

## Using Linet node in data mode

---

### Overview

Each Linet node incorporate fixed I/O-functions for general control network applications. The nodes have documented HW interface and the network protocol is embedded in the system.

The node itself is not, and cannot be, programmed. To build distributed intelligence, a microcontroller is connected to the node. The node can supply operating power and clock signal to the microcontroller. The node has pins for synchronous serial data input, output and clock. The application developer just reads or writes to these pins so virtually none network specific knowledge is required.

### Linnet network principles

In the Linet network, both data and operating power required by the nodes are transmitted in the same single twisted pair cable. The carrier signal frequency is 20kHz and its waveform is close to sinusoidal. Each sequence is used to transmit one bit of information, and the sequence waveform is altered according to the bit to be transmitted. During transmission of the bit as a voltage signal, one bit is recieved as a current signal. It is therefore a full-time full duplex system, and there are no separate modes for reading or writing.

Each Linet node can send and recieve synchronous serial data at a constant rate. The basic data rate is 80 bits/seconds. This is true when the Linet network 'frame size', i.e. the maximum number of nodes in the network, is set to 200 nodes (which is the default). If the frame size is decreased to 100 or 50 nodes, the data rate increases to 160 bits/sec. or 320 bits/sec.

Linnet network uses time division protocol. The data rate is therefore always constant and doesn't depend on the number of nodes in the network, or the network load. If the application is a single data mode node, its data rate is 80 bits/sec.; if it consists of 200 continuously transmitting and recieving data nodes, the rate is still 80 bits/sec. per each node.

### Data modes

The available data modes are the I/O-node, data 8-bit, data 12-bit, data 16-bit and data exchange mode. The I/O-node is to read and set switch position and not discussed here.

In data exchange mode, two nodes form a data pair, where the data output bits from the former node are transmitted to be the data input bits to the latter, and vice versa. The controller can't manipulate the data transmitted between the nodes in data exchange group.

The data  $n$ -bit modes are to transmit information between the Linet controller and the node, where  $n$  is the wordlength of the data to be transmitted. The data can be set using controllers 'OUT' command, and read using 'IN' command.

### The node serial data interface

The Linet node has three data related pins, which are SDI, SDO and SCK (serial data in, serial data out and serial clock). The node generates a rising edge on the SCK pin for each data bit. From the moment of the rising edge, there is about 300 usec. to read the output data bit. If reading is based on interrupts the bit should be captured with a flip-flop.<sup>1</sup>

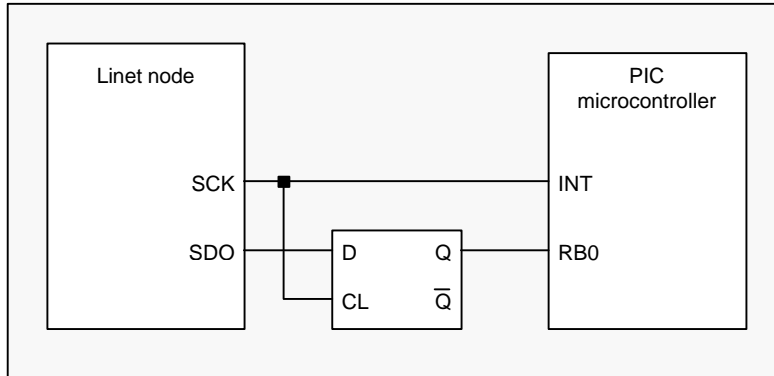


Figure 1. Capturing incoming data.

The input data bit is read at the moment of the rising edge. On interrupt-based systems there should be enough time to set the following data bit at SDI.

### The data configuration

When the controller sends and receives data, it divides the data word into nibbles (groups of 4 bits), and places a zero between the nibbles. A sequence of five consecutive '1's is used to specify the beginning of a frame. A new frame is sent immediately after the previous, even if the data to be transmitted has not changed.

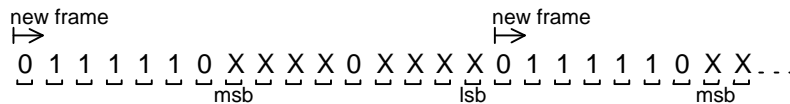


Figure 2. Data 8-bit pattern.

The 8-bit data pattern reserves 16 bits. Transmission of the 8-bit word takes 16/80 sec. = 200 msec. when frame size is set to 200 nodes.

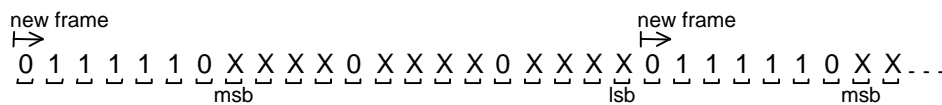


Figure 3. Data 12-bit pattern.

The 12-bit data pattern reserves 21 bits, and takes 260 msec. to transmit when frame size is set to 200 nodes. The internal A/D-converter of the fully integrated node uses this pattern.

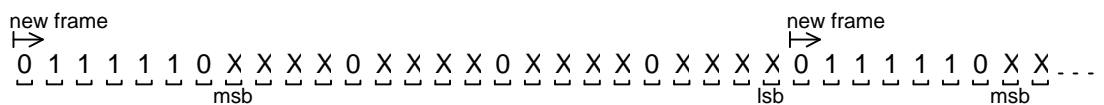


Figure 4. Data 16-bit pattern.

<sup>1</sup> The mentioned flip-flop will be included on-chip in the future node versions. Please verify the update node data sheet.

The 16-bit data pattern reserves 26 bits, and takes 330 msec. to transmit when frame size is set to 200 nodes.

### The integrated A/D-converter

The Linet node from LIN02 upwards includes internal, 12-bit A/D-converter. When used as an analog input node, such node should be configured to be data 12-bit or data exchange group.

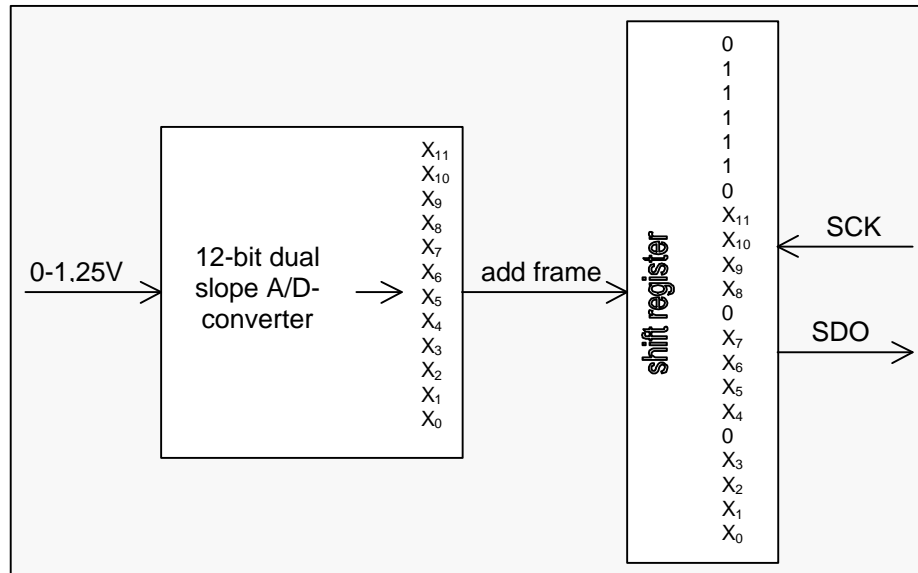


Figure 5. Internal A/D-conversion.

### Data display - application example

The attached design example is a networked data display unit. It displays the received data as an integer on a numeric 4-digit LCD display. The unit consists of a Linet node, a PIC microcontroller, an LCD display driver and the display. It can be configured to be a data 8 bit, data 12 bit or data 16 bit node, when the data to be displayed originates at the controller, or to be in a data exchange group, when the other node in this group is a node that sends data (such as the analog input node).

In this design example, the circuitry is powered by the Linet node. The PIC microcontroller is operated at low frequency - 32kHz - to reduce power consumption.

The Linet node SCK pin is connected to PIC external interrupt pin. The interrupt routine on the PIC reads the input bit from the node's SDO pin, inserts the bit to a shift register and verifies the frame syntax. The main program waits for the complete frame to be received, then makes binary-to-bcd conversion to the inputted data and outputs the bcd data to the display driver.

## APPENDIX A. THE DISPLAY UNIT SOFTWARE

```

        TITLE "Numeric display adapter for Linet"

;
; *****
;
;   Numeric display adapter for Linet
;   Version 0.1, 9.2.1999, Tauno Voipio
;
; *****

radix dec
list n=40,c=110
errorlevel -302

processor 16C711
__config 0x30

include "p16c711_.inc"

;
;   Constants
;   -----

LNBMSK set 0x0f           ;low nibble mask
HNBMSK set 0xf0           ;high nibble mask

PATRIS equ 0x18           ;port A: bits 0-2 out, 3-4 in
PBTRIS equ 0x03           ;port B: bits 0 and 1 in, others out

INIDSP equ 0xaa           ;initial display: '----'
page

;
;   Data framing patterns
;   -----

;
;   8 bit data pattern:           0111110 xxx0xxx x0111110
;   12 bit data pattern:         0111 110xxxx0 xxx0xxx x0111110
;   16 bit data pattern:         0 111110xx xx0xxxx0 xxx0xxx x0111110
;   Bytes in asmbuf:             +4   +3       +2       +1       +0

PMASK0 equ 0x087f         ;common mask 16 LSBs
PDATA0 equ 0x003e         ;common data 16 LSBs

PMASK8 equ 0x0007f        ;8 bit MSB mask
PDATA8 equ 0x0003e        ;8 bit MSB data

PMASK12 equ 0x00fe1       ;12 bit MSB mask
PDATA12 equ 0x007c0       ;12 bit MSB data

PMASK16 equ 0x1fc21       ;16 bit MSB mask
PDATA16 equ 0xf800       ;16 bit MSB data
page

;
;   Data
;   ----

cblock _FIRSTDATA

        save          ;interrupt save for working register
        saves         ;interrupt save for status register

        asmreg:5      ;assembly register
        rxbuf:2       ;receiver data buffer

        flgs          ;bit flags

        cvbbuf:2      ;conversion binary buffer
        bcdbuf:3      ;converted BCD buffer
        cntr          ;conversion counter
        crytmp        ;decimal carry temporary

endc

;
;   Bit codes

```

```

;      -----

#define NSTROBE PORTA,0      ;display strobe bit
#define INTSRV  PORTA,1      ;interrupt service flag (pin 18)
#define MATCH   PORTA,2      ;pattern match flag (pin 1)
#define INBIT   PORTB,1      ;input data bit

#define DRDY8   flgs,0       ;8 bit data ready bit
#define DRDY12  flgs,1       ;12 bit data ready bit
#define DRDY16  flgs,2       ;16 bit data ready bit
      page

;      Startup
;      -----

      org _RESETVECTOR

      bcf INTCON,GIE         ;no surprises
      call init              ;: initialize

      goto mloop            ;: enter main loop
      page

;      Interrupt handler
;      -----

      org _INTVECTOR

      bsf INTSRV             ;: flag interrupt service running
      movwf savew           ;: save W

      swapf STATUS,w        ;: get status - keep flags
      movwf saves           ;: save it

      bsf asmreg,0          ;: preset newest bit to one

      btfsc INBIT           ;: input bit an one?
      goto shift            ;: yes - cannot be pattern match

      bcf asmreg,0          ;: clear newest bit

      movfw asmreg          ;: get 1s byte
      andlw low PMASK0      ;: pick up bits to look at
      xorlw low PDATA0      ;: 1s byte match?
      bnz shift            ;: no - no match

      btfsc asmreg+1,3      ;: 2nd 1s byte match?
      goto shift            ;: no - no match

      movfw asmreg+2        ;: get middle byte
      andlw low PMASK8      ;: pick up bits to look at
      xorlw low PDATA8      ;: 8 bit data match?
      bnz not8             ;: no - proceed to test longer words
                          ;: yes - continued ...

      page

;      8 bit data match

      clrf flgs             ;: reset all flags
      bsf DRDY8             ;: flag 8 bit data available
      clrf rxbuf+1          ;: clear ms buffer
      goto save8           ;: save the data

;      8 bit data did not match - continue testing

not8:  btfss asmreg+2,0      ;: no match if bit 0 == 1
      btfsc asmreg+2,5      ;: no match if bit 5 == 1
      goto shift            ;: no match - exit test

      btfsc asmreg+2,6      ;: not 12 bit if bit 6 == 0
      btfss asmreg+2,7      ;: not 12 bit if bit 7 == 0
      goto not12           ;: no match - try 16 bit

      movfw asmreg+3        ;: get 2nd ms byte
      andlw low(PMASK12>>8) ;: get bits to look at
      xorlw low(PDATA12>>8) ;: 12 bit framing match?
      bnz not12            ;: no match - try 16 bit

```

```

;      12 bit data match

      clrfl flgs           ;; reset all flags
      bsf  DRDY12         ;; flag 12 bit data available
      clrfl rxbuf+1       ;; clear ms buffer
      goto save12         ;; save the data
      page

;      Check for 16 bit data match

not12: btfsc asmreg+4,0    ;; no match if ms bit == 1
      goto shift          ;; no match - exit test

      movfw asmreg+3      ;; get 2nd ms byte
      andlw low(PMASK16>>8) ;; pick up bits to look at
      xorlw low(PDATA16>>8) ;; 2nd ms byte match?
      bnz  shift          ;; no - no match

;      16 bit data match

      clrfl flgs           ;; reset all flags
      bsf  DRDY16         ;; flag 16 bit data available

;      Save 16 bit data

      rrf  asmreg+3,w      ;; get 1s bit of 2 ms byte
      movwf rxbuf          ;; save shifted byte

      rrf  asmreg+2,w      ;; shift the bit into middle byte
      andlw low(HNBMSK<<1) ;; clean shifted high nibble
      movwf rxbuf+1       ;; set up ms nibble of ms byte

      rrf  rxbuf,f         ;; get 2nd 1s bit of ms byte
      rrf  rxbuf+1,f       ;; shift into result
      ;; ... continued ...

      page

;      Save 12 bit data

save12: rrf  asmreg+2,w    ;; get 2nd ms nibble
      andlw LNBMSK        ;; clean up
      iorwf rxbuf+1,f     ;; set up high byte

;      Save 8 bit data

save8:  bsf  MATCH        ;; report match

      rlf  asmreg,w        ;; get ms bit of 1s byte
      rlf  asmreg+1,w      ;; build 1s nibble
      andlw LNBMSK        ;; clean up
      movwf rxbuf          ;; set up

      movfw asmreg+1      ;; get middle nibble
      andlw HNBMSK        ;; clean up
      iorwf rxbuf,f       ;; insert into low byte

      bcf  MATCH          ;; clear match flag

;      Shift the assembly register for next bit

shift:  rlf  asmreg,f       ;;      shift
      rlf  asmreg+1,f      ;;      assembly
      rlf  asmreg+2,f      ;;      register
      rlf  asmreg+3,f      ;;      left
      rlf  asmreg+4,f      ;;      a bit
      ;; ... continued ...

      page

;      Dismiss the interrupt

      swapf saves,w        ;; get saved status
      movwf STATUS        ;; restore status

      swapf savew,f        ;; prepare saved W for restore
      swapf savew,w        ;; restore saved W - keep flags

```

```

        bcf INTCON,INTF      ;; reset external interrupt flag
        bcf INTSRV          ;; clear interrupt handler flag
        retfie              ;; dismiss
        page

;      Main loop
;      -----

mloop:  btfss DRDY8          ;8 bit data ready?
        goto mlp1          ; no - proceed

        call gdata8         ;get 8 bit data
        goto dspit         ;display it

mlp1:   btfss DRDY12        ;12 bit data ready?
        goto mlp2          ; no - proceed

        call gdat12         ;get 12 bit data
        goto dspit         ;display it

mlp2:   btfss DRDY16        ;16 bit data ready?
        goto mloop         ; no - loop

        call gdat16         ;get 16 bit data

;      Display the data - could be different functions
;      for different size data

dspit:  call cnvbcd         ;convert to BCD
        call dspbcd         ;display the data
        goto mloop         ;loop
        page

;      Get 8 bit data from receive buffer
;      -----

gdata8: clrf cvbbuf+1       ;clear ms byte

        bcf INTCON,GIE      ;no surprises

        movfw rxbuf         ;; get ls byte
        movwf cvbbuf       ;; set to conversion buffer

        bcf DRDY8          ;; mark data taken
        retfie             ;; return and restore interrupts

;      Get 12 bit data from receive buffer
;      -----

gdat12: bcf INTCON,GIE      ;no surprises

        movfw rxbuf         ;; get ls byte
        movwf cvbbuf       ;; set to conversion buffer

        movfw rxbuf+1      ;; get ms byte
        movwf cvbbuf+1    ;; set to conversion buffer

        bcf DRDY12        ;; mark data taken
        retfie             ;; return and restore interrupts
        page

;      Get 16 bit data from receive buffer
;      -----

gdat16: bcf INTCON,GIE      ;no surprises

        movfw rxbuf         ;; get ls byte
        movwf cvbbuf       ;; set to conversion buffer

        movfw rxbuf+1      ;; get ms byte
        movwf cvbbuf+1    ;; set to conversion buffer

        bcf DRDY16        ;; mark data taken
        retfie             ;; return and restore interrupts
        page

```

```

;      Convert 16 bit binary to packed BCD
;      -----
cnvbcd: movlw 16          ;get bit count
        movwf cntr      ;set up counter

        clrfsz bcdbuf   ;clear
        clrfsz bcdbuf+1 ; bcd
        clrfsz bcdbuf+2 ;  buffer

cvtlp:  movlw bcdbuf    ; -> converted buffer
        movwf FSR      ;set up pointer

        call decadj     ;process decimal carries
        call decadj     ;process decimal carries

        rlf cvbbuf,f    ;shift binary
        rlf cvbbuf+1,f  ; data left into

        rlf bcdbuf,f    ;converted
        rlf bcdbuf+1,f  ; bcd
        rlf bcdbuf+2,f  ;  buffer

        decfsz cntr,f   ;bump count - done?
        goto cvtlp     ; no - loop

        return         ;return
page

;      Process decimal carries in conversion
;      -----
BCDEXC equ (16-10)>>1   ;decimal to BCD excess shifted one bit right

decadj: movfw INDF      ;get data byte
        addlw BCDEXC    ;add low nibble excess for correction
        movwf crytmp    ;save result

        btfsz crytmp,3  ;correction needed?
        movwf INDF     ; yes - do it

        movfw INDF      ;get data byte
        addlw BCDEXC<<4 ;add high nibble excess for correction
        movwf crytmp    ;save result

        btfsz crytmp,7  ;correction needed?
        movwf INDF     ; yes - do it

        incf FSR,f      ;bump pointer to next data byte
        return         ;return
page

;      Display data from BCD buffer
;      -----
dspbcd: swapf bcdbuf,w  ;get ls nibble - shift up
        andlw HNBMSK    ;clean high nibble
        iorlw 3<<2      ;insert display address
        call wrtout     ;strobe the controller

        movfw bcdbuf    ;get next nibble
        andlw HNBMSK    ;clean high nibble
        iorlw 1<<2      ;insert display address
        call wrtout     ;strobe the controller

        swapf bcdbuf+1,w ;get next nibble - shift up
        andlw HNBMSK    ;clean high nibble
        iorlw 2<<2      ;insert display address
        call wrtout     ;strobe the controller

        movfw bcdbuf+1  ;get next nibble
        andlw HNBMSK    ;clean high nibble
;      iorlw 0<<2      ;insert display address
;      call wrtout     ;strobe the controller

;      return         ;return
page

```

```
;      Send contents of W to display controller
;      -----
wrtout: movwf PORTB      ;set data to controller

        bcf NSTROBE      ;activate strobe
        bsf NSTROBE      ;terminate the strobe

        return           ;return
page

;      Initialization
;      -----
init:   clrf PORTB       ;; clear port B
        clrf PORTA       ;; clear port A
        bsf NSTROBE      ;; preset strobe passive

        bsf STATUS,RP0   ;; select bank 1

        movlw PBTRIS     ;; get port B setup
        movwf TRISB      ;; set up port B direction

        bsf OPTION_REG,NOT_RBPU ;; disable port B pull-ups

        movlw PATRIS     ;; get port A setup
        movwf TRISA      ;; set up port a direction

        bcf STATUS,RP0   ;; return to bank 0

        clrf flgs        ;; clear flags

        clrf asmreg      ;; clear
        clrf asmreg+1    ;;  assembly
        clrf asmreg+2    ;;  register
        clrf asmreg+3    ;;  to
        clrf asmreg+4    ;;  zeroes
        ;;  ... continued ...

page

        movlw INIDSP     ;; get initial display pattern
        movwf bcdbuf     ;; set up
        movwf bcdbuf+1   ;;  initial display pattern

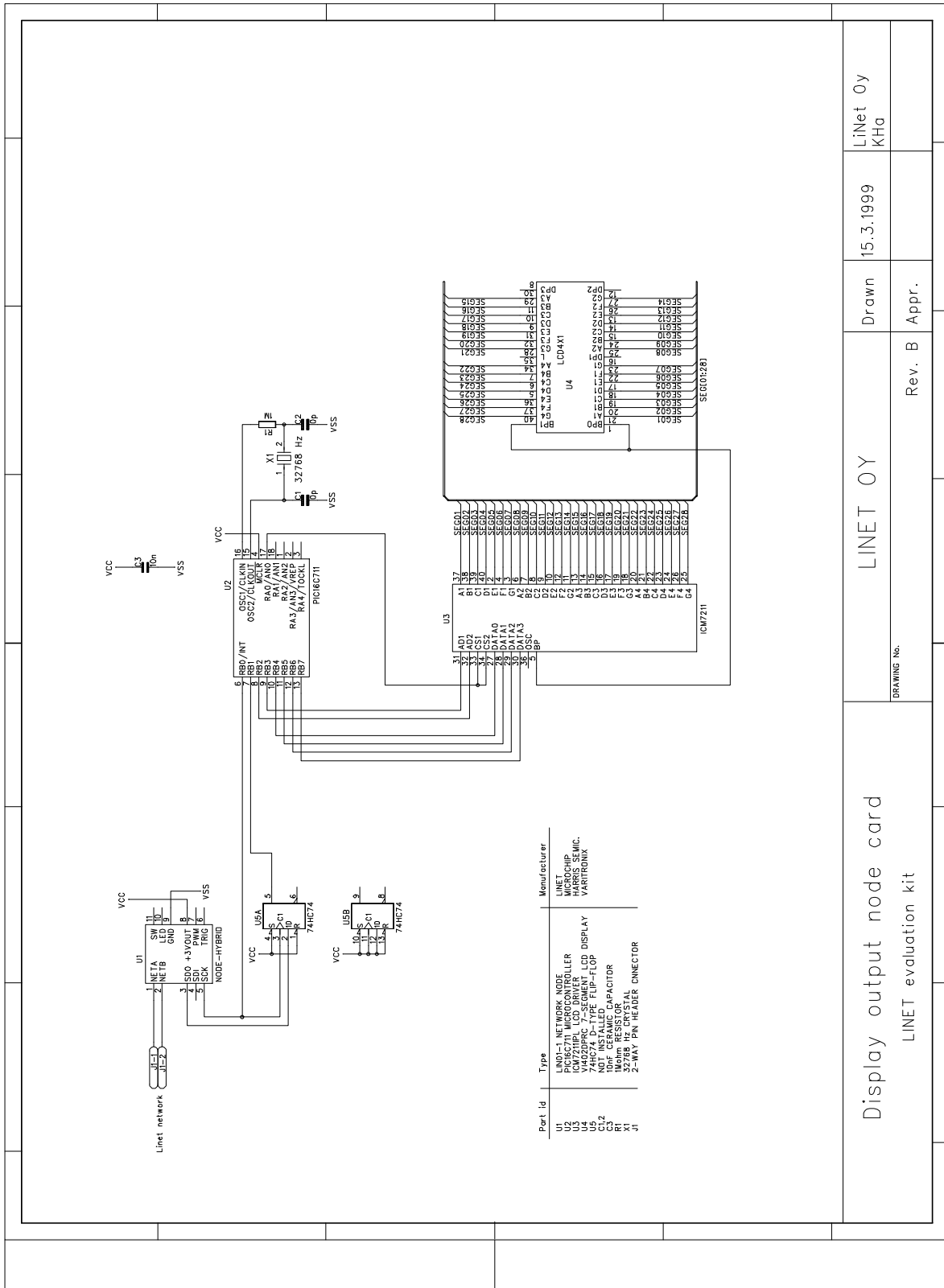
        call dspbcd       ;; display it

        movlw 1<<INTE    ;; get external interrupt bit
        movwf INTCON     ;; enable external interrupts

        retfie           ;; return and enable interrupts

end
```

APPENDIX B. THE DISPLAY UNIT SCHEMATICS



Display output node card  
LINET evaluation kit

Part Id: LINET OY

Drawn: 15.3.1999

Appr.: Rev. B

LiNet Oy  
KHG

DRAWING No.