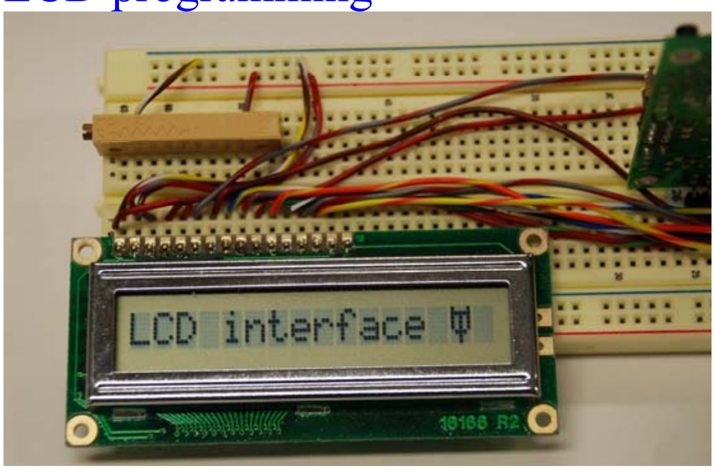
Recap

I/O synchronization
LCD interface
Implementing local variables with a stack frame
Parameter passing

Overview

LCD programming
Fixed-point numbers

LCD programming



Instruction	RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	Code									
Clear display	0	0	0	0	0	0	0	0	0	1
Return home	0	0	0	0	0	0	0	0	1	—
Entry mode set	0	0	0	0	0	0	0	1	I/D	S
Display on/off control	0	0	0	0	0	0	1	D	C	B
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—
Function set	0	0	0	0	1	DL	N	F	—	—
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD

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
F = 1: 5 × 10 dots, F = 0: 5 × 8 dots
BF = 1: Internally operating
BF = 0: Instructions acceptable

- 2-line display (N = 1) (Figure 4)
  — Case 1: When the number of display characters is less than 40 × 2 lines, the two lines are displayed from the head. Note that the first line end address and the second line start address are not consecutive. For example, when just the HD44780 is used, 8 characters × 2 lines are displayed. See Figure 5.

  When display shift operation is performed, the DDRAM address shifts. See Figure 5.

Display position	1	2	3	4	5	...	39	40
DDRAM address (hexadecimal)	00	01	02	03	04	....	26	27
	40	41	42	43	44	....	66	67

Figure 4 2-Line Display

Display position	1	2	3	4	5	6	7	8
DDRAM address	00	01	02	03	04	05	06	07
	40	41	42	43	44	45	46	47
For shift left	01	02	03	04	05	06	07	08
	41	42	43	44	45	46	47	48
For shift right	27	00	01	02	03	04	05	06
	67	40	41	42	43	44	45	46

Figure 5 2-Line by 8-Character Display Example

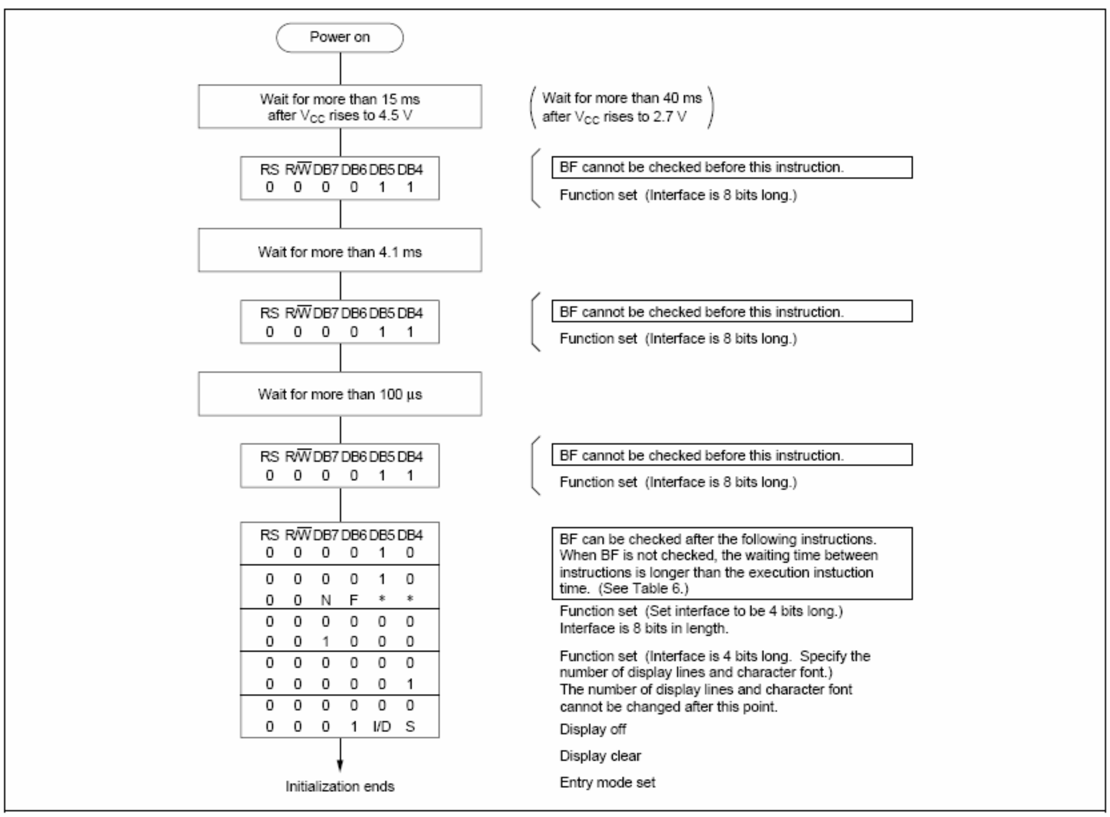


Figure 26 4-Bit Interface

Clear display

```
;   1) outCsr($01)   causes Clear
;   2) blind cycle 1.64ms wait
;   3) outCsr($02)   sends the Cursor to home
;   4) blind cycle 1.64ms wait
```

Move cursor to

```
;   1) outCsr(DDaddr+$80)
```

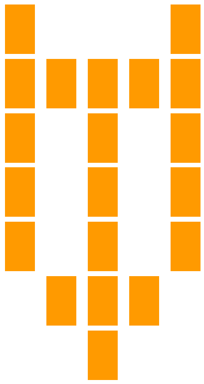
first row (left-most 8) `DDaddr` is 0 to 7

second row (right-most 8) `DDaddr` is $40 to $47

Define up to 8 new fonts

```
// font is the font number 0 to 7
// n0-n7 is the 5 by 8 pixel map
   CGaddr=((font&0x07)<<3)+0x40;
   outCsr(CGaddr); // set CG RAM address
   LCD_OutChar(n0);
   LCD_OutChar(n1);
   LCD_OutChar(n2);
   LCD_OutChar(n3);
   LCD_OutChar(n4);
   LCD_OutChar(n5);
   LCD_OutChar(n6);
   LCD_OutChar(n7);
   outCsr(0x80); // revert back to DD RAM
```

Example



```
//              n0 = $00
//              n1 = $11
//              n2 = $1F
//              n3 = $15
//              n4 = $15
//              n5 = $15
//              n6 = $0E
//              n7 = $04
// set font=5 to this graphic
// CGaddr=((5&0x07)<<3)+0x40 = $68
   outCsr(0x68); // set CG RAM address
   LCD_OutChar(0x00);
   LCD_OutChar(0x11);
   LCD_OutChar(0x1F);
   LCD_OutChar(0x15);
   LCD_OutChar(0x15);
   LCD_OutChar(0x15);
   LCD_OutChar(0x0E);
   LCD_OutChar(0x04);
   outCsr(0x80); // revert back to DD RAM

// To output this graphic
   LCD_OutChar(5);
```

The floating-point format

Bit 31	sign, **s**=0 for positive, **s**=1 for negative
Bits 30:23	8-bit biased binary exponent $0 \le \mathbf{e} \le 255$
Bits 22:0	24-bit mantissa, **m**
	expressed as a binary fraction
	a binary 1 as the most significant bit is implied

$$\mathbf{m} = 1.\mathbf{m}_1\mathbf{m}_2\mathbf{m}_3\ldots\mathbf{m}_{23}$$

s	e_7 e_0	m_1 m_{23}

$$\mathbf{f} = (-1)^s \bullet 2^{e-127} \bullet \mathbf{m}$$

Fixed point numbers

Why? (wish to represent non-integer values)

Next lab measures distance from 0 to 3 cm

E.g., 1.234 cm

When? (range is known, range is small)

Range is 0 to 3cm

Resolution is 0.003 cm

How? (value = Integer*Δ)

16-bit unsigned integer

$\Delta = 10^{-3}$ decimal fixed-point

Range becomes 0.000 to 65.535

The design of LCD_OutFix

What does it mean?

Fixed-point number (integer part) to ASCII

$\Delta = 10^{-3}$ decimal fixed-point

Work through LCD_OutFix with Input = 1234

```
LCD_OutFix
  0) save any registers that will be destroyed by pushing on the stack
  1) allocate local variables letter and num on the stack
  2) initialize num to input parameter, which is the integer part
  3) if number is less or equal to 9999, go the step 6
  4) output the string "*.*** " calling LCD_OutString
  5) go to step 19
  6) perform the division num/1000, putting the quotient in letter,
     and the remainder in num
  7) convert the ones digit to ASCII, letter = letter+$30
  8) output letter to the LCD by calling LCD_OutChar
  9) output '.' to the LCD by calling LCD_OutChar
```

```
  10)perform the division num/100, putting the quotient in letter,
      and the remainder in num
  11)convert the tenths digit to ASCII, letter = letter+$30
  12)output letter to the LCD by calling LCD_OutChar
  13)perform the division num/10, putting the quotient in letter,
      and the remainder in num
  14)convert the hundredths digit to ASCII, letter = letter+$30
  15)output letter to the LCD by calling LCD_OutChar
  16)convert the thousandths digit to ASCII, letter = num +$30
  17)output letter to the LCD by calling LCD_OutChar
  18)output ' ' to the LCD by calling LCD_OutChar
  19)deallocate variables
  20)restore the registers by pulling off the stack
```

How do we test LCD_OutFix?

Stabilize inputs with a test set of inputs

User types in input, your program displays results

Recursion, a function that calls itself, see book Program 3.12

```
// Variable format 1 to 5 digits with no space before or after
// uses recursion to convert a decimal number to an ASCII string
void OutUDec(unsigned short n){
  if(n >= 10){
    OutUDec(n/10);
    n=n%10;
  }
  OutChar(n+'0'); /* n is between 0 and 9 */
}
```

Program 3.12. Print unsigned 16-bit decimal to an output device.

Put variables on the stack

Save and restore variables as needed

End condition

Simple answer,

Return value

Body

Call itself, calculate result

Return value

Fact(1) = 1 end condition

Fact(n) = n*Fact(n-1) if n>1 recursion

```
; recursive implementation
; Input: RegD is n
```

```
; Output: RegD is Fact(n)
n     set  2
Fact  pshd
      pshy
      cpd  #1    ; end condition
      bls  done
      subd #1    ; n-1
      bsr  Fact  ; RegD=Fact(n-1)
      ldy  n,s   ; RegY=n
      emul       ; RegD=n*Fact(n-1)
done  puly
      leas 2,s
      rts
```

Draw a stack frame

The bottom line

Stack is useful for local variables, parameter passing
Draw stack picture by hand executing assembly code
LCD is an output device for embedded systems
- **Text, graphics, touch pads, color**
- **On board controller handles bit-level functions**

Fixed-point allows non-integers without FP hardware