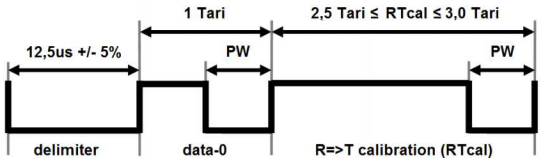


Abbildung 4.2: R=>T Frame-Sync laut [6]

Bei dem eingezeichneten PW-Signal handelt es sich um das *pivot*.

$$Pivot = \frac{RTcal}{2} \tag{4.2.2}$$

Der Transponder misst die Länge von RTcal und berechnet daraus das wichtige *pivot* Symbol. Durch *pivot* können nun die folgenden data-0 und data-1 Symbole unterschieden werden. Alle empfangenen Symbole, die kürzer als *pivot* sind werden als data-0, alle die länger dauern als data-1 Symbole interpretiert. Symbole die länger als 4*RTcal dauern, werden als Fehler erkannt. [6]

| | **Command** | **Target** | **Action** | **MemBank** | **Pointer** | **Length** | **Mask** | **Truncate** | **CRC-16** |
|---|---|---|---|---|---|---|---|---|---|
| # of bits | 4 | 3 | 3 | 2 | EBV | 8 | Variable | 1 | 16 |
| description | 1010 | 000: **Inventoried** (S0)<br>001: **Inventoried** (S1)<br>010: **Inventoried** (S2)<br>011: **Inventoried** (S3)<br>100: SL<br>101: RFU<br>110: RFU<br>111: RFU | See [6], Tabelle 6.19 | 00: RFU<br>01: EPC<br>10:TID<br>11: User | Starting Mask address | Mask length (bits) | Mask value | 0: Disable truncation<br>1:Enable truncation | |

Abbildung 4.3: Select Befehl laut [6]

Nach dem Frame-Sync startet der eigentliche Befehl. Wie aus der in Abbildung 4.3 dargestellten Befehlsstruktur ersichtlich, sind nach dem Command-Identifier (4 Bits) mehrere Selektier-Optionen vorhanden. Alle sich im elektromagnetischen Feld befindlichen Transponder werden im ersten Feld, abhängig von $selectedflag$, $inventoriedflag$ und $session$ in Gruppen unterteilt und können dementsprechend ausgewählt werden. *Action* löst die entsprechende Transponder Antwort aus. Hier wird festgelegt, welche Flags im Tag gesetzt werden, wenn die Selektier-Optionen zutreffen. Mit Hilfe der nachfolgenden Felder

Mask, MemBank, Pointer und Length ist ein Bitmaskenvergleich mit einem bestimmten Speicherbereich möglich. Hier kann z.B. eingestellt werden, dass nur Tags ausgewählt werden, deren EPC mit ”C” beginnt [6]. Das entwickelte RFID-System benötigt zurzeit, wie auch aus der Messung ersichtlich, keine zusätzlichen Auswahlkriterien.

### 4.2.2 Query

Die eigentliche Inventory-Round beginnt erst jetzt mit dem Befehl *Query*. *Query* startet anstelle des Frame-Sync mit ein Preamble, welche ein zusätzliches TRcal Symbol beinhaltet. Abbildung 4.5 zeigt den Ablauf nach [6]. Mit Hilfe dieses zusätzlichen Symbols TRcal ist der Reader in der Lage, die Rücksendefrequenz des Transponders einzustellen.

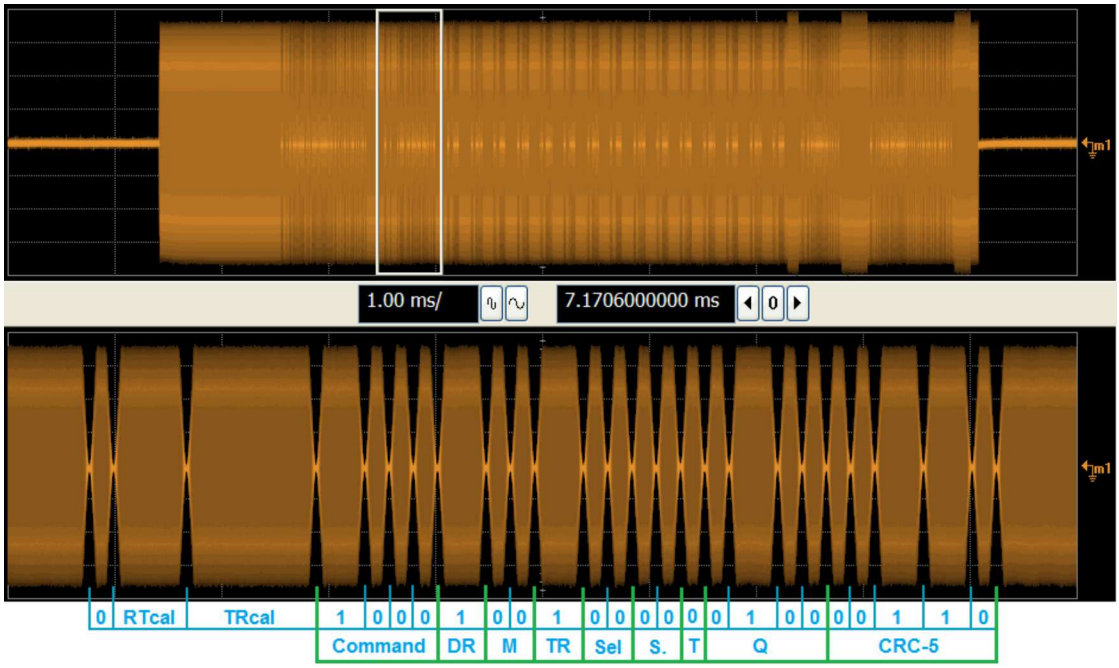


Abbildung 4.4: Query

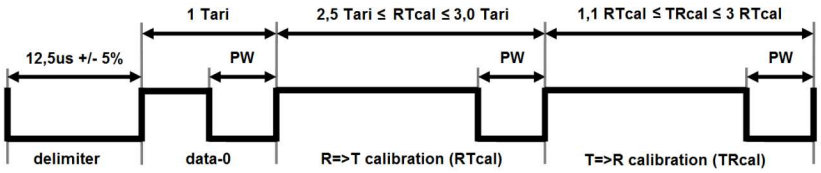


Abbildung 4.5: Preamble laut [6]

Der Transponder misst die Dauer von TRcal und berechnet die daraus resultierende Rücksendefrequenz (backscatter link frequency) LF. Weiters ist es mit einer zusätzlichen Option im *Query* Befehl möglich eine *divideratio* DR einzustellen, welche in die Berechnung der LF wie folgt einfließt. [6]

$$LF = \frac{DR}{TRcal} \tag{4.2.3}$$

Neben DR (der TRcal divide ratio) sind noch die in Abbildung 4.6 dargestellten Query-Optionen vorhanden. Mit $M$ kann die Codierung bzw. die Anzahl der Zyklen pro Symbol eingestellt werden. Die Datenrate und Modulation hängt also von diesem Parameter ab. Zur Auswahl stehen FM0, Miller2, Miller4 und Miller8. Der im System verwendete OCA-Transponder beherrscht, laut Vorgabe der EPCglobal-Spezifikation, alle vier Codierungsarten. Um die Antwort des Transponders leichter anhand der Messung zu dekodieren wurde in unserem Fall FM0 verwendet (d.h. M=00). $TRext$ gibt an, ob bei der Tag-Reader-Kommunikation ein Pilot-Tone verwendet wird. Falls verwendet, leitet dieser das Backscatter-Signal ein. *Sel* gibt an, welcher Tag auf das Query antwortet. Wie der Name vermuten lässt, ist *Session* für die verwendete Session der Inventory Round zuständig. Wie auch *Sel* ist $Target$ (selektiert Tag mit Inventory Flag A oder B) ein Auswahlkriterium der Tags. Treffen diese zu, so wird im Transponder ein interner Slot-Counter auf einen bestimmten Wert, abhängig von $2^{Q-1}$ (bei 4 Bits ist Q also zwischen 0 und 15) und der RN16, gesetzt. Darauf wird im nächsten Punkt näher eingegangen.

| | Command | DR | M | TRext | Sel | Session | Target | Q | CRC-5 |
|---|---|---|---|---|---|---|---|---|---|
| # of bits | 4 | 1 | 2 | 1 | 2 | 2 | 1 | 4 | 5 |
| description | 1000 | 0: DR=8<br>1: DR=64/3 | 00: M=1<br>01: M=2<br>10: M=4<br>11: M=8 | 0: No pilot tone<br>1: Use pilot tone | 00: All<br>01: All<br>10: ~**SL**<br>11: **SL** | 00: S0<br>01: S1<br>10: S2<br>11: S3 | 0: A<br>1: B | 0-15 | |

Abbildung 4.6: Query Befehl laut [6]

### 4.2.3 QueryRep

Wie aus der Messung aus dem vorherigen *Query*-Befehl ersichtlich, wurde Q auf 4 gesetzt. Mit Q (4 Bits) kann, wie bereits erwähnt, nach $2^{Q-1}$ ein Wert zwischen 0 und 15 eingestellt werden. Der Transponder erzeugt mit dem implementierten Zufallszahlengenerator eine 16-bit-breite Zufallszahl (RN16). Der Q-Wert bestimmt nun die Anzahl der verwendeten Bits der RN16, mit der der Slot-Counter befüllt wird (beginnend mit dem LSB). Ist also Q z.B. 15, so entspricht der Wert des Slot-Counters der gesammten generierten 16 Bit Zufallszahl. In unserem Fall werden die letzten 4 Bits der RN16 verwendet. Da der verwendete OCA-Transponder nur über 000Fh als ”Zufallszahl” verfügt (darauf wird

später genauer eingegangen), stellt sich dessen interne Slotcounter auf 15. Dieser wird nun mit jedem *QueryRep* dekrementiert. Besitzt der Slotcounter schlussendlich den Wert 0, antwortet der Tag. Dieser Ablauf ist in Abbildung 4.7 dargestellt. Nach dem 15. *QueryRep* sendet der Transponder die in Abbildung 4.7 herausgezoomte Antwort.

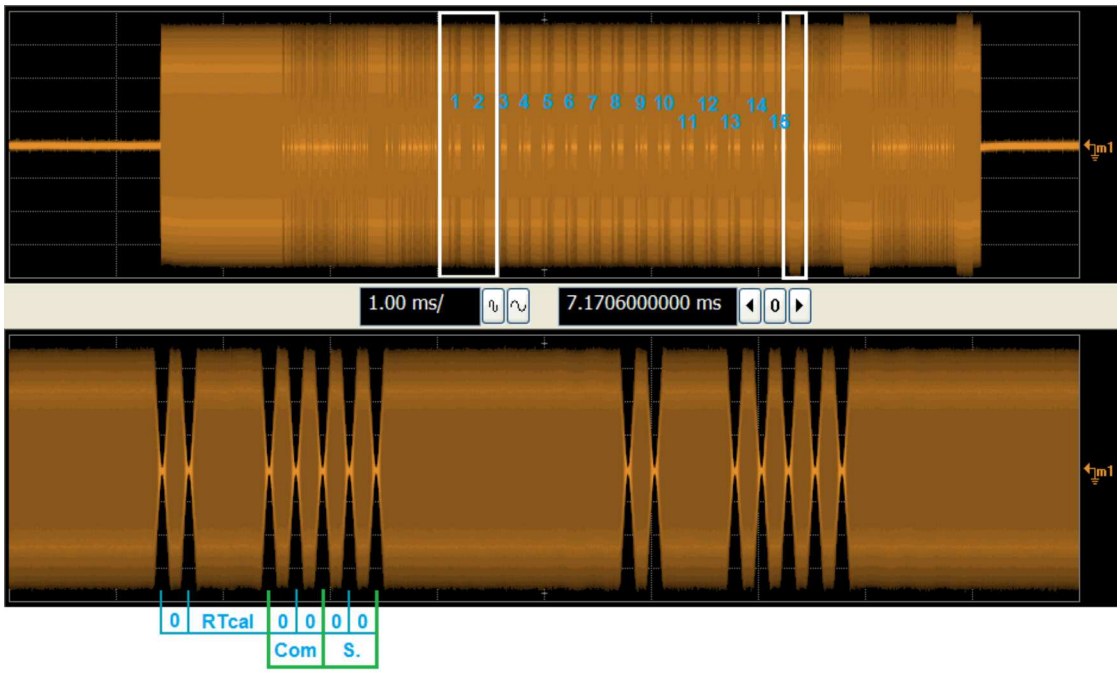


Abbildung 4.7: QueryRep

Sind sehr viele Tags im elektromagnetischen Feld, kann man Q durchaus auf 15 setzten. Bevor die Transponder antworten, werden nun alle durch den hohen Zufallszahlenwert sehr unterschiedlich lang warten. Dadurch ist es fast unmöglich, dass 2 Transponder gleichzeitig antworten. Ist aber beispielsweise nur ein Tag im Feld sollte Q auf 0 gesetzt werden, um unnötige Wartezeit zu vermeiden.

Wie auch im *Query*-Befehl werden hier nur die Tags mit der bestimmten Session angesprochen. Die Session wurde, wie bereits erwähnt durch *Select* einem Tag zugewiesen, um diese in Gruppen zu unterteilen. Dadurch handelt es sich bei *Session* um den einzigen Payload-Parameter von *QueryRep*. Die Befehlsstrukturen sind in Abbildung 4.8 dargestellt.

Die Antwort des Transponders besteht aus einer in [6] definierten Preamble und der RN16. Diese 16-Bit breite Zufallszahl wird von jedem Transponder, der sich jetzt im *Reply*-Status befindet, an den Reader zurückgesendet. Die Wahrscheinlichkeit, dass zwei

Tags die gleiche RN16 erzeugen beträgt nur 0,025 %. Würde dieser Fall trotzdem eintreffen, führt dies zu einem Fehler und der Reader startet die Kommunikation neu.

| | Command | Session |
|---|---|---|
| # of bits | 2 | 2 |
| description | 00 | 00: S0<br>01: S1<br>10: S2<br>11: S3 |

| | Response |
|---|---|
| # of bits | 16 |
| description | RN16 |

Abbildung 4.8: QueryRep Befehl und Tag-Antwort laut [6]

Im verwendeten OCA-Tag ist noch kein Zufallszahlengenerator implementiert der alle von EPCglobal definierten Kriterien erfüllt. Die Random Number wurde lediglich auf 000Fh festgelegt, was aber zu einer erleichterten Verifikation eines einzelnen Tags führt. In den nächsten Versionen des OCA-Tags wird dieser jedoch implementiert.

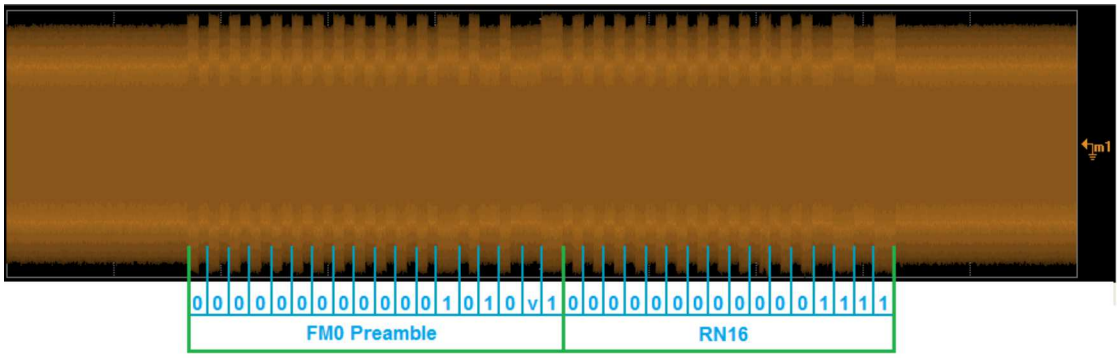


Abbildung 4.9: QueryRep Tag-Antwort

Im entwickelten System wurde, wie auch aus der *Query* Messung ersichtlich, TRext auf 1 gesetzt. Das bedeutet der bereits erwähnte $Pilot - tone$ wird verwendet. Somit ergibt sich die in Abbildung 4.9 dargestellte Preamble mit den 12 Nullen am Anfang. Das in der Preamble vorhandene $v$ entspricht einem Fehler bzw. einer beabsichtigten Verletzung der FM0-Codierung. Nach der FM0-Codierung müsste hier das Symbol invertiert sein. Abbildung 4.10 veranschaulicht diesen Vorgang laut [6].

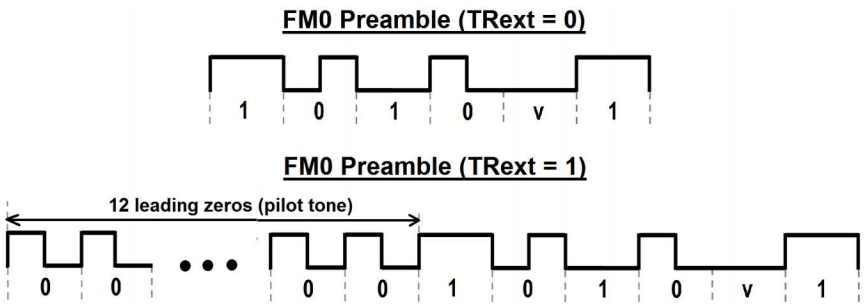


Abbildung 4.10: FM0 T=>R Preamble laut [6]

### 4.2.4 ACK

Der *ACK* Befehl dient als Antwort auf die RN16 des Tags und gibt diese als Echo wieder. Damit bestätigt der Reader die Kommunikation mit dem Transponder. Je nach Zustand des Transponders wird die RN16 oder *handle* gesendet. *handle* wird später genauer beschrieben.

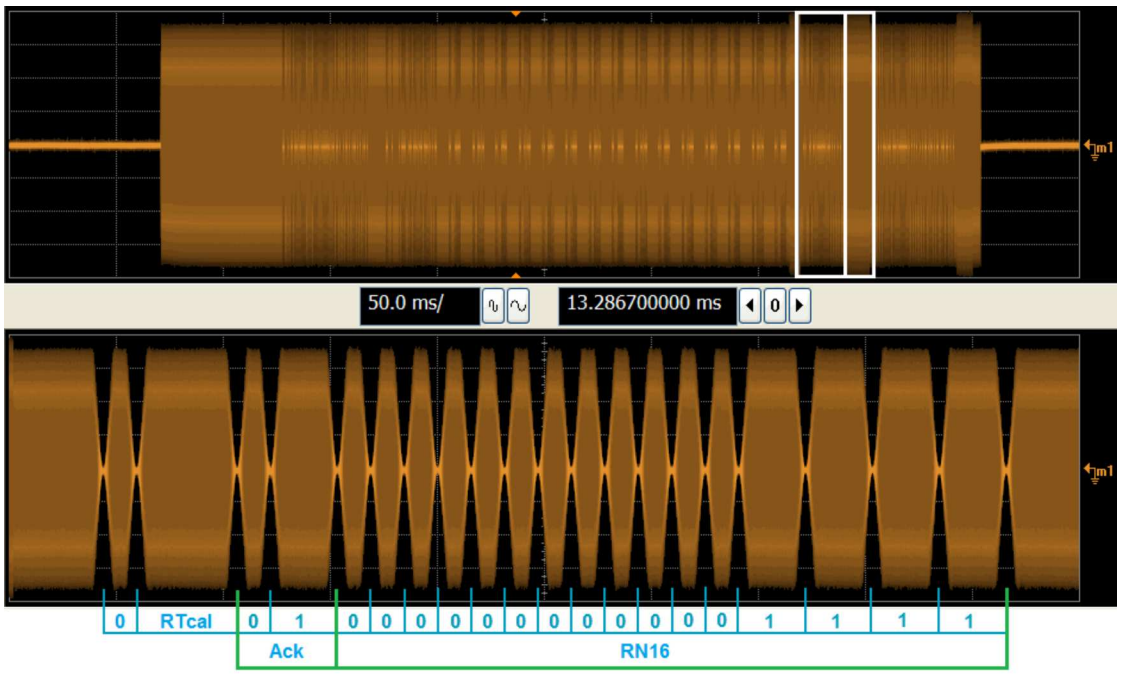


Abbildung 4.11: Acknowledged

Die Antwort des *ACK*-Befehls besteht aus dem PC (Protocol-Control), der EPC und einer CRC-16 als Checksum. Sowohl PC als auch EPC dienen der Identifizierung. Erst jetzt ist der Transponder dem Reader bekannt. Die ersten 5 der insgesamt 16 Protocol-Control-Bits definieren die Länge des EPCs + PC in Words folgendermaßen, wobei ein Word zwei Bytes entspricht.

00000b ⇒ ein Word
00001b ⇒ zwei Words
00010b ⇒ drei Words
... 11111b ⇒ 32 Words

Bei den nächsten 2 Bits des PC handelt es sich um reservierte Bits. Sie sollten bei Class-1 Tags immer auf 0 gesetzt werden. Die restlichen 9 Bits sind sogenannte NSI (numbering system identifier)-Bits, welche die in [6] beschriebenen phyikalischen Informationen und den EPCglobal Header beinhalten.

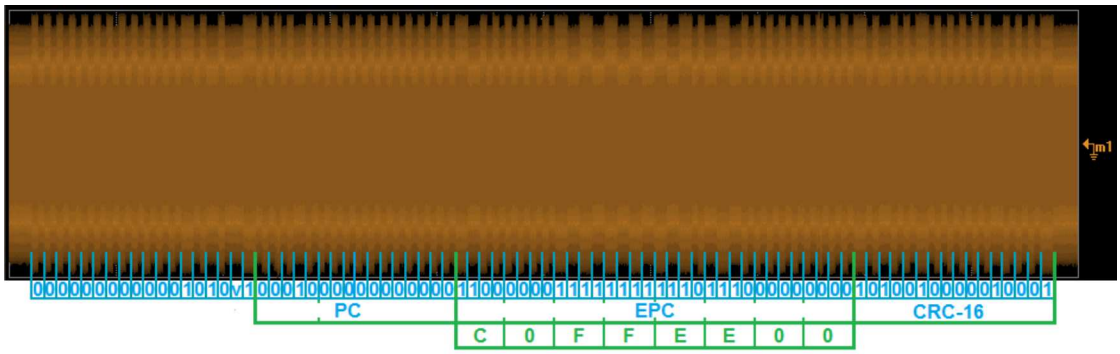


Abbildung 4.12: Acknowledged Tag-Antwort

Der EPC des verwendeten OCA-Tags entspricht, wie auch aus der Messung ersichtlich, dem Wert C0FFE00h. Dabei handelt es sich um keinen richtigen bzw. eindeutig identifizierbaren Elektronic Product Code von EPCglobal. Dieser, an das Wort Kaffee angelehnte "Test-EPC", wurde in den ersten OCA-Prototypen von den Designern definiert. Der EPC (C0FFE00h) besteht aus 2 Words und ergibt addiert mit dem PC (1 Word), durch die ersten 5Bits vorgegebene Länge (00010b = 3 Words).

| | **Response** |
|---|---|
| # of bits | 21 to 528 |
| description | {PC, EPC, CRC-16} OR {$00000_2$, truncated EPC, CRC-16} |

Abbildung 4.13: ACK Tag-Antwort laut [6]

### 4.2.5 Req_RN

Reqest Random Number oder *Req_RN* fordert den Tag auf eine neue *RN*16 oder *handle* zu senden. Er gehört zwar zu den Access und nicht zu den Inventory Commands, wird aber zum Abschluss dieser Inventroy Runde verwendet.

Je nach Zustand von Reader und Tag wird, wie in [6] beschrieben, der Status geändert oder beibehalten. *Req_RN* wird hauptsächlich dazu verwendet, um den Zustand des Tags in Open oder Secured zu ändern. Später kann auf den Speicher zugegriffen werden. *handle* ist im Prinzip auch eine normale *RN*16, welche aber als diese bezeichnet wird, wenn sie zum Ansprechen des Speichers etc. dient.

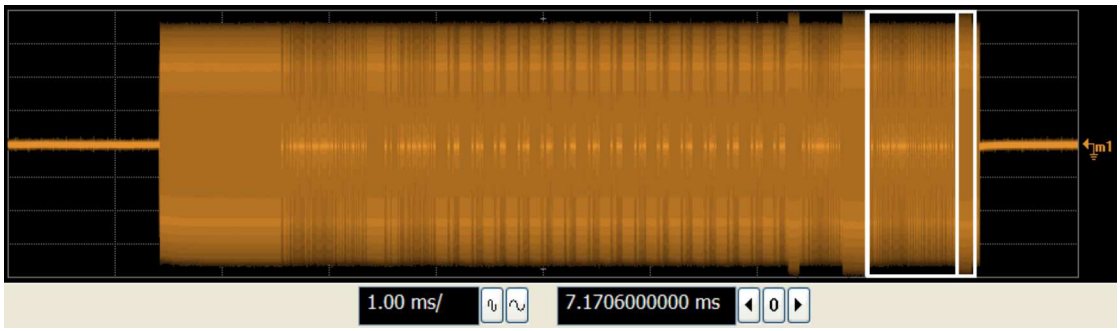


Abbildung 4.14: Req_RN und Req_RN Tag-Antwort

Auf die Messung wird hier nicht näher eingegangen, da sowohl alte als auch neue *RN*16 bzw. *handle* den Wert 000Fh besitzen. Dies is darauf zurückzuführen, dass kein Zufallszahlengenerator im Tag implementiert ist. D.h., wie auch in Abbildung 4.15 dargestellt, wird in unserem Fall die *RN*16 zum Tag gesendet und das neue (000Fh) *handle* empfangen.

| | Command | DR | CRC-16 |
|---|---|---|---|
| # of bits | 8 | 16 | 16 |
| description | 11000001 | Prior RN16 or handle | |

| | RN | CRC-16 |
|---|---|---|
| # of bits | 16 | 16 |
| description | New RN16 or handle | |

Abbildung 4.15: Req_RN Befehl und Tag-Antwort aus [6]

## 4.3 Sensorauswertung WWR-Befehl

Wie im Punkt 3.1.2 detailliert beschrieben, startet der am OCA-Transponder implementierte ADC, wenn ein spezieller Write-Befehl vom Reader gesendet wird. Ebenso

wird über die Bits *ADC_EN* und *Sens_EN*0 bzw. *Sens_EN*1 der gewünschte Sensor ausgewählt. Nach dem Start der Sensormessung darf das elektromagnetische Feld für eine einstellbare Zeit nicht verändert werden, da ansonsten die Ergebnisse verfälscht werden. Nach erfolgreicher ADC-Wandlung speichert der OCA-Transponder die Messergebnisse in der User-Speicherbank. Diese können nun unter Einhalten der Zeitbedingung mit dem Read-Befehl ausgelesen werden.

### 4.3.1 Timing

Der ADC des OCA-Transponders benötigt für die Konvertierung der Sensorwerte mindestens 500 $\mu s$. Die zusätzlichen Verzögerungen, die im Punkt 3.1.2 ausführlich beschrieben werden, summieren sich auf ca. 100 $\mu s$, sind jedoch variabel. Sinnvolle Verzögerungswerte im Write-Write-Read-Befehl sind also Zeiten $\geq 1\,ms$. In diesem Fall wurde im GUI des Systems (LabView) $3\,ms$ als Verzögerung eingestellt. Der ADC und zusätzliche Digitalteil haben somit ausreichend Zeit die Sensorwerte zur Verfügung zu stellen. Das genaue Messergebnis in Abbildung 4.16 ist darauf zurückzuführen, dass die im GUI einstellbare Verzögerung zwischen den Befehlen Write, Write und Read direkt im Reader implementiert wurde.

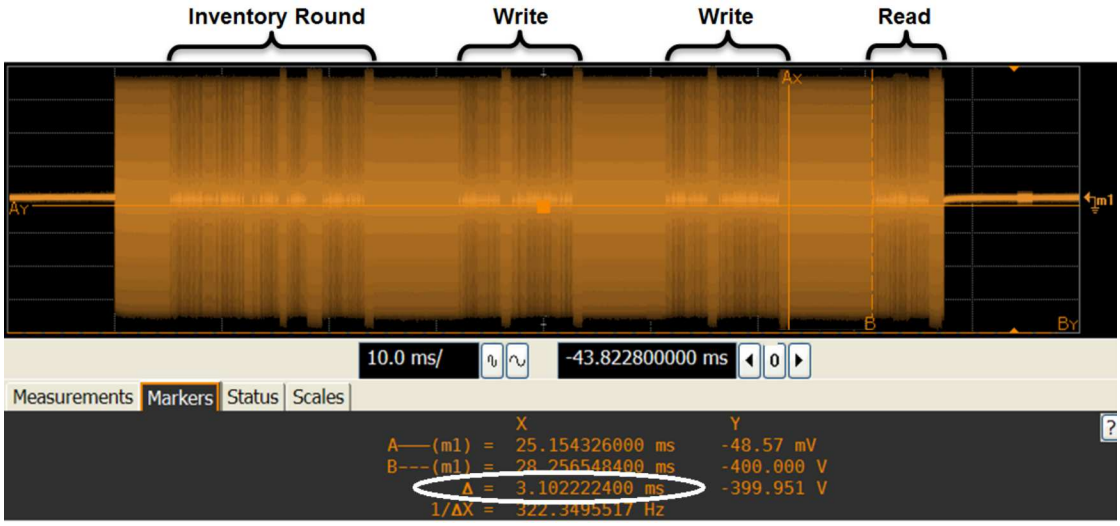


Abbildung 4.16: gesamter Ablauf der Sensorauswertung mit Timing

In Abbildung 4.16 ist weiters die zuvor beschriebene Inventory Round erkennbar. Sie leitet in diesem Fall den insgesamt Write-Write-Read-Ablauf ein. Nach dieser ist der Transponder dem Reader bekannt und die Sensormessungen können gestartet und ausgewertet werden. Im GUI (LabView) wurde eine weitere Variante namens $WWR - Fast$

zur Sensorauswertung implementiert. Diese ermöglicht ein Ausführen von Write-Write-Read-Befehlen, ohne zusätzliche Inventory Rounds. Somit muss sie nur ein einziges Mal ausgeführt werden.

### 4.3.2 Write-Write

Ein kleiner Bug im Digitalteil der ersten Prototypen des OCA-Transponders setzt voraus, dass erst bei einem zweiten Write-Befehl die zu schreibenden Werte übernommen werden. Die nächsten Versionen des OCA-Transponders werden in Zukunft nur einen einzigen Write-Befehl benötigen. Die in Punkt 3.1.2 beschriebenen Kriterien bezüglich der geforderten Zeitabstände zwischen Write und Read müssen natürlich trotzdem beachtet werden.

Ein Write-Befehl kann nur ausgeführt werden, wenn der Tag dem Reader bereits bekannt ist und sich dieser im open oder secured Zustand befindet. Bevor ein Write-Befehl ausgeführt wird, wird daher ein *handle* mit dem bereits beschriebenen Req_RN Befehl angefordert. Nun kann der eigentliche Write-Befehl mit folgenden Parametern gesendet werden.

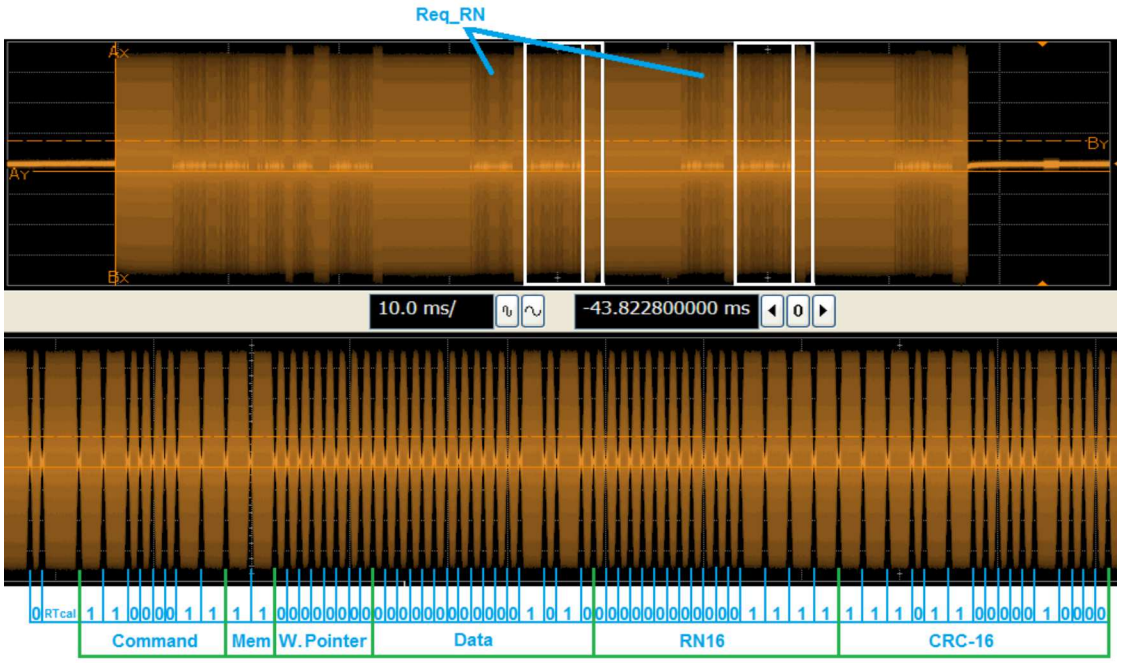


Abbildung 4.17: Write

Die Startbits für die Sensormessung befinden sich wie in Punkt 3.1.2 beschrieben in der Userbank, welche mittels $MemBank = 11$ ausgewählt wurde. Der $WordPtr$ gibt den verwendeten Offset (beginnend mit dem LSB) innerhalb der ausgewählten Bank an. An dieser Stelle werden die Daten geschrieben.

Mit einem einzigen Write-Befehl können maximal 16 Bits bzw. 2 Words übertragen werden. Aufgrund der Sicherheit werden diese mit den 16 Bits der $RN16$ (hier *handle*) mittels XOR verknüpft. In unserem Fall werden mit $0011b = 3$ oder $0101b = 5$ die Sensoren ausgewählt und der ADC gestartet. Verknüpft mit der fixen "Test-RN16" (000Fh) ergeben sich somit die in Abbildung 4.17 dargestellten Daten.

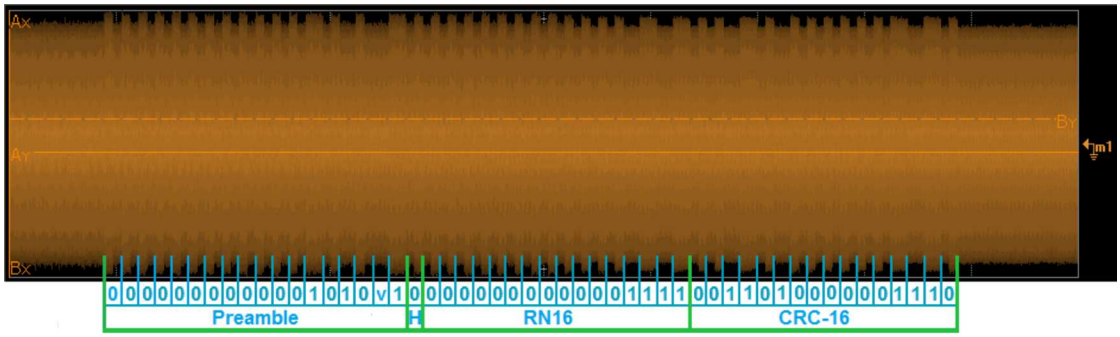


Abbildung 4.18: Tag-Antwort

Bei der Antwort des Transponders handelt es sich zwar lediglich um ein neues *handle*, jedoch wird diesem ein einzelnes Headerbit vorangestellt. Die Checksum (CRC16) wird nun aus *handle* und dem Header vom Transponder berechnet. Empfängt der Reader die in Abbildung 4.18 dargestellte Antwort innerhalb der in [6] vorgegebenen 20 ms, so war der Schreibvorgang erfolgreich.

### 4.3.3 Read

Mit dem Read-Befehl des Readers wird schlussendlich der ermittelte Sensorwert des Transponders ausgelesen. Wie auch im Write werden hier Parameter bezüglich des Speicherbereiches angegeben. Zusätzlich legt der Parameter $Word - Count$ die Anzahl der zu lesenden 16 Bit Words fest. Abbildung 4.19 zeigt den Read-Befehl des Readers, welcher den Sensorwert (zweites Word der Userbank) ausliest.

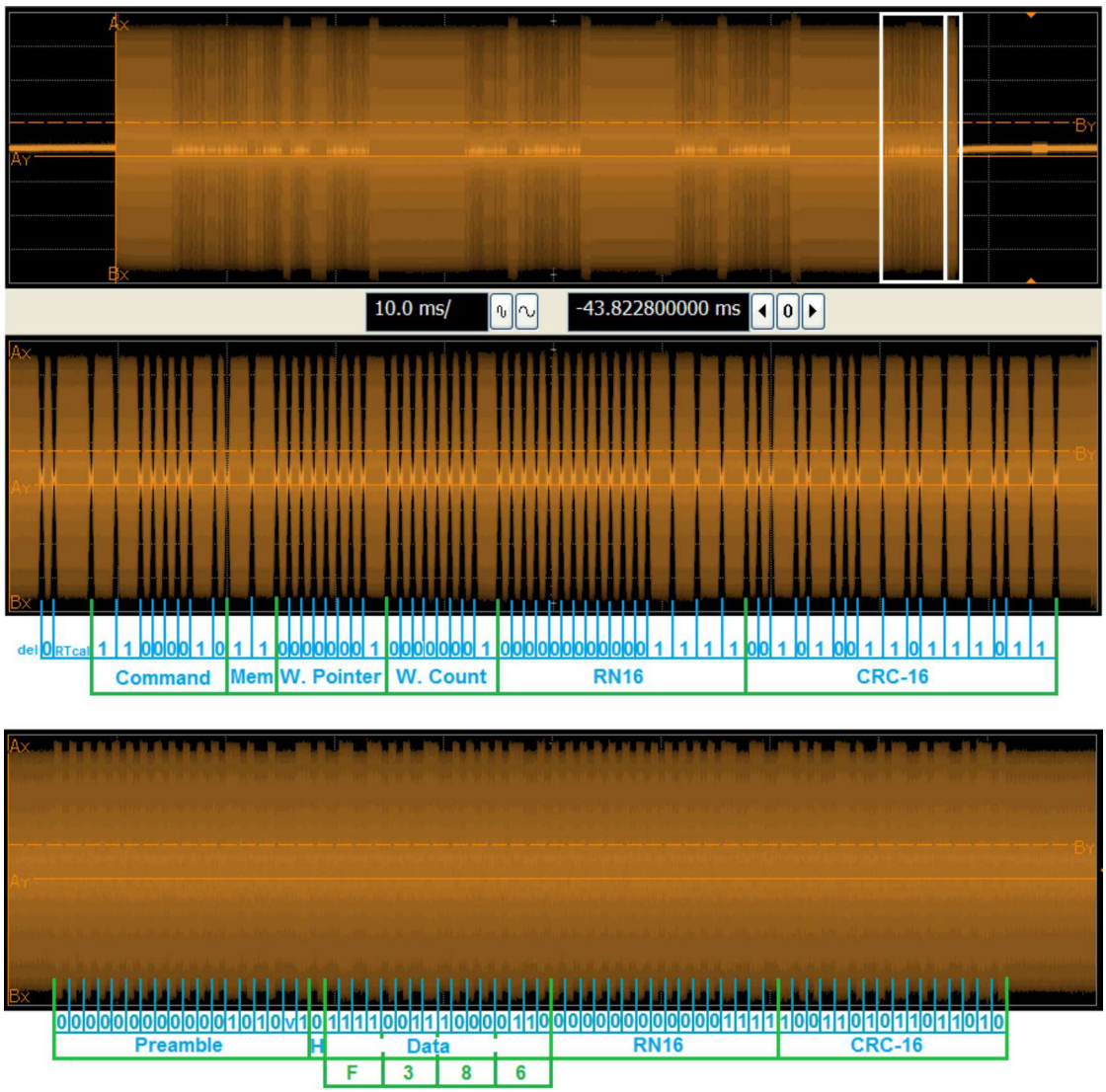


Abbildung 4.19: Read-Befehl und Tag-Antwort

Auch bei der Tag-Antwort des Read-Befehls ist nach der Preamble ein einzelnes Headerbit mit dem Wert 0 vorhanden. Anschließend werden alle angeforderten Datenbits, ein neues *handle* und der berechnete CRC-16 gesendet. Abbildung 4.19 zeigt das ermittelte Sensorergebnis. Die ersten vier Bits der Daten bestätigen die Richtigkeit der nachfolgenden Sensorwerte. Die Messung beweist, dass der in Punkt 3.1.2 besprochene Maximalwert (F3FFh) nicht überschritten wurde und somit die Messung weiterverarbeitet werden kann.

# Kapitel 5: Demosystem

Nach dem erfolgreichen Zusammenbau aller Komponenten des RFID-UHF-Systems soll nun auf dessen Einsatzgebiete näher eingegangen werden. Dazu werden in diesem Kapitel mögliche Einsatzbereiche erörtert und mit einem passenden Demosystem veranschaulicht.

Die Dimensionen des OCA-Transponders könnten theoretisch noch stark verkleinert werden. Beispielsweise kann die Siliziumschicht noch auf wenige $\mu m$ zugeschliffen werden. Betrachtet man mehrere derartige Chips, so trifft die Bezeichnung "Intelligenter Staub" zu. Deren Einsatzgebiet scheint schier unbegrenzt.

## 5.1 Geld/Dokumenten-Verifizierung

Laut [16] ist innerhalb der EU immer weniger Euro-Falschgeld im Umlauf. Dieser Trend ist auf die ständige Weiterentwicklung der Banknotenerkennungsmerkmale und deren Detektion (IR-Licht etc.) zurückzuführen. Die Europäische Zentral Bank (EZB) plant auch schon eine neue, zweite Serie der Eurobanknoten. Diese wird wiederum neue zusätzliche benutzerfreundliche Sicherheitsmerkmale aufweisen. Laut [15] werden die ersten Banknoten 2013 im Umlauf gebracht.

Der österreichische Anteil des gesamten Falschgelds des europäischen Raums beziffert sich auf 1,17 Prozent. In Österreich werden vermehrt kleinere Scheine gefälscht, da diese weniger kontrolliert werden. Der finanzielle Schaden beträgt in Summe "nur" 617,095 Euro.

Ein absolut sicheres Erkennungsmerkmal wäre ein in das Papier eingebetteter OCA-Transponder. Eine Read-Only Variante des Transponders mit starker Security und eindeutiger ID könnte jede Banknote zu einem Unikat machen. In Verbindung mit einem EPC-Global Prinzip (dem Internet der Dinge) könnten die Banknoten sogar überall auf der Welt eindeutig identifiziert und verfolgt werden. Somit würde sogar ein gefälschter OCA-Transponder (hoher Aufwand von Nöten) sofort detektiert werden. Nach dem Erkennen einer doppelten Banknote können sofort dementsprechende Maßnahmen eingeleitet werden.

### 5.1.1 Aufbau

Beim Demosystem handelt sich im Wesentlichen um das gesamte entwickelte UHF-RFID-System. Hier wurden verschiedene OCA-Transponder auf Banknoten aufgeklebt und detektiert. Liegt der Transponder im Bereich der verwendeten Nahfeldantenne, so werden die Geldscheine und deren Wert vom Reader detektiert und am PC ausgegeben. Dies geschieht sowohl mit einem Bild des entsprechenden Geldscheins, als auch akustisch. Abbildung 4.16 zeigt das im Labview implementierte Demosystem. Unter "wait" kann hier eingestellt werden wie lange der detektierte Eurowert angezeigt wird.

Abbildung 5.1: Banknotendetektor


Abbildung 5.2: Dokumenten Überprüfung

Neben der Geldverifizierung gibt es unzählige andere Einsatzgebiete im Bereich der Fälschungssicherheit. Wertvolle Dokumente, Gutscheine, Tickets etc. könnten alle mit einem OCA-Transponder gesichert werden. Außerdem bietet der RFID-Transponder die Möglichkeit zusätzliche Informationen zu speichern. Beispielsweise wurde in Abbildung 5.2 ein Lottoschein mit einem Transponder versehen. Um den Schein nun absolut fälschungssicher zu machen, könnte die Quittungsnummer auf dem Transponder abgespeichert werden. Ein zusätzlicher Vorteil wäre eine erleichterte elektronische Gewinnabfrage.

## 5.2 Bienen-Detektion

Durch seine Größe und sein Gewicht ist der OCA-Transponder auch im Bereich der Insektenforschung einsetzbar. Wie Kühe, Hunde etc. können so auch Kleinstlebewesen markiert und überwacht werden. Im Gebiet der Entomologie (Insektenforschung) wären OCA-Transponder sehr gut einsetzbar. Vor allem staatenbildende Insekten wie Termiten, Ameisen oder auch Bienen könnten eindeutig identifiziert und untersucht werden. Die Bienen-Institute könnten durch den Einsatz der OCA-Transponder beispielsweise einzelne Bienen auf ihre Aktivität und Lebensdauer elektronisch überprüfen. In Verbindung mit

elektronischer Datenverarbeitung und Datenbanken kann auf Änderungen der Population oder des Verhaltens der Völker rückgeschlossen werden. Einigen Phänomenen, wie beispielsweise dem Bienensterben in den USA, kann vielleicht mit Hilfe der OCA-Transponder besser auf den Grund gegangen werden.

> *"Insekten sind unsere wichtigsten Partner bei der Schaffung von Leben auf der Erde, denn oft übernehmen sie die Federführung bei der Gestaltung terrestrischer Ökosysteme. Etwa ein Drittel unserer Nahrung geht direkt auf die Bestäubung durch Insekten zurück. Allein in den USA entspricht diese Bestäubungstätigkeit jährlich einem Wert von mehr als neun Milliarden Dollar. Ohne Insekten gäbe es keine Orangen in Florida, keinen Käse in Wisconsin, keine Pfirsiche in Georgia und keine Kartoffeln in Idaho." May R. Berenbaum 2004 [2]*

Nicht nur Forschungseinrichtungen, sondern auch normale Imker sind an solchen Bienendetektionsmöglichkeiten sehr interessiert. Eine der häufigsten Arbeiten eines Imkers ist es, die Bienenkönigin zu finden. Wird sie vom Imker entdeckt, so gibt diese Aufschluss über den Zustand des gesamten Bienenvolks. Allein die Tatsache, dass die Königin vorhanden ist, erleichtert dem Imker nachfolgende Arbeiten erheblich. Um sie von tausenden anderen Bienen leichter zu unterscheiden, wird sie üblicherweise mit einem Farbtupfer markiert. Die Farbe ist hierbei vorgegeben und gibt Auskunft über das Alter der Bienenkönigin. Diese Farbbedeutungen werden in Tabelle 5.1 zusammengefasst.

| weiß | gelb | rot | grün | blau |
|---|---|---|---|---|
| 2006 | 2007 | 2008 | 2009 | 2010 |
| 2011 | 2012 | 2013 | 2014 | 2015 |
| 2016 | 2017 | 2018 | 2019 | 2020 |

Tabelle 5.1: Jahresmarkierungen der Bienenköniginnen

Meist schützen die Bienen beim Öffnen des Bienenstocks die Königin und sie ist unter mehreren Bienen versteckt. Trotz Markierung ist die Königin daher nicht immer leicht zu finden. In Abbildung 5.3 ist eine markierte Königin dargestellt. Zusätzliche Informationen über den Zustand und Eigenschaften des Bienenvolks protokolliert ein Imker in einem "Stock-Heft" bzw. einer "Stock-Karte" oder schreibt sie mit Kreide direkt auf den Stock.

All dies kann mit Hilfe eines OCA-Tags ersetzt und erleichtert werden. Dabei stellt ein tragbarer Reader mit einer gerichteten Antenne einen Scanner dar, mit dem die Königin sehr schnell geortet werden kann. Zusätzlich können gespeicherten Daten (wie z.B. das Alter oder die Nummer der Königin) ausgelesen werden.

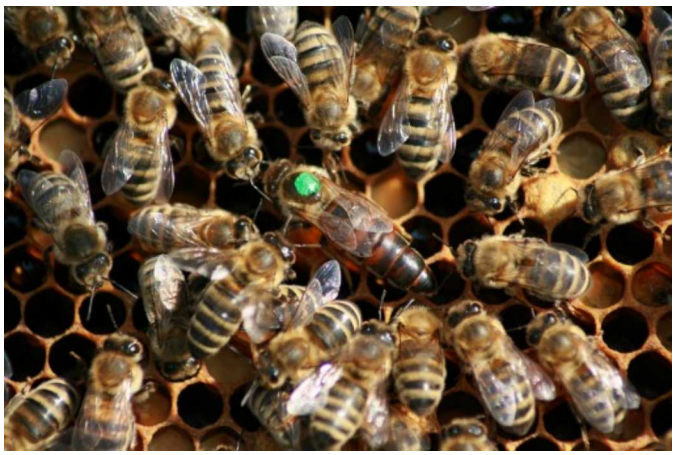

Abbildung 5.3: Markierte Bienenkönigin [17]

Das Demosystem "Bienendetektion" ist vor allem für Forschungszwecke verschiedenster Entomologie-Institute von Interesse. Hier ist am Flugbrettchen des Bienenstocks eine längliche Antenne installiert. Alle Bienen, die mit einem OCA-Transponder gekennzeichnet wurden, werden hier vom Reader detektiert und angezeigt. Somit kann die Aktivität einer Biene, sprich wie oft sie ausfliegt, festgestellt werden. Kommt sie von einem Ausflug nicht mehr zurück, kann davon ausgegangen werden, dass sie tot ist.

Die kleine Nahfeldantenne kann in diesem Fall nicht mehr verwendet werden, da hier eine breitere Detektionsfläche benötigt wird. Deshalb wurde eine Lecher-Leitung als Detektionslinie verwendet.

### 5.2.1 Lecher-Leitung

Jeden, der eine technische Ausbildung hinter sich hat, ist zumindest ein Versuchsaufbau bekannt, welcher stehende Wellen veranschaulicht. Sei es mit einem Seil, einer Wellenmaschine oder der besagten Lecher-Leitung. Bei dem Versuchsaufbau mit dem Seil wird ein Ende direkt an der Wand fixiert und das andere Ende mit einer bestimmten Frequenz bewegt. Je nach Frequenz erhält man eine stehende Welle mit ein oder mehreren Wellenbäuchen. Logischerweise sind bei höheren Frequenzen mehrere Wellenknoten und Wellenbäuche vorhanden als bei niedrigeren. Dadurch lässt sich wiederum auf die Frequenz, mit der das Seil bewegt wird, zurückschließen.

Analog dazu verhält sich die vom Österreicher E. Lecher erfundene symmetrische Doppelleitung (1890). Sie besteht aus zwei parallelen symmetrischen Drähten mit kreisförmigem Querschnitt und eine bestimmten Länge (abhängig von Lambda). Dadurch befindet

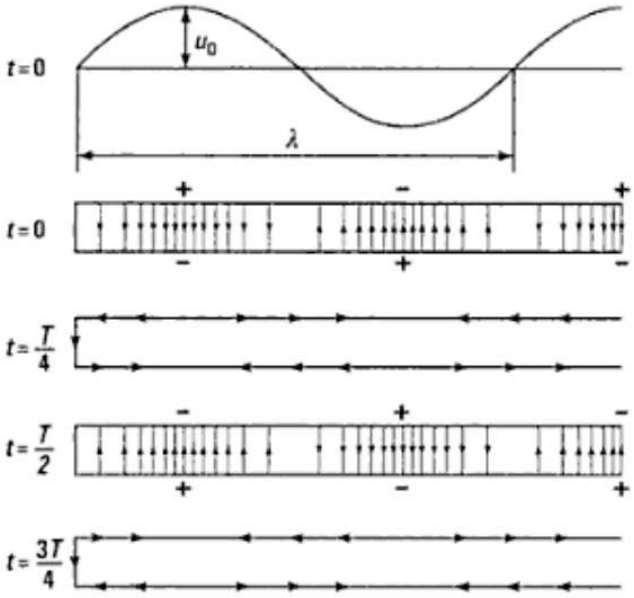


Abbildung 5.4: Strom- und Spannungsverteilung einer links kurzgeschlossene Lecher-Leitung zu verschiedenen Zeitpunkten [12]

sich die meiste Energie des elektromagnetischen Feldes zwischen den Leitern. Störungen durch die Umgebung können somit theoretisch vernachlässigt werden. Die Lecher-Leitung verhält sich wie der beschriebene Versuchsaufbau mit dem Seil (sowohl bei festgehaltenem als auch bei offenem Ende). Sind beispielsweise die zwei Doppelleitungen im Leerlauf, sprich offen, so wird die gesamte einfallende Welle ohne Phasensprung reflektiert. Einfallende und reflektierende Welle addieren sich und ergeben einen Spannungsbauch bzw. einen Stromknoten. Bei einer kurzgeschlossenen Leitung wird auf der Kurzschlussbrücke der Strom maximal, es entsteht ein Spannungsknoten. Es erfolgt eine Reflexion der Spannungswelle mit einem Phasensprung von 180°, welche sich somit am Kurzschluss zu null addieren. Wie auch beim Seil oszillieren hier die entstandenen stehenden Wellen. Dabei ändern die Spannungs- und Strombäuche ständig deren Vorzeichen.

Die Spannung kann als Funktion der Zeit und des Ortes x entlang der Lecher-Leitung angegeben werden. Dabei breitet sich die Welle mit einer Geschwindigkeit von v aus.

$$u(t,x) = u_0 \cdot sin2\pi f(t - \frac{x}{v}) \tag{5.2.1}$$

Ist die Frequenz, mit der die Lecher-Leitung angeregt wird, bekannt, kann nach dem Vermessen der stehenden Welle die Ausbreitungsgeschwindigkeit berechnet werden. Die

Wellenlänge wird hierbei aus den Abständen der Strom- und Spannungsbäuche ermittelt. Die Strombäuche werden mit einer kleinen Glühbirne und einer Spule detektiert. Spannungsbäuche beispielsweise mit einer Neonröhre die quer über die Leitungen gelegt wird. Der Maximalwert des Wellenbauchs entspricht dem Ort wo die jeweilige Lampe am hellsten aufleuchtet.

$$\lambda = \frac{c}{f} \tag{5.2.2}$$

Typische Schulversuchssysteme ermitteln auf diese Art die Lichtgeschwindigkeit (Ausbreitungsgeschwindigkeit der Welle) promillegenau. Nur aufgrund der Permittivität (Dielektrische Leitfähigkeit) der Luft (er= 1.0006) und des Skineffekts weicht diese von der Vakuumlichtgeschwindigkeit leicht ab.

### 5.2.2 Lecher-Leitung als Antenne

Im Allgemeinen wird eine Lecher-Leitung natürlich nicht als Antenne verwendet. Wie bereits erwähnt, konzentriert sich bei der Lecher-Leitung das elektrische und magnetische Feld im Raum zwischen den parallelen Leitern. Somit beschränken sich natürlich auch die Sende- bzw. der Empfangsbereich auf den Raum unmittelbar um die Paralleldrähte. Dennoch ist die Paralleldrahtleitung mit Antennen verwandt. Zwischen den Drähten der Lecher-Leitung herrscht ein homogenes transversales elektromagnetisches Feld. Am Ende der parallelen Drähte spreizen sich das Feld bzw. die Feldlinien jedoch nach außen hin aus. Klappt man nun beispielsweise eine am Ende offene Lecher-Leitung mit einer Länge von Lambda/2 um 90° auf, so erhält man den Hertzschen Dipol. Hier breitet sich das Wellenfeld senkrecht zur Achse aus. Die abgestrahlten Wellen dringen somit tief in den freien Raum ein und ermöglichen eine große Lese-Schreibreichweite. Wie auch bei der Lecher-Leitung ist die Spannung am Ende des Hertzschen Dipols maximal und der Strom gleich 0. An der Einspeisung, sprich Mitte des Dipols, ist der Strom maximal und die Spannung gleich 0.

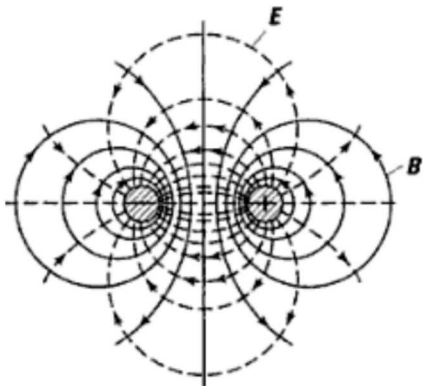


Abbildung 5.5: E- und H-Feld der Lecherleitung [12]

Für den Anwendungsfall der Bienendetektion scheint die Variante der Lecher-Leitung dennoch als ideal. Der standardmäßige Eingang des Bienenstocks entspricht einem

breiten Spalt mit geringer Höhe. Die Bienen passieren hier die sehr breite (vorher definierte) Detektionsfläche der Lecher-Leitung. Durch die geringe Höhe des Spalts reicht die beschränkten Lese-Schreibreichweite völlig aus.

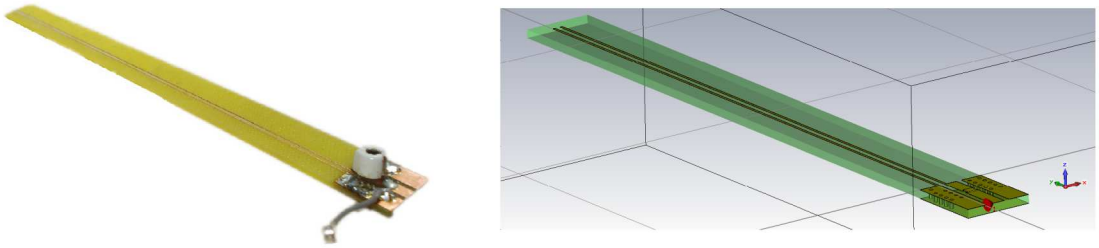

Abbildung 5.6: Gebaute und simulierte Lecher-Leitung

Für die Kommunikation zwischen dem Reader und den verwendeten OCA-Transpondern ist aufgrund der induktiven Kopplung nur das magnetische Feld von Bedeutung. Dieses soll auf der Detektionsfläche einen Maximalwert besitzen. Wie auch in Abbildung 5.5 dargestellt, zeigt die Simulation in Abbildung 5.10 den klaren Verlauf des H-Feldes. Logischerweise entsteht das magnetische Feld nur an der Stelle, an der ein Leiter mit Strom durchflossen wird. Das magnetische Feld verhält sich also proportional zur entstandenen stehende Welle. D.h. am offenen Ende der Lecher-Leitung ist das magnetische Feld minimal, das elektrische Feld aufgrund der anliegenden Spannung hingegen Maximal.

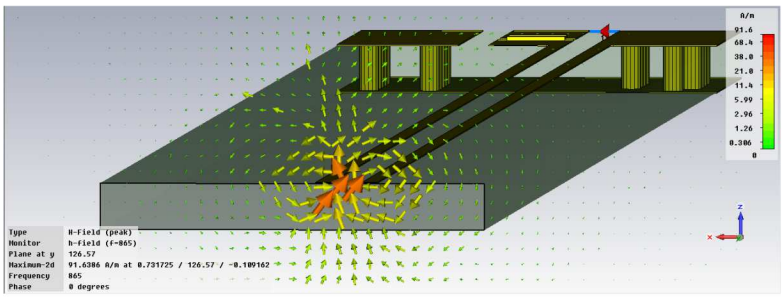


Abbildung 5.7: Magnetische Feldlinien der simulierten Lecher-Leitung

Um das magnetische Feld der gebauten Lecher-Leitung vermessen zu können, wurde einfach der Shuntsensor des OCA-Transponders verwendet. Er misst die, wie im Punkt 3.1 beschriebene, Shunt-Spannung, welche direkt proportional zur magnetischen Feldstärke

ist. Die Spannung kann aufgrund des beschränkten ADC-Bereichs jedoch nur zwischen 300 und 800 mV gemessen werden.

Die Messung erfolgte durch ändern der Position des OCA-Transponders. Bei einer eingestellten Reader-Ausgangsleistung von 20 dBm wurde der Transponder zwischen die Paralleldrahtleitung gelegt. Mit Hilfe der entwickelten Software aus Kapitel 4 wurde der Shuntwert des Transponders alle 5 mm ausgelesen und notiert. Tabelle 5.2 zeigt die aufgenommenen Werte.

| Abstand [mm] | Shuntwert [mV] | | Abstand [mm] | Shuntwert [mV] |
|---|---|---|---|---|
| 0 | 800 | | 95 | 520 |
| 5 | 760 | | 100 | 670 |
| 10 | 740 | | 105 | 770 |
| 15 | 710 | | 110 | 780 |
| 20 | 700 | | 115 | 780 |
| 25 | 700 | | 120 | 800 |
| 30 | 710 | | 125 | 790 |
| 35 | 680 | | 130 | 760 |
| 40 | 590 | | 135 | 710 |
| 45 | 500 | | 140 | 690 |
| 50 | 400 | | 145 | 670 |
| 55 | 300 | | 150 | 600 |
| 60 | 0 | | 155 | 450 |
| 65 | 0 | | 160 | 330 |
| 70 | 0 | | 165 | 300 |
| 75 | 0 | | 170 | 0 |
| 80 | 300 | | 175 | 0 |
| 85 | 410 | | 180 | 0 |
| 90 | 450 | | | |

Tabelle 5.2: Shuntwert des Transponders an bestimmten Positionen auf der Lecher-Leitung

Die nachfolgenden Bilder zeigen die magnetische Feldstärke entlang der Lecher-Leitung. Man erkennt, dass sich eine stehende Welle mit einer Wellenlänge von Lambda auf der Leitung ausbreitet. In den nachfolgenden Bildern wird die Z-Komponente des magnetischen Feldes betrachtet.

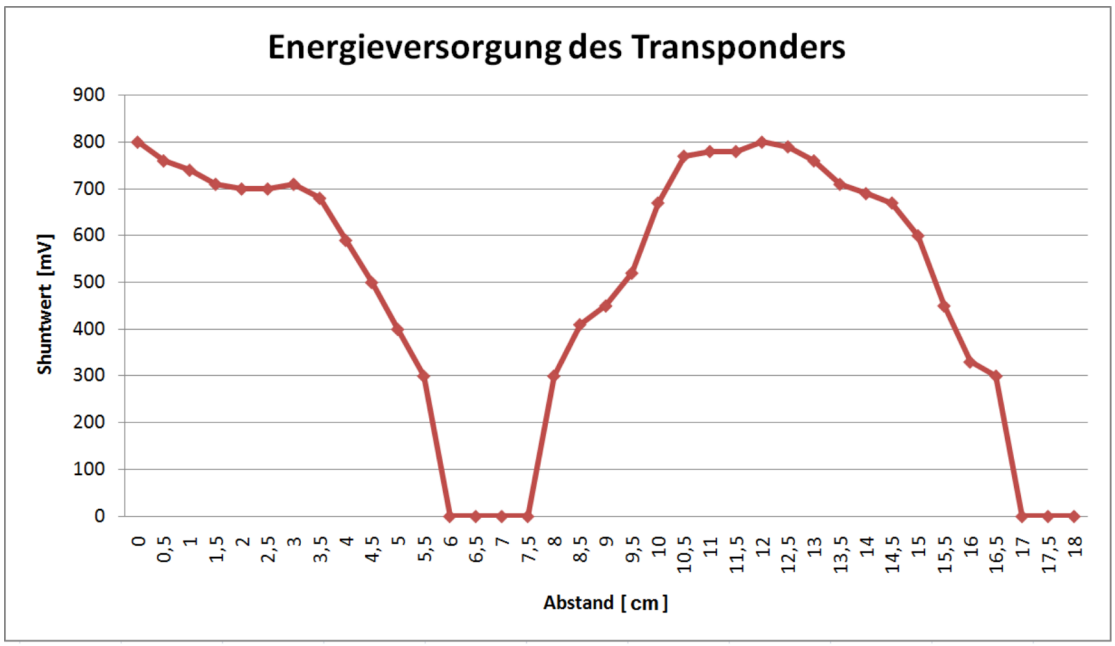


Abbildung 5.8: Diagramm der aufgenommenen Shuntsensorwerte

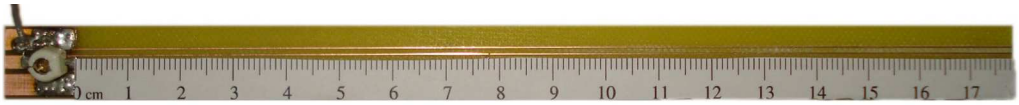


Abbildung 5.9: Lecherleitung

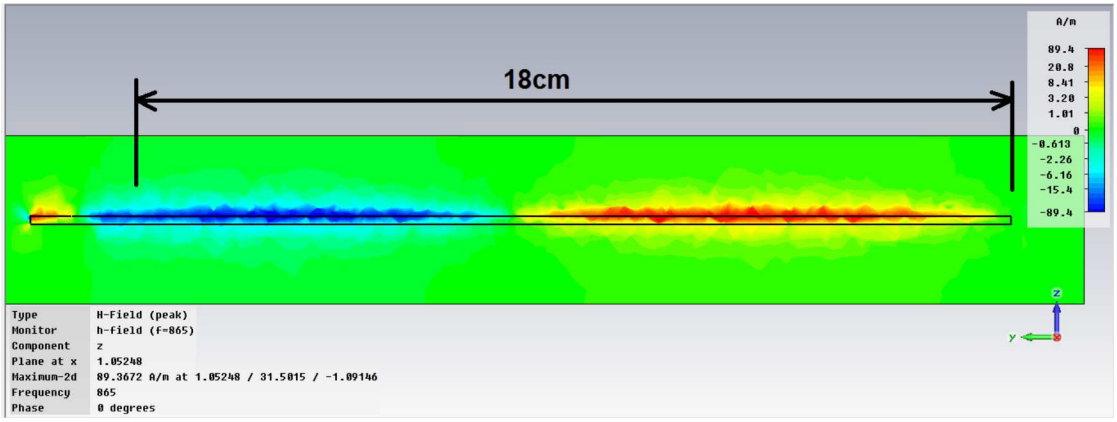


Abbildung 5.10: Simulierte magnetische Feldstärke der Lecherleitung (Seitenansicht)

Ermittelt man die Wellenlänge für eine Frequenz von 865 MHz (UHF) unter Verwendung der Lichtgeschwindigkeit so erhalt man $\lambda = 34,6\,cm$. Ein Wellenbauch, sprich $\lambda/2$, entspricht also einer theoretischen Länge von 17,3 cm. Misst man die Länge eines klar erkennbaren Wellenbauchs (siehe Abbildung 6.8 bzw. 6.10), kommt man auf nur 11 cm.

$$\lambda = \frac{c}{f} = \frac{300 \cdot 10^8 m/s}{865\,MHz} = 34,6\,cm \qquad \frac{\lambda}{2} = 17,3\,cm$$

Die numerische Lösung stimmt also weder mit der Simulation, noch mit der Vermessung der stehenden Welle überein. Die Annahme, Lichtgeschwindigkeit als Phasengeschwindigkeit zu verwenden, ist also falsch. Die Phasengeschwindigkeit verringert sich aber laut [12] in einem Medium mit $\epsilon r$>1 und $\mu$r=1, unabhängig von der Bauform der Doppelleitung, wie folgt.

$$v \approx (\epsilon_r \cdot \epsilon_0 \cdot \mu_0)^{-1/2} = \frac{c}{\sqrt{\epsilon_r}} \tag{5.2.3}$$

In den bereits beschriebenen Schulversuchen zur Messung der Lichtgeschwindigkeit stimmen die errechneten Wellenlängen mit den Gemessenen promillegenau überein, da Luft (ER=1.0006) das Ergebnis nur minimal verfälscht. Typische 1/v Werte von Lecher-Leitungen, die auf Dielektrika aufgebracht werden, liegen laut [12] bei 5 ns/m.

$$\frac{1}{v} = 5\frac{ns}{m} = 200 \cdot 10^8 \frac{m}{s}$$

$$\frac{200 \cdot 10^8 m/s}{865 MHz} = 23,1m \qquad \frac{\lambda}{2} = \underline{11,5cm}$$

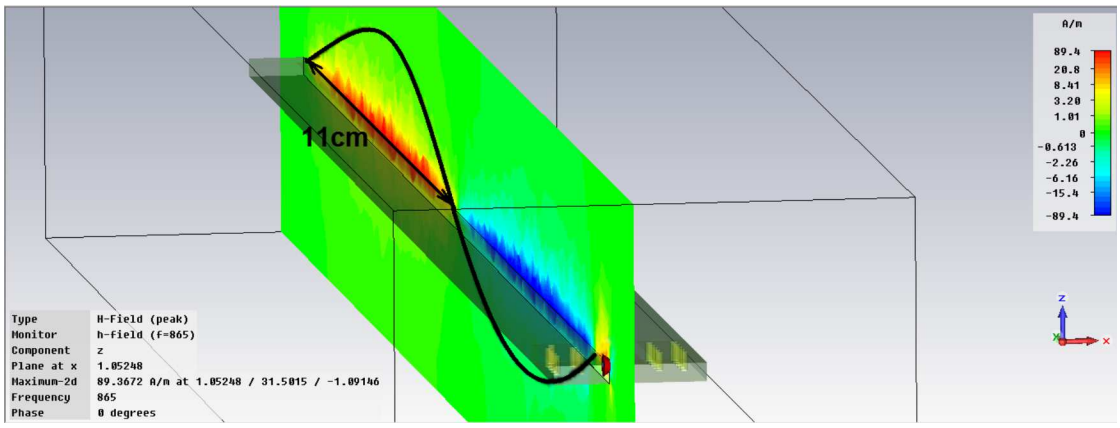


Abbildung 5.11: Stehende Welle

Unterschiedliche Materialien zwischen den parallelen Drähten der Lecher-Leitung rufen natürlich unterschiedliche Kapazitäten hervor. Dielektrische Stoffe, wie in unserem Fall FR4, erhöhend die Kapazität erheblich und verringern nun, wie erwiesen, die Ausbreitungsgeschwindigkeit. Die errechnete Wellenlänge von 11cm stimmt also nach der Berücksichtigung vom Dielektrikum mit der Messung und den Simulationen überein. Der Einfluss vom Dielektrikum auf die Wellenlänge $\lambda$ wird wie folgt ermittelt.

$$v = (\lambda - FR4) \cdot f \tag{5.2.4}$$

$$c \approx (\lambda - Luft) \cdot f \tag{5.2.5}$$

Setzt man die Gleichung 6.2.4 und 6.2.5 ins Verhältnis erhält man:

$$\frac{v}{c} = \frac{(\lambda - FR4) \cdot f}{(\lambda - Luft) \cdot f} \qquad v = \frac{\lambda - FR4}{\lambda - Luft} \tag{5.2.6}$$

Nach dem Einsetzen der Gleichung 6.2.3 erkennt man den Einfluss von Dielektrikum auf die Wellenlänge:

$$\epsilon = \frac{\lambda^2 - Luft}{\lambda^2 - FR4} \tag{5.2.7}$$

Die Dielektrizitätskonstante $\epsilon$ des Umgebungsmaterials (in unserem Fall FR4), unter der Annahme $\epsilon_{Luft} = 1$, ergibt sich also aus dem Verhältnisquadrat der gemessenen Wellenlänge. Abbildung 5.12 zeigt die durchgeführte Anpassung der beschriebenen Antenne.

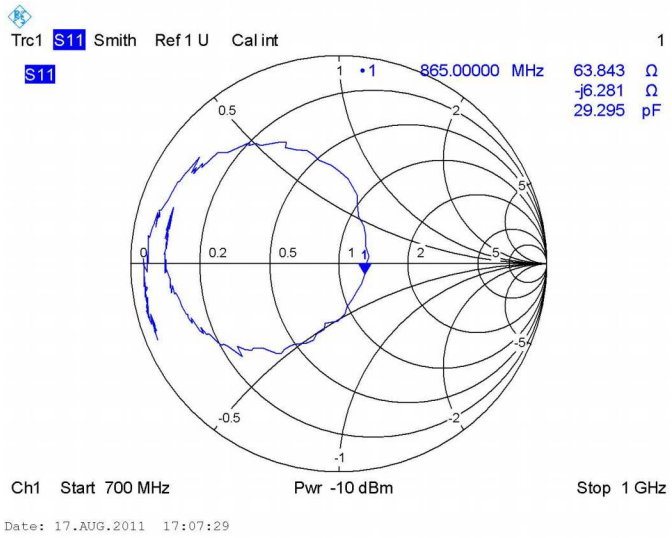


Abbildung 5.12: Anpassung der Lecher-Leitung

### 5.2.3 Demoaufbau

Für den Demoaufbau "Bienendetektion", wurde zu aller erst einen Biene mit dem OCA-Transponder versehen. Der OCA-Transponder wurde wie in der Abbildung, auf die Brust der Biene geklebt. Dazu wurde die Biene kurz mit einem Wollgitter an eine Oberfläche gedrückt, um den Tag mit einer Pinzette zu fixieren. Das Verhalten der Biene wurde dabei nicht verändert, da die Größe des Chips keinerlei Behinderungen darstellt. Das Gewicht ist durchaus mit einer größeren Ladung Pollen zu vergleichen, die eine Biene sammelt, um ihn als Nahrung für den Nachwuchs in den Bienenstock zu bringen.

Abbildung 5.13: Getaggte Biene

Anschließend wurde das Demosystem mit der Lecher-Leitung aufgebaut. Typischerweise landet eine Biene am Flugbrettchen des Bienenstocks und krabbelt durch den breiten Spalt ins Innere. Hierbei simuliert das Demosystem den Eingang des Bienenstocks, indem die Biene einfach unter der Lecher-Leitung hindurchkrabbelt.

Abbildung 5.14: Getaggte Biene

Krabbelt die Biene mit dem an ihr befestigten OCA-Transponder unter der Lecher-Leitung hindurch, wird diese vom Reader detektiert und im entwickelten LabView-Programm (aus Punkt 4.3) visuell dargestellt. Der Bereich zwischen den Wellenbäuchen (hier kann der OCA-Transponder, wie aus der vorherigen Messung ersichtlich, nicht detektiert werden) wird als Stütze verwendet. Die Detektionsfläche erstreckt sich von der Stütze aus nach beiden Seiten jeweils 11 cm. Sobald ein Transponder diese Bereiche passiert, wird er am PC angezeigt.

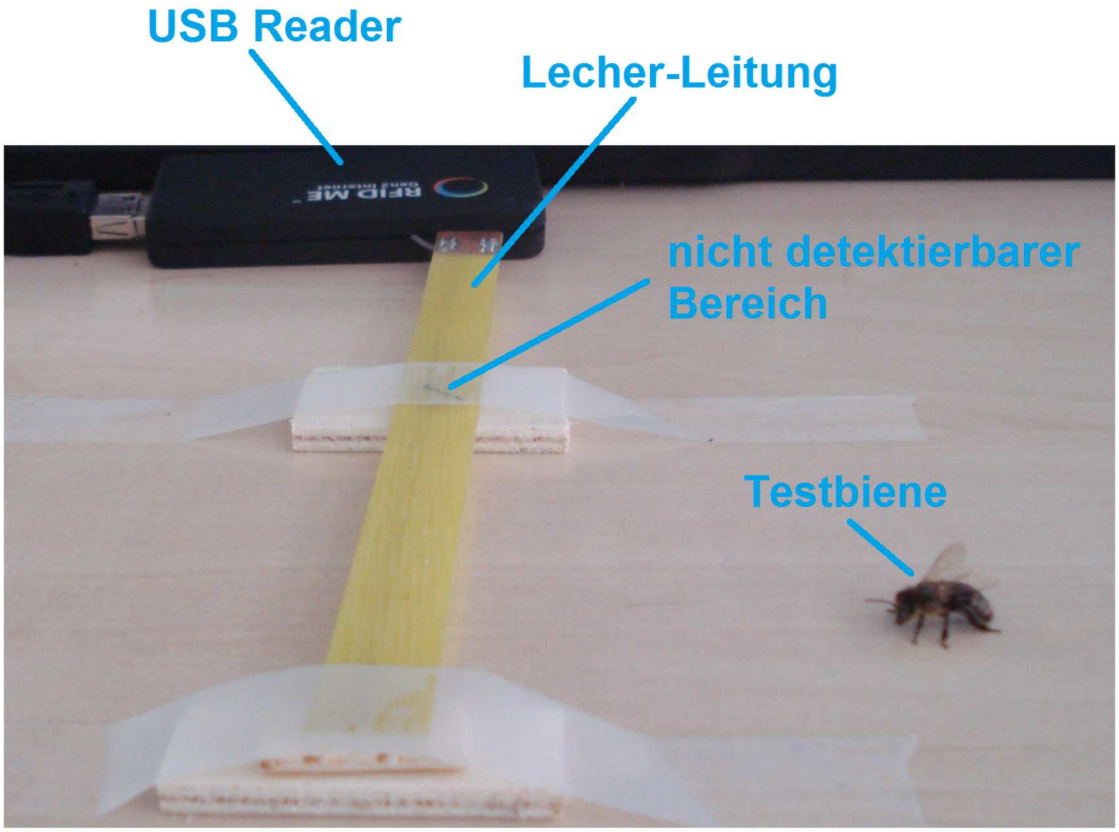


Abbildung 5.15: Bienenscanner

Zur Bienendetektion wird die Maximalleistung des Readers verwendet. 20 dBm bzw. 0,1 W haben wie auch beim Menschen keinerlei physikalischen Einfluss auf die Biene.

# Kapitel 6: Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, ein UHF-RFID-Komplettsystem zu entwerfen, welches in der Lage ist, mit einem von Infineon entwickelten OCA-Chip zu kommunizieren. Es ist zu betonen, dass es sich dabei um einen Test-Chip handelt und bislang keine Kommunikation mit dem Chip ausgetestet wurde.

Die Modifikation eines konventionellen UHF-RFID-USB-Readers von RF-IT-Solutions ermöglichte nach bereits kurzer Zeit die ersten Tests des OCA-Transponders. Unterschiedlich eingestellte Parameter und vor Allem die gut geeignete Nahfeldantenne spielten hier eine maßgebliche Rolle. Schnell wurde klar, dass sich zwischen den OCA-Transpondern starke Performanceunterschiede ergaben. Diese sind auf verschiedene Ausführungsform, Produktionsvariation und Störmaterial zurückzuführen. Werden die Transponder beispielsweise sehr genau entlang der On-Chip-Antenne zugeschnitten, sind sie performancetechnisch der etwas größeren Variante (hier umgibt die Antenne zur Sicherheit eine dünne Schicht von wenigen $\mu m$ aus Silizium und Alu) stark überlegen.

Nach erfolgreichem Aufbau des RFID-Systems mit den OCA-Transpondern konnten alle Funktionen des Chips getestet und verifiziert werden. So wurden beispielsweise auch alle Sensorwerte periodisch ausgelesen und grafisch dargestellt. Dadurch konnten Änderungen der Temperatur-und Shuntwert des Chips grafisch dargestellt werden. Der erwähnte Shuntsensor ermöglichte weiters die Analyse der verwendeten Nahfeldantenne, da der Sensorwert proportional zur Feldstärke der Reader-Antenne ist.

Schlussendlich sind zwei Demosysteme zur Veranschaulichung der Coil-On-Chip-Technologie entwickelt worden. OCA-Transponder, die auf Papier geklebt sind, ermöglichen im ersten Demosystem die Verifikation wertvoller Dokumente oder Geldscheine. Das zweite System zeigt den effektiven Einsatz dieser Technologie im Bereich der Insektenforschung. Hier werden Bienen mit dem sehr kleinen OCA-Transpondern ausgestattet und können mit der geeigneten Vorrichtung detektiert werden.

Das entwickelte System bietet neben der Kommunikation mit den Test-Chips von Infineon eine komplett einstellbare Hardwarekonfiguration des Readers. Somit ist Infineon in der Lage, auch die Änderungen und Weiterentwicklungen des Chips mit Hilfe dieses Systems weiterhin zu testen und zu verifizieren.

# Literaturverzeichnis


[1] A.Bigalke: Erstellen einer DLL-Datei zur Verwendung in LabView 8.2 mittels Microsoft Visual C++ 2005 Express Edition. online, August 2011. http://www.labviewforum.de/attachment.php?aid=6069.

[2] Berenbaum, M. R.: Zitat 2004. online, Juli 2011. http://www.adglossar.de/Insektenkunde.

[3] Cordic, A.: Untersuchung und Entwicklung eines induktiv gekoppelten Transpondersystems für On-Chip-Antennen-Systeme. Master Thesis at TU Graz, Institute of Highfrequency, 2011.

[4] Forum, U. I.: Device Class Definition for Human Interface Devices (HID). online, August 2011. http://www.usb.org/developers/devclass_docs/HID1_11.pdf.

[5] Gebhart, M.: Vorlesung RFID Systems. TU Graz, Institute of Communication Networks and Satellite Communications, Sommersemester 2009.

[6] Germany, G.: EPCglobal Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.0.9. online, Juli 2011. http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

[7] Hannes Reinisch, Stefan Gruber, G. H. M. K. W. P. a. G. H.: A Multifrequency Passive Sensing Tag with On-chip Temperature Sensor and Off-chip Sensor Interface Using EPC HF and UHF RFID Technology. Graz University of Technology, Institute of Electronics,Infineon Technologies Austria AG, Development Center Graz, 2011.

[8] Inc., S. L.: DataSheet of C8051F340/1/2/3/4/5/6/7/8/9/A/B/C/D. online, 2011. http://www.keil.com/dd/docs/datashts/silabs/c8051f34x.pdf.

[9] Instruments, N.: Was ist NI LabVIEW?. online, August 2011. http://www.ni.com/labview/whatis/d/.

[10] Laboratories, S.: USB Debug Adapter User Guide. online, 2011. http://www.silabs.com/Support%20Documents/TechnicalDocs/USB_Debug_Adapter_UG.pdf.

[11] Lichtenberg, G. C.: Zitat aus dem Sudelbuch G. online, Juli 2011. http://www.lichtenberg-gesellschaft.de/leben/l_wirk_sudel_02.html.

[12] Ludwig Bergmann, Clemens Schäfer, R. K. S. B. W. R.: Lehrbuch der Experimentalphysik: zum Gebrauch bei akademischen Vorlesungen und zum Selbststudium. Elektromagnetismus, Band 2. Juli 2011. http://books.google.de/books?id=62zjq-YPku4C&dq=Lehrbuch+der+Experimentalphysik+lecherleitung&source=gbs_navlinks_s.

[13] Microsoft: Windows Driver Kit (WDK). online, August 2011. http://msdn.microsoft.com/en-us/windows/hardware/gg487428.aspx.

[14] Microsystems, A.: DataSheet AS3992 UHF RFID Single Chip Reader EPC Class1 Gen2 Compatible. online, 2011. http://www.austriamicrosystems.com/Products/RF-Products/RFID/AS3992.

[15] Nationalbank, E.: Monatsbericht 2008. online, Juli 2011. http://www.ecb.europa.eu/pub/pdf/other/10thanniversaryoftheecbmb200806de.pdf.

[16] Nationalbank Österreichische: Falschgeld im Umlauf. online, Juli 2011. http://www.oenb.at/.

[17] Reider, G.: Das Ganze ist mehr als das Summen seiner Einzelteile. online, Juli 2011. http://augenblicke.blogger.de/topics/natur/.

[18] Wiessflecker, M.: Machbarkeit, Entwurf und Verifikation von integrierten Sensoren. Master Thesis at TU Graz, Institute of Elektronik, Februar 2010.