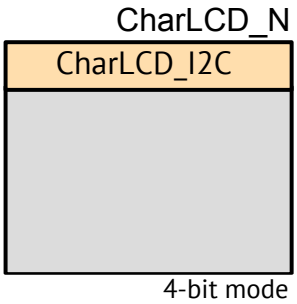
PSoC3, PSoC4 & PSoC5 Character LCD_I2C 4_bit mode v2.0



Features

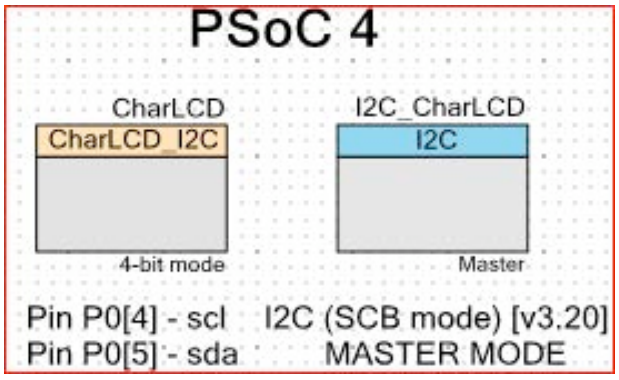
- Implements I2C interface with the PCF 8574AT I2C Adapter PC Board connected to the Industry Standard Hitachi HD44780 LCD controller in 4-bit mode
- Requires only 2 I/O pins of an I2C Master Component
- Contains built-in character editor to create user-defined custom characters
- Supports horizontal and vertical bar graphs

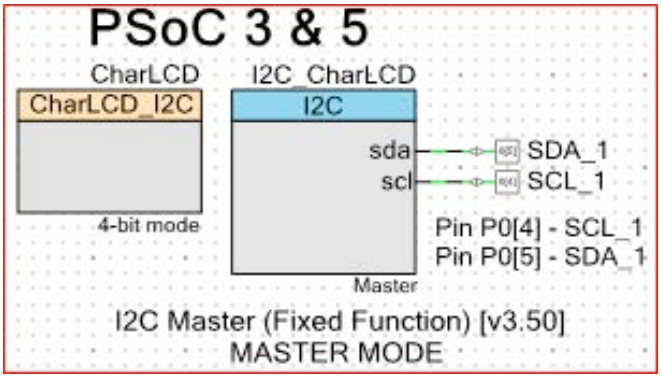
General Description

The CharLCD _I2C 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

IMPORTANT: The **CharLCD_I2C** component requires an **I2C Master Component** with the Instance Name I2C_**'instance name of the CharLCD_I2C component'** to provide the I/O. If the Instance Name for the **CharLCD_I2C Component** is **XYZ_1** then the Instance Name for the **I2C Master Component** must be I2C_**XYZ_1**. This prevents Building and Compiling errors.

For the **PSoC 4** use the **I2C (SBC mode)[v3.2]** Component set to Master Mode. For the PSoC 3 and PSoC 5 use the **I2C Master(Fixed Function)[v3.50]** Component.





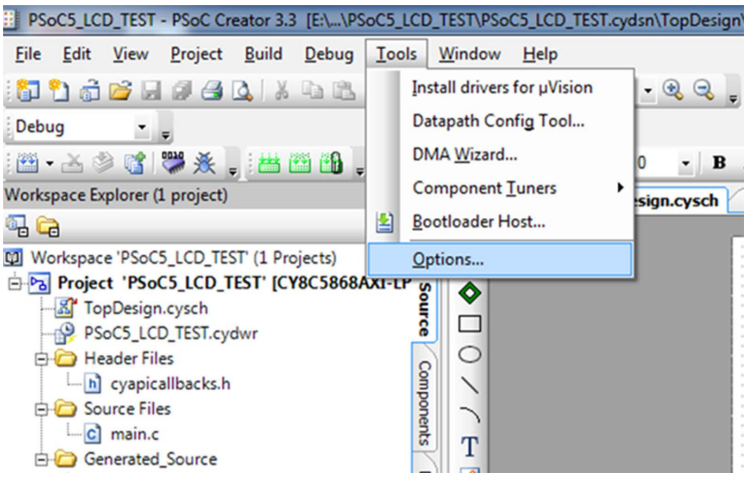
Remember that the I2C Master Component Start API statement needs to come before the CharLCD_I2C Start API statement in the main.c file.

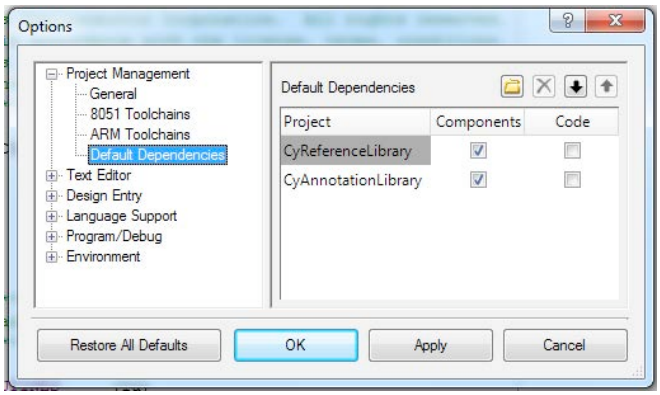
When to Use a CharLCD_I2C

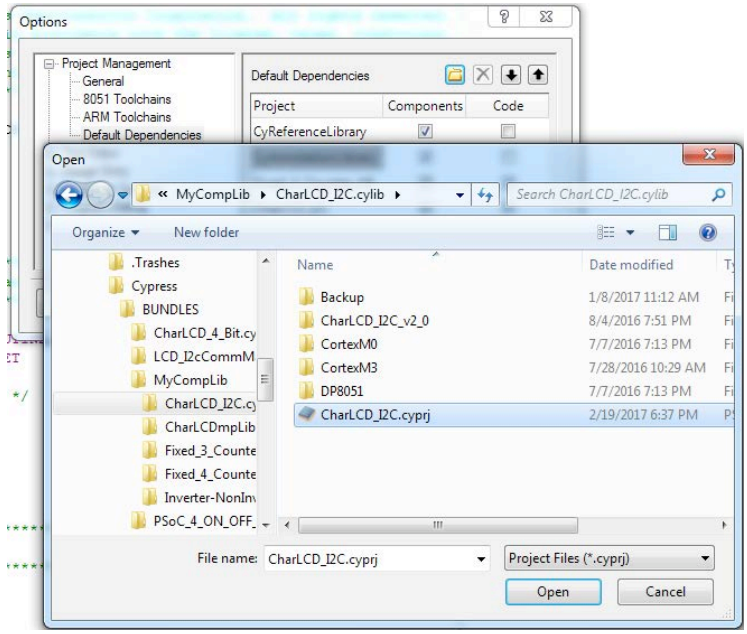
Use the CharLCD_I2C component to display text data to the user of the product or to a developer during design and debug only having to use 2 Data Lines conserving Pins.

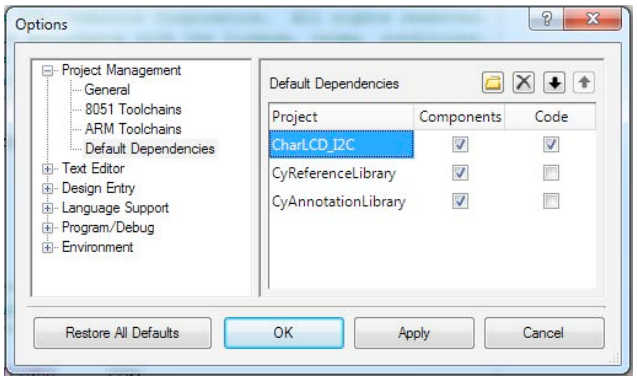
Loading Community Components Using Dependency Method

Go to the **Main Menu Bar of PsoC Creator** and Open the **Tools Menu**. Then open **Options.** Once the **Option Window** is open then expand the **Project Manager** tree and click on **Default Dependencies.** Now click on the **Folder** button and locate the Folder **CharLCD_I2C.cylib**. Open the folder and double click on the **CharLCD_I2C.cyprj** file. **CharLCD_I2C** should now appear in in the **Default Dependencies Window**. There should be a check mark in both the **Components Box** and **Code Box** of the **ChraLCD_I2C** Component. This will make the Component available for **all** your projects.



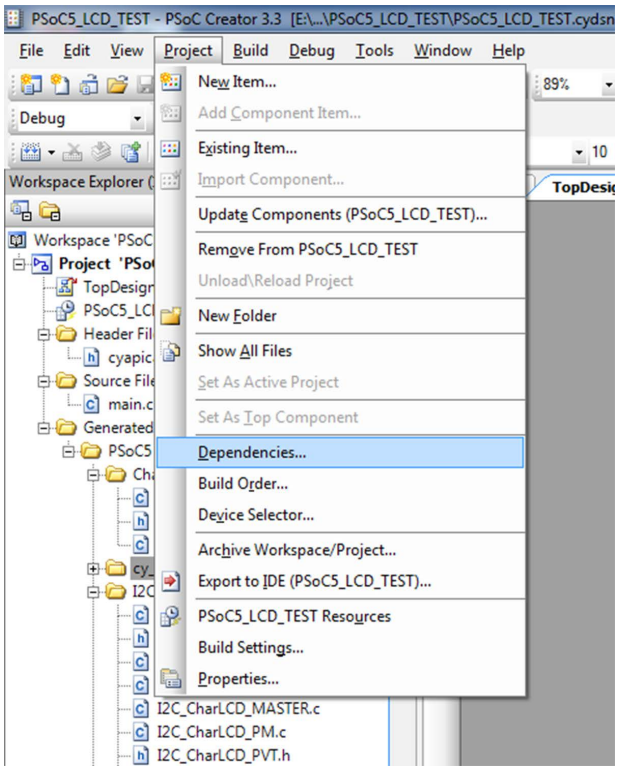


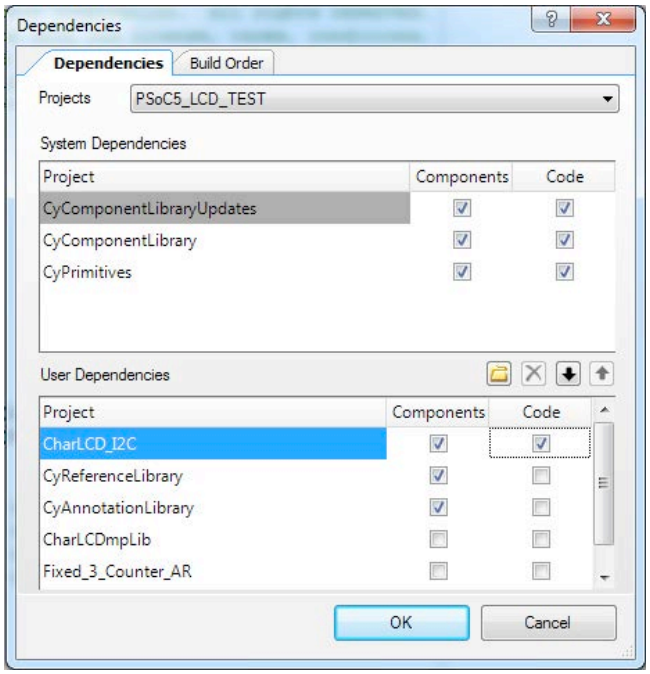


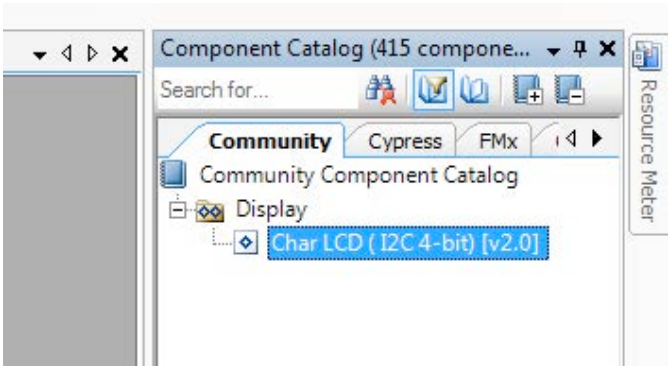


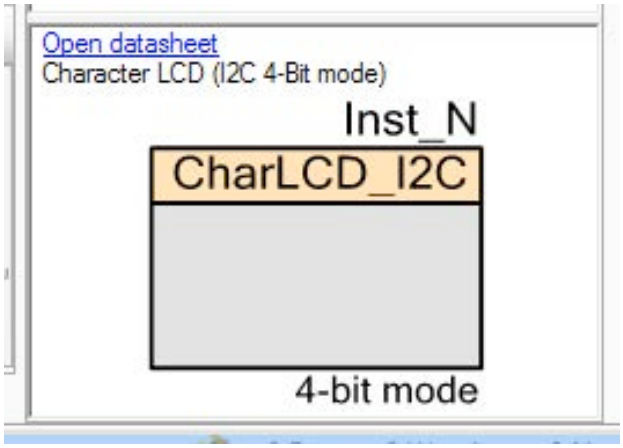
Making Available and Removing Community Components from Projects

Go to the **Main Menu Bar of PSoC Creato**r and open the **Poject Menu** and click on **Dependencies** which will topen he **Dependencies Window**. Locate and click on **CharLCD_I2C.** Make sure there is a check mark in the **Component Box** and the **Code Box** of the CharLCD_I2C component. It should appear under the **Community Tab** of the **Component Catalogue.** Now the **CharLCD_I2C** Component can be dragged into your project. The Component can be **removed** from a project by unchecking the **Components** and **Code** Boxes associated with the Component.









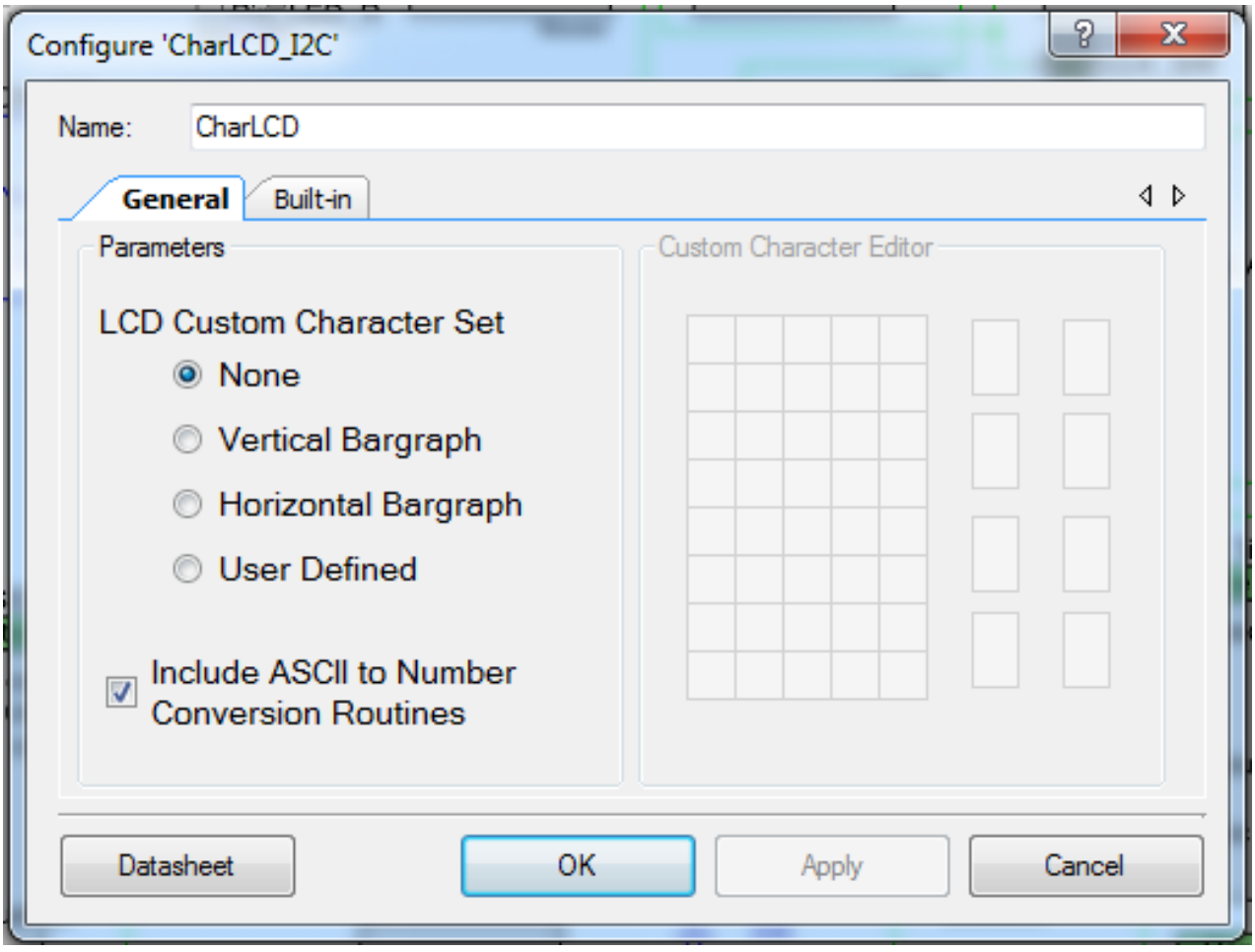
Input/Output Connections

This section describes the various output connections available for the CharLCD_I2C component.

Output	Description
SDA	I2C Data line of an I2C Master Component
SCL	I2C Clock Output of an I2C Master Component

Component Parameters

Drag a CharLCD_I2C 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 CharLCD_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 CharLCD_I2C_Start() routine loads custom characters if any were selected or created.

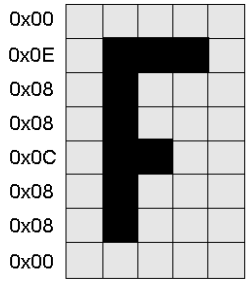
The component's functionality allows you to create custom character sets in the code and load them at run time. In that case, the last loaded character set overwrites the previous one and becomes active. To restore the original custom character set that you created using the component's GUI, you must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

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

At run time, CharLCD_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 "CharLCD_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 "CharLCD."

Functions	Description
CharLCD_Start()	Starts the module and loads custom character set to LCD, if it was defined.
CharLCD_Stop()	Turns off the LCD
CharLCD_DisplayOn()	Turns on the LCD module’s display
CharLCD_DisplayOff()	Turns off the LCD module’s display
CharLCD_PrintString()	Prints a null-terminated string to the screen, character by character
CharLCD_PosPrintString()	Prints a null-terminated string to the screen, at a specified position.

Functions	Description
CharLCD_PutChar()	Sends a single character to the LCD module data register at the current position.
CharLCD_PosPutChar()	Sends a single character to the LCD module data register at a specified position.
CharLCD_Position()	Sets the cursor's position to match the row and column supplied
CharLCD_WriteData()	Writes a single byte of data to the LCD module data register
CharLCD_WriteControl()	Writes a single-byte instruction to the LCD module control register
CharLCD_ClearDisplay()	Clears the data from the LCD module's screen
CharLCD_IsReady()	This function is provided only for compatibility with the standard Character LCD Component.
CharLCD_Sleep()	Prepares component for entering sleep mode
CharLCD_Wakeup()	Restores components configuration and turns on the LCD
CharLCD_Init()	Performs initialization required for component's normal work
CharLCD_Enable()	Turns on the display
CharLCD_SaveConfig()	Empty API provided to store any required data prior entering to a Sleep mode.
CharLCD_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 CharLCD_LoadCustomFonts()function comes with every custom font set, whether it is user-defined or PSoC Creator generated. The CharLCD_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 CharLCD_LoadCustomFonts()). By default, the CharLCD_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
CharLCD_LoadCustomFonts()	Loads custom characters into the LCD module
CharLCD_DrawHorizontalBG()	Draws a horizontal bar graph. Only available when a bar graph character set has been selected.
CharLCD_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
CharLCD_PrintInt8()	Prints a two-ASCII-character hex representation of the 8-bit value to the Character LCD module.
CharLCD_PrintInt16()	Prints a four-ASCII-character hex representation of the 16-bit value to the Character LCD module.
CharLCD_PrintNumber()	Prints the decimal value of a 16-bit value as left-justified ASCII characters
CharLCD_PosPrintInt8()	Prints a two-ASCII-character hex representation of the 8-bit value to the Character LCD module, at a specified position.
CharLCD_PosPrintInt16()	Prints a four-ASCII-character hex representation of the 16-bit value to the Character LCD module, at a specified position.
CharLCD_PosPrintNumber()	Prints the decimal value of a 16-bit value as left-justified ASCII characters, at a specified position.

void CharLCD_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 CharLCD_Stop(void)

Description: Turns off the display of the LCD screen.

Parameters: None

Return Value: None

Side Effects: None

void CharLCD_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 CharLCD_PosPrintString(uint8 row, uint8 col, char8 * string)

Description: Writes a null-terminated string of characters to the screen beginning at the specified location.

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

uint8 row: Row in which to print the string.

uint8 col: Column in which to start printing the string.

Return Value: None

Side Effects: None

void CharLCD_PutChar(char8 character)

Description: Writes an individual character to the screen at the current location. Used to display custom characters through their named values. (CharLCD_CUSTOM_0() through CharLCD_CUSTOM_7().

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

uint8 row: Row in which to print the string.

uint8 col: Column in which to start printing the string.

Return Value: None

Side Effects: None

void CharLCD_PosPutChar(uint8 row, uint8 col, char8 character)

Description: Writes an individual character to the screen at the specified location. Used to display custom characters through their named values. (CharLCD_CUSTOM_0() through CharLCD_CUSTOM_7().

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

uint8 row: Row in which to print the string.

uint8 col: Column in which to start printing the string.

Return Value: None

Side Effects: None

void CharLCD_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 CharLCD_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 CharLCD_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
CharLCD_CLEAR_DISPLAY	Clear display
CharLCD_RESET_CURSOR_POSITION CharLCD_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.)
CharLCD_CURSOR_LEFT	Set left cursor move direction
CharLCD_CURSOR_RIGHT	Set right cursor move direction
CharLCD_DISPLAY_CURSOR_ON	Enable display and cursor
CharLCD_DISPLAY_ON_CURSOR_OFF	Enable display, cursor off
CharLCD_CURSOR_WINK	Enable display, cursor off, set cursor wink
CharLCD_CURSOR_BLINK	Enable display and cursor, set cursor blink
CharLCD_CURSOR_SH_LEFT	Move cursor/Shift display left
CharLCD_CURSOR_SH_RIGHT	Move cursor/shift display right
CharLCD_DISPLAY_2_LINES_5x10	Set display to be 2 lines 10 characters

Return Value: None

Side Effects: None

void CharLCD_ClearDisplay(void)

Description: Clears the contents of the screen and resets the cursor location to be row and column zero. It calls CharLCD_WriteControl() with the appropriate argument to activate the display.

Parameters: None

Return Value: None

Side Effects: Cursor position reset to 0,0.

void CharLCD_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 CharLCD_DisplayOff(void)

Description: Turns the display off, but does not reset the LCD module in any way. It calls function CharLCD_WriteControl() with the appropriate argument to deactivate the display.

Parameters: None

Return Value: None

Side Effects: None

void CharLCD_DisplayOn(void)

Description: Turns the display on, *without* initializing it. It calls function CharLCD_WriteControl() with the appropriate argument to activate the display.

Parameters: None

Return Value: None

Side Effects: None

void CharLCD_Sleep(void)

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

Call the CharLCD_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 CharLCD_Wakeup(void)

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

Parameters: None

Return Value: None

Side Effects: None

void CharLCD_Init(void)

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

Parameters: None

Return Value: None

Side Effects: None

void CharLCD_Enable(void)

Description: Turns on the display.

Parameters: None

Return Value: None

Side Effects: None

void CharLCD_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 CharLCD_RestoreConfig(void)

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

Parameters: None

Return Value: None

Side Effects: None

void CharLCD_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 CharLCD_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 CharLCD_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 CharLCD_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 CharLCD_PosPrintInt8(uint8 row, uint8 col, uint8 value)

Description: Prints a two-ASCII-character representation of the 8-bit value to the Character LCD module at the specified location.

Parameters: uint8 value: The 8-bit value to be printed in hexadecimal ASCII characters.
uint8 row: Row in which to print the string.
uint8 col: Column in which to start printing the string.

Return Value: None

Side Effects: None

void CharLCD_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 CharLCD_PosPrintInt16(uint8 row, uint8 col, uint16 value)

Description: Prints a four-ASCII-character representation of the 16-bit value to the Character LCD module, at the specified location.

Parameters: uint16 value: The 16-bit value to be printed in hexadecimal ASCII characters.
uint8 row: Row in which to print the string.
uint8 col: Column in which to start printing the string.

Return Value: None

Side Effects: None

void CharLCD_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

void CharLCD_PosPrintNumber(uint8 row, uint8 col, uint16 value)

Description: Prints the decimal value of a 16-bit value as left-justified ASCII characters, at the specified position.

Parameters: uint16 value: The 16-bit value to be printed in ASCII characters as a decimal number.
uint8 row: Row in which to print the string.
uint8 col: Column in which to start printing the string.

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.

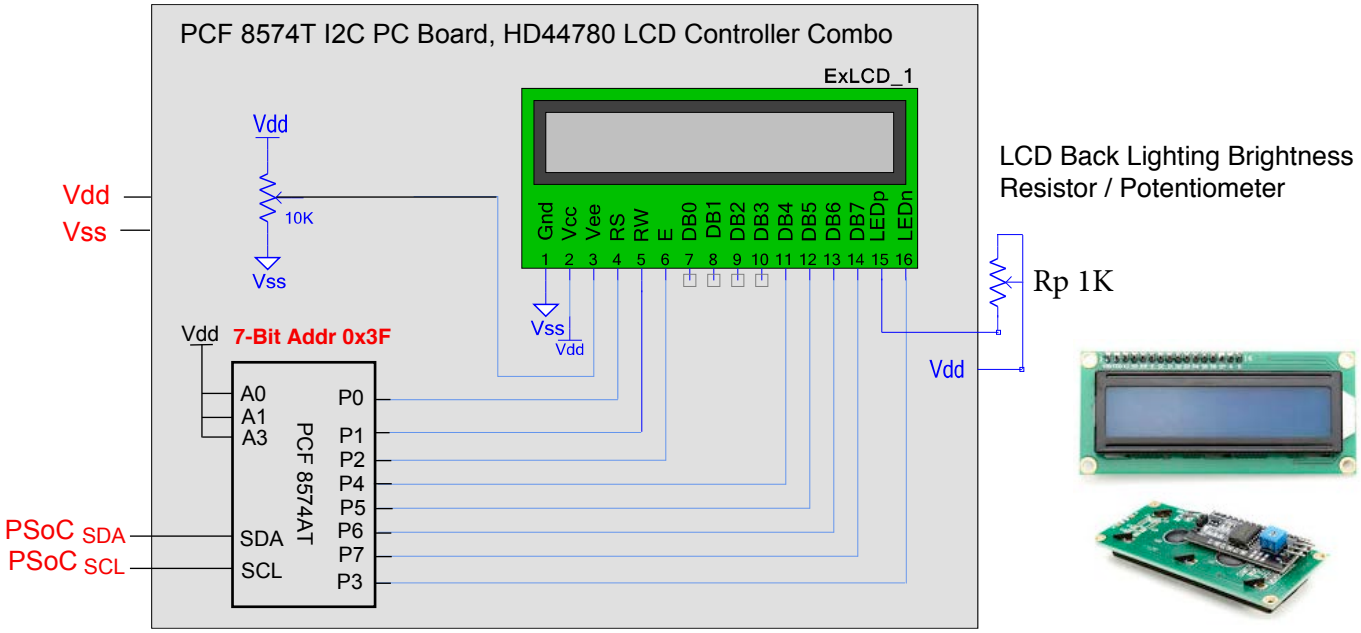
Functional Description

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 with the standard Hitachi HD44780 LCD display Controller using the SDA and SCL inputs of the PCF 8574AT I2C Adapter PC Board which can be soldered pin for pin directly to the LCD Controller board.

PCF8574T	LCD Module Pin	Description
P 7	DB7	Data Bit 7
P 6	DB6	Data Bit 6
P 5	DB5	Data Bit 5
P4	DB4	Data Bit 4
P 3	DB3	LEDn (K) Back Light

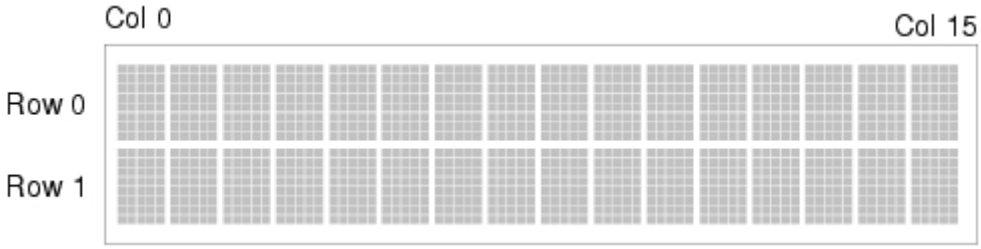
P 2	E	LCD Enable (strobe to transfer new data to the LCD Controller)
P 1	R/!W	Read/not Write
P 0	RS	Register Select (select data or control input data)

Figure 2. Connections of the PCF8574AT I2C PC Board connected to the HD44780 LCD Controller as a single Device with 4 wire input Vdd, Vss, SDA and SCL.



The **CharLCD_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 and down. In a **4x20** display, writing beyond column 19 of Row 0 results in the continuation onto Row 2 and writing beyond column 19 of Row 1 results in the continuation onto Row 3. This is not an issue in the standard **2x16** Hitachi LCD module, writing Charaters beyond column 15 are not displayed.

Figure 3. 2x16 Hitachi LCD Module



Resources

The CharLCD_I2C component only requires the 2 I/O pins, SDA and SCL, of the I2C Master Component.

PCF 8573AT I2C Slave Address

The default 7 bit Hex Address is **0x3F.** At this time, if the Slave Adrress needs to be changed, the **CharLCD_I2C.h** file will have to be modified. This can be done by first going to the main Menu Bar of PSoC Creator and clicking on File ,Open, Project and going to the **CharLCD_I2C.cylib Folder** and double clicking on the **CharLCD_I2C.cyprj** file. Once the Project is open, then click on the component tab and there will be a list of the API files. Double Click on the **CharLCD_I2C.h** file which will open the file so it can be modified. Go to this section of the file where the Slave Address can be changed (0x3Fu).

```
/***************************************
*            Registers                 *
***************************************/
/* I2C slave address for the PCF8574A  A0=1,A1=1,A2=1*/

#define `$INSTANCE_NAME`_I2C_SLAVE_ADDR  (0x3Fu)
```

Then click on the File Menu and Save All. Now when Building a Project using the CharLCD_I2C component, the new Slave Address will be in affect.

The Data Sheet for the PCF8574A chip can be downloaded by right clicking on the CharLCD_I2C component and clicking on Open Component Web Page. There is a table of the 8 different 7 bit Slave Addresses used by the chip.

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The memory usage for all APIs available in the given component configuration has not been determined for this publication.

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
1.10	Removed Statements for While Loop /* Waits until Master Completes Transfer */	Endless While Loop if LCD not connected to the I/O pins of the PSoC causing main.c to hang.
1.20	Corrected API Statement.	No associated Compiler Errors.
1.30	Minor PDF Document changes	Changed Instance Name of the API commands.
1.50	Changed Generated API File Names	Avoid API File Name Conflicts when using with the Character LCD Display Component.
2.0	Changes of Datasheet with New Screen Shots and improved Instructions.	Directions using Dependency Mode to Load the CharLCD_I2C component. Improved Instance Instuctions when naming the Master Component.