



Keywords: Pmod standard for FPGA, Pmod interface, Arduino standard for microcontrollers

APPLICATION NOTE 5902

A CHIMERA OF STANDARDS, OR THE CHALLENGES OF ADAPTING PROTOTYPING STANDARDS

By: Greg Steiert, Senior Member of Technical Staff , Maxim Integrated

Abstract: There are always connections for interfacing programmable devices such as FPGAs and microcontrollers so that software development can begin along with, or before, the hardware. This article explains the very loose pseudo standards for the “expansion interfaces” that have emerged, some more consistent than others, and will discuss the Pmod standard for FPGAs and the Arduino psuedo standard for microcontrollers. It explains how to use the MAX14661 mux to design a simple interface from one standard to another.

"The wonderful thing about standards is that there are so many of them to choose from." Grace Hopper¹

Introduction

Through the years there has been a proliferation of standards, and not many industries have created more than the electronics industry. Do you ever wonder why we have a standard for something that seems to be an odd number or gauge? The debate rages on over whether or not the standard that determined the diameter of the solid rocket boosters for the space shuttle actually originated from the backside of a horse.² Curious and interesting, yes, but, it is more productive to accept the fact that there will always be many standards so we must find ways to make them play well together. There is one specific area for both engineers and makers, where the incompatibility of standards is particularly troublesome: prototyping platforms. Fortunately, there are ways to mitigate these incompatibilities. Let's take a closer look.

Development Board Expansion Standards

For years component manufacturers have offered development systems to assist their customers in designing applications around their parts. For programmable devices such as FPGAs and microcontrollers, there are always connections for interfacing to the other components so software development can begin along with, or prior to, the hardware. With time, very loose pseudo standards for these "expansion interfaces" have emerged, some more consistent than others. FPGA vendors such as Xilinx[®] have driven some of these standards, such as FMC specification, to make it as easy as possible for customers to migrate to the newest platform. Xilinx has also used third-party standards like the Pmod[™] standard developed by Digilent[®] to connect peripheral modules to host controller boards, and there is a wide selection of peripherals for this interface. Microcontroller manufacturers have been somewhat slower to standardize, many utilizing their own proprietary interfaces. However, market forces like the maker movement and the popularity of the Arduino[®] platform are herding them towards pseudo standards too.

Pmod Is Well Suited for FPGAs

The Pmod interface is a great way to mix and match peripherals with an FPGA development board. It provides access to eight pins plus power and ground in a convenient, hand-solderable connector. The flexibility of FPGAs allows its eight signal pins to be used for almost anything. While this enhances its utility for FPGAs, it also makes the interface difficult to use with microcontrollers whose peripherals are assigned to specific pins. To address this problem, Digilent defined several different Pmod pinout types with various functions assigned to specific pins (**Figure 1**).

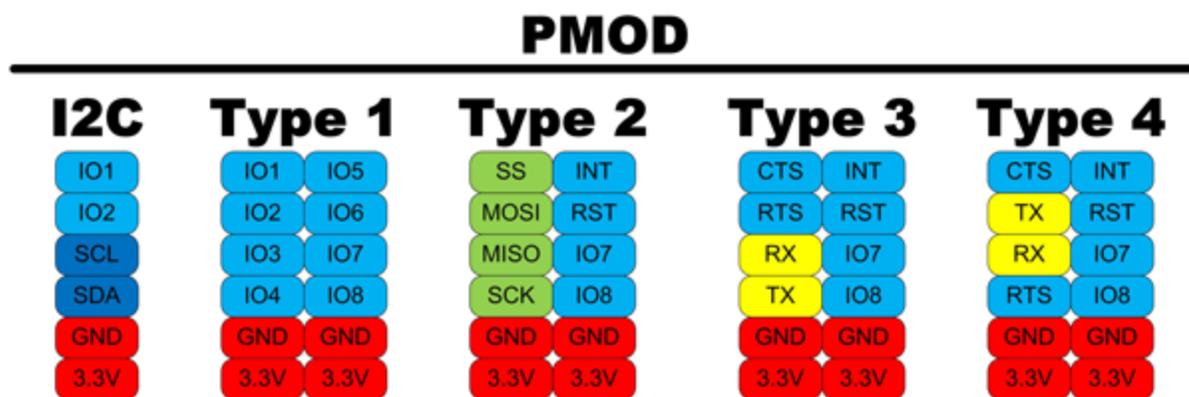


Figure 1. Pmod pinout types have various functions assigned to specific pins.

The type definitions make it easier to use the Pmod interface standard with a microcontroller board, but there are still challenges. The deprecated Type 3 UART interface is a good example of the difficulties involved in implementing a truly universal interface with the limited pin-muxing capabilities of many microcontrollers. Nonetheless, even with its limitations, the Pmod interface is a very useful expansion port for prototyping or educational endeavors.

The Arduino Pseudo Standard

The Arduino pseudo standard is a completely different beast; a different platform developed for a different audience for different reasons. The original Arduino board simply exposed the pins of a simple microcontroller and added enough supporting devices to make it easy to program, yet still affordable for hobbyists. Because of its simple nature, the original pinout was defined by the capabilities of the microcontroller.

As the platform evolved to support more-capable processors, this pseudo standard was fragmented with a myriad of pin-muxing combinations with, arguably, more exceptions than rules. Some issues, such as support for different I/O voltages and the inconsistency of I²C signals, were addressed in revision 3 of the UNO board. Yet, anyone pairing an Arduino board (or any of the Arduino derivatives) with a shield must carefully review the compatibility. There is no shortage of Arduino derivatives to choose from however

but that effort will be left as an exercise for the reader. The task of adaptively mapping the 22 Arduino signals into the different types defined in the Pmod specification (cf. Figure 1) is more manageable, yet still not trivial. Figure 1 shows the pinouts for five different Pmod types. Many of the signals are level sensitive and controlled through software by a GPIO pin. However, some of the signals employ time-sensitive protocols that are easier to work with by utilizing the peripherals inside the microcontroller. The challenge, therefore, is getting all the dedicated pins of the microcontroller mapped to the pin designated in the Pmod specification.

Proposed Solution: Use a Serial-Controlled Crosspoint Switch

One way to address this mapping problem is to put an array of configuration jumpers on the board. While straightforward, this is neither an elegant nor a user-friendly solution. The font size required to fit the instructions on an Arduino-sized board would be illegible. Many of the pins can be bidirectional, so simple logic gates are not an option for signal routing. Analog switches would work, but there are not enough extra pins to control them individually. The pin shortage could be addressed with I²C port expanders. Granted, that tactic would not be much more elegant than the jumpers, but it would be software configurable.

This endeavor would seem hopeless were it not for a serial-controlled 16:2 multiplexer, the [MAX14661](#). At first glance, one might think that four of these devices are required to support all the different Pmod types. Such an approach would hardly seem much better than the port expander solution. That would be true, but...the MAX14661 features a special capability. It allows for any combination of switches to be active concurrently so it can be used as an 8:8 crosspoint switch.

So far, so good. But there is, admittedly, a limitation to the 8:8 crosspoint configuration. It can only pass two independent signals simultaneously, one for each COM_ pin, but what good are two simultaneous signals? As stated before, most signals can be driven by GPIOs and there are only a few signals that require the internal peripheral controllers. If one GPIO is assigned to each pin, only the time-sensitive serial lines need to be routed through the mux. A UART and I²C are each 2-wire buses, so the only serial interface that does not fit through the mux is SPI. There is only one interface defined for SPI; therefore, the SPI signals are routed directly to the connector and used as the GPIO connections for those pins. When one of those pins is needed for a time-sensitive function, simply set the GPIO as a high-impedance input.

What about signal-voltage levels? The original Arduino design and many of its successors used 5V signal levels, but 5V signaling is not as common as it used to be. To address this, an IOREF pin was added to later Arduino boards to indicate the signal level for derivatives based on lower voltage microcontrollers. The Pmod connector does not have an IOREF pin, but the vast majority of its modules support 3.3V signaling. Fortunately, bidirectional level shifters like the [MAX3378E](#) address this issue. Simply connect the Arduino signals to one side of the level shifter powered by IOREF and then connect the other side to the Pmod connector powered by the dedicated 3.3V supply from the Arduino board. The MAX14661 mux tolerates and passes any signal from -5V to +5V with any supply between 1.8V and 5V, so it can be located on either side of the level shifter. Since there are more Arduino connections to the mux than Pmod connections, it makes sense to put the MAX14661 on the Arduino side of the level shifters. That way, only two of the 4-channel devices are needed at the Pmod connector (**Figure 3**).

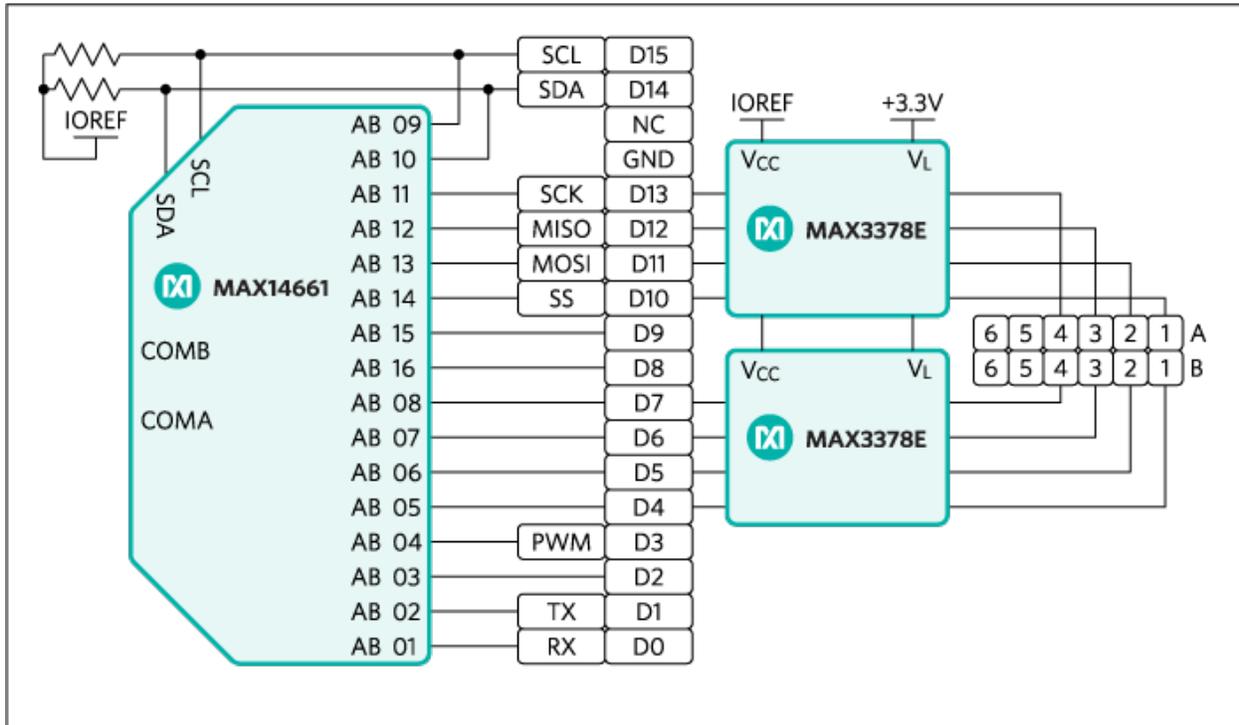


Figure 3. Logical diagram of the proposed solution.

Evaluating this Solution

The key to this solution is that each channel supports multiple simultaneous connections. The COM_ pins are not connected, but are used internal to the mux for routing. Eight of the MAX14661's 16 mux connections are dedicated to the Pmod connector, which is also connected to the four SPI-capable signals and four other GPIOs (Figure 3). The other eight mux connections are tied to the 2-wire serial buses and other time-critical signals like PWM or timer pins. To implement the I²C type, for example, simply configure the mux so the SDA pin and Pmod pin 4 are enabled on channel A, and the SCL pin and Pmod pin 3 are enabled on channel B. Channels A and B are arbitrary and can be swapped freely. In fact, one could even enable pins 3 and 4 on the second row of the Pmod connector to access a second 6-pin I²C Pmod. If the I²C devices have different addresses, both ports can be left active or dynamically activated to support two devices with the same address. Dynamically switching between two devices with the same address is simply not possible using jumpers. But with the MAX14661, it is possible to programmatically configure all the specified Pmod types and even some nonstandard types with a single active device that is only 4mm x 4mm.

Why would anyone design a nonstandard Pmod? One reason is the same pin-muxing problem found with microcontrollers. For example, the MAX14661 supports both I²C and SPI control, depending on the state of a configuration pin. The peripheral module for the MAX14661 is configured to match the SPI Pmod type, but the clock and data pins do not match the I²C pins specified by the Digilent Pmod specification. The MAX14661 on the adapter card can map the SDA and SCL signals to any Pmod pin to support the nonstandard MAX14661 peripheral module, I²C pin mapping. The MAX14661 also has two address pins to support four different I²C device addresses, which means systems can be designed to support multiple

devices on the same bus.

When you use the MAX14661 to multiplex I²C buses, the commands are sent in-band. The switching takes effect synchronously with the I²C bus. It can, admittedly, be challenging to multiplex I²C buses out of band. If care is not taken, you can disconnect while a slave is holding the SDA line low. The next time you select that branch of the bus, it might lock up. The MAX14661 will always transition the bus at the end of an I²C write command when the devices on the other side of the switch should be idle.

Conclusion

Upon inspection, there is no evidence to suggest that either the Pmod specification or Arduino pseudo standard originated from the backside of a horse. However, the legacy of issues and factors that led to their creation is encoded deeply within their DNA. While the differences between the two standards seem irreconcilable, the MAX14661 enables us to bring the mythical Arduino-Pmod chimera to life.

Learn more about [peripheral modules](#) and review the [MAXREFDES72# reference design](#) that implements the circuit described in this article.

References

1. See Simson Garfinkel, Daniel Weise, and Steven Strassmann, eds., **The UNIX-HATERS Handbook**, IDG Books Worldwide, Inc. An International Data Group Company, 1994, page 49, <http://web.mit.edu/~simsong/www/ugh.pdf>.
2. See (www.astrodigital.org/space/stshorse.html).

Arduino is a registered trademark of Arduino, LLC.

Digilent is a registered trademark and Pmod is a trademark of Digilent Inc.

Xilinx is a registered trademark and registered service mark of Xilinx, Inc.

Related Parts

MAX14661	Beyond-the-Rails 16:2 Multiplexer	Free Samples
MAX3378E	±15kV ESD-Protected, 1µA, 16Mbps, Dual/Quad Low-Voltage Level Translators in UCSP	Free Samples

More Information

For Technical Support: <http://www.maximintegrated.com/en/support>

For Samples: <http://www.maximintegrated.com/en/samples>

Other Questions and Comments: <http://www.maximintegrated.com/en/contact>

Application Note 5902: <http://www.maximintegrated.com/en/an5902>

APPLICATION NOTE 5902, AN5902, AN 5902, APP5902, Appnote5902, Appnote 5902

© 2014 Maxim Integrated Products, Inc.

The content on this webpage is protected by copyright laws of the United States and of foreign countries.

For requests to copy this content, [contact us](#).

Additional Legal Notices: <http://www.maximintegrated.com/en/legal>