EIE3105: LCD
(Chapter 13)

Dr. Lawrence Cheung

Semester 1, 2021/22

Topics

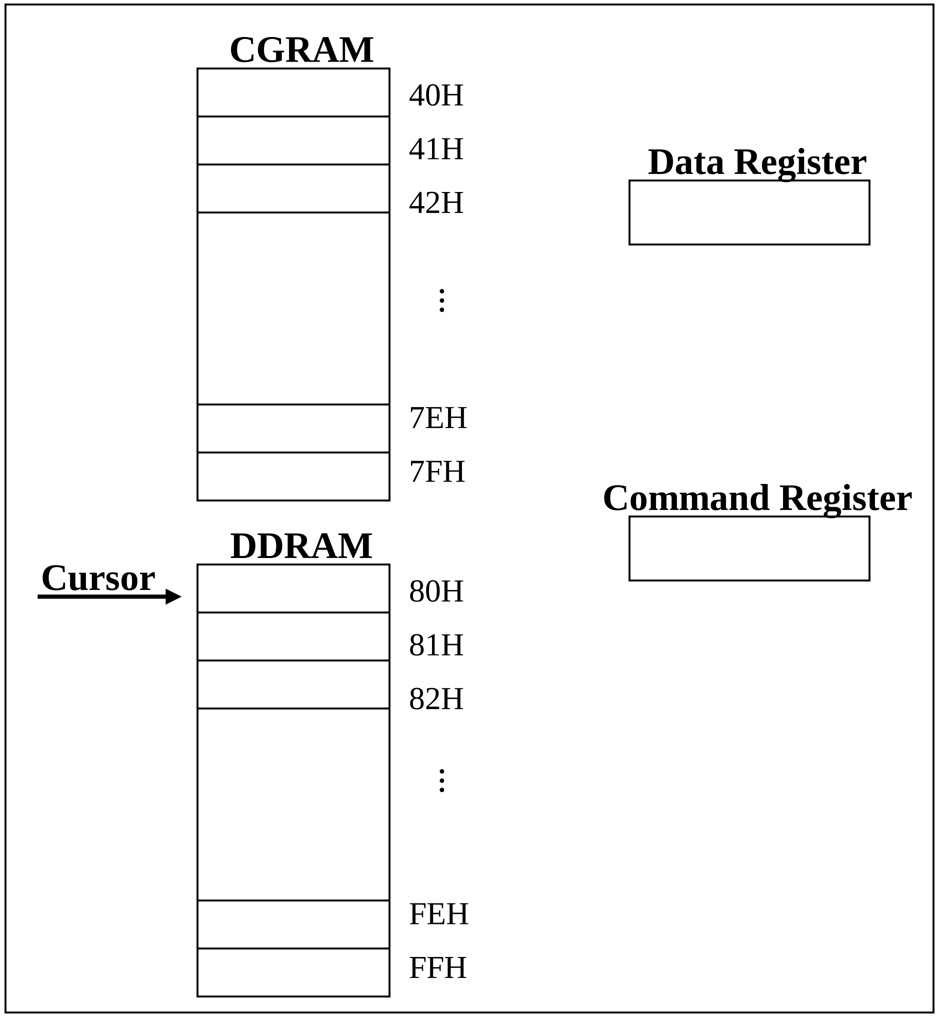
- LCD
  - LCD pin out
  - LCD internal components
  - Writing to data register (example)
  - LCD commands
  - LCD programming

LCD

- Sometimes the embedded system needs to inform the user of something. There are different ways to inform the user, such as LEDs, 7segments and LCDs.
- LCD is one of the most powerful ways; as you can display different texts and icons on it.

LCD internal components

- DDRAM (Data Display RAM)
- CGRAM (Character Generator RAM)
- Cursor (Address Counter)
- Data Register
- Command Register



LCD internal components

- DDRAM (Data Display RAM)
  - It is a 128 x 8 RAM (128 bytes of RAM)
  - Contains the data that should be displayed on the LCD.
  - If we write the ASCII code of a character into the RAM the character will be displayed on the LCD.
- CGRAM (Character Generator RAM)
  - It is a 64x8 RAM (64 bytes of RAM).
  - The fonts of characters 00H to 07H are stored in the RAM.
  - We can change the fonts of the 8 characters by writing into the RAM.

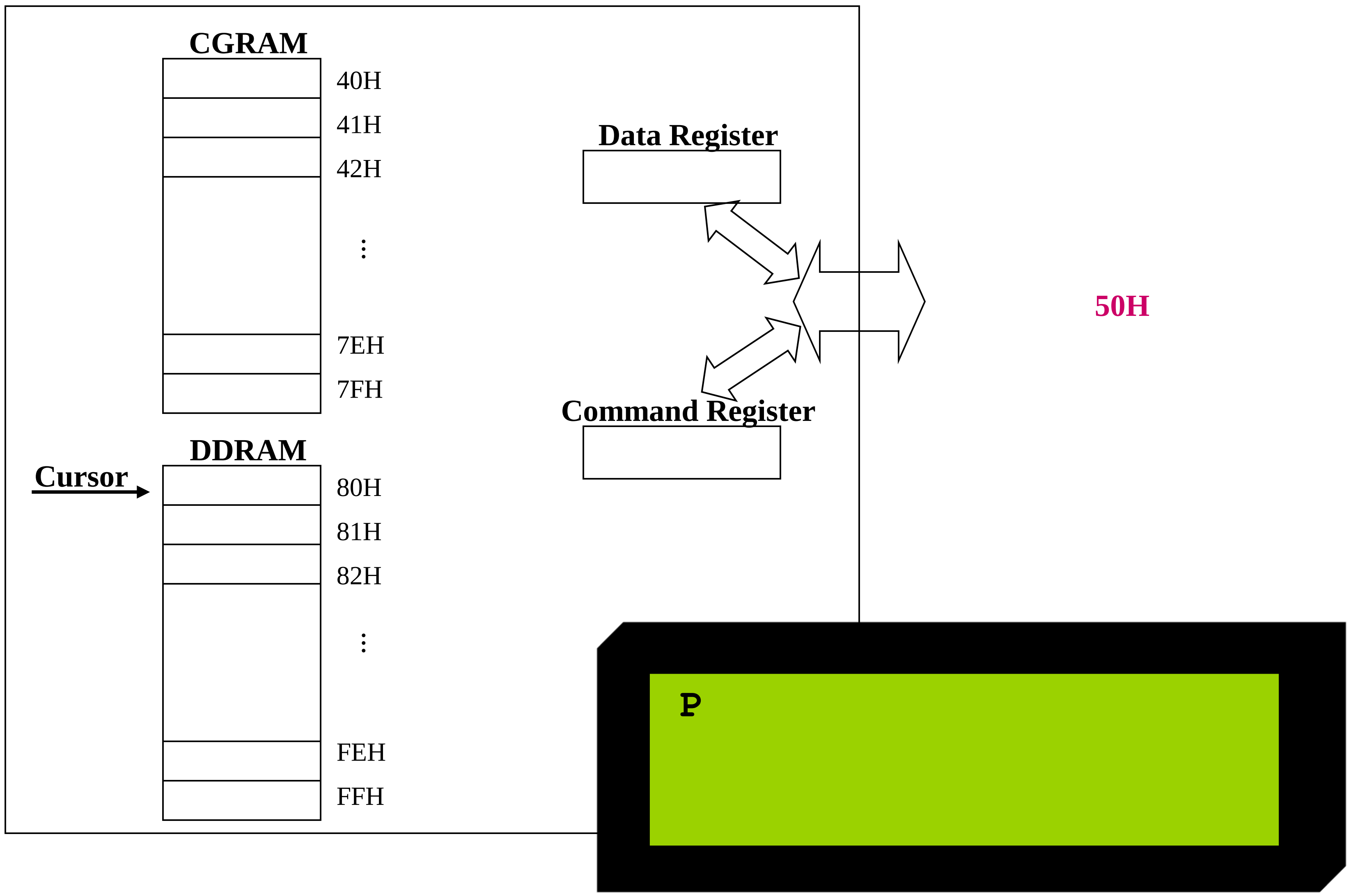
LCD internal components

- Cursor (Address Counter)
  - Cursor is a register which points to a location of DDRAM or CGRAM.
- Data Register
  - It is an 8 bit register.
  - When we write a byte of data into the data register, the data will be written where the cursor points to.
  - For example, if we write a byte of data into the data register while the cursor points to location 80H of DDRAM, the contents of location 80H will be changed to the data, we have written into the data register.

LCD internal components

- Command Register
  - We can command the LCD by writing into the command register.
  - For example, we can ask the LCD, to set the cursor location, or clean the screen, by writing into the command register.

Writing to data register (example)



LCD commands

- We mentioned earlier that we can order the LCD by sending command codes to the command register.
- Some command codes are listed below:

Code (Hex)	Instruction	Code (Hex)	Instruction
1	Clear display screen	**2**	Return home
10	Shift cursor position to left	**14**	Shift cursor position to right
18	Shift display left	**1C**	Shift display right
4	After displaying a character on the LCD, shift cursor to left	**6**	After displaying a character on the LCD, shift cursor to right
80-FF	Set cursor position	**40-7F**	Set CG RAM address
8	Display off, cursor off	**A**	Display off, cursor on
C	Display on, cursor off	**E**	Display on, cursor on
F	Display on, cursor blinking	**38**	Initializing to 2 lines & 5x7 font

Clear Display Screen

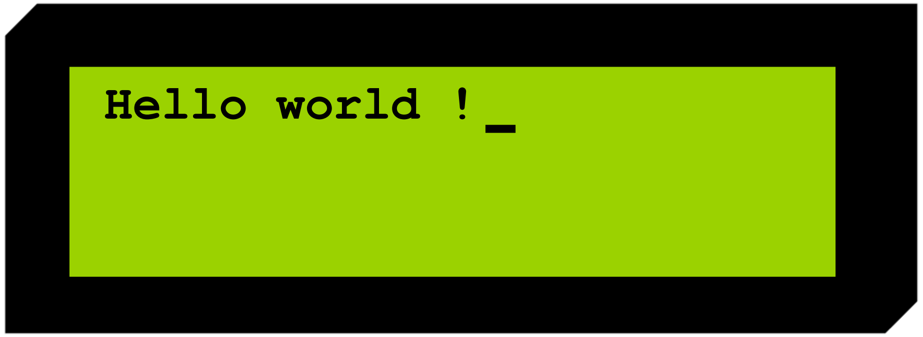
- If we write 01H into the command register, LCD clears the display, and sets the cursor address to 0.

Display and Cursor

Display on cursor blinking (0FH)

Hello world !

Display on cursor on (0EH)



Display on cursor off (0CH)

Hello world !

Display off cursor off (0AH)

Return home

- If we write 02H into the command register, LCD sets the cursor address to 0. It also returns the display to original position if being shifted.

Set cursor position

- We mentioned earlier that each location of the DDRAM, retains the character that should be displayed in a location of LCD.
- The following figures, represent that if you want to display a character in each of the rooms of the LCD, you should write into which location of the DDRAM. (The numbers are in hex.)
- To move the cursor to any location of the DDRAM, write the address of that location into the command register.

Set cursor position

	1	2	3	…	18	19	20
Line 1	80	81	82	…	91	92	93
Line 2	C0	C1	C2	…	D1	D2	D3
Line 3	94	95	96	…	A5	A6	A7
Line 4	D4	D5	D6	…	E5	E6	E7

20x4 LCD

	1	2	3	…	18	19	20
Line 1	80	81	82	…	91	92	93

20x1 LCD

	1	2	3	…	18	19	20
Line 1	80	81	82	…	91	92	93
Line 2	C0	C1	C2	…	D1	D2	D3

20x2 LCD

	1	2	3	…	38	39	40
Line 1	80	81	82	…	A5	A6	A7
Line 2	C0	C1	C2	…	E5	E6	E7

40x2 LCD

	1	2	3	…	14	15	16
Line 1	80	81	82	…	8D	8E	8F
Line 2	C0	C1	C2	…	CD	CE	CF

16x2 LCD

Set cursor position

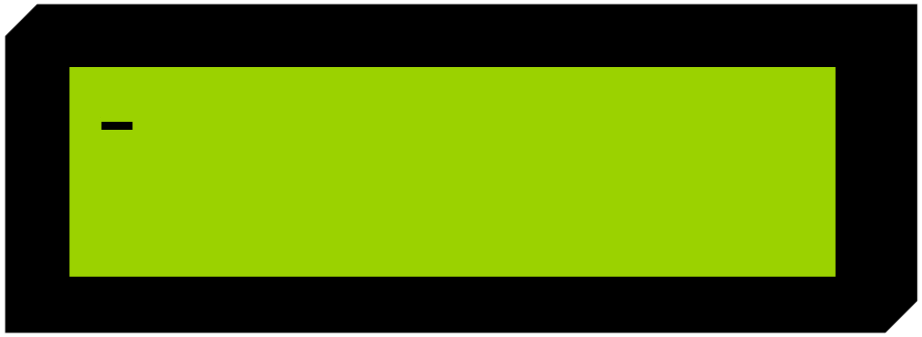
- Example: We want to display a character in line 4 column 1 of a 20 x 4 LCD. What should we write to the command register to move the cursor to?

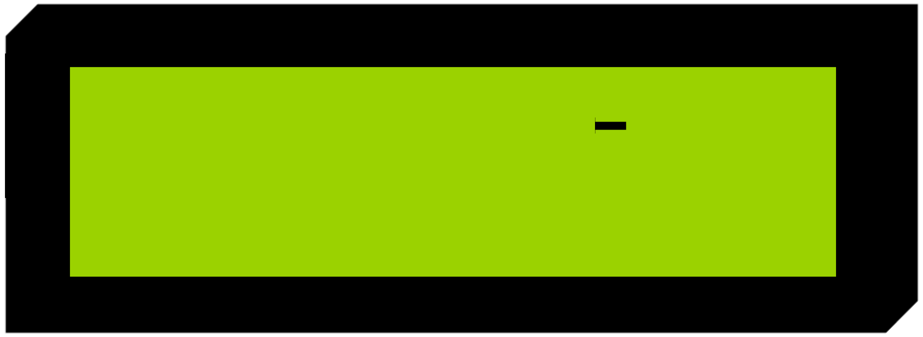
Solution: We should move the cursor to address D4H of the DDRAM. So, we should write D4H, into the command register.

	1	2	3	…	18	19	20
Line 1	**80**	**81**	**82**	…	**91**	**92**	**93**
Line 2	**C0**	**C1**	**C2**	…	**D1**	**D2**	**D3**
Line 3	**94**	**95**	**96**	…	**A5**	**A6**	**A7**
Line 4	**D4**	**D5**	**D6**	…	**E5**	**E6**	**E7**

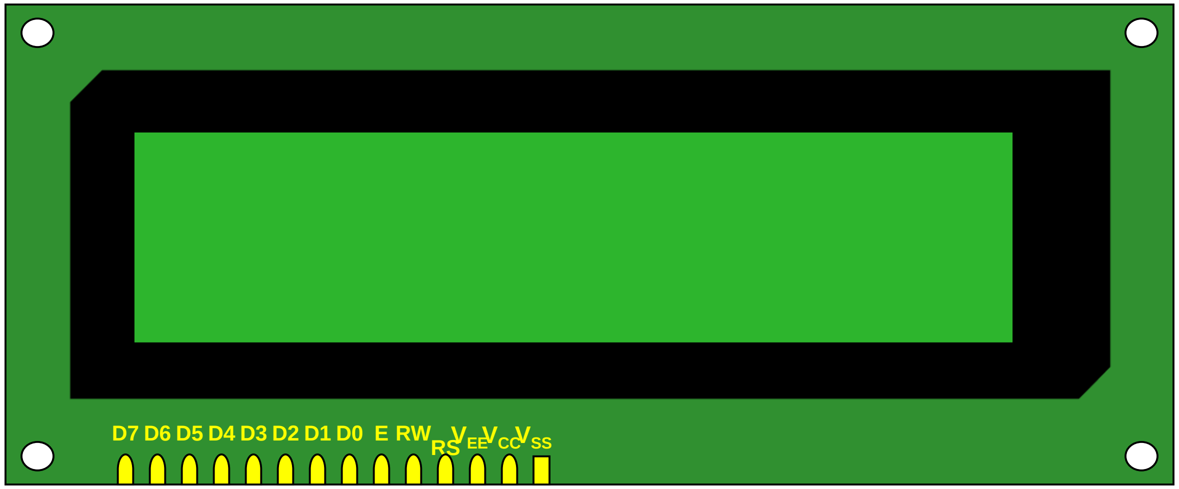
Decrease and increase Cursor

- If you write a byte of data into the data register, the data will be written where the cursor points to, and cursor will be incremented, by default.
  - If you want to make the LCD, to decrement the cursor, you should write 4H into the command register.
  - If you want to make the LCD, to reactivate the default (shift cursor to right) you should write 6H into the command register.


Increment cursor

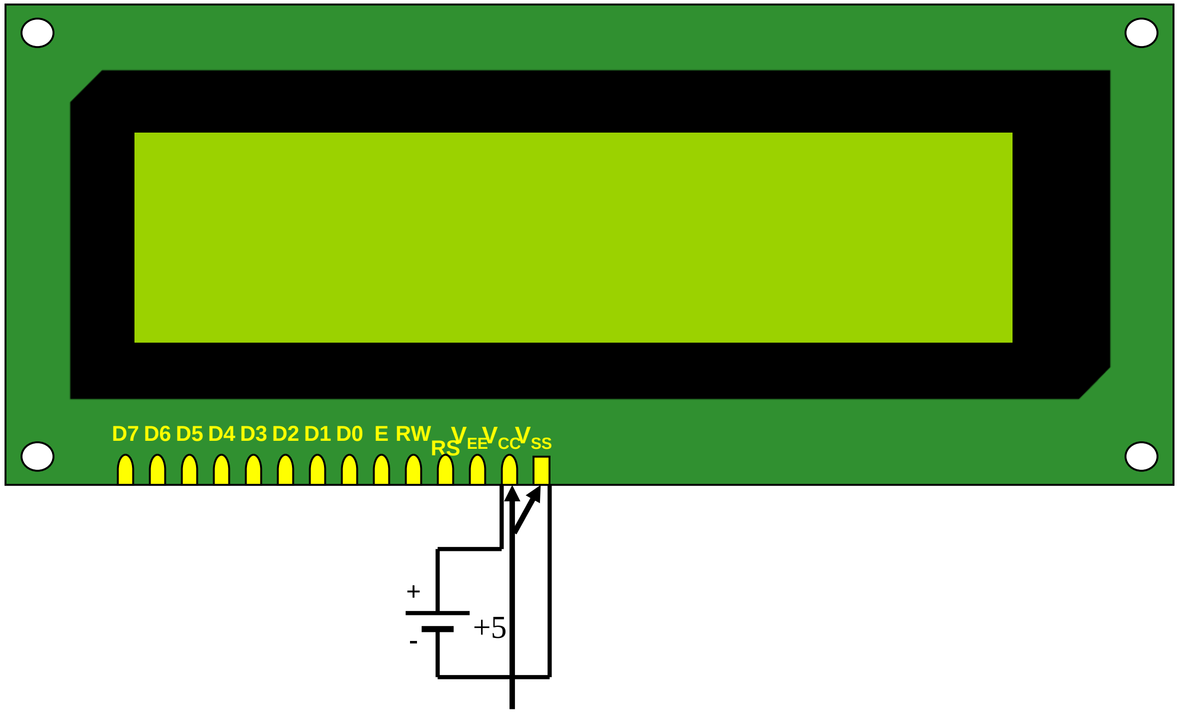

Decrement cursor

LCD pins



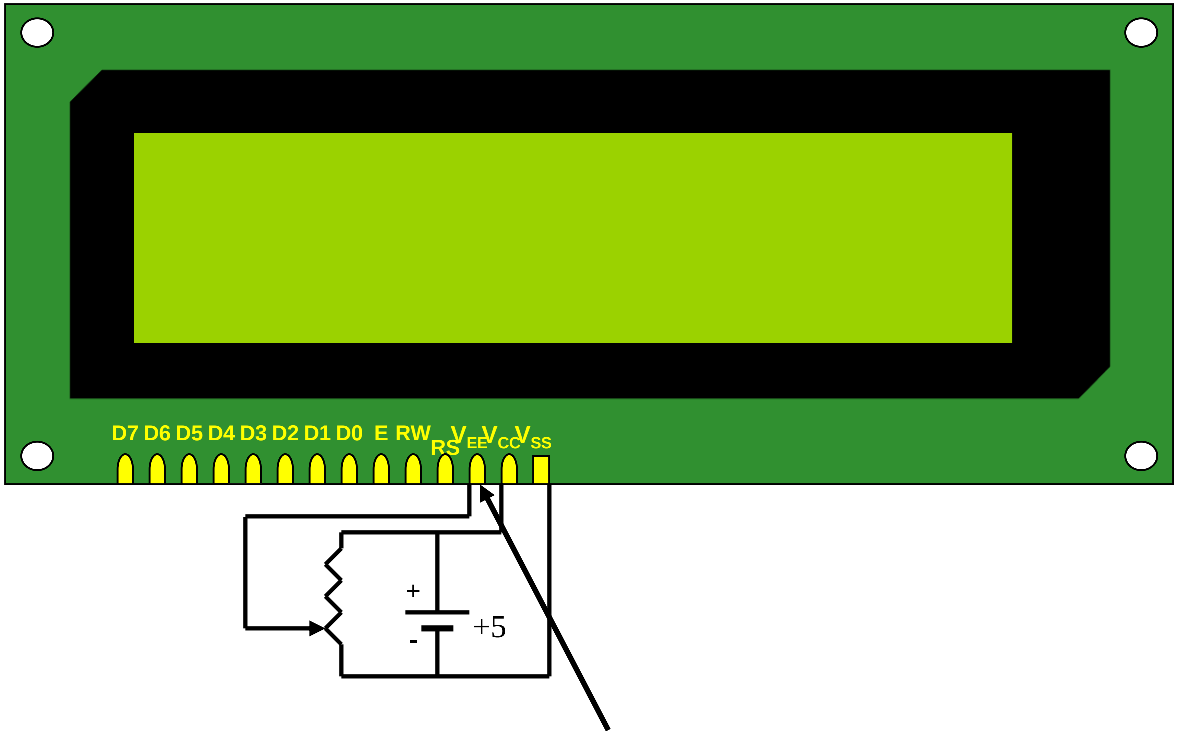
- In this section, you learn the functionalities of the LCD pins.

LCD pins



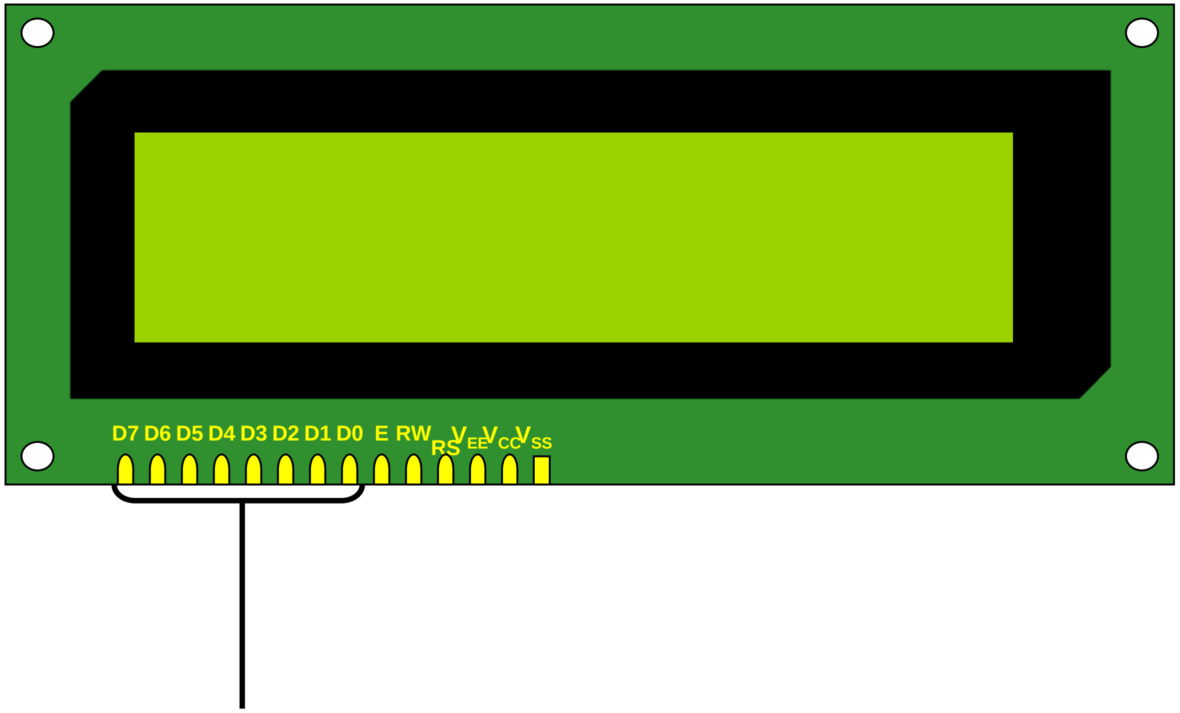
- V_{SS} and V_{CC}: These pins provide the energy to the LCD. We must connect them to +5V.

LCD pins



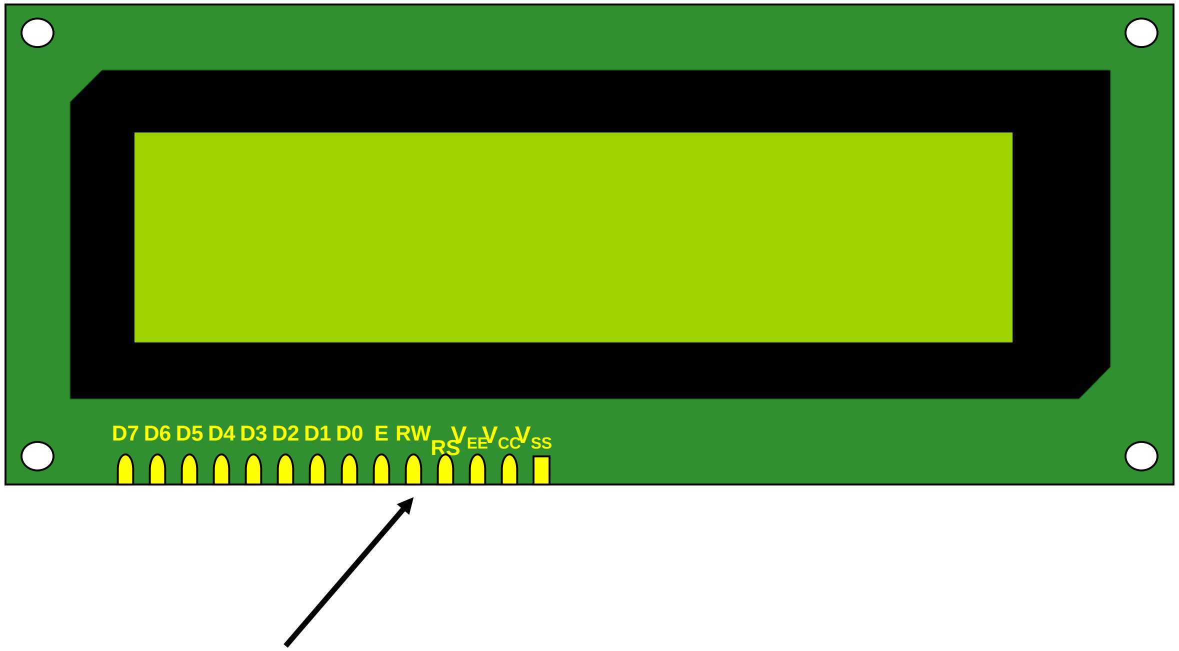
- V_{EE}: We control the contrast of the LCD by giving a voltage between 0V and +5V to the pin.

LCD pins



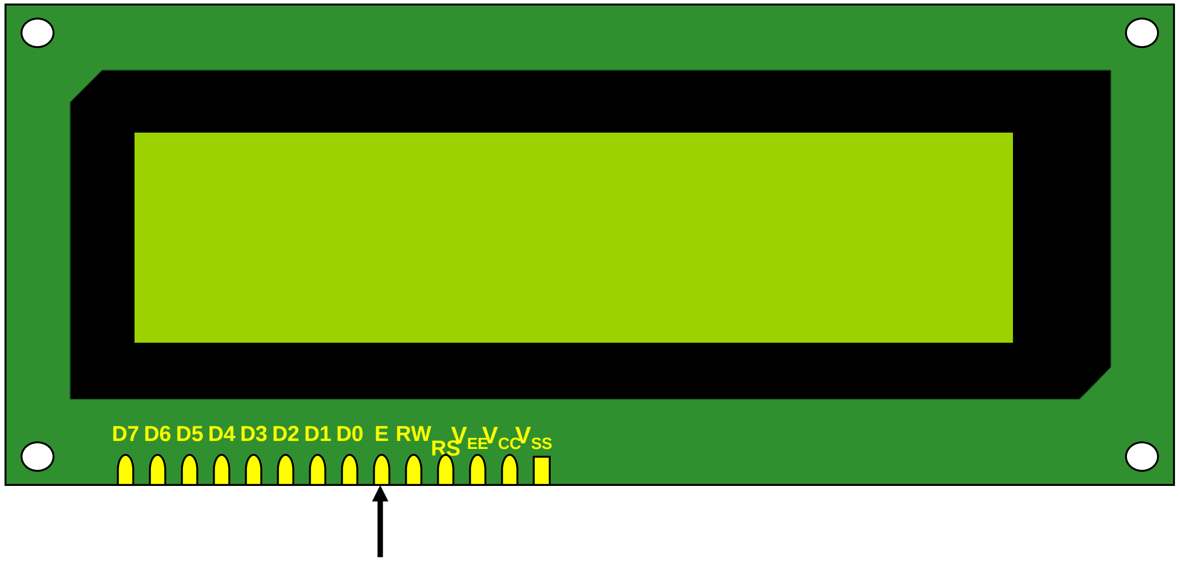
- D0 to D7: LCD sends and receives data, through the 8 pins.

LCD pins

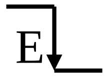


- R/W (Read/Write):
  - When we want to send (write) data to the LCD, we make the pin low.
  - When we want to receive (read) data from the LCD, we set the pin to high.

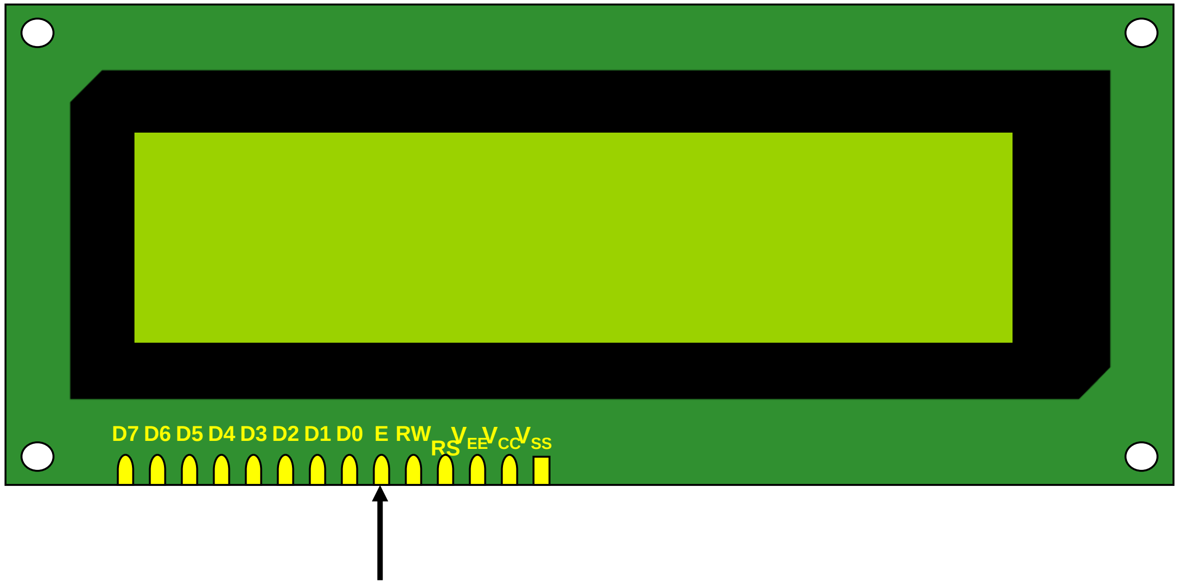
LCD pins

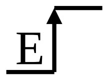


- E (Enable): We activate the pin when we want to send or receive data from the LCD.
  - When we want to send data to the LCD, we make the RW pin, low; and supply the data to data pins (D0 to D7); and then apply a high to low pulse to the **E**nable pin.



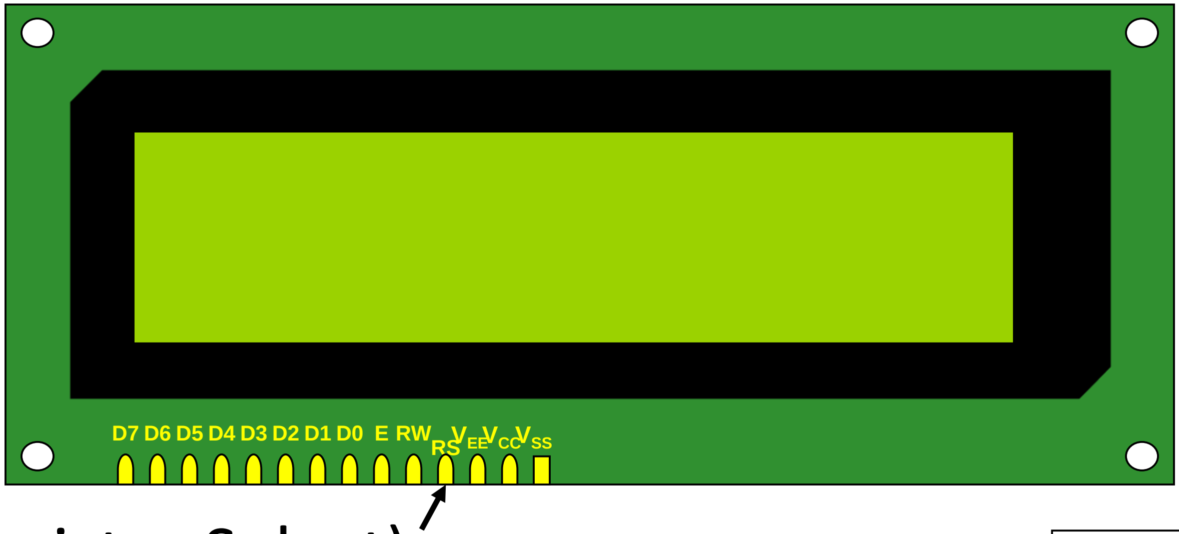
LCD pins



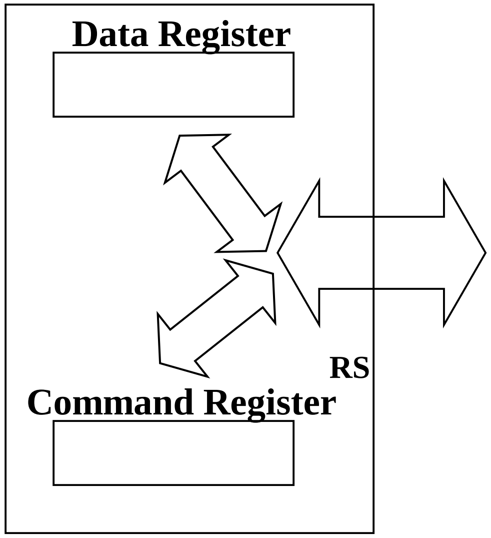


- When we want to receive data from the LCD, we make the RW pin, high; and then apply a low to high pulse to the **E**nable pin. LCD supplies data to the data pins (D0 to D7).

LCD pins



- RS (Register Select):
  - If RS = 1, the data will be located in the data register.
  - If RS = 0, the data will be located in the command register.



LCD programming

- Initialization
  - We must initialize the LCD before we use it.
  - To initialize an LCD, for 5 × 7 matrix and 8-bit operation, 0 x 38, 0 x 0E, and 0 x 01 are sent to the command register.

LCD programming

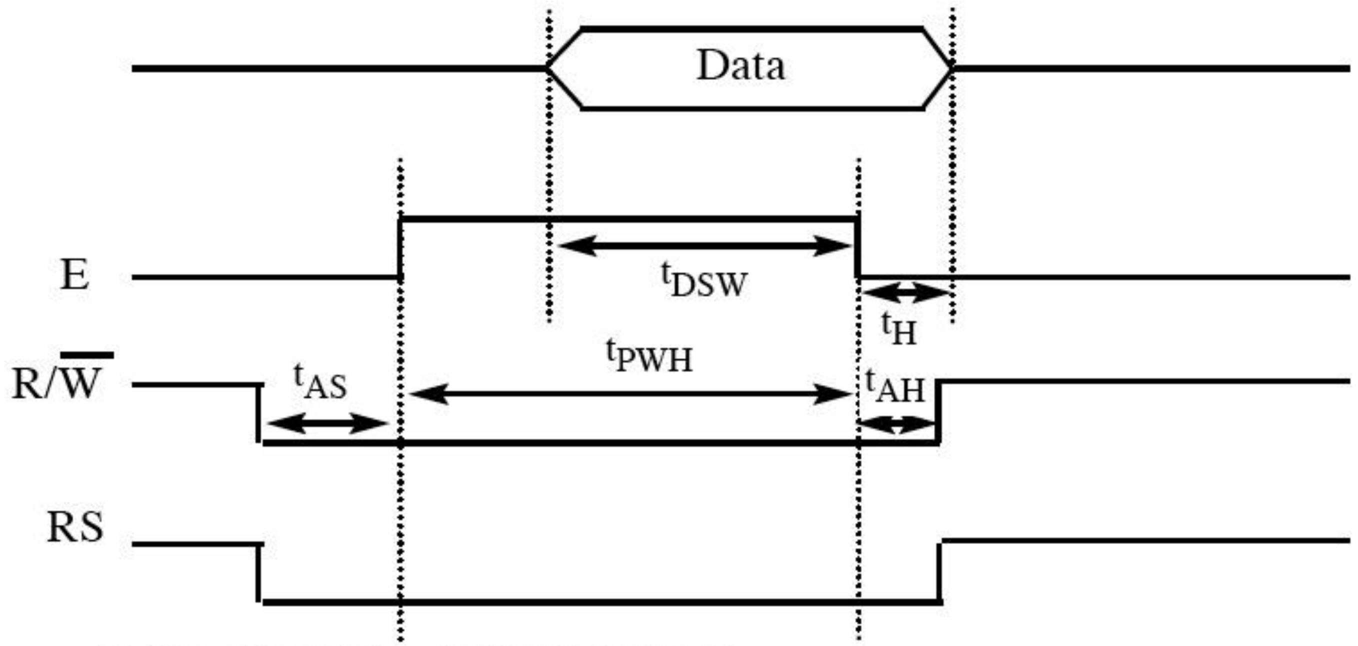
- Sending commands to the LCD
  - Make pins RS = 0 and R/W = 0
  - Put the command code on the data pins (D0–D7)
  - Send a high-to-low pulse to the E pin to enable the internal latch of the LCD (wait about 100 μs after each command)

LCD programming

- Sending data to the LCD
  - Make pins RS = 1 and R/W = 0.
  - Put the data on the data pins (D0–D7)
  - Send a high-to-low pulse to the E pin (wait about 100 μs).

LCD programming

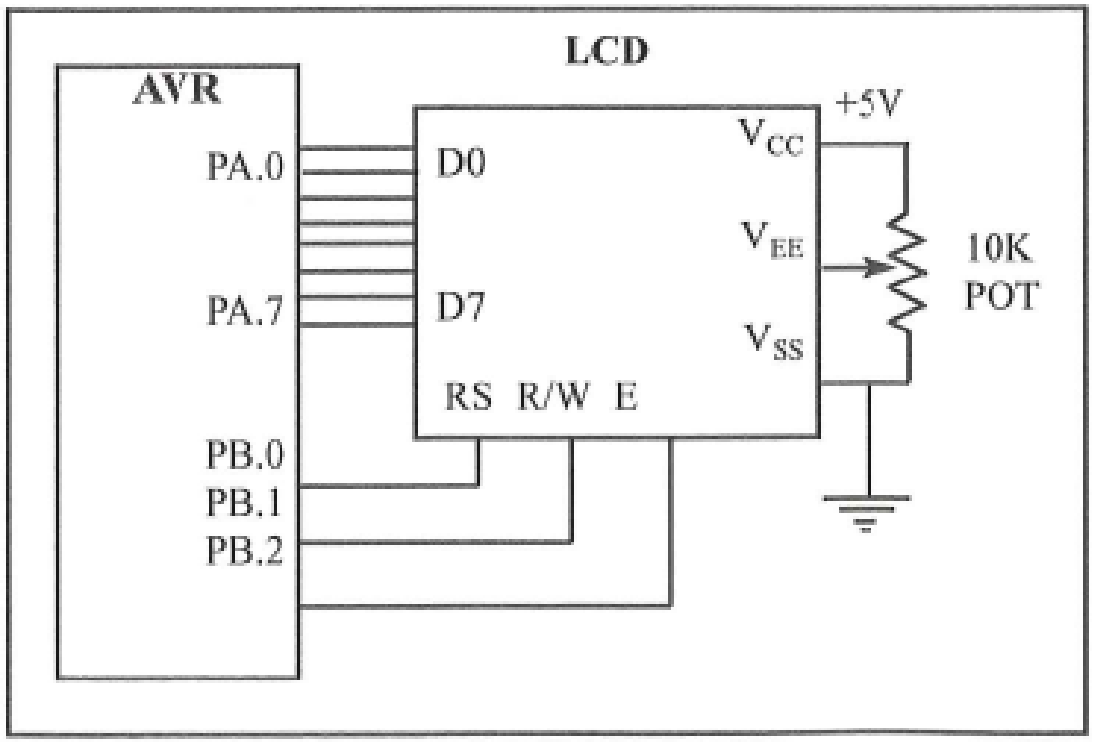
- Timing diagram



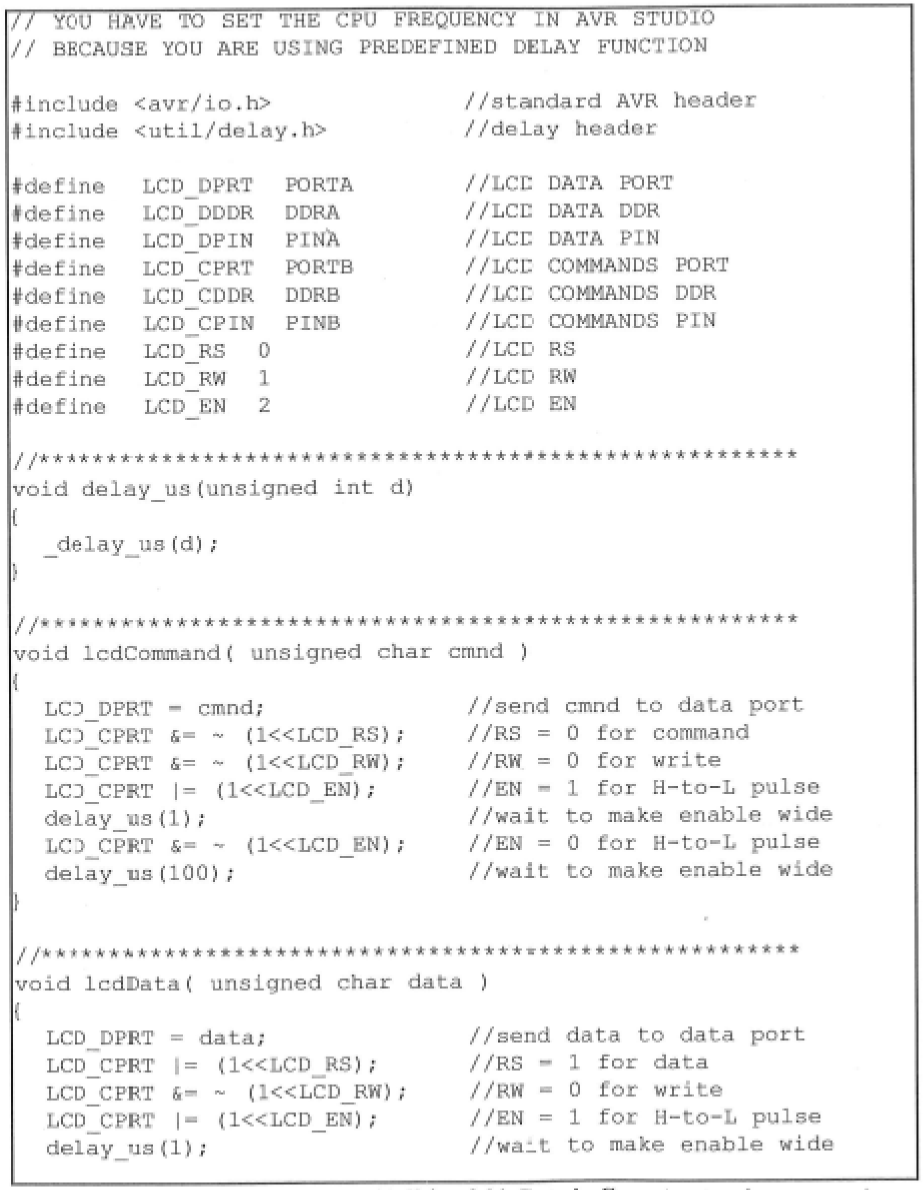
t_{PWH} = Enable pulse width = 450 ns (minimum)
t_{DSW} = Data setup time = 195 ns (minimum)
t_H = Data hold time = 10 ns (minimum)
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

LCD programming

- Block diagram

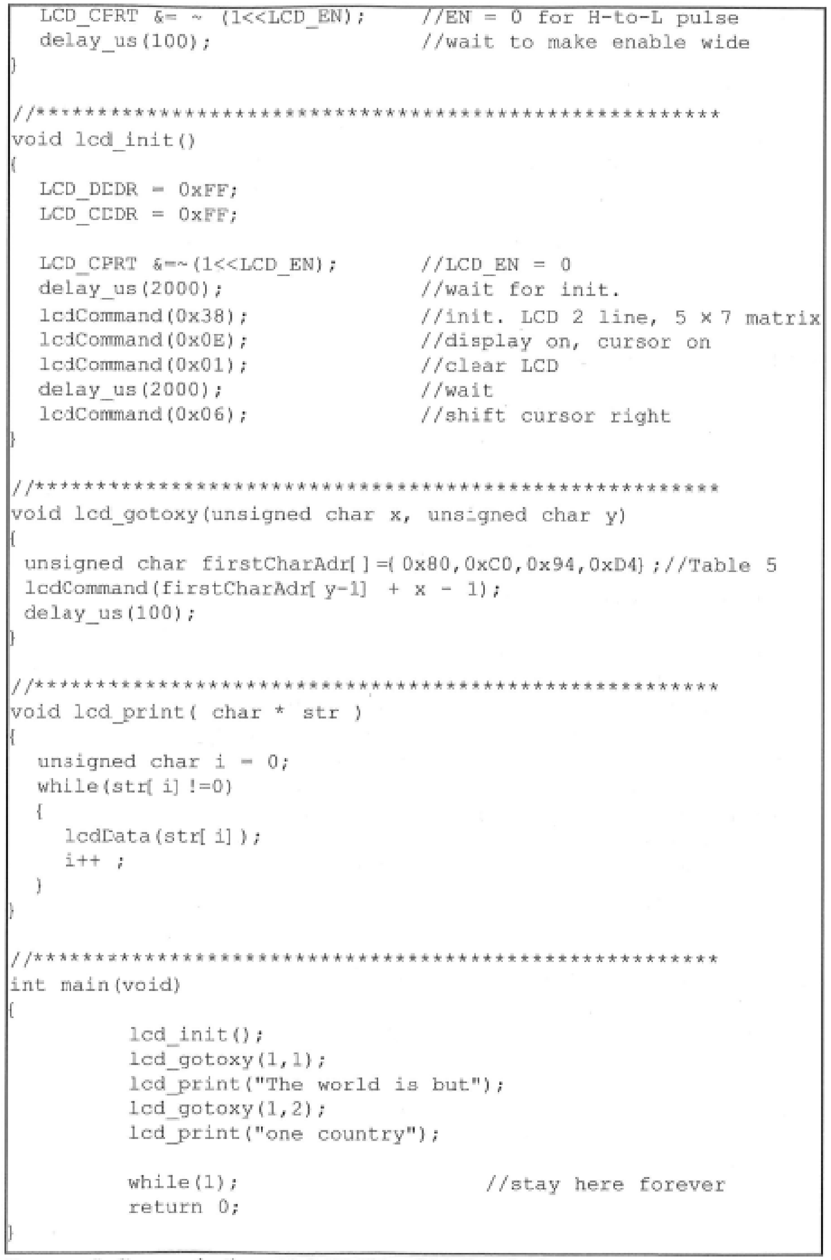


LCD programming



```
// YOU HAVE TO SET THE CPU FREQUENCY IN AVR STUDIO
// BECAUSE YOU ARE USING PREDEFINED DELAY FUNCTION

#include <avr/io.h>              //standard AVR header
#include <util/delay.h>          //delay header

#define   LCD_DPRT   PORTA       //LCD DATA PORT
#define   LCD_DDDR   DDRA        //LCD DATA DDR
#define   LCD_DPIN   PINA        //LCD DATA PIN
#define   LCD_CPRT   PORTB       //LCD COMMANDS PORT
#define   LCD_CDDR   DDRB        //LCD COMMANDS DDR
#define   LCD_CPIN   PINB        //LCD COMMANDS PIN
#define   LCD_RS   0             //LCD RS
#define   LCD_RW   1             //LCD RW
#define   LCD_EN   2             //LCD EN

//*******************************************************
void delay_us(unsigned int d)
{
  _delay_us(d);
}

//*******************************************************
void lcdCommand( unsigned char cmnd )
{
  LCD_DPRT = cmnd;                //send cmnd to data port
  LCD_CPRT &= ~ (1<<LCD_RS);      //RS = 0 for command
  LCD_CPRT &= ~ (1<<LCD_RW);      //RW = 0 for write
  LCD_CPRT |= (1<<LCD_EN);        //EN = 1 for H-to-L pulse
  delay_us(1);                    //wait to make enable wide
  LCD_CPRT &= ~ (1<<LCD_EN);      //EN = 0 for H-to-L pulse
  delay_us(100);                  //wait to make enable wide
}

//*******************************************************
void lcdData( unsigned char data )
{
  LCD_DPRT = data;                //send data to data port
  LCD_CPRT |= (1<<LCD_RS);        //RS = 1 for data
  LCD_CPRT &= ~ (1<<LCD_RW);      //RW = 0 for write
  LCD_CPRT |= (1<<LCD_EN);        //EN = 1 for H-to-L pulse
  delay_us(1);                    //wait to make enable wide
```

LCD programming



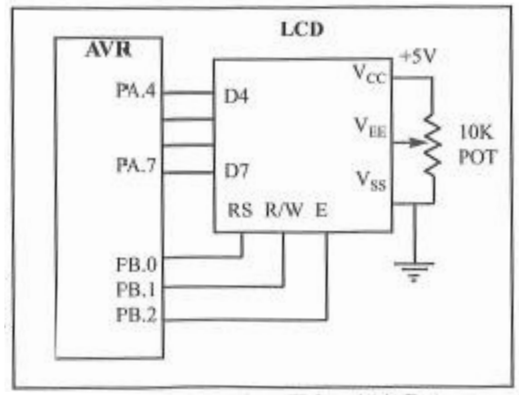
```
  LCD_CPRT &= ~ (1<<LCD_EN);      //EN = 0 for H-to-L pulse
  delay_us(100);                  //wait to make enable wide
}

//*********************************************************
void lcd_init()
{
  LCD_DDDR = 0xFF;
  LCD_CDDR = 0xFF;

  LCD_CPRT &=~(1<<LCD_EN);        //LCD_EN = 0
  delay_us(2000);                 //wait for init.
  lcdCommand(0x38);               //init. LCD 2 line, 5 × 7 matrix
  lcdCommand(0x0E);               //display on, cursor on
  lcdCommand(0x01);               //clear LCD
  delay_us(2000);                 //wait
  lcdCommand(0x06);               //shift cursor right
}

//*********************************************************
void lcd_gotoxy(unsigned char x, unsigned char y)
{
 unsigned char firstCharAdr[]={0x80,0xC0,0x94,0xD4};//Table 5
 lcdCommand(firstCharAdr[y-1] + x - 1);
 delay_us(100);
}

//*********************************************************
void lcd_print( char * str )
{
  unsigned char i = 0;
  while(str[i]!=0)
  {
    lcdData(str[i]);
    i++ ;
  }
}

//*********************************************************
int main(void)
{
         lcd_init();
         lcd_gotoxy(1,1);
         lcd_print("The world is but");
         lcd_gotoxy(1,2);
         lcd_print("one country");

         while(1);                      //stay here forever
         return 0;
}
```

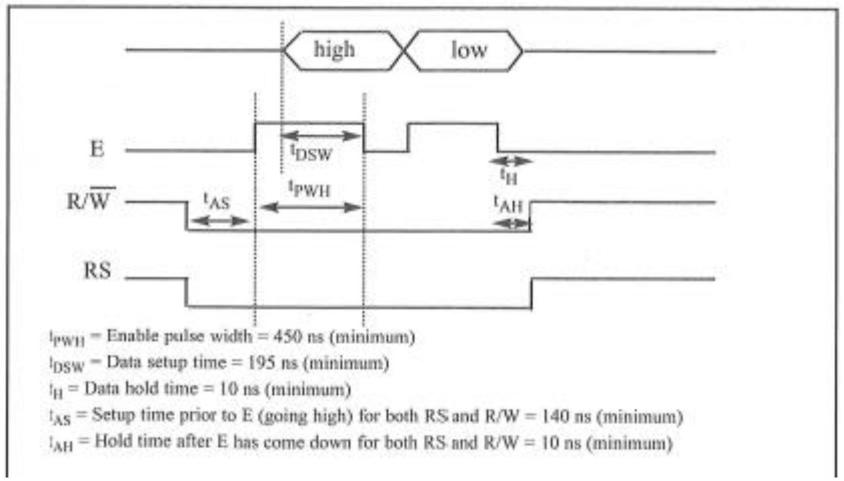
LCD programming in 4-bit mode

- To save pins of the AVR, we can use 4-bit operating mode.
- The initialization of 4-bit mode is somehow different:
  - In 4-bit mode, we initialize the LCD with the series 33, 32, and 28 in hex.
    - This represents nibbles 3, 3, 3, and 2, which tells the LCD to go into 4-bit mode. The value $28 initializes the display for 5 × 7 matrix and 4-bit operation.

LCD programming in 4-bit mode

- Sending commands and data to the LCD
  - Sending data and commands to the LCD is like the 8-bit mode but we should only use D4 – D7.
  - First we should send the high nibble to D4-D7, then, to send the low nibble, swap the low nibble with the high nibble, and send it to D4-D7.

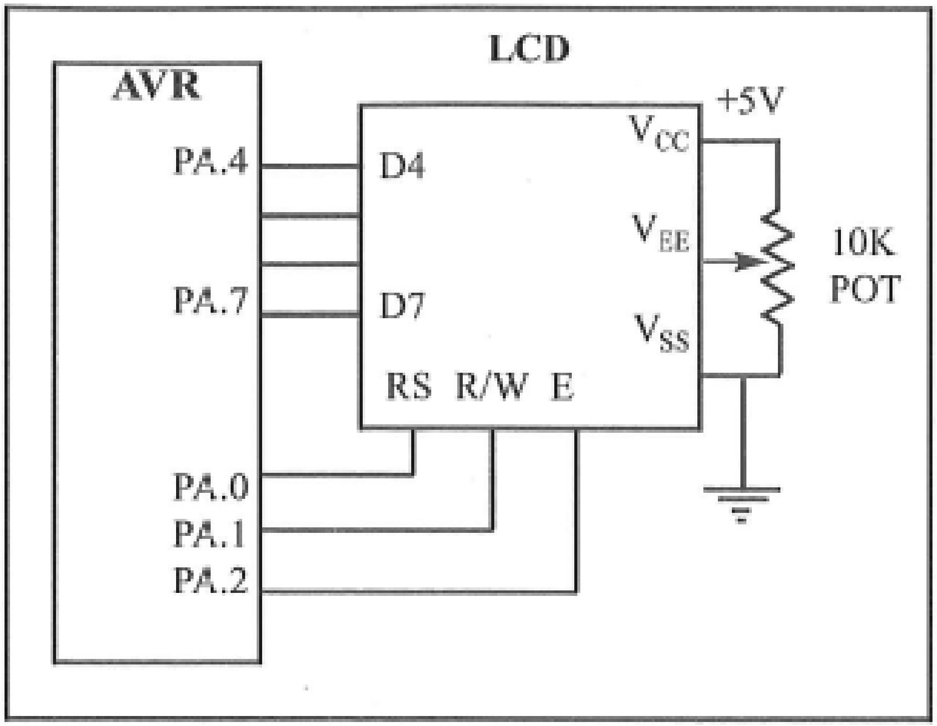




t_{PWH} = Enable pulse width = 450 ns (minimum)
t_{DSW} = Data setup time = 195 ns (minimum)
t_H = Data hold time = 10 ns (minimum)
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

LCD programming in a single port

- Sending commands and data to the LCD by using a single port.



Reference Readings

- Chapter 13 – *The AVR Microcontroller and Embedded Systems : Using Assembly and C*, M. A. Mazidi, S. Naimi, and S. Naimi, Pearson, 2014.

End