

PMODIO

Generated by Doxygen 1.8.6

Fri Mar 18 2016 21:35:30

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	LCD_pos Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	page	5
3.1.2.2	x	5
3.1.2.3	y	5
3.2	PMODIO Struct Reference	5
3.2.1	Detailed Description	6
3.2.2	Field Documentation	6
3.2.2.1	base_addr	6
3.2.2.2	current_pos	6
3.2.2.3	lcd_map	6
4	File Documentation	7
4.1	PMODIO.c File Reference	7
4.1.1	Detailed Description	7
4.1.2	Macro Definition Documentation	8
4.1.2.1	ROT_BTN_MSK	8
4.1.2.2	ROT_CLR_CLR_MSK	8
4.1.2.3	ROT_CLR_SET_MSK	8
4.1.2.4	SELF_TEST_PATTERN	8
4.1.2.5	SELF_TEST_REG_CNT	8
4.1.3	Function Documentation	8
4.1.3.1	PMODIO_init	8
4.1.3.2	PMODIO_LCD_clr	9

4.1.3.3	PMODIO_LCD_getPos	9
4.1.3.4	PMODIO_LCD_setPos	9
4.1.3.5	PMODIO_LCD_write	10
4.1.3.6	PMODIO_ROT_clr	10
4.1.3.7	PMODIO_ROT_isBtnPressed	10
4.1.3.8	PMODIO_ROT_read	10
4.1.3.9	PMODIO_ROT_setMaxCount	11
4.1.3.10	PMODIO_ROT_setMinCount	11
4.1.3.11	PMODIO_ROT_setStep	11
4.1.3.12	PMODIO_selfTest	12
4.2	PMODIO.h File Reference	12
4.2.1	Detailed Description	13
4.2.2	Macro Definition Documentation	13
4.2.2.1	LCD_CHIP_OFF	13
4.2.2.2	LCD_CHIP_ON	13
4.2.2.3	LCD_COL_SET	13
4.2.2.4	LCD_COLS	13
4.2.2.5	LCD_DISP_START	13
4.2.2.6	LCD_PAGE_HEIGHT	14
4.2.2.7	LCD_PAGES	14
4.2.2.8	LCD_ROW_SET	14
4.2.2.9	LCD_X_MAX	14
4.2.2.10	LCD_Y_MAX	14
4.2.2.11	ROT_CNT_ABS_MAX	14
4.2.2.12	ROT_CNT_ABS_MIN	14
4.2.3	Function Documentation	14
4.2.3.1	PMODIO_init	14
4.2.3.2	PMODIO_LCD_clr	15
4.2.3.3	PMODIO_LCD_getPos	16
4.2.3.4	PMODIO_LCD_setPos	16
4.2.3.5	PMODIO_LCD_write	16
4.2.3.6	PMODIO_ROT_clr	17
4.2.3.7	PMODIO_ROT_isBtnPressed	17
4.2.3.8	PMODIO_ROT_read	17
4.2.3.9	PMODIO_ROT_setMaxCount	17
4.2.3.10	PMODIO_ROT_setMinCount	18
4.2.3.11	PMODIO_ROT_setStep	18
4.2.3.12	PMODIO_selfTest	18
4.3	PMODIO_I.c File Reference	19
4.3.1	Detailed Description	19

4.3.2	Macro Definition Documentation	20
4.3.2.1	DELAY_LEN	20
4.3.2.2	LCD_CLR_CNTL_CS_1	20
4.3.2.3	LCD_CLR_CNTL_CS_2	21
4.3.2.4	LCD_CLR_CNTL_EN_OP	21
4.3.2.5	LCD_CLR_CNTL_REG_SEL	21
4.3.2.6	LCD_CLR_CNTL_RST_N	21
4.3.2.7	LCD_SET_CNTL_CS_1	21
4.3.2.8	LCD_SET_CNTL_CS_2	21
4.3.2.9	LCD_SET_CNTL_EN_OP	21
4.3.2.10	LCD_SET_CNTL_REG_SEL	21
4.3.2.11	LCD_SET_CNTL_RST_N	21
4.3.2.12	nop	21
4.3.3	Function Documentation	22
4.3.3.1	PMODIO_LCD_clrCntlBit	22
4.3.3.2	PMODIO_LCD_delay	23
4.3.3.3	PMODIO_LCD_exec	23
4.3.3.4	PMODIO_LCD_execCmd	23
4.3.3.5	PMODIO_LCD_init	23
4.3.3.6	PMODIO_LCD_setCntlBit	24
4.3.3.7	PMODIO_LCD_writeCommit	24
4.3.3.8	PMODIO_LCD_writeDataBits	24
4.4	PMODIO_I.h File Reference	25
4.4.1	Detailed Description	25
4.4.2	Macro Definition Documentation	26
4.4.2.1	GP_RSVD0	26
4.4.2.2	GP_RSVD1	26
4.4.2.3	GP_RSVD2	26
4.4.2.4	LCD_CNTL	26
4.4.2.5	LCD_DATA	26
4.4.2.6	LCD_RSVD0	26
4.4.2.7	PMODIO_mReadReg	26
4.4.2.8	PMODIO_mWriteReg	27
4.4.2.9	ROT_CNT	27
4.4.2.10	ROT_MAX_CNT	27
4.4.2.11	ROT_MIN_CNT	27
4.4.2.12	ROT_RSVD0	27
4.4.2.13	ROT_STEP	27
4.4.2.14	ROT_STS	28
4.4.3	Typedef Documentation	28

4.4.3.1	LCD_CHIP	28
4.4.3.2	LCD_CNTL_BIT	28
4.4.4	Enumeration Type Documentation	28
4.4.4.1	CHIP	28
4.4.4.2	_CNTL_BIT	28
4.4.5	Function Documentation	28
4.4.5.1	PMODIO_LCD_clrCntlBit	28
4.4.5.2	PMODIO_LCD_delay	29
4.4.5.3	PMODIO_LCD_exec	29
4.4.5.4	PMODIO_LCD_execCmd	29
4.4.5.5	PMODIO_LCD_init	29
4.4.5.6	PMODIO_LCD_setCntlBit	29
4.4.5.7	PMODIO_LCD_writeCommit	30
4.4.5.8	PMODIO_LCD_writeDataBits	30
Index		31

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

LCD_pos	5
PMDIO	5

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

PMDIO.c	7
PMDIO.h	12
PMDIO_I.c	19
PMDIO_I.h	25

Chapter 3

Data Structure Documentation

3.1 LCD_pos Struct Reference

```
#include <PMODIO.h>
```

Data Fields

- u8 [x](#)
- u8 [y](#)
- u8 [page](#)

3.1.1 Detailed Description

Tracks current LCD write position

3.1.2 Field Documentation

3.1.2.1 u8 LCD_pos::page

current page

3.1.2.2 u8 LCD_pos::x

current X coordinate

3.1.2.3 u8 LCD_pos::y

current Y coordinate

The documentation for this struct was generated from the following file:

- [PMODIO.h](#)

3.2 PMODIO Struct Reference

```
#include <PMODIO.h>
```

Data Fields

- u32 [base_addr](#)
- [LCD_pos](#) [current_pos](#)
- u8 [lcd_map](#) [[LCD_PAGES](#)][[LCD_COLS](#)]

3.2.1 Detailed Description

One instance of this type must be instantiated for each instance of the [PMDIO](#) peripheral in the system

3.2.2 Field Documentation

3.2.2.1 u32 PMDIO::base_addr

device base address

3.2.2.2 [LCD_pos](#) PMDIO::current_pos

current LCD write position

3.2.2.3 u8 PMDIO::lcd_map[LCD_PAGES][LCD_COLS]

static array to track writes to LCD

The documentation for this struct was generated from the following file:

- [PMDIO.h](#)

Chapter 4

File Documentation

4.1 PMODIO.c File Reference

```
#include "stdbool.h"
#include "xparameters.h"
#include "xil_types.h"
#include "xstatus.h"
#include "PMDIO.h"
#include "PMDIO_l.h"
```

Macros

- #define ROT_BTN_MSK 0x00000001
- #define ROT_CLR_SET_MSK 0x00000002
- #define ROT_CLR_CLR_MSK (ROT_CLR_SET_MSK ^ 0xFFFFFFFF)
- #define SELF_TEST_REG_CNT 3
- #define SELF_TEST_PATTERN 0xF0F0F0F0

Functions

- XStatus PMDIO_init (PMDIO *inst_p, u32 base_addr)
- XStatus PMDIO_selfTest (PMDIO *inst_p)
- XStatus PMDIO_ROT_setStep (PMDIO *inst_p, int step)
- XStatus PMDIO_ROT_setMaxCount (PMDIO *inst_p, int count)
- XStatus PMDIO_ROT_setMinCount (PMDIO *inst_p, int count)
- XStatus PMDIO_ROT_clr (PMDIO *inst_p)
- XStatus PMDIO_ROT_read (PMDIO *inst_p, int *rotary_cnt_p)
- bool PMDIO_ROT_isBtnPressed (PMDIO *inst_p)
- XStatus PMDIO_LCD_setPos (PMDIO *inst_p, u8 x, u8 y)
- LCD_pos PMDIO_LCD_getPos (PMDIO *inst_p)
- XStatus PMDIO_LCD_write (PMDIO *inst_p, u8 data, u8 mask)
- XStatus PMDIO_LCD_clr (PMDIO *inst_p)

4.1.1 Detailed Description

Author

Scott Lawson

Copyright

Scott Lawson, 2016

PMODIO peripheral driver API function implementations

The **PMODIO** peripheral interfaces with the **PMODIO** hardware, which consists of a KS0108-compatible 128 pixel x 64 pixel dot-matrix graphical LCD and a quadrature rotary encoder.

These functions allow an application to interface with the LCD and rotary encoder hardware to draw on the LCD and obtain the rotary encoder's rotation count, and configure the rotary encoder's step size per rotation, count limits, and rotation increment/decrement directions.

Note

The LCD code is derived from the GLCD library from microchip.com. Download link: <http://www.microchip.com/forums/download.axd?file=0;275906>

Modification History:

Ver	Who	Date	Changes
1.0	SRL	16 Mar 2016	Initial release

4.1.2 Macro Definition Documentation**4.1.2.1 #define ROT_BTN_MSK 0x00000001**

rotary encoder pushbutton status

4.1.2.2 #define ROT_CLR_CLR_MSK (ROT_CLR_SET_MSK ^ 0xFFFFFFFF)

clear count bit mask

4.1.2.3 #define ROT_CLR_SET_MSK 0x00000002

set the clear count bit mask

4.1.2.4 #define SELF_TEST_PATTERN 0xF0F0F0F0

pattern written to registers and read back during self-test

4.1.2.5 #define SELF_TEST_REG_CNT 3

number of registers being tested

4.1.3 Function Documentation**4.1.3.1 XStatus PMODIO_init (PMODIO * inst_p, u32 base_addr)**

Initializes the **PMODIO** peripheral

Configures rotary encoder defaults Count step is 1 Negative counts are allowed Max count of 2^{32}

Parameters

<i>inst_p</i>	PMODIO instance address
<i>base_addr</i>	Base address of PMODIO instance

Returns

- XST_SUCCESS if all self-test code passed
- XST_FAILURE if any self-test code failed

Note

This must be the first function called for this peripheral

This function calls PMODIO_SelfTest(), so it's notes apply for this function as well

4.1.3.2 XStatus PMODIO_LCD_clr (PMODIO * *inst_p*)

Clears all pixels on the LCD

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

XST_SUCCESS

4.1.3.3 LCD_pos PMODIO_LCD_getPos (PMODIO * *inst_p*)

Retrieves the current LCD write position

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

LCD_pos current write position of the LCD

4.1.3.4 XStatus PMODIO_LCD_setPos (PMODIO * *inst_p*, u8 *x*, u8 *y*)

Sets the X-Y position for the next data write

X coordinate valid range is 0 - 127, inclusive Y coordinate valid range is 0 - 63, inclusive

Parameters

<i>inst_p</i>	PMODIO instance address
<i>x</i>	x-coordinate

<i>y</i>	y-coordinate
----------	--------------

Returns

- XST_SUCCESS if X and Y coordinates are in the valid range ($x < 128$ and $Y < 64$)
- XST_FAILURE if either coordinate is out of range

4.1.3.5 XStatus PMODIO_LCD_write (PMODIO * *inst_p*, u8 *data*, u8 *mask*)

Writes data to LCD at the current position

Parameters

<i>inst_p</i>	PMODIO instance address
<i>data</i>	data to write
<i>mask</i>	mask to select which bits are written

Returns

XST_SUCCESS

Note

The PMODIO instance's current X position will be automatically incremented by 1 on every write. Since writes to the LCD must be done a full byte at a time, the mask argument is provided to prevent the caller from having to perform read-modify-write cycles when a data write overlaps with a previous write. The driver tracks all writes internally, and when writing new data, it handles the read-modify-write cycles to only write bits to the LCD that are enabled by a set bit in the mask.

4.1.3.6 XStatus PMODIO_ROT_clr (PMODIO * *inst_p*)

Resets the current count to zero

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

XST_SUCCESS

4.1.3.7 bool PMODIO_ROT_isBtnPressed (PMODIO * *inst_p*)

Returns the current state of the rotary encoder pushbutton

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

true if the button is pressed, false otherwise

4.1.3.8 XStatus PMODIO_ROT_read (PMODIO * *inst_p*, int * *rotary_cnt_p*)

Retrieves the current rotary encoder count

Parameters

<i>inst_p</i>	PMODIO instance address
<i>rotary_cnt_p</i>	pointer to the location to populate with the count

Returns

XST_SUCCESS

4.1.3.9 XStatus PMODIO_ROT_setMaxCount (PMODIO * *inst_p*, int *count*)

Sets the maximum count allowed

Parameters

<i>inst_p</i>	PMODIO instance address
<i>count</i>	the maximum allowed count

Returns

- XST_SUCCESS if max count input is valid ($\leq 2^{31} - 1$)
- XST_FAILURE if count value is invalid

4.1.3.10 XStatus PMODIO_ROT_setMinCount (PMODIO * *inst_p*, int *count*)

Sets the minimum count allowed

Parameters

<i>inst_p</i>	PMODIO instance address
<i>count</i>	the maximum allowed count

Returns

- XST_SUCCESS if min count input is valid ($\geq -2^{31}$)
- XST_FAILURE if count value is invalid

4.1.3.11 XStatus PMODIO_ROT_setStep (PMODIO * *inst_p*, int *step*)

Sets the number that the rotary encoder count is incremented or decremented by per rotation

Parameters

<i>inst_p</i>	PMODIO instance address
<i>step</i>	sets number of "counts" to increment or decrement the current count by when the rotary encoder is turned one click - can be negative to invert which rotation direction increments/decrements the count

Returns

XST_SUCCESS

4.1.3.12 XStatus PMODIO_selfTest (PMODIO * inst_p)

Run a self-test on the driver/device

If the hardware system is not built correctly, this function may never return to the caller.

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

- XST_SUCCESS if all self-test code passed
- XST_FAILURE if any self-test code failed

Note

Caching must be turned off for this function to work
Self test may fail if data memory and device are not on the same bus

4.2 PMODIO.h File Reference

```
#include "stdbool.h"
```

Data Structures

- struct [LCD_pos](#)
- struct [PMODIO](#)

Macros

- #define [ROT_CNT_ABS_MAX](#) 2147483647
- #define [ROT_CNT_ABS_MIN](#) (-2147483648)
- #define [LCD_X_MAX](#) 127
- #define [LCD_Y_MAX](#) 63
- #define [LCD_CHIP_ON](#) 0x3F
- #define [LCD_CHIP_OFF](#) 0x3E
- #define [LCD_DISP_START](#) 0xC0
- #define [LCD_COL_SET](#) 0xB8
- #define [LCD_ROW_SET](#) 0x40
- #define [LCD_COLS](#) 128
- #define [LCD_PAGES](#) 8
- #define [LCD_PAGE_HEIGHT](#) 8

Functions

- XStatus [PMODIO_init](#) (PMODIO *inst_p, u32 base_addr)
- XStatus [PMODIO_selfTest](#) (PMODIO *inst_p)
- XStatus [PMODIO_ROT_setStep](#) (PMODIO *inst_p, int step)
- XStatus [PMODIO_ROT_setMaxCount](#) (PMODIO *inst_p, int count)

- XStatus [PMODIO_ROT_setMinCount](#) ([PMODIO](#) *inst_p, int count)
- XStatus [PMODIO_ROT_clr](#) ([PMODIO](#) *inst_p)
- XStatus [PMODIO_ROT_read](#) ([PMODIO](#) *inst_p, int *rotary_cnt_p)
- bool [PMODIO_ROT_isBtnPressed](#) ([PMODIO](#) *inst_p)
- XStatus [PMODIO_LCD_setPos](#) ([PMODIO](#) *inst_p, u8 x, u8 y)
- XStatus [PMODIO_LCD_write](#) ([PMODIO](#) *inst_p, u8 data, u8 mask)
- XStatus [PMODIO_LCD_clr](#) ([PMODIO](#) *inst_p)
- [LCD_pos](#) [PMODIO_LCD_getPos](#) ([PMODIO](#) *inst_p)

4.2.1 Detailed Description

Author

Scott Lawson

Copyright

Scott Lawson, 2016

Header file for [PMODIO](#) driver API

Modification History:

Ver	Who	Date	Changes
1.0	SRL	16 Mar 2016	Initial release

4.2.2 Macro Definition Documentation

4.2.2.1 `#define LCD_CHIP_OFF 0x3E`

LCD command bit pattern: chip off

4.2.2.2 `#define LCD_CHIP_ON 0x3F`

LCD command bit pattern: chip on

4.2.2.3 `#define LCD_COL_SET 0xB8`

LCD command bit pattern: column set

4.2.2.4 `#define LCD_COLS 128`

number of columns on the LCD

4.2.2.5 `#define LCD_DISP_START 0xC0`

LCD command bit pattern: display start

4.2.2.6 #define LCD_PAGE_HEIGHT 8

pixel height of one page

4.2.2.7 #define LCD_PAGES 8

number of pages on the LCD

4.2.2.8 #define LCD_ROW_SET 0x40

LCD command bit pattern: row set

4.2.2.9 #define LCD_X_MAX 127

LCD maximum X coordinate

4.2.2.10 #define LCD_Y_MAX 63

LCD maximum y coordinate

4.2.2.11 #define ROT_CNT_ABS_MAX 2147483647

Rotary encoder absolute maximum count ($2^{(32 - 1)} - 1$)

4.2.2.12 #define ROT_CNT_ABS_MIN (-2147483648)

Rotary encoder absolute minimum count ($-2^{(32 - 1)}$)

4.2.3 Function Documentation

4.2.3.1 XStatus PMODIO_init (PMODIO * inst_p, u32 base_addr)

Initializes the [PMODIO](#) peripheral

Configures rotary encoder defaults Count step is 1 Negative counts are allowed Max count of 2^{32}

Parameters

<i>inst_p</i>	PMODIO instance address
<i>base_addr</i>	Base address of PMODIO instance

Returns

- XST_SUCCESS if all self-test code passed
- XST_FAILURE if any self-test code failed

Note

This must be the first function called for this peripheral

This function calls [PMODIO_SelfTest\(\)](#), so it's notes apply for this function as well

4.2.3.2 XStatus PMODIO_LCD_clr (PMODIO * *inst_p*)

Clears all pixels on the LCD

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

XST_SUCCESS

4.2.3.3 LCD_pos PMODIO_LCD_getPos (PMODIO * *inst_p*)

Retrieves the current LCD write position

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

LCD_pos current write position of the LCD

4.2.3.4 XStatus PMODIO_LCD_setPos (PMODIO * *inst_p*, u8 *x*, u8 *y*)

Sets the X-Y position for the next data write

X coordinate valid range is 0 - 127, inclusive Y coordinate valid range is 0 - 63, inclusive

Parameters

<i>inst_p</i>	PMODIO instance address
<i>x</i>	x-coordinate
<i>y</i>	y-coordinate

Returns

- XST_SUCCESS if X and Y coordinates are in the valid range ($x < 128$ and $Y < 64$)
- XST_FAILURE if either coordinate is out of range

4.2.3.5 XStatus PMODIO_LCD_write (PMODIO * *inst_p*, u8 *data*, u8 *mask*)

Writes data to LCD at the current position

Parameters

<i>inst_p</i>	PMODIO instance address
<i>data</i>	data to write
<i>mask</i>	mask to select which bits are written

Returns

XST_SUCCESS

Note

The [PMODIO](#) instance's current X position will be automatically incremented by 1 on every write. Since writes to the LCD must be done a full byte at a time, the mask argument is provided to prevent the caller from having to perform read-modify-write cycles when a data write overlaps with a previous write. The driver tracks all writes internally, and when writing new data, it handles the read-modify-write cycles to only write bits to the LCD that are enabled by a set bit in the mask.

4.2.3.6 XStatus PMODIO_ROT_clr (PMODIO * *inst_p*)

Resets the current count to zero

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	---

Returns

XST_SUCCESS

4.2.3.7 bool PMODIO_ROT_isBtnPressed (PMODIO * *inst_p*)

Returns the current state of the rotary encoder pushbutton

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	---

Returns

true if the button is pressed, false otherwise

4.2.3.8 XStatus PMODIO_ROT_read (PMODIO * *inst_p*, int * *rotary_cnt_p*)

Retrieves the current rotary encoder count

Parameters

<i>inst_p</i>	PMODIO instance address
<i>rotary_cnt_p</i>	pointer to the location to populate with the count

Returns

XST_SUCCESS

4.2.3.9 XStatus PMODIO_ROT_setMaxCount (PMODIO * *inst_p*, int *count*)

Sets the maximum count allowed

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	---

<i>count</i>	the maximum allowed count
--------------	---------------------------

Returns

- XST_SUCCESS if max count input is valid ($\leq 2^{31} - 1$)
- XST_FAILURE if count value is invalid

4.2.3.10 XStatus PMODIO_ROT_setMinCount (PMODIO * *inst_p*, int *count*)

Sets the minimum count allowed

Parameters

<i>inst_p</i>	PMODIO instance address
<i>count</i>	the maximum allowed count

Returns

- XST_SUCCESS if min count input is valid ($\geq -2^{31}$)
- XST_FAILURE if count value is invalid

4.2.3.11 XStatus PMODIO_ROT_setStep (PMODIO * *inst_p*, int *step*)

Sets the number that the rotary encoder count is incremented or decremented by per rotation

Parameters

<i>inst_p</i>	PMODIO instance address
<i>step</i>	sets number of "counts" to increment or decrement the current count by when the rotary encoder is turned one click - can be negative to invert which rotation direction increments/decrements the count

Returns

XST_SUCCESS

4.2.3.12 XStatus PMODIO_selfTest (PMODIO * *inst_p*)

Run a self-test on the driver/device

If the hardware system is not built correctly, this function may never return to the caller.

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	---

Returns

- XST_SUCCESS if all self-test code passed
- XST_FAILURE if any self-test code failed

Note

Caching must be turned off for this function to work
 Self test may fail if data memory and device are not on the same bus

4.3 PMODIO_I.c File Reference

```
#include "xil_types.h"
#include "xstatus.h"
#include "PMODIO.h"
#include "PMODIO_l.h"
```

Macros

- #define LCD_SET_CNTL_RST_N 0x00000001
- #define LCD_SET_CNTL_EN_OP 0x00000002
- #define LCD_SET_CNTL_REG_SEL 0x00000004
- #define LCD_SET_CNTL_CS_1 0x00000008
- #define LCD_SET_CNTL_CS_2 0x00000010
- #define LCD_CLR_CNTL_RST_N (LCD_clear_CNTL_RST_N ^ 0xFFFFFFFF)
- #define LCD_CLR_CNTL_EN_OP (LCD_clear_CNTL_EN_OP ^ 0xFFFFFFFF)
- #define LCD_CLR_CNTL_REG_SEL (LCD_clear_CNTL_REG_SEL ^ 0xFFFFFFFF)
- #define LCD_CLR_CNTL_CS_1 (LCD_clear_CNTL_CS_1 ^ 0xFFFFFFFF)
- #define LCD_CLR_CNTL_CS_2 (LCD_SET_CNTL_CS_2 ^ 0xFFFFFFFF)
- #define DELAY_LEN 600
- #define nop() __asm__ __volatile__ ("nop")

Functions

- XStatus PMODIO_LCD_init (PMODIO *inst_p)
- XStatus PMODIO_LCD_execCmd (PMODIO *inst_p, u8 cmd, LCD_CHIP chip)
- XStatus PMODIO_LCD_writeCommit (PMODIO *inst_p, u8 data)
- XStatus PMODIO_LCD_writeDataBits (PMODIO *inst_p, u8 pattern)
- XStatus PMODIO_LCD_clrCntlBit (PMODIO *inst_p, LCD_CNTL_BIT bit)
- XStatus PMODIO_LCD_setCntlBit (PMODIO *inst_p, LCD_CNTL_BIT bit)
- XStatus PMODIO_LCD_exec (PMODIO *inst_p)
- void PMODIO_LCD_delay ()

4.3.1 Detailed Description**Author**

Scott Lawson

Copyright

Scott Lawson, 2016

Contains implementations of low-level driver functions. These functions are used by the driver API functions.

This code targets a KS0108-compatible, 128x64 pixel, graphical LCD. It supports the API functions to provide 8-bit pattern writing to arbitray positions on the LCD. For these LCDs, X is the long axis, and Y is the short axis. The display is split into Chip 1 for the left half, and Chip 2 for the right half. The display area is further split into eight rows of pixels (8 pixels tall each), called "pages."

Bit/Pin Mapping:

Abbreviation	Enum Name	Description
RST	RESET_N	Active-low reset
EN	EN_OP	Falling-edge triggered operation enable
RS	REG_SEL	Register select (0=instruction, 1=data)
CS1	CS1	Chip-select 1
CS2	CS2	Chip-select 2

Implementation Note: For the current hardware revision, there weren't enough pins available for every pin on the LCD interface, so the read/write pin was tied to ground, which sets it as write-only. Since we can't read from the LCD memory, we can't do read-write-modify cycles. This is a problem when the application layer wants to write new pixels in a page and column that already has some pixels, or perform a write that is spread across two pages. In both cases, previously written display data will be overwritten if the data byte is simply put on the data bus and written out.

This driver overcomes this limitation by maintaining its own 128-bit x 64-bit array to keep track of display data. Every write to the LCD is also reflected in this array, so read-modify-write cycles are performed using this array to avoid clobbering existing display data in the LCD. This array is maintained in the [PMODIO](#) structure that is allocated for each instance of the [PMODIO](#) peripheral in the Vivado block diagram (it's the `lcd_map` attribute).

Warning

These functions are not intended to be used directly by any code except the [PMODIO](#) driver API functions. They may modify internal state or otherwise interfere with normal driver operation if called from the main application.

Note

This code is derived from the GLCD library from microchip.com. Download link: <http://www.microchip.com/forums/download.axd?file=0;275906>

Modification History:

Ver	Who	Date	Changes
1.0	SRL	16 Mar 2016	Initial release

4.3.2 Macro Definition Documentation**4.3.2.1 #define DELAY_LEN 600**

sets number of iterations for [nop\(\)](#) delay loop in `PMODIO_LCD_exec`

4.3.2.2 #define LCD_CLR_CNTL_CS_1 (LCD_clear_CNTL_CS_1 ^ 0xFFFFFFFF)

LCD control pin clear mask: chip select 1

4.3.2.3 `#define LCD_CLR_CNTL_CS_2 (LCD_SET_CNTL_CS_2 ^ 0xFFFFFFFF)`

LCD control pin clear mask: chip select 2

4.3.2.4 `#define LCD_CLR_CNTL_EN_OP (LCD_clear_CNTL_EN_OP ^ 0xFFFFFFFF)`

LCD control pin clear mask: enable

4.3.2.5 `#define LCD_CLR_CNTL_REG_SEL (LCD_clear_CNTL_REG_SEL ^ 0xFFFFFFFF)`

LCD control pin clear mask: register select

4.3.2.6 `#define LCD_CLR_CNTL_RST_N (LCD_clear_CNTL_RST_N ^ 0xFFFFFFFF)`

LCD control pin clear mask: reset

4.3.2.7 `#define LCD_SET_CNTL_CS_1 0x00000008`

LCD control pin set mask: chip select 1

4.3.2.8 `#define LCD_SET_CNTL_CS_2 0x00000010`

LCD control pin set mask: chip select 2

4.3.2.9 `#define LCD_SET_CNTL_EN_OP 0x00000002`

LCD control pin set mask: enable

4.3.2.10 `#define LCD_SET_CNTL_REG_SEL 0x00000004`

LCD control pin set mask: register select

4.3.2.11 `#define LCD_SET_CNTL_RST_N 0x00000001`

LCD control pin set mask: reset

4.3.2.12 `#define nop() __asm__ __volatile__ ("nop")`

Implements a Microblaze nop

Returns

NONE

Note

C-style signature: void [nop\(\)](#)

4.3.3 Function Documentation

4.3.3.1 XStatus PMODIO_LCD_clrCntlBit (PMODIO * *inst_p*, LCD_CNTL_BIT *bit*)

Clears the specified LCD control bit without modifying others

Parameters

<i>inst_p</i>	PMODIO instance address
<i>bit</i>	bit to clear

Returns

XST_SUCCESS

4.3.3.2 void PMODIO_LCD_delay ()

Executes a NOP loop to meet LCD timing requirements

Returns

NONE

4.3.3.3 XStatus PMODIO_LCD_exec (PMODIO * *inst_p*)

Instructs the LCD to act on the data/command on the data bus

The LCD uses the falling edge of the EN pin as an indicator that the data bus has data to be consumed. The RS pin indicates if the data bus contains an instruction (0) to be executed, or data (1) to be written to display ram. Delay no-op loops are used to meet the LCD timing requirements.

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

XST_SUCCESS

4.3.3.4 XStatus PMODIO_LCD_execCmd (PMODIO * *inst_p*, u8 *cmd*, LCD_CHIP *chip*)

Writes a command to the specified LCD chip

Parameters

<i>inst_p</i>	PMODIO instance address
<i>cmd</i>	the command to execute
<i>chip</i>	the chip to execute the command for

Returns

XST_SUCCESS

Note

The LCD uses its 8-bit data bus both for data and for commands.

4.3.3.5 XStatus PMODIO_LCD_init (PMODIO * *inst_p*)

Initializes the LCD

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

XST_SUCCESS

4.3.3.6 XStatus PMODIO_LCD_setCntlBit (PMODIO * *inst_p*, LCD_CNTL_BIT *bit*)

Sets the specified LCD control bit without modifying others

Parameters

<i>inst_p</i>	PMODIO instance address
<i>bit</i>	bit to set

Returns

XST_SUCCESS

4.3.3.7 XStatus PMODIO_LCD_writeCommit (PMODIO * *inst_p*, u8 *data*)

Commits data to LCD display RAM and driver LCD map

Parameters

<i>inst_p</i>	PMODIO instance address
<i>data</i>	data to write

Returns

XST_SUCCESS

Note

The LCD's current X position will be automatically incremented by 1 on every write.

4.3.3.8 XStatus PMODIO_LCD_writeDataBits (PMODIO * *inst_p*, u8 *pattern*)

Writes a specified pattern to the LCD data bus

Parameters

<i>inst_p</i>	PMODIO instance address
<i>pattern</i>	bit pattern to write to LCD data bus

Returns

XST_SUCCESS

Note

The LCD uses its 8-bit data bus both for data and for commands.

4.4 PMODIO_I.h File Reference

```
#include "PMODIO.h"
```

Macros

- `#define ROT_CNT 0`
- `#define ROT_STEP 4`
- `#define ROT_STS 8`
- `#define ROT_MAX_CNT 12`
- `#define ROT_MIN_CNT 16`
- `#define ROT_RSVD0 20`
- `#define LCD_DATA 24`
- `#define LCD_CNTL 28`
- `#define LCD_RSVD0 32`
- `#define GP_RSVD0 36`
- `#define GP_RSVD1 40`
- `#define GP_RSVD2 44`
- `#define PMODIO_mWriteReg(BaseAddress, RegOffset, Data) Xil_Out32((BaseAddress) + (RegOffset), (u32)(Data))`
- `#define PMODIO_mReadReg(BaseAddress, RegOffset) Xil_In32((BaseAddress) + (RegOffset))`

Typedefs

- `typedef enum CNTL_BIT LCD_CNTL_BIT`
- `typedef enum CHIP LCD_CHIP`

Enumerations

- `enum CNTL_BIT {
 RESET_N, EN_OP, REG_SEL, CS1,
 CS2 }`
- `enum CHIP { CHIP1, CHIP2 }`

Functions

- `XStatus PMODIO_LCD_init (PMODIO *inst_p)`
- `XStatus PMODIO_LCD_execCmd (PMODIO *inst_p, u8 cmd, LCD_CHIP chip)`
- `XStatus PMODIO_LCD_writeDataBits (PMODIO *inst_p, u8 pattern)`
- `XStatus PMODIO_LCD_writeCommit (PMODIO *inst_p, u8 data)`
- `XStatus PMODIO_LCD_setCntlBit (PMODIO *inst_p, LCD_CNTL_BIT bit)`
- `XStatus PMODIO_LCD_clrCntlBit (PMODIO *inst_p, LCD_CNTL_BIT bit)`
- `XStatus PMODIO_LCD_exec (PMODIO *inst_p)`
- `void PMODIO_LCD_delay ()`

4.4.1 Detailed Description

Author

Scott Lawson

Copyright

Scott Lawson, 2016

Header file for low-level driver functions in the [PMDIO](#) peripheral driver.

Warning

This file is only intended for [PMDIO](#) driver use, and should not be directly included by applications. Use [PMDIO.h](#) for API access.

Modification History:

Ver	Who	Date	Changes
1.0	SRL	16 Mar 2016	Initial release

4.4.2 Macro Definition Documentation**4.4.2.1 #define GP_RSVD0 36**

Register offset: General purpose reserved

4.4.2.2 #define GP_RSVD1 40

Register offset: General purpose reserved

4.4.2.3 #define GP_RSVD2 44

Register offset: General purpose reserved

4.4.2.4 #define LCD_CNTL 28

Register offset: LCD control

4.4.2.5 #define LCD_DATA 24

Register offset: LCD data

4.4.2.6 #define LCD_RSVD0 32

Register offset: LCD reserved 0

4.4.2.7 #define PMDIO_mReadReg(BaseAddress, RegOffset) Xil_In32((BaseAddress) + (RegOffset))

Read a value from a [PMDIO](#) register. A 32 bit read is performed. If the component is implemented in a smaller width, only the least significant data is read from the register. The most significant data will be read as 0.

Parameters

<i>BaseAddress</i>	is the base address of the PMODIO device.
<i>RegOffset</i>	is the register offset from the base to write to.

Returns

Data is the data from the register.

Note

C-style signature: u32 [PMODIO_mReadReg](#)(u32 BaseAddress, unsigned RegOffset)

4.4.2.8 `#define PMODIO_mWriteReg(BaseAddress, RegOffset, Data) Xil_Out32((BaseAddress) + (RegOffset), (u32)(Data))`

Write a value to a [PMODIO](#) register. A 32 bit write is performed. If the component is implemented in a smaller width, only the least significant data is written.

Parameters

<i>BaseAddress</i>	is the base address of the PMODIOdevice.
<i>RegOffset</i>	is the register offset from the base to write to.
<i>Data</i>	is the data written to the register.

Returns

None.

Note

C-style signature: void [PMODIO_mWriteReg](#)(u32 BaseAddress, unsigned RegOffset, u32 Data)

4.4.2.9 `#define ROT_CNT 0`

Register offset: Rotary encoder count

4.4.2.10 `#define ROT_MAX_CNT 12`

Register offset: Rotary encoder maximum allowed count

4.4.2.11 `#define ROT_MIN_CNT 16`

Register offset: Rotary encoder minimum allowed count

4.4.2.12 `#define ROT_RSVD0 20`

Register offset: Rotary encoder reserved 0

4.4.2.13 `#define ROT_STEP 4`

Register offset: Rotary encoder step size

4.4.2.14 #define ROT_STS 8

Register offset: Rotary encoder status

4.4.3 Typedef Documentation

4.4.3.1 typedef enum CHIP LCD_CHIP

Available LCD Chips

4.4.3.2 typedef enum CNTL_BIT LCD_CNTL_BIT

LCD Control Bits

4.4.4 Enumeration Type Documentation

4.4.4.1 enum CHIP

Available LCD Chips

Enumerator

- CHIP1** Chip 1 (left)
- CHIP2** Chip 2 (right)

4.4.4.2 enum CNTL_BIT

LCD Control Bits

Enumerator

- RESET_N** Active-low reset
- EN_OP** Falling edge triggered operation enable
- REG_SEL** Register select (0 = instruction, 1 = command)
- CS1** Chip Select 1 (left)
- CS2** Chip Select 2 (right)

4.4.5 Function Documentation

4.4.5.1 XStatus PMODIO_LCD_clrCntlBit (PMODIO * inst_p, LCD_CNTL_BIT bit)

Clears the specified LCD control bit without modifying others

Parameters

<i>inst_p</i>	PMODIO instance address
<i>bit</i>	bit to clear

Returns

XST_SUCCESS

4.4.5.2 void PMODIO_LCD_delay ()

Executes a NOP loop to meet LCD timing requirements

Returns

NONE

4.4.5.3 XStatus PMODIO_LCD_exec (PMODIO * inst_p)

Instructs the LCD to act on the data/command on the data bus

The LCD uses the falling edge of the EN pin as an indicator that the data bus has data to be consumed. The RS pin indicates if the data bus contains an instruction (0) to be executed, or data (1) to be written to display ram. Delay no-op loops are used to meet the LCD timing requirements.

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

XST_SUCCESS

4.4.5.4 XStatus PMODIO_LCD_execCmd (PMODIO * inst_p, u8 cmd, LCD_CHIP chip)

Writes a command to the specified LCD chip

Parameters

<i>inst_p</i>	PMODIO instance address
<i>cmd</i>	the command to execute
<i>chip</i>	the chip to execute the command for

Returns

XST_SUCCESS

Note

The LCD uses its 8-bit data bus both for data and for commands.

4.4.5.5 XStatus PMODIO_LCD_init (PMODIO * inst_p)

Initializes the LCD

Parameters

<i>inst_p</i>	PMODIO instance address
---------------	-------------------------

Returns

XST_SUCCESS

4.4.5.6 XStatus PMODIO_LCD_setCntlBit (PMODIO * inst_p, LCD_CNTL_BIT bit)

Sets the specified LCD control bit without modifying others

Parameters

<i>inst_p</i>	PMODIO instance address
<i>bit</i>	bit to set

Returns

XST_SUCCESS

4.4.5.7 XStatus PMODIO_LCD_writeCommit (PMODIO * *inst_p*, u8 *data*)

Commits data to LCD display RAM and driver LCD map

Parameters

<i>inst_p</i>	PMODIO instance address
<i>data</i>	data to write

Returns

XST_SUCCESS

Note

The LCD's current X position will be automatically incremented by 1 on every write.

4.4.5.8 XStatus PMODIO_LCD_writeDataBits (PMODIO * *inst_p*, u8 *pattern*)

Writes a specified pattern to the LCD data bus

Parameters

<i>inst_p</i>	PMODIO instance address
<i>pattern</i>	bit pattern to write to LCD data bus

Returns

XST_SUCCESS

Note

The LCD uses its 8-bit data bus both for data and for commands.

Index

base_addr
 PMODIO, 6

CHIP1
 PMODIO_I.h, 28

CHIP2
 PMODIO_I.h, 28

CS1
 PMODIO_I.h, 28

CS2
 PMODIO_I.h, 28

CHIP
 PMODIO_I.h, 28

CNTL_BIT
 PMODIO_I.h, 28

current_pos
 PMODIO, 6

DELAY_LEN
 PMODIO_I.c, 20

EN_OP
 PMODIO_I.h, 28

GP_RSVD0
 PMODIO_I.h, 26

GP_RSVD1
 PMODIO_I.h, 26

GP_RSVD2
 PMODIO_I.h, 26

LCD_CHIP
 PMODIO_I.h, 28

LCD_CHIP_OFF
 PMODIO.h, 13

LCD_CHIP_ON
 PMODIO.h, 13

LCD_CNTL
 PMODIO_I.h, 26

LCD_CNTL_BIT
 PMODIO_I.h, 28

LCD_COL_SET
 PMODIO.h, 13

LCD_COLS
 PMODIO.h, 13

LCD_DATA
 PMODIO_I.h, 26

LCD_DISP_START
 PMODIO.h, 13

LCD_PAGE_HEIGHT
 PMODIO.h, 13

LCD_PAGES
 PMODIO.h, 14

LCD_ROW_SET
 PMODIO.h, 14

LCD_RSVD0
 PMODIO_I.h, 26

LCD_X_MAX
 PMODIO.h, 14

LCD_Y_MAX
 PMODIO.h, 14

LCD_pos, 5
 page, 5
 x, 5
 y, 5

lcd_map
 PMODIO, 6

nop
 PMODIO_I.c, 21

PMODIO_I.h
 CHIP1, 28
 CHIP2, 28
 CS1, 28
 CS2, 28
 EN_OP, 28
 REG_SEL, 28
 RESET_N, 28

PMODIO, 5
 base_addr, 6
 current_pos, 6
 lcd_map, 6

PMODIO.c, 7
 PMODIO_LCD_clr, 9
 PMODIO_LCD_getPos, 9
 PMODIO_LCD_setPos, 9
 PMODIO_LCD_write, 10
 PMODIO_ROT_clr, 10
 PMODIO_ROT_isBtnPressed, 10
 PMODIO_ROT_read, 10
 PMODIO_ROT_setMaxCount, 11
 PMODIO_ROT_setMinCount, 11
 PMODIO_ROT_setStep, 11
 PMODIO_init, 8
 PMODIO_selfTest, 11
 ROT_BTN_MSK, 8

PMODIO.h, 12
 LCD_CHIP_OFF, 13
 LCD_CHIP_ON, 13
 LCD_COL_SET, 13

LCD_COLS, [13](#)
 LCD_DISP_START, [13](#)
 LCD_PAGES, [14](#)
 LCD_ROW_SET, [14](#)
 LCD_X_MAX, [14](#)
 LCD_Y_MAX, [14](#)
 PMODIO_LCD_clr, [14](#)
 PMODIO_LCD_getPos, [16](#)
 PMODIO_LCD_setPos, [16](#)
 PMODIO_LCD_write, [16](#)
 PMODIO_ROT_clr, [17](#)
 PMODIO_ROT_isBtnPressed, [17](#)
 PMODIO_ROT_read, [17](#)
 PMODIO_ROT_setMaxCount, [17](#)
 PMODIO_ROT_setMinCount, [18](#)
 PMODIO_ROT_setStep, [18](#)
 PMODIO_init, [14](#)
 PMODIO_selfTest, [18](#)
 PMODIO_LCD_clr
 PMODIO.c, [9](#)
 PMODIO.h, [14](#)
 PMODIO_LCD_clrCntlBit
 PMODIO.l.c, [22](#)
 PMODIO.l.h, [28](#)
 PMODIO_LCD_delay
 PMODIO.l.c, [23](#)
 PMODIO.l.h, [28](#)
 PMODIO_LCD_exec
 PMODIO.l.c, [23](#)
 PMODIO.l.h, [29](#)
 PMODIO_LCD_execCmd
 PMODIO.l.c, [23](#)
 PMODIO.l.h, [29](#)
 PMODIO_LCD_getPos
 PMODIO.c, [9](#)
 PMODIO.h, [16](#)
 PMODIO_LCD_init
 PMODIO.l.c, [23](#)
 PMODIO.l.h, [29](#)
 PMODIO_LCD_setCntlBit
 PMODIO.l.c, [24](#)
 PMODIO.l.h, [29](#)
 PMODIO_LCD_setPos
 PMODIO.c, [9](#)
 PMODIO.h, [16](#)
 PMODIO_LCD_write
 PMODIO.c, [10](#)
 PMODIO.h, [16](#)
 PMODIO_LCD_writeCommit
 PMODIO.l.c, [24](#)
 PMODIO.l.h, [30](#)
 PMODIO_LCD_writeDataBits
 PMODIO.l.c, [24](#)
 PMODIO.l.h, [30](#)
 PMODIO_ROT_clr
 PMODIO.c, [10](#)
 PMODIO.h, [17](#)
 PMODIO_ROT_isBtnPressed
 PMODIO.c, [10](#)
 PMODIO.h, [17](#)
 PMODIO_ROT_read
 PMODIO.c, [10](#)
 PMODIO.h, [17](#)
 PMODIO_ROT_setMaxCount
 PMODIO.c, [11](#)
 PMODIO.h, [17](#)
 PMODIO_ROT_setMinCount
 PMODIO.c, [11](#)
 PMODIO.h, [18](#)
 PMODIO_ROT_setStep
 PMODIO.c, [11](#)
 PMODIO.h, [18](#)
 PMODIO_init
 PMODIO.c, [8](#)
 PMODIO.h, [14](#)
 PMODIO.l.c, [19](#)
 DELAY_LEN, [20](#)
 nop, [21](#)
 PMODIO_LCD_delay, [23](#)
 PMODIO_LCD_exec, [23](#)
 PMODIO_LCD_execCmd, [23](#)
 PMODIO_LCD_init, [23](#)
 PMODIO_LCD_writeCommit, [24](#)
 PMODIO.l.h, [25](#)
 CHIP, [28](#)
 CNTL_BIT, [28](#)
 GP_RSVD0, [26](#)
 GP_RSVD1, [26](#)
 GP_RSVD2, [26](#)
 LCD_CHIP, [28](#)
 LCD_CNTL, [26](#)
 LCD_CNTL_BIT, [28](#)
 LCD_DATA, [26](#)
 LCD_RSVD0, [26](#)
 PMODIO_LCD_delay, [28](#)
 PMODIO_LCD_exec, [29](#)
 PMODIO_LCD_execCmd, [29](#)
 PMODIO_LCD_init, [29](#)
 PMODIO_LCD_writeCommit, [30](#)
 PMODIO_mReadReg, [26](#)
 PMODIO_mWriteReg, [27](#)
 ROT_CNT, [27](#)
 ROT_MAX_CNT, [27](#)
 ROT_MIN_CNT, [27](#)
 ROT_RSVD0, [27](#)
 ROT_STEP, [27](#)
 ROT_STS, [27](#)
 PMODIO_mReadReg
 PMODIO.l.h, [26](#)
 PMODIO_mWriteReg
 PMODIO.l.h, [27](#)
 PMODIO_selfTest
 PMODIO.c, [11](#)
 PMODIO.h, [18](#)
 page
 LCD_pos, [5](#)

REG_SEL
 PMODIO_I.h, [28](#)
RESET_N
 PMODIO_I.h, [28](#)
ROT_BTN_MSK
 PMODIO.c, [8](#)
ROT_CLR_CLR_MSK
 PMODIO.c, [8](#)
ROT_CLR_SET_MSK
 PMODIO.c, [8](#)
ROT_CNT
 PMODIO_I.h, [27](#)
ROT_CNT_ABS_MAX
 PMODIO.h, [14](#)
ROT_CNT_ABS_MIN
 PMODIO.h, [14](#)
ROT_MAX_CNT
 PMODIO_I.h, [27](#)
ROT_MIN_CNT
 PMODIO_I.h, [27](#)
ROT_RSVD0
 PMODIO_I.h, [27](#)
ROT_STEP
 PMODIO_I.h, [27](#)
ROT_STS
 PMODIO_I.h, [27](#)

SELF_TEST_PATTERN
 PMODIO.c, [8](#)

x
 LCD_pos, [5](#)

y
 LCD_pos, [5](#)