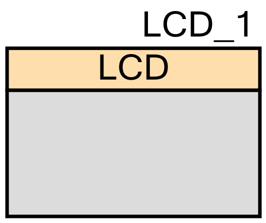
Segment LCD (PDL_LCD)

1.0



Features

- 8 COM (40 segment outputs, LCDRAM up to 40 bytes, bias – 1/3 or 1/4) mode or 4 COM (44 segment outputs, LCDRAM up to 22 bytes, bias) mode is selectable for display mode
- Divider resistors for generating LCD drive power are incorporated allowing selecting 10kΩ or 100kΩ for the resistor values (the LCD drive power can be supplied from the external circuit)
- Sub-clock and PCLK are available for LCD controller operating clock (LCDC clock).
- Blinking (flashing) function is available
- Direct drive for LCD panel is available
- Voltage Booster configuration is available
- Interrupt request is allowed per frame

General Description

The Peripheral Driver Library (PDL) Segment LCD controller (PDL_LCD) component displays the contents of Display Data Memory (LCDRAM) directly to the LCD (Liquid Crystal Display) panel by using segment outputs and common outputs.

This component uses firmware drivers from the PDL_LCD module, which is automatically added to your project after a successful build.

When to Use a PDL_LCD Component

Use the PDL_LCD component when you need to display data on a glass.

Quick Start

1. Drag a PDL_LCD component from the Component Catalog the FMx/Display/Segment LCD folder onto your schematic. The placed instance takes the name LCD_1.
2. Double-click to open the component's Configure dialog. See Component Parameters.
3. Edit parameters in the configure dialog to set display mode, activate SEG and COM outputs, activate interrupt if needed.

4. Build the project to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer and generate configuration data for the LCD_1 instance.
5. In the *main.c* file, initialize the peripheral and start the application.

```
(void)Lcd_Init((stc_lcd_config_t*) &LCD_1_Config);
Lcd_SetDispMode(LCD_1_Config.enDispMode);
```

6. Build and program the device.

Component Parameters

The PDL_LCD component Configure dialog allows you to edit the configuration parameters for the component instance.

Basic Tab

This tab contains the component parameters used in the basic peripheral initialization settings.

Parameter Name	Description
bEnBlankDisp	Display blank (regardless of data in LCD RAM)
bEnDispRevs	Reverse the display
bTimerMod	Run in Timer mode
enDispMode	Select display mode
enDivMode	Divider resistors to control LCD drive power
enDivResVal	Divider resistor values
enInputIoMode	I/O port input control

Booster Tab

This tab contains the Booster configuration settings.

Parameter Name	Description
bEnBooster	Enable booster function
BSTOPT	LCD Booster function available
BSTPD	LCD Booster power control
BTRC	Booster coarse setting bits for VV1 reference voltage
BTRF	Booster fine setting bits for VV1 reference voltage
CENSEL	Control booster C1/C0 pin function

Clock Tab

This tab contains the Clock configuration settings.

Parameter Name	Description
enSrcClk	Controller clock source
u32DivVal	Controller clock division ratio

Common Select Tab

This tab contains the Common Select configuration settings.

Parameter Name	Description
COM*n*	Common select bits, where *n* = 0..3 if display mode set to COM4, if display mode set to COM8 then *n* = 0..7

Interrupts Tab

This tab contains the Interrupts configuration settings.

Parameter Name	Description
bIrqEn	Enable LCD interrupt
bTouchNvic	Update the NVIC with the LCD interrupt
pfnIrqCb	Callback function for LCD interrupts. Note: this generates a declaration only - USER must implement the function

Segment Select Tab

This tab contains the Segment Select configuration settings.

Parameter Name	Description
SEG*n*	Segment select bits, where *n* = 00..39 when display mode COM8 chosen, or *n* = 00..43 when display mode COM4 chosen

VVx Select Tab

This tab contains the VVx Select configuration settings.

Parameter Name	Description
VE*n*	VVx select bits. Where *n* = 0..4

Component Usage

After a successful build, firmware drivers from the PDL_LCD module are added to your project in the pdl/drivers/lcd folder. Pass the generated data structures to the associated PDL functions in your application initialization code to configure the peripheral.

Generated Data

The PDL_LCD component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the component (e.g. *LCD_1_config.c*). Each variable is also prefixed with the instance name of the component.

Data Structure Type	Name	Description
stc_lcd_clk_config_t	LCD_1_ClkConf	Select source clock and configure clock division ratio structure
stc_lcd_vv_sel_t	LCD_1_VvSel	VVx selection bit structure
stc_lcd_seg_sel1_t	LCD_1_SegSel1	These bits control I/O port status and analog switches for SEG outputs
stc_lcd_seg_sel2_t	LCD_1_SegSel2	These bits control I/O port status and analog switches for SEG outputs
stc_lcd_com_sel_t	LCD_1_ComSel	These bits control I/O port status and analog switches for COM outputs
stc_lcd_booster_t	LCD_1_Booster	Booster function structure
stc_lcd_config_t	LCD_1_Config	Configuration structure

Once the component is initialized, the application code should use the peripheral functions provided in the referenced PDL files. Refer to the PDL documentation for the list of provided API functions. To access this document, right-click on the component symbol on the schematic and choose "**Open API Documentation…**" option in the drop-down menu.

Data in RAM

The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the Built-In tab of the Configure dialog set the parameter CONST_CONFIG to make your selection. The default option is to place the data in flash.

Interrupt Support

If the PDL_LCD component is specified to trigger interrupts, it will generate the callback function declaration that will be called from the LCD ISR. The user is then required to provide the actual callback code. If a null string is provided the struct is populated with zeroes and the callback declaration is not generated. In that case it is the user's responsibility to modify the struct in firmware.

The component generates the following function declarations.

Function Callback	Description
LCD_1_ LcdIrqCb	Callback function for LCD interrupts. Note: this generates a declaration only - USER must implement the function

Code Examples and Application Notes

There are numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

Cypress also provides a number of application notes describing how FMx devices can be integrated into your design. You can access the Cypress Application Notes search web page at www.cypress.com/appnotes.

Resources

The PDL_LCD component uses the Segment LCD (LCD) peripheral block.

References

- FM0+ Family of 32-bit ARM® Cortex®-M0+ Microcontrollers Peripheral Manuals
- Cypress FM0+ Family of 32-bit ARM® Cortex®-M0+ Microcontrollers

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.0	Initial Version	

© Cypress Semiconductor Corporation, 2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.