

ELCD SERIES

INTRODUCTION

ALCD is Serial LCD module which is controlled through Serial communication. Most of existing LCD adopts Parallel communication which needs lots of control lines and complicated control. On the other hand, ALCD adopts Serial communication which needs only one or two lines to transmit data and display it on LCD. In addition, ALCD allows users to use LCD with easy even they don't have comprehensive knowledge of LCD module.

GENERAL DESCRIPTION

- Various size from 16x2 lines to 20x4lines
- 3lines-interface (GND, 5V, RX)
- 5V level of RS232C protocol (select one of 19200 and 4800 baud rate)
- Built-in functions such as location-control, screen-clear, cursor management and etc.

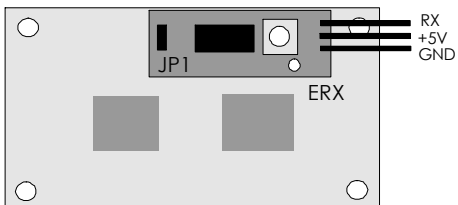
MODEL

Model	Range of display	Backlight
ELCD-162	16 BY 2	N/A
ELCD-162-BL	16 BY 2	LED
ELCD-164	16 BY 4	N/A
ELCD-164-BL	16 BY 4	LED
ELCD-204	20 BY 4	N/A
ELCD-204-BL	20 BY 4	LED
ELCD-162-BIG	16 BY 2	N/A
ELCD-162-BIG-BL	16 BY 2	LED

LAYOUT & OUTLINE



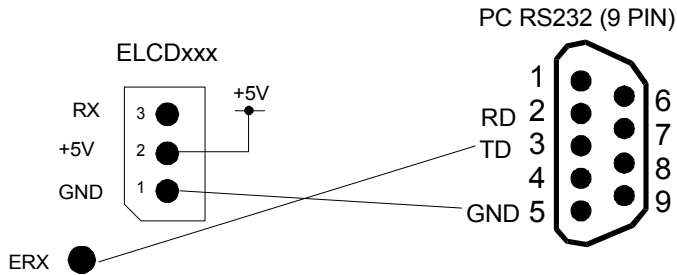
<Picture-1> Front view



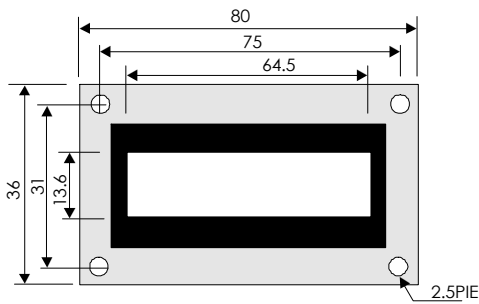
<Picture-2> Rear view

When JP1 is cut, baud rate is 4800. Otherwise, baud rate is 19200. Factory default is that JP1 is shorted. (JP1 is jumper type in some model)

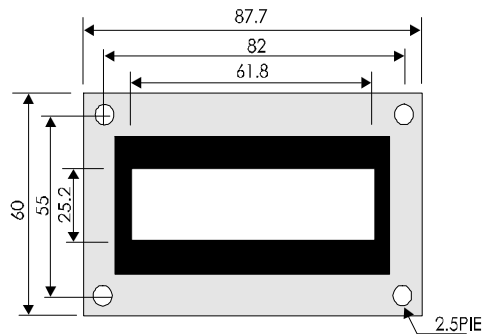
ERX is port receiving $\pm 10V$ of RS232 signal directly. You can make direct control from PC with connection shown as the following picture.



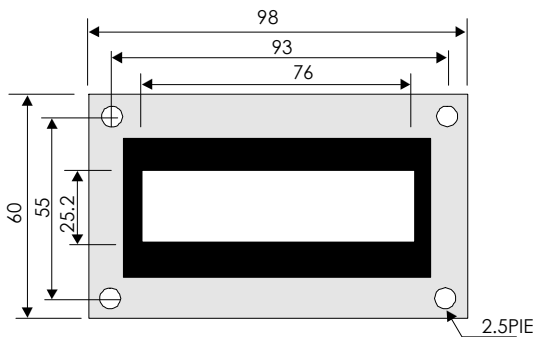
DIMENSION



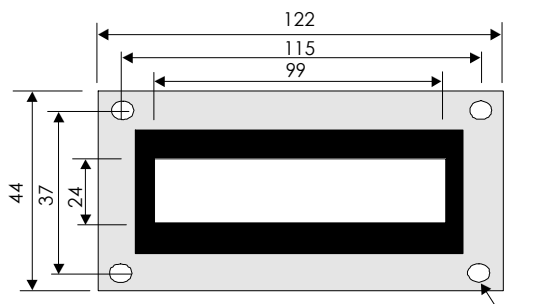
ELCD162 or ELCD162-BL



ELCD164 or ELCD164-BL



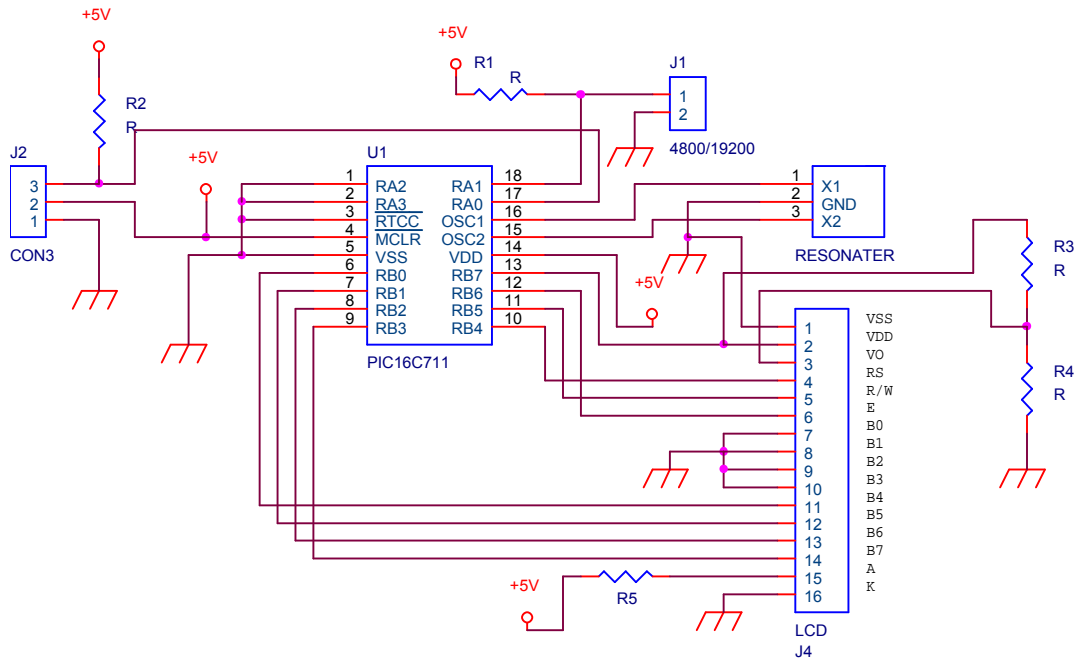
ELCD204 or ELCD204-BL



ELCD162-BIG or ELCD162-BIG-BL

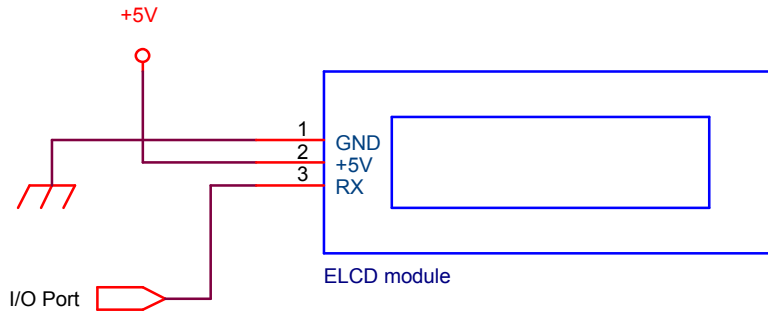
ELCD SERIES

CIRCUIT DRAWING

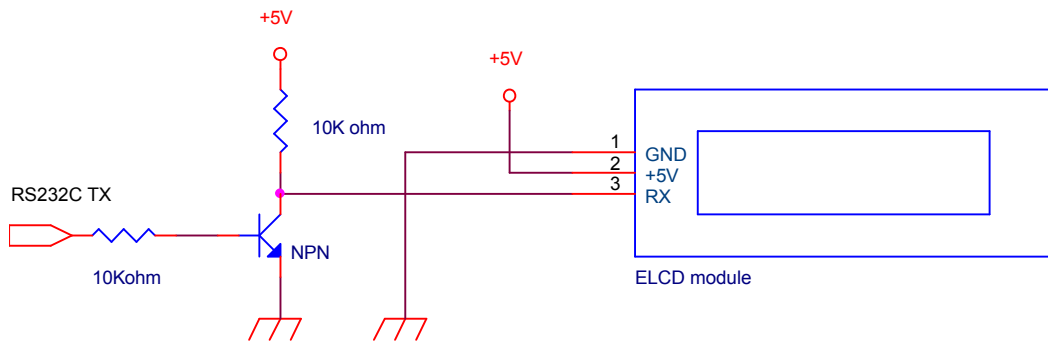


HOW TO USE

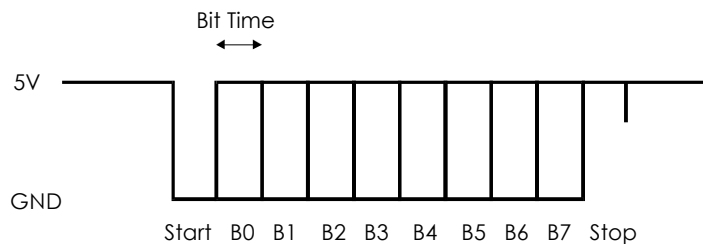
Connect 3pins connector attached to LCD module to HOST. (Microcontroller and PC can be used as HOST), and connect 5V and GND, connect RX line to TX terminal of PC or I/O port of PICmicro.



If you need to connect to RS232C port of PC or other devices, you have to organize extra level conversion circuit as the following picture. Because that $\pm 10V$ flows through RS232C line, you need to convert it into 5V level. If you use ERX port, it is same as using the following circuit.

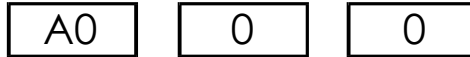


Send 5V level of signal in the forms of 8bit, NONE parity, 1 STOP bit to RX terminal of ELCD module.



ELCD SERIES

Bit Time determines baud rate. When Bit Time is 52mS, related baud rate is 9200. When 104mS, then 4800 baud rate. To display characters on ELCD screen, you have to send data in the forms of command and data. For instance, when you send LOCATE command, you have to send 0A1H, command code, first, and send two bytes of location data (X axis, Y axis) continuously.



The following table describes command code and format of ELCD.

Command (Hexadecimal)	Example	Transmitted bytes	Execution time	Description
A0	A0	1	10mS	Initialize LCD There needs 10mS of delay at least after sending command.
A3 01	A3 01	2		Clear LCD screen Display location is set at (0,0) automatically.
A1 X Y	A1 01 01	3		Appoint location of display (X axis is from 0 to 20. Y axis is from 0 to 3)
A2 String 0	A2 41 42 00	variable		Display characters on LCD screen "0" must be sent at end of string. (End Code)
A3 0C	A3 0C	2		Cursor OFF
A3 0E	A3 0E	2		Cursor ON (Default)
A4 Code Data 8	A4 08 03 01 0B A0 AA A3 80 30	10		Eight character code, from 8 to 15, is available for user-defined area. By sending certain BITMAP data to the area, users can display special codes.

USAGE IN PICBASIC

Because that PICBASIC has dedicated command for control serial LCD, users don't have to take care of LCD command code. They can use serial LCD module with ease by only LOCATE, PRINT, CLS, etc. (Refer to PICBASIC databook for detailed information). Refer to the following simple example.

```
SET PICBUS HIGH
CLS
LOCATE 0,0
PRINT "SERIAL LCD MOD."
LOCATE 9,1
PRINT "COMFILE"
```



HOW TO USE USER-DEFINED AREA

ELCD has eight user-defined areas and users can stored certain BITMAP data in the area, and display on LCD screen. In order the process, PICBASIC uses BUSOUT command. The following example show how to display arrow mark on LCD. (BITMAP has 5*8 of size.)

```
SET PICBUS HIGH
LCDINIT
BUSOUT &HA5,8,0,0,0,15,15,0,0,0
BUSOUT &HA5,9,0,&H10,&H18,&H1C,&H1C,&H18,&H10,0
LOCATE 0,0
PRINT 8,9
```

USAGE WITH PICMICRO (ASSEMBLY LANGUAGE)

The following example is to control serial LCD by PICmicro in Assembly language. Although some devices which have built-in UART such as PIC16C7X can make it by hardware TX function, it is recommended to make it by software. In order to execute the following program, RX terminal of serial LCD must be connected to port0 of PORTB. (Device is PIC16C711 using 4MHz clock)

; The delay time used in the following program is for execution at 48 baud rate, 4MHz.

; If you execute it at 19200 baud rate, you should adjust the delay time.

; When you execute the program, 16x2 of serial LCD displays follows;

```
;
;           LINE 1 : COMFILE TECHNOLO
;           LINE 2 : LCD CONTROLLER..
```

```
LIST    P=16C711, F=INHX8M
```

```
;
;
; FILE DEFINITION
;
```

```
INDIR          EQU          00H          ; PAGE 0
RTCC           EQU          01H
PC             EQU          02H
STATUS        EQU          03H
FSR           EQU          04H
PORTA        EQU          05H
PORTB        EQU          06H
PCLATH        EQU          0AH
```

ELCD SERIES

```

OPTIONR      EQU      01H      ; PAGE 1
PCL          EQU      02H
TRISA        EQU      05H
TRISB        EQU      06H

VARIABLE LOOP_CNT      = 0CH
VARIABLE BF1           = 0DH
VARIABLE DELAY_TIMER = 0EH
VARIABLE DELAY1_TIMER = 0FH
VARIABLE FETCH_SEQ    = 10H

#DEFINE      IO_TX      PORTB,0

:
:      BIT DEFINITION
:

CF          EQU      .0      ; STATUS
DC          EQU      .1
ZF          EQU      .2
PD          EQU      .3
TO          EQU      .4
RP0        EQU      .5

RBIF        EQU      .0      ; INTCON REGISTER
INTF        EQU      .1
RTIF        EQU      .2
RBIE        EQU      .3
INTE        EQU      .4
RTIE        EQU      .5
EEIE        EQU      .6
GIE         EQU      .7

INTEDG      EQU      .6
RBPU        EQU      .7

:
:      MAIN ROUTINE
:

ORG 0
GOTO SIJAK
GOTO SIJAK
GOTO SIJAK
GOTO SIJAK

SIJAK

RAM_CLEAR
MOVLW 0CH
MOVWF FSR

RAM_1
CLRF  INDIR
INCF  FSR
BTFSS FSR,6
GOTO  RAM_1

BSF  STATUS,RP0
MOVLW B'00000000'
MOVWF TRISA

```

```

        MOVLW    B'00000000'
        MOVWF    TRISB
        MOVLW    B'00001111'      ; Enable Watch-dog , 1:8
        MOVWF    OPTIONR
        BCF     STATUS,RP0
        CLRF     PCLATH

;-----
; MAIN PROC
;-----
        MOVLW    0A0H              ; LCD initialization command
        CALL    TX_PROC
        MOVLW    .200              ; DELAY aroud 600US
        CALL    DELAY_US

MAIN_LOOP
LINE_0
        MOVLW    0A1H              ; LOCATE 0,0
        CALL    TX_PROC
        MOVLW    00H              ; ROW 0
        CALL    TX_PROC
        MOVLW    00H              ; COL 0
        CALL    TX_PROC

        MOVLW    0A2H              ; String command (PRINT)
        CALL    TX_PROC

NEXT_0
        BTFSC   FETCH_SEQ,4
        GOTO    NEXT_01
        CALL    DATA0_TBL
        CALL    TX_PROĆ
        INCF    FETCH_SEQ
        GOTO    NEXT_0

NEXT_01
        CLRF    FETCH_SEQ
        MOVLW    00H              ; End of string
        CALL    TX_PROC

LINE_1
        MOVLW    0A1H
        CALL    TX_PROC
        MOVLW    00H
        CALL    TX_PROC
        MOVLW    01H
        CALL    TX_PROC

        MOVLW    0A2H
        CALL    TX_PROC

NEXT_1
        BTFSC   FETCH_SEQ,4
        GOTO    NEXT_11
        CALL    DATA1_TBL
        CALL    TX_PROĆ
        INCF    FETCH_SEQ
        GOTO    NEXT_1

NEXT_11
        CLRF    FETCH_SEQ
        MOVLW    00H
        CALL    TX_PROC
        GOTO    MAIN_LOOP

```

ELCD SERIES

DATA0_TBL

```
MOVF    FETCH_SEQ,W
ANDWF   0FH
ADDWF   PC
RETLW  'C'
RETLW  'O'
RETLW  'M'
RETLW  'F'
RETLW  'I'
RETLW  'L'
RETLW  'E'
RETLW  ''
RETLW  'T'
RETLW  'E'
RETLW  'C'
RETLW  'H'
RETLW  'N'
RETLW  'O'
RETLW  'L'
RETLW  'O'
```

DATA1_TBL

```
MOVF    FETCH_SEQ,W
ANDWF   0FH
ADDWF   PC
RETLW  'L'
RETLW  'C'
RETLW  'D'
RETLW  ''
RETLW  'C'
RETLW  'O'
RETLW  'N'
RETLW  'T'
RETLW  'R'
RETLW  'O'
RETLW  'L'
RETLW  'L'
RETLW  'E'
RETLW  'R'
RETLW  ''
RETLW  ''
```

TX_PROC

```
MOVWF   BF1
MOVLW   .8 ; Because 8BIT transmission,
MOVWF   LOOP_CNT
BCF     IO_TX
CALL    DELAY_ONE
```

TX_1

```
CLRWDT
RRF     BF1
BTFSS   STATUS,CF
BCF     IO_TX
BTFSC   STATUS,CF
BSF     IO_TX
CALL    DELAY_ONE
DECFSZ  LOOP_CNT
GOTO    TX_1
BSF     IO_TX
CALL    DELAY_ONE
```

```

        RETURN

DELAY_ONE      ; 4800-> 208US Delay(64) , 19200->52US Delay (14)
        MOVLW      .14                      ; 19200 Baud Rate
        ;MOVLW     .64                      ; 4800 Baud Rate

DELAY_US
DL_0          MOVWF      DELAY_TIMER
              DECFSZ    DELAY_TIMER
              GOTO      DL_0
              RETURN
              END
    
```

USAGE WITH PICMICRO (C LANGUAGE)

If you use C language, you can control serial LCD much simpler. The following example is for controlling serial LCD by CCS-C (PICmicro C-compiler, CCS) (Device is PIC16C711 using 4MHz clock)

```

#include <16c711.h>
#include delay(clock=4000000) // When 4MHz
#include rs232(baud = 19200, xmit = PIN_B1, rcv= PIN_B0)

#define TRISB 0x85
#define PORTB 5

void main()
{
    char i;
    TRISB = 1;
    delay_ms(200); // Wait for initialization of LCD
    printf("%c%c",0xa3,0xa1); // cls
    while(1){
        printf("%c%c%c",0xa1,0,0); // locate 0,0
        printf("%cCOMFILE SERIAL %c",0xa2,0);
        printf("%c%c%c",0xa1,0,1); // locate 0,1
        printf("%cLCD MODULE%d %c",0xa2,i,0);
        i++;
    }
}
    
```