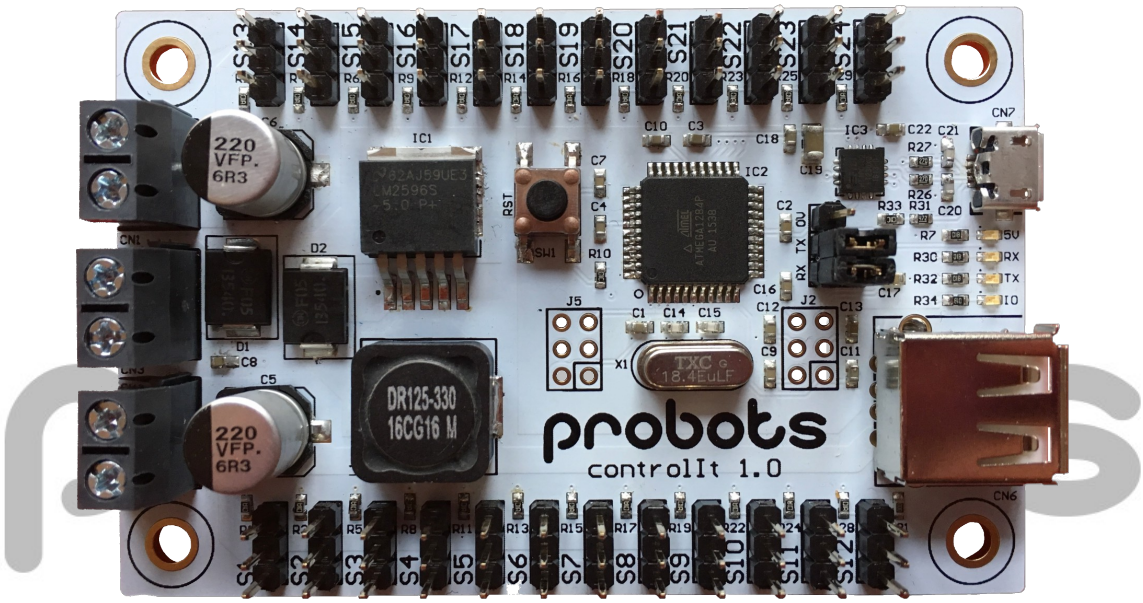
probots



controlIt 1.0 User Guide

Version 1.0 01/11/2016

Contents

Version 1.0 (01/11/16) – Initial release.

Introduction

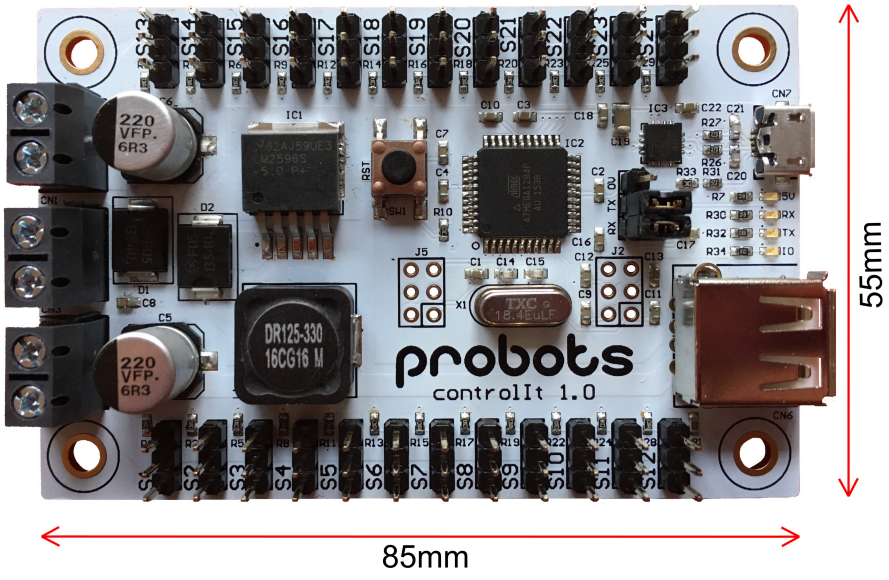
The controlIt device comprises of 24 PWM outputs and a 5V switched-mode power supply (SMPSU) for servo control and external power respectively. Designed to simplify the addition of electromechanical control to a robotic application, it features:

1) Up to 24 PWM servo outputs.
2) Command-based PWM position and duration of movement per channel.
3) 5V regulated power @ up to 3A via USB and screw terminal for external hardware.
4) LED power, serial activity and IO indicators.
5) USB or TTL serial interface with loop-back option.
6) Separate power terminals for logic / servos.
7) Up to 6 spare GPIO pins.
8) Logic reset switch.

Thank you for using controlIt!

Dimensions

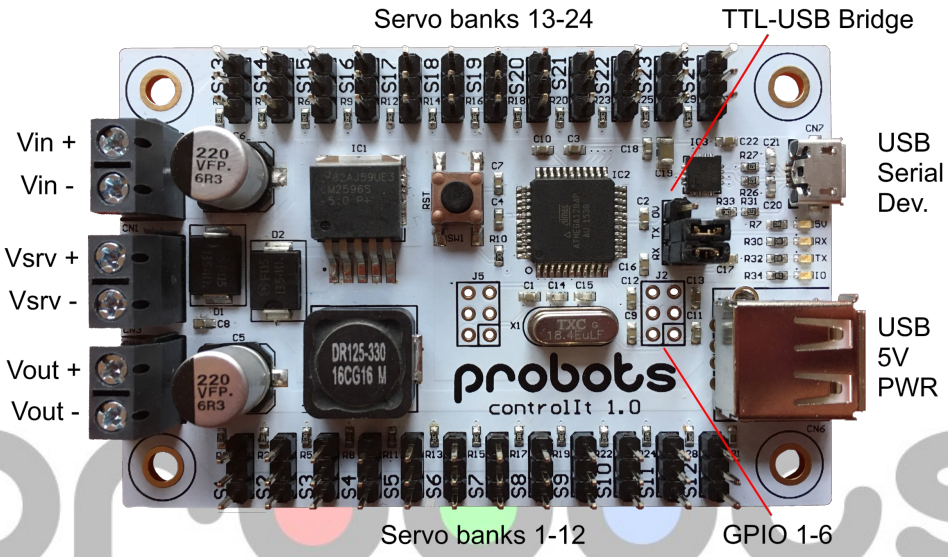
The overall dimensions for controlIt are 85mm x 55mm x 14mm.



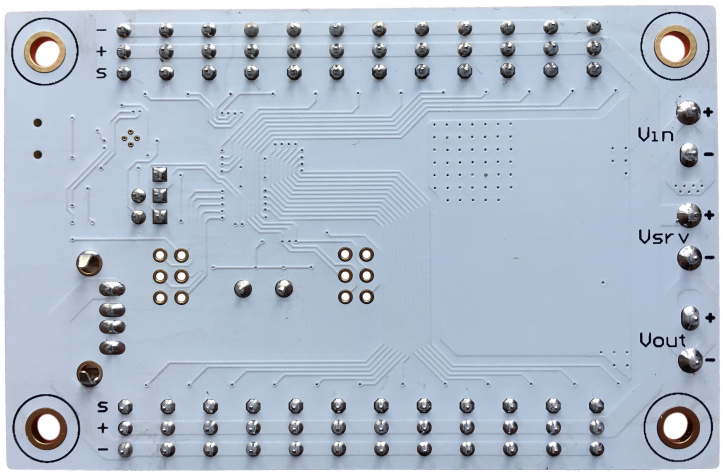
Wiring

The controlIt device both powers a controlling device and receives commands from it. Power to the 3A SMPSU is provided via 'Vin+' and 'Vin-' and can be between 5V and 40V (voltages greater than 5V recommended for best efficiency).

Servo power is supplied to the 'Vsrv+' and 'Vsrv-' terminals and must match the rated voltage of the servo in use (often 4.8V-6V). 'Vsrv+' can be wired to 'Vin+' if the servo voltage and the logic voltage match, else damage may occur to the servos. Note that the power supply for 'Vsrv' must be able to withstand the combined stall currents of all connected servos.

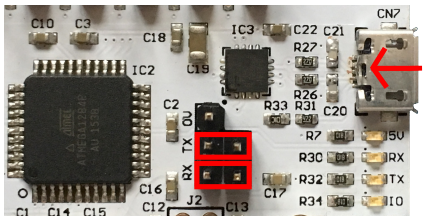


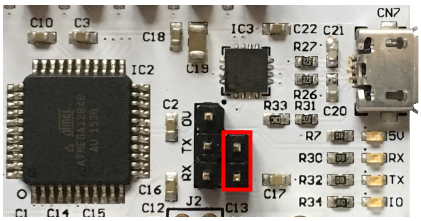
Servos are connected to each of the servo outputs using standard JST or Futaba connections. The servo pinout is identified on the rear of the PCB where GND (black/brown) is connected to the pin near the edge of the board, PWR (red) is the center pin and SIG (yellow/white) is the innermost pin.



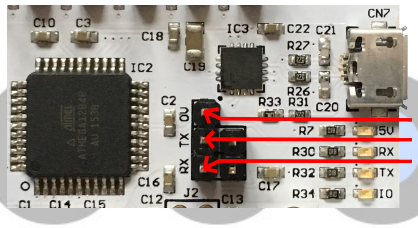
Power for the controlling device or accessory is rated up to 3A at 5V, provided at terminal block 'Vout' for wired devices, or via a Type A USB socket for small computers.

If the controlling device is to communicate via USB, it must be connected via a microUSB cable to controlIt. The onboard FTDI USB Serial transceiver should be automatically recognised as a COM port on most systems. In order to communicate with the servo controller however, jumpers must horizontally bridge the RX and TX pins as shown in the image. USB Serial loopback testing can be performed by vertically shorting the RX and TX pins close to the microUSB port.





If the controlling device is to communicate via TTL (local UART), then the jumpers should be removed from the board to disconnect the transceiver, and wire connections made to the single column of RX, TX and reference 0V pins.



GPIO pins 1-6 on J2 are directly wired to the control IC and are tolerant of voltages of up to 5V only. They must be configured as input or output before use to protect the IC. Pin 1 (lower right) is identified by a surrounding line, with pin 2 directly to the left of it.

GPIO Pin 6 is also connected to the 'IO' LED and can be convenient for communications testing or status indication.

Programming

All programming is done over a serial interface. Over a USB connection the controlIt device acts as a virtual COM port on the host computer, otherwise a direct wired connection to a host microcontroller allows for a standard UART interface. Baud rate is set at 38400, no parity, 8 data bits.

There are 5 commands available to the controlling device. All commands contain a Cyclic-Redundancy Check byte (CRC) to improve immunity over long wire lengths and prevent erratic behaviour.

Transmit format
<'#'><CMD><LEN><DATA><CRC>

<'#'>	Start of transmit command
<CMD>	Command number
<LEN>	Length of data bytes to send
<DATA>	Data bytes to send (if LEN > 0)
<CRC>	Generated CRC

Receive format
<'$'><CMD><LEN><DATA><CRC>

<'$'>	Start of receive command
<CMD>	Command number
<LEN>	Length of data bytes to send
<DATA>	Data bytes to send (if LEN > 0)
<CRC>	Check CRC

The CRC is generated with an XOR of all bytes except the CRC itself. The method below is used to generate and validate outgoing and incoming packets respectively:

Transmitted “Set Servo” Bytes:
<'#'><0x01><0x05><0x01><0x05><0xDC><0x00><0x00><0xFF>

CRC ^= '#'; (CRC = 0x23)

CRC ^= 0x01; (CRC = 0x22)

CRC ^= 0x05; (CRC = 0x27)

CRC ^= 0x01; (CRC = 0x26) – Servo number, 1

CRC ^= 0x05; (CRC = 0x23) – 2-byte Servo position, 1500uS

CRC ^= 0xDC; (CRC = 0xFF)

CRC ^= 0x00; (CRC = 0xFF) – 2-byte Movement duration, 0uS

CRC ^= 0x00; (CRC = 0xFF) – Set as transmitted 9th byte

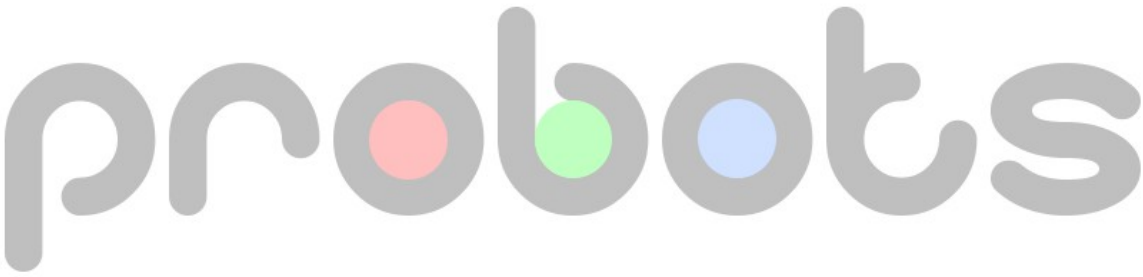
Received “Set Servo” Bytes: <'$'><0x01><0x01><0x01><0x25>

CRC ^= '$'; (CRC = 0x24)

CRC ^= 0x01; (CRC = 0x25)

CRC ^= 0x01; (CRC = 0x24)

CRC ^= 0x01; (CRC = 0x25) – Check against received 5th byte



Command Set

0x00 – Get Version

Returns the touchIt Firmware Version.

Out: <'#'><0x00><0x00><CRC>
In: <'$'><0x00><0x02><0x01><0x00><CRC>

The data received above indicates the firmware is at version 1.0

0x01 – Set Servo

Set numbered servo(s) to a specific position, over a specific duration

Out: <'#'><0x01><0x05><Servo><Pos MSB><Pos LSB><Dur MSB><Dur LSB><CRC>
In: <'$'><0x01><0x01><Servo><CRC>

The 'Servo' byte can be between 1 and 24, Position 'Pos' must be between 0 and 2500uS, but is subject to the mechanical range of your chosen servo. Duration 'Dur' can be between 0 and 65535mS, allowing instant movements or incremental motion lasting up to a minute. The servo number is returned to confirm the task once it has completed.

The length byte is not restricted to 5 bytes; it can be incremented in units of 5 to allow multiple servos to be controlled simultaneously.

0x02 – Get Servo

Returns the current position for the requested servo.

Out: <'#'><0x02><0x01><Servo><CRC>
In: <'$'><0x02><0x03><Servo><Pos MSB><Pos LSB><CRC>

0x03 – Set GPIO

Configure the port direction and pin state for the 6 GPIO pins

Out: <'#'><0x03><0x02><Config><State><CRC>
In: <'$'><0x03><0x01><State><CRC>

The 'Config' Byte defines the port direction; a byte set to XX001010 would set GPIO 2 and 4 as Outputs. The same byte sent as the 'State' byte would then set the respective pins high.

The 'State' byte returned to the host contains the IO Port output state to confirm the requested change.

0x04 – Get GPIO

Retrieve the IO Port input state (required pins must be configured as inputs using 'Set GPIO' beforehand).

Out: <'#'><0x04><0x00><CRC>
In: <'$'><0x04><0x01><State><CRC>