LCD BASICS

- DATA SHEET FOR LCD
- "OLD" NOTEPACK ON THIS SECTION (P74 - P80)

Unlike 7-Segs, LCD is READ/WRITE.

Like 7-Segs, LCD is on 2 ports → PORTH (DATA)
PORTK 0:2 (CONTROL)

ONCE AGAIN, WE MUST FAKE A BUS TO COMMUNICATE W/ THE LCD. SINCE THE LCD IS R/W, AT TIMES PORT H WILL NEED TO SWITCH BETWEEN INPUTS AND OUTPUTS. THIS WILL BE BUILT INTO OUR HELPER CODE SO WE ALWAYS GET IT RIGHT!

PORT K:
- 0 → E (E FOR ENABLE) NOTE: ACTIVE HIGH!!! MUST IDLE LOW
- 1 → R/$\bar{W}$ (READING (1) OR WRITING (0))
- 2 → A (A FOR ADDRESS, LCD ONLY HAS 2)

AS WITH THE 7-SEGS, WE WILL USE HELPER MACROS TO MANAGE THE CONTROL SIGNALS. THE ORDER OF OPERATIONS HERE IS CRITICAL.

IMPORTANT: WHEN READING/WRITING, DDR MUST Δ ON PORT H:

READING? CONFIGURE FOR READING!
NOTE ORDER OF OPERATIONS HERE.
WRITING? CONFIGURE FOR WRITING!

```
// gotta be inputs, set R/W* high
#define lcd_RWUp DDRH = 0; PORTK |= 2;

// set R/W* low, // gotta be outputs
#define lcd_RWDown PORTK &= (~2); DDRH = 0xFF;

// this is *snug*, datasheet says PW_EH must be at least 450ns. (P49 + P58)
//  but &= probably implemented as a BCLR instruction (@ 50ns bus, 4 cycles, around 200ns per instruction)
//  measured on scope at ~300ns (which would make sense (6 cycles up/down instructions),
//   not exactly sure when pin changes state in instruction pair)
//  this is actually too short, but appears to work. either the device is better than expected, or does not match datasheet!
// other LCDs may require a small delay be added to E strobe operations, or data setup time during read.
#define lcd_EUp PORTK |= 1;
#define lcd_EDown PORTK &= (~1);
#define lcd_RSUp PORTK |= 4;
#define lcd_RSDown PORTK &= (~4);
          RS == ADDRESS
#define lcd_MicroDelay { char __x = 1; while (--__x); }
```

REMEMBER: MACROS SWAP LABEL FOR REST OF LINE, SPACE/PERFORMANCE TRADE-OFF.

TO TALK TO THE LCD:

WRITE: PRESENT DATA ON PTH, R/W = 0, STROBE E → DDR AS OUTPUTS.

READ: R/W = 1, E ↑, WAIT, READ PTH, E ↓ → DDR AS INPUTS.

LCD PRODUCES DATA

RS
↑ : DATA
↓ : COMMAND

LCD Instruction Table

Instruction	Code RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0	Description	Execution time (max.) when fcp or fosc is 250 kHz
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter.	15.2ms
Return Home	0	0	0	0	0	0	0	0	1	x	Sets DD RAM address 0 in address counter. Also returns shifted display to original position. DD RAM contents remain unchanged.	15.2ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift or display. These operations are performed during data write and read.	40µs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of entire display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40µs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Moves cursor and shifts display without changing DD RAM contents.	40µs
Function Set	0	0	0	0	1	DL	N	F	x	x	Sets interface data length (DL), number of display lines (N) and character font (F).	40µs
Set CG RAM Address	0	0	0	1	ACG						Sets CG RAM address. CG RAM data is sent and received after this setting.	40µs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. DD RAM data is sent and received after this setting.	40µs
Read Busy Flag & Address	0	1	BF	AC							Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	40µs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40µs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40µs
	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										DD RAM : Display Data RAM CG RAM : Character Generator RAM ACG : CG RAM address ADD : DD RAM address. Corresponds to cursor address. AC : Address counter used for both	Execution time changes when frequency changes. Example: When fcp or fosc is 270kHz:
	R/L=0 : Shifts to the left DL=1 : 8 bits, DL=0 : 4 bits N=1 : 2 lines, N=0 : 1 line F=1 : 5x10 dots, F=0 : 5x7 dots BF=1 : Internally operating BF=0 : Can accept instruction										DD and CG RAM address.	40µs x 250/270 = 37 µs

x = don't care. (No Effect)

When sending a command ($R/\bar{W} = \phi$, $RS = \phi$), the first "on" bit in the data determines the command.

Many commands have single bit options. See decoder table.

Some commands take multi-bit integral arguments that span many bits.

Note: Some commands take quite a bit of time to complete.

The device will ignore commands while busy.

We can always read a "busy flag" from the LCD (except during part of the initialization sequence). Initialization of the LCD is a little odd, as the device may be 4-bit or 8-bit interfaced. The initialization sequence requires some specific minimal delays, so we will use our timer library to help.

Toward the end of the initialization sequence, the busy flag becomes available, and from that point forward, we will use it to determine if the device is ready for a command.

All command functions we create will first check to see if the LCD is not busy (a blocking function).

RS R/W

Read Busy Flag & Address	0	1	BF	AC INTERNAL ADDRESS COUNTER	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	40µs

↳ BUSY FLAG, AS MSB OF THIS REGISTER.

PORT H

lcd_Busy:

```
// wait for the LCD to be not busy (blocking)
// service function, private
void lcd_Busy (void)
{
  unsigned char inVal = 0;

  lcd_RSDown; RS=0  // instruction   } NOTE: MACROS NOT FUNCTIONS!
  lcd_RWUp;   RW=1  // reading

  do
  {
    lcd_EUp;        E=1, GET LCD ATTENTION!
    lcd_MicroDelay    // tested @ 20MHz, delay for tDDR absolutely needed (data sheet says 360ns (P58 + P49))
                      // @50ns per clock, and average 2.5 cycles per instruction, this is about 3 average assembly instruction.
                      // this is not a long delay, but back-to-back instructions are too fast!
    inVal = PTH;      // status (d7) and address (d6:0)
    lcd_EDown;
  }
  while (inVal & 0x80);
}
```

TAKES TIME FOR LCD TO PRODUCE DATA.

READ DATA PRESENTED BY LCD.

↳ RELEASE DEVICE

↳ REPEAT UNTIL DEVICE IS NOT BUSY.

THIS FUNCTION WILL BE USED DURING INIT, AND BEFORE ALL OTHER FUNCTIONS YOU WRITE.

NEXT, WE NEED THE ABILITY TO SEND COMMANDS TO THE LCD

TO SEND A COMMAND TO THE LCD, WE WRITE TO IT, W/RS LOW. WE THEN STROBE E TO HAVE THE DEVICE LATCH THE DATA.

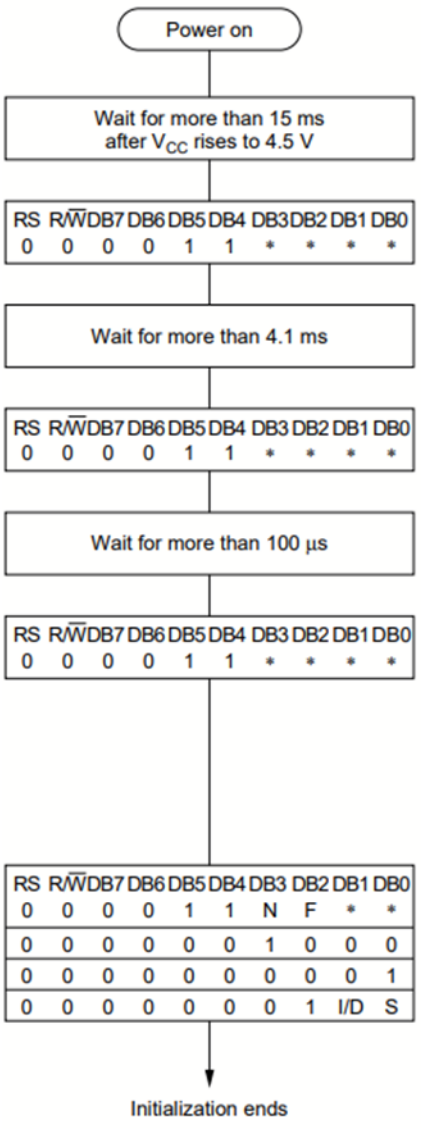
```
// send a byte to the inst reg of LCD on custom port
void lcd_Inst (unsigned char val)
{
  lcd_Busy();          // wait for LCD to be not busy

  lcd_RWDown; R/W=0    // writing
  lcd_RSDown; RS=0     // command

  PTH = val;           // present data on port

  lcd_EUp;
  lcd_EDown; STROBE E  // and... latch
}
```

↳ IDLE STATE

ASSUME DEVICE IS INITIALIZED

DEVICE WAKES UP, SEES R/W = 0, SO GRABS DATA. PROCESSES DATA AS A COMMAND (RS = 0)

WITH lcd_Busy and lcdInst, we can finally initialize the LCD! If the power supply does not come up to speed in < 10ms, the lcd needs an instruction initialization sequence, but we would do this anyway to configure the device:



INITIALIZE TIMER →

Wait for more than 40 ms after V_{CC} rises to 2.7 V

INITIAL PARTS:
E low
RS low (command) + R/W low (writing)
Port H → DDR OUTPUTS
PORT K 0:2 AS OUTPUTS
Delay 100ms (experimental) (Timer_Sleep)

BF cannot be checked before this instruction.
Function set (Interface is 8 bits long.)

PTH = 0x38
E↑ E↓
Delay 15ms (Timer_Sleep)

BF cannot be checked before this instruction.
Function set (Interface is 8 bits long.)

E↑ E↓
Delay 1ms (Timer_Sleep)

BF cannot be checked before this instruction.
Function set (Interface is 8 bits long.)

BF can be checked after the following instructions. When BF is not checked, the waiting time between instructions is longer than the execution instuction time. (See Table 6.)

Function set (Interface is 8 bits long. Specify the number of display lines and character font.) The number of display lines and character font cannot be changed after this point.

Display off

Display clear

Entry mode set

WHAT DO THESE DO?

LCDINST (0x38)
LCDINST (0x0C)
LCDINST (0x01)
LCDINST (0x06)

NOW, WITH THE DEVICE INITIALIZED, IT'S TIME TO WRITE DATA!

lcd_Data is exactly the same as lcd_Inst, except RS is high.

< write this now >

Create a program that inits the LCD, and sends out the letter 'Q' on the LCD: lcd_Data('Q');

You may need to adjust the contrast on the LCD to see the output.

20 character x 4 lines (1/16 duty)

	Column 1	2	3	...	18	19	20
line 1	00	01	02		11	12	13
line 2	40	41	42		51	52	53
line 3	14	15	16		25	26	27
line 4	54	55	56		65	66	67

The LCD tracks the display address internally. If you need to set the address, note that it is really organized as a stacked 2-line display!

Functions to write:

lcd_Inst
lcd_Init
lcd_Busy
lcd_Data

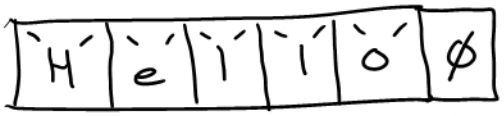
lcd_Addr
lcd_AddrXY
lcd_Clear
lcd_Home
lcd_DispControl

lcd_String
lcd_StringXY

Some functions leverage other functions!

Remember that strings are NUL(Ø) terminated:

"Hello" is really



You can use this to locate the end of the string.

BLINKING NON-CURSOR!

```
PLL_To20MHz();
lcd_Init ();

lcd_Data ('Q');
lcd_AddrXY (1, 3);
lcd_String ("Hello");
lcd_StringXY (0, 2, "This is spam!");
lcd_AddrXY (10, 1);
lcd_DispControl (0, 1);
```

USES TIMER, SO LCD.C WILL INCLUDE TIMER HEADER, PROJECT WILL INCLUDE TIMER.C IN "SOURCES".