HÖHERE TECHNISCHE BUNDESLEHRANSTALT

HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse / Jahrgang:	Gruppe:	Übungsleiter:
5BHEL	10	Prof. Reisinger
Übungsnummer:	**Übungstitel:**	
-	-	
Datum der Übung:	**Teilnehmer:**	
-	Moritz Baldauf, Robert Radu	
Datum der Abgabe:	**Schriftführer:**	**Unterschrift:**
11.04.2021	Baldauf, Radu	

Allgemeiner Teil

Übungsleiter:	Prof. Reisinger
Klasse:	5BHEL
Schriftführer:	Baldauf, Radu

Inhaltsverzeichnis

1 Aufgabenstellung

Generell ist es das Ziel in dieser Übung Demoprogramme für Peripheral Library des Mikrocontrollers STM32F10X zu entwickeln, die auch reale Anwendungen simulieren sollen. Dieses soll auf dem HTL eigenen Mikrocontrollersystem (Cortex M3) funktionieren. Zugriff auf Peripherie (GPIO, ADC, Joystick, …) darf außerdem auch nur über die Standard Peripheral Library erfolgen und das über Keil µVision (Entwicklungsumgebung). Zustandsänderungen müssen über den UART protokolliert werden.

1.1 Individuelle Aufgabenstellung

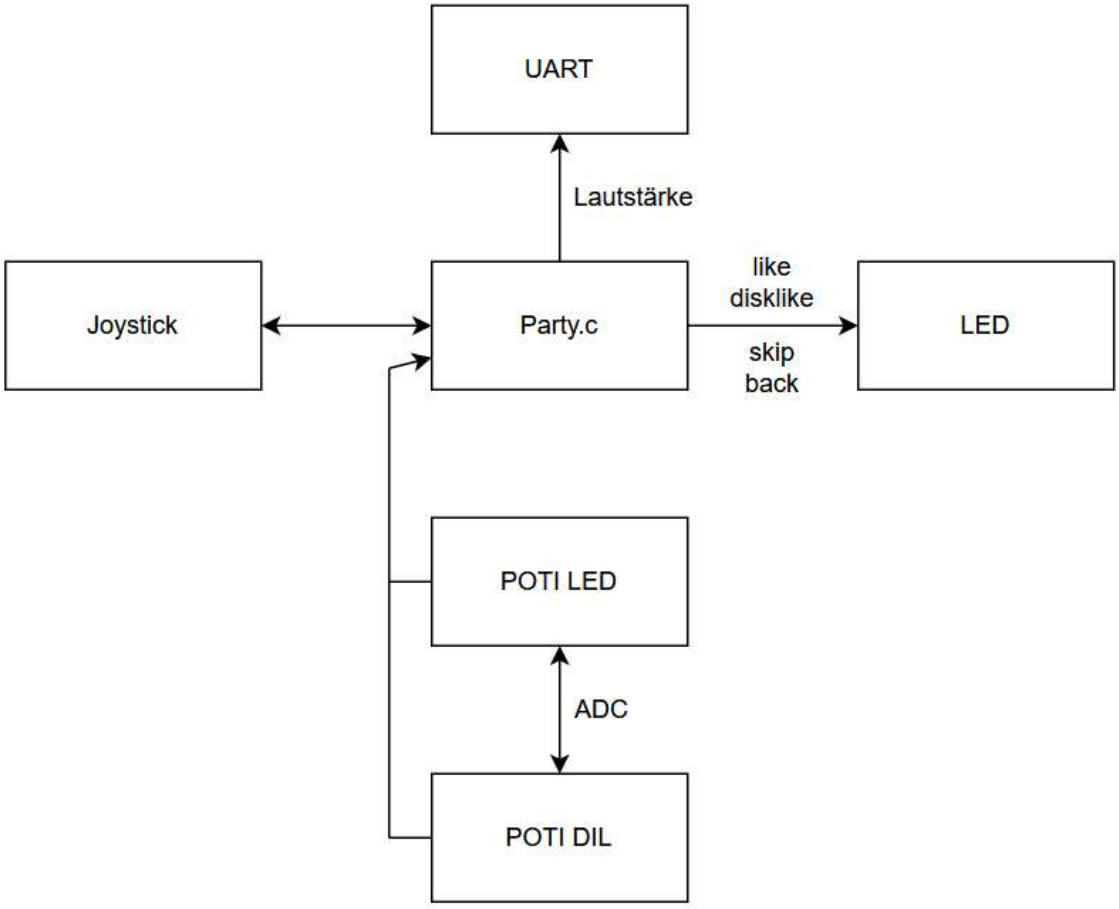
Folgende Peripherie (von CortexM3 Platine) muss verwendet werden:

- Joystick
- POTI (DIL Adapter)
- POTI LED/Schalterplatine
- ADC (ScanMode)
- LCD

1.2 Reale Anwendung

Als Anwendung für die „echte Welt“ hat man hier einen Musik-Player verwirklicht. Mit den POTI-Bauelementen soll es möglich sein die Lautstärke zu verändern und diese muss über den UART sichtbar gemacht werden. Der Joystick soll außerdem die Songwahl steuern mit „skip“ und „back“, aber auch den Song selbst „liken“ oder „disliken“.

2 Blockschaltbild von Demoprogramm



2.1 Kurze Erläuterung wie Aufgabe gelöst wurde

Den Joystick kann man nach oben, unten, links oder rechts drücken und je nachdem wohin er gedrückt wird, gibt das Programm auf der LCD-Anzeige etwas anderes an, bei oben wird ein like ausgegeben, bei unten wird ein dislike ausgegeben, bei links ein back und bei rechts ein skip. Die Potentiometer sind beide für die Lautstärke zuständig, dafür muss man entweder nach links oder nach rechts drehen und die Lautstärke wird entweder kleiner oder größer.

3 Source-Code

```
/*
Name: Party.c
Author: Moritz Baldauf, Robert Radu

Description:
The two POTI's should be run by the ADC in Scanmode,
The joystick should give an message to the LCD if used,
all changes on the POTI's are shown on the LCD in a table
```

```
Version: 1.0
*/
#include "Party.h"
#include "armv10_std.h"
#include "stm32f10x_conf.h"

/*Pins for Joystick
Left:PC6
Down:PC7
Up:PC8
Right:PC9
*/
//Initialise Pins for Joystick
GPIO_InitTypeDef Joyleft =
{
    .GPIO_Pin = GPIO_Pin_6,
    .GPIO_Speed = GPIO_Speed_50MHz,
    .GPIO_Mode = GPIO_Mode_IPU
};
//Initialise Pins for Joystick
GPIO_InitTypeDef JoyDown=
{
    .GPIO_Pin = GPIO_Pin_7,
    .GPIO_Speed = GPIO_Speed_50MHz,
    .GPIO_Mode = GPIO_Mode_IPU
};
//Initialise Pins for Joystick
GPIO_InitTypeDef JoyUp =
{
    .GPIO_Pin = GPIO_Pin_8,
    .GPIO_Speed = GPIO_Speed_50MHz,
    .GPIO_Mode = GPIO_Mode_IPU
};
//Initialise Pins for Joystick
GPIO_InitTypeDef JoyRight =
{
    .GPIO_Pin = GPIO_Pin_9,
    .GPIO_Speed = GPIO_Speed_50MHz,
    .GPIO_Mode = GPIO_Mode_IPU
};
//Internal peripheral Trigger
EXTI_InitTypeDef Joyleft_EXTI =
{
    .EXTI_Line = EXTI_Line6,
    .EXTI_Mode = EXTI_Mode_Interrupt,
    .EXTI_Trigger = EXTI_Trigger_Falling,
    .EXTI_LineCmd = ENABLE
};
//Internal peripheral Trigger
EXTI_InitTypeDef Joydown_EXTI =
{
    .EXTI_Line = EXTI_Line7,
    .EXTI_Mode = EXTI_Mode_Interrupt,
    .EXTI_Trigger = EXTI_Trigger_Falling,
    .EXTI_LineCmd = ENABLE
};
```

```
//Internal peripheral Trigger
EXTI_InitTypeDef Joyup_EXTI =
{
    .EXTI_Line = EXTI_Line8,
    .EXTI_Mode = EXTI_Mode_Interrupt,
    .EXTI_Trigger = EXTI_Trigger_Falling,
    .EXTI_LineCmd = ENABLE
};
//Internal peripheral Trigger
EXTI_InitTypeDef Joyright_EXTI =
{
    .EXTI_Line = EXTI_Line9,
    .EXTI_Mode = EXTI_Mode_Interrupt,
    .EXTI_Trigger = EXTI_Trigger_Falling,
    .EXTI_LineCmd = ENABLE
};
//Vector interrupt control for Pins 5-9
NVIC_InitTypeDef Joy_NVIC =
{
    .NVIC_IRQChannel = EXTI9_5_IRQn,
    .NVIC_IRQChannelPreemptionPriority = 3,
    .NVIC_IRQChannelSubPriority = 0,
    .NVIC_IRQChannelCmd = ENABLE
};

//variable for value from ADC1 channel 9
__IO uint16_t Ch_9 = 0;
//variable for value from ADC1 channel 14
__IO uint16_t Ch_14 = 0;
//string buffer for UART transfer
char buffer[100];

//Protoype for ADC programm
void ch14(void);

void EXTI9_5_IRQHandler()
{
    //Joystick up
    if(EXTI_GetFlagStatus(Joyup_EXTI.EXTI_Line) == SET)
    {
        //Print Like on UART and LCD
        lcd_init();
        lcd_set_cursor(0,0);
        sprintf(&buffer[0],"Like\r\n");
        lcd_put_string(&buffer[0]);
        USART_SendString(USART1,buffer);
        EXTI_ClearFlag(Joyup_EXTI.EXTI_Line);
    };
    //Joystick left
    if(EXTI_GetFlagStatus(Joydown_EXTI.EXTI_Line) == SET)
    {
        //Print Dislike on UART and LCD
        lcd_init();
        lcd_set_cursor(0,0);
        sprintf(&buffer[0],"Dislike\r\n");
        lcd_put_string(&buffer[0]);
```

```
        USART_SendString(USART1,buffer);
        EXTI_ClearFlag(Joydown_EXTI.EXTI_Line);
    };
    //Joystick right
    if(EXTI_GetFlagStatus(Joyright_EXTI.EXTI_Line) == SET)
    {
        //Print Skip on UART and LCD
        lcd_init();
        lcd_set_cursor(0,0);
        sprintf(&buffer[0],"Skip\r\n");
        lcd_put_string(&buffer[0]);
        USART_SendString(USART1,buffer);
        EXTI_ClearFlag(Joyright_EXTI.EXTI_Line);
    };
    //Joystick left
    if(EXTI_GetFlagStatus(Joyleft_EXTI.EXTI_Line) == SET)
    {
        //Print Back on UART and LCD
        lcd_init();
        lcd_set_cursor(0,0);
        sprintf(&buffer[0],"Back\r\n");
        lcd_put_string(&buffer[0]);
        USART_SendString(USART1,buffer);
        EXTI_ClearFlag(Joyleft_EXTI.EXTI_Line);
    };

    NVIC_ClearPendingIRQ(EXTI9_5_IRQn);
}



/*
  Synchronise inputs before
    processing input-data using
    a semaphore rendezvous
*/

void ch14 ()
{
    int perc = 0;
    int old = 0;
    for(;;) {
        //get data from ADC1 channel14
        Ch_14 = ADC1ConvertedValues[1];
        //get data from ADC1 channel9
        Ch_9 = ADC1ConvertedValues[0];

        //calculate combined percentage of both variable resistors
        perc = ((Ch_14 + Ch_9)*100)/8190;

        //Only print if Changes
        //Potis are not totally exact so there is a tolerance for
changes
        if(!((perc >= old-1)&&(perc <= old+1)))
        {
            old = perc;
```

```
            //prepare string buffer for UART
            sprintf(&buffer[0],"Lautstaerke:%d\r\n",perc);
            //send string via UART
            USART_SendString(USART1,buffer);
            lcd_init();
            lcd_clear();
            //shows procentage on LCD screen
            lcd_bargraphXY(1,0,perc);
        }
    }
}

int main (void) {
    //Enalbe Peripheral Clock
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

    //Initilaice GPIO Port and Pins
    GPIO_Init(GPIOC, &Joyleft);
    GPIO_Init(GPIOC, &JoyRight);
    GPIO_Init(GPIOC, &JoyDown);
    GPIO_Init(GPIOC, &JoyUp);

    //Initilaice External Interrupt for IE-Event
    AFIO->EXTICR[6/4] |= (AFIO_EXTICR2_EXTI6_PC);
    AFIO->EXTICR[7/4] |= (AFIO_EXTICR2_EXTI7_PC);
    AFIO->EXTICR[8/4] |= (AFIO_EXTICR3_EXTI8_PC);
    AFIO->EXTICR[9/4] |= (AFIO_EXTICR3_EXTI9_PC);
    EXTI_Init(&Joyup_EXTI);
    EXTI_Init(&Joydown_EXTI);
    EXTI_Init(&Joyright_EXTI);
    EXTI_Init(&Joyleft_EXTI);

    NVIC_Init(&Joy_NVIC);

    //load DMA config
    DMA_Config();
    //initialize ADC1
    init_uart();//initialize UART
    ADC1_Init();
    //Clear Console
    USART_SendData(USART1,12);
    USART_SendString(USART1,
    "\r\n***************\r\n    Musikanlage
\r\n***************\r\n");
    SystemCoreClockUpdate();
    ch14();//start programm for potis
    while(1);
}
```

4 Funktionsnachweis mit Screenshots & Erläuterung

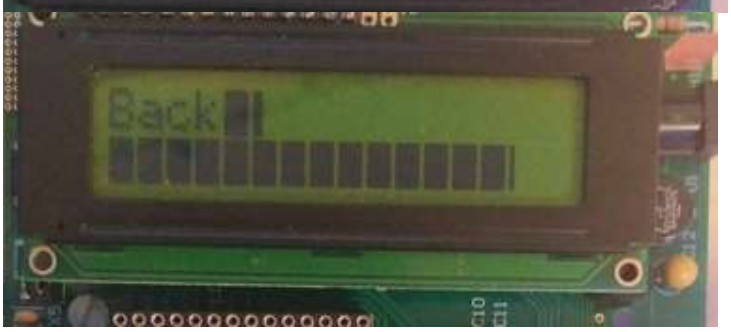
```
**********************
    Musikanlage
**********************
Lautstaerke:49
Lautstaerke:47
Lautstaerke:45
Lautstaerke:43
Lautstaerke:41
Lautstaerke:39
Lautstaerke:37
Lautstaerke:35
Lautstaerke:33
```

Hier kann man klar erkennen wie eines der beiden POTI's verwendet wird, da sich die Lautstärke verändert. Wird direkt ausgegeben bei Veränderung, aber auch wenn es keine gibt

Der Joystick wurde nach oben gedrückt → Der Song wird geliked.

Der Joystick wurde nach unten gedrückt → Der Song wird gedisliked.

Der Joystick wurde nach rechts gedrückt → Song wird übersprungen.



Der Joystick wurde nach links gedrückt → Ein Song davor wird abgespielt.