```
;*****************************************************************
;*****************************************************************
;Skrevet av:        Erik Grindheim,  IFJF - UiB                    *
;Dato:              15. februar 2000                               *
;Revisjon;          1.12                                           *
;Språk:             MPASM (Microchip)                              *
;Processor:         PIC16F84                                       *
;Watchdog timer:    off                                            *
;Code protect:      off                                            *
;*****************************************************************
;*****************************************************************
;                                                                  *
;Programmet starter opp med å gi to LED-blink.                     *
;Deretter vil LED'en lyse eller være slokket i hht. EEPROM data.   *
;                                                                  *
;LED'en kan skrus av eller på ved å sende kommando S (s) eller C (c)  *
;inn til PIC'en. Den vil da svare ved å sende ut enten S eller C.  *
;                                                                  *
;Dersom en vil vite LED status uten å endre tilstand så kan en sende  *
;kommando X (x). Da vil PIC'en svare ved å sende ut enten S eller C.  *
;                                                                  *
;Baudrate er 9600 baud, men kan endres til alt fra 2400 til 19200 bd. *
;                                                                  *
;Hver gang en endring skjer lagres dette i EEPROM slik at ved neste   *
;Power-up vil alltid LED'en ha samme tilstand som ved Power-down.  *
;                                                                  *
;*****************************************************************



       list    p=16f84
       include "p16f84.inc"

       __config _HS_OSC & _WDT_OFF

#define TTL   ;Dersom denne linjen blir gjort gjeldende vil
              ;serie-kommunikasjonen sin polaritet reverserest
              ;slik at det må brukes MAX232 el. lignende inverterende
              ;RS-232 linje driver/receiver.
              ;
              ;       TTL   RS-232
              ;Log.0  0V    5V eller +(3~25)V
              ;Log.1  5V    0V eller -(3~25)V
              ;
```

```
;*********************************************************************
;* KONSTANTER;                                                       *
;*********************************************************************

F_CRYSTAL     equ    .4000000        ;4MHz krystall
BAUD_RATE     equ    .9600           ;9600 baud seriekommunikasjon
                                     ; (kan være fra 2400 til 19200)
F_CLOCK       equ    F_CRYSTAL / 4          ;instruksjons clock-rate

BIT_TIME            equ    ( ( F_CLOCK / BAUD_RATE ) /3 -4 )
BIT_TIME_AND_HALF   equ    ( ( F_CLOCK / BAUD_RATE ) /2 -4 )

Reset_Vector  equ    0x00
Int_Vector    equ    0x04

EEadrLED      equ    0x32         ;EEPROM adresse til å lagre
                                  ;LED-status. Hvilken som helst
                                  ;adresse fra 0x00 til 0x3F kan
                                  ;brukes. Dersom byte-verdien på
                                  ;denne adressen er 0x00 er LED-
                                  ;status "CLEARED". Ellers (ved
                                  ;ulik 0x00) er LED-status "SET"


;*********************************************************************
;* VARIABLER                                                         *
;*********************************************************************

WaitVar1      equ    0x0C   ;Disse to er kladd-variabler for bruk i
WaitVar2      equ    0x0D   ;  forsinkelses-rutinen "wait".
blinks        equ    0x0F   ;Hvor mange ggr. skal LED'en blinke?

BitDelay      equ    0x10   ;Her er variablene for Asynkron seriell
BitCounter    equ    0x11   ;  kommunikasjon...
TxReg         equ    0x12   ;
RxReg         equ    0x13   ;


;*********************************************************************
;* PIN ASSIGNMENTS                                                   *
;*********************************************************************

#define      LED_PIN      PORTA,  1
#define      _rx          PORTB,  0
#define      _tx          PORTB,  4

#define  LED_ON           bcf     LED_PIN
#define  LED_OFF          bsf     LED_PIN


;*********************************************************************
;* OPPSTART                                                          *
;*********************************************************************

       org    Reset_Vector
       goto   init
```

```
;*********************************************************************
;* INTERRUPT RUTINE                                                  *
;*********************************************************************

       org    Int_Vector

rxbaudwait
       decfsz BitDelay, F      ;Forsinkelses-loop for å gi rett
       goto   rxbaudwait       ;  varighet på bit'ene ( 9k6 => 104us )
       movlw  BIT_TIME         ;Når loop'en er ferdig resettes loop-
       movwf  BitDelay         ;  variablen til neste gang (neste bit)
       decfsz BitCounter, F    ;Etter denne instr. ser man hvor mange
       goto   RecvNextBit      ;  bits som gjenstår. Hvis null: "skip"

       movlw  'S'              ;Er nå ferdig med en hel byte
       subwf  RxReg, W         ;Sammenligner denne med 'S'
       btfsc  STATUS, Z        ;Hvis det var en 'S'
       goto   set_routine      ;  -ja, en 'S'
                               ;
       movlw  's'              ;  -nei, ikke 'S'. Fortsett testingen
       subwf  RxReg, W         ;Sammenligner denne med 's'
       btfsc  STATUS, Z        ;Hvis det var en 's'
       goto   set_routine      ;  -ja, en 's'

       movlw  'C'              ;  -nei, ikke 's'. Fortsett testingen
       subwf  RxReg, W         ;Sammenligner denne med 'C'
       btfsc  STATUS, Z        ;Hvis det var en 'C'
       goto   clear_routine    ;  -ja, en 'C'

       movlw  'c'              ;  -nei, ikke 'C'. Fortsett testingen
       subwf  RxReg, W         ;Sammenligner denne med 'c'
       btfsc  STATUS, Z        ;Hvis det var en 'c'
       goto   clear_routine    ;  -ja, en 'c'

       movlw  'X'              ;  -nei, ikke 'c'. Fortsett testingen
       subwf  RxReg, W         ;Sammenligner denne med 'X'
       btfsc  STATUS, Z        ;Hvis det var en 'X'
       goto   status_routine   ;  -ja, en 'X'

       movlw  'x'              ;  -nei, ikke 'X'. Fortsett testingen
       subwf  RxReg, W         ;Sammenligner denne med 'x'
       btfsc  STATUS, Z        ;Hvis det var en 'x'
       goto   status_routine   ;  -ja, en 'x'

character_tested
       movlw  BIT_TIME_AND_HALF
       movwf  BitDelay         ;Gjør klar til mottak av en ny byte
       movlw  .9               ;  ( StartBit + 1 byte  =  9 bits )
       movwf  BitCounter       ;

       bcf    INTCON, INTF     ;Clear eksternt interrupt flagg
       retfie                  ;Interrupt rutine er ferdig, Enable Int

RecvNextBit
       bcf    STATUS, C        ;Sletter Carry-flagget
       ifdef  TTL              ;TTL:
              btfsc _rx        ;  "Skip" hvis Logisk 0 (0V)
       else                    ;RS-232:
              btfss _rx        ;  "Skip" hvis Logisk 0 (5V)
       endif                   ;
       bsf    STATUS, C        ;Setter C dersom Log 1 nivå på RB0
       rrf    RxReg, F         ;Roterer RxReg ett hakk mot høyre
       goto   rxbaudwait       ;Forsinkelse før ny bit...
```

```
;***********************************************************************
;* HOVEDPROGRAM                                                        *
;***********************************************************************

init
       bsf    INTCON, INTE        ;Enabler eksterne interrupt fra RB0
       clrw                       ;
       movwf  PORTA               ;PORTA pinnene settes til 0

       bsf    STATUS, RP0         ;Skifter til Register Bank 1
       movwf  TRISA               ;PORTA er utganger
       movlw  B'11000001'         ;PORTB:  utg:  RB1, RB2, RB3, RB4, RB5
       movwf  TRISB               ;        inng: RB0, RB6, RB7

       ifdef  TTL                 ;Dersom TTL:
         bcf  OPTION_REG, INTEDG  ;  -int. på fallende flanke
       else                       ;ellers
         bsf  OPTION_REG, INTEDG  ;  -int. på stigende flanke
       endif                      ;
                                  ;
       bcf    STATUS, RP0         ; ..og tilbake til Register Bank 0

       movlw  BIT_TIME_AND_HALF
       movwf  BitDelay            ;Gjør klar til mottak av en byte
       movlw  .9                  ;
       movwf  BitCounter          ;

       movlw  .2               ;Gir et par blink i LED'en. Indikerer
       call   blink_LED        ;  at initialiseringen er ferdig
       call   readEEsetLED     ;Leser EEPROM'en og setter LED'en av
                               ;  eller på i hht. EEPROM'ens data.
       bsf    INTCON, GIE      ;Muliggjør interrupt (Globalt)
                               ;
end_loop                       ;
       goto   end_loop         ;Her er program-initialiseringen
                               ;  ferdig, og en venter på interrupt...
```

```
;*********************************************************************
;* SUB RUTINER                                                       *
;*********************************************************************

blink_LED
       movwf  blinks           ;husk hvor mange blink som gjenstår
a_flash                        ;her starter ett enkelt blink
       LED_ON                  ;tenn LED'en
       call wait               ;vent...
       LED_OFF                 ;slukk LED'en
       call wait               ;vent...
       decfsz blinks, F        ;dekrementer blink-variablen
       goto a_flash            ;hvis den fortsatt er større enn 0...
       return                  ;...og hvis den nå ble null: Returner.


readEEsetLED                   ;Leser EEPROM'en, setter LED'en i hht.
                               ;  byteverdi og returnerer /m denne i W
       movlw  EEadrLED         ;EEPROM adressen som holder LED-status
       movwf  EEADR            ;  blir valgt i EEADR-registeret.
       bsf    STATUS, RP0      ;Register Bank 1 velges
       bsf    EECON1, RD       ;Leser en byte fra EEPROM'en
       bcf    STATUS, RP0      ;Går tilbake til Register Bank 0
       movf   EEDATA, W        ;EEPROM-data flyttes til W
       btfss  STATUS, Z        ;Var denne lik 0x00...
       goto   _set             ;  -nei. Set LED'en
       LED_OFF                 ;  -ja. Slukk LED'en
       retlw  'C'              ;Returner med LED-status ('C') i W
_set                           ;
       LED_ON                  ;Tenner LED
       retlw  'S'              ;Returner med LED-status ('S') i W


set_routine                      ;Skriver 0xFF til EEPROM og
                                 ;  kaller 'readEEsetLED' og 'transmit'
       movlw  0xFF               ;Legger inn data som skal skrives til
       call   writeEEPROM        ;  EEPROM, og kaller opp sub-rutinen.
       call   readEEsetLED       ;Denne rutinen returnerer status i W
       call   transmit           ;Her sendes W ut serielt
       goto   character_tested   ;Returnerer til int.rutinens avsluttn.


clear_routine                    ;Skriver 0x00 til EEPROM og
                                 ;kaller 'readEEsetLED' og 'transmit'
       movlw  0x00               ;Legger inn data som skal skrives til
       call   writeEEPROM        ;  EEPROM, og kaller opp sub-rutinen.
       call   readEEsetLED       ;Denne rutinen returnerer status i W
       call   transmit           ;Her sendes W ut serielt
       goto   character_tested   ;Returnerer til int.rutinens avsluttn.


status_routine                   ;Kaller 'readEEsetLED' og 'transmit'
       call   readEEsetLED       ;Denne rutinen returnerer status i W
       call   transmit           ;Her sendes W ut serielt
       goto   character_tested   ;Returnerer til int.rutinens avsluttn.


transmit                       ;Rutine som sender ut W serielt
       movwf  TxReg            ;Lagrer byte'n som skal sendes
       ifdef  TTL              ;
              bcf    _tx       ;Lager startbit - Logisk 0  (0V)
       else                    ;
              bsf    _tx       ;Lager startbit - Logisk 0  (5V)
       endif                   ;
       movlw  BIT_TIME         ;
       movwf  BitDelay         ;
       movlw  .11              ;StartBit + DataBits + StopBits = 11
       movwf  BitCounter       ;Tallet 11 legges inn i bit-telleren
```

```
txbaudwait
       decfsz  BitDelay, F     ;Forsinkelse med 1 bit-tids varighet
       goto    txbaudwait      ;
       movlw   BIT_TIME        ;Reset bit-forsinkelses tid variabel
       movwf   BitDelay        ;
       decfsz  BitCounter, F   ;Er alle bits sendt?
       goto    SendNextBit     ;  -nei, send en bit til
       return                  ;  -ja, returner
SendNextBit
        bsf     STATUS, C      ;Setter C for å gi Logisk 1 StopBits
        rrf     TxReg, F       ;Roterer ny "tx-bit" inn til C
        btfss   STATUS, C      ;Var dette bit'et HØYT ?
        goto    Setlo          ;  -nei, det var LAVT
Sethi
       ifdef  TTL              ;  -ja, bit'et var HØYT
              bsf    _tx       ;Setter tx-pin til logisk 1  (5V)
       else                    ;
              bcf    _tx       ;Setter tx-pin til logisk 1  (0V)
       endif                   ;
       goto    txbaudwait      ;Vent på at neste bit kan sendes...
Setlo
       ifdef  TTL              ;...bit'et som skal sendes er LAVT
              bcf    _tx       ;Setter tx-pin til logisk 0  (0V)
       else                    ;
              bsf     _tx      ;Setter tx-pin til logisk 0  (5V)
       endif                   ;
       goto    txbaudwait      ;Vent på at neste bit kan sendes...


writeEEPROM                    ;Rutine som skriver W inn i EEPROM'en
       movwf  EEDATA           ;Legger inn data for EEPROM-skriving
       movlw  EEadrLED         ;EEPROM adressen som holder LED-status
       movwf  EEADR            ;  blir valgt i EEADR-registeret.
       bsf    STATUS, RP0      ;Register Bank 1 velges
       bsf    EECON1, WREN     ;Muliggjør skriving i EEPROM'en
       movlw  0x55             ;Initierer "write"
       movwf  EECON2           ;  ...ved å skrive 0x55
       movlw  0xAA             ;  og deretter skrive 0xAA
       movwf  EECON2           ;  ...til EECON2
       bsf    EECON1, WR       ;Dette starter selve write-syklusen
       bcf    EECON1, WREN     ;Disable EEPROM skriving
waitEEcomplete                 ;
       btfss  EECON1, WR       ;Sjekk om EEPROM'en fortsatt er opptatt
       return                  ;  med å skrive. Hvis ikke: Returner.
       goto waitEEcomplete     ;Opptatt, vent litt til...


wait   movlw  0xFF             ;Forsinkelses-rutine
       movwf  WaitVar2         ;  bruker 2 variabler:
loop2  movlw  0xFF             ;  'WaitVar1' og 'WaitVar2'
       movwf  WaitVar1         ;
loop1  decfsz WaitVar1, F      ;
       goto   loop1            ;
       decfsz WaitVar2, F      ;
       goto   loop2            ;
       return                  ;


;*********************************************************************
;* SLUTT PÅ PROGRAM                                                  *
;*********************************************************************

       end                     ;Her slutter programmet!

;*********************************************************************
;*********************************************************************
```