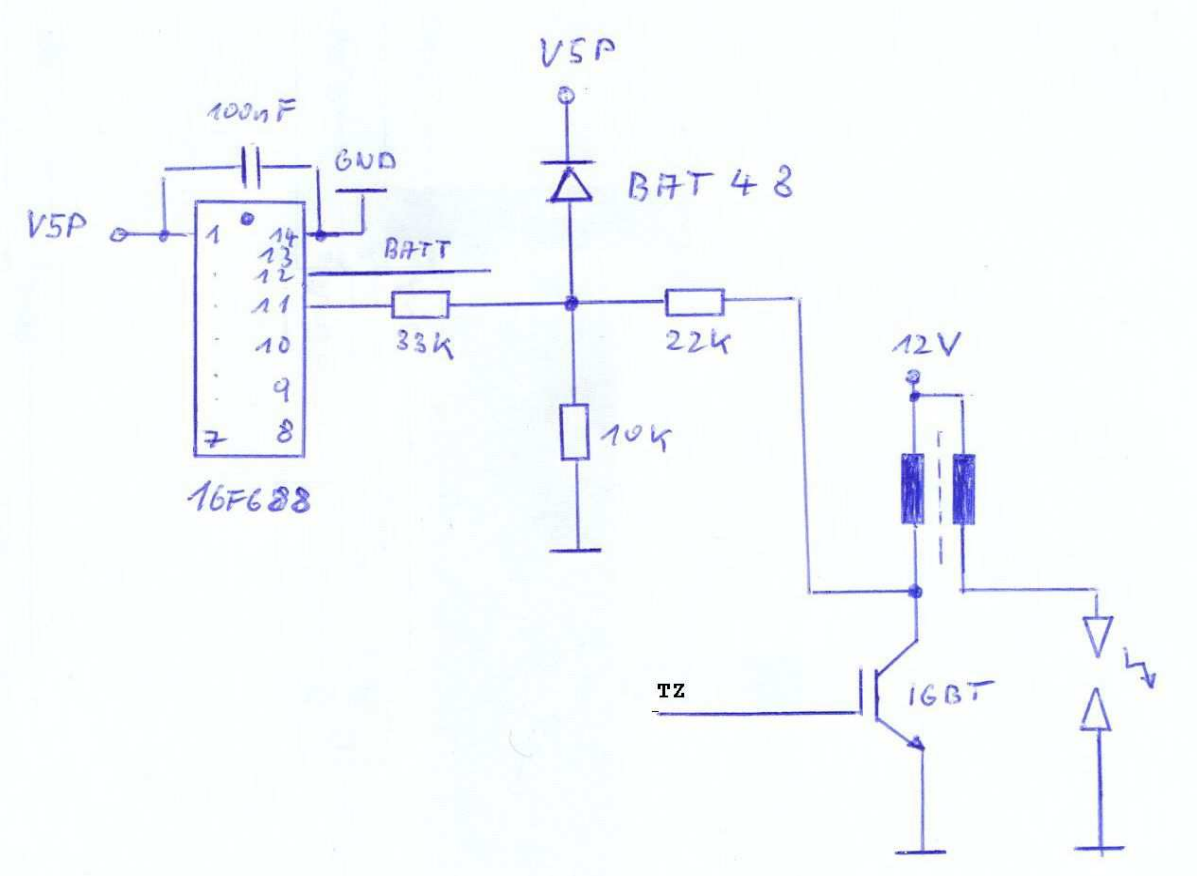
Drehzahlmesser 2-Takt Motor

Doku für einen 2-Takt Motor Drehzahlmesser. Die Drehzahlmessung erfolgt mit einem Microchip 8bit Controller 16F688. Als Display wird ein LCD-Modul C0802-04 mit HD47780 kompatiblen Kontroller verwendet.

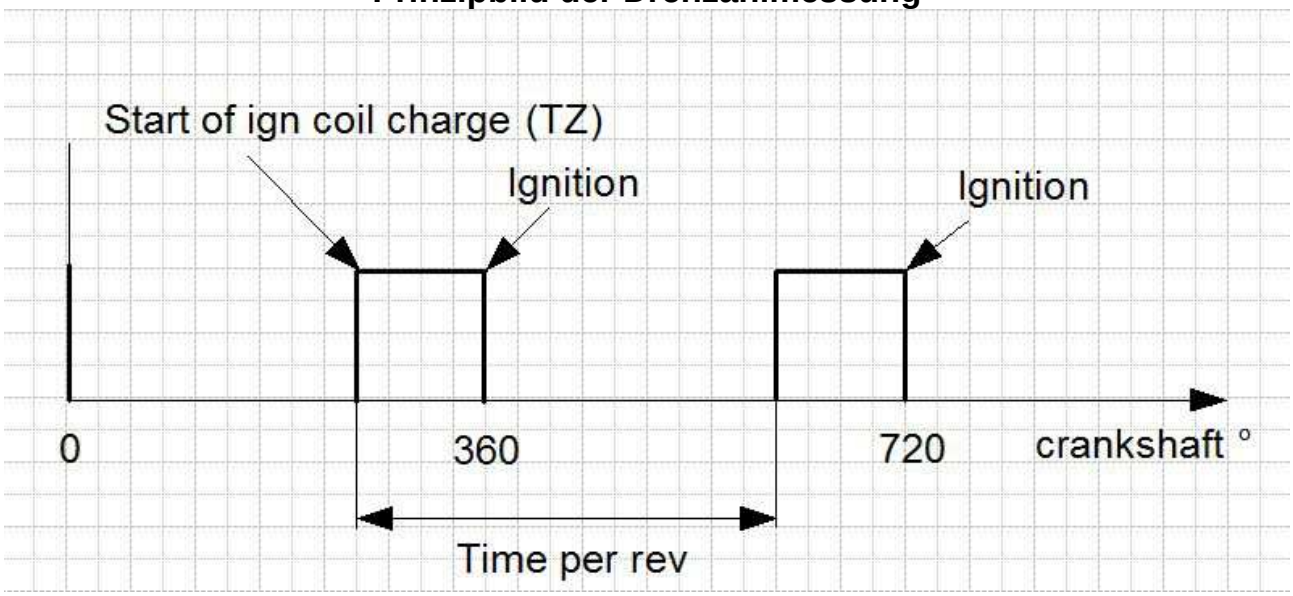
Die Software ist in GCBASIC geschrieben.

http://gcbasic.sourceforge.net/

Interface Drehzahlmesser



Prinzipbild der Drehzahlmessung



Programm Drehzahlmesser

```
'  RPM BATT MON V1.1
'  25.06.2016
'  Compiler v95.02
'                                ------------
'  V+                          -| 1       14 |- GND
'  PORTA.5 LCD_DB4             -| 2       13 |- PORTA.0
'  PORTA.4 LCD_DB5             -| 3       12 |- PORTA.1  BATTERY_IN
'  PORTA.3                     -| 4       11 |- PORTA.2  IGN_PRIM
'  PORTC.5 LCD_DB6             -| 5       10 |- PORTC.0
'  PORTC.4 LCD_DB7             -| 6        9 |- PORTC.1  LCD_RS
'  PORTC.3 LCD_Enable          -| 7        8 |- PORTC.2
'                                ------------
'

#include "display_new.h"

#chip 16F688,4
#config OSC = INTRC_OSC_NOCLKOUT, PWRTE = ON

#define Vf          610     'Forward voltage BAS32
#define VOLT_14P5 14500

#define TMR_CORR        12
#define TIME_13000rpm  576
#define TIME_11718rpm  640
#define TIME_1000rpm  7500

'LCD setting 4 bit mode
#define LCD_SPEED FAST
#define LCD_IO 4
#define LCD_NO_RW 'LCD_RW connected to GND

#define LCD_RS     PORTC.1
#define LCD_Enable PORTC.3
#define LCD_DB4    PORTA.5
#define LCD_DB5    PORTA.4
#define LCD_DB6    PORTC.5
#define LCD_DB7    PORTC.4

'voltage divider 1.5K /(1.5K + 5K), measured 3.28V/15V
'gain corrected from 89 to 86, at 12V displayed 12.4V with gain of 89

'ADC AN1 init
TRISA.1 = 1              ;Set PORTA.1 to input
ANSEL.ANS1 = 1           ;AN1 as analog input
ADCON0 = b'00000101'     ;Left justify, Vdd = Vref, AN1, ADC enabled
ADCON1.ADCS0 = 1         ;FOSC/8
'ADCON1.ADCS1 = 0
'ADCON1.ADCS2 = 0

#define IGN_PRIM PORTA.2
dir IGN_PRIM in

dim ISR1_counter as byte

dim TimerValue,TimerValue1,TimerValue2,TimerValue_copy as word
dim rpm,rpm_raw,rpm_old1,rpm_old2,rpm_old3,rpm_old4 as word

dim battery,battery_max,battery_old1,battery_old2,battery_old3,battery_old4 as word

dim battery_raw,batt_high_counter,case_counter,display_counter,main_counter as byte

TimerValue = 0
TimerValue_copy = 0
```

```
TimerValue1 = 0
TimerValue2 = 0
ISR1_counter = 0

rpm = 0
rpm_raw = 0
rpm_old1 = 0
rpm_old2 = 0
rpm_old3 = 0
rpm_old4 = 0

battery_raw = 0
batt_high_counter = 0
case_counter = 0
display_counter = 0
main_counter = 0

battery = 0
battery_max = 0
battery_old1 = 0
battery_old2 = 0
battery_old3 = 0
battery_old4 = 0

'init
CLS
Print("RPM_BATT")
Locate 1,0
Print("V1.1")
wait 2 s

bcf OPTION_REG,INTEDG ; set external interrupt on falling edge, start of ignition coil
charge

On Interrupt ExtInt0 Call ISR1

InitTimer1 Osc, PS1_8 '4 Mhz/4/8, 8 us resolution, max. 524 ms pulse length
StartTimer 1

CLS

'------------------- Main loop ----------------------------------------------
Do
    battery_raw = ([word]GET_ADC + GET_ADC ) / 2

    TimerValue_copy = TimerValue
    if ((TimerValue_copy <= TIME_11718rpm) or (TimerValue_copy >= TIME_1000rpm)) then
       rpm_raw = 0
    else
       rpm_raw = ([long]240000 / (TimerValue_copy + TMR_CORR)) * 31
    end if

    case_counter++

    Select Case case_counter

    Case 1
    DISPLAY_UPDATE
    battery_old1 = battery_raw
    rpm_old1 = rpm_raw

    Case 2
    battery_old2 = battery_raw
    rpm_old2 = rpm_raw

    Case 3
    battery_old3 = battery_raw
    rpm_old3 = rpm_raw
```

```
    Case 4
    battery_old4 = battery_raw
    rpm_old4 = rpm_raw
    case_counter = 0

    End Select

    battery = (battery_old1 + battery_old2 + battery_old3 + battery_old4) / 4
    battery = (battery * 86) + Vf

    if (battery > battery_max) then battery_max = battery

    main_counter++
    if (main_counter < 3) then
       rpm = (rpm_old1 + rpm_old2 + rpm_old3 + rpm_old4) / 4
    else
       rpm = 0
       main_counter = 3
    end if

    wait 250 ms
Loop

'------------------------------- subs ---------------------------------------
sub ISR1 'external interrupt
    ISR1_counter++
    main_counter = 0
    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
       if (TimerValue <= TIME_13000rpm) then ' masking of spikes due to ignition
          TimerValue = TimerValue_copy
          ISR1_counter = 1
       end if
    end if
end sub 'ISR1
'------------------------------------------------------------------------------
sub DISPLAY_UPDATE
    INTCON.INTE = 0 'disable ExtInt0
    CLS
    if (rpm > 0) then
       print_int(rpm,4)
       print(" rpm")
    end if
    Locate 1,0
    batt_high_counter++
    if ((batt_high_counter >= 60) and (battery_max > VOLT_14P5)) then
       print_2p1(battery_max)
       print(" VH")
    else
       print_2p1(battery)
       print(" V ")
    end if
    ISR1_counter = 0
    if (batt_high_counter >= 64) then batt_high_counter = 0
    INTCON.INTE = 1 'enable ExtInt0
end sub 'DISPLAY_UPDATE
'------------------------------------------------------------------------------
```

```
function GET_ADC as byte
    dim A_Delay as byte

    'Acquisiton delay
    A_Delay = 4 '20 us delay
    LABEL_INC:
    A_Delay--
    if (A_Delay <> 0) then goto LABEL_INC
    ;end Acquisiton delay
    ADCON0.GO = 1 ;Start conversion
    LABEL_ADCONV:
    if (ADCON0.GO = 1) then goto LABEL_ADCONV  ;Is conversion done? No, test again
    GET_ADC = ADRESH ;Read 8 bit ADC value
end function
```

Display Library

```
' LCD Display print routines
' update 11.10.2015

'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
          LCDWriteChar(32) 'print blank
       else
          LCDWriteChar(buffer + 48)
       end if
    end if '(lcd_count < 5)

    div = div / 10

    if (lcd_count = 2) then LCDWriteChar(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 LCDWriteChar(32) 'print blank
if (value < 100) then
   'Print(" 0.0")
   LCDWriteChar(32)
   LCDWriteChar(48)
   LCDWriteChar(46)
   LCDWriteChar(48)
   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
       'Print(" 0.")
       LCDWriteChar(32)
       LCDWriteChar(48)
       LCDWriteChar(46)
    end if

    if (not zero) then LCDWriteChar(buffer + 48)
    'if (zero and (not pos_3)) then Print(" ")

    if (pos_4 and (lcd_count = 2)) then LCDWriteChar(46) 'print dot
    value = value % div
    div = div / 10
next
end sub 'print_2p1
'-------------------------------------------------------------------------------
sub print_byte(In value as byte) 'e.g. _25, value = 25, _ is a blank
dim buffer as byte
dim div as byte
dim 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
       LCDWriteChar(32) 'print blank
    else
       LCDWriteChar(buffer + 48)
    end if

    div = div / 10
next
LCDWriteChar(32) 'print blank
end sub 'print_byte
'--------------------------- print_int -------------------------------------
' unsigned integer print routine 3-5 digits (count), max. 65535 (5 count's)
sub print_int(In value as word, In count as byte)
'e.g. 00123 with count = 5, value = 123
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
       LCDWriteChar(32) 'print blank
    else
       LCDWriteChar(buffer + 48)
    end if
    div = div / 10
next
end sub 'print_int
```