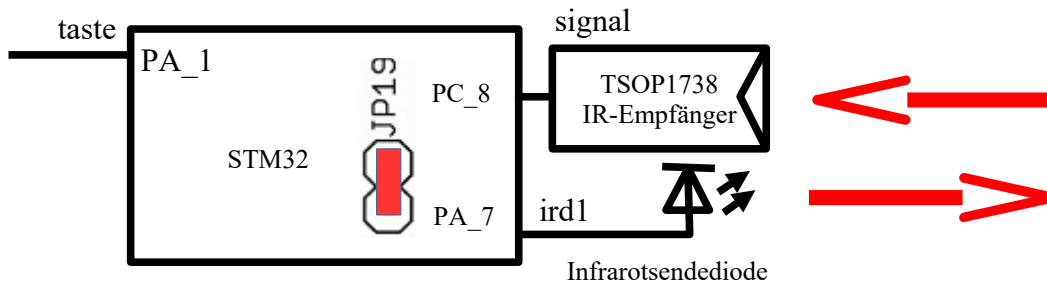


Morsedatenübertragung mit Infrarot

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-Sendediode muss die Brücke JP19 geschlossen sein.

Hinweise:

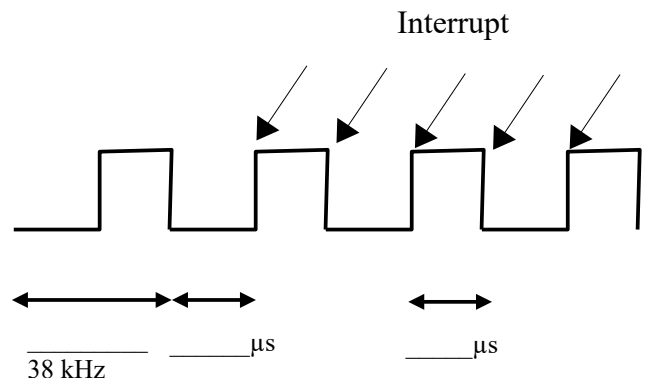
- Vorlage: mbed-os-example-blinky
- Verwendet Library: LCD_i2c_GSOE
- siehe auch Anhang 3

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

- `init()`: Zur Initialisierung von Timer TIM6 so dass alle _____ μs ein Interrupt ausgelöst wird.
- `tim6isr()`: Die das Rechtecksignal an PA_7 ausgibt.
- Hauptprogramm:

- Initialisierung:
 - Taste mit PullDown
 - `init()`
- Endlosschleife:
 - Geben Sie zu Testzwecken PC_8 auf PC_0 aus.
 - Wenn die Taste gedrückt ist soll TIM6 freigegeben 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:

Infrarot

Das Base-Shield ist mit 2 Infrarotsendioden IR_LED10 und IR_LED9 ausgerüstet. Für die Verwendung der beiden Infrarotsendioden ist es erforderlich die Brücke JP19 mit einem Jumper zu schließen.

IR_LED9 wird mit PA6=1 und IR_LED10 wird mit PA_7=1 eingeschaltet.

Für die Verwendung mit den Infrarotempfängern JP20 und JP21 müssen die LEDs mit einem Rechtecksignal der Frequenz von 38kHz betrieben werden.

Einschränkungen aufgrund der Mehrfachverwendung von Ports:

- Bei der Verwendung von IR_LED9 kann Taste PA_6 nicht genutzt werden.
- Bei der Verwendung von IR_LED10 kann der Frequenzgenerator nicht an PA_7 genutzt werden.

Das Base-Shield ist mit 2 Steckplätzen (JP20 und JP21) für Infrarotempfängermodule TSSP4P38. Die Module reagieren auf Infrarotlicht, das mit einer Frequenz von 38kHz gepulst wird.

Die Module sind lowaktiv. Modul JP20 ist an Port PC_8 und Modul JP21 ist an Port PC_9 angeschlossen. Die Module liefern ein Low-Signal wenn Licht mit obiger Frequenz empfangen wird. Durch geringfügige Abweichung von 36kHz kann die Empfindlichkeit herabgesetzt werden.

Anwendungen:

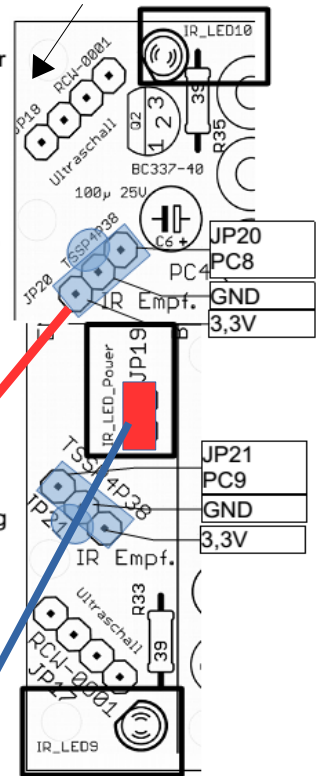
- Infrarotfernbedienungen
- Infrarotdatenübertragung
- Infrarot-Kollisionserkennung bei Fahrzeugen, die LEDs können in die gewünschte Senderichtung gebogen werden.

Einschränkungen aufgrund der Mehrfachverwendung von Ports:

- Infrarotempfänger JP20 kann nicht gemeinsam mit Ultraschallmodul JP18 verwendet werden.
- Infrarotempfänger JP21 kann nicht gemeinsam mit Ultraschallmodul JP17 verwendet werden.

Infrarotempfänger aufstecken

Brücke geschlossen



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 tasteGedueckt 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 gedueckt()                //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

