```
{
================================================================================


                L E D  -  L I N K    P R O G R A M

            -- INTERFACE BETWEEN nRF0433 AND PIC16F84 --


FIL         : Første test-versjon
            :
BESKRIVELSE : Program som mottar søppel fra nRF0433 på Serial Port 1.
            : Når det mottas en en sekvens som starter med "UiB: " vil
            : det neste tegnet (tegnet etter <space>) bli sendt ut til
            : PIC16F84 på Serial Port 0.
            : Hvis programmet mottar en byte fra PIC16F84 vil det skru
            : på senderen til nRF0433 og sende ut en streng bestående
            : av 12 tegn: "abcUiB: $abc". Dollar-tegnet ($) er
            : erstattet av den bokstaven som kom fra PIC16F84.
            :
            :
            : Hardware: "Flashlite NEC V25+" fra JK Microsystems
            :
            :
PROSJEKT    : Dette er en enhet som inngår i fagprøven på ELAB våren 2000.
COPYRIGHT   : Universitetet i Bergen, Inst. for den faste jords fysikk (2000)
SPRÅK       : Turbo Pascal 7.0
TAB         : 2
            :
REV.        : REV   DATO        AV    BESKRIVELSE
            :  ------------------------------------------------------------
            : 1.0   15.02.2000  EG    Basert på "GPS timing-unit" program.
            :



================================================================================
}

{$N-} {No numeric co-processor}


{



  Hardware resources:
  -------------------------------------------------------------


   PORT 1:
   -------
   P1.0 -
   P1.1 -
   P1.2 -
   P1.3 -
   P1.4 -
   P1.5 - LED_Blink utgang. Positiv puls når headeren "UiB: " mottas.
   P1.6 - Toggle ved 240 byte RX. (Utfrekvens blir ca 1 Hz ved 4800 bd.)
   P1.7 - TxEnable. Utgang til å styre TX/RX på nRF0433.  HI=TX , LO=RX
```

```
  ---------------------------------------------------------------



  Software resources:
  ---------------------------------------------------------------

   Trenger ingen andre filer enn programfilen til dette
   programmet for å kjøre på Flashlite NEC V25+

  ---------------------------------------------------------------

}

Program LED_link;

uses
  dos;


const

  SEG   = $F000;           { Segment address for CPU special registers.  }
  P0    = $FF00;           { Port 0 register.                            }
  PM0   = $FF01;           { Mode register for port 0.                   }
  PMC0  = $FF02;           { Mode control register for port 0.           }
  P1    = $FF08;           { Port 1 register.                            }
  PM1   = $FF09;           { Mode register for port 1.                   }
  PMC1  = $FF0A;           { Mode control register for port 1.           }
  P2    = $FF10;           { Port 2 register.                            }
  PM2   = $FF11;           { Mode register for port 2.                   }
  PMC2  = $FF12;           { Mode control register for port 2.           }
  INTM  = $FF40;           { External interrupt mode register.           }
  EXIC0 = $FF4C;           { External int. request control register 0.   }
  EXIC1 = $FF4D;           { External int. request control register 1.   }
  EXIC2 = $FF4E;           { External int. request control register 2.   }

  TMIC0 = $FF9C;           { Timer Unit Int. Request control register 0. }
  TMIC1 = $FF9D;           { Timer Unit Int. Request control register 1. }
  TMIC2 = $FF9E;           { Timer Unit Int. Request control register 2. }
  TMC0  = $FF90;           { Timer Control Register 0.                   }
  TMC1  = $FF91;           { Timer Control Register 1.                   }
  MD1   = $FF8A;           { Timer REgister for Timer1.                  }
  TBIC  = $FFEC;           { Time Base Interrupt req. Control reg.(1-163)}

  RxB0  = $FF60;           { Receive Buffer register 0.                  }
  SCS0  = $FF6B;           { Serial Communications Status register 0.    }

  TxB1  = $FF72;           { Transmit Buffer register 1.                 }
  RxB1  = $FF70;           { Receive Buffer register 1.                  }
  SCM1  = $FF78;           { Serial Communications Mode register 1.      }
  SCC1  = $FF79;           { Serial Communications Control register 1.   }
  SCS1  = $FF7B;           { Serial Communications Status register 1.    }
  SEIC1 = $FF7C;           { Serial Error Interrupt req. Control reg. 1  }
  SRIC1 = $FF7D;           { Serial Receive Interrupt rq. Control reg. 1 }
  STIC1 = $FF7E;           { Serial Transmit Interrupt rq Control reg. 1 }
  BRG1  = $FF7A;           { Baud Rate Generator register 1              }
```

```
var

  CommandReceived : boolean;  { Flagg som sier at en kommando er mottat via nRF0433 }
  ReplyReceived   : boolean;  { Flagg som sier at Byte er mottatt fra PIC16F84      }
  ByteCounter     : byte;     { Dummy-variabel for å gi ut pulser ved noise-RX      }
  TriggerLevel    : byte;     { Hvor mange "Header-bytes" er mottatt?               }
  PIC_Trick,
  nRF_Trick       :           { To luringer brukt i serielt mottak                  }
    RECORD
      CASE Boolean OF
        True  : (Ch : Char);
        False : (n  : byte);
    END;



  (*************************************************************)
  procedure VarInitialize;
  (*************************************************************)
  { Initialize variables and set program default values. }
  begin
    TriggerLevel := 0;
  end;



  (*************************************************************)
  procedure InitPorts;
  (*************************************************************)
  begin

    (* Initialize port 0.               *)
    (* Alle pinner er ubrukte utganger  *)
    mem[SEG:PMC0] := 0;
    mem[SEG:PM0]  := 0;

    (* Initialize port 1. Bit 5 ~ 7 er  *)
    (* utganger, resten er innganger.   *)
    mem[SEG:PMC1] := $00;
    mem[SEG:PM1]  := $1F;
    mem[SEG:P1]   := $00;   { Utgangene er lave (RX, ikke TX)         }


    (*
    mem[SEG:SCM1] :=  $C9;  { Tx & Rx Enable, 8N1, asynchronous mode  }
    *)
    mem[SEG:SCM1] :=  $CD;  { Tx & Rx Enable, 8N2, asynchronous mode  }
    mem[SEG:SCC1] :=    3;  { 3 --> 4800 baud   2--> 9600 baud        }
    mem[SEG:BRG1] :=  130;  { pg.1-177~8~9 / pg.7  (BaudRate Gen.)    }

  end;



  (*************************************************************)
  procedure TX_ON;
  (*************************************************************)
```

```
begin
  mem[SEG:P1] := $80 OR mem[SEG:P1];  { Utgangen P1.7 settes Høy. }
end;



(*****************************************************************)
procedure TX_OFF;
(*****************************************************************)
begin
  mem[SEG:P1] := $7F AND mem[SEG:P1]; { Utgangen P1.7 settes Lav. }
end;



(*****************************************************************)
procedure LED_Blink;
(*****************************************************************)
var
  Dummy : byte;
begin
  mem[SEG:P1] := $20 OR mem[SEG:P1];
  for Dummy := 0 to 255 do;
  mem[SEG:P1] := $DF AND mem[SEG:P1];
end;



(*****************************************************************)
procedure CheckSerial0;   { fra PIC16F84 }
(*****************************************************************)
begin
  if ( (mem[SEG:SCS0] AND $10) = $10 ) then begin   {   Ny byte?  }
    PIC_Trick.n := mem[SEG:RxB0];     { Leser mottatt reply-tegn  }
    ReplyReceived := True             { Setter flagget            }
  end;
end;



(*****************************************************************)
procedure CheckSerial1;   { fra nRF0433 }
(*****************************************************************)
begin
  if ( (mem[SEG:SCS1] AND $10) = $10 ) then begin   {   Ny byte?  }

    Inc(ByteCounter);
    If ByteCounter >239 then begin
      ByteCounter := 0;
      mem[SEG:P1] := $40 XOR mem[SEG:P1];       { Toggle J5-pin13 }
    end;

    nRF_Trick.n := mem[SEG:RxB1];     { Leser mottatt radio-tegn  }

    CASE TriggerLevel of
      5 : begin
            TriggerLevel := 0;
            LED_Blink;
            If (nRF_Trick.Ch = 'S') OR (nRF_Trick.Ch = 's') OR
               (nRF_Trick.Ch = 'C') OR (nRF_Trick.Ch = 'c') OR
```

```
               (nRF_Trick.Ch = 'X') OR (nRF_Trick.Ch = 'x')
             then CommandReceived := True;
           end;
       4 : begin
             if nRF_Trick.Ch = ' ' then TriggerLevel := 5 else TriggerLevel := 0;
           end;
       3 : begin
             if nRF_Trick.Ch = ':' then TriggerLevel := 4 else TriggerLevel := 0;
           end;
       2 : begin
             if nRF_Trick.Ch = 'B' then TriggerLevel := 3 else TriggerLevel := 0;
           end;
       1 : begin
             if nRF_Trick.Ch = 'i' then TriggerLevel := 2 else TriggerLevel := 0;
           end;
       0 : begin
             if nRF_Trick.Ch = 'U' then TriggerLevel := 1;
           end;
     END;  {CASE}

  end;
end;



(*****************************************************************)
procedure ForwardReply;
(*****************************************************************)
var
  Dummy : byte;
  Str   : string;

begin
  TX_ON;

  ReplyReceived := False;

  Str := 'abcUiB: $abc';
  Str[9] := PIC_Trick.Ch;

  for Dummy := 0 to 255 do;
  for Dummy := 0 to 255 do;
  for Dummy := 0 to 255 do;    {Tidligere linjer + dette gir ca 4ms delay}

  for Dummy := 1 to 12 do begin
    mem[SEG:TxB1] := Ord(Str[Dummy]);
    while ( (mem[SEG:SCS1] AND $20) = $00 ) do;
                            { loop som sjekker om TxB er tomt...          }
  end;

  while ( (mem[SEG:SCS1] AND $60) <> $60 ) do;
                            { sjekker at alle bits er gått ut "på luften" }

  for Dummy := 0 to 255 do; { delay }

  TX_OFF;

end;
```

```
(*****************************************************************************)
(*****************************************************************************)
(*****************************************************************************)
(******************                                     **********************)
(******************      H O V E D   P R O G R A M      **********************)
(******************                                     **********************)
(*****************************************************************************)
(*****************************************************************************)
(*****************************************************************************)

begin

  InitPorts;              { Initiate ports P0, P1 and P2  }

  VarInitialize;          { Initialize global variables   }

  { Programet vil gå i en uendelig    }
  { loop helt til en tar av strømmen. }

  While True Do begin
  {hoved-loop}
    CheckSerial0;                           { PIC16F84                  }
    CheckSerial1;                           { nRF0433                   }
    If ReplyReceived then ForwardReply;     { Fra PIC16F84 til nRF0433  }
    If CommandReceived then begin
      Write( UpCase(nRF_Trick.Ch) );        { Skriver kommando til PIC  }
      CommandReceived := False;             { ...og re-setter flagget.  }
    end;
  {hoved-loop end}
  end;

end.
```