23.0331

Clarity™

ADVANCED CHROMATOGRAPHY SOFTWARE

UNI RUBY

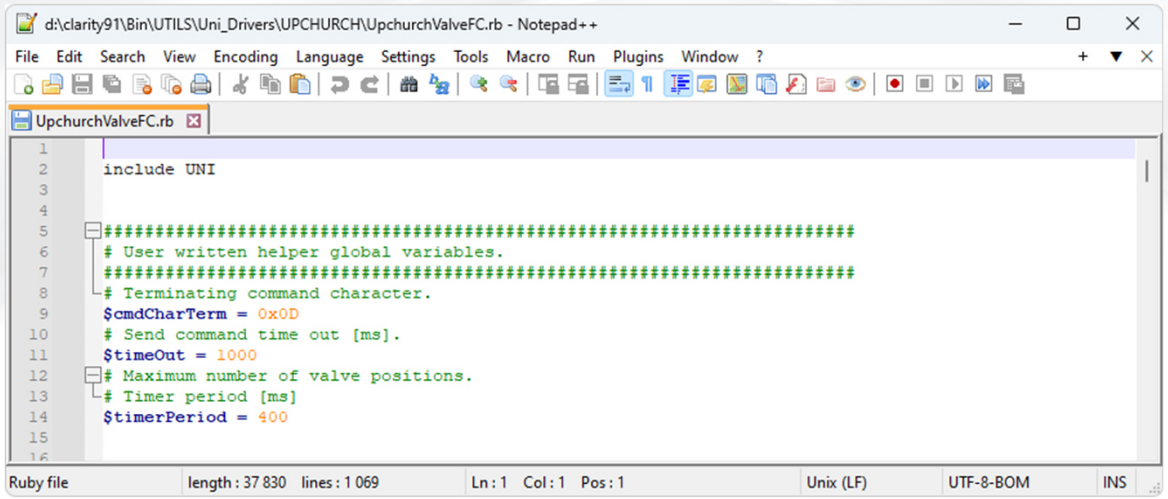
DEVELOPMENT

DataApex

MD1

UNI RUBY

- Text file with defined structure
- Can be edited in text editor (Notepad++ recommended)
- Easy to learn Ruby language https://www.ruby-lang.org/en/
  - ruby gems not supported ☹
- Simple instruments are made with UNI Ruby (pump, thermostat, detector)
- C++ not needed (other than UNI Ruby script development the SDK needs to be used)
- You need to know the protocol (commands issued on HW)



Snímek 2

MD1 Přepsána řádka s C++ not needed - překlepy a nevím co tím chtěl autor říct
Mentlík Daniel; 2023-12-19T12:34:35.477

MD1

SCRIPT WORKFLOW

- Lots of code
- 2 classes (Device and SubDevice)
- You can‘t delete framework methods and classes (commented as „Method expected by framework“).

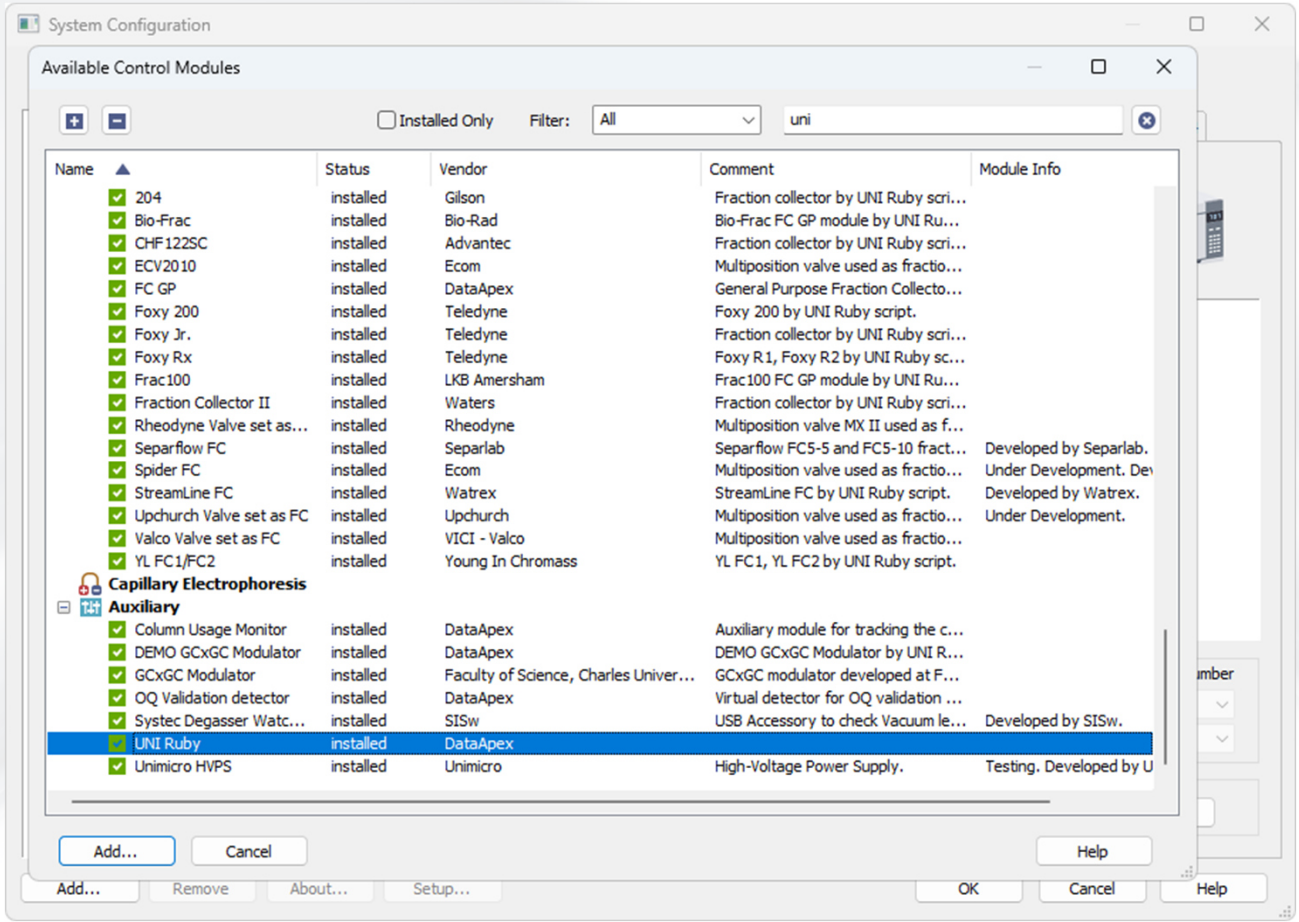
- Common methods are Init, InitCommunication, InitConfiguration, CmdTimer, CmdSendMethod, ...
- Common methods are in each script, but some devices have specific methods in addition

- Global variables at the begining ($) – script scope
- Device variables inside device class (@) – class scope
  - Must be initialized in the CmdOpenInstrument function

Snímek 3

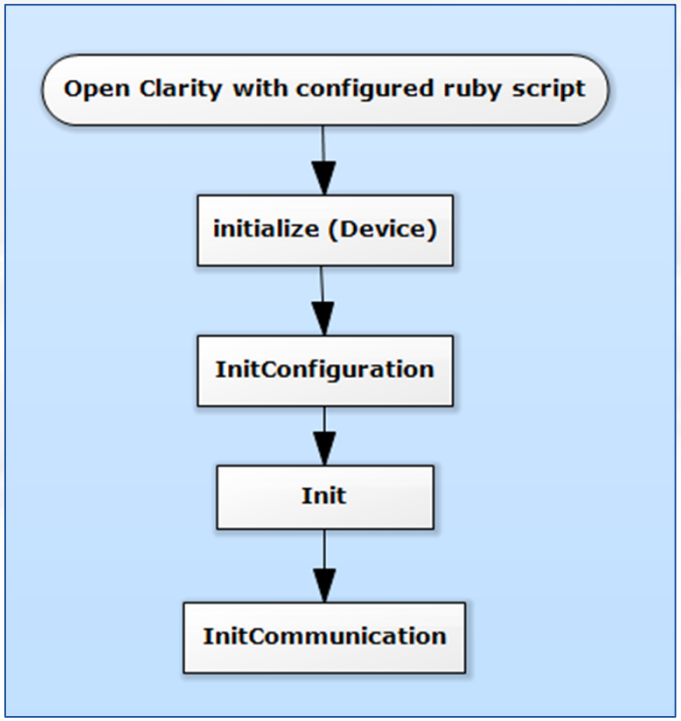
MD1 změněny uvozovky u komentáře, čtyřtečka na trojtečku
Mentlík Daniel; 2023-12-19T12:35:43.999

OPEN SCRIPT IN CLARITY



WHAT'S HAPPENING...

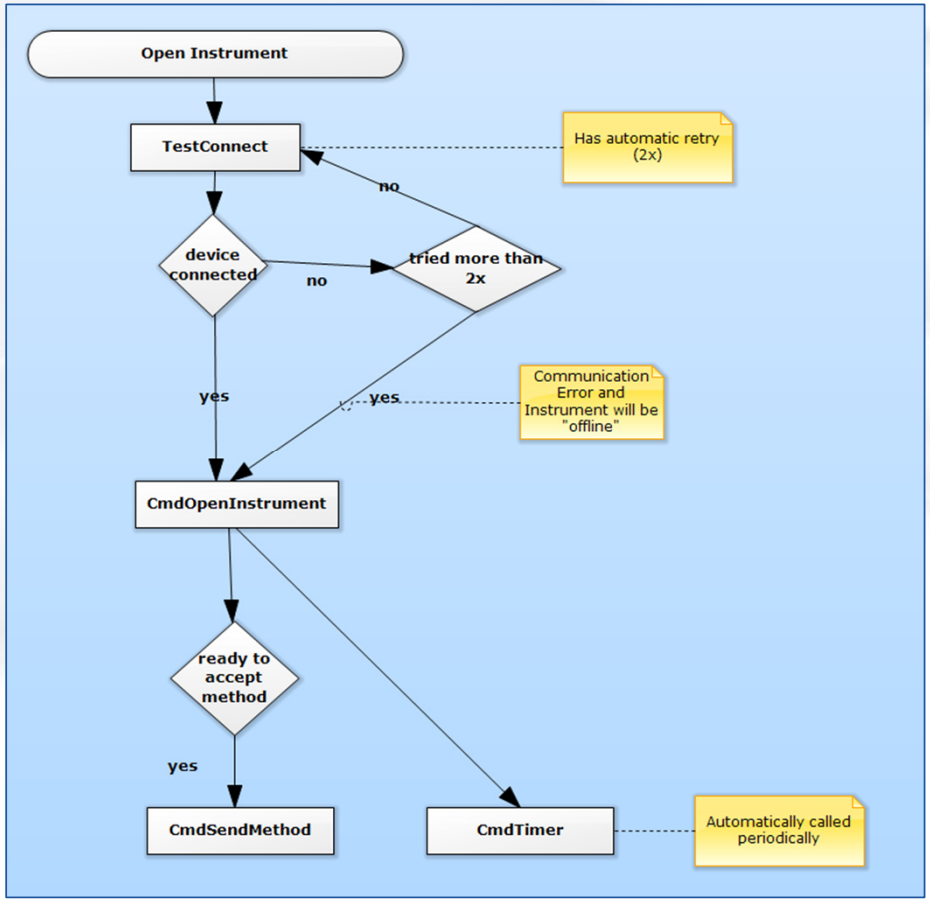
...WHEN YOU OPEN CLARITY



When you modify a script file, you should restart Clarity to see changes.

WHAT'S HAPPENING...

...WHEN YOU OPEN THE INSTRUMENT



MD1

CLARITY OBJECTS IN SCRIPT

- Configuration()
  - HW parameters, can't be changed after your open instrument (changes only from Clarity Configuration dialog)
  - Autodetection, if CmdAutoDetect implemented
    - Device name, Serial number
    - Maximal flow, pressure
    - Other options
- Monitor()
  - State of Device
  - Buttons for Device control (stop pump, perform AutoZero, ...)
- Method()
  - Parameters that are set on the device when CmdSendMethod() is called

Snímek 7

MD1 mezera před trojtečkou, přepis itemu u Method()

Mentlík Daniel; 2023-12-19T12:37:08.011

CLARITY OBJECTS IN SCRIPT

- Create UI Items
  - TextBox *<object>.AddString(...) or .AddInt(...) or .AddDouble*
  - CheckBox *.AddCheckBox*
  - ComboBox *.AddChoiceList and .AddChoiceListItem*
  - Button *.AddButton*
  - Table *.AddTable and .AddTableColumn* (only allowed in Method)
- Read values from UI Items with *.GetString(...), GetInt(...), GetDouble(...), GetTableColumnValues(...)*
  - For ComboBox use *GetString*
  - For CheckBox use *GetInt*
  - For TextBox, it depends on definition

OBJECTS TYPE

MD1

Once you add object to a Configuration, Method or Monitor, a type of the object is stored within Clarity methods/configurations. You should not change the object type after releasing the script, that will break compatibility with already created configuration and method files in Clarity.

e.g.

First version of ruby script:

```
Method().AddString(„myObject“,…)
```

Later, when you change string to double:

```
Method().AddDouble(„myObject“,…)
Method().GetDouble(„myObject“)
```

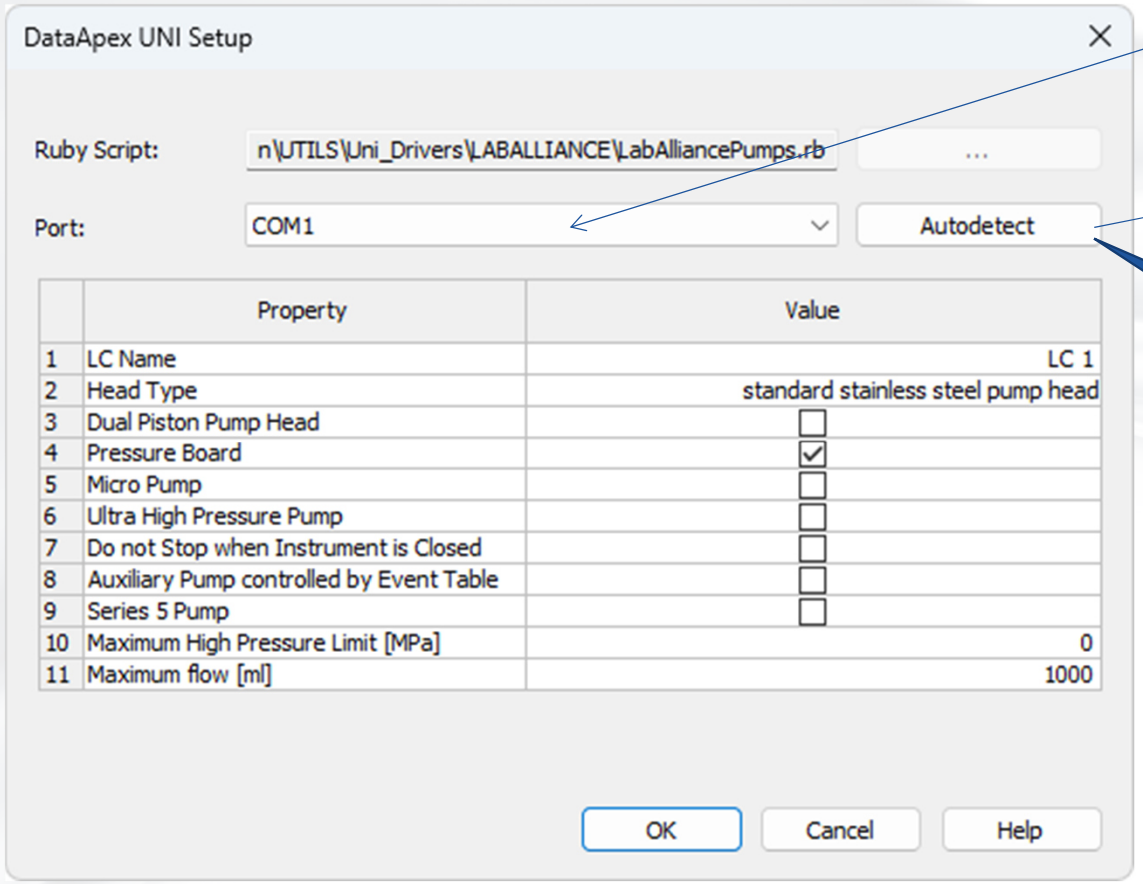
=> ruby exception

When you need to change the type of existing object, it is better to create a brand new object with different ID and deprecate the old item (stop reading it, saving it, ...).

Snímek 9

MD1 zarovnání a text prvního odstavce, faktická správnost v kódovém příkladu
Mentlík Daniel; 2023-12-19T12:42:19.535

CONFIGURATION



InitCommunication

CmdAutoDetect

Hide with *SetHideAutoDetect (true)*

InitConfiguration

- Values can be validated with user-defined function

VALIDATING ITEMS

- You want to check some user input in Configuration table
  Configuration().AddString(..., ..., ..., 'VerifyDetectorName')
- Validation function
  - two parameters in declaration
    - *uiitemcollection, value*

```
#############################################################################
# User written method.
#
# Validates length of LC name.
# Validation function returns true when validation is successful otherwise
# it returns message which will be shown in the Message box.
#############################################################################
def VerifyLCName(uiitemcollection,value)
    if (value.length >= 32)
        return t("NAME_LONG")
    end
    return true
end
```

- When the value is out of range, it returns message
- Messagebox is shown automatically in Clarity from validation functions

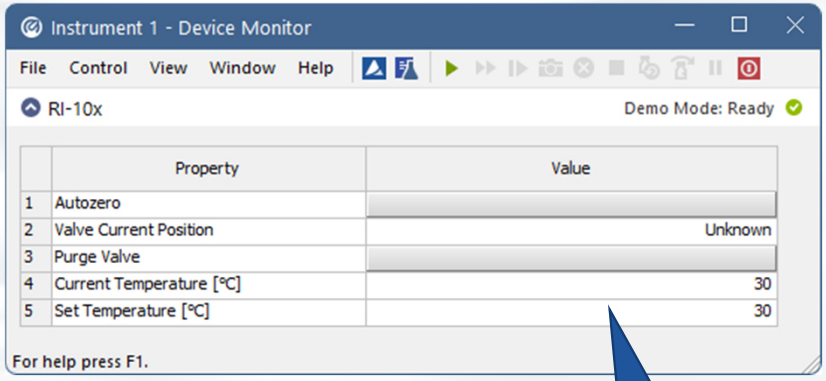
DEVICE MONITOR

- Display status of device (temperature, flow, pressure)

- Define in *Init()*
  - *Monitor().Add....*

- Data are typically read from device in *CmdTimer()*
- Change values in Monitor with SetXyz methods
  - ***Monitor().Set****String(ID, value)*
  - *Monitor().SetInt(ID, x)*

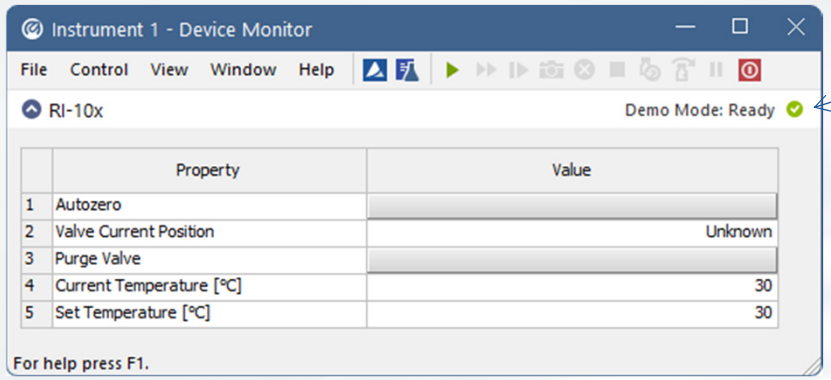
- Call *Monitor().Synchronize()* at the end of *CmdTimer()*



MD1 zarovnání, přepis Set metod

Mentlík Daniel; 2023-12-19T12:46:46.165

DEVICE MONITOR



Device state
Monitor().SetReady(bool)
Monitor().SetStateName(...)
Monitor().SetRunning(bool)

Items defined in *Init*

- Monitor can be hidden
  - *SetHideMonitor(bool)*

DEVICE MONITOR

- Button with action
  - *Monitor().AddButton('ID', 'label', 'buttonText', 'buttonAction')*

Here, all parameters are passed as string, even the name of buttonAction.

buttonAction is your function in ruby script

MD1

DEVICE MONITOR

- The **buttonAction** in the monitor is performed immediately and in parallel with the code in the CmdTimer, which can affect it executing code in the CmdTimer function.
- The solution is to perform only a single buttonAction in one iteration of the CmdTimer.

```
class Device < DeviceWrapper

    def Init
        # Add "Startup Sequence" button to the Device Monitor
        Monitor().AddButton('StartupSequenceButton', t("Startup Sequence"), t("ON"), 'PerformStartupSequence')
    end

    ##########################################################################
    #
    # Perform Startup sequence button in monitor
    # @m_bPerformStartupSequence = true when button is pressed
    ##########################################################################
    def PerformStartupSequence
        @m_bPerformStartupSequence = true
    end


    ##########################################################################
    # Method expected by framework.
    #
    # Periodically called function which should update state
    # of the sub-device and monitor.
    # Returns true when successful otherwise false.
    ##########################################################################
    def CmdTimer
        # In this iteration perform only "Startup Sequence"
        if @m_bPerformStartupSequence
            CmdStartupSequenceOnOff()
            @m_bPerformStartupSequence = false
            # exit from this iteration
            return true
        end


        # another code in the next iteration

    end
end
```

Snímek 15

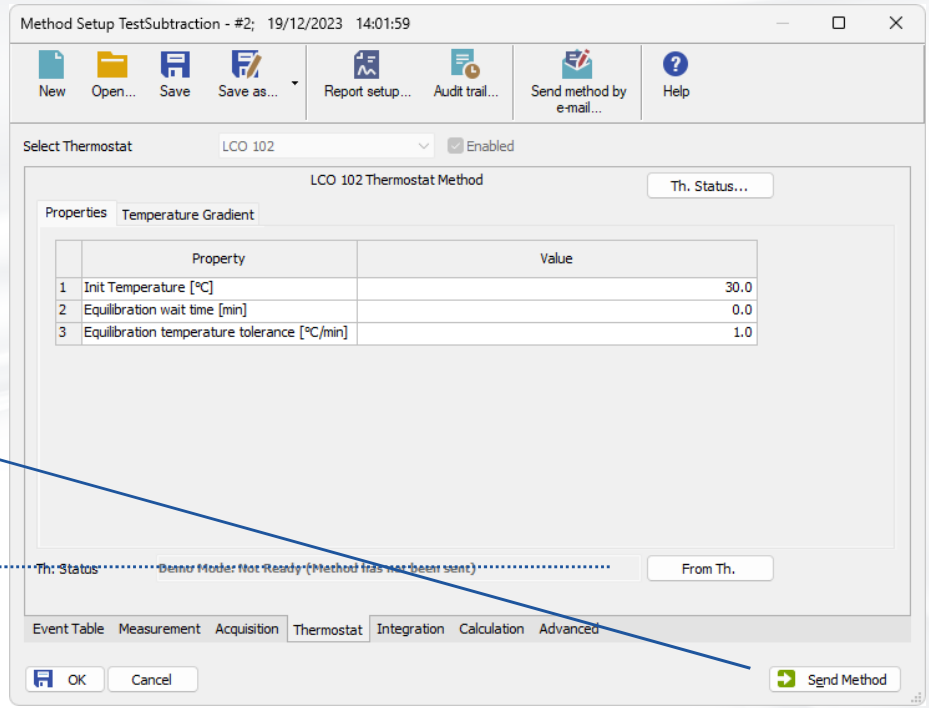
MD1 drobné textové úpravy

Mentlík Daniel; 2023-12-19T12:49:15.231

METHOD

MD1

- Call *Method().Add....* in Init() to define method table
  - Similar to Configuration() and Monitor()
- or Hide whole method with
  - SetHidePropertyPage(true)



Framework functions:

- *CmdSendMethod()*
  - Send parameters to HW
  - *Method().Get....*
- *CmdLoadMethod()*
  - Read HW settings from device
  - hide button with SetHideLoadMethod(true)
- *CheckMethod(situation,method)*
  - Checking whether method is correct as a whole
  - can get information from other places, e.g. vial number from sequence table, ...

Snímek 16

MD1 Změněn obrázek (šipka u Load Method ukazovala na nesouvisející tlačítko), faktická chyba u schovávání metody, přidáno schování vyčítání metody, upravena poslední věta

Mentlík Daniel; 2023-12-19T13:12:06.506

THE INVOKE ORDER OF START FUNCTIONS

1. CmdStartSequence
2. CmdStartRun
3. CmdStartAcquisition

COMMUNICATION WITH HW

- Communication protocol
  - Parameters of communication, set of commands used to control HW
- Communication pipe
  - Serial (rs232)
  - TCP
  - USB

COMMUNICATION WITH HW

- InitCommunication()
  - Set number of pipe configurations for communication. In our case one - serial communication.

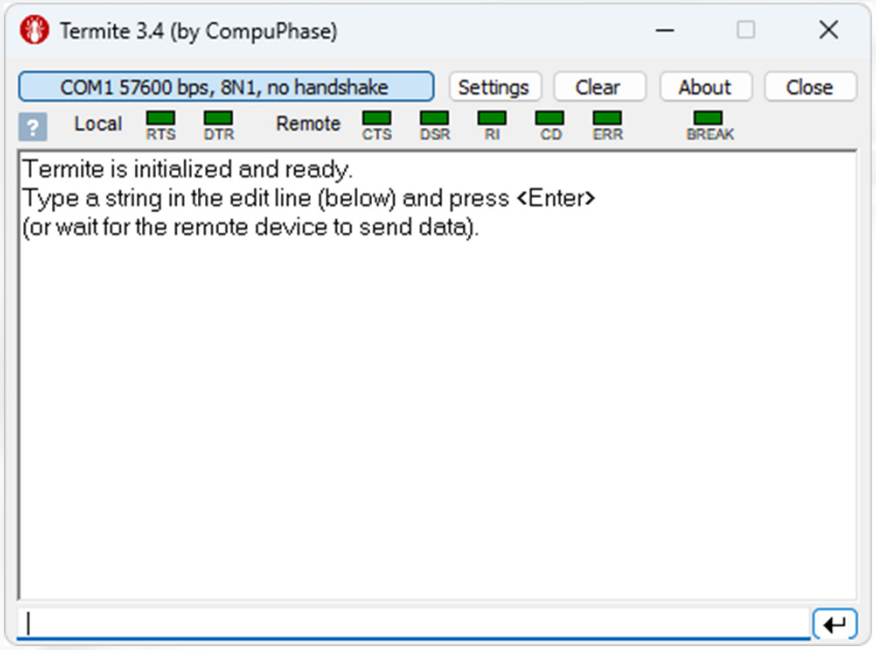
```
Communication().SetPipeConfigCount(1)
```

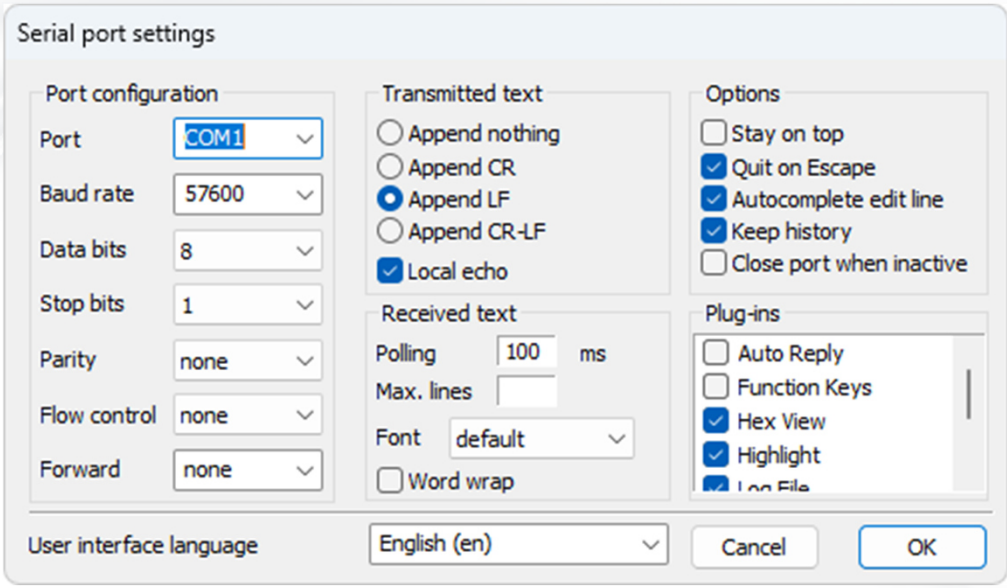
  - Set type for created pipe configuration.
    - EPT_SERIAL Serial pipe.
    - EPT_TCP TCP pipe.
    - EPT_UDP UDP pipe.
    - EPT_GSIOC GSIOC pipe.
    - EPT_USBFTDI USB FTDI chipset pipe. SetUSBDeviceName().
    - EPT_USBXP USB SiUSBXp chipset pipe. SetUSBDeviceName().

```
Communication().GetPipeConfig(0).SetType(EPT_SERIAL)
Communication().GetPipeConfig(0).SetBaudRate(19200)
Communication().GetPipeConfig(0).SetParity(NOPARITY)
Communication().GetPipeConfig(0).SetDataBits(DATABITS_8)
Communication().GetPipeConfig(0).SetStopBits(ONESTOPBIT)
```

COMMUNICATION WITH HW

- Termite
  - **a simple RS232 terminal**
  - http://www.compuphase.com/software_termite.htm
  - Test connection settings, answers from HW





COMMUNICATION WITH HW

- Create and Send Command

```
cmd = CommandWrapper.new(self)
cmd.AppendANSIString(" ")
if (cmd.SendCommand($timeOut) == false)
    return false
end
```

- Read Answer

```
answer = cmd.ParseANSIString()
```

  - Parsed string when successful otherwise false!
  - Parsed string is then removed from cmd
  - other Append and Parse functions - see UNI Ruby documentation

    AppendANSIInt(x)

    AppendANSIDouble(...)

    AppendANSIChar

COMMUNICATION WITH HW

- Example – send command and parse answer

```
cmd = CommandWrapper.new(self)
cmd.AppendANSIString(„send me ok 100")
if (cmd.SendCommand($timeOut) == false)
    return false
end

ok = cmd.ParseANSIString(„ok“)
space = cmd.ParseANSIString(„ “)
number100 = cmd.ParseANSIInt()
thisIsFalse = cmd.ParseANSIString(„x“)
```

Expecting answer: „ok 100“

The last variable will be false, because cmd buffer contains nothing at the moment

COMMUNICATION WITH HW

- Example 2 – send command without expecting answer

```
cmd = CommandWrapper.new(self)
cmd.AppendANSIString("I1S0")
cmd.AppendANSIChar($cmdCharTerm)

if (cmd.SendCommand(0) == false)
    return „command not sent“
else
    return „command sent“
end
```

COMMUNICATION WITH HW

MD1

How UNI ruby gets data from HW

- UNI Ruby automatically checks for incoming data (sequence of bytes)
- Data are then passed into FindFrame function for completion

- ***FindFrame (dataArraySent, dataArrayReceived)***
  - dataArraySent – byte array containing your command
  - dataArrayReceived – byte array containing incoming data

  - Typical implementation - check leading & ending sequence of characters, calculate check sum of data and then mark beginning and end of the frame with framework functions:

```
SetFrameStart(startIndex)
SetFrameEnd(endIndex)
```

  - If sentByteArray is nil – it means you have not send command to HW and it has sent unrequested data
  - Return false, if data are incomplete (e.g. ending character is not found)

Snímek 24

MD1 překlep v poslední větě, faktické pojmenování parametrů.
Mentlík Daniel; 2023-12-19T13:22:04.403

COMMUNICATION WITH HW

- *Example:*

FindFrame(dataArraySent, dataArrayReceived)

```
nEndFrameIdx = dataArrayReceived.index($cmdCharTerm)
if (nEndFrameIdx == nil)
  return false
End
# Set frame start and end indexes!
SetFrameStart(0)
SetFrameEnd(nEndFrameIdx)
return true
```

Frame is complete, if the data received from HW contains a terminating character.

Define $cmdCharTerm as global variable

- Search character in byteArray with .index (or .rindex) function
  - Example – find index of ‚a‘

```
nEndFrameIdx = dataArrayReceived.index(‘A‘.ord)     - correct
nEndFrameIdx = dataArrayReceived.index(0x41)        - correct (hex code)

nEndFrameIdx = dataArrayReceived.index(‘A‘)         - wrong
you can‘t search char in byte array directly (convert array to string or
parameter to code)
```

COMMUNICATION WITH HW

ASCII TABLE

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

COMMUNICATION WITH HW

- You can convert received data to string representation
  - Use *.pack("c*")* => Pack every element as an 8-bit signed integer.

```
str = dataArrayReceived.pack("c*")
```
=> „Hello, world“

  - If you expect non-ascii characters – use parameter "**C***"
    - C* is for 8-bit unsigned (unsigned char)

- Unicode characters
  - .pack(„U*“)

COMMUNICATION WITH HW

- Implement IsItAnswer(dataArraySent, dataArrayReceived)
  - If IsItAnswer() returns false, then frame is not passed to command waiting for the answer and frame is passed into ParseReceivedFrame() for processing as unrequested data
  - common scenarios:

    a) All data from HW can be your answer

```
return true
```

    b) Pass answers to your questions only

```
if (dataArraySent == nil || dataArraySent.length == 0)
    return false
end
```

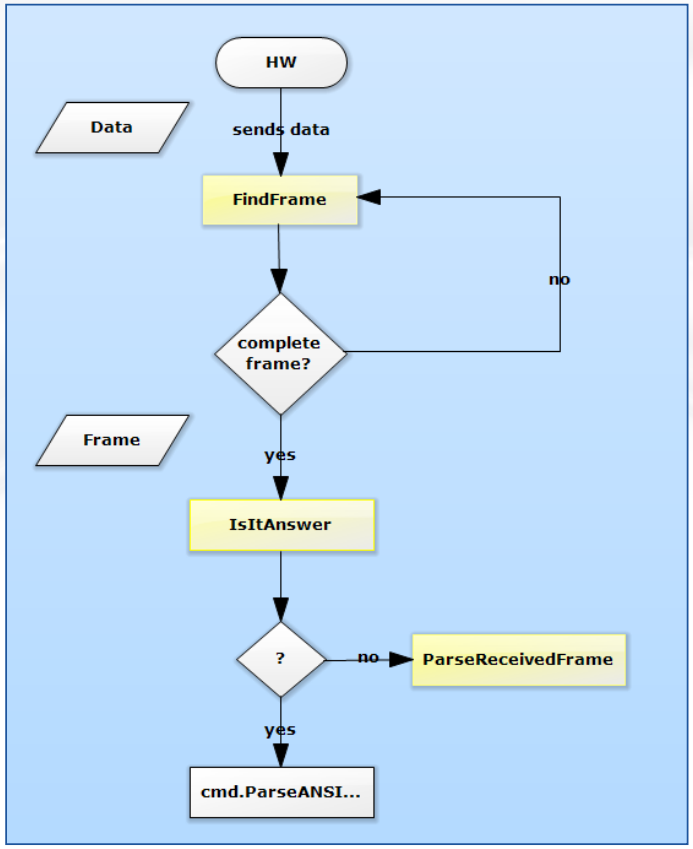
    c) Skip specific answers

```
if (dataArrayReceived[0] == 0x15)
    return false
end
```

- More info can be found in UNI RUBY DataApex Help - DeviceWrapper Class Reference

COMMUNICATION WITH HW

- Implement *ParseReceivedFrame(receivedByteArray)* to processes unrequested data sent by hardware



COMMUNICATION WITH HW

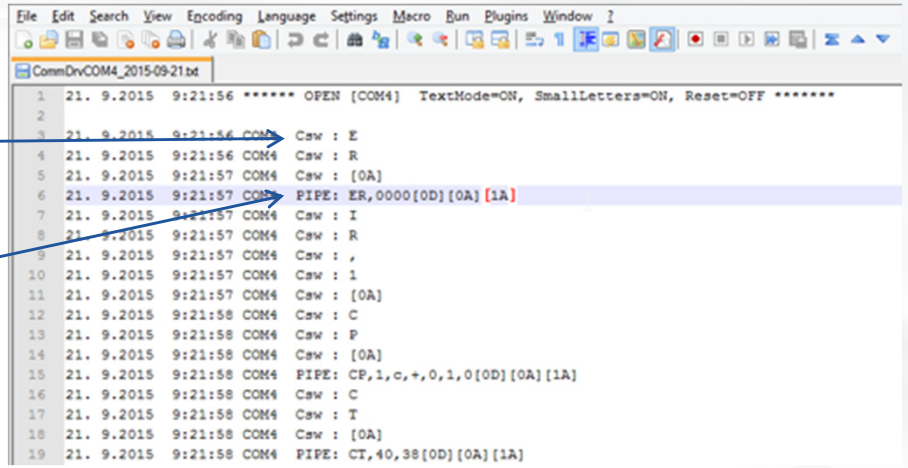
MD1

- Enable communication log for device
  - open CommDrv.ini
  - find your communication pipe (or add it) and set echo=ON
  - restart Clarity for the changes to take effect

```
[COM1]
echo=ON
textmode=ON
filename=CommDrvCOM1_%D.txt
reset=OFF
```

Send command (Csw)

Received answer (PIPE)



Snímek 30

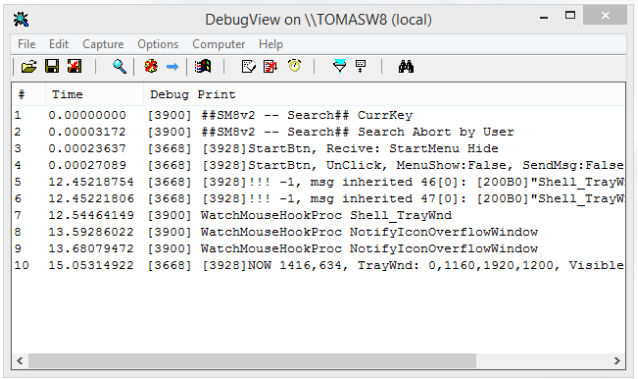
MD1 Přidán doplňující řádek, smazán deprecated ini soubor

Mentlík Daniel; 2023-12-19T13:28:59.731

DEBUGGING SCRIPT

MD1

- *TracePipe (pipeNumber, message)*
  - Function allowing to place specific text right into the pipe communication log when pipe logging is enabled and configured in CommDrv.ini
- *Trace(message)*
  - sending a string to the debug monitor.
  - The given string has to be less than 512 characters long. Output can be captured by DbgView.exe.
  - https://technet.microsoft.com/en-us/sysinternals/debugview.aspx
  - Or newer open-source https://github.com/djeedjay/DebugViewPP



MD1 odstranění deprecated ini souboru

Mentlík Daniel; 2023-12-19T13:30:23.492

ERROR HANDLING

ReportError (errorSeverity, message) – show error in Clarity

EsStopSingle_StopSeq

- Stops both current acquisition and sequence, device gets into initial/idle state. For example pipe error, detector error, auto-sampler motor error. Clarity continues in CONTROL TIME, sub-device causing this error must decide if it has sense for it to continue in CONTROL TIME itself, whatever decision is reported to Clarity via CSubDevice::GetMethodLength. When CSubDevice::GetMethodLength returns METHOD_FINISHED then CONTROL TIME is skipped.

EsStopSingle_RunSeqIfNotCalibration

- Autosampler's error level for reporting MISSING VIAL error.
- Stops current acquisition but tries to continue with sequence if current vial is not calibration. It should be enabled in autosampler method (by some combobox) if Clarity should continue in sequence when non-calibration vial is missing. Clarity continues in CONTROL TIME, sub-device causing this error must decide if it has sense for it to continue in CONTROL TIME itself, whatever decision is reported to Clarity via CSubDevice::GetMethodLength. When CSubDevice::GetMethodLength returns METHOD_FINISHED then CONTROL TIME is skipped.

ERROR HANDLING

ReportError (errorSeverity, message) – show error in Clarity

EsStopSingle_RunSeq

- Autosampler's error level for reporting MISSING VIAL error.
- Stops current acquisition but tries to continue with sequence. It should be enabled in autosampler method (by some combobox) if Clarity should continue in sequence when whatever vial is missing. Clarity continues in CONTROL TIME, sub-device causing this error must decide if it has sense for it to continue in CONTROL TIME itself, whatever decision is reported to Clarity via CSubDevice::GetMethodLength. When CSubDevice::GetMethodLength returns METHOD_FINISHED then CONTROL TIME is skipped.

EsLogInfo

- Information or success to be written into Clarity CDS, sequence and chromatogram log (no message box would appear on screen).

EsLogFailure

- The least severe error, writes into Clarity CDS, sequence and chromatogram log (no message box would appear on screen).

ERROR HANDLING

ReportError (errorSeverity, message) – show error in Clarity

EsAbort

- Little less severe error than esShutDown for reporting LC pressure errors and similar.
- It is possible to customize reaction on CmdAbortRunError signal in device method, for example user wants to switch detector lamps off when running sequence over night but does not want to switch detector lamps off when running attended single runs. Clarity does not continue in CONTROL TIME but chromatogram is created.

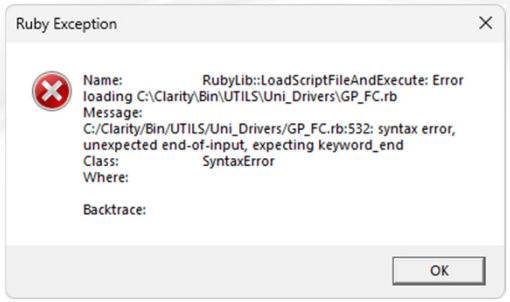
EsCommunication

- Indicates, that particular HW does not communicate.

```
if (cmd.SendCommand($timeOut) == false)
        ReportError(EsCommunication, t("COMMAND_TIMEOUTED") % „xxx")
        return false
end
```

RUBY EXCEPTION

- Check your script with Ctrl+shift+F7 from Notepad++ with UNI Ruby Plugin installed
- When error in ruby script occurs (syntax error, wrong variable used,...), Clarity will show error message with description.
- Look for a line number and message, if you have Notepad++ with script opened, the line should be automatically selected



RUBY SCRIPT LOCALIZATION

- Implement function t(stringID) - copy code to your script from other existing scripts

MD1

```
def t(stringID)
    $locale = GetLanguageCode()       # ask Clarity about current language code
    defaultLang = "ENG"           #TraceLine ("Lang:" << $locale.to_s)
    if (!$locale)
         $locale = defaultLang
    elsif (!$translation.has_key?($locale))        #Missing language - set language to
    default       $locale = defaultLang
    end
    if ($translation.has_key?($locale) && $translation[$locale].has_key?(stringID))
         return $translation[$locale][stringID]
    elsif ($locale != defaultLang && $translation.has_key?(defaultLang) &&
    $translation[defaultLang].has_key?(stringID))
    #Chosen language does not have translation with given ID, use default language instead
         return $translation[defaultLang][stringID]
    end
    return stringID + " string not found"
end
```

Snímek 36

MD1 doplnění první uvozující věty
Mentlík Daniel; 2023-12-19T13:34:54.744

RUBY SCRIPT LOCALIZATION

- Localized string are stored in global variable **`$translation`** at the end of script
- type Hash http://ruby-doc.org/core-2.3.0/Hash.html

```
$translation = {
  "ENG" => {
         "NAME" => "LC Name",
         "NAME_LONG" => "LC name too long.",
         "COMMAND_TIMEOUTED" => "Command %s timeouted.",
         "BAD_ANSWER" => "Command %s wrong answer",
         "BAD_ANSWER2" => "Command %s wrong answer: %s",
         "ERR_10" => "Command not in the specified format",
         "ERR_20" => "Outside of set value limits specified„
         }
}
```

String parameters:
%s - string
%d – integer number
%.3f – float number (3 decimal places)

- Use in script

```
t(„string_ID“)
t(„string with %parameter“) % variable
t(„string with 2 % parameters) % [variable1, variable 2]
```

23.0331

Clarity™

ADVANCED CHROMATOGRAPHY SOFTWARE

UNI RUBY

DEVICES TYPES

DataApex

THERMOSTAT (OVEN)

- SubDevice class
  - class Thermostat < ThermostatSubDeviceWrapper

- Device class Init

```
@m_Thermostat=Thermostat.new
AddSubDevice(@m_Thermostat)
```

- Add Auxiliary signal (optional)

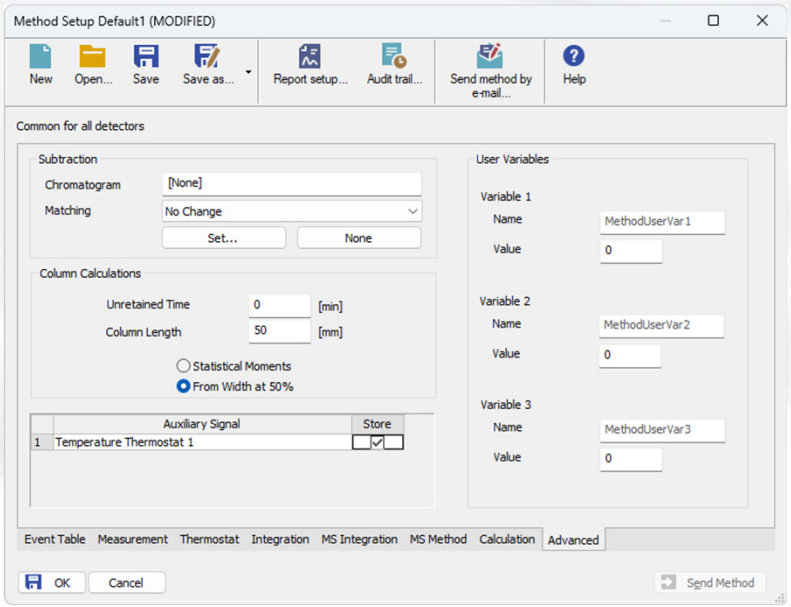
```
AuxSignal().AddSignal("Temperature", t("TEMPERATURE"), EMeaningTemperature)
```

- Doesn't have any special framework methods

Aux. signals can be added in all device types

AUXILIARY SIGNAL

- Check signal in Method Setup > Advanced dialog



PUMP

- SubDevice class
  - class LC < LCSubDeviceWrapper
- Device class Init

  ```
  @m_LC=LC.new
  ```
- Set Pressure limits

  ```
  SetDefaultLowerPressureLimit(...)
  SetDefaultUpperPressureLimit(...)
  ```
- Auxiliary pump

  ```
  @m_LC.SetLCIsAuxiliaryEventTable(Configuration().GetInt("AuxiliaryPump")!=0)
  ```
- Add Auxiliary Signals (optional)
  - if the pump is auxiliary specify EMeaningAuxiliaryFlowRate

    ```
    AuxSignal().AddSignal(@m_LC,"LCFlow", t("LC_FLOW"),
    EMeaningAuxiliaryFlowRate)
    ```
  - or standard flow rate

    ```
    AuxSignal().AddSignal(@m_LC,"LCFlow", t("LC_FLOW"), EMeaningFlowRate)
    ```

PUMP

- Write auxiliary signal in CmdTimer()
  - `AuxSignal().WriteSignal(string signalName, double value)`
- Implement framework methods for pumps
  - ***CmdSendAuxiliaryLCFlow(lc,float)***
    - *float* in ml/min
  - ***CmdSendLCFlow(floats)***
    - Use *floats[0]* !

AUTOSAMPLER

- SubDevice class
  - class Aux < SamplerSubDeviceWrapper
- Device class Init

```
@m_Aux=Aux.new
AddSubDevice(@m_Aux)
```

- Specify minimum and maximum injection volume and possible step in mL

```
SetVolumeRange (double min, double max, double step)
```

- Add Input (optional)
  - *Init*

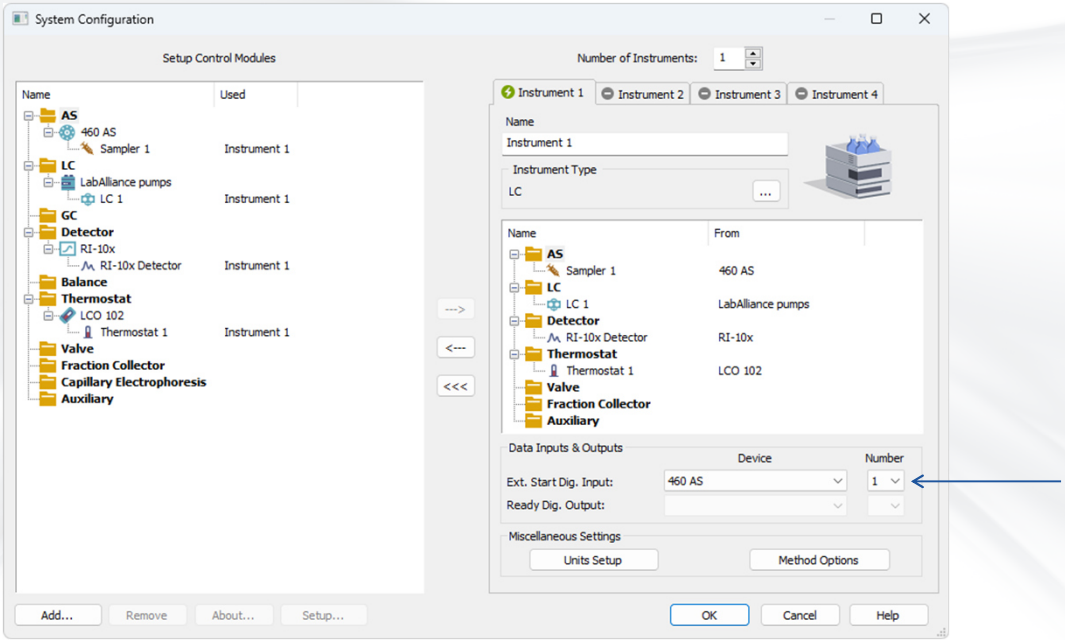
```
Input().AddInput("Input1",  EitDigital, 0,
Configuration().GetString("Input1"), "")
```

  - Sends event of specified input and given value into Clarity CDS to be processed. Change of input can be detected in CmdTimer() or ParseReceivedFrame().

```
Input().WriteInput("Input1",1)
Input().WriteInput("Input1",0)
```

More info can be found in UNI RUBY DataApex Help - DeviceWrapper Class Reference

AUTOSAMPLER - INPUTS



AUTOSAMPLER

- Add Output (optional)
  - see UNI Ruby help
  - *CmdWriteOutputInt*
  - *CmdWriteOutputDouble*
  - *CmdWriteOutputString*

- Implement CmdSendMethod – get vial number from Clarity and set vial for AutoSampler

  Method().GetVial()
- Implement
  - *CmdPerformInjection* - Indicates into samplers, that injection should be performed
  - *CmdByPassInjection*

DETECTOR

- SubDevice class
  - class Detector < DetectorSubDeviceWrapperDevice
- Decice class Init

```
@m_Detector = Detector.new
@m_Detector.SetName(Configuration().GetString('DetectorName')
AddSubDevice(@m_Detector)
```

- Set Rate and Range after AddSubDevice

```
@m_Detector.SetRate (double sampleRate)
@m_Detector.SetRange (double range)
@m_Detector.SetYUnits (strBaseUnit, strAxisName, bool bScalable, double
nCoefToBase)
```

- `@m_Detector.SetYUnits('µRIU', t("REFRACTIVE_INDEX"), false, 1.0)`

DETECTOR

- Implement framework methods:
  - *GetMethodRate(method)*
  - *GetMethodRange(method)*

It can be constant, or you can use parameter to read values from method table: method.Get.....

- Set Rate and Range at the beginning of *CmdSendMethod*

```
@m_Detector.SetRate(...)
@m_Detector.SetRange(...)
```

- Read data from detector and write it to Clarity

```
@m_Detector.WriteSignal (value)
```

  - usually in CmdTimer()
  - you can write multiple values in one period of CmdTimer

FRACTION COLLECTOR

- SubDevice class
  - class FRC < FRCSubDeviceWrapper

- Device class Init

```
@m_FRC=FRC.new
AddSubDevice(@m_FRC)
```

- Initialize vial range - default FRC start/end vial number to be shown in UI, interesting for FRC methods only.

```
SetDefaultFRCStartVial (int vialNumber)
SetDefaultFRCEndVial (int vialNumber)
```

- Implement framework methods:
  - ***CmdFRCCollect()*** - change FRC to collect mode
  - ***CmdFRCWaste()*** - change FRC to waste mode

FRACTION COLLECTOR

- ***CmdNextFRCPosition()*** - Send request to go to next valve position. Called by FC in UNI Ruby during automatic collection MD1
- *CmdGetFRCPosition() (optional – some collectors do not report their position)*

Fraction collector manual control

- from Monitor() - button
- notify framework that FRC changed its state and vial position
  - OnFRC(state, vialPosition)
    - `OnFRC(FrcConfirmVial, Position)`
    - `OnFRC(FrcManualWaste,NOVIAL)`
    - `OnFRC(FrcManualCollect,NOVIAL)`
    - `OnFRC(FrcManualNext,NOVIAL)`
    - `OnFRC(FrcManualChangeVial,setPosition)`

Snímek 49

MD1 Přepis CmdGetFRCPosition popisku
Mentlík Daniel; 2023-12-19T13:41:31.325

FRACTION COLLECTOR

FrcConfirmVial

- Confirm current vial number, can be called from Resume Method button

FrcManualWaste

- Notify, that FRC started collecting as a reaction to user request from monitor, should be called from button click event handler. CmdFRCWaste() will get called from framework!

FrcManualCollect

- Notify, that FRC started collecting as a reaction to user request from monitor, should be called from button click event handler. CmdFRCCollect() will get called from framework!

FRACTION COLLECTOR

FrcManualNext

- Notify, that FRC shifted to next vial as a reaction to user request from monitor, should be called from button click event handler. CmdNextFRCPosition() will get called from framework. If collecting and‚Waste during vial change' is checked, also CmdFRCWaste() and CmdFRCCollect().

option in Method Setup dialog

FrcManualChangeVial

- Notify, that FRC changed vial as a reaction to user request from monitor, should be called from button click event handler. If collecting and‚Waste during vial change is checked', also CmdFRCWaste() and CmdFRCCollect().

FRACTION COLLECTOR

- **Important FRC functions**
  - *method.GetFRCStartVial()* – obtain default start vial set in method
  - *method.GetFRCEndVial()* – obtain default end vial
  - ***Method().GetFRCResetCondition()***
    - ErcStartRun set hardware position to GetFRCStartVial() after start of method run - implement in CmdStartSequence()

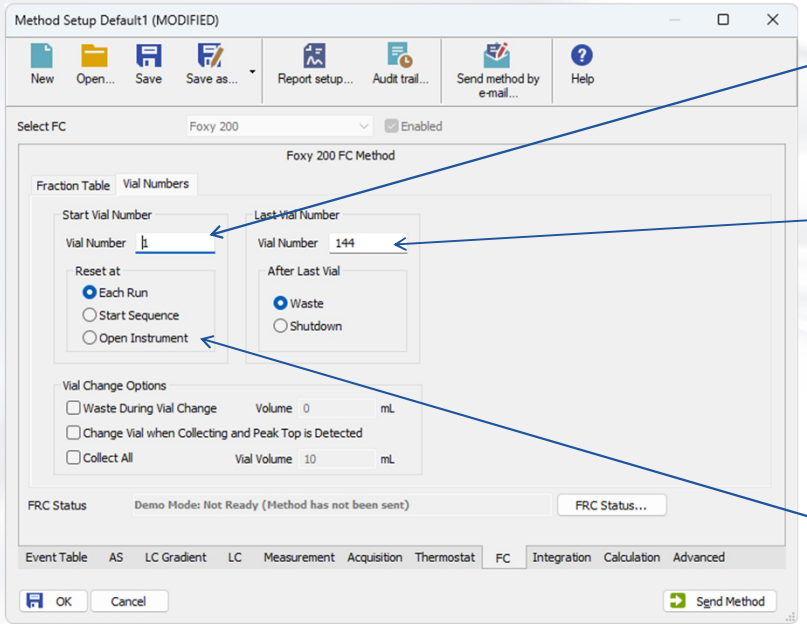
```
if Method().GetFRCResetCondition()==ErcStartRun
      #CmdFRCWaste()
      CmdSetFRCPosition(Method().GetFRCStartVial())
end
```

    - ErcStartSequence set hardware position to GetFRCStartVial() after start of sequence - implement in

```
if Method().GetFRCResetCondition()==ErcStartSequence
      CmdSetFRCPosition(Method().GetFRCStartVial())
end
```

    - ErcOpenInstrument - set hardware position to FRCStartVial() after Instrument window is opened – implement in CmdSendMethod, when method is send for the first time

FRACTION COLLECTOR



Method().GetFRCStartVial()

Method().GetFRCEndVial()

Method().GetFRCResetCondition()

23.0331

Clarity™

ADVANCED CHROMATOGRAPHY SOFTWARE

...

THANK YOU FOR YOUR ATTENTION