**DC Motor**

Inhaltsverzeichnis

[1 Aufgabenstellung 3](#_Toc516516979)

[2 Blockschaltbild 3](#_Toc516516980)

[3 Hardware/Stromlaufplan 4](#_Toc516516981)

[4 Timer – Berechnung 5](#_Toc516516982)

[4.1 Timer 1 – Echtzeituhr 5](#_Toc516516983)

[4.2 Timer 3 – Motoransteuerung 5](#_Toc516516984)

[4.3 Timer 4 – Drehzahlermittlung 5](#_Toc516516985)

[5 Messungen mit dem Oszilloskop 6](#_Toc516516986)

[5.1 PWM: 50% High und 50% Low – Phase 6](#_Toc516516987)

[5.2 PWM max: 90% High und 10% Low – Phase 6](#_Toc516516988)

[5.3 PWM min: 10% High und 90% Low – Phase 7](#_Toc516516989)

[6 Kommentierter Programmcode 8](#_Toc516516990)

[7 24 – Stundentest 14](#_Toc516516991)

[8 Zeitaufwand 15](#_Toc516516992)

[8.1 Berger 15](#_Toc516516993)

[8.2 Geyer 15](#_Toc516516994)

[9 Probleme 16](#_Toc516516995)

[10 Erkenntnisse 16](#_Toc516516996)

# Aufgabenstellung

Es ist ein mit Unterprogrammen strukturiertes Programm für den Cortex M3 zu schreiben, welches die Drehzahl eines DC-Motors mit TachoGenerator erfasst. Weiters soll eine Echtzeituhr realisiert werden (hh:mm:ss:z).

Die Uhr soll in der ersten Zeile des LCD-Displays angezeigt werden und die Drehzahl soll in der zweiten Zeile des LCD-Displays angezeigt werden.

Weiters wurde als Feature eingebaut, dass man mit dem DIL Potentiometer die PWM zur Motoransteuerung ständig verändern kann.

# Blockschaltbild

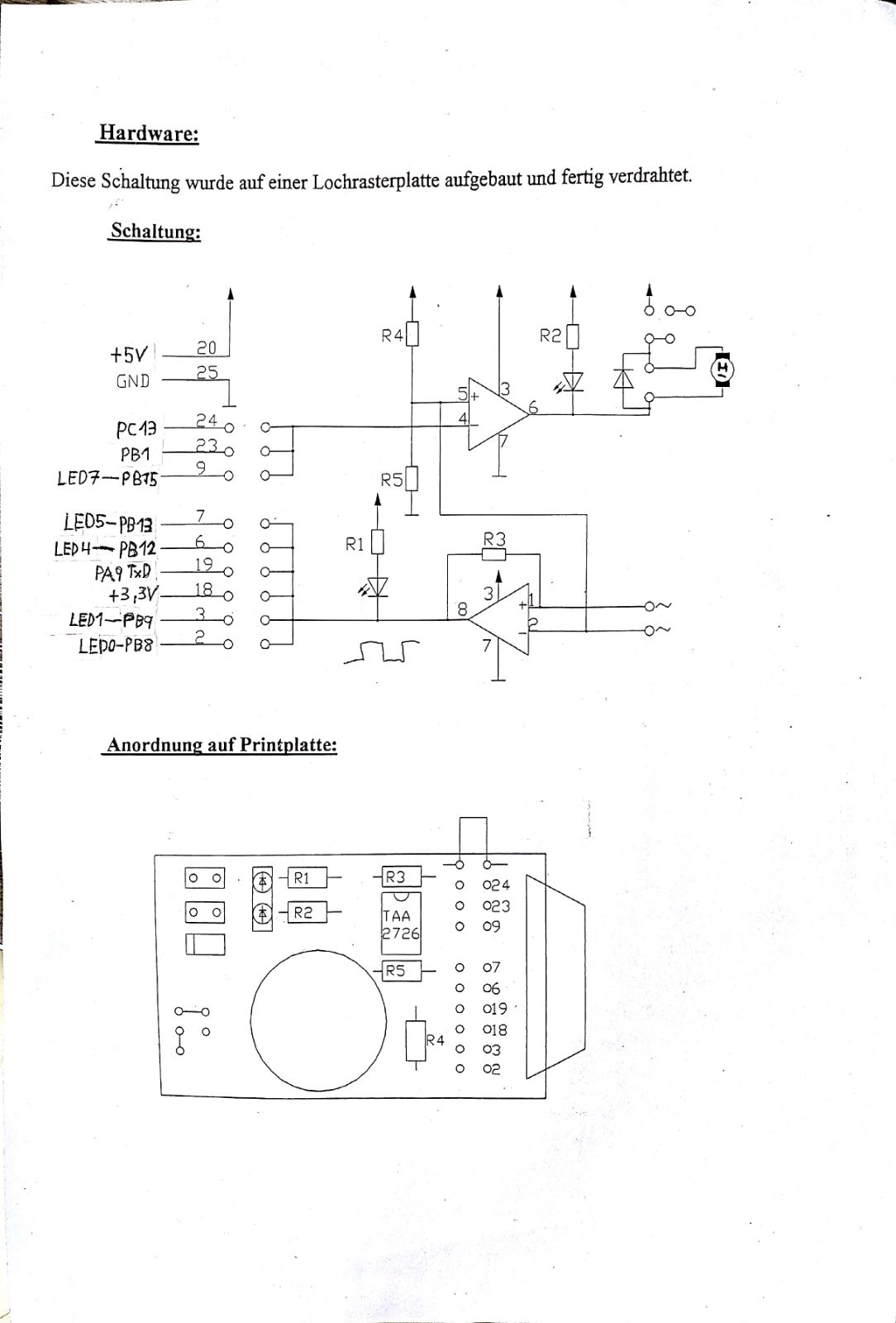
STM32F103RB

µC

außerhalb des µC

# Hardware/Stromlaufplan

In der folgenden Grafik wird gezeigt welche Jumper gesetzt werden müssen, damit PB1 als Motoransteuerung und PB9 zur Drehzahlerfassung verwendet werden kann.



Jumper gesetzt

# Timer – Berechnung

In dem Projekt wurden drei Timer verwendet, die jeweils eine andere Funktion übernommen haben. Für die Berechnung wurde ein Systemtakt von fsys=72 MHz (HSE) vorausgesetzt.

## Timer 1 – Echtzeituhr

Es soll ein Update-Interrupt alle 100ms ausgelöst werden:

(wegen 7200000/150)

## Timer 3 – Motoransteuerung

Bem.: Timer wird als Output Compare – Einheit verwendet

Es soll ebenfalls alle 100ms ein Interrupt ausgelöst werden:

(wegen 7200000/150)

## Timer 4 – Drehzahlermittlung

Bem.: Timer wird als Input Capture – Einheit verwendet

Der Wert des PSC=710 und des ARR=10000 wurden durch ausprobieren ermittelt.

# Messungen mit dem Oszilloskop

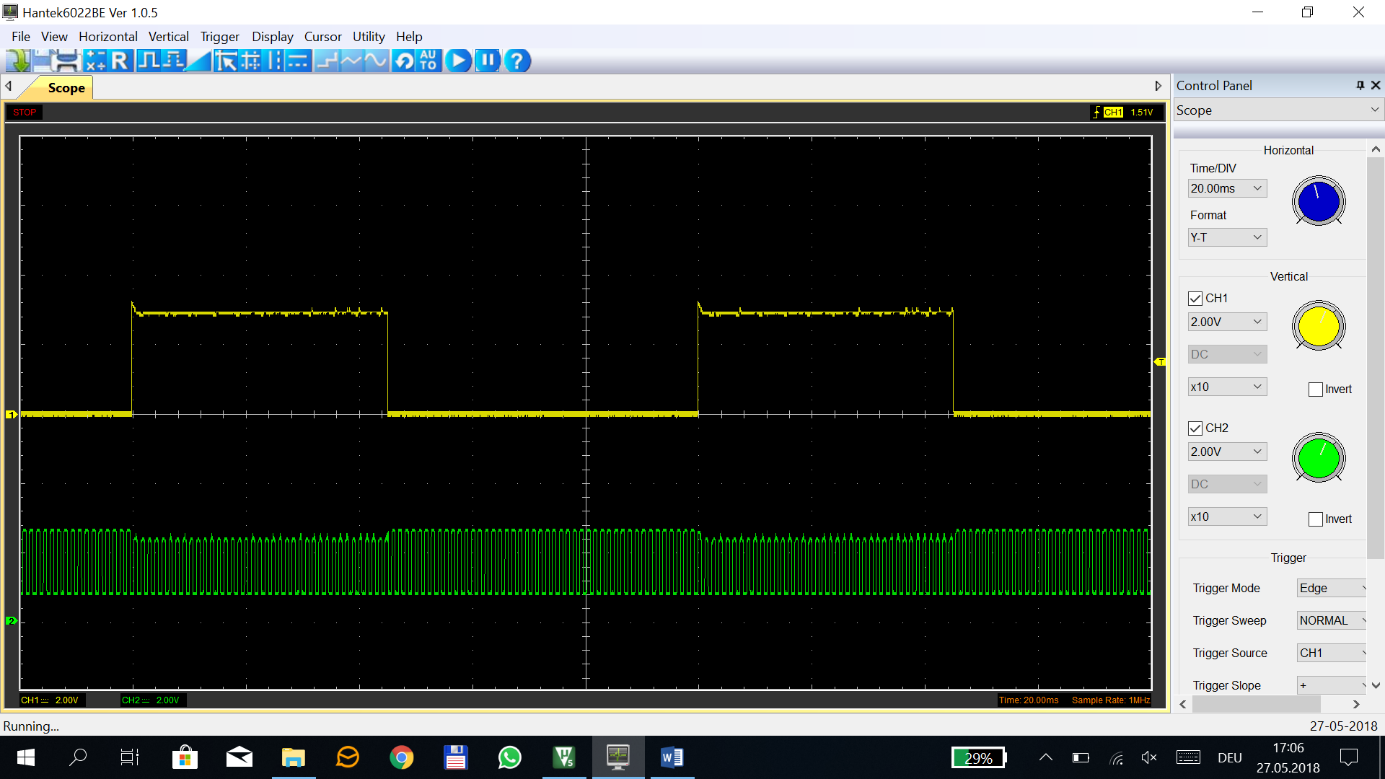
In den folgenden Darstellungen werden drei Fälle bei der Motoransteuerung dargestellt:

1. DIL Poti auf die Hälfte drehen -> PWM 50:50
2. DIL Poti auf Maximum drehen -> PWM 90:10
3. DIL Poti auf Minimum drehen -> PWM 10:90

Dabei wird die PWM zur Motoransteuerung selber gelb angezeigt und die Impulse die vom Motor-Encoder kommen, aus der in weiterer Folge die Drehzahl bestimmt wird, grün angezeigt.

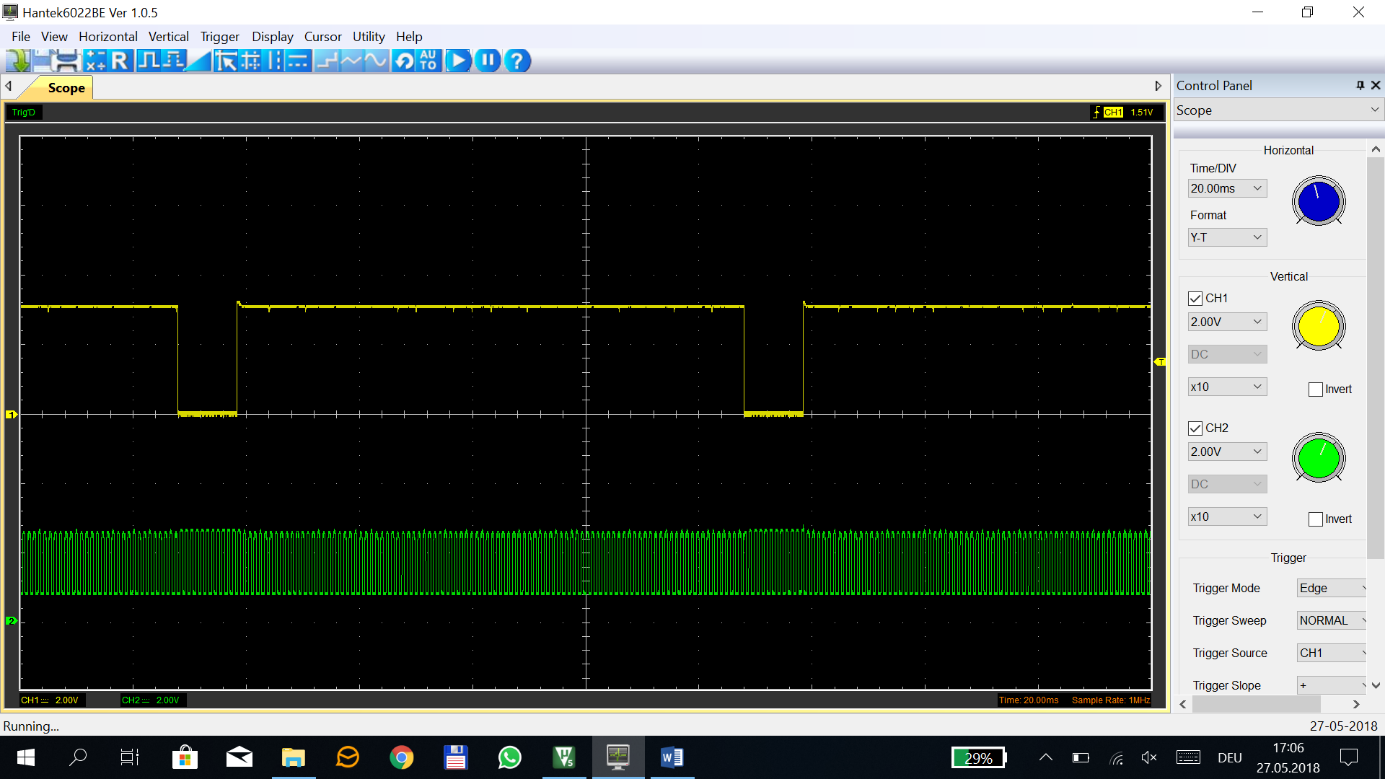
Bem.: Es wurde ein Minimum und ein Maximum realisiert, da eine PWM von 100% High und 0% Low und 0% High und 100% Low verhindert werden soll.

## PWM: 50% High und 50% Low – Phase



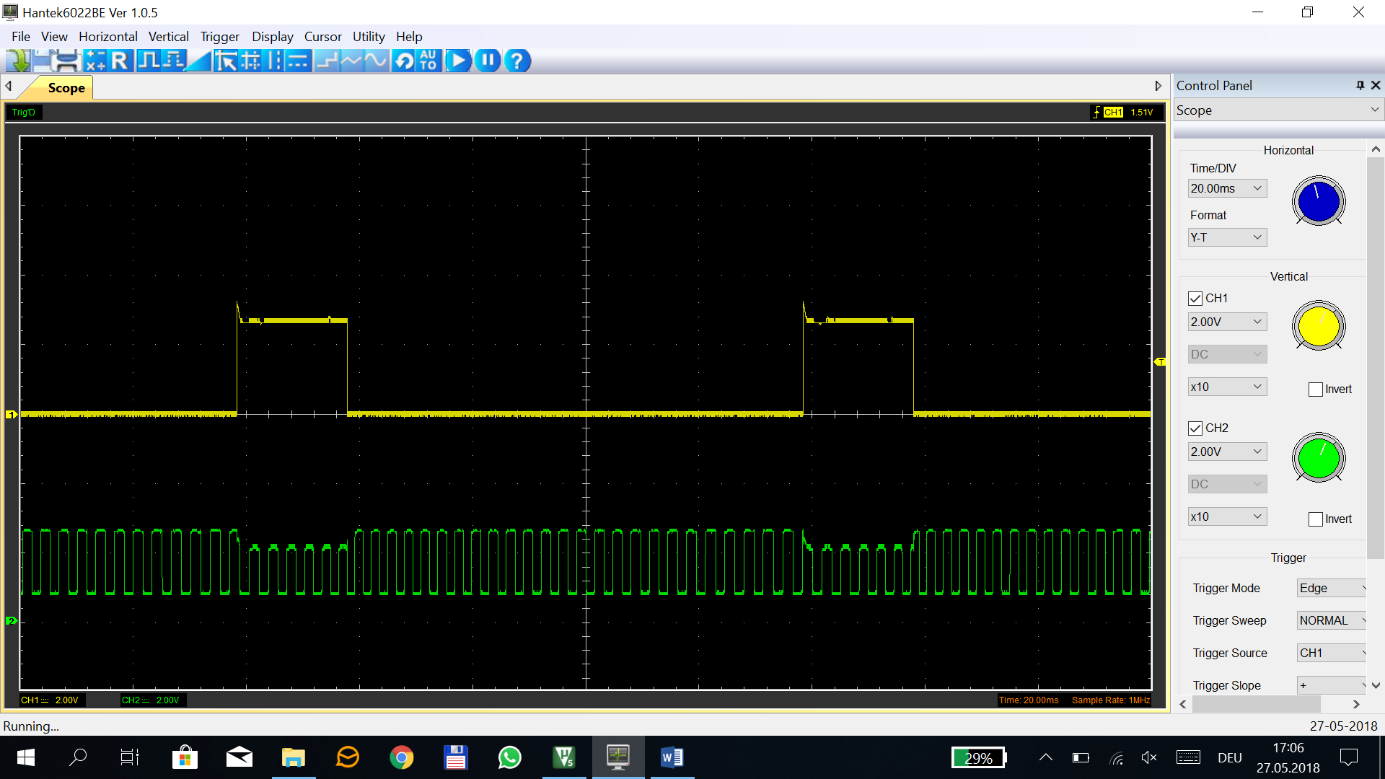
Dies ist ein klassischer Fall mit der PWM-Ansteuerung von ca. 50:50

## PWM max: 90% High und 10% Low – Phase



Bei diesem Fall kann die höchste Drehzahl erreicht werden. Dies wird an der sehr kurzen Impulsdauer der Impulse vom Motor-Encoder deutlich.

## PWM min: 10% High und 90% Low – Phase



Bei diesem Fall kann die geringste Drehzahl erreicht werden. Dies wird an der deutlich längeren Impulsdauer (Impulse kommen vom Motor-Encoder) deutlich.

# Kommentierter Programmcode

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* (C) Copyright PRIVATE All rights reserved. AUSTRIA \*/

/\* \*/

/\* File Name: DC\_Motor.c \*/

/\* Autor: Berger Jonas \*/

/\* Version: V1.00 \*/

/\* Date: 27.05.2018 (last change) \*/

/\* Description: Ansteuerung und Drehzahlbestimmung eines DC-Motors mit Uhr \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* History: V1.00 creation \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "armv10\_std.h"

/\*-------------------------- Function Prototypes -----------------------------\*/

void init\_ports**(**void**);** //initialisiert uC-Ports

void TIM1\_Config**(**void**);** //Timer 1 config, fuer Uhr

void TIM3\_Config**(**void**);** //Timer 3 config, fuer Motoransteuerung

void TIM4\_Config**(**void**);** //Timer 4 config, fuer Input Capture

void NVIC\_init**(**char position**,** char priority**);** //NVIC initialisieren

/\* ------------------------------- Defines -----------------------------------\*/

#define TIM\_IT\_Update ((uint16\_t)0x0001) //TIM\_interrupt\_source

#define TIM\_IT\_CC4 ((uint16\_t)0x0010) //TIM\_interrupt\_source

/\*---------------------------- Static Variables ------------------------------\*/

int sekunden**=**0**,**minuten**=**0**,**stunden**=**0**,**zehntel**=**0**;** //Zeitvariablen

double IC4ReadValue1**=**0**,**IC4ReadValue2**=**0**;** //Input Capture Variablen

int CaptureNumber**=**0**;** //Unterscheidung 1. oder 2. Input Capture

int Capture**=**0**;** //Capture Computation

int help**=**0**;** //help fuer LCD-Update

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Interrupt Service Routine Timer 1 (General Purpose Timer) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TIM1\_UP\_IRQHandler**(**void**)** //Timer 1, loest alle 1ms aus

**{**

TIM1**->**SR **&=** **~**0x01**;** //Interrupt pending bit loeschen (Verhinderung eines nochmaligen Interrupt-ausloesens)

help**=~**help**;** //Hilfsvariable fuer das LCD-Displayupdate (alle 100ms)

**if(**zehntel**==**10**)**

**{**

sekunden**++;**

zehntel**=**1**;**

**}**

**else**

**{**

zehntel**++;**

**}**

**if((**sekunden**)==**60**)**

**{**

minuten**++;**

sekunden**=**0**;**

**}**

**if(**minuten**==**60**)**

**{**

stunden**++;**

minuten**=**0**;**

**}**

**if(**stunden**==**24**)**

**{**

stunden**=**24**;**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Interrupt Service Routine Timer 4 (General Purpose Timer) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TIM4\_IRQHandler**(**void**)**

**{**

**if(((**TIM4**->**SR **&** TIM\_IT\_Update**)** **!=** 0**)** **&&** **((**TIM4**->**DIER**&**TIM\_IT\_Update**)** **!=** 0**))** //Update Interrupt Pending ?

**{**

TIM4**->**SR **&=** **~**TIM\_IT\_Update**;** //Clear TIM4 Update Interrupt pending bit

**}**

**if(((**TIM4**->**SR **&** TIM\_IT\_CC4**)** **!=** 0**)** **&&** **((**TIM4**->**DIER **&** TIM\_IT\_CC4**)** **!=** 0**))** //Capture Interrupt von Channel 4?

**{**

TIM4**->**SR **&=** **~**TIM\_IT\_CC4**;** //Clear TIM4 Capture Interrupt pending bit for Channel 4

**if(**CaptureNumber **==** 0**)**

**{**

IC4ReadValue1**=**TIM4**->**CCR4**;** //Get the Input Capture value for Channel 4

CaptureNumber**=**1**;**

**}**

**else** **if(**CaptureNumber **==** 1**)**

**{**

IC4ReadValue2**=**TIM4**->**CCR4**;** //Get the Input Capture value for Channel 4

**if(**IC4ReadValue2 **>** IC4ReadValue1**)** //Capture computation

**{**

Capture**=(**IC4ReadValue2**-**IC4ReadValue1**);**

**}**

**else**

**{**

Capture**=((**TIM4**->**ARR**-**IC4ReadValue1**)+**IC4ReadValue2**);**

**}**

CaptureNumber**=**0**;**

**}**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Initialization Timer 1 (General Purpose Timer) \*/

/\* Funktion: \*/

/\* initialisiert Timer 1, der fuer die Uhr zustaendig ist ->INT=alle 100ms \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TIM1\_Config**(**void**)**

**{**

/\*-------------------------- Timer 1 konfigurieren -------------------------\*/

RCC**->**APB2ENR **|=** RCC\_APB2ENR\_TIM1EN**;** //Timer 1 Clock enable

TIM1**->**SMCR **&=** **~**0x07**;** //Timer 1 Clock Selection: CK\_INT wird verwendet

TIM1**->**CR1 **&=** 0x00**;** //CMS=00: Edge-aligned mode. DIR=0 --> Upcounter

TIM1**->**PSC **=** 149**;** //Prescaler

TIM1**->**ARR **=** 48000**;** //Auto-Reload Wert = Maximaler Zaehlerstand des Upcounters

/\*--------------------- Interrupt Timer 1 konfigurieren --------------------\*/

TIM1**->**DIER **|=** 0x01**;** //enable Interrupt bei einem UEV (UEberlauf/Unterlauf)

NVIC\_init**(**TIM1\_UP\_IRQn**,**2**);** //Enable Timer 1 Update Interrupt, Priority 2

/\*---------------------------- Enable Timer 1 ------------------------------\*/

TIM1**->**CR1 **|=** 0x01**;** //Enable Timer 1

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Initialization Timer 3 (General Purpose Timer) \*/

/\* Funktion: \*/

/\* initialisiert Timer 3, der fuer die Motoransteuerung zustaendig ist \*/

/\* CC4 CH4(PB1) als Output Compare verwendet ->INT=alle 100ms \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TIM3\_Config**(**void**)**

**{**

/\*------------------------- Timer 3 konfigurieren --------------------------\*/

RCC**->**APB1ENR **|=** RCC\_APB1ENR\_TIM3EN**;** //Timer 3 Clock enable

TIM3**->**SMCR **&=** **~**0x07**;** //Timer 3 Clock Selection: CLK\_INT wird verwendet

TIM3**->**CR1 **|=** 0x010**;** // Auswahl des Timer Modus: Downcounter -> Zaehlt: von ARR bis 0

TIM3**->**PSC **=** 149**;** //Prescaler

TIM3**->**ARR **=** 48000**;** //Auto-Reload Wert = Maximaler Zaehlerstand des Upcounters

/\*--------------------- Interrupt Timer 3 konfigurieren --------------------\*/

TIM3**->**DIER **=** 0x01**;** //enable Interrupt bei einem UEV (UEberlauf / Unterlauf)

/\*------------- Configure Channel 4(PB1) as Output Compare -----------------\*/

TIM3**->**CCR4 **=** 20000**;** //1/4 von ARR-Wert=13200 als Level festgelegt

TIM3**->**CCMR2 **|=** 0x7800**;** //CC4S=00, OC4PE=1, OC4M=111

TIM3**->**CCER **|=** 0x01000**;** //Enable CH4 als Output Compare

/\*---------------------------- Enable Timer 3 ------------------------------\*/

TIM3**->**CR1 **|=** 0x01**;** //Enable Timer 3

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Initialization Timer 4 (General Purpose Timer) \*/

/\* Funktion: \*/

/\* initialisiert Timer 4, der fuer die Drezahl zustaendig ist \*/

/\* CC4 CH4(PB9) als Input Capture verwendet ->INT=alle 10ms \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TIM4\_Config**(**void**)**

**{**

/\*-------------------- Configure Timer 4 CR1 - Register --------------------\*/

RCC**->**APB1ENR **|=** RCC\_APB1ENR\_TIM4EN**;** //Timer 4 Clock enable

TIM4**->**SMCR **&=** **~**0x07**;** //Timer 4 Clock Selection: CLK\_INT wird verwendet

TIM4**->**CR1 **&=** 0x00**;** //CMS=00: Edge-aligned mode. DIR=0 --> Upcounter

TIM4**->**PSC **=** 710**;** //Prescaler

TIM4**->**ARR **=** 10000**;** //Auto-Reload Value

/\*---------------- Configure Channel 4(PB9) as Input Capture ---------------\*/

TIM4**->**CCMR2 **|=** 0x0100**;** //IC4, no filter

TIM4**->**CCER **|=** 0x01000**;**

/\*--------------------- Enable Interrupts for Timer 4 ----------------------\*/

TIM4**->**DIER **|=** 0x011**;** //Enable Update Interrupt, Enable the CC4 Interrupt Request

NVIC\_init**(**TIM4\_IRQn**,**1**);** //Enable Timer 4 Interrupt, Priority 1

/\*---------------------------- Enable Timer 4 ------------------------------\*/

TIM4**->**CR1 **|=** 0x01**;** //Enable Timer 4

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* NVIC\_init(char position, char priority) \*/

/\* Funktion: \*/

/\* UEbernimmt die vollstaendige Initialisierung eines Interrupts im Nested \*/

/\* Vectored Interrupt Controller (Prioritaet setzen, Ausloesen verhindern, \*/

/\* Interrupt enablen) \*/

/\* UEbergabeparameter: "position" = 0-67 (Nummer des Interrupts) \*/

/\* "priority" : 0-15 (Prioritaet des Interrupts) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void NVIC\_init**(**char position**,** char priority**)**

**{**

NVIC**->**IP**[**position**]=(**priority**<<**4**);** //Setzen der Interrupt Prioritaet

NVIC**->**ICPR**[**position **>>** 0x05**]** **|=** **(**0x01 **<<** **(**position **&** 0x1F**));** //Verhindert, dass der Interrupt ausloest sobald er enabled wird

NVIC**->**ISER**[**position **>>** 0x05**]** **|=** **(**0x01 **<<** **(**position **&** 0x1F**));** //Enable Interrupt

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* init\_ports() \*/

/\* Funktion: \*/

/\* initialisiert uC-Ports (PB9 as IC, PB1 as OC) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void init\_ports**(**void**)**

**{**

RCC**->**APB2ENR **|=** RCC\_APB2ENR\_IOPBEN**;** //GPIOB Clock enable

GPIOB**->**CRH **&=** **~**0x000000F0**;** //PB9 Konfigurationsbit loeschen

GPIOB**->**CRH **|=** 0x00000080**;** //PB9 auf IPUP fuer Input Capture: Drehzahl

GPIOB**->**ODR **|=** 0x0200**;** //Pull-Up fuer PB9 aktivieren

GPIOB**->**CRL **&=** **~**0x000000F0**;** //PB1 Konfigurationsbit loeschen

GPIOB**->**CRL **|=** 0x000000B0**;** //PB1 auf AFPP fuer Output Compare: Motorantrieb

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* MAIN function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main**(**void**)**

**{**

set\_clock\_72MHz**();** //SYS\_CLK auf HSE 72MHz einstellen

char buffer**[**50**];** //Variable fuer LCD-Display

int lcd\_sekunden**=**0**,**lcd\_minuten**=**0**,**lcd\_stunden**=**0**,**lcd\_zehntel**=**0**;** /\*Variablen fuer ...\*/

char einh**[**10**],**ein**[**10**],**zeh**[**10**],**hour**[**10**],**min**[**10**],**eimin**[**10**],**sek**[**10**];** /\*... Uhrenanzeige\*/

double um**=**0**;** //Drezahl

int ADC**;** //Wert des DIL-Poti fuer PWM-Einstellung

lcd\_init**();** //LCD Anzeige initialisieren

init\_ports**();** //initialisiert uC-Ports

TIM1\_Config**();** //Timer 1 config, fuer Uhr

TIM3\_Config**();** //Timer 3 config, fuer Motoransteuerung(Output Compare)

TIM4\_Config**();** //Timer 4 config, fuer Drezahl(Input Capture)

**do**

**{**

**if** **(**help **==** 0**)** //loest alle 100ms ein LCD-Update aus; help von TIM1-ISR gesteuert

**{**

lcd\_clear**();**

/\*------------------------- Uhr-Anzeige ----------------------------------\*/

**if** **(**lcd\_zehntel **!=** zehntel**)** //1 Zehntel vergangen

**{**

lcd\_zehntel **=** zehntel**;**

**}**

**if** **(**lcd\_sekunden **!=** sekunden**)** //1 Sekunde vergangen ?

**{**

lcd\_sekunden **=** sekunden**;**

**}**

**if** **(**lcd\_minuten **!=** minuten**)** //1 Minute vergangen ?

**{**

lcd\_minuten **=** minuten**;**

**}**

**if** **(**lcd\_stunden **!=** stunden**)** //1 Stunde vergangen ?

**{**

lcd\_stunden **=** stunden**;**

**}**

lcd\_set\_cursor**(**0**,**0**);** // Cursor auf Ursprung

//Stundenanzeige

int einer **=** lcd\_stunden**/**10**;**

sprintf**(&**einh**[**1**],**"%d"**,** einer**);**

lcd\_put\_string**(&**einh**[**1**]);**

//Einerstundenanzeige

int hou **=** lcd\_stunden**%**10**;**

sprintf**(&**hour**[**1**],**"%d:"**,** hou**);**

lcd\_put\_string**(&**hour**[**1**]);**

//Minutenanzeige Zehnerstelle

int einermin **=** lcd\_minuten**/**10**;**

sprintf**(&**eimin**[**1**],**"%d"**,** einermin**);**

lcd\_put\_string**(&**eimin**[**1**]);**

//Minutenanzeige Einerstelle

int mins **=** lcd\_minuten**%**10**;**

sprintf**(&**min**[**1**],**"%d:"**,** mins**);**

lcd\_put\_string**(&**min**[**1**]);**

//Sekundenanzeige -

sprintf**(&**sek**[**1**],**"%d"**,** lcd\_sekunden**/**10**);**

lcd\_put\_string**(&**sek**[**1**]);**

// - geteilt in einer und zehner stellen

int einersek **=** lcd\_sekunden**%**10**;**

sprintf**(&**zeh**[**1**],**"%d:"**,** einersek**);**

lcd\_put\_string**(&**zeh**[**1**]);**

//Anzeige der zehntel

int zehntel **=** lcd\_zehntel**%**10**;**

sprintf**(&**ein**[**1**],**"%d"**,** zehntel**);**

lcd\_put\_string**(&**ein**[**1**]);**

/\*---------------- Motor PWM-Einstellung mit DIL-Poti --------------------\*/

ADC**=**adc1\_convert**(**14**);** //Wert von 0-4000; Channel 14 entspricht DIL-Poti (PC4)

ADC**=**ADC**\***10**;** //Wert\*10

**if(**ADC**<=**5000**)** //PWM von 100%->High und 0%->Low verhindern

**{**

ADC**=**5000**;**

**}**

**if(**ADC**>=**37440**)** //PWM von 0%->High und 100%->Low verhindern

**{**

ADC**=**37440**;**

**}**

TIM3**->**CCR4**=**ADC**;** //ADC-Wert als Output Compare Level verwenden

/\*--------------------- Drehzahl-Anzeige --------------------------------\*/

um**=**60000**/**Capture**;** //Umdrehungen pro min berechnen

lcd\_set\_cursor**(**1**,**0**);**

sprintf**(&**buffer**[**0**],** "n=%.0lf Umd./min"**,**um**);** //Drezahl Umdr./min

lcd\_put\_string**(&**buffer**[**0**]);**

**}**

**}**

**while(**1**);**

**}**

# 24 – Stundentest

Der 24 – Stundentest ergab eine Abweichung von 3,328 Sekunden.