PMODIO

Generated by Doxygen 1.8.6

Fri Mar 18 2016 23:13:56

Contents

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

2 File Index

2.1 File List

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

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 PMODIO peripheral in the system

3.2.2 Field Documentation

3.2.2.1 u32 PMODIO::base_addr

device base address

3.2.2.2 LCD_pos PMODIO::current_pos

current LCD write position

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

static array to track writes to LCD

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

- PMODIO.h

4 File Documentation

4.1 PMODIO.c File Reference

```
#include "stdbool.h"
#include "xparameters.h"
#include "xil_types.h"
#include "xstatus.h"
#include "PMODIO.h"
#include "PMODIO_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 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)
- LCD_pos PMODIO_LCD_getPos (PMODIO *inst_p)
- XStatus PMODIO_LCD_write (PMODIO *inst_p, u8 data, u8 mask)
- XStatus PMODIO_LCD_clr (PMODIO *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

inst_p	PMODIO instance address
base_addr	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

inst_p	PMODIO instance address

Returns

XST_SUCESS

4.1.3.3 LCD_pos PMODIO_LCD_getPos (PMODIO * inst_p)

Retrieves the current LCD write position

Parameters

inst_p	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

inst_p	PMODIO instance address
x	x-coordinate
y	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

inst_p	PMODIO instance address
data	data to write
mask	mask to select which bits are written

Returns

XST_SUCESS

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 cyles 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

inst_p	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

inst_p	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

inst_p	PMODIO instance address
rotary_cnt_p	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

inst_p	PMODIO instance address
count	the maximum allowed count

Returns

- XST_SUCCESS if max count input is valid (<= 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

inst_p	PMODIO instance address
count	the maximum allowed count

Returns

- XST_SUCCESS if min count input is valid (>= -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

inst_p	PMODIO instance address
step	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

inst_p	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

inst_p	PMODIO instance address
base_addr	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

inst_p	PMODIO instance address

Returns

XST_SUCESS

4.2.3.3 LCD_pos PMODIO_LCD_getPos (PMODIO * *inst_p*)

Retrieves the current LCD write position

Parameters

inst_p	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

inst_p	PMODIO instance address
x	x-coordinate
y	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

inst_p	PMODIO instance address
data	data to write
mask	mask to select which bits are written

Returns

XST_SUCESS

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 cyles 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

inst_p	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

inst_p	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

inst_p	PMODIO instance address
rotary_cnt_p	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

inst_p	PMODIO instance address
count	the maximum allowed count

Returns

- XST_SUCCESS if max count input is valid (<= 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

inst_p	PMODIO instance address
count	the maximum allowed count

Returns

- XST_SUCCESS if min count input is valid (>= -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

inst_p	PMODIO instance address
step	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

inst_p	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_l.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

inst_p	PMODIO instance address
bit	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

inst_p	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

inst_p	PMODIO instance address
cmd	the command to execute
chip	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

inst_p	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

inst_p	PMODIO instance address
bit	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

inst_p	PMODIO instance address
data	data to write

Returns

XST_SUCESS

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

inst_p	PMODIO instance address
pattern	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_l.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 PMODIO peripheral driver.

Warning

This file is only intended for PMODIO driver use, and should not be directly included by applications. Use PMODIO.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 PMODIO_mReadReg(*BaseAddress, RegOffset*) Xil_In32((BaseAddress) + (RegOffset))

Read a value from a PMODIO 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

BaseAddress	is the base address of the PMODIO device.
RegOffset	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

BaseAddress	is the base address of the PMODIOdevice.
RegOffset	is the register offset from the base to write to.
Data	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

inst_p	PMODIO instance address
bit	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

inst_p	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

inst_p	PMODIO instance address
cmd	the command to execute
chip	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

inst_p	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

inst_p	PMODIO instance address
bit	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

inst_p	PMODIO instance address
data	data to write

Returns

XST_SUCESS

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

inst_p	PMODIO instance address
pattern	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