Application Note

Character LCD Interface for the Z8 Encore!® MCU

AN014304-0608

Abstract

This Application Note provides Character Liquid Crystal Display (LCD) driver routines (coded in ANSI C) for Zilog's Z8 Encore!® Flash microcontroller-based embedded software.

The LCD driver routines discussed in this Application Note utilize an LCD driver library built for a generic 16 x 2 character LCD display. This 16x2 character LCD uses the industry-standard 4-bit data transfer mode. With minor modifications, the driver APIs can also be used for character LCD formats such 8x1, 16x1, 20x2, and others, in combination with different LCD controllers. The hardware interface between the LCD controller and the Z8 Encore! microcontroller is built by using seven GPIO pins tied to a single port.

The APIs in this LCD driver library perform the following functions:

- Set up and initialize the LCD controller, (specific to the type of display).
- Turn the display ON and OFF.
- Display an ASCII character/symbol.
- Display a null-terminated string.
- Display a character or symbol a specified number of times.
- Read the character or symbol at the current location.
- Clear the screen.
- Home-position the cursor.
- Backspace the cursor one position.
- Insert one space at the current location.
- Blink the cursor at the current position.
- Turn off cursor blinking.
- Read the current cursor position in DDRAM.
- Set the print location at the line number or column number
- Scroll text from the bottom line to the top line.
- Write a user-defined pixel map in LCD CGRAM from the table.
- Display user-defined character patterns.

Note: *The source code file associated with this application note (*`AN0143-SC01.zip`*) is available for download at* www.zilog.com.

Z8 Encore! Flash MCU Overview

Zilog's Z8 Encore! products are based on the new eZ8™ CPU and introduce Flash Memory to Zilog's extensive line of 8-bit microcontrollers. Flash Memory in-circuit programming capability allows for faster development time and program changes in the field. The high-performance register-to-register based architecture of the eZ8 core maintains backward compatibility with Zilog's popular Z8® MCU.

The Z8 Encore! microcontrollers combine a 20 MHz core with Flash Memory, linear-register SRAM, and an extensive array of on-chip peripherals. These peripherals make the Z8 Encore! suitable for various applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

For more information on the Z8 Encore!® devices, refer to appropriate product specifications available for download at www.zilog.com.

Discussion

Liquid Crystal Display (LCD) is a useful medium of communication in various applications, especially in consumer goods such as washing machines, microwave appliances, and VCRs.

The number of lines displayed and the number of characters displayed per line characterize LCDs into 16x2, 40x2, and 40x4 formats. An LCD requires a controller to control various features of its display. An LCD with a controller is referred to as an LCD module. The following section describes one such LCD module.

Character LCD Module

The Character LCD Module used to demonstrate character LCD interfacing with the Z8 Encore! MCU is a 16x2 character LCD, controlled by a Hitachi controller device (Part Number HD44780). It features the following main units:

- Pins
- Display Data RAM (DDRAM)
- Character Generator RAM (CGRAM)
- Character Generator ROM (CGROM)
- Busy flag

Pin Description

The Character LCD Module, with its onboard HD44780 controller, contains eleven signal pins for controlling the LCD, in addition to the power supply pins. Table 1 provides the pin descriptions of a HD44780-based Character LCD Module.

The connector for the Character LCD Module can be arranged as a 14x1 pin header or as a 7x2 pin header. The pin descriptions listed in Table 1 are applicable for both (14x1 pin header or 7x2 pin header) the arrangements.

Table 1. Pin Descriptions of a Typical HD44780-based Character LCD Module

Pin No.	LCD Pin Name	Description
1	V_{SS}	GND (0 V).
2	V_{DD}	+V (5 V).
3	V_L	Pot. Wiper (Voltage between V_{SS} & V_{DD}).
4	RS	Register Select (1 = Data Input, 0 = Instruction Input).
5	RW	Read/Write mode (1 = Read data from LCD, 0 = Write data).
6	E	Enable pulse signal (High to Low pulse for short duration).
7	D0	Data pin (unused in 4-bit mode).
8	D1	Data pin (unused in 4-bit mode).
9	D2	Data pin (unused in 4-bit mode).
10	D3	Data pin (unused in 4-bit mode).
11	D4	Data pin.
12	D5	Data pin.
13	D6	Data pin.
14	D7	Data pin.

The LCD interface described in this Application Note utilizes a 4-bit data bus to communicate with the HD44780 controller. Therefore, the interface uses four GPIO pins less than a typical 8-bit transfer mode. For a 4-bit wide interface, data is transferred through the D4–D7 pins only, thus, leaving the D0–D3 pins unused.

Data from the higher nibble is transferred first, followed by data from the lower nibble. One complete data transfer between the HD44780 controller and the microcontroller, for a 4-bit wide interface, requires 4-bit data to be transferred twice. Conversely, when the bus is 8 bits wide, data is transferred using of the D0–D7 pins together.

Display Data RAM

DDRAM on the Character LCD Module stores the data to be displayed and is represented by 8-bit ASCII character codes. In the HD44780 controller, the DDRAM capacity is 80x8 bits, representing a total of 80 characters. DDRAM can also be used as a scratch data area (if specified by the manufacturer).

Figure 1 displays the correspondence between DDRAM addresses and positions on the display module for a 16 character x 2 line LCD. The DDRAM address is available in the Address Counter, which is an 8-bit wide register that stores the address value of the currently-accessed memory location.

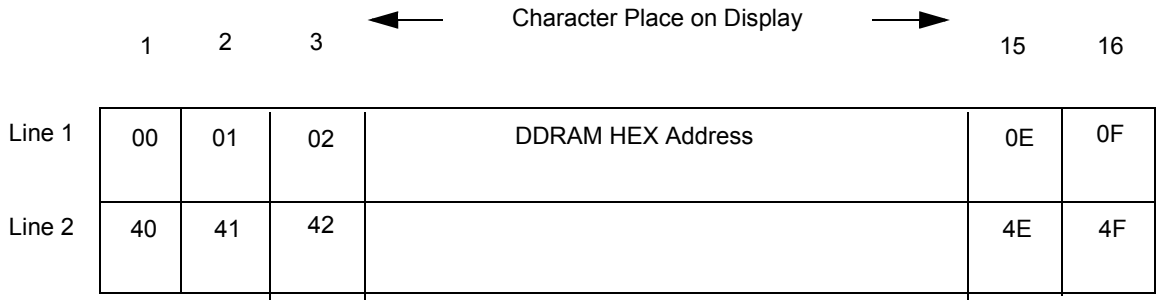

Figure 1. Display Position and DDRAM Address Relation for 16x2 LCD

Character Generator ROM

The LCD Module's HD44780 controller contains a CGROM that stores 5x8-dot or 5x10-dot character patterns in an 8-bit addressable space.

The CGROM contains a total of 208 5x8-dot character patterns and 32 5x10-dot character patterns. The various mask versions exhibit different capacities.

Note: *These character patterns are factory programmed and cannot be altered.*

Character Generator RAM

The CGRAM of the LCD Module's HD44780 controller contains user-defined character patterns. Eight user-defined character patterns in the 5x8 format and four user-defined character patterns in the 5x10 format (using 11 bytes per character) can be stored in CGRAM.

Busy Flag

The Busy flag is the 8th bit in the Address Counter. This bit indicates whether the HD44780 controller is busy processing internal data or is ready to accept new commands.

Character LCD Interfacing to the Z8 Encore!® MCU

This section describes the hardware and software implementation for interfacing the Character LCD with the Z8 Encore! MCU.

Hardware Implementation

Figure 2 displays the block diagram of the LCD Interface with the Z8 Encore! MCU.

In Figure 2, the interface utilizes the HD44780 controller's 4-bit mode, therefore only seven GPIO pins are required.

For software implementation described in this Application Note, a single port of the Z8 Encore! MCU is used to connect to the LCD's signal and data pins.

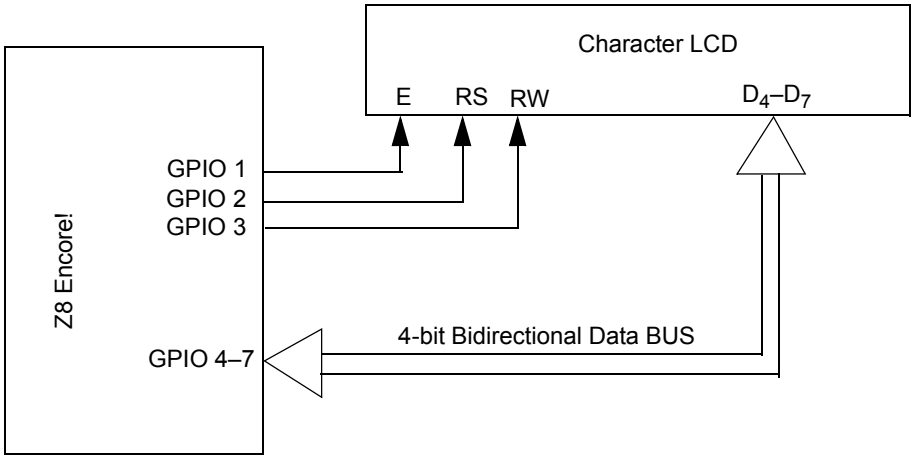

Figure 2. Block Diagram of the LCD Interface with the Z8 Encore! MCU

Software Implementation

As displayed in Figure 3, the LCD device driver is the lowest software layer and the user application layer is above the LCD device driver API layer.

The device driver layer directly interacts with the Z8 Encore! hardware registers and the port pins. The API are utilized by the user application to Read/Write data from/to the Character LCD Module.

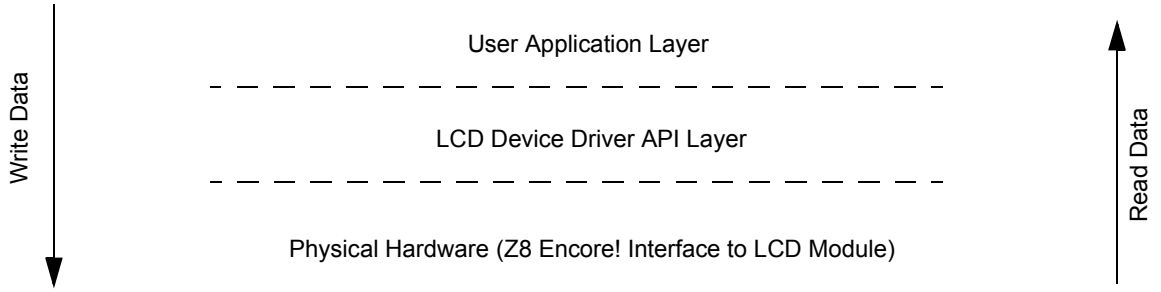

Figure 3. Software Layer Arrangement

Using LCD Device Driver APIs

The software developed for this application implements LCD device driver API code for a 16x2 LCD Module, with appropriate initialization of the LCD controller.

The APIs featured in this Application Note are built for a generic HD44780-based LCD module. The routine `LCD_init()` contains instructions to specifically initialize a 16x2 LCD module.

Note: *To use a 40x2 LCD module, the* `LCD_init()` *routine must be modified appropriately, along with a declaration of* `#define LCD_40x2`, *in the* `LCD_API.h` *file.*

The `LCD_init()` API must be called before any other API.

Follow the steps below to startup the LCD module with Z8 Encore!® MCU:

1. Initialize the Z8 Encore! GPIO pins (using `LCD_gpio_set()`).
2. Initialize Timer0 for 1 ms delay in CONTINUOUS mode (using `init_timer0`).
3. Enable Interrupts (using Macro: `EI`).
4. Provide power-on delay for LCD warm up, minimum 15 ms (using `delayms`).
5. Write instructions sequentially to the LCD module, according to the flowchart displayed in Appendix A—Flowcharts on page 9 (using `LCD_init()`).
6. Turn off the cursor blink (using `LCD_blink_off()`).

The GPIO pins used for the LCD interface are defined in the `LCD_API.h` file. To provide the required delay, the timer Timer0 is used in the CONTINUOUS mode with a time-out of 1 ms.

On initialization, following tasks can be performed to display text on the LCD Module:

- Print an ASCII Character
- Print an ASCII Character String
- Print a User-defined Character
- Replace the Character Before Current Cursor Position with a New One
- Insert a Character in the Middle of a String
- Scroll a Long Line of Text Across the Screen

Print an ASCII Character

Follow the steps below to print an ASCII character ‘Z’ at 5th column of 1st row of LCD module:

1. Set the print location as 5th column of 1st row (using `LCD_setposition(0,5)` API).
2. Call the `LCD_printc` API with character as argument (using `LCD_printc('Z')` API).

Print an ASCII Character String

Follow the steps below to print an ASCII character string ‘`Zilog`’:

1. Clear the screen using `LCD_clear()` API.
2. Print the string using `LCD_prints('Zilog')` API.

Print a User-defined Character

Follow the steps below to print a user-defined character:

1. Define the format of special characters in the `defchar.h` file as 8 bytes per character. The HD44780’s CGRAM can contain a maximum of 64 Bytes, thus limiting it to 8 characters.

Figure 4 displays a construction pattern as a sample character. The character is formed using 7 rows of 5 pixels each. The last row redisplays the underline cursor.

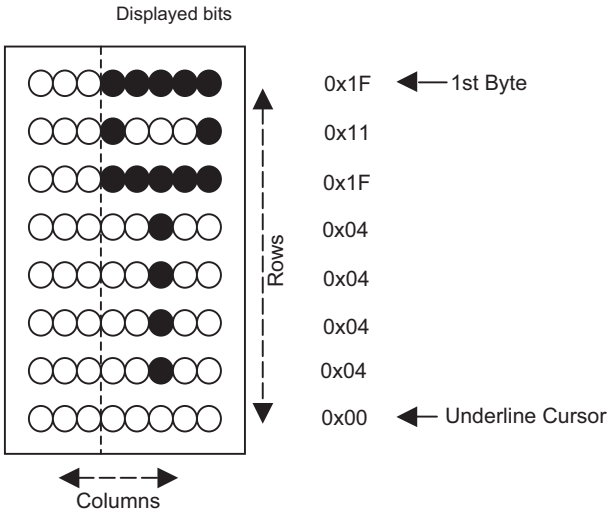

Figure 4. Construction Pattern for a Sample Character

2. Copy the character pattern table to a CGRAM location, for example at `0x03` (using `LCD_copymap()` API).
3. Print the special character (using `LCD_printc_user(0x03)` API).

Note: *The* `LCD_test.c` *file contains the code to display text on the LCD Module.*

Replace the Character Before Current Cursor Position with a New One

Follow the steps below to replace the character before current cursor position with a new one:

1. Delete the character (using `LCD_backspace()` API).
2. Print a new character in its location (using `LCD_printc()` API).

Insert a Character in the Middle of a String

Follow the steps below to insert a character in the middle of a string:

1. Relocate the cursor to the new position (using `LCD_setposition()` API).
2. Print a new character in the location (using `LCD_printc()` API).

Scroll a Long Line of Text Across the Screen

Follow the steps below to scroll a long line of text across the screen:

1. Define the long line as: `unsigned char text[] = "A long line of text ...";`
2. Call the scroll API (using `LCD_scroll(text)`).

Appendix A—Flowcharts on page 9 displays a flowchart for the initialization routine for a HD44780 controller on a 16x2 LCD Module. Initialization routines are usually specified in the LCD manufacturer's datasheets.

Appendix B—LCD Device Driver API Description on page 10 provides a description of all the APIs in the LCD driver library. Example usage of these APIs is demonstrated in the `LCD_test.c` file.

Besides the driver APIs, the software implementation contains support routines that are used to read or write data from/to the LCD controller. These support routines are utilized by the LCD device driver APIs. Appendix B—LCD Device Driver API Description on page 10 provides a list of these support routines.

Testing

The LCD device driver APIs were tested on a 16x2 Character LCD Module using the `LCD_test.c` demo file.

Equipment Used

The equipment required to test the LCD device driver library include:

- Z8 Encore!® Development kit (Z8ENCORE000ZCO).
- ZDS II IDE with C Compiler/Simulator/Online Debugger, v4.7.0.
- Digital Storage Oscilloscope (Tektronix TDA724D).

Procedure and Results

The LCD Module is connected to the Z8 Encore! microcontroller as displayed in Figure 2 on page 4. The GPIO pins are set up according to the configuration in `LCD_API.h` file.

Note: *You must apply proper voltage to the GND, V_{CC}, & V_L (pins 1, 2, and 3 respectively, as indicated in Table 1 on page 2) of the LCD module.*

The results were as expected. The programs run stably, with the LCD displaying text messages at proper locations, with timed delays.

Summary

There are different methods to implement a Character LCD interface using the microcontrollers available in the market today. This Application Note demonstrates a HD44780-based Character LCD signal and data pin to interface with a Z8 Encore! MCU. This configuration uses the single port method with appropriate driver software. The driver software is a ready-to-use library of APIs which allows you to integrate a general Character LCD Module efficiently with the Z8 Encore! MCU.

Reference

The documents associated with Z8 Encore! MCU, eZ8 CPU, and the Hitachi driver are provided below:

- Z8 Encore! XP® F64XX Series Product Specification (PS0199)
- eZ8™ CPU User Manual (UM0128)
- Hitachi HD44780U Dot Matrix LCD Controller/Driver — HD44780U (LCD II), available at Hitachi website

Appendix A—Flowcharts

Figure 5 displays the flowchart for initializing HD44780 Controller on the 16x2 LCD Module.

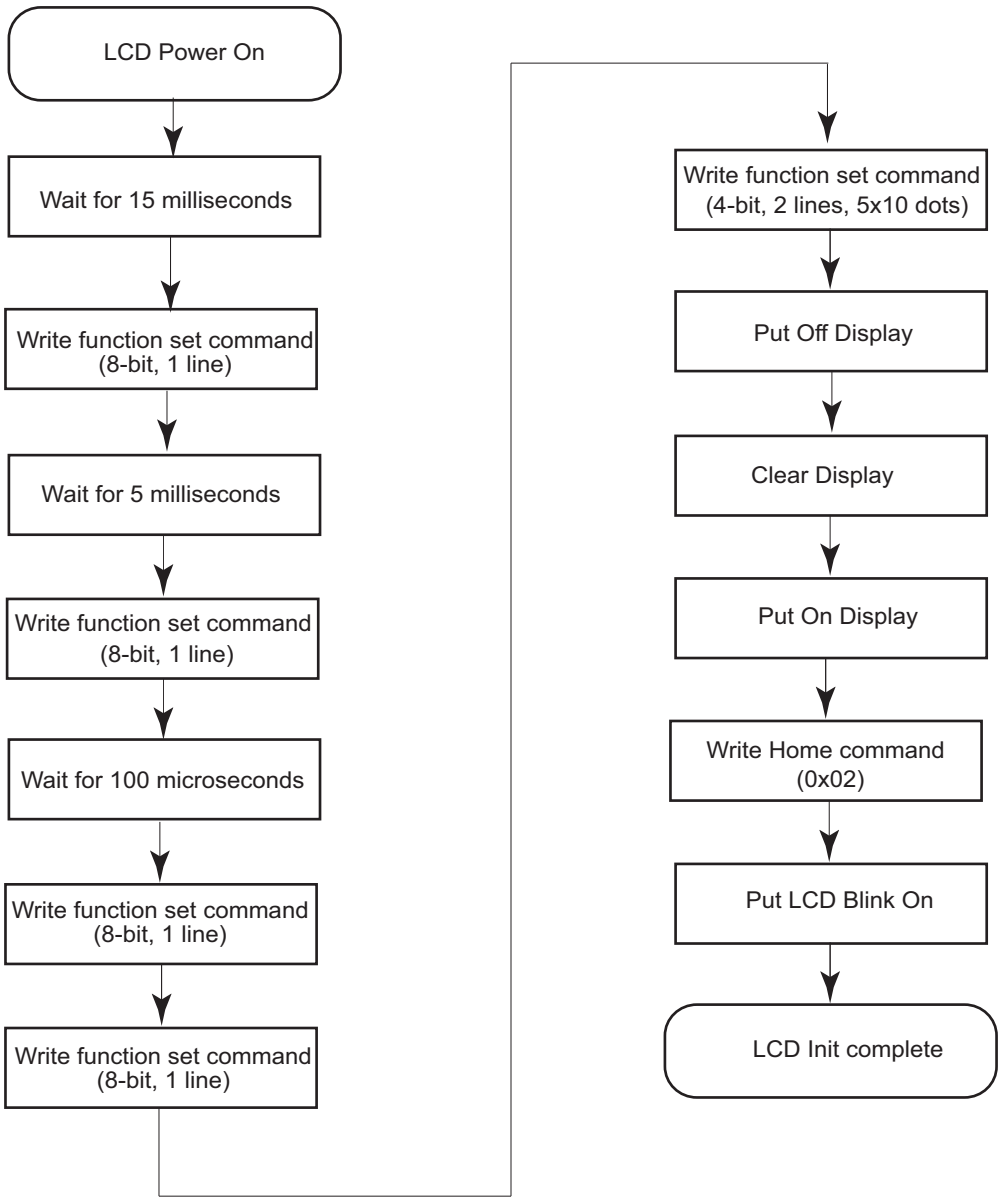

Figure 5. Initializing the HD44780 Controller on the 16x2 LCD Module

Appendix B—LCD Device Driver API Description

This appendix provides a brief description of all the LCD device driver APIs to develop a Character LCD interface for the Z8 Encore!® MCU.

LCD_init

API Detail	Function
Parameters	None
Returns	void
Description	Initializes LCD as per manufacturer's specifications
Usage	`LCD_init();`

LCD_on

API Detail	Function
Parameters	None
Returns	void
Description	Enables printing on the display
Usage	`LCD_on() ;`

LCD_off

API Detail	Function
Parameters	None
Returns	void
Description	Disables display
Usage	`LCD_off();`

LCD_printc

API Detail	Function
Parameters	unsigned char
Returns	void
Description	Prints an ASCII symbol at current cursor position
Usage	`LCD_printc('A');`

LCD_prints

API Detail	Function
Parameters	unsigned char *
Returns	void
Description	Prints a string of ASCII symbols at current cursor position
Usage	`LCD_prints("A String") ;`

LCD_fillchar

API Detail	Function
Parameters	unsigned char, unsigned char
Returns	void
Description	Prints ASCII symbol at current cursor position a specified number of times
Usage	`LCD_fillchar('A',5);`

LCD_readchar

API Detail	Function
Parameters	void
Returns	unsigned char
Description	Returns HEX value of ASCII symbol at current cursor position
Usage	if (LCD_readchar() == 'A') { ; }

LCD_clear

API Detail	Function
Parameters	void
Returns	void
Description	Clears the display screen
Usage	`LCD_clear();`

LCD_home

API Detail	Function
Parameters	void
Returns	void
Description	Brings cursor to first position of first line without changing contents in DDRAM
Usage	`LCD_prints("A String") ;`

LCD_backspace

API Detail	Function
Parameters	void
Returns	void
Description	Relocates the cursor to one place before current cursor position and deletes the content there
Usage	`LCD_backspace();`

LCD_insert

API Detail	Function
Parameters	void
Returns	void
Description	Prints an ASCII space symbol at current cursor position and shifts the rest of the contents one place towards right
Usage	`LCD_insert();`

LCD_blink

API Detail	Function
Parameters	void
Returns	void
Description	Blinks the symbol at current cursor position
Usage	`LCD_blink();`

LCD_blink_off

API Detail	Function
Parameters	void
Returns	void
Description	Stops cursor blinking
Usage	`LCD_blink_off();`

LCD_curpos

API Detail	Function
Parameters	void
Returns	unsigned char
Description	Reads the value of DDRAM address at current cursor position
Usage	`current_position = LCD_cursor();`

LCD_scroll

API Detail	Function
Parameters	unsigned char *
Returns	void
Description	Prints the specified string, then moves the contents of lower lines to one line above and repeats till end of string
Usage	`unsigned char* long_line = "A very long line of text";` `LCD_scroll(long_line);`

LCD_copymap

API Detail	Function
Parameters	unsigned char *
Returns	void
Description	Copies eight data patterns from specified RAM location to CGRAM addresses 0x00 to 0x07
Usage	`unsigned char table[64] = { …. 8x8 data table ...} ; LCD_copymap(table);`

LCD_setposition

API Detail	Function
Parameters	unsigned char, unsigned char
Returns	void
Description	Relocates the cursor to specified position on display
Usage	`LCD_setposition(0,7); // Middle of 1st line on 16x2 LCD`

LCD_delay

API Detail	Function
Parameters	void
Returns	void
Description	Provides fixed delay between commands during Initialization routine
Usage	`LCD_delay();`

LCD_checkbusy

API Detail	Function
Parameters	void
Returns	unsigned char (1=LCD is busy, 0 = LCD is not busy)
Description	Checks whether LCD controller is busy
Usage	`while (LCD_checkbusy) ; // Wait till LCD is busy`

LCD_printc_user

API Detail	Function
Parameters	unsigned char
Returns	unsigned char
Description	Displays a user defined Character from CGRAM location to current cursor position
Usage	`LCD_printc_user(5); // Display the fifth user defined symbol`

A brief functional description for each LCD device driver support API is provided below.

LCD_command

API Detail	Function
Parameters	unsigned char
Returns	void
Description	Writes a command word to the LCD controller
Usage	`LCD_command(0x01); // Clear display command`

LCD_writedata

API Detail	Function
Parameters	unsigned char
Returns	void
Description	Writes a data word to the LCD controller. This API should be preceeded by an instruction to set DDRAM or CGRAM address to where the data is to be written.
Usage	`LCD_writedata(0x20); // Write data word 0x20`

LCD_set_cgram

API Detail	Function
Parameters	unsigned char
Returns	void
Description	Sets CGRAM address to specified value
Usage	`LCD_set_cgram(0x00); // Point to 1st address of CGRAM`

LCD_set_ddram

API Detail	Function
Parameters	unsigned char
Returns	void
Description	Sets DDRAM address to specified value
Usage	`LCD_set_ddram(0x00); // Point to 1st address of DDRAM`

Appendix C—Theory of LCD Operation

You must choose a suitable character LCD Module for your application. This appendix provides a brief description on the basic LCD technology and the components of an LCD.

Construction of an LCD Panel

Figure 6 displays the construction of a typical LCD. A liquid crystal fluid is inserted between two sheets of non-conductive glass. This active medium of fluid is used to create an image by aligning elongated crystals to the fields created by an electric charge. The layers of glass contain conductive patterns along which the crystals align.

The layers of glass are further coated with polarized matter to reflect or pass the incident light. In the absence of an electric charge, the incident light passes through the two glass layers, making the display transparent or unlit. When an electric charge is applied, the incident light is reflected, creating the impression of a lit segment.

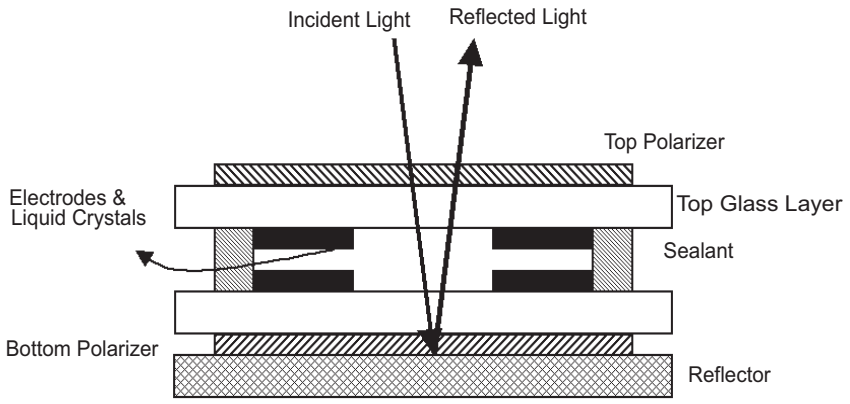

Figure 6. Construction of a Typical LCD Module

Backlighting

The LCD panel relies on reflected light to display its segments. For providing additional light, backlit displays are the norm. Backlight can exist in the form of an LED array arranged at the side of the panel, an electroluminescent (EL) panel, or a Cold Cathode Fluorescent (CCFL) panel.

An LED array provides a soft diffused light, but it is prone to placing undesirable light and dark patterns over the panel. It is powered by the same power rail used for the LCD. CCFL technology is complex and is generally used with large displays. It utilizes AC voltage (through a high-frequency DC-to-AC inverter) for providing light. The EL panel is the most widely used, consuming less wattage compared to others. It also uses a DC-to-AC inverter for operation, but can be easily powered by a battery. The EL and CCFL backlights are used sparingly, as they sustain a shorter life compared to the display module. When expended, the backlight cannot be replaced.

Fluid Technology Options

A liquid crystal is specified in terms of its ability to twist light. Accordingly, LCDs are of following three types:

- Twisted Nematic (TN)
- Super Twisted Nematic (STN)
- Filtered STN (FSTN)

Compared to TN, STN provides a wide-angled view and better contrast but is prone to slow response times. An FSTN is a special type of STN display that provides monochrome images.

Polarizer

A polarizer coating can be one of following three types:

- Reflective,
- Transmissive
- Transflective

With a reflective type of coating, the LCD panel can be operated only in brightly-lit ambience. Conversely, with a transmissive type of coating, a backlight can be used. The transflective type is a combination of the other two and can be used in any type of lighting conditions as it is able to reflect as well as transmit backlight. It is the most commonly used of the three types.

Contrast Adjustment and Temperature Compensation

The alignment of an LCD's fluid is sensitive to temperature fluctuations. An LCD Module tested at 25 ºC exhibits a different contrast ratio than at a higher temperature of 50 ºC. To compensate for this contrast change, a V_L pin is provided in LCD Modules that can be powered with a different drive voltage than required for LCD operation. Figure 4 displays a simple potentiometer used to vary V_L drive voltage.

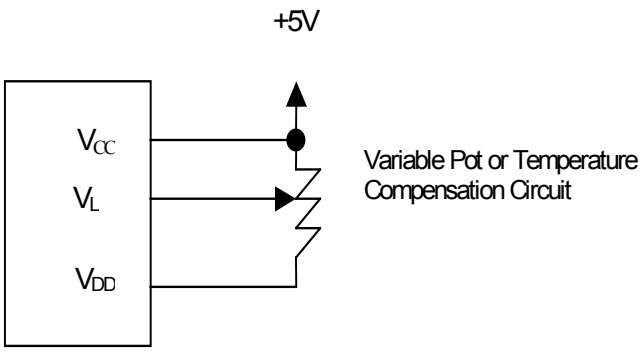

Figure 4. Contrast Control

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer