

General Description

The Millswood Engineering Failsafe Device is designed to fulfil the failsafe requirements of the Australian UAV Outback Challenge. This Search and Rescue event requires competing aircraft to have an on-board failsafe device that activates if the autopilot or communication link fails for more than 5 seconds. If activated, the failsafe device must override the autopilot and set the throttle and control surfaces to predefined states.

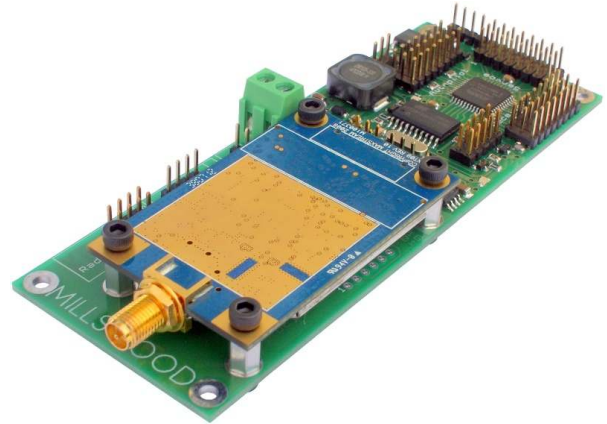
Although the Failsafe Device has been designed to meet the requirements of the UAV Outback Challenge, it can do a lot more than just put your UAV into a death spiral.

The Failsafe Device gathers together all of those annoying electronic bits and pieces that infest UAVs, and integrates them into a single unit. Things like RS232 level converters, radio modem carrier boards, receiver/autopilot multiplexers, battery eliminator circuits, serial servo controllers, and redundant power supply switches. All of these functions have been integrated onto a single PCB that plugs directly into a Digi XTend OEM RF Module.

The Failsafe PTZ is a variant that provides precision control of pan, tilt, zoom and trigger servos using the telemetry radio's uplink.

Features

- Provides mechanical mounting for the Digi XTend radio modem.
- Provides full-duplex RS232 access to both autopilot and radio modem.
- In conjunction with a ServoStation, provides a fully redundant power system for telemetry radio, autopilot, RC receiver and servos.
- High efficiency 2A switching power supply.
- Delayed power to the radio modem on power-up allows autopilot to initialise without RF interference.
- Allows RC control to override failsafe.
- Multiple heartbeat sources – can be a string in the telemetry data stream, or a logic level input (or both).
- Activation of failsafe can be reported on telemetry downlink with a user programmable message string.
- 8 channel serial servo controller built-in, with precision 12-bit hardware PWM generation.
- PWM channel switching performed in hardware by a programmable logic device, giving consistent low-latency signal path delays.
- Pololu and NMEA 0183 style text versions of Pololu commands supported natively. No need to set any jumpers or cycle the power.
- Hardware EEPROM write protection to prevent the Failsafe from responding inappropriately to routine telemetry data.



Theory of Operation

The Failsafe Device can be divided into 3 main sections: the PWM multiplexers (shown in black), the serial data processing logic (shown in blue), and the power system (shown in red).

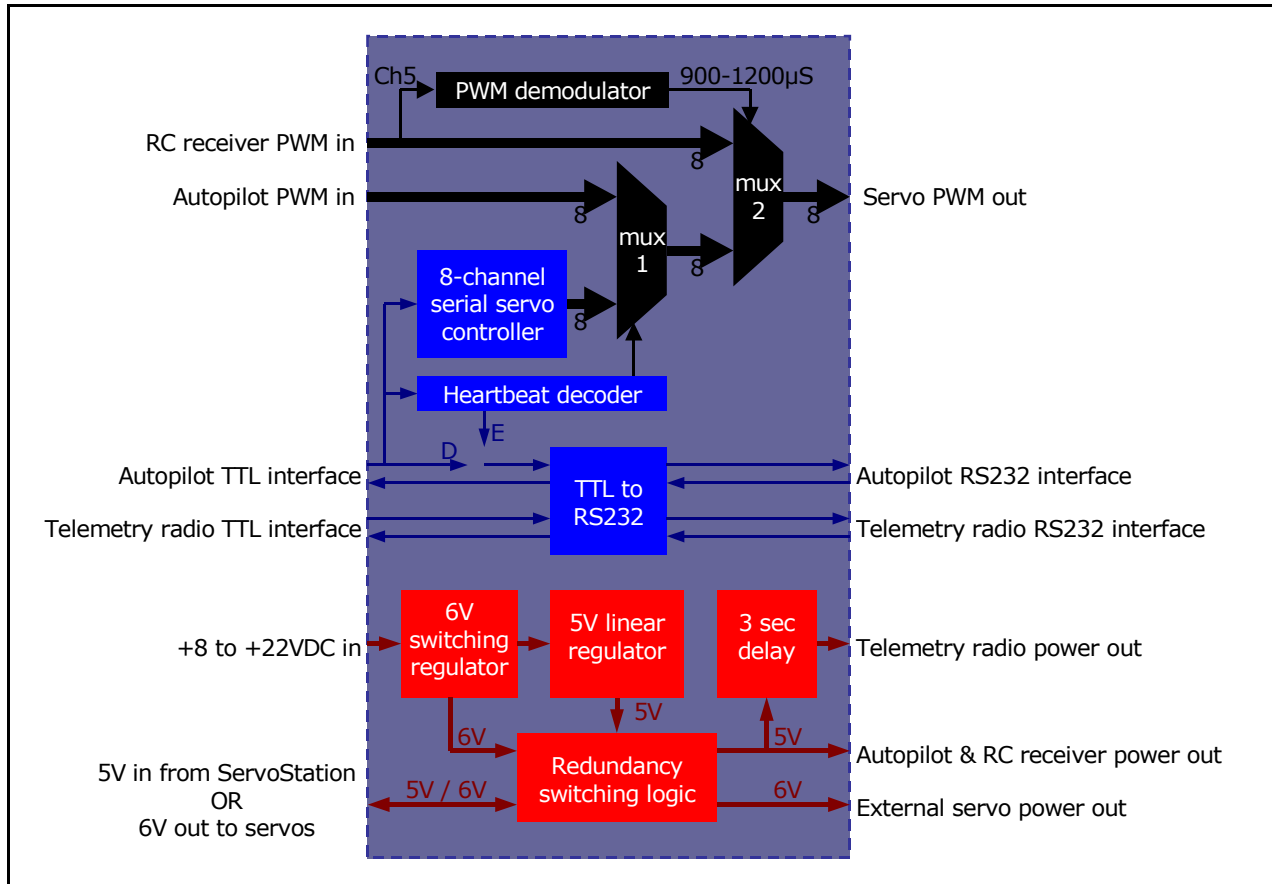


Fig. 2: Internal block diagram

PWM Multiplexers

Each multiplexer (mux) is 8 channels wide, and their combined job is fairly simple: to select one group of 8 channels from RC receiver, autopilot or serial servo controller.

Mux 2 is controlled by RC receiver channel 5; when this channel has a signal with a pulse width between 0.9 and 1.2ms, all 8 channels of the RC receiver are selected. This occurs regardless of the state of mux 1.

Mux 1 only comes into play if the RC receiver is not selected (i.e. channel 5's pulse width is not in the range 0.9 to 1.2ms). Mux 1 selects the autopilot unless a failsafe condition has occurred, in which case it selects the serial servo controller.

Note that the PTZ variant has 4 of the 8 servo outputs (pan, tilt, zoom and trigger) wired up so that they are driven directly by the serial servo controller. Thus these 4 channels are always under telemetry control regardless of the state of the muxes, the autopilot or the RC receiver.

Serial Data Processing Logic

The Serial Servo Controller (SSC) is a fairly simple device: it continuously generates 8 PWM channels under software control. The SSC stores most of its settings in non-volatile memory, and so for failsafe applications it can be programmed once and subsequently forgotten about.

Although the SSC is intended to be a set-and-forget device, it can be manipulated in real-time just like any other serial servo controller. It accepts Mini SSC II commands as well as most of the Pololu SSC command set. Naturally for the servos to respond to these commands the multiplexers must be in the right positions. This can be arranged by programming in a failsafe timeout period of zero and having a pulse width on RC channel 5 of anything other than 0.9 to 1.2ms (or indeed no signal at all).

The serial input for programming the SSC is the input pin of the autopilot interface, which is a TTL level signal. To program the SSC from a PC or laptop (i.e. at RS232 levels), fit a jumper across the "I" and "O" pins of the autopilot interface, and transmit data into the "I" pin of the RS232 AP interface. Data will be returned through the "O" pin of the RS232 AP interface (make sure that the status reporting jumper is fitted in the "Enable" position). If an autopilot is already connected, programming data must be able to pass transparently through it.

The heartbeat decoder monitors the serial data stream at the autopilot interface looking for particular sequences of bytes (the heartbeat strings). If neither of these are found within a certain period of time (the failsafe timeout period), a failsafe event is deemed to have occurred and mux 1 switches to the serial servo controller. Even though a failsafe event has occurred, the RC receiver can still be selected and the serial servo controller can still be manipulated. The Failsafe Device continues searching for the heartbeat strings, and if either is found autopilot control is restored.

The TTL to RS232 level converters are arranged to provide RS232 access to the telemetry radio and the autopilot for programming and debugging purposes. In flight it is anticipated that RS232 access will not be required, and so the RS232 ports should be looped together to complete the telemetry data link. Note that this arrangement assumes a telemetry data loop as shown in figure 3 below – i.e. data originates from a Ground Control Station (GCS), travels up the telemetry radio uplink, passes through the autopilot and then travels back down the telemetry radio downlink to the GCS.

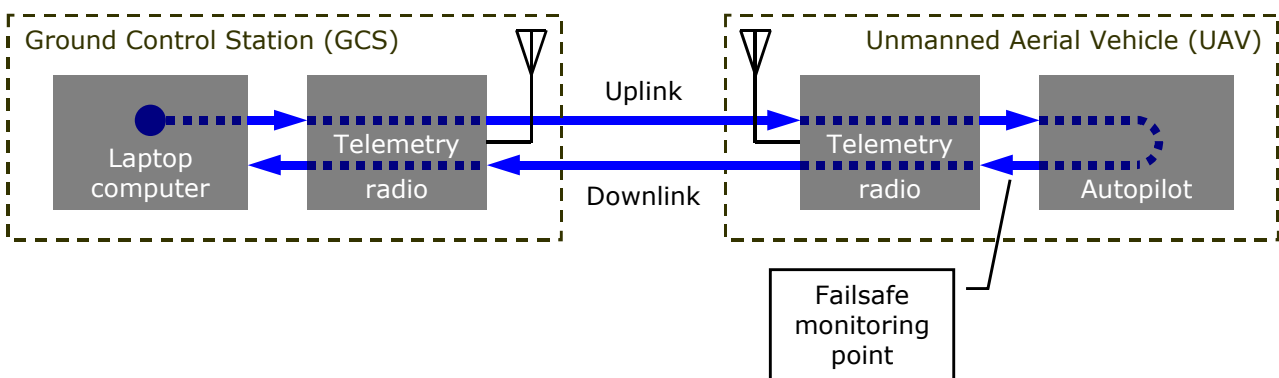


Fig. 3: Telemetry data flow in a typical UAV system

The heartbeat decoder monitors the telemetry data stream just prior to the radio downlink. If the GCS inserts a heartbeat string into the telemetry data on a regular basis, then the detection of this string on the downlink implies correct operation of both uplink and autopilot. Alternatively a heartbeat string can be inserted by the autopilot, in which case its continued presence implies that the autopilot is working (but says nothing about the uplink).

The Failsafe Device cannot monitor downlink integrity – this is the GCS's responsibility.

Power System

Although superficially straightforward, the power system is actually quite complex. Please take care to ensure that the power source jumpers are set correctly (as described in the Connectors and Jumpers section) or the system will not work as expected.

In a stand-alone power system the Failsafe Device supplies +6V to the servos, and +5V to the RC receiver, autopilot and telemetry radio (with power-up delay). The external servo power output is not used. A +3.3V rail is generated internally from the +5V rail for powering the onboard microprocessor and programmable logic device.

In a redundant power system things are more complex. Under normal operating conditions the ServoStation supplies +5V to the Failsafe Device via the female-female servo cables connecting these two devices together. This +5V is then fed to the RC receiver, autopilot and telemetry radio (with power-up delay), and is also used to generate the internal +3.3V rail. If the Failsafe Device's power source disappears for any reason, then nothing really changes as everything is powered from the ServoStation by default.

When the ServoStation's power fails, the external servo power output from the Failsafe maintains the ServoStation's +6V rail, thus powering the servos. The ServoStation's 5V regulator remains active, thus powering the internal PWM buffers.

A Few Things To Bear In Mind

- The GCS and autopilot should not use telemetry data that could be misinterpreted by the Failsafe Device as a command. Avoiding the values 128 (80 hex) and 255 (FF hex) will guarantee that this requirement is met. Attopilot systems meet this requirement because they use pure ASCII telemetry data (all values below 128). If you cannot be sure that your telemetry meets these requirements, then you should fit the EEPROM write protect jumper – this will prevent the Failsafe from responding inappropriately to configuration commands that may appear within the telemetry data stream. Write protection is enabled by fitting a jumper across pins 1 and 3 of the programming connector.
- Do not connect too many large servos directly to the Failsafe Device – its power supply is only rated to 2 Amps. Driving multiple servos simultaneously to their failsafe positions at maximum speed requires a lot of current. If too much current is drawn from the Failsafe Device then voltages will sag, and too much sag will cause the Failsafe Device to reset, aborting the failsafe event. If there is any concern, use a ServoStation or BEC to power the servos independently.
- UAVs are high-vibration environments. The PCB jumpers (also known as shunts) should be secured with tape or a blob of silicon goo, or for maximum security with a soldered wire strap.

Typical Applications

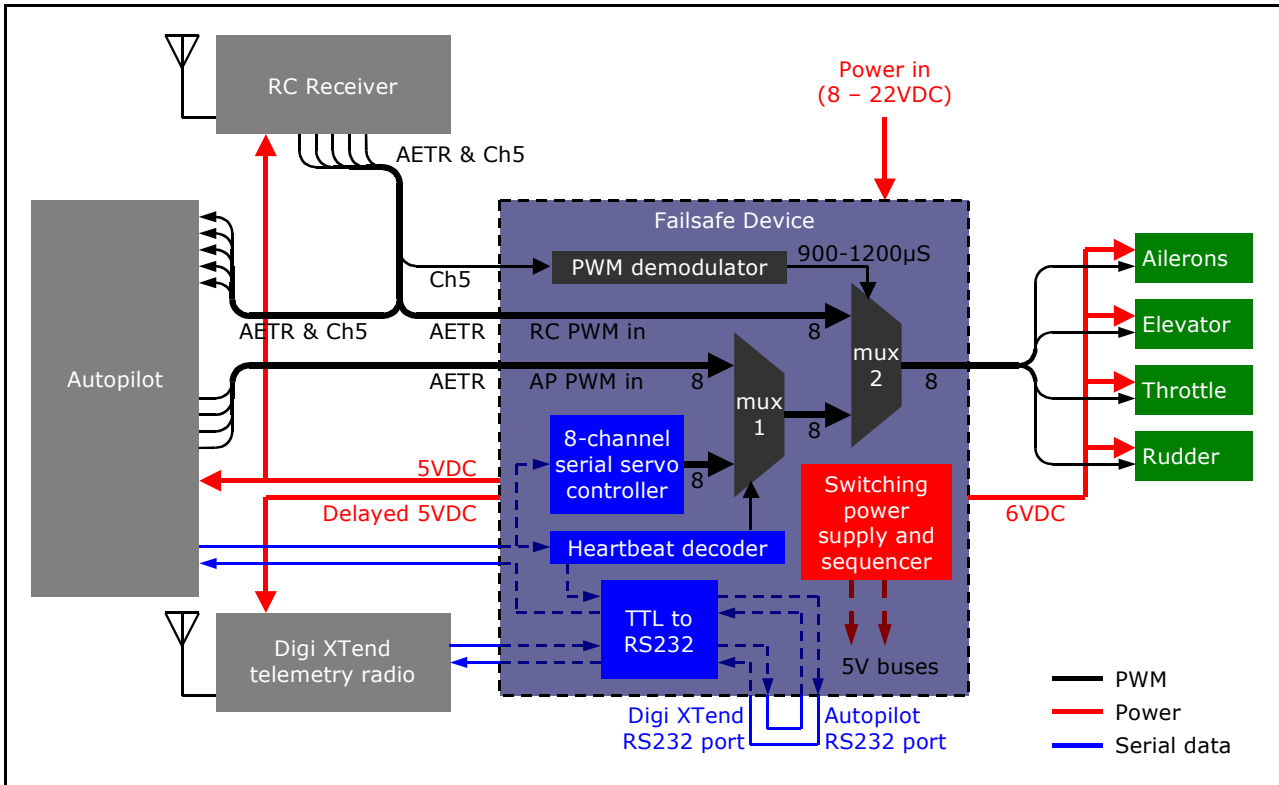


Fig. 4: A typical 4-channel setup showing RC receiver, autopilot, telemetry radio, failsafe and servos

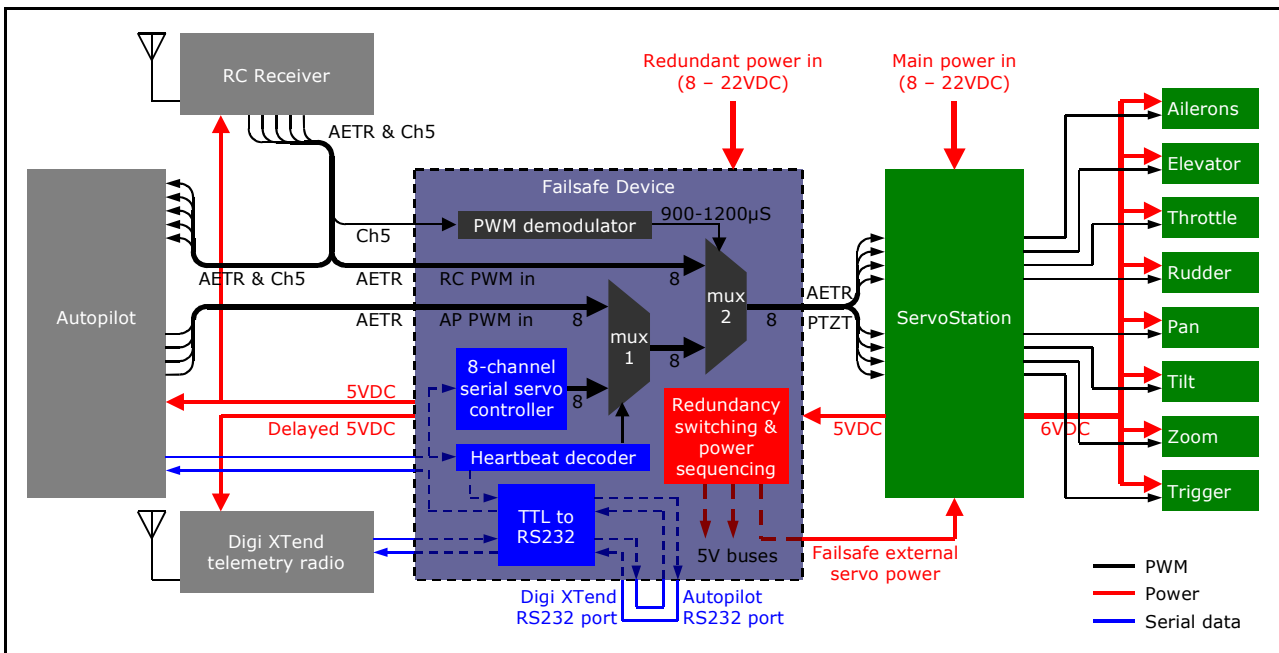


Fig. 5: A Failsafe PTZ configured for operation with redundant power supplies and a ServoStation

Connectors and Jumpers

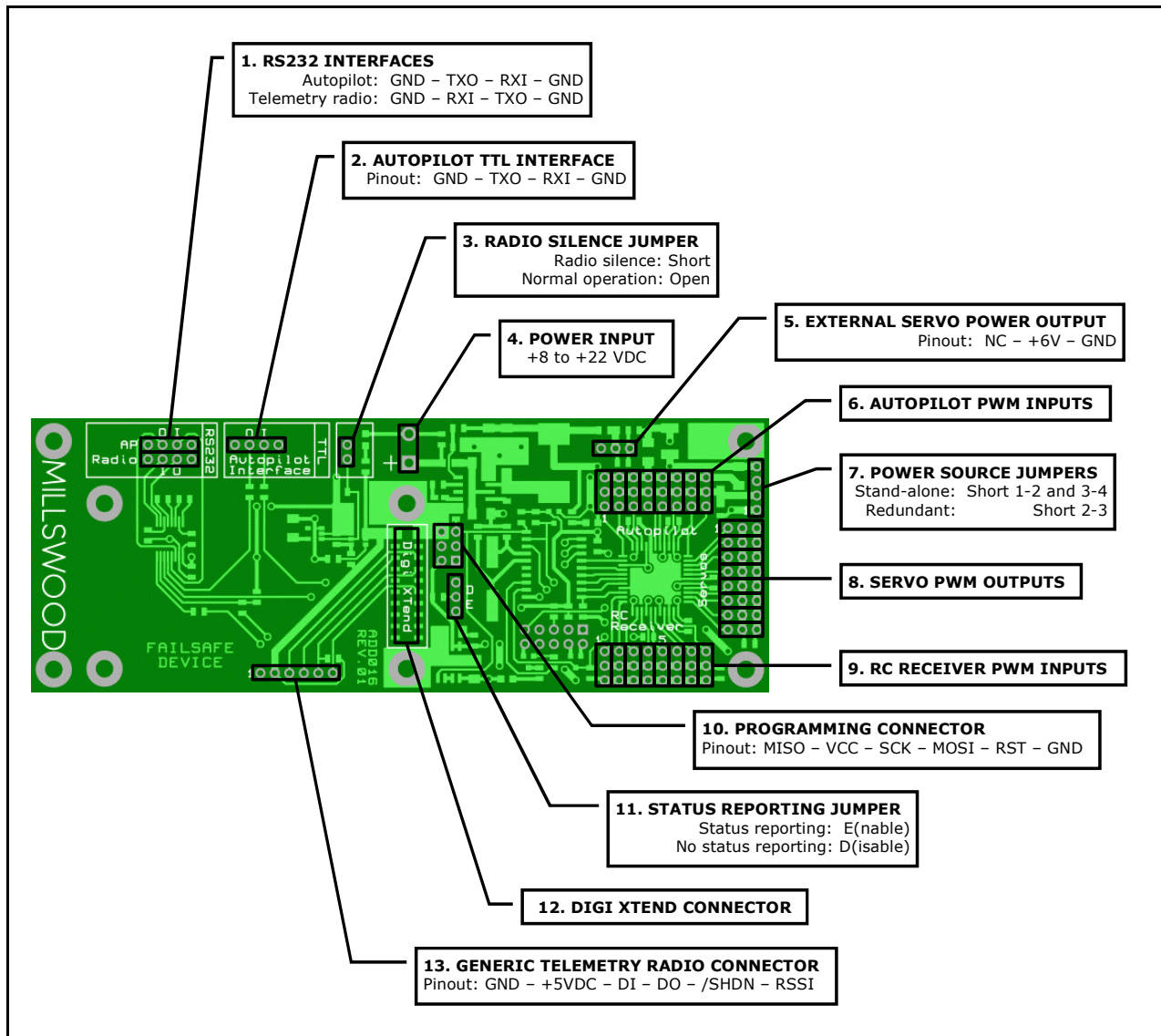
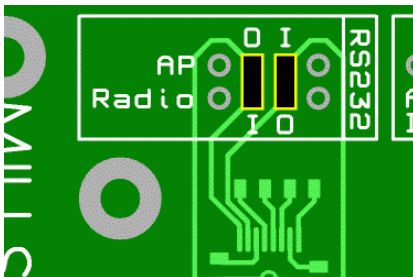


Fig. 6: Locations of connectors and jumpers

1. RS232 interfaces



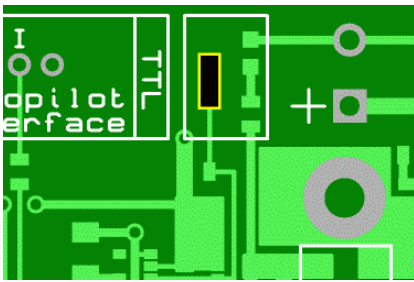
The Failsafe Device provides two RS232 interfaces to access the autopilot and telemetry radio. This is where a laptop or PC can be connected. These interfaces are not normally used during flight and must be linked together to complete the telemetry data link. The autopilot RS232 output "O" must be connected to the radio RS232 input "I" and vice versa. The connectors are arranged to make this easy.

Fig 7: Two jumpers fitted to the RS232 interfaces to complete the telemetry data link.

2. Autopilot TTL interface

This is where the autopilot TTL serial lines are connected. The input is 3V logic compatible and very tolerant of overvoltage and undervoltage. The output is 5V TTL.

3. Radio silence jumper



When fitted this jumper disables the telemetry radio. This is useful for testing in environments when radio silence must be observed. The autopilot, RC receiver, failsafe and servos remain powered-up and active. A green LED next to the jumper illuminates when the radio is powered up.

Fig. 8: Don't forget to remove this jumper before flying!

4. Power input

+8 to +22 VDC. Positive is marked on the PCB (ground is nearest the edge of the PCB). Reverse polarity protected (and thus protects all electronics powered by the Failsafe Device). A green LED next to the large inductor illuminates when power is connected.

5. External servo power output

A nominally +6VDC output which is only used in redundant systems. It provides power to the servos when the primary (ServoStation) power source fails. **It should be connected to one of the ServoStation's servo outputs.** A standard female-female servo cable may be used, but only +6V and ground are actually being provided by this connection. Be very careful to get the polarity correct. If connected incorrectly the system may seem to operate normally, but redundant operation will fail. Connector orientation is the same as the servo outputs.

6. Autopilot PWM inputs

Connect to the autopilot's servo outputs. A "1" on the PCB identifies channel 1. The ground pins are nearest the edge of the PCB.

7. Power source jumpers

These jumpers define whether the Failsafe Device will operate from a stand-alone or redundant power source. The two possible arrangements are:

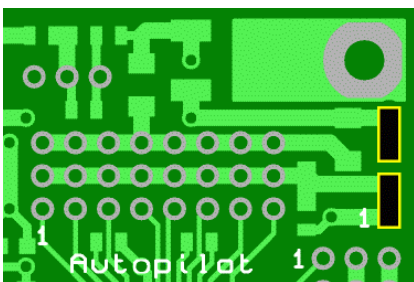


Fig. 9: For stand-alone operation short pin 1 to pin 2, and pin 3 to pin 4.

This connects 6 volts to the servos, and 5 volts to the autopilot, RC receiver and telemetry radio. Any BEC that might be feeding power back up towards the Failsafe must be disconnected or disabled, otherwise the arrangement shown in figure 10 should be used.

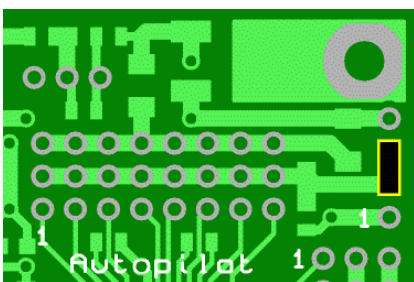


Fig. 10: For operation with a ServoStation or BEC, short pin 2 to pin 3.

With the jumper in this position, 5 volts must be applied to the servo power pins externally – either from a ServoStation or a BEC. With this arrangement there is no need to connect power to the Failsafe's main power connector, but we strongly recommend that you do.

For redundant power supply operation with a ServoStation, a cable must be connected from the Failsafe Device's external servo power output to one of the ServoStation's servo outputs. A primary source of power should be connected to the ServoStation, and a secondary source of power should be connected to the Failsafe Device.

When two power sources are connected, power will be drawn preferentially through the ServoStation. The autopilot, RC receiver, telemetry radio, failsafe and servos will remain operational when either power source fails. The transition is glitch-free and has no brownout phase. The system also tolerates either power supply going short-circuit.

8. Servo PWM outputs

In stand-alone systems this is where the servos connect. They must be rated for 6V operation. Ground pins are those nearest the edge of the PCB. In electrically powered aircraft this is also where the Electronic Speed Controller (ESC) plugs in (usually to channel 3). If the ESC has a Battery Eliminator Circuit (BEC) it must be disconnected or disabled, unless it is being used as part of a redundant system – see the Failsafe Integration guide for details. If using a ServoStation, these outputs connect to the ServoStation's receiver inputs and the power jumpers must be set for redundant operation.

9. RC receiver PWM inputs

Connect the RC receiver's servo outputs here. Channel 1 is marked on the PCB. Channel 5 is also marked because it is used by the Failsafe Device to control the second multiplexer. If the pulse width on channel 5 is between 0.9 and 1.2ms the system will switch to RC control. This overrides any failsafe activation. The ground pins are nearest the edge of the PCB.

10. Programming connector

Used to program the microprocessor, but also used to access the microprocessor for implementing additional functions. A hardware heartbeat input was added at firmware version 1.22, and EEPROM write protection was added at firmware version 1.23.

If the hardware heartbeat is enabled, any transition on pin 4 (the middle pin on the left) of the programming connector ("MOSI") will be interpreted as a valid heartbeat signal. There is an internal pull-up to +3.3V on this pin, allowing direct connection of open-collector output stages (such as found in many common optocouplers).

EEPROM write protection is enabled by fitting a jumper across pins 1 (MISO) and 3 (SCK) as shown in figure 11 below. Commands that do not result in EEPROM change (Set Position commands and heartbeat detection) will still occur normally. Write protecting the EEPROM is useful to prevent the Failsafe from inappropriately responding to Failsafe commands that might appear within routine telemetry data.

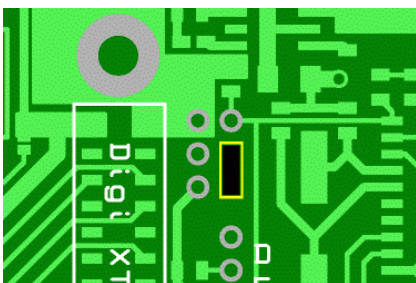


Fig. 11: To prevent any changes to the Failsafe's current programming, fit a jumper across pins 1 and 3 of the programming connector.

Ground (pin 6, top left) and +3.3V (pin 2, bottom left) are also present on the programming connector; these may be used to source a small amount of power to external devices.

The programming connector connects directly to the microprocessor, and so extreme care must be taken to avoid ESD and voltage excursions beyond those listed in the absolute maximum ratings section.

11. Status reporting jumper

This jumper determines whether status information can be added to the downstream telemetry data stream.

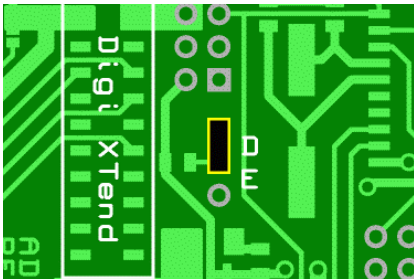


Fig. 12: To disable reporting of failsafe status, install a jumper in the "D" (disable) position.

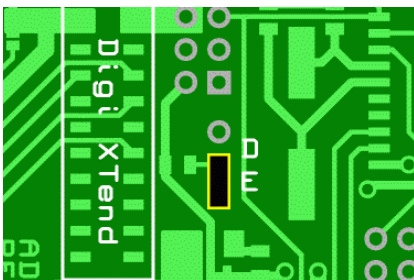


Fig. 13: To enable reporting of failsafe status on the telemetry downlink, install a jumper in the "E" (enable) position. The programming software requires a jumper installed here in order to read back data from the failsafe.

12. Digi XTend connector

Plug your Digi XTend in here. The Failsafe Device provides connections to pins 1 (GND), 2 (VCC), 5 (DI), 6 (DO), 7 (/SHDN) and 11 (RSSI). These connections are also brought out to the generic telemetry radio connector to facilitate:

- Direct access to the TTL serial uplink and downlink data streams.
- Control of the XTend's shutdown pin.
- RSSI monitoring.

The generic telemetry radio connector pinouts are given in section 12 below. There is a 3 second delay on the telemetry radio's 5 volt power supply whenever the 5 volt rail transitions upwards through 4.5 volts. This delay is to allow the autopilot to initialise without radio interference.

When the 5 volt rail is below 4.5 volts, power to the radio is switched off.

13. Generic telemetry radio connector

As well as providing direct access to the Digi XTend radio, this connector provides a place to connect an alternative telemetry radio. The pinouts are:

Pin number	Name	Digi XTend pin number	Comments
1	GND	1	Ground.
2	VCC	2	+5VDC output with a 3 second power-up delay.
3	DI	5	If using an external (non Digi) radio modem, connect to the radio's TTL serial data input.
4	DO	6	If using an external (non Digi) radio modem, connect to the radio's TTL serial data output.
5	/SHDN	7	Includes a 10kΩ pull-up to VCC. Makes no other connection inside the Failsafe.
6	RSSI	11	Makes no other connection inside the Failsafe.

Absolute Maximum Ratings^{Note 1}

Symbol	Parameter	Min	Max	Unit
V _{BI}	Battery input voltage	-25	+25	V
V _{SI} , V _{SO}	PWM signal input & output voltages	-2	+5.75	V
V _{PCI}	Programming connector input voltage	-0.5	+3.6	V
V _{RS232_I}	RS232 input voltage	-25	+25	V
V _{RS232_O}	RS232 output voltage	-13.2	+13.2	V
V _{ESD}	ESD rating as per Mil-Std-883C, method 3015, using the human body model ^{Note 2}	2		kV
V _{RS232_ESD}	ESD rating as per Mil-Std-883C, method 3015, using the human body model ^{Note 2}	15		kV
T _{stg}	Storage temperature range	-40	+85	°C

Note 1: Absolute maximum ratings are those values beyond which damage to the product may occur. Functional operation under these conditions is not implied (or recommended).

Note 2: The human body model is a 100pF capacitor discharged through a 1.5kΩ resistor into each pin.

Recommended Operating Conditions

Symbol	Parameter	Min	Max	Unit
V _{BI}	Battery input voltage	+8	+22	V
T _{op}	Operating temperature range	-40	+85	°C
V _{air}	Air flow velocity	1.5		ms ⁻¹

Electrical Characteristics

Test conditions are $+8 < V_{BI} < +22V$, $-40 < T_{op} < +85^{\circ}C$, $V_{air} = 1.5ms^{-1}$ unless stated otherwise.

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
I_Q	Quiescent current	$T_{op} = +25^{\circ}C$		TBD	TBD	mA
Servo power output (stand-alone operation)						
V_{SPO}	Voltage		5.8	6.0	6.2	V
I_{SPOT}	Total current capability ^{Note 1}		2.0			A
η	Switching power supply efficiency	$1A < I_{SPOT} < 2A$, $T_{op} = +25^{\circ}C$	TBD	TBD		%
f	Switching frequency		315	370	435	kHz
RC receiver, autopilot and telemetry radio power outputs (stand-alone operation)						
V_{RPO}	Voltage		4.8	5.0	5.2	V
I_{RPO}	Total current capability		1.0			A
I_{RPOSC}	Short circuit current			1.5		A
Telemetry radio power output						
t_p	Power-up delay		TBD	3	TBD	S
RC receiver and autopilot PWM signal inputs						
I_{IL}	Input leakage current				± 10	μA
V_{IL}	Input logic low		-0.5		0.8	V
V_{IH}	Input logic high		1.7		5.75	V
Servo PWM signal outputs						
V_{OL}	Output logic low	Sink current=100 μA Sink current=8mA			0.2 0.4	V
V_{OH}	Output logic high	Source current=100 μA Source current=8mA	3.1 2.7			V
Autopilot TTL serial interface						
V_{AP_IL}	Input logic low		-10		1.0	V
V_{AP_IH}	Input logic high		2.0		13.2	V
V_{RAP_OL}	Output logic low	Sink current=1.6mA			0.8	V
V_{RAP_OH}	Output logic high	Source current=1.0mA	4.1	4.6		V
Autopilot and telemetry radio RS232 serial interfaces						
V_{RS232_IL}	Input logic low	$T_{op} = +25^{\circ}C$	-25		0.8	V
V_{RS232_IH}	Input logic high	$T_{op} = +25^{\circ}C$	2.4		25	V
R_{RS232_I}	Input resistance	$T_{op} = +25^{\circ}C$	3	5	7	k Ω
V_{RS232_OS}	Output voltage swing	$R_L = 3k\Omega$ to ground	± 5.0			V
$V_{RS232_OS_CC}$	Output short circuit current			± 35	± 60	mA
Programming connector						
V_{PC_IL}	Input logic low		-0.5		0.9	V
V_{PC_IH}	Input logic high		2.1		3.6	V

Note 1: Current drawn by RC receiver, autopilot and telemetry radio subtracts from this value.

Commands

In addition to the hardware jumpers, there are a number of software parameters that can be set by sending commands to the Failsafe Device. Commands are sent serially, and may be sent at TTL levels (via the "I" pin in the TTL autopilot interface), or at RS232 levels (via the "I" pin in the RS232 AP interface). Commands can also be sent to the Failsafe Device after it is installed in a UAV via the telemetry link.

Note: If the autopilot is not connected when sending commands via the RS232 interface, a jumper shorting the "I" and "O" pins of the TTL autopilot interface should be fitted.

Configuration commands are stored in non-volatile memory as soon as they have been processed, and so there is no need to resend them following a power-down.

In order to control the servos when a failsafe condition arises, the Failsafe Device implements a precision 8-channel serial servo controller. It is quite possible to control all 8 servos in real-time via the telemetry link or even directly via the TTL or RS232 interfaces. The Failsafe command set is based on Pololu's serial servo controllers.

Commands may be sent as bytes or NMEA 0183 format text strings. Using bytes is more bandwidth efficient, but many autopilots will only process and echo printable ASCII characters, and so sometimes it is necessary to use the text commands in order to get them through the autopilot.

Command summary

Command	String
Heartbeat	\$E*FF
Set Position (Scott Edwards)	\$E,FF,00-7F,00-FE*FF
Read EEPROM address ^{Note 4}	\$E,80,00,00-7F*FF
Set Speed (Pololu)	\$E,80,01,01,00-7F,00-7F*FF
Set Position Absolute (Pololu)	\$E,80,01,04,00-7F,00-7F,00-7F*FF
Set Failsafe Position (Pololu)	\$E,80,01,05,00-7F,00-7F,00-7F*FF
Change Servo Numbers (Pololu)	\$E,80,02,00-0F*FF
Set Baudrate	\$E,80,03,00-07*FF
Set Failsafe Timeout Period	\$E,80,04,00-7F*FF
Set Failsafe Heartbeat String	\$E,80,05,STRING*FF
Set Fail Status String	\$E,80,06,STRING*FF
Configure Programming Connector ^{Note 4}	\$E,80,07,BITFIELD*FF

*Note 1: The trailing *FF is a wildcard substitute for the checksum to facilitate hand entry of commands. *FF will always be accepted. To calculate the checksum, XOR the ASCII values of all bytes between (but not including) the \$ and *. Include the commas. If present, checksums are used to validate commands and a checksum mismatch will result in the command being rejected.*

Note 2: The Failsafe command processor is not case sensitive (although the programmable heartbeat string matching routine is).

*Note 3: If sending byte commands, drop the leading \$E and the trailing *FF and send the remaining data as bytes without any spaces or commas. Terminate strings with a zero byte.*

Note 4: Command added at firmware version 1.22.

Commands in detail

Heartbeat is a fixed string that the Failsafe will always interpret as a heartbeat. It is simply **\$E*FF**. It is not case sensitive, and may include a checksum if desired (**\$E*45** or **\$e*65**).

Set Position (Scott Edwards) provides a fast but imprecise way of commanding a servo to move to a given position. The command consists of \$E,FF, followed by the servo number (00 to 7F), followed by the desired servo position (00 to FE). A position of 7F will centre the servo.

For example, to command servo 3 (physical servo 4 – the rudder) to move to approximately 25% (1.25ms), send: **\$E,FF,03,3F*FF**

Any servo group numbers and servo speeds previously programmed into the Failsafe Device will be observed. If the servo number is invalid the command will be ignored.

Read EEPROM address reads a single byte from the specified EEPROM address. The command consists of \$E,80,00, followed by the EEPROM address (00 to 7F).

For example, to read the data stored in the EEPROM at address 0x1A, send: **\$E,80,00,1A*FF**

As the serial link may be echoing other unrelated data when the response is issued, the returning byte is preceded by a backslash. Please note that the status reporting jumper must be set to "Enable" for this command to work.

Set Speed (Pololu) defines the rate at which each servo will move. This is useful for slowing down servos, especially those used in camera pan and tilt applications. Landing gear servos are another good application. The command consists of \$E,80,01,01, followed by the servo number (00 to 7F), followed by the desired speed (00 to 7F).

A speed of 00 is a special case and sets the speed for that servo to maximum. This is the default for all servos. A value of 01 sets the slowest speed possible, 3.6 degrees/second. At this speed most servos will take about 25 seconds to traverse 90 degrees.

For example, to set servo 0 (physical servo 1 – the ailerons) to move at 18 degrees/second, send: **\$E,80,01,01,00,05*FF**

Any servo group numbers previously programmed into the Failsafe Device will be observed. If the servo number is invalid the command will be ignored.

Set Position Absolute (Pololu) performs the same function as the Scott Edwards command of the same name, but does so with greater precision. The penalty is that this command is longer (and therefore takes longer to send), and more difficult to use. The command consists of \$E,80,01,04, followed by the servo number (00 to 7F), followed by the desired servo position expressed as most and least significant bytes (00 to 7F, 00 to 7F).

The position is an integer whose value ranges from 750 to 3000 (corresponding to pulse widths from 0.6 to 2.4ms).

To derive the most and least significant bytes, divide the position by 128. The whole part of the result is the most significant byte, and the remainder is the least significant byte. For example, to set a servo to position 2188 (1.75ms):

$$2188 / 128 = 17.0938$$

The most significant byte is 17. Converting this to hex gives 11.

The remainder is $2188 - (17 \times 128) = 12$. Converting this to hex gives 0C.

So, to set servo 1 (physical servo 2 – the elevator) to position 2188, send: **\$E,80,01,04,01,11,0C*FF**

For reference, here are a few pulse widths and their corresponding most and least significant data bytes:

Pulse width	Position	Data bytes
0.6 ms	750	05,6E
0.8 ms	1000	07,68
1.0 ms (min.)	1250	09,62
1.2 ms	1500	0B,5C
1.4 ms	1750	0D,56
1.5 ms (centre)	1875	0E,53
1.6 ms	2000	0F,50
1.8 ms	2250	11,4A
2.0 ms (max.)	2500	13,44
2.2 ms	2750	15,3E
2.4 ms	3000	17,38

In order to avoid damaging servos by taking them beyond their natural limits it is wise to keep pulse widths between 1 and 2ms, at least initially.

Any servo group numbers and servo speeds previously programmed into the Failsafe Device will be observed. If the servo number is invalid the command will be ignored. If the position requested lies outside the range 750 – 3000, then it will be limited to one of these extremes.

Set Failsafe Position (Pololu) sets the position to which the servos will move when a failsafe condition occurs. The format of this command is identical to the Set Position Absolute command. The only difference is that the third byte is 05 rather than 04.

Thus to set the failsafe position of servo 2 (physical servo 3 – the throttle) to its minimum value, send: **\$E,80,01,05,02,00,00*FF**

Although 0 is not a legal value for servo position, the Failsafe Device will accept the command and program in its minimum value of 750 (corresponding to a pulse width of 0.6ms). Thus when a failsafe condition occurs, the motor will stop. (Electronic speed controllers only; if you do this to a servo it might break.)

Any servo group numbers programmed into the Failsafe Device will be observed, however the speeds may not be. If the servos are moving to their failsafe positions as a result of a failsafe event, they will do so at maximal rates.

If the servo number is invalid the command will be ignored. If the position requested lies outside the range 750 – 3000, then it will be limited to one of these extremes. Executing this command when the Failsafe Device is already controlling the servos (i.e. when a failsafe condition already exists) will set the relevant servo to its new failsafe position whilst observing any speed restriction.

Change Servo Numbers (Pololu) allows the Failsafe Device to respond to a different range of servo numbers. This is only useful if there are other Pololu-compatible devices in the system. The command consists of \$E,80,02, followed by a number from 00 to 0F which is the servo group number.

The default servo group number is 00, which makes the Failsafe respond to servo numbers 0 through to 7. To make the Failsafe respond to servo numbers 8 through to 15, program in a servo group number of 01. The command to do this is: **\$E,80,02,01*FF**

The maximum servo group number is 0F, which corresponds to servo numbers 78 to 7F. Requests for servo group numbers greater than 0F are ignored.

Set Baudrate does exactly that. The command consists of \$E,80,03, followed by a number from 00 to 07 which sets the new baudrate according to the following table:

Baudrate	Data byte
2400 baud	00
9600 baud (default)	01
14.4 kbaud	02
19.2 kbaud	03
28.8 kbaud	04
38.4 kbaud	05
57.6 kbaud	06
115.2 kbaud	07

The new baudrate does not take effect until the Failsafe is powered down and then back up again.

For example, to set the baudrate to 38.4kbaud, send **\$E,80,03,05*FF** and then cycle the power.

The default baudrate is 9600. If the third byte is greater than 07 the command is ignored.

Set Failsafe Timeout Period sets how long the Failsafe Device will wait for a heartbeat string before overriding the autopilot and asserting the failsafe servo positions. The command consists of \$E,80,04, followed by the timeout period in 0.5 second increments (00-7F).

For example, to set a timeout period of 5 seconds, send: **\$E,80,04,0A*FF**

A timeout value of 7F is a special case, and disables the timeout function completely. With the timeout value set to 7F, mux1 will always select the autopilot.

A timeout value of 00 results in an immediate timeout. This is useful for accessing the serial servo controller.

Sending the Set Failsafe Timeout Period command restarts the timeout timer.

Set Failsafe Heartbeat String sets the programmable string that the Failsafe Device will search for in the received telemetry data stream. If the failsafe timeout period elapses without either heartbeat string being found, then the autopilot will be overridden and the failsafe servo positions asserted. The command consists of \$E,80,05, followed by the desired heartbeat string. For reliable operation a minimum of 3 characters is recommended.

For example, to set the heartbeat string to ATTO, send: **\$E,80,05,ATTO*FF**

If using byte commands the string must be terminated with a zero byte; if using text commands the asterisk marks the end of the string (i.e. strings may not contain asterisks within them). The maximum string length is ten characters. Do not program in heartbeat strings that begin with \$; they will never be found. Please note that string matching for the programmable heartbeat string is case-sensitive (but string matching for the built-in heartbeat \$E*FF is not).

Set Fail Status String sets the string that the Failsafe Device will insert into the downlink telemetry data stream when a failsafe event occurs. The fail status string will be resent every timeout period whilst a failsafe condition exists. The command consists of \$E,80,06, followed by the fail status string.

For example, to set the fail status string to \$FAIL, send: **\$E,80,06,\$FAIL*FF**

If using byte commands the string must be terminated with a zero byte; if using text commands the asterisk marks the end of the string (i.e. strings may not contain asterisks

within them). The maximum string length is ten characters. Please note that the status reporting jumper must be set to "Enable" for status reporting to work.

Configure Programming Connector enables and disables additional physical I/O and any associated functionality. Firmware version 1.22 introduced a programmable hardware heartbeat input on pin 4. When enabled, a high-to-low or low-to-high transition on this pin constitutes a valid heartbeat signal and restarts the failsafe timer.

Note that other heartbeat signals may also be enabled and present concurrently, and that for the failsafe timer to expire and the failsafe activate, all heartbeats must cease for one whole failsafe timeout period.

To enable the hardware heartbeat input, send: **\$E,80,07,20*FF**

To disable the hardware heartbeat input, send: **\$E,80,07,00*FF**

Note that disabling the hardware heartbeat input does not disable the failsafe timer. In fact, with systems that use the hardware heartbeat alone, disabling this pin will cause failsafe activation (if timeouts are enabled, and after the timeout period). This provides a useful way to test flight termination, as it can be done without accessing the hardware.

Firmware version 1.23 adds EEPROM write protection on pins 1 and 3, but this is not a software configurable feature. The presence of a jumper across these pins is all that is required to prevent the EEPROM from being modified.

Mechanical Data

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
L	Length			129		mm
W	Width			48		mm
H	Height			19		mm
M	Mass			TBD		g

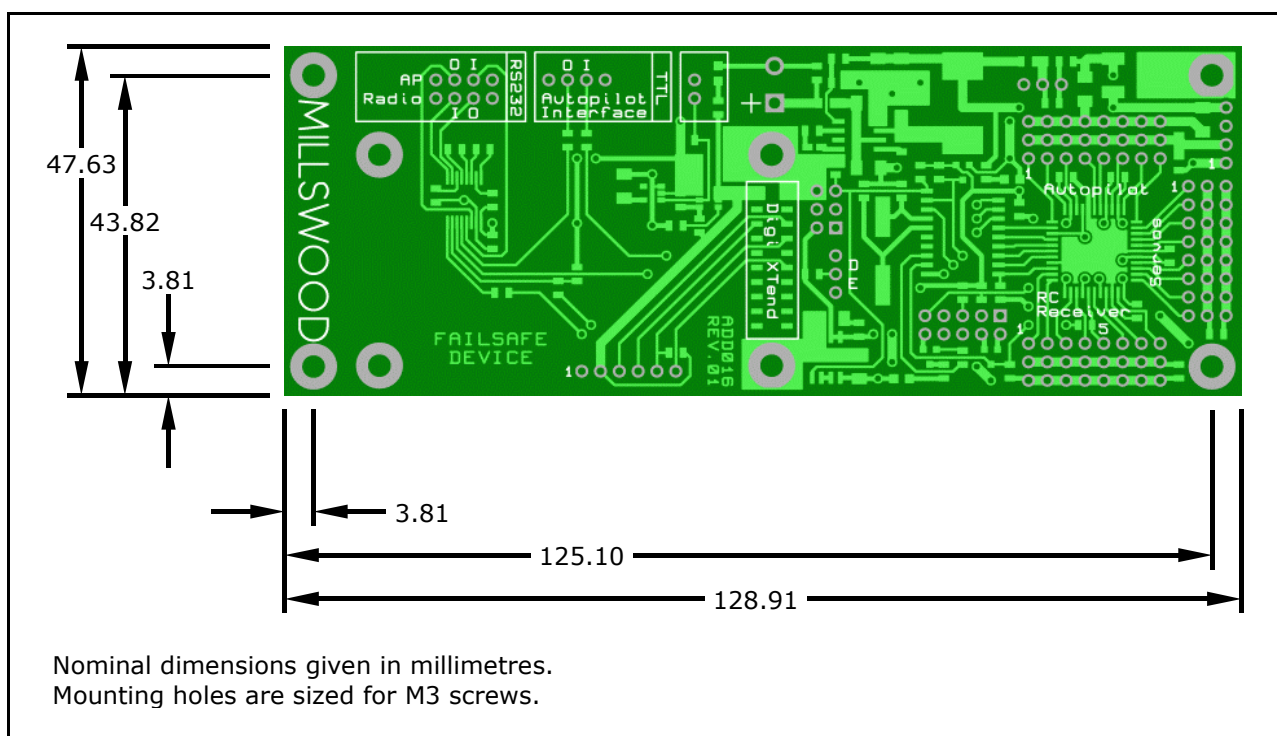


Fig. 14: Physical dimensions

Further Information

Visit us on the web at www.millswoodeng.com.au

Didn't find what you wanted? Send us an email or give us a call – contact details are on our website.



The Fine Print

Regarding this document: Millswood Engineering makes no warranty, representation or guarantee regarding the accuracy or completeness of this document and reserves the right to make changes to specifications and product descriptions at any time without notice.

Regarding this product: Millswood Engineering makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Millswood Engineering assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Millswood Engineering reserves the right to make changes without further notice to any products herein to improve reliability, function or design.

Regarding typical specifications: "Typical" parameters which may be provided in Millswood Engineering datasheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts.

Regarding life support applications: Millswood Engineering products are not designed, intended, or authorised for use as components in systems intended to support or sustain life, or for any other application in which the failure of the Millswood Engineering product could create a situation where personal injury or death may occur.

Regarding intellectual property: No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document.

Copyright Millswood Engineering July 2010. All rights reserved.