HÖHERE TECHNISCHE BUNDESLEHRANSTALT

HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse / Jahrgang:	Gruppe:	Übungsleiter:
5BHEL	Team 8	Prof. Reisinger
Übungsnummer:	**Übungstitel:**	
Übung 3	Timer1, LED-Europlatine, Taster DIL-Adapter UART#1 (Polling)	
Datum der Übung:	**Teilnehmer:**	
10.03.2021	Hasenzagl, Meichenitsch	
Datum der Abgabe:	**Schriftführer:**	**Unterschrift:**
12.04.2021		

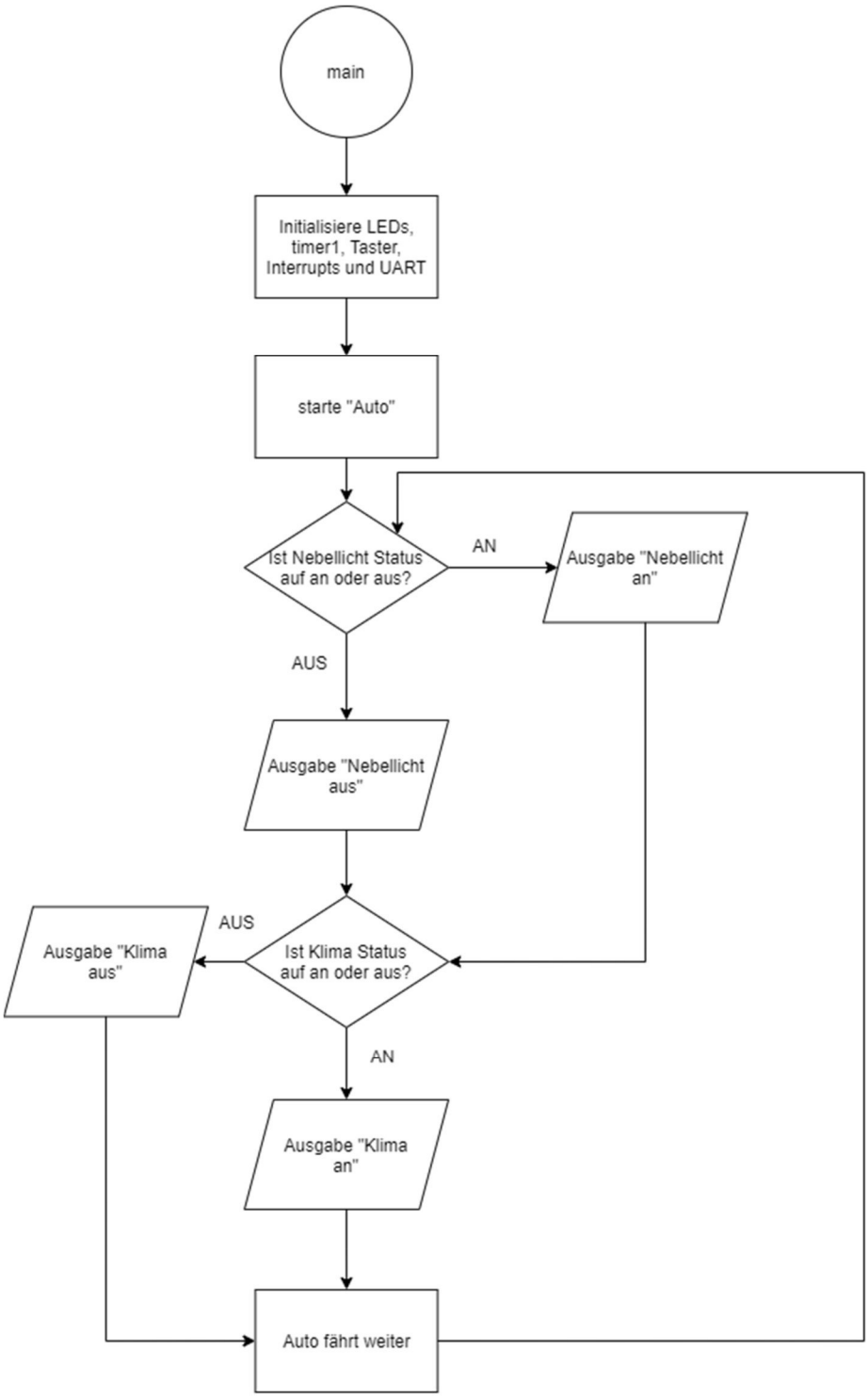
		Beurteilung
Deckbl., Inhaltsverz.		
Aufgabenstellung		
Dokumentation		
Messschaltungen		
Messtabellen		
Berechnungen		
Programmlistings		
Auswertung		
Diagramme		
Berechnungen		
Simulationen		
Schlußfolgerungen		
Kommentare		
Inventarliste		
Messprotokoll		
Form		
Summe		

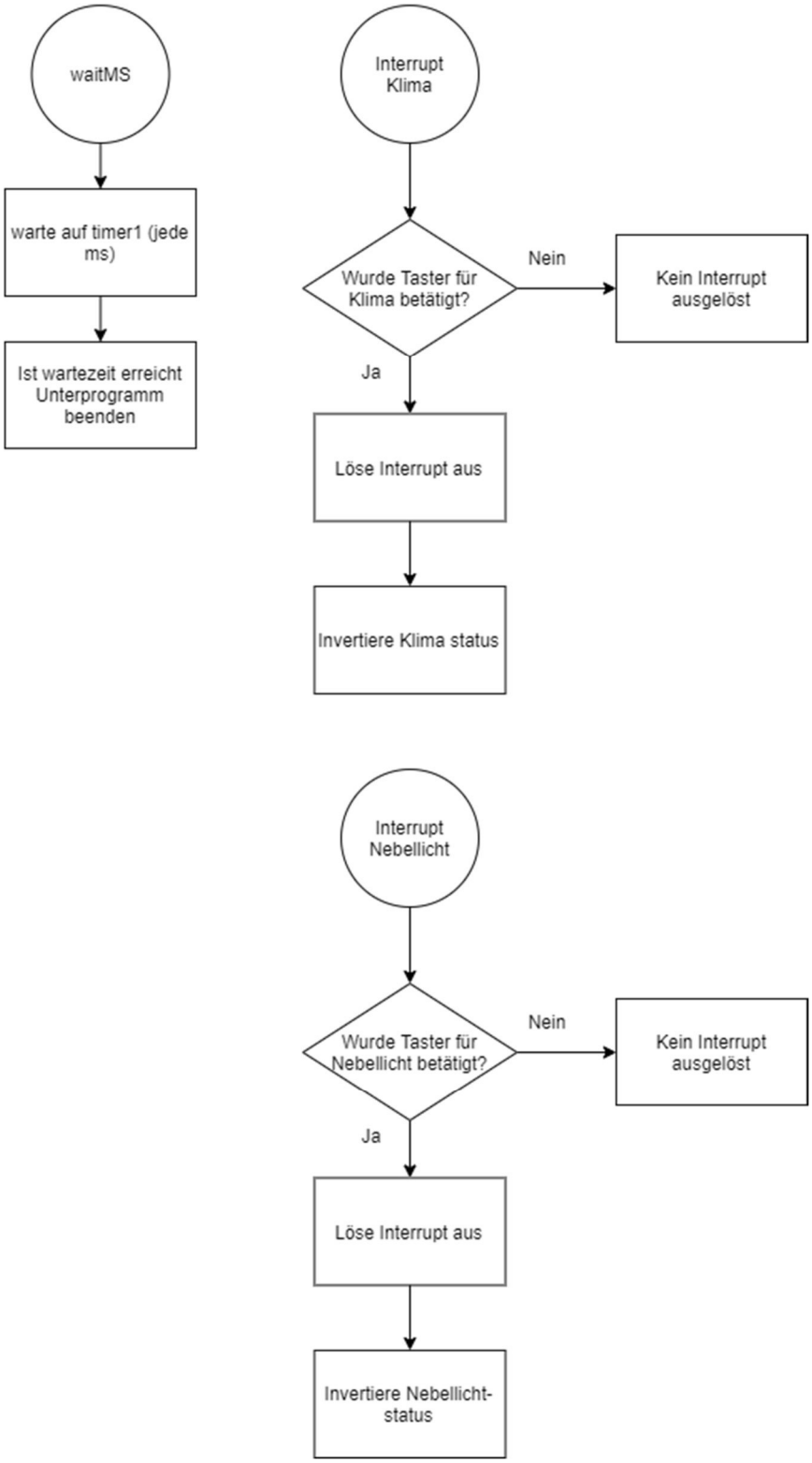
Inhaltsverzeichnis

1 Aufgabenstellung

Es sollte ein Beispielprogramm für timer1, den Tastern am DIL-Adapter, dem LED-Array auf der Euro-Platine und UART#1 (Polling) programmiert werden. Realisiert wurde es als Realbeispiel "Auto mit Klima und Nebellicht".

2 Blockschaltbild




waitMS
warte auf timer1 (jede ms)
Ist wartezeit erreicht Unterprogramm beenden
Interrupt Klima
Wurde Taster für Klima betätigt?
Nein
Kein Interrupt ausgelöst
Ja
Löse Interrupt aus
Invertiere Klima status
Interrupt Nebellicht
Wurde Taster für Nebellicht betätigt?
Nein
Kein Interrupt ausgelöst
Ja
Löse Interrupt aus
Invertiere Nebellicht-status

```c
/******************************************************************************/
/*   (C) Copyright HTL - HOLLABRUNN  2009-2009 All rights reserved. AUSTRIA   */
/*                                                                            */
/* File Name:   main.c                                                        */
/* Autor:     Philipp Hasenzagl / Nicolas Meichenitsch                        */
/* Version:   V1.00                                                           */
/* Date:    11/04/2021                                                        */
/* Description: Beispielprogramm für timer1, Taster am DIL-Adapter,           */
/*              LED-Array auf der Euro-Platine und UART#1 (polling),          */
/*              realisiert als Realbeispiel "Auto mit Klima und Nebellicht"   */
/******************************************************************************/

//Programm uses 8MHz HSE

#include "stm32f10x.h"
#include "stm32f10x_conf.h"
#include "string.h"

//prototypes
void usart_send_buffer(const char *str, int strlen);
void usart_send_string(const char *str);
void init_TIM();
void waitMS(int ms);

//global variable
int flag=0;
int AC=0;
int Nebel=0;

//timer1 and wait function
int timer_value = 0;

TIM_TimeBaseInitTypeDef tim1 =
{
  .TIM_Prescaler = 0x0008, //divides 8MHz by 8 = 1MHz
  .TIM_CounterMode = TIM_CounterMode_Up, //defines that the timer should count up
  .TIM_Period = 999, //counts every clk-cyle 1 up = 0-999 at 1MHz is 1ms
  .TIM_ClockDivision = TIM_CKD_DIV1, //could be divided again (not used)
  .TIM_RepetitionCounter = 0
};

NVIC_InitTypeDef tim1_nvic =
{
  .NVIC_IRQChannel = TIM1_UP_IRQn,
  .NVIC_IRQChannelPreemptionPriority = 2, //sets priority of the interrupt
  .NVIC_IRQChannelSubPriority = 0,
  .NVIC_IRQChannelCmd = ENABLE
};

void TIM1_UP_IRQHandler()
{
  if (TIM_GetFlagStatus(TIM1, TIM_IT_Update)) //when timer's ready do this
  {
    timer_value++; //count time_value up (one count takes one ms)
    TIM_ClearFlag(TIM1, TIM_IT_Update); //clear flag
  }
}

void init_TIM()
{
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1, ENABLE); //EnableClock for timer1

  TIM_TimeBaseInit(TIM1, &tim1); //use tim1 configurations from above

  NVIC_Init(&tim1_nvic);

  TIM_ITConfig(TIM1, TIM_IT_Update, ENABLE);
  TIM_Cmd(TIM1, ENABLE);
}

void waitMS(int ms)
{
  timer_value = 0; //reset timer_value
  while (timer_value < ms); //blocking process until "time is up"
}


```

```
78
79  //LED and Button define
80  GPIO_InitTypeDef LED_Euro_1 =
81  {
82    .GPIO_Pin = GPIO_Pin_6, //Pin definition
83    .GPIO_Speed = GPIO_Speed_50MHz, //Frequency Speed
84    .GPIO_Mode = GPIO_Mode_Out_PP // defines as Push/Pull
85  };
86
87  GPIO_InitTypeDef LED_Euro_2 =
88  {
89    .GPIO_Pin = GPIO_Pin_7, //Pin definition
90    .GPIO_Speed = GPIO_Speed_50MHz, //Frequency Speed
91    .GPIO_Mode = GPIO_Mode_Out_PP // defines as Push/Pull
92  };
93
94  GPIO_InitTypeDef taster_dill_1 =
95  {
96    .GPIO_Pin = GPIO_Pin_5,
97    .GPIO_Speed = GPIO_Speed_50MHz,
98    .GPIO_Mode = GPIO_Mode_IPU // defines as Input
99  };
100
101 GPIO_InitTypeDef taster_dill_2 =
102 {
103   .GPIO_Pin = GPIO_Pin_13,
104   .GPIO_Speed = GPIO_Speed_50MHz,
105   .GPIO_Mode = GPIO_Mode_IPU // defines as Input
106 };
107
108
109 //Interrupt define
110 EXTI_InitTypeDef exti0 =
111 {
112   .EXTI_Line = EXTI_Line5, //defines which EXTI Line is used (is the same as the PIN-Number from the
    Input)
113   .EXTI_Mode = EXTI_Mode_Interrupt, //sets mode as Interrupt
114   .EXTI_Trigger = EXTI_Trigger_Falling, //trigegrs on Falling-Edge
115   .EXTI_LineCmd = ENABLE
116 };
117
118 NVIC_InitTypeDef nvic_exti0 =
119 {
120   .NVIC_IRQChannel = EXTI9_5_IRQn, //EXTI 5 is in a combined interrupt request handler therefore
    EXTI9_5_IRQn
121   .NVIC_IRQChannelPreemptionPriority = 3, //sets priority of the interrupt
122   .NVIC_IRQChannelSubPriority = 0,
123   .NVIC_IRQChannelCmd = ENABLE
124 };
125
126 EXTI_InitTypeDef exti1 =
127 {
128   .EXTI_Line = EXTI_Line13, //defines which EXTI Line is used (is the same as the PIN-Number from
    the Input)
129   .EXTI_Mode = EXTI_Mode_Interrupt, //sets mode as Interrupt
130   .EXTI_Trigger = EXTI_Trigger_Falling, //trigegrs on Falling-Edge
131   .EXTI_LineCmd = ENABLE
132 };
133
134 NVIC_InitTypeDef nvic_exti1 =
135 {
136   .NVIC_IRQChannel = EXTI15_10_IRQn, //EXTI 5 is in a combined interrupt request handler therefore
    EXTI9_5_IRQn
137   .NVIC_IRQChannelPreemptionPriority = 3, //sets priority of the interrupt
138   .NVIC_IRQChannelSubPriority = 0,
139   .NVIC_IRQChannelCmd = ENABLE
140 };
141
142 //Interrupts
143 void EXTI9_5_IRQHandler()
144 {
145   if (EXTI_GetFlagStatus(exti0.EXTI_Line)) //when Interrupt triggered then (when button pushed then)
146   {
147     if (Nebel == 0)
148     {
149       GPIO_SetBits(GPIOB, LED_Euro_1.GPIO_Pin);
150       flag |= 1; //sets flag for main programm
```

```
151         Nebel = 1;
152       }
153       else
154       {
155         GPIO_ResetBits(GPIOB, LED_Euro_1.GPIO_Pin);
156         Nebel = 0;
157         flag |= 2; //sets flag for main programm
158       }
159
160       EXTI_ClearFlag(exti0.EXTI_Line); //resets Interrupt flag
161     }
162   }
163
164   void EXTI15_10_IRQHandler()
165   {
166     if (EXTI_GetFlagStatus(exti1.EXTI_Line)) //when Interrupt triggered then (when button pushed then)
167     {
168       if (AC == 0)
169       {
170         GPIO_SetBits(GPIOB, LED_Euro_2.GPIO_Pin);
171         flag |= 4; //sets flag for main programm
172         AC = 1;
173       }
174       else
175       {
176         GPIO_ResetBits(GPIOB, LED_Euro_2.GPIO_Pin);
177         AC = 0;
178         flag |= 8; //sets flag for main programm
179       }
180
181
182       EXTI_ClearFlag(exti1.EXTI_Line); //resets Interrupt flag
183     }
184   }
185
186   //UART initialize
187   void init_UART()
188   {
189     USART_InitTypeDef usart_init;
190     USART_ClockInitTypeDef usart_clkinit;
191     GPIO_InitTypeDef gpio;
192
193     // Peripheral Clock Enable
194     RCC_APB2PeriphClockCmd(RCC_APB2ENR_IOPAEN, ENABLE);
195     RCC_APB2PeriphClockCmd(RCC_APB2ENR_AFIOEN, ENABLE);
196     RCC_APB2PeriphClockCmd(RCC_APB2ENR_USART1EN, ENABLE);
197
198   //GPIO Pins init
199     //Tx (PA9)
200     gpio.GPIO_Pin = GPIO_Pin_9;
201     gpio.GPIO_Speed = GPIO_Speed_50MHz;
202     gpio.GPIO_Mode = GPIO_Mode_AF_PP;
203     GPIO_Init(GPIOA, &gpio);
204
205     //Rx (PA10)
206     gpio.GPIO_Pin = GPIO_Pin_10;
207     gpio.GPIO_Mode = GPIO_Mode_IN_FLOATING;
208     GPIO_Init(GPIOA, &gpio);
209
210     //USART Clock init
211     usart_clkinit.USART_Clock = USART_Clock_Enable;
212     usart_clkinit.USART_CPHA = USART_CPHA_2Edge;
213     usart_clkinit.USART_CPOL = USART_CPOL_Low;
214     usart_clkinit.USART_LastBit = USART_LastBit_Disable;
215     USART_ClockInit(USART1, &usart_clkinit);
216
217     //USART init
218     USART_StructInit(&usart_init);
219     usart_init.USART_BaudRate = 115200;
220     USART_Init(USART1, &usart_init);
221
222     USART_Cmd(USART1, ENABLE);
223     usart_send_string("UART initialize successful!\r\n");
224
225   }
226
227   //makes use easier (you only have to input the string initially and not the length too)
```

```
228   void usart_send_string(const char *str)
229   {
230     usart_send_buffer(str, strlen(str));
231   }
232
233   //sends 1 character at a time (polling)
234   void usart_send_buffer(const char *str, int strlen)
235   {
236     char *data = (char *)str;
237     for(; strlen>0; strlen--, data++)
238     {
239       USART_SendData(USART1, *data);
240       while(!USART_GetFlagStatus(USART1, USART_FLAG_TXE)); //checks Tx USART-status
241     }
242     while(!USART_GetFlagStatus(USART1, USART_FLAG_TC));
243   }
244
245
246
247   int main()
248   {
249     //enabling used GPIOS
250     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
251     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
252
253     //initallize the Button and LED with their corrosponding GPIO
254     GPIO_Init(GPIOB, &LED_Euro_1);
255     GPIO_Init(GPIOC, &taster_dill_1);
256     GPIO_Init(GPIOB, &LED_Euro_2);
257     GPIO_Init(GPIOC, &taster_dill_2);
258
259     /*
260     Examples for Setting, Resetting and Reading Bits
261     GPIO_SetBits(GPIOB, LED_Euro_1.GPIO_Pin);
262     GPIO_ResetBits(GPIOB, LED_Euro_1.GPIO_Pin);
263     GPIO_WriteBit(GPIOB, LED_Euro_1.GPIO_Pin, Bit_SET);
264     GPIO_ReadInputDataBit(GPIO, taster_dill_1.GPIO_Pin);
265
266     //toggle LED
267     GPIO_WriteBit(GPIOB, LED_Euro_1.GPIO_Pin, ((~GPIO_ReadOutputDataBit(GPIOB,
      LED_Euro_1.GPIO_Pin))&1));
268     */
269
270     RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE); //Enable AFIO for Interrupt
271     AFIO->EXTICR[1] |= AFIO_EXTICR2_EXTI5_PC;
272     AFIO->EXTICR[13/4] |= AFIO_EXTICR4_EXTI13_PC;
273
274     //Initialize Interrupt
275     EXTI_Init(&exti0);
276     NVIC_Init(&nvic_exti0);
277     EXTI_Init(&exti1);
278     NVIC_Init(&nvic_exti1);
279
280     init_TIM();
281     init_UART();
282     usart_send_string("Auto gestartet\r\n");
283     while(1)
284     {
285       if(flag & 1)
286       {
287         usart_send_string("Nebellicht an\r\n");
288         flag &= ~1;
289       }
290       else if(flag & 2)
291       {
292         usart_send_string("Nebellicht aus\r\n");
293         flag &= ~2;
294       }
295       else if(flag & 4)
296       {
297         usart_send_string("Klima an\r\n");
298         flag &= ~4;
299       }
300       else if(flag & 8)
301       {
302         usart_send_string("Klima aus\r\n");
303         flag &= ~8;
```

```
304         }
305         usart_send_string("Auto faehrt\r\n");
306         waitMS(1000); //wait a second
307       }
308     }
309
```

4 Funktionsnachweis

Wie zu sehen ist, wird Anfangs der UART initialisiert und danach das Auto gestartet. Die Nachricht „Auto faehrt“ wird jede Sekunde ausgegeben, und sobald ein Taster auf dem DIL-Adapter gedrückt wird, ändert sich der Status des Nebellichts bzw. der Klima.
Der Funktionsnachweis für die Änderung der LEDs, sind im beigefügten Video zu sehen.