
```

/*****
Firmware für das Autorange DVM TN20140606, Rev. 3 mit ATmega8 @8MHz, Hardware:
LED-Display, gemeinsame Anode Digit1 (rechts) an PC1 - Digit 4 (links) an PC4
Segmente A-G & DP an PD0-PD7 Autorange mit Shunt-R an PB0, ADC-Eingang an PC0
Fuesbits interner RC-Takt: Low=E4, High=D9, Lock=FF, Compiler: AVRStudio 4.17

```

```

V1.00: 02/09/2015 - Initiale Version aus V1.30/Rev.1 (c) Thomas Neveling      566 Byte
V1.10: 06/09/2015 - Bei Messbereichswechsel keine ungültigen Werte anzeigen
                   Anzeige "OFL" (overflow) bei Spannungen >= 102,3V      636 Byte
V1.20: 08/10/2015 - 64 Messungen & Mittelwertbildung, Software Kalibrierung 696 Byte
*****/

```

```

#include <avr/io.h>                // bindet auch den Prozessortyp gemäß Makefile ein.
#include <avr/interrupt.h>

// Zur Kalibrierung der Messbereiche: 0-10V (LO), 10-100V (HI) Wert kleiner => Anzeige höher
#define OFFS_LO 0                  // Maximalwert +1 damit Autorange-Schwelle erreicht wird
#define OFFS_HI 0                  // hier alternativ Trimpoti nutzen

const char SEGMENTE[] = {192,249,164,176,153,146,130,248,128,144}; // Segmentmuster: 0..9

unsigned int display= 1234;        // Auszugebender Messwert: 10 von 16 Bit werden genutzt
unsigned char dot = 3;             // Position des Dezimalpunkts 1 = rechts..4 = links
unsigned char digicnt= 1;          // Zählt die Digits des Displays bei der Ausgabe durch
unsigned char ADCslot= 0;          // Messintervall alle 409,6ms (~ 2 Messungen / Sekunde)
unsigned char SkipUpd= 0;          // steuert Display-Update nach Bereichsumschaltung
unsigned char WERT_A = 192;        // 0 Segmentmuster der einzelnen Ziffern: Einer-Stelle
unsigned char WERT_B = 164;        // 2
unsigned char WERT_C = 121;        // 1. (dot=3) initialisiert mit
unsigned char WERT_D = 198;        // C Firmwareversion 1000er-Stelle

ISR (TIMER0_OVF_vect)             // 2,048ms Timer: Pro Aufruf entweder 1 Digit
{
    PORTD = 0b11111111;           // ansteuern oder den AD-Konverter auslesen
    PORTC = PORTC & 0b11100001;   // alle Segmente aus
    // D1 - D4 aus

    PORTC = 1<<digicnt;          // gewähltes Digit einschalten, im 5. Zyklus ADC auslesen
    switch (digicnt)              // entsprechendes Segmentmuster dazu
    {
        case 1:                   // Digit 1 (rechts) ansteuern
            PORTD = WERT_A;
            break;
        case 2:                   // Digit 2 ansteuern
            PORTD = WERT_B;
            break;
        case 3:                   // Digit 3 ansteuern
            PORTD = WERT_C;
            break;
        case 4:                   // Digit 4 (links) ansteuern
            if (display > 999) PORTD = WERT_D; // Leading Zero Blanking
            break;
        case 5:                   // den ADC auslesen: Das Display bleibt dunkel PC5 = frei
            digicnt = 0;
            ADCslot++;
            if (ADCslot == 40)     // ADC alle 409,6ms auslesen (40 x 5 x 2,048ms)
            {
                SkipUpd = 0;       // kein Messbereichswechsel
                ADCslot = 0;       // Messfolge-Zähler neu initialisieren
                display = 0;       // 64 Messungen aufaddieren
                for (uint8 t i=0; i<64; i++) display = display + ADCW;
                if (dot == 3) display = display / (64 + OFFS_LO); // Mittelwert & skalieren
                else display = display / (64 + OFFS_HI); // je Inc/Dec +/- 0,16V

                if (dot == 3)       // Autorange: Kleiner => großer Messbereich ?
                {
                    if (display > 1005)
                    {
                        dot = 2;
                        DDRB = 0xFF; // PB0 = Ausgang nach Masse (siehe Init in Main)
                        SkipUpd = 1; // Bereichswechsel: Messwert verwerfen
                    }
                }

                if (dot == 2)       // Autorange: Großer => kleiner Messbereich ?
                {
                    if (display < 98)
                    {
                        dot = 3;

```

```
        DDRB = 0xFE; // PB0 = Eingang (offen)
        SkipUpd = 1; // Bereichswechsel: Messwert verwerfen
    }
    if (display == 1023)
    {
        // Überlauf im oberen Bereich: Spannung >= 102,3V
        SkipUpd = 1; // damit kein Zahlenwert übernommen wird
        WERT_A = 0xC7; // "L"
        WERT_B = 0x8E; // "F" OFL (overflow) anzeigen
        WERT_C = 0xC0; // "0"
        WERT_D = 0xFF; // dunkel
    }
}

if (SkipUpd == 0)
{
    // kein Bereichswechsel => Display aktualisieren
    WERT_A = SEGMENTE[(display % 10)];
    WERT_B = SEGMENTE[(display % 100 / 10)];
    if (dot == 2) WERT_B = WERT_B & 0b01111111; // Dezimalpunkt Bit 7 dazu
    WERT_C = SEGMENTE[(display % 1000 / 100)];
    if (dot == 3) WERT_C = WERT_C & 0b01111111; // Dezimalpunkt Bit 7 dazu
    WERT_D = SEGMENTE[(display % 10000 / 1000)];
}
} // if ADCslot
break;
} // switch
digitcnt++;
} // isr

int main (void) // Hauptprogramm mit Initialisierungen
{
    // Data Direction Register setzen
    DDRB = 0xFE; // PB0 zunächst als Eingang ohne Pullup-R, der Rest ist unbenutzt.
    PORTB = 0; // bleibt immer = 0 => je nach DDRB Eingang/offen oder Ausgang nach Masse
    DDRC = 0x1E; // PC0 = Messeingang, PC1-4 = LED Anoden Ausgänge
    DDRD = 0xFF; // PDx = Segmentausgänge Kathoden A-G, DP

    // Den 10 Bit ADC initialisieren
    ADMUX = 0b11000000; // Kanal0, Ausgabe rechtsbündig, 2,56V Referenz intern, C an AREF Pin
    // ADC Enable, Start Conversion, free running, Prescaler = 64 => 8MHz / 64 = 125kHz ADC-Takt
    ADCSRA = ((1 << ADEN) | (1 << ADSC) | (1 << ADFR) | (1 << ADPS2) | (1 << ADPS1));

    /* Timer0 liefert den Zeittakt zum Multiplexen des Displays und zum Auslesen des ADC
    Prescaler = 1/64 => 64/8MHz * 256 = 2,048ms ohne zusätzlichen Timer-Preload */
    TCCR0 = 0x03; // (1 << CS01) | (1 << CS00) => 1/64 Prescaler
    TIMSK = 0x01; // (1 << TOIE0) => Overflow-Interrupt Timer0 freigeben

    sei(); // globale Interrupt-Freigabe

    while(1); // leere Main-Schleife
}
```