```
//nlcd.c
//------

//Newhaven Display NHD-0116GZ-FSW-FBW driver
//  Partially based on the example initialization found in the product specification pdf located at:
//
http://www.newhavendisplay.com/index.php?main_page=product_info&cPath=2_78&products_id=244
//
// This is a 4-bit initialization



#include <avr/pgmspace.h>
#include "nlcd.h"
#include "delay.h"

//Internal functions
void nlcd_command (unsigned char);
void nlcd_inc_pos (void);
void nlcd_dec_pos (void);
void nlcd_nibble (void);
void nlcd_place_data (unsigned char);
void nlcd_put_data (unsigned char);
void nlcd_set_command (void);
void nlcd_set_data (void);
void nlcd_bstring (void);
void nlcd_clear (void);
```

```
void nlcd_backspace (void);

static int next_pos;     //Address counter of the next display position to be written to
static int db_next_pos;  //Counter of the next buffer position (modulo 16) to be written to.

static unsigned char scrolling_enabled;  //Flag for determining when scrolling has been enabled
static unsigned char disp_buffer[DISP_BUFFER_SIZE];  //Buffer to hold displayed characters



//Functions
//---------

//LCD initialization - see page 12 of LCD driver datasheet
//  By default, scrolling is not enabled due to noticeable flicker.
int nlcd_init (void)
{
  DDRX |= (1 << PORTX_E ); //Set pin driver directions for signal E
  DDRX |= (1 << PORTX_RS); //Set pin driver directions for signal RS

  DATA_DDRX |= (0x0f << DATA_PORTX0);  //Set pin driver directions for data

  delay_ms(15);  //Wait >15ms after power is applied

  //Send wake up command 3 times to ensure it is received
  nlcd_command(WAKE_UP);
```

```
  delay_ms(5);
  nlcd_command(WAKE_UP);
  delay_us(100);
  nlcd_command(WAKE_UP);

  //Initialize screen -- see .h file for description of commands
  nlcd_command(FUNCTION_SET);
  nlcd_command(INIT_CURSOR);
  nlcd_command(INIT_DISPLAY);
  nlcd_command(ENTRY_MODE);

  scrolling_enabled = 0;
  //Initialization complete: print title and clear
  nlcd_string(PSTR("Comaidsystem 1.0"));
  delay_ms(1500);
  nlcd_wipe();  //Must call this at end to clear screen and initialize position trackers

  return 0;
}

//Send a character for the LCD to print
void nlcd_char (unsigned char c)
{
  if (c == BACKSPACE)
    nlcd_backspace();
```

```
  else {
    disp_buffer[(db_next_pos++) % DISP_BUFFER_SIZE] = c;

    if (next_pos == (LAST_POS + 1)) {

      if (scrolling_enabled)
        nlcd_bstring();

      else  //Scrolling has not been enabled...
        ;   //Already at last displayable position: do nothing
    }

    else {
      nlcd_set_data();
      nlcd_put_data(c);
      nlcd_inc_pos();
    }
  }
}

//Send a constant string for the LCD to print, erasing the current display.
void nlcd_string (const char *str)
{
  nlcd_wipe();
```

```
  while(pgm_read_byte(str) != EOL)
    nlcd_char(pgm_read_byte(str++));
}

//Send a non-constant (variable) string for the LCD to print, erasing the current display.
void nlcd_vstring(unsigned char* vstr)
{
  nlcd_wipe();

  unsigned char* strptr = vstr;

  while(*strptr != EOL)
    nlcd_char(*strptr++);
}

//Reset position trackers, wiping the buffer, and clears the screen
void nlcd_wipe (void)
{
  next_pos = FIRST_POS;
  db_next_pos = FIRST_POS;

  nlcd_clear();
}
```

```
//Flash text on screen for 'sec' seconds at FPS
void nlcd_flash (int sec)
{
  int i = 0;

  for ( ; i < (sec * FPS); i++) {
    nlcd_command(DISP_OFF);
    delay_ms(FPS_DELAY);
    nlcd_command(DISP_ON);
    delay_ms(FPS_DELAY);
  }

  nlcd_command(INIT_DISPLAY); //Put display back to normal
}

//LCD will scroll when receiving characters past screen edge
void nlcd_enable_scrolling (void)
{
  scrolling_enabled = 1;
  if ((next_pos >= LAST_POS))
    nlcd_bstring(); // Needed in case we enable scrolling after text hits end of screen
}

void nlcd_disable_scrolling (void)
{
```

```
  scrolling_enabled = 0;
}



//Internal functions
//------------------

//Send a command to the LCD
void nlcd_command (unsigned char c)
{
  nlcd_set_command();
  nlcd_put_data(c);
}

//Setting: send data to LCD
void nlcd_set_data (void)
{
  PORTX |=  (1 << PORTX_RS);  // RS high: send data
}

//Setting: send commands to LCD
void nlcd_set_command (void)
{
  PORTX &= ~(1 << PORTX_RS);  // RS low: send command
}
```

```
//Send 8 bits of information (command or character)
void nlcd_put_data (unsigned char c)
{
  nlcd_place_data((c >> 4) & 0x0f);  //Place upper data on output Port
  nlcd_nibble();                  //Clock upper 4 bits

  nlcd_place_data(c & 0x0f);        //Place lower data on output Port
  nlcd_nibble();                  //Clock lower 4 bits

  delay_us(100);  //Allow time for data to be handled by LCD
}

//Place data on output pins
void nlcd_place_data (unsigned char c)
{
  DATA_PORTX &= ~(0x0f << DATA_PORTX0);
  DATA_PORTX |= ((c & 0x0f) << DATA_PORTX0);
}

//Inform LCD there is data on the line
void nlcd_nibble (void)
{
  PORTX |= (1 << PORTX_E);    //Rising edge
  delay_us(1);              //Enable pulse width >=300ns
```

```
  PORTX &= ~(1 << PORTX_E);  //Clock enable: falling edge
  delay_us(1);               //Enable pulse width >=300ns
}

//Increment the position of the next address
void nlcd_inc_pos (void)
{
  next_pos++;

  if (next_pos == HALF_POS)
    nlcd_command(DDRAM_HALF_ADDRESS);  //Move across DDRAM split
}

//Decrement the position of the next address
void nlcd_dec_pos (void)
{
  next_pos--;

  if (next_pos == (HALF_POS - 1))
    nlcd_command(DDRAM_PRIOR_HALF_ADDRESS);  //Move back to before DDRAM split
}

//Print the string currently residing in the buffer: used for pseudo-scrolling
void nlcd_bstring (void)
{
```

```
  int i = FIRST_POS;

  nlcd_clear();  //Clear the display but do not reset position trackers

  nlcd_set_data();     //Print first half of buffer
  while (i < HALF_POS)
    nlcd_put_data(disp_buffer[((i++) + db_next_pos) % DISP_BUFFER_SIZE]);

  nlcd_command(DDRAM_HALF_ADDRESS);  //Jump across DDRAM gap

  nlcd_set_data();          //Print last half of buffer
  while (i < (LAST_POS + 1))
    nlcd_put_data(disp_buffer[((i++) + db_next_pos) % DISP_BUFFER_SIZE]);
}

//Clear the display and return to home position (leftmost)
void nlcd_clear (void)
{
  nlcd_command(CLR_DISPLAY);
  delay_ms(4);
  nlcd_command(RETURN_HOME);
  delay_ms(4);
}

//Backspace requires special handling because of split DDRAM
```

```
void nlcd_backspace (void)
{
  if (next_pos == FIRST_POS)
    ;  //Already at first displayable position: do nothing

  else {
    disp_buffer[(--db_next_pos) % DISP_BUFFER_SIZE] = BLANK;

    if (next_pos == HALF_POS) //Currently right after DDRAM split: need to jump back over it
      nlcd_command(DDRAM_PRIOR_HALF_ADDRESS);

    else
      nlcd_command(MV_CURSOR_LEFT);  //Simply move cursor left to write a blank

    nlcd_set_data();
    nlcd_put_data(BLANK);
    nlcd_command(MV_CURSOR_LEFT);  //Writing a blank moved the cursor right
    nlcd_dec_pos();
  }
}
```