Application Note

Character LCD Interface for eZ80Acclaim!® MCUs

AN015902-0708

Abstract

This Application Note provides Character LCD driver routines, coded in ANSI C, for Zilog's eZ80Acclaim!® Flash microcontroller-based embedded software. The driver library is built for a generic 16x2 character LCD display that is fitted with a Hitachi HD44780 controller. This 16x2 character LCD uses the industry standard 4-bit data transfer mode. The driver APIs can also be used, with minor modifications, for other character LCD dimensions, such as 8x1, 16x1, and 20x2 in combination with different LCD controllers.

The hardware interface between the LCD controller and an eZ80Acclaim! microcontroller unit (MCU) is built by using seven GPIO pins on a single port.

Note: *The source code file associated with this application note,* `AN0159-SC01.zip` *and* `AN0159-SC02.zip` *are available for download at* www.zilog.com.

The APIs in the Character LCD driver library provide services to:

- Set up and initialize the LCD controller specific to the type of display
- Turn 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 a character or symbol at a current location
- Clear the screen
- Home-position the cursor
- Backspace the cursor one position
- Insert one space at a current location
- Blink the cursor at a current position
- Turn off the cursor blink
- Read the current cursor position in DDRAM
- Set the print location at a line number and column number
- Scroll text from the bottom line to the top line
- Write a user-defined pixel map in an LCD CGRAM from a table
- Display user-defined character patterns

eZ80Acclaim! Flash MCU Overview

eZ80Acclaim! on-chip Flash MCUs are an exceptional value for you in designing high performance, 8-bit MCU-based systems. With speeds up to 50 MHz and an on-chip Ethernet MAC (for eZ80F91 only), you have the performance necessary to execute complex applications quickly and efficiently. Combining Flash and SRAM, eZ80Acclaim! devices provide the memory required to implement communication protocol stacks and achieve flexibility when performing in-system updates of application firmware.

The eZ80Acclaim! Flash MCU can operate in full 24-bit linear mode addressing 16 MB without a Memory Management Unit. Additionally, support for the Z80®-compatible mode allows Z80/Z180 customers to execute legacy code within multiple 64 KB memory blocks with minimum modification. With an external bus supporting eZ80®, Z80, Intel, and Motorola bus modes and a rich set of

serial communications peripherals, you have several options when interfacing to external devices.

Some of the many applications suitable for eZ80Acclaim! devices include vending machines, point-of-sale terminals, security systems, automation, communications, industrial control and facility monitoring, and remote control.

Discussion

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

The number of lines displayed and the number of characters displayed per line characterize LCDs into 16x2, 40x2, and 40x4 dimensions. 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 for demonstrating character LCD interfacing with the eZ80Acclaim! MCU is a 16x2 character LCD controlled by the HD44780 controller by Hitachi. It features the following main units:

- Pins
- DDRAM
- CGRAM
- CGROM
- Busy Flag

Pin Description

The Character LCD Module, with its on-board HD44780 controller, contains eleven signal pins for controlling the LCD, in addition to the power supply pins. These pins are listed in Table 1.

Table 1. Pin Descriptions for 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	D_0	Data pin (unused in 4-bit mode)
8	D_1	Data pin (unused in 4-bit mode)
9	D_2	Data pin (unused in 4-bit mode)
10	D_3	Data pin (unused in 4-bit mode)
11	D_4	Data pin
12	D_5	Data pin
13	D_6	Data pin
14	D_7	Data pin

The connector for a Character LCD Module can be arranged as 14x1 pin header or as a 7x2 pin header. In either case, the pin descriptions listed in Table 1 are valid.

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

Data in the higher nibble is transferred first, followed by data in the lower nibble. For a 4-bit wide interface, one complete data transfer between the HD44780 controller and the eZ80Acclaim! MCU requires 4-bit data to be transferred twice. Conversely, when the bus is 8-bits wide, data is transferred using the entire range of D0–D7 pins.

Display Data RAM

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

The correspondence between DDRAM addresses and positions on the display module for a 16-character x 2-line LCD are displayed in Figure 1. 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.

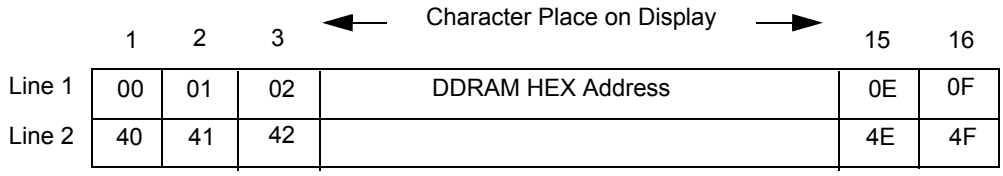

Figure 1. Display Position and Related DDRAM Address for a 16x2 LCD

Character Generator ROM

The LCD Module's HD44780 controller contains a Character Generator ROM (CGROM) that stores 5 x 8 dot or 5 x 10 dot character patterns in an 8-bit addressable space. In all, the CGROM contains 208 5 x 8 dot character patterns and 32 5 x 10 dot character patterns. The various mask versions have different capacities.

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

Character Generator RAM

The Character Generator RAM (CGRAM) of the LCD Module's HD44780 controller contains user-defined character patterns. Eight user-defined character patterns in the 5 x 8 format and four user-defined character patterns in the 5 x 10 format (using 11 bytes per character) can be stored in CGRAM.

Busy Flag

The Busy Flag is the eighth bit in the Address Counter. It indicates that the HD44780 controller is busy processing internal data or is ready to accept new commands.

Interfacing a Character LCD to the eZ80Acclaim! MCU

Hardware Overview

A block-level scheme of the LCD Interface with an eZ80Acclaim! MCU is displayed in Figure 2. The interface utilizes the HD44780 controller's 4-bit mode; therefore only seven GPIO pins are required for connecting to the LCD's signal and data pins.

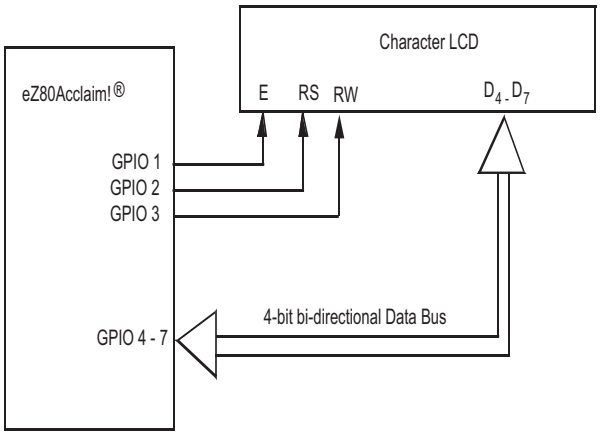

Figure 2. An LCD Interface with the eZ80Acclaim! MCU

Software Implementation

As displayed in Figure 3, the LCD device driver is the lowest software layer; the user application is built above that layer. The device driver layer interacts directly with the eZ80Acclaim! MCU hardware registers and port pins. The API services are utilized by the user application to read and write data to and from the Character LCD Module.

The `startup.asm` start-up module is written with suitable allocation for the eZ80Acclaim! Module's memory maps. This code provides appropriate boot-up of the eZ80Acclaim! Module as per the RAM and Flash peripherals available on the eZ80® Development Platform. For a standalone design using an eZ80Acclaim! MCU, this start-up code can be modified to accommodate the attached peripherals.

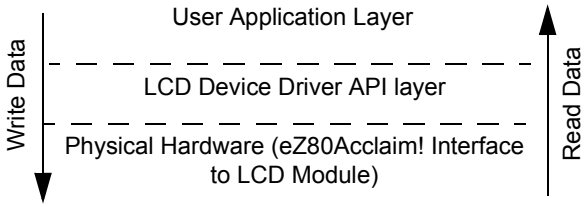

Figure 3. Software Layer Arrangement

Using LCD Device Driver APIs

The software developed for this application implements driver API code for a 16x2 LCD Module with appropriate initialization of the LCD controller. The service APIs featured in this Application Note (refer to the `LCD_API_port.c` file in AN0159-SC01) is built for a generic HD44780-based LCD module. However, the `LCD_init()` routine contains instructions to specifically initialize a 16x2 LCD module.

The `LCD_init()` routine is called before any other API. The sequence of start-up events for the LCD module with the eZ80Acclaim! MCU is listed below:

1. Initialize the microcontroller's GPIO pins using `LCD_gpio_set()`.
2. Initialize Timer0 for a 1 ms delay in CONTINUOUS mode using `init_timer0`.
3. Enable interrupts using the `EI` macro.
4. Provide a power-on delay, minimum 15 ms, for LCD warm up using `delayms()`.
5. Write instructions sequentially to the LCD module using `LCD_init`. For more information, see flowchart displayed in Appendix A—Flowchart on page 8.
6. Turn off the cursor blink function using `LCD_blink_off`.

The GPIO pins used for the LCD interface are defined in the `LCD_API.h` header file (refer to, AN0159-SC01). The timer, Timer0, is used in CONTINUOUS mode with a timeout of 1 ms to provide the required delay.

Upon initialization, displaying text on the LCD Module is achieved by performing the following instructions.

To print the ASCII character *Z* to the fifth column of the first row of the LCD Module:

1. Use `LCD_setposition(0,5)` to set the print location as the fifth column of the first row.
2. Using `LCD_printc('Z')`, call the print API with the *Z* character as the argument.

To print the ASCII character string *Zilog*:

1. Use `LCD_clear` to clear the screen.
2. Print the string using `LCD_prints('Zilog')`.

To print a user-defined character:

1. Define the format of special characters in `defchar.h` header file as 8 bytes per character.
2. The HD44780 controller's CGRAM can hold a maximum of 64 bytes, which limits the number of characters to 8. Figure 4 displays a construction pattern for a sample character. The character is formed using 7 rows containing 5 pixels each. The bottom row represents the underline cursor.
3. Copy the character pattern table to a CGRAM location using `LCD_copymap()`. For example: `0x03`.
4. Print the special character using `LCD_printc_user(0x03)`.

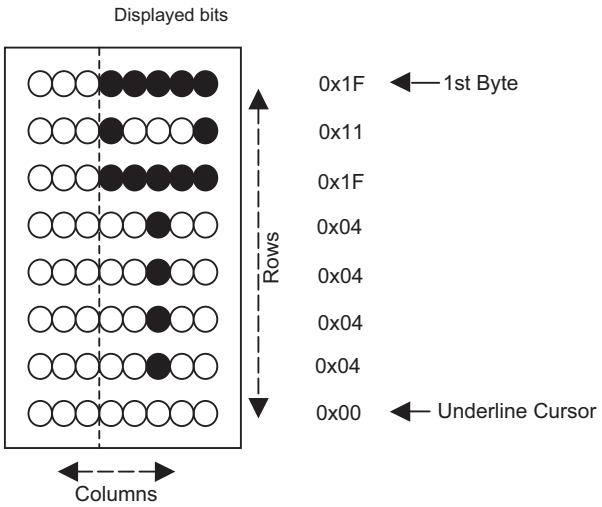

Figure 4. Construction Pattern for a Sample Character

To replace the character before the current cursor position with a new character:

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

To insert a character in the middle of a string:

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

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)`.

Project Build for Interfacing the LCD Module

To implement the software for a project utilizing an eZ80Acclaim! GPIO port, add the following files to a new ZDS II project:

- `LCD_API_port.c` (for API function definitions using port access)
- `LCD_API.h` (for API prototype declarations for port access)
- `LCD_test.c` (for a main routine that contains a user application)
- `defchar.h` (for special user-defined character patterns)
- `eZ80F91.h` (for ZDS II, available with the ZDS II Compiler definitions)

A typical initialization routine flow for the HD44780 controller on a 16x2 LCD Module is displayed in Figure 5 on page 8. Initialization routines are usually specified in the LCD manufacturer's data sheets.

In addition to the driver APIs, the software implementation contains support routines that are used to read or write data to and from the LCD controller. These routines are utilized by the APIs only. A listing of these support routines appears in Appendix B—Software Device Driver API Description on page 9.

Testing

Equipment Used

The following equipment are used for testing:

- Zilog's eZ80Acclaim Development Kit (eZ80F910200ZCO)
- Zilog's eZ80Acclaim*Plus!*™ Development Kit (eZ80F910300ZCOG)
- Zilog's eZ80Acclaim Module (eZ80F915050MODG)
- Zilog's eZ80Acclaim*Plus!* Module (eZ80F916050MODG)
- ZDS II IDE for eZ80Acclaim! v4.11.0

The LCD Module is connected to the eZ80Acclaim! MCU as displayed in Figure 2 on page 4. The GPIO pins are set up as per the configuration in the `LCD_API.h` header file.

Note: *It is important to apply proper voltage to the GND, V_{CC}, and V_L (pins 1, 2, and 3, respectively, as in Table 1 on page 2) of the LCD Module.*

The results are as expected and the program operates in stable fashion, with the LCD displaying the text messages at appropriate locations with timed delays in between.

Summary

There are different ways to implement a Character LCD interface using microcontrollers available in the market today. This Application Note demonstrates signal and data pin interfacing between eZ80Acclaim! MCUs and a HD44780-based Character LCD Module, using the accompanying driver software. The driver software is a ready-to-use library of APIs that enables a you to successfully integrate a general character LCD Module with the eZ80Acclaim! family of microcontrollers.

References

The documents associated with ZDS II and eZ80F91 available on www.zilog.com are provided below:

- HD44780U (LCD II); www.hitachi.com
- eZ80F91 MCU Product Specification (PS0192)
- eZ80F91 Module Product Specification (PS0193)
- eZ80F91 ASSP Product Specification (PS0270)

- Zilog Developer Studio II–eZ80Acclaim! User Manual (UM0144)
- eZ80 CPU User Manual (UM0077)

Appendix A—Flowchart

Figure 5 displays the flowchart for Initializing HD44780 Controller of a 16x2 LCD Module.

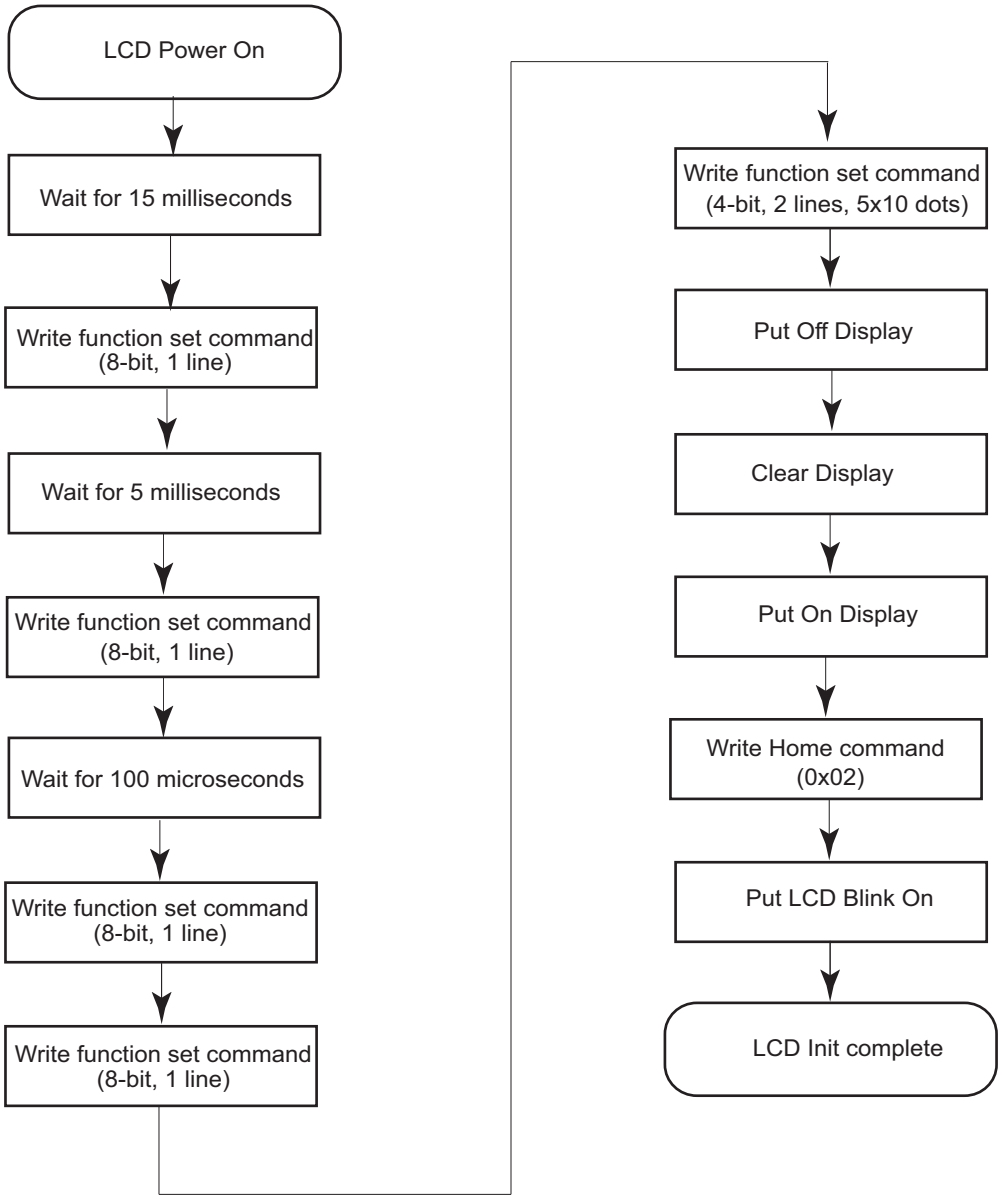

Figure 5. Initializing the HD44780 Controller on a 16x2 LCD Module

Appendix B—Software Device Driver API Description

This appendix lists descriptions for each device driver API and the API support functions.

Device Driver APIs

The tables that follow provide brief descriptions of each of the Device Driver APIs.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

API Support Functions

The tables that follow list brief descriptions of each of the Device Driver support functions.

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

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

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

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

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