

HÖHERE TECHNISCHE BUNDESLEHRANSTALT

HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse / Jahrgang: 5BHEL	Projektteam: Team#3 – Marx Clemens, Pachtrog Jakob
Datum der Abgabe: 10.04.2021	Thema: CM3 Peripheral Library – Inkrementalgeber, Externer Interrupt, UART#1(DMA)

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

Inhaltsverzeichnis

1	Aufgabenstellung.....	3
1.1	Grundidee	3
1.2	Reale Anwendung.....	3
1.3	Blockschaltbild	4
1.4	Inkrementalgeber auf der Euro-Platine	4
1.5	Schaltplan	5
1.6	Signal des Inkrementalgeber	5
2	Funktionsweise.....	6
3	Funktionsnachweis.....	6
4	Source Code	9
5	Quellen	22

1 Aufgabenstellung

1.1 Grundidee

Die Grundidee der Übung besteht darin ein Demoprogramm für die Peripheral Library des Mikrocontrollers STM32F10X zu realisieren, welches eine reale Anwendung simulieren soll.

Das Demoprogramm soll auf dem HTL eigenen Mikrocontrollersystem auf Basis des Cortex M3 Mikrocontroller lauffähig sein. Der Zugriff auf die Peripherieeinheiten (GPIO, ADC, Timer, UART, ...) des Mikrocontrollers ausschließlich über die **Standard Peripheral Library** der Entwicklungsumgebung Keil µVision 5 erfolgen.

Zustandsänderungen im System sollen über UART protokolliert werden und können mit einem Terminalprogramm (PuTTY) angesehen werden. Über dieses Logging wird auch der Funktionsbeweis geführt.

1.2 Reale Anwendung

Die Aufgabe besteht darin, die Anwendung eines Lautstärkereglers im alltäglichen, realen Gebrauch zu simulieren.

Lautstärkereglern kommen bei Radios oder Soundanlagen vor und besitzen oftmals dabei eine Anzeige, mit der aktuellen Lautstärke, angegeben in einer numerischen Zahl.

Bei unserer Simulation geht es um die Lautstärkeregelung in einem Auto mit Anzeige der Lautstärke auf einer numerischen Skala von 0-20. Die entsprechende Anzeige ist in unserem Fall das Terminal und der Lautstärkereglern ist der Inkrementalgeber auf unserer Euro-Platine. Der Inkrementalgeber wird mit externen Interrupts ausgelesen.



Abbildung 1: Lautstärkereglern bei einem Autoradio [WP21]

1.3 Blockschaltbild

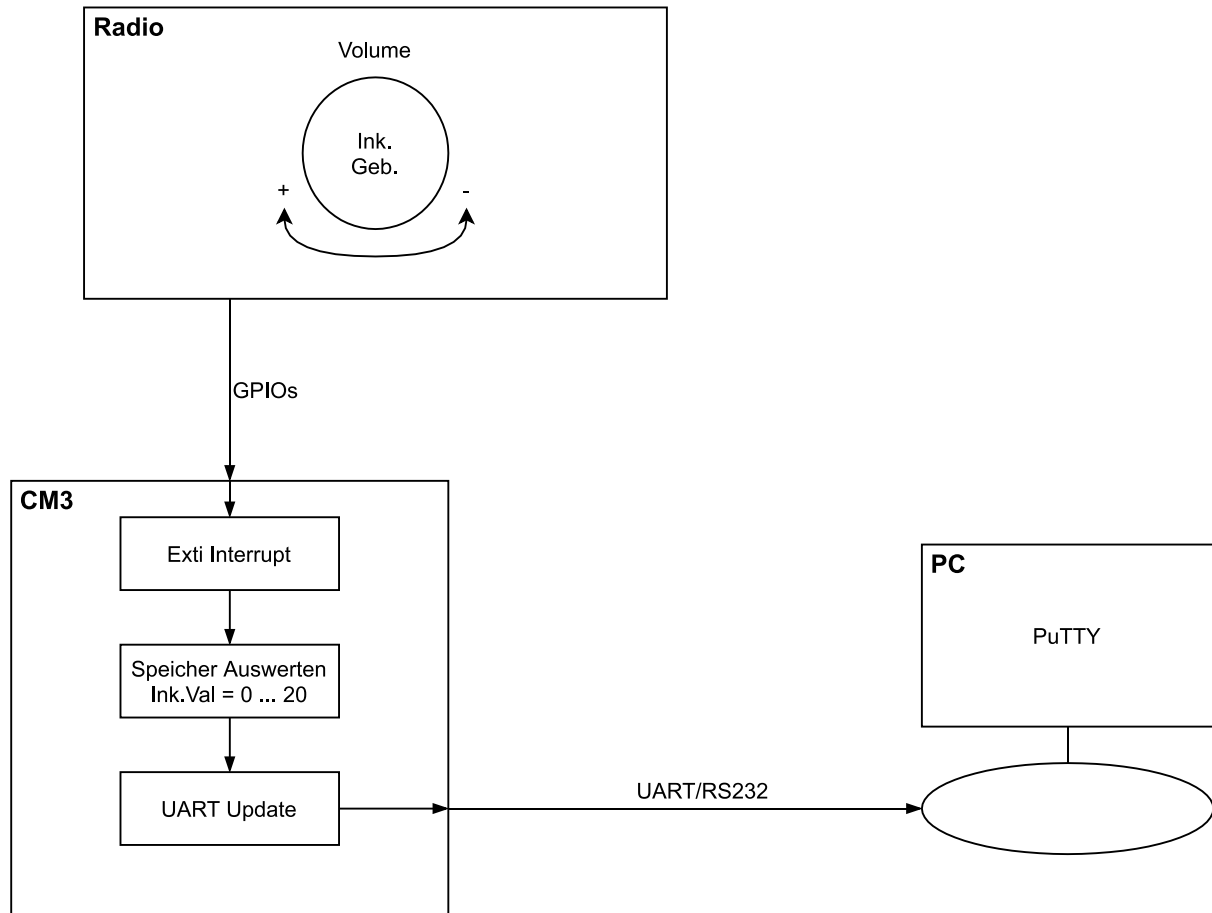


Abbildung 2: Blockschaltbild

1.4 Inkrementalgeber auf der Euro-Platine

Der Inkrementalgeber befindet sich auf der Euro-Platine zwischen den LEDs und dem Joystick. Dieser hat die Bezeichnung STEC12E08 und ist von der Firma ALPS.

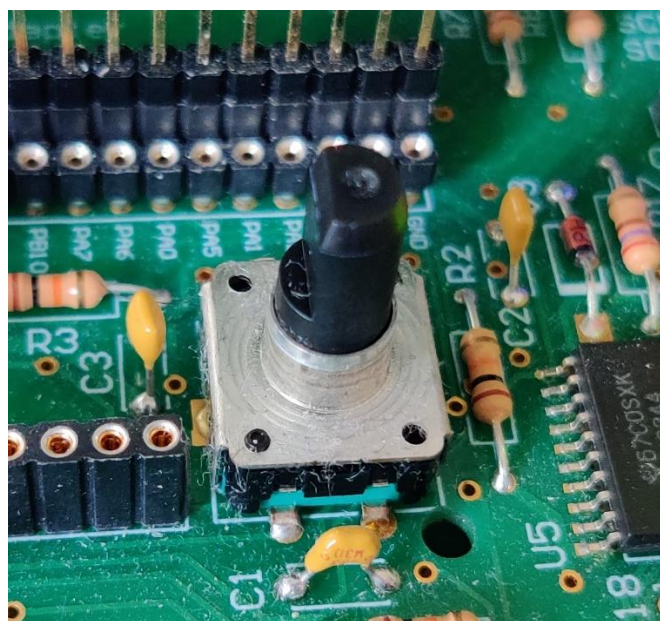


Abbildung 3: Inkrementalgeber auf der Euro-Platine

1.5 Schaltplan

Die Schaltung des Inkrementalgeber auf der Euro-Platine sieht wie folgt aus:

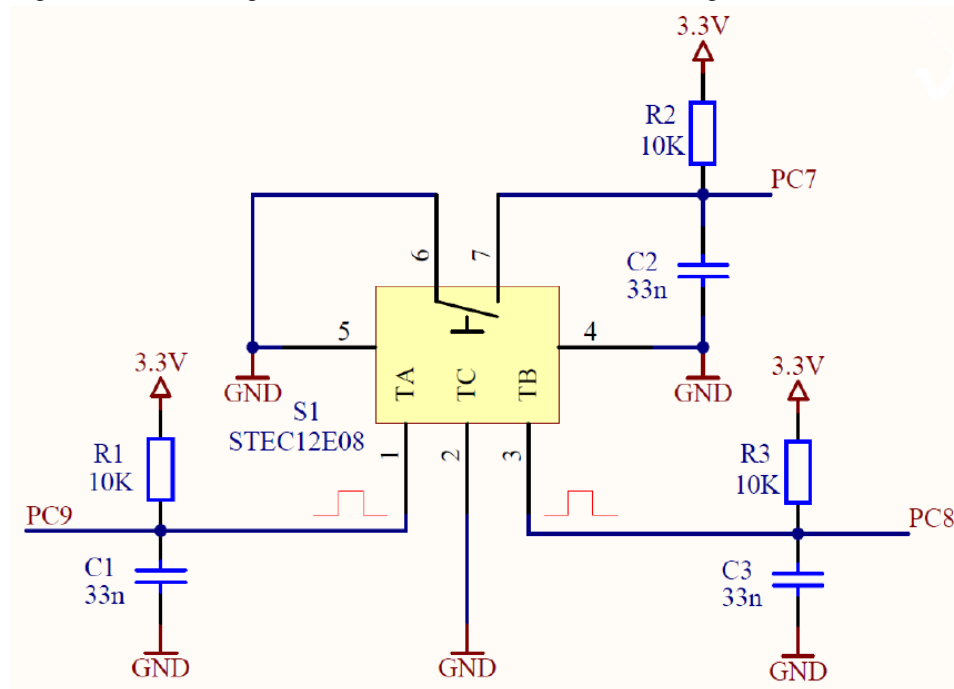


Abbildung 4: Schaltung zum Inkrementalgeber (nach [CM311])

1.6 Signal des Inkrementalgeber

Der Inkrementalgeber (STEC12E08) gibt, während er gedreht wird, ein Rechtecksignal auf seinen zwei Pins aus. Diese sind die I/O-Pins PC8 und PC9 des STM32F103RB ausgeführt. Weiters besitzt der Inkrementalgeber auch einen Taster, wird in unserer Anwendung aber nicht benötigt.

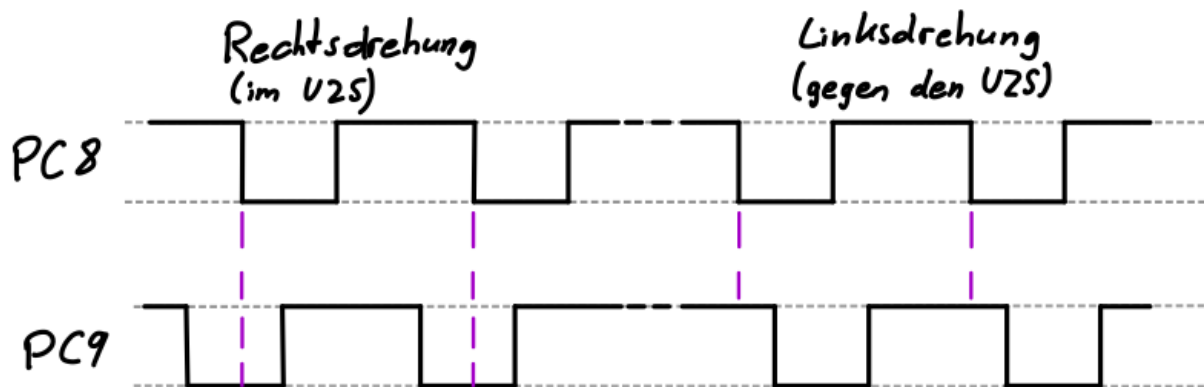


Abbildung 5: Signal Inkrementalgeber

2 Funktionsweise

Im Programm werden zu Beginn die entsprechenden GPIO-Pins für den Inkrementalgeber und den UART initialisiert. Über das EXTICR3 Register wird die Quelle für den externen Interrupt ausgewählt (PC8) und initialisiert. Der externe Interrupt löst demnach nun jedes Mal aus, wenn eine fallende Flanke beim PC8 detektiert wird. In der Interrupt Service Routine wird abgefragt, ob PC9 High oder Low ist. Demnach kann festgestellt werden, ob der Inkrementalgeber nach links oder nach rechts gedreht wird (siehe 1.6 Signal des Inkrementalgeber).

Die Ausgangslautstärke ist ,10' und wird dekrementiert, wenn nach links gedreht wird und inkrementiert, wenn nach rechts gedreht wird. Dies ist dann abhängig von dem jeweils vorherigen Wert. Der numerische Lautstärkewert kann mindestens 0 und maximal 20 erreichen.

Jedes Mal, wenn eine Änderung der Lautstärke vorgenommen wird, wird die Änderung über den UART im DMA-Betrieb ausgegeben und über ein Terminal geloggt.

3 Funktionsnachweis

Zum Loggen wird das Tool PuTTY verwendet. Der UART verwendet eine Baudrate von 115200 und ist über die COM4-Schnittstelle angeschlossen. Mit *Open* wird das Terminal geöffnet.

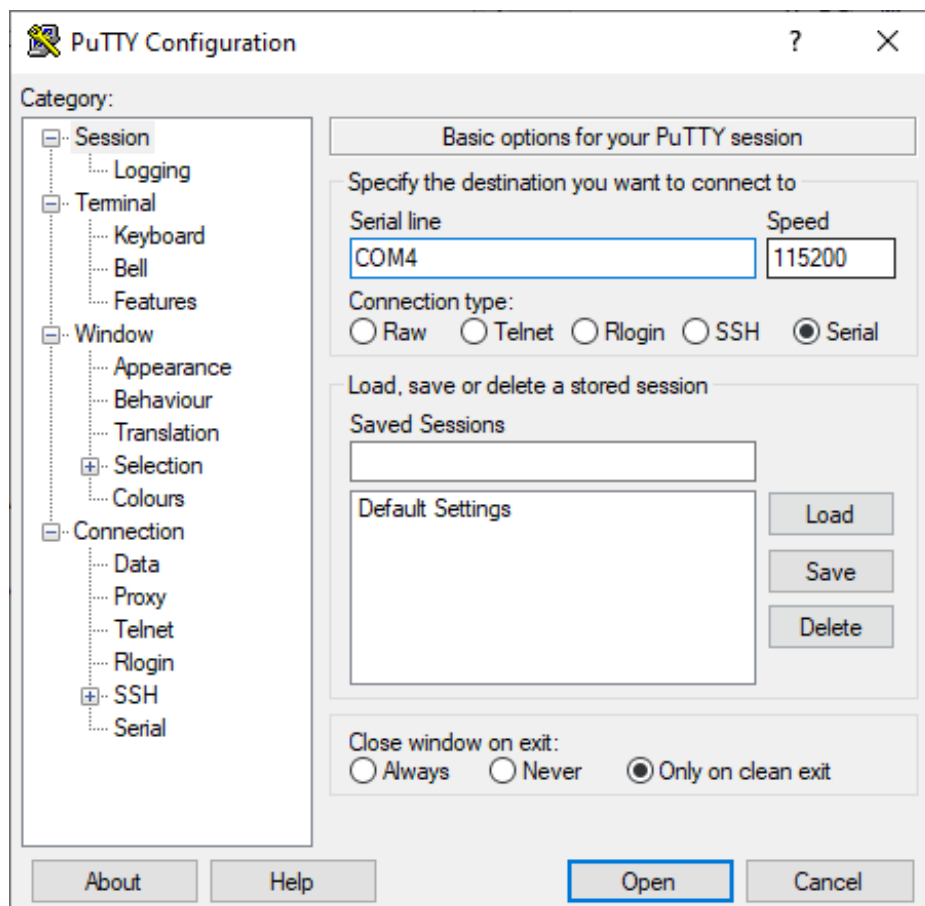


Abbildung 6: PuTTY Konfiguration zum Loggen

Fall 1 – Linksdrehung:

Im ersten Fall wird der Inkrementalgeber nach Links gedreht.

Das Video zum Funktionsnachweis des ersten Falles befindet sich im Abgabeordner unter \DIC_Ü3_Abgabe_Marx_Pachtrog\Funktionsnachweis\Linksdrehung.mp4.

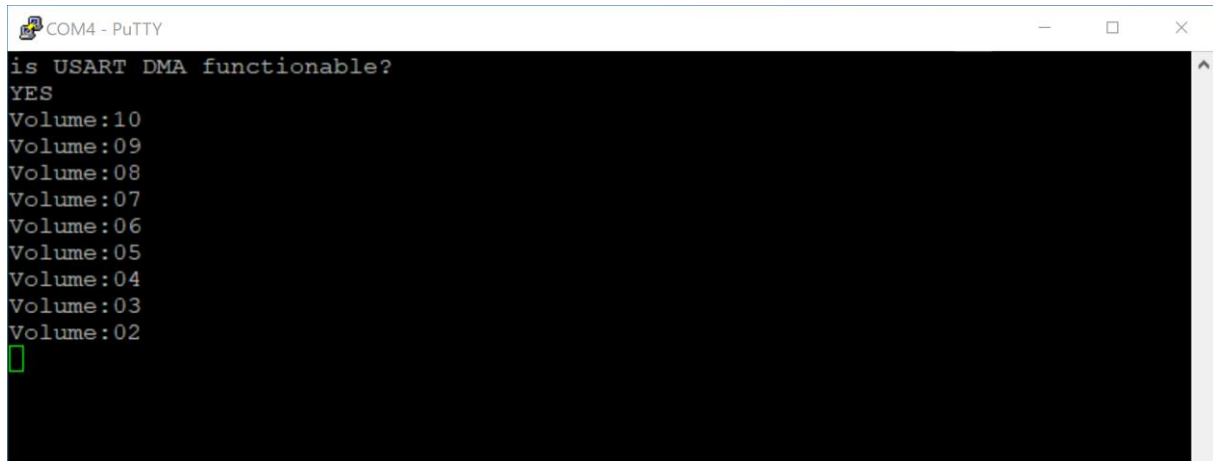
A screenshot of a PuTTY terminal window titled 'COM4 - PuTTY'. The terminal output shows the question 'is USART DMA functionable?' followed by 'YES'. Below this, a series of volume levels are printed: 'Volume:10', 'Volume:09', 'Volume:08', 'Volume:07', 'Volume:06', 'Volume:05', 'Volume:04', 'Volume:03', and 'Volume:02'. A green cursor is visible on the line following 'Volume:02'.

Abbildung 7: Terminal Logging Linksdrehung

Fall 2 – Rechtsdrehung:

Im zweiten Fall wird der Inkrementalgeber nach rechts gedreht.

Das Video zum Funktionsnachweis des zweiten Falles befindet sich im Abgabeordner unter \DIC_Ü3_Abgabe_Marx_Pachtrog\Funktionsnachweis\Rechtsdrehung.mp4.

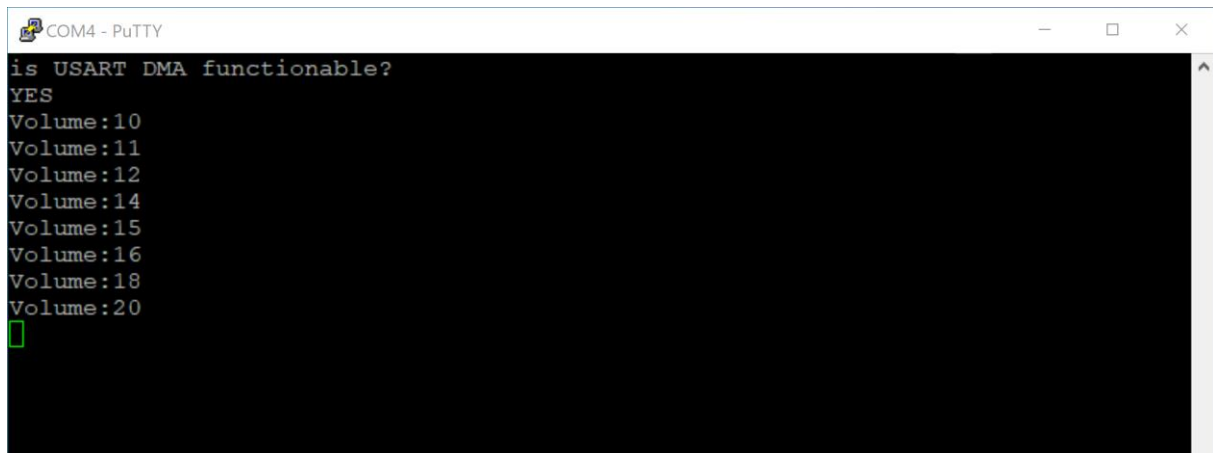
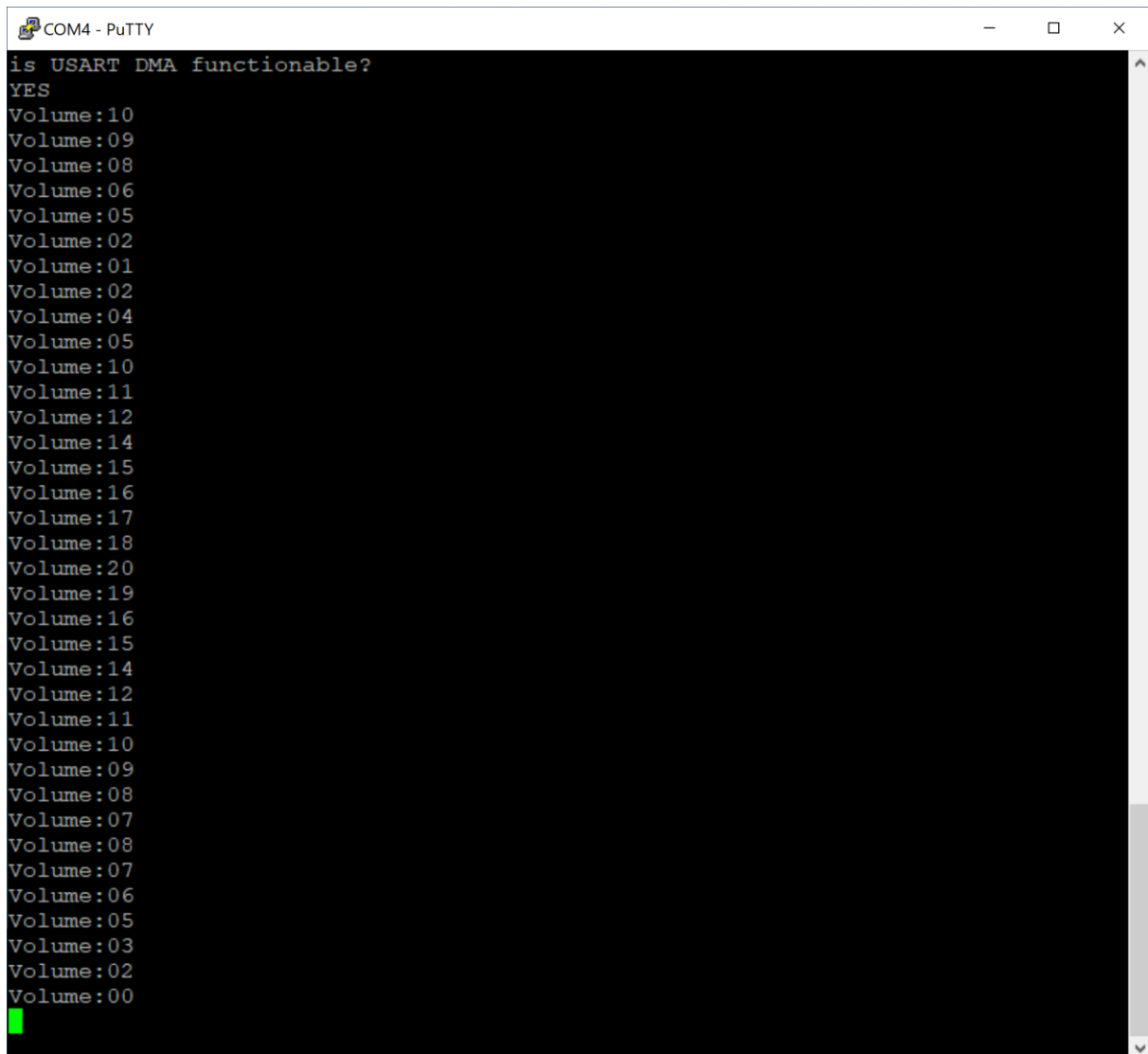
A screenshot of a PuTTY terminal window titled 'COM4 - PuTTY'. The terminal output shows the question 'is USART DMA functionable?' followed by 'YES'. Below this, a series of volume levels are printed: 'Volume:10', 'Volume:11', 'Volume:12', 'Volume:14', 'Volume:15', 'Volume:16', 'Volume:18', and 'Volume:20'. A green cursor is visible on the line following 'Volume:20'.

Abbildung 8: Terminal Logging Rechtsdrehung

Fall 3 – Links-, dann Rechts-, dann Linksdrehung:

Im dritten Fall wird der Inkrementalgeber nach links zum Minimum, dann nach rechts zum Maximum und dann wieder nach links zum Minimum gedreht. Damit wird die funktionierende Änderung in beide Richtungen gewährleistet. In realer Anwendung würde die Lautstärke nach unten, nach oben und dann wieder nach unten geregelt werden.

Das Video zum Funktionsnachweis des dritten Falles befindet sich im Abgabeordner unter \DIC_Ü3_Abgabe_Marx_Pachtrog\Funktionsnachweis\LinksRechtsLinksdrehung.mp4.



```
COM4 - PuTTY
is USART DMA functionable?
YES
Volume:10
Volume:09
Volume:08
Volume:06
Volume:05
Volume:02
Volume:01
Volume:02
Volume:04
Volume:05
Volume:10
Volume:11
Volume:12
Volume:14
Volume:15
Volume:16
Volume:17
Volume:18
Volume:20
Volume:19
Volume:16
Volume:15
Volume:14
Volume:12
Volume:11
Volume:10
Volume:09
Volume:08
Volume:07
Volume:08
Volume:07
Volume:06
Volume:05
Volume:03
Volume:02
Volume:00
```

Abbildung 9: Terminal Logging Links-, Rechts-, Linksdrehung

4 Source Code

```
001 /**
002  * \brief          DIC#3 Project - Standard Peripheral Library
003  * \file           main.c
004  * \date           10.04.2021
005  * \version        1.2
006  * \author         Clemens Marx
007  * \author         Jakob Pachtrog
008  * \copyright      HTL - Hollabrunn
009  *
010  * \details
011  * Inkrementalgeber:   GPIO C8      Event
012  *                      GPIO C9      Direction
013  *
014  * UART:               GPIO A9      Tx
015  *                      GPIO A10     Rx
016  *
017  */
018
019 /**
020  * \}
021  * \defgroup Includes
022  * \{
023  */
024 #include "stm32f10x.h"           /// Device header
025 #include "stm32f10x_conf.h"     /// StdPeriph Driver
026 #include "stdio.h"
027 #include "string.h"
```

```
028
029 /**
030  *  \}
031  *  \defgroup Defines
032  *  \{
033  */
034
035 #define MAX_VOLUME 20
036
037 /**
038  *  \}
039  *  \defgroup Variables Global variables
040  *  \{
041  */
042 char uartDR = 0;
043 int volume = 10;
044 int oldVolume = 0;
045
046 /**
047  *  \}
048  *  \defgroup Variables Incremental Encoder settings
049  *  \{
050  */
051
```

```
052 /**
053  * \brief          Incremental Encoder - GPIO Port: Event
054  */
055 GPIO_InitTypeDef iE_Event =
056 {
057     .GPIO_Pin = GPIO_Pin_8,
058     .GPIO_Speed = GPIO_Speed_50MHz,
059     .GPIO_Mode = GPIO_Mode_IPU
060 };
061
062 /**
063  * \brief          Incremental Encoder - GPIO Port: Dircetion
064  */
065 GPIO_InitTypeDef iE_Direction =
066 {
067     .GPIO_Pin = GPIO_Pin_9,
068     .GPIO_Speed = GPIO_Speed_50MHz,
069     .GPIO_Mode = GPIO_Mode_IPU
070 };
071
072 /**
073  * \brief          Incremental Encoder - External Interrupt
074  */
075 EXTI_InitTypeDef iE_EXTI_Event =
076 {
077     .EXTI_Line = EXTI_Line8,
078     .EXTI_Mode = EXTI_Mode_Interrupt,
079     .EXTI_Trigger = EXTI_Trigger_Falling,
080     .EXTI_LineCmd = ENABLE
081 };
```

```
082
083 /**
084  * \brief          Incremental Encoder - NVIC for External Interrupt
085  */
086 NVIC_InitTypeDef iE_NVIC =
087 {
088     .NVIC_IRQChannel = EXTI9_5_IRQn,
089     .NVIC_IRQChannelPreemptionPriority = 3,
090     .NVIC_IRQChannelSubPriority = 0,
091     .NVIC_IRQChannelCmd = ENABLE
092 };
093
094 /**
095  * \}
096  * \defgroup Variables vCOM settings
097  * \{
098  */
099
100 /**
101  * \brief          vCOM - GPIO Port: Tx
102  */
103 GPIO_InitTypeDef vCOM_Tx =
104 {
105     .GPIO_Pin = GPIO_Pin_9,
106     .GPIO_Speed = GPIO_Speed_50MHz,
107     .GPIO_Mode = GPIO_Mode_AF_PP
108 };
109
```

```
110 /**
111  * \brief          vCOM - GPIO Port: Rx
112  */
113 GPIO_InitTypeDef vCOM_Rx =
114 {
115     .GPIO_Pin = GPIO_Pin_10,
116     .GPIO_Speed = GPIO_Speed_50MHz,
117     .GPIO_Mode = GPIO_Mode_IN_FLOATING
118 };
119
120 /**
121  * \brief          vCOM - Init Struct
122  */
123 USART_InitTypeDef vCOM =
124 {
125     .USART_BaudRate = 115200,
126     .USART_WordLength = USART_WordLength_8b,
127     .USART_StopBits = USART_StopBits_1,
128     .USART_Parity = USART_Parity_No,
129     .USART_Mode = (USART_Mode_Tx | USART_Mode_Rx),
130     .USART_HardwareFlowControl = USART_HardwareFlowControl_None
131 };
132
```

```
133 /**
134  * \brief          vCOM - Clk Init Struct
135  */
136 USART_ClockInitTypeDef vCOM_clk =
137 {
138     .USART_Clock = USART_Clock_Disable,
139     .USART_CPOL = USART_CPOL_Low,
140     .USART_CPHA = USART_CPHA_2Edge,
141     .USART_LastBit = USART_LastBit_Disable
142 };
143
144 /**
145  * \brief          vCOM - NVIC for USART
146  */
147 NVIC_InitTypeDef vCOM_NVIC =
148 {
149     .NVIC_IRQChannel = USART1_IRQn,
150     .NVIC_IRQChannelPreemptionPriority = 3,
151     .NVIC_IRQChannelSubPriority = 0,
152     .NVIC_IRQChannelCmd = ENABLE
153 };
154
```

```
155 /**
156  * \brief          vCOM - NVIC for DMA
157  */
158 NVIC_InitTypeDef vCOM_DMA_NVIC =
159 {
160     .NVIC_IRQChannel = DMA1_Channel1_IRQn,
161     .NVIC_IRQChannelPreemptionPriority = 3,
162     .NVIC_IRQChannelSubPriority = 0,
163     .NVIC_IRQChannelCmd = ENABLE
164 };
165
166 /**
167  * \brief          vCOM - DMA for USART
168  */
169 DMA_InitTypeDef vCOM_DMA =
170 {
171     .DMA_PeripheralBaseAddr = USART1_BASE,
172     .DMA_MemoryBaseAddr = (uint32_t)&uartDR,
173     .DMA_DIR = DMA_DIR_PeripheralDST,
174     .DMA_BufferSize = 32,
175     .DMA_PeripheralInc = DMA_PeripheralInc_Disable,
176     .DMA_MemoryInc = DMA_MemoryInc_Enable,
177     .DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte,
178     .DMA_MemoryDataSize = DMA_MemoryDataSize_Byte,
179     .DMA_Mode = DMA_Mode_Circular,
180     .DMA_Priority = DMA_Priority_Medium,
181     .DMA_M2M = DMA_M2M_Disable
182 };
183
184 /**
```

```
185 * \}  
186 */  
187  
188 /**  
189 * \defgroup Function Interrupt Service Routine  
190 */  
191  
192 /**  
193 * \brief          External Interrupt Service Routine  
194 * \note          If Event-Port (PC8) has a falling flank, this interrupt is triggered.  
195 *                The Direction of the Incremental Encoder (IE) depends on the state of  
196 *                Direction-Port (PC9).  
197 */  
198 void EXTI9_5_IRQHandler()  
199 {  
200     // Direction-Port is set -> direction of the IE is left  
201     if (GPIO_ReadInputDataBit(GPIOC, iE_Direction.GPIO_Pin) == Bit_SET)  
202     {  
203         volume--;  
204         if (volume < 0)  
205             volume = 0;  
206     }  
207  
208     // Direction-Port is not set -> direction of the IE is right  
209     else if (GPIO_ReadInputDataBit(GPIOC, iE_Direction.GPIO_Pin) == Bit_RESET)  
210     {  
211         volume++;  
212         if (volume > MAX_VOLUME)  
213             volume = MAX_VOLUME;  
214     }
```



```
215
216 // Reset the Interrupt
217 EXTI_ClearFlag(iE_EXTI_Event.EXTI_Line);
218 NVIC_ClearPendingIRQ(EXTI9_5_IRQn);
219 }
220
221 /**
222  * \}
223  * \defgroup Function vCOM - Functions
224  * \{
225  */
226
227 /**
228  * \brief          Process received data over UART
229  * \note           Either process them directly or copy to other bigger buffer
230  * \param[in]      data: Data to process
231  * \param[in]      len: Length in units of bytes
232  */
233 void usart_process_data(const void* data, size_t len)
234 {
235     const uint8_t* d = data;
236
237     /*
238      * This function is called on DMA TC and HT events, aswell as on UART IDLE (if enabled) line event.
239      *
240      * For the sake of this example, function does a loop-back data over UART in polling mode.
241      * Check ringbuff RX-based example for implementation with TX & RX DMA transfer.
242      */
243
```

```
244     for (; len > 0; --len, ++d) {
245         USART_SendData(USART1, *d);
246         while (!USART_GetFlagStatus(USART1, USART_FLAG_TXE)) {}
247     }
248     while (!USART_GetFlagStatus(USART1, USART_FLAG_TC)) {}
249 }
250
251 /**
252  * \brief          Send string to USART
253  * \param[in]      str: String to send
254  */
255 void usart_send_string(const char* str)
256 {
257     usart_process_data(str, strlen(str));
258 }
259
260 /**
261  * \}
262  * \defgroup Function Main Programm
263  * \{
264  */
265
266 /**
267  * \brief          Main Programm
268  */
```

```
269 int main()
270 {
271 // initialise IE
272
273 // Enable Peripheral Clock
274 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
275 RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
276
277 // initialise GPIO Port and Pins
278 GPIO_Init(GPIOC, &iE_Event);
279 GPIO_Init(GPIOC, &iE_Direction);
280
281 // initialise External Interrupt for IE-Event
282 AFIO->EXTICR[8/4] |= (AFIO_EXTICR3_EXTI8_PC);
283 EXTI_Init(&iE_EXTI_Event);
284
285 // NVIC Interrupt Initialisieren
286 NVIC_Init(&iE_NVIC);
287
288
289 // UART Init
290
291 // GPIO Clock Aktivieren
292 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
293 RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
294 RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
295
296 // GPIOs Initialisieren
297 GPIO_Init(GPIOA, &vCOM_Tx);
298 GPIO_Init(GPIOA, &vCOM_Rx);
```

```
299
300 // DMA Initialisieren
301 DMA_Init(DMA1_Channel1, &vCOM_DMA);
302 DMA_ITConfig(DMA1_Channel1, DMA1_IT_TC1 | DMA1_IT_HT1, ENABLE);
303
304 NVIC_Init(&vCOM_DMA_NVIC);
305
306 // UART Initialisieren
307 USART_ClockInit(USART1, &vCOM_clk);
308 USART_Init(USART1, &vCOM);
309
310 // NVIC Interrupt Initialisieren
311 NVIC_Init(&vCOM_NVIC);
312
313 // USART Staren
314 DMA_Cmd(DMA1_Channel1, ENABLE);
315
316 // USART Starten
317 USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
318 USART_Cmd(USART1, ENABLE);
319
320 // Send Init Strings
321 usart_send_string("is USART DMA functionable?\r\n");
322 usart_send_string("YES\r\n");
323
```

```
324 while(1)
325 {
326     // if there is a change in the volume
327     if (oldVolume != volume)
328     {
329         char str[16];
330         sprintf(str, "Volume:%02d\r\n", volume);
331         usart_send_string(str);
332         oldVolume = volume;
333     }
334 }
335 }
336
337 /**
338  * \}
339  */
340
341
```

5 Quellen

- [CM311] Michael Mötz – CM3 Arbeitsunterlagen – STM32F103RBT6 Entwicklungsumgebung
V1.3, 04.03.2011
HTBL Hollabrunn Interne Dokumentation
- [WP21] Ralph's – Lautstärkeregler Autoradio Bilddatei
<https://i1.wp.com/www.ralphs.ca/wp-content/uploads/2019/06/Turn-the-volume-up.jpg?ssl=1>
Letzter Aufruf: 10.04.2021