Freescale Semiconductor
Application Note

Document Number: AN3796
Rev. 4, 10/2011

LCD Driver Specification

by: Alejandra Guzmán, Luis Puebla, and Saul Salazar
Microcontroller Solutions Group
Guadalajara, México

1 Introduction

This document describes a driver for twisted nematic displays (TN) or super twisted nematic displays (TNS). This LCD driver allows the user to customize glass requirements with the microcontroller LCD module.

This driver was tested for the following microcontroller demo boards: the TWR-K40X256, TWR-K53N512, MCF51EM256 (DEMOEM), MC9S08LG32 (DEMO9S08LG32), MC9S08LL16 (DEMO9S09LL16), MC9RS08LA8 (DEMO09RS08LA8), and MC9RS08LE4 (DEMO09RS08LE4).

The software architecture was designed to provide seamless migration between these devices.

This document is intended to be used by all software development engineers, test engineers, and anyone else who has to use the microcontrollers with an LCD.

Contents

Issues and suggestions about this document and driver must be provided through the support webpage at www.freescale.com/support.

2 Introduction to the Twisted Nematic (TN) LCD

A twisted nematic (TN) display is used in most LCD applications; this device consists of a nematic liquid crystal sandwiched between two plates of glass. The inner glass surfaces are coated with a transparent metal oxide film that acts as an electrode. The surfaces are used to apply voltages across the cells.

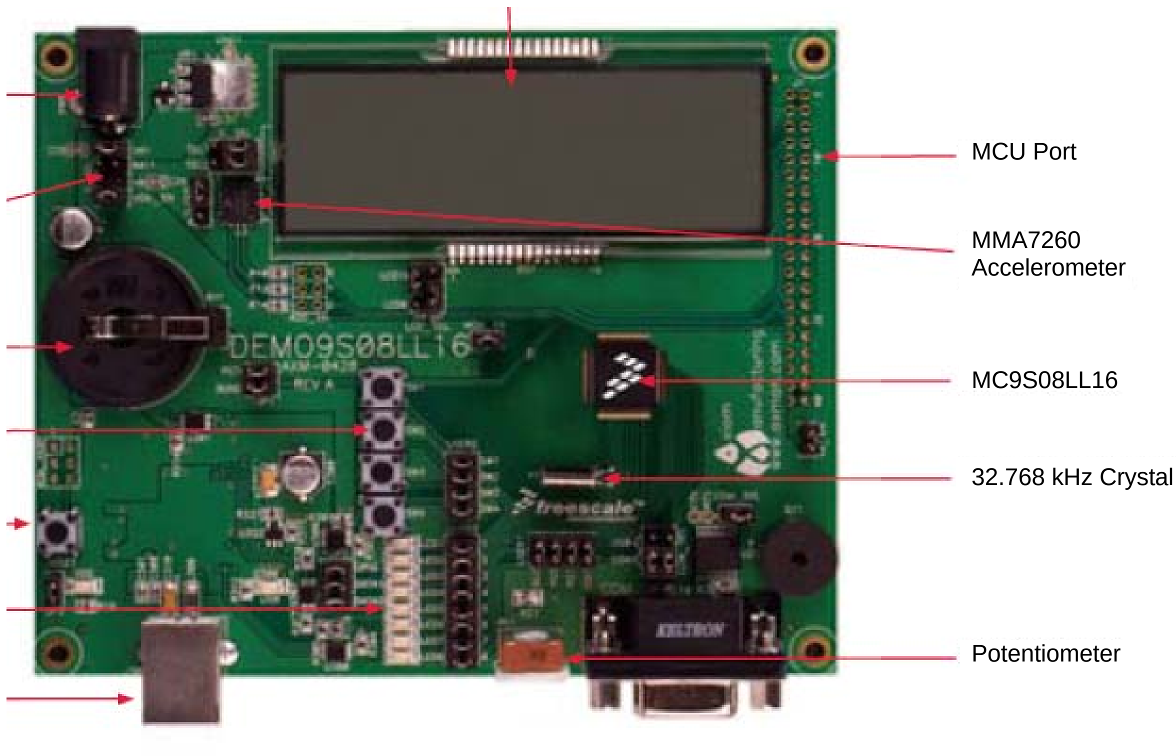


Figure 1. Twisted nematic LCD

2.1 Principle of Operation

The light entering the display from the top is polarized by the top polarizer and as it travels through the liquid crystals its polarization rotates with the molecules. When it emerges its polarization has been changed by 90 degrees and passes through the bottom polarizer.

However, when a voltage is applied to the electrodes the majority of the molecules are arranged vertically. In this case, polarization of the light passing through the liquid crystals remains unchanged and the light beam is blocked by the bottom polarizer.

2.2 Types of TN LCDs

There are two types of LCDs:

- Static LCDs — Have only one backplane (BP) electrode and one frontplane (FP) pin to turn on one segment. Therefore, a 7-segment display needs eight terminals to display one character. If you increase the number of segments you increase the number of terminals, the cost of the LCD, and the total cost of the application.

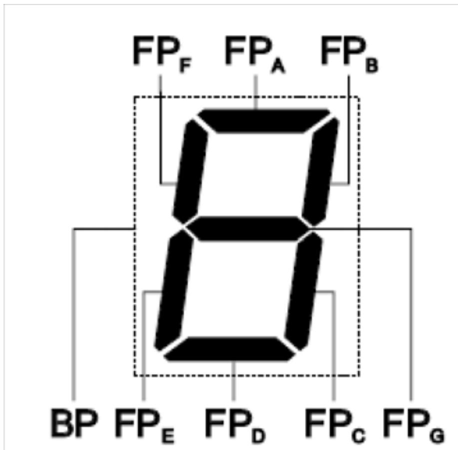


Figure 2. Static LCD

- Dynamic LCDs — Help optimize the required number of terminals needed to drive the LCD. Instead of having one BP and one FP, dynamic displays reduce the number of terminals by multiplexing the segments; this means that two or more segments are connected to the same terminal. The way that dynamic displays work is similar to keyboards.

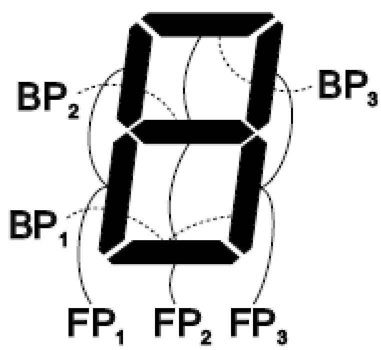


Figure 3. Dynamic LCD

For further information about dynamically driven LCDs, see Freescale application note *XGATE Library: TN/STN LCD Driver* (document AN3219), available on the Freescale.com website.

3 LCD Software

3.1 LCD Software Architecture

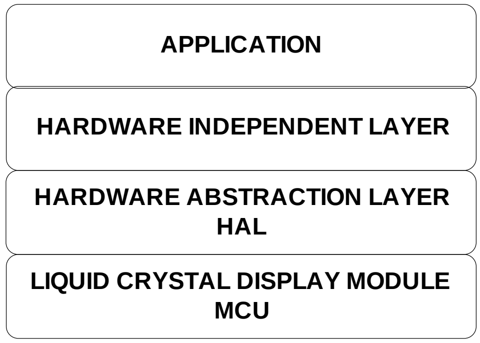


Figure 4. Software architecture

Hardware abstraction layer (HAL) — The hardware abstraction layer is defined as a collection of software components that directly access hardware resources. In this layer, the LCD driver defines macros and functions that configure the custom glass requirements in the LCD module registers.

Hardware independent layer (HIL) — The hardware independent layer is defined as a collection of software components that access hardware resources through HAL. Peripheral drivers are implemented in this layer.

4 Hardware Abstraction Layer (HAL) Implementation

The HAL module defines the macros and functions needed to match the custom glass hardware specifications with the MCU LCD registers. This information is obtained from each custom glass specification. The changes to these macros are performed through modifications on the macro definitions located in the LCD_HAL.h header file. The user functions are located in LCD_HAL.c.

The user is responsible for providing correct definitions for these hardware access macros. Descriptions of the individual macros and examples of their definitions are given below.

4.1 _LCDBACKPLANES

The LCD backplane macros defines the number of backplane electrodes on the custom glass. The duty ratio of the waveforms generated by the LCD module is 1/(_LCDBACKPLANES).

This module supports values from 1–8.

Example 1.

```
#define _LCDBACKPLANES     (8 )          // # of backplanes
```

4.2 _LCDCLKPSL

This macro defines the LCD clock prescaler which in combination with the number of backplanes of the LCD, and determines the LCD frame frequency.

The LCD frame frequency is the number of times that the LCD is energized per second. The LCD module frame frequency must be selected to prevent the LCD display from flickering (LCD module frame frequency is too low) or ghosting (LCD module frame frequency is too high).

To avoid these conditions, an LCD module frame frequency in the range of 28 Hz to 58 Hz is required. Table 1 shows the LCD frame frequency calculations.

Table 1. LCD frame frequency calculations

Duty Cycle	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
Y	16	8	5	4	3	3	2	2
LCLK[2:0]								
0	76.3	76.3	81.4	76.3	81.4	67.8	87.2	76.3
1	61	61	65.1	61	65.1	54.3	69.8	61
2	50.9	50.9	54.3	50.9	54.3	45.2	58.1	50.9
3	43.6	43.6	46.5	43.6	46.5	38.8	49.8	43.6
4	38.1	38.1	40.7	38.1	40.7	33.9	43.6	38.1
5	33.9	33.9	36.2	33.9	36.2	30.1	38.8	33.9
6	30.5	30.5	32.6	30.5	32.6	27.1	34.9	30.5
7	27.7	27.7	29.6	27.7	29.6	24.7	31.7	27.7

4.3 _LCDCLKSOURCE

The _LCDCLKSOURCE macro defines the LCD module clock source. The available clocks are the internal (also called alternate) clock or the external clock of 32.768 kHz. The supported clock range is from 30 kHz to 39.053 kHz.

- 0 — Selects external clock source
- 1 — Selects alternate clock source

4.4 _LCDVSUPPLY

The LCD voltage supply macro defines whether the LCD module power supply is internal or external. These device sources change in each device.

Supply sources for the MCF51EM256 are:

- 0 — Drive VLL2 internally from V_{DD}
- 1 — Drive VLL3 internally from V_{DD}
- 2 — Drive VLL1 internally from V_{LCD}
- 3 — Drive VLL3 externally Or V_{IREG}

Supply sources for the MC9S08LG32 are:

- 0 — Drive VLL2 internally from V_{DD}
- 1 — Drive VLL3 internally from V_{DD}
- 2 — Reserved
- 3 — Drive VLL3 externally

Supply sources for the MC9S08LL16 are:

- 0 — Drive VLL2 internally from V_{DD}
- 1 — Drive VLL3 internally from V_{DD}
- 2 — Drive VLL1 internally from V_{LCD}
- 3 — Drive VLL3 externally Or V_{IREG}

Supply fsources for the MC9RS08LA8:

- 0 — Drive VLL2 internally from V_{DD}
- 1 — Drive VLL3 internally from V_{DD}
- 2 — Reserved
- 3 — Drive VLL3 externally

Supply sources for the MC9RS08LE4:

- 0 — Reserved
- 1 — Drive VLL3 internally from V_{DD}
- 2 — Reserved
- 3 — Drive VLL3 externally

4.5 _LCDCPSEL

The LCD charge pump selector macro defines the type of supply for the LCD voltages VLL1, VLL2, and VLL3. The LCD module provides two options for the supply voltage: resistor network or charge pump.

This option is only available for the MCF51EM256, MC9S08LG32, MC9S08LL16, and MC9RS08LA8 families. The defined value (0 or 1) will not affect the MC9RS08LE4.

- 0 — Selects the resistor network
- 1 — Selects the charge pump

4.6 _LCDLOADADJUST

The load adjust macro configures the LCD module to manage a different LCD glass capacitance. The capacitance of the LCD depends on the custom glass. The value written in this macro is related to the type of voltage source selected (resistor network or charge pump). The results for the different possible combinations of _LCDLOADADJUST and _LCDCPSEL (0 — resistor network, 1 — charge pump) for the MCF51EM256, MC9S08LG32, and MC9S08LL16 are shown in Table 2.

Table 2. _LCDLOADADJUST and _LCDCPSEL Combinations for the MCF51EM256, MC9S08LG32, and MC9S08LL16

_LCDLOADADJUST	_LCDCPSEL = 1	_LCDCPSEL = 0
0	8000 pf	2000 pf
1	6000 pf	2000 pf
2	4000 pf	8000 pf
3	2000 pf	8000 pf

The configuration for the MC9RS08LA8 and MC9RS08LE4 devices are shown in Table 3. For the MC9RS08LA8 the value of _LCDCPSEL must be 0 and for the MC9RS08LE4 the value does not matter.

Table 3. _LCDLOADADJUST Configuration for the MC9RS08LA8 and MC9RS08LE4

_LCDLOADADJUST	
0	2000 pf
1	2000 pf
2	8000 pf
3	8000 pf

NOTE

For further details consult the specific device reference manual.

4.7 _LCDFRAMEINTERRUPT

The LCD frame interrupt macro enables an LCD interrupt that matches the LCD module frame frequency.

- 0 — Disable interrupt
- 1 — Enable interrupt

4.8 _LCDBLINKRATE

The LCD blink rate macro selects the frequency at which the LCD display blinks when the software blink control is enabled. This macro can be any number between 0–7. Equation 1 shows how the LCD blink rate is calculated.

Eqn. 1

$$\text{LCD module blink rate} = \frac{LCDCLK}{2^{(12 + \text{\_LCDBLINKRATE})}}$$

Table 4 shows calculations for all values of BRATE.

Table 4. BRATE values

BRATE[2:0]	0	1	2	3	4	5	6	7
LCD Clock	**Blink Frequency (Hz)**							
30 kHz	7.32	3.66	1.831	.916	.46	.23	.11	.06
32.768 kHz	8	4	2	1	.5	.25	.13	.06
39.063 kHz	9.54	4.77	2.38	1.19	.6	.30	.15	.075

NOTE

The MCF51EM256, MC9S08LG32, MC9S08LL features software control. The MC9RS08LA and MC9RS08LE do not feature software control.

4.9 _CHARNUM

This macro defines the number of alphanumerics for the custom glass.

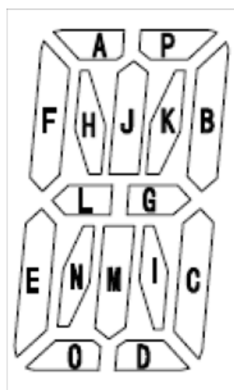


Figure 5. 16-Segment alphanumeric

4.10 _LCDTYPE

This macro defines the number of LCDWF needed to drive one alphanumeric of the custom glass.

4.11 Custom Glass Configuration

The macros explained below interface between the custom glass pinout and the MCU LCD module. The user must take into consideration that these are only definitions, therefore these macros are needed and called in the LCD_HAL.c file.

NOTE

Download the AN3796SW.zip file from www.freescale.com for a spreadsheet that helps in filling-out these definitions.

4.11.1 EnableLCDpins

Description:

Enable the corresponding LCD pin for the LCD operation.

Name:

EnableLCDpins(RegNum, Mask)

Parameters:

- RegNum — Number of the register to write
- Mask — Mask to habilitate LCD pins

Excel Location:

Sheet — Custom Glass Column— L

Example 2.

```
#define     _PEN0()        EnableLCDpins (0,255)        //LCD0-LCD7 are LCD pins
```

4.11.2 EnableBackplanes

Description:

Enable the corresponding LCD pin for the backplane operation.

Name:

EnableBackplanes(RegNum, Mask)

Parameters:

- RegNum — Number of the register to write
- Mask — Mask to select backplanes

Excel Location:

Sheet — Custom Glass Column — M

Example 3.

```
#define     _BPEN0()       EnableBackplanes (0, 15)      // LCD0-LCD3 are backplanes
```

4.11.3 SetBackplane

Description:

Gives the common number (COM) to the previous enabled backplanes.

Name:

SetBackplane(ComNum, LCDn)

Parameters:

- ComNum — COM number

- LCDn — Number of the LCD pin connected to the backplane

Excel Location:

Sheet — Custom Glass Column — N

Example 4.

```
#define     _SETCOM1()     SetBackplane (1, 3)     /LCD3 like COM1
```

4.11.4 _CharacterPlace

Description:

This macro defines the LDCWaveforms directions needed to write the alphanumeric character.

Name:

CharacterPlace(LCDn)

Parameters:

- LCDpin — LCDn number of pin connected

Excel Location:

Sheet — Custom Glass Column — O

Example 5.

```
#define   CHAR1A (5)           //Alphanumeric 1 drive by LCD5 and LCD15
#define   CHAR1B (15)
```

4.11.5 AllSegmentsON

Description:

Turns on all the segments of the custom glass.

Name:

AllSegmentsON(LCDn, Mask)

Parameters:

- LCDn — LCDWF that needs to be turned on
- ComMask — Bits that are turned on (example: 8 COM = 255, 2 COM = 3)

Excel Location:

Sheet — Custom Glass Column — Q

Example 6.

```
#define SEGMENT10_ON AllSegmentsON (14, 255)          //Turns on all the bits from the LCD14
```

4.11.6 SymbolON

Description:

Turns on the segment dedicated to write a specific symbol.

Parameters:

BufferNumber — Number of the LCD pin connected

BitNumber — COM number that turns on that symbol

Excel Location:

Sheet — Special Symbols Column — H

Example 7.

```
#define LCD_CLOCK_ON SymbolON(11,7)
                  //Bit7 from LCD11 turns on the clock symbol
```

4.11.7 SymbolOFF

Description:

Turns off the segment dedicated to write a specific symbol.

Name:

SymbolOFF(LCDn, BitNumber)

Parameters:

- BufferNumber — Number of the LCD pin connected
- BitNumber — COM number that turns on that symbol

Excel location:

Sheet — Special Symbols Column — I

Example 8.

```
#define   _LCD_CLOCK_OFF SymbolOFF(11,7)          // Turns off bit7 from LCD11
```

4.12 LCD fault detection

LCD fault detection capability is included only in Kinetis family.

Most failures, where a LCD segment is either unexpectedly on or off, result in an erroneous information that can mislead a user and cause a dangerous situation. The LCD Display Fault Detect circuit's (LFD) function find faults in the LCD display, display connector, and board connections between the MCU and the display.

The LCD panel can be regarded as a matrix of segments, as shown in the following figures. Any open/short connection changes the capacitive characteristics of the segment matrix. The pullup fault detection checks the capacitive characteristic of an LCD_Pn pin by applying a weak pullup voltage to the display capacitor matrix. The rise time response is sampled and summed within a user defined time frame and stored for post-processing in the FDSR[FDCNT] bitfield. The summing of the digitized values uses the principal that the response of a capacitor for each charge/discharge is constant for the same loading conditions.

For more information on how LCD fault detection operates see “LCD fault detect circuit” section in the Reference Manual of K30 and K40 families available from www.freescale.com/kinetis.

void vfnlcd_pinmux(uint8 mux_val)

Description:

Configure LCD pin to operate in LCD normal operation or for fault detection. To measure the capacitance of the pin all LCD pins through the port control must be configured for ALT7 function which enable the LCD fault circuitry. During LCD normal operation the port control must be configured for ALT0.

NOTE

> When fault detect measurement is performed, a glitch on the LCD is observed due to handling of pull ups used for the fault detect circuit. Once the fault detection is completed the user must ensure to return the port control to ALT0 configuration.

ANSIC prototype:

void vfnlcd_pinmux(uint8 mux_val)

Input parameters:

mux_val = 0 —Configure LCD pins for LCD normal operation

mux_val = 7 — Configure LCD pins for LCD fault detection. Configuring to these values allow the LCD Fault detect circuitry to control the pull-up for proper measurement.

NOTE

> A glitch on the LCD, or darkness on the LCD occurs while Fault detection is on progress.

Return value:

None

uint8 u8fnlcd_measure_pin_cap(uint8 pinid, uint8 pin_type, uint8 fdprs_val, uint8 fdsww_val)

Description: measure capacitance of the pin

ANSIC prototype:

uint8 u8fnlcd_measure_pin_cap(uint8 pinid, uint8 pin_type, uint8 fdprs_val,uint8 fdsww_val)

Input parameters:

pinid (LCDWF number 0-47)
pin_type FP_TYPE=0 or BP_TYPE =1
fdprs_val Fault detect preescaler value 0-7
fdsww_val Fault detect windows width 0-7

Return value:

capacitance value (FDSR_FDCNT)

Example 9.

Measuring capacitance of all LCD pins enabled.

```
void ldc_fault_detection_measure(void)
  {
    uint8  i;
    uint8 lcd_cap_val;

   vfnlcd_pinmux(7);
    for(i=0;i<_LCDUSEDPINS;i++)
    {
      if (i<_LCDFRONTPLANES)
      {
      lcd_cap_val =
u8fnlcd_measure_pin_cap(WF_ORDERING_TABLE[i],FP_TYPE,lcd_fp_fdprs_val,lcd_fp_fdsww_val);
      }
      else
     {
       lcd_cap_val =
u8fnlcd_measure_pin_cap(WF_ORDERING_TABLE[i],BP_TYPE,lcd_bp_fdprs_val,lcd_bp_fdsww_val);
     }
     lcd_pin_cap[i] = lcd_cap_val;
    }
   vfnlcd_pinmux (0);
  }
```

5 LCD HAL Functions

5.1 vfnLCD_Init

Description:

Initializes the LCD with the parameters given in the LCD_HAL.h.

ANSIC prototype:

void vfnLCD_Init (void);

Input parameters:

None

Return value:

None

Typical usage:

```
main()
{
  vfnLCD_Init();
for (;;)
 {  }
}
```

CAUTION

This caution is for MCF51EM256 devices. LCD registers require several cycles between write accesses. Only byte writes should be used to write these registers. Finally, consecutive writes must be seperated by at least one non-write cycle.

5.2 vfnLCD_EnablePins

Description:

Provides the LCD characteristics to the MCU pins previously selected in the LCD_HAL.h file. This function is used on vfnLCD_Init. The LCD characteristics help with the generation of the waveforms that drive the liquid crystal display.

ANSIC prototype:

void vfnLCD_EnablePins (void)

Input parameters:

None

Return value:

None

5.3 vfnLCD_ConfigureBackplanes

Description:

Enables an LCD pin to be a backplane and gives the number of the COM corresponding to each backplane. This function is used for the LCD initialization sequence and therefore is called from vfnLCD_Init.

ANSIC prototype:

void vfnLCD_ConfigureBackplane (void)

Input parameters:

None

Return value:

None

5.4 vfnLCD _Clear_Display

Description:

Turns off all the segments of the LCD. This function uses the previous AllSegmentsON definition.

ANSIC prototype:

void vfnLCD_Clear_Display (void)

Input parameters:

None

Return value:

None

Typical usage:

```
#include “lcd.h”
main()
{
vfnLCD_Init();
vfnLCD_Clear_Display ();
for (;;)
 { }
}
```

5.5 vfnLCD _Set_Display

Description:

Turns on all the segments of the LCD. This function uses the previous AllSegmentsON definition.

ANSIC prototype:

void vfnLCD_Set_Display (void)

Input parameters:

None

Return value:

None

Typical usage:

```
#include “lcd.h”
main()
{
vfnLCD_Init();
vfnLCD_Set_Display ();
vfnLCD_Clear_Display ();
for (;;)
 { }
}
```

5.6 vfnLCD_Home

Description:

Reset the pointer to the first alphanumeric position — LCD_CharPosition = 0.

Input parameters:

none

Return value:

none

Typical usage:

```
#include “lcd.h”
main()
{
vfnLCD_Init();
LCD_writeMSG(“”)
vfnLCD_Home();
LCD_writeMSG(“200”); “LCD result = 200 Volts”
for (;;)
 { }
}
```

5.7 vfnLCD_Write_Char

Description:

Writes one ASCII character on the next alphanumeric. If the alphanumeric counter reaches the maximum value, vfnLCD_Home is called.

ANSIC prototype:

void vfnLCD_Write_Char (UINT8 u8Value);

Input parameters:

- UINT8 — u8Value (ASCII to write on the alphanumeric)

Return value:

none

Typical usage:

```
#inlcude “lcd.h”
main()
{
vfnLCD_Init();
vfnLCD_Write_Char (‘a’);
for (;;)
 { }
}
```

5.8 vfnLCD_Write_MsgPlace

Description:

Write a message on an LCD alphanumeric specific position. Write only the number of characters specified by the user.

ANSIC prototype:

vfnLCD_Write_MsgPlace(UINT8 _POINTER pu8Message, UINT8 u8NumChars, UINT8 u8Place)

Input parameters:

- UINT8*pu8Message — The first character on the array to write
- UINT8 u8NumChars — The numbers of characters to write
- UINT8 u8Plcae— Address alphanumeric

Return value:

None

Typical usage:

```
#inlcude “lcd.h”
const byte cbaMessage1 [ ] = {“123456”};
main()
{
vfnLCD_Init();
vfnLCD_Write_Char (‘a’);
vfnLCD_Write_MsgPlace(&cbaMessage1[0], 3); //Only write “123”
for (;;)
 { }
}
```

5.9 vfnLCD_Write_Msg

Description:

Write a message to the LCD alphanumeric space. If the message is longer than the maximum number of alphanumeric characters, it will truncate the message. If the message is shorter than the maximum number of alphanumeric characters it will write the message and the remaining alphanumeric characters will be written with spaces.

ANSIC prototype:

void vfnLCD_Write_Msg (UINT8 _POINTER pu8Message);

Input parameters:

- UINT8 * pu8Message — The first character on the array to write

Return Value:

none

Typical Usage:

```
#inlcude “lcd.h”
const byte cbaMessage1 [] = {“123456”};
main()
{
vfnLCD_Init();
vfnLCD_Write_Char (‘a’);
vfnLCD_Write_Msg (&cbaMessage1[0];
for (;;)
 { }
}
```

5.10 vfnLCD_Contrast

Description:

The software contrast is updated with the value of the input parameter. The permitted values for the contrast are in the range 0x00–0x0F. By increasing the value the glass becomes more opaque, decreasing the value of the contrast makes the LCD more transparent.

Prototype:

vfnLCD_Contrast (UINT8 Contrast)

Input parameters:

- byte — contrast

Return value:

None

Typical usage:

```
#include “lcd.h”
main()
{
    vfnLCD_Contrast(15);
    for (;;)
{ }
}
```

NOTE

The contrast function can only be used with the MCF51EM256 and MC9S08LL16 devices.

5.11 vfnLCD_GoTo

Description:

Move cursor to a specific position.

Prototype:

vfnLCD_GoTo (Place)

Input parameters:

UINT8 — Place,

Return value:

None

Typical usage:

```
#include “lcd.h”
const byte cbaMessage1 [] = {“123456”};
main()
{
    vfnLCD_Init();
    vfnLCD_Write_Char(“a”);
    vfnLCD_GoTo(3);                 //select the position to begin writing
    vfnLCD_Write_Char(“b”);         //LCD result = “a b”
    for (;;)
{ }
}
```

6 Custom Glass Implementation

Using good practices to configure the LCD hardware pinout simplifies the software. Consider the following:

- Drive one segment with the same backplane at all alphanumerics.
- Arrange all the symbols on the same pin.
- Increasing the number of backplanes increases the number of segments that are possible to drive.

When customizing software for a specific glass, the following steps must be executed:

1. Make a new project for the MCU that you select.

2. Add the files LCD_HAL.h, LCD_HAL.c, and LCD.h to the project.
3. Include LCD.h in main (#include "LCD.h").
4. Open the LCD_HAL.h., customize the LCD configuration macros and uncomment the MCU from that application, leaving the other two MCUs (not used) commented, as shown.

```
//#define MC9RS08LE4
//#define MC9RS08LA8
#define MC9S08LL16
//#define MC9S08LG32
//#define MCF51EM256
```

5. Open the spreadsheet and fill the columns in the custom glass worksheet that are mentioned below.
   — Pin (Column A) — Custom glass pin
   — LCDn (Column B) — Number of the MCU LCD pin connected to the custom glass
   — BP(Column C) — COM number connected to the pin; if that pin is not a backplane leave the cell blank
   — ChNx (Column D) — Select the number of alphanumeric characters that the custom glass is driving. When alphanumerics need more than one pin, the alphabet will be used for identification.

Table 5. Spreadsheet

Pin	LCD*n*	BP	ChNx
1	0	1	
2	1	2	
3	2	3	
4	3	4	
5	4		1a
6	5		1b
7	6		1c
8	7		1d
9	8		2a
10	9		2b
11	10		2c
12	11		2d
13	12		7a
14	13		7b
15	33		7c
16	34		7d
17	41		8a
18	42		8b
19	43		8c
20	14		8d
21	15		9a
22	16		9b
23	28		9c

Table 5. Spreadsheet (continued)

Pin	LCD*n*	BP	ChNx
24	27		9d
25	26		
26	25		
27	24		
28	23		
29	22		6d
30	36		6c
31	35		6b
32	21		6a
33	20		5d
34	19		5c
35	18		5b
36	17		5a
37	32		4d
38	31		4c
39	40		4b
40	39		4a
41	38		3d
42	37		3c
43	30		3b
44	29		3a

6. Copy the information shown in Table 6 from the custom glass worksheet into the LCD_HAL.h.

Table 6. Custom glass worksheet to LCD_HAL.h

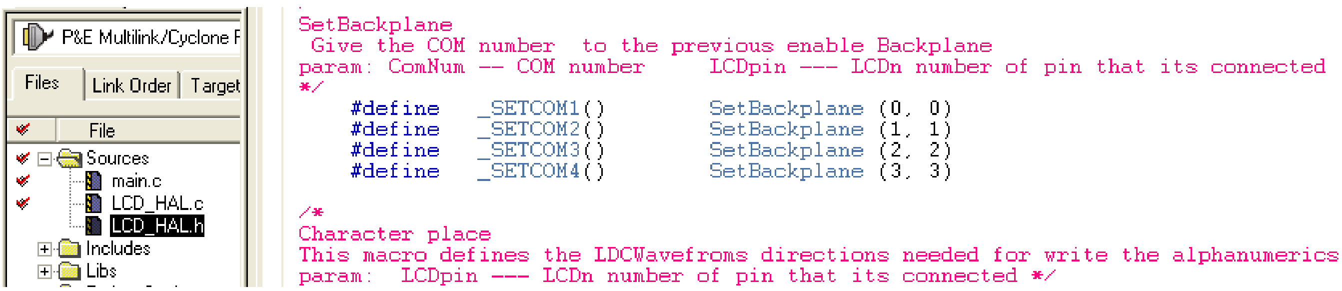
Column	Place in LCD_HAL.h
M	EnableLCDPins
N	EnableBackplanes
O	SetBackplanes
P	CharacterPlace
R	SegmentsON
S	SegmentsOFF

Copy from the worksheet:

Table 7. Example for SetBackplane macros

Set Backplane Macro		
#define	SetCom1()	SetBackplane (0, 0)
#define	SetCom2()	SetBackplane (1, 1)
#define	SetCom3()	SetBackplane (2, 2)
#define	SetCom4()	SetBackplane (3, 3)

Paste to the project:



7. On the worksheet named Special Symbols fill the columns:
   Name (Column C) — Name of the custom glass segments.
   Pin (Column D) — Custom glass connected to that symbol.
   COM (Column E) — Number of backplane or common that turns on that symbol.
8. Copy the columns shown in Table 8 from the special symbols worksheet and paste into the LCD_HAL.h.

Table 8. Special symbols worksheet to LCD_HAL.h

Column	Place in the LCD_HAL.h
H	SymbolON
I	SymbolOFF

9. Fill the columns explained below on the Map Segments worksheet. Copy column D and paste it in the LCD_HAL.h at the Name Segment section.
10. Copy the columns shown in Table 9 from the Custom Glass worksheet and paste into the LCD_HAL.c.

Table 9. Custom glass worksheet to LCD_HAL.c

Column	Place in LCD_HAL.c
Q	aPlace

NOTE

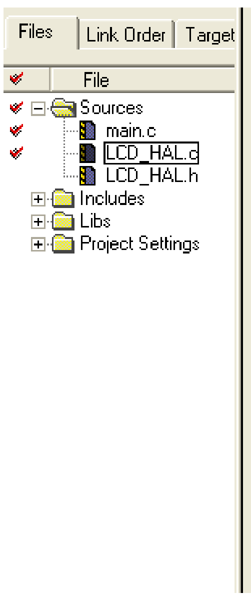
Before pasting, column Q must be sorted in alphabetical order.

Before Sorting

Q
aPlace
CHAR1A, // LCD4 --- Pin:5
CHAR1B, // LCD5 --- Pin:6
CHAR1C, // LCD6 --- Pin:7
CHAR1D, // LCD7 --- Pin:8
CHAR2A, // LCD8 --- Pin:9
CHAR2B, // LCD9 --- Pin:10
CHAR2C, // LCD10 --- Pin:11
CHAR2D, // LCD11 --- Pin:12
CHAR7A, // LCD12 --- Pin:13
CHAR7B, // LCD13 --- Pin:14
CHAR7C, // LCD33 --- Pin:15
CHAR7D, // LCD34 --- Pin:16
CHAR8A, // LCD41 --- Pin:17
CHAR8B, // LCD42 --- Pin:18
CHAR8C, // LCD43 --- Pin:19
CHAR8D, // LCD14 --- Pin:20
CHAR9A, // LCD15 --- Pin:21
CHAR9B, // LCD16 --- Pin:22
CHAR9C, // LCD28 --- Pin:23
CHAR9D, // LCD27 --- Pin:24

After Sorting

CHAR1A, // LCD4 --- Pin:5
CHAR1B, // LCD5 --- Pin:6
CHAR1C, // LCD6 --- Pin:7
CHAR1D, // LCD7 --- Pin:8
CHAR2A, // LCD8 --- Pin:9
CHAR2B, // LCD9 --- Pin:10
CHAR2C, // LCD10 --- Pin:11
CHAR2D, // LCD11 --- Pin:12
CHAR3A, // LCD29 --- Pin:44
CHAR3B, // LCD30 --- Pin:43
CHAR3C, // LCD37 --- Pin:42
CHAR3D, // LCD38 --- Pin:41
CHAR4A, // LCD39 --- Pin:40
CHAR4B, // LCD40 --- Pin:39
CHAR4C, // LCD31 --- Pin:38
CHAR4D, // LCD32 --- Pin:37
CHAR5A, // LCD17 --- Pin:36
CHAR5B, // LCD18 --- Pin:35
CHAR5C, // LCD19 --- Pin:34




```
/** LCD waveform for each alphanumeric */
const UINT8 aPlace [ ] =
{
     CHAR1A,    // LCD4 --- Pin:5
     CHAR1B,    // LCD5 --- Pin:6
     CHAR1C,    // LCD6 --- Pin:7
     CHAR1D,    // LCD7 --- Pin:8
     CHAR2A,    // LCD8 --- Pin:9
     CHAR2B,    // LCD9 --- Pin:10
     CHAR2C,    // LCD10 --- Pin:11
     CHAR2D,    // LCD11 --- Pin:12
     CHAR3A,    // LCD29 --- Pin:44
     CHAR3B,    // LCD30 --- Pin:43
     CHAR3C,    // LCD37 --- Pin:42
     CHAR3D,    // LCD38 --- Pin:41
     CHAR4A,    // LCD39 --- Pin:40
     CHAR4B,    // LCD40 --- Pin:39
     CHAR4C,    // LCD31 --- Pin:38
     CHAR4D,    // LCD32 --- Pin:37
     CHAR5A,    // LCD17 --- Pin:36
     CHAR5B,    // LCD18 --- Pin:35
     CHAR5C,    // LCD19 --- Pin:34
     CHAR5D,    // LCD20 --- Pin:33
     CHAR6A,    // LCD21 --- Pin:32
     CHAR6B,    // LCD35 --- Pin:31
     CHAR6C,    // LCD36 --- Pin:30
     CHAR6D,    // LCD22 --- Pin:29
```

Figure 6. Before sorting and after sorting

11. Create your own alphanumeric definition in the ASCII sheet columns. After defining the segments, copy the AF column and paste it in the baASCII array.

If the application needs to declare text, the arrays that do this must be between the red lines in the main file after including headers and libraries. This is to save data in a special memory segment for the MC9RS08 devices.When using the LL family the memory will not be saved in a special place.

```
#pragma CONST_SEG __FAR_SEG mySEG
const byte Mytext [ ] = {"HELLO WORLD"};
#pragma CONST_SEG DEFAULT
```

- For the MCF51EM256, 13 segments will configure with four lines (chara, charb, charc, and chard). These are defined in aBackup[] array
- For the MC9S08LG32, 13 segments will configure with four lines (chara, charb, charc, and chard).
- For the MC9S08LL16, 16 segments configure the two lines (chara and charb). These are defined in baBackup[] array.
- For the MC9RS08LA8 and MC9RS08LE4, 13 segments configure the two lines (chara and charb).

You can take the information from the ASCII array.

```
/** Backup mask array */
const UINT8 aBackup [ ] =
{
   _Backupd,
   _Backupc,
   _Backupb,
   _Backupa
};
```

7 Conclusion

This driver provides a software interface between the custom glass pinout and the LCD module in low-end Freescale microcontrollers. Minimal changes are required to customize an LCD application and fit specified hardware requirements. These routines are a reliable platform to migrate between the MC9RS08LE, MC9RS08LA, MC9S08LL, MC9S08LG32, MCF51EM256, K40 and K50 families and reduces development time when using Freescale products.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN3796
Rev. 4
10/2011