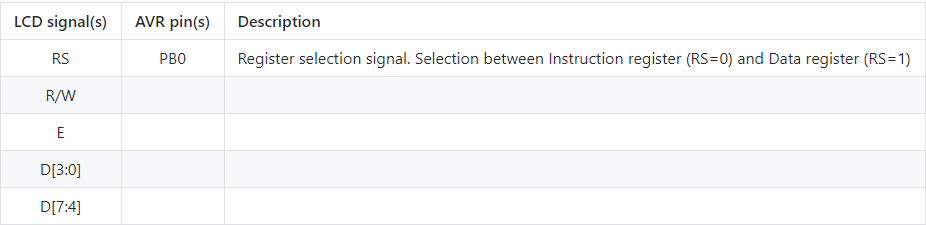
**DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 6**

Name: Demirkan Korbey Baglamac Github Repository Link:

1. What data and control signals are used? What is the meaning of these signals?

sent each symbol (word etc.) in two halves.

PD7:PD4 Data signals, we use only these on 4-bit communcation and since the data is 8bit we sent

- Data signals, but since we are using 4-bit communcation we dont use them.

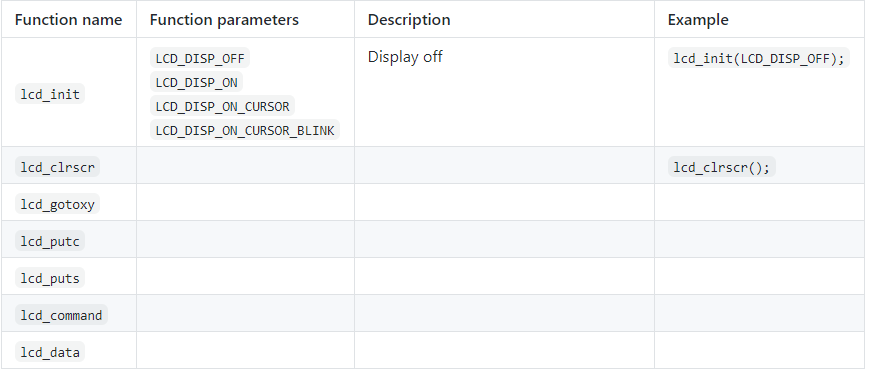
PB1 Enable Signal. We read the signal when this is 1.

GND If it is 1 write data signal, if it is 0 read data signal.

What is the ASCII table? What are the values for uppercase letters A to Z, lowercase letters a to z, and numbers 0 to 9 in this table?

ASCII is American Standard Code for Information Interchange.

|  |  |  |
| --- | --- | --- |
| **HEXADECIMAL** | **BINARY** | **SYMBOL** |
| 30 | 00110000 | 0 |
| 31 | 00110001 | 1 |
| 32 | 00110010 | 2 |
| 33 | 00110011 | 3 |
| 34 | 00110100 | 4 |
| 35 | 00110101 | 5 |
| 36 | 00110110 | 6 |
| 37 | 00110111 | 7 |
| 38 | 00111000 | 8 |
| 39 | 00111001 | 9 |
| 41 | 01000001 | A |
| 42 | 01000010 | B |
| 43 | 01000011 | C |
| 44 | 01000100 | D |
| 45 | 01000101 | E |
| 46 | 01000110 | F |
| 47 | 01000111 | G |
| 48 | 01001000 | H |
| 49 | 01001001 | I |
| 4A | 01001010 | J |
| 4B | 01001011 | K |
| 4C | 01001100 | L |
| 4D | 01001101 | M |
| 4E | 01001110 | N |
| 4F | 01001111 | O |
| 50 | 01010000 | P |
| 51 | 01010001 | Q |
| 52 | 01010010 | R |
| 53 | 01010011 | S |
| 54 | 01010100 | T |
| 55 | 01010101 | U |
| 56 | 01010110 | V |
| 57 | 01010111 | W |
| 58 | 01011000 | X |
| 59 | 01011001 | Y |
| 5A | 01011010 | Z |
| 61 | 01100001 | a |
| 62 | 01100010 | b |
| 63 | 01100011 | c |
| 64 | 01100100 | d |
| 65 | 01100101 | e |
| 66 | 01100110 | f |
| 67 | 01100111 | g |
| 68 | 01101000 | h |
| 69 | 01101001 | i |
| 6A | 01101010 | j |
| 6B | 01101011 | k |
| 6C | 01101100 | l |
| 6D | 01101101 | m |
| 6E | 01101110 | n |
| 6F | 01101111 | o |
| 70 | 01110000 | p |
| 71 | 01110001 | q |
| 72 | 01110010 | r |
| 73 | 01110011 | s |
| 74 | 01110100 | t |
| 75 | 01110101 | u |
| 76 | 01110110 | v |
| 77 | 01110111 | w |
| 78 | 01111000 | x |
| 79 | 01111001 | y |
| 7A | 01111010 | z |

1. 

void Clear display and set cursor to home pos. .position.

uint8\_t x, uint8\_t y Set cursor to specified position lcd\_gotoxy(0,0);

x: horizontal position, y: vertical pos.

x horizontal position

(0: left most position)

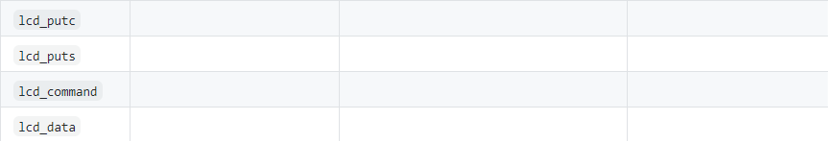
y vertical position

(0: first line)

|  |  |
| --- | --- |
| **x** | horizontal position (0: left most position) |
| **y** | vertical position (0: first line) |

Display on, Cursor off Display on, Cursor on Display on, Cursor on flashing

lcd\_init(LCD\_DISP\_ON); lcd\_init(LCD\_DISP\_ON\_CURSOR); lcd\_init(LCD\_DISP\_ON\_CURSOR\_BLINK);



uint8\_t cmd Send LCD controller instruction command. lcd\_command(1<<LCD\_CGRAM);

uint8\_t data Send data byte to LCD controller. lcd\_data(customChar[i]);  
;

char c Display character at current cursor. lcd\_putc(“c”);

const char \* c Display string without auto linefeed. lcd\_puts(“hello”);

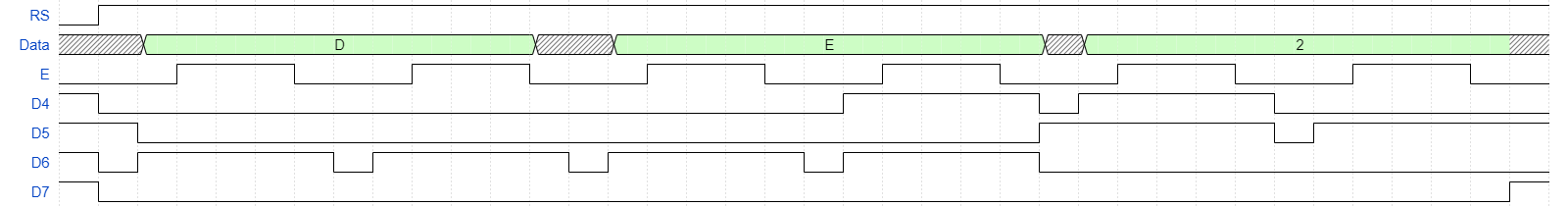


Table 1: Waveform for DE2

1. Listing of TIMER2\_OVF\_vect interrupt routine with complete stopwatch code (minutes:seconds.tenths) and square value computation,

/\*\*

\* ISR starts when Timer/Counter2 overflows. Update the stopwatch on

\* LCD display every sixth overflow, ie approximately every 100 ms

\* (6 x 16 ms = 100 ms).

\*/

ISR(TIMER2\_OVF\_vect)

{

static *uint8\_t* number\_of\_overflows = 0;

static *uint8\_t* tens = 0; // Tenths of a second

static *uint8\_t* secs = 0; // Seconds

static *uint8\_t* mins = 0; // Minutes

char lcd\_string[2] = " "; // String for converting numbers by itoa()

char lcd\_longstring[4];

number\_of\_overflows++;

if (number\_of\_overflows >= 6)

{

// Do this every 6 x 16 ms = 100 ms

number\_of\_overflows = 0;

// Update the tenths of a second

tens++;

if (tens >= 10)

{

tens = 0;

//Update the seconds

secs++;

if(secs >= 60)

{

secs = 0;

// Update the minutes

mins++;

if(mins >= 60)

mins = 0;

// Display minutes

lcd\_gotoxy(1,0);

if(mins < 10)

lcd\_putc('0');

*itoa*(mins, lcd\_string, 10);

lcd\_puts(lcd\_string);

}

//Display Seconds

lcd\_gotoxy(4,0);

if(secs < 10)

lcd\_putc('0');

*itoa*(secs, lcd\_string, 10);

lcd\_puts(lcd\_string);

// Display the square value of the Seconds

lcd\_gotoxy(11, 0);

*itoa*((secs \* secs), lcd\_longstring, 10);

lcd\_puts(lcd\_longstring);

if(secs == 0)

lcd\_puts(" ");

}

// Display hundredths of seconds

lcd\_gotoxy(7,0);

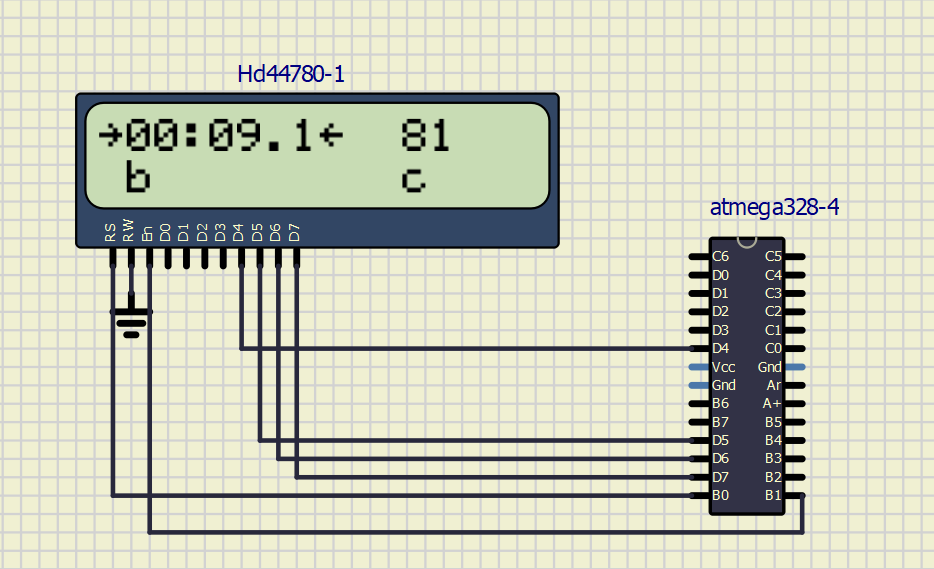
// Converting cnt0 in decimal to string

*itoa*(tens, lcd\_string, 10);

lcd\_puts(lcd\_string);

}

}

Screenshot of the circuit:

1. Listing of TIMER0\_OVF\_vect interrupt routine with a progress bar,

/\*\*

\* ISR starts when Timer/Counter0 overflows. Update the progress bar on

\* LCD display every 16 ms.

\*/

ISR(TIMER0\_OVF\_vect)

{

static *uint8\_t* symbol = 0;

static *uint8\_t* position = 0;

lcd\_gotoxy(1 + position, 1);

lcd\_putc(symbol);

symbol++;

if(symbol >= 6)

{

symbol = 0;

position++;

if(position >= 10) {

position = 0;

lcd\_gotoxy(1,1);

for(*uint8\_t* i; i < 10; i++) {

lcd\_putc(0);

}

}

}

}

Screenshot of the circuit:

