PSoC WS0010 Component

Thanasis Georgiou

Latest update: Version 1.0, 27 February 2016

Requirements

This display requires 7 pins to use in 4-bit mode:

- RS
- R/W
- Enable
- D4-D7

Pins D4 to D7 must be next to each other, the rest can be placed wherever it's convenient. There isn't any buffering done so the memory requirements should be minimal.

API

For readability reasons, the following will assume the component is named `'WS0010'`. Of course it can be named anything desired.

Macros

General Control Commands

- `LCD_CLEAR_DISPLAY`: Use to clear the display's contents.
- `LCD_RETURN_HOME`: Return the cursor to (0, 0)
- `LCD_FUNCTION_SET`: Use to change to 4/8-bit mode and to change fonts.
- `LCD_SET_CGRAM_ADDR`: Set a specific address in the custom character RAM.
- `LCD_SET_DDRAM_ADDR`: Set a specific address in the graphics RAM.

Entry Mode Commands

- `LCD_ENTRY_MODE_SET`: Use with the following flags to change how text is entered
  - `LCD_ENTRY_RIGHT`: Right-to-left text
  - `LCD_ENTRY_LEFT`: Left-to-right text
  - `LCD_ENTRY_SHIFT_INCREMENT`: 'Right justify' text
  - LCD_ENTRY_SHIFT_DECREMENT: 'Left justify' text

On/Off & Cursor Control

- `LCD_DISPLAY_CONTROL`: Use with the following flags to turn the display on/off and change the cursor's behaviour.
    - `LCD_DISPLAY_ON`: Turn the display on
    - `LCD_DISPLAY_OFF`: Turn the display off
    - `LCD_CURSOR_ON`: Make the cursor visible
    - `LCD_CURSOR_OFF`: Make the cursor invisible
    - `LCD_BLINK_ON`: Turn on cursor blinking
    - `LCD_BLINK_OFF`: Turn off cursor blinking

Scrolling Commands

- `LCD_CURSOR_SHIFT`, `LCD_DISPLAY_MOVE`: Use to scroll the display without changing RAM's contents.
- `LCD_CURSOR_MOVE`: Move the cursor
- `LCD_MOVE_RIGHT`: Scroll the display right.
- `LCD_SET_DDRAM_ADDR:` Scroll the display left.

Power & Character/Graphic Mode Selection

- `LCD_POWER_ON`: Turn on the display's power circuits.
- `LCD_POWER_OFF`: Turn off the display's power circuits, use to conserve power.
- `LCD_GRAPHIC_MODE`: Put the display in Graphic mode.
- `LCD_CHARACTER_MODE`: Put the display in Character mode.

4/8-Bit Switching and Fonts

- `LCD_8BIT_MODE`: Put module in 8-bit mode.
- `LCD_SET_DDRAM_ADDR:` Put module in 4-bit mode.
- `LCD_JAPANESE`: Default English + Japanese charset.
- `LCD_EUROPEAN_I`: Extended latin charset.
- `LCD_RUSSIAN`: Russian charset.
- LCD_EUROPEAN_II: Extended latin + greek charset.

High-level functions

- `WS0010_Start(void);`

  Prepare the display for usage. Just an alias for WS0010_Init(void) for now.

- `WS0010_Init(void);`

  This command restarts the display, puts it in 4-bit mode, clears all contents and turns the display back on.

- `WS0010_Position(uint8_t column, uint8_t row);`

  Move the cursor to the given position.

- `WS0010_Sleep(void);`

Puts the module in sleep mode. Power consumption during sleep mode should be around 0.8mA. **Side effect:** This will reset cursor visibility and blinking.

- `WS0010_Wakeup(void);`

  Wakes the module app.

- `WS0010_PutChar(char data);` *MACRO*

  Prints a single character at the cursor's position.

- `WS0010_PrintString(char *string);`

Prints a null terminated ('\0') string to the current cursor position. No automatic line wrapping.

- `WS0010_PrintByte(unsigned char byte);`

  Prints a byte to the current cursor position in hexadecimal notation.

- `WS0010_ClearDisplay(void);` *MACRO*

  Clears the display's contents.

- `WS0010_DisplayOn(void);` *MACRO*

  Turns the display on. **Side effect:** This will reset cursor visibility and blinking.

- `WS0010_DisplayOff(void);` *MACRO*

  Turns the display off. **Side effect:** This will reset cursor visibility and blinking.

Low-level functions

- `WS0010_SendNibble(unsigned char nibble);`

  Sends 4-bits to the display.

- `WS0010_SendByte(unsigned char byte);`

Sends a whole byte to the display (in two chunks of 4-bits each). Using WS0010_WriteData() and WS0010_WriteControl() instead of this is probably a good idea.

- `WS0010_WriteControl(unsigned char cmd);`

Sends a command to the display. These commands can be constructed using the macros defined by this component and some bitwise operations.

- `WS0010_WriteData(unsigned char data);`

  Sends some data to the display. Same as WS0010_PutChar().

- `WS0010_IsReady();`

  Stalls code execution until the display is ready to receive the next command.

Example Code

```
WS0010_Init();

WS0010_PrintString("Hello PSoC! ");
WS0010_Position(0, 1);
WS0010_PrintString("Second line.");
```

Changelog

Version 1.0 - 27 February 2016

- Initial Release.