Project · Height measure

```
#include "ABlocks_LiquidCrystal_I2C.h"
#include <inttypes.h>
#include <Arduino.h>
#include <Wire.h>

// When the display powers up, it is configured as follows:
//
// 1. Display clear
// 2. Function set:
//    DL = 1; 8-bit interface data
//    N = 0; 1-line display
//    F = 0; 5x8 dot character font
// 3. Display on/off control:
//    D = 0; Display off
//    C = 0; Cursor off
//    B = 0; Blinking off
// 4. Entry mode set:
//    I/D = 1; Increment by 1
//    S = 0; No shift
//
// Note, however, that resetting the Arduino doesn't reset the LCD, so we
// can't assume that its in that state when a sketch starts (and the
// LiquidCrystal constructor is called).

LiquidCrystal_I2C::LiquidCrystal_I2C(uint8_t  lcd_addr,  uint8_t  lcd_cols,  uint8_t
lcd_rows, uint8_t charsize)
{
      _addr = lcd_addr;
      _cols = lcd_cols;
      _rows = lcd_rows;
      _charsize = charsize;
      _backlightval = LCD_BACKLIGHT;
}

void LiquidCrystal_I2C::begin() {
      Wire.begin();
      _displayfunction = LCD_4BITMODE | LCD_1LINE | LCD_5x8DOTS;

      if (_rows > 1) {
```

Project · Height measure

```
      _displayfunction |= LCD_2LINE;
}

// for some 1 line displays you can select a 10 pixel high font
if ((_charsize != 0) && (_rows == 1)) {
      _displayfunction |= LCD_5x10DOTS;
}

// SEE PAGE 45/46 FOR INITIALIZATION SPECIFICATION!
// according to datasheet, we need at least 40ms after power rises above 2.7V
// before sending commands. Arduino can turn on way befer 4.5V so we'll wait 50
delay(50);

// Now we pull both RS and R/W low to begin commands
expanderWrite(_backlightval); // reset expanderand turn backlight off (Bit 8 =1)
delay(1000);

//put the LCD into 4 bit mode
// this is according to the hitachi HD44780 datasheet
// figure 24, pg 46

// we start in 8bit mode, try to set 4 bit mode
write4bits(0x03 << 4);
delayMicroseconds(4500); // wait min 4.1ms

// second try
write4bits(0x03 << 4);
delayMicroseconds(4500); // wait min 4.1ms

// third go!
write4bits(0x03 << 4);
delayMicroseconds(150);

// finally, set to 4-bit interface
write4bits(0x02 << 4);

// set # lines, font size, etc.
command(LCD_FUNCTIONSET | _displayfunction);
```

Project · Height measure

```
	// turn the display on with no cursor or blinking default
	_displaycontrol = LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKOFF;
	display();

	// clear it off
	clear();

	// Initialize to default text direction (for roman languages)
	_displaymode = LCD_ENTRYLEFT | LCD_ENTRYSHIFTDECREMENT;

	// set the entry mode
	command(LCD_ENTRYMODESET | _displaymode);

	home();
}

/********** high level commands, for the user! */
void LiquidCrystal_I2C::clear(){
	command(LCD_CLEARDISPLAY);// clear display, set cursor position to zero
	delayMicroseconds(2000);  // this command takes a long time!
}

void LiquidCrystal_I2C::home(){
	command(LCD_RETURNHOME);  // set cursor position to zero
	delayMicroseconds(2000);  // this command takes a long time!
}

void LiquidCrystal_I2C::setCursor(uint8_t col, uint8_t row){
	int row_offsets[] = { 0x00, 0x40, 0x14, 0x54 };
	if (row > _rows) {
		row = _rows-1;    // we count rows starting w/0
	}
	command(LCD_SETDDRAMADDR | (col + row_offsets[row]));
}

// Turn the display on/off (quickly)
void LiquidCrystal_I2C::noDisplay() {
	_displaycontrol &= ~LCD_DISPLAYON;
	command(LCD_DISPLAYCONTROL | _displaycontrol);
```

Project · Height measure

```
}
void LiquidCrystal_I2C::display() {
      _displaycontrol |= LCD_DISPLAYON;
      command(LCD_DISPLAYCONTROL | _displaycontrol);
}

// Turns the underline cursor on/off
void LiquidCrystal_I2C::noCursor() {
      _displaycontrol &= ~LCD_CURSORON;
      command(LCD_DISPLAYCONTROL | _displaycontrol);
}
void LiquidCrystal_I2C::cursor() {
      _displaycontrol |= LCD_CURSORON;
      command(LCD_DISPLAYCONTROL | _displaycontrol);
}

// Turn on and off the blinking cursor
void LiquidCrystal_I2C::noBlink() {
      _displaycontrol &= ~LCD_BLINKON;
      command(LCD_DISPLAYCONTROL | _displaycontrol);
}
void LiquidCrystal_I2C::blink() {
      _displaycontrol |= LCD_BLINKON;
      command(LCD_DISPLAYCONTROL | _displaycontrol);
}

// These commands scroll the display without changing the RAM
void LiquidCrystal_I2C::scrollDisplayLeft(void) {
      command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVELEFT);
}
void LiquidCrystal_I2C::scrollDisplayRight(void) {
      command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVERIGHT);
}

// This is for text that flows Left to Right
void LiquidCrystal_I2C::leftToRight(void) {
      _displaymode |= LCD_ENTRYLEFT;
      command(LCD_ENTRYMODESET | _displaymode);
}
```

Project · Height measure

```
// This is for text that flows Right to Left
void LiquidCrystal_I2C::rightToLeft(void) {
      _displaymode &= ~LCD_ENTRYLEFT;
      command(LCD_ENTRYMODESET | _displaymode);
}

// This will 'right justify' text from the cursor
void LiquidCrystal_I2C::autoscroll(void) {
      _displaymode |= LCD_ENTRYSHIFTINCREMENT;
      command(LCD_ENTRYMODESET | _displaymode);
}

// This will 'left justify' text from the cursor
void LiquidCrystal_I2C::noAutoscroll(void) {
      _displaymode &= ~LCD_ENTRYSHIFTINCREMENT;
      command(LCD_ENTRYMODESET | _displaymode);
}

// Allows us to fill the first 8 CGRAM locations
// with custom characters
void LiquidCrystal_I2C::createChar(uint8_t location, uint8_t charmap[]) {
      location &= 0x7; // we only have 8 locations 0-7
      command(LCD_SETCGRAMADDR | (location << 3));
      for (int i=0; i<8; i++) {
            write(charmap[i]);
      }
}

// Turn the (optional) backlight off/on
void LiquidCrystal_I2C::noBacklight(void) {
      _backlightval=LCD_NOBACKLIGHT;
      expanderWrite(0);
}

void LiquidCrystal_I2C::backlight(void) {
      _backlightval=LCD_BACKLIGHT;
      expanderWrite(0);
}
```

Project · Height measure

```
/*********** mid level commands, for sending data/cmds */

inline void LiquidCrystal_I2C::command(uint8_t value) {
      send(value, 0);
}

inline size_t LiquidCrystal_I2C::write(uint8_t value) {
      send(value, Rs);
      return 1;   //fixed for new compiler version IDE 1.6+ !!! by Juanjo
}


/************ low level data pushing commands **********/

// write either command or data
void LiquidCrystal_I2C::send(uint8_t value, uint8_t mode) {
      uint8_t highnib=value&0xf0;
      uint8_t lownib=(value<<4)&0xf0;
      write4bits((highnib)|mode);
      write4bits((lownib)|mode);
}

void LiquidCrystal_I2C::write4bits(uint8_t value) {
      expanderWrite(value);
      pulseEnable(value);
}

void LiquidCrystal_I2C::expanderWrite(uint8_t _data){
      Wire.beginTransmission(_addr);
      Wire.write((int)(_data) | _backlightval);
      Wire.endTransmission();
}

void LiquidCrystal_I2C::pulseEnable(uint8_t _data){
      expanderWrite(_data | En);    // En high
      delayMicroseconds(1);         // enable pulse must be >450ns

      expanderWrite(_data & ~En);   // En low
```

Project · Height measure

```
      delayMicroseconds(50);        // commands need > 37us to settle
}

void LiquidCrystal_I2C::load_custom_character(uint8_t char_num, uint8_t *rows){
      createChar(char_num, rows);
}

void LiquidCrystal_I2C::setBacklight(uint8_t new_val){
      if (new_val) {
            backlight();            // turn backlight on
      } else {
            noBacklight();          // turn backlight off
      }
}

void LiquidCrystal_I2C::printstr(const char c[]){
      //This function is not identical to the function used for "real" I2C displays
      //it's here so the user sketch doesn't have to be changed
      print(c);
}
```