

DATE	17MAY2023
TEAM ID	NM2023TMID06791
PROJECT NAME	WOWKI PROJECT USE OF ULTRASONIC SENSOR

"""Provides an API for talking to HD44780 compatible character LCDs."""

import time

class LcdApi:

"""Implements the API for talking with HD44780 compatible character LCDs.

This class only knows what commands to send to the LCD, and not how to get them to the LCD.

It is expected that a derived class will implement the hal_*** functions.

"""

The following constant names were lifted from the avrlib lcd.h header file, however, I changed the definitions from bit numbers to bit masks.

 # HD44780 LCD controller command set

LCD_CLR = 0x01 # DB0: clear display
LCD_HOME = 0x02 # DB1: return to home position

LCD_ENTRY_MODE = 0x04 # DB2: set entry mode
LCD_ENTRY_INC = 0x02 # --DB1: increment
LCD_ENTRY_SHIFT = 0x01 # --DB0: shift

LCD_ON_CTRL = 0x08 # DB3: turn lcd/cursor on
LCD_ON_DISPLAY = 0x04 # --DB2: turn display on
LCD_ON_CURSOR = 0x02 # --DB1: turn cursor on
LCD_ON_BLINK = 0x01 # --DB0: blinking cursor

LCD_MOVE = 0x10 # DB4: move cursor/display
LCD_MOVE_DISP = 0x08 # --DB3: move display (0-> move cursor)
LCD_MOVE_RIGHT = 0x04 # --DB2: move right (0-> left)

LCD_FUNCTION = 0x20 # DB5: function set
LCD_FUNCTION_8BIT = 0x10 # --DB4: set 8BIT mode (0->4BIT mode)
LCD_FUNCTION_2LINES = 0x08 # --DB3: two lines (0->one line)
LCD_FUNCTION_10DOTS = 0x04 # --DB2: 5x10 font (0->5x7 font)
LCD_FUNCTION_RESET = 0x30 # See "Initializing by Instruction"

section

LCD_CGRAM = 0x40 # DB6: set CG RAM address
LCD_DDRAM = 0x80 # DB7: set DD RAM address

LCD_RS_CMD = 0
LCD_RS_DATA = 1

LCD_RW_WRITE = 0
LCD_RW_READ = 1

```

def __init__(self, num_lines, num_columns):
    self.num_lines = num_lines
    if self.num_lines > 4:
        self.num_lines = 4
    self.num_columns = num_columns
    if self.num_columns > 40:
        self.num_columns = 40
    self.cursor_x = 0
    self.cursor_y = 0
    self.implicit_newline = False
    self.backlight = True
    self.display_off()
    self.backlight_on()
    self.clear()
    self.hal_write_command(self.LCD_ENTRY_MODE | self.LCD_ENTRY_INC)
    self.hide_cursor()
    self.display_on()

def clear(self):
    """Clears the LCD display and moves the cursor to the top left
    corner.
    """
    self.hal_write_command(self.LCD_CLR)
    self.hal_write_command(self.LCD_HOME)
    self.cursor_x = 0
    self.cursor_y = 0

def show_cursor(self):
    """Causes the cursor to be made visible."""
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY |
                           self.LCD_ON_CURSOR)

def hide_cursor(self):
    """Causes the cursor to be hidden."""
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY)

def blink_cursor_on(self):
    """Turns on the cursor, and makes it blink."""
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY |
                           self.LCD_ON_CURSOR | self.LCD_ON_BLINK)

def blink_cursor_off(self):
    """Turns on the cursor, and makes it no blink (i.e. be solid)."""
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY |
                           self.LCD_ON_CURSOR)

def display_on(self):
    """Turns on (i.e. unblanks) the LCD."""
    self.hal_write_command(self.LCD_ON_CTRL | self.LCD_ON_DISPLAY)

def display_off(self):
    """Turns off (i.e. blanks) the LCD."""
    self.hal_write_command(self.LCD_ON_CTRL)

def backlight_on(self):
    """Turns the backlight on.

```

This isn't really an LCD command, but some modules have backlight

```

        controls, so this allows the hal to pass through the command.
        """
        self.backlight = True
        self.hal_backlight_on()

def backlight_off(self):
    """Turns the backlight off.

    This isn't really an LCD command, but some modules have backlight
    controls, so this allows the hal to pass through the command.
    """
    self.backlight = False
    self.hal_backlight_off()

def move_to(self, cursor_x, cursor_y):
    """Moves the cursor position to the indicated position. The
cursor
    position is zero based (i.e. cursor_x == 0 indicates first
column).
    """
    self.cursor_x = cursor_x
    self.cursor_y = cursor_y
    addr = cursor_x & 0x3f
    if cursor_y & 1:
        addr += 0x40    # Lines 1 & 3 add 0x40
    if cursor_y & 2:    # Lines 2 & 3 add number of columns
        addr += self.num_columns
    self.hal_write_command(self.LCD_DDRAM | addr)

def putchar(self, char):
    """Writes the indicated character to the LCD at the current
cursor
    position, and advances the cursor by one position.
    """
    if char == '\n':
        if self.implied_newline:
            # self.implied_newline means we advanced due to a
wraparound,
            # so if we get a newline right after that we ignore it.
            pass
        else:
            self.cursor_x = self.num_columns
    else:
        self.hal_write_data(ord(char))
        self.cursor_x += 1
    if self.cursor_x >= self.num_columns:
        self.cursor_x = 0
        self.cursor_y += 1
        self.implied_newline = (char != '\n')
    if self.cursor_y >= self.num_lines:
        self.cursor_y = 0
    self.move_to(self.cursor_x, self.cursor_y)

def putstr(self, string):
    """Write the indicated string to the LCD at the current cursor
    position and advances the cursor position appropriately.
    """
    for char in string:

```

```

        self.putchar(char)

def custom_char(self, location, charmap):
    """Write a character to one of the 8 CGRAM locations, available
    as chr(0) through chr(7).
    """
    location &= 0x7
    self.hal_write_command(self.LCD_CGRAM | (location << 3))
    self.hal_sleep_us(40)
    for i in range(8):
        self.hal_write_data(charmap[i])
        self.hal_sleep_us(40)
    self.move_to(self.cursor_x, self.cursor_y)

def hal_backlight_on(self):
    """Allows the hal layer to turn the backlight on.

    If desired, a derived HAL class will implement this function.
    """
    pass

def hal_backlight_off(self):
    """Allows the hal layer to turn the backlight off.

    If desired, a derived HAL class will implement this function.
    """
    pass

def hal_write_command(self, cmd):
    """Write a command to the LCD.

    It is expected that a derived HAL class will implement this
    function.
    """
    raise NotImplementedError

def hal_write_data(self, data):
    """Write data to the LCD.

    It is expected that a derived HAL class will implement this
    function.
    """
    raise NotImplementedError

def hal_sleep_us(self, usecs):
    """Sleep for some time (given in microseconds)."""
    time.sleep_us(usecs)

```