ADVANCED VALVE HUB COMMUNICATION PROTOCOL

This document provides the information needed to communicate with the Advanced Valve Hub module through direct UART communication.

UART Communication Protocol for Advanced Valve Hub / Version: 01.01.00 / Date: August 2024

Introduction

The Advanced Valve Hub can be controlled directly via an adapter (intermediary between the module and the computer) or it can be controlled via an Advanced Control Center. If you are using the Valve Hub with the Advanced Control Center, refer to the Control Center communication protocol documentation for additional information.

The Advanced Valve Hub offers 16 valve connection ports, divided into 2 rows that can either be 12V or 24V (controlled by the switch next to each row).

Each port status (activated or not) can be controlled individually or all ports can be controlled together.

Table of contents

Serial connection settings

Baud rate: 230400

Data bits: 8

Stop bit: 1

Parity: none

Termination character: '\n'

Syntax

Command syntax

char 0: '<' to start the query

char 1 to 5: command name

char 6: '?' 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 following table to check the possible error codes related to the Valve Hub module:

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
P0	Pause error: this command can not be processed while pause is set to 1
B0	Argument value out of bound

List of 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'

Other auxiliary tables after this one.

Parameter	Mandatory arguments	Arguments	W	R	Number of characters returned	Example query	Typical answer	Note
VALVE	**int** : channel (1 to 16)	**bool:** valve state return true (1) for valve activated, false (0) for valve not connected	X	X	17	<VALVE?:4 <VALVE!:4:1	>VALVE?|00|04:01 >VALVE!|00|04:01	**Control a single valve channel at a time**
IDN		**str:** device name		X	22	<_IDN_?	>_IDN_?|00|VALVE_HUB_	

DEVSN		**str:** SN		X	18	<DEVSN?	>DEVSN?|00|V00001	
FIRMV		**str:** firmware version		X	21	<FIRMV?	>FIRMV?|00|v01.03.01	
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)**
VALVS		**int:** 16 bits register for all valve states (0 to 65535)	X	X	17	<VALVS? <VALVS!:65535	>VALVS?|00|65535 >VALVS!|00|65535	**Control all valves of the device via a single instruction (cf VALVS argument calculation method)**
PINGA				X	17	<PINGA?	>PINGA?|00|65535	**Status of valves taken as a whole and not individually (see VALVS command)**

STOP_		**int:** stop status	X	X	14	<STOP_? <STOP_!:1	>STOP_?|00|00 >STOP_!|00|01	**Stop in ESI** **0 by default** **Force all valve state to 0**
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)**

VALVS argument calculation method

The VALVS command allows you to control the 16 valves individually in one command, i.e. you can set all 16 valves to individual positions in one go via one command argument. To calculate the value of this argument, you should use the following :

1. Let's consider a binary number of 16 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	bit 5	bit 6	bit 7	bit 8	bit 9	bit 10	bit 11	bit 12	bit 13	bit 14	bit 15	bit 16
valve 1	valve 2	valve 3	valve 4	valve 5	valve 6	valve 7	valve 8	valve 9	valve 10	valve 11	valve 12	valve 13	valve 14	valve 15	valve 16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For readability reasons, the rest of the example is restricted to 4 valves. The method extends to 16 valves.

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
$0*2^3$	$\mathbf{1}*2^2$	$\mathbf{1}*2^1$	$0*2^0$

4. In this example, this results in :

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

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

For clarity, below are steps 3 to 5 in case of 5 valves :

6. 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	bit 5
valve 1	valve 2	valve 3	valve 4	valve 5
1	0	1	1	0
$1*2^4$	$0*2^3$	$1*2^2$	$1*2^1$	$0*2^0$

7. In this example, this results in :

$2^4 + 0 + 2^2 + 2^1 + 0 = 16 + 4 + 2 = 22$

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

<VALVS!:22\n>