```
'SPEEDY.BAS
'Hardware:  32768 KHz-Quarz für Timer2-RTC
'           8 MHz intern. RC-Taktgenarator

'Fuses:     CKSEL = 0100
'           SUT = 10

'Framesize auf 32 setzen
'Softstack auf 32
'Hardstack auf 80

'Timer0: zählt Zündimpulse; wird alle 500ms ausgelesen
'Timer1: asynchron von Uhrenquarz; für Sekundenbildung und
'        Messung Radumdrehungszeit (InputCapture)
'Timer2: asynchron von Uhrenquarz; läuft alle 500ms über: dabei Auslesen Timer0

'-------------------------------------------------------------------------------
'$DBG
'$SIM
'$PROG &B11111111 , &B11100100 , &B11001001,

$Regfile = "m8def.dat"                  'ATmega8-Deklarationen
$Crystal = 8000000                      'Der AVR-Takt kommt vom RC-Oszillator
$baud = 9600

CONFIG LCDPIN = PIN , DB4 = PORTC.0 , DB5 = PORTC.1 , DB6 = PORTC.2 , DB7 = PORTC.3 , E = PORTC.5 , RS =
PORTC.4
Config LCD = 16 * 2
cls
cursor off noblink

'-------------------------------------------------------------------------------

CONFIG INT1 = FALLING                   'fallende Flanke
On INT1 OnInt1                          'PD3 = PIN 5 (INT1)

ddrd.3 = 0                              'als Interrupt-Eingang
portd.3 = 1                             'Pullup

ddrb.1 = 0                              'als Tasten-Eingänge
ddrb.2 = 0
portb.1 = 1                             'Pull-Up
portb.2 = 1

Taste1Direct alias pinb.1               'benötigt für "langes Drücken"-Erkennen

enable INT1

'-------------------------------------------------------------------------------

CONFIG TIMER0 = COUNTER , EDGE = FALLING       'kein Interrupt!
TIMER0 = 0

ddrd.4 = 0                              'PD4 = PIN6 (T0) (für Pickup)
portd.4 = 1                             'Pullup

'-------------------------------------------------------------------------------

'CONFIG TIMER1:
On OVF1 OnOVF1

ddrd.5 = 0                              'PD5 = PIN11 (T1) mit Uhrentakt (10) versorgt
'portd.5 = 1                            'kein Pullup
TCCR1A = &B00000000
TCCR1B = &B00000110                     'fallende Flanke an T1
TIMSK.TOIE1 = 1                         'Timer1-Overflow-Interrupt an


On OC1A OnOC1A
COMPARE1A = &H8000
TIMSK.OCIE1A = 1                        'Timer1-Compare-Interrupt bei 1/2 (=1sek) an


On ICP1 OnICP1
ddrb.0 = 0                              'PB0 = PIN14 (ICP) (für Radsensor)
portb.0 = 1                             'PullUp
TCCR1B.ICES1 = 1                        'fallende Flanke
'TCCR1B.ICNC1 = 1                       ' Noise Cancel
TIMSK.TICIE1 = 1                        'Input-Capture-Interrupt an

'-------------------------------------------------------------------------------

'CONFIG TIMER2:
```

```
On OVF2 OnOVF2                          'Interrupt-Routine für Timer2-Overflow

ASSR.AS2 = 1                            'Bit AS2=1: Asynchroner Timer2-Betrieb
TCCR2 = &B00000100                      'Takt 32768Hz/64= 0,5 Sek
TIMSK.TOIE2 = 1                         'Timer2-Overflow-Interrupt an

'-------------------------------------------------------------------------------

'CONFIG ACI:
On ACI OnACI
ACSR = &B01001011                       'Bandgap + Interr. enable bei rising edge
ddrd.7 = 0                              'Komparator-Eingang
ddrd.6 = 1                              'Discharge-Ausgang

'-------------------------------------------------------------------------------

'WaitMs 1000                             'Eine Sekunde warten bis Quarz schwingt
SREG.7 = 1                              'Interrupts global einschalten

'-------------------------------------------------------------------------------

dim wUmfang as word                     'Radumfang

dim lSekCount as long                   'zählt Sekunden
'(
dim wTimer1Capture as word              'für DragsterMode zur Zeitmessung
dim wTimer1Overflow as word             'dito
dim lKmDrag_mm as long                  'Km-Zähler für Dragster Mode
')
dim wCapture as word                    'captured Zählerstand1 für Geschwindigk.
dim wCaptureMin as Word                 'für Vmax
dim bfirstmax as Bit                    'Flag: Vmax zum 1. Mal zurücksetzen

dim bUsek as Byte                       'Motor-Umdrehungen / Sek

dim lTurnsTotCount as long              'Km-Zähler total zählt Radumdrehungen!!
dim lSekCountU as long                  'zählt Sek., wenn Motor läuft

'dim lKm1_mm as long                     'Tages-Km-Zähler 1+2
'dim lKm2_mm as long                     'zählt mm!!
dim lTurns1 as long                     'Tages-Km-Zähler 1+2
dim lTurns2 as long                     'zählt Radumdrehungen!!
dim lSekCount1U as long                 'Zeit Tag1
dim lSekCount2U as long                 'Zeit Tag2

dim bAnzeige as Byte                    'Anzeigemodus

dim EeDummy as ERAM word                'kann evtl. überschrieben werden
dim EeUmfang as ERAM Word               'Wert von wUmfang im EEPROM
dim EeCaptureMin as ERAM Word           'letzter Vmax
dim EeTurnsTotCount as ERAM long        'Km total
'dim EeKm1_mm as ERAM long               'Km Tag1
'dim EeKm2_mm as ERAM long               'Km Tag2
dim EeTurns1 as ERAM long               'Km Tag1
dim EeTurns2 as ERAM long               'Km Tag2
dim EeSekCountU as ERAM long            'Betriebsstundenzähler
dim EeSekCount1U as ERAM long           'Zeit Tag1
dim EeSekCount2U as ERAM long           'Zeit Tag2


dim sTemp as string * 16                'Arbeitsvariablen
dim aTemp(17) as byte at sTemp overlay
dim sTemp2 as string * 16
dim bTemp as Byte


declare sub intro()
declare sub first()
declare sub maximum()
declare sub total()
declare sub tageskm1()
declare sub tageskm2()
declare sub set_Umfang()
declare sub set_totalkm()
declare sub set_totalhours()

declare Function ClearButton(byval x as byte , byval y as byte) as byte
declare sub calc_kmh(w as word , s as string)
declare sub calc_time(seks as long , s as string)
declare sub calc_km_total(turns as long , s as string)
'declare sub calc_km12(mm as long , s as string)
declare sub calc_kmd12(Turns as long , s as string)
```

```
declare sub form_x_leer(byval x as byte , s as string)
declare sub form_x_0(byval x as byte , s as string)

declare Function is_key() as Byte
declare Function get_key() as Byte

const MainMod = 1                       'AnzeigeModi
const Vmax = 2
const Tag1Mod = 3
const Tag2Mod = 4
const TotalMod = 5
const stopp = 6

const UmfangMod = 7
const kmMod = 8
const hoursMod = 9

const Taste1 = 1
const Taste2 = 2
const beideTasten = 3
'      Taste1Direct = pinb.2 siehe oben


$EEPROM
'DATA 0%, 1500% , &HFFFF% , 0& , 0& , 0& , 0& , 0& , 0&       'als Defaultwerte
DATA 0% , 1500% , 0% , 0& , 0& , 0& , 0& , 0& , 0&       'als Defaultwerte
$DATA

wUmfang = EeUmfang                      'aus EEPROM
wCaptureMin = EeCaptureMin              'letzter Vmax
lTurnsToTCount = EeTurnsTotCount        'Km total
'lKm1_mm = EeKm1_mm                      'Km Tag1
'lKm2_mm = EeKm2_mm                      'Km Tag2
lTurns1 = EeTurns1                      'Km Tag1
lTurns2 = EeTurns2                      'Km Tag2
lSekCountU = EeSekCountU                'Betriebsstundenzähler
lSekCount1U = EeSekCount1U              'Zeit Tag1
lSekCount2U = EeSekCount2U              'Zeit Tag2

bAnzeige = MainMod                      'Default

call intro()
call first()

Do                                      'Hauptschleife
   bTemp = get_key()
   if bTemp = Taste2 then
      incr bAnzeige
      if bAnzeige = stopp then
         bAnzeige = MainMod
      end if

   end if

   if bTemp = beideTasten then
      bAnzeige = UmfangMod
   end if

   select case bAnzeige
     case MainMod : call first()        'Hauptanzeige
     case Vmax : call maximum()
     case Tag1Mod : call tageskm1()
     case Tag2Mod : call tageskm2()
     case TotalMod : call total()

     case UmfangMod : call set_Umfang()
                      incr bAnzeige
     case kmMod : call set_totalkm()
                  incr bAnzeige
     case hoursMod : call set_totalhours()       'danach
                     bAnzeige = MainMod       'zurück zur Hauptanzeige
   end select
Loop


end                                     'end program

'-------------------------------------------------------------------------------

sub intro()
  local b as byte
' ----------------
```

```
'  SPEEDY V. 1.0
' (c) by DiLi-Soft
' ----------------
  cls
  locate 1 , 17
  lcd " SPEEDY V. 1.0"
  locate 2 , 17
  lcd "(c) by DiLi-Soft"
  wait 1
  for b = 1 to 16
     shiftlcd left
     waitms 150
  next
  wait 2
  for b = 1 to 16
     shiftlcd right
     waitms 150
  next
  waitms 250
end sub


sub first()
local f as single , b as byte
' ----------------
' km/h    : ###.#
' 1000/min: ###.#
' ----------------
  cls
  do
    upperline
    call calc_kmh(wCapture , sTemp)
    lcd "km/h    : " ; sTemp

    f = bUsek * 0.12
    sTemp = Fusing(f , "#.#")
    call form_x_leer(5 , sTemp)
    lowerline
    lcd "1000/min: " ; sTemp

    b = is_key()
    waitms 100
  loop until b > 0                      'wenn Taste gedrückt: raus
end sub


sub maximum()
local b1 as byte , b2 as byte
' ----------------
' Vmax:
'       ###.#*
' ----------------
  cls
  do
    upperline
    lcd "Vmax:"

    lowerline
    call calc_kmh(wCaptureMin , sTemp)
    lcd spc(5) ; sTemp ; "*"

    b1 = is_key()
    waitms 100
    if b1 = Taste1 then
       b2 = ClearButton(11 , 2)
       if b2 = 2 then
          wCaptureMin = &HFFFF
          EeCaptureMin = wCaptureMin
       end if
    end if
  loop until b1 > Taste1
end sub


sub total()
local l as long , b as byte
' ----------------
' total km:######
' hours:   ######
' ----------------
  cls
  do
```

```
    upperline
    call calc_km_total(lTurnsTotCount , sTemp)
    call form_x_leer(6 , sTemp)
    lcd "total km:" ; sTemp

    l = lSekCountU \ 3600
    sTemp = str(l)
    call form_x_leer(9 , sTemp)
    lowerline
    lcd "hours:" ; sTemp

    b = is_key()
    waitms 100
  loop until b > 0
end sub


sub tageskm1()
local b1 as byte , b2 as byte , f as single , l as long
' ----------------
' Day 1:  ###.###*
' time : ####:##
' ----------------
  cls
  do
    'call calc_km12(lKm1_mm , sTemp)
    call calc_kmd12(lTurns1 , sTemp)
    upperline
    lcd "Day 1:  " ; sTemp ; "*"

    lowerline
    call calc_time(lSekCount1U , sTemp)
    lcd "time : " ; sTemp

    b1 = is_key()
    waitms 100
    if b1 = Taste1 then
       b2 = ClearButton(16 , 1)
       if b2 = 2 then
          'lKm1_mm = 0
          lTurns1 = 0
          lSekCount1U = 0
          EeTurns1 = lTurns1
          EeSekCount1U = lSekCount1U
       end if
    end if
  loop until b1 > Taste1
end sub


sub tageskm2()
local b1 as byte , b2 as byte , f as single
' ----------------
' Day 2:  ###.###*
' time : ####:##
' ----------------
  cls
  do
    'call calc_km12(lKm2_mm , sTemp)
    call calc_kmd12(lTurns2 , sTemp)
    upperline
    lcd "Day 2:  " ; sTemp ; "*"

    lowerline
    call calc_time(lSekCount2U , sTemp)
    lcd "time : " ; sTemp

    b1 = is_key()
    waitms 100
    if b1 = Taste1 then
       b2 = ClearButton(16 , 1)
       if b2 = 2 then
          'lKm2_mm = 0
          lTurns2 = 0
          lSekCount2U = 0
          EeTurns2 = lTurns2
          EeSekCount2U = lSekCount2U
       end if
    end if
  loop until b1 > Taste1
end sub
```

```
sub set_Umfang()
local b1 as byte , b2 as byte
' ----------------
' set perimeter:
'     #### mm
' ----------------
  cls
  cursor on noblink
  lcd "set perimeter:"
  lowerline

  sTemp = str(wUmfang)
  call form_x_0(4 , sTemp)
  lcd "    " ; sTemp ; " mm"

  b1 = 1 + 4                            '1. Zeichen auf Position 5
  do
    locate 2 , b1
    select case get_key()
    case Taste1 : b2 = aTemp(b1 - 4)    'Array overlay zu sTemp
                  incr b2
                  if b2 > 57 then
                     b2 = 48            'ASCII 9 -> 0
                  endif
                  aTemp(b1 - 4) = b2
                  locate 2 , b1
                  lcd chr(b2)

    case Taste2 : incr b1
    end select
  loop until b1 > 8

  wUmfang = val(sTemp)
  EeUmfang = wUmfang                    'ins EEPROM
  cursor off noblink
  home
end sub


sub set_totalkm()
local b1 as byte , b2 as byte , l as long , f as single
' ----------------
' set total km:
'      ######
' ----------------
  cls
  cursor on noblink
  upperline
  lcd "set total km:"

  call calc_km_total(lTurnsTotCount , sTemp)
  call form_x_0(6 , sTemp)
  sTemp2 = sTemp
  lowerline
  lcd spc(5) ; sTemp

  b1 = 1 + 5                            '1. Zeichen auf Position 6
  do
    locate 2 , b1
    select case get_key()
    case Taste1 : b2 = aTemp(b1 - 5)    'Array overlay zu sTemp
                  incr b2
                  if b2 > 57 then
                     b2 = 48            'ASCII 9 -> 0
                  endif
                  aTemp(b1 - 5) = b2
                  locate 2 , b1
                  lcd chr(b2)

    case Taste2 : incr b1
    end select
  loop until b1 > 11

  if sTemp <> sTemp2 then               'nur, wenn geändert (wg. Rundungsfehler)
    if wUmfang > 0 then
       l = val(sTemp)
       f = l * 1000000
       lTurnsTotCount = f / wUmfang
       EeTurnsTotCount = lTurnsTotCount
    end if
  end if
```

```
  cursor off noblink
  home
end sub


sub set_totalhours()
local l as long , b1 as byte , b2 as byte
' ----------------
' set total time:
'      ######
' ----------------
cls
  cursor on noblink
  upperline
  lcd "set total time:"

  l = lSekCountU \ 3600
  sTemp = str(l)
  call form_x_0(6 , sTemp)
  sTemp2 = sTemp
  lowerline
  lcd spc(5) ; sTemp

  b1 = 1 + 5                            '1. Zeichen auf Position 6
  do
    locate 2 , b1
    select case get_key()
    case Taste1 : b2 = aTemp(b1 - 5)    'Array overlay zu sTemp
                  incr b2
                  if b2 > 57 then
                     b2 = 48            'ASCII 9 -> 0
                  endif
                  aTemp(b1 - 5) = b2
                  locate 2 , b1
                  lcd chr(b2)

    case Taste2 : incr b1
    end select
  loop until b1 > 11

  if sTemp2 <> sTemp then               'nur wenn geändert (wg. Rundungsfehler)
    l = val(sTemp)
    lSekCountU = l * 3600
    EeSekCountU = lSekCountU
  end if

  cursor off noblink
  home
end sub

'------------------------------- Hilfsroutinen --------------------------------

Function ClearButton(byval x as byte , byval y as byte) as byte
local b1 as byte , b2 as byte , l as long
   b1 = get_key()                       'damit Taste abgeholt wurde
   locate y , x
   cursor on blink
   do
   loop until is_key() > 0
   if is_key() = Taste1 then
      b1 = get_key()
      waitms 100

      l = lSekCount
      b2 = 0
      do
         b2 = Taste1Direct              '1: nicht gedrückt

         b1 = lSekCount - l
         if b1 > 2 then
            b2 = 2
         end if

      loop until b2 > 0                 '1: vorher losgelassen ; 2: lange gedrückt
   end if
   cursor off noblink
   ClearButton = b2
end function


sub calc_kmh(w as word , s as string)
```

```
  local f as single
  f = wUmfang * 117.9648                '=(32768 / 1000) * 3,6
  if w > 0 then                         'div0-Error verhindern
     f = f / w
  else
    f = 0                               'wenn zu langsam
  end if
  if w = &HFFFF then
    f = 0
  end if
  s = fusing(f , "#.#")
  s = mid(s , 1 , 5)
  call form_x_leer(5 , s)
end sub


sub calc_time(seks as long , s as string)
  local l as long , s2 as string * 2
  l = seks \ 3600
  s = str(l)
  call form_x_leer(4 , s)
  l = seks \ 60
  l = l mod 60
  s2 = str(l)
  call form_x_0(2 , s2)
  s = s + ":" + s2
end sub


sub calc_km_total(turns as long , s as string)
  local l as long , f as single
  f = wUmfang * turns
  f = f / 1000000
  l = f
  s = str(l)
end sub

'(
sub calc_km12(mm as long , s as string)
  local f as single
  f = mm / 1000000
  if f > 999.999 or f < 0then           'nicht größer!
     mm = 0
     f = 0
  end if
  s = fusing(f , "#.###")
  call form_x_leer(7 , s)
end sub
')

sub calc_kmd12(Turns as long , s as string)
  local f as single
  f = wUmfang * Turns
  f = f / 1000000
  if f > 999.999 then                   'nicht größer!
     Turns = 0
     f = 0
  end if
  s = fusing(f , "#.###")
  call form_x_leer(7 , s)

end sub

sub form_x_leer(byval x as byte , s as string)
   x = x - len(s)
   if x = 0 then
      exit sub
   end if
   s = space(x) + s                     'bei x=0 -> 255x!!
   'DBG
   'Stcheck
end sub


sub form_x_0(byval x as byte , s as string)
   x = x - len(s)
   if x = 0 then
      exit sub
   end if
   s = string(x , 48) + s               'bei x=0 -> 255x!!
   'DBG
   'Stcheck
```

```
end sub

'----------------------------- Interrupt-Routinen: -----------------------------

OnACI:                                  'Comparator-Eingang; bei PowerDown!!
  cls
  lcd "Powerdown"
  lowerline
  lcd "saving... ";
  EeCaptureMin = wCaptureMin            'letzter Vmax
  EeTurnsTotCount = lTurnsToTCount      'Km total
  'EeKm1_mm = lKm1_mm                    'Km Tag1
  'EeKm2_mm = lKm2_mm                    'Km Tag2
  EeTurns1 = lTurns1
  EeTurns2 = lTurns2
  EeSekCountU = lSekCountU              'Betriebsstundenzähler
  EeSekCount1U = lSekCount1U            'Zeit Tag1
  EeSekCount2U = lSekCount2U            'Zeit Tag2
  wait 1
  lcd "bye!"
  portd.6 = 1                           'Discharge ausführen
  disable interrupts
  do : loop                             'alles stoppen
Return


OnOVF2:                                 'Timer2-Overflow-Interrupt-Routine (alle 0.5 sek)
  bUsek = timer0                        'zählt Zündimpulse
  timer0 = 0
Return


dim bOverflow as Byte                   'Zähler für Timer1-Überlauf alle 2 Sek
OnOVF1:                                 'Timer1-Overflow alle 2 sek

  incr lSekCount                        'Echtzeit-Sekundenzähler

  if bUsek > 0 then                     'nur zählen, wenn Motor läuft
     incr lSekCountU                    'Betriebsstundenzähler
     incr lSekCount1U                   'Tages-Km1
     incr lSekCount2U                   '         2
  end if

  incr bOverflow                        'merkt sich Überläufe
  if bOverflow > 2 then                 'mehrfach übergelaufen:
    wCapture = 0                        'lange kein Impuls vom Radgeber
  end if
return


OnOC1A:                                 'Timer1-Compare bei 1/2 (auch alle 2 sek)

  incr lSekCount                        'Echtzeit-Sekundenzähler

  if bUsek > 0 then                     'nur zählen, wenn Motor läuft
     incr lSekCountU                    'Betriebsstundenzähler
     incr lSekCount1U                   'Tages-Km1
     incr lSekCount2U                   '         2
  end if
return


dim wCaptureNeu as word , wCaptureAlt as word
OnICP1:                                 'Timer1-InputCapture; zählt Dauer und
                                        'Anzahl der Radumdrehungen
  wCaptureNeu = CAPTURE1
  wCapture = wCaptureNeu - wCaptureAlt  'Zeitdifferenz

  if bOverflow = 1 then                 'einmal übergelaufen; siehe OnOVF1
     if wCaptureNeu > wCaptureAlt then
        wCapture = 0                    '1x überholt
        goto ICPret2
     end if                             'ansonsten ist 1x ok!
  end if

  if bOverflow > 1 then                 'mehrfach übergelaufen:ungültig
     wCapture = 0
  end if

  ICPret:
  bOverflow = 0
```

```
  ICPret2:
  wCaptureAlt = wCaptureNeu             'Wert für nächsten Vergleich merken

  if wCapture > 0 then                  'wenn gültig
    if bfirstmax = 0 then
      bfirstmax = 1                     '1. Mal aussetzen
    else
      if wCapture < wCaptureMin then
        wCaptureMin = wCapture          'für Maximalgeschwingkeit
      end if
    end if
  end if

  if bUsek > 0 then                     'nur zählen, wenn Motor läuft
    incr lTurnsTotCount                 'Total-Km
    incr lTurns1
    incr lTurns2
    'lKm1_mm = LKm1_mm + wUmfang         'Km-Zähler 1
    'lKm2_mm = LKm2_mm + wUmfang         '          2
  end if

return

'--------------------------------------------------------------------------------

dim bKey as byte                        ' "Tastaturpuffer"

Function is_key() as Byte
  is_key = 0
  if bKey > 0 then
     is_key = bKey
  end if
End Function


Function get_key() as Byte
  get_key = bKey                        'zuletzt gedrückte Taste(n)
  bKey = 0                              'ist jetzt abgeholt!
End function


OnInt1:
  if bKey = 0 then                      'wenn leer
    waitms 25                           'damit man 2 Tasten gleichzeitig drücken kann!!
    bKey = pinb
    bKey = not bkey                     'Low-Aktiv
    bKey = bKey and &B00000110          'PB1 + PB2 für Tasten 1+2
    shift bKey , right , 1
  end if
  GIFR.INTF1 = 1                        'evtl. Tastenpreller entfernen
return
```