Serial

Cleanflight has enhanced serial port flexibility but configuration is slightly more complex as a result.

Cleanflight has the concept of a function (MSP, GPS, Serial RX, etc) and a port (VCP, UARTx, SoftSerial x).
Not all functions can be used on all ports due to hardware pin mapping, conflicting features, hardware, and software
constraints.

Serial port types

- USB Virtual Com Port (VCP) - USB pins on a USB port connected directly to the processor without requiring
  a dedicated USB to UART adapter. VCP does not 'use' a physical UART port.
- UART - A pair of dedicated hardware transmit and receive pins with signal detection and generation done in hardware.
- SoftSerial - A pair of hardware transmit and receive pins with signal detection and generation done in software.

UART is the most efficient in terms of CPU usage.
SoftSerial is the least efficient and slowest, SoftSerial should only be used for low-bandwidth usages, such as telemetry transmission.

UART ports are sometimes exposed via on-board USB to UART converters, such as the CP2102 as found on the Naze and Flip32 boards.
If the flight controller does not have an on-board USB to UART converter and doesn't support VCP then an external USB to UART board is required.
These are sometimes referred to as FTDI boards. FTDI is just a common manufacturer of a chip (the FT232RL) used on USB to UART boards.

When selecting a USB to UART converter choose one that has DTR exposed as well as a selector for 3.3v and 5v since they are more useful.

Examples:

- FT232RL FTDI USB To TTL Serial Converter Adapter (http://www.banggood.com/FT232RL-FTDI-USB-To-TTL-Serial-Converter-Adapter-Module-For-Arduino-p-917226.html)
- USB To TTL / COM Converter Module buildin-in CP2102 (http://www.banggood.com/Wholesale-USB-To-TTL-Or-COM-Converter-Module-Buildin-in-CP2102-New-p-27989.html)

Both SoftSerial and UART ports can be connected to your computer via USB to UART converter boards.

Serial Configuration

Serial port configuration is best done via the configurator.

Configure serial ports first, then enable/disable features that use the ports. To configure SoftSerial ports the SOFTSERIAL feature must be also be enabled.

Constraints

If the configuration is invalid the serial port configuration will reset to its defaults and features may be disabled.

- There must always be a port available to use for MSP/CLI.
- There is a maximum of 2 MSP ports.
- To use a port for a function, the function's corresponding feature must be also be enabled.
  e.g. after configuring a port for GPS enable the GPS feature.
- If SoftSerial is used, then all SoftSerial ports must use the same baudrate.
- Softserial is limited to 19200 baud.
- All telemetry systems except MSP will ignore any attempts to override the baudrate.
- MSP/CLI can be shared with EITHER Blackbox OR telemetry. In shared mode blackbox or telemetry will be output only when armed.
- Smartport telemetry cannot be shared with MSP.
- No other serial port sharing combinations are valid.
- You can use as many different telemetry systems as you like at the same time.
- You can only use each telemetry system once. e.g. FrSky telemetry cannot be used on two port, but MSP Telemetry + FrSky on different ports is fine.

Configuration via CLI

You can use the CLI for configuration but the commands are reserved for developers and advanced users.

The `serial` CLI command takes 6 arguments.

1. Identifier (see serialPortIdentifier_e in the source)
2. Function bitmask (see serialPortFunction_e in the source)
3. MSP baud rate
4. GPS baud rate
5. Telemetry baud rate (auto baud allowed)
6. Blackbox baud rate

Baud Rates

The allowable baud rates are as follows:

Identifier	Baud rate
0	Auto
1	9600
2	19200
3	38400
4	57600
5	115200
6	230400
7	250000

Passthrough

Cleanflight can enter a special passthrough mode whereby it passes serial data through to a device connected to a UART/SoftSerial port. This is useful to change the configuration of a Cleanflight peripheral such as an OSD, bluetooth dongle, serial RX etc.

To initiate passthrough mode, use the CLI command `serialpassthrough` This command takes four arguments.

```
serialpassthrough <id> [baud] [mode] [DTR PINIO]
```

ID is the internal identifier of the serial port from Cleanflight source code (see serialPortIdentifier_e in the source). For instance UART1-UART4 are 0-3 and SoftSerial1/SoftSerial2 are 30/31 respectively. Baud is the desired baud rate, and mode is a combination of the keywords rx and tx (rxtx is full duplex). The baud and mode parameters can be used to override the configured values for the specified port. `DTR PINIO` identifies the PINIO resource which is optionally connected to a DTR line of the attached device.

For example. If you have your MWOSD connected to UART 2, you could enable communicaton to this device using the following command. This command does not specify the baud rate or mode, using the one configured for the port (see above).

```
serialpassthrough 1
```

If a baud rate is not specified, or is set to 0, then `serialpassthrough` supports changing of the baud rate over USB. This allows tools such as the MWOSD GUI to dynamically set the baud rate to, for example 57600 for reflashing the MWOSD firmware and then 115200 for adjusting settings without having to powercycle your flight control board between the two.

To use a tool such as the MWOSD GUI, it is necessary to disconnect or exit Cleanflight configurator.

To exit serial passthrough mode, power cycle your flight control board.

In order to reflash an Arduino based device such as a MWOSD via `serialpassthrough` if is necessary to connect the DTR line in addition to the RX and TX serial lines. The DTR is used as a reset line to invoke the bootloader. The DTR line may be connected to any GPIO pin on the flight control board. This pin must then be associated with a PINIO resource, the instance of which is then passed to the serialpassthrough command. The DTR line associated with any given UART may be set using the CLI command `resource` specifying it as a PINIO resource.

For example, the following configuration for an OpenPilot Revolution shows the UART6 serial port to be configured with TX on pin C06, RX on pin C07 and a DTR connection using PINIO on pin C08.

```
resource SERIAL_TX 1 A09
resource SERIAL_TX 3 B10
resource SERIAL_TX 4 A00
resource SERIAL_TX 6 C06
resource SERIAL_RX 1 A10
resource SERIAL_RX 3 B11
resource SERIAL_RX 6 C07

resource PINIO 1 C08
```

To assign the DTR line to another pin use the following command.

```
resource PINIO 1 c05
```

To disassociate DTR from a pin use the following command.

```
resource PINIO 1 none
```

Having configured a PINIO resource assocaited with a DTR line as per the above example, connection to an MWOSD attached to an Openpilot Revolution could be achieved using the following command.

```
serialpassthrough 5 0 rxtx 1
```

This will connect using UART 6, with the baud rate set over USB, full duplex, and with DTR driven on PINIO resource 1.

A (desirable) side effect of configuring the DTR line to be associated with a PINIO resource, is that when the FC is reset, the attached Arduino device will also be reset.

Note that if DTR is left configured on a port being used with a standard build of MWOSD firmware, the display will break-up when the flight controller is reset. This is because, by default, the MWOSD does not correctly handle resets from DTR. There are two solutions to this:

1. Assign the DTR pin using the resource command above prior to reflashing MWOSD, and then dissasociate DTR from the pin.
2. Rebuild MWOSD with MAX_SOFTRESET defined. The MWOSD will then be reset correctly every time the flight controller is reset.