Resource Remapping Command

NOTE: this command is only available in version 3.1 and newer of Betaflight.

The IO remapping allows you to configure the pins on the MCU to be utilised for various functions. This is the starting framework - more work can be done.

Pins are remapped using the resources command line interface command.

`resource [function name] [index] [pin]` (e.g. `resource MOTOR 1 A1`)

Where MOTOR is the function, 1 is the motor index (1 based e.g. 1-4 on a quad) and A1 is Port A pin 1 or more commonly referred to as PA1 in STM datasheet documentation.

To remove a mapping, use `NONE` in place of PIN, e.g. `resource MOTOR 5 NONE`

Where a function does not require an index (i.e. there is only 1 possible pin assignment), e.g. `BEEPER`, `SONAR_ECHO` or `SONAR_TRIGGER` then the index **must** be omitted (e.g. `resource BEEPER B6`)

`resource` on its own will list all the available configurable options, and their current setting. This is the output to be added to the `dump` for use in backing up and restoring configuration. Note that this command will list all configured that would be allocated if used.

As an example `resource` will show motors 1-8, but if your mixer is set to QuadX then only motors 1-4 will actually be used, if you change to Oct as the mixer (and reboot) then all 8 motors will be configured.

`resource list` (or `resource show` in more recent versions of Betaflight) will list all pins and their current assignments, including all those in use by system components and **not** configurable by the user. It will also list the currently active DMA utilisation. Note for any adjustments made a save and reboot is required in order for those changes to be visible here. Consider this command the output of the currently active state.

Note that the `save` command must be used after changing pin mapping via the CLI.

```
<center>
<img src="https://cloud.githubusercontent.com/assets/14850998/21921215/c5d3521c-d9a9-11e6-8ed8-c53afdbda50f.jpg" width="70%"><br>
Figure: How different resource command variation works
</center>
```

Available functions, values and constraints

Function	Index	Description	Constraints
`ADC_BATT`	-	Battery voltage sensor	Can only be mapped to other ADC pin
`ADC_CURR`	-	Battery current sensor	Can only be mapped to other ADC pin
`ADC_RSSI`	-	Analog RSSI input from receiver	Can only be mapped to other ADC pin
`BEEPER`	-	Signal buzzer	Usually hard-wired to the base of a transistor to switch the buzzer.
`CAMERA_CONTROL`	-	FPV camera OSD control	Emulates joystick key presses. Needs hard-wired resistor and/or capacitor. Therefore existing CAMERA*CONTROL pin _may* only be usable for certain low frequency applications, like SmartAudio.
`ESCSERIAL`	?	?	?
`I2C_SCL`	`1-n`		
`I2C_SDA`	`1-n`		
`INVERTER`			
`LED`	`1-3`	Indicator LEDs	Don't confuse with LED_STRIP!

Function	Index	Description	Constraints
`LED_STRIP`	-	WS2812 LED data	Usually very good choice for alternative usage, if no WS2812-LEDs are connected/used
`MOTOR`	`1-n`	Motor signal	Mapping between other motor pins) (swapping) should always work fine. Other pins may not work (DMA conflict with DShot).
`PPM`	-	Receiver PPM input	Usually good choice for alternative usage
`PWM`	`1-n`	Receiver PWM input	?
`SERIAL_RX`	`1-n`	Serial receive pin	Can not be remapped to any other pins, but can be used for different functions (including software serial ports).
`SERIAL_TX`	`1-n`	Serial send pin	`SERIAL_RX` / `SERIAL_TX` `11-12` are software serial ports #1 and #2
`SERVO`	`1-n`	Servo signal	?
`SONAR_ECHO`	-		
`SONAR_TRIGGER`	-		
`SPI_SDI`	`1-n`	Used to be called `SPI_MISO`	
`SPI_SDO`	`1-n`	Used to be called `SPI_MOSI`	
`SPI_SCK`	`1-n`		

Wiki pages with examples of using the Resource Commands:

Remapping Motor outputs Using Servos & SERVO_TILT Setup on a Fixed Wing Aircraft

Youtube

Joshua Bardwell

Ivan Efimov

Links

Oscar Liang

VitroidFPV

Community

Discord

IntoFPV

Feeds

RSS

Atom

JSON

Built with Docusaurus

made with ❤️ by **VitroidFPV** and **un!t**