ENCS 411 x86 Computer System Lab

MDA-8086 LCD Applications

Birzeit University

Information Technology Faculty

Computer Systems Engineering Department

Abstract

This experiment aims at understanding, configuring and testing the LCD used on the MDA 8086 kit. Experiment includes applications such as displaying the outcome of an Analog to Digital Converter ADC804 as a Volt Meter on the LCD.

PART I Theoretical and Technical Introduction

1.1 The LCD Display

1.1.1 Introduction

The LCD display is Liquid Crystal Display used to provide readable information for the user on the kit. The LCD that we will be using in this experiment is 16 character x 2 lines. This means 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

The LCD has three control lines are referred to as **EN**, **RS**, and **RW**.

The **EN** line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring **EN** high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

The **RS** line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which should be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

The **RW** line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands--so RW will almost always be low.

1.1.2 PIN Connection

The LCD has the following PIN Connection. Usually the LCDs has the same pins structure.

Pin NO.	Symbol	Level	Function	
1	Vss	-	0V	Power supply
2	Vdd	-	5V	
3	VL	-	-	
4	RS	H/L	H : Data input L : Instruction input	
5	R/W	H/L	H : Data read L : Data write	
6	E	H. H→L	Enable signal	
7	D0	H/L	Data bus line	
8	D1	H/L		
9	D2	H/L		
10	D3	H/L		
11	D4	H/L		
12	D5	H/L		
13	D6	H/L		
14	D7	H/L		

1.1.3 INSTRUCTIONS

Instruction	CODE										Description	Execution time(max) fosc is 250 KHz
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display	1.64 ㎳
Return Home	0	0	0	0	0	0	0	0	1	*	Returns display being shifted to original position	1.64 ㎳
Entry Mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display	40 ㎲
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	D : Display ON/OFF C : Cursor ON/OFF B : Cursor Blink/Not	40 ㎲
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor and Shifts display	40 ㎲
Function Set	0	0	0	0	1	DL	N	F	*	*	Refer to Remark	40 ㎲
Set CGRAM	0	0	0	1	ACG						Sets CG RAM Addr.	40 ㎲
Set DD RAM Addr.	0	0	1	ADD							Sets DD RAM Address	40 ㎲
Read Busy Flag & Addr	0	1	BF	AC							BF : Busy flag Reads AC contents.	40 ㎲
Write Data CG or DD	1	0	Write data								Writes data into DD RAM or CG RAM	40 ㎲
Read Data from CG or DD RAM	1	1	Read data								Reads data from DD RAM or CG RAM	40 ㎲
Remark	I/D= 1: Increment 0: Decrement S= 1: Accompanies display shift S/C=1:Display shift. 0:cursor move R/L=1:Shift right. 0: Shift left. DL= 1 : 8bits 0 : 4 bits N = 1 : 2 lines 0 : 1 lines F = 1 : 5×10dots 0 : 5×7dots BF = 1: Internally operating 0: Can accept instruction * NO EFFECT										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 DD and CG RAM address	

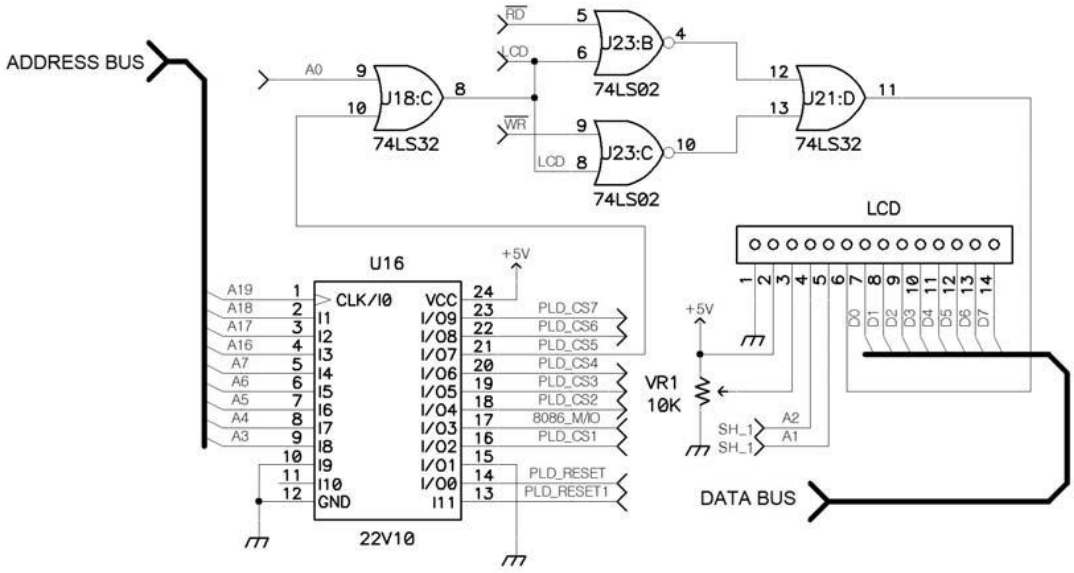
1.1.4 INITIALIZING SEQUENCE

SEQUENCE
POWER ON
⇓
Wait till VCC is 4.5V min
⇓
RS = 0, WRITE 38H * (Execution time : 40 μs)
⇓
RS = 0, WRITE 0EH (Execution time : 40 μs)
⇓
RS = 0, WRITE 08H (Execution time : 40 μs)
⇓
RS = 0, WRITE 02H (Execution time : 1.64 ms)
⇓
RS = 0, WRITE 01H (Execution time : 1.64 ms)
⇓
RS = 0, WRITE ADDR. ** (Execution time : 40 μs)
⇓
RS = 1, WRITE DATA ** (Execution time : 40 μs)

* . Should use this instruction only once in operation.
* *. ADDR is the setting data cursor position to debug. In data, MSB(D7) should be "1" and other 7 bits (D0～D6) are cursor position.
*** . DATA mean the ASCII codes.

1.1.5 LCD Interface

Study the schematics shown for the LCD interface on the MDA-8086 kit.



1.1.6 CHARACTER FONT TABLE

상위 4비트 / 하위 4비트	0000	0010	0011	0100	0101	0110	0111	1000	1010	1011	1100	1101	1111
XXXX0000	CG RAM (1)		0	@	P	`	p		ー	夕	ミ	ム	円
XXXX0001	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ム	9
XXXX0010	(3)	"	2	B	R	b	r	「	イ	ツ	メ	Ｐ	8
XXXX0011	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
XXXX0100	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	H	Ω
XXXX0101	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	Ü
XXXX0110	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
XXXX0111	(8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
XXXX1000	(1)	(	8	H	X	h	x	ィ	ク	ネ	リ	J	X
XXXX1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	'	9
XXXX1010	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
XXXX1011	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	×	万
XXXX1100	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
XXXX1101	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
XXXX1110	(7)	.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
XXXX1111	(8)	/	?	O	_	o	←	ッ	ソ	マ	゜	ö	■

NOTE : CGRAM is a CHARACTER GENERATOR RAM having a storage function of character pattern which enable to change freely by users program

PART III Practices

3.1 PRACTICE I: SCROLLING A MESSAGE ON LCD in ASM

WARNING: Don't touch any exposed wiring or the pins of any of the ICs.

Step1: Write the following code and save it.

```
CODE SEGMENT
ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE
;
; STACK EQU 0540H
;
LCDC            EQU     00H
LCDC_S          EQU     02H
LCDD            EQU     04H
;
1000 ORG 1000H
;
        XOR AX,AX
        MOV SS,AX
        MOV SP,STACK
;
        CALL ALLCLR
;
        CALL ENTMODE
 L1:
        CALL CUSOR1
        MOV SI,OFFSET DATA
         CALL STRING
        JMP L1
;
DATA DB '8086 Training Kit Good !',00H
;
; LCD instruction
ALLCLR:
        MOV AH,01H
        JMP LNXX
;
ENTMODE:
        MOV AH,00000111B
        JMP LNXX
;
CUSOR1:
```

```
        MOV AH,90H
;
LNXX:
        CALL BUSY
        MOV AL,AH
        OUT LCDC,AL
        RET

; busy flag check
BUSY:
        IN AL,LCDC_S
        AND AL,10000000B
        JNZ BUSY
        RET
;
; 1 char. LCD OUT
; AH = out data
CHAROUT:
        CALL BUSY
;
        MOV AL,AH
        OUT LCDD,AL
        RET
;
STRING:
        MOV AH,BYTE PTR CS:[SI]
        CMP AH,00H
        JE STRING1

        ; out
        CALL BUSY
        CALL CHAROUT
        INC SI
        CALL TIMER
        JMP STRING

STRING1:
        RET
;

TIMER:
        MOV CX,1
        TIMER2: PUSH CX
        MOV CX,0
TIMER1:
        NOP
        NOP
```

```
        NOP
        NOP
        LOOP TIMER1
        POP CX
        LOOP TIMER2
        RET
;
CODE ENDS
END
```

TASKS:

1. Explain how the code works?
2. What is the purpose of code under the LNXX label?
3. Modify the code above to display your name.

3.2 PRACTICE II: DISPLAY ON LCD in C

Step1: Write the following code and save it, and upload it to the Kit.

```
#define _LCD                /* You must define it, because using LCD Function in the HeaderFile. */
#include "mde8086.h"
                            /* To the LCD Output String( Delay One character) */
void string( char *str )
{
        while( *str ) {
                            LCD_putch( *str );
                            str ++;
                            wait( 10000 );
                        }
}

void main( void )           /*my not need this line if compiled in Linux*/
{
        LCD_init;                           /* LCD Initial */
        string( "Serial monitor !" );
        LCD_LN21;
        string( "Midas 335-0964/5" );
        do {
                LCD_DISPOFF;
                wait( 20000 );
                LCD_DISPON;
                wait( 20000 );
            } while(1);
}
```

TASKS:

1. Explain how the code works?
2. What is the purpose of following instructions:
    a. LCD_init
    b. LCD_DISPOFF
    c. LCD_DISPON
3. Modify the code above to display your name.
4. Write a code that emulates the result of the assembly code in Secion 3.1 . Use the help of the following fuctions.
    a. LCD_init;
    b. LCD_ALLCLR;
    c. LCD_puts("TEXT");
    d. LCD_RShift;
    e. wait(#);

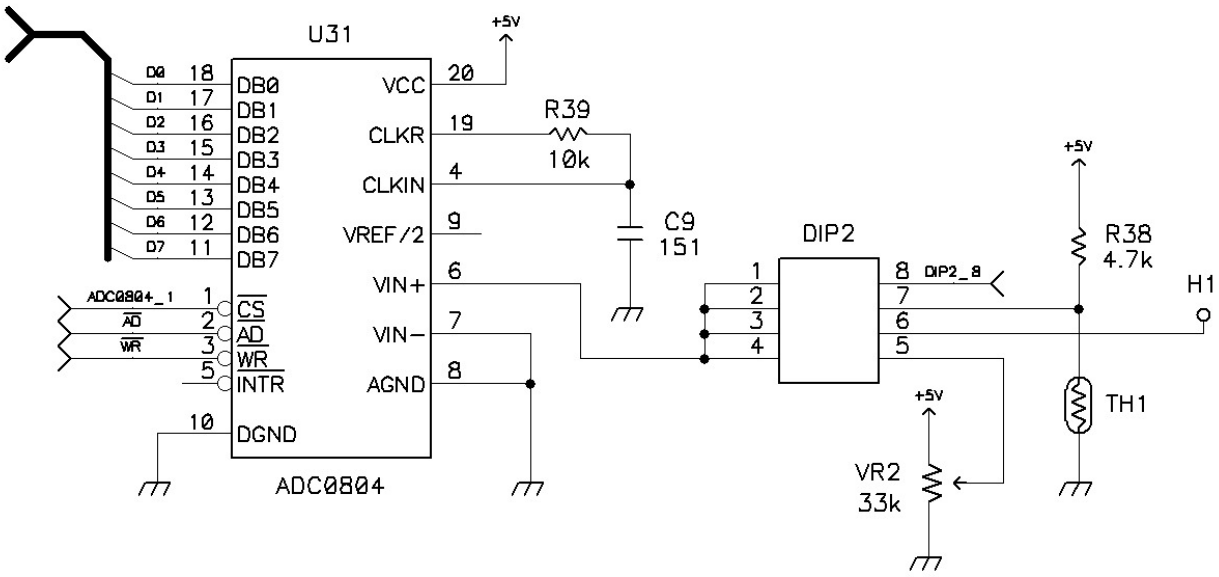
3.3PRACTICE III: DISPLAY OUTPUT of ADC804 on LCD

The ADC804 is an 8-bit analog to digital converter with a resolution (step voltage) of

-Vref / 255

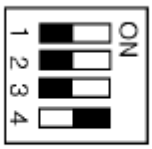
NOTE: Students are expected to review their course material on ADC804 as well as DAC830 converters. This part only deals with the ADC804 as it uses the LCD. Lab instructor may demo the usage of the DAC830 as well if time permits.

The ADC schematics portion is shown below;



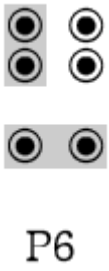
Execute the following steps:

1. Set the DIP2 switch near ADC0804 as follow (i.e. switch 4 is ON)



DIP2

2. Move the jumpers on P6 as follow;



3. Execute the C code below, then

4. Turn volume resister (VR2), potentiometer, around and observe the changes on the LCD

CODE TO EXECUTE:

```
#define                    _LCD
#include    "mde8086.h"
/* Output Fixed Point
          v     : Output Data
          max   : Output Location( Integer Inclusion )
          point : Point Location
*/
void     LCD_putf( long v, int max, int point )
{
   char  temp[20];
   char  temp1[20];
   int      len, i;

   ltoa( v, temp, 10 );
   memset( temp1, '0', max );
   temp1[max] = 0;

   len = strlen( temp );
   memmove( temp1+(max-len), temp, len );
```

```
   /* Output Integer */
   for( i = 0;  i < max-point; i ++ )  LCD_putch( temp1[i] );

   LCD_putch( '.' );
   LCD_puts( temp1+i );
}
void    main( void )
{
   long v;
   char buf[20], temp[10];
         int                 i;

   LCD_init;
   LCD_puts( "   Volt Meter" );

   do  {
     outportb( ADC, 0xff);
     wait( 20000 );

     v = inportb(ADC)*(500000I/256);

     v /= 100;
     LCD_Iout( 0xc5 );

     LCD_putf( v, 4, 3 );
     LCD_puts( " V" );

   } while( 1);
}
```

TASKS:

1. Explain the following lines of the **do** loop:

```
v = inportb(ADC)*(500000I/256);

v /= 100;
LCD_Iout( 0xc5 );
```

```
LCD_putf( v, 4, 3 );
LCD_puts( " V" );
```

2. Make changes to display the outcome in milli-volt, i.e. mV.