;-------------------------------------------------------------------------  
; Titel :  
;-------------------------------------------------------------------------  
; Funktion :  
; Schaltung :  
;-------------------------------------------------------------------------  
; Prozessor : ATmega8  
; Takt : 3686400 Hz  
; Sprache : Assembler (GNU)  
; Datum : 29.2.2012  
; Version : 1.0  
; Autor :  
; Programmer:  
; Port :  
;-------------------------------------------------------------------------  
; created by myAVR-CodeWizard  
;-------------------------------------------------------------------------  
;  
.include    "avr.h"  
  
begin:      rjmp    main                ; 1 POWER ON RESET  
            reti                        ; 2 Int0-Interrupt  
            reti                        ; 3 Int1-Interrupt  
            reti                        ; 4 TC2 Compare Match  
            reti                        ; 5 TC2 Overflow  
            reti                        ; 6 TC1 Capture  
            rjmp    onTimer1Cmp         ; 7 Timer1 Interrupt bei Vergleichswert  
            reti                        ; 8 TC1 Compare Match B  
            reti                        ; 9 TC1 Overflow  
            reti                        ;10 TC0 Overflow  
            reti                        ;11 SPI, STC Serial Transfer Complete  
            reti                        ;12 UART Rx Complete  
            reti                        ;13 UART Data Register Empty  
            reti                        ;14 UART Tx Complete  
            reti                        ;15 ADC Conversion Complete  
            reti                        ;16 EEPROM Ready  
            reti                        ;17 Analog Comparator  
            reti                        ;18 TWI (IC) Serial Interface  
            reti                        ;19 Store Program Memory Ready  
  
;------------------------------------------------------------------------  
; Initialisierungen  
;------------------------------------------------------------------------  
main:  
;--- Stack Initialisierung ---  
            ldi     r16,hi8(RAMEND)  
            out     SPH,r16  
            ldi     r16,lo8(RAMEND)  
            out     SPL,r16  
; Ports initialisieren  
            cbi     DDRD,4              ; PORTD4 auf Eingang mit PullUp  
            sbi     PORTD,4  
;--- Timer 1 initialisieren ---  
            ldi r16,0b00000100 ; Teiler 1/1  
            ori r16,0b00001000 ; Modus: Zählen bis Vergleichswert (WGM12=1)  
            out TCCR1B,r16 ; Teiler+Modus schreiben  
            ldi r16,lo8(18064) ; Vergleichswert speichern  
            ldi r17,hi8(18064)  
            out OCR1AH,r17  
            out OCR1AL,r16  
            in r16,TIMSK ; Interrupt bei Vergleichswert  
            ori r16,0b00010000  
            out TIMSK,r16  
  
  
  
;--- Interrupts erlauben ---  
;----------LCD initialisieren-----------------  
            rcall   lcdInit  
            rcall   lcdClear  
            sei  
;------------------------------------------------------------------------  
; Hauptprogramm-Schleife  
;------------------------------------------------------------------------  
mainloop:   wdr  
; ...  
            rjmp    mainloop            ; Sprung zum Beginn der Hauptprogrammschleife  
;--------------------------------------------------------------------  
; onTimer1Cmp - Timer1 Interrupt bei Vergleichswert  
; aktuelle Einstellung:  10 Hz  100 ms  
; PE: ...  
; PA: ...  
;--------------------------------------------------------------------  
onTimer1Cmp:  
            rcall   lcdInit  
            mov     r16,r0  
            inc     r16  
            mov     r0,r16  
            rcall   lcdZahl  
            reti                        ;Rücksprung  
;--------------------------------------------------------------------  
ausgabe:  
            rcall   lcdInit  
            ldi     r16,  
            inc     r16  
            rcall   lcdZahl  
;-------------------------------------------------------------------  
; myWait\_ms - Warte-Routine für x-Millisekunden  
; ein Millisekundenzyklus dauert 1,052 ms  
; PE: r16 = Anzahl der zu wartenden Milisekunden  
; PA: r16 = 0  
;--------------------------------------------------------------------  
myWait\_ms:  
            push    r16  
            ldi     r16,1  
myWait\_ms\_3:  
            push    r16  
            ldi     r16,5  
myWait\_ms\_2:  
            push    r16  
            ldi     r16,255  
myWait\_ms\_1:  
            dec     r16  
            brne    myWait\_ms\_1  
            pop     r16  
            dec     r16  
            brne    myWait\_ms\_2  
            pop     r16  
            dec     r16  
            brne    myWait\_ms\_3  
            pop     r16  
            dec     r16  
            brne    myWait\_ms  
            ret  
  
;-----------------------------------------------------------------------------  
lcdOut:     mov     r17,r16  
            swap    r17  
            andi    r16,0b11110000  
            or      r16,r18  
            andi    r17,0b11110000  
            or      r17,r18  
            out     PORTD,r16  
            rcall   lcdEnable  
            out     PORTD,r17  
            rcall   lcdEnable  
            rcall   lcdWait  
            ret  
;------------------------------------------------------------------------  
;--------------------------------------------------------------------  
; Warte-Routine für 5 ms  
; die Routine wartet inclusive Aufruf 5,031 ms  
;--------------------------------------------------------------------  
lcdWait:  
            ldi     r16,5  
            rcall   myWait\_ms  
            ret  
;------------------------------------------------------------------------  
;-----------------------------------------------------------------------------  
lcdCmd:     ldi     r18,0b0000000       ; RS = lo  
            rjmp    lcdOut  
            ret  
;------------------------------------------------------------------------  
;-----------------------------------------------------------------------------  
lcdData:    ldi     r18,0b0000100       ; RS = hi  
            rjmp    lcdOut  
            ret  
;------------------------------------------------------------------------  
;-----------------------------------------------------------------------------  
lcdOn:      ldi     r16,0x0E  
            rcall   lcdCmd  
            rcall   lcdWait  
            ret  
;------------------------------------------------------------------------  
;-----------------------------------------------------------------------------  
lcdClear:   ldi     r16,0b00000001      ; Display löschen  
            rcall   lcdCmd  
            rcall   lcdWait  
            ret  
;------------------------------------------------------------------------  
;-----------------------------------------------------------------------------  
lcdOff:     ldi     r16,0b00001000  
            rcall   lcdCmd  
            rcall   lcdWait  
            ret  
;------------------------------------------------------------------------  
;-----------------------------------------------------------------------------  
lcdEnable:  sbi     PORTD,3             ; Enable high  
            nop                         ; kurz warten  
            nop  
            nop  
            cbi     PORTD,3             ; Enable wieder low  
            ret  
;------------------------------------------------------------------------  
;-----------------------------------------------------------------------------  
lcdInit:    sbi     DDRD,2              ; LCD RS = OUT  
            sbi     DDRD,3              ; LCD E  = OUT  
            sbi     DDRD,4              ; LCD D4 = OUT  
            sbi     DDRD,5              ; LCD D5 = OUT  
            sbi     DDRD,6              ; LCD D6 = OUT  
            sbi     DDRD,7              ; LCD D7 = OUT  
            cbi     PORTD,2             ; LDC RS = Low  
  
; warte bis PowerUp  
            ldi     r18,20  
powerUp:    rcall   lcdWait  
            dec     r18  
            brne    powerUp             ; Power-Up Wartezyklus min 30 ms  
  
; sende Resetsequenz kompatibel zu HD44780 Industriestandard  
            ldi     r16,0b00110000      ; Reset-Sequenz Teil 1  
            out     PORTD,r16  
            rcall   lcdEnable           ; Enable-Impuls  
            rcall   lcdWait  
            ldi     r16,0b00110000      ; Reset-Sequenz Teil 2  
            out     PORTD,r16  
            rcall   lcdEnable           ; Enable-Impuls  
            rcall   lcdWait  
            ldi     r18,100             ; Wartezyklus bei RESET LCD min 100 s  
resetLCD:  
            nop  
            nop  
            nop  
            dec     r18  
            brne    resetLCD  
            ldi     r16,0b00110000      ; Reset-Sequenz Teil 3  
            out     PORTD,r16  
            rcall   lcdEnable           ; Enable-Impuls  
            rcall   lcdWait  
  
; sende init 1  
            ldi     r16,0b00100000      ; 4 Bit Modus aktivieren  
            out     PORTD,r16  
            rcall   lcdEnable           ; Enable-Impuls  
            rcall   lcdWait  
            ldi     r16,0b00101000  
            rcall   lcdCmd              ; Function Set 4 Bit, 2 Zeilen, 5x7  
            rcall   lcdOff  
            rcall   lcdClear  
            ldi     r16,0x06  
            rcall   lcdCmd              ; Entry Mode Set, increase+shifted  
            rcall   lcdOn  
            ret  
;------------------------------------------------------------------------  
;------------------------------------------------------------------------  
; Sendet einen mit 0x00 abgeschlossenen String von Flash an UART  
; PE: Z=StartAdresse  
  
;------------------------------------------------------------------------  
lcdZahl:  
            mov     r1,r16              ; Zwischenspeicher und Einer-Stelle  
            clr     r3  
            clr     r2  
hundert:  
            ldi     r16,100             ; solange die 100 in die Restzahl (r1) passt  
            cp      r1,r16  
            brlo    zehner  
            sub     r1,r16              ; 100 abziehen und  
            inc     r3                  ; Hunderter-Stelle erhöhen  
            rjmp    hundert  
zehner:  
            ldi     r16,10              ; wie bei den Hundertern  
            cp      r1,r16  
            brlo    einer  
            sub     r1,r16  
            inc     r2  
            rjmp    zehner  
einer:      ;       der\_Rest\_sind\_die\_Einer  
            ldi     r21,0x30            ; 30 hex für Zahl  
            mov     r16,r3  
            add     r16,r21             ; hunderter + 30 hex  
            rcall   lcdData  
            mov     r16,r2  
            add     r16,r21             ; zehner + 30 hex  
            rcall   lcdData  
            mov     r16,r1  
            add     r16,r21             ; einer + 30 he  
            rcall   lcdData  
            ret  
;------------------------------------------------  
;------------------------------------------------------------------------  
lcdString:  
            push    r16  
            push    r30  
            push    r31  
lcdString1:  
            lpm     r16,Z+  
            cpi     r16,0  
            breq    lcdString2  
            rcall   lcdData  
            rjmp    lcdString1  
lcdString2:  
            pop     r31  
            pop     r30  
            pop     r16  
            ret