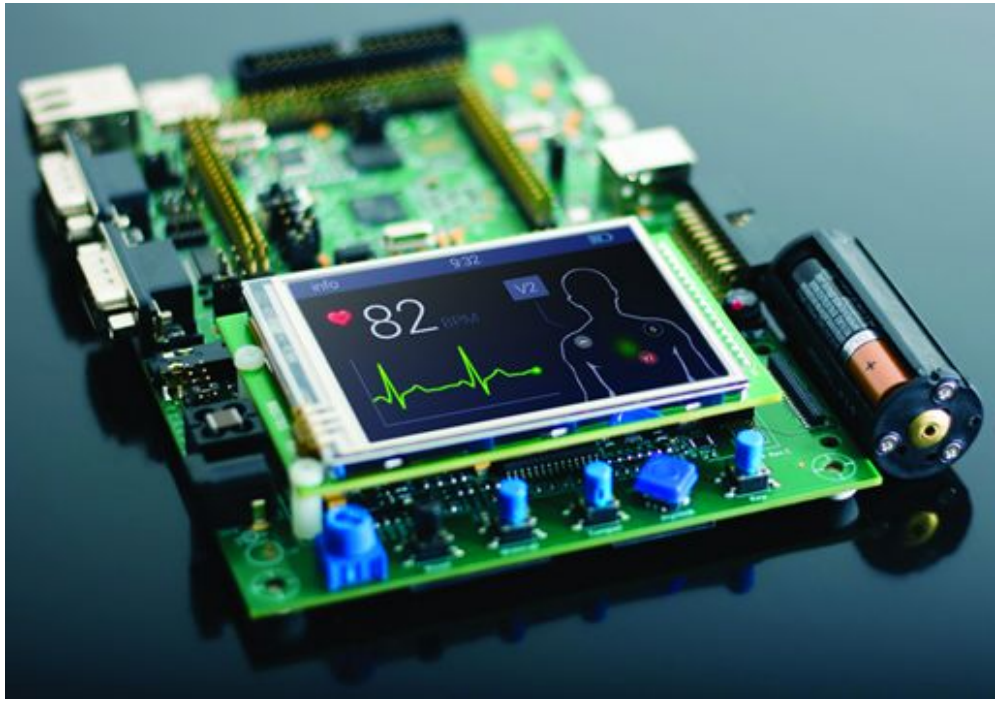
Embedded Software

CS 145/145L



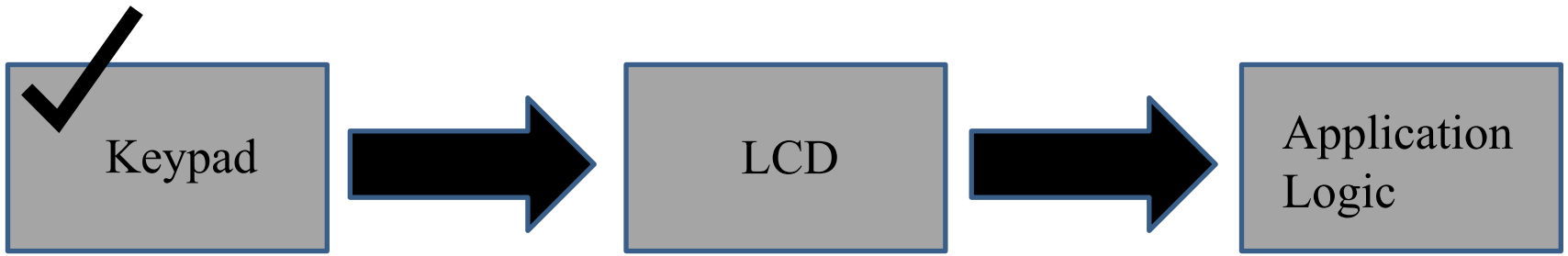
Caio Batista de Melo

Recap - Soldering

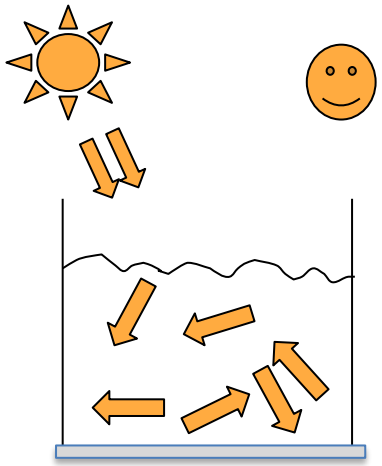
- https://www.youtube.com/watch?v=oRt_jOJ8IRU
- https://www.youtube.com/watch?v=cjDOye2xHKQ

Feel free to use the soldering iron in the lab;
You can ask your TAs for help!

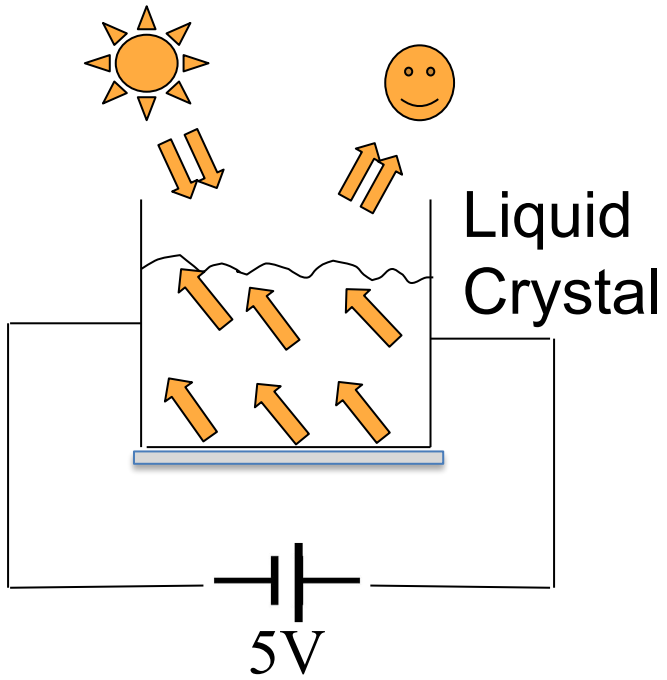
Project 2 Roadmap



How LCDs Work

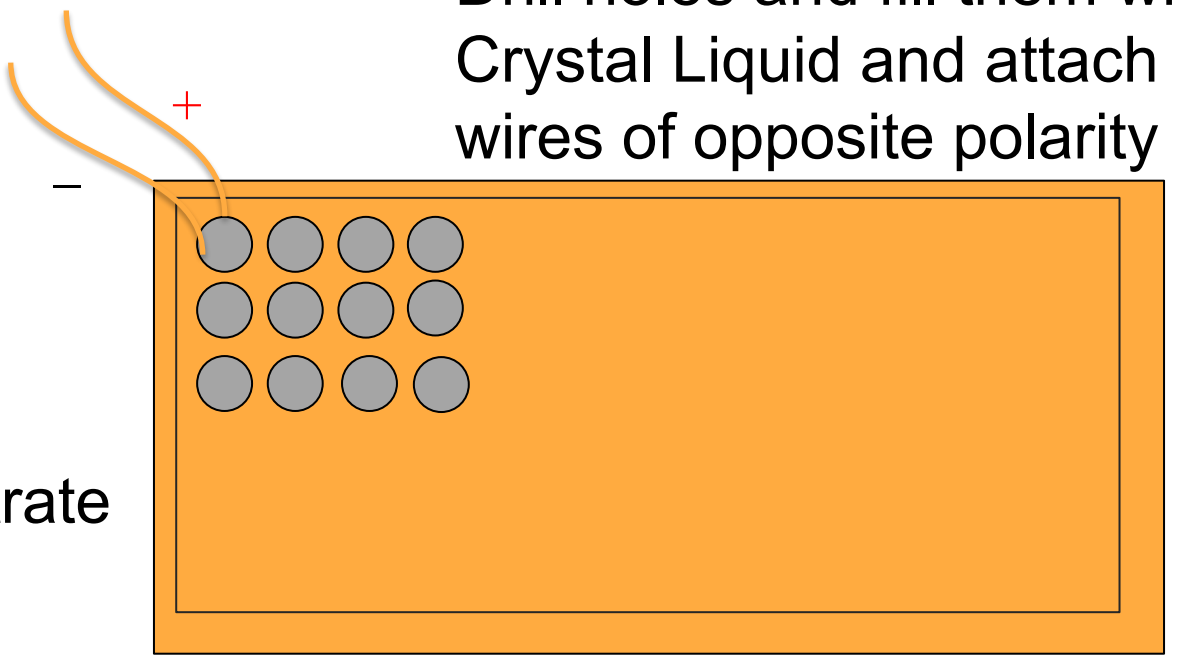


Nematic Phase of Liquid

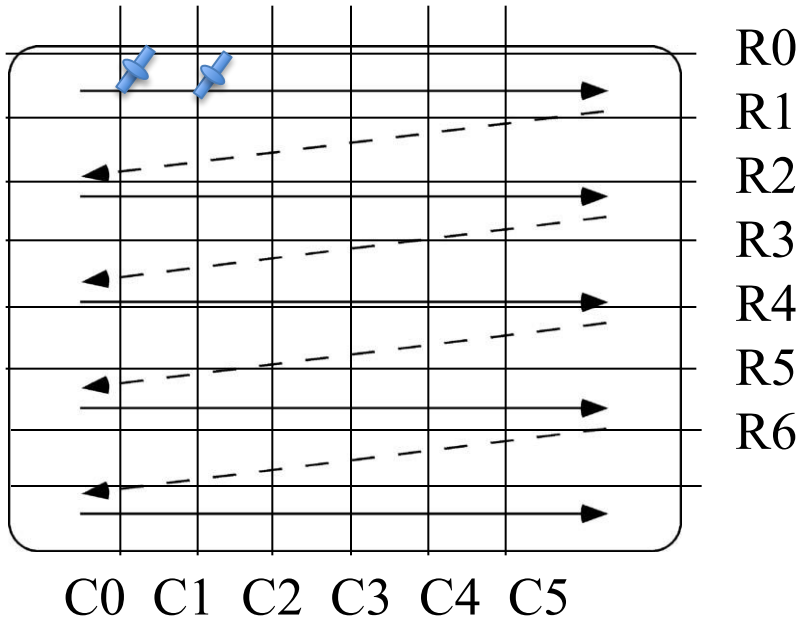


Non-aligned after applying battery and the orientation of the molecules does not allow reflection.

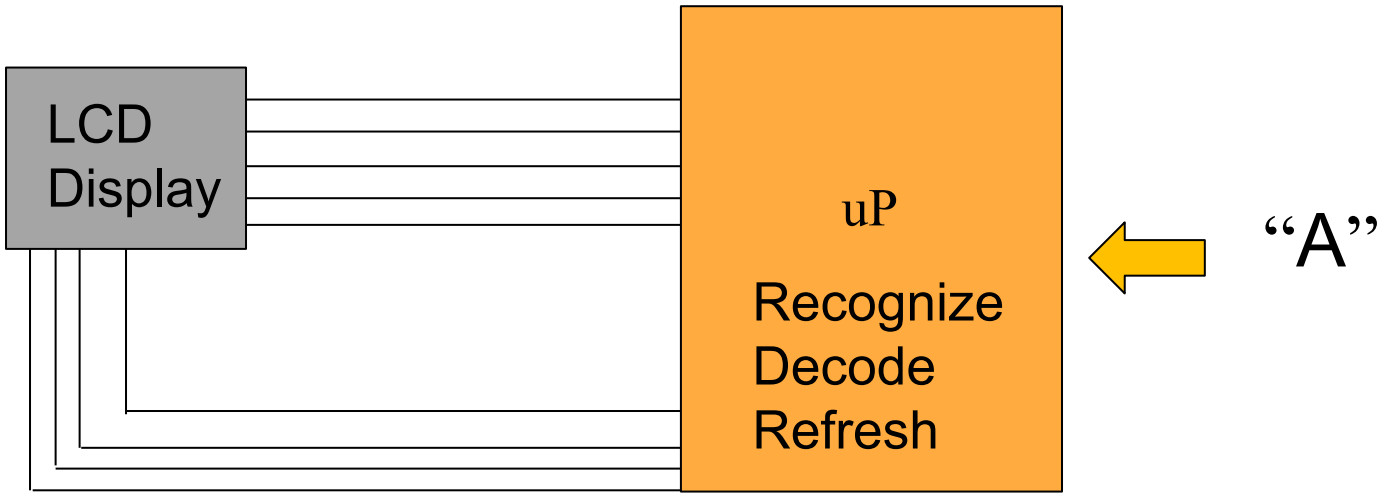
Physical Structure of LCD Display



LCD Grid

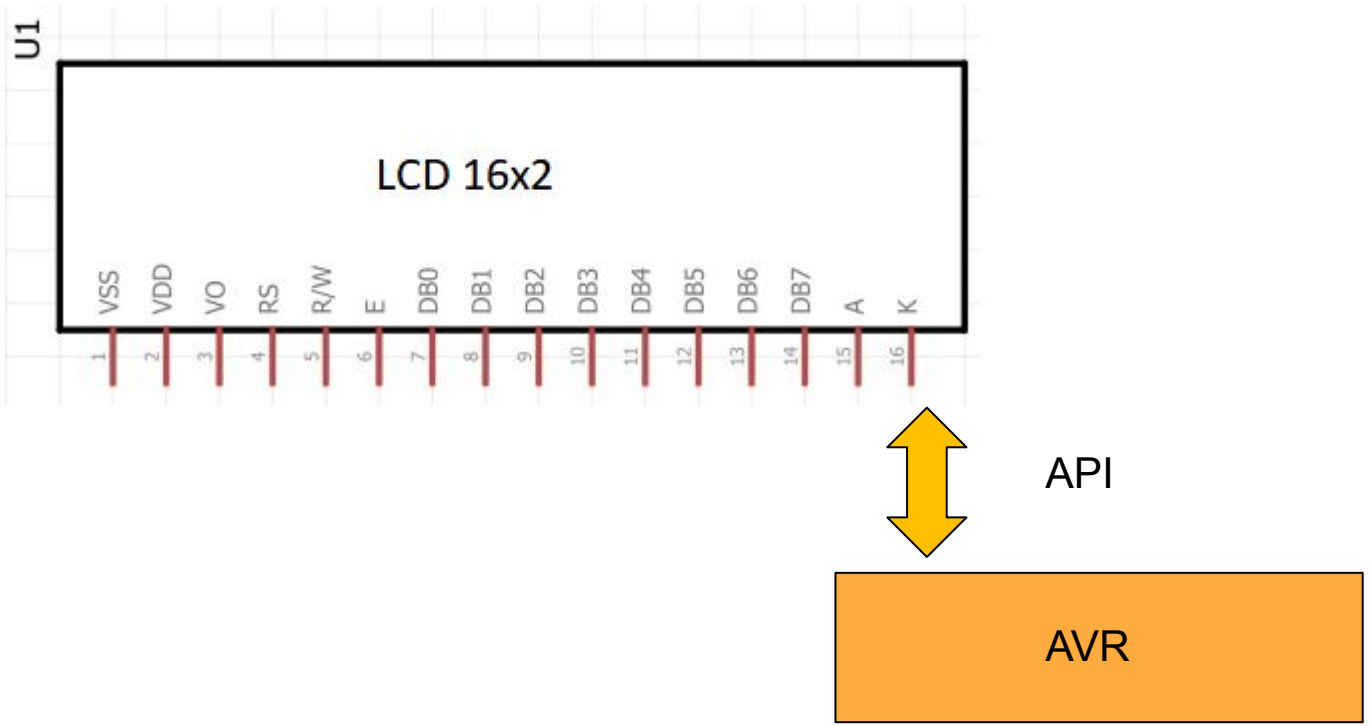


General Processor Connection and Tedious Logic



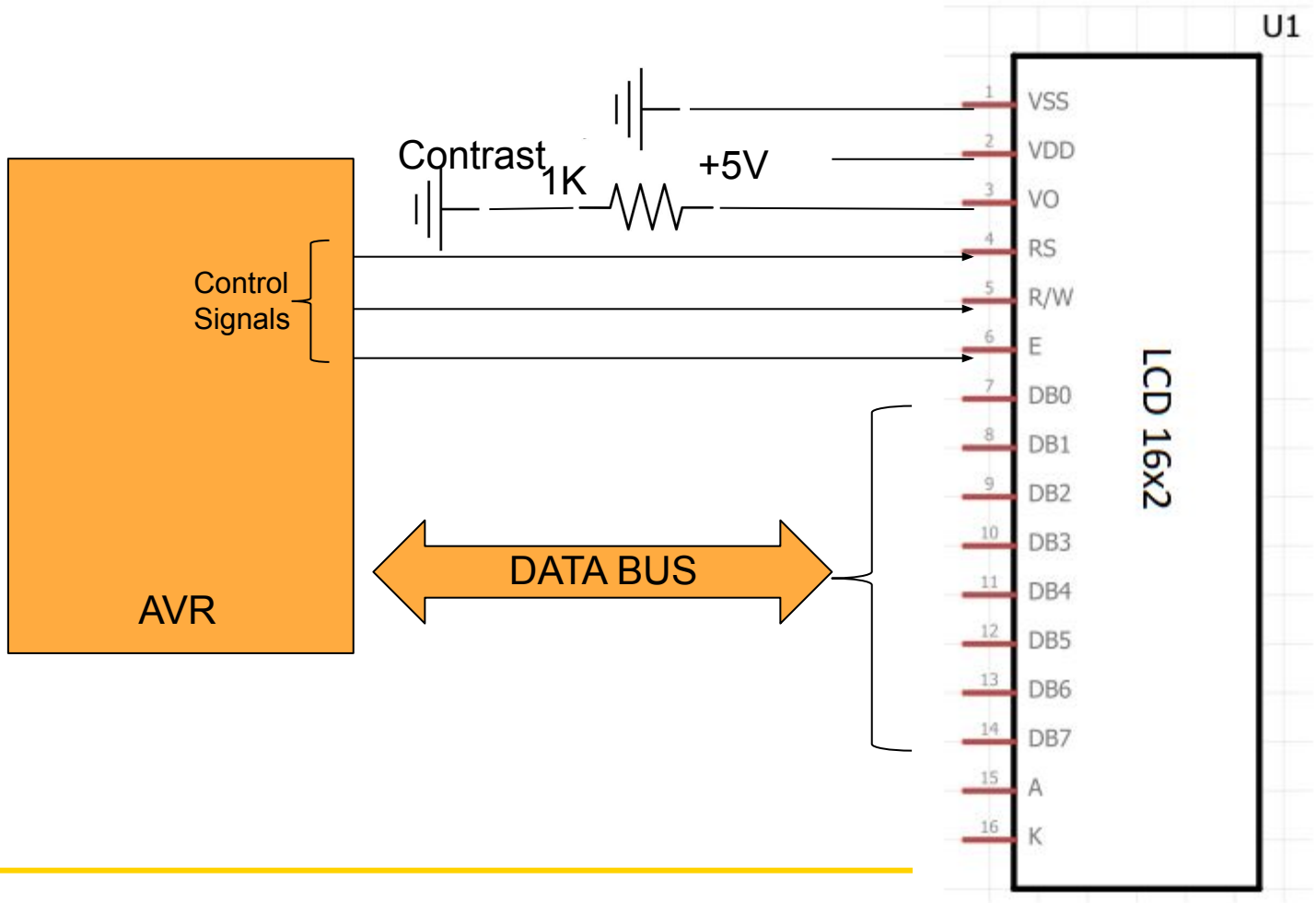
Tedious task of recognizing characters and decoding logic and also refreshing

Importance of LCD Module



Our LCD display is a small embedded system on its own and we don't need to think about small tasks of managing many connections and refreshing, scanning, pixel maps, …

Schematic Layout of Connections



Pins Layout and Description (Refer to page 11 on LCD manual)

Pin Number	Symbol
1	V_{ss}
2	V_{cc}
3	V_{ee}
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7

Pin Assignment

Signal name	No. of Lines	Input/Output	Connected to	Function
DB4 ~ DB7	4	Input/Output	MPU	4 lines of high order data bus. Bi-directional transfer of data between MPU and module is done through these lines. Also DB_7 can be used as a busy flag. These lines are used as data in 4 bit operation.
DB0 ~ DB3	4	Input/Output	MPU	4 lines of low order data bus. Bi-directional transfer of data between MPU and module is done through these lines. In 4 bit operation, these are not used and should be grounded.
E	1	Input	MPU	Enable - Operation start signal for data read/write.
R/W	1	Input	MPU	Signal to select Read or Write "0": Write "1": Read
RS	1	Input	MPU	Register Select "0": Instruction register (Write) : Busy flag; Address counter (Read) "1": Data register (Write, Read)
Vee	1		Power Supply	Terminal for LCD drive power source.
Vcc	1		Power Supply	+5V
Vss	1		Power Supply	0V (GND)
E1	1	Input	MPU	Enable 1 - Operation start signal for data Read/Write of upper 2

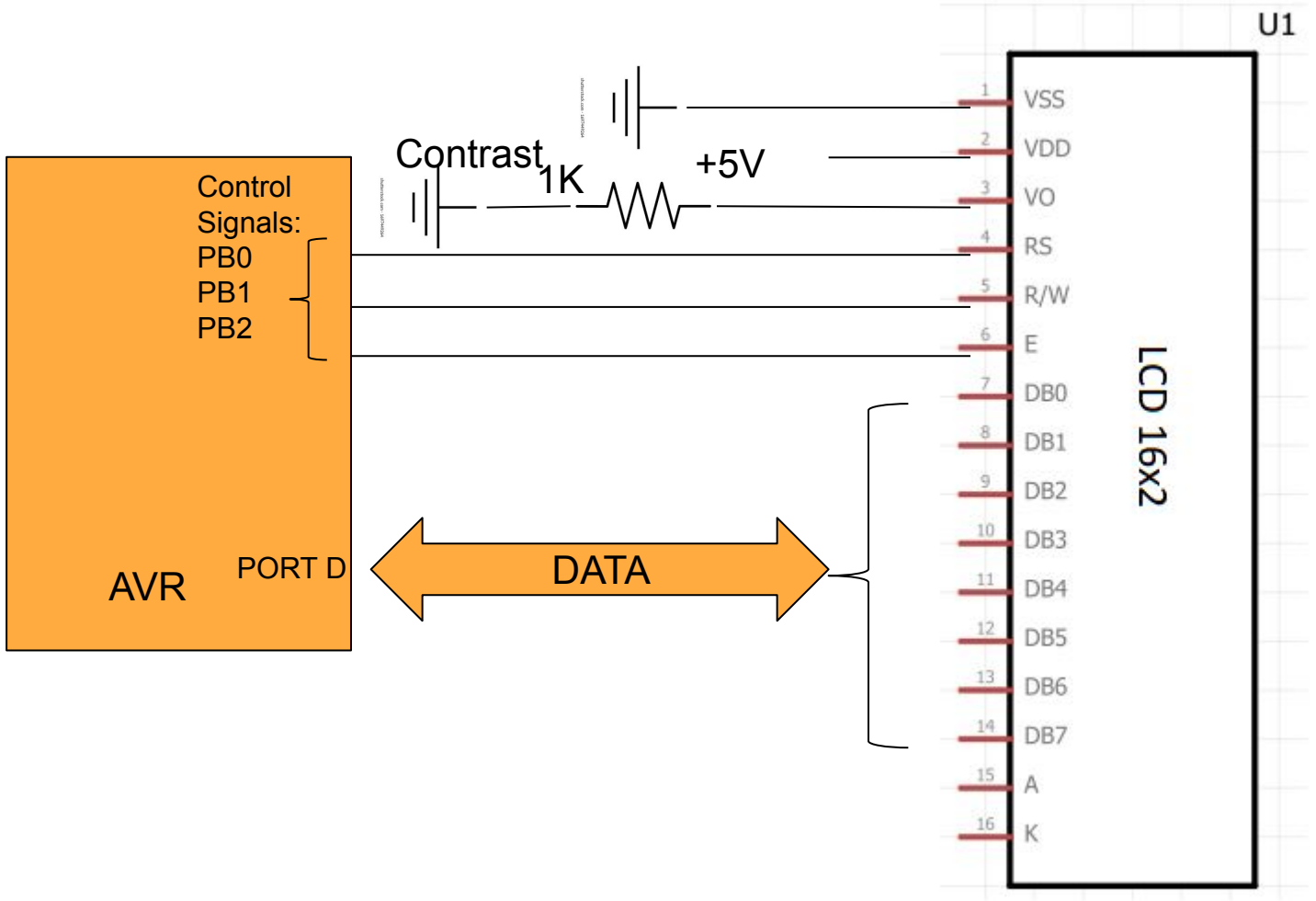
Pin Description

Connections with PORTB

```
...
#define DDR     DDRB
#define PORT    PORTB
#define RS_PIN 0
#define RW_PIN 1
#define EN_PIN 2
...
```

lcd.c

Final Layout



LCD Header Working

```
/**
 * lcd.h
 * Copyright (C) 2001-2020, Tony Givargis
 */

#ifndef _LCD_H_
#define _LCD_H_

void lcd_init(void);

void lcd_clr(void);

void lcd_pos(unsigned char r, unsigned char c);

void lcd_put(char c);

void lcd_puts(const char *s);

#endif /* _LCD_H_ */
```

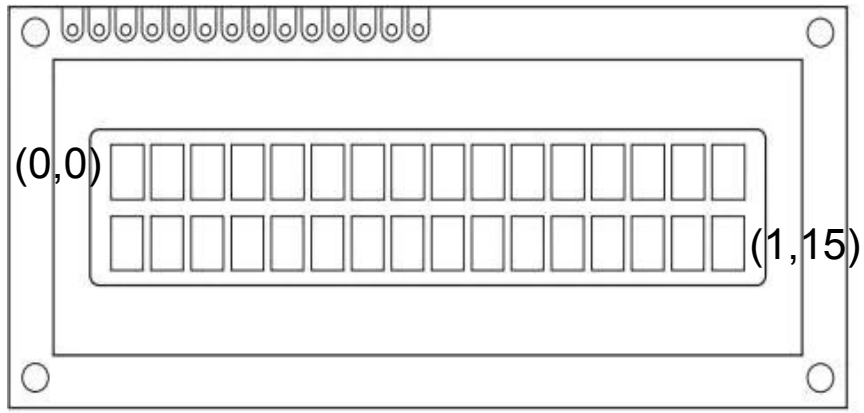
It does everything needed to initialize the display and get it ready to work

Clears All

Cursor to desired Location
1st index for both rows and columns is (0,0)

Print a character and move cursor right by one location

Print bunch of characters



Write Operation Timing Diagram (Refer to page 47 on LCD manual)

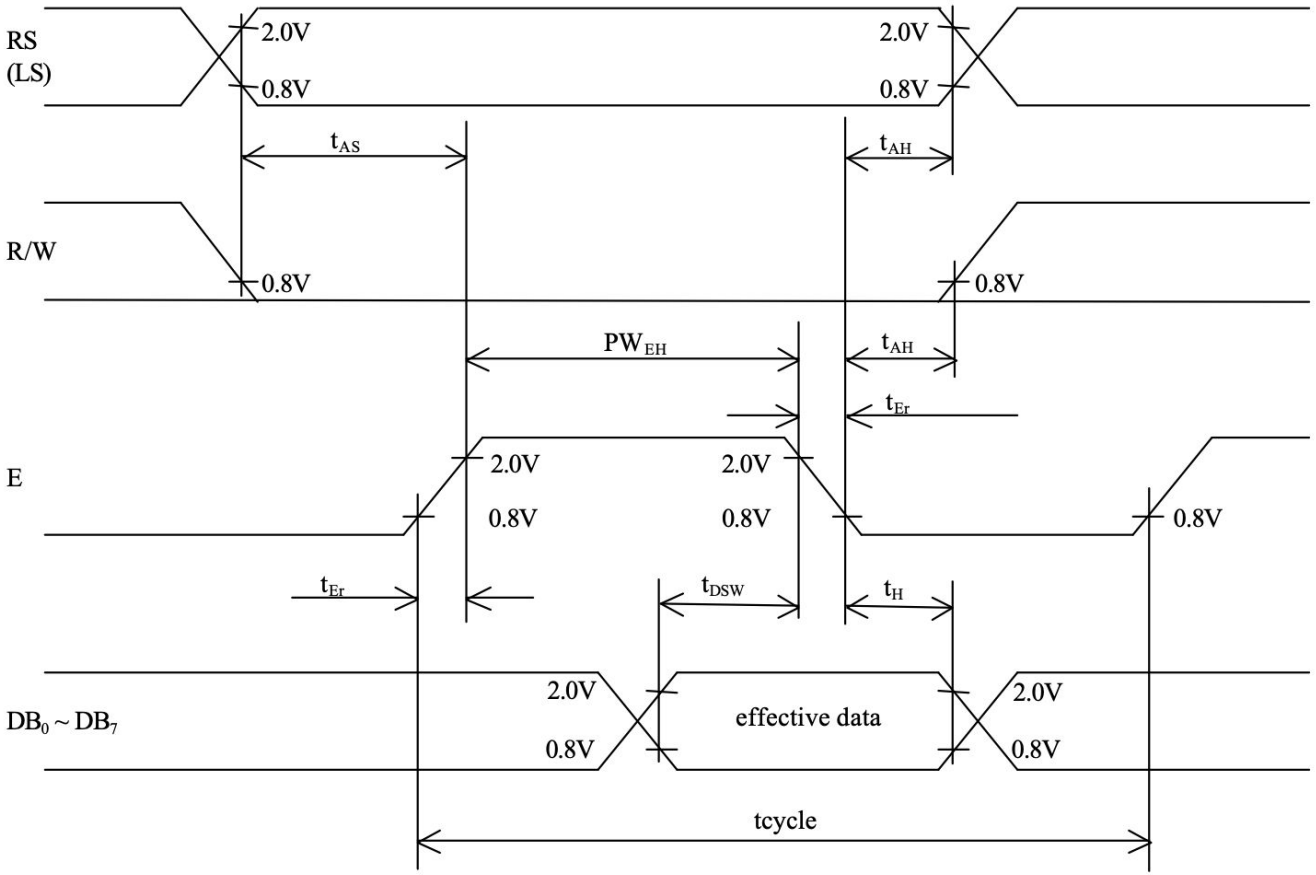


Fig. 4.1
Bus Write Operation Sequence.
(Writing data from MPU to Module)

Write Operation Timing Diagram (Refer to page 49 on LCD manual)

Write timing characteristics (Fig. 4.1)

Item		Symbol	Specs. Value		Unit
			Min.	Max.	
Enable cycle time		tcycle	1000	-	ns
Enable pulse width	"High" level	PW_{EH}	450	-	ns
Enable rising, falling time		t_{Er}, t_{Ef}	-	25	ns
Set up time	RS, R/W-E	t_{AS}	140	-	ns
Address hold time		t_{AH}	10	-	ns
Data set up time		t_{DSW}	195	-	ns
Data hold time		t_H	10	-	ns

Read Operation Timing Diagram (Refer to page 48 on LCD manual)

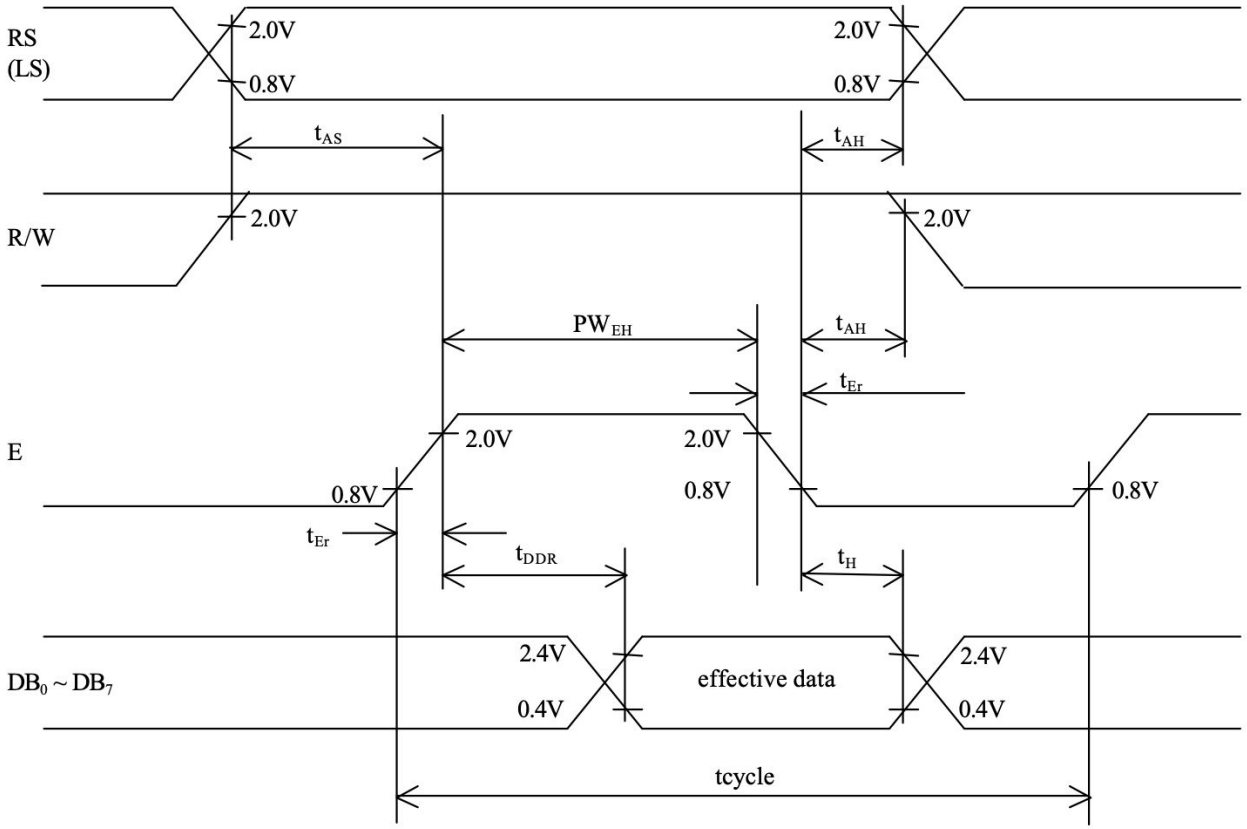


Fig. 4.2
Bus Read Operation Sequence
(Reading data from Module to MPU)

Read Operation Timing Diagram (Refer to page 49 on LCD manual)

Read timing characteristics (Fig. 4.2)

Item		Symbol	Specs. Value		Unit
			Min.	Max.	
Enable cycle time		tcycle	1000	-	ns
Enable pulse width	"High" level	PW_{EH}	450	-	ns
Enable rise, fall time		t_{Er}, t_{Ef}	-	25	ns
Set up time	RS, R/W-E	t_{AS}	140	-	ns
Data delay time		t_{DDR}	-	320	ns
Data hold time		t_H	20	-	ns

Delay / Wait Loop

```
static void
sleep_700ns(void)
{
   NOP();
   NOP();
   NOP();
}
```

```
...

#define SET_BIT(p,i) ((p) |=  (1 << (i)))
#define CLR_BIT(p,i) ((p) &= ~(1 << (i)))
#define GET_BIT(p,i) ((p) &   (1 << (i)))

#define NOP() asm volatile("nop"::)

...
```

avr.h

Code to Write Data

```
static void
output(unsigned char d, unsigned char rs)
{
    if (rs) {
        SET_BIT(PORT, RS_PIN);
    }
    else {
        CLR_BIT(PORT, RS_PIN);
    }
    CLR_BIT(PORT, RW_PIN);
    set_data(d);
    SET_BIT(PORT, EN_PIN);
    sleep_700ns();
    CLR_BIT(PORT, EN_PIN);
}
```

```
static void
set_data(unsigned char x)
{
    PORTD = x;
    DDRD = 0xff;
}
```

Code to Read Data

```
static unsigned char
get_data(void)
{
    DDRD = 0x00;
    return PIND;
}
```

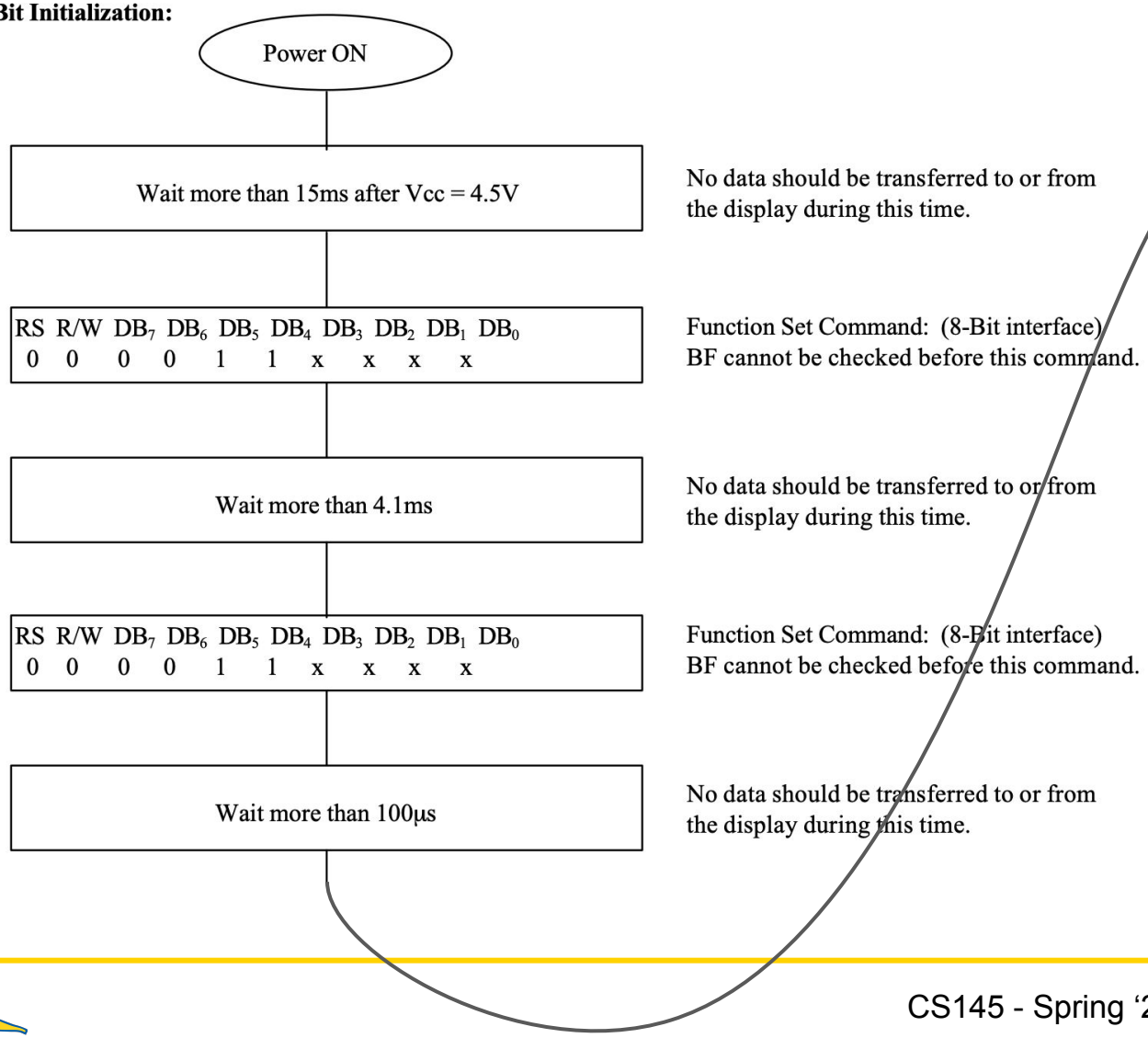
```
static unsigned char
input(unsigned char rs)
{
    unsigned char d;

    if (rs) {
        SET_BIT(PORT, RS_PIN);
    }
    else {
        CLR_BIT(PORT, RS_PIN);
    }
    SET_BIT(PORT, RW_PIN);
    get_data();
    SET_BIT(PORT, EN_PIN);
    sleep_700ns();
    d = get_data();
    CLR_BIT(PORT, EN_PIN);
    return d;
}
```

Check if LCD Module is Busy

```
static void
write(unsigned char c, unsigned char rs)
{
   while (input(0) & 0x80);
   output(c, rs);
}
```

Initialization Procedure (Refer to page 32 on LCD manual)

8 - Bit Initialization:



No data should be transferred to or from the display during this time.

Function Set Command: (8-Bit interface)
BF cannot be checked before this command.

No data should be transferred to or from the display during this time.

Function Set Command: (8-Bit interface)
BF cannot be checked before this command.

No data should be transferred to or from the display during this time.

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	x	x	x	x	Function Set Command: (8-Bit interface) After this command is written, BF can be checked.
0	0	0	0	1	1	N	F	x	x	Function Set (Interface = 8 bits, Set No. of lines and display font)
0	0	0	0	0	0	1	0	0	0	Display OFF
0	0	0	0	0	0	0	0	0	1	Clear Display
0	0	0	0	0	0	0	1	I/D	S	Entry Mode Set:
0	0	0	0	0	0	1	1	C	B	Display ON (Set C and B for cursor/Blink options.)

Initialization Complete, Display Ready.

Note: BF should be checked before each of the instructions starting with Display OFF.

LCD Initialization

```
void
lcd_init(void)
{
    SET_BIT(DDR, RS_PIN);
    SET_BIT(DDR, RW_PIN);
    SET_BIT(DDR, EN_PIN);
    avr_wait(16);
    output(0x30, 0);
    avr_wait(5);
    output(0x30, 0);
    avr_wait(1);
    write(0x3c, 0);
    write(0x0c, 0);
    write(0x06, 0);
    write(0x01, 0);
}
```

LCD Clear

```
void
lcd_clr(void)
{
   write(0x01, 0);
}
```

3.1.1 Clear Display

	RS	R/W	DB_7	DB_6	. . .	DB_1	DB_0
Code	0	0	0	0	0	0	1

Writes the space code “20” (hexadecimal) into all addresses of DD RAM. Returns display to its original position if it was shifted. In other words the display clears and the cursor or blink moves to the upper left edge of the display. The execution of clear display instruction sets entry mode to increment mode.

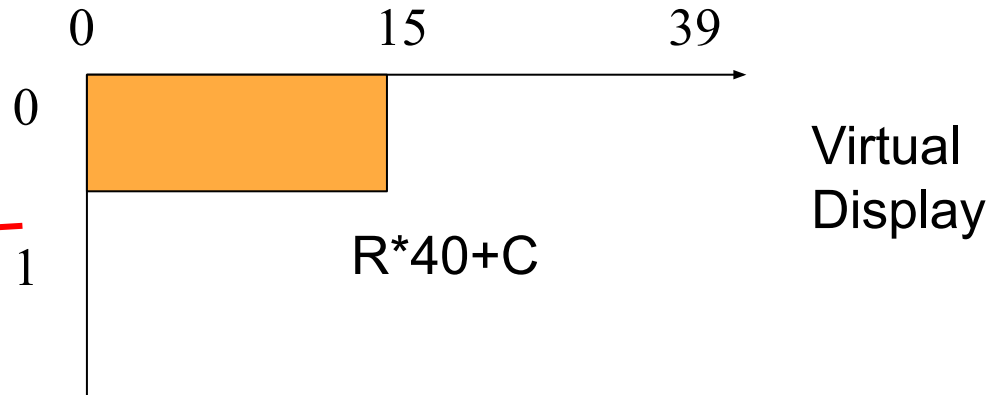
Write Text to LCD

```
void
lcd_put(char c)
{
   write(c, 1);
}
```

```
void
lcd_puts(const char *s)
{
   char c;
   while ((c = *(s++)) != 0) {
      write(c, 1);
   }
}
```

LCD Position Function

```
void
lcd_pos(unsigned char r, unsigned char c)
{
    unsigned char n = r * 40 + c;

    write(0x02, 0);
    while (n--) {
        write(0x14, 0);
    }
}
```



3.1.2 Return Home

	RS	R/W	DB_7	DB_6	. . .	DB_1	DB_0
Code	0	0	0	0	0	1	x

Note: x = Don't Care

Shifts to the right by one

Sets the DD RAM address "0" in address counter. Return display to its original position if it was shifted. DD RAM contents do not change.

The cursor or the blink moves to the upper left edge of the display. Text on the display remains unchanged.

Cursor Shift

0x14 == 20 == 0b00010100

3.1.5 Cursor or Display Shift

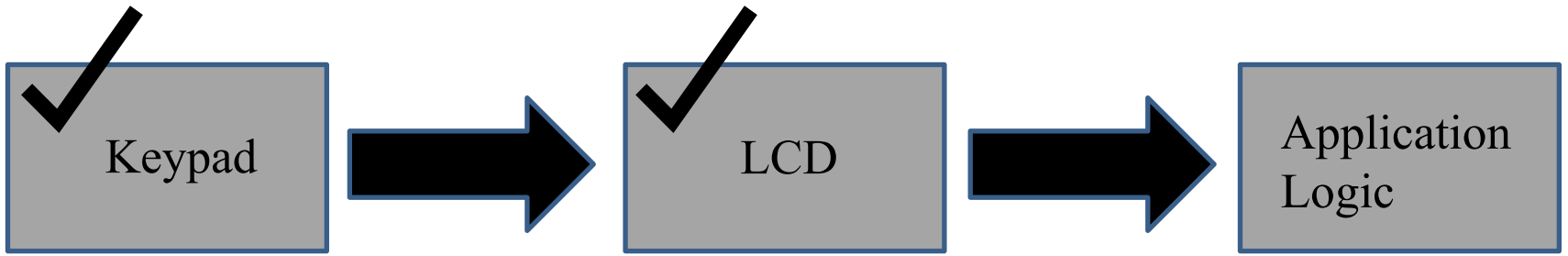
	RS	R/W	DB_7	DB_6	. . .			DB_1	DB_0
Code	0	0	0	0	0	1	S/C	R/L	x	x

Note: x = Don't Care

Shifts the cursor position or display to the right or left without writing or reading display data. This function is used to correct or search for the display. In a 2-line display the cursor moves to the 2nd line when it passes the 40th digit of the 1st line. Notice that the 1st and 2nd line displays will shift at the same time.

When the displayed data is shifted repeatedly each line only moves horizontally but the 2nd line display does not shift into the 1st line position.

Project 2 Roadmap



Date and Time Data Structure

```
typedef struct {
  int year;
  unsigned char month;
  unsigned char day;
  unsigned char hour;
  unsigned char minute;
  unsigned char second;
} DateTime;
```

int uses 2 bytes
in the ATmega32
-32,768 ~ 32,767

```
void init_dt(DateTime *dt) {
  dt->year = 2022;
  ...
  dt->second = 0;
}
```

```
void advance_dt(DateTime *dt) {
  ++dt->second;
  if (dt->second == 60) {
    // Increase minute and check.
    // Repeat process as needed.
  }
}
```

```
void print_dt(const DateTime *dt) {
  char buf[17];
  // Print date on top row.
  lcd_pos(0, 0);
  sprintf(buf, "%04d-%02d-%02d",
                          dt->year,
                          dt->month,
                          dt->day)
  lcd_puts(buf);
  // Do similar thing to print time on bottom row.
}
```

Main Function for Digital Clock

```
int main() {
  DateTime dt;
  lcd_init();
  init_dt(&dt);
  while (1) {
    avr_wait(1000);
    advance_dt(&dt);
    print_dt(&dt);
  }
  return 0;
}
```

Check keypad press before function calls.

See you next time :)

Q & A