Hardware Design Description

Plantation Productions Open Source/Open Hardware Data Acquisition System

**Table of Contents**

[1 Introduction 3](#_Toc6090883)

[1.1 Purpose 3](#_Toc6090884)

[1.2 Scope 3](#_Toc6090885)

[1.3 Contents of the Document 3](#_Toc6090886)

[1.4 Document Conventions 3](#_Toc6090887)

[2 Hardware Architecture 5](#_Toc6090888)

[2.1 Design Considerations 6](#_Toc6090889)

[2.1.1 Assumptions and Dependencies 6](#_Toc6090890)

[2.1.2 Related Software or Hardware 6](#_Toc6090891)

[2.1.3 End-User Characteristics 6](#_Toc6090892)

[2.1.4 System Interfaces 6](#_Toc6090893)

[2.1.5 Fail-Safe 7](#_Toc6090894)

[2.1.6 Hardware Basis 7](#_Toc6090895)

[2.1.7 User Interfaces 7](#_Toc6090896)

[2.1.8 Hardware Interfaces 7](#_Toc6090897)

[2.1.9 Software Interfaces 8](#_Toc6090898)

[2.1.10 General Constraints 8](#_Toc6090899)

[2.1.11 Goals and Guidelines 9](#_Toc6090900)

[2.1.12 Developmental Methods 10](#_Toc6090901)

[3 Hardware Design Description 11](#_Toc6090902)

[3.1 Design Stakeholders 11](#_Toc6090903)

[3.2 Design Concerns 11](#_Toc6090904)

[3.2.1 Dow Site Management 11](#_Toc6090905)

[3.2.2 DTRR Operators 11](#_Toc6090906)

[3.2.3 NRC 11](#_Toc6090907)

[3.2.4 DAQ Project Management 11](#_Toc6090908)

[3.2.5 DAQ Hardware Engineering/Development 11](#_Toc6090909)

[3.2.6 DAQ Hardware Engineering/Testing 12](#_Toc6090910)

[4 DAQ Detailed Design 13](#_Toc6090911)

[4.1 DAQ\_IF 13](#_Toc6090912)

[4.1.1 CPU Interface Board 13](#_Toc6090913)

[4.1.2 Logic Level Conversion 14](#_Toc6090914)

[4.1.3 Logic Signal Buffering 16](#_Toc6090915)

[4.1.4 SPI Chip Select 17](#_Toc6090916)

[4.1.5 I2C Bus Expansion 17](#_Toc6090917)

[4.1.6 Application-Defined I/O Pin 19](#_Toc6090918)

[4.1.7 DAQ\_IF Watchdog Timer 19](#_Toc6090919)

[4.1.8 DAQ\_IF Power LED 21](#_Toc6090920)

[4.1.9 DAQ\_IF RS-232 22](#_Toc6090921)

[4.1.10 PPDO Connection 23](#_Toc6090922)

[4.1.11 PPDIO96 Connection 24](#_Toc6090923)

[4.1.12 DAQ\_IF Power Supply 24](#_Toc6090924)

Plantation Productions Open Source/Open Hardware Data Acquisition System

**HARDWARE DESIGN DESCRIPTION**

# Introduction

This document is the Hardware Design Description (HDD) for the hardware contained in the Digital Data Acquisition System (DAQ) created by Plantation Productions, Inc.

The DAQ system provides digital and analog I/O capabilities for embedded systems; specifically, it was designed to provide data acquisition for TRIGA™ Research Reactors (TRIGA is a registered trademark of General Atomics).

## Purpose

The purpose of this document is to describe the hardware design for the DAQ system. The intended audiences for this document are the engineering, product assurance and management personnel involved in the hardware development.

## Scope

The scope of the software, which is based on the DAQ Hardware Requirements Specification (DAQ HRS) per the requirements that have been allocated to the DAQ system hardware. From this document the hardware will be developed.

## Contents of the Document

The general description including product perspective, product functions, user characteristics and general constraints is included in Section 2.

The design descriptions are included in Section 3.

The HDD to HRS traceability (reverse traceability) is documented in a separate document (Excel spreadsheet).

## Document Conventions

All tags shall take the form:

<whitespace> [DAQ\_HDD\_xxx]

where "xxx" is a three-digit number reserved for HDD usage.

For HDD tags, should the need arise to insert a new HDD tag between two other values (e.g., add a requirement between 030 and 031) then a decimal fractional number shall be appended to the HDD tag number (e.g., 031.5). Any number of decimal point suffixes can be added, if needed (e.g., 030.5.2).

Examples:

From the HDD:

<whitespace> [DAQ\_HDD\_053]

Note: because an external script may be used to extract requirements from this document, it is very important that actual requirements in this document begin on a new line (with nothing but white space preceding the requirement) and that the requirement take exactly the form shown above. The regular expression used by the requirement extraction script is the following:

<whitespace>\* '[DAQ\_HDD\_' {0-9}{0-9}{0-9}('.' {0-9}+)\* ']' <whitespace>\* ':' <whitespace>\* .\* '\n'

Where ".\*" represents an arbitrary sequence of characters not including a new line and "<whitespace>\*" represents zero or more tab or space characters. The actual string "<whitespace>" appears in front of the examples in this section so that they will *not* be captured by this script; the string "<whitespace>" should not appear in front of actual requirements (though actual whitespace is certainly permissible).

# Hardware Architecture

The DAQ digital data acquisition system (Hereafter, "DAQ") consists of several main components:

1. DAQ\_IF (DAQ interface board). This circuit board interfaces to a single-board computer such as a Netburner MOD54415 Evaluation Board, a Raspberry Pi 3 Model B, or a Teensy 3.2.
2. PPDIO96 (96-input digital I/O board). This board connects to the PPDIO96 bus connector on the DAQ\_IF board and provides 96 digital I/O pins. Up to six PPDIO96 boards can be connected together in a daisy chain off one DAQ\_IF board.
3. PPOPTO-12 (12-channel digital input opto-isolation). This board provides 12 channels of digital opto-isolation. This board connects to one of the 12-input bank connectors on the PPDIO96. Up to eight PPOPTO-12 boards can be connected to a single PPDIO96 board providing 96 opto-isolated inputs.
4. PPBreakout (12-channel breakout board for PPDIO96). This board connects to one of the 12-input bank connectors on the PPDIO96. It provides 12 sets of two-terminal screw terminals for the 12 I/O pins on that PPDIO96 bank. The PPBreakout board is useful when feeding inputs to the PPDIO96 that do not require isolation or for connecting digital outputs from the PPDIO96 to the rest of the system.
5. PPRELAY-12 (12-channel mechanical relay). this board connects to the PPDO bus connector on the DAQ\_IF board and provides 12 mechanical relay digital outputs (with NC/NO terminals). In addition, for digital I/O pins capable of sinking up to 150 mA are also available. In theory, an unlimited number of these boards (and PPSSR-16 boards) can be daisy-chained off the PPDO connector. In practice, fan-out limits the system to 10 or fewer boards.
6. PPSSR-16 (16-channel Solid-State Relay). This board connects to the PPDO bus connector on the DAQ\_IF board (or daisy-chains with other PPSSR-16 and PPRELAY-12 boards). It provides 16 solid-state relay controlled outputs. In theory, an unlimited number of these boards (and PPRELAY-12 boards) can be daisy-chained off the PPDO connector. In practice, fan-out limits the system to 10 or fewer boards.
7. PPAIO-16/4 (16 analog inputs, 4 analog outputs). This board connects to one of the four 6-pin I2C connectors on the DAQ\_IF board and provides 16 single-ended 16-bit analog inputs (or 8 double-ended inputs) and four 12-bit analog outputs. The analog inputs support up to a 0-5V range (with programmable gain amplifiers allowing the full-range input of smaller signals). The analog outputs include amplifier circuitry to support outputs in the range -10V to +10V (easily programmable as -5V to +5V, if desired).
8. PPAC4 (4-channel analog conditioning). This board provides analog isolation and conditioning. The four inputs to this board are single-ended analog signals in the range -10 to +10V. The four outputs from this board are double-ended analog signals (0-4.096V) that provide a full 16-bit range to the PPAIO-16/4 analog inputs. This board also contains isolation amplifiers that isolate the double-ended output from the single-ended input signal.
9. PPAC420 (8-channel 4-20mA analog conditioning). This board accepts 4-20 mA current loop inputs and produces a single-ended -1.25 to +5V output. Typically, this outputs would be fed as inputs to the PPAC4 (to provide isolation and full-range ADC conversion), though if isolation is not required the inputs could be fed directly into the single-ended inputs on the PPAIO-16/4 board.

## Design Considerations

### Assumptions and Dependencies

In the most common use case, the DAQ system will be connected to some host computer via Ethernet, USB, or RS-232 Serial. While it is possible for the DAQ system to operate at a stand-alone data acquisition system, such usage will be rare.

### Related Software or Hardware

When running with a Netburner MOD54415 module, the system will be running the µC/OS real-time operating system. When running on a Raspberry Pi, the system will likely be running a variant of the Linux operating system. When running with a Teensy 3.2 module, the software will (likely) be using the Arduino-style library modules provided for the Teensy 3.2.

Because the DAQ system operates using I2C and SPI interfaces, it is quite possible to hook up other I2C and SPI devices. This document will not consider such related hardware.

### End-User Characteristics

There are three types of "end-users" associated with the DAQ system: system end users, technicians, and system designers.

System end users (those individual using the final system) may not even be aware of the DAQ system – they are only interested in using the system as a whole and the DAQ system might be a small or hidden part of that whole system.

Technicians are those individuals responsible for maintaining and calibrating the system. Their responsibilities will likely include maintaining and calibrating the components of the DAQ system. As such, they are likely to be concerned with the electronic design and operation of the DAQ system boards.

System designers are those indivduals who design a system around the DAQ system. Clearly, they will require the most knowledge and experience with the DAQ system hardware.

### System Interfaces

The DAQ system provides for up to three system interfaces (depending on the computer module controlling the DAQ\_IF board): Ethernet (Netburner and Raspberry Pi), USB (Teensy 3.2), and RS-232 (Netburner, Raspberry Pi, and Teensy 3.2).

### Fail-Safe

The DAQ system operates in a fail-safe fashion. After a power on restart the PPDIO96 comes up with all the pins programmed as inputs; the PPRELAY-12 and PPSSR-16 digital output boards come up with all the relays turned off (NC in the "C" state, NO in the "O" state, and solid-state relays in the high-impedance state).

The DAQ-IF includes a watch-dog timer that trips a relay if it is not refreshed within 5-10 seconds. If the firmware on the CPU module hangs up or otherwise fails to refresh the watchdog timer within the specified amount of time, a relay will be actuated. The NC/NO outputs from this relay are available for use in the rest of the system (application-defined) to handle the software anamoly. In addition, this signal also appears on the PPDIO96 and PPDO busses. Activation of this signal will place the PPDIO96, PPRELAY-12, and PPSSR-16 boards in the fail-safe state.

Finally, a software programmable line (reset) appearing on the PPDIO96 and PPDO busses will also put the boards into their fail-safe state. Note that the reset line will also reset the watchdog timer if it has expired.

The DAQ\_IF board uses a separate digital output line from the CPU module to refresh the watchdog timer. The CPU module must pulse this line (low-to-high edge) at least once every timeout period to keep the watchdog timer from timing out. Note that the watchdog timer is a hardware (RC-based) circuit, it does not depend on software to produce a watchdog timeout.

When designing an end-system around the DAQ system, you should carefully consider the fail-safe design of the DAQ system and wire up any safety-critical systems in an appropriate fashion (e.g., SCRAM loops in a nuclear power reactor).

### Hardware Basis

The DAQ hardware is based the data acquisition needs commonly found in TRIGA™ Research Reactors. The design considered the maximum data acquisition requirements for the most complex reactor in existence and then tripled or quadrupled those requirements (in terms of input and output channels) and that became the basis for the design. Because of the generous requirements (up to 576 digital inputs [or outputs], up to 160 relay-controlled outputs, up to 128 analog inputs, and up to 32 analog outputs) the DAQ system should be more than capable for the vast majority of data acquisition projects someone could come up with.

### User Interfaces

This document assumes that the DAQ system will be controlled by some host PC. As such, most of the user interface design associated with the DAQ system will occur on the host PC. However, certain user-interface components are present on the DAQ hardware. In particular, all boards will contain a power LED indication power applied to the board (or not). Furthermore, most digital I/O devices provide LEDs for each bit of digital I/O indicating whether the digital I/O pin is active or inactive.

### Hardware Interfaces

The DAQ system provides several hardware interfaces to external (to the DAQ) components and systems. As noted earlier, the DAQ system provides Ethernet, USB, and RS-232 interfaces to host computer systems. On the hardware side there are also interfaces to digital inputs, interfaces to digital outputs, interfaces to analog inputs, interfaces to analog outputs, and interfaces to other modules using the I2C and SPI busses.

#### Ethernet Interfaces

The Netburner and Raspberry Pi 3 Model B single-board computers (SBCs) both provide on-board Ethernet interfaces. These SBCs both contain an operating system with a full TCP/IP stack allowing easy communication across the Ethernet using standard socket communications.

The Teensy 3.2 module does not have a built-in Ethernet interface. However, it is possible to purchase an SPI-based Ethernet interface and wire it to the SPI bus (PPDIO96 bus) on the DAQ\_IF interface board if Ethernet access is desired when using a Teensy 3.2 SBC. Note, however, that such connections are beyond the scope of this document.

#### USB Interfaces

The Teensy 3.2 SBC includes a small micro-USB connector. You may connect the DAQ\_IF board (with Teensy 3.2 installed) to a host computer using a USB-to-micro-USB cable. To the host computer, the Teensy 3.2 looks like a really fast COM port.

In theory, the Netburner can also be programmed to act as a USB device. This document will not consider that option.

The Raspberry Pi 3, while it supports USB, cannot be used as a USB device (it's a USB host). Newer Raspberry Pi Zero units can be programmed as a device (using USB On-the-go) but this document will not consider that option.

### Software Interfaces

There are software libraries available for the Teensy 3.2 and Raspberry Pi 3 Model B SBCs that allow you to easily program all the I/O chips in the DAQ system. However, the main software written to support the DAQ system runs on the Netburner MOD54415 Evaluation board plugged into the DAQ\_IF board. The software interface for this system is completely described in the SRS (Software Requirements Specification), SDD (Software Design Description), and DAQ Operations manual.

### General Constraints

* Hardware or software environment – The majority of the software is written in the C/C++ programming language. In addition, there are various "make" files and system scripts used to build or run the system. Some test software is written in C++ running on the Teensy 3.2 or Raspberry Pi, but the majority of the system code runs under µC/OS on the Netburner.
* The software is developed under Windows using the Netburner software development kit and software development tools.
* Availability or volatility of resources – Run within the confines of the Netburner MOD54415 hardware.
* Interface/Protocol Requirements – The basic intertask communication will be handled by Ethernet socket communications.
* Data repository and distribution requirements – In a preemptive multitasking operating system, shared data objects must be protected to insure all access is serialized. Typically, this is done with semaphores, mutexes, or critical sections. To provide the necessary data access protection for globals, the code will include a global semaphore to be used by most tasks. This method will insure serialized data access.
* Memory and other capacity limitations - There will be no memory constraints other than the total amount of system RAM installed MOD54415 module. As the software runs on µC/OS, the system limitations 64 MB RAM and 2 MB Flash/ROM.
* Performance Requirements – The system shall perform all necessary tasks well within the performance capabilities of the MOD54415. During normal operations, all normal I/O and user interface tasks much be completely in less than 25% of the CPU cycles available. Occasional exceptions, lasting no more than a few seconds (less than the DAQ\_IF watchdog timeout period), are acceptable as long as these events are infrequent.
* Network Communications – The system shall use TCP/IP Ethernet communications protocols to communicate between the DAQ system and other devices.
* Testing – Testing will follow the standard Plantation Productions, Inc., practices for testing systems, as defined by the HTP (Hardware test procedures) and STP (Software Test Procedures) documents.
* Environmental -- The software shall perform properly while the unit is operating within the environmental constraints for the hardware.

#### Operating System

The DAQ software will utilize the µC/OS pre-emptive priority-based multitasking operating system kernel for microprocessors. µC/OS was not developed by Plantation Productions, Inc., and may be considered Software of Unknown Provenance (SOUP). However, µC/OS is open-source and its implementation on the Netburner modules has a long history so there is a high confidence in the reliability of this software.

The host computers contain their own software. As the software for those devices is outside the scope of this document, it shall not be considered here.

#### Criticality of the Application

The hardware may be used as part of a research reactor. The hardware's design allows the support of safety critical systems with an appropriate verification and validation (V&V) process in place.

#### Personnel Safety and Security Considerations

The hardware must be designed to protect the safety of personnel who use and maintain it.

### Goals and Guidelines

**Basic hardware goals:**

The hardware is designed to operate as programmed under normal circumstances. In anamolous situations (power up, software hang) the hardware shall enter a special fail-safe mode in which all digital I/O ports that are programmable as digital inputs or outputs revert to inputs and all strict digital output ports are programmed in the "off" (or "open") state. To fully realize this in a system, the following guidelines apply:

1. The PPDIO96 board should be programmed as input only. Although the system will automatically reset the I/O expander chips to input pins on a reset or watchdog timeout condition, keep in mind that if the pins are actually used as outputs the pins will be floating at that point and won't be in a guaranteed (fail-safe) state immediately after the reset or timeout.
2. Before writing to a digital output board (PPRELAY-12 or PPSSR-16) the system software should first verify that a watchdog timeout has not occurred (which will force all the outputs to "off" or "open"). If a timeout has occurred, then the system should reinitialize all the digital outputs to some reasonable state.

### Developmental Methods

The developmental method used to design the DAQ system hardware was based loosely on the Prototype/Iterative Model. In this Model, being a process, the following phases were followed in order:

1. Requirements Specification
2. Hardware Design
3. Circuit board creation and assembly
4. Testing (validation)
5. Repeat steps 1-4 for each new feature or site port.

# Hardware Design Description

## Design Stakeholders

The initial design of the DAQ system hardware was to support the data acquisition requirements for the Dow TRIGA Research Reactor (DTRR) at Dow Chemical in Midland Michigan. For this project, the project stakeholders are the following:

* Dow Chemical (site management)
* Dow Chemical (site operators/users)
* NRC
* Plantation Productions, Inc., project management
* Plantation Productions, Inc., software engineering/development
* Plantation Productions, Inc., hardware engineering/development
* Faircloth Engineering (systems hardware engineering development)
* Plantation Productions, Inc., software quality assurance

Although this design was originally created for DTRR and Dow Chemical, the design is sufficiently generic to be adopted for a wide variety of embedded applications. As such, the stakeholders can easily be swapped for more stakeholders more pertinent to a given application.

## Design Concerns

### Dow Site Management

The DAQ design shall provide like-for-like functionality of the existing data acquisition system for DTRR.

### DTRR Operators

The console design shall provide a reasonable approximation of the legacy QNX console to reduce the learning curve. This is achieved by providing the like-for-like functionality of the existing console using a similar user interface. In particular, the DAQ system will provide a transparent replacement for obsolete ISA-bus analog and digital I/O boards.

### NRC

The DAQ shall follow software quality assurance requirements for research reactors as specified by ANSI 15.15-1978.

### DAQ Project Management

The design shall allow implementation, testing, and deployment of the console on schedule and under budget

### DAQ Hardware Engineering/Development

The hardware engineering team shall have the following design concerns:

* The design shall provide the functionality required by the SOW, SyRS, and HRS.
* The design shall be efficient in terms of space, power usage, and functionality.
* The design and implementation shall be reliable.
* The design and implementation shall be maintainable.

### DAQ Hardware Engineering/Testing

The hardware testing team shall have the following design concerns:

* The design shall produce an appropriate set of test cases.
* The design and hardware shall be such that the system is easy to test.

# DAQ Detailed Design

## DAQ\_IF

### CPU Interface Board

1. [DAQ\_HDD\_001]

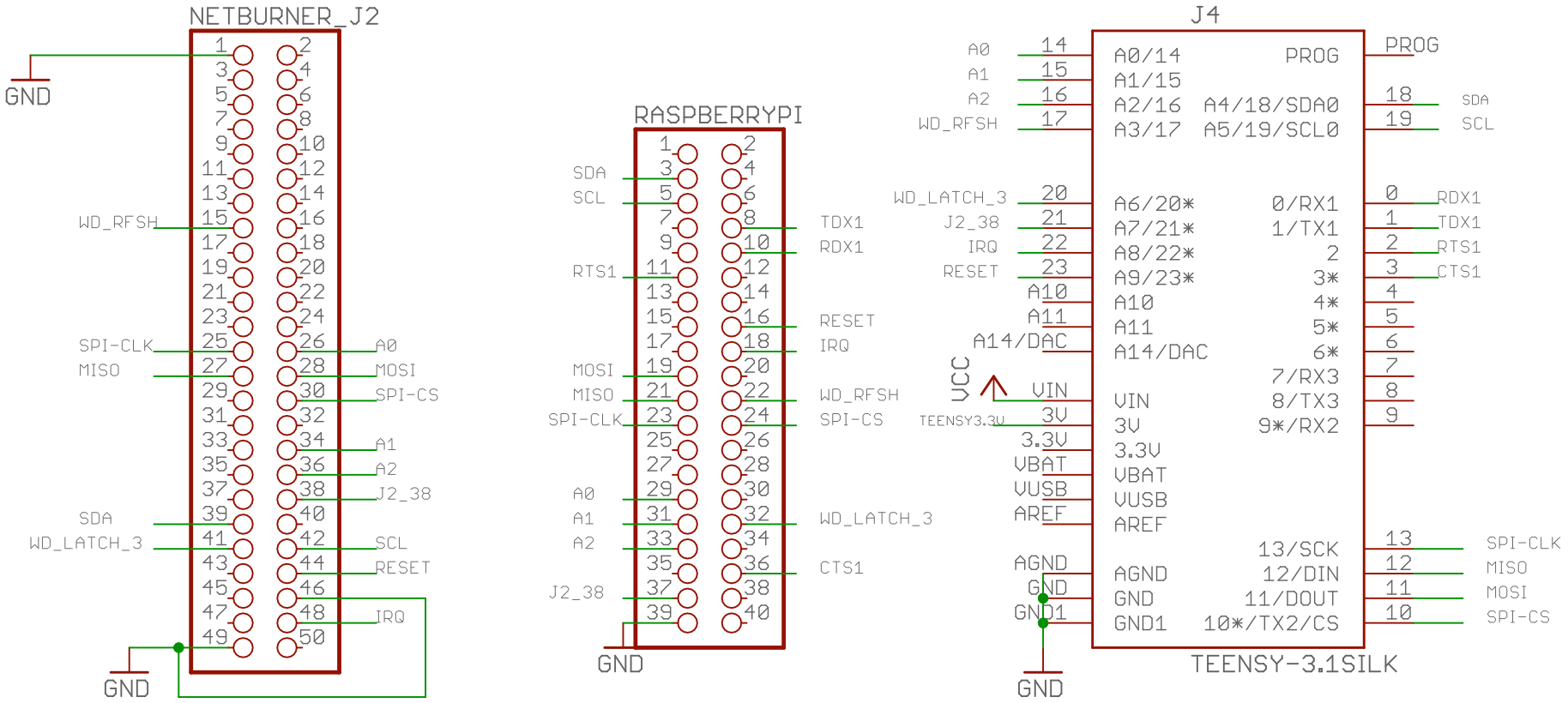
The original DAQ system design was created using a Netburner MOD54415 Evaluation Board as the controlling computer module. However, to make the design slightly more generic and open up the application domain a bit, the design should also include support for other CPU board modules. In the initial design, this should include at least the following modules:

* A Linux-based CPU module (Raspberry Pi 3 Model B was chosen)
* An Arduitno-based CPU module (Teensy 3.2 was chosen).

The DAQ\_IF board was designed to interface to a CPU module using a single I2C bus, a single SPI bus (and single SPI chip select), and handful of digital I/O pins, and an optional (TTL-level) RS-232 port.

The Netburner MOD54415 Evaluation Board largely defined the signals used by the DAQ\_IF (other than RS-232, the MOD54415 Eval Board provides its own RS-232 ports). Pretty much every free available digital I/O line that was available on the MOD54415 Eval board was put to use on the DAQ\_IF board. The Raspberry Pi 3's (on its 40-pin GPIO connector) and the Teensy 3.2's I/O pins were mapped to provide this same functionality. The Raspberry Pi and Teensy modules also had TTL-level serial port signals that the DAQ-IF uses for serial communication (note that the DAQ\_IF board provides level conversion to RS-232 signals).

In theory, there is no reason one couldn't connect a different computer system to the DAQ\_IF board by wiring appropriate signals to the Netburner, Raspberry Pi, or Teensy 3.2 connections on the DAQ\_IF board. The only thing to note is that the Netburner, Pi, and Teensy[[1]](#footnote-1) modules are all 3.3V logic levels so any system connecting to those pins must also support 3.3V logic levels.



Pins in use by the Single-Board Computer interface. All signals are 3.3V logic signals.

|  |  |  |  |
| --- | --- | --- | --- |
| Netburner | Raspberry Pi | Teensy 3.2 | Function |
| 15 | 22 | 17 | Output: Watchdog refresh |
| 25 | 23 | 13 | Output: SPI Clock |
| 26 | 29 | 14 | Output: A0 (SPI chip select L.O. bit) |
| 27 | 21 | 12 | Input: MISO (SPI data in) |
| 28 | 19 | 11 | Output: MOSI (SPI data out) |
| 30 | 24 | 10 | Output: SPI CS (SPI master chip select) |
| 34 | 31 | 15 | Output: A1 (SPI chip select middle bit) |
| 36 | 33 | 16 | Output: A2 (SPI chip select H.O. bit) |
| 38 | 37 | 21 | I/O: Reserved for future expansion |
| 39 | 3 | 18 | I/O: I2C data line |
| 41 | 32 | 20 | Input: WD timeout latch input |
| 42 | 5 | 19 | I/O: I2C clock line |
| 44 | 16 | 23 | Output: Reset (DAQ hardware) |
| 48 | 18 | 22 | Input: IRQ (Interrupt Request) |
| xx | 8 | 1 | Output: TDX1 (RS-232 transmit) |
| xx | 10 | 0 | Input: RDX1 (RS-232 receive) |
| xx | 11 | 2 | Input; RTS1 (RS-232 request to send) |
| xx | 36 | 3 | Output: CTS1 (RS-232 clear to send) |
|  |  |  |  |

### Logic Level Conversion

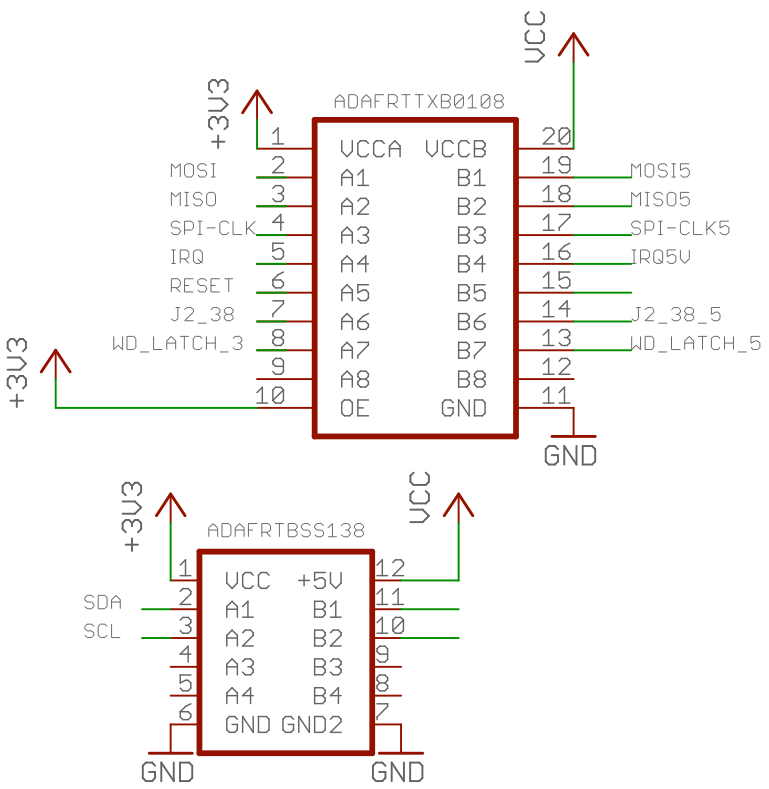
1. [DAQ\_HDD\_002]

All of the DAQ system modules except the CPU interface operate using TTL/5V logic level signals. Therefore, all the 3.3V signals on the CPU lines must be translated to 5V signals for the rest of the system.

For the non-I2C signals, an Adafruit TXB0108 breakout board provides appropriate level conversion between 3.3V and 5V logic signals. The TXB0108 IC is a surface-mount device (SMD). One design rule for the DAQ system is to use through-hole parts and avoid SMDs and system maintenance and trouble-shooting is much more difficult when using SMDs. Breakout boards provide a solution when only SMDs are available for a given function. Breakout boards are small circuit boards containing SMD circuitry that bring out the pins on the surface mount devices to 0.1" header pins. The DAQ system treats these breakout boards like any other IC. In the case of the TXB0108 IC, Adafruit provides a small 20-pin breakout board that contains the TXB0108, decoupling capacitors, and a resistor all in the same package. It is very easy to connect test leads to the pins and if the device is damaged, you can replace the whole breakout board and not worry about having to desolder and solder SMD parts on the DAQ\_IF board.

The TXB0108 board provides an array of eight bi-directional level shifter circuits. The DAQ\_IF board uses seven of these circuits (leaving one free for future use). These signals are the following:

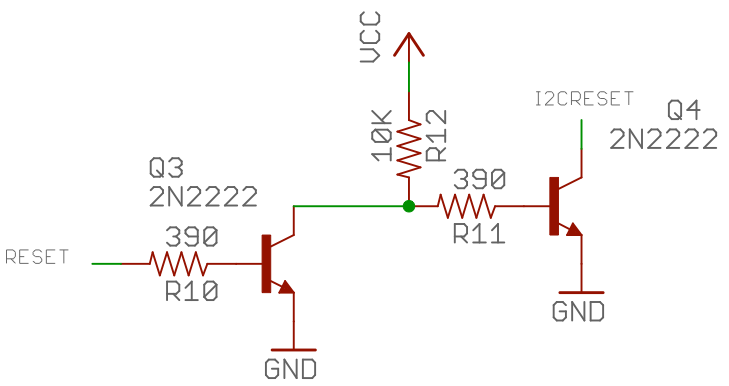
* MOSI (SPI data output)
* MISO (SPI data input)
* SPI CLK (SPI clock output)
* IRQ (interrupt request input)
* J2\_38 (Spare I/O line, input/output)
* WD\_Latch (Watchdog latch input)



Because of the open-collector nature of the I2C the I2C signals cannot pass through the TXB0108 IC. Adafruit makes a specialized I2C level-shifting breakout board, the BSS138 that uses a set of BSS132 Field-Effect Transistors (FETs) to to level shifting for the I2C bus. The SCL and SDA lines from the CPU interface pass through this level shifter.

The reset line is a special case. This is an output line from the CPU interface that instructs all the hardware modules to reset/initialize themselves and enter a fail-safe condition. The DAQ\_IF board needs to pass this signal to an I2C multiplexer (TCA9585 breakout board, also on the DAQ\_IF). However, this I2C breakout board has a pull-up resistor attached to this line. Pull-up resistors play havoc with the BX0108 level shifter chip so connecting the RST pin on the TCA9584 breakout board directly to the 5V side of the TXB0108 will force the reset pin constantly high (meaning you can never reset anything in the system with this pin).

To solve this problem, the (3.3V) reset signal is fed into a transistor pair (the second transistor exists simply to invert the inverted signal from the first transistor) and use that as a switch to pull the RST pin on the I2C multiplexer low when the reset signal is active (also low).



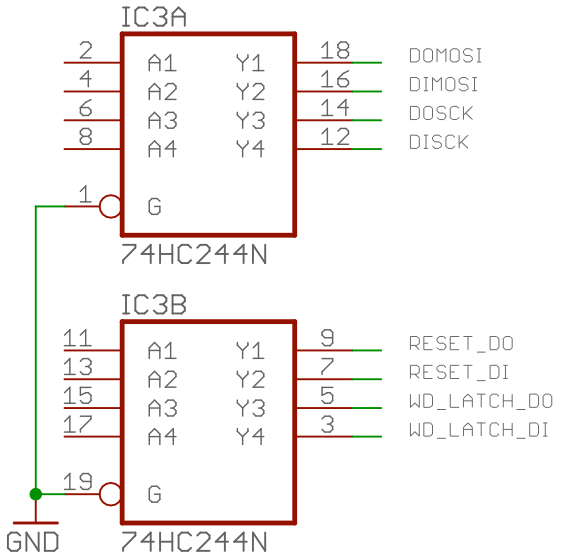
### Logic Signal Buffering

1. [DAQ\_HDD\_003]

The DAQ\_IF board transmits several of the digital signals it produces along the PPDO and PPDIO96 busses. A system designer can connect up to six PPDIO96 (or other SPI interfaces) to the PPDIO96 bus and a virtually unlimited number of PPRELAY or PPSSR (or other SPI interfaces) to the PPDO bus. Even if each board in the daisy-chain presents only one TTL load on this signals, the fanout could quickly render the signals unreliable and noisy.

To solve this problem, the DAQ\_IF board includes a 74HC244 (dual 4-channel buffer) IC to provide buffering for the following signals:

* MOSI (SPI output )
* SPK CLK
* RESET
* WD\_Latch



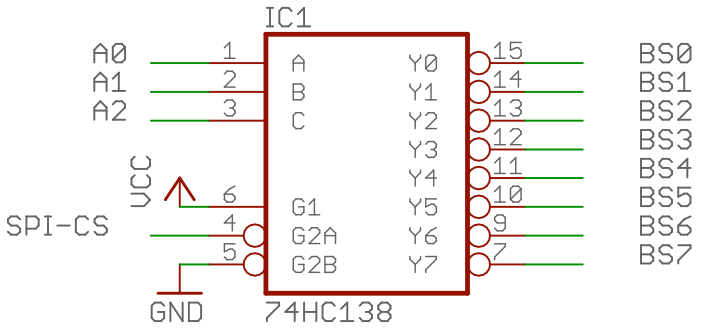
The DAQ\_IF board actually produces two copies of each of these signals. One buffered copy is provided on the PPDIO96 bus and the second buffered copy is placed on the PPDO bus. The 74HC255 chip supports the equivalent of 15 TTL loads on each of these two busses. As long as the boards attached to these busses present only 1 TTL load each (as specified by the hardware requirements) one could, in theory, attach up to 15 boards to each bus. In practice, the PPDIO96 bus is limited to 6 boards due to the number of SPI chip select lines present on the bus and the PPDO bus is probably limited to around 10 boards due to cable capacitance and other issues.

### SPI Chip Select

1. [DAQ\_HDD\_004]

The DAQ\_IF hardware requirements assume that there is a single SPI CS (chip select) line available from the CPU module (this could be a hard-wired SPI chip select line that is automatically handled by the on-board SPI hardware or a generic digital output line that is programmed via software). In addition to this chip select line, the DAQ\_IF expects three software-programmable output lines that can be decoded to produce eight separate SPI chip select lines.

The DAQ\_IF circuit labels the three software-driven chip select lines A0, A1, and A3. These form a 3-bit binary number producing a chip select value of 0 through 7. Connecting these lines to a 3-to-8 decoder (e.g., 74hc138) produces eight chip select lines (labeled BS0-BS7 for board select zero through board select seven). The actual SPI chip select line (usually handled by hardware) is used as one of the decoder/demultiplexer enable inputs.



The BS0 (board select zero) signal controls the PPDO bus. Boards attached to the PPDO bus (e.g., PPRELAY-12 and PPSSR-16) operate in a serial shift register fashion. That is, bits are shifted into the first board in the PPDO daisy chain and the previous bits from the first board are shifted to the second board; the previous bits from the second board are shifted to the third, etc. As such, only a single SPI chip select line is necessary to control any number of PPDO boards.

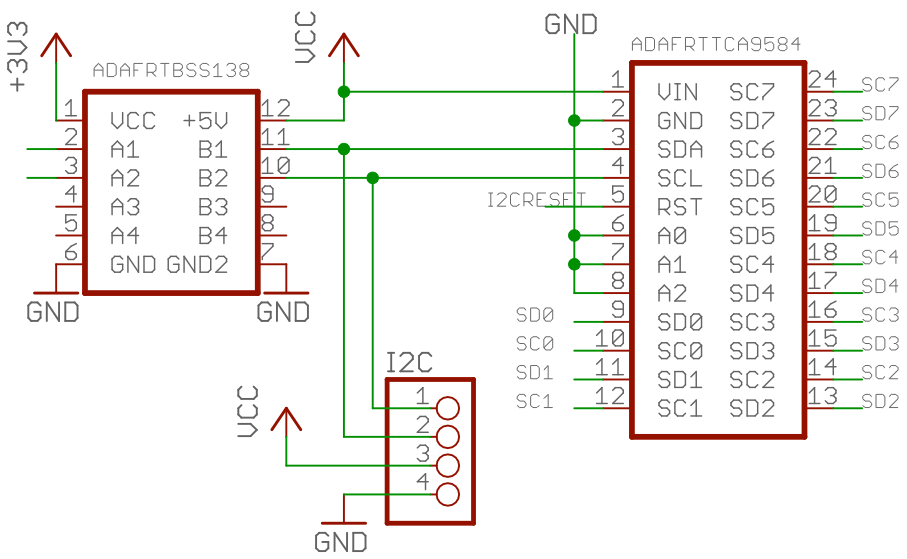
The BS1 through BS6 lines connect to the PPDIO96 bus connector. These six SPI chip select lines allow a system designer to connect six PPDIO96 boards (or other SPI devices) to the PPDIO96 bus.

The BS7 chip select line is intentionally left disconnected. The intent is to program the decoder with A0, A1, A2 equal to (1, 1, 1) whenever none of the other seven chip selects are active. Note that this convention is redundant – if the SPI-CS line is not active (low) then none of the BS*n* lines will be active (low).

### I2C Bus Expansion

1. [DAQ\_HDD\_005]

The DAQ\_IF uses an Adafruit TCA9584 breakout board to provide I2C expansion. The single I2C bus provided by the CPU module is fed into the TCA9584 to produce eight separate I2C busses. The TCA9584 IC consumes a single address in the I2C address space. Writing a byte to this I2C address with a single bit set activates the corresponding I2C bus on the TCA9584 until a different value is written to that address. On the DAQ\_IF, that magic address is 0x70 (obtained by wiring the three address lines on the TCA9584 breakout board to ground).



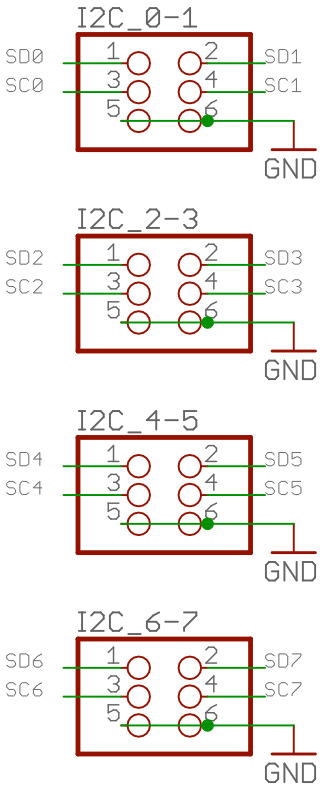
On power up, or whenever the I2CRESET signal is pulsed (low), the TCA9584 initializes to "no busses selected. Writing a zero to I2C address 0x70 also disables all busses. Note that the DAQ\_IF hardware includes a four-pin header with I2C signals (plus Vcc and Gnd) that is always active (as it sits before the TCA9584 breakout board).

Unlike the SPI bus and certain other signals, the DAQ\_F does not buffer the I2C bus signals. This is because (when using the PPAIO-16/4 boards) there aren't that many (TTL) loads placed on each I2C bus. However, if you connect other devices to the eight expansion I2C busses be aware of the fanout issues.

The DAQ\_IF board routes the eight expansion I2C busses to a set of four six-pin headers. These headers have the following pin out:

|  |  |  |  |
| --- | --- | --- | --- |
| Pin | Description | Pin | Description |
| 1 | SDA*n* | 2 | SDA*n+1* |
| 3 | SCL*n* | 4 | SCL*n+1* |
| 5 | Gnd | 6 | Gnd |

Each six-pin header comprises two I2C busses. The odd-numbered pins make up the first bus, the even-numbered pins make up the second bus. Between the four connectors, there are a total of eight I2C busses:



In a typical DAQ system setup, the PPAIO-16/4 boards connect to the I2C expansion busses. You may daisy-chain two PPAIO-16/4 boards on the same bus. The first PPAIO-16/4 uses the first I2C bus for its ADC modules, the second ADC module uses the second I2C bus for its modules.

Both PPAIO-16/4 boards use both busses for the DAC modules. The first board will deploy Adafruit MCP4725 DAC modules (I2C addresses 0x62 and 0x63) while the second board will use Sparkfun MCP4725 DAC modules (I2C addresses 0x60 and 0x61).

It would have been possible to have each board utilitize only a single I2C bus by mixing Adafruit and Sparkfun DACs on the same PPAIO-16/4 board. However, the current design allows you to provide up to 16 DACs in the system (and 64 single-ended ADCs) using parts from a single supplier (Adafruit) by using four non-daisy-chained PPAIO-16/4 boards connected to the four I2C bus connectors. You would only have to deal with a second supplier (Sparkfun) if you need ore than 16 DACs in the system.

### Application-Defined I/O Pin

1. [DAQ\_HDD\_006]

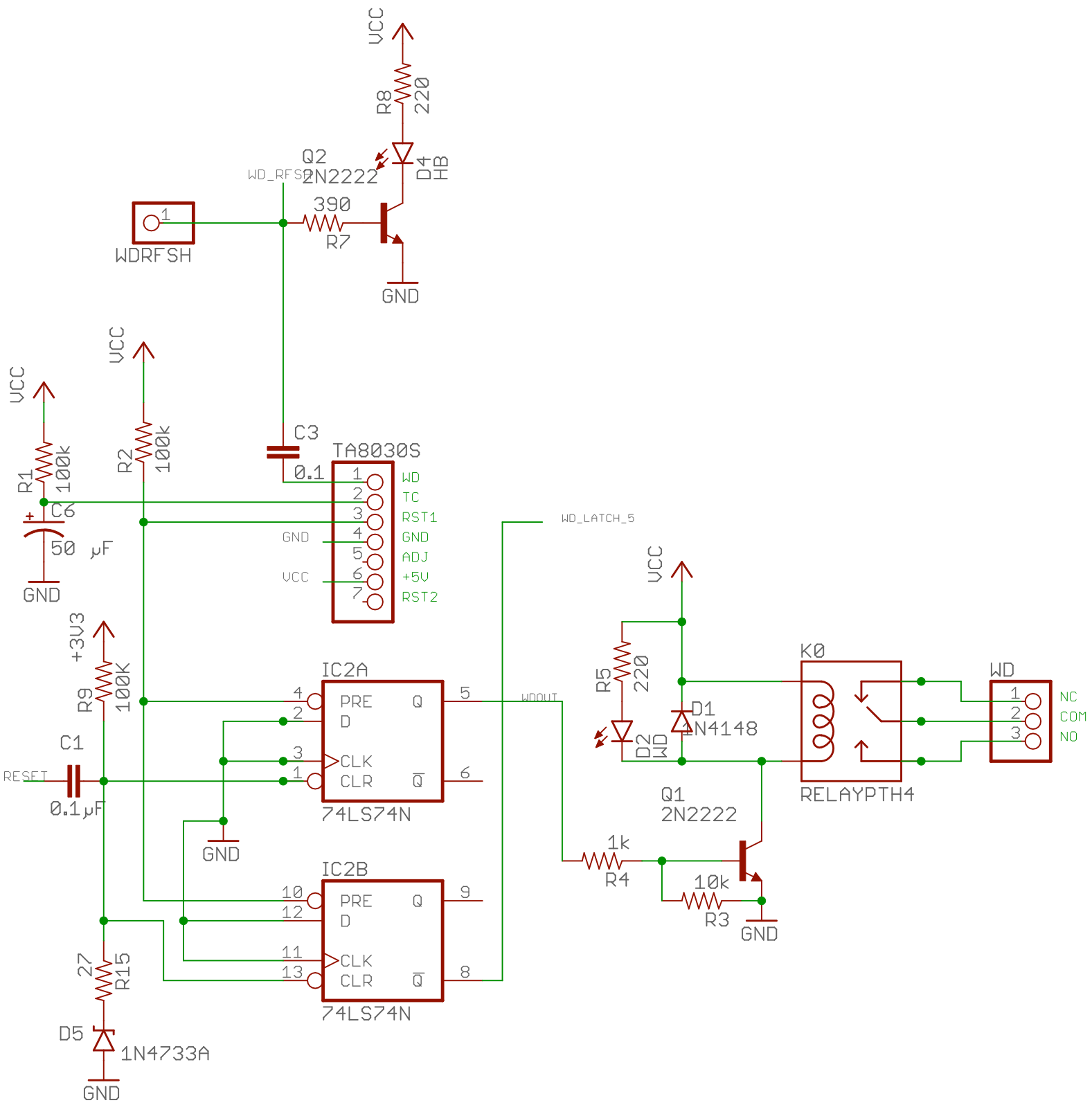
The DAQ\_IF board requires a single application-defined I/O pin. This signal will be present on both the PPDO and PPDIO96 busses. The signal is named "J2\_38" (the name comes from the fact that this is pin 38 on the J2 connector of the Netburner MOD54415 Evaluation Board; obviously this name doesn't have a direct meaning on the Teensy or Pi CPU modules).

This pin is application-defined and may be programmed as an input or an output pin on all CPU modules. Note that this pin does not run through a buffer, so there is a limit to the number of inputs (TTL loads) it can drive on the busses when being used as an output pin.

### DAQ\_IF Watchdog Timer

1. [DAQ\_HDD\_007]

The DAQ\_IF board provides a watchdog timer circuit with latching relay.



The watchdog timeout is provided by a Toshiba TA8030S watchdog timer chip. This chip produces an active low signal on the RST1 pin whenever a power on condition occurs or whenever a timeout period occurs. The timeout period is specified by the RC time constant obtained from R1 and C6 in the above schematic (100K \* 50 µC is approximately 5 seconds plus or minus the accuracy of the RC components). The requirements state that the watchdog timeout period is a minimum of two seconds and a maximum of ten seconds. This RC circuit safely implements that timeout period.

The TA8030S documentation states that maximum width of the reset pulse is 0.6\*R1\*C6 in the circuit above. This implies that the RST1 signal could be held low for as long as three seconds (nominally) after a watchdog timeout period. As this signal drives the preset lines on a pair of 74HC74 flip-flops (the watchdog timeout latches), this means that the system should not attempt to reset (clear) the watchdog latches until the RST1 circuit goes high. There is no input to the system (software) telling it that this signal is no longer active. Therefore, whenever the system sees that the WD\_LATCH signal is active (this is the WD\_LATCH\_5 signal coming from the second 74HC74 flip-flop above), the system should wait five seconds before attempting to reset the watchdog latches.

The system resets the watchdog latches by sending an active-low pulse on the RESET output line. Note that activating this signal will also reset all the other boards in the system (this toggles the reset pins on all the MCP23S17 I/O expander chips found on the PPDIO96 boards, it puts the PPRELAY-12 and PPSSR-16 boards into fail-safe mode, and it resets the I2C expander chip on the DAQ\_IF board).

In the circuit above, a capacitor appears in series with the RESET signal and the CLR lines on the flip-flops. This prevents an errant always-low signal on the RESET line from constantly clearing the watchdog state on the 74HC74 flip-flops. A pull-up resistor keeps this signal high when it's not being actively pulled low. The combination of the two produces a brief pulse to clear the flip-flops when an active-low signal is first sent. A zener diode (1N4733A) is added to the circuit to shunt excessively high voltages to ground (this occurs on the rising edge of the reset signal) to protect the inputs on the 74HC74 chip.

The WD\_RFSH signal refreshes the TA8030S watchdog timer. The CPU module must pulse this pin (typically a low-to-high-to-low pulse) every two seconds, or faster, to prevent the watchdog timer module from timing out. Note that this pin also drives an LED. The intent is to invert the WD\_RFSH at (better than) twice the normal refresh interval (i.e., invert the pin faster than once per second) to obtain a "heartbeat" flashing LED effect.

Note that both halves of the 74HC74 package are provided with the same inputs. One flip-flop produces the WD\_LATCH\_5 signal which the DAQ\_IF passes on to the CPU module and two the PPDO and PPDIO96 busses. The boards on the busses can use this signal to determine that a watchdog failure has occurred (and switch to fail-safe mode, if desired). The CPU can monitor this signal to determine if there has been a watchdog failure (assuming the CPU is still running) and send an appropropriate reset signal after dealing with the fact that there has been a watchdog timeout.

The other flip-flop drives the relay circuit. It goes to a transistor that turns on a relay whenever the watchdog condition is latched. This also illuiminates an LED so a human can see that the WD timeout condition is active. The 1N4148 diode across the relay coil pins is a standard clamping diode used to short high-voltages produced by inductive kick-back across the relay coil.

The NC/NO outputs from the relay are presented on a three-pin connector (NC/COM/NO) and are application-defined. The DAQ\_IF board does not use the relay except as a way of signaling external hardware that a timeout has occurred.

### DAQ\_IF Power LED

1. [DAQ\_HDD\_008]

The DAQ\_IF board, like most boards in the DAQ system, provides an LED that illuminates whenever power is present on the board.

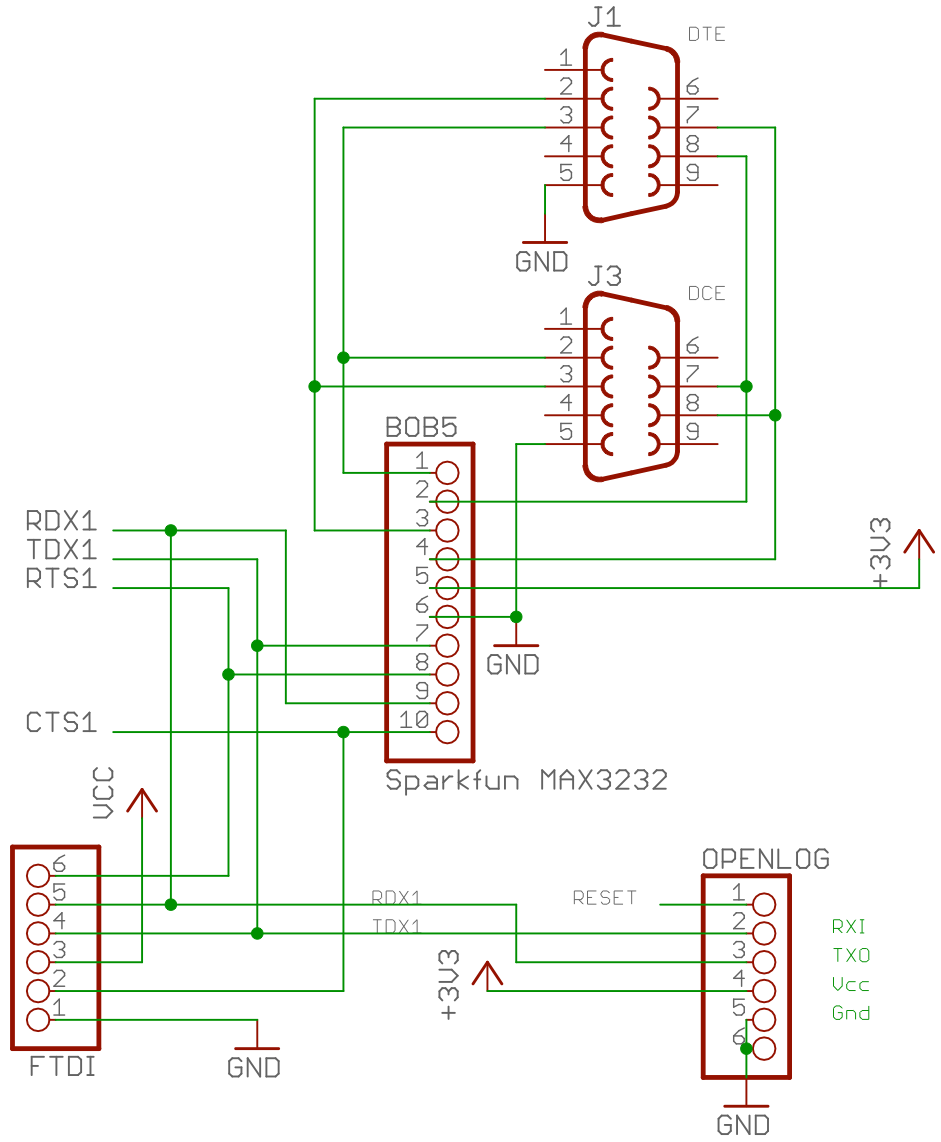


### DAQ\_IF RS-232

1. [DAQ\_HDD\_009]

The DAQ\_IF board provides RS-232 level shifting to take TTL level serial communication signals and amplify them to RS-232 levels. In addition to the level shifting, the DAQ\_IF board also provides two DB-9 connectors (male and female, DTE and DCE) to allow connection to standard RS-232 serial devices.

The DAQ\_IF board alternately allows connection of the TTL serial signals to an FTDI-compatible header (allowing the connection of an FTDI-compatible USB-to-serial cable) or the connection of a SparkFun "OpenLog" serial logging module. Note that only one of these three options (RS-232 level shifting, FTDI cable, or OpenLog module) may be populated on the board. Installing two or more of these devices would create signal conflicts.



If RS-232 level shifting (to the DB-9 connectors) is desired, then the DAQ\_IF should populate the BoB5 socket with a SparkFun MAX3232 breakout board. This breakout board contains a Maxim IC (SMD) that will convert between TTL level signals and RS-232 signals. The MAX3232 breakout board automatically produces the ±12V needed for RS-232 communications without any additional power supplies. Note that the DTE and DCE DB9 connectors are wired in parallel with each other (with appropriate pins swapped). Generally you would only connect to one of these connectors at a time (depending on what you're connecting to). Connectiing active signals to both connectors would create signal conflicts.

When using the Sparkfun MAX3232 breakout board, you must leave the FTDI header and the OpenLog circuits unpopulated.

An alternative to using the DB9 RS-232 connectors is to use the FTDI header. FTDI is a special 6-pin USB-to-serial cable used for microcontroller programming (e.g., many Arduino-type boards use this cable). FTDI cables are very common and relatively inexpensive (typically under $10 US).

When using the FTDI header with an FTDI cable, you must leave the SparkFun MAX3232 and OpenLog sockets unpopulated.

A third alternative is to use a Sparkfun OpenLog module. This module contains a small CPU and a connection for a flash card to log all data written to the serial port (using the OpenLog protocol). In practice, the OpenLog only makes sense when using the Teensy 3.2 CPU module. On the Raspberry Pi you (presumably) have lots of Flash storage to write log files to. The Netburner module cannot access the serial facilities on the DAQ\_IF (the Netburner has its own serial ports and doesn't use those on the DAQ\_IF).

When using the OpenLog header on the DAQ\_IF, you must leave the SparkFun MAX3232 and FTDI sockets unpopulated.

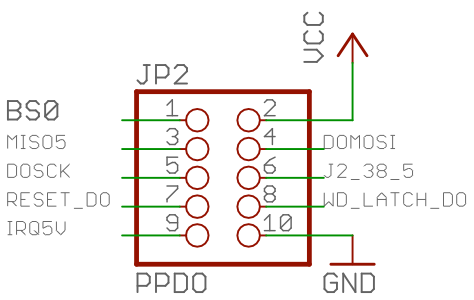
As just noted above, the Netburner MOD54425 Evaluation board has two (level-shifted) RS-232 ports present on the eval board itself. Therefore, RS-232 communications take place directly on the Netburner board itself, those signals are not routed to the DAQ\_IF circuitry.

### PPDO Connection

1. [DAQ\_HDD\_010]

The PPDO bus on the DAQ\_IF board is a 10-pin header (2x5) with the following signals:

|  |  |  |  |
| --- | --- | --- | --- |
| Pin | Description | Pin | Description |
| 1 | Output: BS0 (SPI Chip Select) | 2 | Vcc |
| 3 | Input: MISO (SPI input) | 4 | Output: MOSI (SPI output) |
| 5 | Output: SPI Clk | 6 | I/O: J2\_38 |
| 7 | Output: RESET | 8 | Output: WD\_LATCH |
| 9 | Input: IRQ | 10 | Gnd |



The PPDO bus is intended for use by the PPRELAY-12 and PPSSR-16 boards. Note that these boards do not use the IRQ, J2\_38, and MISO signals (the boards do not have any need for interrupt capabilities, hence no IRQ support; J2\_38 is reserved for end-user applications, and these board are strictly output boards so there is no need for the MISO signal). However, these signals are included for future expansion on the PPDO bus.

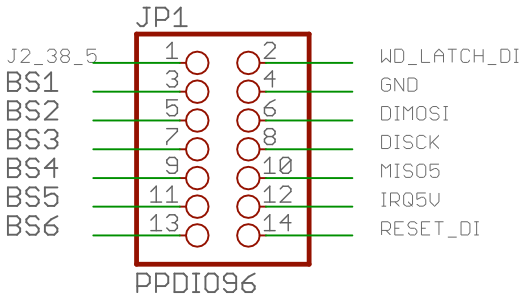
All signals on the PPDO bus are 5V logic signals having passed through the Adafruit TXB0108 bi-directional level shifter. Futhermore, the MOSI, SPI-Clk, Reset, and WD\_Latch signals are also buffered via a 74HC244 dual quad non-inverting buffer chip,

### PPDIO96 Connection

1. [DAQ\_HDD\_011]

The PPDIO96 bus on the DAQ\_IF board is a 14-pin header (2x7) with the following signals:

|  |  |  |  |
| --- | --- | --- | --- |
| Pin | Description | Pin | Description |
| 1 | I/O: J2\_38 | 2 | Output: WD\_LATCH |
| 3 | Output: BS1 | 4 | Gnd |
| 5 | Output: BS2 | 6 | Output: MOSI (SPI data out) |
| 7 | Output: BS3 | 8 | Output: SPI Clk |
| 9 | Output: BS4 | 10 | Input: MISO (SPI data in) |
| 11 | Output: BS5 | 12 | Input: IRQ |
| 13 | Output: BS6 | 14 | Output: Reset |

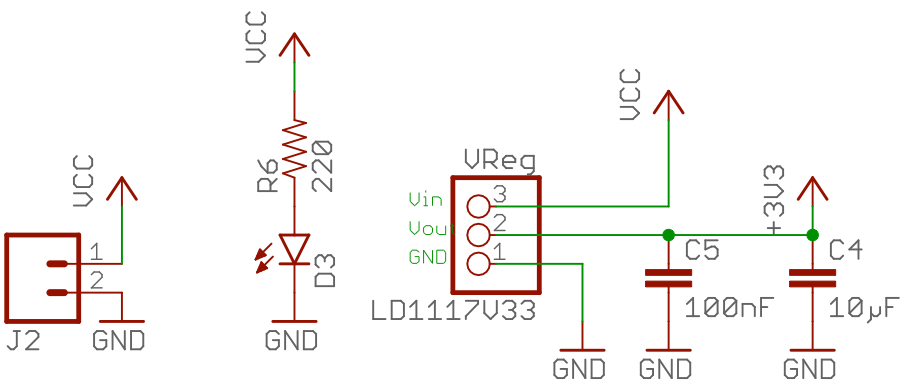


All signals on the PPDIO96 bus connector are 5V logical signals. The MOSI, SPI Clk, WD\_Latch, and Reset signals are also buffered (independently of the same signals on the PPDO connector). The BSn signals are not buffered, but as they are intended for consumption by a singla PPDIO96 board, this shouldn't be a problem; however, if you connect a different board (or many boards) with many TTL loads to one of these signals, you should consider buffering them on that board(s).

### DAQ\_IF Power Supply

1. [DAQ\_HDD\_012]

The DAQ\_IF board shall use a single +5V power supply that connects to the board via a screw-terminal module with two terminals (+5V and Gnd). Application of power to this power supply jack will illuminate the power on LED on the board. Because several components on the board require 3.3V in addition to the components that require 5V, the DAQ\_IF board provides a 3.3V voltage regulator circuit:



The 3.3V regulator does not provide power to the Teensy 3.2, Raspberry Pi, or Netburner modules (all of which require far more power than the LD1117v33 regulator can provide). The Teensy 3.2 gets its power from the +5V supply, the Pi and Netburner modules have separate power supply inputs. On the DAQ\_IF board, the 3.3V supply is largely used for the level shifters.

1. Note that the Teensy 3.2 is 5V tolerant, but is a 3.3V processor. The DAQ\_IF interface board uses 3.3V signals when connected to a Teensy 3.2. [↑](#footnote-ref-1)