OEM instruments - Control Center

Communication protocol

Introduction

This document provides the information necessary for communicating with the OEM devices through direct UART communication to the OEM Control Center.

Document status & Revision history

Version	Author	Release date	Comments
V01.00.00	David Bretaud	05/04/24	Launch version

Table of content

Serial connection settings

Baud rate : 115200

Data bits : 8

Stop bit : 1

Parity : none

Termination character : '\n'

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 Subcards

char 0 : '[' to start the query.

char 1-7: Subcard 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 an answer, after the read/write character, '|xx|' with xx 2 hexadecimal numbers are also sent and indicating the error code associated with the request. '00' means no error. The following error codes are found in the section Errors at the end of the document.

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 run in each microcontroller has a firmware version associated with it. We advise regularly updating the device firmware to ensure access to the latest fixes and newly added functionalities.

Control Center to daughterboard

The Control Center is equipped with five M12 connectors, facilitating UART connections with daughterboards. By cycling through standard UART commands, the Control Center can identify any daughterboards connected either before power up or in hotwire mode. When connected, the basic information is retrieved continuously by the Control Center with a PINGA command. When the daughterboard 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 of the daughterboard as well as its type. This enables it to send a command to a daughterboard from a computer connected to its Control Center. The new syntax identifies the daughterboard 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 daughterboard attached to the multiplexor is recognized by the motherboard, allowing for command dispatch from the PC to the Control Center, enabling a dispatch from the PC to the Control Center, then to the Hub, then to the daughterboard.

Speed VS robustness

Although communication speed is crucial, the OEM range prioritizes communication robustness. Increasing the number of daughterboards does not impact the motherboard's performance or reactivity. However, using UART commands from a PC to extract information from each daughterboard will naturally take more time as the number of daughterboards 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 daughterboard to the computer using an adapter. This approach allows you to take advantage of the native speed of 230k bauds.

How to

Handling subcards

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

The Control Center and Hub identifies the type and Serial number of the sub cards directly connected to them via the **GETSN** command.

Answers : ">GETSN?|00|**%02d:%.6s**:%02d:%.6s:%02d:%.6s:%02d:%.6s:%02d:%.6s:%03d%s"

%02d integer characterizing the subcard (2/6 : Hub, 3: OB1, 7, Pressure Controller, etc...)

%06s Serial Number

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 subcard to stay informed of the most up to date information regarding those cards.

Example : for a Pressure Controller card of serial number '48V200', the command to send is [48V200:PINGA?

Theoretically all commands from the subcards can be sent via the Control Center, with the reading/writing option, as well as all the normal arguments as if you were addressing the subcard directly

Note: for example, to read the pressure on the first channel of a Pressure Controller card of serial number '48V200', the command to send is [48V200:PRESS?:00

Configuring a remote feedback loop across two daughtercards

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.

The connection between a Pressure Controller daughtercard and a Pressure Controller/Sensor HubD daughtercard is possible with the **CNECT** command. The command is directed at the daughterboard controlling the pressure regulator. The second serial number provided as an argument specifies the device that transmits the sensor information.

ex : [48V300:CNECT!:01:48V200:0

Managing the integrated sequencing process

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

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), and 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's important to design each sequence step with an **inherent 1 millisecond wait**. Almost every if not all physical actions controlled by the OEM range have reaction time of at least tens if not hundreds of milliseconds,

Configuration

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

SEQST read only -> %03d:%03d:%03d:%010d : (int)current_step, (int)nb_of_steps, (int) errors, (int)sequencer.clock

SREAD read the sequencer

Modification

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

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 : for index dev, for Pressure Controller card it means 0-> reg value, 1->sens value, otherwise it means sens value channel (for Sensor Hub oem)

ex : <S_A_I!:48V200:000000:09:08:1000:01:10.0:01:00

S_A_C add a standard command to be applied to a subcard

ex : <S_A_C!:48V200: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

*N*ote : the sequencer needs to be saved to ensure this feature functions correctly.

NAMES sets the sequencer name for easier identification

ex : <NAMES?

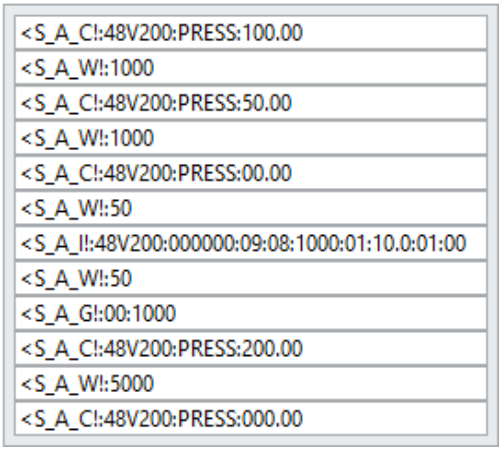
Modification

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

Example

A cycle from 100 mbar to 0 mbar with one intermediate step and a 1s delay for each step. When reaching 0 mbar, checks the sensor value of the Pressure Controller card, if above 10.0 (a.u.), executes the final sequence otherwise starts 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 down to 0 mbar.



List of Control Center commands

Parameter	Mandatory arguments	Arguments	W	R	Example query	Typical answer	Note
IDN		**char[10] :** device ID		X	<_IDN_?	>_IDN_?|00|MOTHERCARD	
DEVSN		**char[6] :** device serial number		X	<DEVSN?	>DEVSN?|00|M00072	
FIRMV		**char[9] :** firmware version		X	<FIRMV?	>FIRMV?|00|v01.00.00	
VALVE	**int :** channel number (1 to 4)	**bool :** return true for valve connected, false for valve not connected	X	X	<VALVE?:1 <VALVE!:0:1	>VALVE?|00|01:00 >VALVE!|00|00:01	Control a single valve channel at a time
VALVS		**int :** register representing the valve states	X	X	<VALVS?	>VALVS?|00|13	Control all valves of the device via a single instruction (cf correspondence table further below)
GETSN		**int :** channel 1 type **char[6] :** channel 1 SN **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		X	<GETSN?	>GETSN?|00|06:X00008:00:FFFFFF:00:FFFFFF:00:FFFFFF:00:FFFFFF:000	Get the device type and the Serial Number of the daughter cards connected to the Control Center

SEQCD		**int :** new status	**X**	**X**	<SEQCD? <SEQCD!:02	>SEQCD?|00|01 >SEQCD!|00|02	0 : stop, 1: pause, 2: run
SEQST		**int :** current step **int :** total steps **int :** number of errors **int :** timer from start (ms)		**X**	<SEQST?	>SEQST?|00|00265:500:0000 00017:000000000512	Returns the status of the sequencer
S_A_G	**int :** new step to go to **int :** maximum command applications		**X**		<S_A_G!:000:1000	>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 = **total number of steps : new step to go to : maximum command applications**
S_A_W	**int :** time to wait		**X**		<S_A_W!:50	>S_A_W!|00|500:00050	Add a WAIT command to the sequencer returned values = **total number of steps : time to wait**
S_A_V	**int :** valve register		**X**		<S_A_V!:15	>S_A_V!|00|500:00015	Add a VALVS command to the sequencer returned values = **total number of steps : valve**

							register
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**		<S_A_I!:48V200:000000:09:08:1000:01:10.0:01:00	>S_A_I!:48V200:000000:09:08:1000:01:10.0:01:00	Add a IF command to the sequencer Comparaison value : 0 = '<' 1 = '>'
S_A_C	**char[5]** : Command **char[6]** : Serial Number .	**(See note)**	**X**		<S_A_C!:PRESS:A00544:300	>S_A_C!:PRESS:A00544:003:000:xxxxxx	Add a device command to the sequencer. Refer to the daughter card 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 the sequencer
SREST			**X**				Full reset of the sequencer (note : does not affect what is saved in memory)

SREAD	**int :** sequence step	**char[6] :** Serial Number **int :** command id **bool :** write/read **char[6] :** target_arg **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**	<SREAD?:265	>SREAD?|00|265:S00176:0015:00:xxxxxx:00000.00:00000.00:000:000:000:000:000:000	Read the details of the sequencer step
EEPRS			**X**	**X**	<EEPRS! <EEPRS?	>EEPRS!|00| >EEPRS?|00|	Save the sequence to memory or load it from memory
SCHAN		**int** : sequencer channel to select	**X**	**X**	<SCHAN!:01	>SCHAN!|00|001:5500	Controls which sequencer is being selected for reading and writing returned values = **sequencer channel to select : total number of steps**
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 (note: the sequence obviously needs to be saved for this

							parameter to work)
NAMES		**char[10] :** sequencer name	**X**	**X**	<NAMES? <NAMES!:sequence1	>NAMES?|00|mysequence >NAMES!|00|sequence1	Sequencer name
SGETE	**int :** previous_error	**int :** error ID **int :** sequence step **int :** relative timer **int :** abs time, hour **int :** abs time, mins **int :** abs time, sec		**X**			
RESET							Firmware reset of the instrument

Constants

Device types

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

VALVS correspondence table

VALVS argument value in integer (to be used in UART command)	corresponding VALVS argument value in binary	Meaning			
		valve 0	valve 1	valve 2	valve 3
0	0000	off	off	off	off
1	0001	off	off	off	ON
2	0010	off	off	ON	off
3	0011	off	off	ON	ON
4	0100	off	ON	off	off

5	0101	off	ON	off	ON
6	0110	off	ON	ON	off
7	0111	off	ON	ON	ON
8	1000	ON	off	off	off
9	1001	ON	off	off	ON
10	1010	ON	off	ON	off
11	1011	ON	off	ON	ON
12	1100	ON	ON	off	off
13	1101	ON	ON	off	ON
14	1110	ON	ON	ON	off
15	1111	ON	ON	ON	ON

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 daughtercard concerned is not connected on the Control Center
P0	Pause error: this command can not be processed while pause is set to 1

List of commands available with the S_A_C sequencer command

Command list

See compatible device communication protocol for more information.

Command	Compatible devices
VALVS	Control Center, Valve Hub
VALVE	Control Center, Valve Hub
PRESS	Pressure Controller
SENSC	Pressure Controller
LISTN	Pressure Controller
POSTN	MuxD
SETPI	Pressure Controller
SENCA	Pressure Controller, Sensor Hub
SENLT	Pressure Controller, Sensor Hub
SENRE	Pressure Controller, Sensor Hub
USRSO	Pressure Controller, Sensor Hub
SETMT	Pressure Controller, Sensor Hub
USRPL	Pressure Controller, Sensor Hub
ERLOG	Pressure Controller
PIRUN	Pressure Controller
WAVCT	Pressure Controller