

# Interfacing an LCD to the MC9S08LC60

by: Steven Torres  
Austin, TX, USA

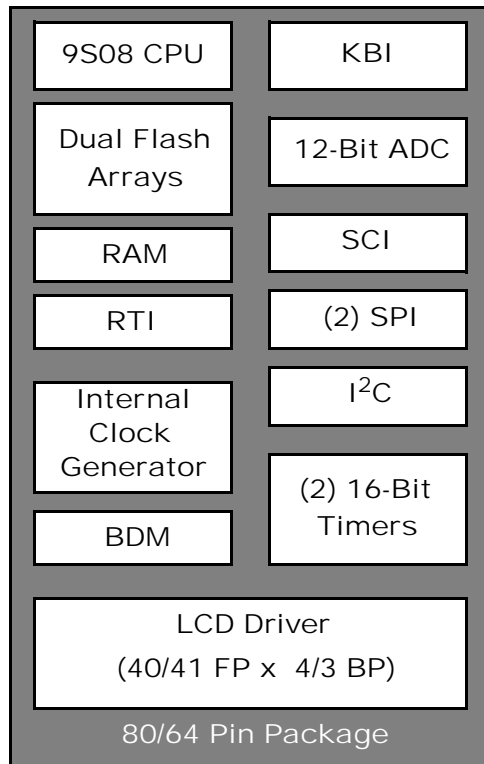
## 1 MC9S08LC60 Introduction

The MC9S08LC60 is an HCS08 device that combines a liquid crystal display (LCD) driver module with a rich set of peripherals. This high level of integration reduces the total system cost by providing a single chip solution for applications needing a display with up to 160 segments. LCDs are commonplace in many applications, including thermostats, calculators, digital multi-meters, medical monitoring devices, toys, and appliances.

The LCD module integration in the MC9S08LC60 provides a high-level of configurability allowing for lower power operation. In fact, the LCD driver can be configured to operate a display even when the MC9S08LC60 enters stop3 sleep mode. [Figure 1](#) provides a block diagram of the MC9S08LC60. In addition to the built-in LCD driver, the MC9S08LC60 has notable peripherals such as a 12-bit ADC and dual Flash arrays (for EEPROM emulation), as shown in the block diagram.

### Contents

1	MC9S08LC60 Introduction	1
1.1	Type of LCD Glass Compatible with MC9S08LC60	2
1.2	Number of Segments and Using Segments in LCD	3
1.3	LCD Glass Specification	4
2	LCD Module Overview	4
2.1	LCD Module Power Supply Configuration	4
2.2	LCD Module Clock	5
2.3	LCD Blinking	6
3	Interfacing the LCD MCU to an LCD Glass	7
3.1	LCD Glass Detailed Description	8
3.2	Mapping LCD Glass Segments to the MCU LCD RAM Registers	11
3.3	Mapping the MCU LCD RAM to Alphanumeric Segment Groups	12
4	Example Application Using the DEMO9S08LC60	16
4.1	DEMO9S08LC60 Overview and Configuration	17
4.2	Example Application Software Overview	18
4.3	Example Application Operation and Quick Start	25



**Figure 1. MC9S08LC60 Block Diagram**

This application note introduces the MC9S08LC60 and details a procedure for interfacing a LCD glass to the MC9S08LC60 from a hardware and software perspective. Using the DEMO9S08LC60 EVB hardware, the relationship between the LCD glass segments and the individual bits in the MC9S08LC60 LCDRAM registers is demonstrated. The demo validates that the LCD glass was properly interfaced to the MC9S08LC60.

Sample software to illustrate a basic application that uses a LCD is included in this document. The document contains both an overview of the demo software and a quickstart for the demo. All code for the demo is provided in the file AN3280SW1.zip, which can be found on the Freescale web page ([freescale.com](http://www.freescale.com)).

## 1.1 Type of LCD Glass Compatible with MC9S08LC60

Two common LCD types are TN (twisted nematics) and STN (super twisted nematics). TN-type LCD glass is very cost-effective and is compatible with multiplexing LCD drivers using fewer than 16 backplanes. The multiplexing LCD driver uses multiple backplanes to reduce the number of connections between the LCD and the LCD driver. Using multiplexing, each frontplane pin can be connected to as many segments as there are backplanes. With up to four backplanes and running in 1/3 bias mode, the MC9S08LC60 is compatible with TN-type LCD glass.

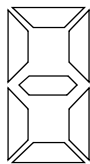
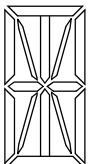
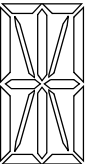
The multiplexing configuration relates to the LCD duty cycle. The duty cycle indicates the amount of time the LCD panel segment is energized during each LCD module frame cycle. The available duty cycle options for the MC9S08LC60 LCD module are 1/2, 1/3, and 1/4. The denominator of the duty cycle indicates the number of backplanes being used to drive a LCD panel.

This application note does not provide an introduction to LCD theory, types of LCD, LCD waveforms, or principles of operation. AN3219, *XGATE Library: TN/STN LCD Driver*, provides a detailed description of driving TN and STN LCDs using general-purpose input/output (GPIO) pins. Even though the MC9S08LC60 has a built-in LCD driver module, the discussions presented in AN3219 on multiplex waveforms and LCD theory are relevant.

## 1.2 Number of Segments and Using Segments in LCD

Depending on package and LCD configuration, the MC9S08LC60 can support up to 160 segments. The number of segments is equal to the number of backplanes times the number of frontplanes. With multiplex LCD drivers like the MC9S08LC60, each frontplane pin can be connected to as many segments as there are backplanes. With up to 160 segments, the MC9S08LC60 can be used in displays that are numeric, alpha-numeric, symbolic, or a combination of these. Numeric displays require seven segments for each digit; alpha-numeric displays require either 14 or 16 segments per character. [Table 1](#) lists the available configurations and the potential number of digits when using a 7-, 14-, or 16-segment configuration.

**Table 1. 9S08LC60 Segment Configuration Based on Package**

Package	Number of Backplanes	Number of Frontplanes	Total Number of Segments	Potential Number of Characters (Remainder)		
				7 segment 	14 segment 	16 segment 
64 pin	4	32	128	18 (2)	9 (2)	8 (0)
	3	33	99	14 (1)	7 (1)	6 (3)
80 pin	4	40	160	22 (6)	11 (6)	10 (0)
	3	41	123	17 (4)	8 (11)	7 (11)

### 1.3 LCD Glass Specification

When obtaining TN-type LCD glass for an application, several parameters should be considered. These include:

- LCD operating voltage — Typical values for LCD glass operating voltage are 3 V or 5 V. The operating voltage is the nominal voltage level threshold to power on a segment.
- Current consumption — The current consumption is related to the total segment area.
- View area — This is related to the size of the LCD glass.
- Driving mode — Duty cycle (1/2, 1/3, and 1/4) and bias (1/3)
- View angle — 3 o'clock, 6 o'clock, 9 o'clock, and 12 o'clock
- View mode — Reflective, transmissive, and transfective
- Operating temperature — Varies

## 2 LCD Module Overview

### 2.1 LCD Module Power Supply Configuration

The LCD module provides several configurations for its power supply. The LCD module can be powered via  $V_{DD}$  or, alternatively, by an external LCD power supply connected to the  $V_{LCD}$  pin (the  $V_{LCD}$  voltage range is 1.4 to 1.8 V). The LCD module power supply configuration is determined by the VSUPPLY[1:0] bit field. A block diagram of the power supply architecture is shown in Figure 2. The designer must choose from one of the power modes shown in Figure 2. This decision will be based on application requirements.

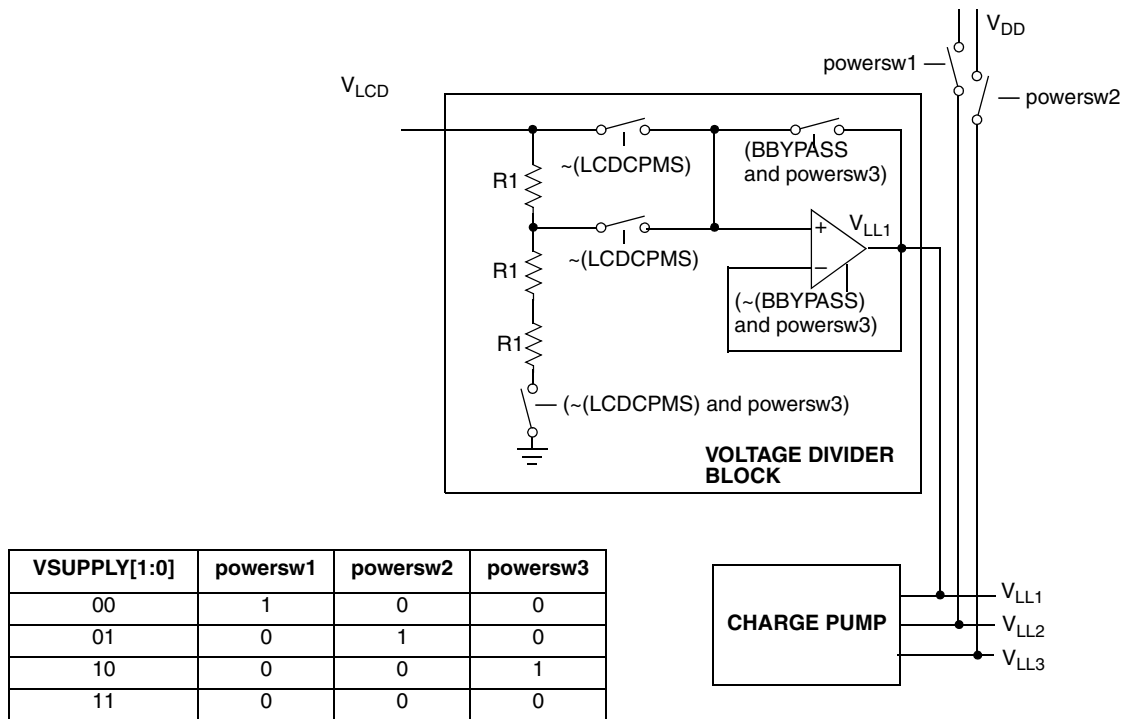


Figure 2. LCD Module Power Supply Configuration Options

Another factor affecting power supply configuration has to do with the LCD glass panel. Typical driving voltages for LCD glass panels are 3 V and 5 V. The subsequent sections summarize power supply configuration options when driving both 3 V and 5 V LCD glass.

### 2.1.1 Configuration Options for 3 V LCD Glass

When driving 3 V LCD glass, the following configurations are available:

- Power the LCD module via  $V_{LCD}$ , where  $V_{LCD}$  is nominally 1.5 volts, with the LCD module configured for doubler mode.
- Power the LCD module via  $V_{DD}$ , where  $V_{DD}$  is nominally 3 V, with the LCD module voltage supply switch (VSUPPLY[1:0]) configured to generate  $V_{LL3}$  from  $V_{DD}$
- Power the LCD module via  $V_{DD}$ , where  $V_{DD}$  is nominally 2 V, with the LCD module voltage supply switch (VSUPPLY[1:0]) configured to generate  $V_{LL2}$  from  $V_{DD}$

### 2.1.2 Configuration Options for 5 V LCD Glass

When driving 5 V LCD glass, the following configurations are available:

- Power LCD module via  $V_{LCD}$ , where  $V_{LCD}$  is nominally 1.67 volts, with the LCD module configured for tripler mode.
- Power the LCD module via  $V_{DD}$ , where  $V_{DD}$  is nominally 3.3 volts, with the LCD module voltage supply switch (VSUPPLY[1:0]) configured to generate  $V_{LL2}$  from  $V_{DD}$

## 2.2 LCD Module Clock

Figure 3 illustrates the MC9S08LC60 clock source, which can either be sourced from the MC9S08LC60 internal clock reference or an external clock source. The LCD module is designed to operate using a 32.768 kHz clock regardless of clock source, so clock dividers must be used to achieve the target 32.768 kHz clock target. In Figure 3, the 32.768 kHz clock target is labeled LCDCLK. LCDCLK is used to determine the following LCD operational settings:

- LCD base frequency
- LCD blink rate
- LCD charge pump rate

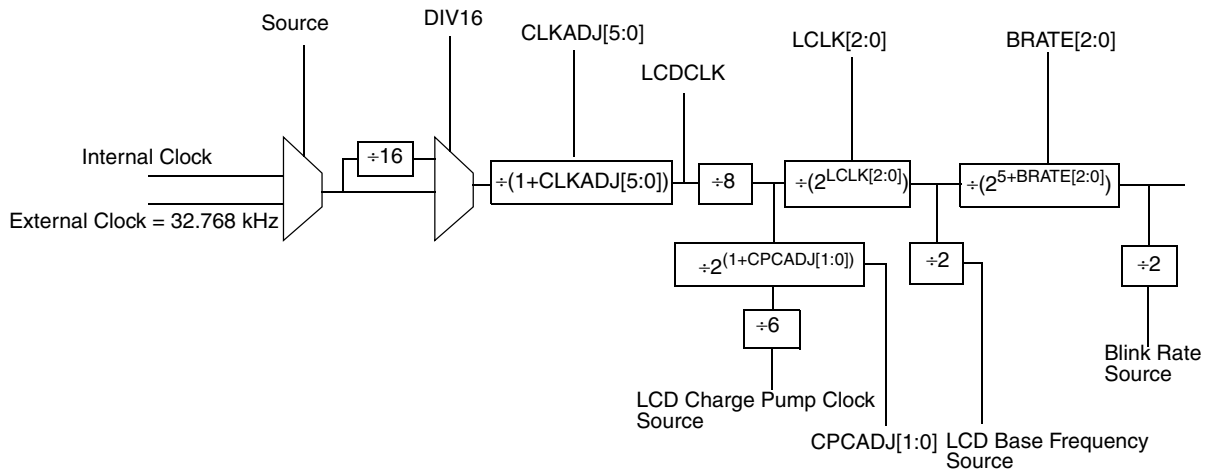


Figure 3. LCD Module Clock Tree

### 2.2.1 External Crystal and the LCD Frame Frequency

The lowest-power configuration for the MC9S08LC60 uses the external clock. It is also advantageous to select the slowest acceptable clock rate for the LCD base frequency, the LCD frame frequency, and the LCD charge pump rate that supports the LCD glass in the application. Reducing the clock rate reduces the current consumption of the LCD module.

### 2.3 LCD Blinking

One of the most outstanding features of the MC9S08LC60 is the flexible support for segment blinking. Some features of the segment blinking implementation are:

- Blink capability is programmable for the entire display or individual segments
- Multiple programmable options for the blink rate are available
- Blink functionality is available in stop3 mode

The LCDDRMS bit in the LCDCMD register is used in conjunction with the LCDRAM registers to provide blinking control for individual segments. If the LCDDRMS bit is cleared, the LCDRAM registers control the display on/off state for segments on the LCD display. If LCDDRMS bit is set, the LCDRAM registers control the blink enable on/off state for the corresponding segment. See [Section 4.2.3.2.1, “Non-alphanumerics,”](#) for example software functions that use the LCDDRMS bit to toggle the LCDRAM between its function to control the display on/off state and to control blink enable on/off state.

To configure all LCD segments to blink regardless of the contents of the LCDRAM registers while the LCDDRMS bit is set, the BLKMODE bit in the LCDBCTL control register must also be set.

### 3 Interfacing the LCD MCU to an LCD Glass

This application uses the DEMOLC60 EVB to detail the requirements for interfacing an LCD glass to the MC9S08LC60. Figure 4 shows the top layer silk screen of the DEMO9S08LC60 with the LCD glass panel. The LCD glass panel is shown with all LCD segments turned on. This section will review the LCD glass design in detail and discuss the interfacing of the LCD module to the LCD glass via hardware and software.

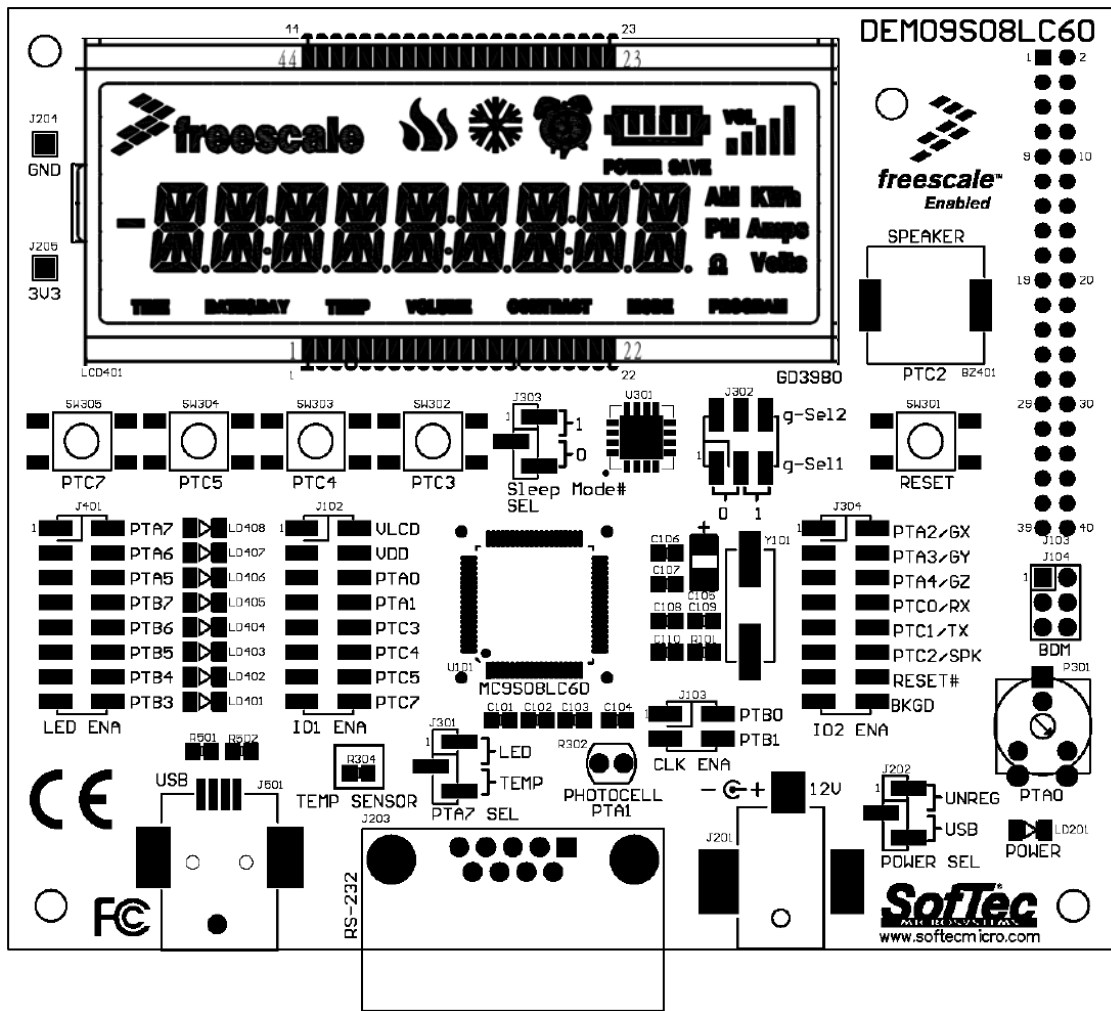


Figure 4. DEMO9S08LC60

### 3.1 LCD Glass Detailed Description

The LCD panel used in the DEMO9S08LC60 has a viewable area of approximately 75 mm × 25 mm and is shown in Figure 5. It is a custom 160-segment LCD TN-type glass that combines nine individual alphanumeric segment groups with numerous individual segments in the form of labels, punctuation, icons, etc. The alphanumeric segment groups are labeled 1 to 9 starting from the leftmost character position. The LCD panel is manufactured by S-Tek Displays (part number GD3980P).

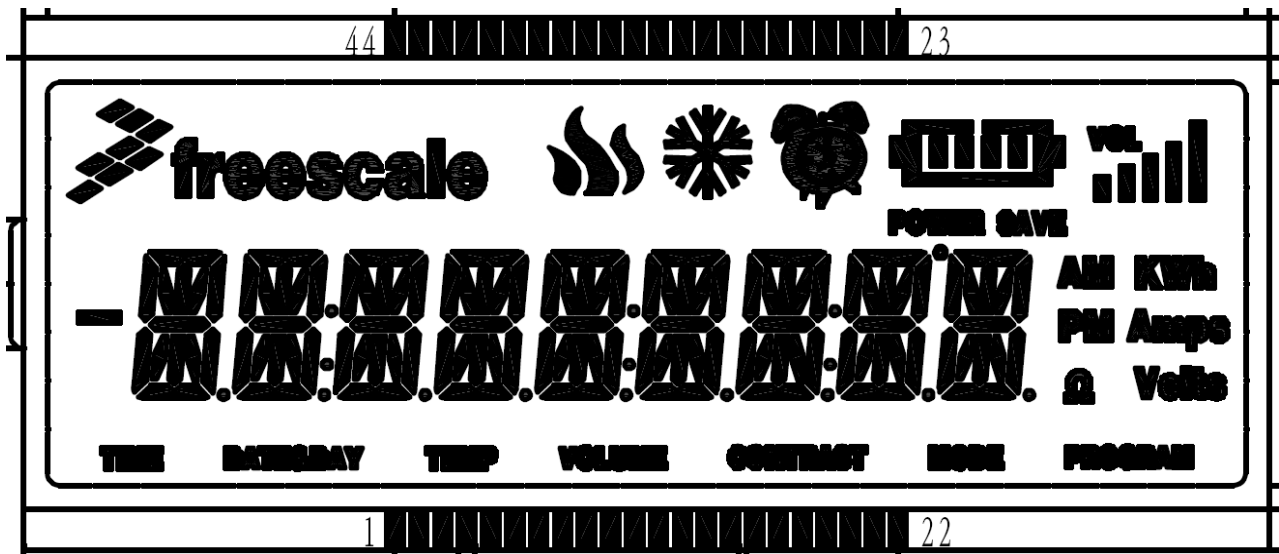


Figure 5. Custom LCD from S-Tek Displays

The LCD panel’s segment layout is not the only important aspect of its design. Other LCD glass specifications, such as those mentioned in Section 1.1, also dictate how the LCD glass must be driven by an LCD driver for optimal performance. Table 2 provides a summary of GD3980P parameters.

Table 2. GD3890P Parameters

GD3890P Parameters	Units	Values
Driving Voltage	Volts	3
Duty		1/4
Driving Frequency	Hz	64
Operating Temperature	C	0-50
Display Mode		Positive, Reflective
Viewing Angle	o'clock	6

Figure 6 shows all 160 segments labeled individually. This information is provided by the LCD manufacturer. See the appendix for the full GD3980P specification for S-Tek Displays.

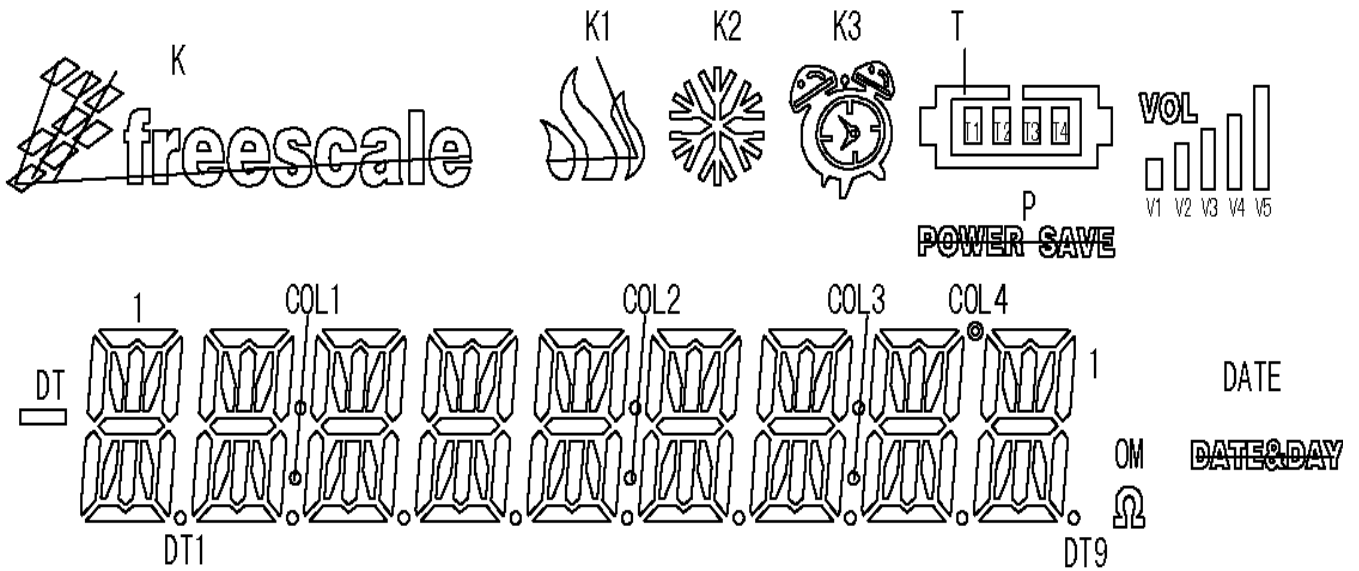


Figure 6. Segment Labels

Table 3 lists the LCD glass pin out along with each of the 160 segment labels. The LCD glass has a total of 44 pins, including four pins for backplane connections and 40 for frontplane connections.

**Table 3. LCD Pin-Out Specification**

PIN	COM1	COM2	COM3	COM4
1	COM1	—	—	—
2	—	COM2	—	—
3	—	—	COM3	—
4	—	—	—	COM4
5	1H	1F	1E	1N
6	1A	1J	1G	1M
7	1K	1L	1D	TIME
8	1B	1C	DT1	DT
9	2H	2F	2E	2N
10	2A	2J	2G	2M
11	2K	2L	2D	DATE
12	2B	2C	COL1	DT2
13	7H	7F	7E	7N
14	7A	7J	7G	7M
15	7K	7L	7D	P
16	7B	7C	COL3	DT7
17	8H	8F	8E	8N
18	8A	8J	8G	8M
19	8K	8L	8D	CONTRAST
20	8B	8C	DT8	MODE
21	9H	9F	9E	9N
22	9A	9J	9G	9M
23	9K	9L	9D	COL4
24	9B	9C	DT9	V1
25	V2	AM	PM	OM
26	KWh	Amps	Volts	Program
27	VOL	V3	V4	V5
28	T4	T3	T2	T1
29	6B	6C	DT6	T
30	6K	6L	6D	K3
31	6A	6J	6G	6M
32	6H	6F	6E	6N
33	5B	5C	COL2	DT5
34	5K	5L	5D	VOLUME
35	5L	5J	5G	5M
36	5H	5F	5E	5N
37	4B	4C	DT4	K2
38	4K	4L	4D	TEMP
39	4A	4J	4G	4M
40	4H	4F	4E	4N
41	3B	3C	DT3	K1
42	3K	3L	3D	K
43	3A	3J	3G	3M
44	3H	3F	3E	3N

### 3.2 Mapping LCD Glass Segments to the MCU LCD RAM Registers

After the relationship between the individual segments and the LCD pins is understood, the relationship between the LCD pins and the MCU pins and MCU LCD RAM registers needs to be established. The MCU and LCD pins are wired together, making sure that the respective backplane and frontplane pins are matched. [Figure 7](#) shows connections between the GP3980P and the MC9S08LC60 for the DEMO9S08LC60.

BP0	1	44	FP8
BP1	2	43	FP9
BP2	3	42	FP10
BP3	4	41	FP11
FP0	5	40	FP12
FP1	6	39	FP13
FP2	7	38	FP14
FP3	8	37	FP15
FP4	9	36	FP16
FP5	10	35	FP17
FP6	11	34	FP18
FP7	12	33	FP19
FP24	13	32	FP20
FP25	14	31	FP21
FP26	15	30	FP22
FP27	16	29	FP23
FP28	17	28	FP39
FP29	18	27	FP38
FP30	19	26	FP37
FP31	20	25	FP36
FP32	21	24	FP35
FP33	22	23	FP34

**Figure 7. MCU-to-LCD pin connections**

With the MCU pins connected to the LCD glass pins, a simple relationship between the LCD display segments and the MCU LCD RAM registers can be established. The mapping is determined by both the MCU-to-LCD pin connections ([Figure 7](#)) and the LCD pin out specification ([Table 3](#)). Each bit in the MCU RAM registers is mapped to an individual segment on the LCD glass.

[Table 4](#) shows the MCU LCD RAM registers, where the individual bits are labelled FPxBPy. [Figure 7](#) shows both the “BPy” and the “FPx” relationships between the MCU pinout and LCD panel pins. For example:

- BP3 is connected to the LCD pin 4. (It is important to note this is COM4.)
- FP15 is connected to LCD pin 37.

Using this information and Table 3 (for COM4 and LCD pin 37), one finds that FP15BP3 is connected to the label “K2.” Using Figure 6, the label “K2” is associated with the snowflake icon segment. Using this approach, all the relationships between the LCD display segments and the MCU LCD RAM registers can be established. Powering individual segments on and off is a matter of toggling the correct bit associated with the segment. For example, when the FP15BP3 bit is set to 1, the corresponding segment (the snowflake icon) will become visible on the LCD glass. When the FP15BP3 bit is set to 0, the segment is not displayed.

**Table 4. MCU LCD RAM**

0x1848	LCDRAM0	FP1BP3	FP1BP2	FP1BP1	FP1BP0	FP0BP3	FP0BP2	FP0BP1	FP0BP0
0x1849	LCDRAM1	FP3BP3	FP3BP2	FP3BP1	FP3BP0	FP2BP3	FP2BP2	FP2BP1	FP2BP0
0x184A	LCDRAM2	FP5BP3	FP5BP2	FP5BP1	FP5BP0	FP4BP3	FP4BP2	FP4BP1	FP4BP0
0x184B	LCDRAM3	FP7BP3	FP7BP2	FP7BP1	FP7BP0	FP6BP3	FP6BP2	FP6BP1	FP6BP0
0x184C	LCDRAM4	FP9BP3	FP9BP2	FP9BP1	FP9BP0	FP8BP3	FP8BP2	FP8BP1	FP8BP0
0x184D	LCDRAM5	FP11BP3	FP11BP2	FP11BP1	FP11BP0	FP10BP3	FP10BP2	FP10BP1	FP10BP0
0x184E	LCDRAM6	FP13BP3	FP13BP2	FP13BP1	FP13BP0	FP12BP3	FP12BP2	FP12BP1	FP12BP0
0x184F	LCDRAM7	FP15BP3	FP15BP2	FP15BP1	FP15BP0	FP14BP3	FP14BP2	FP14BP1	FP14BP0
0x1850	LCDRAM8	FP17BP3	FP17BP2	FP17BP1	FP17BP0	FP16BP3	FP16BP2	FP16BP1	FP16BP0
0x1851	LCDRAM9	FP19BP3	FP19BP2	FP19BP1	FP19BP0	FP18BP3	FP18BP2	FP18BP1	FP18BP0
0x1852	LCDRAM10	FP21BP3	FP21BP2	FP21BP1	FP21BP0	FP20BP3	FP20BP2	FP20BP1	FP20BP0
0x1853	LCDRAM11	FP23BP3	FP23BP2	FP23BP1	FP23BP0	FP22BP3	FP22BP2	FP22BP1	FP22BP0
0x1854	LCDRAM12	FP25BP3	FP25BP2	FP25BP1	FP25BP0	FP24BP3	FP24BP2	FP24BP1	FP24BP0
0x1855	LCDRAM13	FP27BP3	FP27BP2	FP27BP1	FP27BP0	FP26BP3	FP26BP2	FP26BP1	FP26BP0
0x1856	LCDRAM14	FP29BP3	FP29BP2	FP29BP1	FP29BP0	FP28BP3	FP28BP2	FP28BP1	FP28BP0
0x1857	LCDRAM15	FP31BP3	FP31BP2	FP31BP1	FP31BP0	FP30BP3	FP30BP2	FP30BP1	FP30BP0
0x1858	LCDRAM16	FP33BP3	FP33BP2	FP33BP1	FP33BP0	FP32BP3	FP32BP2	FP32BP1	FP32BP0
0x1859	LCDRAM17	FP35BP3	FP35BP2	FP35BP1	FP35BP0	FP34BP3	FP34BP2	FP34BP1	FP34BP0
0x185A	LCDRAM18	FP37BP3	FP37BP2	FP37BP1	FP37BP0	FP36BP3	FP36BP2	FP36BP1	FP36BP0
0x185B	LCDRAM19	FP39BP3	FP39BP2	FP39BP1	FP39BP0	FP38BP3	FP38BP2	FP38BP1	FP38BP0
0x185C	LCDRAM20	0	0	0	0	FP40BP3	FP40BP2	FP40BP1	FP40BP0

### 3.3 Mapping the MCU LCD RAM to Alphanumeric Segment Groups

Using the procedure above, turning individual segments on and off is very straightforward. For the alphanumeric displays (the DEMO9S08LC60 contains 13-segment displays), the procedure is basically the same except that individual segments are grouped and modified together depending on the alphanumeric character that is specified to be activated in the display.

### 3.3.1 GP3890P Alphanumeric Segment Groups

The alphanumeric segment group for the DEMO9S08LC60 is a modified version of a standard 14-segment display group. This custom segment group for alphanumeric displays in the GP3890P LCD glass, shown in [Figure 8](#), uses only 13 segments. The difference between this group and a standard 14-segment group is the single segment “G”, which normally consists of two segments of equal size. The GP3890P has a total of nine 13-segment displays. Referencing [Figure 6](#), the leftmost 13-segment display group is the first position. With a 13-segment display layout, each alphanumeric character will require 13 bits of MCU LCD RAM.

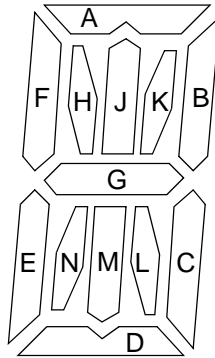


Figure 8. GP3890P Custom 13-Segment Display Layout Pattern

### 3.3.2 Alphanumeric Segment Group MCU LCD RAM Mapping

This section gives an example of the mapping of an alphanumeric segment group for the leftmost 13-segment display (position 1) on the GP3890P to display the letter “M”. [Figure 9](#) shows the layout of the 13-segment display, an excerpt from the LCD pinout specification table, and the pinout connections between the MCU and the LCD glass. [Table 5](#) shows the corresponding MCU LCD RAM registers. Note that two adjacent MCU LCD RAM registers, LCDRAM1 and LCDRAM0, are used for the 13-segment alphanumeric segment group.

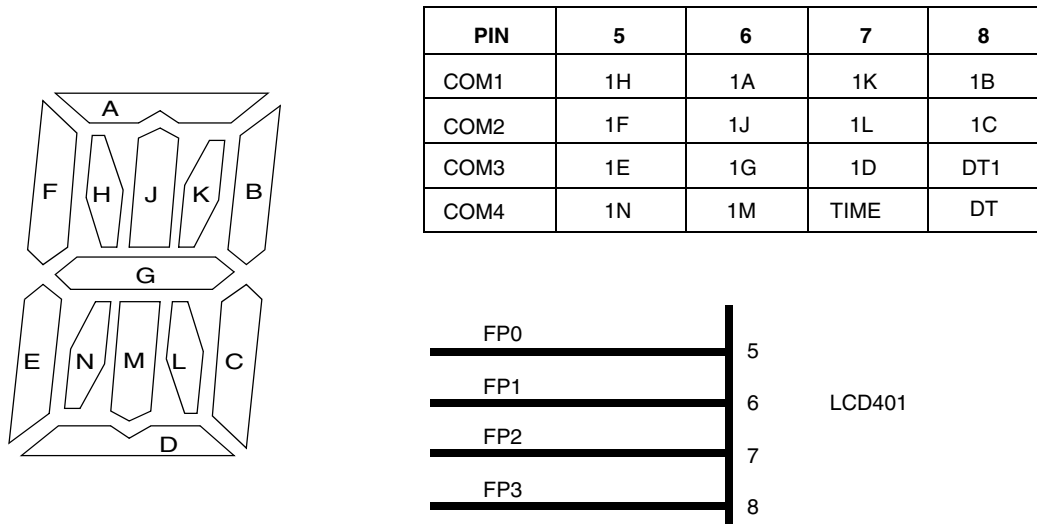


Figure 9. Example Mapping MCU LCD RAM to LCD Glass Alphanumeric Segment Groups

Table 5. LCD RAM Values for the Leftmost GD3890P Alphanumeric Character to Display an “M”

LCDRAM0	FP1BP3 “1M”	FP1BP2 “1G”	FP1BP1 “1J”	FP1BP0 “1A”	FP0BP3 “1N”	FP0BP2 “1E”	FP0BP1 “1F”	FP0BP0 “1H”
	1	0	1	1	0	1	1	0

LCDRAM1	FP3BP3 “DT”	FP3BP2 “DT1”	FP3BP1 “1C”	FP3BP0 “1B”	FP2BP3 “TIME”	FP2BP2 “1D”	FP2BP1 “1L”	FP2BP0 “1K”
	X	X	1	1	X	0	0	0

In Table 5, the MCU LCD RAM registers are shown with both the bit names and the corresponding labels from Figure 6. The register bits values in Table 5 are provided for the display of the alphanumeric character “M” in the LCD display. A value of “X” in the table is provided for the bit locations where the value is not associated with the 13-segment display. For these locations, the bit value is irrelevant. In this case (for alphanumeric characters in character position 1), these are the bits for the labels “TIME”, “DT”, and “DT1”. Table 6 provides a more detailed tabular form of the decoding from MCU pins to LCD segments.

**Table 6. Detailed Alphanumeric Segment Mapping**

MCU Pin Function	MCU Pin # in 80 Pin Package	MCU Pin # in 64 Pin Package	LCD Pin	LCD Segment / MCU LCD RAM Bit			
				COM1/BP0	COM2/BP1	COM3/BP2	COM4/BP3
FP0	7	6	5	1H/FP0BP0	1F/FP0BP1	1E/FP0BP2	1N/FP0BP3
FP1	6	5	6	1A/FP1BP0	1J/FP1BP1	1G/FP1BP2	1M/FP1BP3
FP2	5	4	7	1K/FP2BP0	1L/FP2BP1	1D/FP2BP2	TIME/FP2BP3
FP3	4	3	8	1B/FP3BP0	1C/FP3BP1	DT1/FP3BP2	DT/FP3BP3

The same mapping approach can be used for the remaining alphanumeric character positions. An important observation is that the other character positions follow the MCU LCD RAM mapping organization and layout pattern (see [Figure 8](#)) using two adjacent registers. The shaded bit locations in [Table 6](#) are used arbitrarily for all other segments not used in an alphanumeric group. [Table 7](#) lists the MCU LCD RAM registers used for alphanumeric groups 1–9 (LCDRAM[0:17], respectively). The LCDRAM[18:19] registers are not used for alphanumeric groups. LCDRAM20 is not used at all because FP40 is not used in 1/4 duty mode. In 1/4 duty mode, the multiplexed BP3/FP40 pin is configured as BP3.

**Table 7. MCU LCD RAM Registers for Alphanumeric Groups**

Alphanumeric Segment Group	MCU LCD RAM
1	LCDRAM0 LCDRAM1
2	LCDRAM2 LCDRAM3
3	LCDRAM4 LCDRAM5
4	LCDRAM6 LCDRAM7
5	LCDRAM8 LCDRAM9
6	LCDRAM10 LCDRAM11
7	LCDRAM12 LCDRAM13
8	LCDRAM14 LCDRAM15
9	LCDRAM16 LCDRAM17

## 4 Example Application Using the DEMO9S08LC60

This section provides a brief description of a basic software demo using the DEMO9S08LC60 demonstration board which uses the MC9S08LC60. Figure 10 shows the a photo of the DEMO9S08LC60 Example Application.



Figure 10. DEMO9S08LC60 Example Application

The demo captures alphanumeric data from SCI0 and then displays the data on the LCD display. Alphanumeric data can be transmitted to MC9S08LC60's SCI0 using a PC and a COM terminal program (such as Hyperterminal or Tera Term). Other features of the demo are:

- Demonstrates the usage of an MC9S08LC60 with a 32.768 kHz external crystal clock source
  - Demonstrates configuration of the ICG
- Demonstrates the reception of data via SCI0
  - Demonstrates configuration of the SCI0 baud rate
- Demonstrates the usage of the LCD modules
  - Demonstrates the LCD module power configuration to drive the GP3890P (3 volts LCD glass)
  - Demonstrates the LCD module clock configuration
  - Demonstrates the LCD module segment blinking
  - Demonstrates the LCD module driving of alphanumeric characters
- Demonstrates the reception of data via SCI0
- Demonstrates the use of GPIO
  - LEDs toggle when user presses buttons

All the software for the basic demo is provided as a companion to this application note and can be downloaded from the Freescale web site. The file name is AN3280SW1.zip. Use the software at your own risk. The software is provided “as-is,” with no warranties, guarantees, or support.

### NOTE

Out of box, the DEMO9S08LC60 comes preprogrammed with an LCD demo. This preprogrammed demo application is provided by SofTech Microsystems and it is operational when the DEMO9S08LC60 is powered. The SofTech Microsystems demo and the demo described in this application note, though having similar functions, have slightly different source code implementations. The SofTech Microsystems demo source code is provided on a CD in the DEMO9S08LC60 kit.

## 4.1 DEMO9S08LCD60 Overview and Configuration

The DEMO9S08LC60 (see [Figure 4](#)) is a full-featured customer evaluation PCB with a built-in USB-to-BDM programmer. Besides providing a programming interface, the USB cable can also be used to power the device.

The DEMO9S08LC60 has the following user interface elements:

- LEDs
- Buttons
- Potentiometer
- Serial port
- Photo cell
- Temp sensor
- Speaker
- 3-Axis accelerometer

For this demo, the DEMO9S08LC60 must be configured with the jumper positions given in [Table 8](#).

**Table 8. DEMO9S08LC60 Configuration for Example Application**

Jumper(s)	Setting
LED EN	all ON
IO1 ENA	all ON
IO2 ENA	all ON
PTA7 SEL	LED
CLK ENA	all ON
POWER SEL	USB
g-Sel1	don't care
g-Sel1	don't care
g-Sleep Mode	don't care

## 4.2 Example Application Software Overview

This section provides an overview of the configuration of the example application demo. The overview reviews settings for:

- ICG
- SCI
- LCD

These LCD drivers are a subset of the application. The application could be an energy meter or a toy.

The goal of this section is not to describe every function in detail. Instead, only select functions will be described. For example, a description will be provided for the implementation of a procedure for the mapping of the MCU LCD RAM to alphanumeric segment groups as described in the previous section.

### 4.2.1 ICG Configuration

The ICG has four possible configurations, including the following three modes:

- FLL bypassed, external clock (FBE) mode
- FLL engaged, internal clock (FEI) mode
- FLL engaged, external clock (FEE) mode

Both FBE and FEE use the on-board 32.768 kHz crystal on the DEMO9S08LC60. For these modes, the DEMO9S08LC60 CLK ENA jumper must be installed. For the lowest power mode, the FBE mode is recommended. If the use of the LCD in stop3 is desired, FBE mode is also recommended. If lower system/BOM (bill of materials) cost is the priority, the FEI mode should be used. Cpu.h defines variables to select the desired ICG clock mode.

While the demo software is pre-configured to use the ICG FBE mode, the demo project can be modified so that the user can also operate the demo in FEI and FEE ICG modes. When using either FEI and FEE ICG modes, the demo uses a faster SCI baud setting (9600 bps). The code excerpt below from Cpu.h shows the ICG options built-in to the project.

```
//=====
//Select a clock mode
//=====
//FBE = FLL ByPassed External Clock
#define FBE_32KHZ          1 /* 32 kHz crystal and 16 kHz fbus */
//FEE = FLL Enabled using External Clock
#define FEE_32K8BUS       0 /* 32 kHz crystal and 8.39 MHz fbus */
#define FEE_32K16BUS     0 /* 32 kHz crystal and 16.775 MHz fbus */
//FEI = FLL Enabled Internal Clock
#define FEI_8MBUS        0 /* fbus 17.77/2 MHz; untrimmed */
```

## 4.2.2 SCI Configuration

The SCI configuration requires:

- The enabling of the SCI receiver and transmitter
- The configuration of the SCI baud registers
- Configuration of other SCI parameters

The code excerpt below provides a typical scheme for SCI configuration.

```
SCIC1 = 0;
/* SCI1C3: R8=0,T8=0,TXDIR=0,??=0,ORIE=0,NEIE=0,FEIE=0,PEIE=0 */
SCIC3 = 0;
SCIBD = SCIDIVIDER; //Set in sci.h file
SCIC2_TE = 1;          /* Enable transmitter */
SCIC2_RE = 1;          /* Enable receiver */
```

Depending on the ICG configuration, the SCIDIVIDER variable may need to be modified to obtain a particular baud rate. The SCI is clocked by the bus clock. With a 32.768 kHz crystal and FBE mode, a 9600 baud rate cannot be achieved. (The value of SCIDIVIDER is approximately equal to the bus clock / 16 / desired baud rate). In the demo application, SCIDIVIDER can be configured in sci.h. In the case of a 32.768 kHz crystal in FBE mode, SCIDIVIDER is selected (in sci.h) such that the baud rate is 110 bps.

## 4.2.3 LCD Configuration

An LCD driver can be developed using a variety of approaches. LCD drivers provide the ability to initialize the LCD module and write to the display. Because of this, LCD driver should provide the following functionality:

- LCD initialization
  - Initialization of the LCD clock
  - Initialization of the LCD frame frequency and blink rate
  - Initialization of the LCD power supply
  - Setup of the waveform type
  - Setup of the charge pump frequency
  - Setup of the LCD buffer
  - Setup of the LCD buffer driver capability
  - Enabling of the frontplanes and the LCD
- LCD methods
  - Write to the display (for alphanumerics and non-alphanumerics)
  - Clear the display (for alphanumerics and non-alphanumerics)
  - Configure, start, and end blinking
- LCD events
  - LCD interrupt enable
  - LCD interrupt handler

LCD drivers to write to the display are simple to implement after the relationship of the LCD segments to the LCD RAM is determined. For this example application, the method to determine the LCD segment to LCD RAM relationship was described in [Section 3](#). [Section 3](#) also discussed an approach to display alphanumeric characters. This approach will be implemented in software.

### 4.2.3.1 LCD Initialization Drivers LCD\_init()

In the example application, the initialization is accomplished via the LCD\_init() function. The source code for LCD\_init() is provided below. LCD\_init() calls functions that configure the power, clocks, and other parameters for the LCD. Many of these parameters are compile time decisions. Each of the functions called by LCD\_init() can be found in lccdrv.c.

```
void LCD_init() {  
  
    //Configure clock source  
    CONFIG_CLKSOURCE();  
  
    //power supply configuration  
    SET_CONFIG_VSUPPLY();  
  
    //Configure operation in stop/wait  
    SET_LCDCR1_REG();  
  
    //Configure frame frequency  
    SET_LCD_FRAME_FREQU();  
  
    //Configure blink rate  
    CONFIG_BLINKING( 2 /* Hz*/, OFF);  
  
    //Enable Frontplanes  
    ENABLE_FP();  
  
    //Enable LCD  
    ENABLE_LCD(ON);  
  
    //Map RAM to ARRAY  
    MAP_LCDDRIVER_TO_LCDGLASS();  
}
```

The example application uses lccdrv.h to set up the LCD module power, clock, and other initialization options using several define directives. Some of these directive-defined variables are also tied to variables defined in cpu.h because LCD clock configuration parameters are dependent on the ICG configuration. An excerpt of variables defined in lccdrv.h is provided below, and shows the options for LCD power configuration and LCD duty cycle.

```
//=====  
//LCD power configuration  
//=====  
#define LCDPWR_VDD 1  
#define LCDPWR_VLCD 0  
#define LCDPWR_VLLLN 0  
.  
.  
.
```

```
//=====
//Define LCD duty cycle
//=====
#define DUTY1BY2      0
#define DUTY1BY3      0
#define DUTY1BY4      1
```

Using the directive-defined variables in `lccdrv.h`, `cpu.h`, and `sci.h`, the demo and the LCD module configuration can be changed quickly.

### 4.2.3.2 LCD Methods

The functions listed in this section manage the displaying and clearing of segments in the LCD display during the execution of the program. The functions manage either non-alphanumeric (icons, labels, etc.) or alphanumeric (13-segment display group) segments.

#### 4.2.3.2.1 Non-alphanumerics

The functions available to handle non-alphanumerics are provided below. In this implementation, the non-alphanumerics were grouped into functional categories. The selection of the grouping is not important from an application standpoint.

For the non-alphanumerics, these functions also allow the setting of the blink mode for those individual segments. The `lcdramm` variable tells the function if the LCDRAM[20:0] bits control the display on/off function or the blink enable on/off function. When calling these functions with `lcdramm` set to `SEGONOFF`, the LCDRAM[20:0] bits control the display on/off function; when calling these functions with `lcdramm` set to `SEGBLINKEN`, the LCDRAM[20:0] bits control the blink enable on/off function.

```
void SET_LOGO(char lcdramm, char k);
void SET_VOL(char lcdramm, char v, byte numbars);
void SET_BATT(char lcdramm, char t, byte numbars);
void SET_POWERSAVE(char lcdramm, char p);
void SET_TIMESEGS(byte lcdramm, byte am, byte pm, byte k3);
void SET_TEMPSEGS(byte lcdramm, byte k1, byte k2, char col4);
void SET_DMMLABLES(char lcdramm, char kwatt, char volt, char amp, char om) ;
void SET_COLONS(char lcdramm, char col1, char col2, char col3, char dt) ;
void SET_DECIMALS(char lcdramm, char dt1, char dt2, char dt3, char dt4, char dt5, char
dt6, char dt7, char dt8, char dt9);
void SET_MENU(char lcdramm, char lab1, char lab2, char lab3, char lab4, char lab5, char
lab6, char lab7);
```

The functional arguments after the `lcdramm` argument point to individual bits in the LCD RAM. In most cases, the variable name shown matches the LCD panel label for the segments. For example, “k” in `SET_LOGO(char lcdramm, char k)` points to the Freescale logo segment. To control the on/off state, an ON and an OFF variable are provided. An example of the typical usage of these functions is provided below:

```
//Configure the display of am label, pm label and clock icon
SET_TIMESEGS(SEGONOFF, OFF,ON,ON); //am label is OFF while pm label and clock icon are ON
//Configure the blink enable of am label, pm label and clock icon
SET_TIMESEGS(SEGBLINKEN, OFF,ON,ON); //pm label & clock icon are configured to blink
```

### 4.2.3.2.2 Alphanumerics

The example application provides the `output_strg_lcd(tU08 *string, tU08 length)` function to display alphanumeric characters to the LCD display. The complete source code for `output_strg_lcd()` can be found in `lcddrv.c`.

The `output_strg_lcd()` function writes the contents of the string argument to the LCD display. It starts by writing the rightmost character of the string argument to the rightmost alphanumeric position on the LCD glass (i.e., the ninth position; see [Figure 6](#)). The `output_strg_lcd()` function loops through the string argument variable characters and indexes to the next LCD glass character position to complete the display.

To display the alphanumeric characters of the string argument, the software must determine the correct segments of the 13-segment display to power on while leaving the others off. The software uses a lookup table to determine the correct segments. The excerpt of the `output_strg_lcd()` source code shown below illustrates the usage of the lookup table and the manipulation of the proper LCDRAM registers to display the correct alphanumeric. When a new value is to be written to the LCDRAM registers, it must be ORed with the previous value. This is necessary because there are other segments in the LCDRAM register that are not part of the 13-segment display and their values must be preserved (see [Figure 9](#)).

```
c = ((string[i]-0x20)*2);    // convert ASCII to table index (in bytes)
if (c < 128) {
    *LCD_pos[LCD_position*2-2] = *LCD_pos[LCD_position*2-2] | (tU08)ascii[c];
    *LCD_pos[LCD_position*2-1] = *LCD_pos[LCD_position*2-1] | (tU08)ascii[c+1];
}
```

The look-up table is contained in the `ascii[]` array, which is shown below. This array determines what values need to be written to the MCU LCD RAM registers for a given alphanumeric value. The values of the look-up table were determined using the procedure discussed in [Section 3.3.2](#). An Excel worksheet is provided in `AN3280SW1.zip` that helps determine the values for the `ascii[]` array.

```
const tU08 ascii[] = {
    /* */ 0x00,0x00,
    /*!*/ 0x00,0x00,
    /*"*/ 0x00,0x00,
    /*#*/ 0x00,0x00,
    /*$*/ 0x00,0x00,
    /*%*/ 0x00,0x00,
    /*&*/ 0x00,0x00,
    /*'*/ 0x0,0x00,
    /*(*/ 0x00,0x00,
    /*)*/ 0x00,0x00,
    /***/ 0xe9,0x3,
    /*+*/ 0xe0,0x0,
    /*,/ 0x00,0x00,
    /*-*/ 0x40,0x0,
    /*./ 0x00,0x00,
    /* */ 0x00,0x00,
    /*0*/ 0x16,0x34,
    /*1*/ 0xA0,0x0,
    /*2*/ 0x54,0x14,
    /*3*/ 0x50,0x34,
    /*4*/ 0x42,0x30,
    /*5*/ 0x52,0x24,
    /*6*/ 0x56,0x24,
```

```

/*7*/ 0x18,0x01,
/*8*/ 0x56,0x34,
/*9*/ 0x52,0x34,
/*:*/ 0x10,0x80,
/*;*/ 0x00,0x00,
/*<*/ 0x00,0x00,
/*=*/ 0x00,0x00,
/*>*/ 0x00,0x00,
/*?*/ 0x00,0x00,
/*@*/ 0x00,0x00,
/*A*/ 0x56,0x30,
/*B*/ 0x56,0x34,
/*C*/ 0x16,0x04,
/*D*/ 0x16,0x34,
/*E*/ 0x56,0x4,
/*F*/ 0x56,0x0,
/*G*/ 0x16,0x24,
/*H*/ 0x46,0x30,
/*I*/ 0xa0,0x0,
/*J*/ 0x0,0x34,
/*K*/ 0xA0,0x3,
/*L*/ 0x6,0x4,
/*M*/ 0x7,0x31,
/*N*/ 0x7,0x32,
/*O*/ 0x16,0x34,
/*P*/ 0x56,0x10,
/*Q*/ 0x16,0x36,
/*R*/ 0x56,0x12,
/*S*/ 0x52,0x24,
/*T*/ 0xb0,0x0,
/*U*/ 0x6,0x34,
/*V*/ 0xe,0x1,
/*W*/ 0xa6,0x34,
/*X*/ 0x9,0x3,
/*Y*/ 0x81,0x1,
/*Z*/ 0x18,0x5,
/*[ */ 0x16,0x4,
/*\ */ 0x1,0x2,
/*] */ 0x10,0x34,
/*^ */ 0x3,0x00,
/*_ */ 0x00,0x4,
};

```

#### 4.2.3.2.3 Scrolling a String Across the LCD Display — Scroll\_String()

The example application provides the Scroll\_String(tU08 \*string, byte local\_length) function to scroll an alphanumeric character string across the LCD display. The complete source code for Scroll\_String() can be found in lcddrv.c. The Scroll\_String() function accepts a string argument of length n, manipulates it, and calls output\_strg\_lcd(). Table 9 shows a specific example of how the string is manipulated by the Scroll\_String() function for a 5 character LCD display. In this example, the string argument provided to the Scroll\_String() function is “HELLO.” A call to the CLEAR\_ALL\_ALPNUM() function, a delay, and a call to the output\_strg\_lcd() function are executed between each of the sequences.

**Table 9. String Argument Manipulation by the Scroll\_String Function**

String Argument Manipulation Sequence
H
HE
HEL
HELL
HELLO
ELLO
LLO
LO
O

### 4.2.3.3 LCD Events / Interrupts

For the demo application, the LCD interrupts are initialized in lcddrv.h and the interrupt handler is found in vectors.c. The LCD interrupt handler toggles the GPIO port (PTC2), configured as an output, which is connected to the DEMO9S08LC60 speaker.

### 4.2.4 main()

The main() program function for this application is very simple, providing the following functionality:

- Initialize ports, ICG, SCI, LCD
- Initialize interrupts
- Display the default message
- Provide application loop

Below is the source code for the program loop in main():

```
//Program LOOP
while (1) {
    //Check incoming data
    result = AS1_RecvChar(&c);
    if (result == ERR_OK ) {
        sci_input[inputcounter] = c;//toupper(c);
        inputcounter = inputcounter + 1;
        if (c == '\r') {
            sci_input[inputcounter-1] = '\0';
            string = sci_input;
            Scroll_String(string, strlen(string));
            inputcounter = 0;
            cmdstring[0] = ' ';
            for (i=0; i <= 30; i++) sci_input[i] = 0;
        }
    }
}
```

```

//Change BAT and VOL randomly
SET_VOL(SEGONOFF,ON, 5-1);
SET_BATT(SEGONOFF,ON, 1);
if (l == 4) l=0;
else l++;;

//Display default message
DefaultDisplayMessage();
}

}
}

```

The loop polls the SCI0 data register and, when validate is available, stores the SCI0 data into the array `sci_input[inputcounter]`. The `inputcounter` variable counts the total number of SCI0 characters stored in `sci_input`. When the SCI0 data received is equal to a carriage return, ‘r’, the `Scroll_String()` function is called with `sci_input` as an argument. After the scrolling of the data on the LCD display, the `sci_input` array is cleared and a default LCD message is displayed. Afterwards, the program loop continues polling the SCI0 data register for the next user input.

### 4.3 Example Application Operation and Quick Start

The demo uses the FBE ICG configuration so it used the DEMO9S08LC60 32.768 kHz crystal clock source. The code in AN3280SW1.zip is pre-configured to drive the LCD panel at 3 Volts with a 1/4 duty cycle and a frame frequency of 64 Hz. The following instructions provide a walkthrough of the demo operation.

1. Unpack the DEMO9S08LC60 and place jumpers as described in [Table 8](#).
2. Connect a serial cable from a PC to the DEMO9S08LC60.
3. Connect an USB cable from a PC to the DEMO9S08LC60.
4. Unzip the AN3280SW1.zip file
5. Start Code Warrior and open the project file for the demo found in the AN3280SW1.zip file
  - Project file is named AN3280\_CWPRJ.mcp
  - Ensure the Code Warrior version supports the MC9S08LC60. For Code Warrior 5.1, an MC9S08LC60 patch is available from Freescale.com. Alternatively, Code Warrior 5.x can be used with an MC9S08LC60 driver provided by SofTech Microsystems. The SofTech Microsystems is provided on a CD in the DEMO9S08LC60 kit.
6. With the AN3280\_CWPRJ.mcp open, select SofTech as the programming target and then start the program download. This will start the Code Warrior debugger, Hiwave. The Code Warrior IDE icon that executes the command to start the debugger is the following:



7. At this point, the Hiwave debugger will erase and reprogram the DEMO9S08LC60 — several dialog boxes will show the reprogramming progress.

## Example Application Using the DEMO9S08LC60

8. After the reprogramming is complete, press Hiwave debugger the run icon to start the demo. The Hiware icon that executes the command to run the program is the following:



9. LCD will display a “HELLO WORLD” message.
10. After the “HELLO WORLD” message, the default LCD text message – “9S08LC60” – is displayed.
11. Start a terminal program on the PC and configure the baud rate to 110 bps. (Note AN3280SW1.zip includes a pre-configured HyperTerminal session file, AN3280\_Term.ht, that can be used for this part of the demo)
12. Type an alphanumeric character sequence in the terminal program and then press the Enter key.
  - a) The program will not change the display until the Enter key is pressed.
  - b) You can press the Enter key without entering alphanumeric.
  - c) Every time the Enter key is pressed, the LCD battery and volume indicators change.
13. After the Enter key is pressed, the new data will scroll across the LCD display. As the message scrolls, the program will not accept any additional input from the terminal program.
14. When the display is complete, the default LCD text message will be displayed.



THIS PAGE IS INTENTIONALLY BLANK

## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **E-mail:**

[support@freescale.com](mailto:support@freescale.com)

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
+1-800-521-6274 or +1-480-768-2130  
[support@freescale.com](mailto:support@freescale.com)

### **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)  
[support@freescale.com](mailto:support@freescale.com)

### **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](mailto:support.japan@freescale.com)

### **Asia/Pacific:**

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
[support.asia@freescale.com](mailto: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 303-675-2140  
Fax: 303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2006. All rights reserved.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Document Number: AN3280  
Rev. 0  
07/2006

