

## **LIXX Monitor**

Bauanleitung für einen LiPo/LiFePo Monitor mit Ausgang für ein externes LCD Display und einem externen 12V Piezo Summer für den Unterspannungsalarm.

Der Unterspannungsalarm ist für einen 3-4S LiPo/LiFePo ausgelegt. Die Alarmschwelle liegt bei 3.5 Volt für LiPo und 2.9 für LiFePo. Das externe LCD-Display zeigt die Spannung der einzelnen Zellen an. Das externe Display verwende ich für mehete Anwendungen. Das Display eignet sich auch gut als Debug Monitor.

Abweichend zum Schaltplan LIXX Monitor verwende ich ein PIC16F1705 als Microcontroller. Der Summer ist ein 12 Volt Summer mit integrierter Piezo-Ansteuerung vom Reichelt.

Artikel-Nr.: AL-28SW12-DT :: Summer, Dauerton 12V DC, f=3 kHz, Ø=28 mm

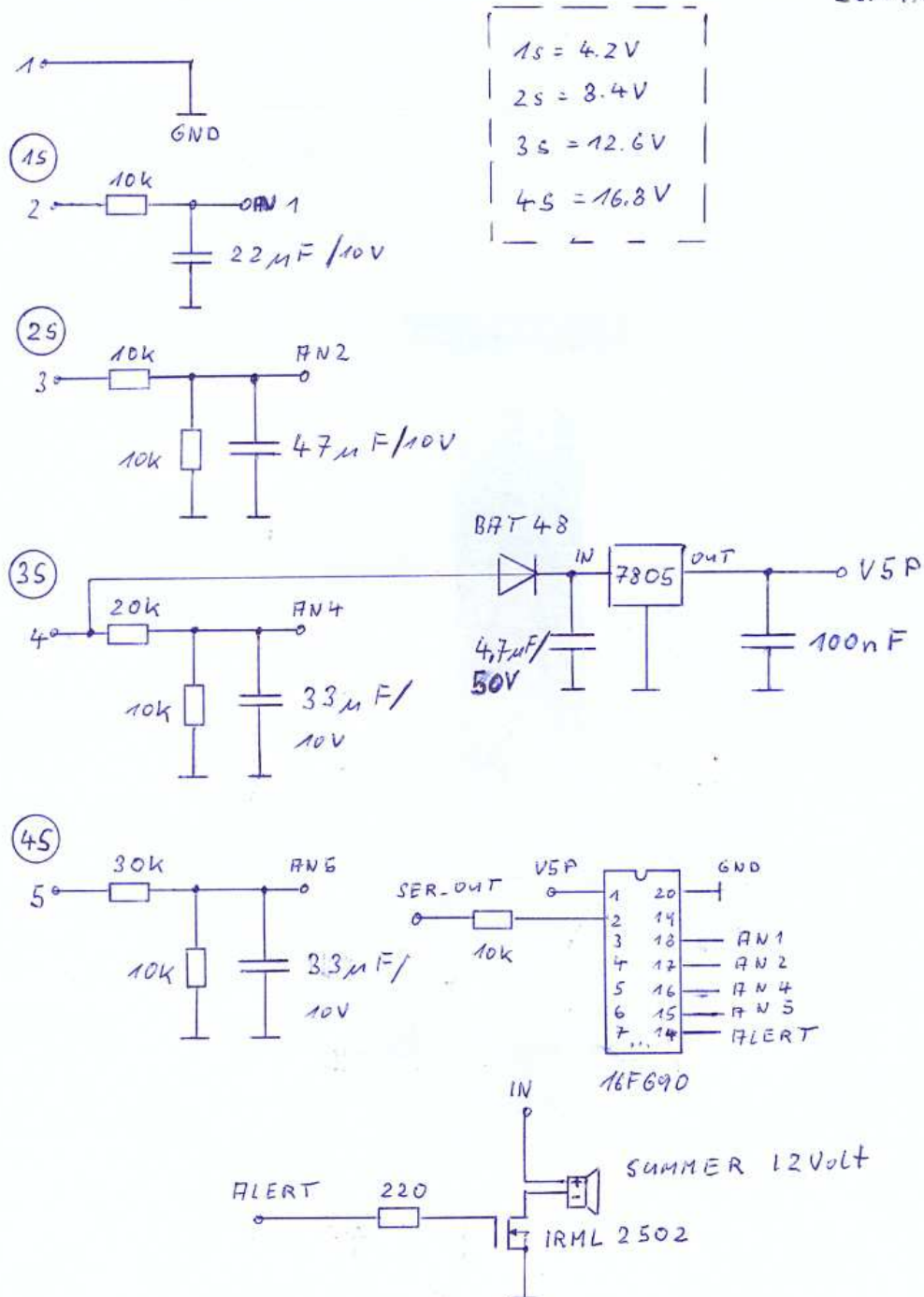
Die Software ist in GCBASIC,Version (0.95 2015-12-27) geschrieben.

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

# Schaltplan LIXX Monitor

LIXX 3-4 Zellen Monitor

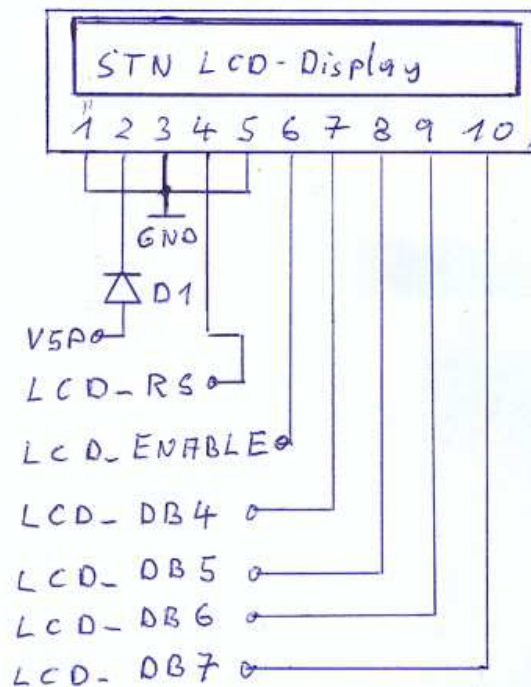
M. Schulz  
28.04.2017



## Serielles LCD

LCD Display

M. Schulz  
28.04.2017



- 1 GND
  - 2 VDD (+5V)
  - 3 V<sub>0</sub> (kontrastepg.)
  - 4 RS
  - 5 RIW (connected to GND)
  - 6 ENABLE
  - 7 D4 (Datenleitung)
  - 8 D5     - - -
  - 9 D6     - - -
  - 10 D7    - - -
- D1 BAT 48

## Program LIXX Monitor

```
' LIXX_4Z, V1.0
' 30.04.2017, IDE v95.02
'
' -----
' V+          -| 1      20 | - GND
' PORTA.5      -| 2      19 | - PORTA.0
' PORTA.4      -| 3      18 | - PORTA.1 AN1
' PORTA.3      -| 4      17 | - PORTA.2 AN2
' PORTC.5      -| 5      16 | - PORTC.0 AN4
' PORTC.4      -| 6      15 | - PORTC.1 AN5
' PORTC.3      SER_OUT -| 7      14 | - PORTC.2 ALERT
' -----

#include "software_uart_v2.h"

#chip 16F1705,8
#config OSC = INTOSC, PWRTE = ON

#define CELL1 AN1
#define CELL2 AN2
#define CELL3 AN4
#define CELL4 AN5

#define TIMES 4 'number of adc samples

#define VOLT_2P9 2900
#define VOLT_3P5 3500
#define VOLT_3P9 3900
#define VOLT_5P0 5000

#define VOLT_1P0 51 '8 bit

' Gain AN1 = 1
' Gain AN2 = 1/2
' Gain AN4 = 1/3
' Gain AN5 = 1/4

#define BAUD_RATE 2400
#define SER_OUT PORTC.3
dir SER_OUT out

dir PORTA.1 in 'AN1
dir PORTA.2 in 'AN2
dir PORTC.0 in 'AN4
dir PORTC.1 in 'AN5

#define ALERT PORTC.2
dir ALERT out

dim cell_voltage_1,cell_voltage_2,cell_voltage_3,cell_voltage_4 as word
dim cell_voltage_1_old1,cell_voltage_1_old2,cell_voltage_1_old3,cell_voltage_1_old4 as word
dim cell_voltage_2_old1,cell_voltage_2_old2,cell_voltage_2_old3,cell_voltage_2_old4 as word
dim cell_voltage_3_old1,cell_voltage_3_old2,cell_voltage_3_old3,cell_voltage_3_old4 as word
dim cell_voltage_4_old1,cell_voltage_4_old2,cell_voltage_4_old3,cell_voltage_4_old4 as word

dim ALERT_COND,alert_counter,conf_counter,case_counter,LIPO,LIXX_3S as byte
dim ALERT_THLD as word

cell_voltage_1 = 0
cell_voltage_2 = 0
cell_voltage_3 = 0
cell_voltage_4 = 0

cell_voltage_1_old1 = 0
cell_voltage_1_old2 = 0
cell_voltage_1_old3 = 0
cell_voltage_1_old4 = 0

cell_voltage_2_old1 = 0
cell_voltage_2_old2 = 0
cell_voltage_2_old3 = 0
cell_voltage_2_old4 = 0

cell_voltage_3_old1 = 0
cell_voltage_3_old2 = 0
cell_voltage_3_old3 = 0
cell_voltage_3_old4 = 0

cell_voltage_4_old1 = 0
cell_voltage_4_old2 = 0
cell_voltage_4_old3 = 0
cell_voltage_4_old4 = 0
```

```

ALERT_COND = FALSE
LIPO = FALSE 'PH
LIXX_3S = FALSE

alert_counter = 0
conf_counter = 0
case_counter = 0

ALERT_THLD = VOLT_2P9 'LiFePo = PH, default setting

'Power Up Test
set ALERT on
wait 1 s
set ALERT off
wait 2 s

LCD_Clear

cell_voltage_4 = Average(ReadAD(CELL4),ReadAD(CELL4))
LIXX_3S = (cell_voltage_4 < VOLT_1P0)

GET_ADC
LIPO = (cell_voltage_1 >= VOLT_3P9) and (cell_voltage_2 >= VOLT_3P9) and (cell_voltage_3 >= VOLT_3P9)

if LIPO then ALERT_THLD = VOLT_3P5
'----- Main -----
Do
  GET_ADC

  case_counter++

  Select Case case_counter
  Case 1
    cell_voltage_1_old1 = cell_voltage_1
    cell_voltage_2_old1 = cell_voltage_2
    cell_voltage_3_old1 = cell_voltage_3
    cell_voltage_4_old1 = cell_voltage_4

  Case 2
    cell_voltage_1_old2 = cell_voltage_1
    cell_voltage_2_old2 = cell_voltage_2
    cell_voltage_3_old2 = cell_voltage_3
    cell_voltage_4_old2 = cell_voltage_4

  Case 3
    cell_voltage_1_old3 = cell_voltage_1
    cell_voltage_2_old3 = cell_voltage_2
    cell_voltage_3_old3 = cell_voltage_3
    cell_voltage_4_old3 = cell_voltage_4

  Case 4
    cell_voltage_1_old4 = cell_voltage_1
    cell_voltage_2_old4 = cell_voltage_2
    cell_voltage_3_old4 = cell_voltage_3
    cell_voltage_4_old4 = cell_voltage_4
    case_counter = 0

  End Select

  cell_voltage_1 = (cell_voltage_1_old1 + cell_voltage_1_old2 + cell_voltage_1_old3 + cell_voltage_1_old4) / 4
  cell_voltage_2 = (cell_voltage_2_old1 + cell_voltage_2_old2 + cell_voltage_2_old3 + cell_voltage_2_old4) / 4
  cell_voltage_3 = (cell_voltage_3_old1 + cell_voltage_3_old2 + cell_voltage_3_old3 + cell_voltage_3_old4) / 4
  cell_voltage_4 = (cell_voltage_4_old1 + cell_voltage_4_old2 + cell_voltage_4_old3 + cell_voltage_4_old4) / 4

  if LIXX_3S then cell_voltage_4 = VOLT_5P0

  ALERT_COND = (cell_voltage_1 < ALERT_THLD) or (cell_voltage_2 < ALERT_THLD) or (cell_voltage_3 < ALERT_THLD) or
    (cell_voltage_4 < ALERT_THLD)

  if ALERT_COND then
    conf_counter++
    if (conf_counter >= 6) then
      conf_counter = 6
      alert_counter++
      if (alert_counter <= 2) then set ALERT on
      if (alert_counter > 2) then set ALERT off
      if (alert_counter >= 4) then alert_counter = 0
    end if
  else
    set ALERT off
    alert_counter = 0
    conf_counter = 0
  end if

```

```

LCD_Line1
  print_2p1(cell_voltage_1)
  print_2p1(cell_voltage_2)
LCD_Line2
  print_2p1(cell_voltage_3)
  if !LIXX_3S then print_2p1(cell_voltage_4)

  wait 500 ms
Loop
'----- end main -----

'----- subroutines -----

sub GET_ADC
  cell_voltage_1 = 0
  cell_voltage_2 = 0
  cell_voltage_3 = 0
  cell_voltage_4 = 0

  Repeat TIMES
    cell_voltage_1 = cell_voltage_1 + ReadAD(CELL1)
    cell_voltage_2 = cell_voltage_2 + ReadAD(CELL2)
    cell_voltage_3 = cell_voltage_3 + ReadAD(CELL3)
    cell_voltage_4 = cell_voltage_4 + ReadAD(CELL4)
  End Repeat

  cell_voltage_1 = cell_voltage_1 / TIMES
  cell_voltage_2 = cell_voltage_2 / TIMES
  cell_voltage_3 = cell_voltage_3 / TIMES
  cell_voltage_4 = cell_voltage_4 / TIMES

  cell_voltage_1 = (cell_voltage_1 * 202) / 200 'calibration
  cell_voltage_2 = (cell_voltage_2 * 202) / 200 'calibration
  cell_voltage_3 = (cell_voltage_3 * 202) / 200 'calibration
  cell_voltage_4 = (cell_voltage_4 * 203) / 200 'calibration

  cell_voltage_2 = 2 * cell_voltage_2 - cell_voltage_1
  cell_voltage_3 = (3 * cell_voltage_3 - cell_voltage_2) - cell_voltage_1
  cell_voltage_4 = ((4 * cell_voltage_4 - cell_voltage_3) - cell_voltage_2) - cell_voltage_1

  cell_voltage_1 = (cell_voltage_1 * 39) / 2
  cell_voltage_2 = (cell_voltage_2 * 39) / 2
  cell_voltage_3 = (cell_voltage_3 * 39) / 2
  cell_voltage_4 = (cell_voltage_4 * 39) / 2
end sub

```

## Program serielles Display

```

' Serial LCD Display Gateway for HD44780 LCD Display Controller V1.0
' 25.07.2016, compiled with v95
'
' -----
' V+          -| 1      14 | - GND
' PORTA.5     -| 2      13 | - PORTA.0  LCD_RS
' PORTA.4     -| 3      12 | - PORTA.1  LCD_Enable
' PORTA.3     -| 4      11 | - PORTA.2  LCD_DB4
' PORTC.5 SerIn -| 5      10 | - PORTC.0  LCD_DB5
' PORTC.4     -| 6       9 | - PORTC.1  LCD_DB6
' PORTC.3 LCD_DB7 -| 7      8 | - PORTC.2
' -----
'
#chip 16F688,8
#config OSC=INTRC_OSC_NOCLKOUT, PWRT = ON

'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

'SerIn
#define SerInPort PORTC.5
dir SerInPort In

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

#define LCD_RS      PORTA.0
#define LCD_Enable  PORTA.1
#define LCD_DB4     PORTA.2
#define LCD_DB5     PORTC.0
#define LCD_DB6     PORTC.1
#define LCD_DB7     PORTC.3

dim display_ctrl as bit

'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

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

```

## UART Library

```
' Software UART LIB with non inverted software UART and some print routines
' M. Schulz
```

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

```
#script
  if BAUD_RATE_4Mhz = 2400 then oneBitdelay = 101
  if BAUD_RATE = 2400 then oneBitdelay = 204
  if BAUD_RATE = 9600 then oneBitdelay = 48
#endscript
```

```
'-----
'***** AsyncTx *****
' * Asynchronous serial transmit byte routine *
' * 1 start bit + 8 data bits + 1 stop bit *
' * non 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(in dout as byte)
  set SER_OUT off ' startbit
  for bitCount = 1 to 9
    uart_delay
    Carry = 1 ' incl. stopbit
    rotate dout right
    if (Carry = 1) then set SER_OUT on
    if (Carry = 0) then set SER_OUT off
  next
  uart_delay
end sub
```

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

```
sub uart_delay ; Loop needs oneBitdelay us
  dim tmp as byte

  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(PrintData(SysPrintTemp))
  next
end sub 'SerPrint
```

```
'-----
sub LCD_Clear ' Clear Display
  uart(254)
  uart(1)
  wait 2 ms
end sub 'LCD_Clear
```

```
'-----
sub LCD_Line1 ' Display Line 1, pos 1
  uart(254)
  uart(128)
end sub 'LCD_Line1
```

```
'-----
sub LCD_Line2 ' Display Line 2, pos 1
  uart(254)
  uart(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(32) 'print blank
            else
                uart(buffer + 48)
            end if
        end if ' (lcd_count < 5)

        div = div / 10

        if (lcd_count = 2) then uart(46) 'print dot
    next
end sub 'print_2p2
'-----
'Print unsigned integer as float, max 65536 = 65.5
sub print_2pl(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(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_2pl
'-----
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(32) 'print blank
        else
            uart(buffer + 48)
        end if
        div = div / 10
    next
    uart(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(32) 'print blank
    else
        uart(buffer + 48)
    end if

    div = div / 10
next
    uart(32) 'print blank
end sub 'print_byte
'-----
sub print_int3(In value as word) 'max. 999 (3 count's)
    dim buffer,div as byte
    div = 100

    for lcd_count = 1 to 3
        buffer = value / div
        value = value % div
        uart(buffer + 48)
        div = div / 10
    next
end sub 'print_int3

```