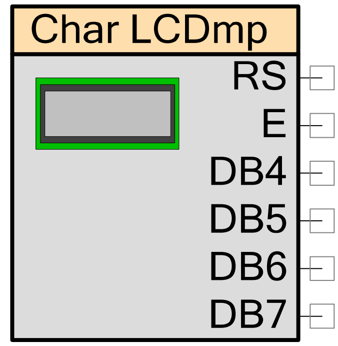
Character LCD (mp)

1.00



Features

- Implements the industry-standard Hitachi HD44780 LCD display driver chip protocol
- Requires only six I/O pins.
- Pins do not have to be sequential or be on the same port.
- Contains built-in character editor to create user-defined custom characters
- Supports horizontal and vertical bar graphs

General Description

The Character LCD (mp) 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. This LCD component allows the user to connect the signals across Multiple Ports (mp).

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 output connections available for the Character LCD component.

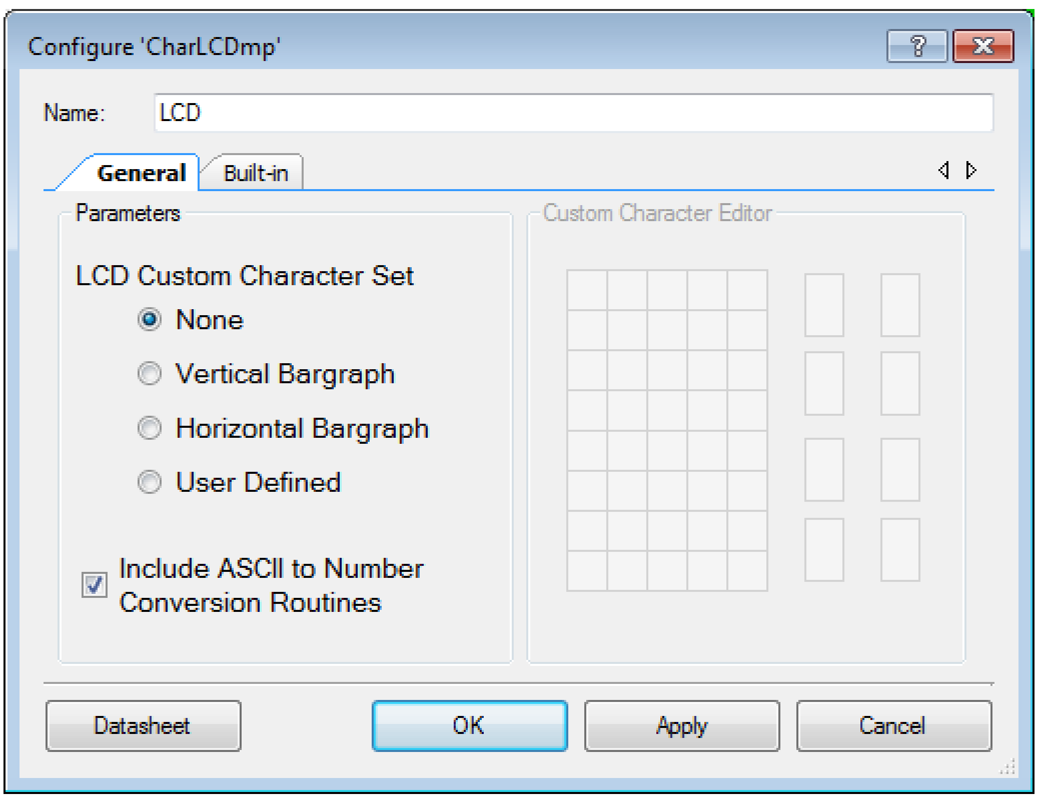
Output	Description
RS	Selects between the command and data registers in the LCD.
E	Used to strobe the data during a write.

DB4	LCD signal used for data bits D4 and D0.
DB5	LCD signal used for data bits D5 and D1.
DB6	LCD signal used for data bits D6 and D2.
DB7	LCD signal used for data bits D7 and D3.

The LCD component uses six GPIO pins that may be placed on any port and any order. There is no need to use sequential pins or the same port for all pins. To place the Character LCD (mp) into you design, simply connect the all output signals to Digital Output Pins. Use the Design-Wide Resource Editor to assign the pins to specific port pins.

Component Parameters

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



Parameters

LCD Custom Character Set

This parameter enables the selection of the following options:

- **None** (Default) – Do not do anything with custom characters.
- **Vertical Bar Graph** – Generate custom characters and API to manipulate a vertical bar graph.
- **Horizontal Bar Graph** – Generate custom characters and API to manipulate a horizontal bar graph.
- **User Defined** –Create custom characters and API to manipulate them.

After the component has loaded in the characters, the LCD_Char_PutChar() function and the custom character constants (from the header file) can be used to display them.

Conversion Routines

Selecting the **Include ASCII to Number Conversion Routines** option adds several API functions to the generated code. (Refer to the API table or function descriptions for more information about 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 8 characters can be up to 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, click on the chosen pixel in the enlarged character view. You may also click and drag to toggle multiple pixels.

After creating a custom character set, the GUI will generate a look-up array of eight custom characters. Then the look-up array can be loaded to a LCD module. By default, the LCD_Char_Start() routine loads custom characters if any were selected or created.

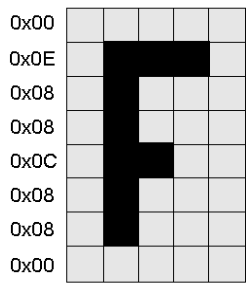
The component's functionality allows you to create custom character sets in the code and load them at run time. In that case, the last loaded character set overwrites the previous one and becomes active. To restore the original custom character set that you created using the component's GUI, you must add the following line to the top of your source code:

```
extern uint8 const CYCODE LCD_Char_customFonts[];
```

At run time, LCD_Char_LoadCustomFonts() can use that code as a parameter to load the original character set to the LCD module.

Figure 1 shows a custom character encoded into an 8-byte custom character lookup array row.

Figure 1. Custom Character Encoding




```
            Custom character «F»:
{0x00, 0x0E, 0x08, 0x08, 0x0C, 0x08, 0x08, 0x00}
```

As shown in the diagram, each row of a character is encoded as a single byte, from which only the five least-significant bits are used. The top row of the first character is encoded in the first byte of the custom font array. The next row of the first character is the second byte in the array. The first row of the second character is the ninth byte in the array, and so on. The entire custom font array consists of eight custom characters, creating a total array size of 64 bytes.

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()	Starts the module and loads custom character set to LCD, if it was defined.
LCD_Char_Stop()	Turns off the LCD
LCD_Char_DisplayOn()	Turns on the LCD module's display
LCD_Char_DisplayOff()	Turns off the LCD module's display
LCD_Char_PrintString()	Prints a null-terminated string to the screen, character by character
LCD_Char_PutChar()	Sends a single character to the LCD module data register at the current position.

Functions	Description
LCD_Char_Position()	Sets the cursor's position to match the row and column supplied
LCD_Char_WriteData()	Writes a single byte of data to the LCD module data register
LCD_Char_WriteControl()	Writes 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()	This function is provided only for compatibility with the standard Character LCD Component.
LCD_Char_Sleep()	Prepares component for entering sleep mode
LCD_Char_Wakeup()	Restores components configuration and turns on the LCD
LCD_Char_Init()	Performs initialization required for component's normal work
LCD_Char_Enable()	Turns on the display
LCD_Char_SaveConfig()	Empty API provided to store any required data prior entering to a Sleep mode.
LCD_Char_RestoreConfig()	Empty API provided to restore saved data after exiting a Sleep mode.

The following optional functions are included, when needed, if a user-selected custom font is selected. The LCD_Char_LoadCustomFonts()function comes with every custom font set, whether it is user-defined or PSoC Creator generated. The LCD_Char_LoadCustomFonts() function can be used to load the user-defined or the bar graph characters into the LCD hardware. If loading custom fonts created by the tool, you will need to import a pointer to the custom font to your project prior to using this function (refer to the description of LCD_Char_LoadCustomFonts()). By default, the LCD_Char_Init() routine loads the user-selected custom font. 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()	Draws a horizontal bar graph. Only available when a bar graph character set has been selected.
LCD_Char_DrawVerticalBG()	Draws 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()	Prints a two-ASCII-character hex representation of the 8-bit value to the Character LCD module.
LCD_Char_PrintInt16()	Prints a four-ASCII-character hex representation of the 16-bit value to the Character LCD module.

LCD_Char_PrintNumber()	Prints 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:

- Enables 4-bit interface
- Clears the display
- Enables auto cursor increment
- Resets the cursor to start position

It also loads a custom character set to LCD if it was defined in the customizer's GUI.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_Stop(void)

Description: Turns off the display of the LCD screen.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_PrintString(char8 * string)

Description: Writes 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: Writes 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: Moves the cursor to the 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 dByte)

Description: Writes data to the LCD RAM in the current position. Upon write completion, the position is incremented or decremented depending on the entry mode specified.

Parameters: dByte: A byte value to be written to the LCD module.

Return Value: None

Side Effects: None

void LCD_Char_WriteControl(uint8 cByte)

Description: Writes a command byte to the LCD module. Different LCD models can have their own commands. Review the specific LCD datasheet for commands valid for that model.

Parameters: cByte: 8-bit value representing the command to be loaded into the command register of the LCD module. Valid command parameters are specified in the table below:

Value	Description
LCD_Char_CLEAR_DISPLAY	Clear display
LCD_Char_RESET_CURSOR_POSITION LCD_Char_CURSOR_HOME	Return cursor and LCD to home position. (Wait at least 1.5 mSec before another LCD function is called after executing either of these two commands.)
LCD_Char_CURSOR_LEFT	Set left cursor move direction
LCD_Char_CURSOR_RIGHT	Set right cursor move direction
LCD_Char_DISPLAY_CURSOR_ON	Enable display and cursor
LCD_Char_DISPLAY_ON_CURSOR_OFF	Enable display, cursor off
LCD_Char_CURSOR_WINK	Enable display, cursor off, set cursor wink
LCD_Char_CURSOR_BLINK	Enable display and cursor, set cursor blink
LCD_Char_CURSOR_SH_LEFT	Move cursor/Shift display left
LCD_Char_CURSOR_SH_RIGHT	Move cursor/shift display right
LCD_Char_DISPLAY_2_LINES_5x10	Set display to be 2 lines 10 characters

Return Value: None

Side Effects: None

void LCD_Char_ClearDisplay(void)

Description: Clears the contents of the screen and resets the cursor location to be row and column zero. It calls 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: This is a null function that is only provided for compatibility with the standard Character LCD component.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_DisplayOff(void)

Description: Turns the display off, but does not reset the LCD module in any way. 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: 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_Sleep(void)

Description: This is the preferred routine to prepare the component for sleep. The LCD_Char_Sleep() routine saves the current component state. Then it calls the LCD_Char_Stop() function and calls LCD_Char_SaveConfig() to save the hardware configuration.

Call the LCD_Char_Sleep() function before calling the CyPmSleep() or the CyPmHibernate() function. Refer to the PSoC Creator *System Reference Guide* for more information about power management functions.

Parameters: None

Return Value: None

Side Effects: Doesn't change component pins' drive modes. Use Port Component APIs for that purpose. Because Character LCD is an interface component that has its own protocol, you need to reinitialize the component after you have saved or restored component pin states.

void LCD_Char_Wakeup(void)

Description: Restores component's configuration and turns on the LCD.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_Init(void)

Description: Performs initialization required for the component's normal work. LCD_Char_Init() also loads the custom character set if it was defined in the Configure dialog.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_Enable(void)

Description: Turns on the display.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_SaveConfig(void)

Description: Empties API provided to store any required data prior to entering Sleep mode.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_RestoreConfig(void)

Description: Empties API provided to restore saved data after exiting Sleep mode.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Char_LoadCustomFonts(const uint8 * customData)

Description: Loads eight 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.

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 that may have been stored in the LCD module.

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

Description: Draws a horizontal bar graph. Only available if a horizontal or vertical bar graph was selected.

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 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: Draws a vertical bar graph. Only available if a horizontal or vertical bar graph was selected.

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 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: Prints 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: Prints 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: Prints 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

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

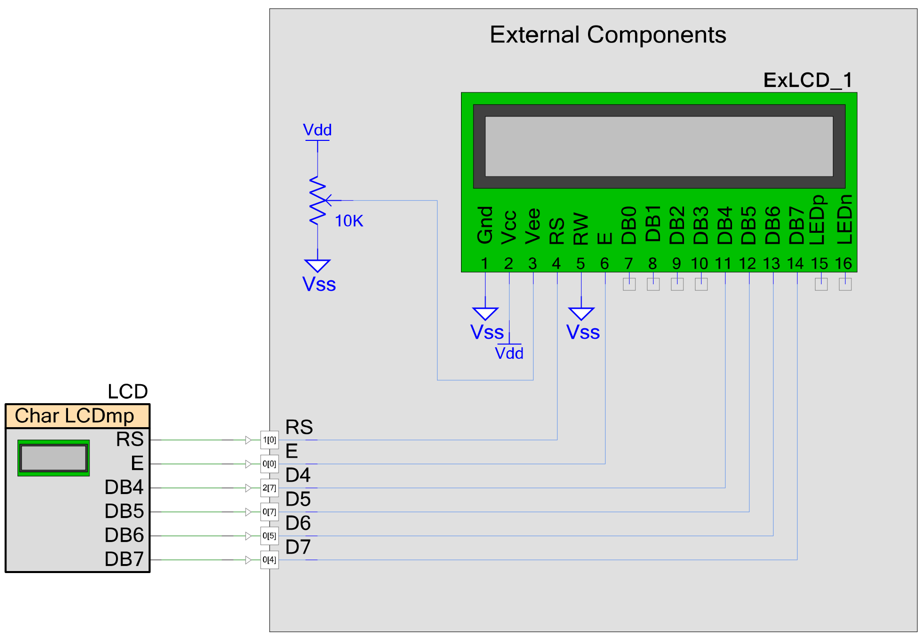
Refer to the "Find Example Project" topic in the PSoC Creator Help for more information.

Functional Description

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

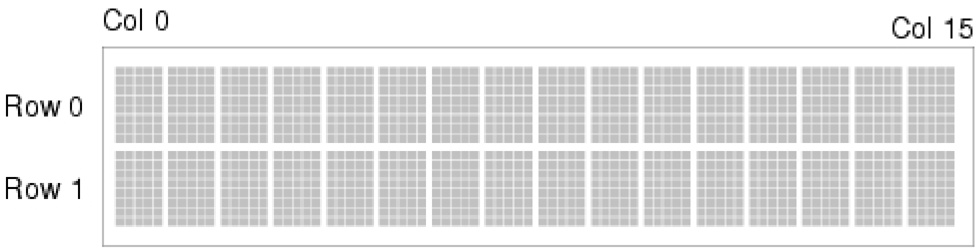
Logical Port Pin	LCD Module Pin	Description
Any GPIO	DB4	Data Bit 0
Any GPIO	DB5	Data Bit 1
Any GPIO	DB6	Data Bit 2
Any GPIO	DB7	Data Bit 3
Any GPIO	E	LCD Enable (strobe to confirm new data available)
Any GPIO	RS	Register Select (select data or control input data)
Vss	R/!W	Read/not Write (Not used)

Figure 2. Connections between component and actual LCD



The LCD_Char_Position() function manages display addressing 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, writing beyond column 19 of row 0 can result in row 2 being corrupted because 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



Resources

The Character LCD component uses 6 I/O pins and a control register.

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
None		3		6
Vertical		3		6
Horizontal		3		6
User Defined		3		6
None + Conversion Routines		3		6

DC and AC Electrical Characteristics

N/A

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.00	Initial Release	NA