//YWROBOT

#ifndef LiquidCrystal\_I2C\_h

#define LiquidCrystal\_I2C\_h

#include <inttypes.h>

#include "Print.h"

#include <Wire.h>

// commands

#define LCD\_CLEARDISPLAY 0x01

#define LCD\_RETURNHOME 0x02

#define LCD\_ENTRYMODESET 0x04

#define LCD\_DISPLAYCONTROL 0x08

#define LCD\_CURSORSHIFT 0x10

#define LCD\_FUNCTIONSET 0x20

#define LCD\_SETCGRAMADDR 0x40

#define LCD\_SETDDRAMADDR 0x80

// flags for display entry mode

#define LCD\_ENTRYRIGHT 0x00

#define LCD\_ENTRYLEFT 0x02

#define LCD\_ENTRYSHIFTINCREMENT 0x01

#define LCD\_ENTRYSHIFTDECREMENT 0x00

// flags for display on/off control

#define LCD\_DISPLAYON 0x04

#define LCD\_DISPLAYOFF 0x00

#define LCD\_CURSORON 0x02

#define LCD\_CURSOROFF 0x00

#define LCD\_BLINKON 0x01

#define LCD\_BLINKOFF 0x00

// flags for display/cursor shift

#define LCD\_DISPLAYMOVE 0x08

#define LCD\_CURSORMOVE 0x00

#define LCD\_MOVERIGHT 0x04

#define LCD\_MOVELEFT 0x00

// flags for function set

#define LCD\_8BITMODE 0x10

#define LCD\_4BITMODE 0x00

#define LCD\_2LINE 0x08

#define LCD\_1LINE 0x00

#define LCD\_5x10DOTS 0x04

#define LCD\_5x8DOTS 0x00

// flags for backlight control

#define LCD\_BACKLIGHT 0x08

#define LCD\_NOBACKLIGHT 0x00

#define En B00000100 // Enable bit

#define Rw B00000010 // Read/Write bit

#define Rs B00000001 // Register select bit

class LiquidCrystal\_I2C : public Print {

public:

LiquidCrystal\_I2C(uint8\_t lcd\_Addr,uint8\_t lcd\_cols,uint8\_t lcd\_rows);

void begin(uint8\_t cols, uint8\_t rows, uint8\_t charsize = LCD\_5x8DOTS );

void clear();

void home();

void noDisplay();

void display();

void noBlink();

void blink();

void noCursor();

void cursor();

void scrollDisplayLeft();

void scrollDisplayRight();

void printLeft();

void printRight();

void leftToRight();

void rightToLeft();

void shiftIncrement();

void shiftDecrement();

void noBacklight();

void backlight();

void autoscroll();

void noAutoscroll();

void createChar(uint8\_t, uint8\_t[]);

void setCursor(uint8\_t, uint8\_t);

#if defined(ARDUINO) && ARDUINO >= 100

virtual size\_t write(uint8\_t);

#else

virtual void write(uint8\_t);

#endif

void command(uint8\_t);

void init();

////compatibility API function aliases

void blink\_on(); // alias for blink()

void blink\_off(); // alias for noBlink()

void cursor\_on(); // alias for cursor()

void cursor\_off(); // alias for noCursor()

void setBacklight(uint8\_t new\_val); // alias for backlight() and nobacklight()

void load\_custom\_character(uint8\_t char\_num, uint8\_t \*rows); // alias for createChar()

void printstr(const char[]);

////Unsupported API functions (not implemented in this library)

uint8\_t status();

void setContrast(uint8\_t new\_val);

uint8\_t keypad();

void setDelay(int,int);

void on();

void off();

uint8\_t init\_bargraph(uint8\_t graphtype);

void draw\_horizontal\_graph(uint8\_t row, uint8\_t column, uint8\_t len, uint8\_t pixel\_col\_end);

void draw\_vertical\_graph(uint8\_t row, uint8\_t column, uint8\_t len, uint8\_t pixel\_col\_end);

private:

void init\_priv();

void send(uint8\_t, uint8\_t);

void write4bits(uint8\_t);

void expanderWrite(uint8\_t);

void pulseEnable(uint8\_t);

uint8\_t \_Addr;

uint8\_t \_displayfunction;

uint8\_t \_displaycontrol;

uint8\_t \_displaymode;

uint8\_t \_numlines;

uint8\_t \_cols;

uint8\_t \_rows;

uint8\_t \_backlightval;

};

#endif