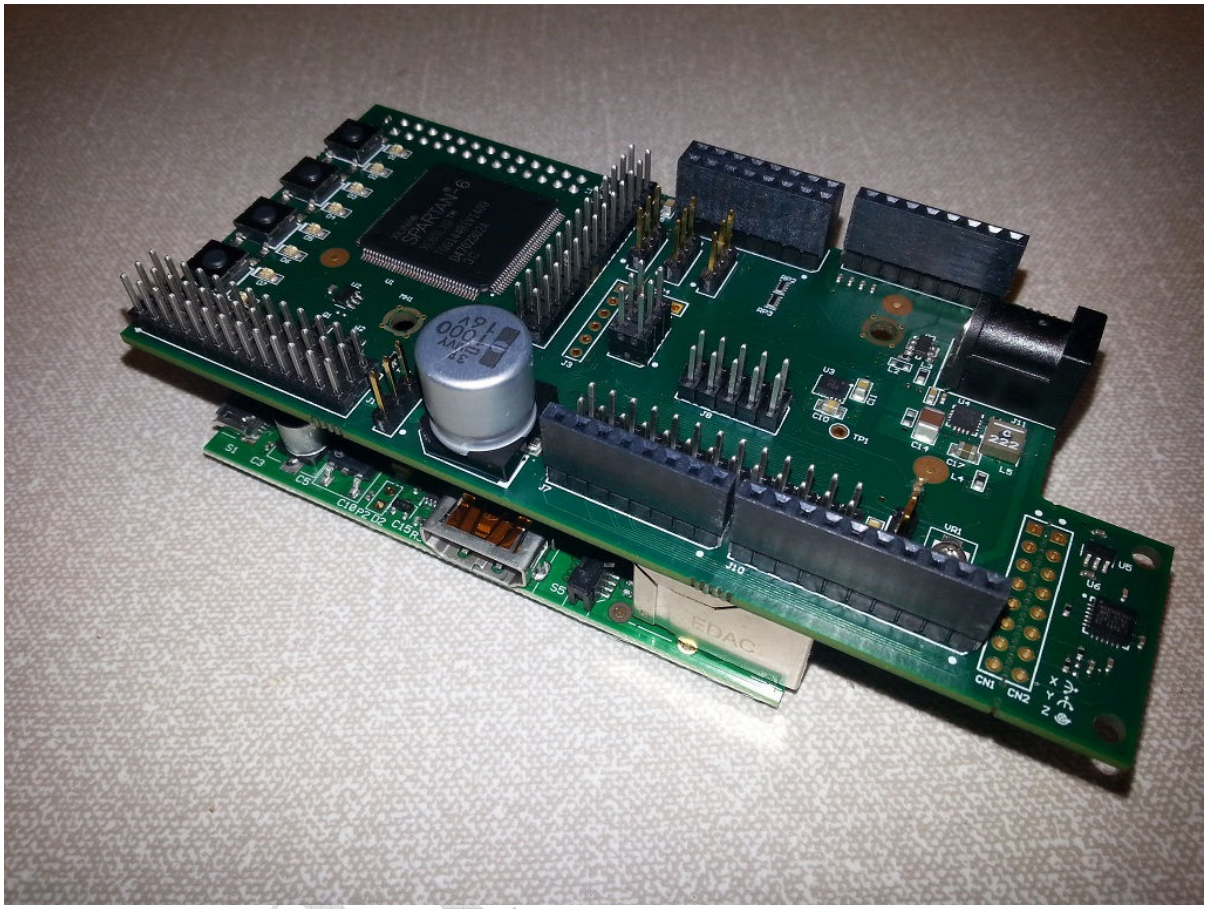
Application Note AN-0018

Using the Switches & LEDs on the PiXi$^{2.0}$



Summary

The PiXi$^{2.0}$ add-on board is designed to expand the general-purpose I/O capabilities of the Raspberry Pi and provide a low cost means of introducing the user to the world of digital electronics and FPGA technology as well as giving the 'Pi Enthusiast' a few more features to play with. The low product cost and feature-packed specification of the PiXi$^{2.0}$ makes it ideal for applications in computing, hobby-electronics, education, training and product development.

This application note explains how to use the general purpose switches and LEDs found on the PiXi$^{2.0}$. The SPI interface on the Raspberry Pi can be used to read the status of the switches or change the status of the LEDs and the PiXi-Tools software provided with the PiXi$^{2.0}$ makes it easy to quickly use the Raspberry Pi's SPI interface to communicate with the LEDs & switches and build these in to custom applications on the Raspberry Pi.

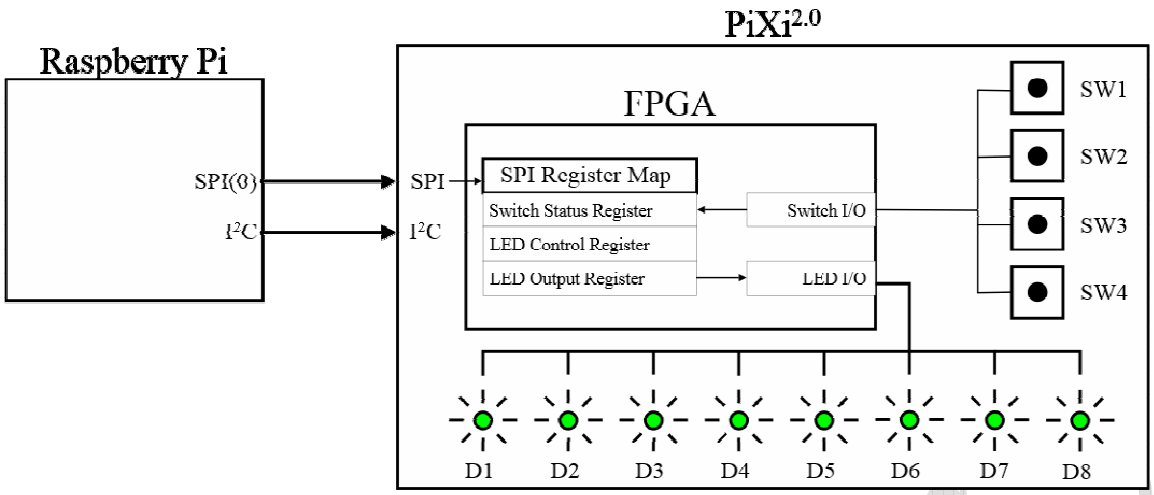


Figure 1 Accessing the Switches & LEDs on the PiXi

Install PiXi-Tools

PiXi-Tools provides a library of routines to help the user quickly access the functions on the PiXi. Application note AN-020 describes how to install and use PiXi-Tools on the Raspberry Pi. Once installed the user can use basic command-line functions to read the status of the switches or control the output of the LEDs through the SPI register map within the PiXi$^{2.0}$ FPGA. If the user is familiar with using the SPI interface on the Raspberry Pi then the switches & LEDs can also be accessed without using PiXi-Tools.

Using PiXi-Tools to Read the Switches and Program the LEDs

Once PiXi-Tools has been installed, the following generic SPI I/O commands can be used from the command-line:

Read switch register:

pio spi-get 0 0x3A

Response:
[n]
(where n is the value found in the switch status register)

Example: (assumes the user is pressing switch SW2)

pio spi-get 0 0x3A
34 (0x22)

This response shown in the switch status register provides two pieces of information. Firstly the first four bits (bits 3:0) provide a current status of each switch. Bit 0 indicates the status of switch SW1 where a value of '0' means the switch is inactive and a value of '1' indicates that the switch is pressed. In this example switch SW2 has been pressed so bit 1 is active. Secondly the second four bits (bits 7:4) provide an indication if the switch has

been pressed in the past – even if it's not currently pressed. As soon as each switch is pressed then the 'event bit' for that switch is set in the switch status register. In this example switch S2 was pressed so bit 5 of the switch status register is '1' to indicate this. With bits 1 and 5 set this equates to a binary value of "00100010" on the switch status register, or 34 in decimal.

The event bits of the switch status register are reset each time the switch status register is read so that they are automatically ready to detect the next event.

The event detection provided can help ease software development by reducing the need to quickly and repeatedly poll the switches to detect if one has been pressed. Also, because the switch event detection built into the FPGA is sensitive to the 'edge' of the signal from the switch, there is virtually no chance that even the quickest press of a switch would go un-detected, something that is virtually impossible to do with a simple polling of a switches current status.

Write to LED Control register:

pio spi-set 0 0x37 [n]

Response:
[n]
(where n is the mode of operation for the LEDs)

Write to LED Output register:

pio spi-set 0 0x36 [n]

Response:
[n]
(where n is the value written to the LED control output register)

Example: (assumes the user is want to illuminate LED D2)

pio spi-set 0 0x37 0
pio spi-set 0 0x36 0xC
Response:
50 (0x32)

This example writes a value of 0 to the LED control register to place the LEDs into a simple register-controlled output mode. The example then writes a value of 50 (decimal) or 0x32 (hexadecimal) to illuminate D2. Each LED requires two bits to be set to enable it, there are four states that each LED can have when controlled from the register. The LED can be off, flashing slowly, flashing quickly or it can be permanently on. For LED D2 bits 3:2 must be set to change the state of D2. Setting bits 3:2 to "11" will turn the LED on permanently, "10" will flash the LED quickly, "01" will flash the LED slowly and "00" will turn the LED off.

Having the LED flash function built into the FPGA can help simplify software by taking away the need to repeatedly write to the LED setting it first on, then off, then on, then off

again and so on. This function is built into the FPGA which provides a 1Hz or 2Hz flash rate.

It's also entirely possible to control the brightness of the LEDs by using a PWM driver built into the FPGA. At this time this function does not exist within the FPGA but this is something that could be built into the FPGA quite easily, however this process is outside the scope of this application note.

The LED Control register supports several modes for the LEDs. For example, mode 1 (write 1 to register 0x36) will force the LEDs to directly respond to the current switch status. Mode 12 will force the LEDs to implement a "walking '1'" pattern where one LED turn on followed by the adjacent LED turning on and previous LED turning off. This pattern repeats backwards and forwards over the line of LEDs.

Other modes of operation are available but these are generally used for testing the FPGA and are therefore beyond the scope of this application note. However it is a relatively simple task to create new modes of operation and implement these within the FPGA.

Register Map

The standard FPGA on the PiXi$^{2.0}$ implements register-mapped control & status registers which can be accessed through the SPI interface on the Raspberry Pi. The switches and LEDs can be accessed over SPI at the following addresses:

Address:	Read / Write	Register Function
0x36 [54]	R/W	LED Output register Bits(7:0) => LED status (D8, D7, D6, D5, D4, D3, D3, D1) '1' = ON, '0' = OFF
0x37 [55]	R/W	LED Control Register Bits(7:0) => LED output mode
0x3A [58]	R	Switch Status Register Bits(3:0) => Current Switch Status (SW4, SW3, SW2, SW1) Bits(7:4) => Switch Event Status (SW4, SW3, SW2, SW1)

Further Reading

If you want to learn more about the programming process, Xilinx application note XAPP502 describes the basics of using a micro computer such as the Raspberry Pi to program a Xilinx FPGA. The datasheet for the FPGA also contains more detailed information about the different programming modes.

PiXi-Tools is described in more detail in application note AN-020 "Installing PiXi-Tools on the Raspberry Pi".

The full register map for the PiXi$^{2.0}$ can be found in application note AN-025 "PiXi$^{2.0}$ SPI & I2C Register Map".

All of these documents and other documents are available for download from www.astro-designs.com.

Acknowledgements

"Raspberry Pi" is a trademark of the Raspberry Pi Foundation.

"Xilinx", "Spartan", "ISE", "Impact" & "WebPack" are all registered trademarks of Xilinx.