Pmod544IO Product Guide
Revision 1.1
(Last Updated 30-Dec-2015)

Overview

NOTE: ALL REFERENCES TO NEXYS™4 REFER TO BOTH THE DIGILENT NEXYS4 DDR AND THE ORIGINAL NEXYS4 FPGA DEVELOPMENT BOARDS.

The Digilent Nexys4 board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7™ Field Programmable Gate Array (FPGA) from Xilinx. With its large, high-capacity FPGA (Xilinx part number XC7A100T-1CSG324C), generous external memories, and collection of USB, Ethernet, and other ports, the Nexys4 can host designs ranging from introductory combinational circuits to powerful embedded processors. Several built-in peripherals, including an accelerometer, temperature sensor, MEMs digital microphone, a speaker amplifier, and a lot of I/O devices allow the Nexys4 to be used for a wide range of designs without needing any other components.

While the Nexys4 board has an excellent feature set for its price, it is lacking several functions that we use in ECE 544. There is no text display and there are no human interface input devices except the pushbuttons and switches. Fortunately, Digilent provides the missing functionality through the PMOD expansion connectors on the board. For ECE 544 we will add key functionality with a PmodCLP (2 line by 16 character LCD) and a PmodENC (Rotary Encoder with pushbutton and slide switch).

Functional Description

Pmod544IO is a package of custom hardware and driver code that provides a register-based interface to the Pmod's used in ECE 544. *Pmod544IO* works in conjunction with *Nexys4IO* (another package of custom IP and drivers) to provide a consistent and (hopefully) easy to use interface to the PmodCLP and PmodENC. Providing this IP allows you to focus on your applications - we've done much of the hard work for you. Specifically, the *Pmod544IO* and *Nexys4IO* duo provides the following capabilities:

- Read the debounced push-buttons and switches, including the push-button and slide switch on the PmodENC.
- Control of the discrete LEDs on the Nexys4 board
- Control of the two RGB LEDs on the Nexys4 board
- Control of the 8 seven segment display digits and decimal points on the Nexys4
- Control and status (including the count) of the PmodENC rotary encoder
- Support for the 2 line x 16 character display on the PmodCLP including writing text and numbers to the display, clearing the display, positioning the cursor, and shifting the characters on the display.

As a user of *Pmod544IO* you will be able to configure and get count and switch and button state information from the PmodENC and control the LCD display from a Microblaze using C library calls.

Feature Summary

- AXI Lite slave interface to PmodCLP and PmidENC providing command-driven, providing memory mapped register-based access to a Microblaze or Zynq-based embedded system.
- Packaged as IP for Vivado. Can be added to your design using either the Vivado project manager or the IP Integrator.

Required Hardware

Pmod544IO assumes that a single PmodCLP has been connected to the JA and JB PMod headers on the Nexys4 and that a PmodENC has been connected to the bottom row of the PMod JD header on the board. There is nothing that precludes using *Pmod544IO* on other systems with PMod ports, the Digilent ZYBO board, for example, other than changing the mapping of the PMod headers to correspond to the selected development board. Refer to the Port Description section of this document for the interface signal names and functionality.

To add an instance of *Pmod544IO* to an existing design make the **IP Catalog** tab visible. Use the search box to locate/select *Pmod544IO* and double click on it or right click and select *Add IP*. In the Block Design editor select either individually, or as a group, all of the inputs and outputs of *Pmod544IO* and make them *External*. You may use the Connection Wizard in the Block editor to connect the *Pmod544IO* to the AXI bus

NOTE: IF YOU DO NOT SEE PMOD544IO IN THE IP CATALOG YOU NEED TO MAKE THE IP INTEGRATOR AWARE OF THE LOCATION OF YOUR REPOSITORY. THIS CAN BE DONE BY ADDING THE REPOSITORY LOCATION TO YOUR PROJECT OPTIONS.

IMPORTANT: IT IS BEST TO INSERT/REMOVE THE PMODCLP AND PMODENC BOARDS ONLY WHEN THE BOARD IS POWERED-DOWN.

Device Drivers

The Pmod544IO hardware is supported by a driver library that provides a (hopefully) easy-to-use application program interface (API). The API is provided as C source code that should be automatically included in the software board support package for a project that includes an instance of Pmod54IO. The drivers are implemented for the Xilinx standalone OS and are not thread-safe (i.e. could cause problems in multitasking applications such as those build with Xilkernel). The API provides the following functionality:

- Base-level functionality for the most commonly used functions provided by the PmodCLP LCD panel. Functions include write character, write string, set cursor, clear display, and shift the display contents.

- Base-level functionality for the rotary encoder. Functions include get rotary count , set amount to increment/decrement the count on each turn, and enable/disable a negative count

The API is documented separately in the *Pmod544IO Device Driver User Guide* and is also available in HTML format in the /doc directory for the Pmod544IO drivers directory and can be accessed by your favorite browser by navigating to the html/index.html. This documentation was generated from the driver source code using doxygen (http://doxygen.org).

Chapter 2: Product Specification

Port Descriptions

The *PMod544IO* peripheral provides an AXI4_Lite slave interface to a Microblaze-based system. It also provides the interface signals between the PMod connectors on the Nexys4 board and a PmodCLP™ (Digilent parallel display) and PmodENC™ (Digilent rotary encoder).

Signal Name	Interface	I/O	Initial State	Description
s_axi_aclk	Clock	I	NA	AXI Clock
s_axi_aresetn	Reset	I	NA	AXI Reset, active-Low
s_axi_*	S_AXI	NA	NA	AXI4-Lite Slave Interface signals. See Appendix A of the AXI Reference Guide (UG761) for AXI4-Lite signals
PmodENC_A, PmodENC_B	PMod544IO	I	NA	A and B inputs from the PModENC rotary encoder. The two signals are quadrature encoded such that the direction of rotation can be determined.
PmodENC_BTN,	PMod544IO	I	0000h	Rotary encoder pushbutton on the PmodENC. This signal is debounced and made available to an application through the ROTLCD_STS register
PmodENC_SWT	PMod544IO	I	0h	Slide switch on the PmodENC. This signal is debounced and made available to an application through the ROTLCD_STS register
PmodCLP_RS, PmodCLP_RW, PmodCLP_E	PMod544IO	O	0h	Control signals for the PmodCLP display. PmodCLP_RS is used to select between the data register and the control register on the display. PmodCLP_RW specifies whether the display should perform a read or a write and PmodCLP_E is the enable signal for the display.
PmodCLP_DataBus[7:0]	PMod544IO	O	0h	8-bit data bus to the display

Register Space

The registers for *Pmod544IO* are mapped into the Microblaze address space (memory mapped I/O as is typically the case for RISC CPU's. The base address for the Pmod544IO register space is included in `xparameters.h` which is generated when the board support package is built.

Address Offset	Register Name	Description
0h	ROTLCD_STS	Rotary encoder and LCD status register. Supports handshaking between the peripheral and its driver.
04h	ROT_CNTRL	Rotary encoder control. Supports configuration and handshaking between the peripheral and its driver
08h	ROT_COUNT	Rotary encoder count. Increments or decrements depending on the direction the rotary encoder shaft is rotated. The rotary count is a 16-bit unsigned integer
0Ch	LCD_CMD	LCD command register. Supports handshaking between the peripheral and its driver. The peripheral supports most of the possible operations (ex: write character, clear display, move cursor, etc.) that the display is capable of.
10h	LCD_DATA	LCD data register. Contains the data associated with the command in the LCD_CMD
14h-1Fh	RSVD	Reserved

Register Descriptions

Rotary Encoder/LCD Status Register (ROTLCD_STS)

This register can be read by the Microblaze to retrieve the status of the Rotary encoder and LCD PMod's. The register also contains the *PMod544IO* "self-test" flag. The bits in this register are used to support handshaking between the *PMod544IO* peripheral and the peripheral driver.

NOTE: THIS IS A READ-ONLY REGISTER. WRITES TO THIS REGISTER ARE IGNORED; THE REGISTER WILL RETURN THE CURRENT STATUS WHENEVER IT IS READ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SelfTst	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LCDBusy	Reserved							ENCBusy	Reserved					ENCSw	ENCBtn

Note: ROTLCD_STS[15:8] are also referred to as the LCD_STATUS register and ROTLCD[7:0] are also referred to as the ROTENC_STATUS register.

Bit	Name	Access Type	Reset Value	Description
31	SelfTst	Read	0	Self-test flag. Returns 1 when the PModECE544IO

				peripheral is executing its self-test function, 0 otherwise. The driver should check this bit when it is being initialized before performing any operations
30-16	0	Read	0	Returns 0s.
15	LCDBusy	Read	0	LCD Busy. Returns 1 when an LCD operation is in progress; 0 when the LCD is ready to perform an operation. The driver should check this bit before starting an operation.
14-8	Reserved	Read	N/A	Reserved
7	ENCBusy	Read	0	Rotary Encoder Busy. Returns 1 when a Rotary Encoder operation (such as updating the rotary count) is in progress; 0 when the rotary encoder is ready to perform an operation. The driver should check this bit before starting an operation.
6-2	Reserved	Read	N/A	Reserved
1	ENCSw	Read	-	PmodENC slide switch state. Returns 1 when the switch is "on" (up), 0 otherwise. The switch is debounced by the PModECE544IO logic.
0	ENCBtn	Read	-	Rotary encoder pushbutton state. Returns 1 when the shaft of the rotary encoder is pressed, 0 otherwise. The pushbutton is debounced by the PModECE544IO logic.

Rotary Encoder Control Register (ROT_CNTRL)

This register can be written by the Microblaze to configure the rotary encoder or clear the rotary encoder count. The bits in this register are used to support handshaking between the rotary encoder logic in the *PMod544IO* peripheral and the peripheral driver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reserved								ClrCnt	LdCfg	0	NoNeg	IncDecCnt			

Bit	Name	Access Type	Reset Value	Description
31-16	0	Read/Write	0000h	Returns 0 on a Read. Ignored on a Write
15-8	Reserved	N/A	-	Reserved
7	ClrCnt	Read/Write	0h	Clear Rotary Count. The driver should toggle this signal from 0->1->0 to clear the rotary count.
6	LdCfg	Read/Write	0h	Load Rotary Encoder logic configuration. The

				driver should toggle this signal from 0->1->0 to configure the rotary encoder logic. NOTE: The Rotary encoder logic should only be configured once during initialization. The logic is not designed to support dynamic configuration.
5	0	Read/Write	0h	Returns 0 on a Read. Ignored on a write
4	NoNeg	Read/Write	0h	"No Negative count" configuration bit. Write a 1 to this bit when configuring the Rotary encoder logic to keep the rotary encoder count >=0. Writing a 0 to the bit allows the rotary count to decrement and wrap around to 0xFFFF
3-0	IncrDecCnt	Read/Write	1h	Increment/Decrement count configuration bit. Write a non-zero value to set the amount the rotary encoder count increases or decreases per shaft rotation. For example, writing a 5 to this field will cause the rotary encoder to count 0->5->10,…

Rotary Count Register (ROT_COUNT)

The *PMod544IO* peripheral supports a 16-bit unsigned rotary encoder count. The count starts at 0 and increments or decrements by the value of *IncDecCnt* every time the rotary shaft is rotated. Rotating the shaft to the right causes the count to increase. Rotating the shaft to the left causes the count to decrease. The direction can be changed in hardware by rewiring the PmodENC_A and PmodENC_B signals. The rotary count is cleared when *ClrCnt* is toggled from 0->1->0. The count will range from 0 to 0xFFFF if *NoNeg* is set during configuration even if the count should wrap around on a decrement from a count of 0. If *NoNeg* is reset during configuration the count will wrap around on a decrement from 0 to 0xFFFF. See the ROT_CNTRL register description for details.

NOTE: THIS IS A READ-ONLY REGISTER. WRITES TO THIS REGISTER ARE IGNORED; THE REGISTER WILL RETURN THE CURRENT STATUS WHENEVER IT IS READ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Rotary Count															

Bit	Name	Access Type	Reset Value	Description
31-16	0	Read	0	Returns 0's
15-0	Rotary_Count	Read	-	Rotary count. The count is cumulative from the last time it was cleared by toggling the *ClrCnt* bit in the ROT_CNTRL register

LCD Command Register (LCD_CMD)

The *PMod544IO* peripheral controls the 2 line x 16 character display on the PmodCLP. This display is based on the pseudo-industry standard *HD44780 parallel interface* controller command set and control signals. The display is capable of performing common operations for a display of this type (clear display, write character, shift the display, enable/disable cursor, etc.).

The LCD Command register supports handshaking between the peripheral and its driver and is capable of executing the following commands in conjunction with the LCD_DATA register:

Refer to the Digilent PmodCLP Reference Manual and the Samsung KS0066 (or equivalent) datasheets for details on the display.

CMD Code	Command Name	LCD Data	Description
0x00	NOP	NONE	No Operation
0x01	Set Cursor Position	LCD_DATA[7:4] – Line # LCD_DATA[3:0] – Char #	Position the cursor to (line, char). The display is formed of 2 lines (1-2) of 16 characters (0 – 15) each.
0x02	Write Character	LCD_DATA[7:0] – ASCII character code for character to be written (if address mode is set to DD). LCD_DATA[7:0] – Bit field for user generated characters (if address mode is set to CG)	Writes the character in *LCD_DATA* to the LCD display at the current cursor position if the address mode is set to DD. Writes the bit field in LCD_DATA to the LCD character generator ROM at the current CG address if the address mode is set to CG
0x03	Read Character NOTE: THIS COMMAND IS NOT IMPLEMENTED IN PMOD544IO	*lcd_sts*[6:0] – low order 7 bits of the ASCII character at the current cursor position. The Read Character Command does not use the *lcd_data* register	Returns the ASCII character code for the character at the current cursor position of the LCD display if the address is set to DD Returns the bit field at the current Character ROM address if the address mode is set to CG
0x04	Clear Display	NONE	Clears the display and returns the cursor position to line 1, position 0 (top left character position). Writes spaces (ASCII 0x20) to all of the character positions.
0x05	Return Cursor Home	NONE	Position the cursor at the home position (top-left corner). DD RAM contents are

			unaffected. Also returns the display being shifted to the original position.
0x06	Set CG Address	LCD_DATA[7:6] – Not used LCD_DATA[5:0] – New CG Address	Sets the initial CG RAM address. After this command all subsequent read or write operations to the display are to/from CG RAM (e.g. the display is in CG mode)
0x07	Set DD Address	LCD_DATA[7] – Not used LCD_DATA[6:0] – New DD (display data memory) Address.	Sets the initial DD RAM address. After this command all subsequent read or write operations to the display are to/from DD RAM (e.g. the display is in DD mode)
0x08	Set Display Entry Mode	LCD_DATA[7:2] – Not used LCD_DATA[1] – Auto Increment (1) or decrement (0) address counter LCD_DATA[0] – Shift entire display in the direction controlled by LCD_DATA[1] (1) or disable shifting (0)	Sets the cursor move direction and specifies whether or not to shift the display. These operations are performed during data reads and writes
0x09	Display On/Off	LCD_DATA[7:3] – Not used LCD_DATA[2] – Display characters (1) or don't display characters (0). LCD_DATA[1] – Display cursor (1) or do not display cursor (0) LCD_DATA[0] – Blink the cursor (1) or do not blink cursor (0)	Turns the display on or off, controlling all characters, cursor and cursor position character (underscore) blink.

0x0A	Shift Display Left	NONE	Shifts the display left one position without changing DD 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
0x0B	Shift Display Right	NONE	Shifts the display right one position without changing DD 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
0x0C	Move Cursor Left	NONE	Position the cursor one position to the left in order to modify an individual character, or to scroll the display window left to reveal additional data stored in the DD RAM beyond the 16th character on a line. The cursor automatically moves to the first line when it moves beyond the 0th location on the second line (I think)
0x0D	Move Cursor Right	NONE	Position the cursor one position to the right in order to modify an individual character, or to scroll the display window right to reveal additional data stored in the DD RAM beyond the 16th character on a line. The cursor automatically moves to the second line when it moves beyond the 40th location on the first line.

An LCD command is executed by writing the command and data values to LCD_CMD and LCD_DATA and then toggling (0->1->0) the DoCmd bit in the LCD_CMD register. The driver should check that the LCD controller is ready (ROTLCD_STS[LCDBusy] = 1'b0) before the command is executed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Reserved																								DoCmd	0	0	LCD Command				

Bit	Name	Access Type	Reset Value	Description
31-7	Reserved	Read/Write	-	Reserved

7	DoCmd	Read/Write	0h	Do Command. The driver should toggle this signal from 0->1->0 to execute the command in the CLD Command field.
6-5	0	Read/Write	00	Returns 0 on Read. Ignored on Write
4-0	LCD Command	Read/Write	0h	Specifies the command that should be executed by the LCD controller. See the previous table for commands.

LCD Data Register (LCD_DATA)

The LCD Data register works in conjunction with the LCD_CMD register to execute LCD commands. The commands are listed in the table shown for the LCD_CMD register. The safest method for executing commands on the display is to first write the command and data to LCD_CMD and LCD_DATA and then toggle LCD_CMD[DoCmd] from 0->1->0 without changing the remaining contents of LCD_CMD or LCD_DATA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Reserved																								LCD Data							

Bit	Name	Access Type	Reset Value	Description
31-8	Reserved	N/A	-	Reserved
7-0	LCD Data	Read/Write	00h	Data for the command in LCD_CMD.

Revision History

1.0	05-Jan-2015	Roy Kravitz	First release based on n4eif User Guide
1.1	30-Dec-2015	Roy Kravitz	Minor edits. No functionality changes.

Related Documents

- *Pmod544IO Device Driver User Guide*; Roy Kravitz; January 2015
- *Digilent Nexys4™ Board Reference Manual ,* Copyright Digilent, Inc.
- *Digilent PmodCLP™ Parallel LCD Reference Manual*. Digilent document 502-142 (28-Apr-2008)). Copyright Digilent, Inc.
- *Digilent PmodENC™ Reference Manual*. Digilent document 502-117 (31-Oct-2011)). Copyright Digilent, Inc.
- *Samsung KS0066 Data sheet*. Copyright Samsung, Inc.
- Chapman, Ken, *Initial Design for Spartan-3E Starter Kit (LCD Display Control);* Xilinx Application Note; 16-February-2006.
- Chapman, Ken, *Rotary Encoder Interface for Spartan-3E Starter Kit*; Xilinx Application Note; 20-February- 2006