OPTREX IM50240
PWB50240-CEW

SAMPLE PROGRAM OPTREX.C

LCD Using Hitachi HD44100.

This display requires a constant 50% duty cycle clock
See HD44100 datasheet under ‚Static Drive' section for more info.

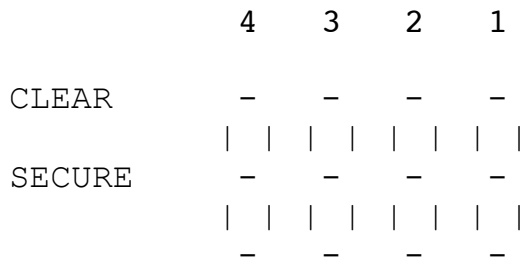
pin from left to right

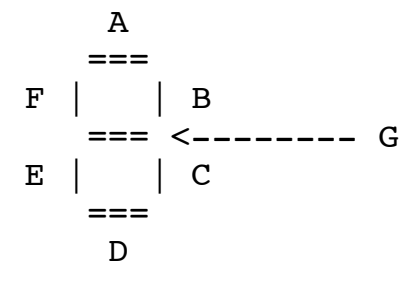
1 input data
2 latch clock
3 shift clock
4 input square waves of 50% Duty cycle (20-200Hz)
5 NC
6 Vcc +5V
7 GND -
8 SW1 CLEAR
9 SW2 SECURE
10 CHASSIS GND (Connect to 7)

OPTREX IM50240
PWB50240-CEW

35-shift registers.

Bits marked X are unused. There are no decimal points on this display.






```
 1 SECURE
 2 CLEAR
 3 X
 4 1G
 5 1F
 6 1E
 7 1D
 8 1C
 9 1B
10 1A
11 X
12 2G
13 2F
14 2E
15 2D
16 2C
17 2B
18 2A
19 X
20 3G
21 3F
22 3E
23 3D
24 3C
25 3B
26 3A
27 X
28 4G
29 4F
30 4E
31 4D
32 4C
33 4B
34 4A
35 X
```

```
// optrex.c
// Optrex IM50240 test program
// Christopher Netherton
// cnetherton@gmail.com
// November 18, 2007
// compiled using HI-TECH Pic C Lite
//
// Feel free to use this code.  Please leave comments
// in and simply add to them.  I would love to hear of any
// fixes, improvements or enhancements....Christopher
//
// tested on 16F877A with 20MHz oscillator
// I deliberately left the interrupt slow in order to
// see the test routines.  For more speed set the
// prescaler bits to 0b000. Also, TMR0 has been left at 0
// so it has to count ~255 times before the interupt pops
// with a 1:1 prescaler.  I have not tried changing TMR0 values yet.
//
// Uses PORTB
//               BIT 0 is 50% duty clock
//               BIT 1 is data to be shifted in
//               BIT 2 is shift clock (shift on falling edge)
//               BIT 3 is latch clock (latch on falling edge)


#include   <htc.h>

// macros for simplifying setting/clearing individual bits
// in the word written to the output port
#define bitset(var,bitno) ((var) |= 1 << (bitno))
#define bitclr(var,bitno) ((var) &= ~(1 << (bitno)))


/*   variables for optrex interrupt routine */
volatile bit CLOCK_BIT;
volatile bit DATA_BIT;
volatile bit DATA_READY;
volatile bit DATA_LOADED;
volatile bit DATA_CLOCKED;
volatile bit LATCH_READY;
volatile bit LATCH_CLOCKED;
unsigned int PORT_BYTE;
/*   end variables for optrex interrupt routine */

// bit mapping for individual segments in optrex display
unsigned int LED_MAP[16] =
     {0b01111110,      //0   msb not used, segments a-b use bits 6-0
     0b00110000,       //1
     0b01101101,       //2
     0b01111001,       //3
     0b00110011,       //4
     0b01011011,       //5
     0b01011111,       //6
     0b01110000,       //7
     0b01111111,       //8
     0b01111011,       //9
```

```
      0b01110111,       //A
      0b00011111,       //b
      0b01001110,       //C
      0b00111101,       //d
      0b01001111,       //E
      0b01000111};      //F
 unsigned int shift_string[5];
 unsigned int digit;
 unsigned int shift;
 unsigned int i;
 unsigned int j;

 static void interrupt
 isr(void)
 {
 if (!CLOCK_BIT)
      {
      CLOCK_BIT = 1;
      bitset(PORT_BYTE,0);
      if (DATA_BIT)
            {
            bitset(PORT_BYTE,1);
            }
      else
            {
            bitclr(PORT_BYTE,1);
            }
      bitset(PORT_BYTE,2);    //data clock bit high (always redy to
shift)
      bitset(PORT_BYTE,3);    //latch clock bit high (always ready to
latch)
      if (DATA_READY)         //ensures data bit set at output before
falling edge
            {
            DATA_READY = 0;
            DATA_LOADED = 1;
            }
      }
 else
      {
      CLOCK_BIT = 0;
      bitclr(PORT_BYTE,0);    //clocking byte of port
      if (DATA_BIT)
            {
            bitset(PORT_BYTE,1);
            }
      else
            {
            bitclr(PORT_BYTE,1);
            }
      if (DATA_LOADED)
            {
            DATA_LOADED = 0;
            bitclr(PORT_BYTE,2);    //falling edge shifts data into
register
            DATA_CLOCKED = 1;
            }
      if (LATCH_READY)
```

```
            {
            LATCH_READY = 0;
            bitclr(PORT_BYTE,3);    //falling edge latches shift
register to display
            LATCH_CLOCKED = 1;
            }
      }
 PORTB = PORT_BYTE;
 T0IF = 0;
 }

 void
 putoptrex(int dig_4, int dig_3, int dig_2, int dig_1)
 {
 DATA_READY = 0;
 DATA_CLOCKED = 0;
 LATCH_READY = 0;
 LATCH_CLOCKED = 0;

 //start test routine 1
 //test display by writing 0's then 1's then 0's
 DATA_BIT = 0;
 for (i=1;i<=35;i++)
      {
      DATA_READY = 1;
      while (DATA_CLOCKED == 0);
      DATA_CLOCKED = 0;
      }
 LATCH_READY = 1;
 while (LATCH_CLOCKED == 0);
 LATCH_CLOCKED = 0;
 DATA_BIT = 1;
 for (i=1;i<=35;i++)
      {
      DATA_READY = 1;
      while (DATA_CLOCKED == 0);
      DATA_CLOCKED = 0;
      }
 LATCH_READY = 1;
 while (LATCH_CLOCKED == 0);
 LATCH_CLOCKED = 0;
 DATA_BIT = 0;
 for (i=1;i<=35;i++)
      {
      DATA_READY = 1;
      while (DATA_CLOCKED == 0);
      DATA_CLOCKED = 0;
      }
 LATCH_READY = 1;
 while (LATCH_CLOCKED == 0);
 LATCH_CLOCKED = 0;
 //end test routine 1

 //start test routine 2
 // write 1 bit to disply, incrementing which bit is set
 for (j=1;j<=35;j++)
 {
 for (i=1;i<=35;i++)
```

```
     {
     if (i==j)
           {
           DATA_BIT = 1;
           }
     else
           {
           DATA_BIT = 0;
           }
DATA_READY = 1;
     while (DATA_CLOCKED == 0);
     DATA_CLOCKED = 0;
     }
LATCH_READY = 1;
while (LATCH_CLOCKED == 0);
LATCH_CLOCKED = 0;
}
//end test routine 2
shift_string[1] = LED_MAP[dig_1];
shift_string[2] = LED_MAP[dig_2];
shift_string[3] = LED_MAP[dig_3];
shift_string[4] = LED_MAP[dig_4];

//set bit 1 of 35 - 'secure' indicator on LCD
DATA_BIT = 1;
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
//set bit 2 of 35 - 'clear' indicator on LCD
DATA_BIT = 0;
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
//set bit 3 of 35 - not used data in shift register
DATA_BIT = 1;
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;

for (digit=1;digit<=4;digit++)
{
     if ((shift_string[digit] & 0b00000001) > 0)
     {
           DATA_BIT = 1;
     }
     else
    {
           DATA_BIT = 0;
     }
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
     if ((shift_string[digit] & 0b00000010) > 0)
     {
           DATA_BIT = 1;
     }
     else
    {
```

```
            DATA_BIT = 0;
      }
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
      if ((shift_string[digit] & 0b00000100) > 0)
      {
            DATA_BIT = 1;
      }
      else
     {
            DATA_BIT = 0;
      }
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
      if ((shift_string[digit] & 0b00001000) > 0)
      {
            DATA_BIT = 1;
      }
      else
     {
            DATA_BIT = 0;
      }
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
      if ((shift_string[digit] & 0b00010000) > 0)
      {
            DATA_BIT = 1;
      }
      else
     {
            DATA_BIT = 0;
      }
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
      if ((shift_string[digit] & 0b00100000) > 0)
      {
            DATA_BIT = 1;
      }
      else
     {
            DATA_BIT = 0;
      }
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
      if ((shift_string[digit] & 0b01000000) > 0)
      {
            DATA_BIT = 1;
      }
      else
     {
            DATA_BIT = 0;
      }
DATA_READY = 1;
```

```
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
     if ((shift_string[digit] & 0b100000000) > 0)
     {
           DATA_BIT = 1;
     }
     else
    {
           DATA_BIT = 0;
     }
DATA_READY = 1;
while (DATA_CLOCKED == 0);   //wait for data to be shifted out
DATA_CLOCKED = 0;
LATCH_READY = 1;
while (LATCH_CLOCKED == 0);
LATCH_CLOCKED = 0;
}
}
void
main(void)
{
     TRISB0 = 0;       //set port b bits 0-3 as output
     TRISB1 = 0;
     TRISB2 = 0;
     TRISB3 = 0;

     T0IE = 1;         // Enable interrupt on TMR0 overflow
     T0CS = 0;         // use internal clock
     PSA = 0;          // prfescaler used for timer0
     PS0 = 0;          // prescaler set to 1:16
     PS1 = 1;
     PS2 = 0;
     INTEDG = 1;       // falling edge trigger the interrupt
     INTE = 0;         // enable the external interrupt
     GIE = 1;          // Global interrupt enable

putoptrex(4,5,6,7);    //call display routine
     for(;;){    // Idly kick the dog

     }
}
```