

Drehzahlmesser Brushless Motor

Bauanleitung für einen einfachen MCU Brushless Drehzahlmesser. Die Drehzahlmessung erfolgt mit einem Microchip 8bit Controller 16F688. Der Drehzahlmesser wird an einem der drei Motorphasen angeschlossen. Über einen Spannungsteiler wird die Phasenspannung für die MCU entsprechend heruntergeteilt. Die Phasenspannung muss gefiltert werden, da beim Schalten der anderen Phasen spikes entstehen, welche sonst zu Fehlmessungen führen. Der PIC misst die Periodendauer einer elektrischen Umdrehung. Über die Polpaarzahl wird die mechanische Drehzahl bestimmt.

Die Software ist in GCBASIC geschrieben.

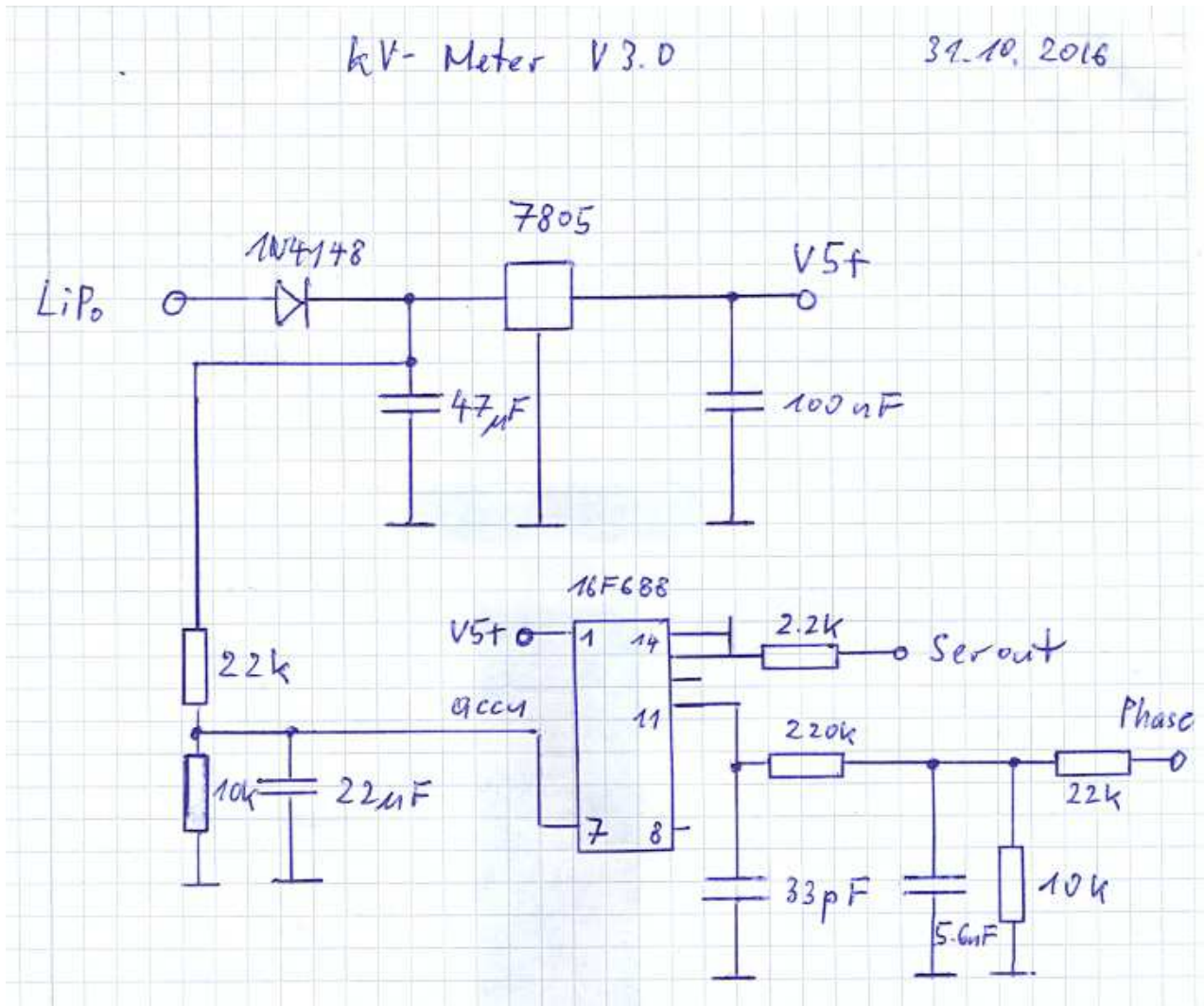
<http://gcbasic.sourceforge.net/>

Brushless Fundamentals:

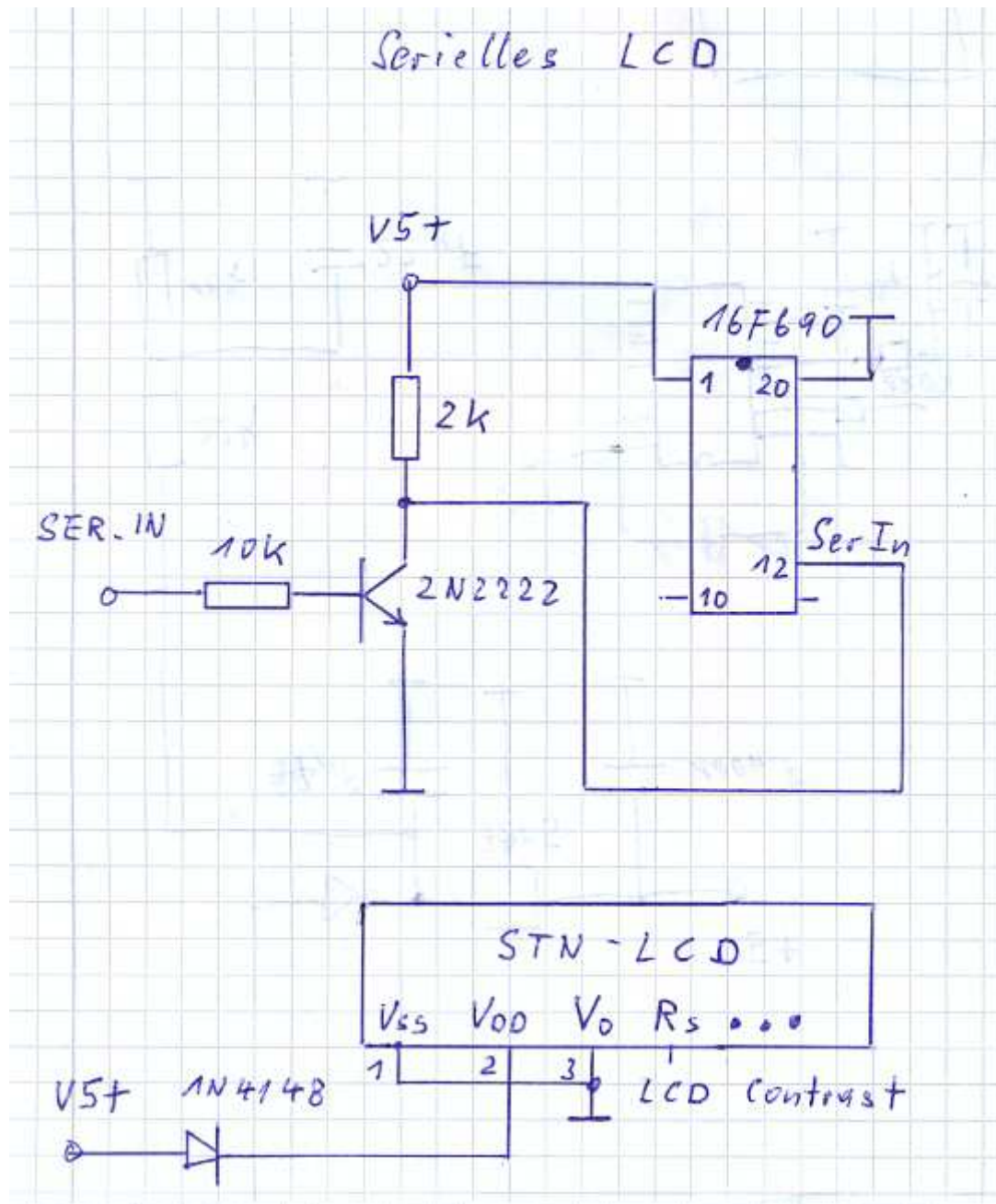
<http://ww1.microchip.com/downloads/en/AppNotes/00885a.pdf>

<http://ww1.microchip.com/downloads/en/AppNotes/01083a.pdf>

Schaltplan Drehzahlmesser



Serielles LCD



Program Drehzahlmesser

```
' KV-Meter V3.03
' 30.10.2016
' Compiler v95 2015-12-27
'
' -----
' V+          -| 1    14 |- GND
' PORTA.5      -| 2    13 |- PORTA.0 SER_OUT
' PORTA.4      -| 3    12 |- PORTA.1
' PORTA.3      -| 4    11 |- PORTA.2 MOT_PHASE
' PORTC.5      -| 5    10 |- PORTC.0
' PORTC.4      -| 6     9 |- PORTC.1
' PORTC.3 ACCU -| 7     8 |- PORTC.2
'
' -----

#include "software_uart_v1.h"

#chip 16F688,8
#config OSC=INTRC_OSC_NOCLKOUT

#define Vf 900 'Forward voltage 1N4148

'Serial settings
#define BAUD_RATE 2400
#define SER_OUT PORTA.0
dir SER_OUT out

'Voltage divider 10K/(10K+22K), 5.6nF||10K
' 220K to PORTA.2, 33pF from PORTA.2 to GND
#define MOT_PHASE PORTA.2
dir MOT_PHASE in

'Voltage divider 10K/(10K+22K), 22uF||10K
dir PORTC.3 in 'AN7 accu voltage

dim accu,ISR1_counter as byte

dim TimerValue,TimerValue1,TimerValue2,rpm as word

accu = 0
ISR1_counter = 0

TimerValue = 0
TimerValue1 = 0
TimerValue2 = 0
rpm = 0

'init
SerPrint("V3.03")
wait 3 s

bsf OPTION_REG,INTEDG ; set external interrupt on rising edge

On Interrupt ExtInt0 Call ISR1

InitTimer1 Osc, PS1_2 '8 Mhz/4/2, 1 us resolution
StartTimer 1

'----- Main loop -----
```

```

Do
  IntOff
  accu = ReadAD(AN7)
  LCD_Clear
  print_2p1([word]accu * 63 + Vf)
  SerPrint(" V")
  if (TimerValue > 40) then
    rpm = ([long]192000 / (TimerValue)) * 31
    if (rpm > 5000) then
      rpm = rpm / 7 'outrunner with 14 poles
    end if
    LCD_Line2
    print_int(rpm * 10,5)
  end if
  IntOn

  wait 1 s
Loop

sub ISR1 'external interrupt
  ISR1_counter++
  if (ISR1_counter = 1) then TimerValue1 = Timer1
  if (ISR1_counter = 2) then
    ISR1_counter = 0
    TimerValue2 = Timer1
    if (TimerValue2 >= TimerValue1) then
      TimerValue = TimerValue2 - TimerValue1
    else
      TimerValue = (0xFFFF - TimerValue1) + TimerValue2
    end if
  end if
end sub 'ISR1
'-----

```

Program serielles Display

```
' Serial LCD Display Gateway for HD44780 LCD Display Controller V1.0
'
' 26.07.2016
' compiled with v95 2015-12-27
'
'
' -----
' V+                -| 1    20 |- GND
' PORTA.5           -| 2    19 |- PORTA.0
' PORTA.4           -| 3    18 |- PORTA.1
' PORTA.3(MCLR)     -| 4    17 |- PORTA.2
' PORTC.5           -| 5    16 |- PORTC.0 LCD_DB4
' PORTC.4           -| 6    15 |- PORTC.1 LCD_DB5
' PORTC.3 LCD_Enable -| 7    14 |- PORTC.2 LCD_DB6
' PORTC.6 LCD_RS     -| 8    13 |- PORTB.4 LCD_DB7
' PORTC.7 LCD_RW     -| 9    12 |- PORTB.5 SerInPort
' PORTB.7           -| 10   11 |- PORTB.6
' -----

#chip 16F690,8
#config OSC=INTRC_OSC_NOCLKOUT

'LCD setting 4 bit mode HD44780
#define LCD_IO 4
#define LCD_RS  PORTC.6
#define LCD_RW  PORTC.7
#define LCD_Enable PORTC.3
#define LCD_DB4  PORTC.0
#define LCD_DB5  PORTC.1
#define LCD_DB6  PORTC.2
#define LCD_DB7  PORTB.4

'Serial settings
COMPORT = 1 'Needs comport to be set first, otherwise sub HSerReceive is not working in v95 2015-12-27
#define USART_BAUD_RATE 2400
#define USART_BLOCKING

#define SerInPort PORTB.5
dir SerInPort In

dim Temp as byte
Temp = 0

'Init
display_ctrl = 0
CLS
```

'----- Main loop -----

Do

'Get a byte from RS232 Rx
HSerReceive Temp

'Received Display CTRL command, ASCII DEZ 254
if (Temp = 254) then
 display_ctrl = 1
end if

' Clear Display, received ASCII DEZ 254, 1
if (Temp = 1) and (display_ctrl = 1) then
 CLS
 display_ctrl = 0
end if

' Display Line 1, pos 1, received ASCII DEZ 254, 128
If (Temp = 128) and (display_ctrl = 1) then
 Locate 0, 0
 display_ctrl = 0
end if

' Display Line 2, pos 1, received ASCII DEZ 254, 192
If (Temp = 192) and (display_ctrl = 1) then
 Locate 1, 0
 display_ctrl = 0
end if

'Received ASCII DEZ between 32 and 127
if (Temp >= 32) and (Temp <= 127) then
 LCDWriteChar(Temp)
end if

Loop

UART Library

' Software UART LIB with inverted software UART and some print routines
' M. Schulz Update 17.08.2016, uart_inv() needs less program memory

#define Carry STATUS.0 ' Carry Bit of STATUS register

#script

if BAUD_RATE = 2400 then oneBitdelay = 204

if BAUD_RATE = 9600 then oneBitdelay = 48

#endscript

'-----
'***** AsyncTx invert *****
'* Asynchronous serial transmit byte routine *
'* 1 start bit + 8 data bits + 1 stop bit *
'* INVERTED *
'* Baud Rate: 2400 or 9600 *
'* Processor clock assumed: 8Mhz *
'* *
'* Uses: dout(byte to transmit) as input *
'* Uses: SER_OUT as SerOut PIN, SerOut pin must be *
'* defined at program start #define SER_OUT PORT.X *
'* *
'* Uses: BAUD_RATE for oneBitdelay setup, *
'* #define BAUD_RATE 2400 or 9600 at program start *
'*****

sub uart_inv(in dout as byte)

set SER_OUT on ' startbit

for bitCount = 1 to 9

uart_delay

Carry = 1 ' incl. stopbit

rotate dout right

if (Carry = 1) then set SER_OUT off

if (Carry = 0) then set SER_OUT on

next

uart_delay

end sub

'-----serial Display routines -----

sub uart_delay ; Loop needs oneBitdelay us

dim tmp as byte 'at 160 defines memory location

movlw oneBitdelay

movwf tmp

decf tmp,f

btfss STATUS,Z ;test if ZERO

goto \$-2

end sub

'-----

sub SerPrint(PrintData As String)

PrintLen = PrintData(0)

'Write Data

for SysPrintTemp = 1 to PrintLen

uart_inv(PrintData(SysPrintTemp))

next

end sub 'SerPrint


```

'-----
sub LCD_Clear          ' Clear Diplay
    uart_inv(254)
    uart_inv(1)
end sub 'LCD_Clear
'-----

sub LCD_Line1          ' Diplay Line 1, pos 1
    uart_inv(254)
    uart_inv(128)
end sub 'LCD_Line1
'-----

sub LCD_Line2          ' Diplay Line 2, pos 1
    uart_inv(254)
    uart_inv(192)
end sub 'LCD_Line2
'-----

'Print unsigned integer as float, max 65536 = 65.54
sub print_2p2(In value as word) 'e.g. _9.99, value = 9999, _ is a blank
    dim buffer as word
    dim div as word
    dim value_copy as word

    div = 10000
    value_copy = value

    for lcd_count = 1 to 5
        buffer = value / div
        value = value % div
        if (lcd_count < 5) then
            if (value_copy < 10000) and (lcd_count = 1) then
                uart_inv(32) 'print blank
            else
                uart_inv(buffer + 48)
            end if
        end if 'lcd_count < 5)

        div = div / 10

        if (lcd_count = 2) then uart_inv(46) 'print dot
    next
end sub 'print_2p2
'-----

'Print unsigned integer as float, max 65536 = 65.5
sub print_2p1(In value as word)
    dim buffer as word
    dim div as word

    div = 10000
    zero = TRUE
    pos_3 = FALSE
    pos_4 = FALSE

    rest = value % 100

    if (rest > 50) then
        value = value + 100 - rest
    end if

```

```

if (value >= 1000) then pos_4 = TRUE
if (value > 99 and value < 1000) then pos_3 = TRUE
if (value > 999) and (value < 10000) then SerPrint(" ")
if (value < 100) then
    SerPrint(" 0.0")
    Exit Sub
end if

for lcd_count = 1 to 3
    buffer = value
    buffer = buffer / div
    if (buffer <> 0) then zero = FALSE
    if (pos_3 and (lcd_count = 3)) then SerPrint(" 0.")

    if (not zero) then uart_inv(buffer + 48)
    'if (zero and (not pos_3)) then SerPrint(" ")

    if (pos_4 and (lcd_count = 2)) then SerPrint(".")
    value = value % div
    div = div / 10
next
end sub 'print_2p1
'-----
sub print_int(In value as word, In count as byte) 'max. 65535 (5 count's), min. 3 count's
    dim buffer as word
    dim div as word

    dim zero as byte
    zero = TRUE

    div = 100
    if (count = 4) then div = 1000
    if (count = 5) then div = 10000

    for lcd_count = 1 to count
        buffer = value / div
        value = value % div
        if (buffer > 0) then zero = FALSE
        if ((zero) and (lcd_count < count)) then
            uart_inv(32) 'print blank
        else
            uart_inv(buffer + 48)
        end if
        div = div / 10
    next
    uart_inv(32) 'print blank
end sub 'print_int
'-----

```

```

sub print_byte(In value as byte) 'e.g. _25, value = 25, _ is a blank
  dim buffer as byte
  dim div,zero as byte

  div = 100
  zero = TRUE

  for lcd_count = 1 to 3
    buffer = value / div
    value = value % div
    if (buffer > 0) then zero = FALSE

    if ((zero) and (lcd_count < 3)) then
      uart_inv(32) 'print blank
    else
      uart_inv(buffer + 48)
    end if

    div = div / 10
  next
  uart_inv(32) 'print blank
end sub 'print_byte
'-----

```