Dragonfly macros



Index

Purpose **3**

Simple Macros **3**

Step 1: when (trigger) 4

Step 2: if (condition) 4

Step 3: then (action) 5

Step 4: extra action 5

Step 5: test your macro 5

Step 6: "Apply now" the macro... twice! 6

Advanced macros **7**

Define "when"; ignore "if" 7

Write an advanced macro 8

Overall syntax: big picture 8

Advanced macro syntax: actions—Part 1 9

Advanced macro syntax: actions—Part 2 10

Park 10micron mount 10

Humidifier control 11

Execute another macro 12

Delayed execution of another macro 12

Stop execution of macro 12

Advanced macro syntax: multiple actions 12

Advanced macro syntax: counter 13

Advanced macro syntax: conditions—Part 1 14

Advanced macro syntax: conditions—Part 2 14

Advanced macro syntax: negating a condition 16

Advanced macro syntax: "if this AND that", "if this OR that" 16

Advanced macro syntax: multiple conditions 17

Advanced macro syntax: multiple conditions... otherwise 18

Appendix 1: Manual macro execution **19**

Method #1: Using the Dragonfly application 19

Method #2: Using Exec. now with the Dragonfly Configurator 19

Method #3: Run from another macro 20

Purpose

The Dragonfly can take actions when a condition is met. These actions can be checked and executed from the Windows PC. This is called a "***script***", and you can find more information in the Dragonfly manual (Dragonfly - Users manual v. 2.0). Scripts are very powerful, but they need the Windows PC to be ON and working.

The Dragonfly allows you to define actions to be performed by the Dragonfly itself when certain conditions are met, too. They are called "***macros***" and have the advantage of being self-contained in the Dragonfly (hence, they don't need the PC to be connected) but have the disadvantage of being less powerful—specifically, macros cannot do ASCOM[1].

Nevertheless, the fact that the PC is not needed makes macros a great tool worthy of being known. You can define up to 20 macros. The most common cases (open/close relays, send pulse and a few more) can be easily created using a step-by-step process explained in "**Simple Macros**". For more elaborate cases, the advanced macros are also available and covered later in this manual.

Simple Macros

Simple macros are defined very easily using the "Dragonfly Configurator" application:

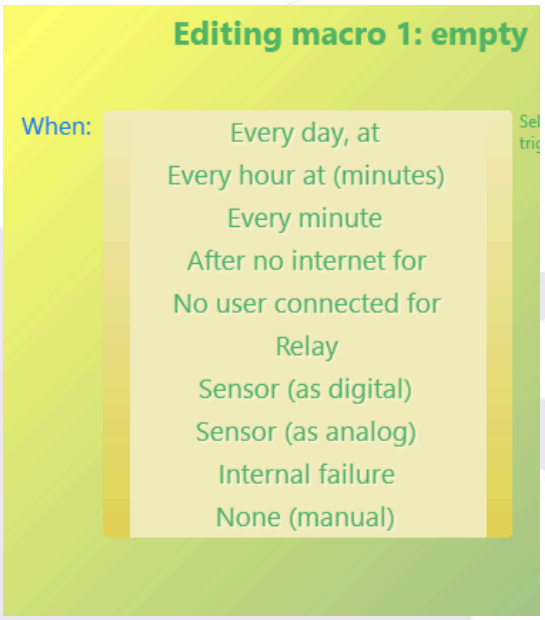


[1] Keep in mind that ASCOM is Windows-centric, so it only makes sense when being executed on a Windows PC

To create or modify one of them, simply click on the desired macro and follow the guided instructions:

Step 1: when (trigger)

As we said, a macro will be executed when a certain condition is met. Said condition is called the "*trigger*" for the macro and is defined in the first step. The following triggers are supported:



—**Every day** at a specific hour and minute

—At a specific minute, **every hour**

—When there is **no internet connection** for a specific number of minutes

—When **no connection to the Dragonfly** occurs for a specific number of minutes

—When a **relay** is opened or closed

—When a **sensor defined as digital** gets open or closed

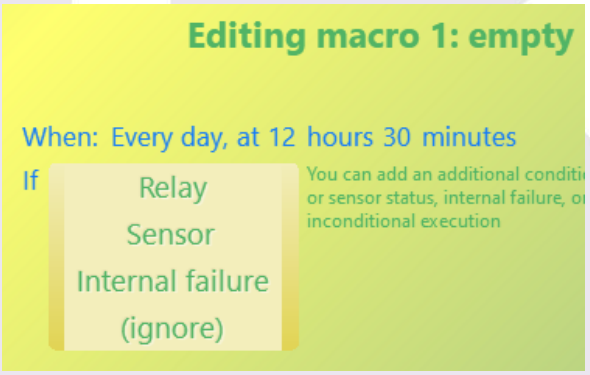
—When a **sensor defined as analogue** goes above or below a specific threshold

—When there is an **internal failure** within the Dragonfly

—The "None (manual)" option allows you to define an action that only you can execute by explicitly commanding so. See Appendix 1 for details on how to run these macros.

Step 2: if (condition)

Sometimes, you may want to define the macro at a specific time but also take the status of a relay/sensor into account, or execute the macro when a relay closes—but only if another relay/sensor has a given status.



Supported conditions are:

—A specific **relay** is open or closed

—A **sensor** is open, closed, above or below a specific threshold

—The **Dragonfly has failed**

—**No condition** (only take step #1 into consideration)

A typical case would be a macro that will close the roof if the CloudWatcher detects unsafe weather. Such a macro will be executed if the relay the CloudWatcher is connected to closes, and only if the sensor that detects that the roof is opened indicates we have not closed the roof yet.

Step 3: then (action)

Finally, you can define the action to be performed by the Dragonfly. As we said, they are simple actions and do not require the PC to be connected.

Supported actions are:

—Open a relay

—Close a relay

—Send a pulse to a relay's output[2]

—Change the status of a relay: close it if open; open it if closed

—Send an e-mail[3]

—Send a PushBullet or PushOver notification[4]

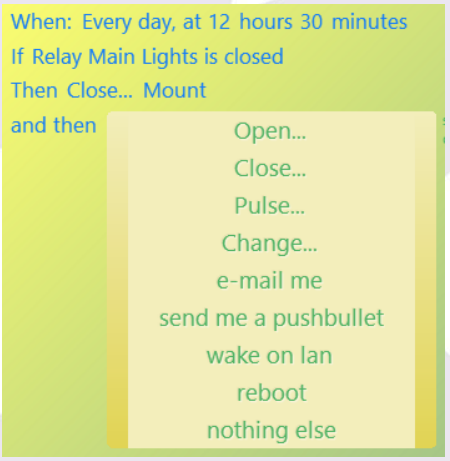
—Send a Wake On Lan notification to a specific MAC address

—Reboot the Dragonfly

—Advanced macro: see below.

Step 4: extra action

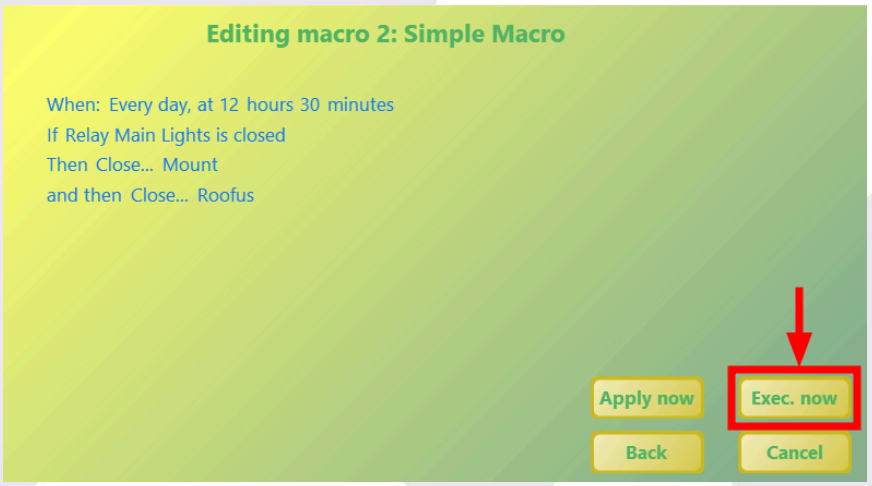
Simple macros allow you to perform **up to two actions**, so after defining the first action, you are presented with the same list of actions (replacing "*Advanced macro*" with "*nothing else*") for you to define the second action, if any.



[2] See Dragonfly's manual for instructions on how to define a pulse
[3] See Dragonfly's manual for instructions on how to define the e-mail address
[4] User's e-mail address is used for this

Step 5: test your macro

Optionally, before saving your macro, you can **test how it works** by clicking on the "*Exec. now*" button. This button will execute the macro without waiting for the "*when*" so that you can check your work right away.

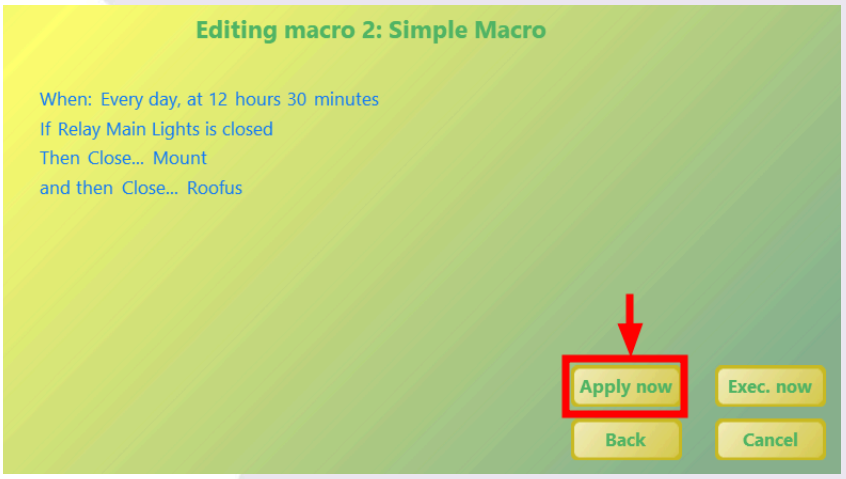


If your macro works as expected, go to the next section to store it into your Dragonfly. Otherwise, you can click on the "Back" button to navigate back to previous steps so that you modify what you need to.

Step 6: "Apply now" the macro... twice!

This is an important step some people miss.

First, you have to click on "*Apply now*" on the window you have been creating and testing your macro on:



This will take you back to the list of macros. In order to actually store the macro into your Dragonfly, it is important you also click on "*Apply now*" in this window too.



Advanced macros

But, hey! Macros can do more than this. However, exposing all their capabilities would overcomplicate the interface necessary to define them. So instead of making the process more obscure, we decided to leave the simple most common tasks easy and expose Dragonfly's macro syntax in the "Advanced macro" option, for cases that *do* need them.

To define an advanced macro, you will have to learn this "**Dragonfly's macro syntax**" and write it in step 3 of the wizard explained in Simple Macros.

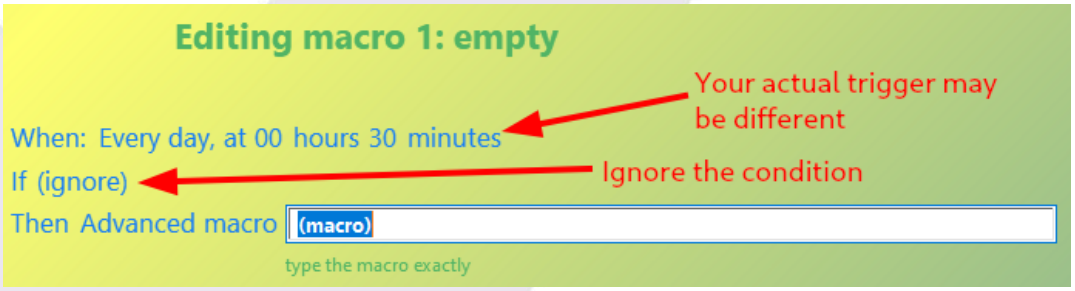
Let 's get started!

Define "when"; ignore "if"

As we said, the advanced macro can be entered in "Step 3" of the macro definition process (see Step 3: then (action)), so in order to get there, you first need to define *when* to execute the macro in Step 1: when (trigger).

However, if you are to write an advanced macro, you must select "ignore" in Step 2: if (condition). This is because advanced macros **already have a syntax to express this condition**. So if you write an advanced macro, the condition you set in "Step 2" will actually be ignored in favour of your advanced macro.

After this, you will be in a situation similar to this:



Write an advanced macro

Just in case you want to try advanced macros as you read this, let's see how to write and apply an advanced macro:

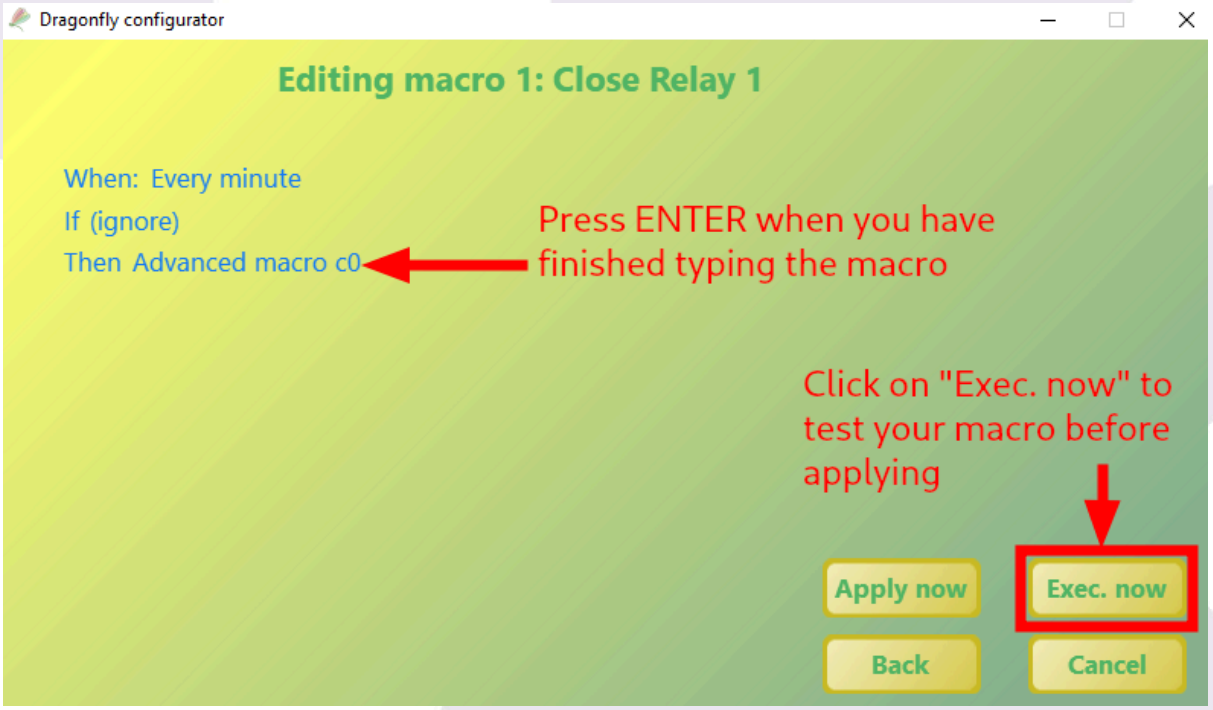
—We start like in the image above

—Type the expression for the macro you want to try

—Press ENTER when done

—The "*Apply now*" and "*Exec. now*" buttons become enabled then

—Before you actually save the macro, it is wise to test it and verify it works as you expect. To do so, press "*Exec. now*" button



—If, when testing your macro, you find that it does not work as you want, you can modify it by clicking on the "Back" button. This will re-enable the edit box and you can modify your macro.

When you are done editing and testing your macro, follow the steps in "Apply now" to actually save the changes.

Overall syntax: big picture

Before entering into the detail of each part of an advanced macro, let's see the overall syntax, so that you know what we're talking about in each section.

Just like a simple macro has a condition and actions, so does an advanced macro. In the following sections, we will see how to write conditions (see Conditions—Part 1 and Part 2) and how to perform actions (see Actions—Part 1 and Part 2).

Since conditions are actually optional (you can perform an unconditional action), we will change the order and explain actions first.

Advanced macro syntax: actions—Part 1

Finally, it is time to actually write our advanced macros. Every action supported by simple macros is, of course, supported here. Let's see them:

Action	Syntax
Open a relay	Use the letter "o" (for "open", lowercase) followed by the number of the relay to open. Internally, relay numbers start in zero instead of one. ***Example***: `o1` *(open relay number two[5])*
Close a relay	Use the letter "c" (for "close", lowercase) followed by the number of the relay to close. Internally, relay numbers start in zero instead of one. ***Example***: `c0`

[5] Remember, internally, the first relay is zero, so "o1" actually refers to relay number 2.

	(close relay number 1)
Send a pulse	Use the letter "p" (for "pulse", lowercase) followed by the number of the relay, then a comma (",") and the length of the pulse in milliseconds. ***Example***: `p1,1000` *(send a 1 second[6] pulse on relay number 2)*
Change a relay	Use the letter "g" followed by the number of the relay. ***Example***: `g2` *(open relay number 3 if it is closed; otherwise open it)*
Send an e-mail	Use the letter "e" followed by the message, surrounded by single quotes. ***Example***: `e'This is an email'` *(sends an e-mail to the address defined in ZeroConf configuration with the text "This is an email")*
Send PushBullet or PushOver	Use the letter "h" followed by the message, surrounded by single quotes. ***Example***: `h'This is a push message'` *(sends a pushbullet or pushover (whichever one you have set up) notification to the e-mail defined in ZeroConf configuration with the text "This is a push message")*
Wake On Lan	Use the letter "k" followed by the MAC address to send the Wake On Lan message, surrounded by single quotes. ***Example***: `k'b8:27:eb:75:de:ea'` *(send WOL message to b8:27:eb:75:de:ea)*
Reboot	Just type "boot", lowercase, without the quotes ***Example***: `boot` *(reboots the Dragonfly)*

Advanced macro syntax: actions—Part 2

But that is not all. The Dragonfly supports other actions (not included in the simple macros to keep them actually simple).

[6] 1 seconds are 1000 milliseconds

Park 10micron mount

Note: reach out to us if you're looking to park another type of network-enabled mount!

If you own a 10micron mount and you have your Dragonfly updated to software version 2.5 and firmware version 4.6.7, you are in luck! You can park your mount with a Dragonfly advanced macro. This means **you no longer need the PC to be working to safely park your mount**.

The syntax is:

```
s'park 10micron <ip-address> <port>'
```

Where *<ip-address>* and *<port>* must be replaced with the IP address and port where your **10micron** is installed (by default, the port is 10000)

Example: `s'park 10micron 192.168.1.100 10000'`
(park 10micron mount installed in 192.168.1.100, port 10000)

Note that the expression after the initial "s" is enclosed between single vertical quotes (‘). It is important to do it that way, or otherwise the macro will not work at all.

Humidifier control

Also new to Dragonflies updated to software version 2.5 and firmware 4.6.7 is the ability to **read weather data** from a Lunatico SOLO device. Specifically, you can switch a humidifier OFF or ON depending on how close the dew-point is to the ambient temperature, to protect your equipment.

The logic is as follows: if the ambient temperature and dew point are too close, the humidifier should start working until the difference is big enough. As a result, we will need two thresholds:

- ON threshold: if the difference between the ambient temperature and the dew point is lower than this threshold, the humidifier will be switched on
- OFF threshold: if the difference between ambient temperature and dew point is higher than this threshold, the humidifier will be switched off.

And remember: you need a SOLO device to read the weather data from. So take note of its IP address.

Bundled with software version 2.5 and firmware 4.6.7, there are two ways to handle your humidifier:

- **Execute a macro** to switch it ON and another macro to switch it OFF
- **Close a relay** to switch it ON, open this relay to switch it OFF

For the **first case**, use this syntax:

```
s'humctlmacro <solo-ip> <on-threshold> <on-macro>
          <off-threshold> <off-macro>'
```

Example: `s'humctlmacro 192.168.1.132 4 0 6 1'`
(Read weather data from the SOLO in 192.168.1.132 and execute macro #0 if the difference is lower than 4 grades degrees, or execute macro #1 if the difference is higher than 6 degrees)

For the **second case**, use this syntax:

```
s'humctlrelay <solo-ip> <on-threshold> <off-threshold>
                      <relay>'
```

Example: `s'humctlrelay 192.168.1.132 4 6 2'`
(Read weather data from SOLO in 192.168.1.132 and close relay #2 if the difference is lower than 4 grades degrees, or open it if the difference is higher than 6 grades)

Again, note that the expression after the initial "s" has to be enclosed between single vertical quotes (').

Execute another macro

If a macro becomes too complex, you may need to divide it into several steps. Or maybe you have macros for simple tasks such as closing the roof or turning off lights and you want to create a "shut all down" macro that executes both. The "Execute another macro" action allows you to do this.

Use the letter "t" followed by the number of the macro to execute. As with relays, the first macro is zero.

Example: `t1`

(execute macro number 2)

Delayed execution of another macro

This is similar to the previous one, only the macro is executed after waiting for a while. This is useful if, for example, you have a macro to park the mount, another macro to close the roof, and you want to write a "shut all down" macro that closes both, but you must give some time for the mount to park before closing the roof.

Use the letter "y" followed by the number of the macro to execute, then a comma (",") and the number of milliseconds to wait before executing the macro.

Example: `y0,2000`
(execute macro number 1 after two seconds; as with relays, the first macro is zero)

Stop execution of macro

This is the advanced-macro-version of the "nothing else" action in the simple macro wizard. For example, say you want to turn on humidifiers if the roof is open and, if not, do nothing. The "do nothing" part can be expressed with this action. We will see an example of this action when we explain the syntax for conditions, we'll simply use the letter "q" and the macro will terminate.

Advanced macro syntax: multiple actions

You can perform **several actions** within the same advanced macro. You are no longer limited to two actions, as when defining a simple macro.

Separate each action with a comma (",") and you're done[7].

For example: `c0,c1,o2,o3,y1,1000`

This means:

—*Close relay number 1 (remember, 0 is the first relay)*
—*Close relay number 2*
—*Open relay number 3*

[7] It is actually not mandatory to use the comma to separate each action, but it makes things much more readable. Compare "*c0c1o2o3y1,1000*" with "*c0,c1,o2,o3,y1,1000*". The latter is much easier to understand.

—Open relay number 4

—Wait for one second, then execute macro number 2

Advanced macro syntax: counter

This is a more complicated one. Let's start with a use case; say you have the CloudWatcher connected to the Dragonfly, and you want to close the roof if the weather is not safe—but you want to be sure it is unsafe and only close the roof if it has been unsafe for 5 consecutive minutes.[8]

We can write a macro that executes every minute and checks the CloudWatcher sensor and, if unsafe... ... Ok, we need to count the number of unsafe's. The Dragonfly offers that via the "@" symbol and the following actions:

Action	Explanation
@n	"n" is an integer value, such as "3", for example. This assigns the value "n" to the counter. *In our example above, we have to do "@0" (assign zero to the counter) if the weather is safe, because we only increment if the weather is unsafe.*
@+	Increments the value of the counter. *In the example above, we do this if the weather is not safe.*
@-	Decrements the value of the counter.

Advanced macro syntax: conditions—Part 1

Do you remember that we said you should ignore the condition if you were going to write an advanced macro? That's because there is an extended syntax for conditions.

The construct we are dealing with is: "if our ***condition*** is met, then execute the ***actions to be executed if the condition is met***; otherwise, execute the ***actions to be executed if the condition is not met***.

In Dragonfly's language, such a construct is:

```
<condition>?<actions-if-condition-is-met>:<actions-if-condition-not-met>
```

[8] This is just an example! Never do that, always trust the *CloudWatcher*!

Note: the "otherwise" part is optional, so a valid condition can simply be:

`<condition>?<actions-if-condition-is-met>`

Let's see some examples you will fully understand after you read the next section:

- If relay #1 is closed, then close relay 2, otherwise open relay 2

`r0?c1:o1`

- If sensor #2 has an analog value higher than 10, then close send one-second pulse to relay 3, otherwise open relay 3

`a1>10?p2,1000:o2`

To fully understand the examples above, we need to know the conditions Dragonfly supports.

Advanced macro syntax: conditions—Part 2

The conditions are pretty much the same as the ones offered in the condition section, plus the counter. Let's see them:

Syntax	Condition met if...
r[n]	Where [n] is the number of a relay (zero is the first relay). The condition is met if the given relay is closed ***Example***: `r0` *(the condition is met if the first relay is closed)*
d[n]	Where [n] is the number of a sensor (zero is the first sensor). The condition is met if the given sensor is active. ***Example***: `d2` *(the condition is met if the sensor number 3 is active)*
a[n]>[value]	Where [n] is the number of a sensor (zero is the first sensor). The condition is met if the given sensor value is greater than [value]. ***Example***: `a0>100` *(the condition is met if the sensor number 1 has a value greater than 100)*

@>[value]	The condition is met if the counter's current value is greater than [value]. ***Example***: `@>5` *(the condition is met if counter's value is greater than 5)*
w>[seconds][9]	The condition is met if the Dragonfly has received no communication from outside (for example, the application in Windows PC) for more than [seconds]. ***Example***: `w>10` *(the condition is met if the Dragonfly not received any external communication for more than 10 seconds)*
i>[checks]	The condition is met if Dragonfly has not detected a valid internet connection for more than [checks] consecutive checks. ***Example***: `i>100` *(condition is met if Dragonfly has not detected a valid internet connection for more than 100 consecutive checks)*
f	The condition is met if the Dragonfly has detected an internal failure.

Conditions using the ">" (greater than) operand can also be used with:

- "<": lesser than
- "=": equals

Now we can fully understand the examples in previous sections, as well as others:

- `r0?c1:o1`
- `a1>10?p2,1000:o2`
- If the counter is less than 10, open relay 1, otherwise close it
  `@<10?o0:c0`
- If no communication for 10 seconds, then reboot the Dragonfly:
  `w=10?boot`

[9] "*w*" stands for "*watchdog*"

Advanced macro syntax: negating a condition

We have seen how to check if a relay is closed ("`r[n]`") or if a digital sensor is active ("`d[n]`"). Now we want to do the opposite—check if a relay is *open* or a digital sensor is *not active*.

To do this, we have to negate the original statement, using the exclamation mark ("!") before the expression. See these examples:

Positive expression	**Syntax**	**Negated expression**	**Syntax**
Relay 2 is closed	`r1`	Relay 2 in **NOT** closed (open)	`!r1`
Sensor 3 is active	`d2`	Sensor 3 is **NOT** active (inactive)	`!d2`

Let's see some more complex examples:

- If relay 1 is open, then open relay 2
  `!r0?o1`
- If sensor 3 is inactive, then increase counter, otherwise set the counter to zero
  `!d2?@+:@0`

Advanced usage: You can also negate expressions such us "`a3>100`" (*analogue sensor #4 reads more than 100*), but you have to use brackets to enclose the condition you want to negate. For example:

- If analogue sensor #4 is **NOT** greater than 100, open relay #2
  `!(``a3>100`**`)`**`?o1`
- If counter is **NOT** 5, open relay #3
  `!(``@=5`**`)`**`?o2`

Advanced macro syntax: "if this AND that", "if this OR that"

Conditions are not always simple; they can be compounded and linked to one another with:

- "AND" meaning **both** conditions must be met or else the action will not be executed.

- "OR" meaning that if **either** of the conditions is met, the action is executed.

This syntax, which is not available in simple macros, is available in advanced macros.

- "AND" uses the "**&**" symbol
- "OR" uses the "**|**" (pipe) symbol

For example, say we have:

—The roof-close mechanism connected to relay number 3, so that it is closed when this relay is closed;

—the CloudWatcher connected to relay #1, so that if closes when weather is not safe;

—and a presence sensor in relay #2, so that this relay is open when I am in the observatory and closed when I am away.

Say we want the roof to close automatically when the weather is not safe **AND** I am away. In this case, I want to close relay #3 y both relay #1 **AND** #2 are closed. Otherwise, open relay #3. This would be syntax:

```
r0&r1?c2:o2
```

Another option would be to open the roof if I am at the observatory **OR** the weather is safe. This is, open relay #3 (roof) if relays #1 or #2 are open[10].

```
!r0|!r1?o2:c2
```

Advanced macro syntax: multiple conditions

One of the strengths of advanced macros is the ability to execute multiple actions... and also **check multiple conditions**. To check multiple conditions, we just have to nest several "`<condition>?<actions>:<actions>`" constructs.

For example, say we want to close relay number 5 if sensor number 1 is inactive and relay number 2 is closed too. Another way to say this would be: *if sensor number 1 is inactive, then if relay number 2 is closed, then close relay number 5*. In

[10] Remember that because "`r1`" asks if relay #2 is closed, to ask whether it is open we have to use the negation ("!")

this more weird way to say the same thing, we are actually nesting the construct we are to use.

Let's translate that into Dragonfly's syntax:

- "if sensor number 1 is inactive", this is, "if sensor number 1 is **NOT** active": `!d0`
- "if relay number 2 is closed": `r1`
- Close relay number 5: `c4`

So, combining it all:

```
!d0?r1?c4
```

Using this technique, you can check the conditions you need.

Advanced macro syntax: multiple conditions... otherwise

Conditions have a "positive" part (the actions to be performed if the condition is met) and an "*otherwise*" part (the actions to be performed "otherwise"). Now that we know we can nest several conditions, there are more complicated statements that we can make.

For example, say you want to:

- If relay #1 is closed:
    - If sensor #2 is greater than 100, then close relay #3, otherwise open relay #3
- Otherwise, open relay #3

That would look something like this:

```
r0?a1>100?c2:o2:o2
```

Confusing, right? Well, if you do nest several conditions with the "*otherwise*" part, there is one rule you must follow, and that is to **enclose inner conditions between brackets**. So, in the example above, type this instead:

```
r0?(a1>100?c2:o2):o2
```

That's more like it. And following that rule, you can now nest conditions and use the "otherwise" clause on the combination of them.

Appendix 1: Manual macro execution

Say you have defined a macro to be executed manually by choosing "*None (manual)*" in the "when" part, or you simply want to test any existing macro instead of waiting for the "*when*" trigger to apply. There are three ways to do this:

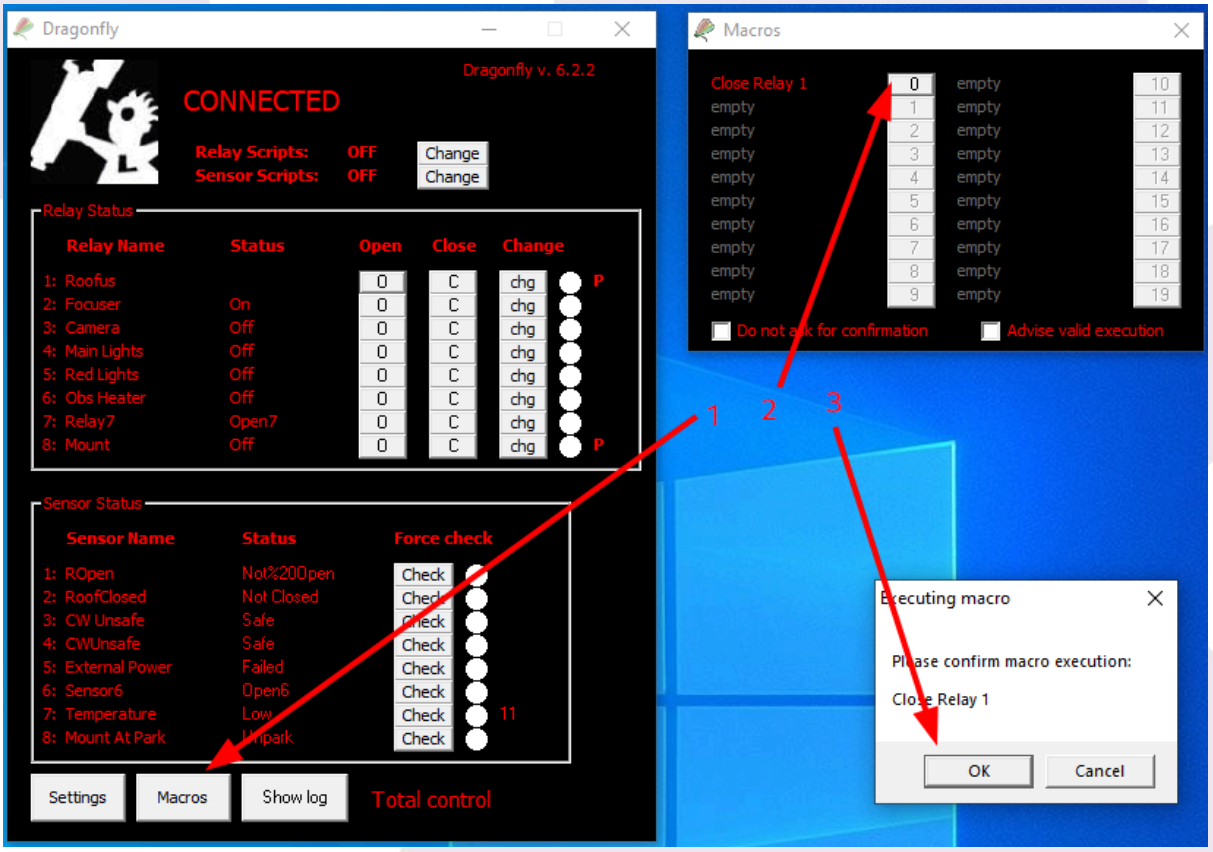
Method #1: Using the Dragonfly application

—Open the Dragonfly application, then click on the *Macros* button. A small window will appear with a button for each macro.

—Only the buttons associated with macros actually defined will be enabled. In our example below, only the first macro is defined, so the rest of the buttons are disabled.

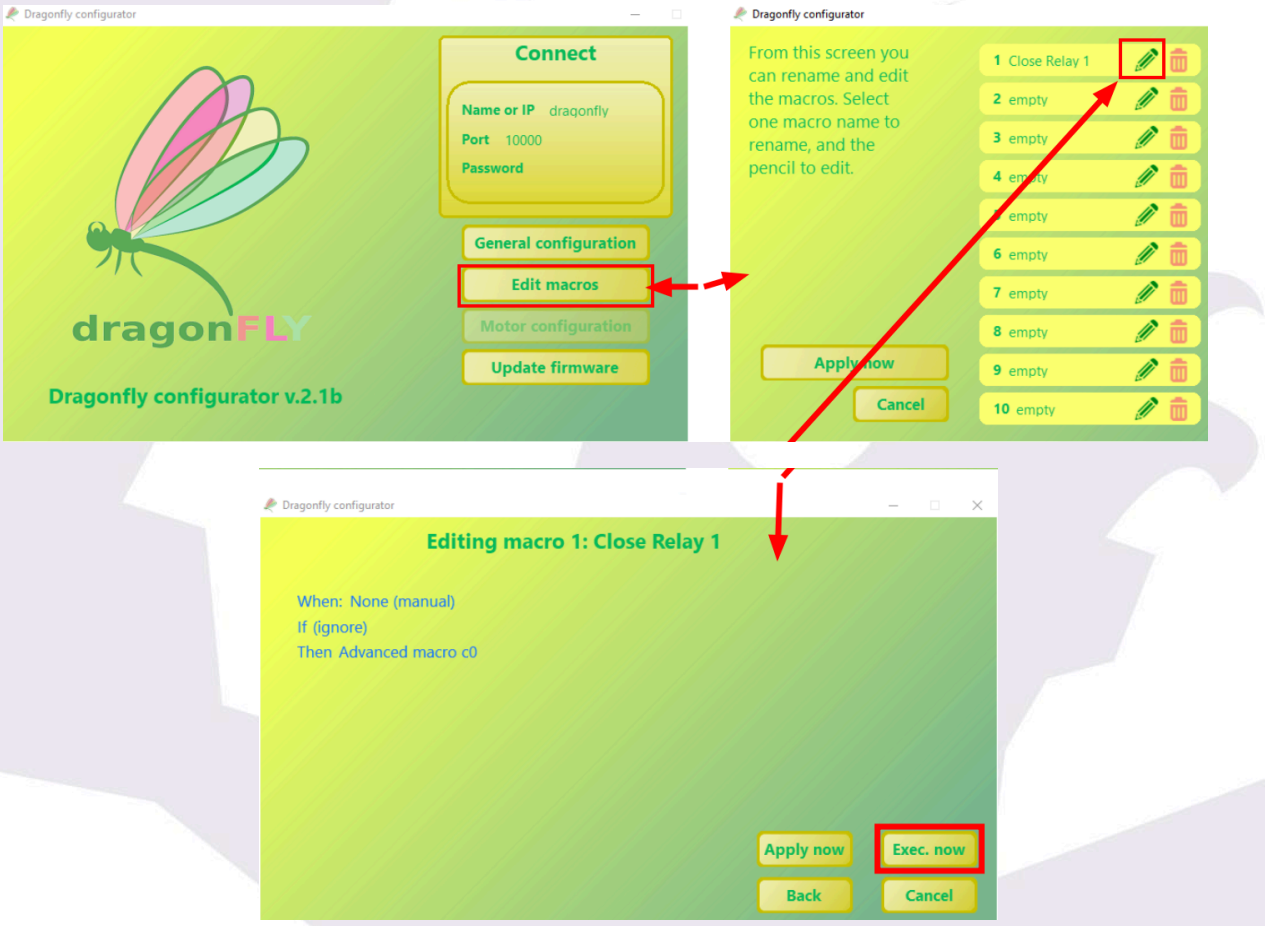
—To run a macro, simply click on the button next to it.

—A confirmation message will appear. Press "Ok" and the macro will be run.



Method #2: Using *Exec. now* with the Dragonfly Configurator

—Open the Dragonfly Configurator application and click on "Edit macros" to display the list of available macros.
—Then, click on the pencil icon of the macro you want to run. The definition of the macro will appear.



—If you then click on the *Exec. now* button, the macro will be run.

Method #3: Run from another macro

As we have seen, the "t" (execute macro) and "y" (delayed macro execution) actions do exactly this: run a macro from another macro.

Questions? As usual, feel free to check our forum or reach out at support@lunaticoastro.com