|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumentation**  für  **RF-ID** | | |  |
|  | | | |
| Klasse | Teammitglied | Signature | |
| 4AHELS | Patrik Staudenmayer |  | |
| Abgabedatum | Teammitglied | Signature | |
| 07.06.2019 | 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](#_Toc11067740)

[1.1 Lesegerät 2](#_Toc11067741)

[1.1.1 Schaltung 2](#_Toc11067742)

[1.1.3 Berechnungen 4](#_Toc11067743)

[1.1.3.1 Resonanzfrequenz RC-Oszillator 4](#_Toc11067744)

[1.1.3.2 Resonanzfrequenz Schwingkreis 4](#_Toc11067745)

[1.2 Tag 5](#_Toc11067746)

[1.2.1 Schaltung 5](#_Toc11067747)

[1.2.2 Berechnungen 6](#_Toc11067748)

[1.2.2.1 Schwingkreis 6](#_Toc11067749)

[1.2.3 Messungen 7](#_Toc11067750)

[1.2.3.1 Übertragungsfunktion der Spannungsversorgung des Tags 7](#_Toc11067751)

[1.2.3.1.1 Messschaltung 7](#_Toc11067752)

[1.2.3.1.2 Messung 7](#_Toc11067753)

[1.3 Übertragung zwischen Lesegerät und Tag 8](#_Toc11067754)

[1.3.1 Übertragung von Tag zu Reader 8](#_Toc11067755)

[1.3.2 Übertragung von Reader zu Tag 8](#_Toc11067756)

[2 Software 9](#_Toc11067757)

[2.1 Aufgabenstellung 9](#_Toc11067758)

[2.2 Initialisierung 10](#_Toc11067759)

[2.3 Manchester Kodierung 11](#_Toc11067760)

[2.4 Hamming Code 12](#_Toc11067761)

[2.5 Einlesen und Speichern der Daten 13](#_Toc11067762)

[2.6 Senderoutine 15](#_Toc11067763)

[2.6.1 Senderoutine Starten 15](#_Toc11067764)

[2.6.2 Timer der Senderoutine 16](#_Toc11067765)

[2.6.2.1 Berechnung der Timer Parameter 16](#_Toc11067766)

[2.6.2.2 Interruptroutine des Timer 3 Capture Interrupt 17](#_Toc11067767)

[2.7 Empfangsroutine 19](#_Toc11067768)

[2.7.1 Timer der Empfangsroutine 20](#_Toc11067769)

[2.7.1.1 Berechnung der Timer Parameter 20](#_Toc11067770)

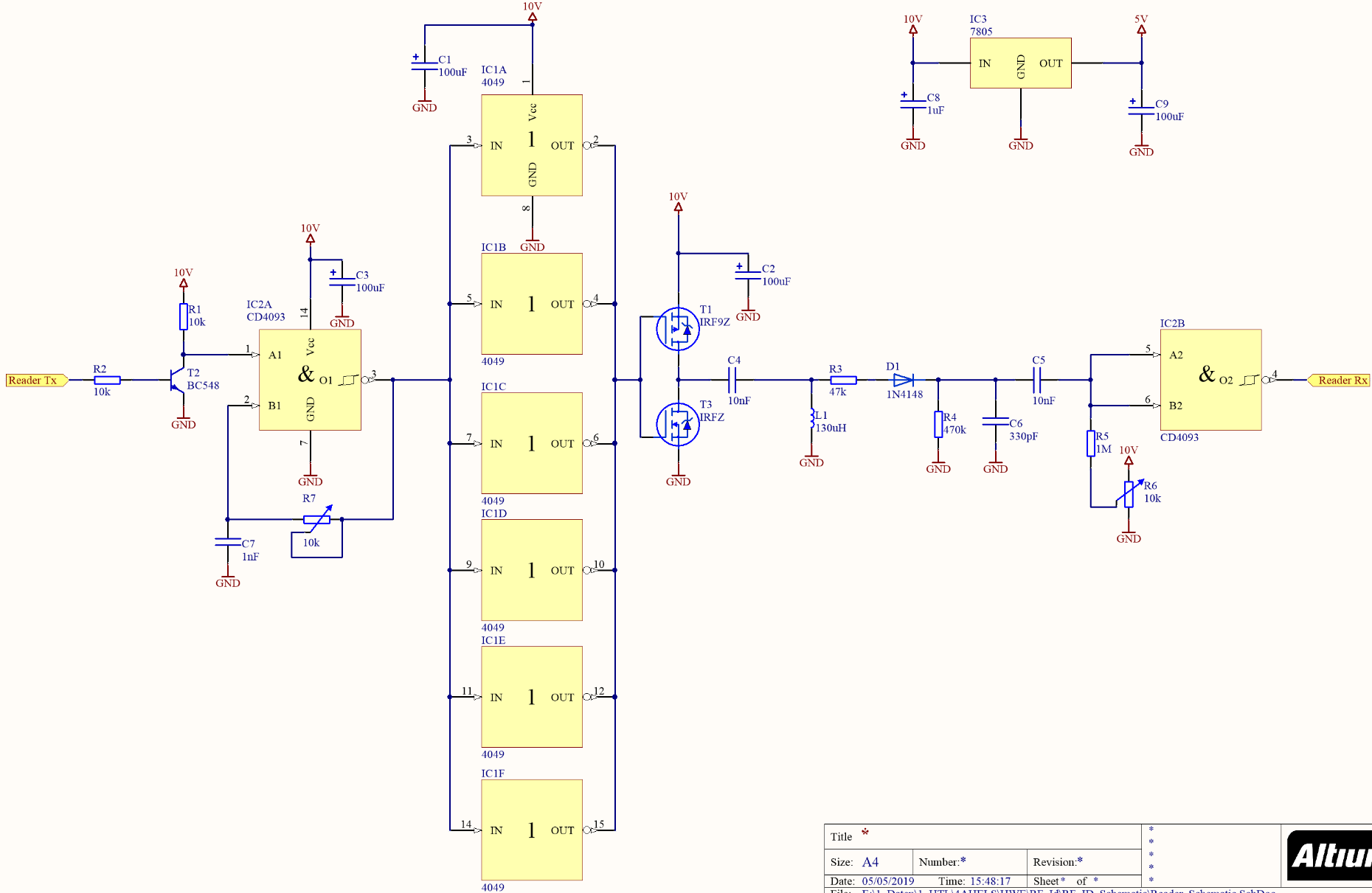
[2.7.2 Flussdiagram der Empfangsroutine 21](#_Toc11067771)

[2.7.3 Code der Empfangsroutine 22](#_Toc11067772)

# Hardware

## Lesegerät

### Schaltung



Endstufe

Treiber

Demodulator

Schwingkreis

RC-Oszillator ()

Abbildung : Schaltung Lesegerät



Abbildung : 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, des Schieberegister IC5 verwendet.

### Berechnungen

#### Resonanzfrequenz RC-Oszillator

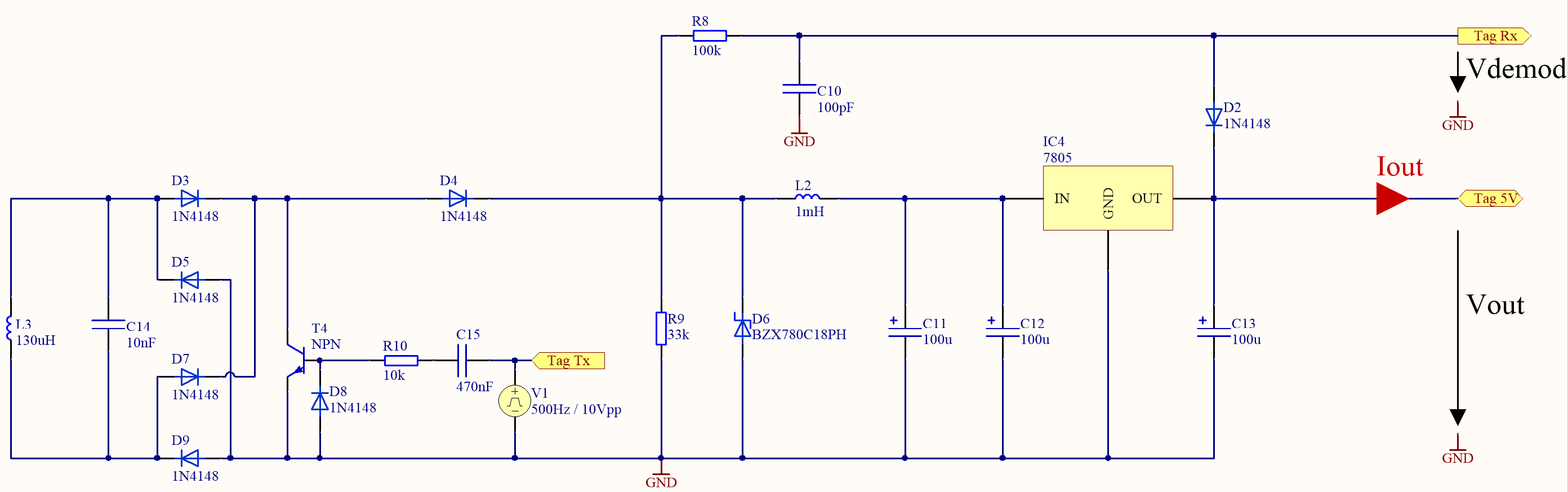
Durch Testen wurde festgestellt das ein 10k Potentiometer ausreicht, zum einstellen der gewünschten Resonanzfrequenz.

#### 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 : 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

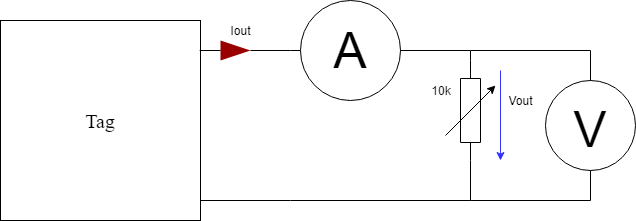


Abbildung : Messchaltung für Übertragungsfunktion Spannungsversorgung des Tags

Durch Variierung des 10kΩ Widerstands wurde die Last simuliert.

Abbildung : 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 |

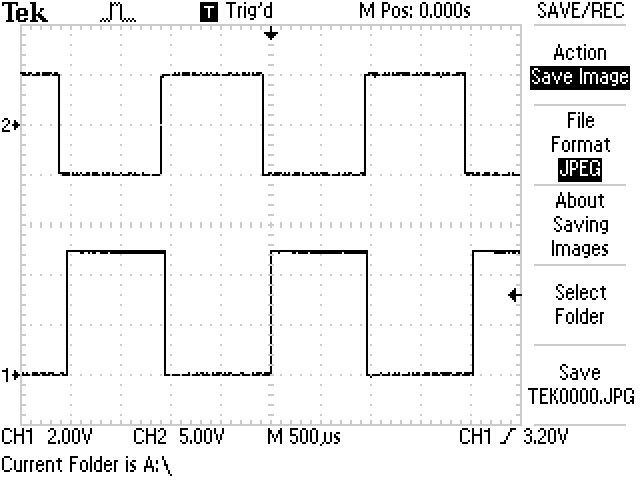
Abbildung : Messtabelle Übertragungsfunktion Spannungsversorgung des Tags

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

Abbildung : Diagramm Übertragungsfunktion Spannungsversorgung des Tags

## Übertragung zwischen Lesegerät und Tag

### Übertragung von Tag zu Reader

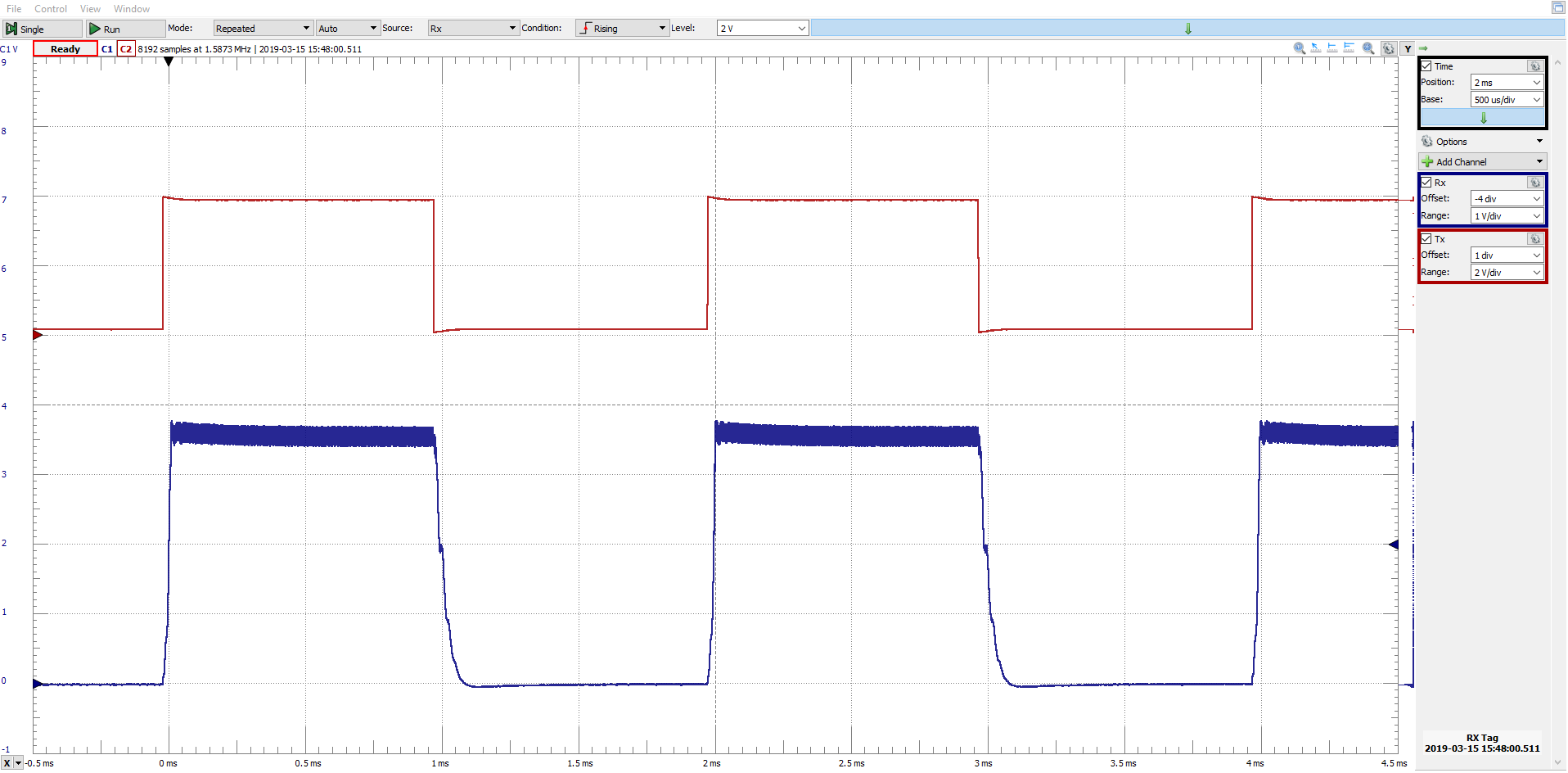


Tag RX

Tag TX

Abbildung : Oszillogramm Übertragung von Tag zu Reader

### Übertragung von Reader zu Tag



Tag Rx

1V/division

Reader Tx

2V/division

Abbildung : 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

// RX instantiation

char GsaveD[RX\_DATA\_LENGTH + 1] = {0};

*int8\_t* Gcounter = -4;

// TX instantiation

char GsendData[TX\_DATA\_LENGTH] = {0};

*int8\_t* GsendCounter = 0;

char RXflag = FALSE;

void RfIDinit()

{

// Initialize Receiver Port

RX\_PORT\_DDR &= ~RX\_PIN\_MASK;

// Initial Transmitter Port

TX\_PORT\_DDR |= TX\_PIN\_MASK;

TX\_PORT |= TX\_PIN\_MASK; // Set Tx to high to deliver energy

// Initialize Input port for shift register

DATA\_INPUT\_PORT\_DDR = 0x00;

// Initialize Pin change interrupt

DDRD &= ~(1 << DDD3); // Initialize PD3 as Input

EICRA |= (1<<ISC31) | (1<<ISC30); // Trigger interrupt on rising and

falling edge

EIMSK |= (1<<INT3); // Locally enable Interrupt for INT3

// Initialize Timer1 for receiving

TCCR1B = TCCR1B | (1<<WGM13) | (1<<WGM12); // Set Timer1 to CTC-Mode with ICR1

as TOP

ICR1 = fICR; // Set TOP Value

OCR1A = fOCR1A; // Set threshold for COMP\_A

OCR1B = fOCR1B; // Set threshold for COMP\_B

TIMSK0 = TIMSK0 | (1<<OCIE0B) | (1<<OCIE0A) | (1<<ICIE1); //COMP\_A, COMP\_B and

Input capture as overflow

// Initialize Timer3 for sending

TCCR3B = TCCR3B | (1<<WGM33) | (1<<WGM32); // Set Timer1 to CTC-Mode with ICR3

as TOP

ICR3 = TOP\_TX; // Set TOP Value

TIMSK3 |= (1 << ICIE3); // Enable Interrupt on ICIE3

}

## Manchester Kodierung

Aus zeitlichen Gründen wurde keine Manchester Kodierung programmiert.

Funktion der Manchester Kodierung:

Wenn Daten Manchester kodiert gesendet werden, werden aus einem Datenbit 2 Datenbits, aus einer 0 wird 10 und aus einer 1 wird 01. Weiters wird der Taktgehalt in der Übertragung erhöht, wodurch eine leichtere Synchronisierung von Sender und Empfänger möglich ist.

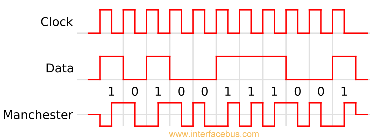


Abbildung : Manchester Kodierung

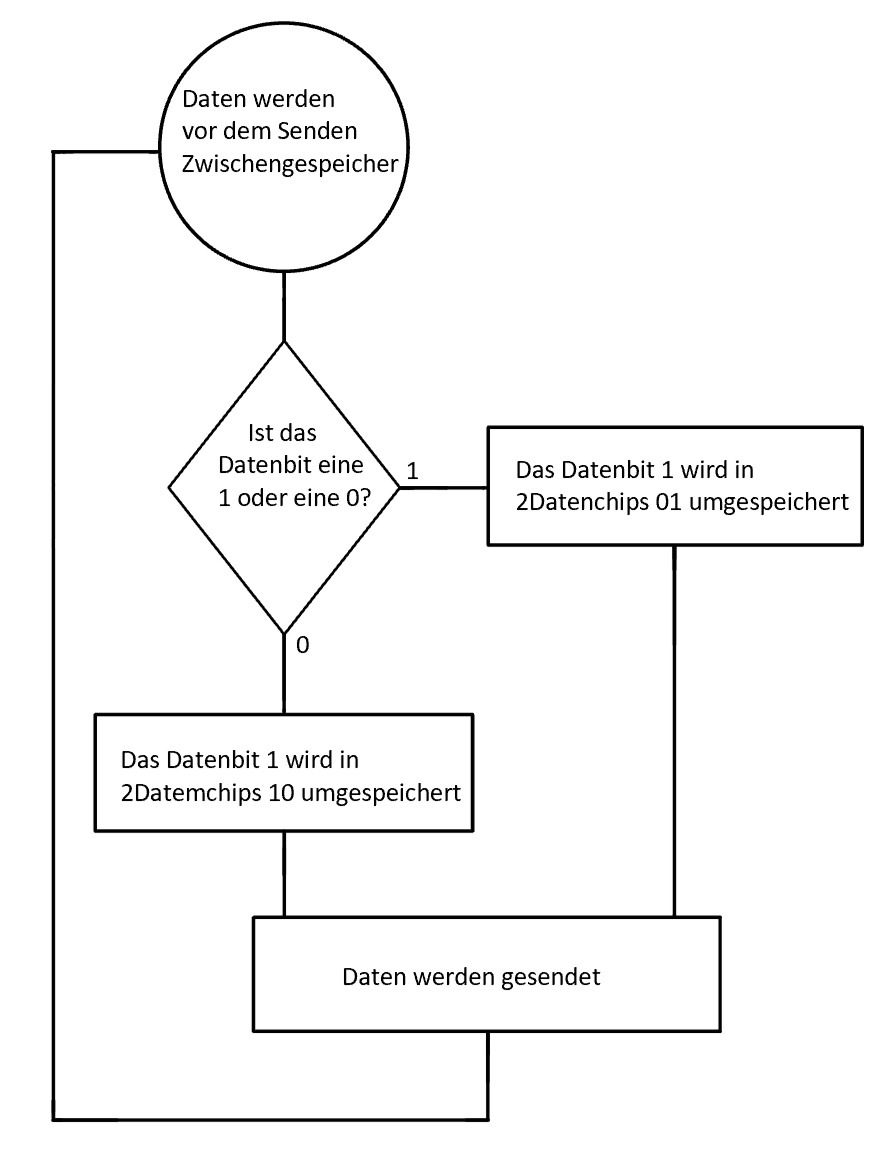


Abbildung : Flussdiagramm der Manchester Kodierung

## Hamming Code

Aus zeitlichen Gründen konnte die Hamming Codierung nicht mehr implementiert werden.

Der Hamming Code ist ein relativer Einfacher Code mit dem Hamming Abstand 3. Dies bedeutet das zwei Fehler erkannt werden können und ein Fehler korrigiert werden kann.

## Einlesen und Speichern der Daten

Aus zeitlichem Mangel konnte das Einlesen und Speichern der Daten nicht rechtzeitig fertig programmiert werden, daher wurde ein Flussdiagramm erstellt, um einen Überblick über den Ablauf der Einlese und Speicher Routine zu geben.

false

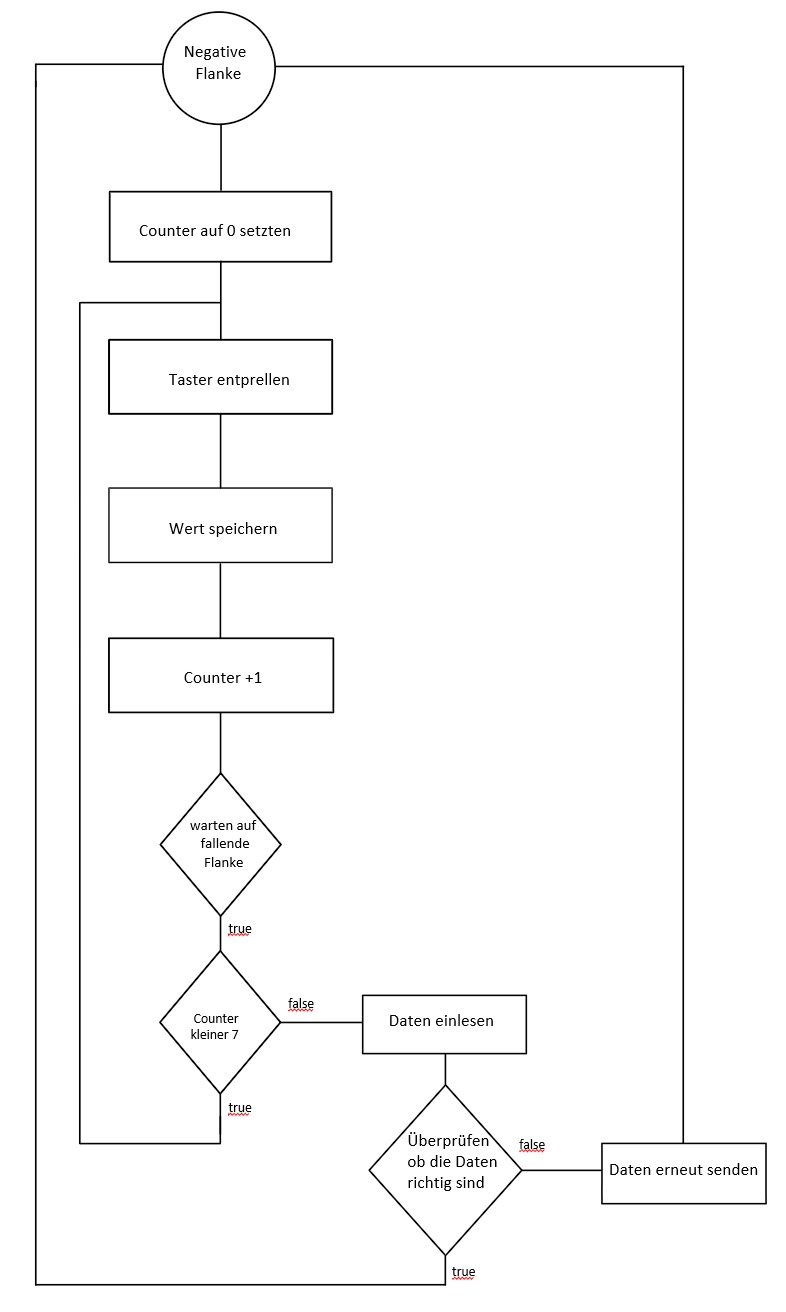


Abbildung : Flussdiagramm der Sende- und Einlese-Routine

Code : Interrupt Routine für Einlesen und Speichern

//Interrupt Routine

ISR(INT1\_vect)

{

InputCounter++;

If(InputCounter == 7)

{

INPUT\_MSB\_FIRST(PORTB);

InputCounter = 0;

}

}

Code : Initialisierung der Einlese und Speicher Routine

// Input-Counter PD1(INT1)

#define DATA\_LENGTH 7

int InputCounter = 0;

// Initialize Interrupt1 for Input-Counter

DDRD &= ~(1<<DDD1);

EIMSK = EIMSK | (1<<INT1); //locally enable Interrupt for INT1

EICRA = EICRA | (1<<ISC10) | (1<<ISC11); //set INT1 to generate an Interrupt at the

falling edge

Code : Speichern der Daten

#define INPUT\_MSB\_FIRST(input) ({ \

unsigned char ret[DATA\_LENGTH]; \

char tmp = input; \

do { \

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

{ \

ret[DATA\_LENGTH - 1 - i] = (((1 << i) & tmp) == (1 << i) ? 1: 0);\

} \

} while(0); \

ret; \

})

## Senderoutine

### Senderoutine Starten

Um die Senderoutine zu starten wird das Makro RF\_ID\_SEND (Code 1) verwendet. Diese führt die in der Abbildung 14: Flussdiagram des Sende Makros 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 13: Flussdiagram des Sende Makros

Code 4: 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 : 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

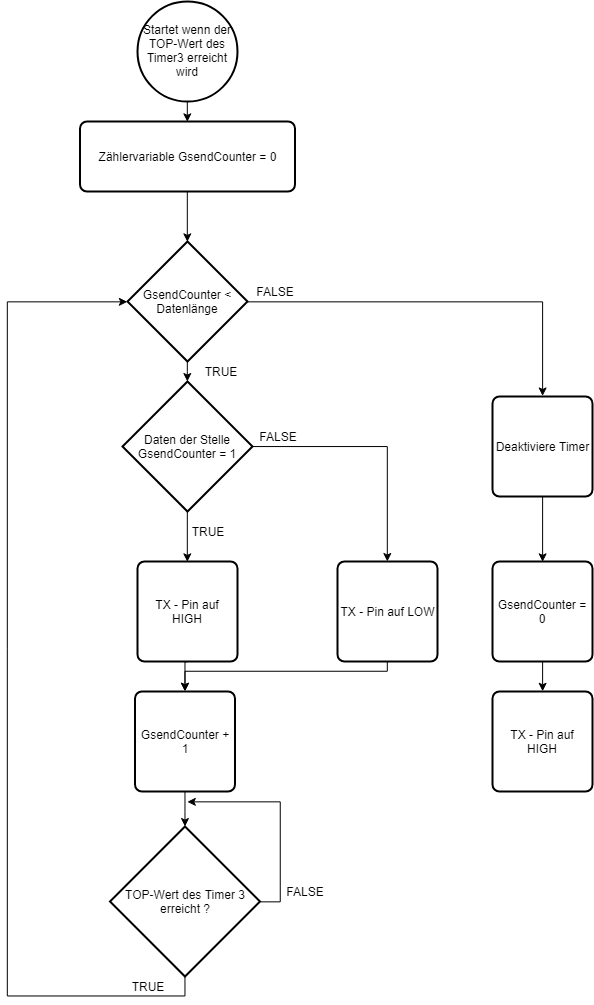
In Abbildung 16: Flussdiagram Interrupt Routine Timer 3 Capture Interrupt ist dargestellt wie das senden in der Interrupt Routine bewerkstelligt wird.

Abbildung 15: Flussdiagram Interrupt Routine Timer 3 Capture Interrupt

Code : 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

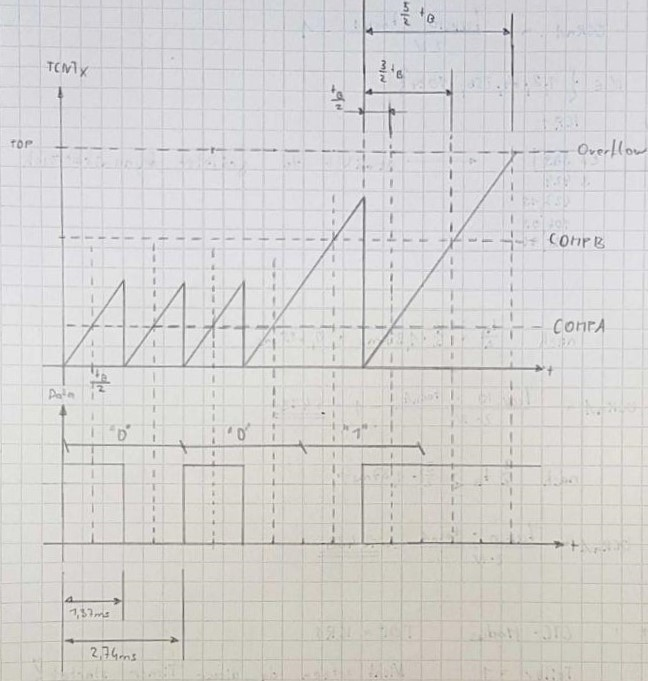


Abbildung : Timing Diagramm der Empfangsroutine

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 : 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 : 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 : Mögliche Werte für den Wert des ICRn Registers

### Flussdiagram der Empfangsroutine

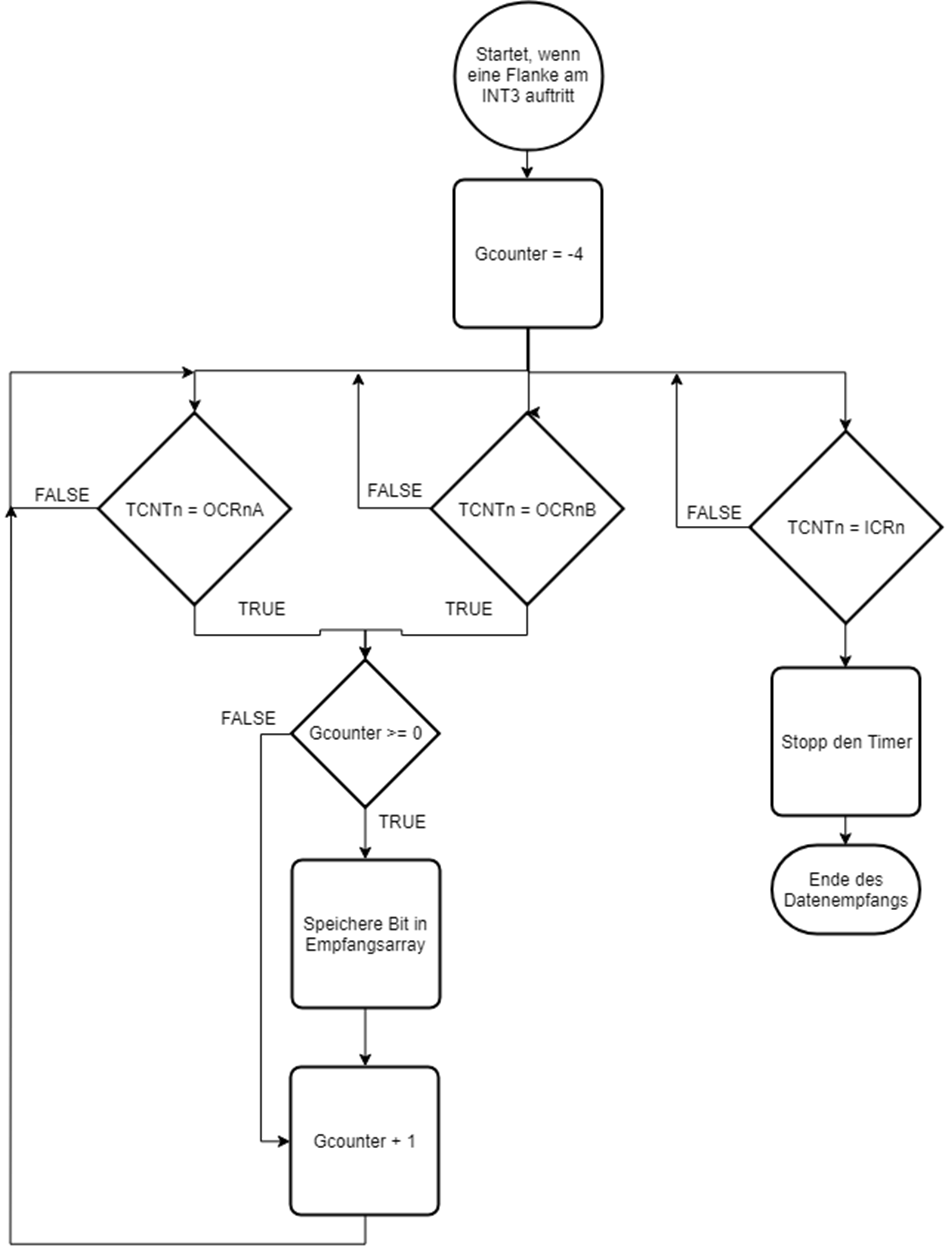


Abbildung : Flussdiagram der Empfangsroutine

### Code der Empfangsroutine

Code : 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

}