The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN204779

F²MC-16FX Family, LC-Display

This application note gives a rough overview about how this kind of display works and illuminates some different methods to control a Liquid Crystal Display (LCD) by a microcontroller.

Contents

1 Introduction....1
2 Introduction in Liquid Crystals....1
2.1 The Physics....1
2.2 Addressing....3
3 Controlling of Liquid Crystals....6
3.1 MB963xx Microcontrollers with LCD Controller/Driver....6
3.2 External LCD Controller/Driver....10
3.3 Alphanumeric LCD Module....11
4 LCD Examples....14
4.1 Interfacing MB9638x to LCD Glass DE133....14
4.2 Interfacing MB963xx to LCD Module HD44780....15
5 Conclusion....21
6 Additional Information....21
7 Document History....22

1 Introduction

Segment displays based on liquid crystal material are very common if only a small number of characters or symbols should be displayed.

This application note gives a rough overview about how this kind of display works and illuminates some different methods to control a Liquid Crystal Display (LCD) by a microcontroller.

Further, the connection to LCD glass and also to alphanumeric LCD modules (with integrated segment and LCD controller) will be discussed.

2 Introduction in Liquid Crystals

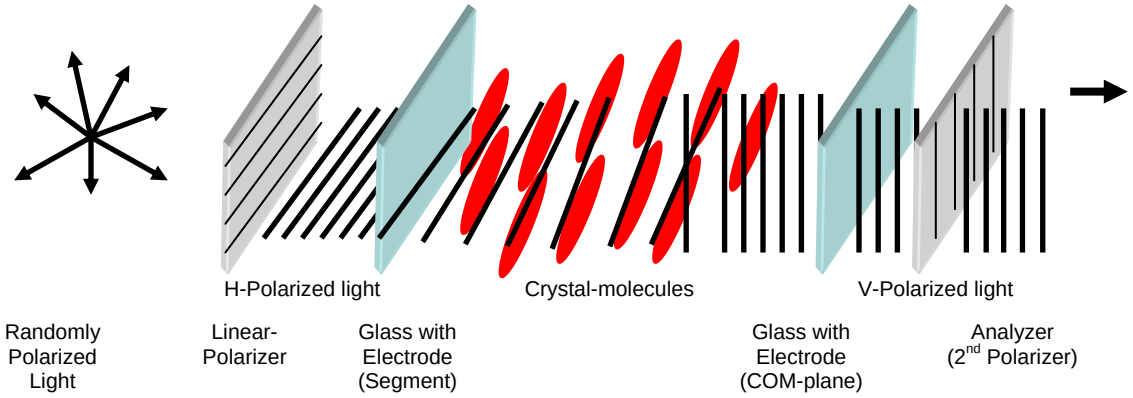
This chapter reflects the technical background of passive LC-segments

2.1 The Physics

The main characteristic of Liquid Crystal is based on anisotropic material. Most common are Nematic LC molecules all tend to align in the same direction to each other.

In the following, the basic construction and basic function of a standard display will be explained. A first special glass will polarize random ambient light. When entering the next layer the twisted structure of the crystal-molecules rotates the polarization of linearly polarized incident light by 90 degrees, so that it passes through a second polarizer, and the device appears white.

Figure 1.LC Display Polarized with Ambient Light



If an electric field is applied to the cell, the molecular axes align with the field and the structure no longer twists the polarization of the incident light, so the emerging light is absorbed by the second polarizer and the device appears black.

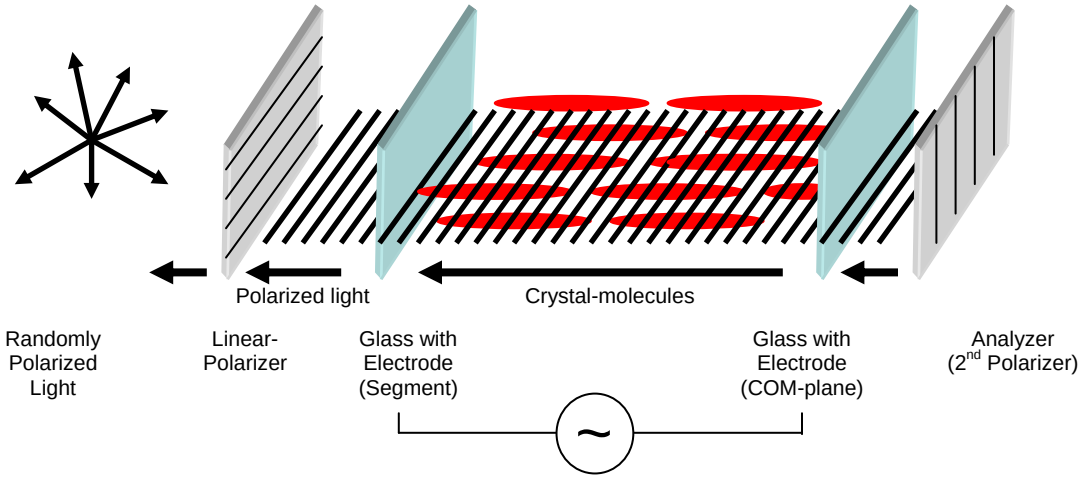
If the field is removed the Nematic relaxes to the twisted structure.

The mobility of the crystal depends very much on the temperature; in cold environment the movement is very slow.

Seen from the electrical side one element can be compared with a capacitor, together with the signal-lines it can be handled as a RC-element.

Direct current will attack Liquid crystals and last destroy it. So any DC-offset should be prevented from the molecules. In general, datasheets of LC-displays allow less than 50mV.

Figure 2. LC Display with Electric Field Applied



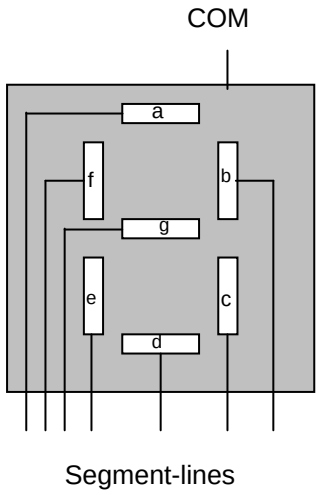
2.2 Addressing

Addressing is the process to turn on and off a pixel in order to create an image. There are two main types of addressing, direct and multiplexing.

2.2.1 Static addressing

Direct addressing is convenient for displays with only a few elements that have to be activated. With direct addressing, each pixel in the display has its own drive circuit. The control-voltage has to be applied to each element.

Figure 3. 7-Segment Display using Static Addressing

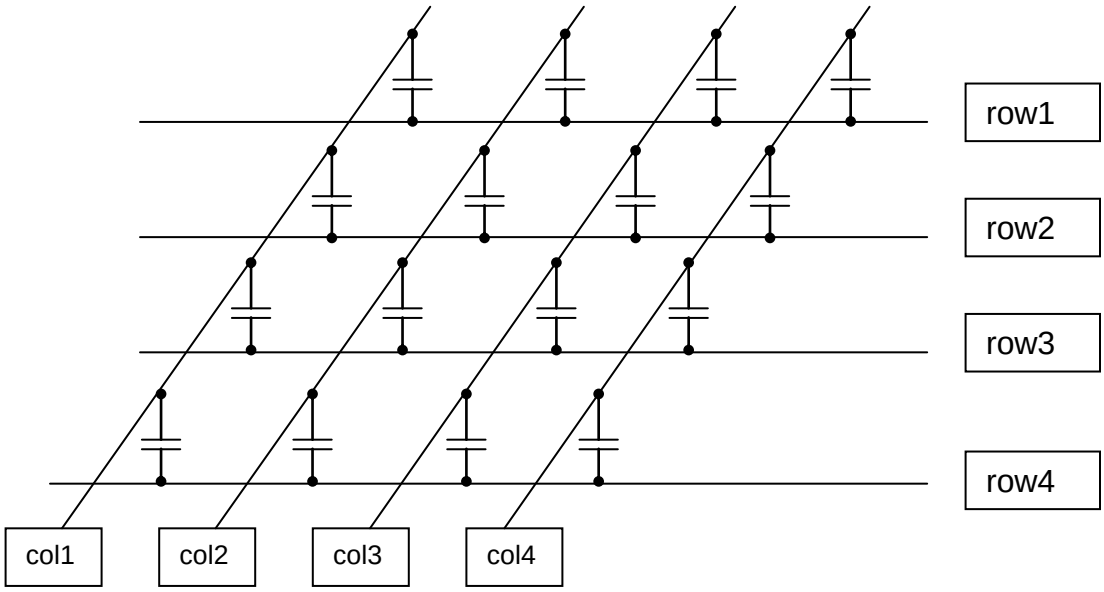


A common application of direct addressing is the traditional seven-segment liquid crystal display, found in wristwatches and similar devices.

2.2.2 Multiplexed addressing

In multiplex addressing, a larger number of pixels are involved. When the elements are in a regular order, they can be addressed by their row and column instead of each element being driven separately.

Figure 4. Multiplexed Addressing



This reduces the complexity of the circuitry because each pixel no longer needs its own driver circuit. If you have a 4x4 matrix of pixels, with direct addressing, you need 16 individual drivers. However, if you use multiplex addressing, you only need eight drivers, one for each row and one for each column.

Figure 5. 7-Segment Display using Multiplexed Addressing

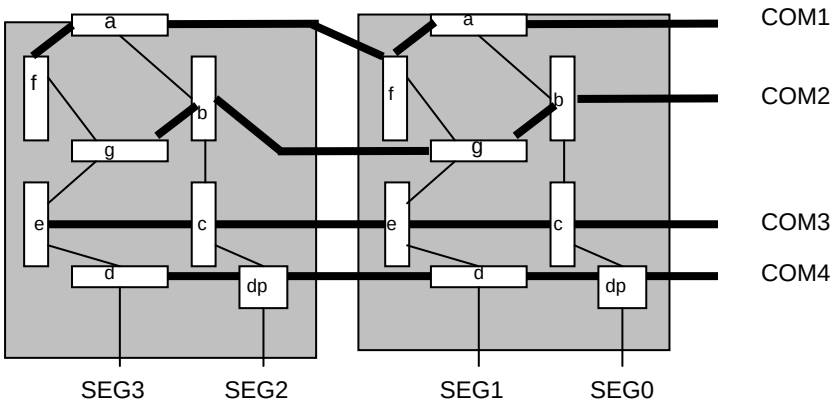
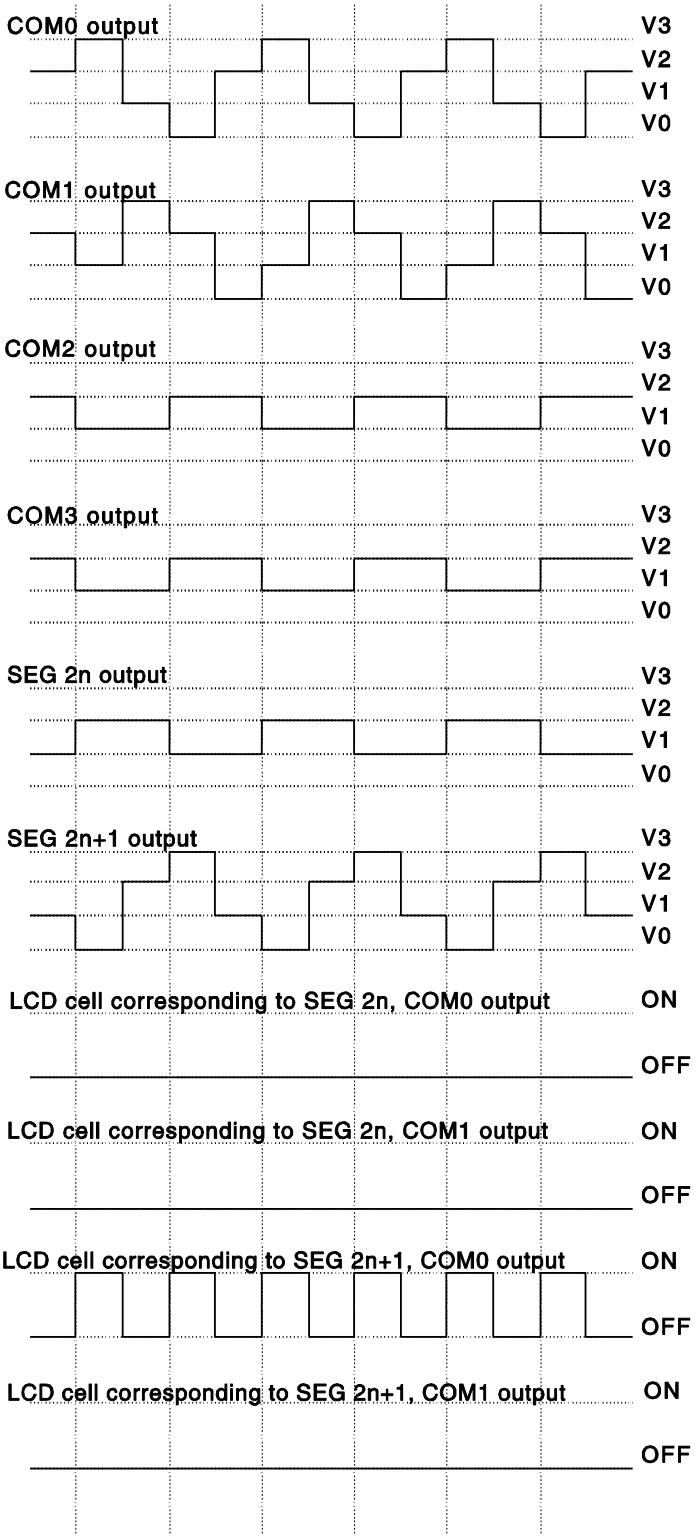


Figure 6. Output Waveform (1/2 duty cycle, 1/3 Bias)



Example:

Figure 6 at the right represents the 3x2 format (bias 1:3, duty 1:2).

Only two COM-lines (duty 1:2) are used and three output-voltages V1, V2 and V3 (bias 1:3) will be output.

The shown waveform demonstrates two segments: SEG_{2n}=OFF and SEG_{2n+1}=ON.

The algebraically summed voltages between the COM- and the segment-lines are depicted in the last 4 waveforms.

In order to prevent deterioration of the crystals by DC power the LCD is driven with a two-frame AC waveform.

Voltage levels:

V0 = 0V , V1= Vcc/3

V2 = 2Vcc/3, V3 = Vcc

(1) $(V_3 – V_1) = - (V_0 – V_1)$

(2) $V_1 = V_3/3$

(3) $V_2 = 2V_3/3$

In addition, the time-interval each signal is applied has to be constant, too.

If the number of common planes (COMx) is increased, the time each pixel is controlled (duty cycle) decrease what result in worse contrast and angle view.

The liability is the increased complexity of drive circuitry.

The method of drive for multiplexed displays is essentially a time division multiplex with the number of time divisions equal to twice the number of common planes used in a given format. As is the case with conventional LCDs, in order to prevent irreversible electrochemical action from destroying the display, the voltage at all segment locations must be caused to reverse polarity periodically so that zero net DC voltage is applied.

This is the reason for the doubling in time divisions: Each common plane must be alternately driven with a voltage pulse of opposite polarity.

3 Controlling of Liquid Crystals

This chapter reflects how to control passive LC-segments

3.1 MB963xx Microcontrollers with LCD Controller/Driver

The following microcontrollers in the 16FX Family have integrated line-drivers:

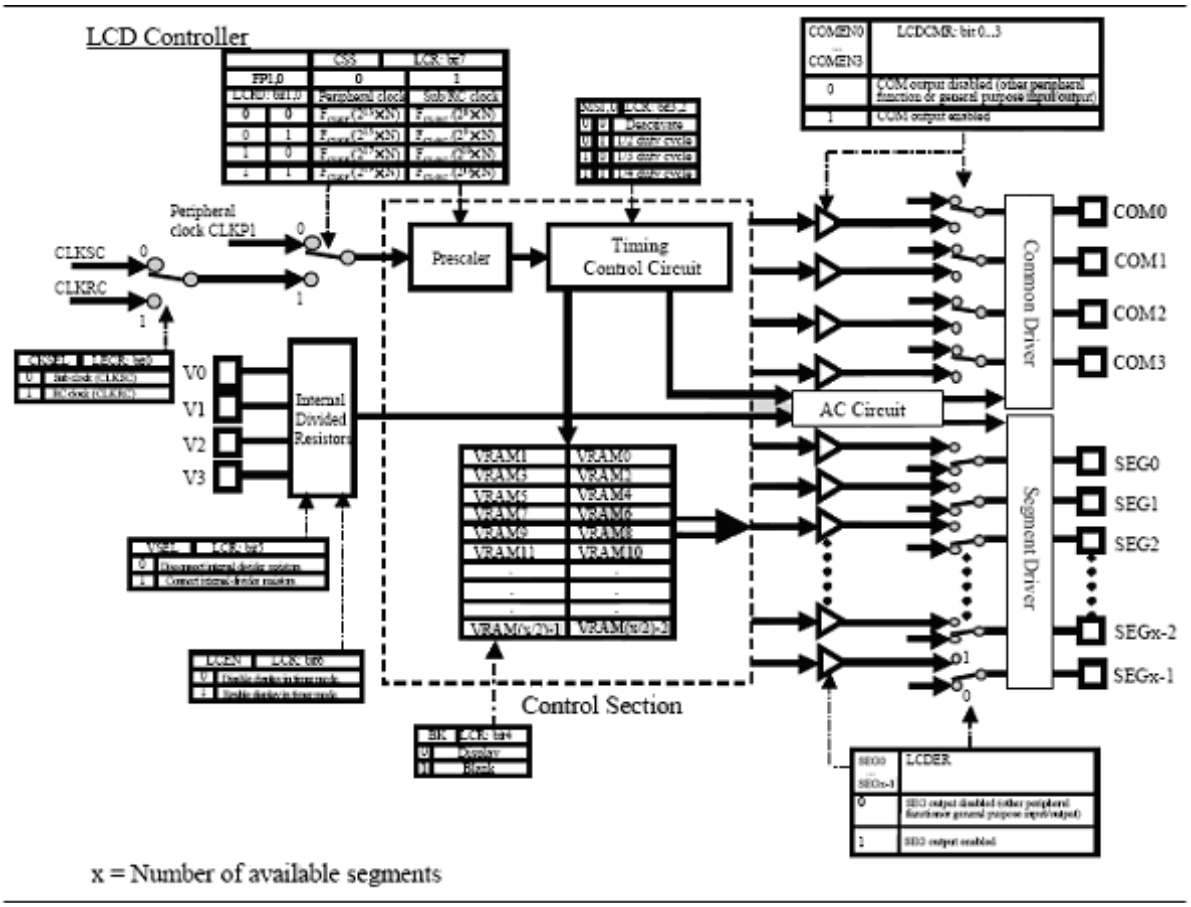
Table 1. MB963xx Microcontrollers with LCD Controller

MCU series	LCD-lines
MB96370	4x72
MB96380	4x65

3.1.1 Internal LCD Controller

The following figure shows the block diagram of the LCD controller of the 16FX Family:

Figure 7. Block Diagram of LCD Controller on MB963xx



The LCD controller/driver has up to 36 × 8-bit built-in display data memory, and controls the LCD display by means of four common outputs and 72 segment outputs. Three duty cycle output modes are available; one can be selected among the same. A LCD panel can be driven directly. The LCD controller/driver provides functions for directly displaying the contents of display data memory on the LCD panel by means of segment output and common output.

Features:

- LCD drive voltage divider resistance is built-in. External divider resistance can also be connected.
- Up to 4 common outputs (COM0 to COM3)
- Up to 72 segment outputs (SEG0 to SEG71)
- 36-byte display data memory (display RAM) is built-in
- The duty can be selected as 1/2, 1/3, 1/4
- Drives the LCD directly

3.1.2 LCD Control Registers

This section describes the list of registers that controls the LCD operation.

The LCR register configures the frame period, duty cycle setting and also controls bias and clock.

Table 2. LCR

bit7 — Register: LCR — bit0

CSS	LCEN	VSEL	BK	MS1	MS0	FP1	FP0
Clock Selection		Int./Ext. Bias	Display-Blanking	Display Mode Duty Cycles		Frame Period	

Note: MS0-MS1 bits will decide which of the COM lines among COM0 to COM3 needs to be interfaced to the LCD glass. However, the unused COM lines cannot be used as general purpose I/O lines or any other shared resource lines since it always outputs some signal (as shown in hardware manual) irrespective of the configured duty cycle. Hence up to two port pins needs to be compromised.

The LECR register also controls the clock.

Table 3. LECR

bit7 — Register: LECR — bit0

							CKSEL
							Clock Selection

Table 4. LCDCMR

bit7 — Register: LCDCMR — bit0

DTCH				COM3	COM2	COM1	COM0
Bias Control				Common Driver Enable			

Note: It should be noted that in order to use LCD, LCDCMR register needs to be set to a value equal to 0x0F. Setting any other value than this will not enable the LCD.

The LCDERn registers enable the LCD (SEGxx) segment outputs. If there are 72 LCD segments (x = 72) then there are 9 (x/8-1) such registers form LCDER0 to LCDER8. Each bit is corresponding to one LCD segment. If the bit is 0 then the LCD segment is disabled else it is enabled.

bit7 — Register: LCDER0 — bit0

SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
LCD Segment Enable							

•
•
•

Table 5. LCDER

bit7 — Register: LCDER(x/8-1) — bit0

SEG(x-1)						…	SEG(x-8)
LCD Segment Enable							

The LCDVER register enables the V0 to V3 pins to input the external bias. It should be noted that pins V0 to V3 are shared with some of the segment lines. Hence if any of the V0 to V3 pins is used, then the corresponding segment lines would be disabled and cannot be used.

Table 6. LCDVER

Register: LCDVER

bit7				V3	V2	V1	bit0 V0
				LCD Voltage Enable			

As discussed before, there can be up to 36 × 8-bit built-in display RAM. The total number VRAM registers depends on the total number of LCD segments. Considering 65 segments, there would be such 33 VRAM registers from VRAM0 to VRAM32. Each VRAM register controls two consecutive LCD segment lines.

e.g. VRAM0 register controls LCD segment line 0 and 1. Bits DL0 to DL3 corresponds to segment line 0 and DH0 to DH3 corresponds to segment line 1.

Bit DL0 should be set to 1, if the LCD segment at the junction of COM0 and SEG0 is to be lit.

Bit DL1 should be set to 1, if the LCD segment at the junction of COM1 and SEG0 is to be lit.

Bit DL2 should be set to 1, if the LCD segment at the junction of COM2 and SEG0 is to be lit.

Bit DL3 should be set to 1, if the LCD segment at the junction of COM3 and SEG0 is to be lit.

Bit DH0 should be set to 1, if the LCD segment at the junction of COM0 and SEG1 is to be lit.

Bit DH1 should be set to 1, if the LCD segment at the junction of COM1 and SEG1 is to be lit.

Bit DH2 should be set to 1, if the LCD segment at the junction of COM2 and SEG1 is to be lit.

Bit DH3 should be set to 1, if the LCD segment at the junction of COM3 and SEG1 is to be lit. So on and so forth.

Table 7. VRAM

Register: VRAM0

bit7							bit0
DH3	DH2	DH1	DH0	DL3	DL2	DL1	DL0
Segment 1 Control				Segment 0 Control			

•
•
•

Register: VRAM(x/2-1)

bit7							bit0
DH3	DH2	DH1	DH0	DL3	DL2	DL1	DL0
Segment x Control				Segment x-1 Control			

The LCD drive voltage is controlled either by external voltage divide resistors or internal voltage divide resistors. If the internal voltage divide resistors are to be used, VSEL bit of LCR register needs to be set to 1. In this case all the segment lines those are shared with V0 to V3 can be used.

The brightness can be controlled by connecting a variable resistor between the V_{CC} and V3 pins. Also the V3 bit of LCDVER register needs to be set to 1. In this case the segment line which is shared with the V3 cannot be used.

If the external voltage divide resistors are to be used, LCDVER register needs to be set to a value of 0x0F. In this case none of the segment lines those are shared with V0 to V3 can be used.

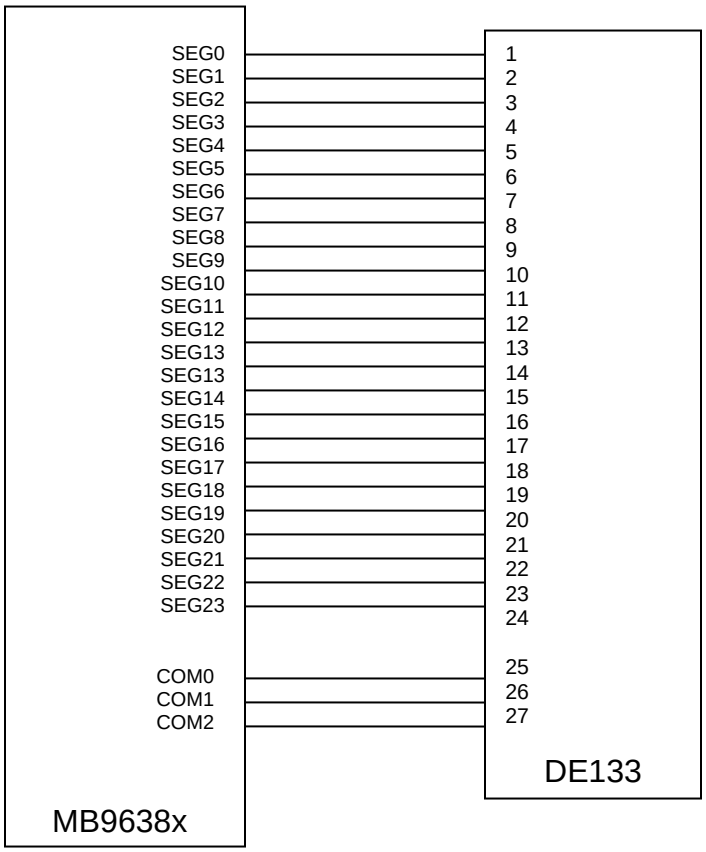
The circuit diagrams for above three cases are discussed in the hardware manual.

3.1.3 Interfacing MB9638x to LCD Glass via On-chip LCD Controller

The below connection diagram shows interfacing of MB9638x to DE133 series LCD (of Display Elektronik GmbH make). Here 3 COM lines and 24 SEG lines are used. Hence total 72 segments can be addressed. LCD controller is required to be used in ⅓ Duty Cycle mode.

It should also be note that the frame period (which is set by `LCR:FP`) of the LCD controller should be within the allowable driving frequency of LCD glass. For DE133 the driving frequency range is 30 Hz to 100 Hz.

Figure 8. Interfacing MB9638x to LCD DE133



The example which illustrates the software part of this interfacing is discussed in section 4.1.

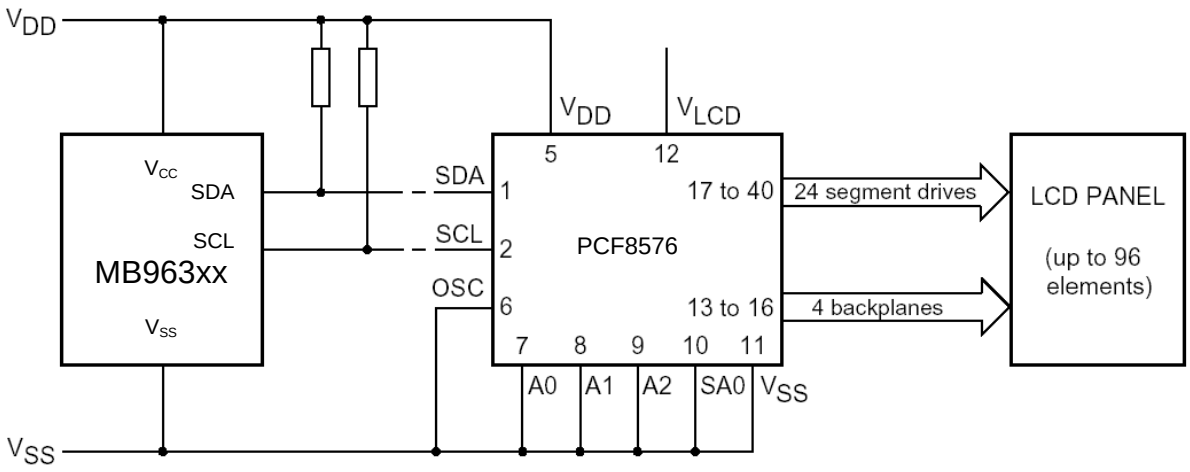
3.2 External LCD Controller/Driver

Some semiconductor companies offer different kinds of external LCD-controllers. The functionality is more or less the same but differences can be found in number of Common-/Segment-lines and of course the interface circuitry to the microcontroller.

3.2.1 Interfacing MB963xx to LCD Glass via External LCD Controller

The PCF8576 is a single-chip LCD-controller from Philips with included LCD-driver and data-RAM for up to 160 segments. By only 2-wires it can easily be connected to the microcontroller.

Figure 9. Interfacing MB963xx to LCD Glass via PCF8576



Here an uncompleted list of manufactures for external LCD-controllers:

Table 8. List of LCD Controller Manufacturer

Company	Web-site	Remark
Hitachi	www.hitachi.com	miscellaneous
Holtek	www.holtek.com	HT162x (4x32 … 8x64Segments)
MAXIM	www.maxim-ic.com	miscellaneous
NXP/Philips	www.nxp.com	I2C-interface: PCF8576 (4x40)
SAMSUNG	www.samsungelectronics.com	miscellaneous

3.3 Alphanumeric LCD Module

In some applications it is required to display alphabets (a,b,A,B etc.) and symbols (@, #, ↑, ↓ etc.) along with numbers (1,2,3 etc.). The LCD glass explained in section 3.1.3 cannot be used since it can only display only numbers and some alphabets. In such cases, it is required to use alphanumeric LCD module.

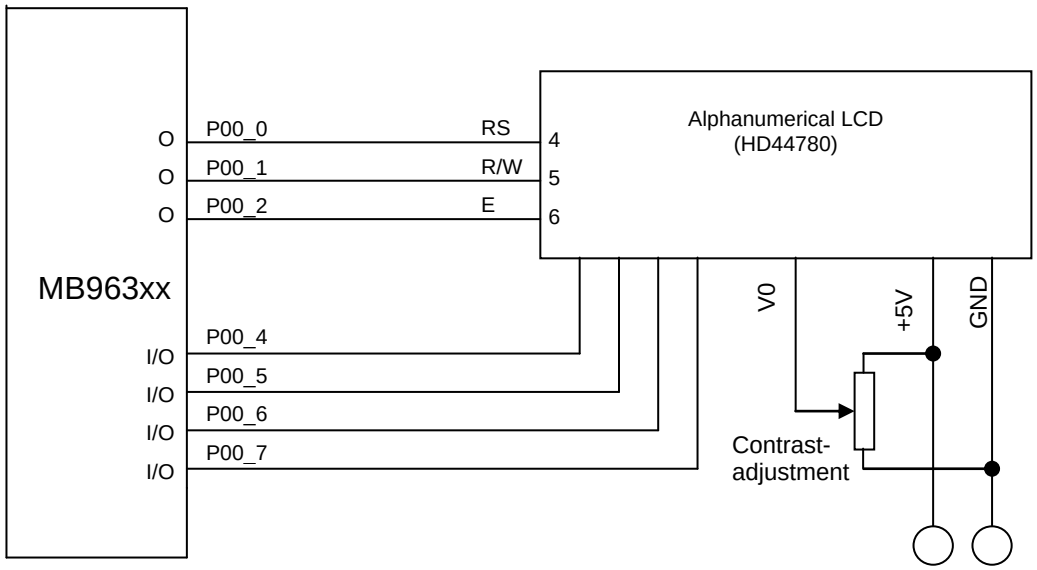
There are a plenty of such LCD modules available. These modules include LCD Controller as well as segment drivers. Additionally it provides a simple parallel interface to establish communication with a microcontroller. In mostly all modules, the HD44780 or derivative can be found. Hitachi Semiconductor settled like a standard with introducing this controller that makes it easy to use software libraries.

Up to 80 alphanumeric characters can be displayed, very common are displays in configurations like 2x14, 2x16, 2x20, 2x24, 2x40 and 4x20 characters.

3.3.1 Interfacing MB963xx to LCD Module via I/O Ports

For interfacing MB963xx to LCD Module HD44780 only seven digital I/O pins have to be reserved. Figure 10 shows how to connect such a standard LCD module. In this example, port 00 is used, but any other port can be used, too. The display supports two modes: Three Control-Lines (E, R/W, RS) are used, but for the data 4 or 8 data-lines can be used. In order to minimize the number of port-pins this example works in 4-bit mode. Every data byte is transmitted in two nibbles using a handshake protocol.

Figure 10. Interfacing MB963xx to HD44780



The example which illustrates the software part of this interfacing is discussed in section 4.2.

The same pin-out can be found at the most modules:

Table 9. HD44780 Pin Description

Pin	Symbol	Level	I/O-Function	Function (8-bit mode)	Function (4-bit mode)
1	Vss	-	-	Power supply (GND)	Power supply (GND)
2	Vcc	-	-	Power supply (+5V)	Power supply (+5V)
3	Vee	-	-	Contrast adjust	Contrast adjust
4	RS	0/1	I	0 = Instruction 1 = data	0 = Instruction 1 = data
5	R/W	0/1	I	0 = Write to 1=Read from LCD	0 = Write to 1=Read from LCD
6	E	1 → 0	I	Enable signal	Enable signal
7	D0	0/1	I/O	Data bus line 0 (LSB)	not used
8	D1	0/1	I/O	Data bus line 1	not used
9	D2	0/1	I/O	Data bus line 2	not used
10	D3	0/1	I/O	Data bus line 3	not used
11	D4	0/1	I/O	Data bus line 4	Data bus line 0 (LSB) and 4
12	D5	0/1	I/O	Data bus line 5	Data bus line 1 and 5
13	D6	0/1	I/O	Data bus line 6	Data bus line 2 and 6
14	D7	0/1	I/O	Data bus line 7 (MSB)	Data bus line 3 and 7 (MSB)

An initial sequence has to be sent to the module in order to select the 4bit interface as well as some basic display-settings. Data transfer between the controller and the LCD consists of instructions or data (ASCII-compatible character set).

Table 10. HD44780 Commands

Instruction	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor home
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position
Entry mode set	0	0	0	0	0	0	0	1	ID	S	Sets cursor move direction (I/D),
Display On/Off	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor
Cursor- / display-shift	0	0	0	0	0	1	SL	RL	*	*	Sets cursor-move or display-shift
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number
Set CGRAM address	0	0	0	1	a	a	a	a	a	a	Sets the CGRAM address.
Set DDRAM address	0	0	1	a	a	a	a	a	a	a	Sets the DDRAM address.
Read busy-flag	0	1	BF	a	a	a	a	a	a	a	Reads Busy-flag (BF) and address counter
Write to CG- or DD-RAM	1	0	d	d	d	d	d	d	d	d	Writes data to CGRAM or DDRAM.
Read from CG- or DD-RAM	1	1	d	d	d	d	d	d	d	d	Reads data from CGRAM or DDRAM.

DDRAM = Display Data RAM. CGRAM = Character Generator RAM. a=address. d=data

Table 11. HD44780 Commands Description

I/D: 0=Decrement cursor position,1=Increment cursor position	S: 0 = No display shift, 1 = Display shift
D: 0 = Display off, 1 = Display on	C: 0 = Cursor off, 1 = Cursor on
B: 0 = Cursor blink off, 1 = Cursor blink on	S/L: 0 = Move cursor, 1 = Shift display
R/L: 0 = Shift left 1 = Shift right	DL: 0 = 4-bit interface, 1 = 8-bit interface
N: 0 = 1/8 or 1/11 Duty (1 line), 1 = 1/16 Duty (2 lines)	F: 0 = 5x7 dots, 1 = 5x10 dots
BF: 0=Can accept instruction, 1=Internal operation in progress	

4 LCD Examples

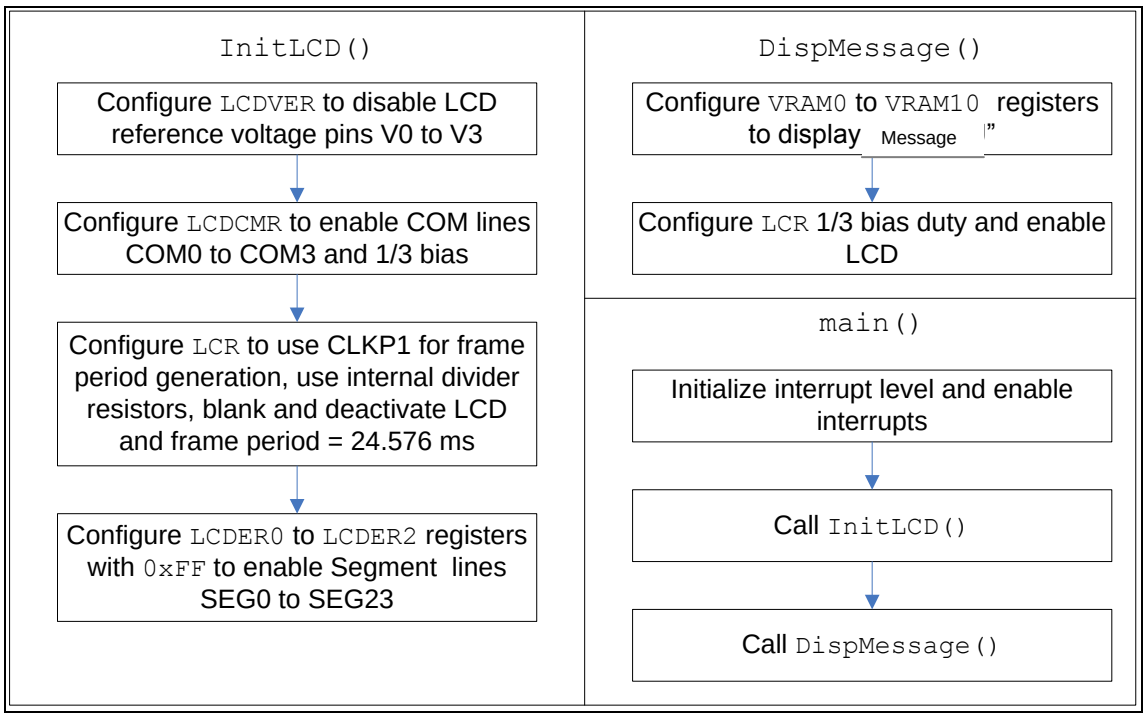
Examples for LCD

4.1 Interfacing MB9638x to LCD Glass DE133

The following software example initializes LCD with ⅓ bias and frame period of 40.69 Hz (considering CLKP1 of 4 MHz). Then it initializes display RAM (in accordance with the data to be displayed), enables the LCD in ⅓ duty mode and displays message.

For the corresponding connection diagram please refer section 3.1.3.

4.1.1 Flowchart



4.1.2 Code

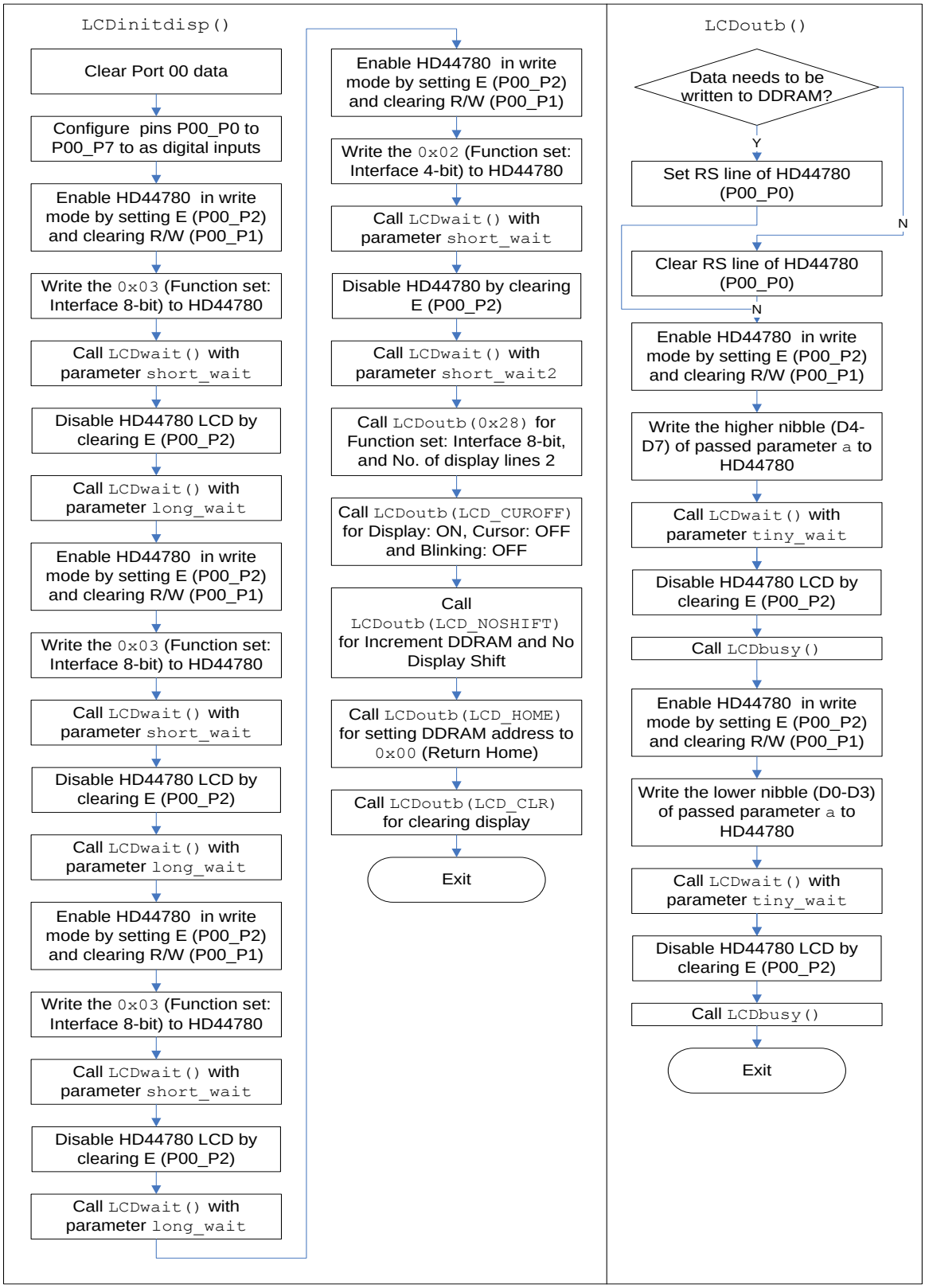
```
void InitLCD(void)
{
    LCDVER = 0x00;     // disable V0 to V3
    LCDCMR = 0x0F;     // enable COM lines, 1/3 bias
    LCR = 0x31;        // CLKP1, internal divider resistor, blank LCD, deactivate LCD,
                       // frame period = 4 MHz/(2^15*3) = 40.69 Hz
    LCDER0 = 0xFF;     // enable SEG0 to SEG7
    LCDER1 = 0xFF;     // enable SEG8 to SEG15
    LCDER2 = 0xFF;     // enable SEG16 to SEG23
}

void DispMessage(void)
{
    VRAM0 = 0x13;
    VRAM1 = 0x30;
    VRAM2 = 0x34;
    VRAM3 = 0x52;
    VRAM4 = 0x03;
    VRAM5 = 0x30;
    VRAM6 = 0x63;
    VRAM7 = 0x12;
    VRAM8 = 0x27;
    VRAM9 = 0x43;
    VRAM10 = 0x03;

    LCR_MS = 2;        // 1/3 duty
    LCR_BK = 0;        // enable LCD
}

void main(void)
{
    InitIrqLevels();
    __set_il(7);               // allow all levels
    __EI();                    // globally enable interrupts

    InitLCD();
    DispMessage();

    while(1);
}
```

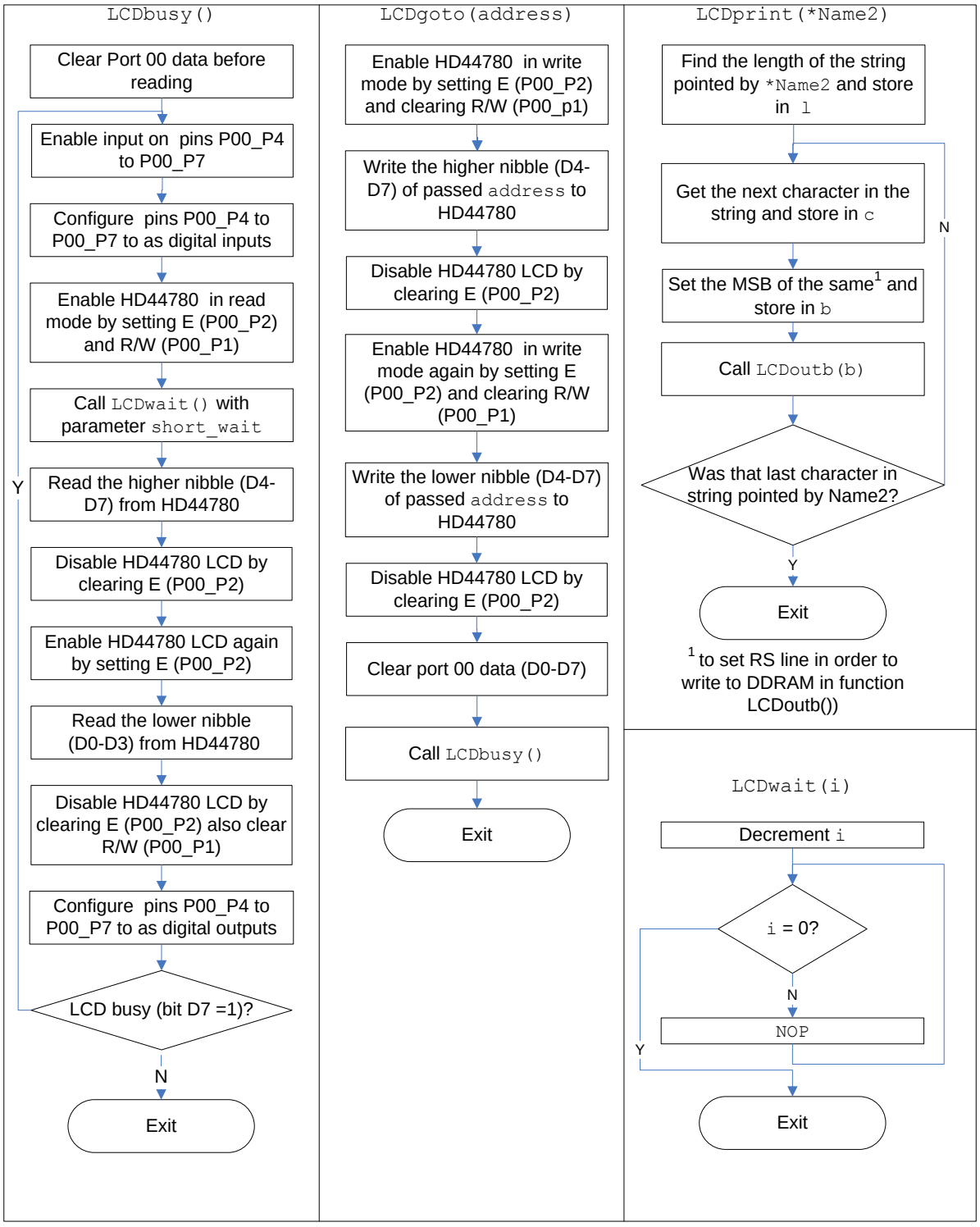
4.2 Interfacing MB963xx to LCD Module HD44780

The following software example initializes LCD module HD44780 in 4-bit mode and then displays message.

For the corresponding connection diagram please refer section 3.3.1.

4.2.1 Flowchart




LCDbusy()
Clear Port 00 data before reading
Enable input on pins P00_P4 to P00_P7
Configure pins P00_P4 to P00_P7 to as digital inputs
Enable HD44780 in read mode by setting E (P00_P2) and R/W (P00_P1)
Call LCDwait() with parameter short_wait
Read the higher nibble (D4-D7) from HD44780
Disable HD44780 LCD by clearing E (P00_P2)
Enable HD44780 LCD again by setting E (P00_P2)
Read the lower nibble (D0-D3) from HD44780
Disable HD44780 LCD by clearing E (P00_P2) also clear R/W (P00_P1)
Configure pins P00_P4 to P00_P7 to as digital outputs
LCD busy (bit D7 =1)?
Y
N
Exit
LCDgoto(address)
Enable HD44780 in write mode by setting E (P00_P2) and clearing R/W (P00_p1)
Write the higher nibble (D4-D7) of passed address to HD44780
Disable HD44780 LCD by clearing E (P00_P2)
Enable HD44780 in write mode again by setting E (P00_P2) and clearing R/W (P00_P1)
Write the lower nibble (D4-D7) of passed address to HD44780
Disable HD44780 LCD by clearing E (P00_P2)
Clear port 00 data (D0-D7)
Call LCDbusy()
Exit
LCDprint(*Name2)
Find the length of the string pointed by *Name2 and store in l
Get the next character in the string and store in c
Set the MSB of the same[1] and store in b
Call LCDoutb(b)
Was that last character in string pointed by Name2?
N
Y
Exit
[1] to set RS line in order to write to DDRAM in function LCDoutb())
LCDwait(i)
Decrement i
i = 0?
N
Y
NOP
Exit

4.2.2 Code

lcd.c

```
#include "lcd.h"

/*****************************************************************************
 *  Initdisp : Resets the LCD and configures the I/F for 4-Bit bus
 *****************************************************************************/
void LCDinitdisp(void)
{
        LCD_PDR=0;                      /* Port 0 Off           */
        LCD_DDR=0x0FF;                  /* Set Port 0 to output */
        LCD_PDR=0x34;                   /* Startup sequence     */
        LCDwait(short_wait);
        LCD_PDR=0x30;
        LCDwait(long_wait);
        LCD_PDR=0x34;
        LCDwait(short_wait);
        LCD_PDR=0x30;
        LCDwait(long_wait);
        LCD_PDR=0x34;
        LCDwait(short_wait);
        LCD_PDR=0x30;
        LCDwait(long_wait);
        LCD_PDR=0x24;
        LCDwait(short_wait);
        LCD_PDR=0x20;
        LCDwait(short_wait2);
        LCDoutb(0x028);         /* Switch to 4-bit mode */
        LCDoutb(LCD_CUROFF);            /* No Cursor            */
        LCDoutb(LCD_NOSHIFT);           /* No display shift     */
        LCDoutb(LCD_HOME);              /* Cursor home          */
        LCDoutb(LCD_CLR);       /* Display clear        */
}
/*****************************************************************************
 *  OUTB      sends one byte to the display
 *  PARAMETERS:     8-bit data
 *  RETURNS:        None
 *****************************************************************************/
void LCDoutb(unsigned char a)
{
        unsigned char b;
        b = (a & 0x70);         /* upper nibble    */
        if (a & 0x80)                   /* check MSB       */
        {
          b = (b | 0x01);               /* set RS line     */
        }
        b = (b | 0x04);         /* set E line      */
        LCD_PDR = b;                    /* send to LCD     */
        LCDwait(tiny_wait);
        b = (b & 0xFB);         /* clear E line    */
        LCD_PDR = b;                    /* send to LCD     */
        LCDwait(tiny_wait);

        b = (a & 0x0F);         /* lower nibble    */
        b = b << 4;                     /* to upper nibble */
        if (a & 0x80)                   /* check MSB       */
        {
          b = (b | 0x01);               /* set RS line     */
        }
        b = (b | 0x04);         /* set E line      */
        LCD_PDR = b;                    /* send to LCD     */
        LCDwait(tiny_wait);
        b = (b & 0xFB);         /* clear E line    */
        LCD_PDR =  b;                   /* send to LCD     */
        LCDbusy();                      /* check LCD       */
}
```

```
/*****************************************************************************
 *  BUSY            polls the busy-line (waits for the LCD to be ready)
 *  PARAMETERS:     None
 *  RETURNS:        None
 *****************************************************************************/
void LCDbusy(void)
{
        unsigned char b;
        LCD_PDR = 0;                    /* Port Off before reading ! */
        b = 0x80;
        while (b & 0x80)                /* LCDwait for Busy-line     */
        {
         LCD_DDR = 0x0F;                /* set Bus as input to read LCD-busy-flag (LCD_D3) */
         LCD_PDR = 0xF6;                /* busy request              */
         LCDwait(short_wait);
         b = LCD_PDR;                   /* read Port                 */
         LCD_PDR_P02 = 0;
         LCD_PDR_P02 = 1;               /* toggle E                  */
         LCD_PDR_P02 = 0;
         LCD_PDR_P01 = 0;
         LCD_DDR = 0xFF;                /* reset Port to output      */
        }
}

/*****************************************************************************
 *  GOTO            jump cursor to address
 *  PARAMETERS:     address
 *  RETURNS:        None
 *****************************************************************************/
void LCDgoto(unsigned char address)
{
  LCD_PDR = (address & 0xF0) | 0x04;          /* upper nibble, set E line  */
  LCD_PDR_P02 = 0;                            /* clear E line              */
  LCD_PDR = ((address<<4) & 0xF0) | 0x04;     /* lowe nibble, set E line   */
  LCD_PDR_P02 = 0;                            /* clear E line              */
  LCD_PDR = 0;
  LCDbusy();
}

/*****************************************************************************
 *  PRINT           displays a string
 *  PARAMETERS:     pointer to string
 *  RETURNS:        None
 *****************************************************************************/
void LCDprint(char *Name2)
{
 unsigned char c;
 unsigned char b;
 unsigned short i,l;

 l=strlen(Name2);
 for (i=0; i<l; i++)          /* go through string */
  {
        c=(Name2[i]);         /* pick char         */

    b=(c | 0x80);
    LCDoutb(b);
  }
}

/*****************************************************************************
 *  VERY PRIMITIVE DELAY LOOP *
 *****************************************************************************/
void LCDwait(unsigned long i)
{
 while(i--)
   __asm("\tNOP");
}
```

lcd.h

```
/*****************************************************************
LCD.H
Prototypes and constants for functions
*****************************************************************/

#ifndef _LCD_H
#define _LCD_H

/* LCD Port */
#define LCD_PDR               PDR00
#define LCD_DDR               DDR00
#define LCD_PDR_P01           PDR00_P1
#define LCD_PDR_P02           PDR00_P2

/* Control defines (use with LCDoutb) : */
#define LCD_CLR               0x01
#define LCD_HOME              0x03
#define LCD_CUROFF            0x0C
#define LCD_CURON             0x0F
#define LCD_NOSHIFT           0x06
#define LCD_1st_line          0x80
#define LCD_2nd_line          0xC0
#define LCD_3rd_line          0x94
#define LCD_4th_line          0xD4

/* Prototypes */
void LCDinitdisp(void);
void LCDoutb(unsigned char);
void LCDgoto(unsigned char address);
void LCDprint(char *Name2);
void LCDprintnum(int n);
void LCDprinthex(unsigned long n, unsigned char digits);
void LCDmovscr(char *Name2, unsigned long delay);
void LCDwait(unsigned long i);
void LCDbusy(void);

/* Wait times (depends on used MCU speed) */
#define tiny_wait   20
#define short_wait  400
#define short_wait2 700
#define long_wait   50000

#endif /* LCD_H */
```

main.c

```
void main(void)
{
  unsigned char ch;

  InitIrqLevels();
  __set_il(7);                /* allow all levels */
  __EI();                     /* globally enable interrupts */

  LCDinitdisp();              /* initialize the LC display */

  LCDgoto(LCD_1st_line);
  LCDprint("    XXX      "); /*Display message*/”
  LCDgoto(LCD_2nd_line);
  LCDprint("YYY");          /*Display message*/”

  while (1);
}
```

5 Conclusion

This chapter summarizes the results of the different control-mechanism of LCD.

The most useful way how to control a LCD depends very much on the application.

For industrial and automotive application, where lifetime and temperature influence is a very important point, a dedicated LCD-controller should be used. A microcontroller with built-in LCD-controller may help to save costs, because no further external components are necessary. In case that a big number of segments have to be controlled, then an external LCD-controller is indispensable.

If the application requires displaying text-messages then alphanumeric LC-modules will be a good choice, because hardware and software overhead can be reduced.

6 Additional Information

Information about Cypress Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

The software example related to this application note is:

96380_lcd_glass

96340_lcd_module

It can be found on the following Internet page:

http://www.cypress.com/16lx

Document History

Document Title: AN204779 - F²MC-16FX Family, LC-Display

Document Number:002-04779

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	07/20/2007	Initial release
			09/07/2007	Updated the entire document with review findings from PHu,
*A	5084284	NOFL	04/14/2016	Migrated Spansion Application Note MCU-AN-300230-E-V11 to Cypress format
*B	5870038	AESATP12	09/01/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2007-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.