# Morsedatenübertragung mit Infrarot

Technologieschema:



An den vorderen Ecken sind Infrarotsendedioden angebracht (siehe Anhang). Eine 1 an PA\_7 schaltet das Infrarotlicht ein, eine 0 schaltet das Infrarotlicht aus. Der Infrarotlichtempfänger an PC\_8 empfängt Infrarotlicht, das mit einer Frequenz von 38 kHz ein- und ausgeschaltet wird. Dadurch werden Fremdlichteinflüsse eliminiert. 

Es bedeuten:

0: Licht empfangen

1: kein Licht empfangen

Hinweis: Für den Betrieb der Infrarot-Sendedioden muss die Brücke JP19 geschlossen sein.

Hinweise:

* Vorlage: mbed-os-example-blinky
* Verwendet Library: LCD\_i2c\_GSOE
* siehe auch Anhang 3

Aufgabe 1: Zum Senden muss die Sendediode mit 38 kHz gepulst werden:



D. h. Solange gesendet werden soll muss PA\_7 immer zwischen High (1) und Low (0) umgeschaltet werden.

Hinweis: Um die Frequenz möglichst exakt einstellen zu können wird mit:

TIM6->PSC=0; //Prescaler 32MHz

der Timertakt von TIM6 auf 32MHz eingestellt.

Füllen Sie zunächst die fehlenden Zeiten im Diagramm rechts aus.

Berechnen Sie den erforderlichen Wert für das Autoreloadregister ARR.

TIM6->ARR = \_\_\_\_\_\_\_\_;

Aufgabe 2: Schreiben Sie in C: Die Unterprogramme

a) *init(): Z*ur Initialisierung von Timer TIM6 so dass alle \_\_\_\_\_ µs ein Interrupt ausgelöst wird.

b)  *tim6isr():* Die das Rechtecksignal an PA\_7 ausgibt.

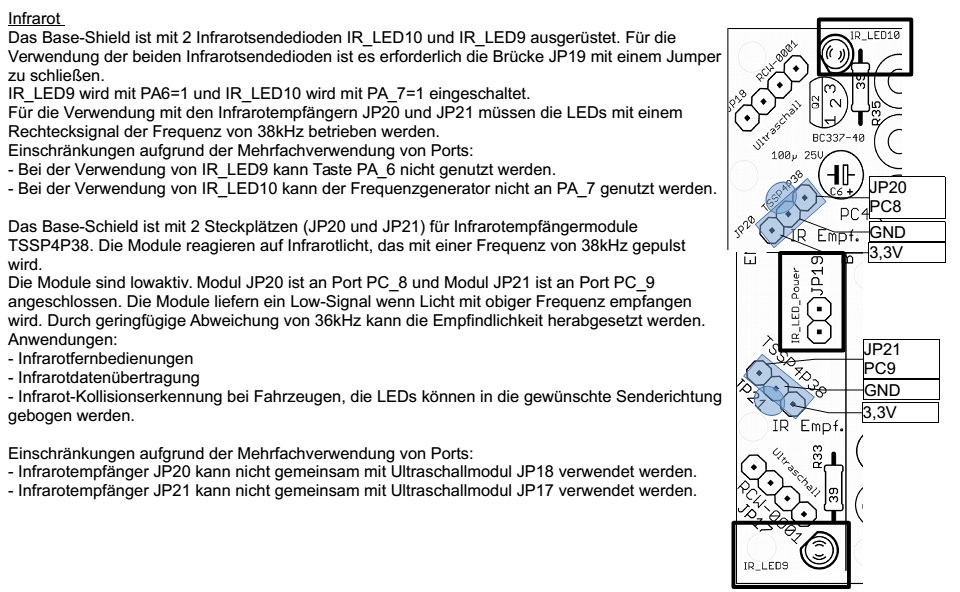
c) Hauptprogramm:

* Initialisierung:
  + Taste mit PullDown
  + init()
* Endlosschleife:
  + Geben Sie zu Testzwecken PC\_8 auf PC\_0 aus.
  + Wenn die Taste gedrückt ist soll TIM6 fregegeben werden, sonst nicht
  + Entprellen mit thread\_sleep\_for(20);

Aufgabe 3: Dekodierung der Morsesignale

Ergänzen Sie den Code im Anhang 3 entsprechend dem Zustandsdiagramm Anhang 4.

Anhang 1:



Anhang 2: Internationaler Morse-Code:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Buchst.** | **Morse** |  | **Buchst.** | **Morse** |
| A | **.-** |  | N | **-.** |
| B | **-...** |  | O | **---** |
| C | **-.-.** |  | P | **.--.** |
| D | **-..** |  | Q | **--.-** |
| E | **.** |  | R | **.-.** |
| F | **..-.** |  | S | **...** |
| G | **--.** |  | T | **-** |
| H | **....** |  | U | **..-** |
| I | **..** |  | V | **...-** |
| J | **.---** |  | W | **.--** |
| K | **-.-** |  | X | **-..-** |
| L | **.-..** |  | Y | **-.--** |
| M | **--** |  | Z | **--..** |

Anhang 3: Programmrumpf

#include "mbed.h"

#include "platform/mbed\_thread.h"

#include "LCD.h"

//Zustände

#define wartenAufTaste 0

#define tasteGedrueckt 1

#define kurz 2

#define lang 3

#define wartenAufNaechsteTaste 4

#define zeichenFertig 5

#define initialisieren 6

#define zeit TIM7->CNT

DigitalOut ird1(PA\_7);

DigitalIn taste(PA\_1);

DigitalIn start(PA\_10);

PortOut z(PortC,0xFF);

InterruptIn signal(PC\_8);

lcd mylcd;

int zustand=initialisieren;

int code=0, laenge=0;

void tim6isr(void)

{

**//Hier Ihr Code Aufgabe 2**

TIM6->SR=0;

HAL\_NVIC\_ClearPendingIRQ(TIM6\_IRQn);

}

void starteZeit()

{

zeit=0;

}

void gedrueckt() //ISR signal PC\_8 fallend Flanke

{

**// hier Ihr Code Aufgabe 3**

}

void speichern(int wert)

{

laenge++;

code=(code<<1)+wert;

}

void losgelassen() //ISR signal PC\_8 steigende Flanke

{

**// hier Ihr Code Aufgabe 3**

}

char dekodiere(char c, char dl)

{

switch (dl)

{

case 1: if (c==0) return 'E';

if (c==1) return 'T';

break;

case 2: switch (c)

{

case 0: return 'I';

case 1: return 'A';

case 2: return 'N';

case 3: return 'M';

};

break;

case 3: switch (c)

{

case 0: return 'S';

case 1: return 'U';

case 2: return 'R';

case 3: return 'W';

case 4: return 'D';

case 5: return 'K';

case 6: return 'G';

case 7: return 'O';

};

break;

case 4: switch(c)

{

case 0: return 'H';

case 1: return 'V';

case 2: return 'F';

case 3: return 'u';

case 4: return 'L';

case 5: return 'a';

case 6: return 'P';

case 7: return 'J';

case 8: return 'B';

case 9: return 'X';

case 10: return 'C';

case 11: return 'Y';

case 12: return 'Z';

case 13: return 'Q';

case 14: return 'o';

case 15: return ' ';

};

break;

default: return ' ';

}

return ' ';

}

void ausgeben() //lcd mylcd) //setzt code und laenge zurück

{

mylcd.printf("%c",dekodiere(code,laenge));

laenge=0;

code=0;

}

void abgelaufen(void) //ISR TIM7

{

**//Hier Ihr Code Aufgabe 3**

TIM7->SR=0;

HAL\_NVIC\_ClearPendingIRQ(TIM7\_IRQn);

}

/\* Timer init function \*/

void init(void)

{

//Eingänge

taste.mode(PullDown);

start.mode(PullDown);

signal.fall(&gedrueckt);

signal.rise(&losgelassen);

**//Hier Ihr Code Aufgabe 2**

//TIM7 initialisieren

RCC->APB1ENR|=0b100000; //Clock Enable

TIM7->PSC=31999; //Prescaler 1ms

TIM7->ARR=1000; //Autoreload 421\*1/32e6 = 13,156µs

TIM7->CNT=1000;

TIM7->DIER=1; //UIE = 1 (Update Interrupt Enable)

TIM7->SR=0; //UIF =0 (Update Interrupt Flag)

TIM7->CR1=1;

/\* TIM7\_IRQn interrupt configuration \*/

NVIC\_SetVector(TIM7\_IRQn, (uint32\_t)&abgelaufen);

HAL\_NVIC\_EnableIRQ(TIM7\_IRQn);

//Display

mylcd.clear();

mylcd.cursorpos(0);

}

int main()

{

init();

**//Hier Ihr Code Aufgabe 3**

while (true) {

if (taste==1) TIM6->CR1=1; else TIM6->CR1=0; //CEN=1 (Counter Enable)

z=zustand;

thread\_sleep\_for(20); //entprellen

}

}

Anhang 4: Zustandsdiagramm

