**#include** <msp430.h>

**#include** <inttypes.h>

**#include**<stdio.h>

**#define** CMD 0

**#define** DATA 1

**#define** LCD\_OUT P1OUT

**#define** LCD\_DIR P1DIR

**#define** D4 BIT4

**#define** D5 BIT5

**#define** D6 BIT6

**#define** D7 BIT7

**#define** RS BIT2

**#define** EN BIT3

**#define** SW BIT0

**#define** LED BIT1

/\*\*

\*@brief Delay function for producing delay in 0.1 ms increments

\*@param t milliseconds to be delayed

\*@return void

\*\*/

**void** **delay**(uint16\_t t)

{

uint16\_t i;

**for**(i=t; i > 0; i--)

**\_\_delay\_cycles**(100);

}

/\*\*

\*@brief Function to pulse EN pin after data is written

\*@return void

\*\*/

**void** **pulseEN**(**void**)

{

LCD\_OUT |= EN; // Giving a falling edge at EN pin

delay(1);

LCD\_OUT &= ~EN;

delay(1);

}

/\*\*

\*@brief Function to write data/command to LCD

\*@param value Value to be written to LED

\*@param mode Mode -> Command or Data

\*@return void

\*\*/

**void** **lcd\_write**(uint8\_t value, uint8\_t mode)

{

**if**(mode == CMD)

LCD\_OUT &= ~RS; // Set RS -> LOW for Command mode

**else**

LCD\_OUT |= RS; // Set RS -> HIGH for Data mode

LCD\_OUT = ((LCD\_OUT & 0x0F) | (value & 0xF0)); // Write high nibble first

pulseEN();

delay(1);

LCD\_OUT = ((LCD\_OUT & 0x0F) | ((value << 4) & 0xF0)); // Write low nibble next

pulseEN();

delay(1);

}

/\*\*

\*@brief Function to print a string on LCD

\*@param \*s pointer to the character to be written.

\*@return void

\*\*/

**void** **lcd\_print**(**char** \*s)

{

**while**(\*s)

{

lcd\_write(\*s, DATA);

s++;

}

}

/\*\*

\*@brief Function to move cursor to desired position on LCD

\*@param row Row Cursor of the LCD

\*@param col Column Cursor of the LCD

\*@return void

\*\*/

**void** **lcd\_setCursor**(uint8\_t row, uint8\_t col)

{

**const** uint8\_t row\_offsets[] = { 0x00, 0x40};

lcd\_write(0x80 | (col + row\_offsets[row]), CMD);

delay(1);

}

/\*\*

\*@brief Initialize LCD

\*\*/

**void** **lcd\_init**()

{

LCD\_DIR |= (D4+D5+D6+D7+RS+EN);

LCD\_OUT &= ~(D4+D5+D6+D7+RS+EN);

delay(150); // Wait for power up ( 15ms )

lcd\_write(0x33, CMD); // Initialization Sequence 1

delay(50); // Wait ( 4.1 ms )

lcd\_write(0x32, CMD); // Initialization Sequence 2

delay(1); // Wait ( 100 us )

// All subsequent commands take 40 us to execute, except clear & cursor return (1.64 ms)

lcd\_write(0x28, CMD); // 4 bit mode, 2 line

delay(1);

lcd\_write(0x0C, CMD); // Display ON, Cursor OFF, Blink OFF

delay(1);

lcd\_write(0x01, CMD); // Clear screen

delay(20);

lcd\_write(0x06, CMD); // Auto Increment Cursor

delay(1);

lcd\_setCursor(0,0); // Goto Row 1 Column 1

}

/\*\*

\* @brief

\* These settings are wrt enabling ADC10 on Lunchbox

\*\*/

**void** **main**(**void**) {

WDTCTL = WDTPW | WDTHOLD; //! Stop Watchdog (Not recommended for code in production and devices working in field)

lcd\_init();

P1DIR |= BIT1; // Set LED pin -> Output

P1DIR &= ~SW; // Set SW pin -> Input

**while**(1)

{

**if**((P1IN & SW)) // If Output of IR Sensor and P!IN is low

{

lcd\_setCursor(0,0); // Set the cursor in the LCD

lcd\_print("Current Status"); // Printing this command

lcd\_setCursor(1,0); // Set the cursor in the LCD

lcd\_print("----NO FIRE---"); // Printing this command

}

**else**{ // If Output is high

P1DIR |= BIT1; // Set as input

delay(5000);

lcd\_setCursor(0,0); // Set the cursor in the LCD

lcd\_print("Current Status"); // Printing this command

lcd\_setCursor(1,0); // Set the cursor in the LCD

lcd\_print("!!FIRE ALERT!!"); // Printing this command

}

P1DIR &= ~ BIT1; // Set as Input

}

}