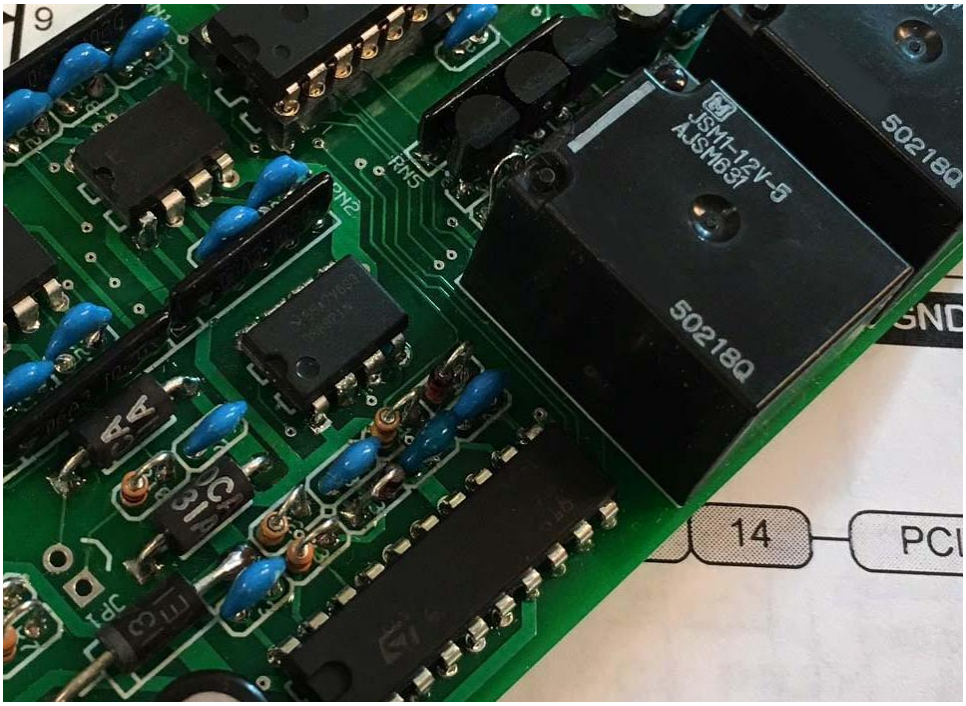
FALCON-AVR™ ARDUINO COMPATIBLE MOTION CONTROLLER USER GUIDE



1. DESCRIPTION

The Falcon-AVR™ is a general-purpose Arduino compatible motion controller, using the Microchip/Atmel ATMega328P microprocessor. Originally developed for aircraft applications, it provides three H-bridge or six unipolar motor drivers, general purpose inputs and outputs, relay and lamp drivers, and a long-haul serial interface in a robust, environmentally hardened design. Operating at input voltages from 9 to 35 volts, the device is ideally suited to automotive, marine and aeronautical applications.

Arduino (www.arduino.cc) is an open-source hardware/software integrated development environment (IDE), originally created as an educational tool. It has been widely adopted by hobbyists, students, and small to medium sized engineering companies as the platform of choice for quick-turn development of custom microcontroller applications. Developers worldwide support the millions of Arduino users, providing readily available, low cost hardware and open-source software.

Unfortunately, the basic Arduino Uno processor does not actually do anything other than blink a light! To make it more useful, you must purchase or design custom 'shields' that plug into the Arduino platform, design or purchase a proper power supply and find an enclosure for the project. Even so, despite a working prototype, the robustness and reliability of the project is questionable. Suspect electrical systems, static discharge, strong radio frequency fields, and accidental short circuits or even residual lightning strikes make for a hostile environment. What works in the lab, may not work in the field! In the end, the low-cost Arduino platform, shields, and power supplies that have been put together end up costing many times the original purchase cost of the Arduino processor board and the result is not production worthy.

The Falcon-AVR, however, was developed to be a production-ready prototyping system. Fully compatible with the Arduino or Atmel Studio development environments, it provides robust power conditioning and four levels of protection from electrostatic discharge, over voltages, and short circuits. It uses reliable industry standard D-subminiature connector for low-current connections and robust fast-on tabs for high-current connections. Best of all, it saves time and money by fitting into a standard Hammond enclosure.

The Falcon-AVR intentionally uses through-hole components to facilitate hardware customization, field modifications and long-term serviceability. Five external connectors are provided: The primary connector is a 25-pin D-subminiature receptacle that provides pins for the low-current (1-2 Amp) motor driver channels and miscellaneous I/O. The secondary connector uses fast-on tabs to support the high current (10 Amp intermittent) relay-based motor drivers.

Provided on board is a six-pin rectangular in-circuit serial programming (ICSP), a six-pin linear bootloader connector for Arduino programming via USB, and a WiFi/I2C connector that supports plug-in modules for LCD displays, wireless communication and other peripheral devices.

FEATURES:

- Microchip/Atmel ATMega328P based three channel full-bridge or six channel single-ended motor and lighting controller.
- Compatible with the Arduino, Atmel Studio, and other commercial development tools for ease of software development.
- Designed entirely with through-hole components to facilitate customization, field modification and maintenance.
- Operates from 9 to 35 Volts DC with -40 °C to +85 °C ambient temperatures and provides extensive power and I/O conditioning to increase reliability in harsh environments.
- I/O capability
    - Nine general-purpose protected I/O pins, including three with 5 Volt analog input capabilities and three with 50 Volt withstanding capability. Configurable to six 5 Volt analog inputs.
    - Four 1 Amp PWM outputs plus two 10 Amp relay outputs for driving motors or other heavy loads. Configurable in bridge mode for forward/reverse motor control.
    - Two 50 Volt open-collector I/O channels for driving external lamps or relays, paired with input capabilities for read-back.
    - One general purpose 50 Volt open collector output for driving external lamps or relays.
    - One on board adjustable potentiometer.
    - Provides a full duplex RS-422 communications interface for reliable long-haul communications. Compatible with RS-232 for short-haul applications. Pins also usable for general purpose I/O.
    - Provides on-board header pin connectors for in-circuit serial programming (ICSP), a USB to serial (U2S) interface for bootloading, an I2C or serial display, and a WiFi wireless module.
- Uses a standard Hammond enclosure to reduce packaging costs in end-use applications.

APPLICATIONS:

- Automotive, Marine or Aeronautical control systems
- Lighting controller
- General purpose laboratory controller
- Data acquisition system

2. APPLICATION EXAMPLES

AIRCRAFT TWO AXIS AUTO-TRIM PLUS FLAPS CONTROLLER

The application example shown in Figure 1 demonstrates an aircraft two-axis auto-trim plus flaps controller. It controls the pitch and roll trim servos using the PWM H-bridge motor drive channels and the flaps motor using the on-board H-bridge relay deck.

The EFIS/EMS System (compatible with Dynon SkyView or Garmin G3X) provides important airspeed/attitude data, sensor data and control data via a serial RS-232 link at 115,200 kilobits per second. This information is extracted to provide control of the trims and flaps depending on airspeed, sensor values and autopilot control information. Advanced features such as speed-sensitive flaps control, trim/flaps preset, trim speed scheduling and auto-trim are all possible by interpreting the serial data link information. A switch input and lamp is used for field programming of settings and operational mode configuration, such as flaps preset.

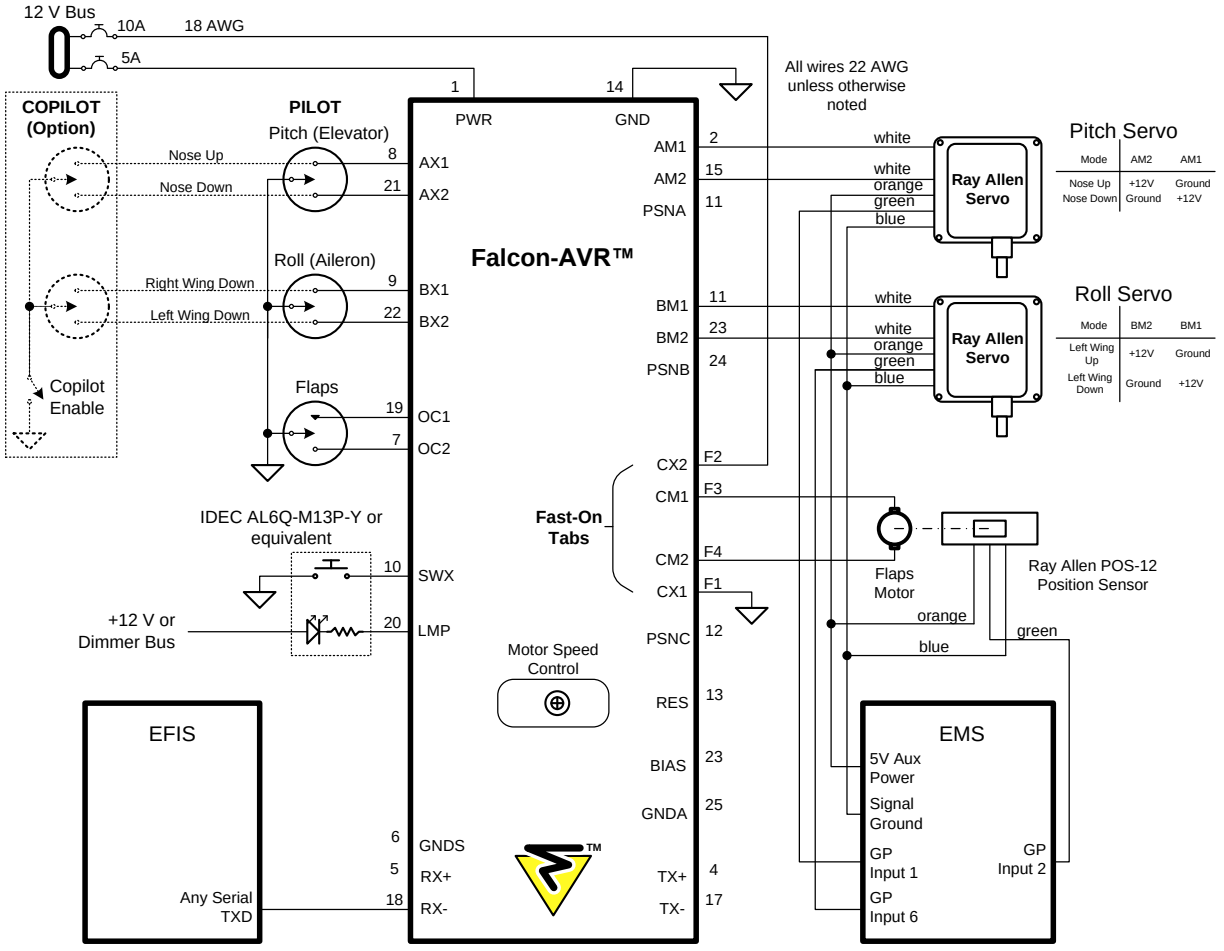


Figure 1. Aircraft Two Axis Trim and Flaps Controller, EFIS/EMS Mode.

STAND-ALONE AIRCRAFT TWO AXIS TRIM PLUS FLAPS CONTROLLER

Figure 2 shows an example of how an application similar to the example in Figure 1 can be supported without an EFIS/EMS system attached. In this case, the controller reads the trim and flaps positions directly using the sensors embedded in the trim motors or the external flaps position sensor. For trim speed scheduling, an external airspeed switch may be used on the RX+ input pin.

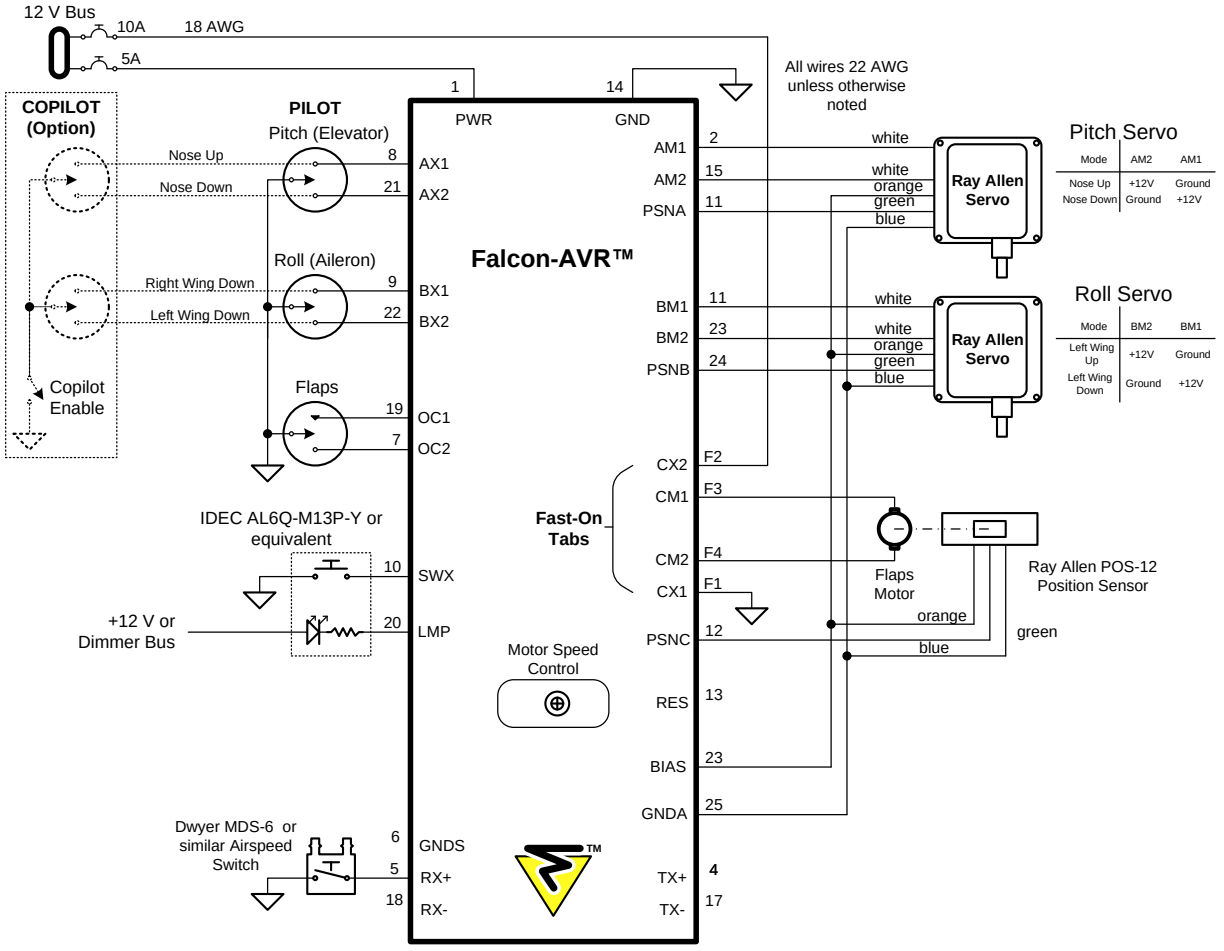


Figure 2. Aircraft Two-Axis Trim and Flaps Controller, Stand-Alone Mode.

AUTOMATIC FUEL PUMP & LIGHTING CONTROLLER

Figure 3 shows an example of how the Falcon-AVR may be configured as a general purpose automatic fuel pump and lighting controller.

The fuel pump switch has Off, Auto and On positions. Fuel pressure is derived from the fuel pressure switch/sensor. Alternatively, if a compatible EFIS/EMS is installed, fuel pressure may be extracted from the incoming serial data stream. In the On position, the fuel pump switch directly controls the fuel pump relay without software intervention. In the Off or Auto positions, the fuel pump is controlled in software. The software detects when fuel pressure is too low, as indicated by the fuel pressure switch/sensor. Then, the fuel pump is turned on by using the LMP pin as a relay driver.

The landing and taxi light switches select Off, Flash (Wig-Wag) or On. In the On position, the switches control the OC1 and OC2 pins, which directly drive the onboard relays without software intervention. In the Off or Auto positions, the lights are controlled in software. Lamp power is provided with the onboard relays, up to 5 Amps each. The software can generate custom flashing or wig-wag patterns as an anti-collision warning. If an external pressure sensor switch is added, patterns can be changed depending on airspeed. Instead, if a compatible EFIS/EMS system is installed, flashing patterns can be changed according to airspeed, altitude or other conditions extracted from the incoming serial data stream.

Two dimmer busses are provided for instrument and map lights, controlled by the corresponding adjustable controls. Up to 1 Amp loads on AM1 or AM2 are supported using pulse-width-modulation (PWM) to control the lamp intensity.

Appendix A shows an example Arduino sketch that implements the Application as described, in stand-alone mode without an EFIS/EMS or Airspeed Switch.

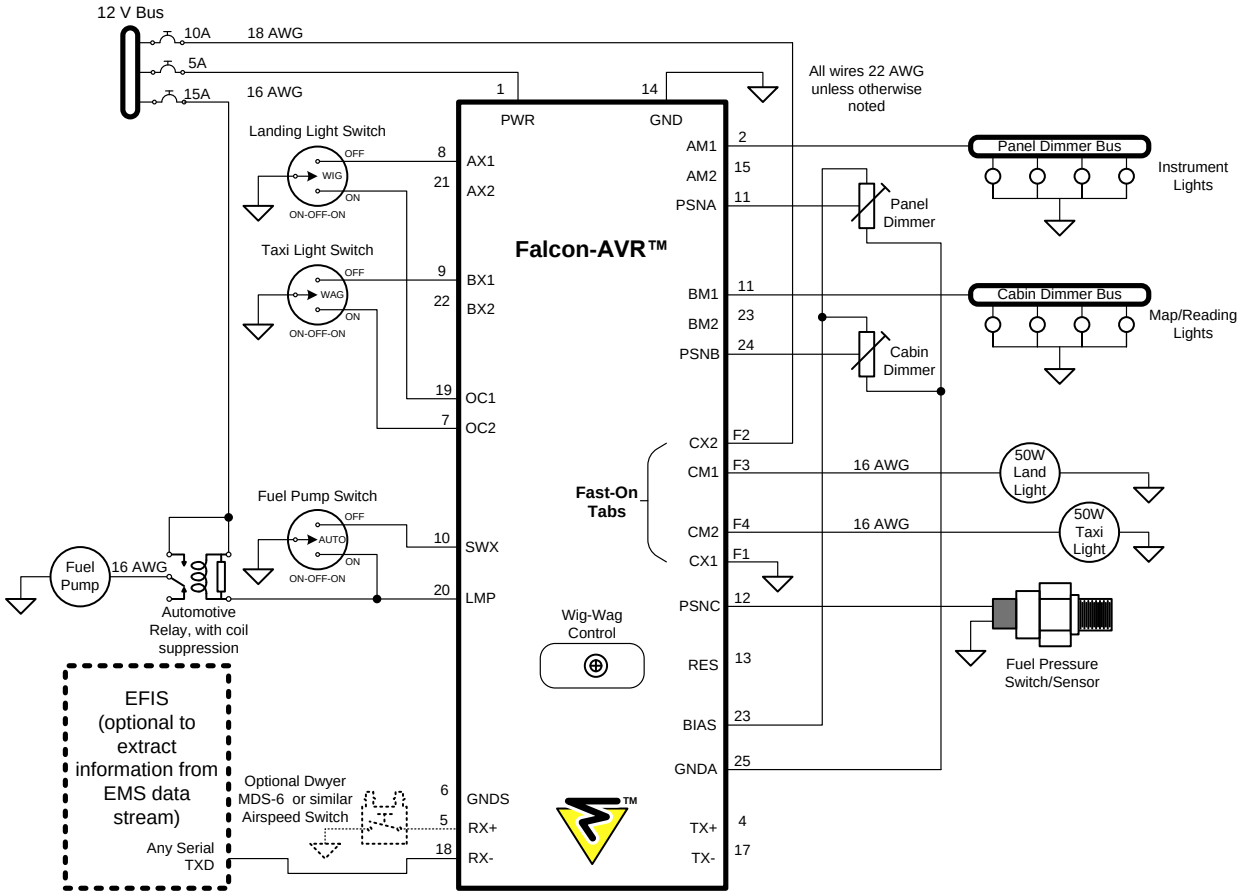


Figure 3. Aircraft Fuel Pump and Lighting Controller.

3. HARDWARE

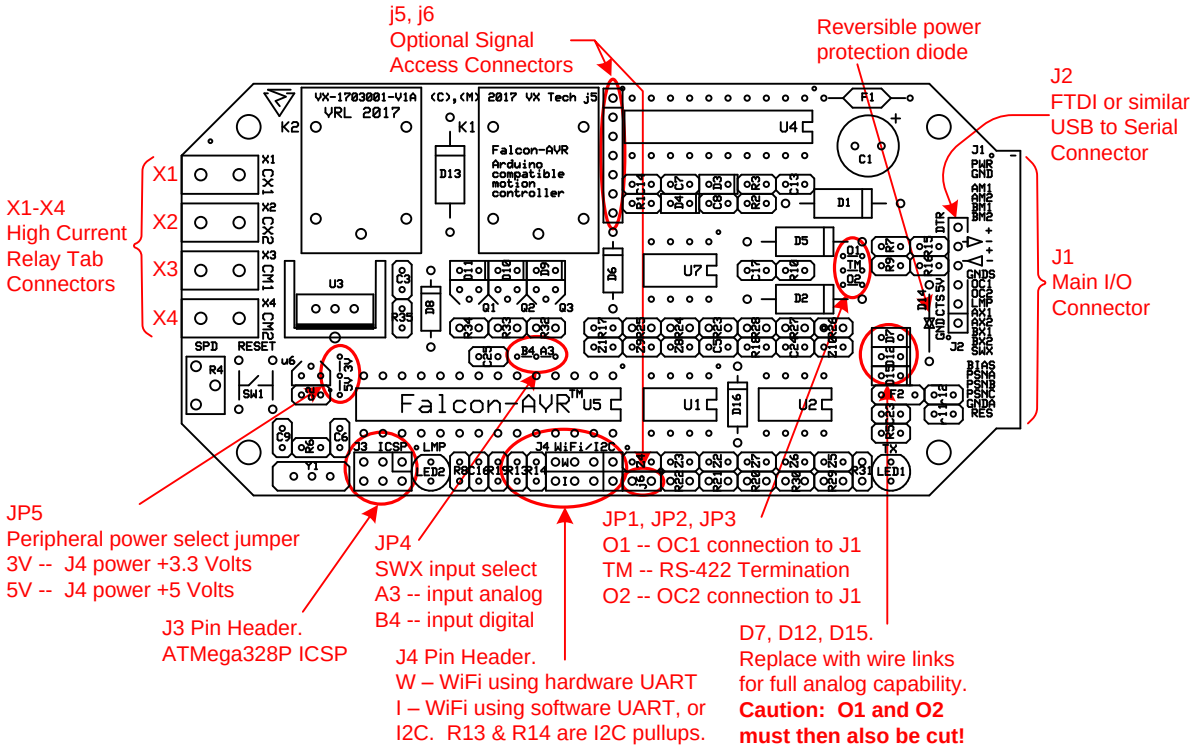


Figure 4. Falcon-AVR Connector & Jumper Placement

The Falcon-AVR device has seven connectors, as shown in Figure 4.

- Connector J1:
  J1 is a standard female 25-pin D-subminiature for connecting the main board power, special and general purpose inputs and outputs, RS-422 communication and the low-current motor drivers.

- Connector J2:
  J2 supports an FTDI or Arduino USB 2 Serial Micro card, available from a number of sources. It is a standard 6-pin female socket that is easy to convert to a male plug by using standard pin headers strips inserted into the socket.

- Connector J3:
  J3 is a 6-pin rectangular in-circuit serial programmer (ICSP) port for directly programming the ATMega328P processor chip using an Atmel Studio IDE, or an Arduino Uno configured as an ICSP.

- Connector J4:
  J4 serves two purposes. The top part of J4, labeled 'W' is compatible with common 5 Volt ESP8266 WiFi adapter modules and connects to the on-board hardware serial port.

The lower part of J4, labeled 'I', is similar but supports software serial on CPU pins 27 and 28 (Arduino A4/SDA and A5/SCL). These pins on the 'I' section of J4 are also used for a standard I2C (Inter-Chip Communications) interface or software serial on pins A4/A5.

The top 'W' part of J4 can be used independently of the lower 'I' part. For example, an ESP8266 WiFi module can be plugged into the top part and a LCD display module (either serial or I2C) can be connected to the lower part. Note: if the 'W' part of J4 is in use, remove any hardware serial devices installed on J2.

Connectors j5 and j6:
j5 and j6 are optional connectors that provide access to internal signals.

- Fast-On Tabs:
  The four fast-on tabs, X1-X4 are for the connection of high power loads, controlled by the onboard relays.

The Jumper options are described in the schematic diagram (Section 6) and in the Pin Description, below.

PIN DESCRIPTION

J1 25 Pin DSub Pinout				
Pin Number	**Pin Name**	**AVR Function**	**Arduino Function**	**Pin Description**
1	PWR	PWR	--	9 to 35 Volt power Input.
2	AM1	PD6	6	Motor driver channel A1.
3	BM1	PB3	11	Motor driver channel B1
4	TX+	TX+	TX/1	*Serial Port TX+ output, logically paired with TX-.
5	RX+	RX+	RX/0	*Serial Port RX+ input, logically paired with RX-.
6	GNDS	GND	GND	Shield Ground for serial communications.
7	OC2	PC5, PB2	A5/19, 10	Input with analog, plus open collector output (K2 relay drive) if JP3 is installed. Also connected to J4, pin 8 as I2C SCL pin.
8	AX1	PD2	2	General purpose I/O.
9	BX1	PD7	7	General purpose I/O.
10	SWX	PB4	12	General Purpose I/O, 30 Volt tolerant.
11	PSNA	PC0	A0/14	General purpose I/O, with analog input capability.
12	PSNC	PC2	A2/16	General purpose I/O, with analog input capability.
13	RES	PC6/RESET	RESET	Reset or general purpose I/O. 100 kΩ pullup to 5V.
14	GND	GND	GND	Main power ground.
15	AM2	PD5	5	Motor driver channel A2.
16	BM2	PD3	3	Motor driver channel B2.
17	TXD-	TXD-	TX/1	*Serial Port TX- output, logically paired with TX+.
18	RXD-	RXD-	RX/0	*Serial Port RX- input, logically paired with RX+.
19	OC1	PC4, PB1	A4/18, 9	Input with analog, plus open collector output (K1 relay drive) if JP1 is installed. Also connected to J4, pin 7 as I2C SDA pin.
20	LMP	PB5	13	Open Collector Out for driving relays or lamps.
21	AX2	PD4	4	General purpose I/O.
22	BX2	PB0	8	General purpose I/O.
23	BIAS	--	--	5 Volt sensor power.
24	PSNB	PC1	A1/15	General purpose I/O, with analog input capability.
25	GNDA	GND	GND	Analog sensor Ground.

*NOTE ON RS-232 COMPATIBILTY

For most short-haul applications, the built-in RS-422 interface is compatible with modern RS-232 data links. The RS-422 interface transmits and receives signals between 0 and 5 Volts. Simply by using the inverting signals (TX- and RX-) to make the appropriate RS-232 connections, serial data can be sent over several meters.

The RS-422 interface is also usable for general purpose input (on RX+ or RX-) or output (on TX+ or TX-) when not used for serial data communications. RX+ and TX+ may also be directly connected to the RX and TX pins on the Arduino Uno for serial bootloading of sketches. For proper RS-232 reception, any unused RX+ or RX- input should be left unconnected.

J2 6 Pin U2S Serial Bootloader Device or WiFi Module Pinout

Pin Number	Pin Name	AVR Function	Arduino Function	Pin Description
1	GND	GND	--	Ground for U2S device.
2	GND	--	--	Ground for WiFi module.
3	5V	VCC	5V	External Power from U2S device, if required. Reverse D14 to enable powering an external 5 Volt WiFi module.
4	RX	PD0	RX/0	Serial Port RX In, from TX on U2S device or WiFi module.
5	TX	PD1	TX/1	Serial Port TX Out, to RX on U2S device or WiFi module.
6	DTR	RESET	RESET	Data Terminal Ready from U2S device

J3 6 Pin ICSP Pinout

Pin Number	Pin Name	AVR Function	Arduino Function	Pin Description
1	MISO	PB4/MISO	12/MISO	SPI Master In/Slave Out
2	5V	VCC	5V	5 Volt Power into Falcon-AVR
3	SCL	PB5/SCL	13/SCL	SPI Serial Clock Input
4	MOSI	PB3/MOSI	11/MOSI	SPI Master Out/Slave In
5	RES	PC6/RESET	RESET	Chip Reset or PortC, 6
6	GND	GND	GND	Ground

J4 (W Section) 8 Pin WiFi/I2C Connector Pinout

Pin Number	Pin Name	AVR Function	Arduino Function	Pin Description
1	GND	GND	GND	Ground
2	VDD			5 or 3.3 Volt power to external device
3	RXD	PD0/RXD	0/RX	Receive serial data in
4	TXD	PD1/TXD	1/TX	Transmit serial data out

J4 (I Section) 8 Pin WiFi/I2C Connector Pinout				
Pin Number	**Pin Name**	**AVR Function**	**Arduino Function**	**Pin Description**
5	GND	GND	GND	Ground
6	VDD			5 or 3.3 Volt power to external device and I2C pull-ups
7	SDA	PC4/SDA	19/A4	I2C serial data or general purpose analog/digital I/O
8	SCL	PC5/SCL	18/A5	I2C serial clock or general purpose analog/digital I/O

j5 (optional) 7 Pin Access Connector Pinout				
Pin Number	**Pin Name**	**AVR Function**	**Arduino Function**	**Pin Description**
1	PD6	PD6	6	Port D, 6 output
2	PD5	PD5	5	Port D, 5 output
3	PB3	PB3	11/MOSI	Port B, 3 output
4	PD3	PD3	3	Port D, 3 output
5	5V	VCC	5V	5 Volt power
6	3.3V			Optional 3.3 Volt peripheral power
7	AREF	AREF	AREF	Analog reference voltage

j6 (optional) 2 Pin Access Connector Pinout				
Pin Number	**Pin Name**	**AVR Function**	**Arduino Function**	**Pin Description**
1	PC3	PC3	A3	PC3 or analog input A3
2	GND	GND	GND	Ground

X1-X4 4 Pin Fast-On Pinout				
Pin Number	**Pin Name**	**AVR Function**	**Arduino Function**	**Pin Description**
1	CX1	--	--	Relay Pole 1. Normally connected to Ground.
2	CX2	--	--	Relay Pole 2. Normally connected to Power.
3	CM1	PB1	15	Relay Output, Channel 1. PB1 also drives J1, OC1 when O1 jumper is installed.
4	CM2	PB2	16	Relay Output, Channel 2. PB2 also drives J1, OC2 when O2 jumper is installed.

JUMPER CONFIGURATION

Jumper Options				
Jumper	**Function**	**Default**	**Option**	**Note**
O1	OC1 output enable	Enable OC1 as an output (connected).	To disable OC1 as a high drive open collector output, but retain the high voltage input capability, remove O1 wire. For normal analog/digital I/O or I2C operation on the J1-OC1 pin, also remove diode D7 and replace with a jumper wire.	Default mode allows both high voltage input and open-collector output functions. To allow I2C operation on J4 (and not on J1), just remove diode D7.
TM	RS-422 Rx Termination	Not connected.	To enable RS-422 input termination, jumper together.	Leave open for RS-232 input.
O2	OC2 output enable	Enable OC2 as an output (connected).	To disable OC2 as a high drive open collector output, but retain the high voltage input capability, remove O2 wire. For normal analog/digital I/O or I2C operation on the J1-OC2 pin, also remove diode D12 and replace with a jumper wire connection	Default mode allows both high voltage input and open-collector output functions. To allow I2C operation, on J4 (and not on J1), just remove diode D12.
B4, A3	SWX digital/ analog	SWX high voltage input . B4 pin jumpered to the center pin.	To connect the SWX input to the low voltage analog capable pin PC3/ADC3, remove the existing B4 jumper and connect the A3 jumper to the center pin.	Remove associated analog components on PC3/ADC3 processor input, as required. See Note*
3V, 5V	Peripheral Power	+5 Volts. 5V jumpered to the center pin.	For + 3.3 Volt peripheral power, remove 5V jumper and jumper 3V to the center pin.	For peripheral devices that require 3.3 Volt power (250 mA max).

*Note: Configuring an Additional Analog Input on J1:

To convert the main connector (J1) SWX pin from a digital-only high voltage tolerant I/O to a general purpose 5 Volt digital/analog I/O:

- Remove R4 and z1 (if installed).
- remove D15 and replace it with a wire jumper,
- remove the existing B4 jumper and connect the center and A3 pins together, and
- change the software assignment of SWX to PC3 or Arduino A3.

This connects processor pin PC3/ADC3 (pin 26, Arduino A3/17) to the J1 SWX pin.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Min	Max	Units	Conditions
PWR	Power Supply Voltage	-1	60	Volts	With respect to GND. Clamped internally. Exceeding limits may blow internal fuse.
V_{IN}	Input Voltage, standard I/O	-12	17	Volts	10 seconds maximum. Limited by heating of internal input protection resistor.
V_{IN}	Input Voltage, SWX input	-12	60	Volts	Negative voltage, 10 seconds maximum. Limited by heating of internal input protection resistor and diode.
V_{IN}	Input Voltage, RX+, RX-	-7	+7	Volts	Internally clamped to 7 Volts.
V_{OUT}	Output Voltage, standard I/O	-12	17	Volts	10 seconds maximum. Limited by heating of internal protection resistor.
V_{OUT}	Output Voltage, open collector I/O	-0.5	60	Volts	Negative voltage, 10 seconds maximum.
V_{OUT}	Output Voltage, TX+, TX-	-7	+7	Volts	
V_{OUT}	Output Voltage, BIAS pin	-0.5	+5.5	Volts	
I_S	AM1, AM2 or BM1, BM2		2.8	Amps RMS	Total load on each pair of outputs.
I_{SPK}	AM1, AM2 or BM1, BM2		7.1	Amps peak	Instantaneous load on each pair of outputs.
I_{S12}	CX1, CX2, CM1, CM2		15	Amps	Switching current load through onboard relays (14 Volt option).
I_{S24}	CX1, CX2, CM1, CM2		10	Amps	Switching current load through onboard relays (28 Volt option).
T_A	Ambient Operating Temperature	-40	+55	°C	Non-condensing.

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units	Conditions
PWR	Power Supply Voltage	9	35	Volts	With respect to GND.
V_{IN}	Input Voltage, standard I/O	-0.3	5.5	Volts	
V_{IN}	Input Voltage, SWX input	-0.3	50	Volts	
V_{IN}	Input Voltage, RX+, RX-	-0.3	5.5	Volts	
V_{OUT}	Output Voltage, standard I/O	-0.3	5.5	Volts	
V_{OUT}	Output Voltage, open collector I/O	-0.3	50	Volts	
V_{OUT}	Output Voltage, TX+, TX-	-0.3	5.5	Volts	
I_S	AM1, AM2, or BM1, BM2		1.4	Amps RMS	Total load for each pair of pins.
I_S	AM1, AM2, BM1, BM2		2.8	Amps RMS	Total load on all pins.
I_S	CX1, CX2, CM1, CM2		10	Amps	Total peak or sustained current load through either or both of the onboard relays.
FSW	PWM operating frequency		100	KHz	
DATA RATE	RS-422 operating rate		250	Kbit/s	

4. CUSTOMIZING THE INPUTS AND OUTPUTS

The Falcon-AVR is designed for customization. It uses through-hole components for ease of modification, assembly or rework in the field.

The best time to make changes to the component types or values is before initial assembly. Usually this will involve input/output (I/O) circuits as shown as options A, B or C in Figure 5. Input Configuration Options. To determine the correct modifications to make, the final application must be analyzed. In particular, the input voltage levels and types must be determined.

Option A: Typically, input voltages will be between 0 and V_{DD} (5 Volts), such as seen from devices like pressure sensors or hall-effect current sensors. In this case, a 1.0 kΩ input resistor and an optional filter capacitor is all that is required in the associated 'Zxx' position on the circuit board. In this configuration, either analog or digital signals can be accommodated and the pin can also be configured as an output.

Option B: Sometimes you need to measure higher level signals, such as battery voltages. In this case, the standard Option A circuit needs to be modified to provide input attenuation.

For example, to measure a 15 Volt input, the input attenuator should be set to 3:1 (the ratio of 15:5) by configuring the input voltage divider as shown in Figure 5. Input Configuration Options This ensures that the microprocessor input does not exceed 5 Volts. Even though there are on-board protection devices to prevent over voltages and damage, it's not a good idea to depend on these for regular operation. In this case, the internal microprocessor pull-up resistors should not be enabled.

It is also possible (and maybe advisable to limit power dissipation) to scale the resistor values up or down, depending on the application, as long as the ratio stays the same. For example, an input resistor of 10 kΩ and an attenuation resistor in the associated 'Zxx' position of 5.0 kΩ will have a power dissipation of 0.019 Watts, which is well within the resistor specifications.

To calculate the resistor values, use the following formula:

Let V_{in} = 30 Volts and V_{DD} = 5.0 Volts
$R_A = K/(1-K) * R_S$

Example, if R_S = 10 kΩ
K = 5.0/30 = 0.167
R_A = 0.167/(1-0.167) * 10 kΩ = 2.0 kΩ.

To calculate the total resistor power dissipation, use the following formula:

$P_{DISS} = (V_{in}^2)/(R_{total}) = (30^2)/(10k + 2.0k) = 900/12.0k = 75$ mW. Use 1/8 or 1/6 W resistors.

Option C: For higher level *digital* signals, three of the device inputs are configured with protection diodes. These diodes allow up to 50 Volts on the inputs. When used as inputs, these must always be configured with the internal microprocessor pull-up resistors selected and R_S must be at most 1.0 KΩ.

When configuring these pins as outputs, the pins cannot sink (pull down) any current, only drive (pull up) current through the diode and resistor. While it is possible to configure these as analog inputs as well, the presence of the diode and pull-up resistor may significantly alter measured voltages. In this case, the diodes can be replaced with jumpers to convert the pins to Option A applications which support both analog and digital signals.

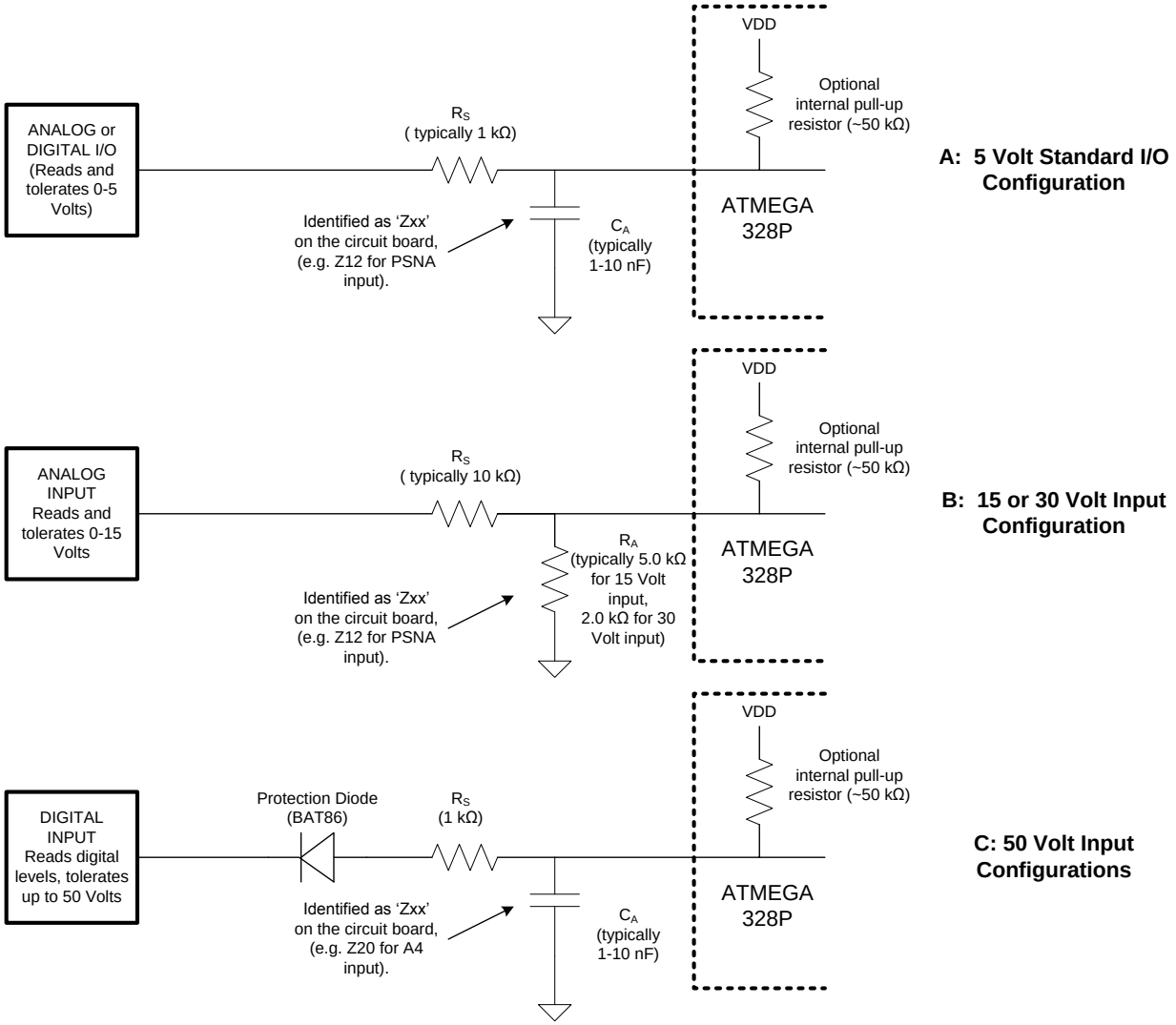


Figure 5. Input Configuration Options

5. CONFIGURING THE ARDUINO ENVIRONMENT

HARDWARE SETUP

The Falcon-AVR provides a convenient way of using the Arduino IDE to download sketches to the board and use all of the Arduino tools and libraries. The following description provides the options for working with the Falcon-AVR to load Arduino sketches. *Please ensure that the Falcon-AVR board is configured for an RS-232 and not an RS-422 serial interface. Jumper JP2 must be open (the factory default).*

OPTION 1: USING THE ARDUINO OR FTDI USB 2 SERIAL MICRO

Purchase an Arduino USB 2 Serial Micro or the similar FTDI device. You may also need a 6-pin header strip, depending on how the USB to serial device is equipped. These devices plug into the U2S connector on the Falcon-AVR and connect to the host computer with a USB cable. With the appropriate USB software drivers installed, you are now ready to program with the Arduino IDE.

OPTION 2: RELOCATING THE PROCESSOR CHIP

This option assumes that you have a working Arduino Uno.

The ATMega328P processor chip from the Arduino Uno must be carefully removed from the Uno. If there is not a processor with a bootloader already installed on the Falcon-AVR , insert the one removed from the Uno into the available socket. The Uno processor contains the bootloader that allows the Arduino IDE to talk to the Falcon-AVR. For more information on how to load a bootloader on the ATMega328P device, refer to the Burning the Bootloader section.

Note: The ATMega328P is a static sensitive device. Ensure you are at a properly grounded ESD (electrostatic discharge) workstation and you are properly grounded yourself. Use proper tools to extract or insert the processor, the pins are easily damaged.

CONNECT THE WIRES

Connect jumper wires between the Arduino Uno to the Falcon-AVR using one of the following options:

Arduino Uno	**Falcon-AVR (option 1)** **J1 25-pin Connector**	**Falcon-AVR (option 2)** **J2 6-pin U2S Connector**
5V	23 (BIAS)	3 (5V)
GND	6 (GND)	1 (GND)
RX	5 (RX+)	4 (RX)
TX	4 (TX+)	5 (TX)
RESET	13 (RES)	6 (DTR)

Note: DO NOT connect to a 3.3 Volt Arduino board! Use only the 5 Volt Arduino Uno.

TESTING

Using the Arduino USB 2 Serial Micro or equivalent (Option 1) or the Arduino Uno (Option 2, with the processor chip removed), configure the serial port (Tools:Port:<choose>) and select Arduino Uno from the target list (Tools:Board: Arduino/Genduino Uno). Load the example program 'Blink' (File:Examples:01.Basics:Blink) and modify the program as follows:

TEST SKETCH

```
void setup() {
// initialize the TX+ pin as an output.
   pinMode(1, OUTPUT);
}

void loop() {
   digitalWrite(1, HIGH);   // turn the LED off (HIGH is the voltage level)
   delay(1000);             // wait for a second
   digitalWrite(1, LOW);    // turn the LED on by making the voltage LOW
   delay(1000);             // wait for a second
}
```

LED1 should now be blinking on the Falcon-AVR. If it is not, or you get an error message from the Arduino IDE, check the configuration and wiring (Option 2).

Assuming that the 'Blink' sketch worked on the Falcon-AVR, you are now ready to develop your own sketches. Congratulations!

BURNING THE BOOTLOADER

Sometimes it is necessary to burn a bootloader onto the ATMega328P chip. Normally, this is required when:

- Installing newly purchased ATMega328P chips,
- After using the Atmel Studio IDE (which erases the bootloader) then you wish to use the Arduino IDE, or
- You have accidentally erased or overwritten the bootloader on the ATMega328P chip.

The procedure for burning a bootloader requires that you have a functional Arduino Uno to serve as a programmer. The procedure is as follows:

1. Connect pins 1, 2, 3, 4, and 6 of the Falcon-AVR board's ICSP connector to the same pins on the Arduino Uno board's ICSP connector (the one nearest the ATMega328P chip). Ensure that you have correctly identified the proper pins on each board. Pin 1 is usually marked with a dot or a square surrounding the pin. See Appendix C. Falcon-AVR Schematic Diagram for more information on the ICSP connector.
2. Connect pin 5 of the Falcon-AVR board ICSP connector to the Uno board's pin 10 (marked as ~10 on the Uno board).
3. Plug in the Uno and properly configure the Arduino environment, then load the sketch "ArdunioISP" from (File:Examples:01.Basics:11.ArduionoISP:ArduinoISP) and click the Upload arrow → to program the Uno.
4. When it's done Uploading, select Tools:Burn Bootloader. After a few seconds, the operation will be completed and the message "Done burning bootloader" will be displayed.
5. Unplug the Uno and remove the connections to the Falcon-AVR ICSP connector before proceeding.

The ATMega328P chip on the Falcon-AVR will now be able to accept Arduino sketches when configured according to the instructions in the Hardware Setup section.

APPENDIX A. EXAMPLE ARDUINO SKETCH FOR FUEL PUMP AND LIGHTING CONTROL

```
//
// Falcon-AVR Demonstration Program
// "Fuel Pump and Lights"
// Written by Vernon Little for the Falcon-AVR(tm) Motion Controller.
// Version 1.1, April, 2017
//
// Fuel Pump Operation:
// Assumes a fuel pressure sensor connected to the PSNC input
// and an ON-OFF-ON FP switch.
// One pole of the FP switch is connected to the SWX pin.
// The center pole of the FP switch is grounded,
// and the third pole connected to the LMP pin.
// The coil of a 12 volt relay is connected to the LMP pin.
// When the FP switch is set to 'Auto', the device automatically
// switches on the fuel pump relay when
// the Pressure sensor connected to the PSNC input is high,
// which indicates low pressure.
// If the PSNC input is low, indicating a normal pressure,
// the fuel pump relay is switched off.
// When the FP switch is 'On', the fuel pump is always on.
// When FP switch is 'Off', the fuel pump is always off.
//
// Landing and Taxi Light Operation:
// Assumes an ON-OFF-ON switch for each of the Landing or Taxi lights.
// One pole of the Landing light switch is connected to AX1,
// the center pole is grounded and the third pole is connected to OC1.
// Similarly, one pole of the Taxi light switch is connected to BX1,
// the center pole is grounded, and the third pole is connected to OC2.
// The switches can turn the lights Off, Flash, or On,
// with the center position being the Flash position.
// When flashing, each lamp follows an 8-tick binary pattern
// as defined in the flash1 and flash2 variables.
//
// Lamp Dimmer Operation:
// Assumes a dimmer control for each of the two dimming channels,
// Panel Dimmer and Cabin Dimmer.
// The analog dimmer values control the pulse-width-modulation of the
// AM1 and BM2 outputs, respectively.
// Each output can drive up to 1 Amp in total load.
// The program can easily be modified to also drive AM2 and BM2
// if more loads are required.
//

//
// Falcon-AVR pin cross reference to Arduino pin definitions
//
```



```
const byte AM1 = 6, AM2 = 5, BM1 = 11, BM2 = 3;
const byte TX = 1, RX = 0;
const byte PSNA = 14, PSNB = 15, PSNC = 16, SPD = 17; //A0, A1, A2, A3
const byte OC1IN = 18, OC2IN = 19; //A4, A5
const byte OC1OUT = 9, OC2OUT = 10;
const byte AX1 = 2, AX2 = 4, BX1 = 7, BX2 = 8;
const byte SWX = 12, LMP = 13, RES = 1;
const byte PressureThreshold = 20;  // PSI. Only if analog pressure sender used.
const byte shift = 1;
byte flash1 = 0b10101100;
byte flash2 = 0b01010011;
int sensorValue;
byte outputValue;
byte FuelPressure;
byte FPump;


//
// Byte rotate function for flashing patterns
//

byte rotateRight (byte value, byte amount)
 {
  amount &= 0b00000111;
  return (value>>amount|value<<(8-amount));
 }

void setup() {
  pinMode(AM1, OUTPUT);
  pinMode(AM2, OUTPUT);
  pinMode(BM1, OUTPUT);
  pinMode(BM2, OUTPUT);
  pinMode(TX, OUTPUT);
  pinMode(OC1OUT, OUTPUT);
  pinMode(OC1IN, INPUT_PULLUP);
  pinMode(OC2OUT, OUTPUT);
  pinMode(OC2IN, INPUT_PULLUP);
  pinMode(AX1, INPUT_PULLUP);
  pinMode(AX2, INPUT_PULLUP);
  pinMode(BX1, INPUT_PULLUP);
  pinMode(BX2, INPUT_PULLUP);
  pinMode(SWX, INPUT_PULLUP);
  pinMode(PSNA, INPUT);
  pinMode(PSNB, INPUT);
  pinMode(PSNC, INPUT_PULLUP);

  Serial.begin(9600);
}

//
// Main program loop, repeats at a rate set by the SPD control.
//
```



```
void loop() {

    delay(analogRead(SPD)*2);  // Sets the flash rate and program scanning rate

//
// Landing and Taxi light flashing (wig-wag)
// Adjust variables flash1 and flash2 to set individual flash pattern sequences.
// Flash pattern repeats every eight cycles.
//

    flash1 =  rotateRight (flash1, shift);
    flash2 =  rotateRight (flash2, shift);

    if ((flash1 & bit(0)) && (byte) digitalRead(AX1)) digitalWrite (OC1OUT, HIGH);
else digitalWrite (OC1OUT, LOW);

    if ((flash2 & bit(0)) && (byte) digitalRead(BX1)) digitalWrite (OC2OUT, HIGH);
else digitalWrite (OC2OUT, LOW);

    Serial.println ("tick");

//
// Panel Dimmer
// External Panel Dimmer control changes the pulse width on AM1.
// Up to 1 Amp of lighting loads may be supported.
// Program may be modified to also drive AM2 for more loads.
//

     sensorValue = analogRead(PSNA);
     outputValue = map(sensorValue, 0, 1023, 0, 255);
     analogWrite(AM1, outputValue);

//
// Map/Reading Light Dimmer
// External Map/Reading light Dimmer control changes the pulse width on BM1.
// Up to 1 Amp of lighting loads may be supported.
// Program may be modified to also drive BM2 for more loads.
//

     sensorValue = analogRead(PSNB);
     outputValue = map(sensorValue, 0, 1023, 0, 255);
     analogWrite(BM1, outputValue);

//
// Fuel Pump Switch
// External ON-OFF-ON FP switch:
// If the FP switch is in the AUTO position,
// the fuel pump is turned on automatically if the external
// Pressure Switch on the POSNC input is off (high), indicating low fuel pressure.
// If the FP switch is in the OFF position, the fuel pump is turned off.
// If the FP switch is in the ON position, the switch will overide
// the internal logic and force the fuel pump on.
//
```



```
// Analog pressure senders are easily accommodated by changing to an
// analogRead of the input, followed by a
// threshold comparison as indicated in the notes below.
//

      FuelPressure = digitalRead(PSNC);
      FPump = digitalRead(SWX);
      if (FuelPressure & FPump) digitalWrite(LMP, LOW);
      else digitalWrite(LMP, HIGH);

//      FuelPressure = analogRead(PSNC);
//      FPump = digitalRead(SWX);
//      if ((FuelPressure > PressureThreshold) & FPump) digitalWrite(LMP, LOW);
//      else digitalWrite(LMP, HIGH);
}
```


6. APPENDIX B. PIN CONFIGURATION CROSS-REFERENCE

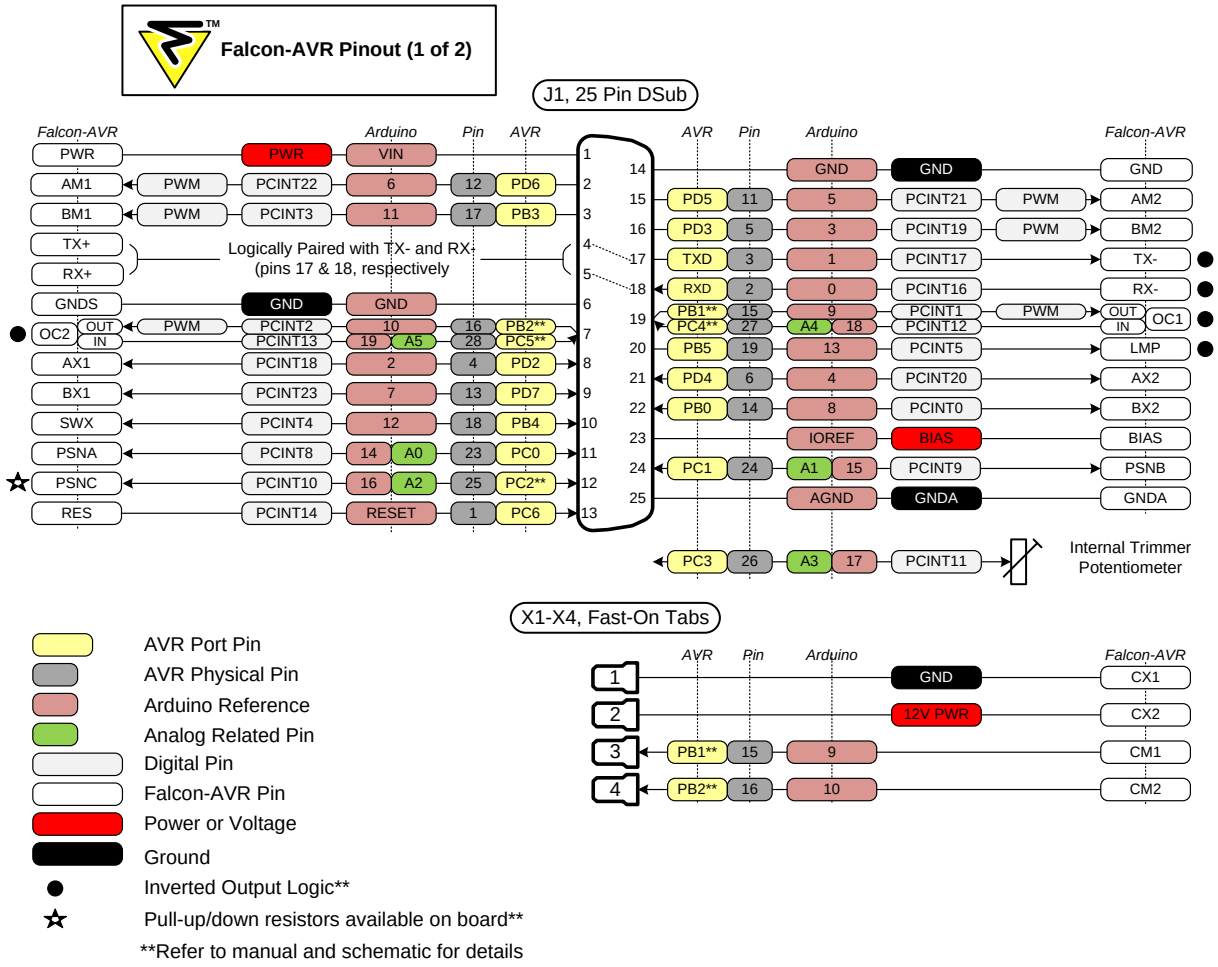


Figure 6. Falcon-AVR Pin Configuration Cross-Reference (1/2)

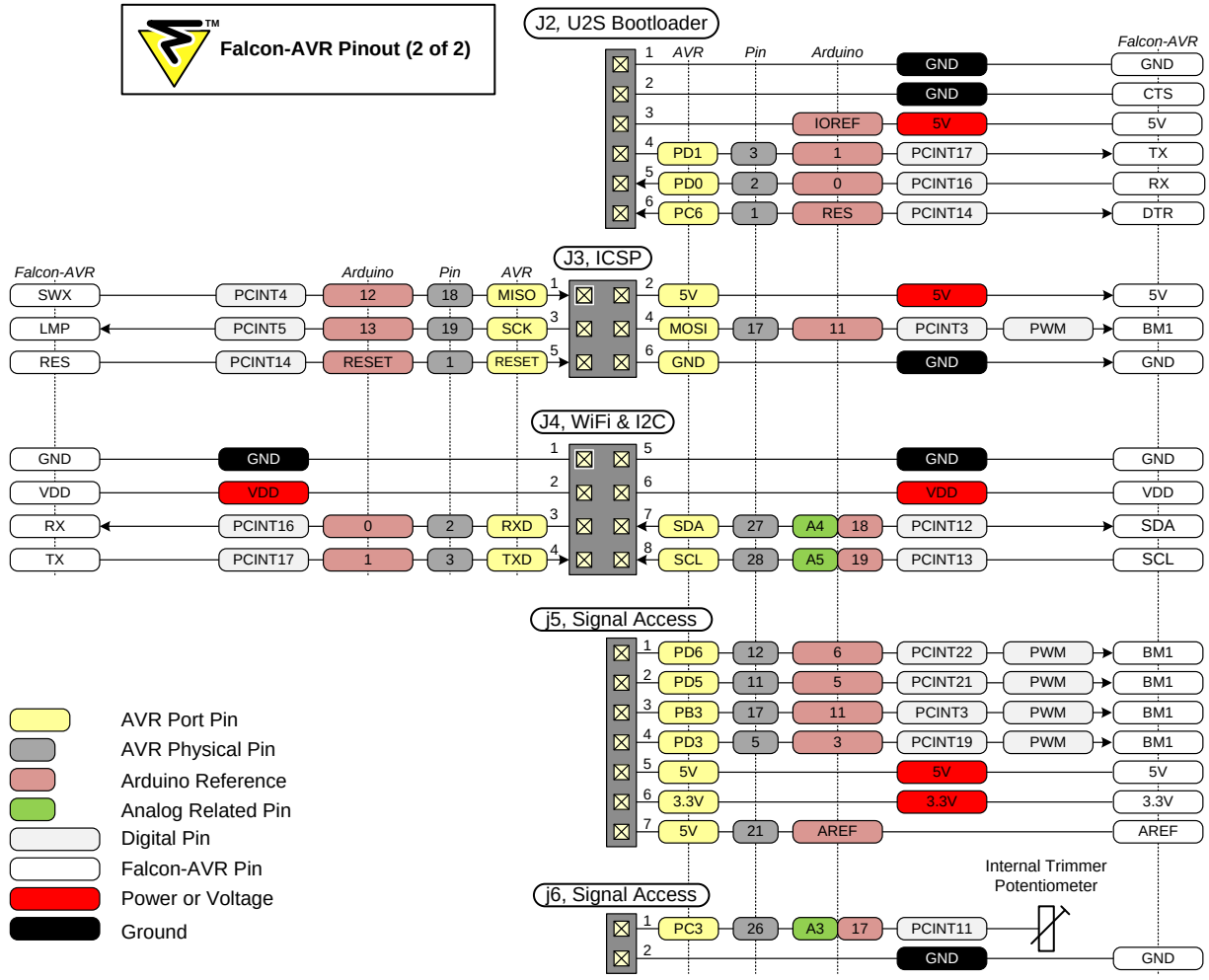
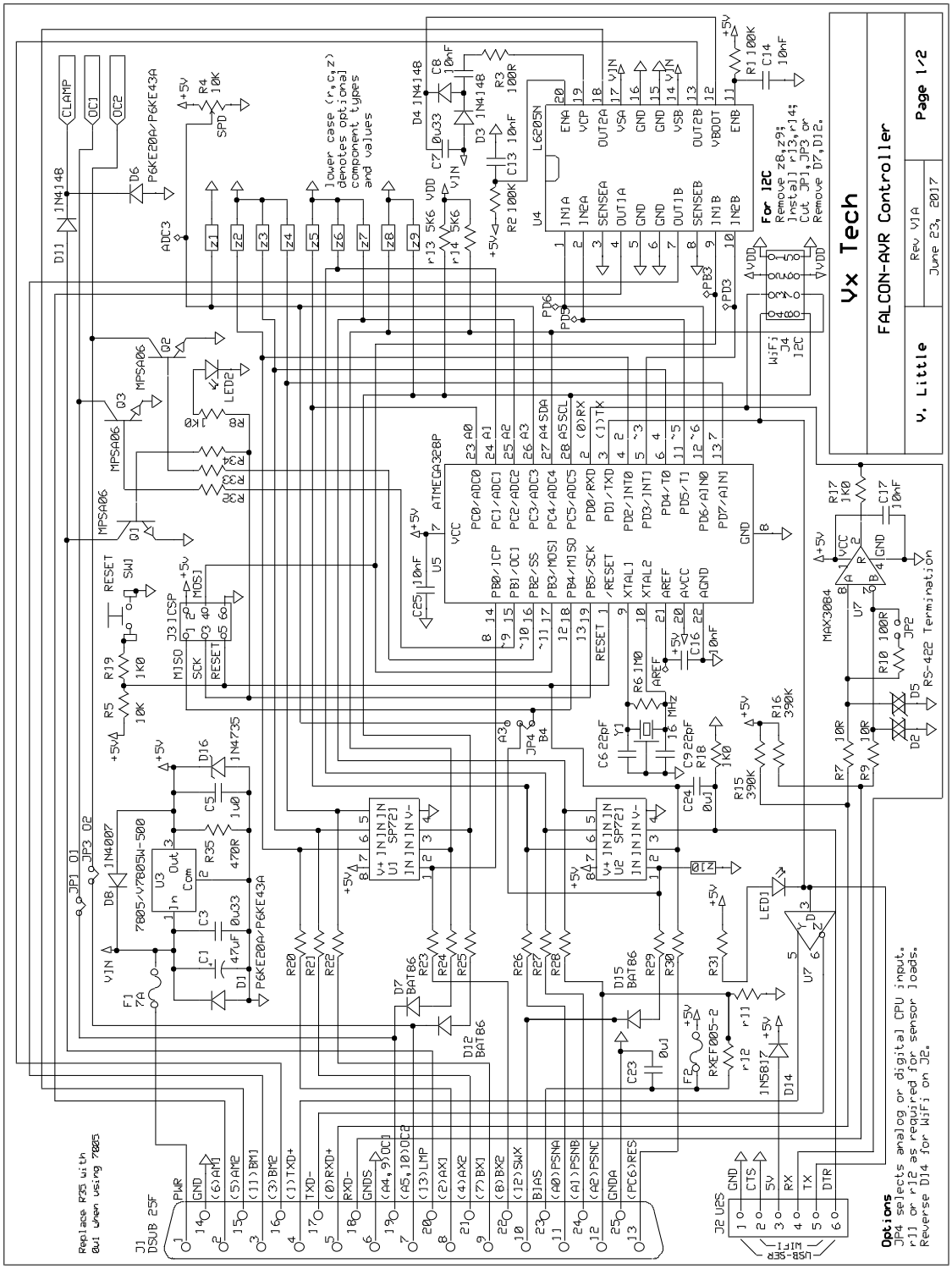
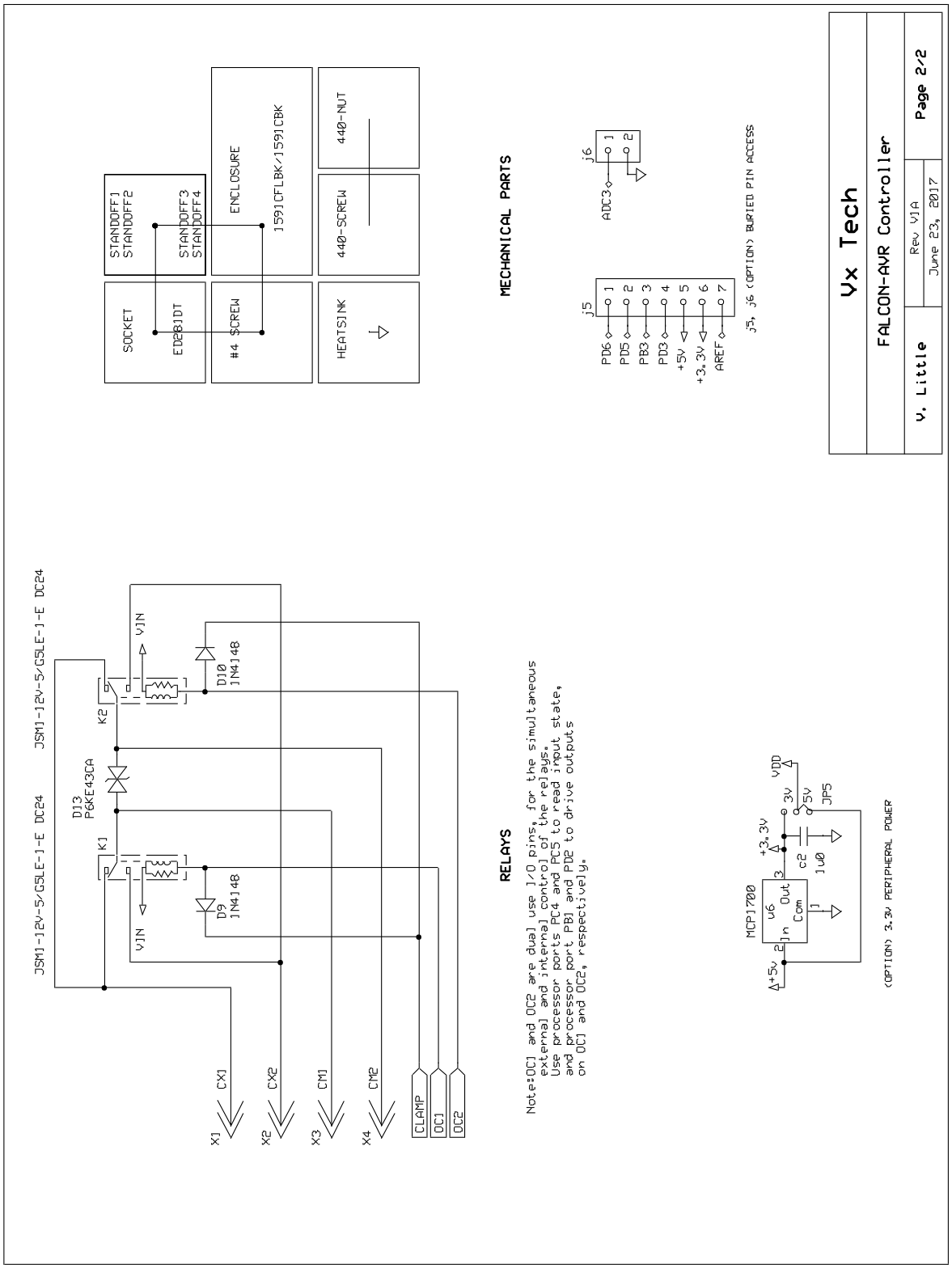


Figure 7. Falcon-AVR Pin Configuration Cross-Reference (2/2)

7. APPENDIX C. FALCON-AVR SCHEMATIC DIAGRAM





8. APPENDIX D. ENCLOSURE

The Falcon-AVR mounts directly into a standard Hammond 1591CFLBK or 1591CBK case. The end plates of the case need to be machined to allow access to the device connectors, as shown in Figure 7.

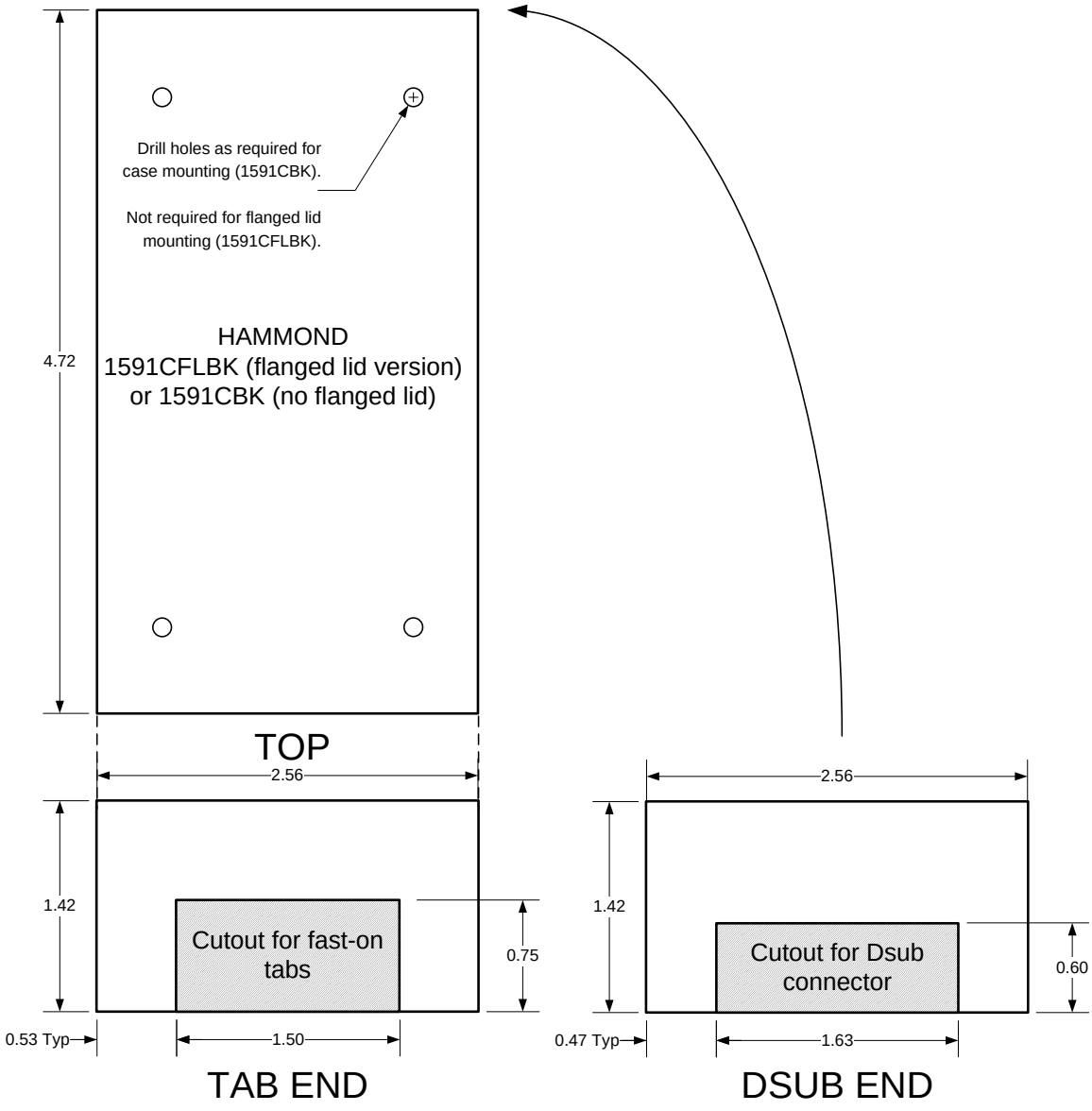


Figure 8. Case Machining Guide

9. DOCUMENT REVISION HISTORY

Issue Number	Date	Purpose
VXD-1703001V1	May 28, 2017	Product Release
VXD-1703001V1A	June 23, 2017	Updated schematic, circuit board and document to support 3.3 Volt peripheral power, optional signal access connectors and future CPU upgrades. Updated pin cross-reference diagrams and jumper descriptions.
VXD-1703001V1B	October 23, 2017	Updated Figure 8. Case Machining Guide.