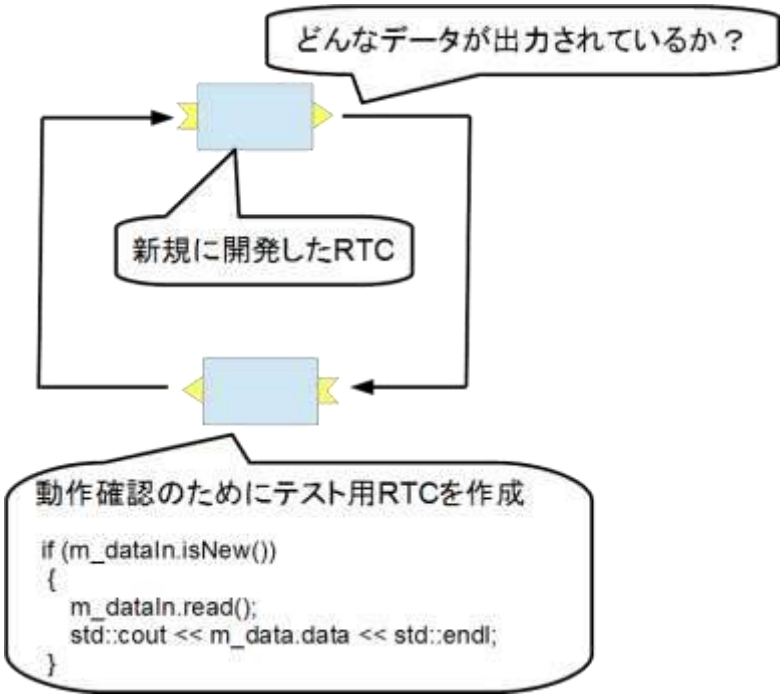
表計算ソフトによるRTコンポーネントの動作確認手順について

宮本 信彦
国立研究開発法人産業技術総合研究所
ロボットイノベーション研究センター
ロボットソフトウェアプラットフォーム研究チーム

RTCの動作確認

テスト用RTCの作成



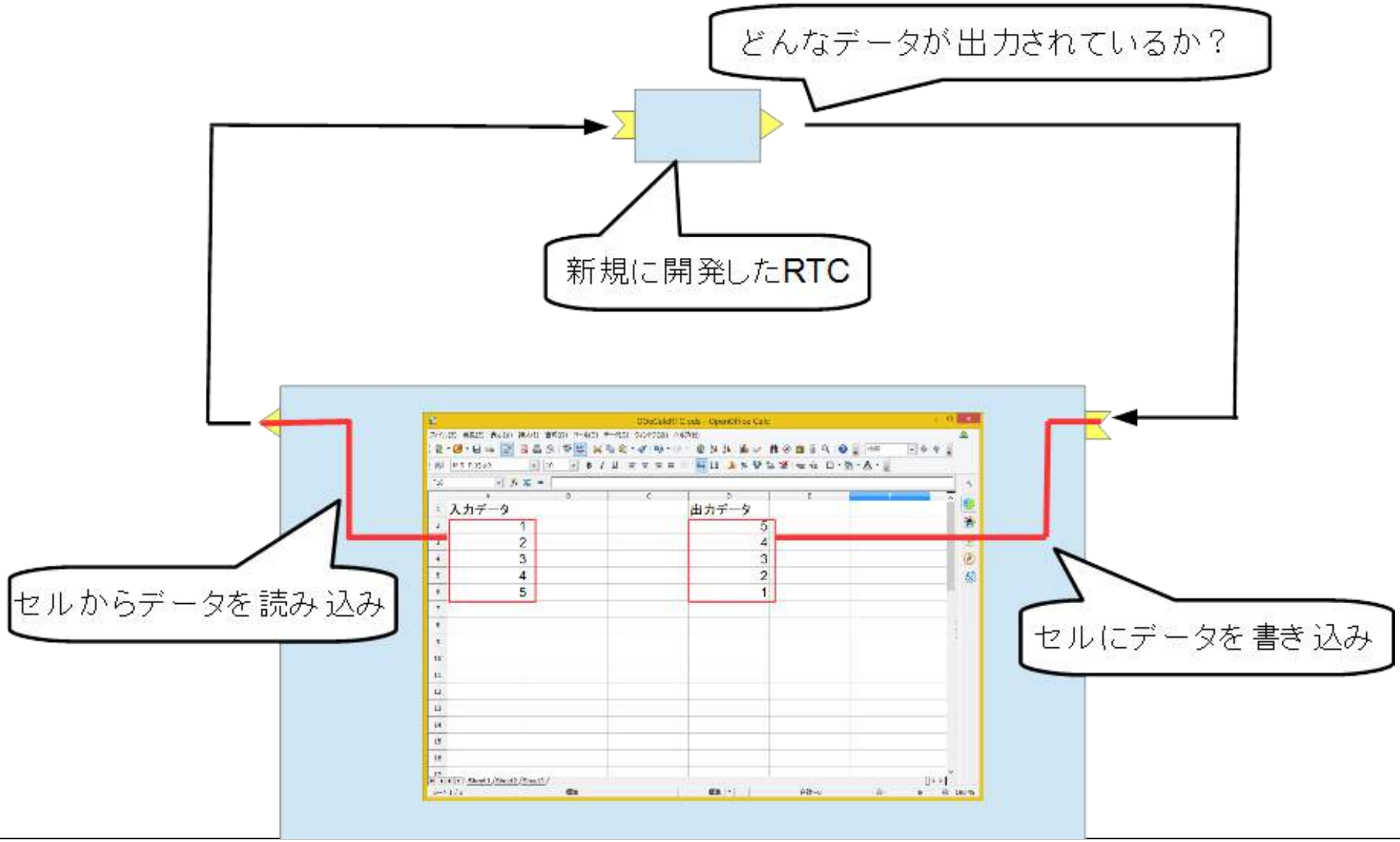
- 新規にRTCを作成するのが面倒
- 複雑なデータを入力するのが難しい
  - ファイルから読み込むなどの手順が必要
- グラフに表示するのが難しい

rtshellの利用

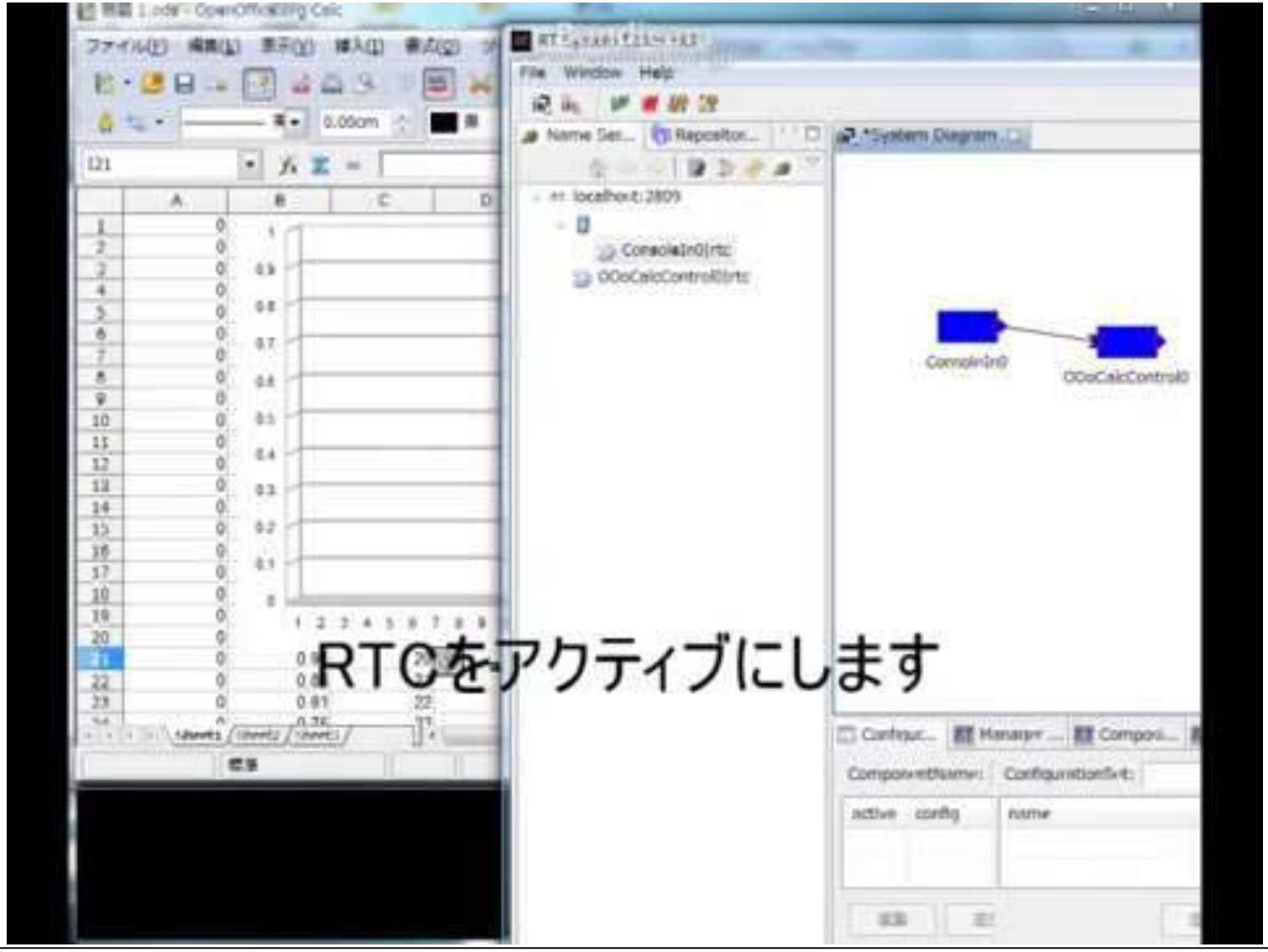
- rtprintコマンド
  - rtprint /localhost/ConsoleIn0.rtc:out
- rtinjectコマンド
  - rtinject /localhost/ConsoleOut0.rtc:in -c 'RTC.TimedLong({time}, 42)'

- コマンドによる操作が必要なため直感的ではない
- 複雑なデータを入力するのが難しい
  - ファイルからデータを読み込む方法があるのかどうか不明
- グラフに表示する方法があるかどうか不明

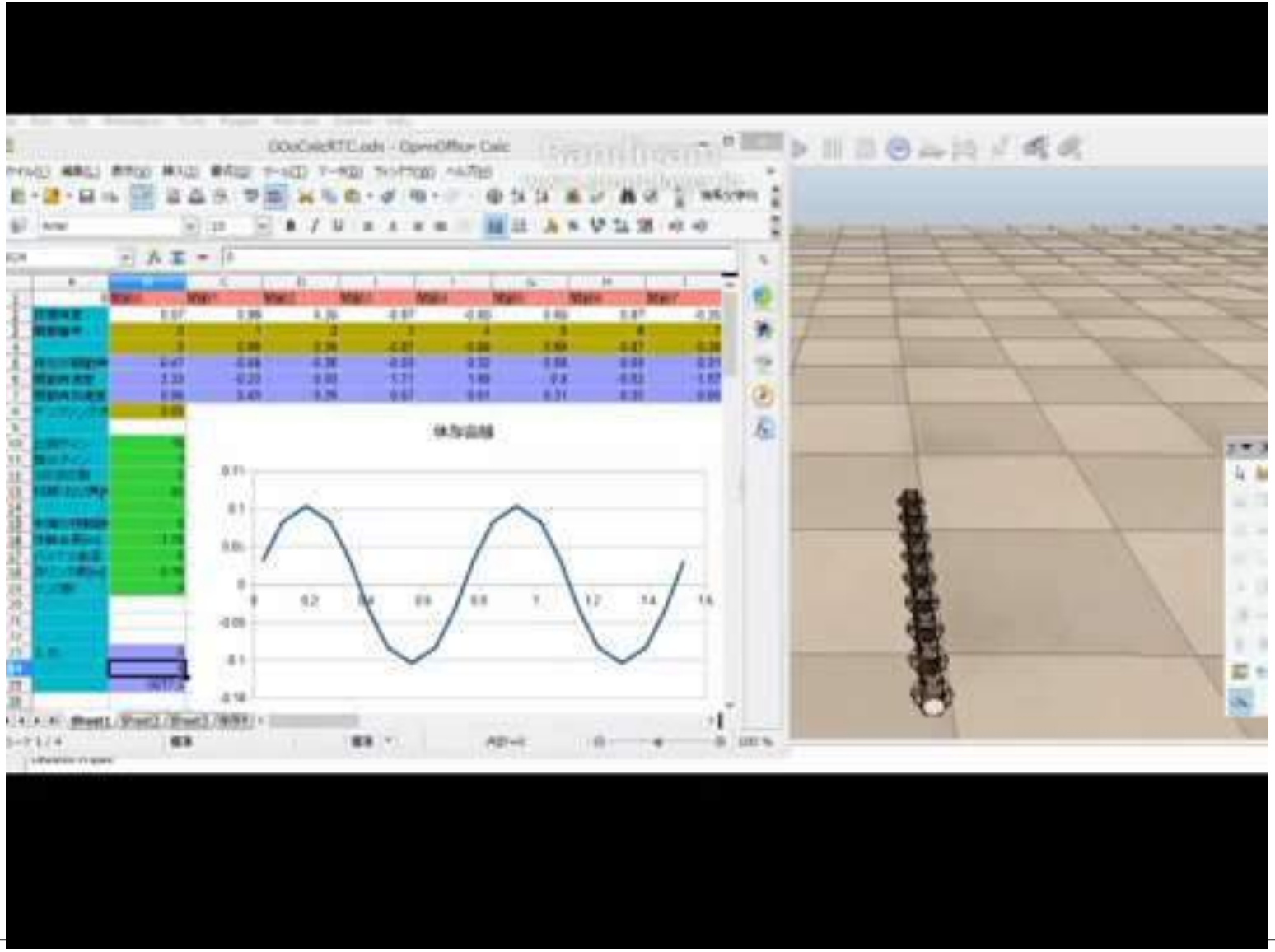
表計算ソフトによるデータ入出力



デモ動画

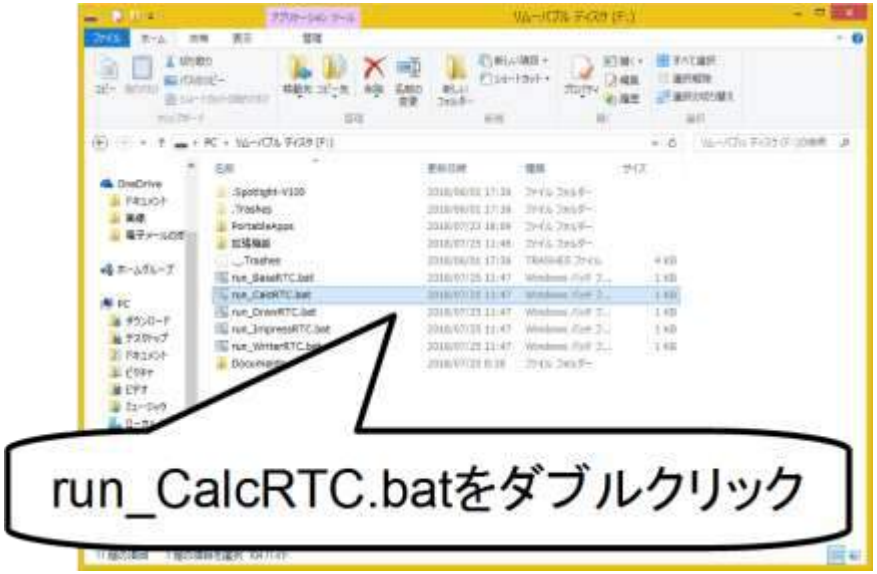


デモ動画



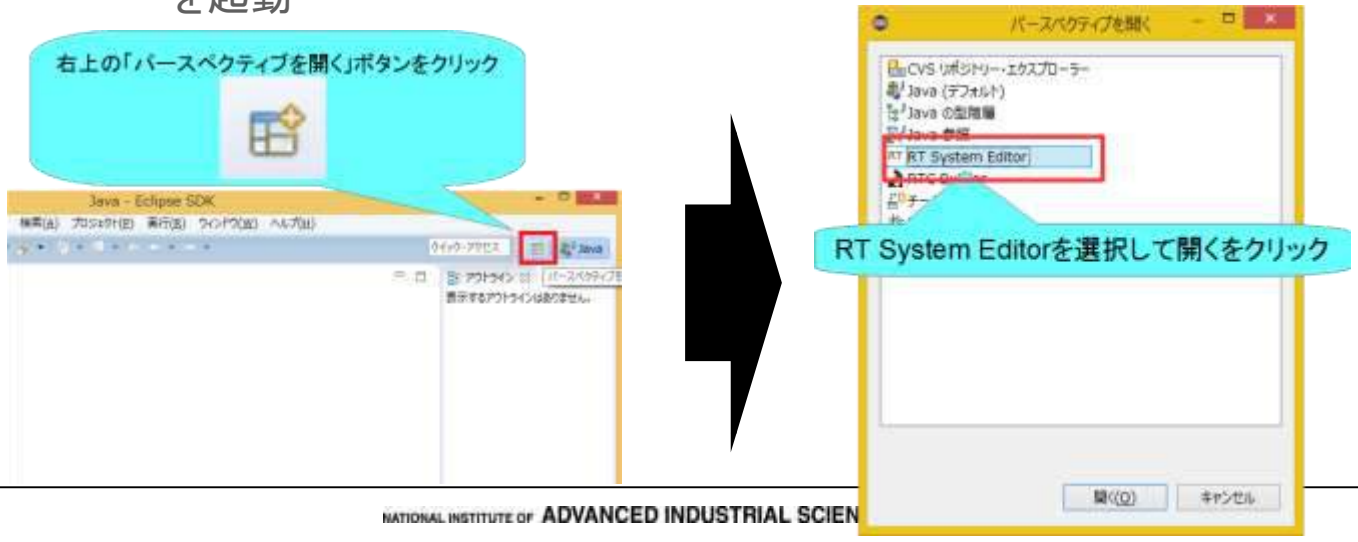
ポータブル版LibreOffice対応RTC

- 配布のUSBメモリに以下のソフトウェアを同梱
  - ポータブル版LibreOffice
  - OpenRTM-aist-Python
  - OpenOffice用RTコンポーネント



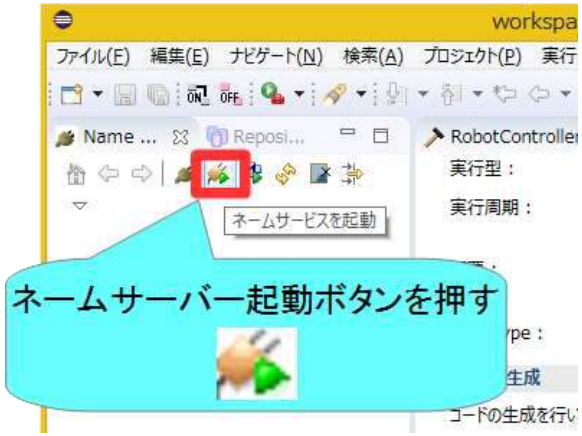
事前準備

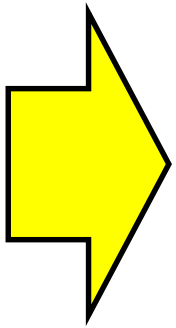
- OpenRTPを起動
  - Windows 7
    - 「スタート」→「すべてのプログラム」→「OpenRTM-aist 1.2.0」→「Tools」→「OpenRTP」
  - Windows 8.1
    - 「スタート」→「アプリビュー(右下矢印)」→「OpenRTM-aist 1.2.0」→「OpenRTP」
    - ※同じフォルダに「RTSystemEditorRCP」がありますが、これはRTC Builderが使えないので今回は「OpenRTP」を起動してください。
  - Windows 10
    - 左下の「ここに入力して検索」にOpenRTPと入力して、表示されたOpenRTPを起動

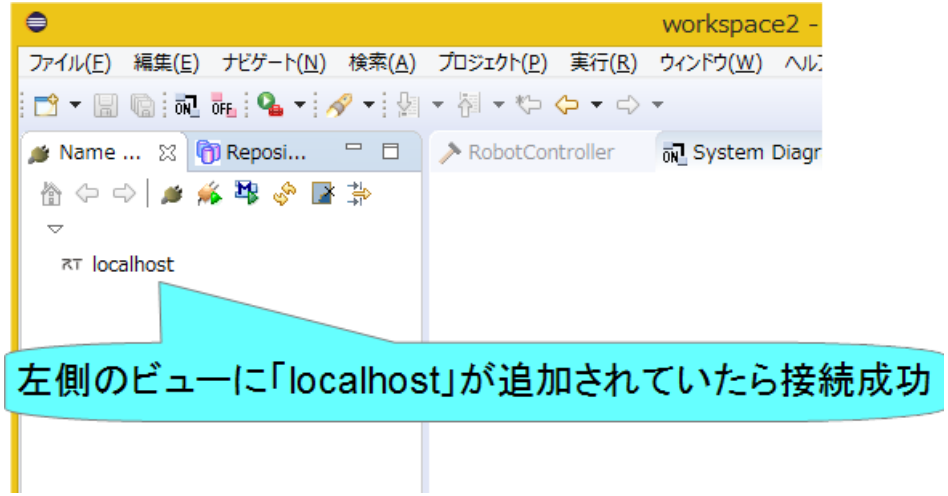


事前準備

- ネームサーバーを起動

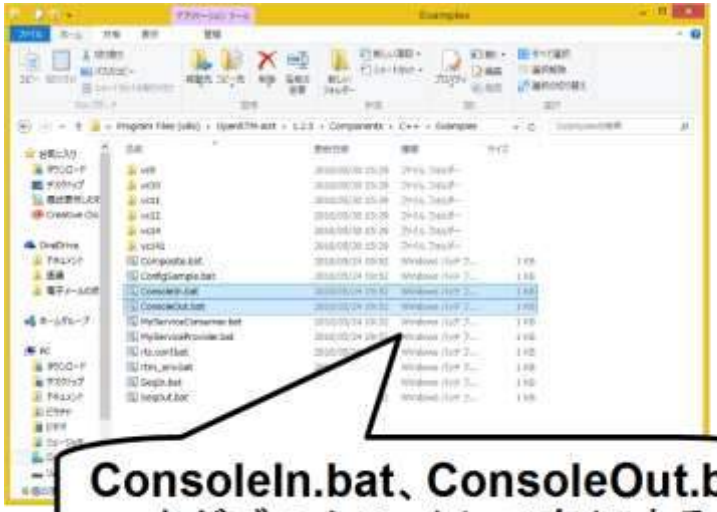




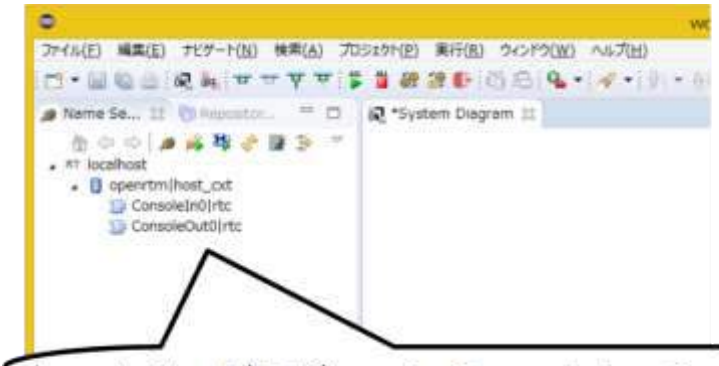


動作確認用のRTCを起動

- ConsoleIn、ConsoleOutのサンプルコンポーネントを起動する
  - Windows 7
    - 「スタート」→「すべてのプログラム」→「OpenRTM-aist 1.2.0」→「Tools」→「C++_Examples」
  - Windows 8.1
    - 「スタート」→「アプリビュー(右下矢印)」→「OpenRTM-aist 1.2.0」→「C++_Examples」
  - Windows 10
    - 左下の「ここに入力して検索」にC++_Examplesと入力して、表示されたC++_Examplesをクリック



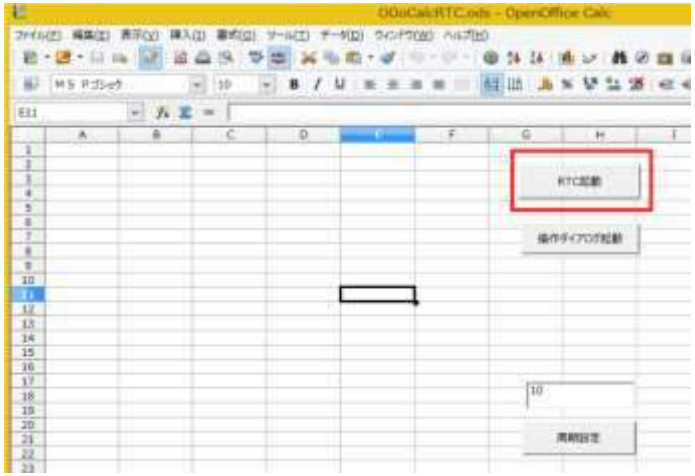
ConsoleIn.bat、ConsoleOut.bat
をダブルクリックして実行する

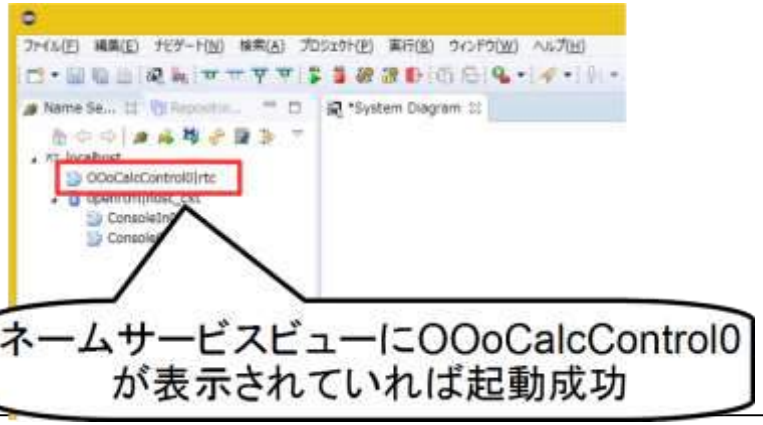


ネームサービスビューにConsoleIn、ConsoleOutが
表示されていれば起動成功

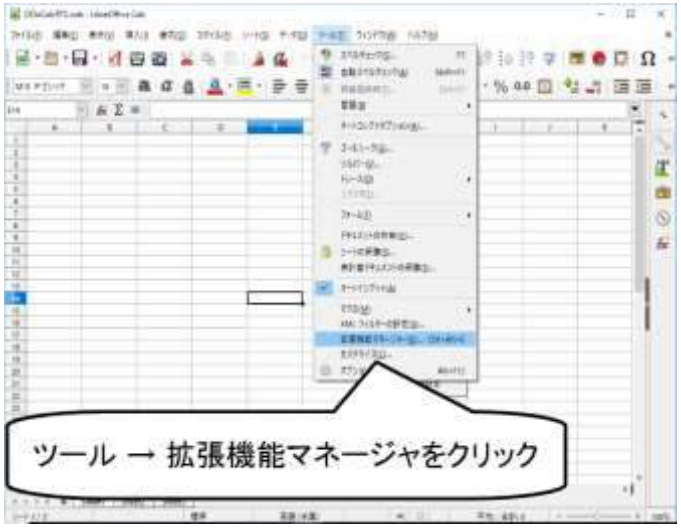
Calc用RTCを起動

- LibreOffice Calcの「RTC起動」ボタンをクリックする

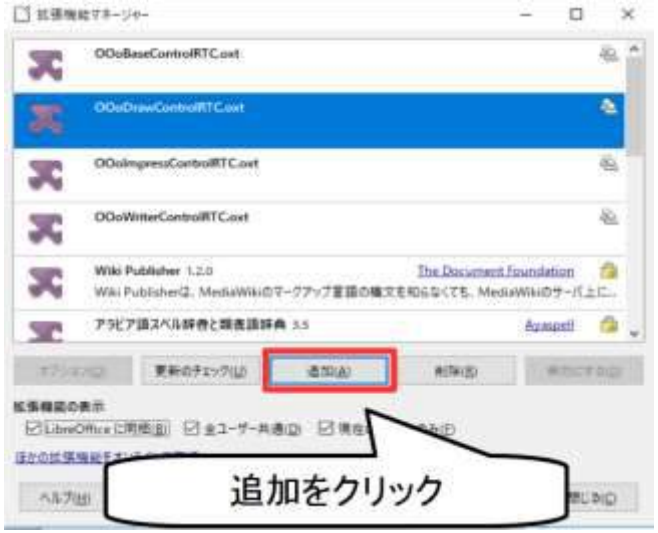


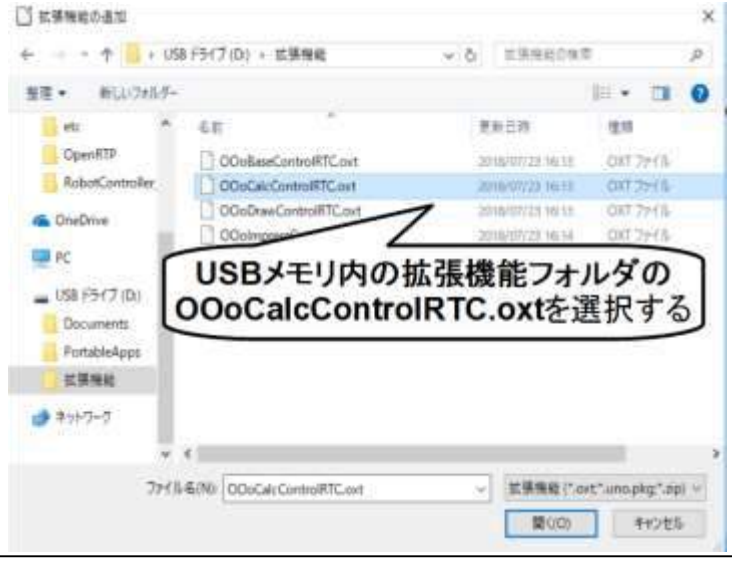


起動に失敗する場合


ツール → 拡張機能マネージャをクリック

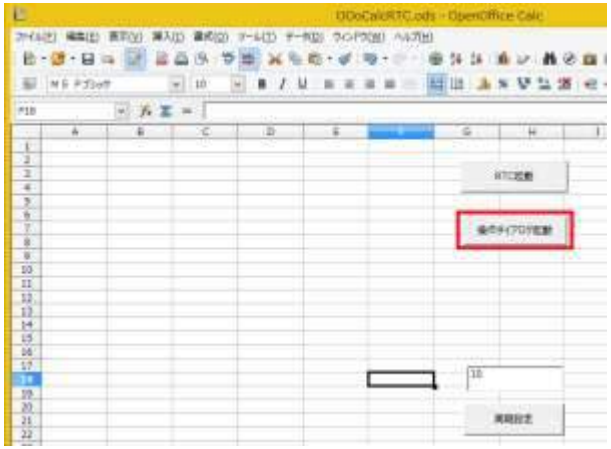

OOoCalcControlRTC.oxtを選択
削除をクリック

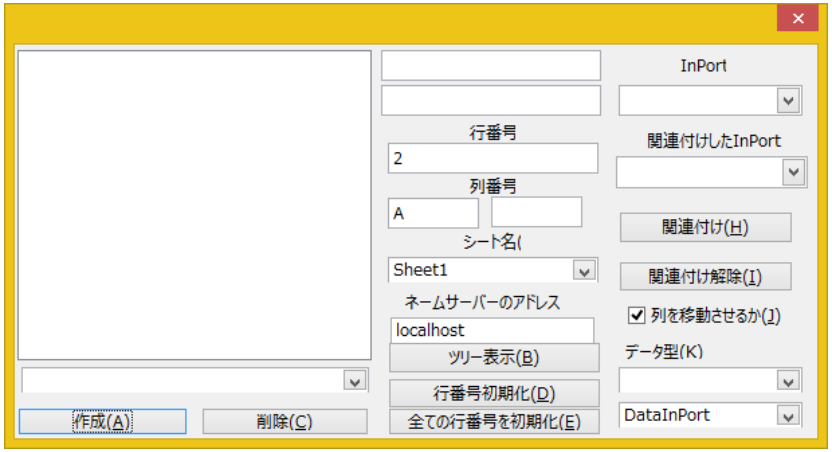

追加をクリック


USBメモリ内の拡張機能フォルダの
OOoCalcControlRTC.oxtを選択する

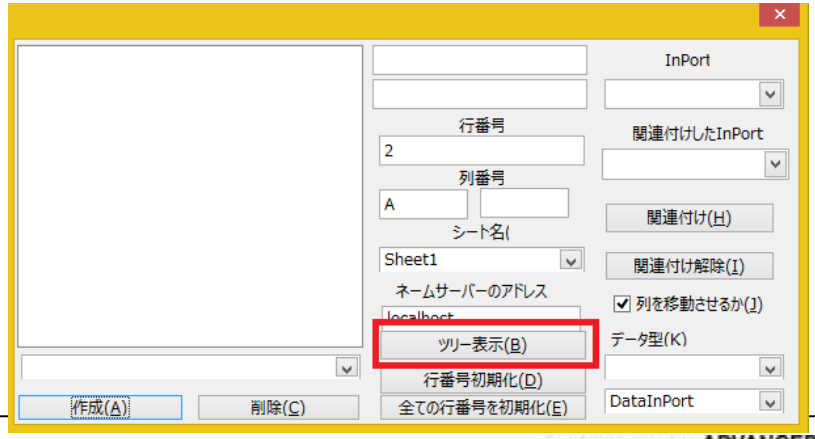
OutPortの接続

- LibreOffice Calcの「操作ダイアログ起動」ボタンをクリックする



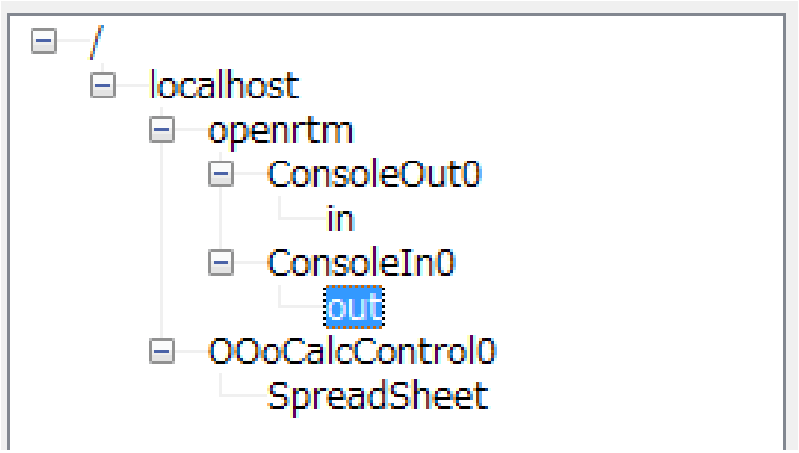


- 「ツリー表示」ボタンをクリックする

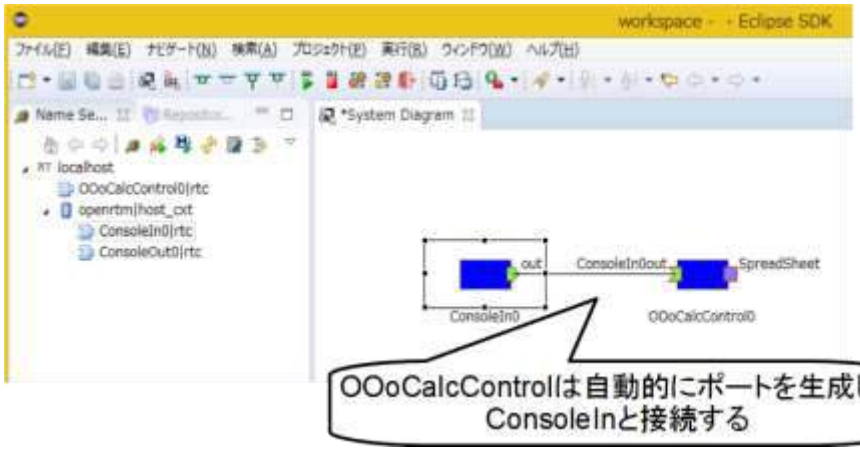


OutPortの接続

- ツリーからConsoleInのoutを選択



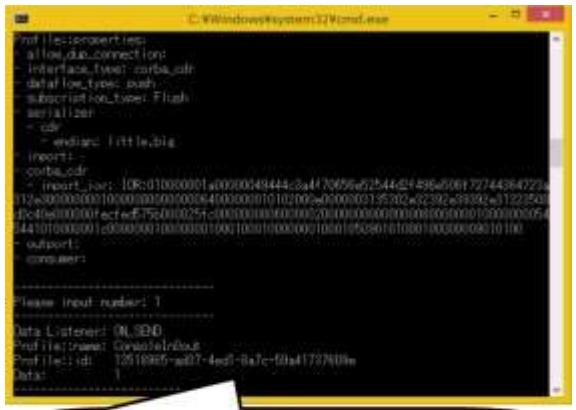
- 「作成」ボタンをクリックする

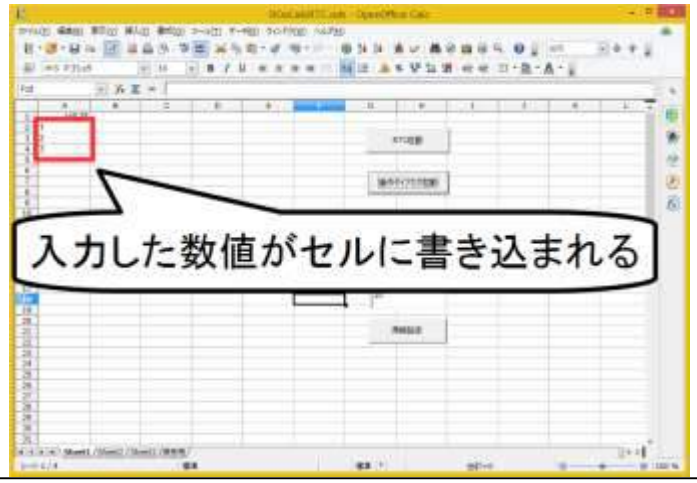


OutPortの動作確認

- RT System EditorでRTCをアクティブ化する

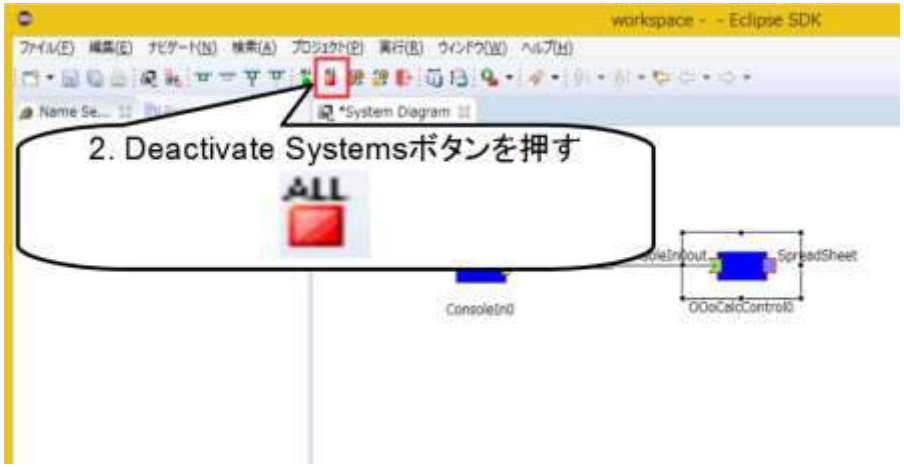
- ConsoleInのウィンドウに数値を入力する





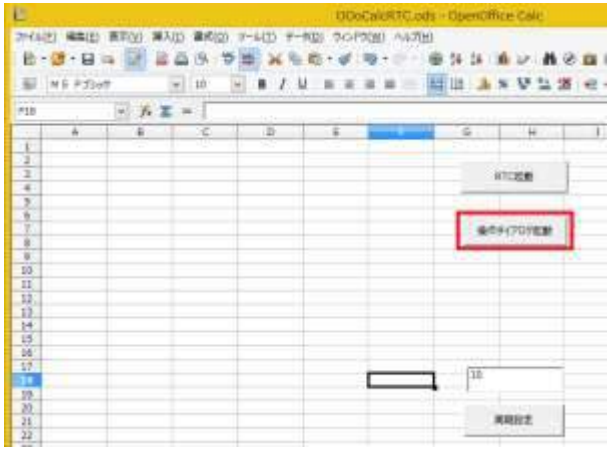
OutPortの動作確認

- 一旦、RTCを非アクティブ化する

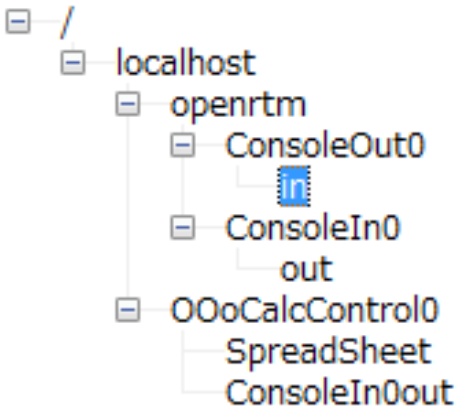


InPortの接続

- LibreOffice Calcの「操作ダイアログ起動」ボタンをクリック後、「ツリー表示」ボタンをクリック



- ツリーからConsoleOutのinを選択



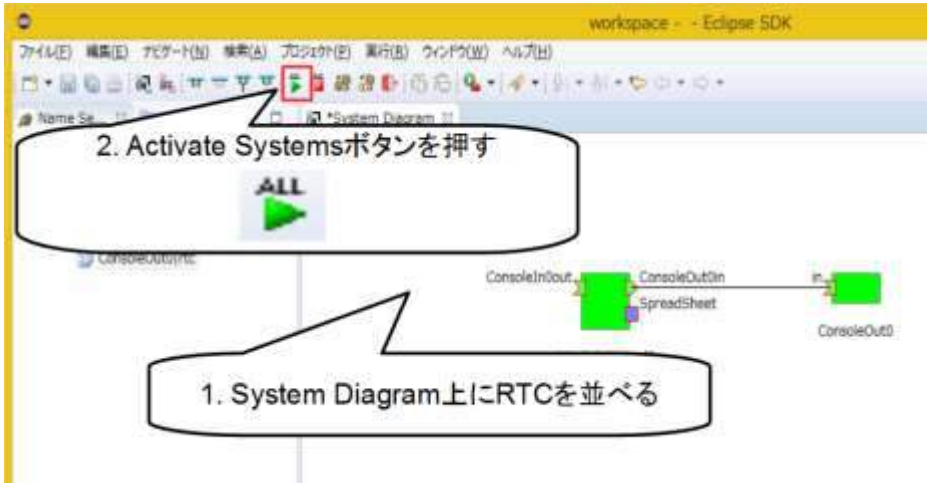
InPortの接続

- 列番号をBに設定
- 「列を移動させるか」のチェックを外す
  - ※2回クリックしないとチェックが外れない

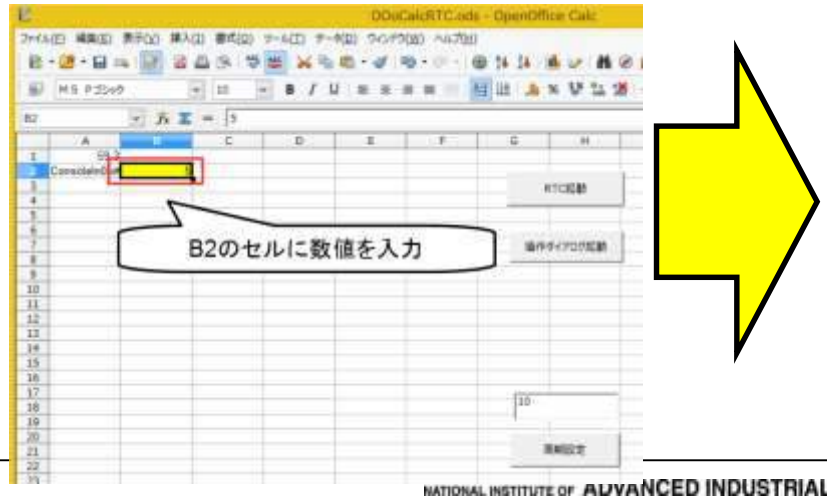
- 「作成」ボタンをクリックする

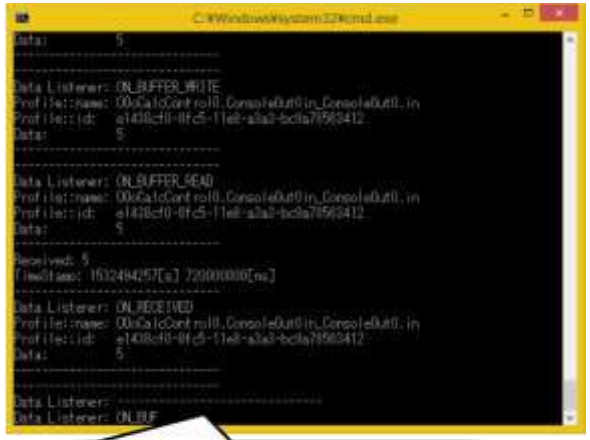
InPortの動作確認

- RT System EditorでRTCをアクティブ化する



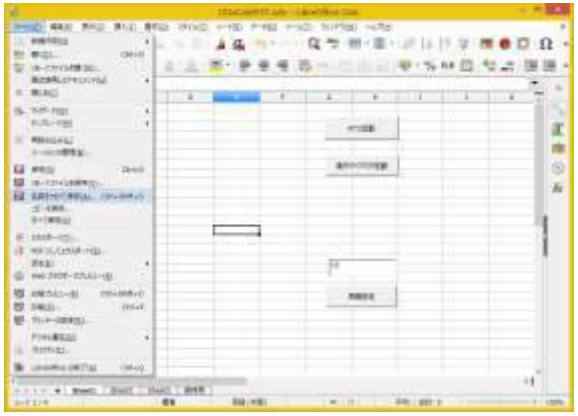
- B2のセルに数値を入力する



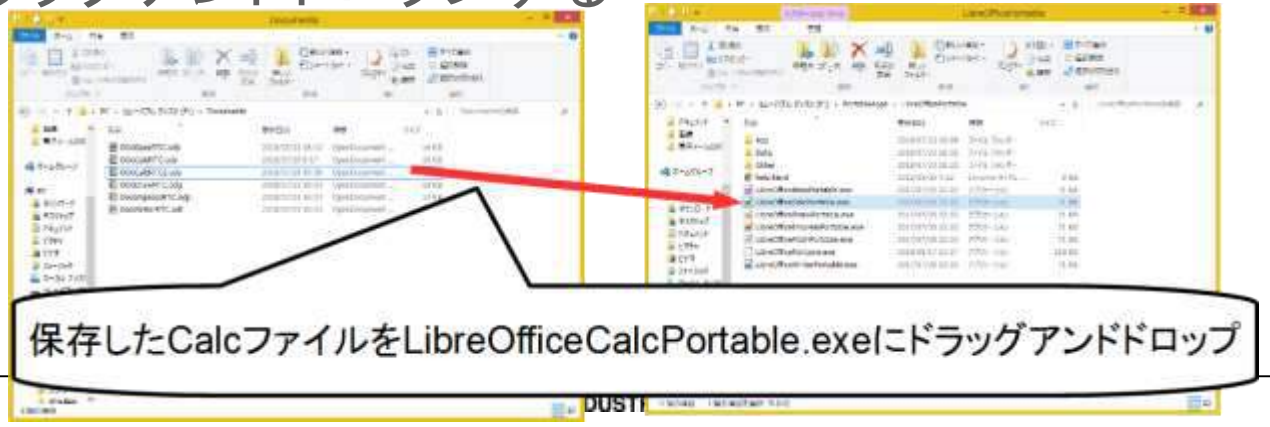


保存

- Calcファイル(.ods)を名前を付けて保存する
  - 接続したポートの情報などはCalcファイル(.ods)に保存される

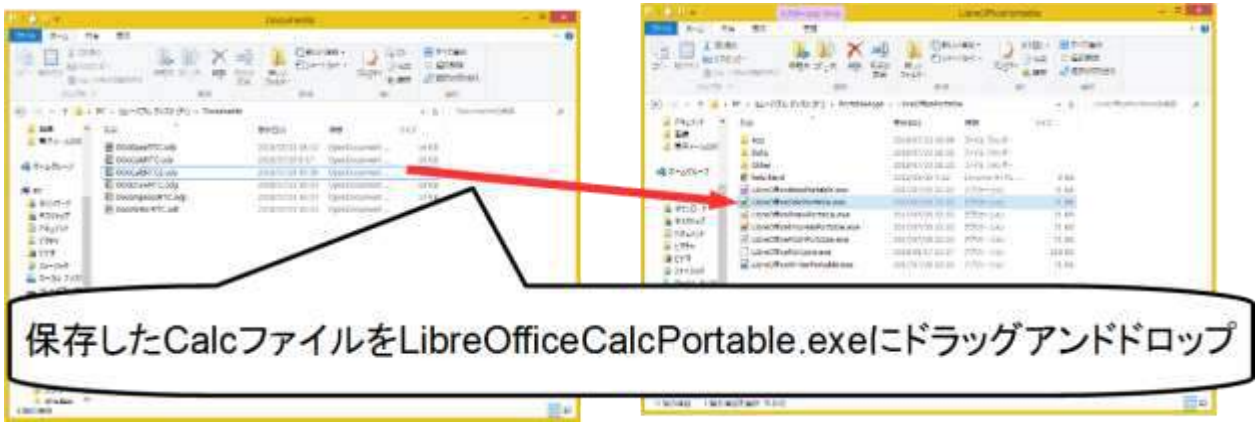


- 保存したCalcファイルをUSBメモリ内の PortableApps¥LibreOfficePortable¥**LibreOfficeCalcPortable.exe** にドラッグアンドドロップする

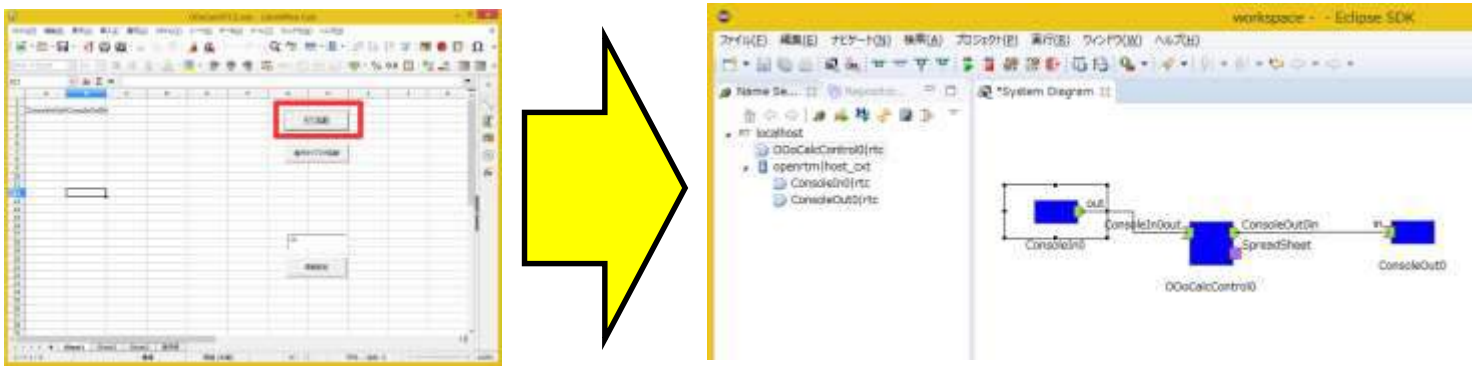


復元

- 保存したCalcファイルをUSBメモリ内の PortableApps¥LibreOfficePortable¥**LibreOfficeCalcPortable.exe** にドラッグアンドドロップする



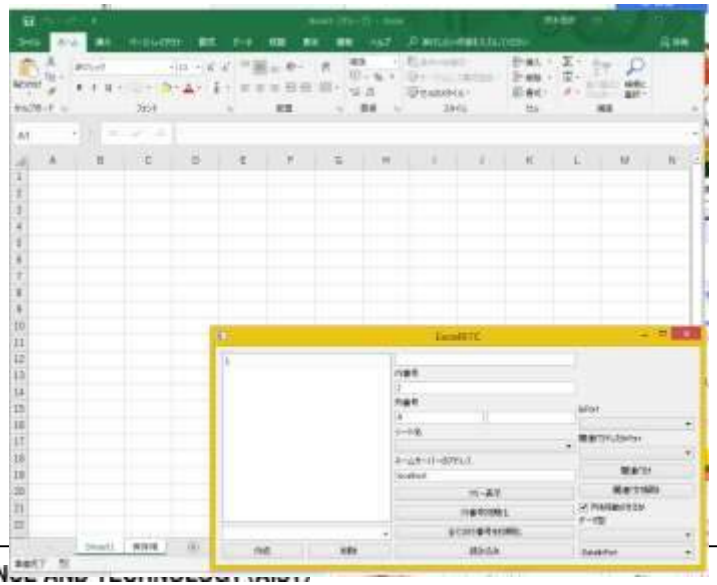
- 「RTC起動」ボタンを押すと、ポートに自動接続して状態を復元する



おわりに

- 表計算ソフトを用いたRTCの動作確認の手順を紹介
  - ポータブル版LibreOffice＋OpenRTM-aist実行環境のUSBメモリを配布
  - サンプルコンポーネントのInPort、OutPortに接続して動作確認
    - データポートを指定すると自動的に接続する
    - 「列を移動する」のチェックボックスが有効の場合は、対象となるセルの位置が下へ移動する

- Microsoft Office用のRTCについては、OpenRTM-aist-1.2ではデフォルトでインストールされます
  - ※Excelのインストールが必要



Lua版RTミドルウェア (OpenRTM Lua)の紹介

※RTミドルウェアコンテスト2018発表予定

OpenRTM Lua

- プログラミング言語Luaに対応したRTミドルウェア
  - RTCを**Lua**、もしくは**MoonScript**で開発可能にする

Lua

```lua
ConsoleIn.new = function(manager)
        local obj = {}
        -- RTObjectをメタオブジェクトに設定する
        setmetatable(obj, {__index=openrtm.RTObject.new(manager)})
        -- データ格納変数
        obj._d_out = openrtm.RTCUtil.instantiateDataType("::RTC::TimedLong")
        -- アウトポート生成
        obj._outOut = openrtm.OutPort.new("out",obj._d_out,"::RTC::TimedLong")

        -- 初期化時のコールバック関数
        -- @return リターンコード
        function obj:onInitialize()
                -- ポート追加
                self:addOutPort("out",self._outOut)

                return self._ReturnCode_t.RTC_OK
        end
        -- アクティブ状態の時の実行関数
        -- @param ec_id 実行コンテキストのID
        -- @return リターンコード
        function obj:onExecute(ec_id)
                io.write("Please input number: ")
                local data = tonumber(io.read())
                -- 出力データ格納
                self._d_out.data = data
                -- 出力データにタイムスタンプ設定
                openrtm.OutPort.setTimestamp(self._d_out)
                -- データ書き込み
                self._outOut:write()
                return self._ReturnCode_t.RTC_OK
        end

        return obj
end
```

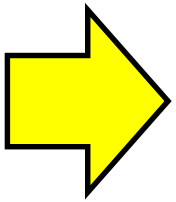
MoonScript

```
class ConsoleIn extends openrtm_ms.RTObject
        -- コンストラクタ
        -- @param manager マネージャ
        new: (manager) =>
                super manager
                -- データ格納変数
                self._d_out = openrtm_ms.RTCUtil.instantiateDataType("::RTC::TimedLong")
                -- アウトポート生成
                self._outOut = openrtm_ms.OutPort("out",self._d_out,"::RTC::TimedLong")

        -- 初期化時のコールバック関数
        -- @return リターンコード
        onInitialize: =>
                -- ポート追加
                @addOutPort("out",self._outOut)

                return self._ReturnCode_t.RTC_OK


        -- アクティブ状態の時の実行関数
        -- @param ec_id 実行コンテキストのID
        -- @return リターンコード
        onExecute: (ec_id) =>
                io.write("Please input number: ")
                data = tonumber(io.read())
                -- 出力データ格納
                self._d_out.data = data
                -- 出力データにタイムスタンプ設定
                openrtm_ms.setTimestamp(self._d_out)
                -- データ書き込み
                self._outOut\write()
                return self._ReturnCode_t.RTC_OK
```

OpenRTM Lua

- プログラミング言語Luaに対応したRTミドルウェア
  - Luaは軽量なプログラミング言語であり、RTC実行環境一式で**2MB程度**
    Lua(2MB)>>>>C++(8MB)>Python(10MB以上)>>>Java(笑)
  - **スクリプト言語**であるため、Pythonと同様に効率的な開発が可能
  - LuaJITによる**高速な動作**
    - Luaはゲーム開発で主に使用されるプログラミング言語のため、10000体×onExecute関数1000回のキャラクターの当たり判定にかかる時間を計測

言語	結果[s]
Lua	4.3324
LuaJIT	1.2459
C++	0.6142
Python	6.4802



Pythonを圧倒し、C++に匹敵する性能

OpenRTM Lua

- プログラミング言語Luaに対応したRTミドルウェア
  - 他のソフトウェアへの組み込み

ゲームエミュレータ

ロボットシミュレータ

物理シミュレータ

WEBサーバー