ADVANCED CONTROL CENTER COMMUNICATION PROTOCOL

This document provides the necessary information to communicate with the Advanced devices through direct UART communication with the Advanced Control Center. In the following documentation, the modules connected to the control center are referred to as satellite modules or simply modules. General information on settings, syntax and procedures are provided, as well as all the commands needed.

UART Communication Protocol for Advanced Control Center / Version: 01.01.03 / Date: May 2025

Introduction

The Advanced Control Center is the brain of the Advanced range ecosystem. This device allows to:

- communicate with up to 25 modules via **only one connection, either USB or RS232, to the PC.** The Advanced ecosystem requires only one connection between the Advanced Control Center and the PC.
- create **sequences of commands** to control precisely and repeatedly modules in a system. No need to send consecutive commands manually thanks to the embedded sequencer of the Control Center.
- create a **remote feedback loop** between the pressure control of one module and the sensing values of another one (see Advanced Pressure Controller documentation for details).

Table of contents

Elveflow, plug & play microfluidics All rights reserved.

Serial connection settings

Baud rate : 115200

Data bits : 8

Stop bit : 1

Parity : none

Termination character : '\n'

Syntax

Command syntax

Command syntax addressed to Control Center	
char 0	'<' to start the query
char 1-5	command name
char 6	'?' to read, '!' to write
then ':' to start a value. Can iterate over many arguments	

Command syntax addressed to satellite modules	
char 0	'[' to start the query
char 1-7	Module Serial Number
char 8	':'
char 9-13	command name
char 14	'?' to read, '!' to write
then ':' to start a value. Can iterate over many arguments	

Error handling

In the answer to any command, the first information displayed after the read / write character is the error code '|xx|'. It is two characters that indicate the error code associated with the request. '00' means no error. Refer to the section Errors at the end of the document to check the possible error codes related to the Control Center module.

Elveflow, plug & play microfluidics All rights reserved.

Context

This section introduces the physical behavior underneath the Control Center and its interaction with modules. Although not required to operate the UART protocol, the information presented below may help diagnose situations without having to contact customer support.

Devices

Each device operates under the control of a microcontroller, equipped with a serial UART port for communication. The communication syntax is standardized which enables easy software integration of individual devices into anyone's code.

Each firmware present in each microcontroller has a firmware version associated with it. We advise you to update the modules' firmwares regularly to ensure access to the latest fixes and newly added functionalities.

Control Center to satellite module

The Control Center is equipped with five M12 connectors, facilitating UART connections with satellite modules. By cycling through standard UART commands, the Control Center can identify any module connected either before or after power up. When connected, the basic information is retrieved continuously by the Control Center with a PINGA command. When the module is disconnected, a timeout is detected and the device is removed from the Control Center memory.

The Control Center retains information about the serial number and type of all satellite modules connected and hence, it simplifies communication. In case of complex systems, the module in question is addressed directly, without having to specify the communication path. The information sent from the computer via the Control Center is directly transmitted to the right module. The syntax (Command syntax addressed to satellite modules) identifies the module to interact with, and the Control Center handles the 'routing' of the UART command back and forth.

This logic also applies when a Hub is connected to the Control Center. Any module connected to a Hub is recognized by the Control Center, allowing for command dispatch from the computer to the Control Center, then to the Hub, then to the module.

Speed VS robustness

Although communication speed is crucial, the Advanced range prioritizes communication robustness. Increasing the number of satellite modules does not impact the Control Center's performance or reactivity. However, using UART commands from a computer to extract information from each module will naturally take more time as the number of satellite modules increases. With a baud rate of 115k bauds, a practical guideline is that approximately 10 UART characters are transmitted per millisecond.

If speed is the most important criteria for your application, we recommend connecting each satellite module directly to the computer using an adapter. This approach allows you to take advantage of the native speed of 230k bauds.

Elveflow, plug & play microfluidics All rights reserved.

Procedures

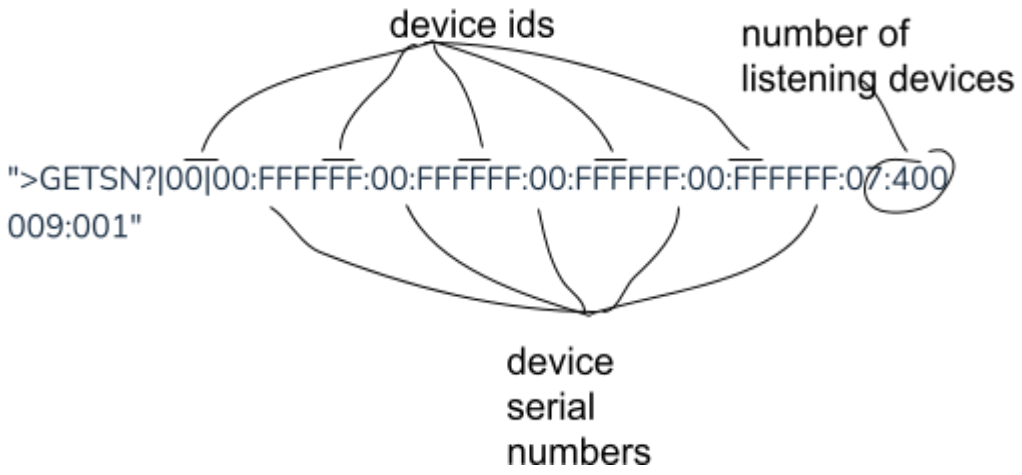
This section presents specific use cases using the Control Center commands that are summarized at the end of the communication protocol. Two specific features of the Control Center are highlighted to help you master their use easily: the **possibility to create a feedback loop between two different satellite modules** and the **embedded sequencer** of the Control Center.

All the commands listed in this section and additional ones are grouped together in the commands table.

How to handle satellite modules

The Control Center needs first to be identified via pnputils or other utilities to determine the appropriate COM port.

The Control Center identifies the type and serial number of the module directly connected to it via the **GETSN** command (cf Device types table at the end of this document).



In the example above, only 1 Pressure Controller is connected to the Control Center.

> *Note: The Control Center is addressed via "**<GETSN?**" whereas a Hub with SN "XXXXXX" attached to the Control Center will be addressed via "[XXXXXX:GETSN?"*

Once the list of appropriate devices is established, a general PING, the **PINGA**, can be sent on a regular basis to the module to stay informed of the most up to date information regarding those modules.

> *Example: for a Pressure Controller module of serial number A00122, the command to send is [A00122:PINGA?*

Elveflow, plug & play microfluidics All rights reserved.

Correspondence table for device types and serial numbers

Device type	SN's first character
Control Center	M
Hub	X
Pressure Controller	A, B, C, Y or Z (according to pressure range)
Sensor Hub	S
Valve Hub	V
RotaValve	R

Correspondence table for pressure range and serial numbers

Pressure controller range	SN's first character
0 mbar - 200 mbar	A
0 mbar - 2000 mbar	B
0 mbar - 8000 mbar	C
-900 mbar - 1000 mbar	Y
-900 mbar - 6000 mbar	Z

All commands for the modules can be sent either via the Control Center or directly via the adapter with the reading (?) / writing (!) option. When communicating to a module via the Control Center, it is necessary to address the module with its serial number.

> *Note: for example, to read the pressure on the channel of a Pressure Controller module of serial number A00122 via the Control Center, the command to send is "[A00122:PRESS?:00\n"*
>
> *To send the same command via an adapter directly to the Pressure Controller, use "<PRESS?:00\n".*

How to configure a remote feedback loop across two satellite modules

One of the major functions of the Control Center is to enable cards to communicate sensor values between them and establish a PI feedback loop.

Elveflow, plug & play microfluidics All rights reserved.

The connection between a Pressure Controller module and a Pressure Controller/Sensor Hub module is possible with the **CNECT** command. The command is addressed to the Pressure Controller module that performs the pressure control (PI control). The second serial number, provided as an argument, specifies the device (Pressure Controller or Sensor Hub) that transmits the sensor information.

> *Example: [A00123:CNECT!:01:S00543:0*
>
> *This command connects A00123 to S00543. Serial number A00123 is the Pressure Controller that performs the PI control based on the sensor data coming from the Sensor Hub S00543*

(See Pressure Controller communication protocol for more detailed information on how to set up a PI control and check the MFS user guide to learn more about flow control and feedback loop)

How to manage the integrated sequencing process

Two categories of functions are available for interacting with the embedded sequencer: **configuration** and **modification**.

A sequencer is an integrated process that allows the creation, saving and execution of a series of instructions for microfluidics systems made of Elveflow Advanced devices. Commands to modules are organized in relation to each other thanks to logical commands like if conditions, wait times, loops (all detailed afterwards).

Multiple **sequencers** are available to run parallel sequences. All functions interacting with a sequencer will **implicitly target** the one being in focus, determined by the SCHAN access function. There are 5 sequencer channels (0 to 4), each one with 128 steps. The 128 steps of a sequence are used to create simple or complex command lists. The minimum time period between 2 steps is 1 ms, which can be increased with a wait command. The device defaults to channel 0 at startup. Reading the channel value before using any sequencer functions is strongly recommended.

Configuration

The sequencer runs a loop to query the next step to apply. Since the loop iterates every millisecond, it is important to design each sequence step with an **inherent 1 millisecond wait**. Almost every, if not all, physical actions controlled by the Advanced range have reaction time of at least tens if not hundreds of milliseconds,

/!\ The sequencer's first step is identified by index 0.

General set up of a sequence

Here is how you would generally handle the set up of a sequence.

1. Select the sequencer channel (from 0 to 4, ie 5 slots available for 5 different sequences in memory)

command sent	response received
<SCHAN!:01	>SCHAN!|00|001:128

2. Set the different steps you want in your sequence with the commands beginning with S_ :

Among other possibilities, you can control a sequence 2 from a sequence 1 with the command **S_A_R**.

Elveflow, plug & play microfluidics All rights reserved.

The following example represents the **addition of a step to the sequence in focus** (sequence number 1, according to the previous command SCHAN here) that **pauses** (value 1 means pause, value 0 means stop, value 2 means start) the **sequence number 2**.

<S_A_R!:00**2**:00**1**	>S_A_R!|00|002:001

3. Optional: save the sequence in Control Center hard memory. Needed before setting the sequence to start up when the control center powers up (see next step).

command sent	response received
<EEPRS!	>EEPRS!|00|

4. Optional: set the sequence to automatically start when the control center powers up (first sequence needs to be saved in memory, see previous step).

command sent	response received
<STARS!:00	>STARS!|00|00

5. Manually start the sequence, for example to test it (here value 2 means run, and as we put the sequence 1 in focus with the first step, it will be the sequence 1 being played).

command sent	response received
<SEQCD!:02	>SEQCD!|00|02

Possible commands related to the sequencer

SEQCD + (int)state : read/write the status of the sequencer, 0 : stop, 1: pause, 2: run

SEQST read only : (int)current_step, (int)nb_of_steps, (int) errors, (int)sequencer.clock

SREAD read the sequencer

EEPRS saves into or loads from the memory

SREST resets the entire sequencer

ex : <SREST!:0

S_A_G adds a GOTO command arguments: index to go to, number of time to do it

ex: <S_A_G!:00:1000

S_A_W adds a WAIT command, argument: timeout(int) in ms

ex: <S_A_W!:50

Elveflow, plug & play microfluidics All rights reserved.

S_A_I adds an IF command : SN dev1, SN dev2 (“000000” if comparison with a fixed value), index to go if true, index to go if wrong, timeout (ms), comparison (0: <, 1:>), (float) fixed value, index dev1, index dev2

Note : index dev applies only for :

- *Pressure Controller modules: 0 means focus on regulator value, 1 means sensor value,*
- *otherwise it means sensor value channel (for Advanced Sensor Hub)*

ex : <S_A_I!:A00012:000000:09:08:1000:01:10.0:01:00

value role	Sn of device 1	Sn of device 2 or “000000” if comparison with a fixed value	index to go if true	index to go if wrong	timeout(ms)	comparison (0 means <, 1 means >)	(float) fixed value	index device 1	index device 2
example values	A00012	:000000	:09	:08	:1000	:01	:10.0	:01	:00
example value explanation	Sn of device 1	means comparison with fixed value	sequencer index to go to if true	sequencer index to go to if false	timeout in ms	means >	float fixed value	sensor value from device 1	NA because comparison is with fixed value and not another device

S_A_C adds a standard command to be applied to a satellite module.

ex : <S_A_C!:A00012:PRESS:00.00

Note : the index value in the direct access mode is removed here in anticipation of the Pressure Controller 1 channel only

STARS determines whether the sequencer should immediately run at startup

ex : <STARS!:01

Note : the sequencer needs to be saved to ensure this feature functions correctly.

NAMES sets the sequencer name for easier identification

ex : <NAMES?

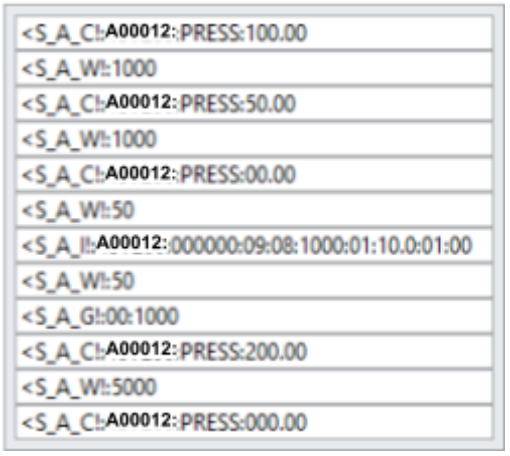
Each sequencer can be individually saved in memory, allowing for loading at startup. The **EEPRS** command facilitates both reading and writing these settings.

Elveflow, plug & play microfluidics All rights reserved.

Example

A cycle from 100 mbar to 0 mbar with one intermediate step (50 mbar) and a 1s delay for each step. When reaching 0 mbar, check the sensor value of the Pressure Controller module, if above 10.0 (a.u.), execute the final sequence otherwise start again for up to a 1000 times.

The final part of the sequence is a 200 mbar step applied for 5 seconds then the pressure is brought back to 0 mbar.



Elveflow, plug & play microfluidics All rights reserved.

List of Control Center commands

- W means ‘write’, ie set value, available for this command if ticked
- R means ‘read’, ie get value, available for this command if ticked
- ‘Mandatory arguments’ corresponds to mandatory arguments to attach to command in R mode
- ‘Arguments’ corresponds to additional mandatory arguments to attach to command in W mode after the ‘Mandatory arguments’
- ‘Arguments’ also corresponds to additional values in the answer resulting from the sent command (in W or R mode), after the ‘Mandatory arguments’

Parameter	Mandatory arguments	Arguments	W	R	Example query	Typical answer	Note
IDN		**char[10] :** device ID		X	<_IDN_?	>_IDN_?|00|CONTROLCEN	Get the name of the device returned values : - Cmd Type : <_IDN_? - Error Code : |00| - Device name : CONTROLCEN
DEVSN		**char[6] :** device serial number		X	<DEVSN?	>DEVSN?|00|M00072	Get the serial number of the device returned values : - Cmd Type : <DEVSN? - Error Code : |00| - Device Serial Number: MXXXXX
FIRMV		**char[9] :** firmware version		X	<FIRMV?	>FIRMV?|00|v01.00.00	Get the firmware version of the device

Elveflow, plug & play microfluidics All rights reserved.

							returned values : - Cmd Type : >FIRMV? - Error Code : |00| - Device name : vXX.XX.XX
VALVE	**int :** channel number (1 to 4)	**bool :** return true (1) for valve activated, false (0) for valve not connected	**X**	**X**	<VALVE?:1 <VALVE!:2:1	>VALVE?|00|01:00 >VALVE!|00|02:01	Get state or Control a single valve on a specific channel at a time returned values : - Cmd Type : >VALVE? - Error Code : |00| - channel : 00 or 01 or ... - state : 00 or 01
VALVS		**int :** register representing the valve states	**X**	**X**	<VALVS? <VALVS!15 <VALVS!:16	>VALVS?|00|0013 >VALVS!|00|0015 >VALVS!|C0|0016	Get state or Control all valves of the device via a single instruction (cf VALVS argument calculation method) returned values : - Cmd Type : >VALVE? - Error Code : |00| or |C0| - valves's state : [0000 to 0015]
GETSN		**int :** channel 1 type **char[6] :** channel 1 SN		**X**	<GETSN?	>GETSN?|00|06:X00008:00:FFFFFF:00:FF	Get the device type and the Serial Number of the

Elveflow, plug & play microfluidics All rights reserved.

		int : channel 2 type **char[6] :** channel 2 SN **int :** channel 3 type **char[6] :** channel 3 SN **int :** channel 4 type **char[6] :** channel 4 SN **int :** channel 5 type **char[6] :** channel 5 SN **int :** number of listening devices				FFFF:00:FFFFFF:00 :FFFFFF:000	satellite module connected to the Control Center returned values : - Cmd Type : >GETSN? - Error Code : |00| - sensor type on ch1: [0 to 44] - sensor serial number on ch1: XXXXXX - Same for each channel
SEQCD		**int :** new status	**X**	**X**	<SEQCD? <SEQCD!:2 <SEQCD!:3	>SEQCD?|00|001 >SEQCD!|00|002 >SEDCD!|I0|003	Get state or Set state of sequencer: returned values : - Cmd Type : >SEQCD? - Error Code : |00| or |I0| - state : 0 (stop) or 1 (pause) or 2 (run)
SEQST		**int :** target channel **int :** current step **int :** total steps **int :** number of errors **int :** timer from start (ms)		**X**	If total step = 500 : <SEQST?:0	>SEQST?|00|00:00265 :500:000000017: 000000000512	Get status of the sequencer of a channel returned values : - Cmd Type : >SEQST? - Error Code : |00| - target channel : [00 to 03] - current step: 00265 - total steps : 500

Elveflow, plug & play microfluidics All rights reserved.

							- number of errors : 000000017 - timer from start (ms) : 000000000512
S_A_G	**int :** new step to go to **int :** maximum command applications		**X**		If total step = 500 : <S_A_G!:000:01000	>S_A_G!|00|500:000:01000	Add a GOTO command to the sequencer. Once the maximum number of iterations is reached, this command will be skipped returned values : - Cmd Type : >S_A_G! - Error Code : |00| - total step: 500 - new step to go to : 000 - maximum command applications : 01000
S_A_W	**int :** time to wait		**X**		If total step = 500 : <S_A_W!:50	>S_A_W!|00|500:00050	Add a WAIT command to the sequencer returned values : - Cmd Type : >S_A_W! - Error Code : |00| - total step: 500 - time to wait (ms) ; 50
S_A_V	**int :** valve register		**X**		<S_A_V!:15	>S_A_V!|00|500:00015	Add a VALVS command to the sequencer

Elveflow, plug & play microfluidics All rights reserved.

							returned values : - Cmd Type : >S_A_V! - Error Code : |00| | - valves's state : [0000 to 0015]
S_A_I	**char[6]**: Serial Number device 1 **char[6]:** Serial Number device 2 **int** : new step if TRUE **int** : new step if FALSE **int** : timeout **int** : comparaison type **float** : compared value **int** : channel device 1 **int** : channel device 2		**X**		If total step = 500 : <S_A_I!:A00012:000000:09:08:1000:01:10.0:01:00	>S_A_I!:500:09:08:1000:01:10.0:01:00	Add a IF command to the sequencer Comparaison value : 0 = '<' 1 = '>' returned values : - Cmd Type : >S_A_I! - Error Code : |00| - total step: 500 - new step if TRUE : 09 - new step if FALSE : 08 - timeout (ms) : 1000 - comparison type : 00 or 01 - compared value : XX - channel device 1 : [00 to 03] - channel device 2 : [00 to 03]
S_A_C	**char[5]** : Command	**char[5] :** According to device	**X**		<S_A_C!:A00054:P	>S_A_C!|00|:xxx:003:000:A0	Add a device command to

Elveflow, plug & play microfluidics All rights reserved.

	char[6] : Serial Number .	command look at device's list of command specs (See note)			RESS:300	0544 • xxx : index of step of your sequence • 003 : cmd ID (See note) • 000: channel number of target device	the sequencer. Refer to the module communication protocol to determine available commands and their required arguments.
S_A_R	**int :** sequencer channel **int :** new state		**X**		<S_A_R!:002:001	>S_A_R!|00|002:001	Add a STATUS command to a specific sequencer new state value meaning : 0 : stop, 1: pause, 2: run returned values : - Cmd Type : >S_A_R! - Error Code : |00| - target sequencer: 002 - new state : 0 / 1 / 2
SREST			**X**		<SREST!	>SREST!|00|500:001:000000000512	Full reset of the sequencer (note : only affect the RAM) returned values : - Cmd Type : >SREST! - Error Code : |00| - total steps: 500

Elveflow, plug & play microfluidics All rights reserved.

							- sequencer errors : 001 - timer from start (ms) : 000000000512
SREAD	**int :** sequence step	**char[6] :** Serial Number **int :** command id **bool :** write/read **char[6] :** target step **float :** f_arg_1 **float :** f_arg_2 **int :** i_arg_1 **int :** i_arg_2 **int :** i_arg_3 **int :** i_arg_4 **int :** i_arg_5 **int :** i_arg_6		**X**	If total step = 500 : <SREAD?:265 <SREAD?:500	>SREAD?|00|265:S00176:0015:00:xxxxx:00000.00:00000.00:000:000:000:000:000:000 >SREAD?|I0|500:001 (total step & sequence error)	Read the details of the sequencer step returned values : - Cmd Type : >SREAD? - Error Code : |00| - target step: 265 - Serial Number : 09 - step type : 08 - write/read : 00 or 01 - ...(args of the specific step)
EEPRS			**X**	**X**	<EEPRS! <EEPRS?	>EEPRS!|00| >EEPRS?|00|	Save the sequence to memory or load it from memory returned values : - Cmd Type : >EEPRS? or ! - Error Code : |00|
SCHAN		**int** : target sequencer	**X**	**X**	If total step = 500 : <SCHAN!:01	>SCHAN!|00|001:500 >SCHAN?|00|001:500	Controls which sequencer is being selected, put in focus for all other sequence related commands

Elveflow, plug & play microfluidics All rights reserved.

					<SCHAN?		returned values : - Cmd Type : >SCHAN? or ! - Error Code : |00| - target sequencer : 001 - total steps : 500
STARS		**bool :** run sequence at startup	**X**	**X**	<STARS? <STARS!:00	>STARS?|00|01 >STARS!|00|00	Enable the sequence to run immediately at startup or check if it's enabled (note: the sequence obviously needs to be saved for this parameter to work) returned values : - Cmd Type : >STARS? or ! - Error Code : |00| - run sequence at startups : 00 or 01
NAMES		**char[10] :** sequencer name	**X**	**X**	<NAMES? <NAMES!:sequence1	>NAMES?|00|mysequence_name >NAMES!|L0|current_name	Get or set Sequencer name (however set is locked per default) returned values : - Cmd Type : >NAMES? or ! - Error Code : |00| or |L0| - sequencer name: XXXXXXXXXX

Elveflow, plug & play microfluidics All rights reserved.

NUKES			X		<NUKES!	>NUKES!|00|	Erase all sequences from device's memory returned values : - Cmd Type : >NUKES ! - Error Code : |00|
RESET						<RESET	reset firmware, i.e. soft reset of the device, i.e. simulates a power off-power on which resets all volatile variable (not saved in hard memory) - No value returned

VALVS argument calculation method

The VALVS command allows you to control the 4 valves individually in one command, i.e. you can set all 4 valves to individual positions via one command argument. To calculate the value of this argument, you should use the following :

1. Let's consider a binary number of 4 bits all set to 0. Here, bit 1 corresponds to valve 1 status (0 is disabled, 1 is activated), similarly for bit 2, bit 3 and bit 4. At this stage, this represents a configuration of valves all disabled :

bit 1	bit 2	bit 3	bit 4
valve 1	valve 2	valve 3	valve 4

Elveflow, plug & play microfluidics All rights reserved.

0	0	0	0

2. Now, set to 1 the bits corresponding to the valves you want to activate. For instance valve 2 and 3 :

bit 1	bit 2	bit 3	bit 4
valve 1	valve 2	valve 3	valve 4
0	**1**	**1**	0

3. Finally, to obtain the VALVS argument value, you should convert this binary number into a decimal one like bellow:

bit 1	bit 2	bit 3	bit 4
valve 1	valve 2	valve 3	valve 4
0	**1**	**1**	0
$0*2^3$	$1*2^2$	$1*2^1$	$0*2^0$

4. In this example, it gives:

$0 + 2^2+2^1 + 0 = 4 + 2 = 6$

5. This leads to the following command to send in order to activate valves 2 and 3 while having valves 1 and 4 disabled :

<VALVS!:6\n

Elveflow, plug & play microfluidics All rights reserved.

Constants

Device types

Device type	Meaning
0	No device
1-5	Reserved
6	Advanced Hub
7	Advanced Pressure Controller
8	Advanced Sensor Hub
9	Advanced Valve Hub
10	Advanced RotaValve

Errors

Error code	Meaning
00	No error
C0	Channel error: wrong channel requested
L0	Locking error: you do not have writing access to this parameter
I0	Impossible command: this query can not be processed
D0	Device error : wrong device cannot run command
NC	Not connected : the module concerned is not connected on the Control Center
P0	Pause error: this command can not be processed while pause is set to 1

Elveflow, plug & play microfluidics All rights reserved.

List of commands available with the S_A_C sequencer command

See compatible device communication protocol for more information.

Command	Compatible devices	ID
VALVS	Control Center, Valve Hub	012
VALVE	Control Center, Valve Hub	002
PRESS	Pressure Controller	003
SENSC	Pressure Controller	004
POSTN	Rotavalve	007
SETPI	Pressure Controller	009
SENCA	Pressure Controller, Sensor Hub	017
SENLT	Pressure Controller, Sensor Hub	018
SENRE	Pressure Controller, Sensor Hub	019
USRSO	Pressure Controller, Sensor Hub	013
SETMT	Pressure Controller, Sensor Hub	014
USRPL	Pressure Controller, Sensor Hub	010
ERLOG	Pressure Controller	011
PIRUN	Pressure Controller	009
WAVCT	Pressure Controller	015

Elveflow, plug & play microfluidics All rights reserved.

List of commands available with the SREAD sequencer command

See compatible device communication protocol for more information.

Command	Compatible devices	ID
VALVS	Control Center, Valve Hub	010
PRESS	Pressure Controller	004
SENSC	Pressure Controller	005
POSTN	Rotavalve	011
SETPI	Pressure Controller	012
SENCA	Pressure Controller, Sensor Hub	024
SENLT	Pressure Controller, Sensor Hub	025
SENRE	Pressure Controller, Sensor Hub	026
WAVET		009
SENSI		023
S_A_W		1000
S_A_G		1001
S_A_I		1002
S_A_V		1010

Elveflow, plug & play microfluidics All rights reserved.