AN_651X_017 JULY 2005

Liquid Crystal Display (LCD) Interface

Teridian Semiconductor's 71M651x has a variety of interfacing ports for data access and display of information. This application note was developed to provide details on interfacing LCD displays to the 71M651x chips.

This application note applies to 71M651x Demo Boards with Varitronix VIM-808 Liquid crystal displays, but the general rules apply to all static and multiplexed LCDs. Details on LCD display routines used in the demo board firmware are also provided.

LCD Basics

LCDs use voltage-sensitive organic molecules with a helical structure that either block or permit the passage of polarized light. Areas filled with molecules that form parts of the display are called segments or pixels. For proper function, alternating current (AC) has to be applied to the segments.
LCDs with only a few segments can be operated in static mode, i.e. each segment has its own wire or pin that is connected to an AC voltage source (driver). In order to keep the number of connections low, LCDs with medium or large density are usually operated in multiplexed mode, i.e. individual segments share pins with others, and the display is driven by selecting one group of segments for a brief period of time and then moving on to the next group. The inertia of the organic molecules in a segment keeps the segment "ON" while the driver is accessing another group of segments. Depending on their operation mode, LCDs are usually categorized as:

- **Static Drive**: LCD Glass or LCD Modules with a simple segment displays are the only parts that have an option of Static Drive. The Static Drive configuration means that there is an individual control line to select each LCD segment and there is only a single common line that connects to them all. This configuration produces the best display with the widest temperature range, but it requires more interconnections that a multiplexed display would require.
- **Multiplexed Drive**: The Multiplexed Drive configuration means that each control line selects several LCD segments and that the final selection is made by selecting the correct common signal that also connects to several LCD segments. This configuration uses fewer interconnections which is cost effective for smaller displays. This configuration degrades the temperature and image performance slightly.

71M651X Hardware Supporting LCDs

The 71M651X is capable of driving LCDs directly in static or multiplexed mode. No driver or other interface circuitry is necessary.

With its maximum of 32 pins useable to drive segments, the 71M6511 device is capable of driving 4x32 = 128 pixels at 25% duty cycle. At eight segments per digit, the LCD can be designed for a maximum of 16 digits. In the minimum configuration, i.e. using only the dedicated segment pins SEG0…SEG2 and SEG8…SEG19, 15 x 4 = 60 segments can be driven, resulting in seven digits with eight segments each.

With its maximum of 42 pins useable to drive segments, the 71M6513 device is capable of driving 4x42 = 168 pixels at 25% duty cycle. At eight segments per digit, the LCD can be designed for a maximum of 21 digits. In the minimum configuration, i.e. using only the dedicated segment pins SEG0…SEG2 and SEG8…SEG13, 19 x 4 = 76 segments can be driven, resulting in nine digits with eight segments each.

Table 1 shows the assignment of DIO pins and SEG pins for the 71M6511 and 71M6513 chips.

DIO Pin	Segment Pin	Alternate Function	Availability	
			6511	6513
DIO21	SEG41			X
DIO20	SEG40			X
DIO19	SEG39			X
DIO18	SEG38			X
DIO17	SEG37		X	X
DIO16	SEG36		X	X
DIO15	SEG35		X	X
DIO14	SEG34		X	X
DIO13	SEG33			X
DIO12	SEG32			X
DIO11	SEG31		X	X
DIO10	SEG30		X	X
DIO9	SEG29		X	X
DIO8	SEG28		X	X
DIO7	SEG27	VARPULSE	X	X
DIO6	SEG26	WPULSE	X	X
DIO5	SEG25	SDATA	X	X
DIO4	SEG24	SCLK	X	X
	SEG23			X
	SEG22			X
	SEG21			X
	SEG20			X
	SEG19		X	X
	SEG18		X	X
	SEG17		X	X
	SEG16		X	X
	SEG15		X	X
	SEG14		X	X
	SEG13		X	X
	SEG12		X	X
	SEG11		X	X
	SEG10		X	X
	SEG9		X	X
	SEG8		X	X
	SEG7	MUX_SYNC	X	X
	SEG6	SRDY	X	X
	SEG5	SFR	X	X
	SEG4	SSDATA	X	X
	SEG3	SCLK	X	X
	SEG2	TEST2	X	X
	SEG1	TEST1	X	X
	SEG0	TEST0	X	X
DIO3				X
DIO2				X
DIO1				X
DIO0				X

Table 1: DIO/SEG Map

The LCD driver circuitry is grouped into 4 common outputs (COM0 to COM3) and up to 42 segment outputs. The COM0 through COM3 outputs are more than logical outputs, since they generate wave forms that, in conjunction with the wave forms generated by the SEG outputs, combine to true AC signals supplied to each individual segment. This means that the COM voltage is below the SEG voltage for some periods of time and above the SEG voltage for other periods of time.

The voltage present at the VLCD input is used internally by the chip to generate the LCD drive signals. A charge pump suitable for driving VLCD is included on-chip. This circuit creates 5VDC from the 3.3VDC supply using an external capacitor and diode network connected to the VDRV output (see Figure 1). This feature can be enabled with the *LCD_BSTEN* register.

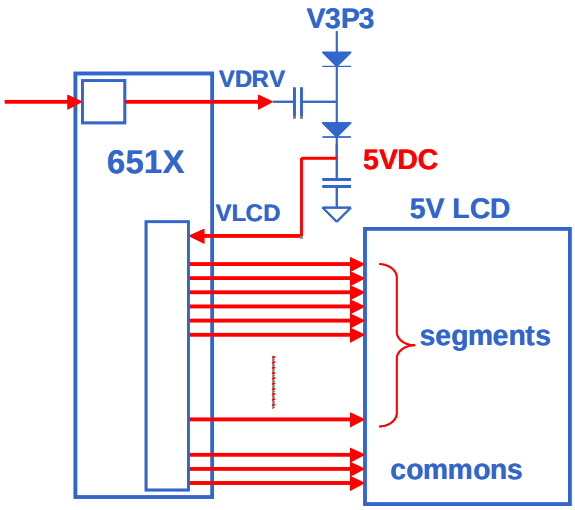

Figure 1: LCD Boost Circuit

71M651X Segment and LCD Control Registers

LCD operation is configured and controlled by registers in the configuration memory (I/O RAM). Configuration memory starts at physical address 0x2000, and access to all registers is possible using the serial interface (>**RI** command).

MPU firmware can easily reconfigure the LCD and change displayed information by writing to the LCD control registers.

LCD Mode

When configuring the LCD, the first parameter that has to be defined is the LCD operation mode, using the *LCD_MODE* register. The operation mode depends on the type of the LCD that is used with the 71M651X chip. For the 651X Demo Boards, the operation mode is 001, or 3-states, 1/3 bias. A maximum of four states to a minimum of one state (for static display) can be configured.

Register Name	Address [bits]	R/W	Description
LCD_MODE[2:0]	2021[4:2]	R/W	The LCD bias mode. 000: 4 states, 1/3 bias 001: 3 states, 1/3 bias 010: 2 states, ½ bias 011: 3 states, ½ bias 100: static display

LCD Operating Voltage

The type of LCD used for a particular implementation will determine whether 3.3VDC or 5VDC will be used for driving the LCD. As explained in the previous section, the 71M651X provides a charge pump capable of boosting the 3.3VDC supply voltage up to 5.0VDC. The boost circuit is enabled with the *LCD_BSTEN* register. The 651X Demo Boards have the boost circuit enabled by default.

Register Name	Address [bits]	R/W	Description
LCD_BSTEN	2020[7]	R/W	Enables the LCD voltage boost circuit.

LCD Clock

The next parameter to be configured is the LCD clock using the *LCD_CLK* register. This is the frequency at which the COM pins change states. A slower clock means lower power consumption, but if the clock is too slow, visible flicker can occur. The default clock frequency for the 71M651X Demo Boards is 150Hz (*LCD_CLK* = 01).

Register Name	Address [bits]	R/W	Description
LCD_CLK[1:0]	2021[1:0]	R/W	Sets the LCD clock frequency, i.e. the frequency at which SEG and COM pins change states. Note: f_w = CKADC/128 = 38,400 00: $f_w/2^9$, 01: $f_w/2^8$, 10: $f_w/2^7$, 11: $f_w/2^6$

LCD Control Registers

The display outputs are enabled by setting the *LCD_EN* register to 1.

Register Name	Address [bits]	R/W	Description
LCD_EN	2021[5]	R/W	Enables the LCD display. When disabled, VLC2, VLC1, and VLC0 are ground as are the COM and SEG outputs.

Depending on the hardware configuration surrounding the 71M651X chip, more or less dual-purpose I/O pins are used for display. By writing a value of 1 through 18 to the LCD_NUM register, the number of segment pins used for display is determined.

Register Name	Address [bits]	R/W	Description
LCD_NUM[4:0]	2020[4:0]	R/W	Number of dual-purpose LCD/DIO pins to be configured as LCD outputs. This will be a number between 0 and 18. The first dual-purpose pin to be allocated as LCD is SEG41/DIO21. Thus, if *LCD_NUM*=2, SEG41 and SEG 40 will be configured as LCD. The remaining SEG39 to SEG24 will be configured as DIO19 to DIO4.

The first dual-purpose pin to be allocated as LCD is SEG37/DIO17. Tables 2 (71M6511) and 3 (71M6513) below list the achievable combination of SEG and DIO pins as a function of the *LCD_NUM* value.

	71M6511	
LCD_NUM	**LCD Segments Available**	**DIO Pins Available**
1-4	None	DIO4-11, DIO14-17
5	SEG37	DIO4-11, DIO14-16
6	SEG36-37	DIO4-11, DIO14-15
7	SEG35-37	DIO4-11, DIO14
8-10	SEG34-37	DIO4-11
11	SEG34-37, SEG31	DIO4-10
12	SEG34-37, SEG30-31	DIO4-9
13	SEG34-37, SEG29-31	DIO4-8
14	SEG34-37, SEG28-31	DIO4-7
15	SEG34-37, SEG27-31	DIO4-6
16	SEG34-37, SEG26-31	DIO4-5
17	SEG34-37, SEG25-31	DIO4
18	SEG34-37, SEG24-31	None

Table 2: LCD_NUM Controlling SEG and DIO (6511)

	6513	
LCD_NUM	**LCD Segments Available**	**DIO Pins Available**
0	SEG0-23	DIO0-3, DIO4-21
1	SEG0-23, SEG41	DIO0-3, DIO4-20
2	SEG0-23, SEG40-41	DIO0-3, DIO4-19
3	SEG0-23, SEG39-41	DIO0-3, DIO4-18
4	SEG0-23, SEG38-41	DIO0-3, DIO4-17
5	SEG0-23, SEG37-41	DIO0-3, DIO4-16
6	SEG0-23, SEG36-41	DIO0-3, DIO4-15
7	SEG0-23, SEG35-41	DIO0-3, DIO4-14
8	SEG0-23, SEG34-41	DIO0-3, DIO4-13
9	SEG0-23, SEG33-41	DIO0-3, DIO4-12
10	SEG0-23, SEG32-41	DIO0-3, DIO4-11
11	SEG0-23, SEG31-41	DIO0-3, DIO4-10
12	SEG0-23, SEG30-41	DIO0-3, DIO4-9
13	SEG0-23, SEG29-41	DIO0-3, DIO4-8
14	SEG0-23, SEG28-41	DIO0-3, DIO4-7
15	SEG0-23, SEG27-41	DIO0-3, DIO4-6
16	SEG0-23, SEG26-41	DIO0-3, DIO4-5
17	SEG0-23, SEG25-41	DIO0-3, DIO4
18	SEG0-41	DIO0-3

Table 3: LCD_NUM Controlling SEG and DIO (6513)

The contrast of the LCD can be controlled with the *LCD_FS* register. Values from 00 to 0x1F can be written to this register causing the VLCD voltage to be adjusted with an on-chip DAC from 70% of VLCD to 100% of VLCD.

Register Name	Address [bits]	R/W	Description
LCD_FS[4:0]	2022[4:0]	R/W	Controls the LCD full scale voltage, VLC2: $VLC2 = VLCD \cdot (0.7 + 0.3\frac{LCD\_FS}{31})$

Finally, the individual segments of the LCD have to be controlled. Since each pixel is addressed individually, the LCD display can be a combination of alphanumeric digits and custom enunciator symbols. The data for the segments is placed into the registers LCD_SEG0 through LCD_SEG41.
Writing the datum 0x06 into address 0x2030 will activate the corresponding segments when COM0 and COM1 outputs are active if the display is operated in two or three-state mode.

Register Name	Address [bits]	R/W	Description
LCD_SEG0[3:0] *…* *LCD_SEG41[3:0]*	2030[3:0] … 2059[3:0]	R/W	LCD Segment Data. Each word contains information for 1 to 4 time divisions of each segment. In each word, bit 0 corresponds to COM0, on up to bit 3 for COM3.

LCD Signals

Figure 2 shows typical LCD waveforms for four states at 1/3 bias for one frame of data. The voltage levels for V1, V2 and V3 depend on the voltage applied to the VLCD input pin. Assuming that 5V are supplied to VLCD (with voltage boost enabled and *LCD_FS* set to 31), the resulting amplitudes for V1, V2 and V3 will be 1.66, 3.33 and 5.0V.

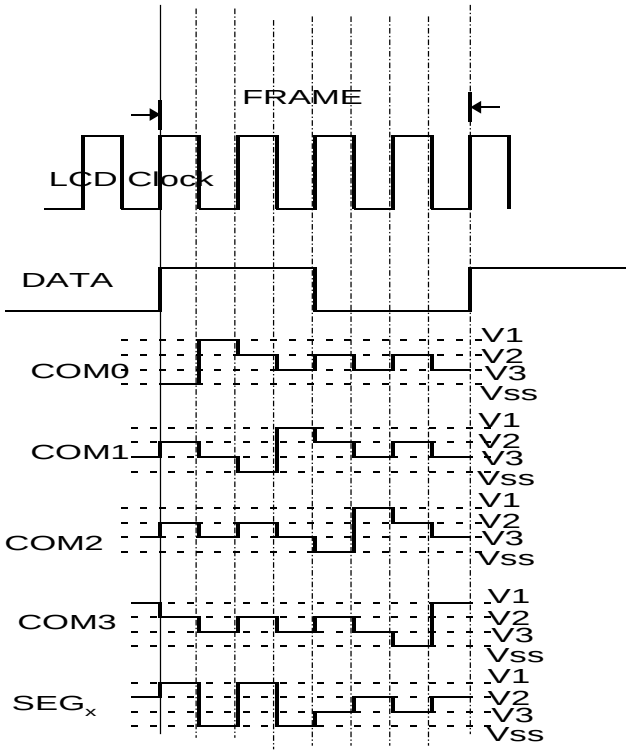

Figure 2: LCD Wave Forms

In Figure 2, the data for segment X changes from 1 to zero in the middle of the frame.

The LCD Module Used on the Demo Board

The Varitronix VIM-808 module is used for both the 6511 and 6513 Demo Boards. This LCD module has three common pins and provides eight digits consisting of seven segments, including decimal points for each digit and three colons located at digits 2, 4, and 6. The colons can be used for time display, e.g. “12:55:42”. Figure 3 shows the layout of the VIM-808 (digit numbers are shown above the segments).

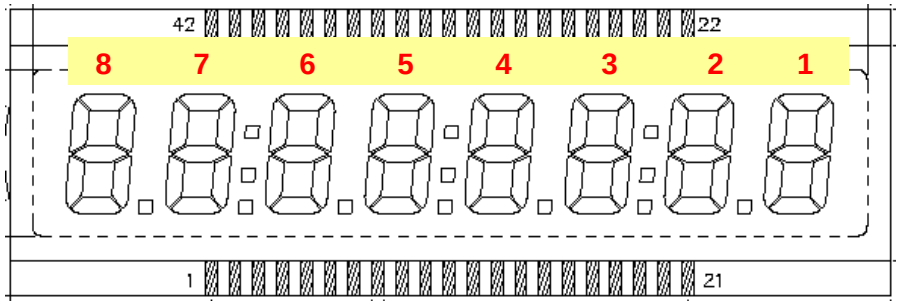

Figure 3: VIM-808 Top View

Figure 3 also shows the location of the connection pins for the VIM-808. Figure 4 shows a segment/pixel map for one digit. Note that only digits 2, 4, and 6 have the L segments forming the colon, and that the colon segments are connected to the 71M651X chip only on the 6513 Demo board.

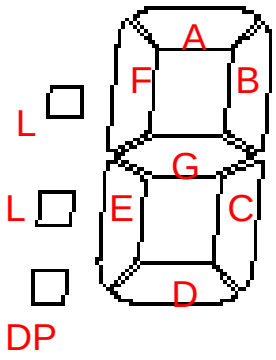

Figure 4: Segment Definitions

The pin-out map for the VIM-808 is shown in table 4. Twelve pins are not connected to any segment (NC). For example, pin 1 controls segments/pixels B and C of digit 8 plus the decimal point of digit 7.

Pin No.	Segment	Pin No.	Segment	Pin No.	Segment
1	8B, 8C, 7DP	15	3B, 3C, 2DP	29	3F, 3E, NC
2	NC	16	NC	30	4A, 4G, 4D
3	7B, 7C, 6DP	17	NC	31	4F, 4E, NC
4	NC	18	2B, 2C, 1DP	32	NC, NC, 4L
5	NC	19	NC	33	5A, 5G, 5D
6	6B, 6C, 5DP	20	1B, 1C, NC	34	5F, 5E, NC
7	NC	21	**COM 3**	35	6A, 6G, 6D
8	NC	22	**COM 1**	36	6F, 6E, NC
9	5B, 5C, 4DP	23	1A, 1G, 1D	37	NC, NC, 6L
10	NC	24	1F, 1E, NC	38	7A, 7G, 7D
11	NC	25	2A, 2G, 2D	39	7F, 7E, NC
12	4B, 4C, 3DP	26	2F, 2E, NC	40	8A, 8G, 8D
13	NC	27	NC, NC, 2L	41	8F, 8E, NC
14	NC	28	3A, 3G, 3D	42	**COM 2**

Table 4: VIM-808 Pin Map

The electrical connections implemented on the Demo Boards along with the VIM-808 pin map combine to the connection map shown in table 5. This table shows which LCD output pin on the 6511 is connected to which pin of the LCD. The NC pins from table 5 are not shown in Table 4.

VIM-808 LCD pin	I/O Address	Connected to		Affected Segments		
		6511 Pin	6513 Pin	COM1	COM2	COM3
1	2030	SEG00	SEG00	8B	8C	7DP
3	2031	SEG 01	SEG 01	7B	7C	6DP
6	2032	SEG 02	SEG 02	6B	6C	5DP
9	2033	SEG 03	SEG 03	5B	5C	4DP
12	2034	SEG 04	SEG 04	4B	4C	3DP
15	2035	SEG 05	SEG 05	3B	3C	2DP
18	2036	SEG 06	SEG 06	2B	2C	1DP
20	2037	SEG 07	SEG 07	1B	1C	--
21 (COM3)	--	**COM2**	**COM2**	--	--	--
22 (COM1)	--	**COM0**	**COM0**	--	--	--
23	204C	SEG 28	SEG 28	1A	1G	1D
24	204D	SEG 29	SEG 29	1F	1E	
25	204E	SEG 30	SEG 30	2A	2G	2D
26	204F	SEG 31	SEG 31	2F	2E	
27	--	--	SEG32	--	--	2L
28	2038	SEG 08	SEG 08	3A	3G	3D
29	2039	SEG 09	SEG 09	3F	3E	--
30	203A	SEG 10	SEG 10	4A	4G	4D
31	203B	SEG 11	SEG 11	4F	4E	--
32	--	--	SEG32	--	--	4L
33	203C	SEG 12	SEG 12	5A	5G	5D
34	203D	SEG 13	SEG 13	5F	5E	--
35	203E	SEG 14	SEG 14	6A	6G	6D
36	203F	SEG 15	SEG 15	6F	6E	--
37	--	--	SEG35	--	--	6L
38	2040	SEG 16	SEG 16	7A	7G	7D
39	2041	SEG 17	SEG 17	7F	7E	--
40	2042	SEG 18	SEG 18	8A	8G	8D
41	2043	SEG 19	SEG 19	8F	8E	--
42 (COM2)	--	**COM1**	**COM1**	--	--	--

Table 5: 651X LCD Pin/Segment Map

LCD Mapping

Figure 5 shows the physical mapping of the COM signals to the segments of the VIM-808 LCD.

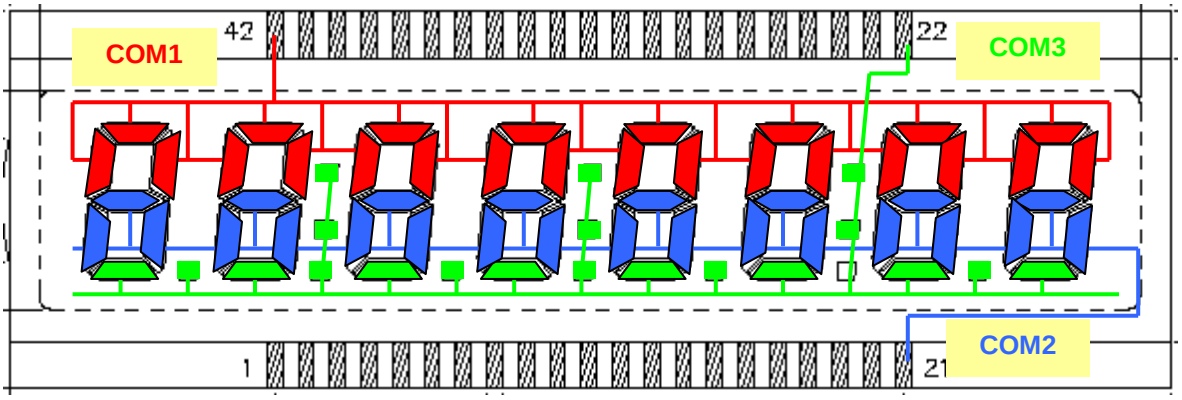

Figure 5: Physical Mapping of COM Signals

Supporting Firmware Routines

LCD Data Declarations for VIM-808 LCD:

```
#define COM(x) (x << 6)
enum COMMONS { COM0 = COM(0), COM1 = COM(1), COM2 = COM(2) };

enum SEGS                // List all segments of display in com/segment order.
{
    //COM0 SEGMENTS
    SEG_8B = COM0 | 0, SEG_7B = COM0 | 1, SEG_6B = COM0 | 2, SEG_5B = COM0 | 3,
    SEG_4B = COM0 | 4, SEG_3B = COM0 | 5, SEG_2B = COM0 | 6, SEG_1B = COM0 | 7,
    SEG_3A = COM0 | 8, SEG_3F = COM0 | 9, SEG_4A = COM0 |10, SEG_4F = COM0 |11,
    SEG_5A = COM0 |12, SEG_5F = COM0 |13, SEG_6A = COM0 |14, SEG_6F = COM0 |15,
    SEG_7A = COM0 |16, SEG_7F = COM0 |17, SEG_8A = COM0 |18, SEG_8F = COM0 |19,
    SEG_1A = COM0 |28, SEG_1F = COM0 |29, SEG_2A = COM0 |30, SEG_2F = COM0 |31,

    //COM1 SEGMENTS
    SEG_8C = COM1 | 0, SEG_7C = COM1 | 1, SEG_6C = COM1 | 2, SEG_5C = COM1 | 3,
    SEG_4C = COM1 | 4, SEG_3C = COM1 | 5, SEG_2C = COM1 | 6, SEG_1C = COM1 | 7,
    SEG_3G = COM1 | 8, SEG_3E = COM1 | 9, SEG_4G = COM1 |10, SEG_4E = COM1 |11,
    SEG_5G = COM1 |12, SEG_5E = COM1 |13, SEG_6G = COM1 |14, SEG_6E = COM1 |15,
    SEG_7G = COM1 |16, SEG_7E = COM1 |17, SEG_8G = COM1 |18, SEG_8E = COM1 |19,
    SEG_1G = COM1 |28, SEG_1E = COM1 |29, SEG_2G = COM1 |30, SEG_2E = COM1 |31,

    //COM2 SEGMENTS
    SEG_7DP= COM2 | 0, SEG_6DP= COM2 | 1, SEG_5DP= COM2 | 2, SEG_4DP= COM2 | 3,
    SEG_3DP= COM2 | 4, SEG_2DP= COM2 | 5, SEG_1DP= COM2 | 6,
    SEG_3D = COM2 | 8, SEG_4D = COM2 |10, SEG_5D = COM2 |12, SEG_6D = COM2 |14,
    SEG_7D = COM2 |16, SEG_8D = COM2 |18, SEG_1D = COM2 |28, SEG_2D = COM2 |30,
    SEG_2L = COM2 |32, SEG_4L = COM2 |33, SEG_6L = COM2 |34
};

// List all segments of all LCD icons.
enum SEGS code Digit_1[] =                                          // 1st digit.
    { SEG_1A, SEG_1B, SEG_1C, SEG_1D, SEG_1E, SEG_1F, SEG_1G };
enum SEGS code Digit_2[] =                                          // 2nd digit.
    { SEG_2A, SEG_2B, SEG_2C, SEG_2D, SEG_2E, SEG_2F, SEG_2G };
enum SEGS code Digit_3[] =                                          // 3rd digit.
    { SEG_3A, SEG_3B, SEG_3C, SEG_3D, SEG_3E, SEG_3F, SEG_3G };
enum SEGS code Digit_4[] =                                          // 4th digit.
    { SEG_4A, SEG_4B, SEG_4C, SEG_4D, SEG_4E, SEG_4F, SEG_4G };
enum SEGS code Digit_5[] =                                          // 5th digit.
    { SEG_5A, SEG_5B, SEG_5C, SEG_5D, SEG_5E, SEG_5F, SEG_5G };
enum SEGS code Digit_6[] =                                          // 6th digit.
    { SEG_6A, SEG_6B, SEG_6C, SEG_6D, SEG_6E, SEG_6F, SEG_6G };
enum SEGS code Digit_7[] =                                          // 7th digit.
    { SEG_7A, SEG_7B, SEG_7C, SEG_7D, SEG_7E, SEG_7F, SEG_7G };
enum SEGS code Digit_8[] =                                          // 8th digit.
    { SEG_8A, SEG_8B, SEG_8C, SEG_8D, SEG_8E, SEG_8F, SEG_8G };
enum SEGS code Icon_DP[]  =                                         // Decimal points.
    { SEG_1DP, SEG_2DP, SEG_3DP, SEG_4DP, SEG_5DP, SEG_6DP, SEG_7DP };
enum SEGS code Icon_L[]  =
    { SEG_2L, SEG_4L, SEG_6L };                                     // Colons ':'.

U08 code * code icons[] =
{
    Digit_1, Digit_2, Digit_3, Digit_4,
    Digit_5, Digit_6, Digit_7, Digit_8,
    Icon_DP, Icon_L
};
```

```
// Number of segments in each icon.
U08 code num_segs[] =
{
    sizeof (Digit_1), sizeof (Digit_2), sizeof (Digit_3), sizeof (Digit_4),
    sizeof (Digit_5), sizeof (Digit_6), sizeof (Digit_7), sizeof (Digit_8),
    sizeof (Icon_DP), sizeof (Icon_L)
};
```

The demo code low-level libraries provide the following functions for controlling the LCD:

Initialization, Configuration, Read, and Write Routines

```
/*****************************************************************************
 * This code and information is provided "as is" without warranty of any   *
 * kind, either expressed or implied, including but not limited to the     *
 * implied warranties of merchantability and/or fitness for a particular   *
 * purpose.                                                                *
 *                                                                         *
 * Copyright (C) 2003 TERIDIAN SemiConductor, Corporation. All Rights Reserved. *
 *                                                                         *
 *****************************************************************************/
//*****************************************************************************
//
//  DESCRIPTION: 71M651x POWER METER - LCD Routines.
//
//*****************************************************************************
//
// File: LCD.C
//
#include "options.h"
#include "api.h"

#define  LCD_SEG_COM0 BIT0           // LCD Segment Data COM0.
#define  LCD_SEG_COM1 BIT1           // LCD Segment Data COM1.
#define  LCD_SEG_COM2 BIT2           // LCD Segment Data COM2.
#define  LCD_SEG_COM3 BIT3           // LCD Segment Data COM3.

/*
 +-----a-----+  Mapping of character to digit/elements.
 |           |    Bitmapped  Mask
 |           |
 f           b    '0'  -,-,f,e, d,c,b,a
 |           |    '1'  -,-,-,-, -,c,b,-
 |           |    '2'  -,g,-,e, d,-,b,a
 +-----g-----+    '3'  -,g,-,-, d,c,b,a
 |           |    '4'  -,g,f,-, -,c,b,-
 |           |    '5'  -,g,f,-, d,c,-,a
 e           c    '6'  -,g,f,e, d,c,-,-
 |           |    '7'  -,-,-,-, -,c,b,a
 |           |    '8'  -,g,f,e, d,c,b,a
 +-----d-----+    '9'  -,g,f,-, -,c,b,a
*/

/*** External functions used within this module ***/
// None

/*** External variables used within this module ***/
extern U08r * code icons[];         // Defined by user application.
extern U08r num_segs[];             //
extern U16r idx2bit[];

/*** Public functions declared within this module ***/
//  LCD API.
//
void LCD_Init (void);
void LCD_Command (U08 LcdCmd);
```

```
void LCD_Config (U01 boost, U08 num, enum LCD_BIAS bias, enum LCD_CLK clock, U08
voltage);
U16  LCD_Data_Read (U08 Icon);
void LCD_Data_Write (U08 icon, U16 Mask);

//*****************************************************************************
// LCD_Command (LCD_CLEAR);       Clear display.
// LCD_Command (LCD_DISPLAY_ON);  Display ON.
// LCD_Command (LCD_DISPLAY_OFF); Display OFF.
//*****************************************************************************

#define LCD_CLEAR       0x01   // Clears Entire LCD.
#define LCD_DISPLAY_ON  0x0C   // Display ON.
#define LCD_DISPLAY_OFF 0x08   // Display OFF.

/*** Public variables declared within this module ***/
// None.

/*** Private functions declared within this module ***/
static void lcd_clear (void);
static void lcd_ctrl (U08 ctrl);

/*** Private variables used within this module ***/
#define SEGMENT_MASK  0x3F              // Number of SEGMENT drivers.
#define COM_MASK      ~SEGMENT_MASK     // Number of COMMONs.
#define COM_ALIGN     6

static U08r com2bit[] = { LCD_SEG_COM0, LCD_SEG_COM1, LCD_SEG_COM2, LCD_SEG_COM3 };

//  LCD API.
//
void LCD_Init (void)                // This routine should be called on power-up
{
    LCD_Command (LCD_CLEAR);            // Display Cleared.
    LCD_Command (LCD_DISPLAY_ON );      // Display ON.
}

void LCD_Command (U08 LcdCmd)
{
    if (LcdCmd & LCD_CLEAR)
       lcd_clear ();
    else
       lcd_ctrl (LcdCmd);
}

U16 LCD_Data_Read (U08 Icon)
{                                          // Handle up to 16 segments per icon.
    U08 code *seg;
    U08 xdata i, xdata common, xdata seg_drv;
      U16 Value = 0;                        // Assume no active seqments.

    seg = icons[ Icon ];

    for (i = 0; i < num_segs[ Icon ]; i++)
    {
       seg_drv = *seg & SEGMENT_MASK;
       common  = com2bit[ *seg >> COM_ALIGN ];

         if (LCD_SEG[ seg_drv ] & common)
          Value |= idx2bit[ i ];           // Segment is active.

       seg++;
    }

    return (Value);
}
```

```
void LCD_Data_Write (U08 Icon, U16 Mask)
{                                              // Handle upto 16 segments per icon.
    U08 code *seg;
    U08 xdata i, xdata common, xdata seg_drv;

    seg = icons[ Icon ];

    for (i = 0; i < num_segs[ Icon ]; i++)
    {
       seg_drv = *seg & SEGMENT_MASK;
       common  = com2bit[ *seg >> COM_ALIGN ];

       if (Mask & 1)
          LCD_SEG[ seg_drv ] |= common;    // Activate segment.
       else
          LCD_SEG[ seg_drv ] &= ~common;   // DeActivate segment.

       seg++;
       Mask >>= 1;
    }
}

void LCD_Config (U01 boost, U08 num, enum LCD_BIAS bias, enum LCD_CLK clock, U08
voltage)
{
    LCDX = (boost << 7) | num;
    LCDY = (LCDY & ~(LCD_MODE | LCD_CLK_)) | (bias << 2) | clock;
    LCDZ = (LCDZ & ~LCD_FS) | voltage;
}

static void lcd_clear (void)
{
    memset_x (LCD_SEG, 0, sizeof (LCD_SEG));
}

static void lcd_ctrl (U08 ctrl)
{
    if (LCD_DISPLAY_ON == ctrl)
       LCDY |= LCD_EN;
    else
       LCDY &= ~LCD_EN;
}
```

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600