ECE 544 Pmod544IO Driver Documentation

Roy Kravitz
Version 1.0
30-Dec-2014

Include Files

```
#include "xil_types.h"
#include "xil_io.h"
#include "xstatus.h"
#include "stdbool.h"
#include "PMod544IOR2_l.h"
```

Macros

Bit Masks*Bit masks for the PModECE544IO registers.*

All of the registers in the PModECE544IO peripheral are 32-bits wide

#define PMOD544IO_SELFTEST_MSK 0x80000000
#define PMOD544IO_LCDBUSY_MSK 0x00008000
#define PMOD544IO_ENCBUSY_MSK 0x00000080
#define **PMOD544IO_ENCSW_MSK** 0x00000002
#define **PMOD544IO_ENCBTN_MSK** 0x00000001
#define **PMOD544IO_CLRCNT_MSK** 0x00000080
#define **PMOD544IO_LDCFG_MSK** 0x00000040
#define **PMOD544IO_NONEG_MSK** 0x00000010
#define PMOD544IO_INCDECCNT_MSK 0x0000000F
#define PMOD544IO_ROTENC_COUNT_MSK 0x0000FFFF
#define PMOD544IO_ROTENC_CNTLO_MSK 0x000000FF
#define PMOD544IO_ROTENC_CNTHI_MSK 0x0000FF00
#define PMOD544IO_LCD_DOCMD_MSK 0x00000080
#define PMOD544IO_LCD_CMD_MSK 0x0000001F
#define PMOD544IO_LCD_DATA_MSK 0x000000FF

Literals and constants

Literals and constants used for selecting specific devices

- enum **_PMOD544IO_lcdcmds** { **LCD_NOP** = 0x00, **LCD_SETCURSOR** = 0x01, **LCD_WRITECHAR** = 0x02, **LCD_READCHAR** = 0x03, **LCD_CLRD** = 0x04, **LCD_HOME** = 0x05, **LCD_SETCGADDR** = 0x06, **LCD_SETDDADDR** = 0x07, **LCD_SETMODE** = 0x08, **LCD_SETONOFF** = 0x09, **LCD_SHIFTLEFT** = 0x0A, **LCD_SHIFTRGHT** = 0x0B, **LCD_MOVELEFT** = 0x0C, **LCD_MOVERGHT** = 0x0D, **LCD_RSVD00** = 0x0E, **LCD_RSVD01** = 0x0F }

Function Prototypes

- int **PMDIO_initialize** (u32 BaseAddr)
- int **PMDIO_ROT_init** (int inc_dec_cnt, bool no_neg)
- int **PMDIO_ROT_clear** (void)
- int **PMDIO_ROT_readRotcnt** (int *RotaryCnt)
- bool **PMDIO_ROT_isBtnPressed** (void)
- bool **PMDIO_ROT_isSwOn** (void)
- int **PMDIO_LCD_docmd** (u32 lcdcmd, u32 lcddata)
- int **PMDIO_LCD_setcursor** (u32 row, u32 col)
- int **PMDIO_LCD_wrchar** (char ch)
- int **PMDIO_LCD_shiftl** (void)
- int **PMDIO_LCD_shiftr** (void)
- int **PMDIO_LCD_setcgadr** (u32 addr)
- int **PMDIO_LCD_setddadr** (u32 addr)

- int **PMDIO_LCD_clrd** (void)
- char * **PMDIO_LCD_itoa** (int value, char *string, int radix)
- int **PMDIO_LCD_wrstring** (char *s)
- int **PMDIO_LCD_puthex** (u32 num)
- int **PMDIO_LCD_putnum** (int num, int radix)

Detailed Description

Author:

Roy Kravitz (`roy.kravitz@pdx.edu`)

Copyright:

Portland State University, 2014, 2015

This header file contains identifiers and driver function definitions for the PMod544IO custom peripheral. The peripheral provides access to the PMOD's required for ECE544. These are:

- PMODCLP - A 2 line by 16 character LCD with an 8-bit parallel interface
- PMODENC - A rotary encoder with a pushbutton and slide switch

The PMODCLP and PMODENC are both controlled by Picoblaze-based firmware in the peripheral. The rotary encoder pushbutton and slide switch are debounced in hardware with an instance of debounce.v (also used in the Nexys4IO custom peripheral).

```
MODIFICATION HISTORY:
```

```
Ver   Who  Date     Changes----- ---- --------
-----------------------------------------------1.00a   rhk 12/20/14    First release of driver
```

Note:

The function prototypes were made to be (mostly) compatible with the N4EIF driver API used in previous terms. This was done to simplify conversion of existing code.

Function Documentation

int PMDIO_initialize (u32 *BaseAddr*)

Initialize the PModECE544IO peripheral driver

Saves the Base address of the PModECE544IO registers and runs the self test (only the first time the peripheral is initialized). If the self-test passes the function sets the rotary encoder mode and clears the rotary encoder count. Finishes off by clearing the LCD.

Parameters:

BaseAddr	is the base address of the NEXYS4IO register set

Returns:

XST_SUCCESS Initialization was successful.

Note:

This function can hang if the peripheral was not created correctly
The Base Address of the PmodECE544IO peripheral will be in *xparameters.h*

int PMDIO_LCD_clrd (void)

LCD_clrd() - Clear the display

Writes blanks to the display and returns the cursor home.

Parameters:

NONE	

Returns:

XST_SUCCESS

int PMDIO_LCD_docmd (u32 *lcdcmd*, u32 *lcddata*)

Execute an LCD command

Executes the LCD command in "lcdcmd" using the data in "lcdata". Controls the handshaking between the driver and the peripheral.

Parameters:

lcdcmd	ia rhw LCD command to execute
lcddata	is the data for the command. Not all commands have data

Returns:

XST_SUCCESS

Note:

only the lower 8-bits of lcdcmd and lcdata are used

char* PMDIO_LCD_itoa (int *value*, char * *string*, int *radix*)

Converts an integer to ASCII characters

algorithm borrowed from ReactOS system libraries

Converts an integer to ASCII in the specified base. Assumes string[] is long enough to hold the result plus the terminating null

Parameters:

value	is the integer to convert
**string*	is the address of buffer large enough to hold the converted number plus the terminating null
radix	is the base to use in conversion,

Returns:

pointer to the return string

Note:

No size check is done on the return string size. Make sure you leave room for the full string plus the terminating null

int PMDIO_LCD_puthex (u32 *num*)

Write a 32-bit unsigned hex number to LCD display in Hex

Writes 32-bit unsigned number to the LCD display starting at the current cursor position.

Parameters:

num	is the number to display as a hex value

Returns:

XST_SUCCESS

Note:

No size checking is done to make sure the string will fit into a single line, or the entire display, for that matter. Watch your string sizes.

int PMDIO_LCD_putnum (int *num*, int *radix*)

Write a 32-bit number in Radix "radix" to LCD display

Writes a 32-bit number to the LCD display starting at the current cursor position. "radix" is the base to output the number in.

Parameters:

num	is the number to display
radix	is the radix to display number in

Returns:

XST_SUCCESS

Note:

No size checking is done to make sure the string will fit into a single line, or the entire display, for that matter. Watch your string sizes.

int PMDIO_LCD_setcgadr (u32 *addr*)

Set the character generator RAM Address

Sets the CG RAM address to the value in "addr". The function also serves the purpose of telling the LCD controller that the character data should be written to the character generator RAM instead of the data RAM. The character generator RAM contains 8 user defined custom characters

Parameters:

addr	is the address in tha character generator ROM

Returns:

XST_SUCCESS

Note:

: only the low order 6-bits of the address are used

int PMDIO_LCD_setcursor (u32 *row*, u32 *col*)

Position the LCD cursor at {line, col}

Position the cursor at the specified position. The next character will be written there.

The display is formed of 2 lines of 16 characters and each position has a corresponding address as indicated below.

```
            Character position
    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

Line 1 - 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F Line 2 - C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF

Parameters:

line	is the row on the display to set the cursor to. There are two lines
col	in the character position on the line. There are 16 characters per line

Returns:

XST_SUCCESS

int PMDIO_LCD_setddadr (u32 *addr*)

LCD_setddadr() - Set the data RAM Address

Sets the data RAM address to the value in "addr". The function also serves the purpose of telling the LCD controller that the character data should be written to the display RAM instead of the character generator RAM.

Parameters:

addr	is the address in the diplay RAM

Returns:

XST_SUCCESS

Note:

use LCD_setcursor() to set the position using {row, col} addressing
only the low order 7-bits of the address are used

int PMDIO_LCD_shiftl (void)

Shifts the entire display left one position

Shifts the display left one position without changing display RAM contents. When the displayed data is shifted repeatedly, both lines move horizontally. The second display line does not shift into the first display line.

Parameters:

NONE	

Returns:

XST_SUCCESS

int PMDIO_LCD_shiftr (void)

Shifts the entire display right one position

Shifts the display right one position without changing display RAM contents. When the displayed data is shifted repeatedly, both lines move horizontally. The second display line does not shift into the first display line.

Parameters:

NONE	

Returns:

XST_SUCCESS

int PMDIO_LCD_wrchar (char *ch*)

Write a character to the LCD display

Writes an ASCII character to the LCD display at the current cursor position

Parameters:

ch	is the character to write to the display

Returns:

XST_SUCCESS

int PMDIO_LCD_wrstring (char * *s*)

Write a 0 terminated string to the LCD display

Writes the null terminated string "s" to the LCD display starting at the current cursor position

Parameters:

**s*	is a pointer to the string to display

Returns:

XST_SUCCESS

Note:

No size checking is done to make sure the string will fit into a single line, or the entire display, for that matter. Watch your string sizes.

int PMDIO_ROT_clear (void)

Clear the rotary encoder count

Sets the rotary encoder count back to 0

Parameters:

NONE	

Returns:

XST_SUCCESS

int PMDIO_ROT_init (int *inc_dec_cnt*, bool *no_neg*)

Initialize the Rotary Encoder control logic

Configures the rotary encoder logic

Parameters:

inc_dec_cnt	is the count for how much the rotary encorder increments or decrements each time the rotary encoder is turned. The count is truncated to 4 bits
no_neg	permits or prevents the rotary count from going below 0., no_neg true say do not allow negative counts.

Returns:

XST_SUCCESS

Note:

Although it should be possible to change the configuration of the rotary encoder logic dynamically this is not recommended. There is a subtle bug that prevents consistent results. So it's best to cnly configure the rotary encoder logic during initialization. You could, of course, try it but caveat emptor.

bool PMDIO_ROT_isBtnPressed (void)

returns the state of the rotary encoder pusbbutton

Reads the ROTLCD_STS register to determine whether the rotary encoder shaft pushbutton is pressed

Parameters:

NONE	

Returns:

true if the button is pressed, false otherwise

bool PMDIO_ROT_isSwOn (void)

returns the state of the slide switch on the PmodENC

Reads the ROTLCD_STS register to determine whether the slide switch is on (up)

Parameters:

NONE	

Returns:

true if the slide switch is on, false otherwise

int PMDIO_ROT_readRotcnt (int * *rotaryCnt_p*)

Read the rotary encoder count into the u32 variable pointed to by rotaryCnt_p

Returns the rotary count. The rotary count is a 16-bit unsigned integer returned to an unsigned int.

Parameters:

rotaryCnt_p	is a pointer to the u32 variable that will receive the rotary count

Returns:

XST_SUCCESS

PMod544IOR2_l.h File Reference

Macros

- #define **PMOD544IO_mWriteReg**(BaseAddress, RegOffset, Data) Xil_Out32((BaseAddress) + (RegOffset), (u32)(Data))
- #define **PMOD544IO_mReadReg**(BaseAddress, RegOffset) Xil_In32((BaseAddress) + (RegOffset))

Registers*Register offsets for this device.*

#define PMOD544IO_ROTLCD_STS_OFFSET 0
#define PMOD544IO_ROT_CNTRL_OFFSET 4
#define PMOD544IO_ROT_COUNT_OFFSET 8
#define PMOD544IO_LCD_CMD_OFFSET 12
#define PMOD544IO_LCD_DATA_OFFSET 16
#define PMOD544IO_RSVD00_OFFSET 20
#define PMOD544IO_RSVD01_OFFSET 24
#define PMOD544IO_RSVD02_OFFSET 28

Functions

- XStatus **PMOD544IO_Reg_SelfTest** (u32 baseaddr)

Detailed Description

Author:

Roy Kravitz (`roy.kravitz@pdx.edu`)

Copyright:

Portland State University, 2014, 2015

This header file contains identifiers and low level driver functions for the PMod544IO custom peripheral. The peripheral provides access to the PMOD's required for ECE544. These are:

- PMODCLP - A 2 line by 16 character LCD with an 8-bit parallel interface
- PMODENC - A rotary encoder with a pushbutton and slide switch

The PMODCLP and PMODENC are both controlled by Picoblaze-based firmware in the peripheral. The rotary encoder pushbutton and slide switch are debounced in hardware with an instance of *debounce.v* (also used in the Nexys4IO custom peripheral).

```
MODIFICATION HISTORY:
```

Ver	Who	Date	Changes
1.00a	rhk	12/20/14	First release of driver

Macro Definition Documentation

#define PMOD544IO_mReadReg(BaseAddress, RegOffset) Xil_In32((BaseAddress) + (RegOffset))

Read a value from a PMOD544IO 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 PMOD544IO 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 **PMOD544IO_mReadReg(u32 BaseAddress, unsigned RegOffset)**

#define PMOD544IO_mWriteReg(BaseAddress, RegOffset, Data) Xil_Out32((BaseAddress) + (RegOffset), (u32)(Data))

Write a value to a PMOD544IO 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 PMOD544IOdevice.
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 **PMOD544IO_mWriteReg(u32 BaseAddress, unsigned RegOffset, u32 Data)**

Function Documentation

XStatus PMOD544IO_Reg_SelfTest (u32 *baseaddr*)

Run a self-test on the driver/device. Note this may be a destructive test if resets of the device are performed.

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

Parameters:

baseaddr	is the base address of the PMOD544IO instance to be worked on.

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.
This test assume the existence of a Serial port in the system (used for xil_printf)

Run a self-test on the driver/device. Note this may be a destructive test if resets of the device are performed.

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

Parameters:

baseaddr_p	is the base address of the PMOD544IOR2instance to be worked on.

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.

Index