|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumentation**  für  **RF-ID** | | |  |
|  | | | |
| Klasse | Teammitglied | Signature | |
| 4AHELS | Patrik Staudenmayer |  | |
| Abgabedatum | Teammitglied | Signature | |
|  | Marie Maier |  | |
| Lehrer | Teammitglied | Signature | |
| Tillich, Gruber, Crha | - |  | |
| Note | Teammitglied | Signature | |
|  | - |  | |
| Projektbeschreibung  Aufbau und Programmierung eines RF-ID Tags und Lesegerät | | | |
| USED DEVICES   |  |  |  |  | | --- | --- | --- | --- | | Nummer | Gerät | Firma | Typ | | 1 | Labornetzteil |  |  | | 2 | Oszilloskope |  |  | | | | |
|  | | | |
| Cover Sheet E2014 v3 | | | |

Inhaltsverzeichniss

[1 Hardware 2](#_Toc9949948)

[1.1 Lesegerät 2](#_Toc9949949)

[1.1.1 Schaltung 2](#_Toc9949950)

[1.1.2 Berechnungen 3](#_Toc9949951)

[1.1.2.1 Resonanzfrequenz RC-Oszillator 3](#_Toc9949952)

[1.1.2.2 Resonanzfrequenz Schwingkreis 3](#_Toc9949953)

[1.2 Tag 4](#_Toc9949954)

[1.2.1 Schaltung 4](#_Toc9949955)

[1.2.2 Berechnungen 5](#_Toc9949956)

[1.2.2.1 Schwingkreis 5](#_Toc9949957)

[1.2.3 Messungen 6](#_Toc9949958)

[1.2.3.1 Übertragungsfunktion der Spannungsversorgung des Tags 6](#_Toc9949959)

[1.2.3.1.1 Messschaltung 6](#_Toc9949960)

[1.2.3.1.2 Messung 6](#_Toc9949961)

[1.3 Übertragung zwischen Lesegerät und Tag 7](#_Toc9949962)

[1.3.1 Übertragung von Tag zu Reader 7](#_Toc9949963)

[1.3.2 Übertragung von Reader zu Tag 7](#_Toc9949964)

[2 Software 8](#_Toc9949965)

[2.1 Aufgabenstellung 8](#_Toc9949966)

[2.2 Initialisierung 9](#_Toc9949967)

[2.3 Senderoutine 9](#_Toc9949968)

[2.3.1 Senderoutine Starten 9](#_Toc9949969)

[2.3.2 Timer der Senderoutine 10](#_Toc9949970)

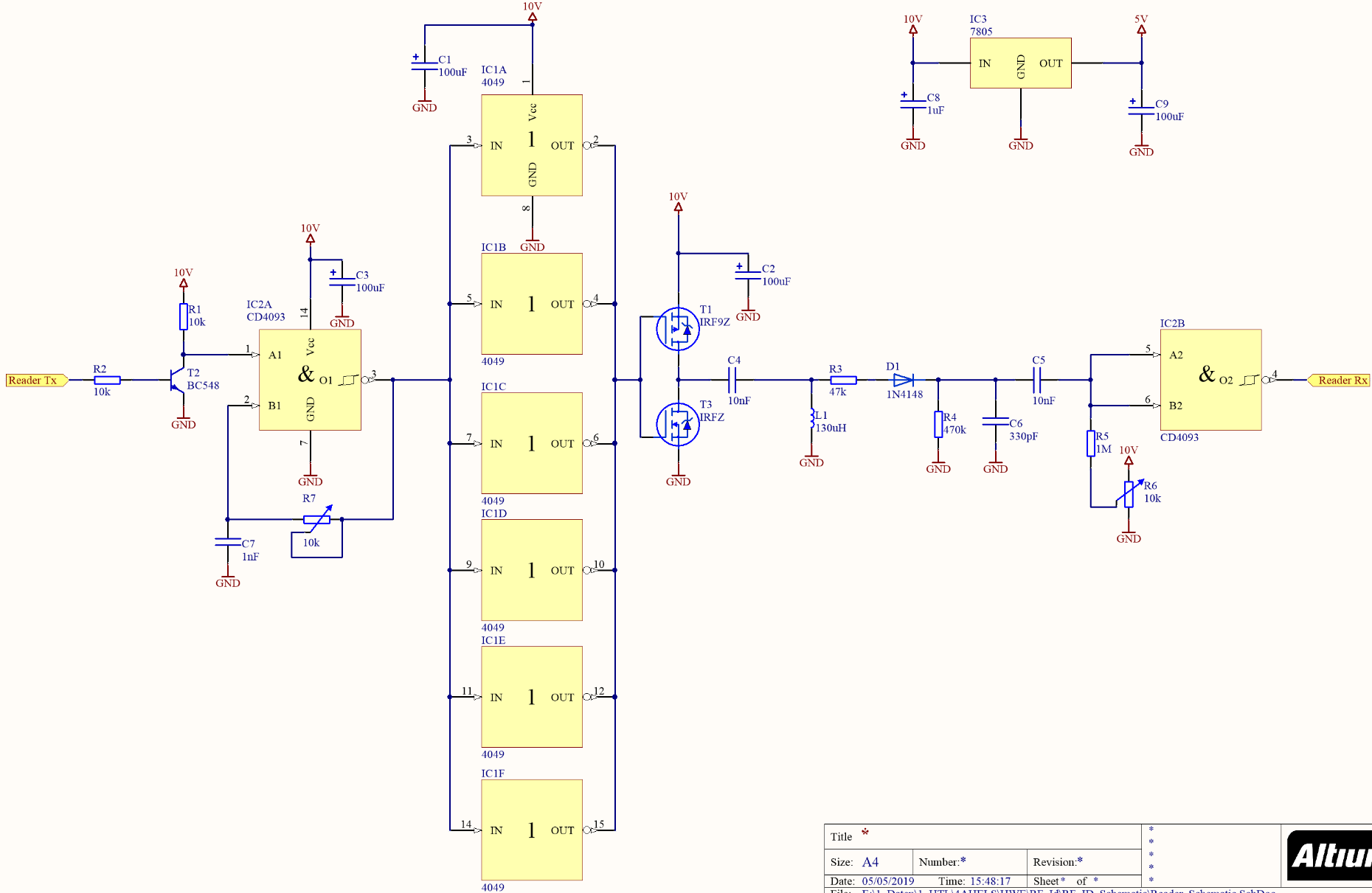
[2.3.2.1 Berechnung der Timer Parameter 10](#_Toc9949971)

[2.3.2.2 Interruptroutine des Timer 3 Capture Interrupt 11](#_Toc9949972)

# Hardware

## Lesegerät

### Schaltung



Endstufe

Treiber

Demodulator

Schwingkreis

RC-Oszillator ()

Abbildung 1: Schaltung Lesegerät



Abbildung 2: Schaltung Lesegerät Daten einlesen

Zum einlesen der Daten wird ein 8 Bit Schieberegister verwendet. Nachdem nur 7 Bits benötigt werden wurden nur sieben der acht Parallelen Ausgänge verwendet.

### Berechnungen

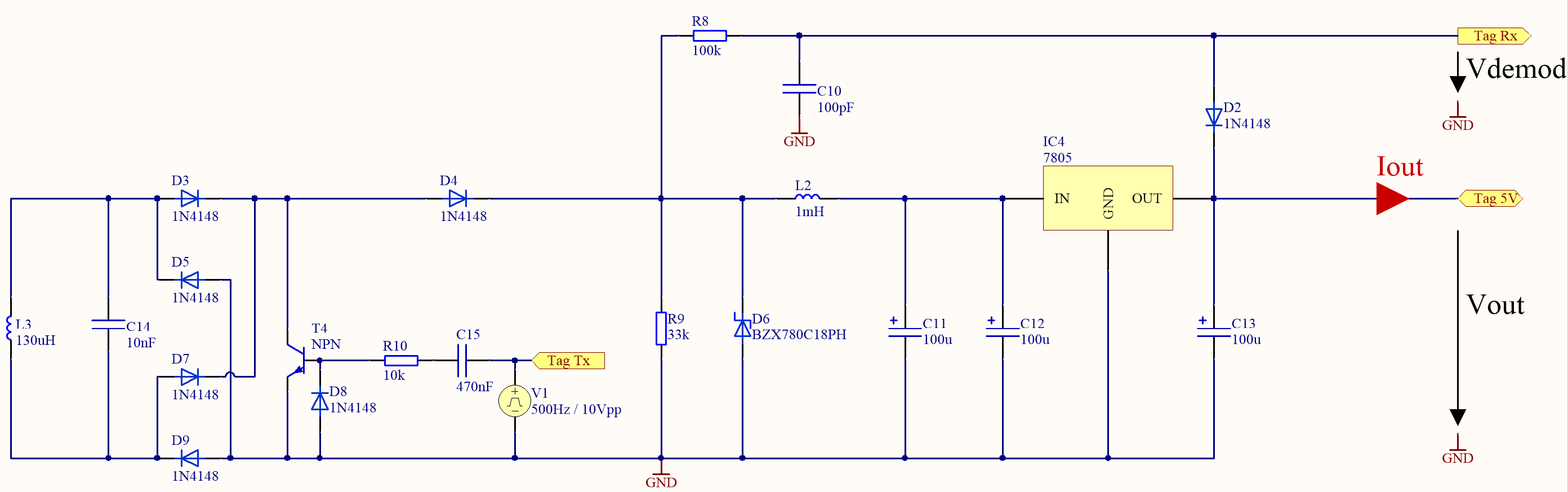
#### Resonanzfrequenz RC-Oszillator

#### Resonanzfrequenz Schwingkreis

Ein 10nF Kondensator wurde verwendet, da die Oszillatorfrequenz mit dem Potentiometer abgeglichen werden kann.

## Tag

### Schaltung



Längsregler 5V

Spannungsbegrenzung

Lastmodulator

Gleichrichter

Schwingkreis

Abbildung 3: Schaltung Tag

### Berechnungen

#### Schwingkreis

Ein 10nF Kondensator wurde verwendet, da die Oszillatorfrequenz mit dem Potentiometer am Lesegerät abgeglichen werden kann.

### Messungen

#### Übertragungsfunktion der Spannungsversorgung des Tags

##### Messschaltung

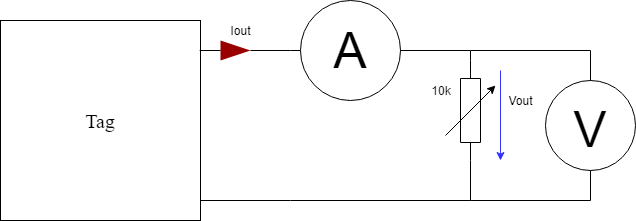


Abbildung 4: Messchaltung für Übertragungsfunktion Spannungsversorgung des Tags

Durch Variierung des 10kΩ Widerstands wurde die Last simuliert.

Abbildung 5: Messanordnung der Spulen

Spule Lesegerät

Plastik Abstandhalter

Spule Tag

5mm

##### Messung

|  |  |
| --- | --- |
| Vou | Iout |
| 5.60 V | 0.04 mA |
| 5.60 V | 6.63 mA |
| 5.60 V | 13.74 mA |
| 5.00 V | 18.74 mA |
| 4.60 V | 20.67 mA |
| 2.60 V | 23.24 mA |
| 0.60 V | 26.11 mA |

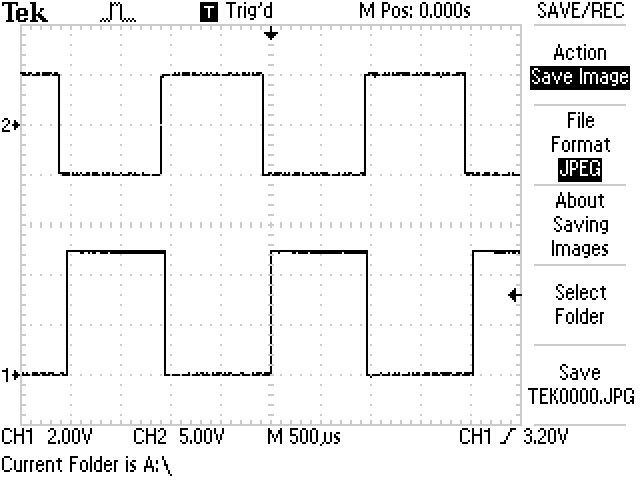
Abbildung 6: Messtabelle Übertragungsfunktion Spannungsversorgung des Tags

Der ATmega32u4 benötigt 14mA bei einer Taktfrequenz von 16MHz und einer Versorgungsspannung von 5V. Dies kann gerade so abgedeckt werden.

Abbildung 7: Diagramm Übertragungsfunktion Spannungsversorgung des Tags

## Übertragung zwischen Lesegerät und Tag

### Übertragung von Tag zu Reader

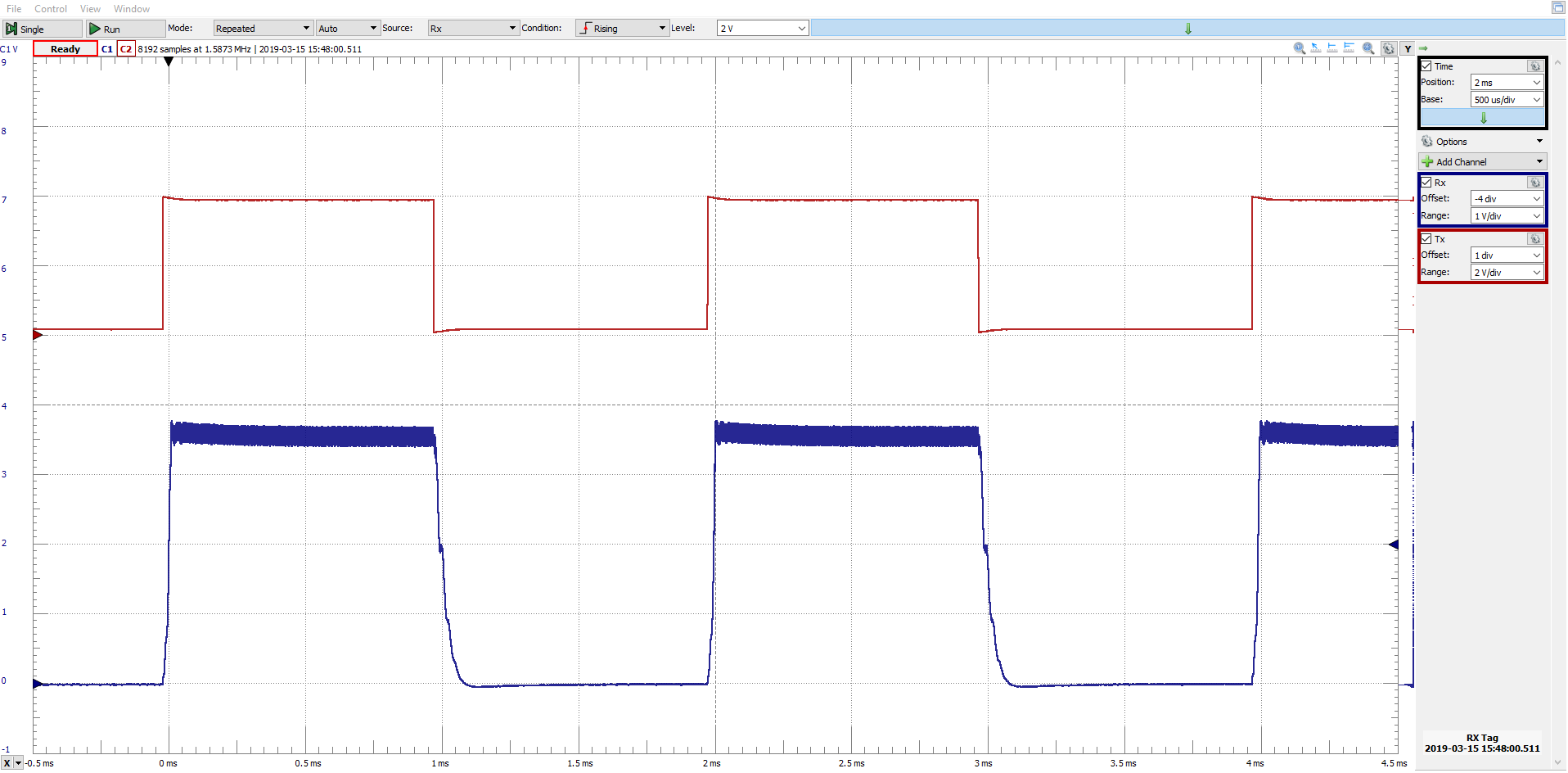


Tag RX

Tag TX

Abbildung 8: Oszillogramm Übertragung von Tag zu Reader

### Übertragung von Reader zu Tag



Tag Rx

1V/division

Reader Tx

2V/division

Abbildung 9: Oszillogramm Übertragung von Reader zu Tag

# Software

Das verwendete Microcontrollerboard ist der „Flipp“ der HTBLuVA St.Pölten Abteilung für Elektronik und technische Informatik, welcher den ATmega32u4 Microcontroller beherbergt.

## Aufgabenstellung

Es soll eine Datenübertragung zwischen zwei Geräten mit folgenden Eigenschaften stattfinden:

* Zum einlesen Der Daten auf einer Seite soll ein SIPO (Seriell In, Parallel Out) Schieberegister verwendet werden.
* Die Daten sollen MSB First gespeichert werden.
* Zum Senden sollen die Daten geschoben werden.
* Die Bitdauer soll 1,37ms betragen.
* Es soll das MSB zuerst gesendet werden.
* Zur Synchronisierung sollen zwei Startbits gesendet werden.
* Zur Datensicherung soll der Hamming Code verwendet werden.

Hamming (11/4) … insgesamt 11 Bit und davon 4 Paritätsbits

* Als Leitungscode soll die Manchesterkodierung verwendet werden

## Initialisierung

## Manchester Kodierung

## Hamming Code

## Senderoutine

### Senderoutine Starten

Um die Senderoutine zu starten wird das Makro RF\_ID\_SEND (Code 1) verwendet. Diese führt die in der Abbildung 7 dargestellten Tätigkeiten durch. Hierbei ist zur Implementierung anzumerken, dass, wenn die Übergabe der Daten mittels Globaler Variablen implementiert wird, welche sowohl im Main als auch in der Library verwendbar sein sollen, so muss das C Schlüsselwort extern verwendet werden.

Variablen, die mit dem Schlüsselwort extern deklariert werden, werden bei der Deklaration noch kein Speicher zugewiesen, dies geschieht erst bei der Definition welche bei der hier gewählten Implementation in dem „RF\_ID.c“ File stattfindet.

Um die Synchronität zu gewährleisten wird am Anfang einer jeden Transaktion die Bits „11“ Manchester kodiert gesendet.

Abbildung 10: Flussdiagram des Sende Makros

Code 1: Makro zum Starten der Senderoutine

/\*

\*@function RfIDsend

\*@abstract Handles the transmission of the given data

\*@discussion Calculates the Hamming code of the given and encodes it to Manchester code then handles the correct set up of the timer

\* and assigns the encoded data to GsendData.

\*@param data Is the pointer to the transmitted data

\*/

#define RF\_ID\_SEND(data) ({ \

for(int i = 0; i < TX\_DATA\_LENGTH; i++) \

{ \

GsendData[i] = data[i]; \

} \

GsendCounter = 0; \

\

TCCR3B |= (1 << CS30); /\* Start Timer3 (Set divider to 1) \*/ \

})

### Timer der Senderoutine

Um eine fixe Bitrate zu gewährleisten wurde der Timer 3 des ATmega32u4 im CTC Modus verwendet. Ein Bit wird gesendet, wenn dieser Timer den Topwert erreicht hat, welcher mittels des ICR3 Registers festgelegt wurde und welcher beim Erreichen den Timer 3 Capture Interrupt auslöst, wenn dieser freigeschalten ist.

#### Berechnung der Timer Parameter

Anforderung: Es soll alle 1,37ms ein Bit gesendet werden.

Der Microcontroller wird mit einer von 16MHz betrieben.

Mithilfe von Excel wurde folgende Tabelle erstellt:

gewählt

|  |  |
| --- | --- |
| Teiler N | ICR3 |
| 1 | 10959.00 |
| 8 | 1369.00 |
| 64 | 170.25 |
| 256 | 41.81 |
| 1024 | 9.70 |

Abbildung 11: Mögliche Werte für den Timer der Senderoutine

Die Kriterien, die hier erfüllt werden mussten war:

* , da Timer 3 ein 16 Bit Timer ist.
* , um falls nötig die Grenze noch verstellen zu können.

#### Interruptroutine des Timer 3 Capture Interrupt

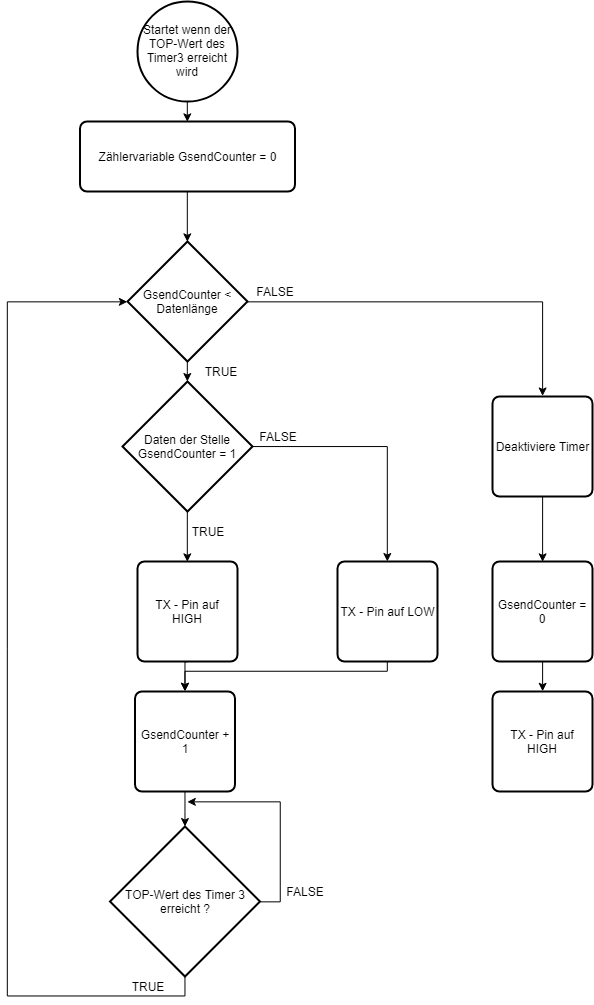
In Abbildung 8 ist dargestellt wie das senden in der Interrupt Routine bewerkstelligt wird.

Abbildung 12: Flussdiagram Interrupt Routine Timer 3 Capture Interrupt

Code 2: Sende Interrupt Routine des Timer 3 Capture Interrupt

ISR(TIMER3\_CAPT\_vect)

{

// Send Interrupt

if (GsendCounter < TX\_DATA\_LENGTH)

{

if(GsendData[GsendCounter] == 1)

{

TX\_PORT |= TX\_PIN\_MASK;

}

else

{

TX\_PORT &= ~TX\_PIN\_MASK;

}

GsendCounter++;

}

else

{

GsendCounter = 0;

TCCR3B &= ~(1 << CS32) & ~(1 << CS31) & ~(1 << CS30);

// Deactivate Timer3

TCNT3 = 0; // Reset Timer3

TX\_PORT |= TX\_PIN\_MASK;

}

}

## Empfangsroutine

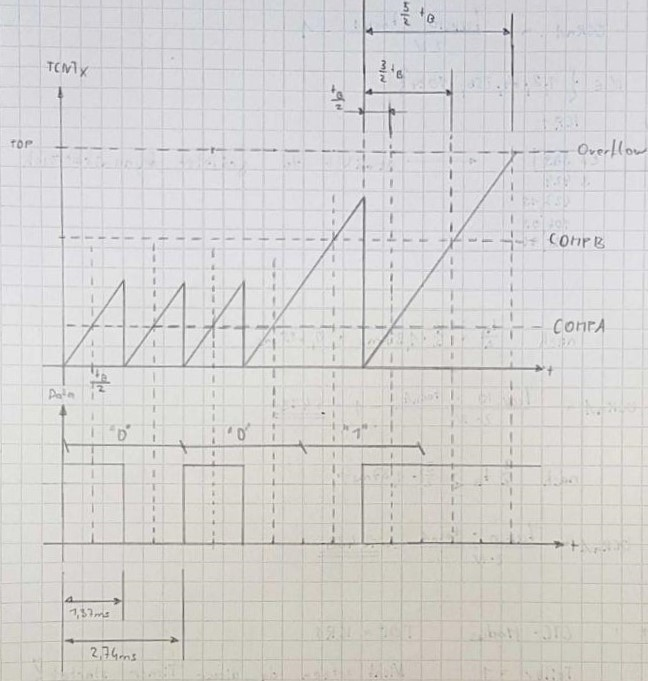


Abbildung 13: Timing Diagramm der Empfangsroutine

Zum Empfangen der gesendeten Daten wird der 16-Bit Timer 1 im CTC-Modus verwendet. Der COMPA Interrupt wird verwendet, um das erste Bit des Symbols zu empfangen, der COMPB Interrupt wird verwendet, um das erste Bit des darauffolgenden Symbols zu empfangen, falls die beiden verschiedene Symbole, des Manchester Codes, hintereinander gesendet wurden. Der Overflow Interrupt dient zum Erkennen des Endes der Datenübertragung.

Der Pin-Change Interrupt INT3 – PD3 des ATmega32u4 – wird verwendet, um den Sender und Empfänger Synchron zu halten. Im Interrupt wird bei einer Flanke das Counter Register des Timer 1 auf 0 gesetzt und dadurch ist die Synchronität des Sender und Empfängers gewährleistet.

### Timer der Empfangsroutine

#### Berechnung der Timer Parameter

Anforderung:

* COMPA soll nach 685ns erreicht werden.
* COMPB soll nach 2,055ms erreicht werden.
* TOP soll nach 3,425ms erreicht werden.

Der Microcontroller wird mit einer von 16MHz betrieben.

Mithilfe von Excel wurden folgende Tabellen erstellt:

gewählt

|  |  |
| --- | --- |
| Divider N | OCRnA (Register zum Festlegen der Grenzte für den COMPA Interrupt) |
| 1 | 5479.00 |
| 8 | 684.00 |
| 64 | 84.63 |
| 256 | 20.41 |
| 1024 | 4.35 |

Abbildung 14: Mögliche Werte für den Wert des OCRnA Registers

gewählt

|  |  |
| --- | --- |
| Divider N | OCRnB (Register zum Festlegen der Grenze für den COMPB Interrupt) |
| 1 | 16439.00 |
| 8 | 2054.00 |
| 64 | 255.88 |
| 256 | 63.22 |
| 1024 | 15.05 |

Abbildung 15: Mögliche Werte für den Wert des OCRnB Registers

gewählt

|  |  |
| --- | --- |
| Divider N | ICRn (Register zum Festlegen der Grenze für den Input Capture Interrupt) |
| 1 | 27399.00 |
| 8 | 3424.00 |
| 64 | 427.13 |
| 256 | 106.03 |
| 1024 | 25.76 |

Abbildung 16: Mögliche Werte für den Wert des ICRn Registers

### Flussdiagram der Empfangsroutine

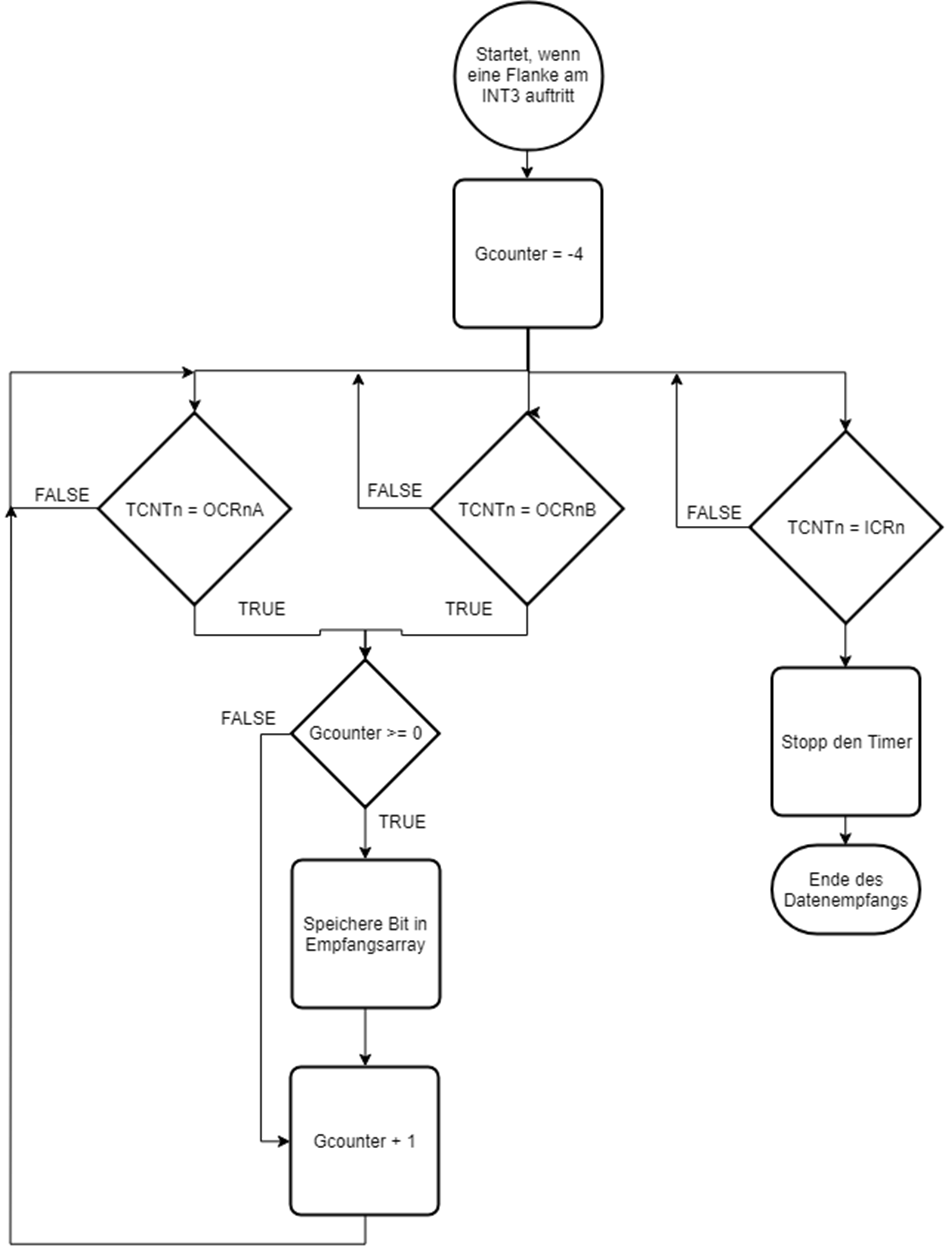


Abbildung 17: Flussdiagram der Empfangsroutine

### Code der Empfangsroutine

Code 3: Implementierung der Empfangsroutine

ISR(TIMER1\_COMPA\_vect)

{

// First routine to be triggered when something received

if (Gcounter >= 0)

{

GsaveD[Gcounter] = RX\_PORT & RX\_PIN\_MASK;

}

Gcounter++;

}

ISR(TIMER1\_COMPB\_vect)

{

// Second routine to be triggered when something received

if (Gcounter >= 0)

{

GsaveD[Gcounter] = RX\_PORT & RX\_PIN\_MASK;

}

Gcounter++;

}

ISR(TIMER1\_CAPT\_vect)

{

// Ends the transaction

Gcounter = -4;

RXflag = TRUE;

TCCR1B = TCCR1B & ~(1<<CS10) & ~(1<<CS11) & ~(1<<CS12); //Timer1 stopped

}