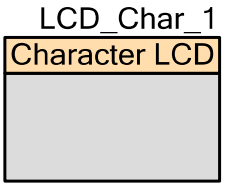
Character LCD
1.30



Features

- Implements the industry standard Hitachi HD44780 LCD display driver chip protocol
- Requires only seven I/O pins on one I/O port
- Built in character editor to create user defined custom characters
- Horizontal and vertical bar graphs

General Description

The Character LCD component contains a set of library routines that enable simple use of one, two, or four line LCD modules that follow the Hitachi 44780 standard 4-bit interface. The component provides APIs to implement horizontal and vertical bar graphs or you can create and display your own custom characters.

When to use a Character LCD

Use the Character LCD component to display text data to the user of the product or to a developer during design and debug.

Input/Output Connections

This section describes the various input and output connections available for the Character LCD.

LCD_Port – Pin Editor

The LCD uses 7 consecutive pins of a physical port. In order to place the Character LCD onto your desired port, use the Design-Wide Resources Pin Editor. The Pin Editor enables the placement of this component's digital port on any free output port.

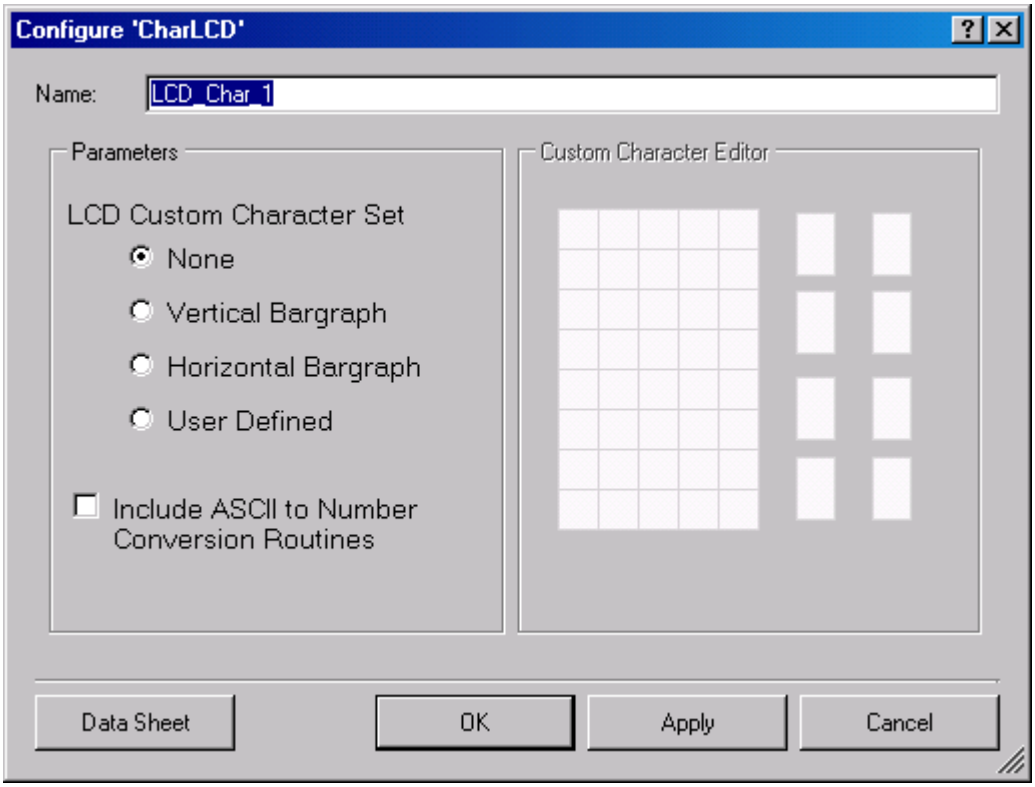
Note The 7 pins can be placed to start at either Pin 1 or Pin 0 of the selected port, but it may not span ports. These pins are for the exclusive use of the LCD port and cannot be used for any other purpose.

No direct access to the Character LCD's port is needed as the software APIs handle all reads and writes for you. The pin connections between an LCD module and a PSoC logical port are detailed in Functional Description.

Parameters and Setup

Drag a Character LCD component onto your design and double-click it to open the Configure dialog.

Figure 1 Configure LCD_Char Dialog



Parameters

A parameter enables the selection of the following options:

- None (Default) – Don't do anything with custom characters.
- Vertical Bar Graph – Generate the custom characters and API to manipulate a vertical bar graph.
- Horizontal Bar Graph – Generate the custom characters and API to manipulate a Horizontal bar graph
- User Defined Characters – Enable the user to create custom characters and API to manipulate them

Whether they are user defined or bar graphs generated by PSoC Creator, every set of custom characters needs to be loaded into the LCD. Once the component has loaded in the characters, the function LCD_Char_PutChar() and the custom character constants (from the header file) can be used to display them.

Conversion Routines

Selecting to include the ASCII conversion routines adds several API functions to the generated code. (See API table or function descriptions for more info on these routines.)

Custom Character Editor

The Custom Character Editor makes user defined character sets easy to create through the use of a GUI. Each of the eight characters can be up to a 5x8 pixels, though some hardware may not display more than the top 5x7.

To use the Custom Character Editor, select “User Defined” as the option for the LCD Custom Character Set. Then, click on the thumbnail of the character you want to edit. To toggle a pixel in your character, simply click the mouse down on the chosen pixel in the enlarged character view. The pixel will toggle; if you click and drag, any pixel you move the mouse over will be set to the same state of the original pixel.

Clock Selection

Not Applicable.

Placement

Not Applicable. This is a software component.

Resources

None.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function together with related constants provided by the "include" files. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "LCD_Char_1" to the first instance of a component in a given project. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "LCD_Char".

Functions	Description
LCD_Char_Start()	Start the module
LCD_Char_Stop()	Does nothing. Provided for consistency.
LCD_Char_DisplayOn()	Turn on the LCD module's display
LCD_Char_DisplayOff()	Turn off the LCD module's display
LCD_Char_PrintString(char8 * string)	Prints a null-terminated string to the screen character by character
LCD_Char_PutChar(char8 character)	Send a single character to the data register of the LCD module at the current position.
LCD_Char_Position(uint8 row, uint8 column)	Set the cursor's position to match the row and column supplied
LCD_Char_WriteData(uint8 value)	Write a single byte of data to the LCD module data register
LCD_Char_WriteControl(uint8 command)	Write a single byte instruction to the LCD module control register
LCD_Char_ClearDisplay()	Clears the data from the LCD module's screen
LCD_Char_IsReady()	Polls LCD until the ready bit is set

The following optional functions are included when needed by your selection of a custom font type. The function LCD_Char_LoadCustomFonts() comes with every custom font set, be they user-defined or PSoC Creator generated. It loads the user-defined or the bar graph characters into the LCD hardware. The draw bar graph commands are generated when a bar graph is selected and enable the easy, dynamic adjustment of bar graphs.

Optional Custom Font Functions	Description
LCD_Char_LoadCustomFonts()	Loads custom characters into the LCD module
LCD_Char_DrawHorizontalBG(uint8 row, uint8 column, uint8 maxCharacters, uint8 value)	Draw a horizontal bar graph. Only available when a bar graph character set has been selected.
LCD_Char_DrawVerticalBG(uint8 row, uint8 column, uint8 maxCharacters, uint8 value)	Draw a vertical bar graph. Only available when a bar graph character set has been selected.

The following optional functions are included when needed by your selection:

Optional Number to ASCII Conversion Routines	Description
LCD_Char_PrintInt8(uint8 value)	Print a 2-ASCII character hex representation of the 8-bit value to the Character LCD module.
LCD_Char_PrintInt16(uint16 value)	Print a 4-ASCII character hex representation of the 16-bit value to the Character LCD module.
LCD_Char_PrintNumber(uint16 value)	Print the decimal value of a 16-bit value as left justified ASCII characters

void LCD_Char_Start(void)

Description: This function initializes the LCD hardware module as follows:
- Enable 4-bit interface
- Clear the display
- Enable auto cursor increment
- Resets the cursor to start position

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_Stop(void)

Description: Provided for consistency with other components, this API has no function.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_PrintString(char8 * string)

Description: Write a null-terminated string of characters to the screen beginning at the current cursor location.

Parameters: char8 * string: Null-terminated array of ASCII characters to be displayed on the LCD module's screen.

Return Value: None

Side Effects: None

void LCD_Char_PutChar(char8 character)

Description: Write an individual character to the screen at the current cursor location. Used to display custom characters through their named values. (LCD_Char_CUSTOM_0 through LCD_Char_CUSTOM_7)

Parameters: char8 character: ASCII character to be displayed on the LCD module's screen.

Return Value: None

Side Effects: None

void LCD_Char_Position(uint8 row, uint8 column)

Description: Move cursor to location specified by arguments row and column.

Parameters: uint8 row: The row number at which to position the cursor. Minimum value is zero.

uint8 column: The column number at which to position the cursor. Minimum value is zero.

Return Value: None

Side Effects: None

void LCD_Char_WriteData(uint8 value)

Description: This function writes the input argument, value, to the LCD's display data register.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_WriteControl(uint8 command)

Description: This function writes the input argument, command, to the LCD's control register. Several commands are included in the Character LCD header file. Review the specific LCD datasheet for commands valid for that model.

Parameters: command: 8-bit value representing valid command to be loaded into the command register of the LCD module.

Return Value: None

Side Effects: None

void LCD_Char_ClearDisplay(void)

Description: This macro clears the contents of the screen and resets the cursor location to be row zero and column zero. It calls function LCD_Char_WriteControl with the appropriate argument to activate the display.

Parameters: None

Return Value: None

Side Effects: Cursor position reset to 0,0.

void LCD_Char_IsReady(void)

Description: Polls LCD until the ready bit is set

Parameters: None

Return Value: None

Side Effects: Changes pins to High-Z.

void LCD_Char_DisplayOff(void)

Description: This macro turns the display off, but does not reset the LCD module in anyway. It calls function LCD_Char_WriteControl with the appropriate argument to deactivate the display.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_DisplayOn(void)

Description: This macro turns the display on, *without* initializing it. It calls function LCD_Char_WriteControl with the appropriate argument to activate the display.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_LoadCustomFonts(const uint8 * customData)

Description: This function loads 8 custom characters (bar graph or user defined fonts) into the LCD module to use the custom fonts during runtime. Only available if a custom character set was selected in the customizer. This function must be called prior to the use of custom characters, be they bar graph or user defined characters.

Parameters: Const uint8 * customData: Pointer to the head of an array of bytes. Array should be 64 bytes long as 5x8 characters require 8 bytes per character.

Return Value: None

Side Effects: Overwrites any previous custom characters which may have been stored in the LCD module.

void LCD_Char_DrawHorizontalBG(uint8 row, uint8 column, uint8 maxCharacters, uint8 value)

Description: Draw a horizontal bar graph. Only available if a horizontal or vertical bar graph was selected. Bar graph characters must be loaded into the LCD first using `LoadCustomFonts(customData);`

Parameters: uint8 row: The row of the first character in the bar graph.

uint8 column: The column of the first character in the bar graph.

uint8 maxCharacters: Number of whole characters the bar graph consumes. Represents height or width depending upon the user's bar graph selection. Each character is 5 pixels wide and 8 pixels high.

uint8 value: number of shaded pixels to draw. May not exceed total pixel length (height) of the bar graph.

Return Value: None

Side Effects: None

void LCD_Char_DrawVerticalBG(uint8 row, uint8 column, uint8 maxCharacters, uint8 value)

Description: Draw a vertical bar graph. Only available if a horizontal or vertical bar graph was selected. Bar graph characters must be loaded into the LCD first using LoadCustomFonts(customData);

Parameters: uint8 row: The row of the first character in the bar graph.

uint8 column: The column of the first character in the bar graph.

uint8 maxCharacters: Number of whole characters the bar graph consumes. Represents height or width depending upon the user's bar graph selection. Each character is 5 pixels wide and 8 pixels high.

uint8 value: number of shaded pixels to draw. May not exceed total pixel length (height) of the bar graph.

Return Value: None

Side Effects: None

void LCD_Char_PrintInt8 (uint8 value)

Description: Print a two ASCII character representation of the 8-bit value to the Character LCD module.

Parameters: uint8 value: The 8-bit value to be printed in hexadecimal ASCII characters.

Return Value: None

Side Effects: None

void LCD_Char_PrintInt16 (uint16 value)

Description: Print a four ASCII character representation of the 16-bit value to the Character LCD module.

Parameters: uint16 value: The 16-bit value to be printed in hexadecimal ASCII characters.

Return Value: None

Side Effects: None

void LCD_Char_PrintNumber(uint16 value)

Description: Print the decimal value of a 16-bit value as left justified ASCII characters

Parameters: uint16 value: The 16-bit value to be printed in ASCII characters as a decimal number.

Return Value: None

Side Effects: None

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the Character LCD component. This example assumes the component has been placed in a design with the default name "LCD_Char_1."

Note If you renamed your component you must also edit the example code as appropriate to match the component name you specified; otherwise, this example code will not work.

```
#include <device.h>
#include "LCD_Char_1.h"

uint8 count8 = 0, spaces = 0;
uint8 numColumns = 20; /* Assign the correct number of columns */

void main()
{
    LCD_Char_1_Start();

    while(1)
    {
        for(count8 = 0; count8 < numColumns; count8++)
        {
            /* First Row: Scroll words right on the screen */
            for(spaces = 0; spaces < count8; spaces++)
            {
                LCD_Char_1_Position(0, spaces);
                LCD_Char_1_PutChar(' ');
            }

            LCD_Char_1_Position(0, count8);
            LCD_Char_1_PrintString("Hello World");

            /* Second Row : Move second row text back and forth once during
             * the scrolling of the first row
             */
            LCD_Char_1_Position(1,0);

            if(count8 % 2)
                LCD_Char_1_PrintString("LCD Module Test ");
            else
                LCD_Char_1_PrintString(" LCD Module Test");

            /* Add delay to make display more readable - adjust as needed */
            CyDelay(50);
        }

        /* Clear first row and restart scroll */
        for(spaces = 0; spaces < numColumns; spaces++)
        {
            LCD_Char_1_Position(0, spaces);
            LCD_Char_1_PutChar(' ');
        }
    }
}
```

Interrupt Service Routine

Not Applicable.

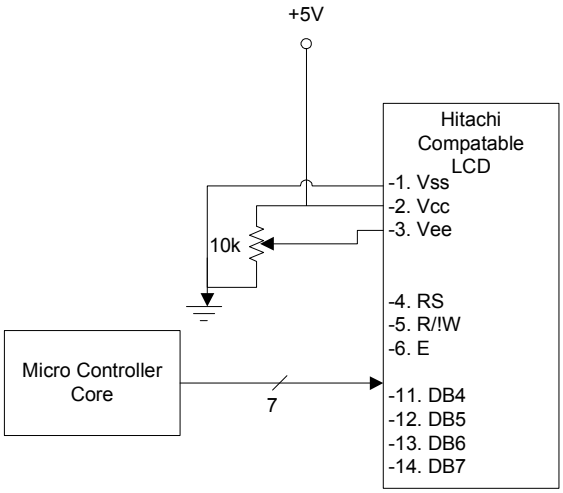
Functional Description

The LCD module provides a visual display for alphanumeric characters as well as limited custom fonts. The APIs configure the PSoC device as necessary to easily interface between the standard Hitachi LCD display driver and the PSoC device.

The following table describes the LCD logical port pin to physical LCD module pin mapping. It should be noted that the LCD's Logical port can be mapped to start on the first or second physical pin of a port; it may not span ports. That is, LogicalPort_0 could theoretically be Port 2, Pin 0 or Port 2, Pin 1. Using the pin editor to force the LCD logical port to begin at pin 0 improves efficiency by reducing the number of shifts required to align data for a write.

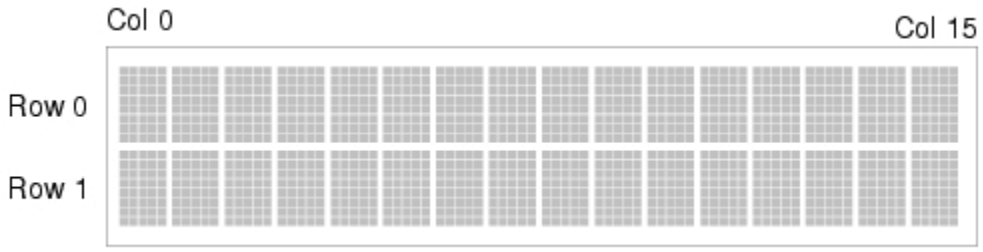
Logical Port Pin	LCD Module Pin	Description
LogicalPort_0	DB4	Data Bit 0
LogicalPort_1	DB5	Data Bit 1
LogicalPort_2	DB6	Data Bit 2
LogicalPort_3	DB7	Data Bit 3
LogicalPort_4	E	LCD Enable (Strobe to confirm new data available)
LogicalPort_5	RS	Register Select (Select Data or Control input data)
LogicalPort_6	R/!W	Read/not Write (Toggle for polling the ready bit of the LCD)

Figure 2 Pin Editor Diagram



The LCD_Char_Position function handles addressing of the display as follows. Row zero, column zero is in the upper left corner with the column number increasing to the right. In a four line display, it should be noted that writing beyond the column 19 of row 0 can result in row 2 being corrupted as the addressing maps row 0, column 20 to row 2, column 0. This is not an issue in the standard 2x16 Hitachi module.

Figure 3 2x16 Hitachi LCD Module



Block Diagram and Configuration

Not Applicable

References

Not Applicable

DC and AC Electrical Characteristics

Not Applicable

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.30	Replaced Digital Port component with a Pins component in the Char LCD schematic.	Old Digital Port was deprecated and was replaced with the new Pins component.
	Added description of IsReady function.	There was no mention about IsReady() in the old version of data sheet.
	Deleted the function DelayUS and replaced it with CyDelay() or IsReady().	This was to address a timing issue with the component which caused some failures.
1.20.a	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
1.20	Updated Symbol	To comply with corporate standard.
1.10.a	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
1.10	Various updates from version 0.2	0.2 version was included with alpha builds, but was not a completely functional component.