PMODIO User Guide

Revision

Date 19 March 2016

Author Scott Lawson

*Notice: All trademarks are property of their respective owners.*

# Table of Contents

[Table of Contents i](#_Toc446125904)

[Revision History ii](#_Toc446125905)

[Overview 1](#_Toc446125906)

[Document Scope 1](#_Toc446125907)

[Project Description 1](#_Toc446125908)

[Conventions Used in this Document 1](#_Toc446125909)

[Notes 1](#_Toc446125910)

[Physical Hardware Description 2](#_Toc446125911)

[LCD Detail 3](#_Toc446125912)

[Rotary Encoder Detail 3](#_Toc446125913)

[PMODIO Peripheral Description 4](#_Toc446125914)

[Port Descriptions 4](#_Toc446125915)

[PMODIO Peripheral Architecture 5](#_Toc446125916)

[Register Map 6](#_Toc446125917)

[Driver Software 9](#_Toc446125918)

[Known Issues 11](#_Toc446125919)

[Resources 12](#_Toc446125920)

[Product Links 12](#_Toc446125921)

[Reference Documents 12](#_Toc446125922)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Author** | **Date** | **Change Description** |
| 1.0 | SRL | 18-Mar-16 | Initial Release |

# Overview

## Document Scope

This document is intended to provide detailed information about the design, implementation, and usage of both the PMODIO physical hardware and the PMODIO peripheral, including its drivers. It is intended to provide the reader an understanding of how the various components of the project work together and how to use them in a design.

*This document assumes the reader is already familiar with the Xilinx™ Vivado Design Suite™, the Xilinx MicroBlaze™, and their usage of intellectual property (IP) blocks.*

## Project Description

The Pmod Input/Output (PMODIO) project is intended to expand the capabilities of the Digilent™ Nexys 4 DDR (NX4) board by provided a rotary encoder and graphical liquid-crystal display (LCD) for use with its expansion headers. The PMODIO device is attached to the NX4 via the JA and JB expansion headers. Along with the physical PMODIO device, a peripheral intellectual property (IP) block is provided for instantiation in a Xilinx™ MicroBlaze™ (MB) design in Xilinx’s field programmable gate array (FPGA) development software, the Vivado Design Suite™. Additionally, drivers are provided to interface with the PMODIO IP block to control the device in a Xilinx SDK project.

## Conventions Used in this Document

|  |  |
| --- | --- |
| **Term** | **Refers to…** |
| Physical Hardware | Physical PMODIO PCB and attached components |
| HDL Hardware | Verilog code in the IP block |
| Peripheral | PMODIO IP block for Xilinx Vivado and accompanying drivers |

Table 1: Document Conventions

## Notes

* This project targets the Nexys 4 DDR, but may also work on the Nexys 4 if pin mappings are updated

# Physical Hardware Description

The PMODIO physical hardware consists primarily of a Newhaven Display Intl™ NHD-12864MZ-FSW-GBW-L 128x64 graphical LCD and a Bourns™ Inc PEC11R-4215K-S0012 quadrature rotary encoder. Various other components are also present to support voltage level shifting for the LCD, as it operates at 5v, and the NX4 operates at 3.3v. Physical hardware is connected as follows:



Figure 1: Physical Block Diagram

## LCD Detail

The LCD is a 128x64 graphical LCD with backlight. It utilizes a parallel data/control bus connected over a 20-pin header. Pin description:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pin #** | **Symbol** | **Connection** | **Header Pin** | **Description** |
| 1 | VDD | 5.0v Power Supply | N/A | Power supply for logic |
| 2 | VSS | Ground | N/A | Ground |
| 3 | V0 | -3.3v Supply | N/A | Power supply for contrast |
| 4-11 | DB0-DB7 | FPGA GPIO | JB10, JB4, JB9, JB3, JB8, JB2, JB7, JB1 | 8-bit data bus |
| 12 | CS2B | FPGA GPIO | JA4 | Chip select, chip 2 |
| 13 | CS1B | FPGA GPIO | JA3 | Chip select, chip 1 |
| 14 | /RST | FPGA GPIO | JA2 | Active-low reset |
| 15 | R/W | Ground | N/A | Read/write select (0=write, 1=read) |
| 16 | RS | FPGA GPIO | JA7 | Register select (0=instruction, 1=data) |
| 17 | E | FPGA GPIO | JA1 | Falling edge triggered operation enable |
| 18 | VSS | Ground | N/A | Ground |
| 19 | LED+ | 5.0v Power Supply | N/A | Power supply for LED backlight |
| 20 | LED- | Ground | N/A | Ground for LED backlight |

Table 2: LCD Pin Descriptions

### Pin Notes

#### Chip Selects

The LCD datasheet says that CS1B and CS2B are active-low, but they do not seem to behave this way. The PMDIO drivers treat them as if they are active-high.

#### R/W Pin

The R/W pin is grounded because between the rotary encoder and LCD, there were seventeen total signals that needed to fit in two NX4 expansion headers, but each header has eight data pins, for a total of only sixteen pins available. It was decided that read capability from the LCD was not necessary, so the R/W pin was grounded, permanently putting the LCD in write mode.

#### Data/Control Pins

All “FPGA GPIO” connected pins are actually connected to the output of the 8-bit transceivers shown in Figure 1 to shift the signals from 3.3v to the LCD’s operating voltage of 5.0v. (This does not apply to the FPGA GPIO pins for the rotary encoder below.)

## Rotary Encoder Detail

The rotary encoder is a contact-switched, quadrature rotary encoder with a pushbutton. Pin description:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Name** | **Connection** | **Header Pin** | **Description** |
| SW1 | 3.3v Power Supply | N/A | Pushbutton signal w/ pull-down |
| SW2 | FPGA GPIO | JA8 | Pushbutton supply |
| A | FPGA GPIO | JA10 | A output, w/ pull-up |
| B | FPGA GPIO | JA9 | B output, w/ pull-up |
| C | Ground | N/A | Common ground for A and B outputs |

Table 3: Rotary Encoder Pin Description

# PMODIO Peripheral Description

The PMODIO peripheral is an AXI4\_Lite compliant IP block that communicates with the MicroBlaze via memory-mapped registers, and uses external ports to connect to the LCD and rotary encoder:



Figure 2: PMODIO Interfaces

## Port Descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal Name** | **Connection** | **I/O** | **Initial State** | **Description** |
| pmodio\_aclk | Clock | I | N/A | AXI Clock |
| pmodio\_aresetn | Reset | I | N/A | AXI Reset, active-low |
| pmodio\_\* | AXI4\_Lite Bus | N/A | N/A | AXI4\_Lite Slave Interface signals - see Appendix A of the AXI Reference Guide (UG761) for AXI4-Lite signals |
| ROT\_ENC\_A, ROT\_ENC\_B | Rotary Encoder: A, Rotary Encoder: B | I | N/A | A and B signals from the rotary encoder |
| ROT\_ENC\_BTN | Rotary Encoder: Button | I | N/A | Rotary encoder pushbutton |
| LCD\_DATA[7:0] | LCD: DB | O | 0x00 | LCD data bus |
| LCD\_EN\_OP | LCD: EN | O | 0x0 | LCD enable operation |
| LCD\_RESET\_N | LCD: /RST | O | 0x0 | LCD active-low reset |
| LCD\_REG\_SEL | LCD: RS | O | 0x0 | LCD register select |
| LCD\_CS\_1 | LCD: CS1B | O | 0x0 | LCD chip select 1 |
| LCD\_CS\_2 | LCD: CS2B | O | 0x0 | LCD chip select 2 |

Table 4: PMODIO Port Descriptions

## PMODIO Peripheral Architecture



Figure 3: PMODIO Module Block Diagram

## Register Map

### Rotary Encoder Current Count Register (ROT\_ENC\_CNT\_REG)

**Offset: 0x00**

Contains the current count output from the rotary encoder counter module. The value stored here is signed.

*This register is read-only.*

|  |
| --- |
| 31 … 0 |
| Current Signed Count |

### Rotary Encoder Step Register (ROT\_ENC\_STEP\_REG)

**Offset: 0x04**

Contains the value by which a single step of the rotary encoder will increment or decrement the count. The value stored here is signed. Due to the way the rotary encoder position is decoded, two clicks on the rotary encoder is one step. If the value here is positive, clockwise rotations will increment the count and counter-clockwise rotations will decrement the count. This behavior will be reversed if the value stored here is negative.

|  |
| --- |
| 31 … 0 |
| Step Size |

### Rotary Encoder Status Register (ROT\_ENC\_STS\_REG)

**Offset: 0x08**

Contains the rotary encoder status, which is currently only the pushbutton state.

Pushbutton State:

0 = Button not pushed

1 = Button pushed

*ROT\_ENC\_STS\_REG[0] is read-only.*

|  |  |
| --- | --- |
| 31 … 0 | 0 |
| Reserved | Button Status |

### Rotary Encoder Maximum Count Register (ROT\_ENC\_MAX\_CNT\_REG)

**Offset: 0x0C**

Contains the maximum value the rotary encoder is allowed to count to, inclusive. The value stored here is signed.

*Warning: setting this register to a value lower than ROT\_ENC\_MIN\_CNT\_REG results in undefined behavior.*

|  |
| --- |
| 31 … 0 |
| Maximum Allowed Count |

### Rotary Encoder Minimum Count Register (ROT\_ENC\_MIN\_CNT\_REG)

**Offset: 0x10**

Contains the minimum value the rotary encoder is allowed to count to, inclusive. The value stored here is signed.

*Warning: setting this register to a value higher than ROT\_ENC\_MAX\_CNT\_REG results in undefined behavior.*

|  |
| --- |
| 31 … 0 |
| Minimum Allowed Count |

### Rotary Encoder Reserved Register 0 (ROT\_RSVD0\_REG)

**Offset: 0x14**

Reserved. Used for write/read test during self-test by driver.

|  |
| --- |
| 31 … 0 |
| Reserved |

### LCD Data Register (LCD\_DATA\_REG)

**Offset: 0x18**

Wired directly to the LCD data bus ports. Register bit numbers correspond to data bus bit numbers. (Ex: LCD\_DATA\_REG[3] is connected to DB[3].)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 … 08 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | LCD Data Bus[7:0] | | | | | | | |

### LCD Control Register (LCD\_CNTL\_REG)

**Offset: 0x1C**

LCD\_CNTL\_REG[0:5] are wired directly to the LCD control pins. This register is used to assert and deassert pins on the LCD controller. Masks for accessing these pins are defined in PMODIO\_l.h.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 … 05 | 4 | 3 | 2 | 1 | 0 |
| Reserved | CS2B | CS1B | RS | EN | /RST |

### LCD Reserved Register 0 (LCD\_RSVD0\_REG)

**Offset: 0x20**

Reserved. Used for write/read test during self-test by driver.

|  |
| --- |
| 31 … 0 |
| Reserved |

### General Purpose Reserved Registers (GP\_RSVD0\_REG, GP\_RSVD1\_REG, GP\_RSVD2\_REG)

**Offset: 0x24, 0x28, 0x2C**

Reserved. Used for write/read test during self-test by driver.

|  |
| --- |
| 31 … 0 |
| Reserved |

## Driver Software

The driver source code is contained in the following files:

|  |  |
| --- | --- |
| **File Name** | **Description** |
| PMODIO\_l.c | Low-level driver code, only intended to be called by the high-level drivers |
| PMODIO\_l.h | Low-level driver code header file, only intended to be included in other driver source files |
| PMODIO.c | High-level driver API function implementations |
| PMODIO.h | High-level driver API header file, intended to be included in MicroBlaze applications that use the PMODIO peripheral |

Table 5: PMODIO Driver Files

### Rotary Encoder

The rotary encoder driver code is very simple, because all of the counting behavior is implemented in hardware. The driver is only responsible for reading count and button status from the registers, and setting count parameters by writing to the registers.

### LCD

The LCD driver is more complex, because it must resolve X-Y coordinates into column-page addresses in the regions that the LCD is divided into and handle writes to any location:



Figure 4: LCD Display Regions

At the driver API level, LCD write positions handled as simple X-Y coordinates for ease of use. When writing to the LCD, all writes must be page-aligned. Any non-page-aligned write actually causes two writes to the LCD to draw on both pages. Since writes are done a full byte at a time (bytes are written vertically, where DB[0] is written at the lowest Y coordinate), this introduces the possibility of overwriting previously written pixels. For example, a write at position (0, 4) means that DB[3:0] are written to column 0, page 0, while DB[7:4] are written to column 0, page 1. Without performing a read-modify-write (RMW) cycle, the currently displayed pixels at column 0, page 0[0:3] and column 0, page 1[4:7] will be overwritten. However, the RW pin on the LCD is tied low on the PCB, disabling reads.

The driver handles this case by maintaining a 1KB static array (128b x 64b = 1KB), where each bit corresponds to a pixel on the LCD. Every write to the LCD is also written to this array. When a write occurs that would clobber previously written bits, the bits that need to be retained are retrieved from the array and are combined with the new data with bit shifts and masks before being written both to the LCD and back to the array.

Note that the driver’s write data function that is available through its API has this signature:

PMODIO\_LCD\_write(PMODIO \*inst\_p, u8 data, u8 mask)

It accepts a mask argument in addition to the data bits to write. When a write to the LCD occurs, only the bits set in the mask argument will be written to the LCD. This allows per-pixel control over writes, preventing previously written data from being clobbered. If the mask is not 0xFF, even page-aligned writes will trigger a RMW cycle.

# Known Issues

1. Noisy rotary encoder

<https://github.com/SLawson/PMODIO/issues/1>

1. Rotary encoder doesn’t obey non-negative minimum counts

<https://github.com/SLawson/PMODIO/issues/2>

1. Incorrect xparamsters.h base addresses

<https://github.com/SLawson/PMODIO/issues/2>

See GitHub bug tracker for more details: <https://github.com/SLawson/PMODIO/issues>

# Resources

## Product Links

Digilent Nexys 4 DDR

<http://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/>

Newhaven Display NHD-12864MZ-FSW-GBW-L

<http://www.newhavendisplay.com/nhd12864mzfswgbwl-p-492.html>

Bourns Inc PEC11R-4215K-S0012

<http://www.bourns.com/docs/Product-Datasheets/PEC11R.pdf?sfvrsn=3>

## Reference Documents

AXI Reference Guide

<http://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf>

NT7108 Application Notes (LCD Controller)

<http://www.newhavendisplay.com/app_notes/NT7108.pdf>