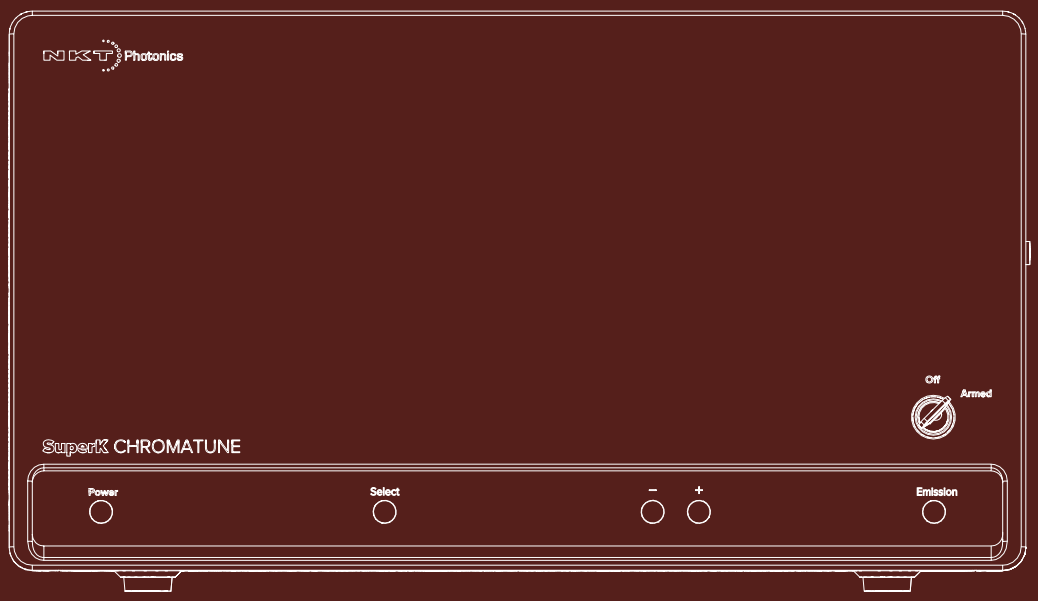
800-642-003

SuperK
CHROMATUNE

Scripting Guide

Revision 1.0 09-2023



NKT Photonics

SCRIPTING GUIDE

This guide includes information for the following NKT Photonics products:

SuperK CHROMATUNE

Wavelength Tunable Laser

WARNING: Before executing a CHROMATUNE script, always check the beam path and ensure it is safe.

Manufactured by:

NKT Photonics A/S
Blokken 84, Birkerød-3460 Denmark

Guide Overview

This product guide is intended to provide automated scripting information for SuperK CHROMATUNE laser systems.

WARNING: Do not operate the laser before first reading and understanding all warnings, cautions and handling information stated within the document:

SuperK CHROMATUNE Laser Safety, Handling and Regulatory Information

The paper copy of this document is included with your laser however it can also be downloaded from:

https://www.NKT Photonicshotonics.com/product-manuals-and-documentation/

CAUTION: Use of controls or adjustments or performance of procedures other than those specified herein may result in hazardous radiation exposure.

WARNING: Before executing a CHROMATUNE script, always check the beam path and ensure it is safe.

Target Audience

This guide is for technical personnel involved in the selection, planning and deployment of lasers in laboratory and industrial settings. The guide assumes a reasonable knowledge level of lasers, photonic principles and electrical interface connectivity.

NOTE: CHROMATUNE lasers are designed for anyone to operate. Other than safety measures, laser expertise is not required.

Chapters Inside

This guide includes the following chapters:

- Chapter I "CHROMASCRIPT Introduction" — Describes the interface and the details in using it and introduces the script elements used for a CHROMASCRIPT.
- Chapter 2 "Step by Step Programming example" — Provides all basic details in creating a CHROMASCRIPT and executing it.
- Chapter 3 "Script elements" — Describes code elements individually including: commands, operators, variables, and arrays.
- Appendix A— "Sample scripts" – A listing of four sample scripts which you can use or base other scripts from.

Added information and Safety Notices

Lasers are highly dangerous devices that can cause serious injury and property damage. This guide use the following symbols to either highlight important safety information or provide further information in relation to a specific topic.

NOTE: Highlights additional information related to the associated topic and/or provides links or the name of the NKT guides describing the additional information.

CAUTION: Alerts you to a potential hazard that could cause loss of data, or damage the system or equipment.

WARNING: The laser safety warning alerts you to potential serious injury when using the laser.

Revision

The section records the document revision details.

Release date	Version and changes
09-2023	First release

Contents

1 CHROMASCRIPT Introduction

Overview

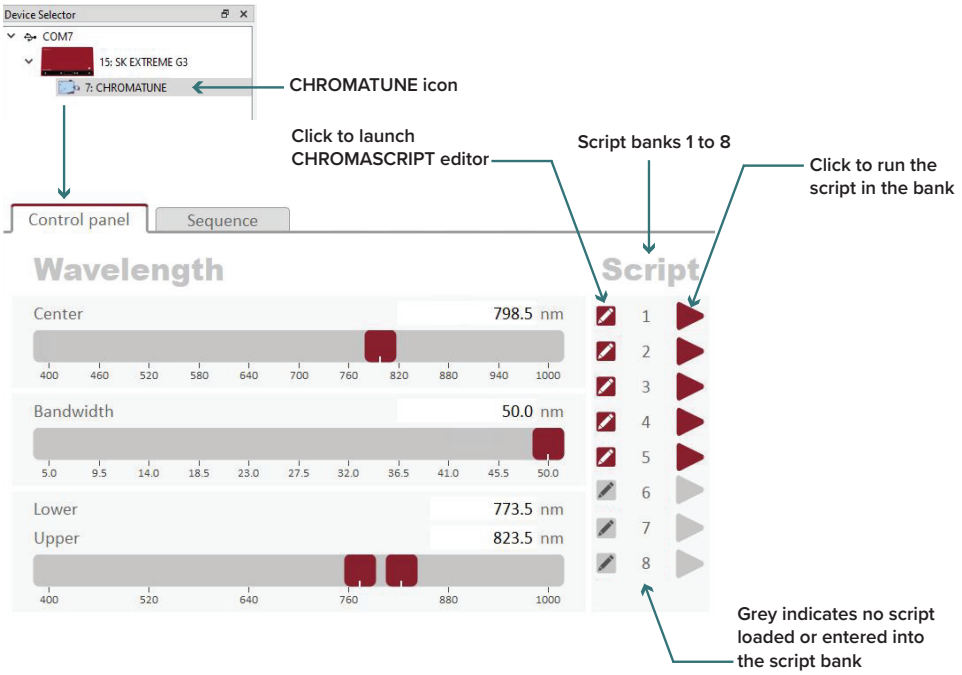
Using the CHROMASCRIPT scripting interface you can automate the spectral output of the CHROMATUNE laser. Automation is based on the executing script setting the laser's filters which adjust emission center wavelength, bandwidth, and output power. Multiple different emission settings can be sequenced automatically by the script on a time basis and additionally the script can be configured to wait for an external input signal to synchronize emission with the application. You can also program the script to show and hide fluorophore datasets in the *Spectrum View* window during script execution.

Launching the CHROMASCRIPT editor

In CONTROL you can create and store up to 8 scripts, stored in an individual CHROMASCRIPT bank within the laser's CHROMATUNE module. You create or load scripts through the CHROMASCRIPT editor. To create a script do the following:

1. Start CONTROL and connect it to the CHROMATUNE laser.
2. In the Device Selector, click on the CHROMATUNE icon.
3. Select the Control Panel tab.
4. Under the Script heading to the right of the panel, click the pencil button next to any of the script banks marked 1 to 8.
5. The CHROMASCRIPT editor launches - see "CHROMASCRIPT editor" on page 10.

Figure 1 Launching the CHROMASCRIPT editor



WARNING: Before executing a CHROMATUNE script, always check the beam path and ensure it is safe.

SCRIPT Banks 1 to 8

As Figure 1 shows, there are 8 script banks. The banks are in the CHROMATUNE module and in each bank you can create and store a single CHROMASCRIPT for execution by clicking the arrow button next to the bank number.

Line numbers

CHROMASCRIPT code uses line numbers. This means the execution of the script is sequenced by line number unless a command (Goto, Gosub) directs the execution pointer to another line.

CHROMASCRIPT editor

The CHROMASCRIPT editor opened on an empty bank is shown in Figure 2. Available commands in grey are listed in the empty script entry field to help begin coding. The white script entry field accepts the script lines of code and the grey area to the right shows the code lines and pointer.

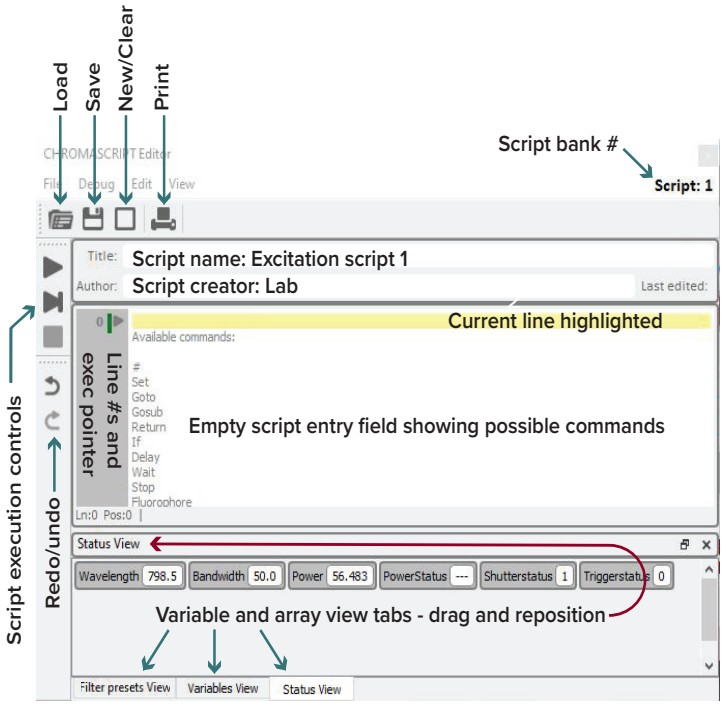
Entering scripts

To enter a script either type or copy and paste your code into the white field.

Loading scripts

You can load a saved script by clicking the FILE FOLDER button at the top left and selecting it from the PC file system.

Figure 2 Empty CHROMASCRIPT editor



Saving scripts

Save a script to the PC file system by clicking the FLOPPY DISK icon along the top of the window. Select the folder and name and click *Save*.

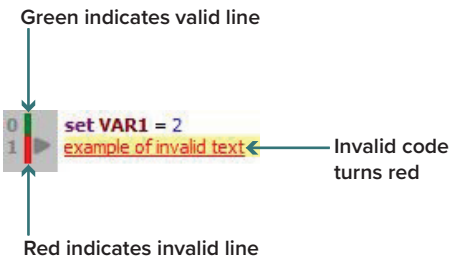
Clear and enter a new script

To clear the code field of and currently entered script, click the EMPTY SQUARE icon along the top of the window. The editor clears any code and the field is empty, ready for any new entry.

Invalid errors when entering code

If you enter invalid code into a script line, the CHROMASCRIPT editor marks the text red and the vertical bar next to the line number changes from green to red as shown in Figure 3.

Figure 3 Invalid text



Undo/Redo

The two buttons marked with curled arrows on the left provide undo and redo functions when entering and editing code.

Print the script

You can print the script by clicking the PRINT icon which is the icon furthest to the right along the top. Select a printer (and any properties or preferences) and click print.

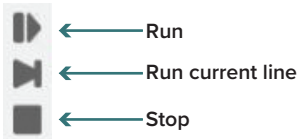
Pointer and yellow line highlight

A script line is highlighted in yellow when the execution pointer points to it. This is the current line.

Executing the script

The buttons along the left column of the editor window control the script execution. Remember, the script executes within the CHROMATUNE module.

Figure 4 Execution buttons



- Run – click this button or F5 to execute the script.
- Run current line – click this button or F6 to execute the current line only.
- Stop – click this button or F8 to stop executing the script.

Externally Triggered script control
You can include code in the script to pause execution and wait for an external trigger at the *Trig In* BNC port before continuing – see "Wait" on page 28.

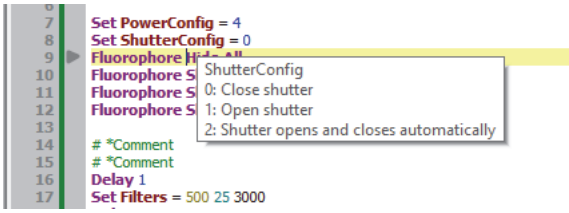
Triggering external devices
You can add an output trigger in the script code to signal an external device when the CHROMATUNE filter tuning is completed – see "TriggerConfig" on page 36.

Title Enter a descriptive name of the script in this field. When the file is saved, this title is kept in the file header and will appear in this field next time you load the file.

Author Enter the script creator's name. When the file is saved, this name is kept in the file header and will appear in this field next time you load the file.

Hover (over) function Hovering your mouse over many components (including code) of the editor will result in a box with help information being displayed.

Figure 5 Hover function example: ShutterConfig command



Variable and array view tabs The view tabs allow you to monitor the contents of variables and arrays of the system during script execution. You can monitor these contents to check the operation of the script, particularly for debugging purposes.

Reposition the tabs
You can drag the tabs out from the bottom of the editor window into their own separate windows which you can resize and position on the screen as is suitable.Figure 6 shows all the tabs repositioned. The *Filter presets view* is moved to the top of the editor by click-hold dragging it out and releasing the mouse when the background turns blue in the desired area. The *Status view* and *Variables view* tabs are completely dragged out and floating on top of the editor and they have been resized with all variables in view.

Status view

See also "System variables" on page 15. The Status view tab shows the current status and reading of the following filter and system variables:

- *Wavelength* – the emission wavelength set in nm.
- *Bandwidth* – the emission bandwidth set in nm.
- *Power* – the setpoint power level in mW.
- *PowerStatus* – The power control mode set.
- *ShutterStatus* – The position of the shutter and operation mode.
- *TriggerStatus* – The signal level on the *Trig out* BNC port on the rear panel.

Variables View

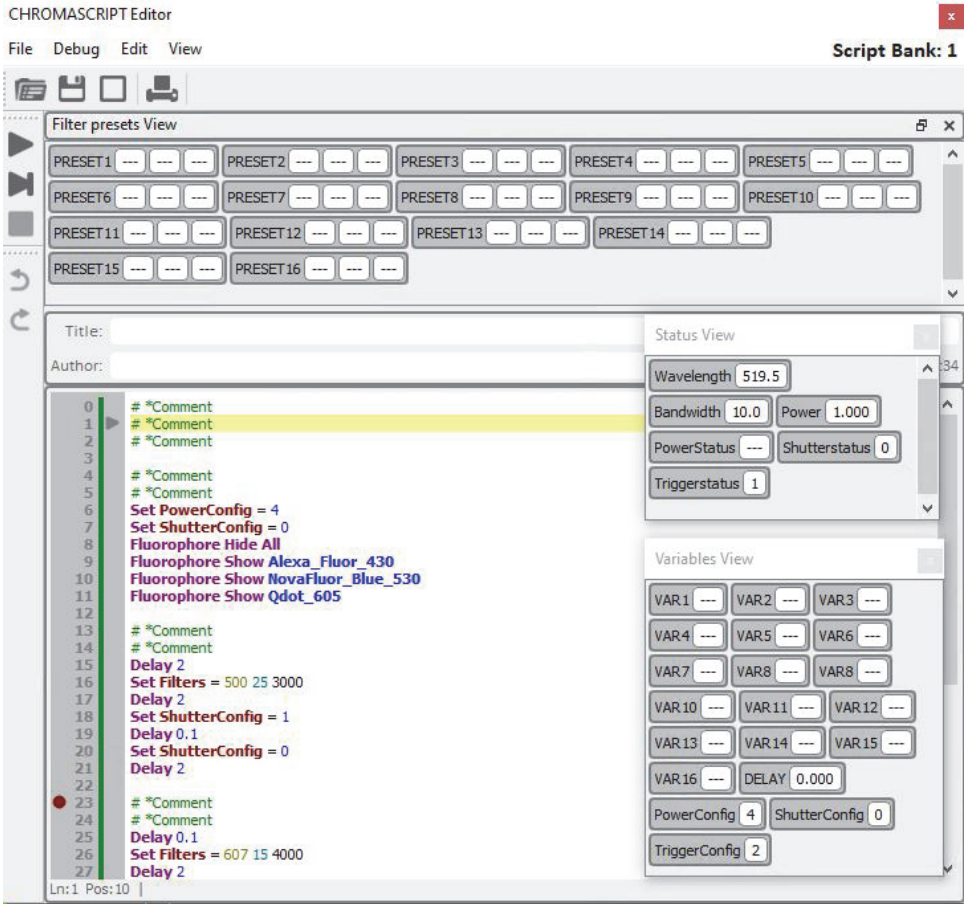
The *Variables view* tab shows the content of some system variables and all user-defined variables. The contents are updated as the script executes or an operator manually modifies the laser controls.

- User-defined variables: *VAR1* to *VAR16* – see "User-defined variables" on page 15
- System variables – see "System variables" on page 15

Filter presets view

This view tab shows the contents of all Preset arrays for *PRESET1* to *PRESET16*. Their contents are updated according to the script execution. See - "Array variables" on page 15.

Figure 6 View tabs dragged out or repositioned



Script elements

Commands

There are 9 commands or actions you can use when you create a CHROMASCRIPT. All code lines must have at least one command. The command set is described in Table 1 below.

Table 1 Commands

Command	Description	Details
Set	Configures the value of a variable.	Set on page 25
Delay	Pause the script for a specified duration in milliseconds.	Delay on page 25
If	Conditional program control	If on page 26
Goto	Unconditional program jump	Goto on page 27
Gosub	Jump to a subroutine section of code.	Gosub on page 27
Return	Return from a subroutine section of code.	Return on page 28
Wait	Pause the script until trigger detection or timeout.	Wait on page 28
Fluorophore	Show or hide a fluorophore dataset.	Fluorophore on page 29

Command	Description	Details
Stop	Ends script execution.	Stop on page 30

Operators

When writing a CHROMASCRIPT, operator symbols are used with commands and variables to perform either an arithmetic function or a comparison (relational). Operators types are listed below in Table 2 with links to their detail.

Table 2 Operators

Type	Description	Details
Arithmetic	Increase or decrease a variable.	+ and - on page 31
Relational	Compare a variable with variable or value.	= , != , < , <= , > and >= on page 31

Variables

User-defined variables

Constant variables *VAR1* to *VAR16* hold any integer or decimal value and are used as utilities to set system variables, delays, and counters.

Filter variables

Filter variables are *Wavelength*, *Bandwidth*, and *Power*. When you configure one of these variables to a new value, the CHROMATUNE module controlling the emission output is set to the value. These filter variables are listed with the system variables in Table 3.

System variables

These variables set certain laser parameters to modify the laser configuration such as power mode of operation, shutter position and its trigger output levels.

All variables are listed below in Table 3.

Table 3 Variables

Variables	Description	Details
Wavelength	Sets the emission center wavelength.	Wavelength on page 33
Bandwidth	Sets the emission bandwidth.	Bandwidth on page 33
Power	Sets the emission power level.	Power on page 34
VAR1-VAR16	16 utility variables to hold any constant.	VAR1-VAR16 on page 34
Powerconfig	Sets the power operation mode.	PowerConfig on page 35
Shutterconfig	Sets the shutter position or operation.	Shutterconfig on page 35
Triggerconfig	Sets the signal level at the Trig out BNC port.	TriggerConfig on page 36

Array variables

Filters (system array)

This array stores system filter settings that control the CHROMATUNE module emission configuration. You can use this array to set all filter settings with one command. The three system settings within the array contents are, in order:

(Wavelength or WL), (Bandwidth or BW), (Power or PW)

PRESETS1 TO 16

You can use these arrays to store filter setting triplets made up of WL, BW, and PW. You can then set the Filter system array at logical points in the script to the three values in one of the 16 presets.

Table 4 Array variables

Type	Description	Details
Filter	Contains the filter settings of the laser.	Filters on page 37
PRESET1-16	An array that contains preset filter values.	PRESET1 to PRESET16 on page 37

Script debug

Disable emission

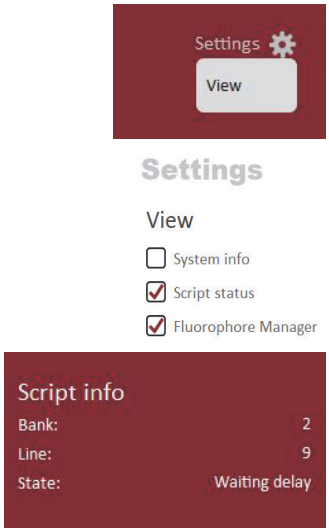
To save laser run time, it is recommended to always disable emission when debugging scripts.

Script state

You can monitor the current state of an executing script by turning on the *Script Info* in the Status panel of the CHROMATUNE. Do the following:

Procedure 1 Enable Script Info in the Status panel

1. Click the CHROMATUNE icon in the *Device Selector* panel.
2. Click the *Settings* gear wheel in the upper right of the *Status* panel.



3. Click *View* in the drop down menu..
4. Put a check mark next to *Script Status*.
5. You can now observe script information by viewing the *Script Info* in the status panel.

- *Bank* – which of the 8 CHROMASCRIPT banks is currently executing.
- *Line* – the script line number currently executing.
- *State* – Informational message indicating the current script action or errors.

Single stepping

To debug a script not working as expected or it has an error, you can step through the code using F6 or the *Run current script line* button – see Figure 4 on page 11 . Do the following:

Procedure 2 Single stepping

1. Press F6, under Script Info, the *State*: changes to *SingleStepping*.

2. Continue pressing F6 while carefully watching the value of variables in the *Variables View* and *Status View* and also the *Script Info: State* message in the status panel.

3. If there is a logic fault in the program, you will likely notice one of the variables being set incorrectly than what you expected or the script jumps to the wrong line of code.

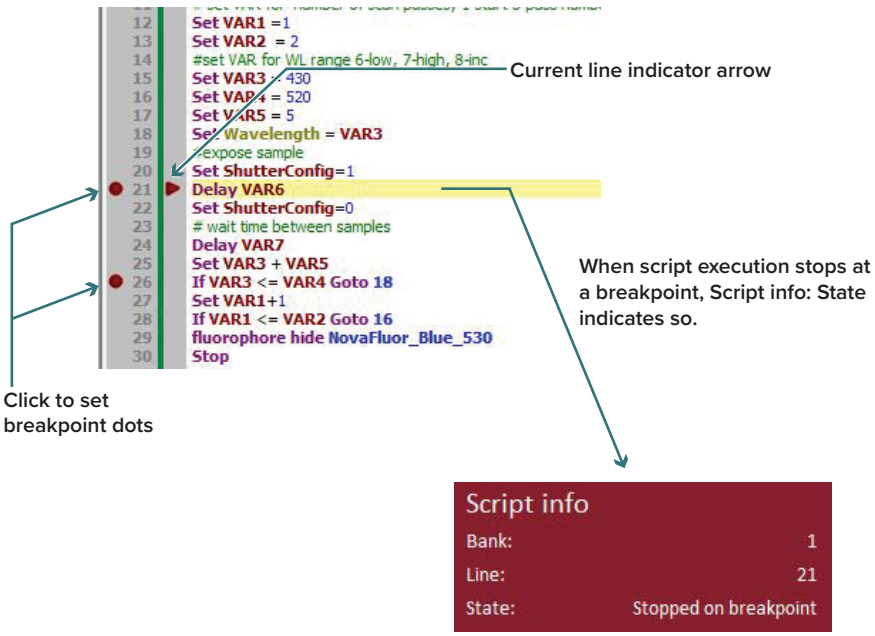
4. Check the validity of the line. If the line is valid you must work backwards through the code to find the bad logic that caused the fault.

Breakpoint

You can set an intentional pause or breakpoint at any line by clicking in the column to the left of a line number. A red dot will appear next to the line indicating the breakpoint is set. Start the script, it only executes until it reaches the breakpoint line. At that point you can inspect variables and try to deduce what occurred. You can then:

- Stop the script - press F8.
- Continue executing the script - press F5.
- Single step through the script - press F6.

Figure 7 Two breakpoints set



Script editing

You can edit a script any time by clicking the pencil item next to one of the 8 banks containing a script for execution. You can only edit the script when it is stopped or not executing. Directly enter new text and delete text in each script line or add new lines, it is that simple.

Offline editing

To edit a script offline, using any text editor to open a script that has been created in the CHROMASCRIPT editor or to create a new script.

2 Step by Step Programming example

This section describes a straight-forward programming example from start to finish. The example includes the following:

1. “Goal” - describe the function of the script – Goal

2. “Script creation” – a method to create a script fully described.

3. “Script execution” – how to run the script including “Monitor the indicators”

4. “Flowchart” – a visual representation of the script flow.

Goal

In this example, the script’s function is to test excitation across a range of wavelengths. Both power and exposure time is constant; however, both can be varied before executing the script. The script should also launch a fluorophore dataset in the *Spectrum View* to help monitor its function and it should be able to repeat for as many iterations as needed.

Script creation

The following steps are suggestions and one method to use when creating CHROMASCRIPTS. Once you are familiar with the interface, you can write any valid script code with a subsequent execution flow that is logical to you.

Suggested script creation steps.

You can find all code for this script here: “Repetitive wavelength scanner” on page 40.

1. Write a small description of your script function which includes all emission parameters to be controlled by the script. You can use the description to write comments in the header of your script. Below is a description for the example script of this chapter:

“The script runs an iterative scan across a range of center wavelengths with a settable number of passes. The script includes a wavelength increment for each pass to be tested. The script can also set the emission power along with the exposure and sample collection times.”

2. Map a set of user-defined variables based on the scripts function. Variables in CHROMASCRIPT are fixed names, therefore mapping them beforehand makes it easier to write your script code and debug or modify later. Here are the variables for our example:

- We need to set the number of passes to test our range of wavelengths, so assign:

  VAR1 = The starting pass #, it should be always 1.

VAR2 = The ending pass or number of iterations.

- Now we need variables for the range of center wavelengths to be checked and an increment. So:

  VAR3 = The minimum center wavelength to test in nm.

  VAR4 = The maximum center wavelength to test in nm.

  VAR5 = The wavelength increment to increase the center wavelength until the range maximum is reached.

- Finally we will set user-defined variables for exposure and collection times. So the times needed are:

  VAR6 = The the exposure time in seconds to excite the fluorophore.

  VAR7 = The collection time in seconds to allow an external device/ application to collect the data and prepare to receive the next emission sample.

3. We will need some system and filter variables:

"PowerConfig" - to set how the power is controlled, this can be varied.

"Shutterconfig" - to open and close the shutter for exposure and collection.

"Power" - to set the emission power level.

"Bandwidth" - to set the emission bandwidth.

4. Now let's work out the script code:

A. First the script needs to set some system and filter variables so:

Set PowerConfig = 4 - set the power to continually update its level from internal feedback.
Set ShutterConfig = 0 - close the shutter.
Set Power = **1000** - set power to 1 mW.
Set Bandwidth = 20 - set the emission bandwidth to 20 nm.

B. The script should turn on a Fluorophore in the *Spectrum View* window using the command:

Fluorophore Show "Fluorophore dataset filename"

C. Next begin setting variables:

VAR6 = "exposure time in seconds"
VAR7 = "collection time in seconds"

VAR1 = "the start pass 1"
VAR2 = "the number of passes, any integer."

D. Next, set the center wavelength variables and configure the *Wavelength* system variable:

VAR3 = "the minimum wavelength of the range in nm"
VAR4 = "the maximum wavelength of the range in nm"
VAR5 = "the increment to increase the wavelength"
Set Wavelength = VAR3 - the system configures the wavelength to the VAR3 value.

E. With all variables configured and the wavelength system variables set to VAR3, we are ready to create the functional part of the script. The script is going to open the shutter to excite the fluorophore, close the shutter and then increase the wavelength by the wavelength increment. Finally the script repeats the entire range of wavelength emissions for the number of iterations or passes set in VAR2. So here is the code:

Set ShutterConfig = 1 - the shutter opens.
Delay VAR6 - pause for VAR6 seconds.
Set ShutterConfig = 0 - close the shutter.
Delay VAR7 - pause for VAR7 seconds.
Set VAR3 + VAR5 - increment the wavelength by VAR8.
If VAR3 <= VAR4 Goto 'line #' - this **If** loop checks if the wavelength to be tested is still within the range and if so jumps execution to the line in the script that sets the wavelength to VAR3 (step D). When all wavelengths are checked move to the next line of code to increment the number of passes.
Set VAR1+1 - increment the pass number by 1.
If VAR1 <= VAR2 Goto 16 - this **If** loop checks if the maximum number of passes set is reached. If not, repeat the loop over the wavelength range for another pass. When the **If** comparison statement is false all passes are complete so move the program counter to the next line.
fluorophore hide NovaFluor_Blue_530 - turn off the fluorophore dataset.
Stop - end script execution and reset the program counter.

F. All of the previous script (and other example scripts) is found in the appendix along under section "Repetitive wavelength scanner" on page 40. You can copy the code into a CHROMASCRIPT bank and then set the variables as applies for the case.

Script execution

WARNING: Before executing a CHROMATUNE script, always check the beam path and ensure it is safe.

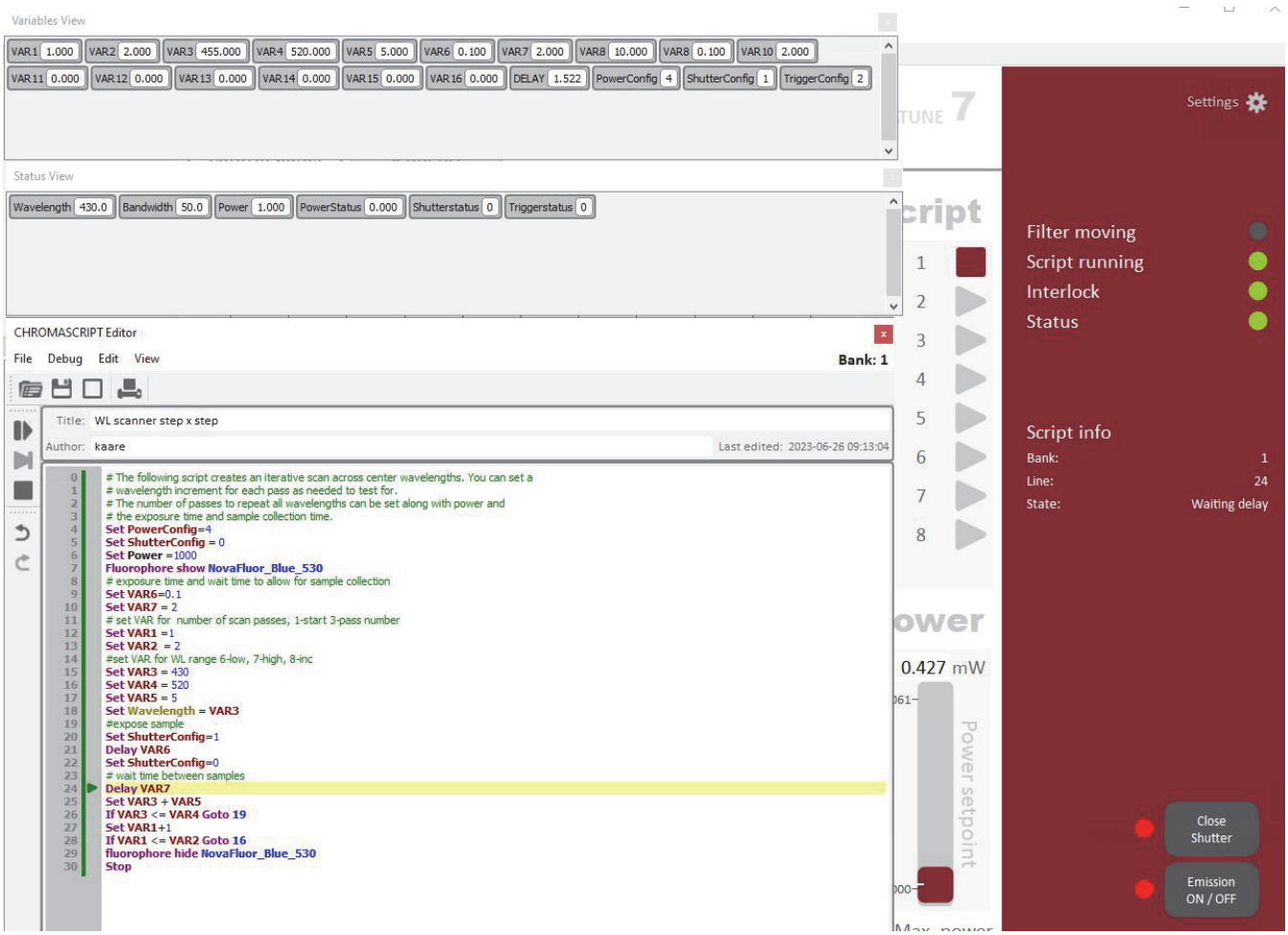
Run the sample script

You can copy the entire script as described in "Suggested script creation steps." on page 19 from "Repetitive wavelength scanner" on page 40 into one of the CHROMASCRIPT banks. To launch it from the editor click the run button or F5. While the script executes you can watch all variables and system parameters by dragging the *Variables View* and *Status View* panels as shown in Figure 8.

Monitor the indicators

You should also monitor the *Script running* indicator and *Script Info* in the status panel. If there are errors to be corrected refer to "Script debug" on page 16.

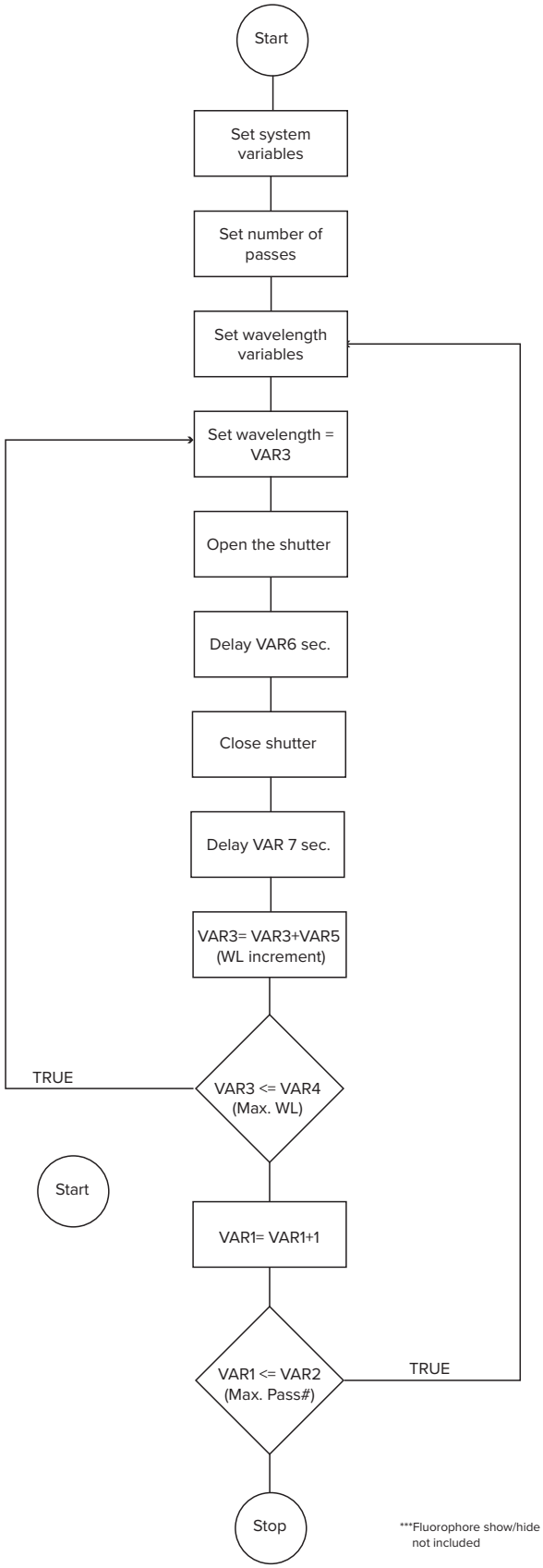
Figure 8 Executing a script and viewing variables



Flowchart

A flowchart of the script is shown in Figure 9 on page 23.

Figure 9 Flow chart of example script



3 Script elements

This chapter provides a description of the “Commands”, “Operators” on page 31, “Variables” on page 33, and “Array variables” on page 37 you can use to write and execute a CHROMASCRIPT with.

Commands

Set

Type
Command

Description
Assigns a value to any variable or array in conjunction with one of three operators: equal, plus or minus.

Syntax

set [*variable*] = [*value* | *variable*]

set [*variable*] + [*value* | *variable*]

set [*variable*] - [*value* | *variable*]

set [*array*] = [*WL (BW) (PW)* | *Preset]*

where:

variable is any variable as described in “Variables” on page 33.

array is any array as described in “Array variables” on page 37.

Preset is any of the 16 preset arrays described in “PRESET1 to PRESET16” on page 37.

value is any value that falls within the range of the variable.

Usage
Values assigned must be within the permitted range of the variable or array. When using *Set* to assign an array you must assign another array or up to three values: wavelength, bandwidth, and power. Note that Preset arrays cannot be set to the Filters array.

Examples

```
set Wavelength = 498
set Filter = PRESET1
set VAR1 + 20
set VAR2 - Bandwidth
set PRESET1 = 498 10 200
set PRESET12 = 530 20
set PRESET10 = 780
set PRESET5 = PRESET7
```

Delay

Type
Command

Description
This command pauses script execution for the duration of the value assigned.

Syntax

Delay [*value* | *variable*]

where:

value is time in seconds from 0.001 to 600 seconds in millisecond precision.

variable can be one of VAR1 to VAR16

Usage
You can insert a delay command anywhere in the script.

Examples
```
Delay 1.52
Delay VAR1
```

If

Type
Command

Description
This command is a conditional statement that compares a variable with a value or another variable. If the result is true, the program jumps. The command is always associated with a relational operator, a variable, and the *Goto* or *Return* commands. If the relation indicated in the command is true, the program jumps to another section of code as defined by the *Goto* or *Return* command.

Syntax

If [variable] = [*value* | *variable*] [Goto [line number] | Return]

If [variable] != [*value* | *variable*] [Goto [line number] | Return]

If [variable] < [*value* | *variable*] [Goto [line number] | Return]

If [variable] <= [*value* | *variable*] [Goto [line number] | Return]

If [variable] > [*value* | *variable*] [Goto [line number] | Return]

If [variable] >= [*value* | *variable*] [Goto [line number] | Return]

where:

value is any value that falls within the range of the variable.

variable is any variable as described in “Variables” on page 33.

Goto is an associated command as described in “Goto” on page 27.

Return is an associated command as described in “Return” on page 28.

Usage
You can use the *If* command to set a condition based on a comparison using relational operators. If the condition is true, the command can be set to either:

- Jump to another section/line of code using the associated *Goto* command.
- When the *Return* form of the command is used, the script interpreter returns the execution pointer to the next script line subsequent to the *Gosub* that initiated the subroutine.

Examples

```
If Wavelength >= 960 Goto 10
If Var1 != Var2 Return
```

Goto

Type
Command

Description
Goto causes an unconditional jump in the program pointer to another line in the script.

Syntax

Goto [line]

where:

line is a script line number greater or less than the current line.

Usage
Assigning a line number equal to the *Goto* command line number will stop execution of the script.

Example

```
Goto 10
```

Gosub

Type
Command

Description
Jump the script execution pointer to a subroutine section of code in order to perform another function before returning.

Syntax

Gosub [line]

where:

line is a script line number greater or less than the current line.

Usage
When *Gosub* is issued in the script, the program pointer jumps to a subroutine section of code to execute a function. The subroutine section of code must

include a *Return* command to return the script pointer to the section of code in the next line below the calling Gosub command.

Example

```
Gosub 20
```

Return

Type

Command

Description

Returns script control from a subroutine back to the line below the originating *Gosub* statement to continue the script's main execution.

Syntax

Return

If [variable] = [*value* | *variable*] Return

If [variable] != [*value* | *variable*] Return

If [variable] < [*value* | *variable*] Return

If [variable] <= [*value* | *variable*] Return

If [variable] > [*value* | *variable*] Return

If [variable] >= [*value* | *variable*] Return

where:

variable is any variable as described in section "Variables" on page 33.

value is any floating point value.

Usage

When the subroutine function is completed, you can return script control to the main body of the script using a simple *Return* statement in a single line. Alternatively you can use *Return* with the conditional command "If" on page 26 to jump the execution pointer back to main script body.

Examples

```
Return
If VAR3 > Power Return
```

Wait

Type

Command

Description

Use this command to pause script execution and wait for an external signal at the *Trig in* BNC port to signal the laser to continue executing the script.

Syntax

Wait [Trigger] [Low | High | HighLow | LowHigh]

where:

Trigger is a keyword used with the *Wait* commend.

Low continues execution when a low signal is detected.

High continues execution when a high signal is detected.

LowHigh continues execution when a low to high transition is detected.

HighLow continues execution when a high to low transition is detected.

Usage

Use the *Wait* command when you want to control script execution from an external application using a hardware signal. When the signal connected to the *Trig in* BNC port meets the command's signal requirement, the script execution continues. The keyword *Trigger* must always follow the Wait command.

Examples

```
Wait Trigger Low
Wait Trigger High
Wait Trigger HighLow
Wait Trigger LowHigh
```

Fluorophore

Type

Command

Description

Use this command to display or hide fluorophore graphs in the *Spectrum View* window.

Syntax

Fluorophore [Show | Hide] [Fluorophore]

where:

Show is a keyword used to display available fluorophore data in the Spectrum View window.

Hide is a keyword used to remove fluorophore data from the Spectrum View window.

Fluorophore is the name of a fluorophore dataset loaded into CONTROL.

Usage

Before issuing the command, the fluorophore dataset must be added to CONTROL using the *Fluorophore Manager* in the CHROMATUNE control panel. Execute the command in the script to load the fluorophore data set into the *Spectrum View* window prior to an emission command. You can view emission in the window to visually observe the result of the excitation. Before the next emission the dataset can be removed from the window and a new set loaded. In this way fluorophore data sets can be shown or hid to best suit the tuning requirement.

Example

```
Fluorophore Show NovaFluor_Blue_530
```

Stop

Type
Command

Description
Place a *Stop* command at the end of the main script code body to end execution of the script.

Syntax

stop

Usage
When the *Stop* command is encountered during script execution the program counter is reset and control is returned to the CONTROL interface to launch another script or other activity. If you do not include a Stop command, the interpreter by default will end execution and return control when it has no more lines to read.

Example
```
stop
```

Operators

+ and -

Type
Arithmetic operator

Description
Use the + operator to increase a variable and use the - operator to decrease a variable.

Syntax

Set [variable] + [value | variable]

Set [variable] - [value | variable]

where:

variable is any variable as described in section "Variables" on page 33.

value is any unsigned decimal or integer number.

Usage
Ensure not to exceed a variable range when using the operators.

Example

```
set wavelength + 20
set power + 100.20
set VAR1 + Bandwidth
```

NOTE: Increasing or decreasing a system variable resulting in an invalid value will cause an illegal value error during execution.

= , != , < , <= , > and >=

Type
Relational operators

Description
Use one of these operators to make a conditional comparison of a variable with another variable or value.

Syntax

See command "If" on page 26.

Usage
When one of the operators is used with the *If* command, the result is true based on implementing the following relational operator comparisons:

= when the left side variable is equal to the right side value or variable.

!= when the left side variable is NOT equal to the right side value or variable.

< when the left side variable is less than the right side value or variable.

<= when the left side variable is less than or equal to the right side value or variable.

> when the left side variable is greater than the right side value or variable.

>= when the left side variable is greater than or equal to the right side value or variable.

Example

“If” on page 26

Variables

Wavelength

Type
Filter variable

Description
The value stored in the *Wavelength* variable sets the emission center wavelength.

Data specifications

Size: 32 bit floating point

Unit: 1 nm, 0.1 nm

Range: ~ 400.0 to 1000.0 nm - per the laser's specifications.

Usage
Combine the *Wavelength* variable with commands, operators and other variables or values to control the center wavelength of the laser emission. Decimal places that are input beyond the first decimal place are rounded.

Example

```
Set Wavelength = 990.1
Set Wavelength - VAR2
If Wavelength <= 358.8 Return
```

Bandwidth

Type
Filter variable

Description
The value stored in the *Bandwidth* variable sets the emission bandwidth.

Data specifications

Size: 32 bit floating point

Unit: 1 nm, 0.1 nm

Range: ~ 5,0 to 50.0 nm

Usage
The minimum bandwidth increases from 5 to 10 nm as the center wavelength increases from its minimum. Combine the *Bandwidth* variable with commands, operators and other variables or values to control the bandwidth of the laser emission. Decimal places that are input beyond the first decimal place are rounded.

Example

```
Set Bandwidth = 20.2
Set Bandwidth + VAR1
If Bandwidth < 10 Goto 43
```

Power

Type
Filter variable

Description
The value stored in the *Power* variable sets the emission power.

Data specifications

Size: 32 bit integer

Unit: 1 µW

Range: 1 to ~ 62000[1]

Usage
The actual maximum power level when set to maximum will in practice be limited depending on the configured center wavelength and bandwidth.

Example
```
set power = 1000
```

VAR1-VAR16

Type
User-defined variable

Description
When creating a script, you can use the 16 user-defined variable VAR1 to VAR16 to hold any unsigned value desired.

Data specifications

Size: 32 bit floating point

Unit: none

Range: none

Usage
You can use VAR user-defined variables to hold counters, values, or any number useful to the script execution.

Examples
```
set VAR1 = 540
set VAR2 = 0.6
set VAR3 + VAR1
If VAR4 > VAR3 Goto 22
```

1. Refer to the laser's factory test sheet for actual specifications.

PowerConfig

Type
System variable

Description
Set the *PowerConfig* variable to select the laser power mode of operation.

Syntax

powerconfig [mode]

where *mode* is one of four power operation modes: 1 to 4

Usage
Set the *powerconfig* variable to 1 will cause the laser to operate with maximum power.

Setting the *powerconfig* variable to 2 will cause the laser to use internal mapping tables to set the output power based on the power setpoint and other filter settings.

Setting the *powerconfig* variable to 3 will cause the laser to first use the mapping tables to set a level and then make adjustments based on internal feedback until the setpoint power value is reached.

Setting the *powerconfig* variable to 4 will cause the laser to first use the mapping tables to set a level and then make adjustments based on internal feedback on a continuous basis to maintain the setpoint power value.

Example

```
Set PowerConfig 3
```

Shutterconfig

Type
System variable

Description
Set this variable to control the shutter position.

Syntax

Set ShutterConfig = [value | variable]

Set ShutterConfig + [value | variable]

Set ShutterConfig - [value | variable]

where:

value is an integer from 0 to 2, 0=closed, 1= open, 2 = automatic operation

variable is any variable as described in

Usage
You can set the shutter position with this variable. When the *ShutterConfig* variable is set to 0, the shutter closes and when set to 1, the shutter opens. If you set *ShutterConfig* to 2 in the script, the shutter will automatically open and

close based on the CHROMATUNE Module filters state. If the filters are adjusting its configurations, the shutter closes. When the filters are static and not in the process of a configuration change, the shutter opens.

Examples

```
Set ShutterConfig = 0
Set ShutterConfig = VAR2
```

TriggerConfig

Type

System variable

Description

Use this variable to allow the script to control the output signal level at the *Trig Out* BNC port on the rear of the laser.

Syntax

Set TriggerConfig = [value | variable]

where:

value is an integer from 0 to 2, 0=trig output low, 1= trig output high, 2 = trig output automatic

variable is any variable as described in section “Variables” on page 33.

Usage

You can use the *Trig out* BNC port to connect and synchronize the script with an external device or application. When the system controls the Trig Out port, the port is set high when the CHROMATUNE filters are in place i.e. the emission parameters are set, when the filters or settings are changing their setting, the port is set low. Setting this system variable in a script overrides the system control of the port.

Examples

```
Set TriggerConfig = 0
Set TriggerConfig = VAR8
```

Array variables

Filters

Type
System array

Description
You can set the *Filters* array within the script to immediately adjusts the system center wavelength, bandwidth and power in one command.

Syntax

set Filters [WL value] (BW value) (PW Value)

set Filters = PRESET[Number]

where:

WL is a center wavelength from ~400.0 to 1000.0

BW is a bandwidth from 5.0 to 50.0 nm

PW is 1 to ~62000 see “Power” on page 34

PRESET(Number) is any Preset array described in “PRESET1 to PRESET16”.

Usage
You can set three parameters of the laser in one line using the *Filters* array; however, you have the option to set the WL value only, the WL and BW only, or the WL, BW and PW. You can store multiple filter settings in the Preset arrays 1 to 16 and then configure the laser in one line by setting the *Filters* array to the values contained in a Preset array. The *Filters* array cannot be used with arithmetic operators.

Example

```
set Filters = 485
set Filters = 485 35
set Filters = 485 35 1000
set Filters = PRESET1
```

PRESET1 to PRESET16

Type
User-defined array

Description
You can store 16 arrays made up of three filter variables for use in setting the Filters array at any time during the script execution. Each PRESET1 to PRESET16 array can contain the filter variable values for wavelength, bandwidth, and power.

Syntax

set PRESET1 = [WL value] (BW value) (PW Value)

set Filters = PRESET1

Where:

WL is a center wavelength from ~400.0 to 1000.0

BW is a bandwidth from 5.0 to 50.0 nm

PW is 1 to ~62000 see “Power” on page 34

Usage

During script execution you can store up to 16 arrays which can be applied to the CHROMATUNE filter using the set *Filters* command. You cannot set a Preset array to the Filters array.

NOTE: You can see the contents of each PRESET1 to 16 in the *Filter presets view* tab at the bottom of the CHROMASCRIPT editor – see “Variable and array view tabs” on page 12.

Example

```
set PRESET1 = 675
set PRESET2 = 408 34
set PRESET5 = 856 22 12900
set Filters = PRESET16
```

A Sample scripts

Sequence two wavelengths

Description The script sets the laser to a 450 nm wavelength with a 10 nm bandwidth and 1 mW power setpoint. After 10 seconds the script changes the filter variables to 900 nm WL, 50 nm BW, and a 5 mW power setpoint.

Script

```
Set ShutterConfig = 1
Set powerconfig = 2
set Wavelength = 450
set bandwidth = 10
set power = 1000
delay 10
Set ShutterConfig = 0
set wavelength = 950
set bandwidth = 50
set power = 5000
Set ShutterConfig = 1
delay 10
Set ShutterConfig = 0
stop
```

Output Initially a 450 nm beam @ 10 nm BW is emitted with 1 mW of power. After 10 seconds the shutter closes and the laser tunes to 950 nm @ 50 nm BW and power set to 50% for 10 seconds where the shutter is opened,

Usage comments The script is intended to provide a two wavelength setup for an application.

Repetitive wavelength scanner

Description This script causes the laser to emit a range of center wavelengths over a given range. The wavelengths increase from a minimum to a maximum wavelength and for each emission the script increases the wavelength by an increment until the maximum wavelength is reached. You can repeat emitting the different wavelengths by setting the pass number greater than 1.

Script

```
Set PowerConfig = 4
Set ShutterConfig = 0
Set Power = 1000
Set Bandwidth = 20
# Load the Fluorophore dataset into the Spectrum View
Fluorophore Show NovaFluor_Blue_530
# Set the exposure and collection times
Set VAR6 = 0.1
Set VAR7 = 0.5
# Set the number of passes to test all wavelengths
Set VAR1 = 1
Set VAR2 = 3
# Set the wavelength variables
Set VAR3 = 430
Set VAR4 = 520
Set VAR5 = 10
# Configure the laser center wavelength
Set Wavelength = VAR3
# Emit for the exposure time and wait for collection
Set ShutterConfig = 1
Delay VAR6
Set ShutterConfig = 0
Delay VAR7
# Increment the WL and check if all WL are tested
Set VAR3 + VAR5
If VAR3 <= VAR4 Goto 17
# Increment the pass number and check if completed.
Set VAR1+1
If VAR1 <= VAR2 Goto 13
# Turn off the fluorophore dataset and stop the script
fluorophore Hide NovaFluor_Blue_530
Stop
```

Output Initially a 430 nm beam @ 20 nm BW is emitted at a 1 mW power setpoint. The wavelength loop increases each emission by VAR5 or 10 nm until 520 nm is surpassed, The process repeats entirely by the number set in VAR2.

Usage comments The script is described in Chapter 2 “Step by Step Programming example” on page 19.

Gosub demo

Description This script demonstrates the use of a Gosub command.

Script

```
# This script demonstrates the use of the Gosub command
# First set some wavelength boundaries and an increment
Set VAR1 = 450.50
Set VAR2 = 650.50
Set VAR3 = 7.5

# Go and do sweep based on the VAR1-3
Gosub 11
Stop

# Sweep from VAR1 to VAR2 in VAR3 steps
Set Wavelength = VAR1
Delay 0.5
Set VAR1 + VAR3
# Check all wavelengths swept? if true go to line 17 and return
If Wavelength < VAR2 Goto 11
Return
```

Output The wavelength variables are loaded and then the script jumps to the Set and If loop to sweep through the wavelengths. When done i.e. reached VAR2 or exceeded, the script returns and stops.

Usage comments A Gosub subroutine is useful if you need to perform a separate function outside the main code sequence. You can jump from the sequence and return with for example a new value from an action performed.

Using presets and Filters

Description This script sets 3 presets first which it then in sequence sets the FILTER system array to. Each Preset emits light according to its array contents and for a specified time in which the shutter is opened.

Script

```
# Close the shutter
Set ShutterConfig = 0
# Create three individual filter settings using Presets1 to 3
Set PRESET1 = 500 10 2000
Set PRESET2 = 600 15 1000
Set PRESET3 = 800 23 5000
# Set the power mode of operation to mode 3
Set PowerConfig = 3
# Set the CHROMATUNE filter to Preset1
Set Filters = PRESET1
# Open the shutter for 2 seconds
Set ShutterConfig = 1
Delay 2
Set ShutterConfig = 0
# Set the CHROMATUNE filter to Preset2
Set Filters = PRESET2
# Open the shutter for 3.5 seconds
Set ShutterConfig = 1
Delay 3.5
Set ShutterConfig = 0
# Set the CHROMATUNE filter to Preset3
Set Filters = PRESET3
# Open the shutter for 5.5 seconds
Set ShutterConfig  =1
Delay 3.5
Set ShutterConfig = 0
Stop
```

Output Light is emitted as follows:

- 500 nm wavelength, 10 nm bandwidth, 2 mW for 2 seconds
- 600 nm wavelength, 15 nm bandwidth, 1 mW for 2 seconds
- 800 nm wavelength, 23 nm bandwidth, 5 mW for 2 seconds

Usage comments Using the presets you can precisely set the output emission for multiple cases in a sequence.

Item:	800-642-03
Customer Revision:	1.0
NKT Photonics Revision:	1-0
Release Date:	09-2023

NKT Photonics A/S
Blokken 84, Birkerød-3460 Denmark

support@nktphotonics.com