Application Note

1

egnite Software GmbH

Embedded Ethernet

# AN-001: Connecting an LC-Display

Version 1.0.0 PRELIMINARY

EMBEDDED ETHERNET

# Ethernut Application Note 001

egnite Software GmbH
Westring 303 • D-44629 Herne
Fon +49 (0)2323-925 375 • Fax +49 (0)2323-925 374
http://www.egnite.de

# Contents

Chapter 1

# Introduction

*Six I/O lines are enough.*

This application note demonstrates how to connect a HD44780 based LCD to the Ethernut board. The Hitachi HD44780 chip is one of the most common LCD controllers used in alphanumeric displays. Its interface is either 4 or 8 bits. In order to save I/O lines, the 4 bit mode is used here, requiring a total of six I/O lines. However, an additional I/O port may be used to switch an optional LED backlight on or off. This backlight may require several hundred milliampere. Therefore the LCD board has been equipped with its own power supply regulator.



**WARNING: As with all computer equipment, the Ethernut board and the LCD may be severely damaged by electrostatic discharge (ESD). Be sure to take proper precautions.**

Chapter 2

# Hardware

*Creating a simple add-on board.*

## I/O Lines

The following table shows, which lines of the Ethernut expansion connector are used to connect the LCD:

| Pin | Signal | Ethernut Connection |
|---|---|---|
| 1 | Power supply (GND) | DC and GND via LCD on-board regulator |
| 2 | Power supply (+5V) | |
| 3 | Contrast adjust | None |
| 4 | L = Instruction register<br>H = Data register | PE2 |
| 5 | Read/Write control<br>L = Write to LCD<br>H = Read from LCD | None, tied to GND |
| 6 | Enable (data strobe) | PE3 |
| 7 | Data line bit 0 | None, tied to GND in 4-bit mode |
| 8 | Data line bit 1 | |
| 9 | Data line bit 2 | |
| 10 | Data line bit 3 | |
| 11 | Data line bit 4 | PD4 |
| 12 | Data line bit 5 | PD5 |
| 13 | Data line bit 6 | PD6 |
| 14 | Data line bit 7 | PD7 |
| 15 | Backlight supply + | DC and GND via LCD on-board regulator and power MOSFET controlled by PB6 |
| 16 | Backlight supply - | |

# Schematic

The schematic is quite simple, but contains one or two features which may need some explanation.

Two sides of the board may be plugged into the Ethernut expansion connector. This offers the possibility to install both boards either in the sandwhich way (using X1 and X2) or side by side (using X3 and X4).

In order to save pins and reduce EMI, each of the two expansion connectors is divided into two separate connectors. The higher speed data and address bus lines are not routed to the board. However, enough pads are on the board to insert the full 64-pin expansion connector of Ethernut version 1.3. Previous Ethernut versions got a 56-pin expansion connector, which requires to reduce X2 or X4 by two ground pins.

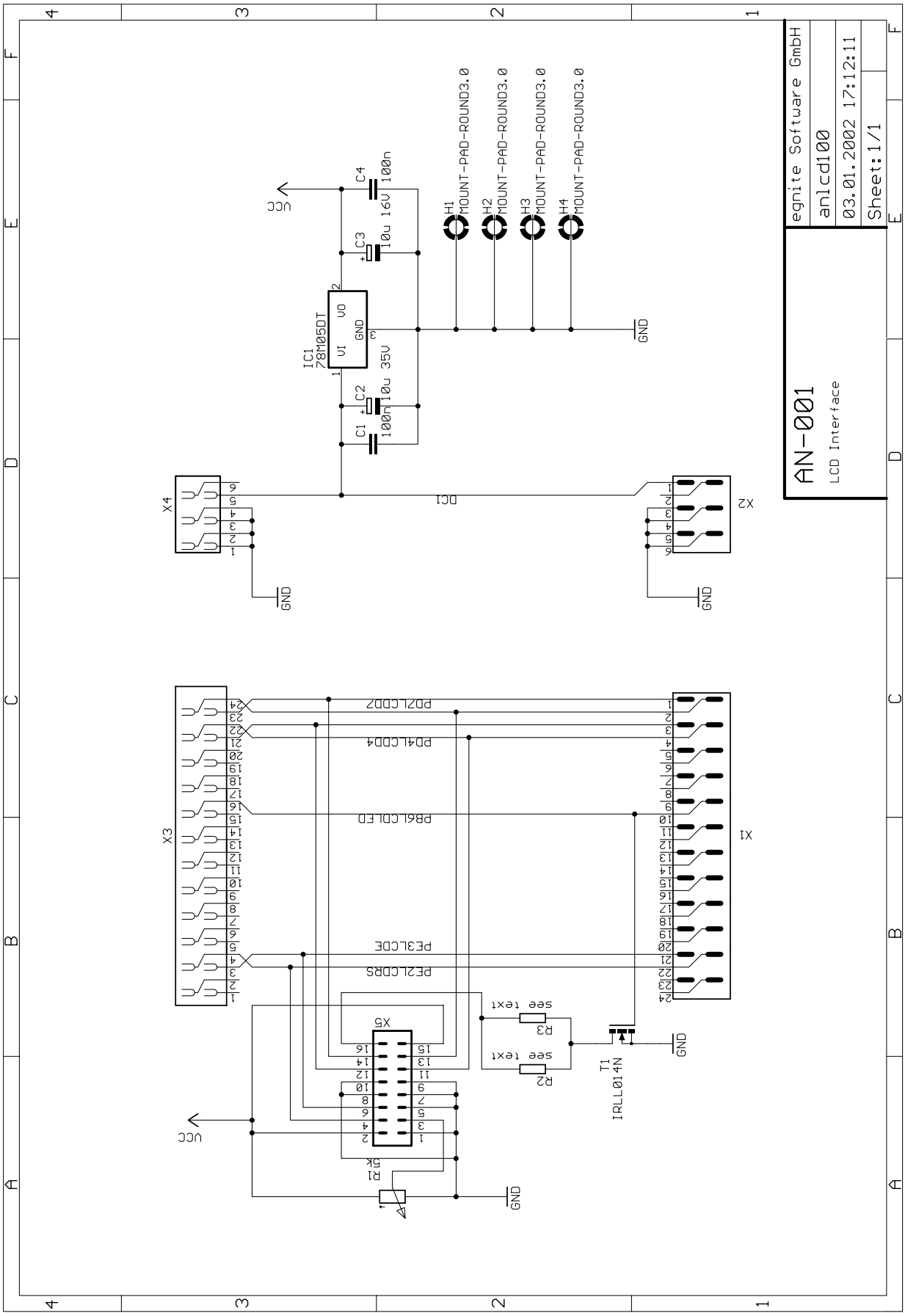
An n-channel power MOSFET is used to switch the LCD backlight. There are many types of these transistors avaiblable and will probably work as long as they are logic level types. Otherwise the 5 Volts from the Ethernut port may not be able to fully switch on the drain/source path. Standard MOSFETs need 10 Volts to do so.

Most LCD backlight supplies need an external serial resistor. Its value depends on the current drawn by the backlight LEDs and can be easily calculated by Ohm's law. Two resistors of different sizes are provided, R2 for 1206 pads and R3 for 2512 pads. Only one needs to be stuffed, but the resistor must be able to consume enough power.

As mentioned earlier, LED backlights might draw several hundred milliampere, which would be too much for the voltage regulator on the Ethernut board. Therefore the LCD board comes with its own regulator fed by the DC line of the Ethernut expansion connector. Make sure, that the power supply is able to provide enough current.

Finally the resistor pot on the board is used to trim the LCD contrast.

Figure 1: Schematic



# Board Layout

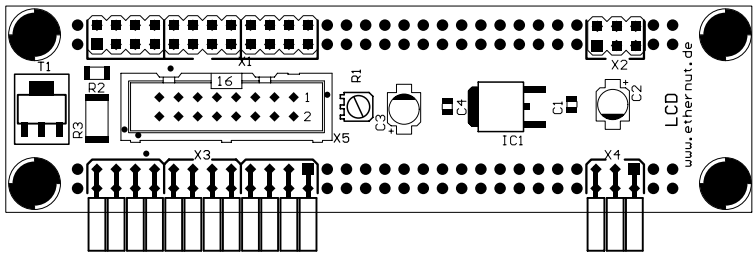



Figure 2: Component Placement

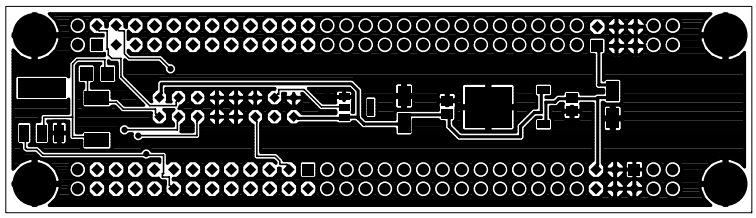


Figure 3: Top Layer

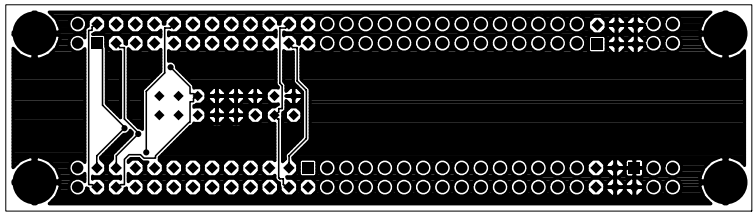


Figure 4: Bottom Layer

Chapter 3

# Software

*Driving the LCD.*

# Initializing the LCD

Some definitions first. As we find in the schematic, the LCD data port is connected to the upper 4 bits of port D.

```
#define LCD_DATA_DDR    DDRD
#define LCD_DATA_PORT   PORTD
```

The two control lines are connected to bits 2 and 3 of port E.

```
#define LCD_CTRL_DDR    DDRE
#define LCD_CTRL_PORT   PORTE
#define LCD_CTRL_E      PE3
#define LCD_CTRL_RS     PE2
```

The LCD backlight is controlled by bit 6 on port B.

```
#define LCD_LIGHT_DDR   DDRB
#define LCD_LIGHT_PORT  PORTB
#define LCD_LIGHT_BIT   PB6
```

A single routine is used to initialize all I/O lines and setup the LCD. Most notable is the for loop somewhere in the middle of the routine. When the LCD is powered up, the LCD controller will start in 8 bit mode. Switching it into 4 bit mode requires a specific command sequence. The loop will send three nibbles of value 3, two of value 2 and finally of value 8.

```
void LcdInit(void)
{
    u_char i;

    /*
     * Set LCD register select low.
     */
    cbi(LCD_CTRL_PORT, LCD_CTRL_RS);
    sbi(LCD_CTRL_DDR, LCD_CTRL_RS);

    /*
     * Set LCD enable low.
     */
    cbi(LCD_CTRL_PORT, LCD_CTRL_E);
    sbi(LCD_CTRL_DDR, LCD_CTRL_E);
```

```
    /*
     * Set direction register of LCD data port.
     */
    sbi(LCD_DATA_DDR, 4);
    sbi(LCD_DATA_DDR, 5);
    sbi(LCD_DATA_DDR, 6);
    sbi(LCD_DATA_DDR, 7);

    /*
     * Initialize for 4-bit operation.
     */
    NutDelay(10);
    for(i = 0; i < 3; i++) {
        LcdSendNibble(0x33);
        NutDelay(4);
    }
    LcdSendNibble(0x22);
    NutDelay(1);

    /*
     * 4-bit mode, 2/4 lines, 5x7 matrix.
     */
    LcdWriteCmd(0x28, 1);

    /*
     * Switch display, cursor and cursor blinking off.
     */
    LcdSwitch(0, LCD_SWITCH_ALL);

    /*
     * Clear display.
     */
    LcdClear();

    /*
     * Increment cursor when writing.
     */
    LcdWriteCmd(0x06, 1);

    /*
     * Lit LED and switch display on.
     */
    sbi(LCD_LIGHT_DDR, LCD_LIGHT_BIT);
    sbi(LCD_LIGHT_PORT, LCD_LIGHT_BIT);
    LcdSwitch(LCD_SWITCH_DISPLAY, 0);
}
```

# Controlling the LCD

The initialization routine already uses other subroutines to send data bytes and instructions to the LCD controller. Remember, that we drive the LCD in 4 bit mode. A special routine named LcdSendNibble will send the upper four bits of a character variable to the LCD data port. It sets the data lines and toggles the enable line.

```
static inline void LcdSendNibble(u_char nib)
{
    outp((inp(PORTD) & 0x0F) | (nib & 0xF0), PORTD);
    sbi(LCD_CTRL_PORT, LCD_CTRL_E);
    asm volatile("nop\n\tnop");
    cbi(LCD_CTRL_PORT, LCD_CTRL_E);
}
```

Another routine called LcdSendByte sends a complete byte to the LCD controller by calling LcdSendNibble twice.

```
static inline void LcdSendByte(u_char ch, u_char xt)
{
    LcdSendNibble(ch & 0xF0);
    LcdSendNibble(ch << 4);
    NutDelay(xt);
}
```

Finally two additional routines are provided to either send an instruction or a data byte. Note, that only instructions need a specific delay time.

```
static void LcdWriteCmd(u_char cmd, u_char xt)
{
    cbi(LCD_CTRL_PORT, LCD_CTRL_RS);
    LcdSendByte(cmd, xt);
}

void LcdWriteChar(u_char ch)
{
    sbi(LCD_CTRL_PORT, LCD_CTRL_RS);
    LcdSendByte(ch, 1);
}
```

One of the most often used instructions to be send to the display is the clear command, which blanks the display. A special routine LcdClear will do this.

```
void LcdClear(void)
{
    LcdWriteCmd(0x01, 2);
}
```

The initialization routine uses another command routine named LcdSwitch, which switches on and off certain mode bits in the LCD controller.

```
void LcdSwitch(u_char on, u_char off)
{
    switch_control |= on & LCD_SWITCH_ALL;
    switch_control &= ~(off & LCD_SWITCH_ALL);
    LcdWriteCmd(switch_control, 1);
}
```

When calling LcdWriteChar instead of LcdWriteCmd, the characters of the specified codes are displayed and the cursor is automatically advanced to the next character position. In addition to the normal 7 bit ASCII character set the controller can display many other characters. It is even possible to define up to 8 custom characters, which however isn't explained in this application note.


Figure 5: LCD Character Set

Another routine called LcdSetCursor may be used to place the cursor to a specific position.

```
void LcdSetCursor(u_char row, u_char col)
{
    u_char row_start[] = { 0x00, 0x40, LCD_ROW2_START, LCD_ROW3_START };

    LcdWriteCmd(0x80 | row_start[row & 0x03] | col, 1);
}
```

The LCD controller doesn't support cursor positioning on a specific column or row, but requires a memory address. This is not a big deal on 2 line displays, but address locations for the third and the second row depend on the number of columns being displayed. On 4 line displays with 16 columns each, the values are:

```
#define LCD_ROW2_START  0x10
#define LCD_ROW3_START  0x50
```

Chapter 4

# Links

*Where to find additional information.*

http://www.ethernut.de/

Information about the Ethernut board.

http://www.egnite.de/

Home of egnite Software GmbH, the developer of the Ethernut hardware.

Chapter 5

# Index