MIKROFORTH - FRTH – Changed to text v3

1 - ADC_MAX7219_Display.frth

```
( Spannung messen mit ADC3 und Doppel-7-Segment-Anzeige mit MAX_7219; )
( Stellt Spannung als Dezimalzahl - z. B. 3,7 - dar )
( Für das Komma {Punkt} bei der linken Stelle muss 128 zum Code adiert werden )
( Jedes Kommando besteht aus einem Befehls-Code und einem Parameter. )
( Vor dem Kommando muss CS auf Low gesetzt werden, danach auf High! )

: CS_low   2  0  outPortB ;
: CS_high  2  1  outPortB ;
: Kommando  swap  CS_low  5  waitus  spi_send  drop  spi_send  drop  CS_high 5 waitus ;
( Stack beim Wort Kommando: operation paramter -- )

: normal_operation  12  1   Kommando ;
: alle_stellen      11  1   Kommando ;
: helligkeit        10  15  Kommando ;
: decode_mode       9   15  Kommando ;
: nullsetzen        1   0   Kommando  2  0  Kommando ;
: display_init  normal_operation  alle_stellen  helligkeit  decode_mode nullsetzen ;

( Wegen Belastung durch Display: VRef ca. 4,5 V )
( Skalierung des Messwerts 255 -> 4,5: Division durch 56 )
: zehnerstelle  56  /  drop  128  +  1  swap  Kommando ;
: einerstelle   56  /  6  /  drop  2  swap  Kommando  drop ;

: CS_SPI_display_init  CS_high  81  spi_master_init  display_init ;
```

```
: >display            dup  zehnerstelle  einerstelle ;

: messung   adc3  >display  100  waitms ;

: main      initadc3  CS_ SPI_display_init  begin  messung  0  until ;
```

2 - adc3_to_com.frth

```
( adc3-Test: Spannungswert wird an LCD ausgegeben, Poti L:GND, M:A3, R:+5V )

: messung  adc3  >com  100  waitms ;
: main     initadc3  initcom  begin  Messung  0  until ;
```

3 - adc3_to_lcd.frth

```
( adc3-Test: Spannungswert wird als Dezimalzahl an LCD ausgegeben )
( Poti L:GND, M:A3, R:+5V; in Wirklichkeit ist die Spannung etwas kleiner. )

: wert_out  50  /  swap  48  +  lcd_out  44  lcd_out  5  /  drop  48  + lcd_out  86  lcd_out ;
: messung   adc3  wert_out  100  waitms  lcd_clear ;
: main      initadc3  lcd_init  lcd_clear  begin  Messung  0  until ;
```

4 - alle_zeichen_auf_LCD.frth

```
(Alle Zeichen auf Pollin-LCD ausgeben )

: Anzeige_Code_Zeichen  dup  lcd_number  58  lcd_out  32  lcd_out lcd_out  250  waitms  lcd_clear ;
: main   lcd_init  255  0  do  I  Anzeige_Code_Zeichen  loop  end ;
```

5 - ampel.frth

```
( Ampel: rote, gelbe und grüne LED an virtuellem Port E.2, E.1 bzw. E.0 )
: initPortE    7  ddrE ;
: warte        3  wait ;
: rotesLicht   2 ;
: gelbesLicht  1 ;
: grünesLicht  0 ;
: an           1  outportE ;
: aus          0  outportE ;
: rotphase     rotesLicht   an  warte  rotesLicht   aus ;
: grünphase    grünesLicht  an  warte  grünesLicht  aus ;
: gelbphase    gelbesLicht  an  warte  gelbesLicht  aus ;
: ampelzyklus  rotphase  grünphase  gelbphase ;
: main         initPortE  ampelzyklus  ampelzyklus end ;
```

6 - com_test.frth

```
( empfängt Zahl von COM, multipliziert sie mit 2 und gibt das Ergebnis über COM aus )

: main   initcom  begin  com>  2  *  >com  0  until ;
```

7 - dot_test.frth

```
( dot-Test; LED-ARRAY an virtuellem Port E; Zahl vom Terminal an LED-Array ausgeben )

: main   initcom  begin  com>  .  0  until ;
```

8 - dotdot_test.frth

```
( dotdot - Test: Zahlen in Endlos-Schleife an terminal ausgeben )

: main   0  begin  1  +  dup  ..  100  waitms  0  until ;
```

9 - einzel_servo.frth

```
( Test 3 für servo : Servo dreht sich in 11 Schritten nach links, dann nach rechts usw. )
: links   10  0  do  I  10  *  100  +  servo  250  waitms  loop ;
: rechts  10  0  do  I  10  *  200  swap  -  servo  250  waitms  loop ;
: main    begin  links  rechts  0  until ;
```

10 - eprom_schreiben_und_lesen.frth

```
( eprom schreiben und lesen )

( eprom-test: Zahlen 10, 20, 50  in Zelle 1, 2, 5 abspeichern )
: schreiben  10  1  >eprom  20  2  >eprom  50  5  >eprom ;

( 10 Werte aus dem EEPROM über COM an Terminal senden )
: eprom2com  eprom>  >com  100  waitms ;
: holen      initCom  10  1  do  I  eprom2com  loop ;

: main       schreiben  10  wait  holen ;
```

11 - i2c_pcf8591_LDR2com.frth

```
( i2c mit PCF8591-Y0027, Analog-Input LDR mit knr 3, Ausgabe über COM )

: i2c_init   i2cstop  i2cstart  144  i2cwrite  drop  3  i2cwrite  drop ;
: init_messen  i2cstop  i2cstart  145  i2cwrite  drop ;
: messen       0  i2cread  >com  100  waitms ;
: main i2c_init  init_messen initCom  begin  messen  0  until  i2cstop ;
```

12 - i2c_pcf8591_LED_dimmen.frth

```
( i2c mit PCF8591-Y0027, LED an Analog-Output mit knr 64; dimmen )
: i2c_init   i2cstop  i2cstart ;
: PCF_AnalogOut_init  144  i2cwrite  drop  64  i2cwrite  drop ;
: dimmen  255  0  do  I  i2cwrite  drop  5  waitms  loop ;
: main  i2c_init  PCF_AnalogOut_init  begin  dimmen  0  until  i2cstop ;
```

13 - i2c_suche_gerade_adresse_2.frth

```
( suche i2c-Adresse )

( gibt die Schreib-Adresse aller angeschlossenen I2C-Geräte an )


: anzeige  swap  >com ;

: main  initcom  127  0  do  i2cstop  i2cstart  I  2  *  dup  i2cwrite  1  =  dup  skipif  anzeige  not  skipif
drop  i2cstop  loop ;
```

14 - INT0_Interrupt.frth

(Beeper zwischen D.6 und Masse)

(Beeper_An_Aus erzeugt eine Fortwährenden Ton von 500 Hz)

(Taster Ta0: zwischen D.2 und Masse; Taster Ta1: zwischen D.3 und Masse)

(Ist Ta1 gedrückt, wird die Test-LED an Port B.5 durch den)

(INT0-Interrupt {ausgelöst durch Drücken von Ta0} eingeschaltet;)

(andernfalls wird die Test-LED ausgeschaltet.)

(Die Unterbrechung durch durch den Interrupt kann man hören!)

```
: int0   pushreg  5  Ta1?  Not  OutPortB  popreg  reti ;
: BeeperAnAus  6  1  outportD  wait1ms  6  0  outPortD  wait1ms ;
: Init_B5_D6   5  1  DDBitB  6  1  DDBitD ;
: main   Init_B5_D6  0  initInt0  begin  BeeperAnAus  0  until ;
```

15 - konstanten.frth

```
( test: constant )
constant merke1
constant merke2
: merken    123  merke1  >eprom  246  merke2  >eprom ;
: anzeigen  merke1  eprom>  merke2  eprom> ;
: main      merken  anzeigen  .  . ;
```

16 - lcd_source_und_test.frth

```
( LCD-Test: gibt "ABC Z" in der 1. Zeile und "#" in der 2. Zeile rechts )
( auf Pollin-Display aus, blinkt )
: lcd_RS     6  swap  outPortE ;
: lcd_E      4  swap  outPortE ;
: lcd_Clock  10  waitus  1  lcd_E  10  waitus  0  lcd_E  10  waitus ;
: lcd_out    16  /  swap  .  1  lcd_RS  lcd_Clock  .  1  lcd_RS  lcd_Clock  0  lcd_RS ;
: lcd_code   .  lcd_clock ;
: lcd_clear  0  lcd_code  1  lcd_code  2  waitms ;
: lcd_cursor 64  *  +  128  +  16  /  swap  lcd_code  lcd_code  1  waitms ;
: lcd_digit   48  +  lcd_out ;
: lcd_number  100  /  swap  lcd_digit  10  /  swap  lcd_digit  lcd_digit  ;
: lcd_init1   3  lcd_code  5  waitms  3  lcd_code  wait1ms  3  lcd_code  wait1ms ;
: lcd_init2   2  lcd_code  wait1ms  2  lcd_code  8  lcd_code  wait1ms  0  lcd_code  12  lcd_code  wait1ms  0
lcd_code  1  lcd_code  wait1ms  0  lcd_code  6  lcd_code  wait1ms ;
: lcd_init  lcd_init1  lcd_init2 ;
: textausgabe  65  lcd_out  66  lcd_out  67  lcd_out  32  lcd_out  90  lcd_out  7  1  lcd_cursor  35  lcd_out ;
: main   lcd_init  begin  textausgabe  2  wait  lcd_clear  1  wait  0  until  end ;
```

17 - lcd_Zahlen.frth

```
( LCD-Test: Zahlen von 1 bis 100 ausgeben, dann CLS und Text "Ende" )
: Ende_Text      69  lcd_out  78  lcd_out  68  lcd_out  69  lcd_out ;
: main lcd_init  100  1  do  I  lcd_number  100  waitms  lcd_clear   loop  1  wait  Ende_Text  end ;
```

18 - LM75A_to_com.frth

```
( Temperaturmessung mit dem LM75A ; LM75A erwartet 2 Read-Vorgänge )
: messung   i2cstop  i2cstart  157  i2cwrite  1  i2cread  0  i2cread  drop ;
: main      initcom  begin  messung  >com  200  waitms  0  until ;
```

19 - Nano_LED.frth

```
( test nano mit atmega328: LED auf Nano-Board für 10 s einschalten )
: main   32  ddrb  32  portb  10  wait  0  portb ;
```

20 - PortE_DDRE.frth

```
( Test für virtuelles Port E )

: main   255  DDRE  255  0  do  I  PortE  10  waitms  loop ;
```

21 - PortE_DDRE_2.frth

```
( Test für virtuelles Port E )

: main   255  PortE  255  0  do  I  DDRE  100  waitms  loop ;
```

22 - PWM.frth

```
( teste PWM: Periodendauer 20 ms, Pulsbreite 1,5 ms )
: main   156  64  11  184  FastPWM ;
```

23 - rekursionstest.frth

```
( Rekursiv-Test: Zahlen von 1 bis 32 auf Port E anzeigen, zuerst vorwärts, dann rückwärts )
(         L  H  addresse {kann sich veraendern wenn programm veraendert wird})
: starten   0  177  >r  >r ;
: rektest   dup  dup  .  255  waitms  1  +  dup  32  equal  skipIf  starten  .  255  waitms ;
: main      1  rektest . ;
```

24 - servo_grundwoerter.frth

```
( Definition von Servo-Wörtern; für 7=6+1 Servos; bei ServoNächsteNr berücksichtigen! )

( Array mit 28 Bytes für 7*4 Servo-Daten plus 1 Byte für die Nr. des aktuellen Servos )


( Servo0: portNr=2; Aktiv=1; Pulsweite=60 )
( Servo1: portNr=3; Aktiv=1; Pulsweite=60 )
( Servo2: portNr=4; Aktiv=1; Pulsweite=60 )
( Servo3: portNr=5; Aktiv=1; Pulsweite=60 )
( Servo4: portNr=6; Aktiv=1; Pulsweite=60 )
( Servo5: portNr=7; Aktiv=1; Pulsweite=60 )
( Servo6: portNr=0; Aktiv=0; Pulsweite=256*256-1200 + 60*6 )
( zunächst direkt Presetwerte statt Pulswerte eingeben: )

: ServoArrayInit   29  initarray   184  252  0  0   195  255  1  7   195  255  1  6   195  255  1  5   195  255
1  4   195  255  1  3   195  255  1  2   0   28  0  do  I  >array  loop ;

( kürzer? )


: ZeigeArray   28  0  do  I  array>  >com  10  waitms  loop ;


: ServoSetzeAktuelleNr   0  >array ;

: ServoHoleAktuelleNr    0  array> ;

: ServoNächsteNr         0  array>  1  +  7  /  swap  drop  0  >array ;
```

```
: ServoHolePortNr         4*  1  +  array> ;
: ServoHoleAktiv          4*  2  +  array> ;

: ServoHolePresetwert     4*  3  +  dup  array>  swap  1  +  array> ;
: ServoHolePresetwertH    4*  3  +  array> ;
: ServoHolePresetwertL    4*  4  +  array> ;
: ServoSetzePresetwertH   4*  3  +  >array ;
: ServoSetzePresetwertL   4*  4  +  >array ;
: ServoPuls               swap  ServoHolePortNr  swap  outPortD ;

: Pulsweite2Preset        255  255  swap16  sub16 ;
: Preset2Pulsweite        255  255  swap16  sub16 ;

( Restzeitbetrachtung für 6 + 1 Servos, jetzt mit Schleife; SPW=ServoPulsweitenwert )
: SPW                     ServoHolePresetwert  Preset2Pulsweite ;
: ServoPulsweitenSumme    0  SPW  5  1  do  I  SPW  add16  loop ;
: ServoRestPresetwert     255  255  4  175  sub16  ServoPulsweitenSumme  add16 ;
: ServoSetzeDummy         ServoRestPresetwert  6  ServoSetzePresetwertL  6  ServoSetzePresetwertH ;

: ServoStart       0  ServoSetzeAktuelleNr  0  1  ServoPuls  4  0  ServoHolePresetwert  initT1ovf ;
```

```
: ServoPulsAus+    ServoHoleAktuelleNr  0   ServoPuls  ServoNächsteNr ;

: ServoPulsAn      ServoHoleAktuelleNr  dup ServoHoleAktiv  ServoPuls ;

: ServoPulsPreset  ServoHoleAktuelleNr  ServoHolePresetwert  SetTimer1 ;

: T1ovf            pushreg  ServoPulsAus+  ServoPulsAn  ServoPulsPreset  popreg  reti ;


: SSP              swap  255  swap  -  swap  ServoSetzePresetwertL  ServoSetzeDummy ;

( ServoSetzePulsweite; pulsweitenwert index -- ; PresetH bleibt! )


: ServoProgramm1   begin  60  0  SSP  2  wait  90  0  SSP  2  wait  120  0  SSP  2  wait  90  1  SSP  3  wait
120  1  SSP  2  wait  0  until ;

: ServoProgramm2   begin  12  6  do  I  10  *  0  SSP  1  wait  loop  0  until ;

: ServoProgramm3   60  0  SSP  70  1  SSP   80  2  SSP   90  3  SSP  100  4  SSP   120  5  SSP  begin  0  until
;


: main             252  ddrd  ServoArrayInit  ServoStart  ServoProgramm3  ;

( 252 ddrd zu ServoStart schlagen? )
```

25 - SPI_Max_7219_Display_dimmen.frth

```
( Test SPI-Master mit Display_MAX_7219 )
( Jedes Kommando besteht aus einem Operationscode und einem Parameter. )
( Vor dem Kommando muss SS auf Low gesetzt werden, danach auf High! )

: SS_low      2  0  outPortB ;
: SS_high     2  1  outPortB ;
: Kommando    swap  SS_low  5  waitus  spi_send  drop  spi_send  drop  ss_high  5  waitus ;
: normal_operation   12  1   Kommando ;
: alle_stellen       11  1   Kommando ;
: helligkeit         10  15  Kommando ;
: decode_mode        9   15  Kommando ;
: nullsetzen         1   0   Kommando  2  0  Kommando ;
: display_init       normal_operation  alle_stellen  helligkeit  decode_mode  nullsetzen ;
: zehnerstelle       10  /  drop  1  swap  Kommando ;
: einerstelle        10  /  2  swap  Kommando  drop ;
: alles_init         SS_high  81  spi_master_init  display_init ;
: >display           dup  zehnerstelle  einerstelle ;
: main               alles_init  65  >display  15  0  do  15  I  -  10  swap  Kommando  250  waitms  loop ;
```

26 - SPI_Max_7219_Display_zaehlen

```
( Test SPI-Master mit Display_MAX_7219 )
( Jedes Kommando besteht aus einem Befehlscode und einem Parameter. )
( Vor dem Kommando muss SS auf Low gesetzt werden, danach auf High! )

: CS_low     2 0 outPortB ;
: CS_high    2 1 outPortB ;
: Kommando   swap CS_low 5 waitus spi_send drop spi_send drop CS_high 5 waitus ;
: normal_operation   12  1   Kommando ;
: alle_stellen       11  1   Kommando ;
: helligkeit         10  15  Kommando ;
: decode_mode        9   15  Kommando ;
: nullsetzen         1   0   Kommando  2  0  Kommando ;
: display_init normal_operation   alle_stellen  helligkeit  decode_mode  nullsetzen ;
: zehnerstelle       10  /  drop  1  swap  Kommando ;
: einerstelle        10  /  2  swap  Kommando  drop ;
: alles_init         CS_high  81  spi_master_init  display_init ;
: >display           dup  zehnerstelle  einerstelle ;
: main          alles_init  99  0  do  I  >display  250  waitms  loop ;
```

27 - sram_test.frth

```
( sram-test: 0, 1, ... auf Stapel legen )
( dann Inhalt von [$100], [$101] ,... über COM ausgeben )

: stapeln     40  0  do  I  loop ;
: main        10  wait  initcom  stapeln  40  0  do  1  I  sram>  >com  wait1ms  loop ;
```

28 - stackcount.frth

```
( test zu stackcount: Kompletten Stapel über COM anzeigen, ohne den Stapel abzubauen )

: stack2com   initcom  stackcount  1  -  0  do  1  I  sram>  >com  250  waitms  loop ;
: main        11  22  33  44  55  66  77  88  99  stack2com ;
```

29 - stackcount2.frth

```
( test zu stackcount: Zuerst die Zahlen 0 bis 49 auf den Stack legen )
( dann den kompletten Stapel über COM anzeigen, ohne den Stapel abzubauen )
( am Ende den Stapel über COM ausgeben )

: stack2com          initcom  stackcount  1  -  0  do  1  I  sram>  >com  50  waitms  loop ;
: zahlen_auf_stack   49  0  do  I  loop ;
: stack_ausgeben     stackcount  1  -  0  do  >com  50  waitms  loop ;
: main               zahlen_auf_stack  stack2com  stack_ausgeben ;
```

30 - test_setioreg.frth

```
( LED an PortD.5 einschalten; SRAM-Nr von PortD ist $2B=43 )

: _Portd    $2B  swap  setIOreg ;

: main      32  ddrd  begin  32  _Portd  1  wait  0  _Portd  1  wait  0  until ;
```

31 - textsuche2.frth

```
( textsuche mit Array; sobald HALLO gefunden, geht LED an B.5 an und Suchen wird beendet )
: anzahl      5 ;
: >suchtext   anzahl  +  >array ;
: suchtext>   anzahl  +  array> ;
: suchtexteingabe  72  65  76  76  79   anzahl  1  -  0  do  I  >suchtext  loop ;

( Schieberegister sreg )
: >sreg    >array ;
: sreg>    array> ;
: schiebe  anzahl  2  -  0  do  anzahl  2  -  I  -  sreg>  anzahl  1  -  I  -  >sreg  loop  0  >sreg ;

( vergleicheZeichen [Index -- 1/0 ] )
: vergleicheZeichen dup suchtext> swap sreg> = ;
( VergleicheRegister [ -- a ] a = Anzahl der Übereinstimmungen )
: vergleicheRegister   0  anzahl  1  -  0  do  I  vergleicheZeichen  +  loop ;

: initAll              anzahl  2  *  initArray  suchtexteingabe  32  ddrb  initCom ;
: main                 initAll  begin  com>  schiebe  vergleicheRegister  anzahl  =  until  32  portB ;
```

32 - timer0_ovf_interrupt.frth

```
( Timerinterrupt-Test : Unterteiler 1024, 156 Counts bis zum Overflow, )
( Toggle-Periode also 2* 1024 * 156 / Taktfrequenz s )
( Bei 16 MHz ist die Toggle-Periode ungefähr 2 * 160 000 / 16 000 000 s = 1/50 s )
( Messung der Periode des Signals an D.2 mit Oszi bestätigt dies. )

: t0Ovf    pushreg  99  settimer0  toggleD  popreg  reti ;
: main     252  DDRD  85  PortD  5  99  initt0ovf ;
```