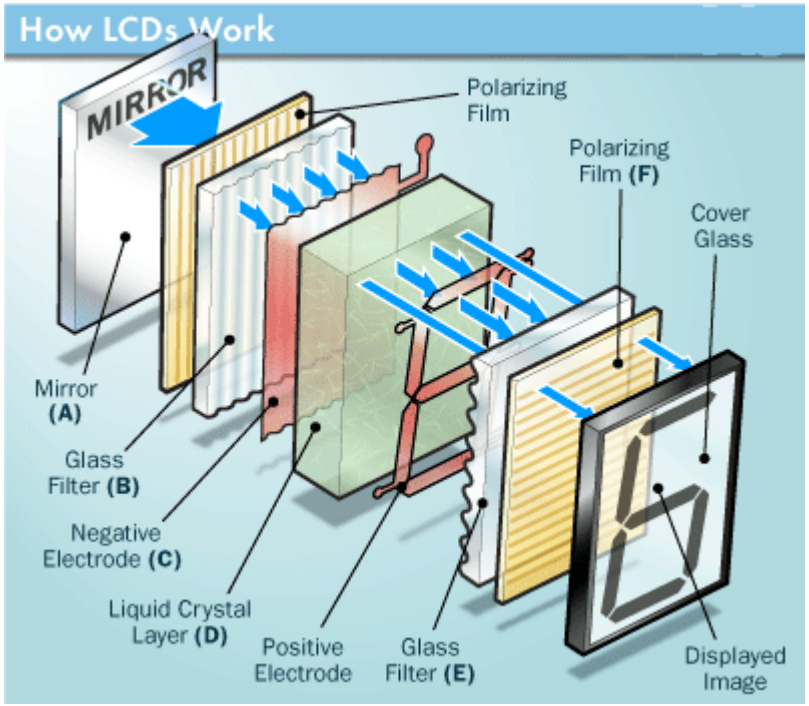
AVR Microcontroller

LCD Display

ECE 372A

LCD Character Display

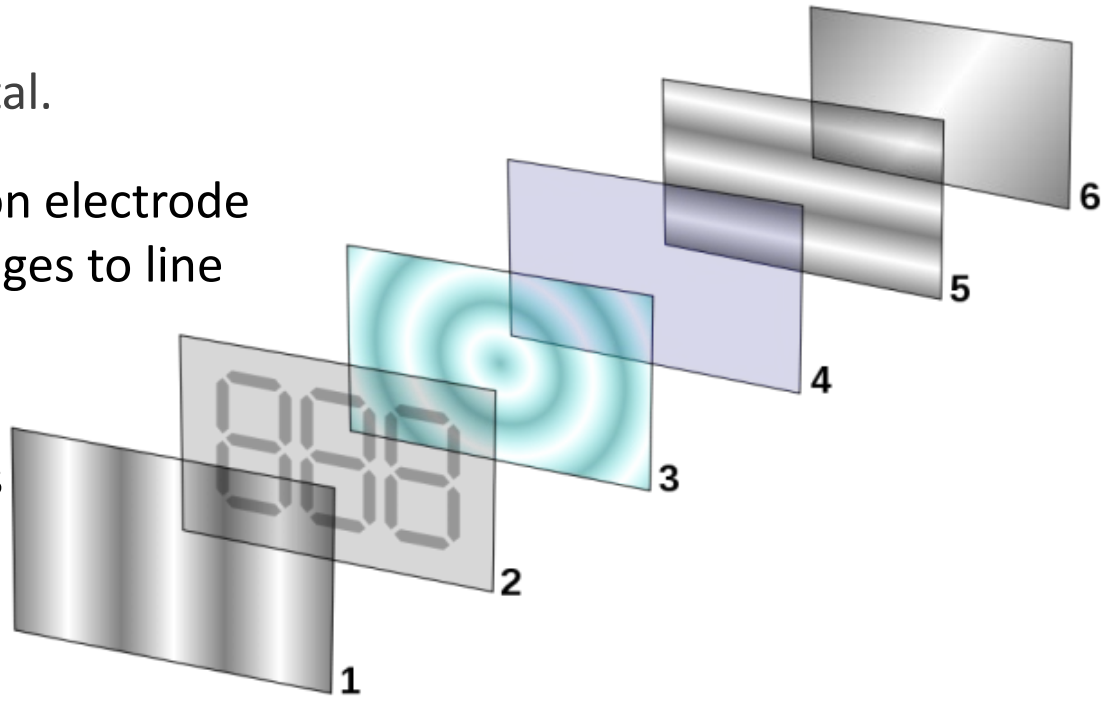
- A liquid crystal cell consists of a thin layer (about 10 u m) of a liquid crystal sandwiched between two glass sheets with transparent electrodes deposited on their inside faces.
- With both glass sheets transparent, the cell is known as *transmittive type cell.*
- When one glass is transparent and the other has a reflective coating, the cell is called *reflective type.*
- The LCD does not produce any illumination of its own. It, in fact, depends entirely on illumination falling on it from an external source for its visual effect



Liquid crystals have properties of both liquid and solid crystals

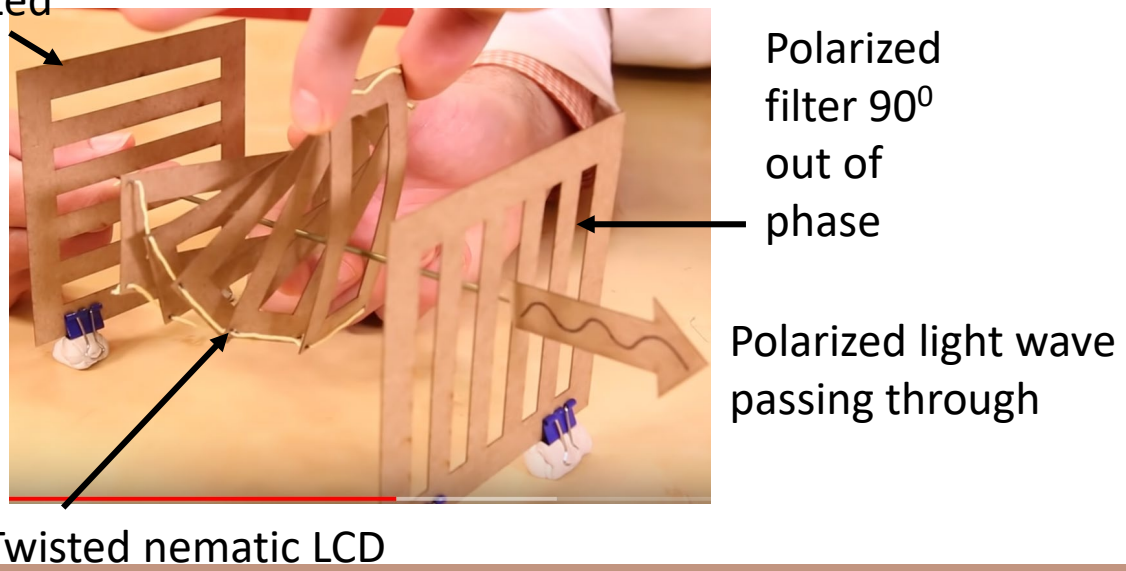
LCD Structure

1. Polarizing filter film with a vertical axis to polarize light as it enters.
2. Glass substrate with Indium tin oxide (ITO) electrodes. The shapes of these electrodes will determine the shapes that will appear when the LCD is turned ON. Vertical ridges etched on the surface are smooth.
3. Twisted nematic liquid crystal.
4. Glass substrate with common electrode film (ITO) with horizontal ridges to line up with the horizontal filter.
5. Polarizing filter film with a horizontal axis to block/pass light.
6. Reflective surface to send light back to viewer. (In a backlit LCD, this layer is replaced with a light source.)



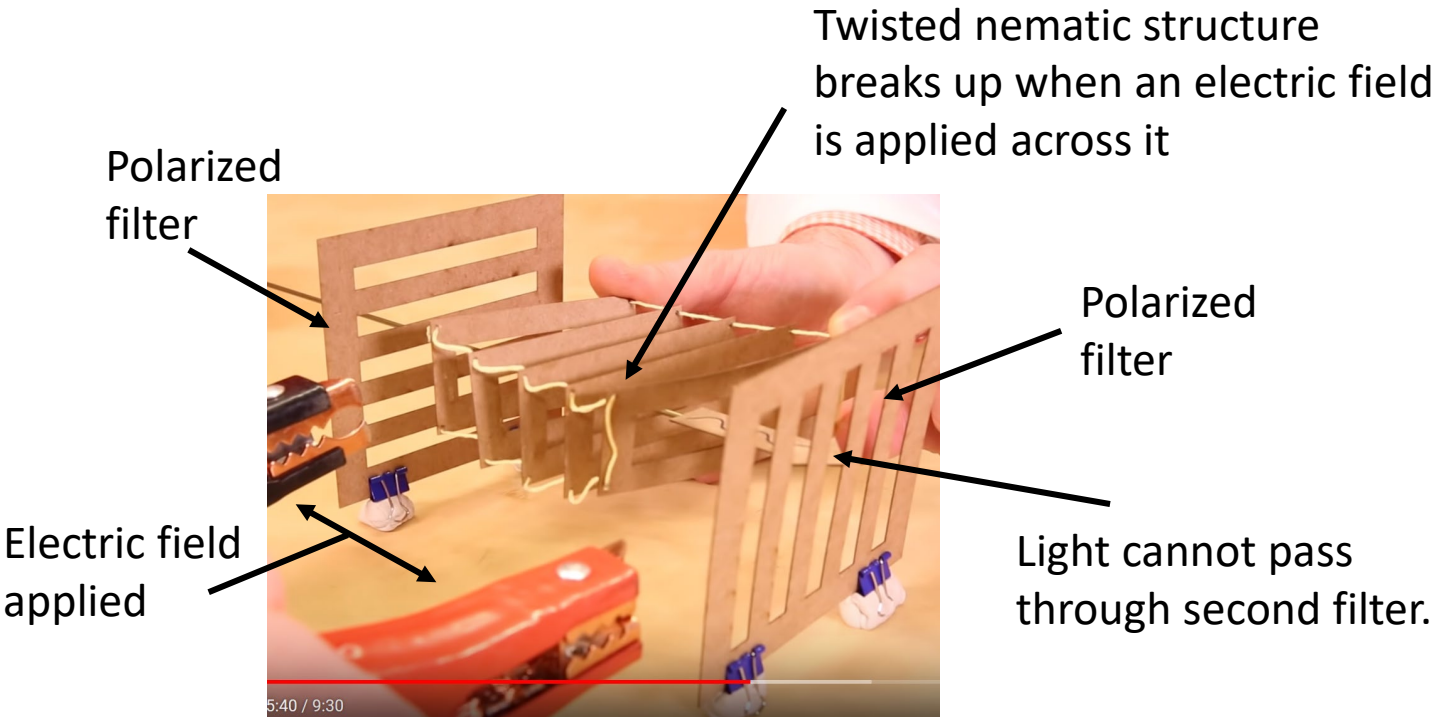
How it works

- LCDs use twisted nematic crystals.
- Polarized light can only pass through a filter that is orientated in the direction of the polarized light wave.
- Polarized light passes through the first filter. The light interacts with the liquid crystal material and rotates it's polarization so that it can exit through a filter that is polarized 90^0 out of phase with respect to the first filter.
- The light's polarization has been twisted and can pass through the exit filter.

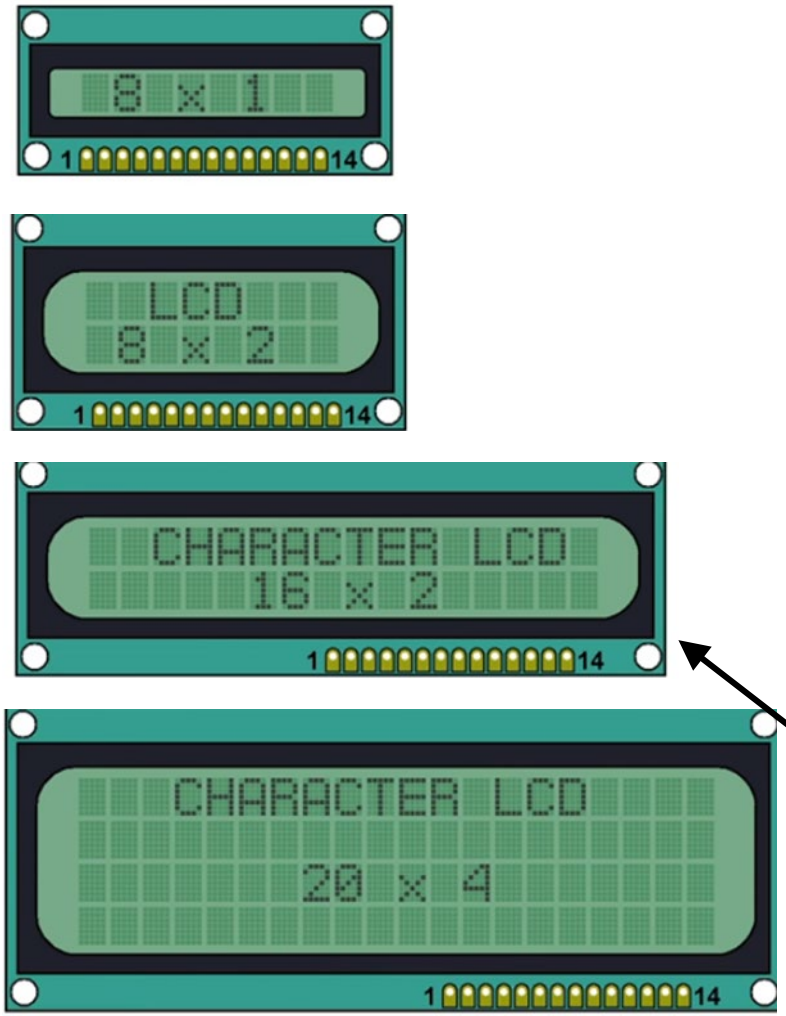


How it works

- When you apply an electric field across the LCD, the twisted nematic structure breaks up and no longer can twist the polarized light passing into the crystal (blocked).



LCD Character Displays



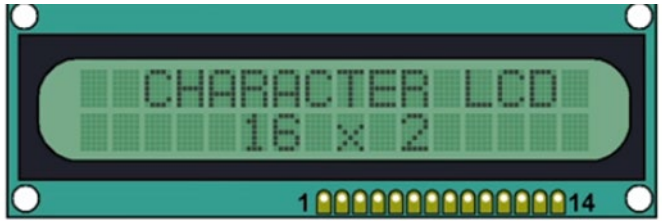
- Very commonly used in many applications
- There are different sizes and different colors
- Can even have custom designed LCD for specific application.

Your Arduino kit has a 16 x 2 LCD.

The LCD Module HDM08216H-3

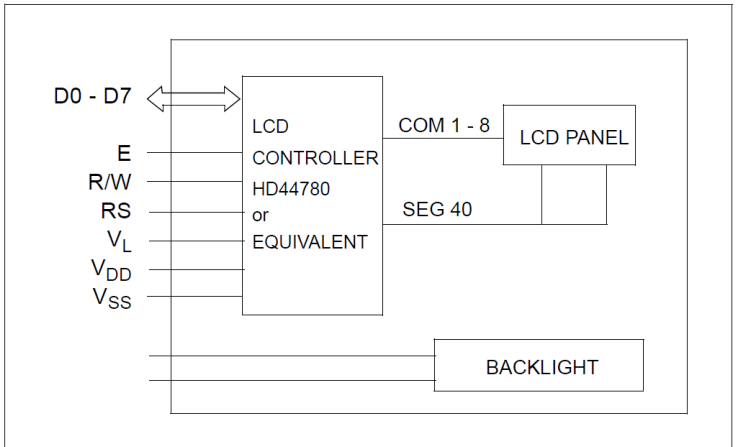
- 16 x 2 LCD
- The LCD Display Module actually has an LCD controller within the module that our ATMega 2560 sends data and instructions to.
- We are not driving the LCD directly but that we are sending instructions or writing characters to the chip on the LCD board.

Front



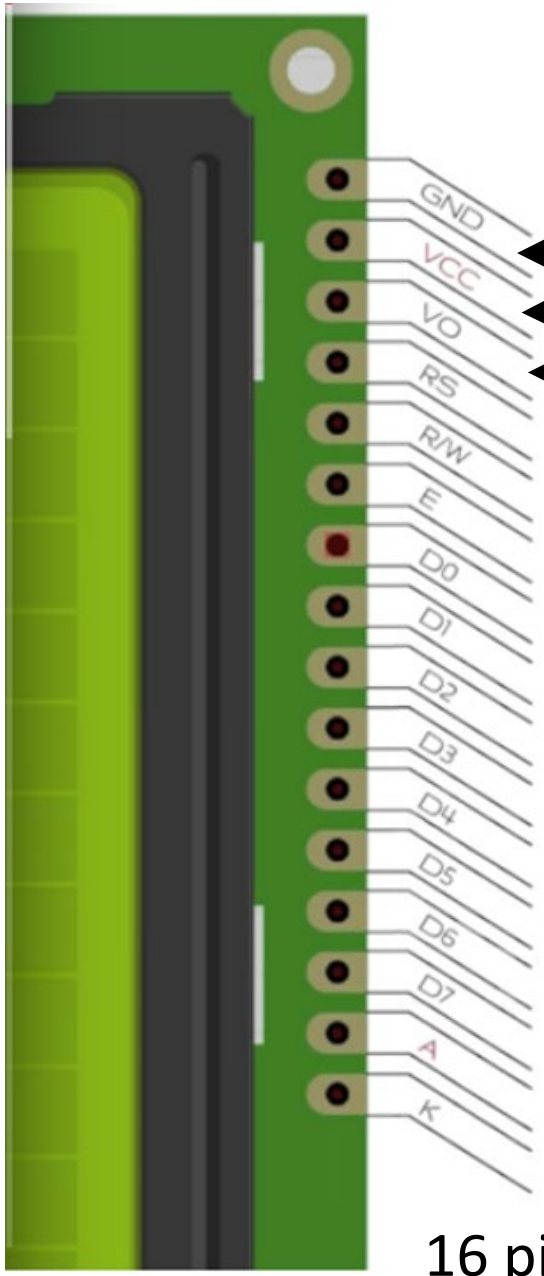
Back

Block Diagram



The LCD we are using has a controller chip interface built into the board

Pinout of LCD



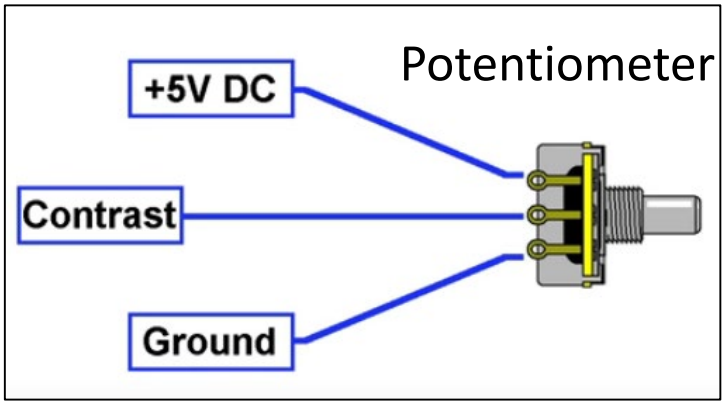
Ground

+ 5V

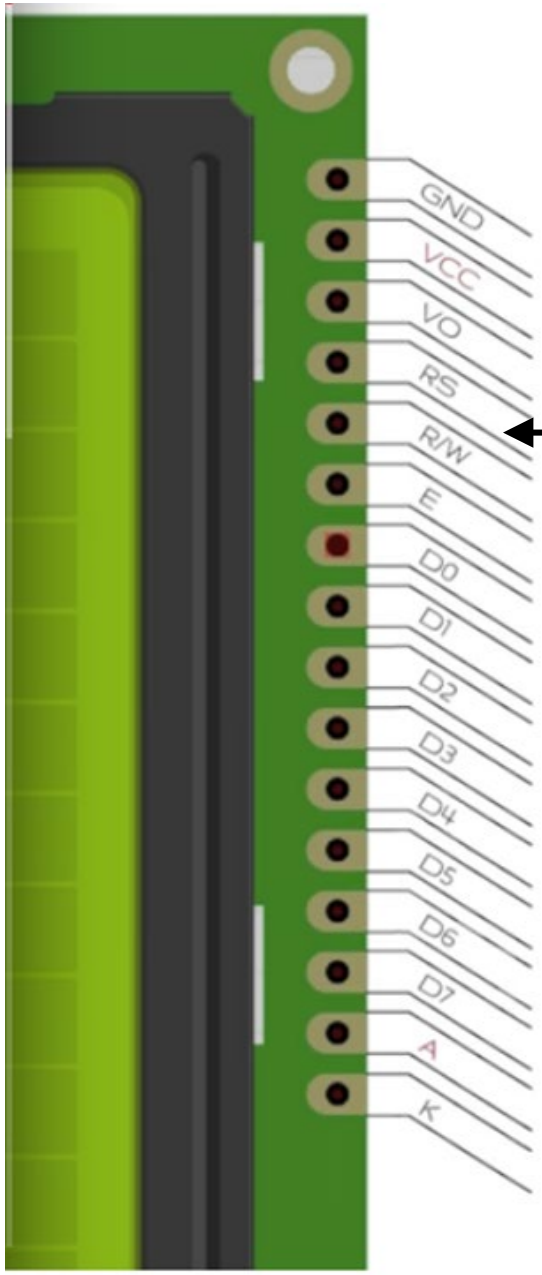
Backlight control (contrast)

16 pins total

Contrast control adjustment



Pinout of LCD

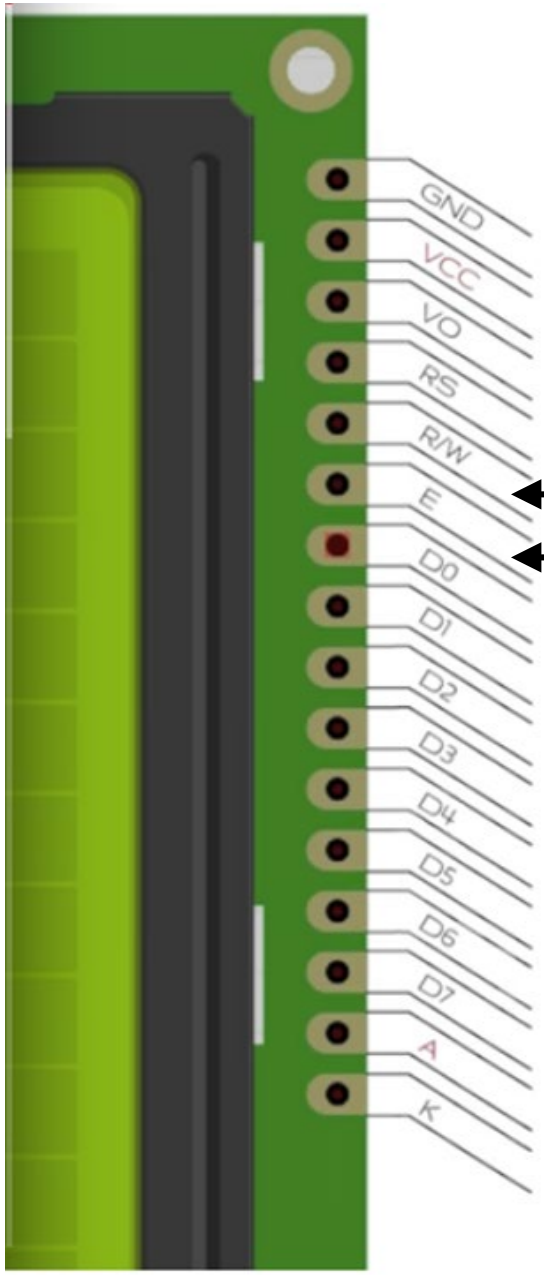


← Register Select

High (Text data input)
Low (Command data input)

- Text data input displays ASCII characters
- Command data input examples:
  - Clear display
  - Turn off LCD
  - Move cursor
  - Clear display
  - …….

Pinout of LCD



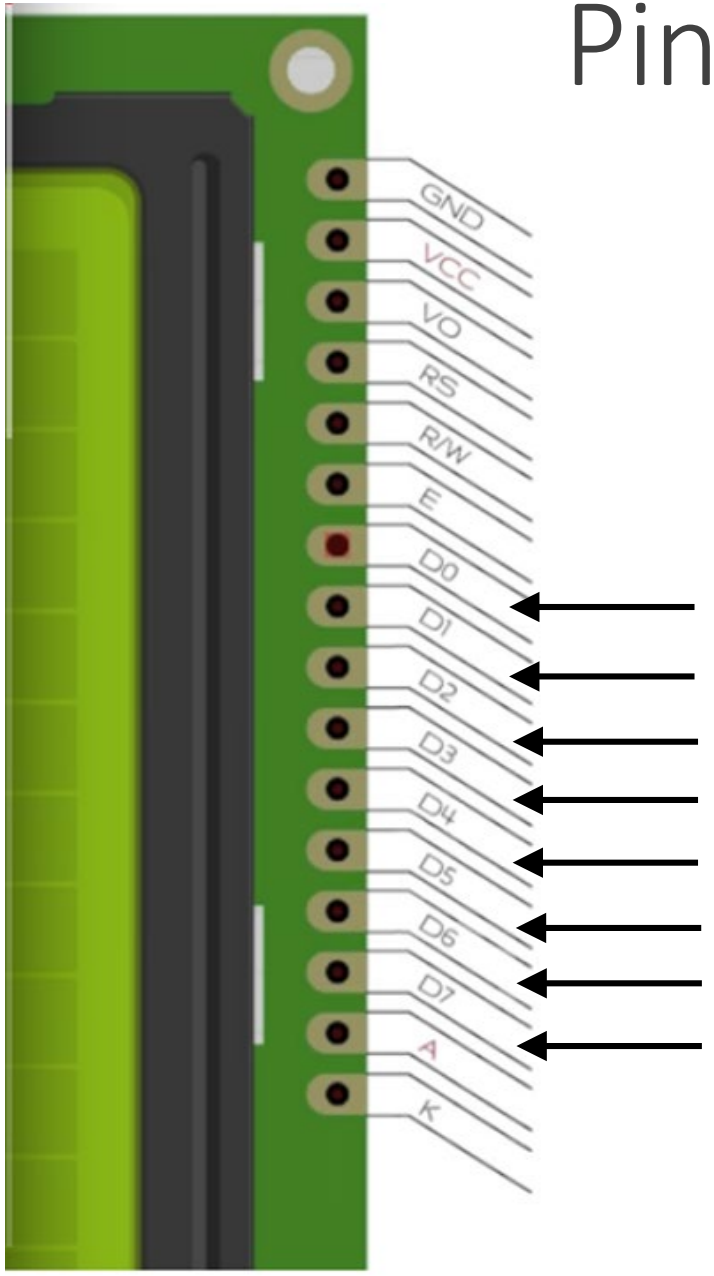
Read/Write

High (Data read)
Low (Data Write)

Enable Pin

- When Enable is logic low, the LCD does not do anything with the data on the data bus pins.
- When Enable goes from low to high, (data D0 through D7 are fed into LCD).

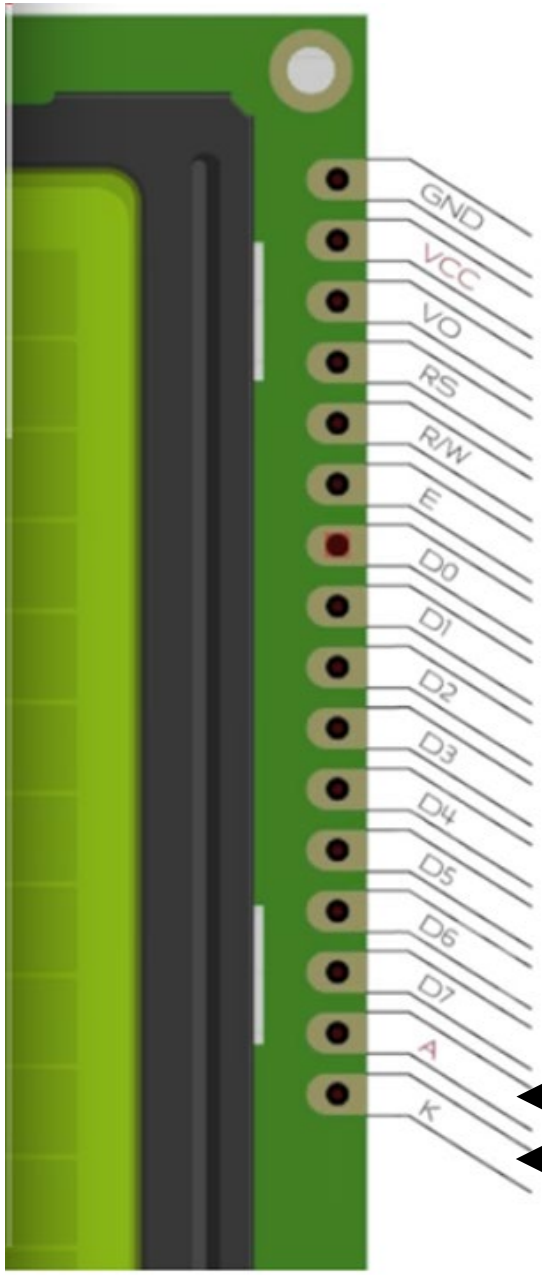
Pinout of LCD



8 bit parallel data port for LCD –used for

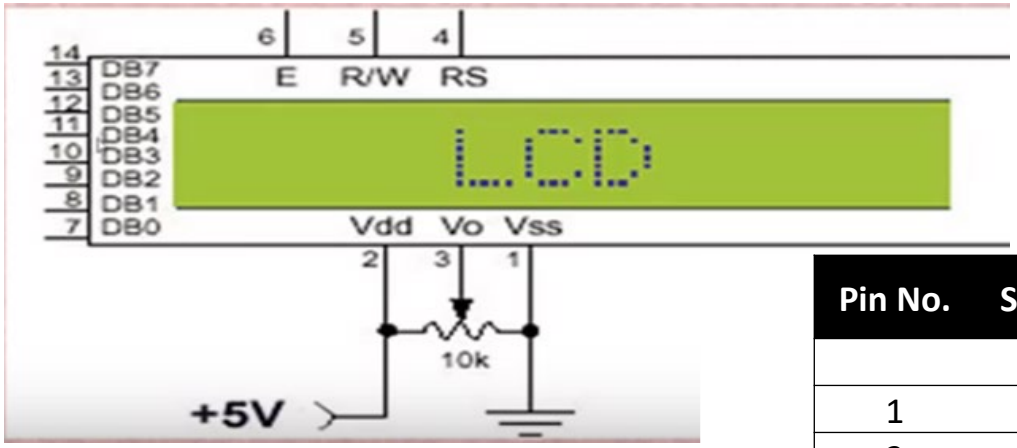
- Displaying 8 bit ASCII character or
- Executing 8 bit command

Pinout of LCD



Backlighting control

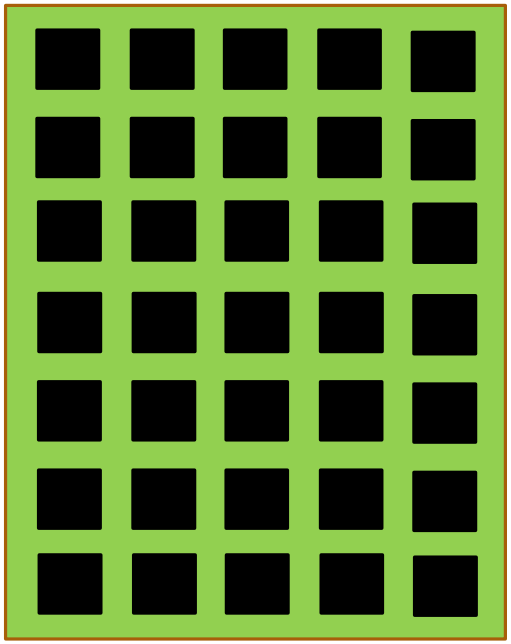
LCD Interfacing

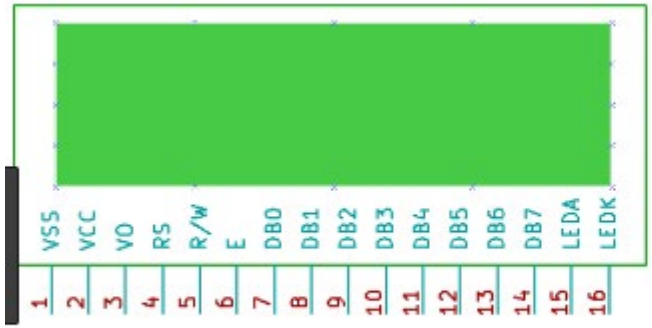


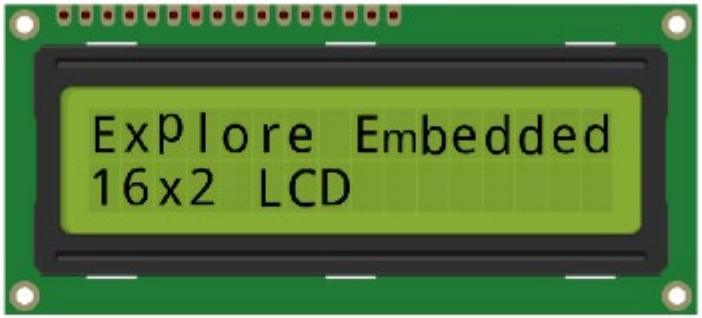
Pin No.	Symbol	Function
1	Gnd	Power Supply Ground
2	Vcc	5V Supply
3	VEE	Contrast adjustment voltage
4	RS	Register select (H: data; L: instruction)
5	R/W	Read/Write data (H: LCD -> uC, L: uC -> LCD)
6	E	Enable pulse
7	D0	Data bit 0
8	D1	Data bit 1
9	D2	Data bit 2
10	D3	Data bit 3
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7
15	A	Anode of backlight LED
16	K	Cathode of backlight LED

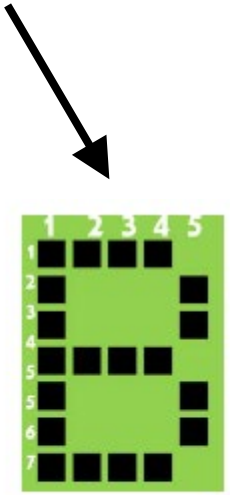
Display mode

5 x 7 matrix of LCD elements for each character









LCDs & Interfacing

Addressing with 4 bits vs 8 bits

- This LCD has two interfaces
  - One 8-bit interface or 2 4-bit interfaces
  - We will be using the 4-bit interface
- The major difference in 4 bit and 8 bit mode
  - # of Data pins used.
  - LCD initializing commands (not really too different).
  - In 4 bit mode only four data pins are used. Character 8-bit ascii value is divided in to two 4-bit nibbles.
    - High nibble is sent first following by the lower nibble.
  - In 8 bit mode, an 8-bit ascii value of character is sent at a single time period and displayed on the LCD.
  - Thus the 4-bit mode generates latency. Although 4-bit mode generates latency it on the other hand saves 4 gpio pins. Which can be utilized elsewhere.
- We are going to use 4 bits instead of tying up all 8 pins.
  - 8 bit vs 4 bit

LCD Command Table

- In the HDM08216H-3 LCD datasheet, there are a set of commands that can be issued for the LCD.
- This table specifies what value each pin must be to effectively send a command.
- However, we must first know how to send a command of any type.

COMMANDS FOR CHARACTER MODULES

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged.	40μs~1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and enables/disables the display.	40μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B).	40μs
Cursor & Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing the DD RAM contents.	40μs
Function Set	0	0	0	0	1	DL	N$	RE	*	#	Sets the data width (DL), the number of lines in the display (L), and the character font (F).	40μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data can be read or altered after making this setting.	40μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. Data may be written or read after making this setting.	40μs
Read Busy Flag & Address	0	1	BF	AC							Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents.	1μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	46μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	46μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift. S/C= 1: Display shift S/C = 0: cursor move R/L= 1: Shift to the right. R/L= 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line RE = 1: Ext. Reg. Ena. F = 0: 5 x 7 dots BF = 1: Busy BF = 0: Can accept data *# Set to 1 on 24x4 modules* *$ With KS0072 is Address Mode.*										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address.	Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times.

Sending LCD Commands

- This LCD has two interfaces
  - One 8-bit interface or a 4-bit interface
  - We will be using the 4-bit interface

- All commands require 8 bits, so how does a 4 bit interface work?
  - Send the 4 MSb, then the 4 LSb.

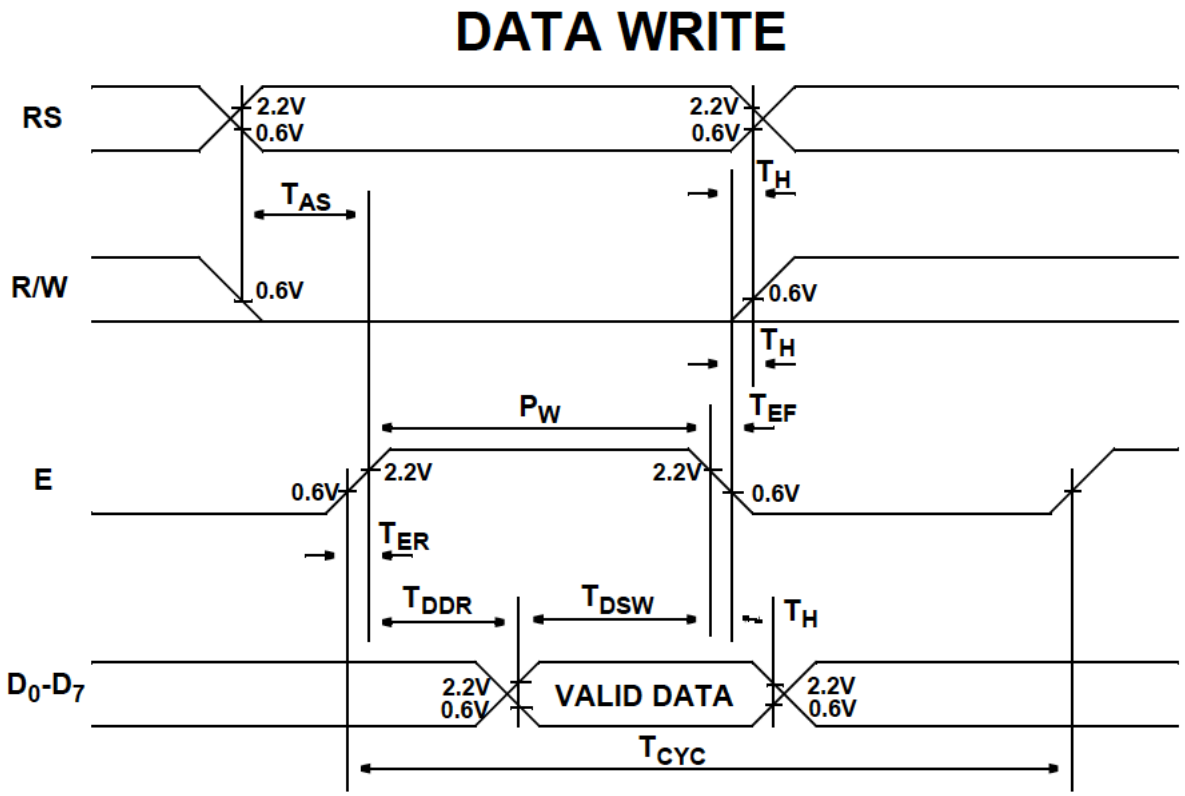
- So let’s see how to send 4 bits.

Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

This is a timing diagram.

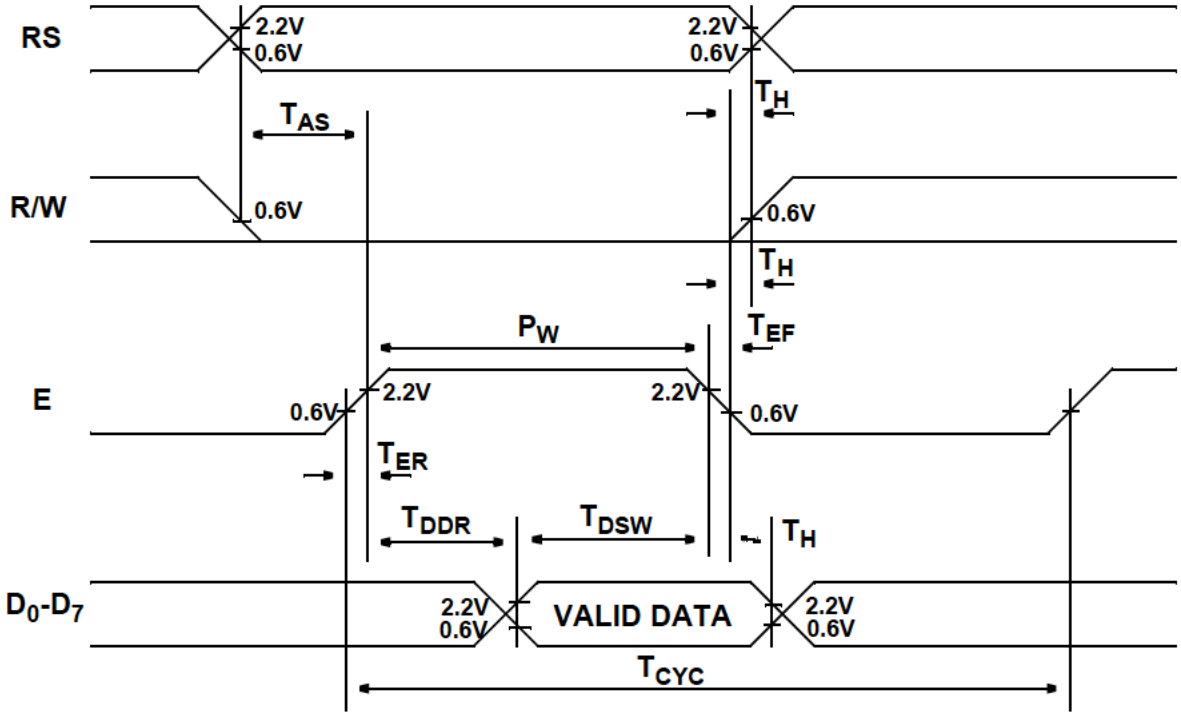


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

This is a timing diagram.



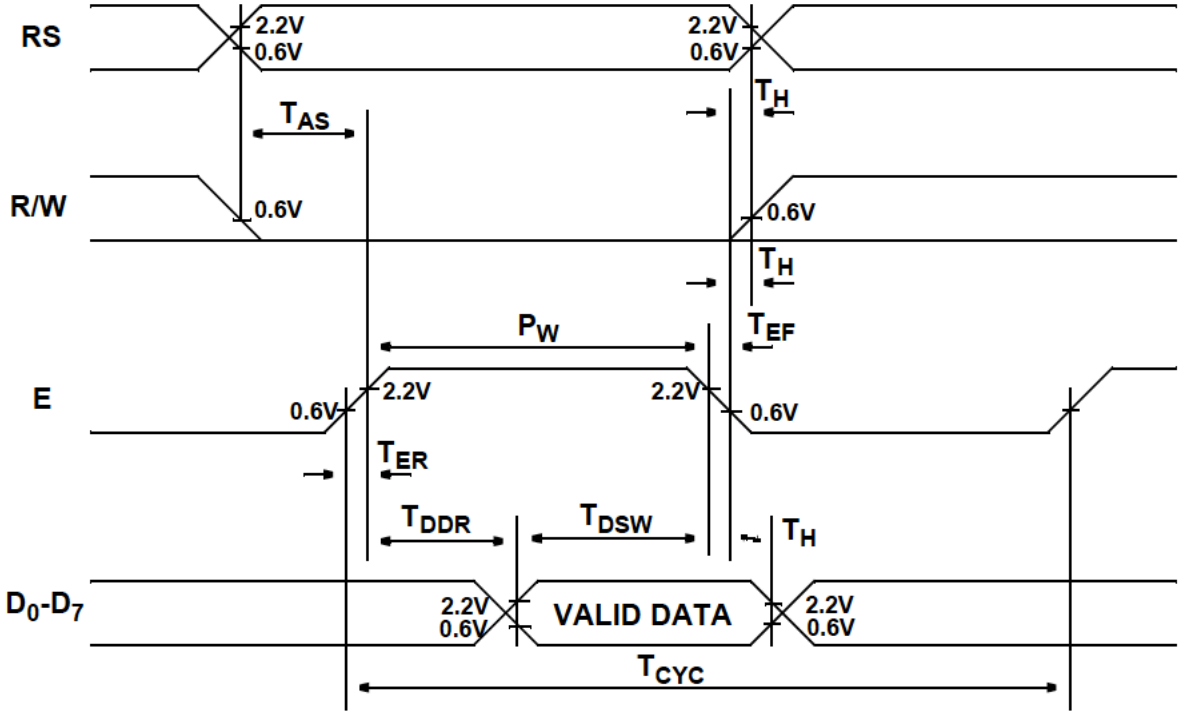
Increasing Time

Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

This describes the order in which pin values should be changed to write data.

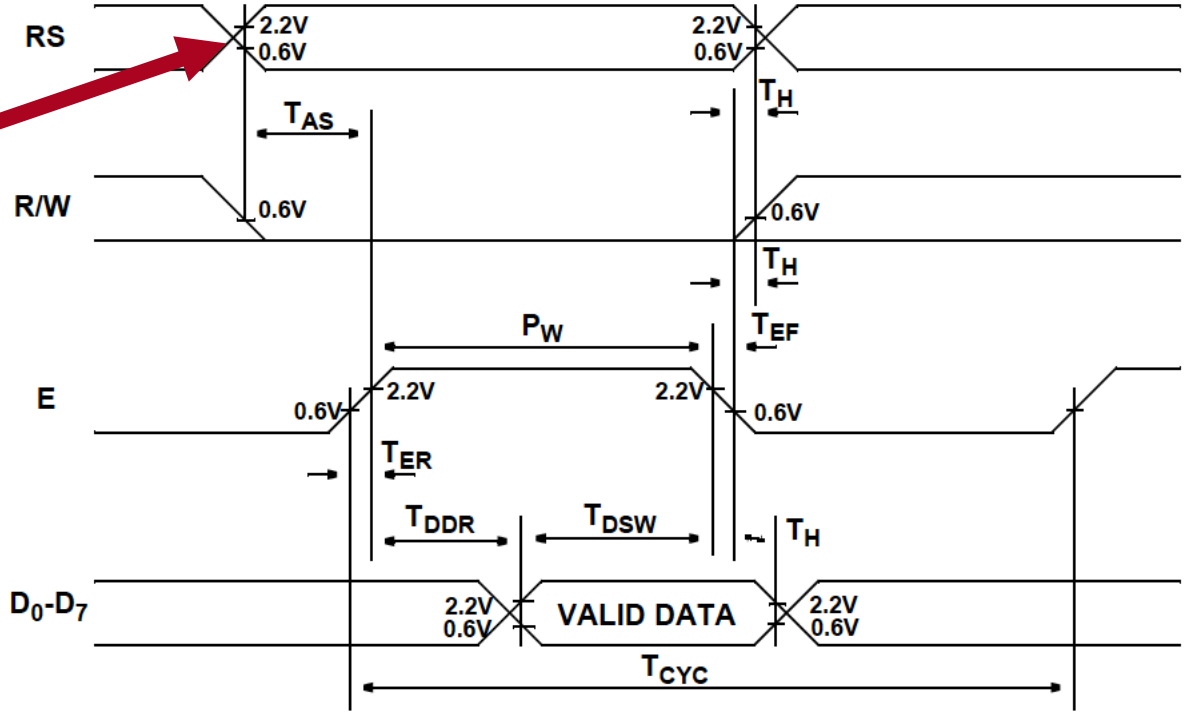


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

This cross of lines implies "the data changes"

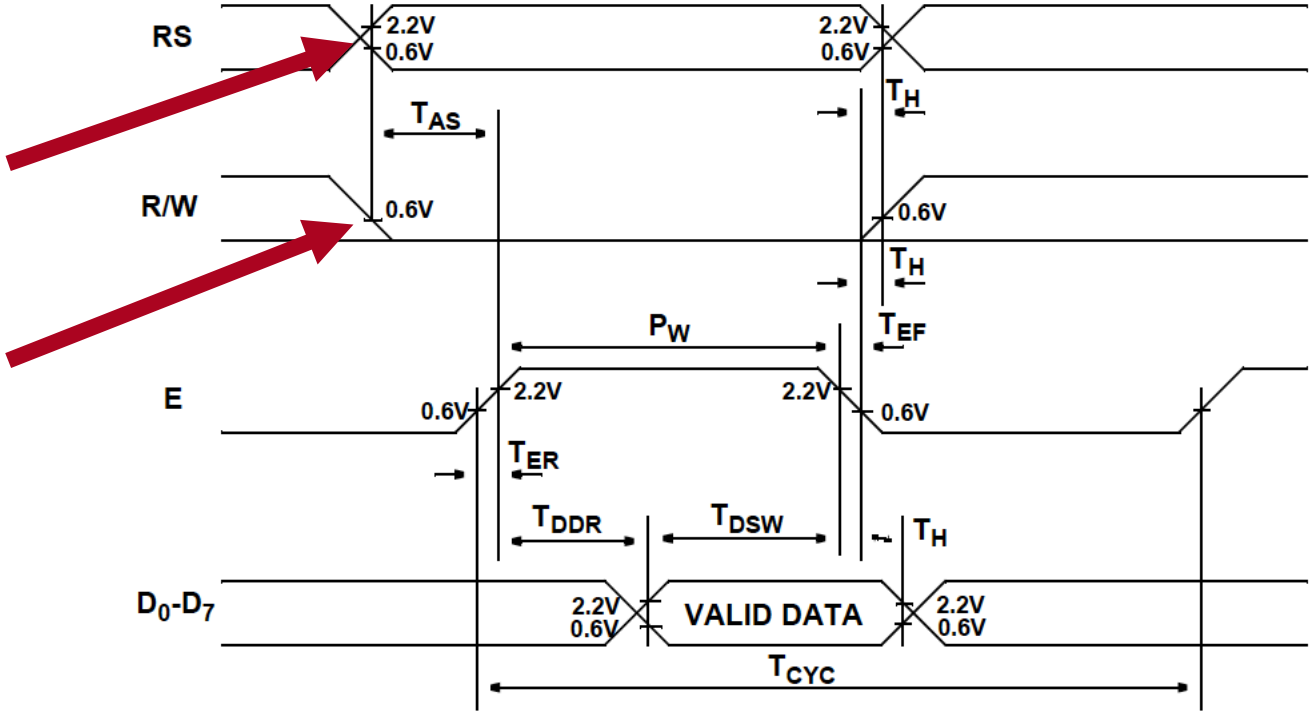


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

So the RS and R/W pins must be set to appropriate values first.

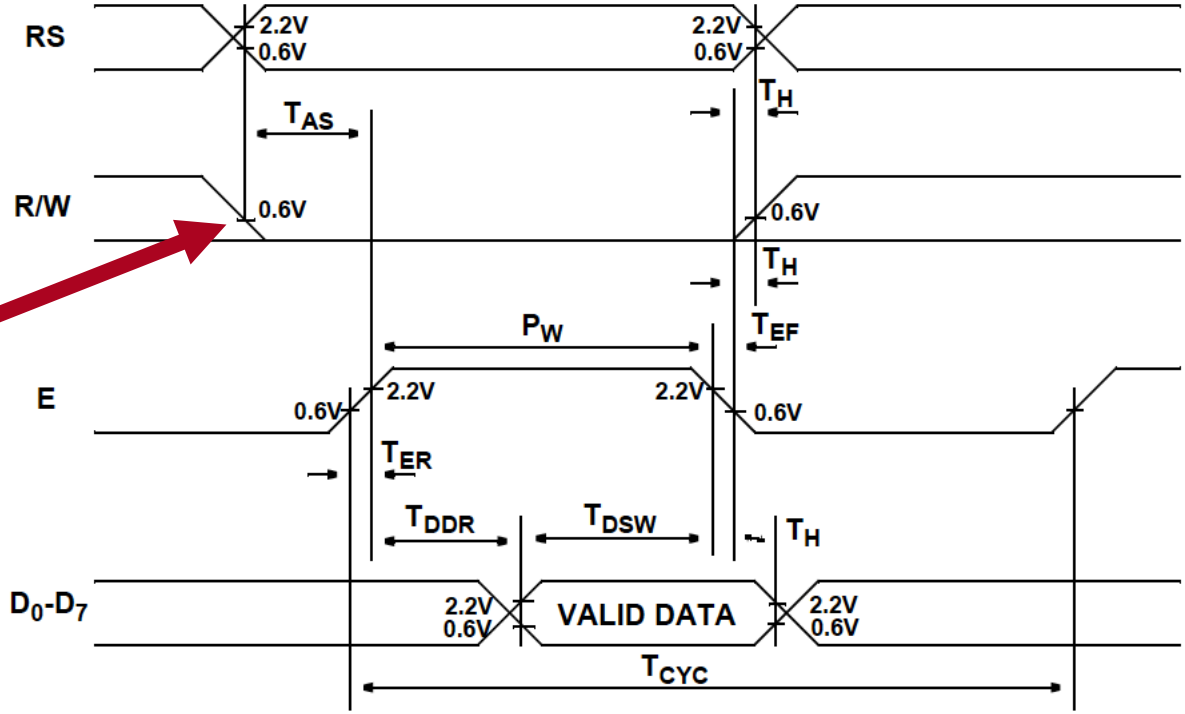


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

R/W doesn't have crossed lines because it must necessarily go LOW for a write command.

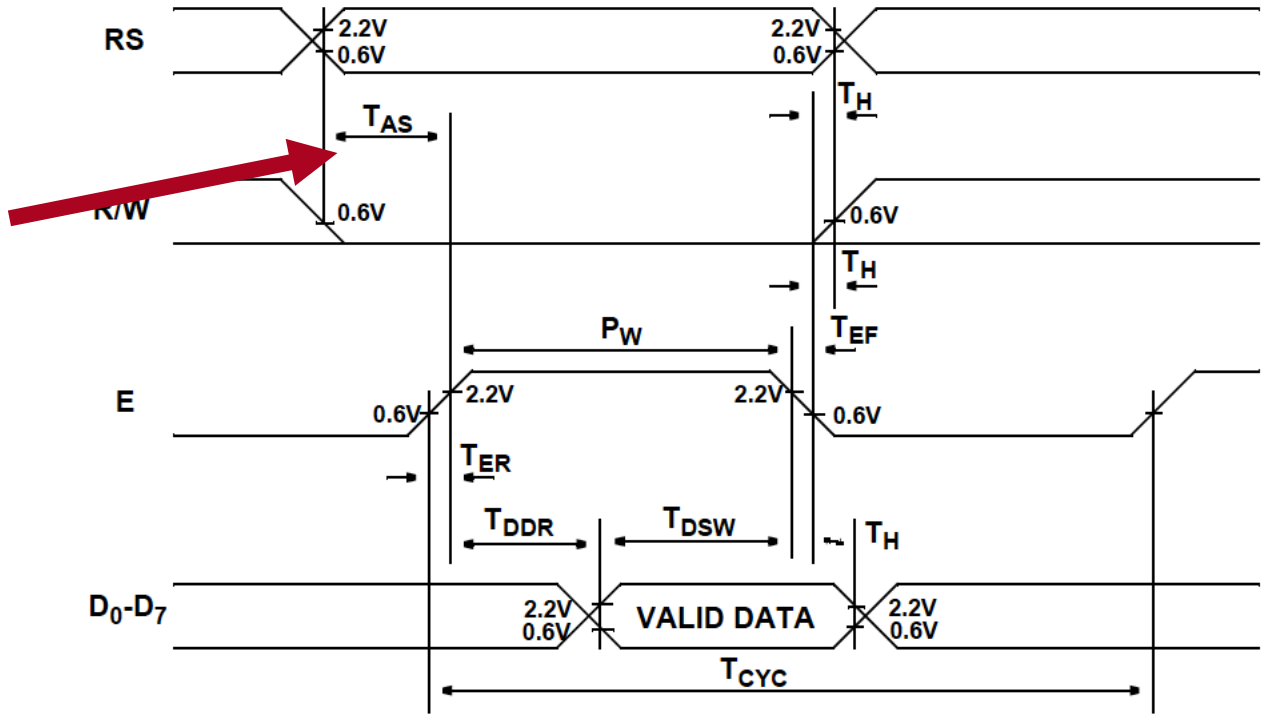


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

There is a minimum time that must elapse before the next event.

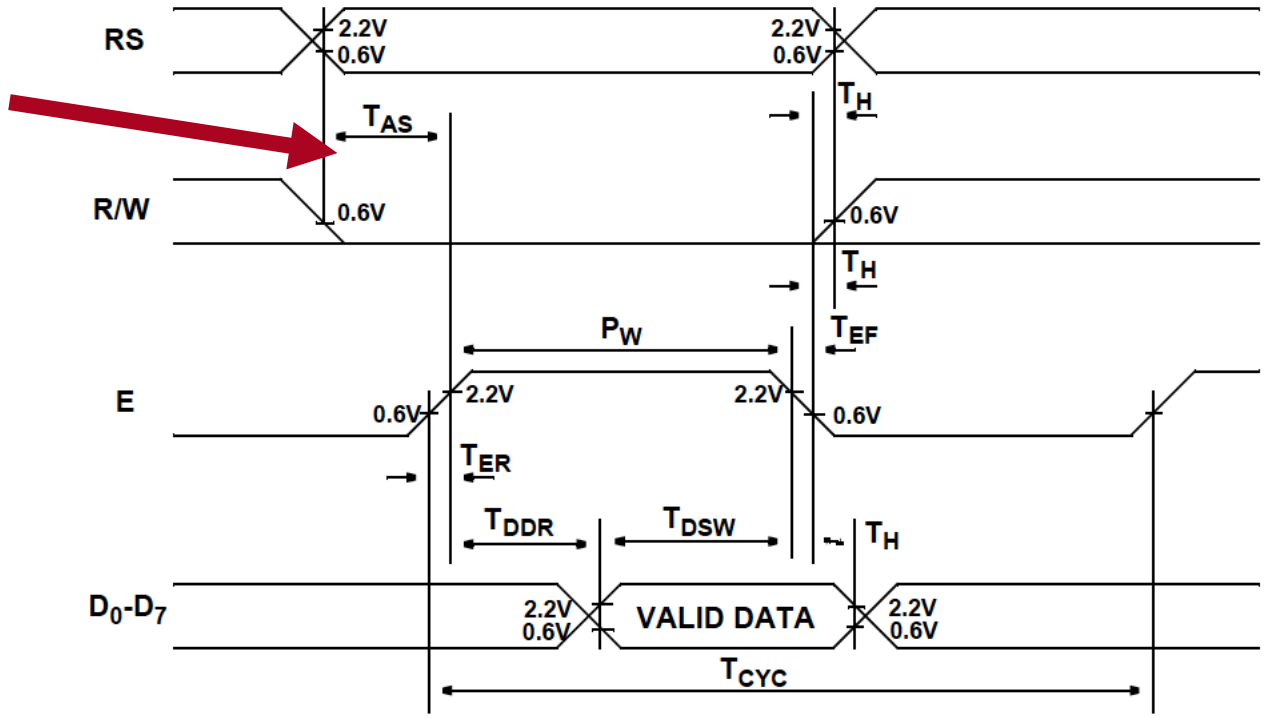


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

T_{AS} in the data sheet is 40 nS.

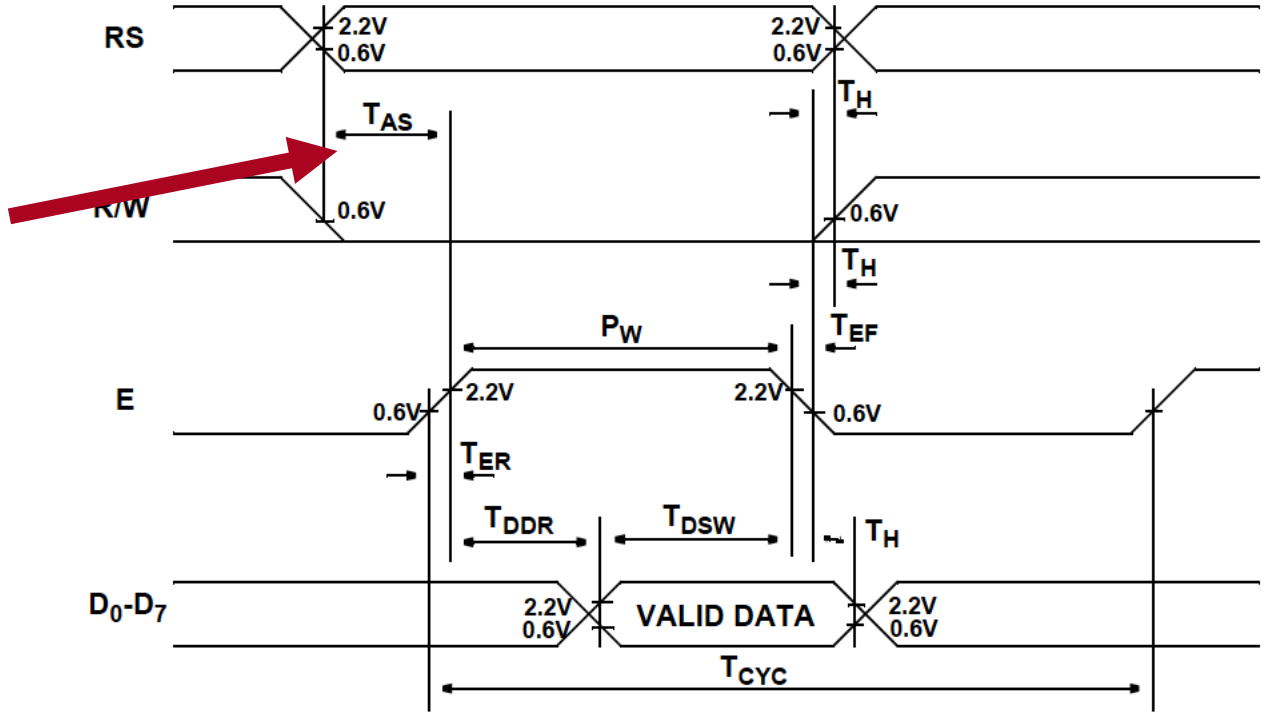


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

Our clock speed is not fast enough to change signals this fast.

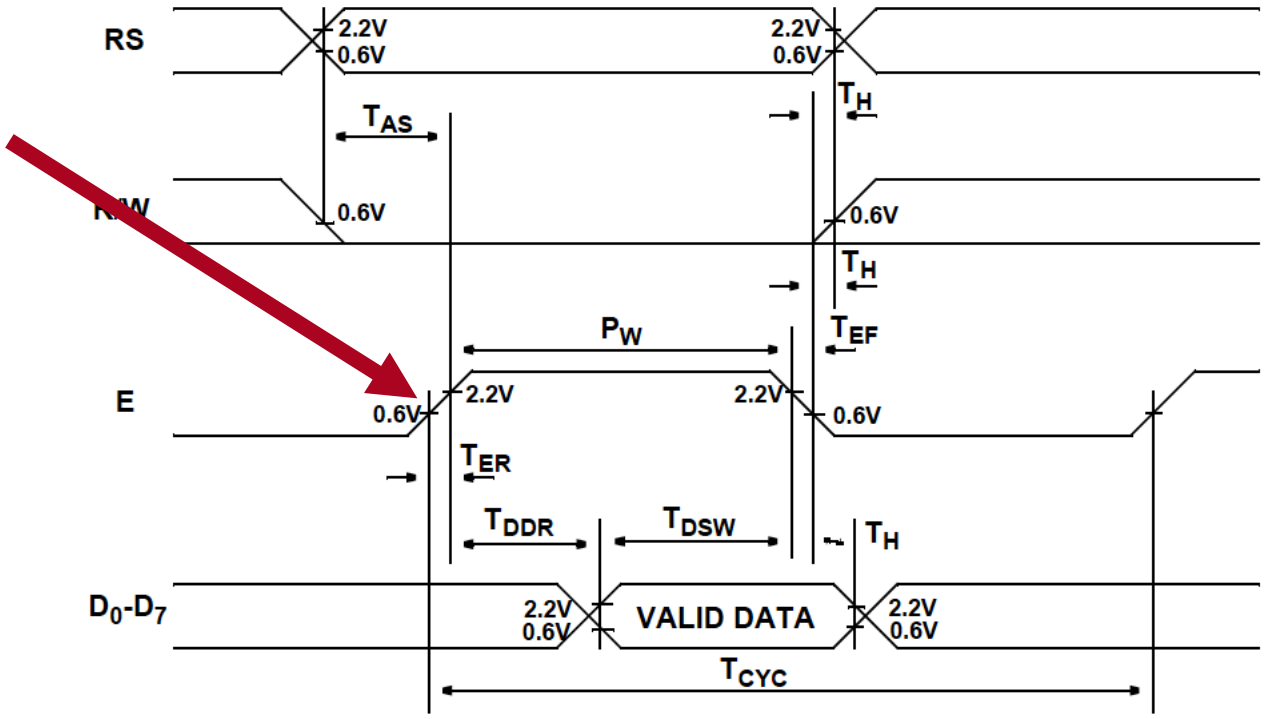


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

Then we put set the enable signal to HIGH.

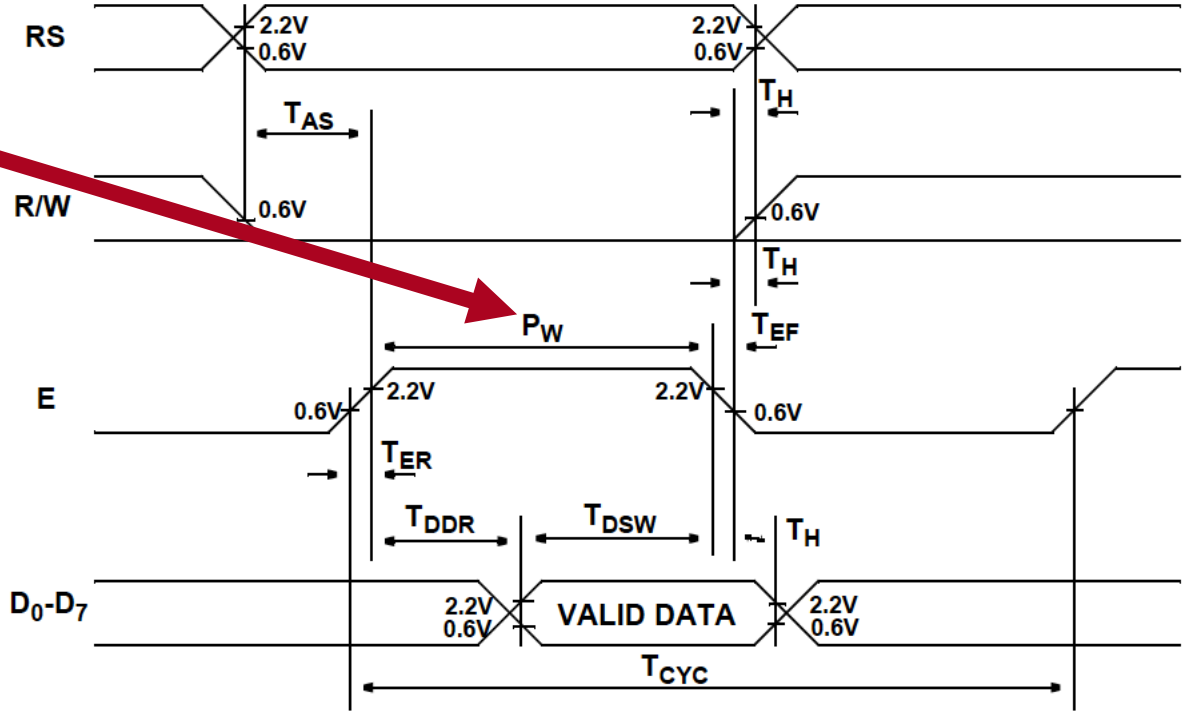


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

This has to be for at least P_w which is 230 ns.

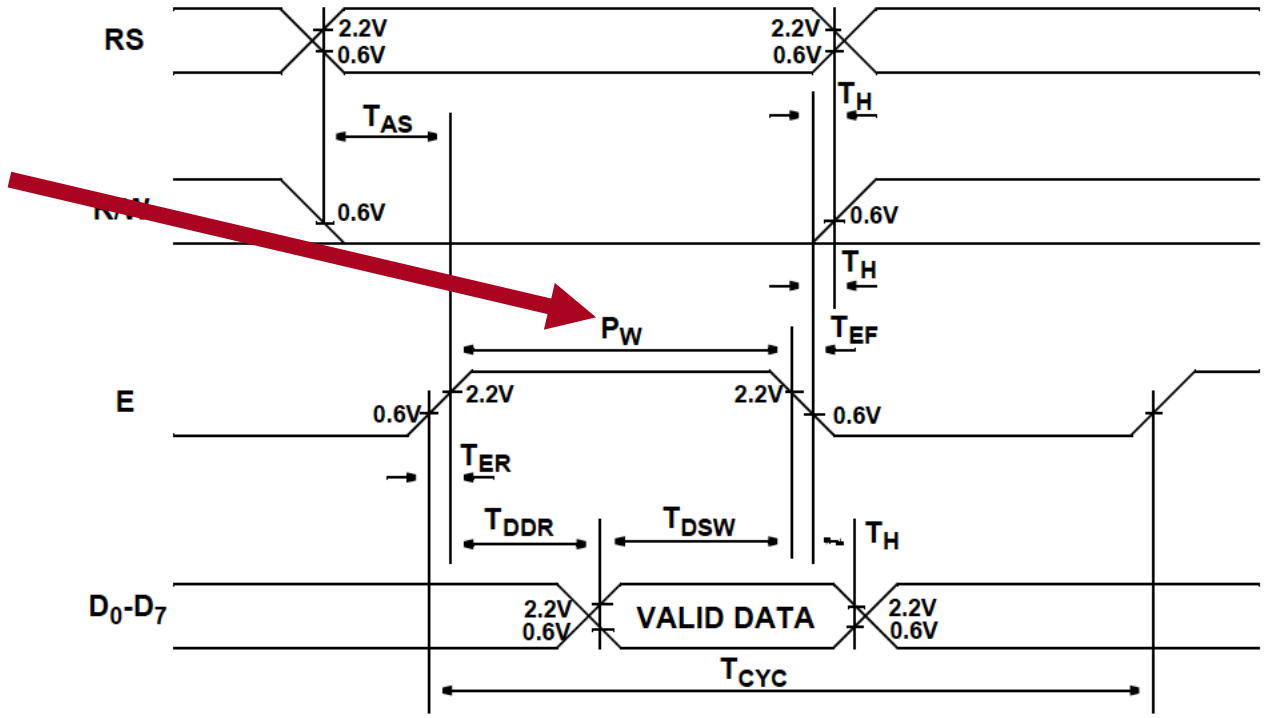


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

Again our clock speed is not fast enough to delay less than this.

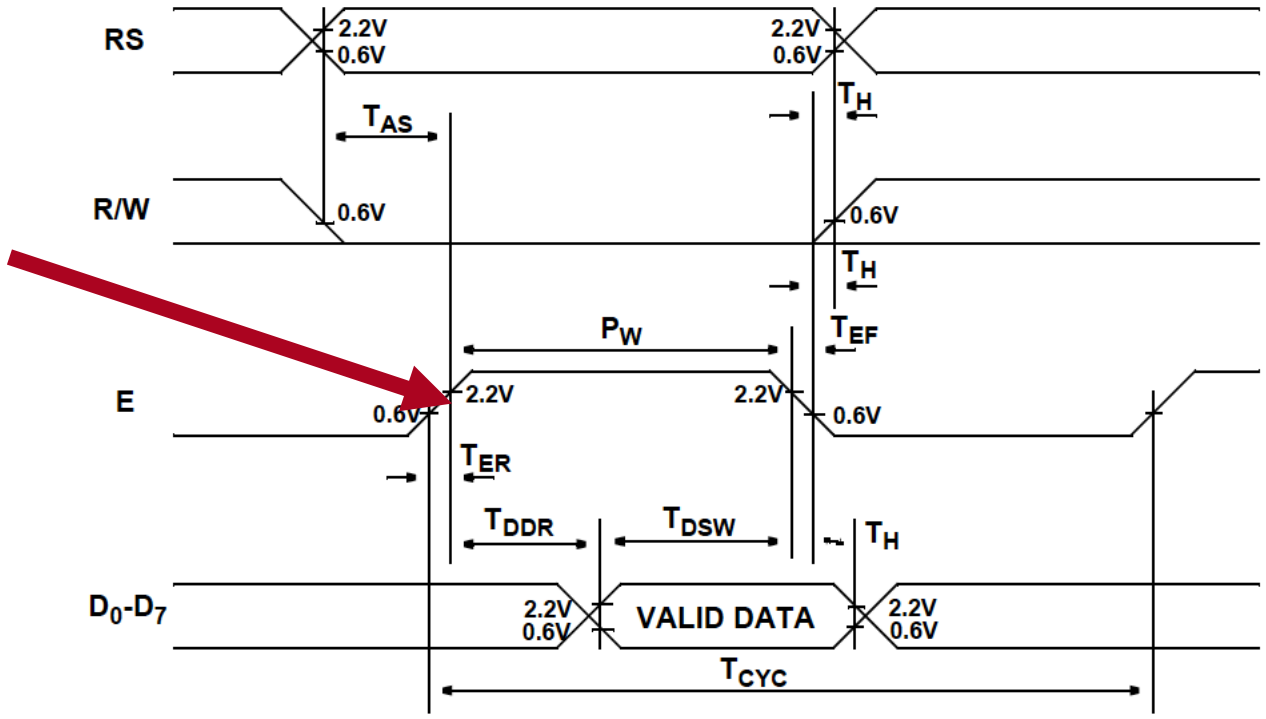


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

However, it takes non-zero time for the enable signal to go HIGH after setting a PORT bit.

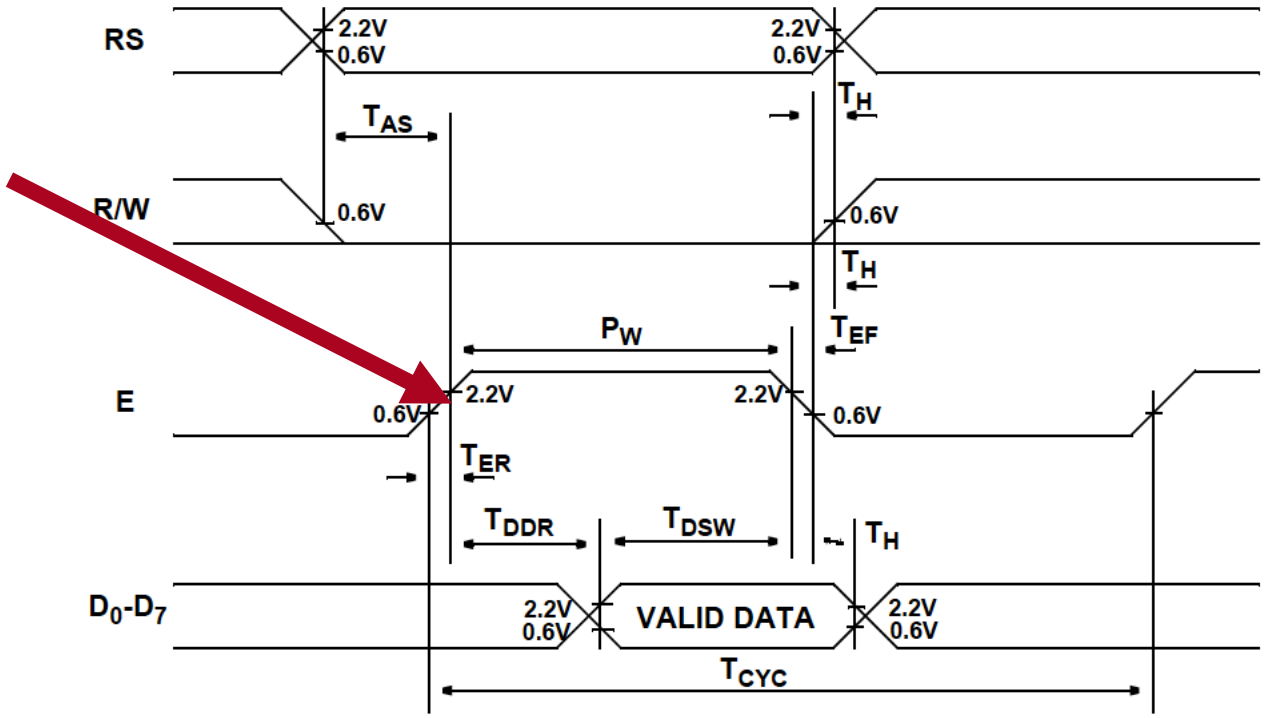


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

So it's better to delay at least 1 microsecond here.

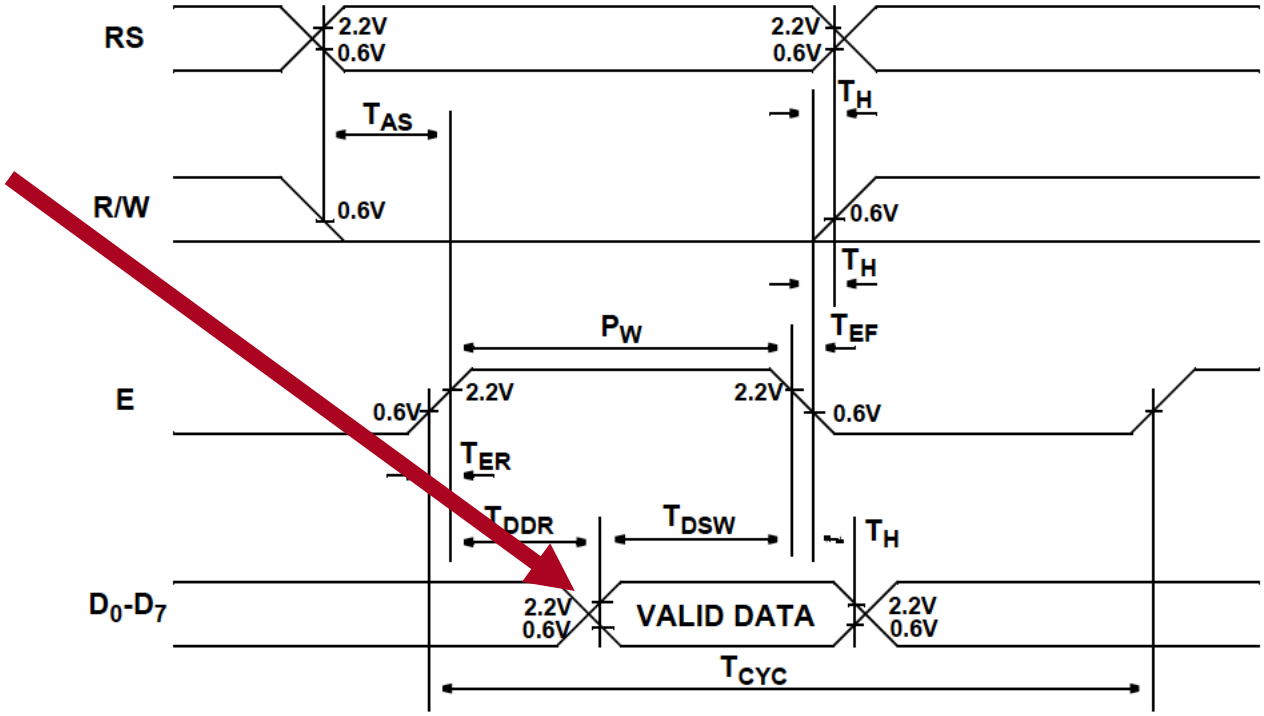


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

Finally, we can change the values of our data pins.

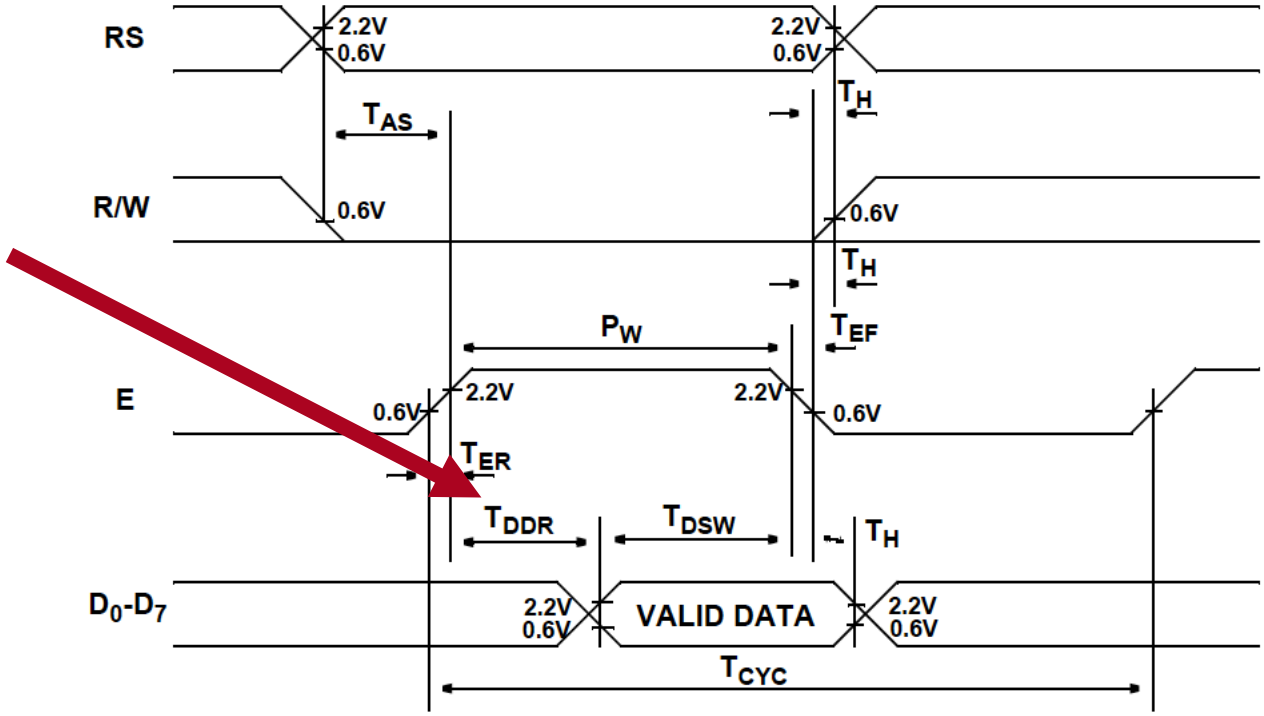


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

However, since T_{DDR} is 360 ns MAXIMUM, we cannot change our data quickly enough!!

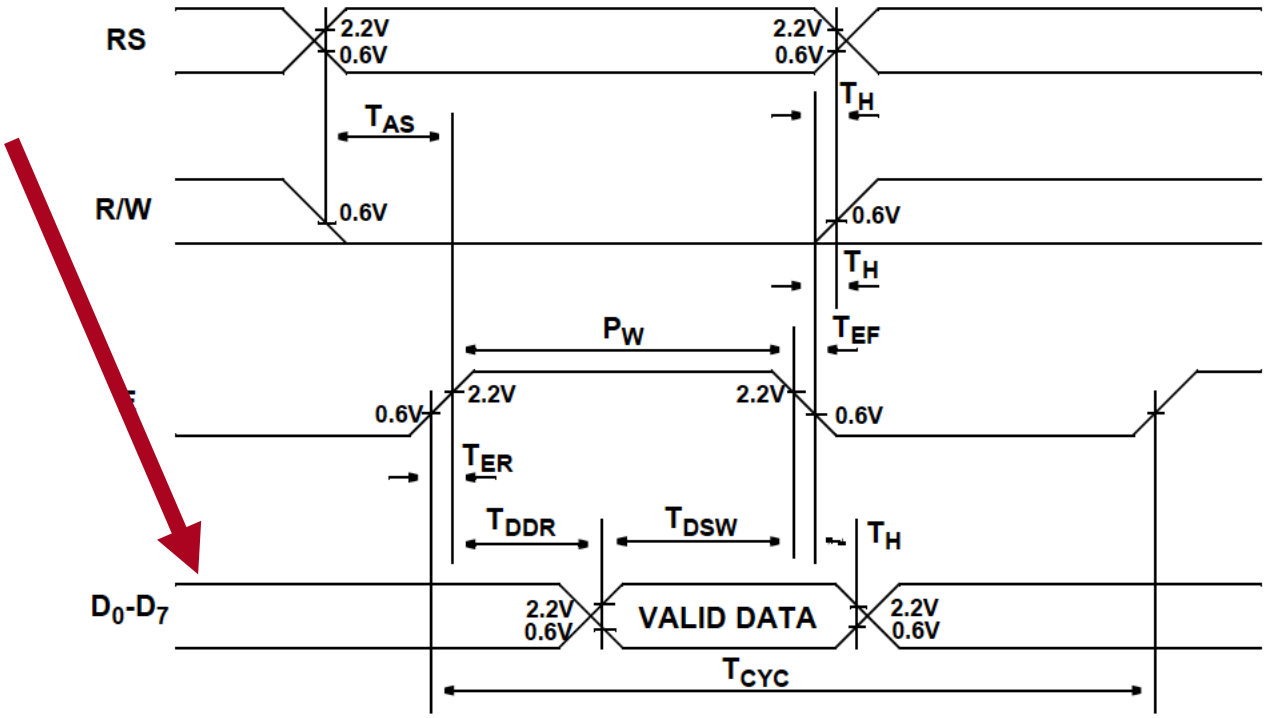


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

So… we can just change our data back here.

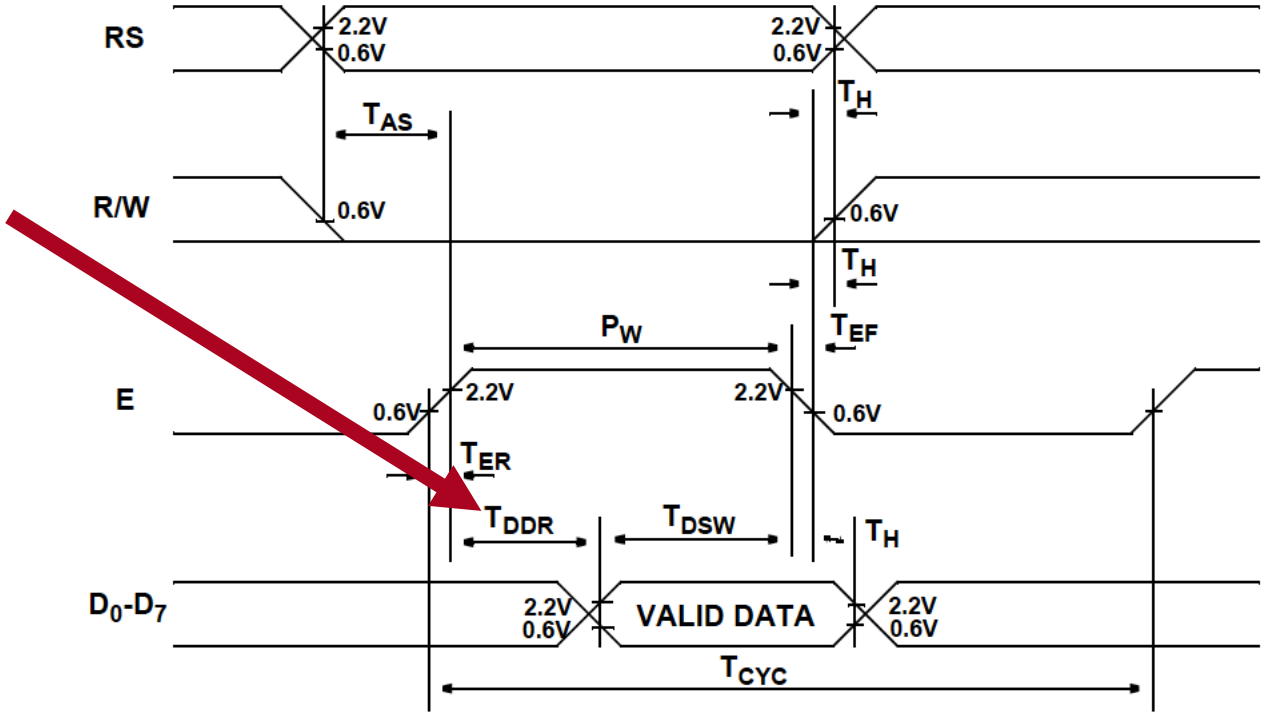


Sending LCD Commands

TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

This just says how long after the enable signal is high to have our data READY.

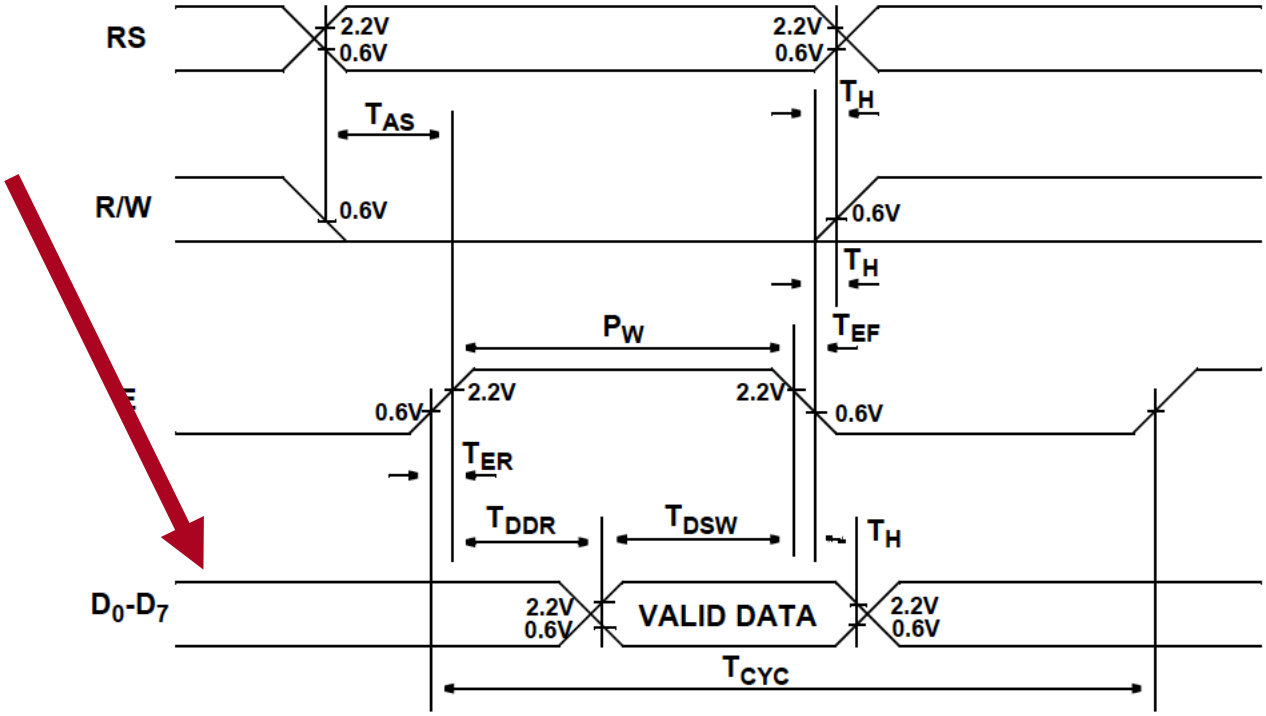


Sending LCD Commands

TIMING CHARACTERISTICS

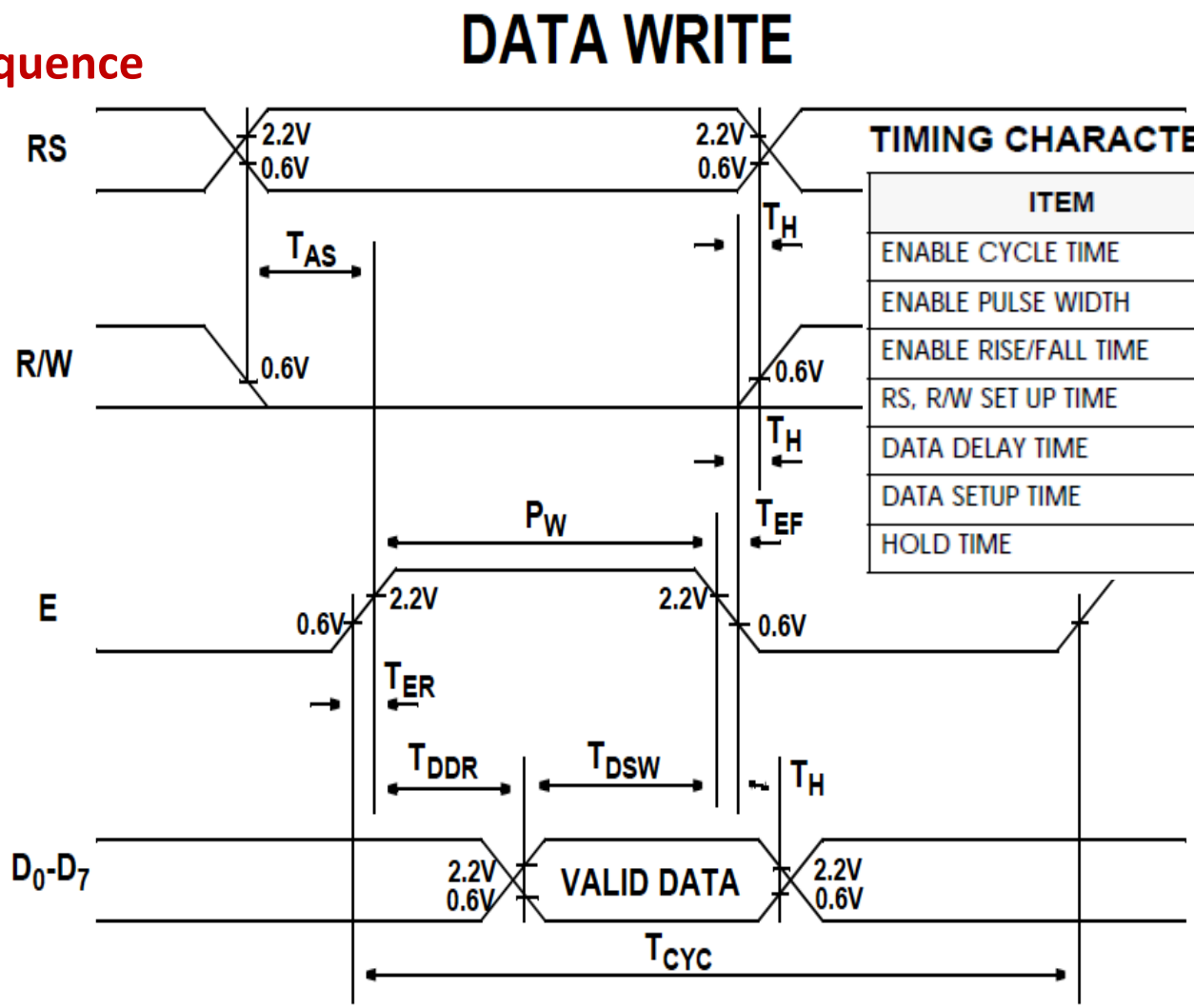
ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

Nothing wrong with having it ready beforehand.



Timing sequence for a Write to LCD

Sequence



TIMING CHARACTERISTICS

ITEM	SYMBOL	MAX.	MIN.	UNIT
ENABLE CYCLE TIME	T_{CYC}		500	nS
ENABLE PULSE WIDTH	P_W		230	nS
ENABLE RISE/FALL TIME	T_{ER}, T_{EF}	20		nS
RS, R/W SET UP TIME	T_{AS}		40	nS
DATA DELAY TIME	T_{DDR}	360		nS
DATA SETUP TIME	T_{DSW}		60	nS
HOLD TIME	T_H		10	nS

A 4-bit Command

So to send a 4-bit LCD command:

1. Set appropriate values to RS, R/W, and data pins.
2. Set the enable pin to HIGH.
3. Delay a microsecond
4. Set the enable pin to LOW.

Command Table

SNO	Instruction for LCD	Hex code
1	If you want to display content in one line in 5x7 matrix	0x30
2	If you want to display content in two lines in 5x7 matrix	0x38
3	If you display 4 bit data in one line in 5x7 matrix	0x20
4	If you display 4 bit data in two lines in 5x7 matrix	0x28
5	entry mode	0x06
6	To clear the display without clearing the ram content	0x08
7	Making the cursor on and also display on	0x0E
8	Making the cursor off and also display off	0x0C
9	Displaying the data on cursor blinking	0x0F
10	Shifting complete display data to left side	0x18
11	Shifting complete display data to right side	0x1C
12	Moving cursor to one place or one character left	0x10
13	Moving cursor to one place or one character RIGHT	0x14
14	Clearing the complete display including RAM DATA	0x01
15	Set DDRAM address on cursor position	0x80+add

Command vs Display mode

- For sending command to LCD
  - E=1; enable pin should be high
  - RS=0; Register select should be low
  - R/W=0; Read/Write pin should be low.
- For displaying data in LCD
  - E=1; enable pin should be high
  - RS=1; Register select should be high
  - R/W=0; Read/Write pin should be low.

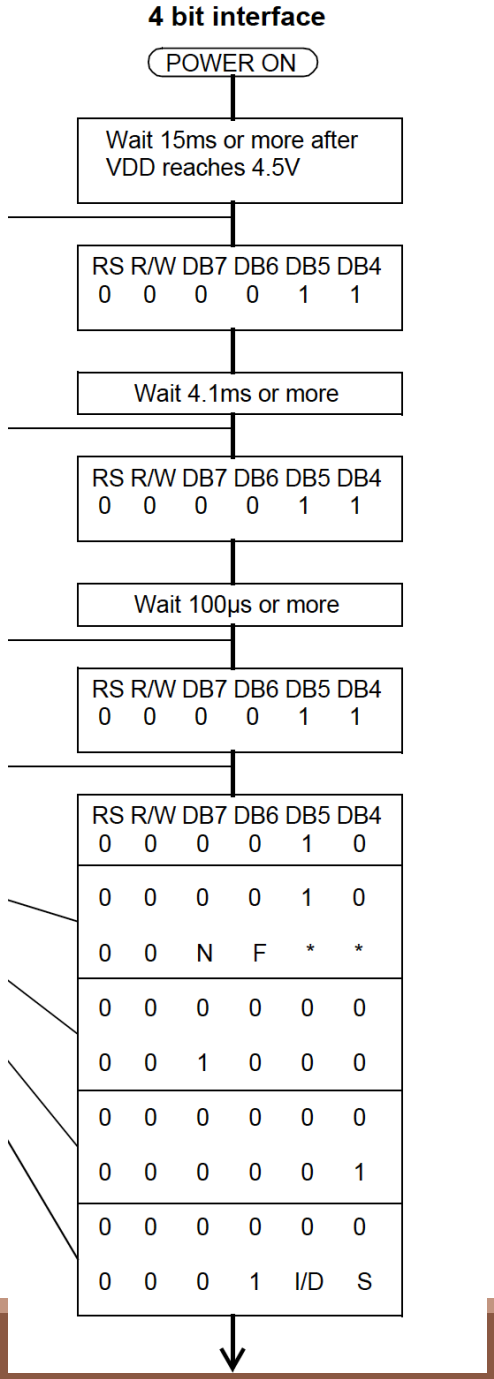
LCD Initialization

LCD Initialization

- The most difficult part of working with the LCD is properly initializing it
- This is really the most difficult because sending a command correctly is difficult and the Datasheet is not that clear!

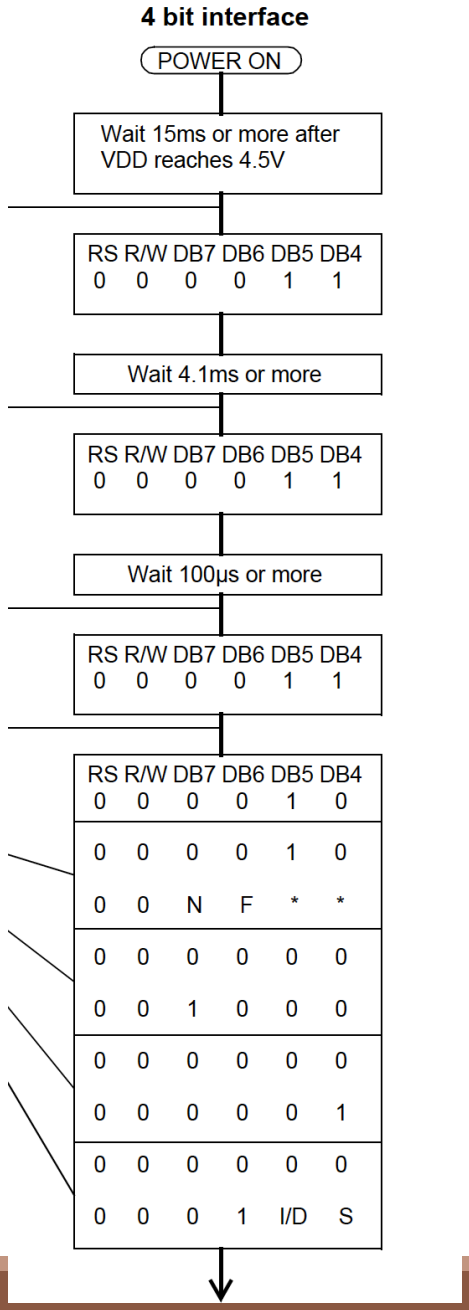
LCD Initialization

Here is the LCD initialization sequence.



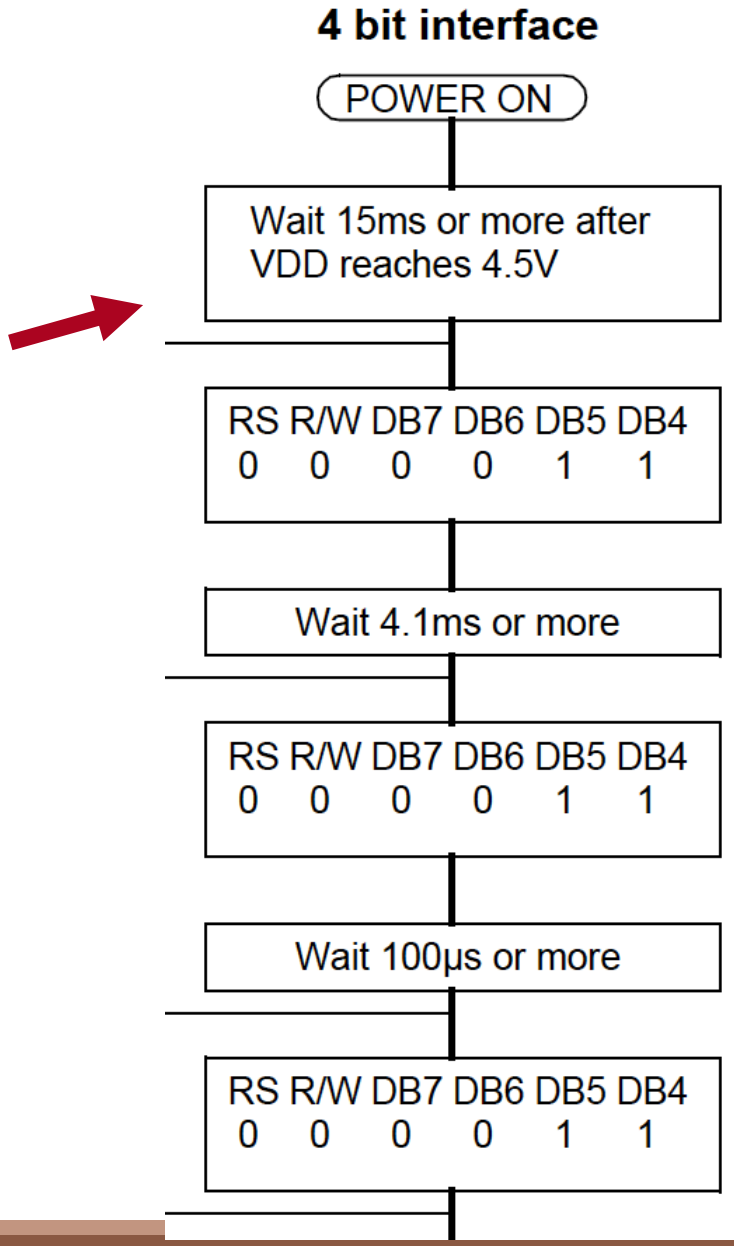
LCD Initialization

Let’s zoom in on the first half.



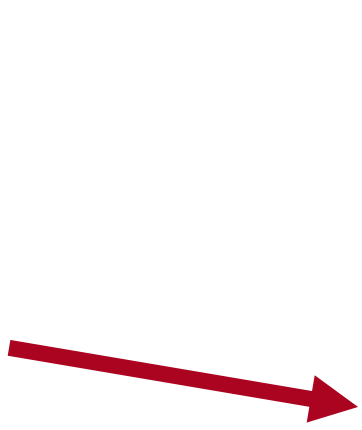
LCD Initialization

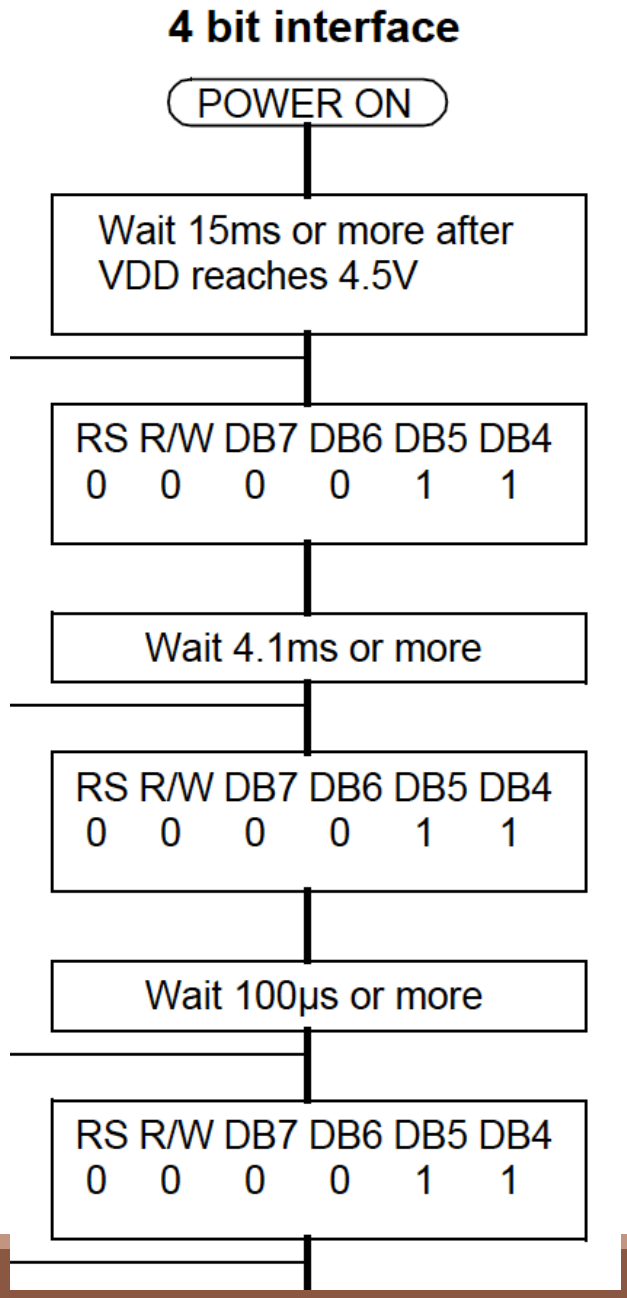
So the first thing is just to delay 15 ms. That's simple enough.



LCD Initialization

Next thing is to set the pins to these values.

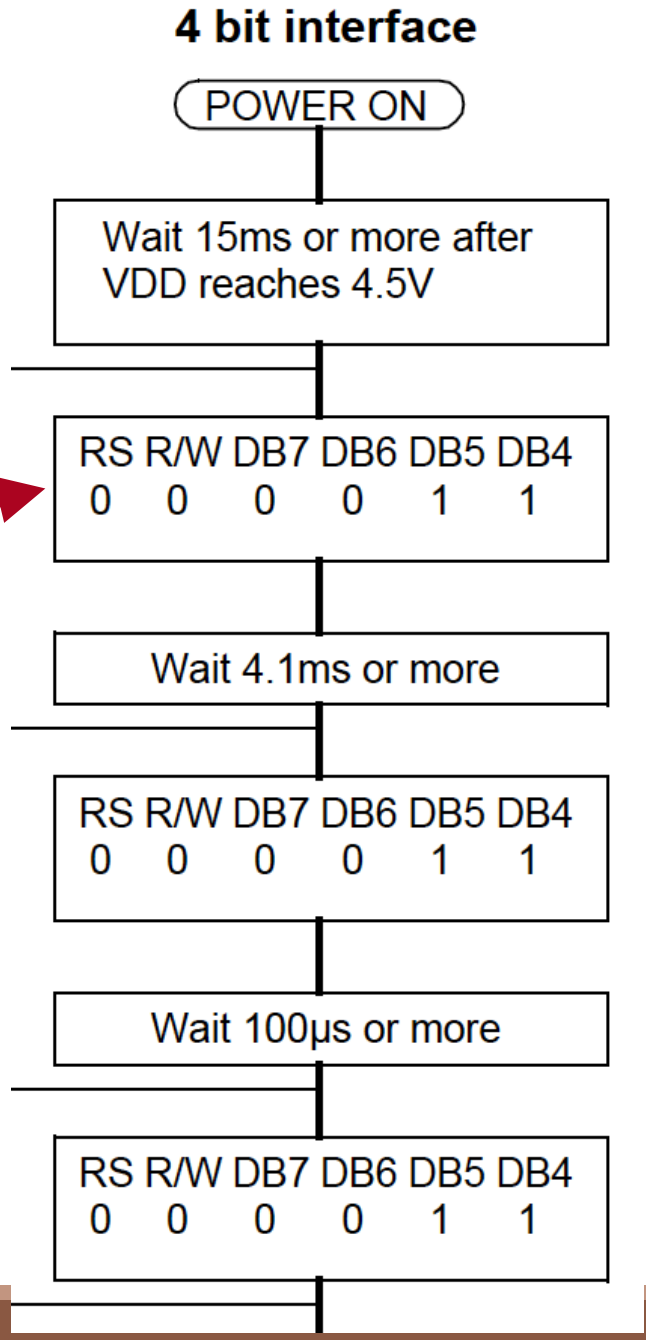




LCD Initialization

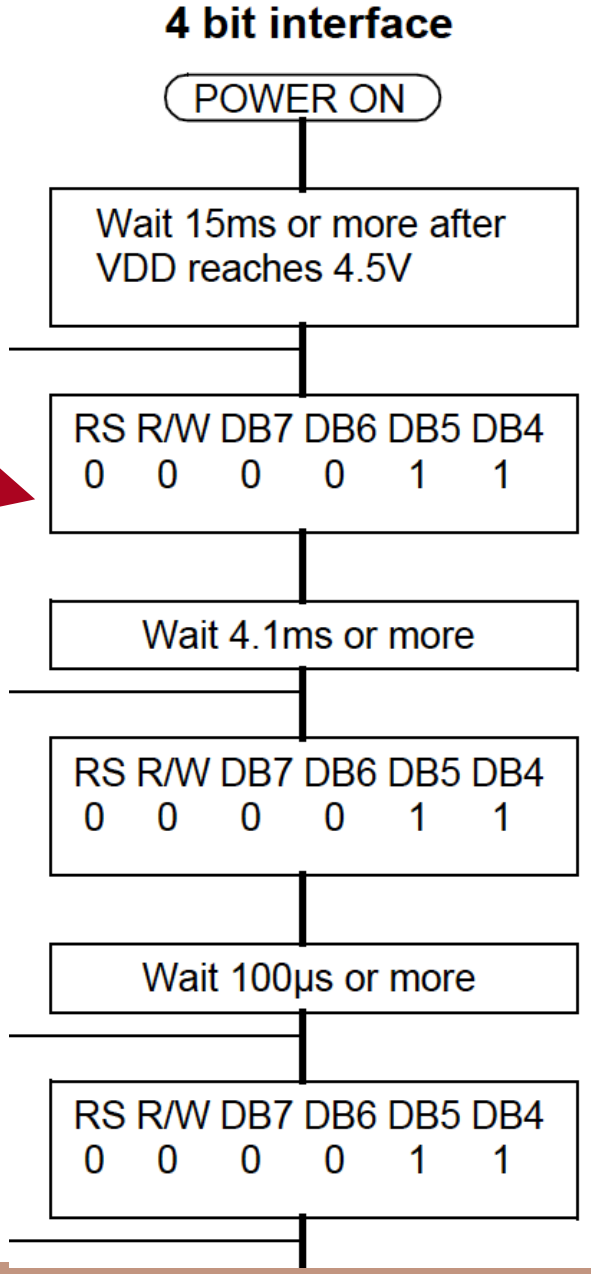
Next thing is to set the pins to these values.

It’s not clear here, but it wants you to send this as a command, meaning you need to use the enable signal like we mentioned before.



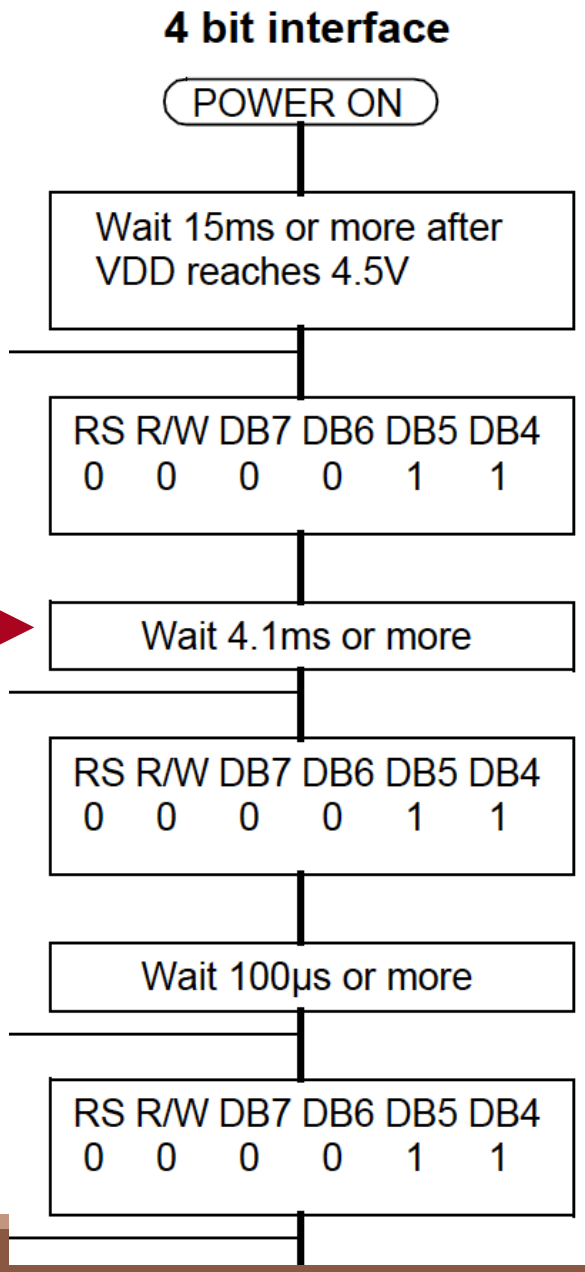
LCD Initialization

Also, we will be sending 8-bit commands, but the initialization starts with 4-bit commands.



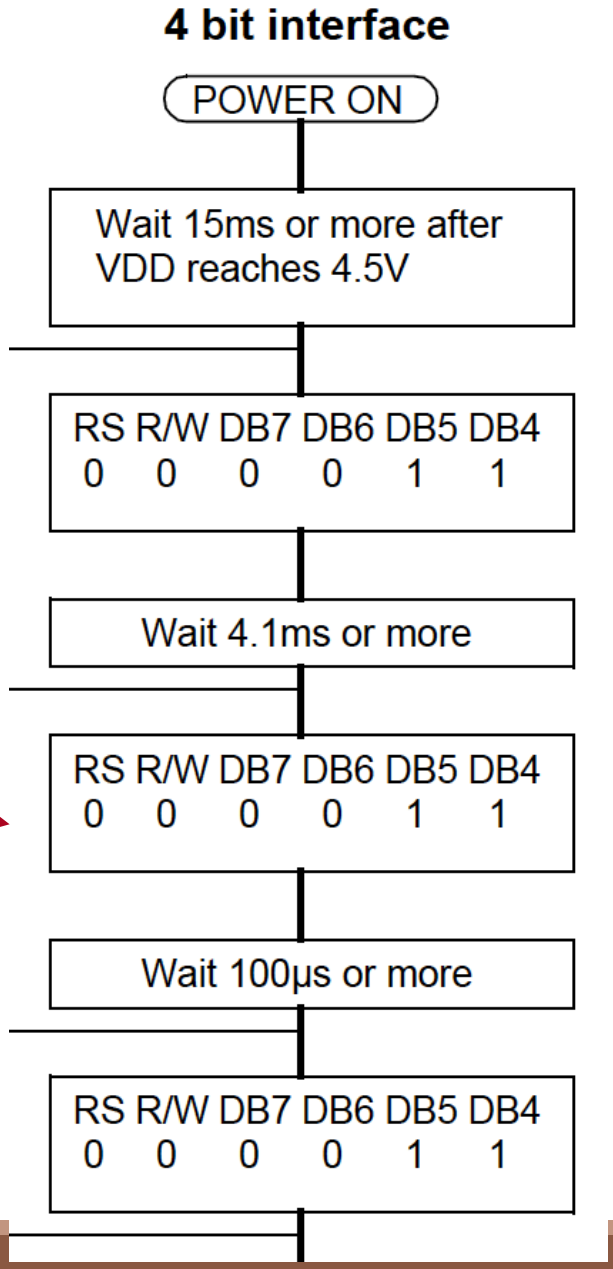
LCD Initialization

Delay again.

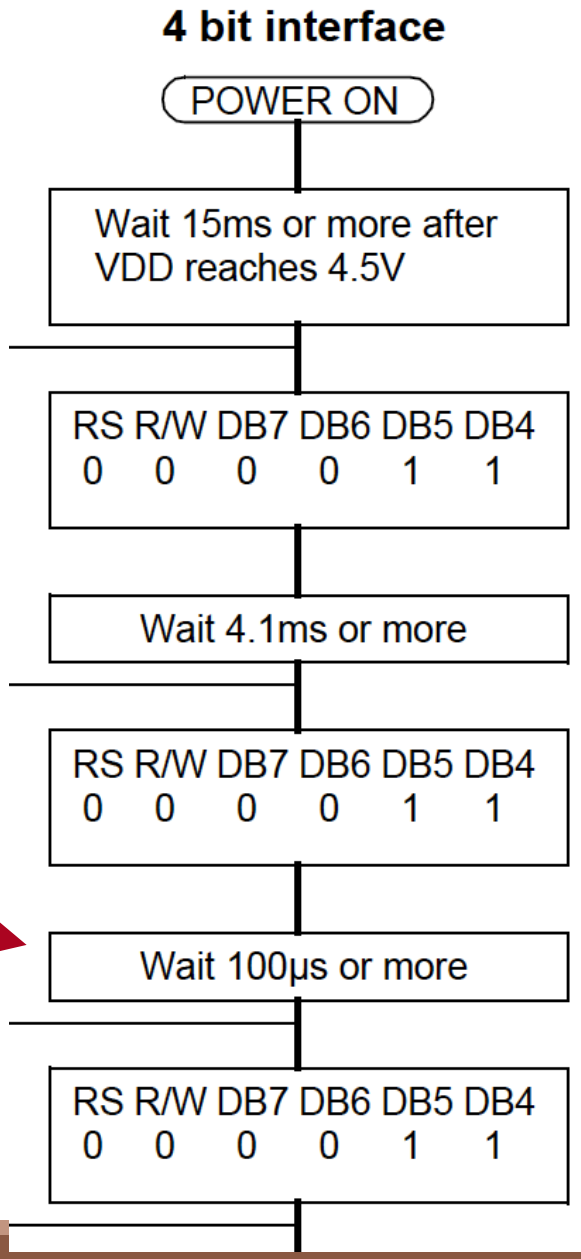


LCD Initialization

Another command



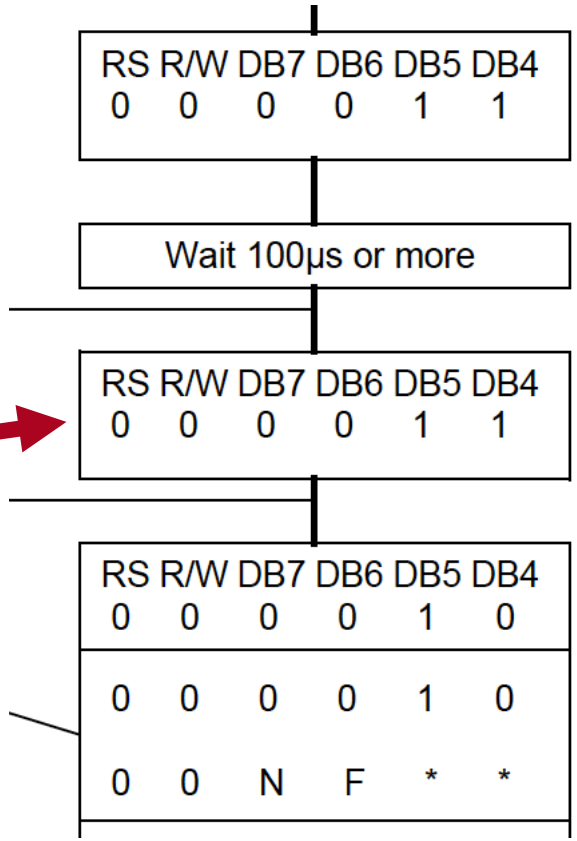
LCD Initialization



Another delay

LCD Initialization

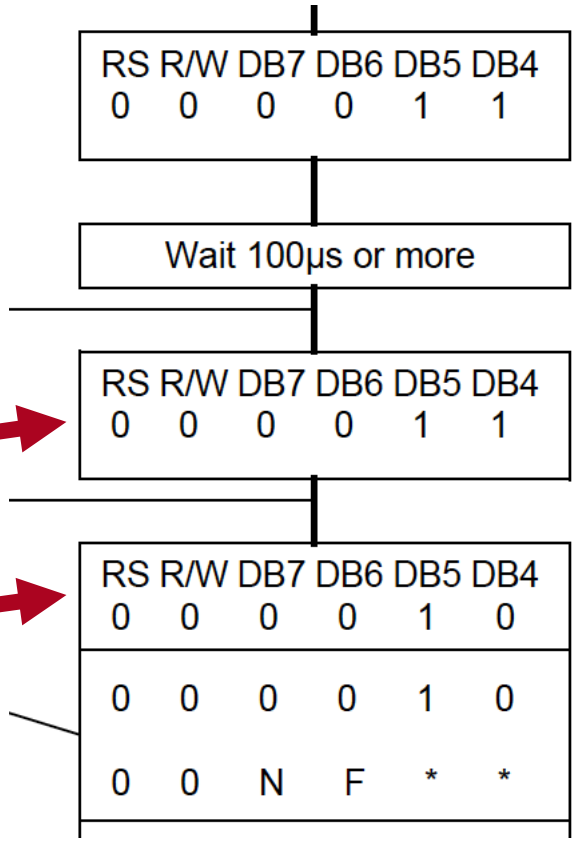
Ok, so there's this command here.



LCD Initialization

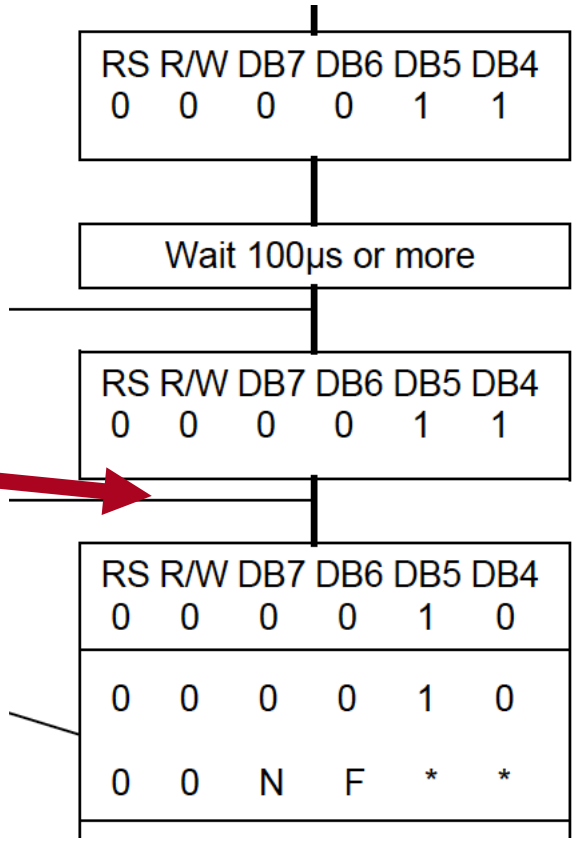
Ok, so there's this command here.

But there's also another here…



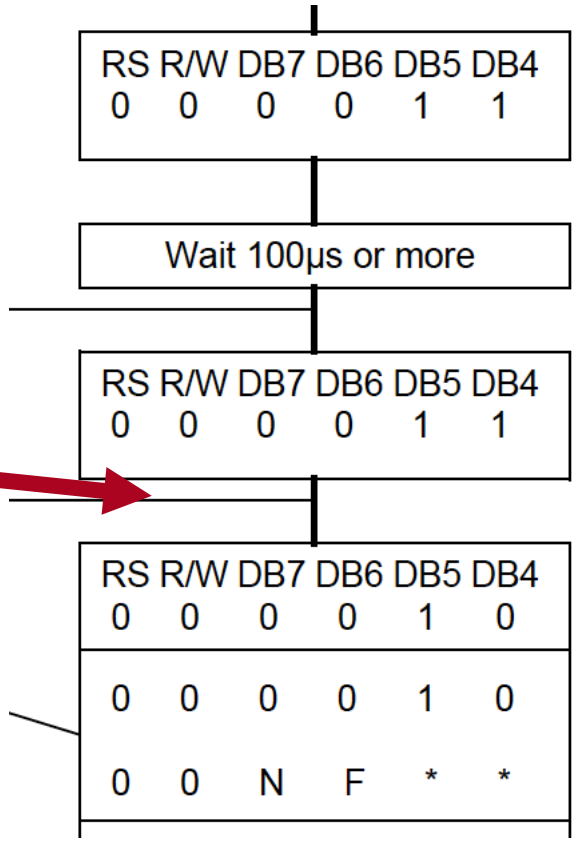
LCD Initialization

What happens in-between?



LCD Initialization

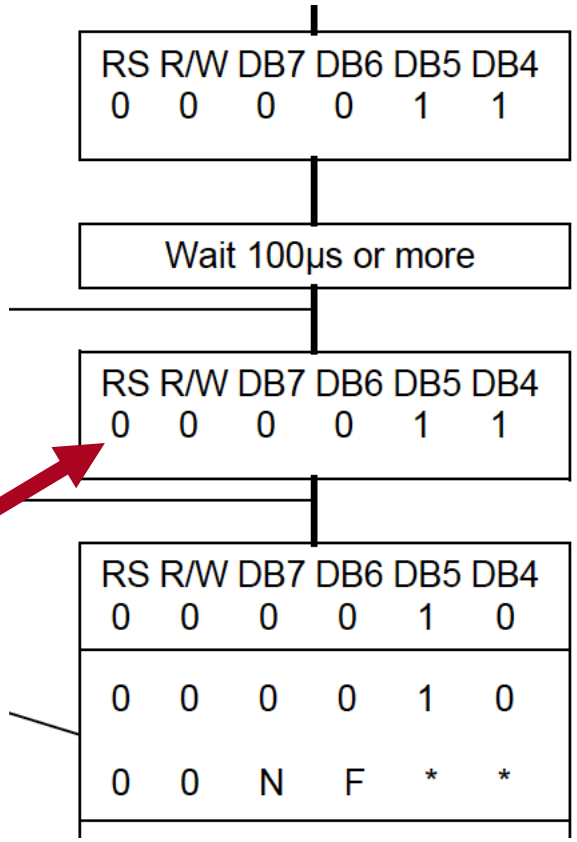
The data sheet puts a gap here...



LCD Initialization

The data sheet puts a gap here…

To show that after this command

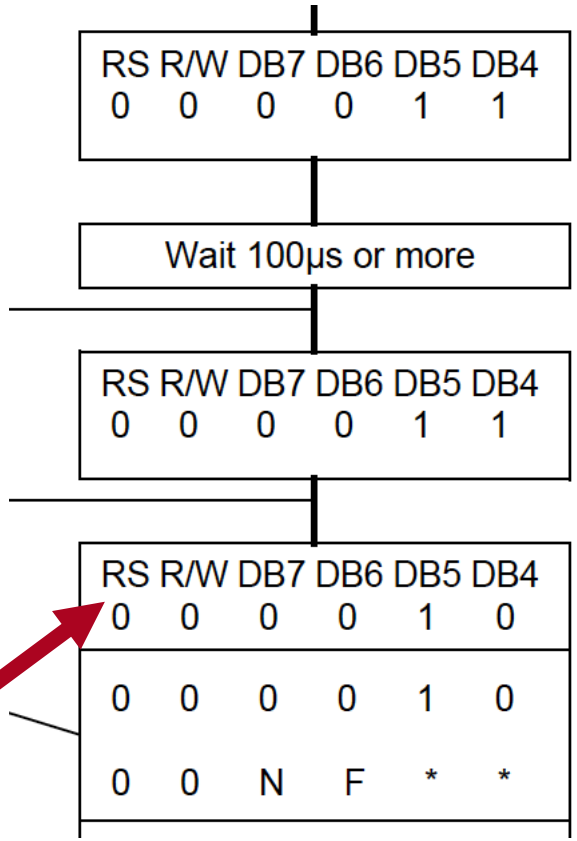


LCD Initialization

The data sheet puts a gap here…

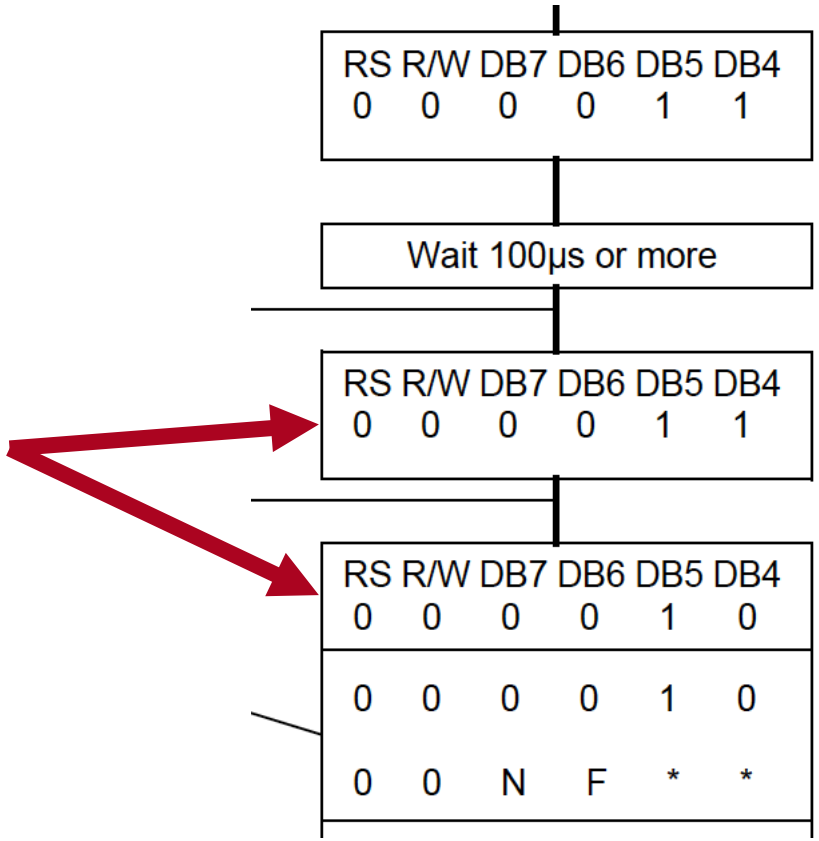
To show that after this command

We are now in 4-bit interface mode.



LCD Initialization

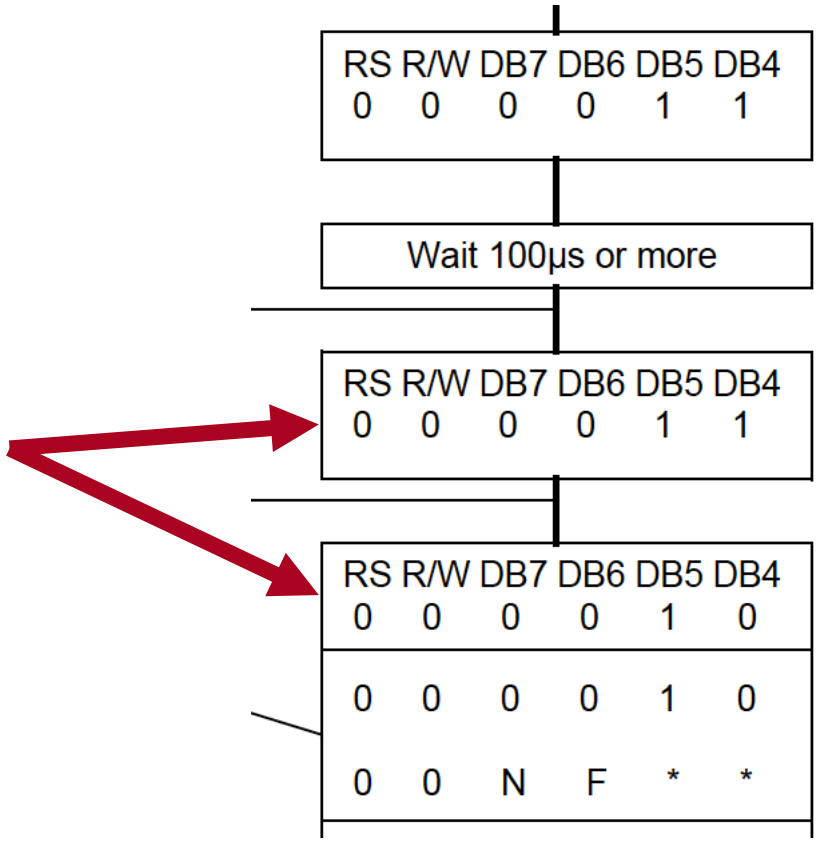
So these two 4-bit commands



LCD Initialization

So these two 4-bit commands

Are collectively our first 8-bit command.



8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

So here is a row from the command table.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

We have a name for the command.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

Values for all of the digital pins.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

How do we send 8 bits using the 4-bit interface?
The data sheet is not clear on this.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

First send the top four bits.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

Then the next four bits.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

A description is here too.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

And an execution time.

8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

What's the importance of this?

8-bit Commands

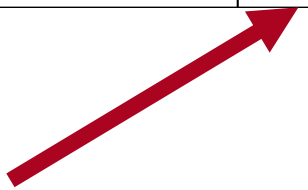
Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

After issuing this command.

8-bit Commands

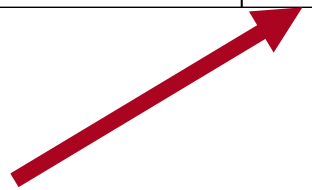
Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms

The LCD ignores input for this long.



8-bit Commands

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms



So when issuing these 8-bit commands, we have to delay afterwards.

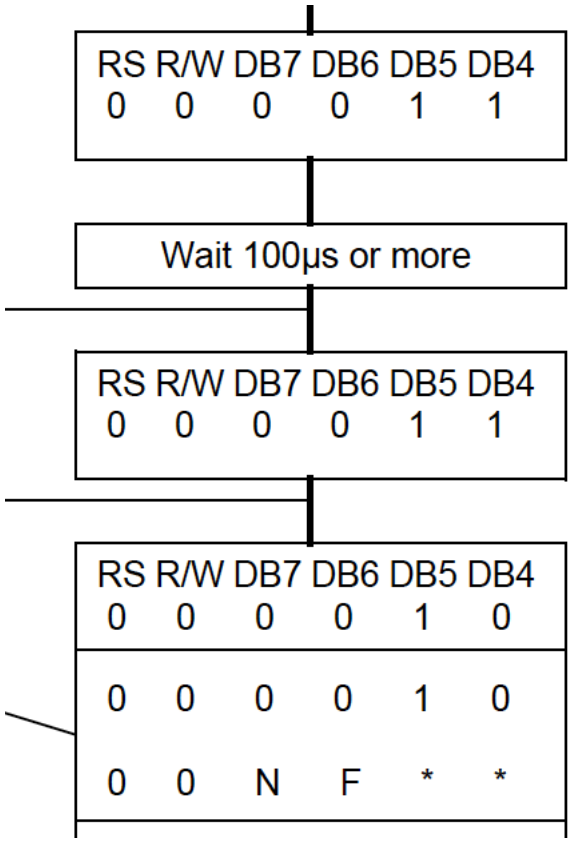
8-bit Commands

1. Set RS, R/W and the four MSb to appropriate data pins
    1. Enable pin HIGH
    2. Delay 1 microsecond
    3. Enable pin LOW
2. Set four LSb to appropriate values
    1. Enable pin HIGH
    2. Delay 1 microsecond
    3. Enable pin LOW
3. Delay for execution time

8-bit Commands

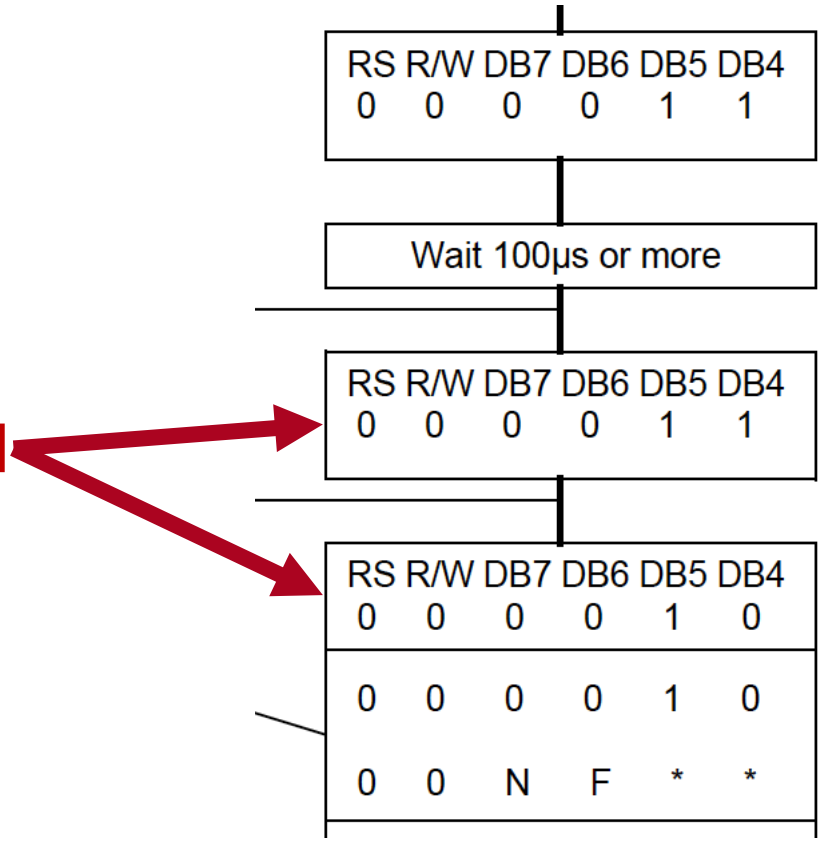
1. Send top four bits
2. Send bottom four bits
3. Delay for execution time

LCD Initialization



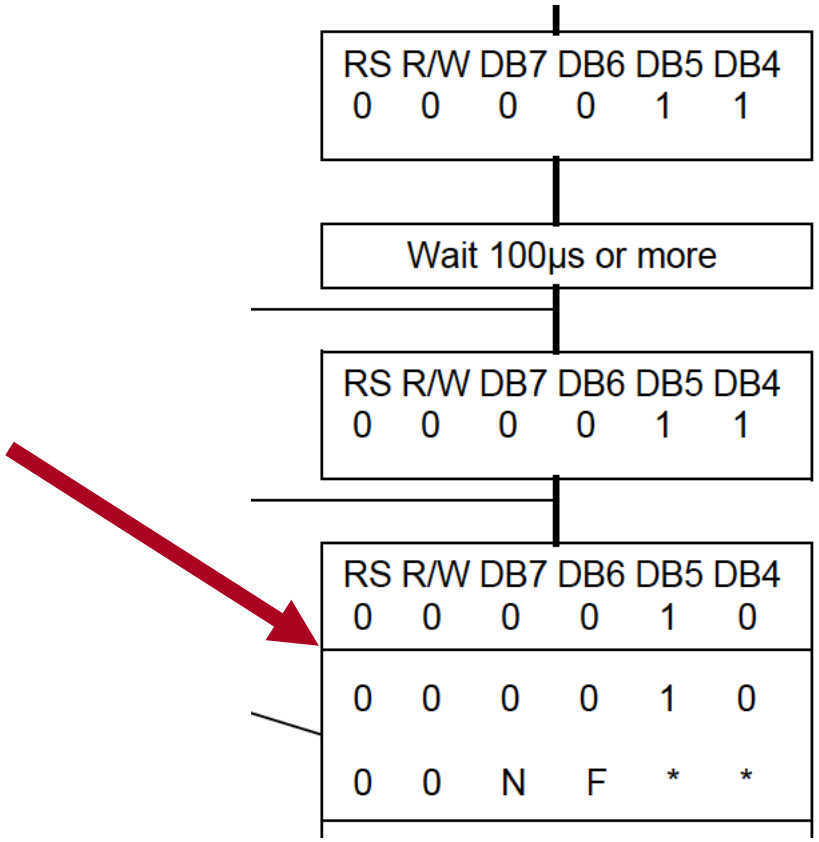
LCD Initialization

Since this is an 8-bit command



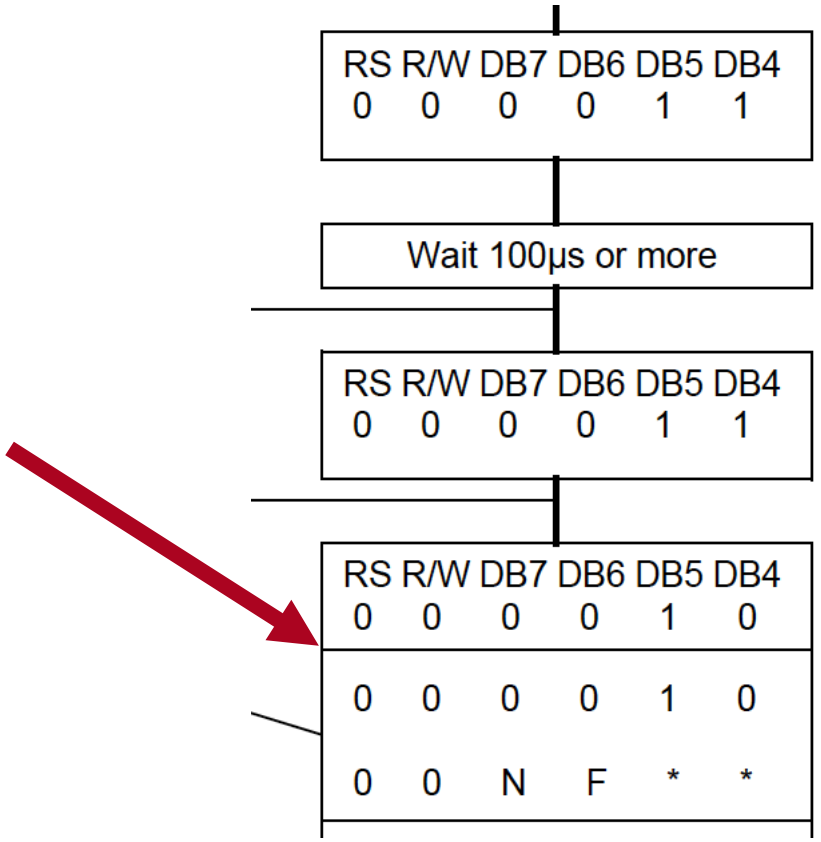
LCD Initialization

We need to delay afterwards according to which command this is.



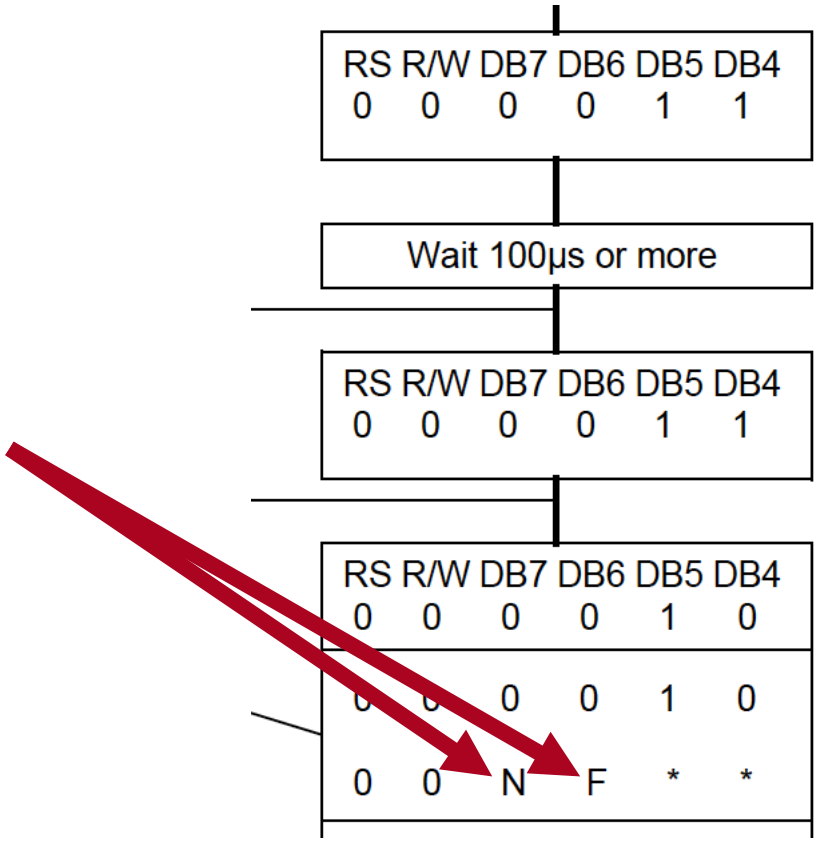
LCD Initialization

The same goes for the remaining 8-bit commands.



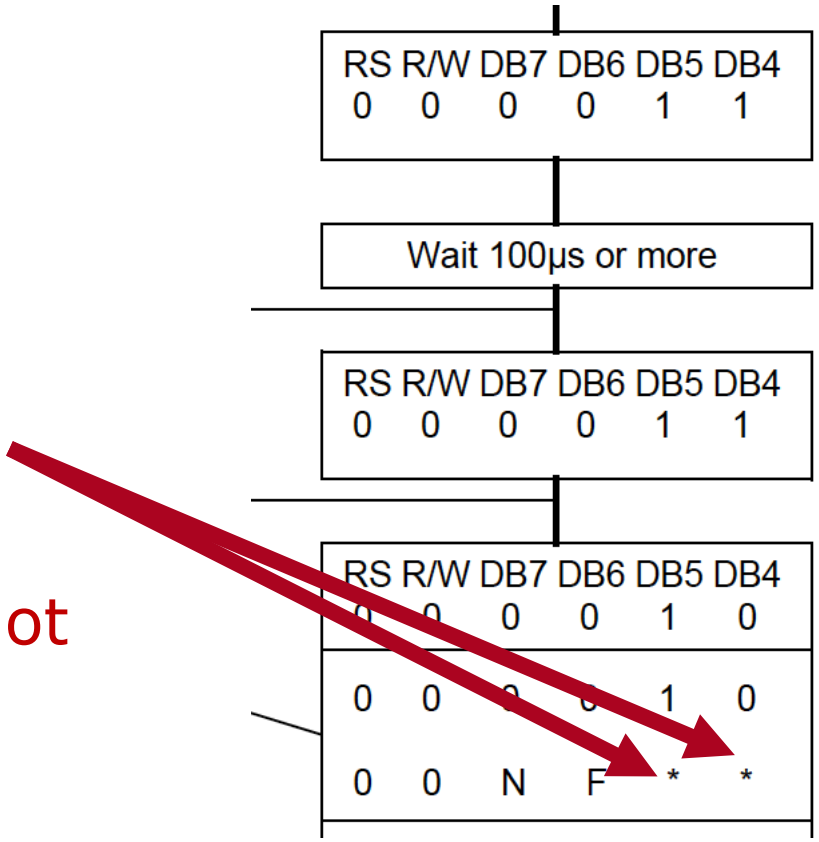
LCD Initialization

Read the data sheet to know what to put in these positions.



LCD Initialization

Asterisks almost always denote “don’t care bits” meaning their value does not affect the outcome.



More information on the Initialization of LCD

Initializing LCD

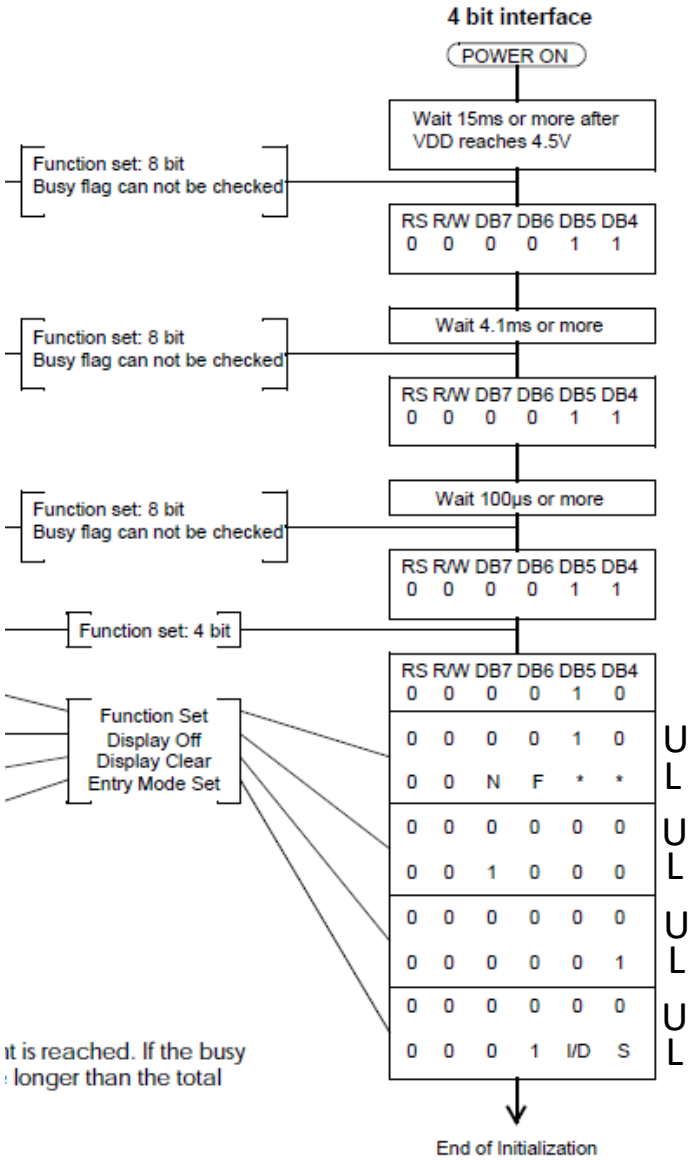
- The 'Internal Reset' technique described in the data sheet is relied upon by many programmers but this is not a wise choice.
- The disclaimer note clearly states that this technique will fail if the power supply does not meet certain specifications, specifications that are buried elsewhere in the datasheet.
- It is prudent to write a software to initialize the LCD

```
void initLCDProcedure ()
```

- The flowcharts showing the initialization procedure listed on many of the datasheets floating around are all equally ambiguous! The best explanation that I've found is located here:

http://web.alfredstate.edu/faculty/weimandn/lcd/lcd_initialization/lcd_initialization_index.html

LCD Initialization by instruction - 4 bit interface



← 4 bit command using pins D4:D7

← 4 bit command using pins D4:D7

← 4 bit command using pins D4:D7

← 4 bit command using pins D4:D7

← 8 bit command using pins D4:D7 (upper bits then lower bits)

← 8 bit command using pins D4:D7 (upper bits then lower bits)

← 8 bit command using pins D4:D7 (upper bits then lower bits)

← 8 bit command using pins D4:D7 (upper bits then lower bits)

Initialization by Software Instruction

Step 1. Power on, then delay > 50 ms
The data sheet says to delay 15 ms. Other datasheets say to delay 40 ms. I chose 50 ms. Since this delay only occurs once it doesn't make sense to try to speed up program execution time by skimping on this delay.

Step 2. Instruction 0b0011 , then delay > 4.1 ms
This is a special case of the **Function Set** instruction where the lower four bits are irrelevant. These four bits are not shown on the flowcharts because the host microcontroller does not usually implement them at all (as opposed to the 8-bit mode where they are implemented as '0's). This first instruction, for some unexplained reason, takes significantly longer to complete than the ones that come later.

Step 3. Instruction 0b0011 then delay > 100 us
This is a second instance of the special case of the **Function Set** instruction. The controller does not normally expect to receive more than one 'Function Set' instruction so this may account for the longer than normal execution time.

Initialization by Software Instruction – 4bit

Step 4. Instruction 0b0011, then delay > 100 us
This is a third instance of the special case of the **Function Set** instruction. By now the LCD controller realizes that what is really intended is a 'reset', and it is now ready for the real **Function Set** instruction followed by the rest of the initialization instructions. The flowcharts do not specify what time delay belongs here. I have chosen 100 us to agree with the previous instruction. It may be possible to check the busy flag here.

Step 5. Instruction 0b0010, then delay > 100 us
Here is where the LCD controller is expecting the 'real' **Function Set** instruction which, in the 8-bit mode, would start with 0011. Instead, it gets a **Function Set** instruction starting with 0010. This is it's signal to again ignore the lower four bits, switch to the four-bit mode, and expect another 'real' **Function Set** instruction. Once again the required time delay is speculation.

The LCD controller is now in the 4-bit mode. This means that the LCD controller reads only the four high order data pins each time the Enable pin is pulsed . To accommodate this, the host microcontroller must put the four high bits on the data lines and pulse the enable pin, it must then put the four low bits on the data lines and again pulse the enable pin. There is no need for a delay between these two sequences because the LCD controller isn't processing the instruction yet. After the second group of data bits is received the LCD controller reconstructs and executes the instruction and this is when the delay is required.

Initialization by Software Instruction – 4bit

- **Step 6. Instruction 0b0010, then 0b1000, then delay > 53 us or check BF**
  This is the real **Function Set** instruction. This is where the interface, the number of lines, and the font are specified. Since we are implementing the 4-bit interface we make D = 0. The number of lines being specified here is the number of 'logical' lines as perceived by the LCD controller, it is NOT the number of 'physical' lines (or rows) that appear on the actual display. This should almost always be two lines so we set N=1 (go figure). There are very few displays capable of displaying a 5x10 font so the 5x7 choice is almost always correct and we set F=0.

- **Step 7. Instruction 0b0000, then 0b1000 then delay > 53 us or check BF**
  This is the **Display on/off Control** instruction. This instruction is used to control several aspects of the display but now is NOT the time to set the display up the way we want it. The flow chart shows the instruction as 00001000, not 00001DCB which indicates that the Display (D), the Cursor (C), and the Blinking (B) should all be turned off by making the corresponding bits = 0.

  Step 8. Instruction 0b0000, then 0b0001 then delay > 3 ms or check BF
  This is the **Clear Display** instruction which, since it has to write information to all 80 DDRAM addresses, takes more time to execute than most of the other instructions. On some flow charts the comment is incorrectly labeled as 'Display on' but the instruction itself is correct.

Initialization by Software Instruction – 4 bit

- **Step 9. Instruction 0b0000, then 0b0110, then delay > 53 us**
  This is the **Entry Mode Set** instruction. This instruction determines which way the cursor and/or the display moves when we enter a string of characters. We normally want the cursor to increment (move from left to right) and the display to not shift so we set I/D=1 and S=0. If your application requires a different configuration you could change this instruction, but my recommendation is to leave this instruction alone and just add another **Entry Mode Set** instruction where appropriate in your program.

  Step 10. Initialization ends
  This is the end of the actual intitalization sequence, but note that step 6 has left the display off.

  Step 11. Instruction 0b0000, then 0b1100, then delay > 53 us or check BF
  This is another **Display on/off Control** instruction where the display is turned on and where the cursor can be made visible and/or the cursor location can be made to blink. This example shows the the display on and the other two options off, D=1, C=0, and B=0.

About Delays

- Make sure that the LCD controller has finished executing an instruction before sending it another one, otherwise the second instruction will be ignored.
- The data sheet gives specific times for the power up delay and for the first few instructions. After that it specifies that you must either check the busy flag to see if the LCD controller is finished or wait a sufficient amount of time, a time that is longer than the instruction execution time.
- The Instruction Set has a column which lists 'typical' instruction execution times and at the bottom of that column most data sheets indicate the clock speed (LCD controller clock) for which that time is valid. Data sheet indicates that you should add 10% to the value when using software time delays.

In summary…

In summary

- An LCD is one way to display information graphically
  - We use one that displays 16x2 ASCII
- Sending a command to the LCD requires sending 4-bits in parallel twice to make a single 8-bit command

4 bit commands and 8 bit commands are both implemented using data pins [D4:D7]

- Why are we only using 4 pins on the LCD board?
  - This is important to learn for embedded systems that have limited number of pins from microcontroller.
  - The LCD 1602 interface chip provides the ability of using 4 data pins to implement an 8 bit command. (setup during LCD initialization).
- Which commands require 8 bits? (Most of them)
  - character write commands.
  - Move cursor command.
  - All of the commands listed on page 48 of the Hantronix data sheet.
  - Software initialization –the last 4 steps of the sequence
- Which commands require 4 bits?
  - Only the first four commands in the initialization step.

Writing to LCD

To send an LCD command:

1. Set appropriate values to RS, R/W, and data pins.
2. Set the enable pin to HIGH.
3. Delay a microsecond (this delay is to allow for data to be set up and written to LCD.
   - After the enable pin goes HIGH there needs to be a delay before you set the enable LOW again.
   - This is not the execution delay time for the particular commant
4. Set the enable pin to LOW.
5. Now delay for the execution time that the LCD driver needs perform the command.

For Lab#3 you will have to write the following Functions in LCD.cpp

```
void initLCDPins()  Initialize pins to output

void fourBitCommandWithDelay(unsigned char data, unsigned int delay)

void eightBitCommandWithDelay(unsigned char command, unsigned int delay)

void writeCharacter(unsigned char character)

void writeString(const char *string)

void moveCursor(unsigned char x, unsigned char y)

void initLCDProcedure()

void initLCD(){
       initLCDPins();
       initLCDProcedure();
```

Character strings

- When you need to display “words” or “character strings” on the LCD display
  - In C language you can present this as a “characters array” and use a “pointer” to point to the array.
  - The pointer is just the address of the first character in the array, and “pointer +1” is the second character in the array, “pointer +2” is the third…
  - By passing the pointer to an LCD_write_string function, you can cite all the characters in the array in this function and send them one by one and show them on the screen.

Character strings – LCD write

- When you are passing a string, its better use a string pointer and increment the pointer, if you are incrementing a pointer it will automatically go the next address of the variable in which you can store your character which you wanted to display. See two examples adjacent.

```
void LCD_write_string(const char *str)
{
  while(*str != '\0'){
    LCD_write(*str);
    str++;
  }
}
```

```
void LCD_write_string(unsigned char *str)

/*store address value of the string in
pntr *str */
{
int i=0;
while(str[i]!='\0')
/* loop will go on till the NULL character
 in the string */
{ LCD_write(str[i]);
 // sending data on LCD byte by byte
i++;
}
return;
}
```

4 bit command with delay

void fourBitCommandWithDelay(unsigned char data, unsigned int delay){

- Bit mask data with PORTn (bottom 4 bits of data assigned to PORTn[3:0]
- Set RS pin to low
- Set Enable to high
- Delay 1 us
- Set Enable to low
- Delay specified amount so driver can interface with LCD

8 bit Command with delay

void eightBitCommandWithDelay(unsigned char command, unsigned int delay){

- Bit mask PORTn with top four bits of command
- SET RS bit to 0
- Set enable bit to 1
- Delay 1 us
- Set enable bit to 0
- Bit mask PORTn with bottom four bits
- Set enable bit to 1
- Delay 1 us
- Set enable bit to 0
- Delay