

RTミドルウェアツール紹介： 利用可能なRTコンポーネントや ツールについて

宮本 信彦

国立研究開発法人産業技術総合研究所
インテリジェントシステム研究部門

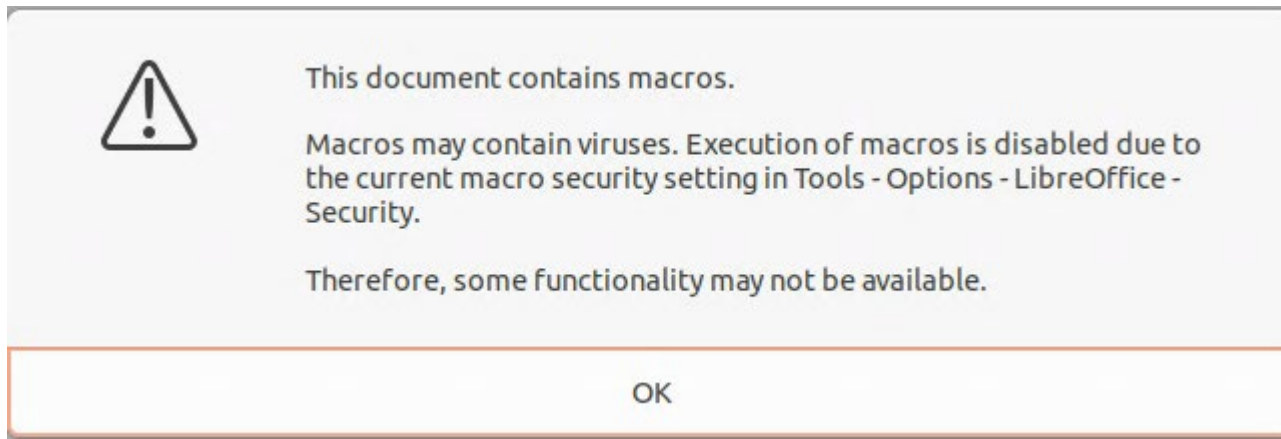


ツールのダウンロード

- 講義資料
 - <https://openrtm.org/openrtm/sites/default/files/7361/2026SummerCamp-04.pdf>
- Windowsの場合
 - USBメモリで配布
- Ubuntuの場合
 - インストール作業が必要のため次のスライドで説明します
- 上記URLはSlackでお知らせします

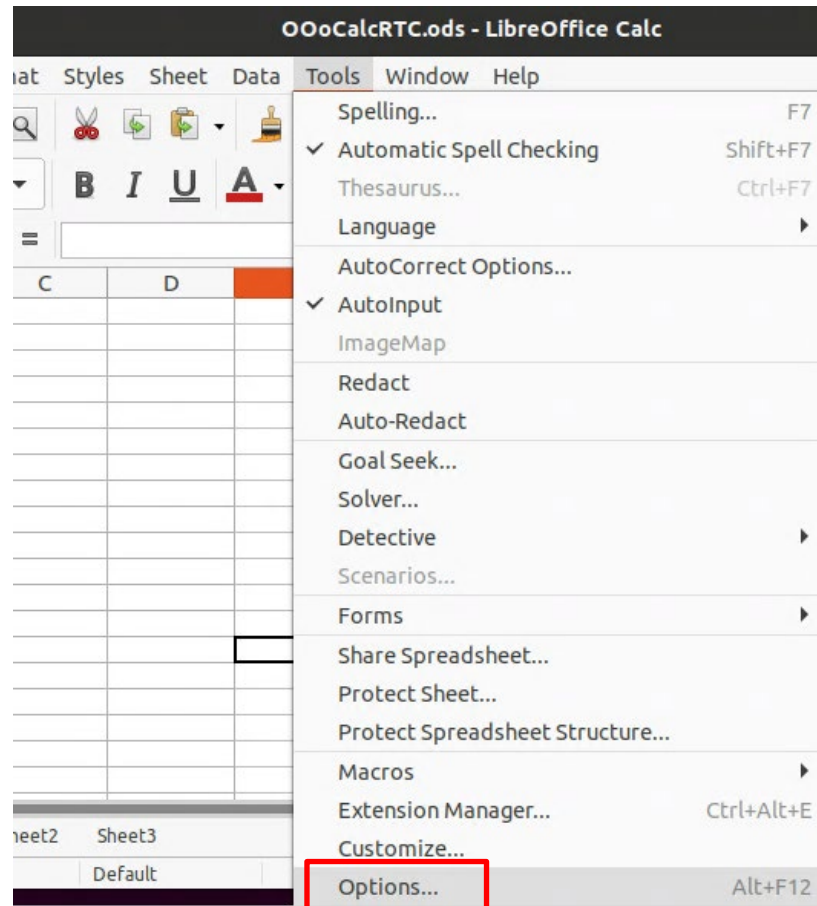
ツールのインストール(Ubuntu)

- 以下のコマンドを実行する
 - `sudo apt install libreoffice-script-provider-python`
 - `git clone https://github.com/Nobu19800/OOoRTCs`
 - `cd OOoRTCs`
 - `sh install.sh`
- OOoRTCs/OOoCalcRTC/OOoCalcRTC.odsをダブルクリックして開く
 - 以下の画面が表示されたらセキュリティの設定を変更する



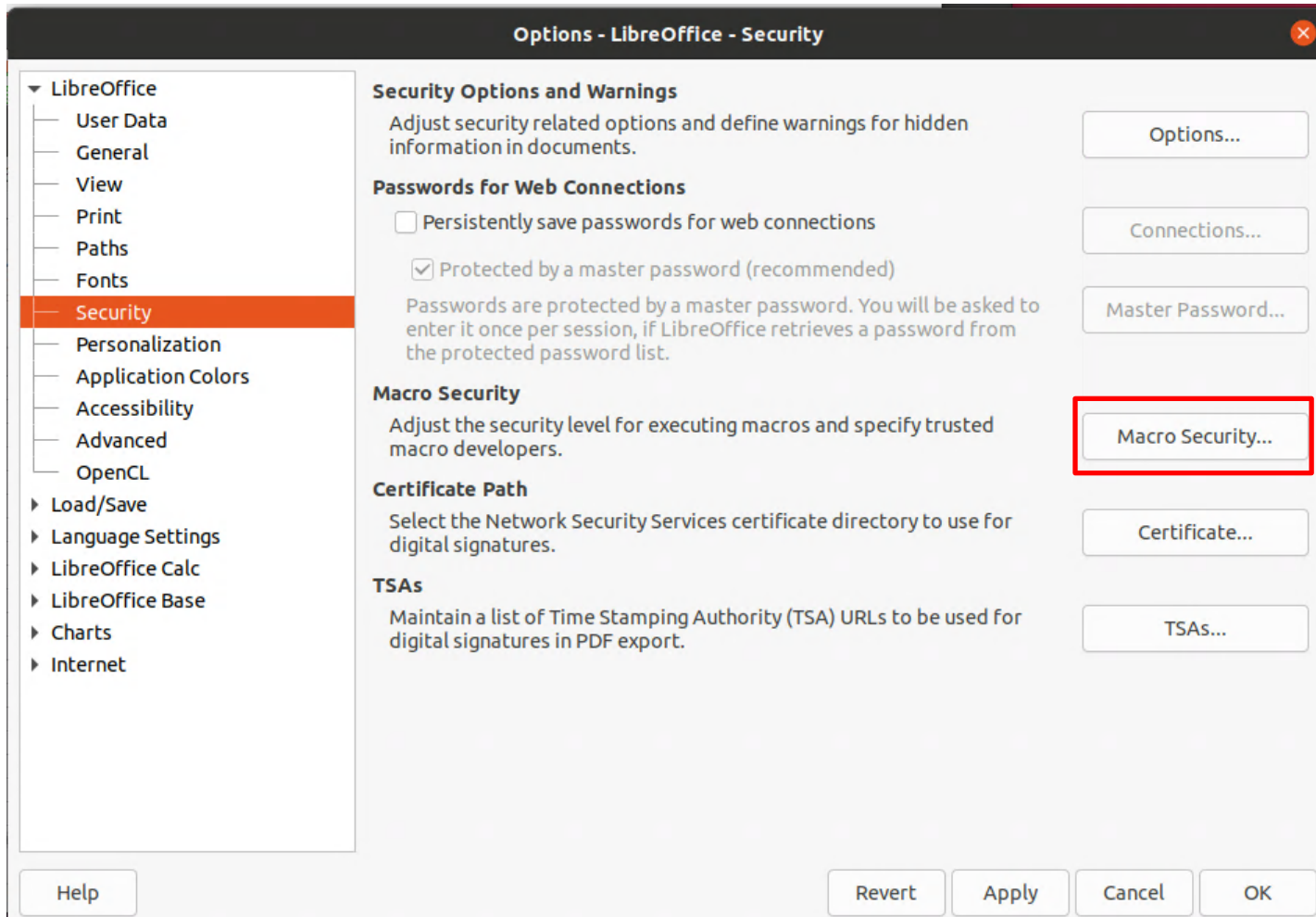
ツールのインストール(Ubuntu)

- Tools -> Options



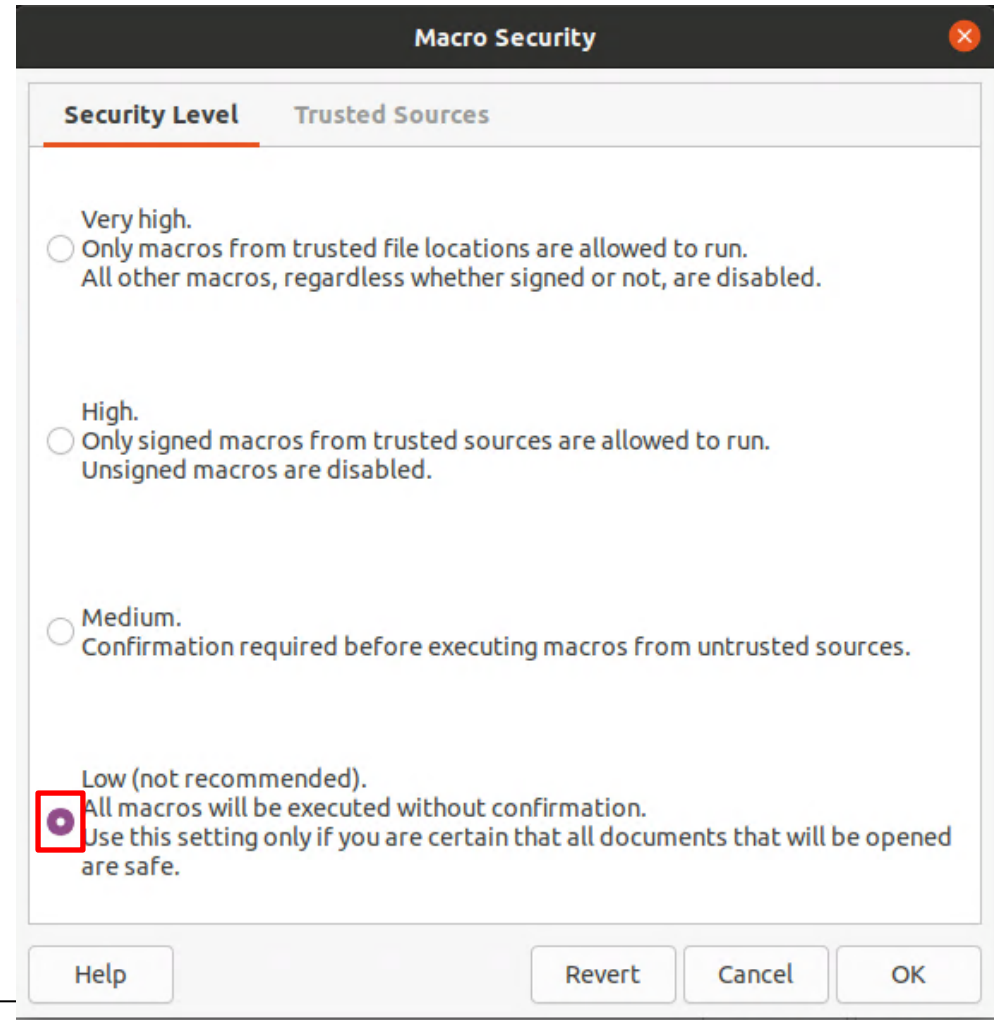
ツールのインストール(Ubuntu)

- Security -> Macro Security...



ツールのインストール(Ubuntu)

- Security LevelをLowに設定する
- 設定後、Calcは再起動する

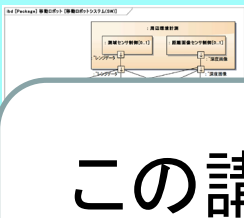
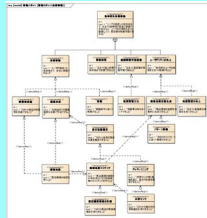


これからどうやってロボットシステム
を開発するのか？

設計

要求図

内部ブロック図

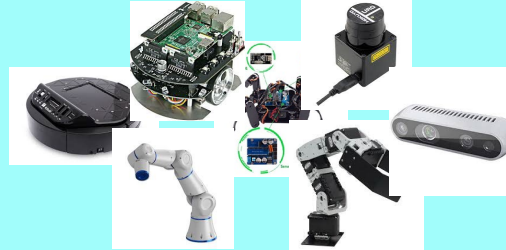


- システムの要求
- 使用するハードウェア
- 再利用するソフトウェア

昨日説明があった

RTC Builderの
利用方法は講習
会で説明済み

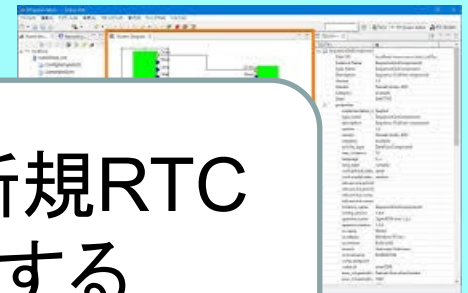
ハードウェア、既存の RTCの動作確認



- RTCの導入方法、使用方法の調査
- RTCのインストール
- RTCの入出力の確認

RT System Editor
の利用方法は講習
会で説明済み

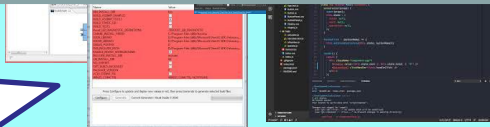
システムの構築、 動作確認



この講義ではRTCの動作確認、新規RTC の開発で有用なツールを紹介する

- RT System Editor
- rtshell
- 動作確認

rtshellについては
前の講義で説明
があった



- RTCの詳細な仕様を決める
- コーディング、ビルド
- RTCの動作確認、デバッグ

紹介するツール

- **RtStorage**

- 元々セックでOpenRTM.NETを開発していた人が作ったツール
- OutPortから出力されたデータをファイルに記録
- 保存したデータを、RTコンポーネントのInPortに対して再生

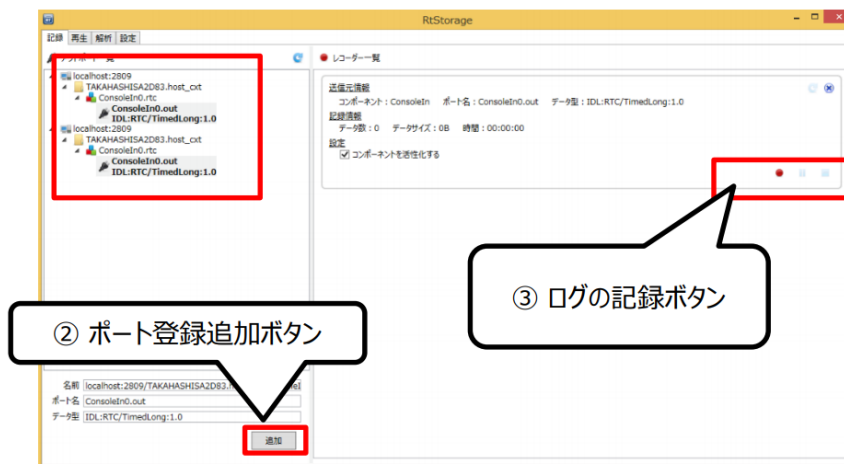
- **表計算ソフトとRTCを連携させるためのツール**

- Excel、LibreOfficeCalcのセルの値をデータポートから入出力する
- 新規に開発したRTCの動作確認が簡単にできる
 - OutPortから出力された値の確認
 - InPortに任意の値を入力したときの挙動
- 既存のRTCについても、動作がよく分からない場合に確認できる

RtStorage

RtStorageの概要

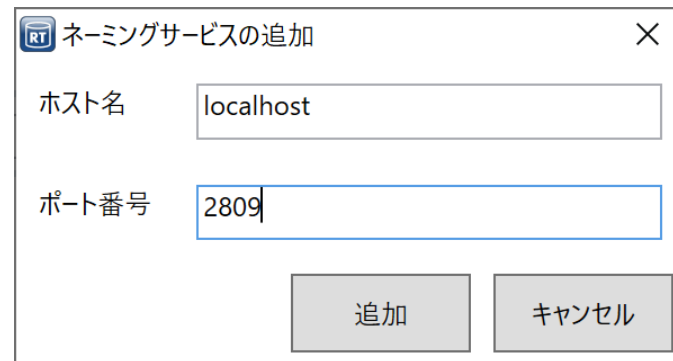
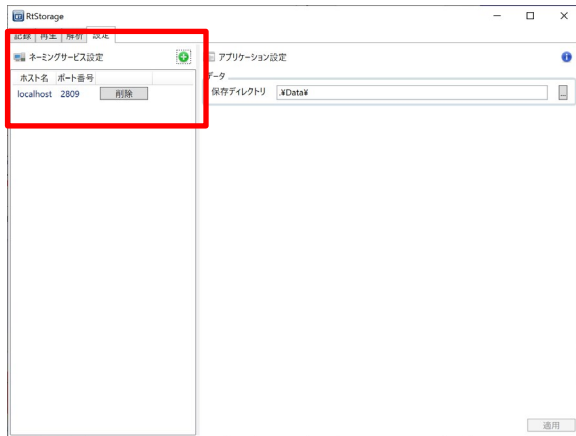
- できる事
 - OutPortから出力されたデータをファイルに記録
 - 保存したデータを、RTコンポーネントのInPortに対して再生
 - 上記の保存、再生がGUIで簡単にできる



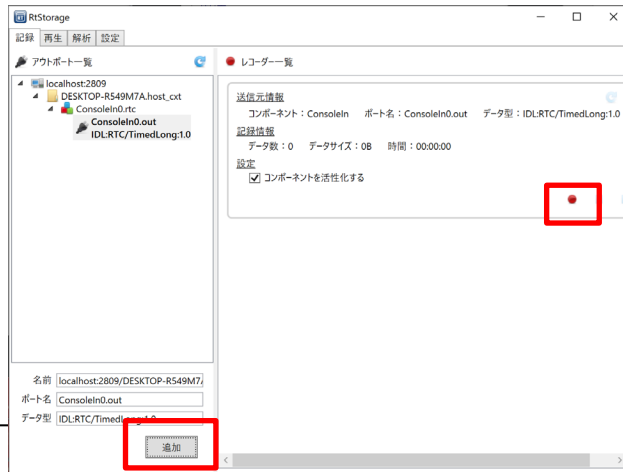
- インストール手順
 - 以下からRtStorage-ver.0.3.1.zipをダウンロードして展開
 - <https://github.com/zoetrope/RtStorage/releases/tag/ver.0.3.1>
 - RtStorage.slnをVisualStudioで開いてソリューションをビルド
 - RtStorage¥bin¥ReleaseのRtStorage.exeを実行すると起動する

RtStorageの使い方

1. RtStorage.exeを実行する
2. ネームサーバーを登録する



3. ポートを登録追加、記録の開始



表計算ソフトとRTCの連携

表計算ソフトによるデータ入出力

どんなデータが出力されているか？

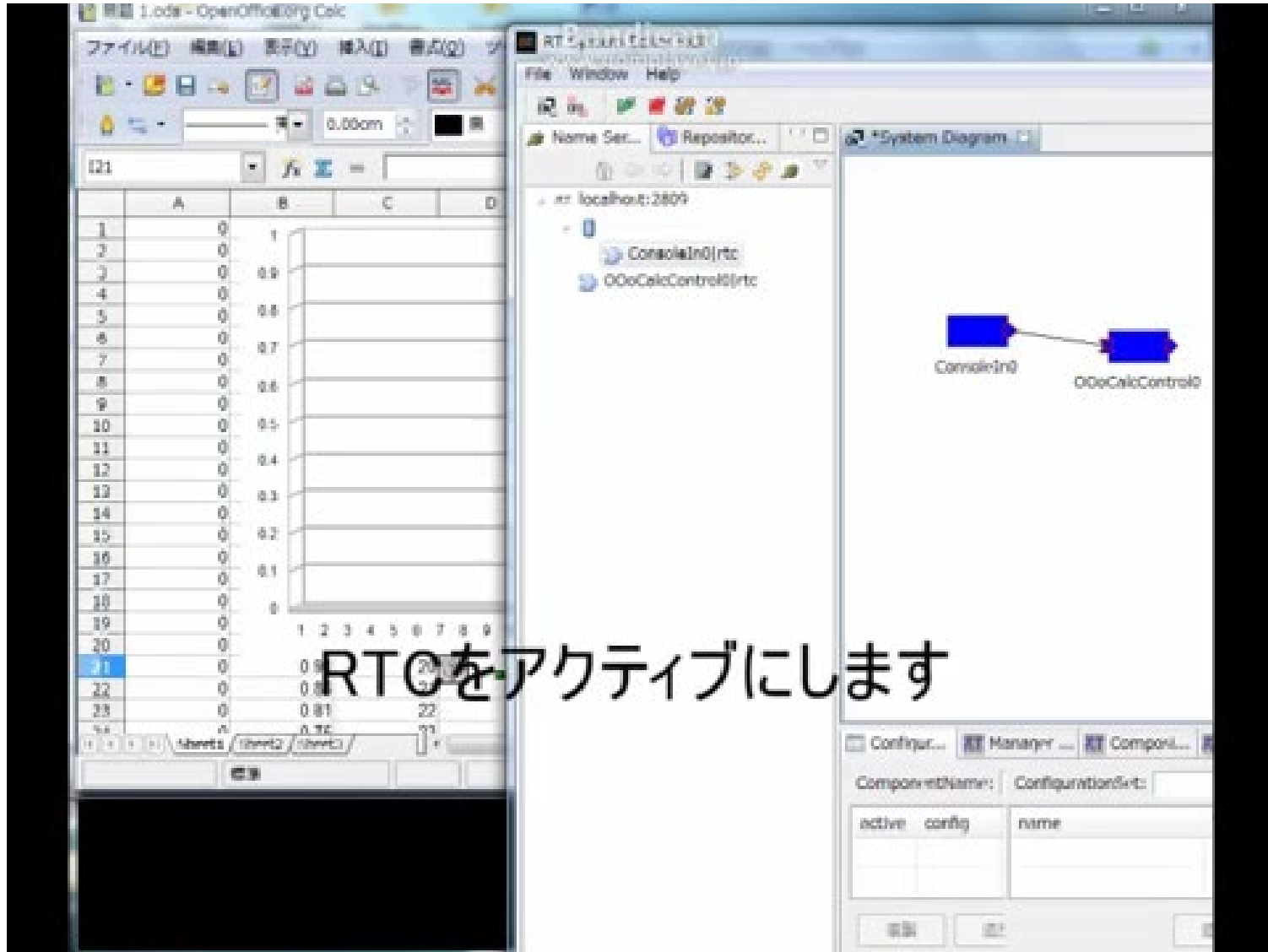
新規に開発したRTC

セルからデータを読み込み

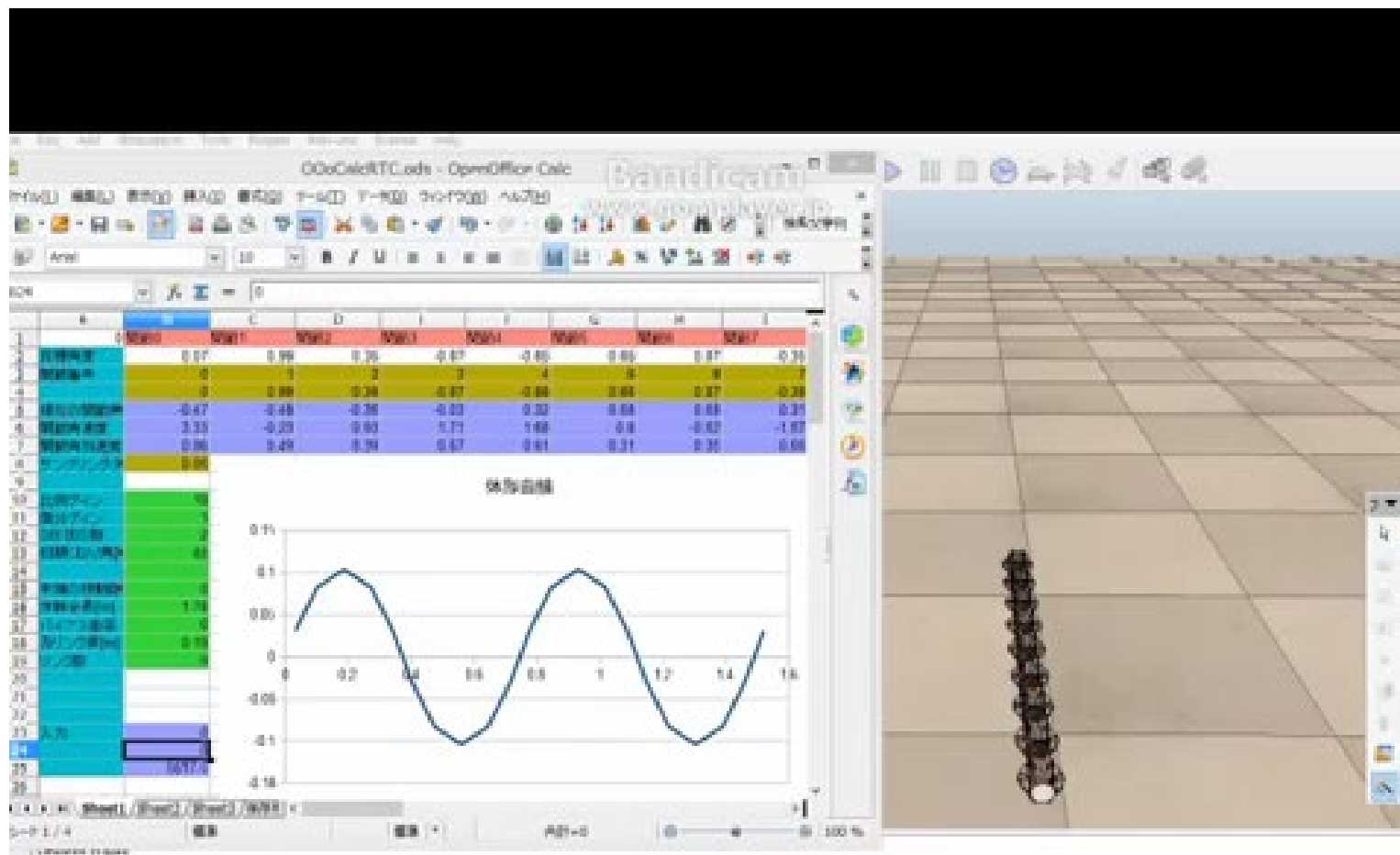
セルにデータを書き込み

入力データ	出力データ
1	5
2	4
3	3
4	2
5	1

デモ動画

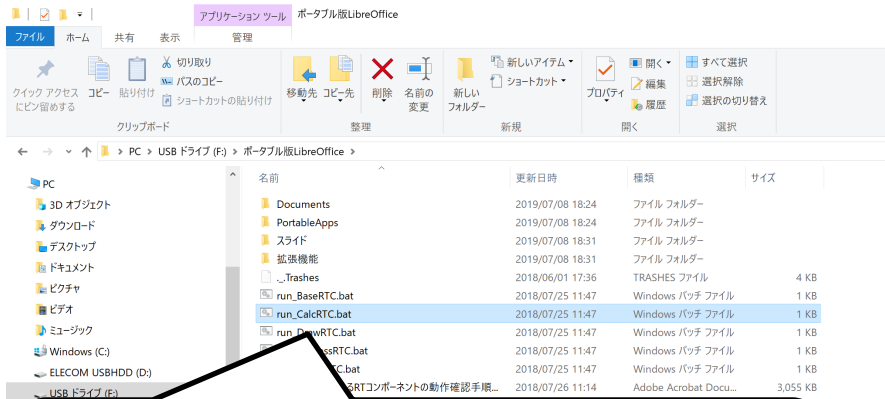


デモ動画

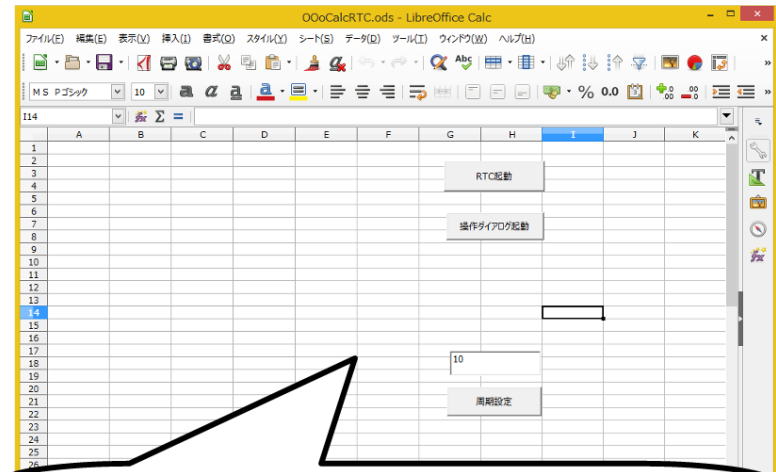


ポータブル版LibreOffice対応RTC

- 配布ファイルに以下のソフトウェアを同梱(Windows)
 - ポータブル版LibreOffice
 - OpenRTM-aist-Python
 - OpenOffice用RTコンポーネント
- 起動手順(Windows)



ポータブル版LibreOffice¥run_CalcRTC.bat
をダブルクリック

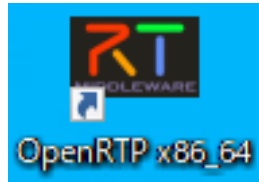


LibreOffice Calcが起動する

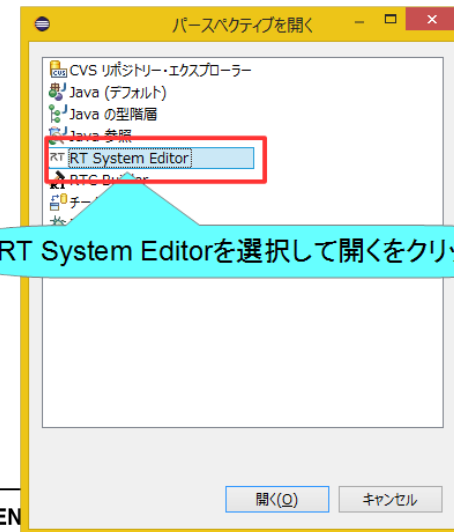
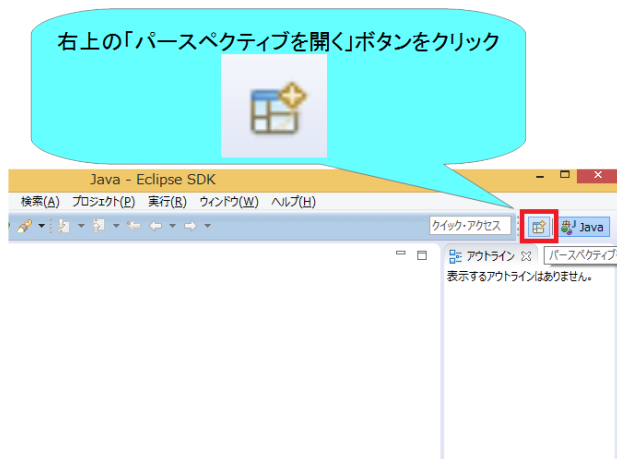
- 起動手順(Ubuntu)
 - OOoRTCs/OOoCalcRTC/OOoCalcRTC.odsをダブルクリック

事前準備

- OpenRTPを起動
 - Windows
 - デスクトップのショートカットをダブルクリックする

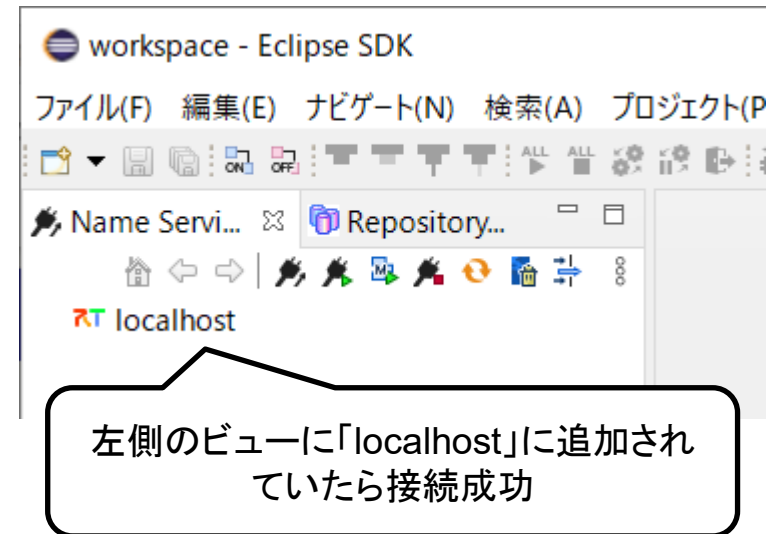
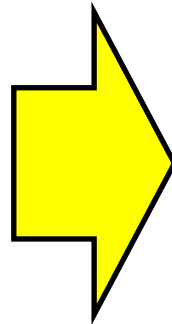
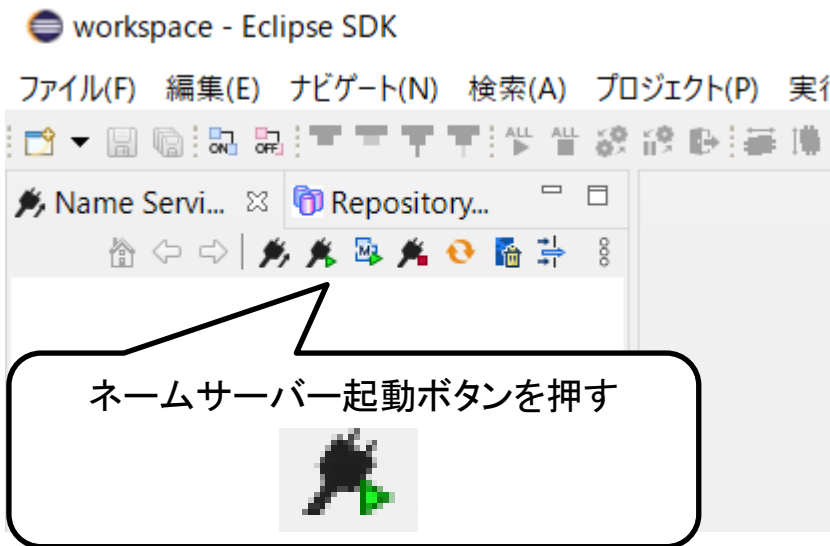


- Ubuntu
 - 以下のコマンドを実行
 - openrtp2



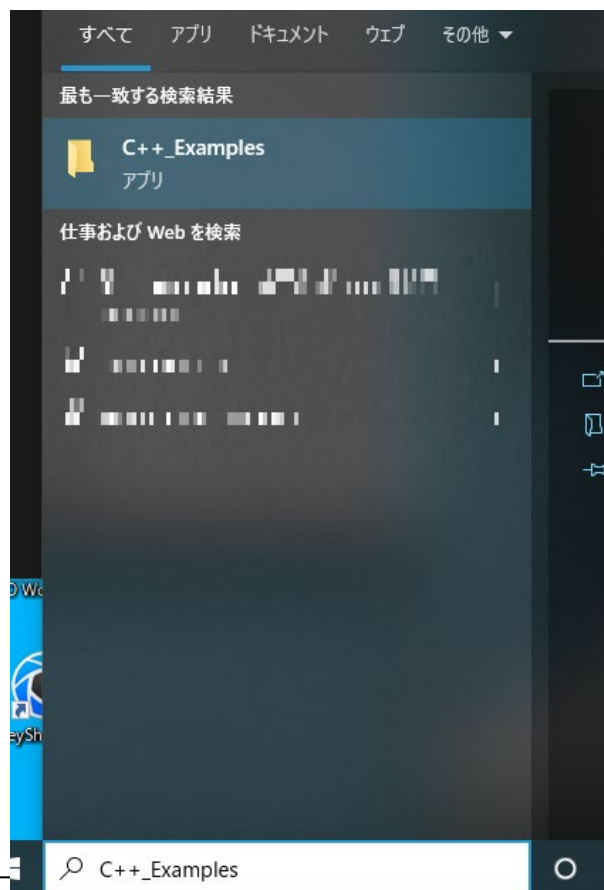
事前準備

- ネームサーバーを起動



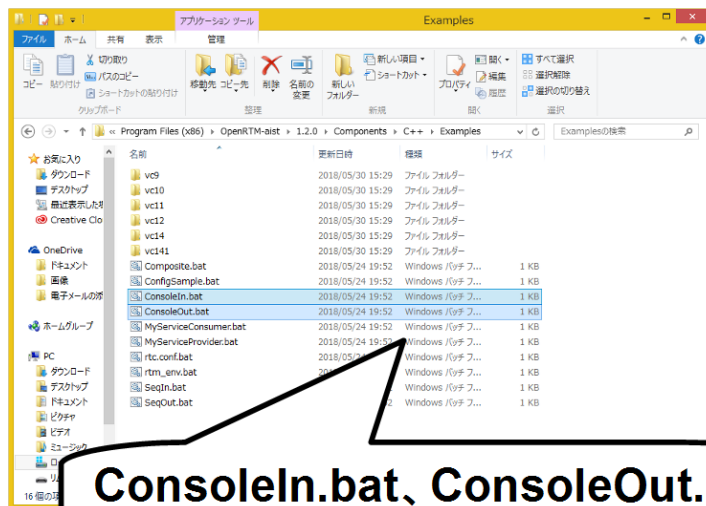
動作確認用のRTCを起動

- ConsoleIn、ConsoleOutのサンプルコンポーネントを起動する
 - Windows 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

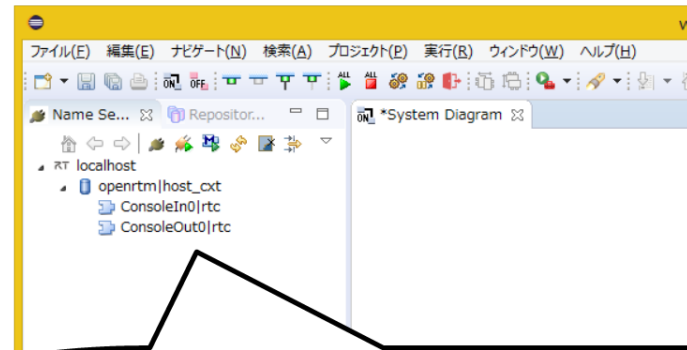


動作確認用のRTCを起動

- ConsoleIn、ConsoleOutのサンプルコンポーネントを起動する
 - Windows 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



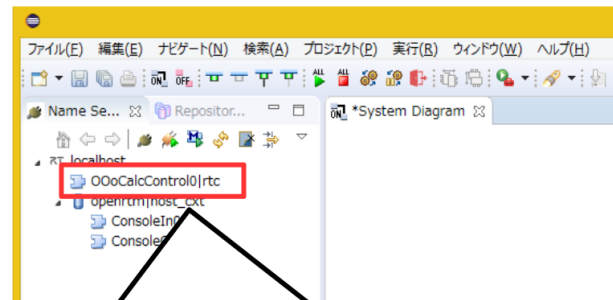
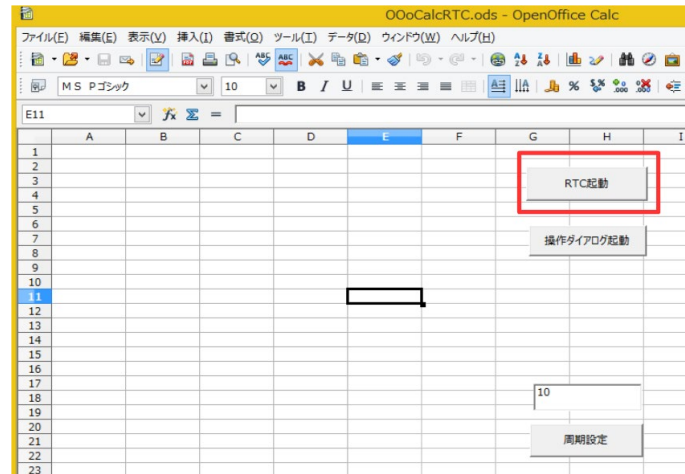
ネームサービスビューにConsoleIn、ConsoleOutが表示されていれば起動成功

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
 - Ubuntu
 - 以下のコマンドを実行
 - /usr/share/openrtm-2.0/components/c++/examples/ConsoleInComp
 - /usr/share/openrtm-2.0/components/c++/examples/ConsoleOutComp

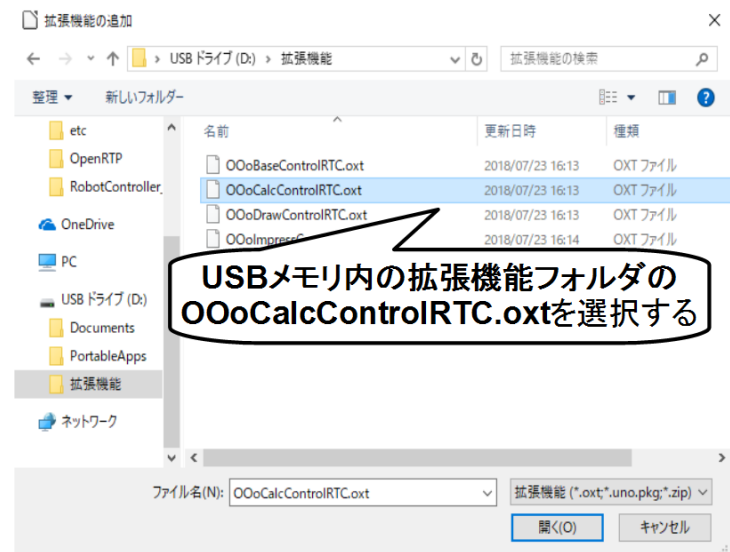
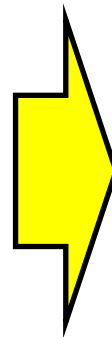
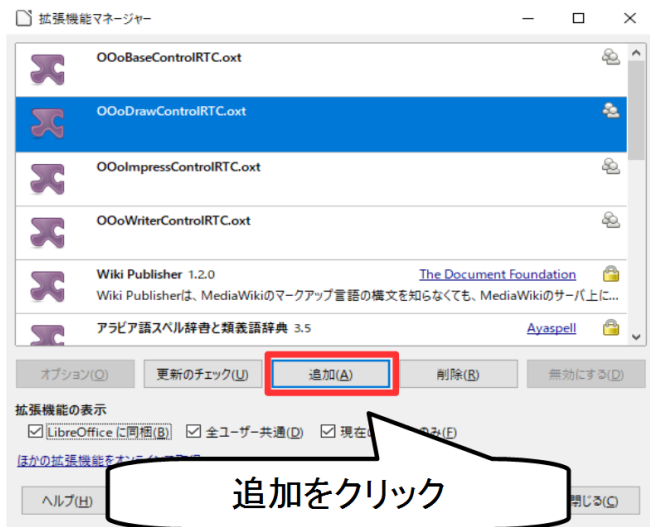
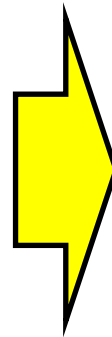
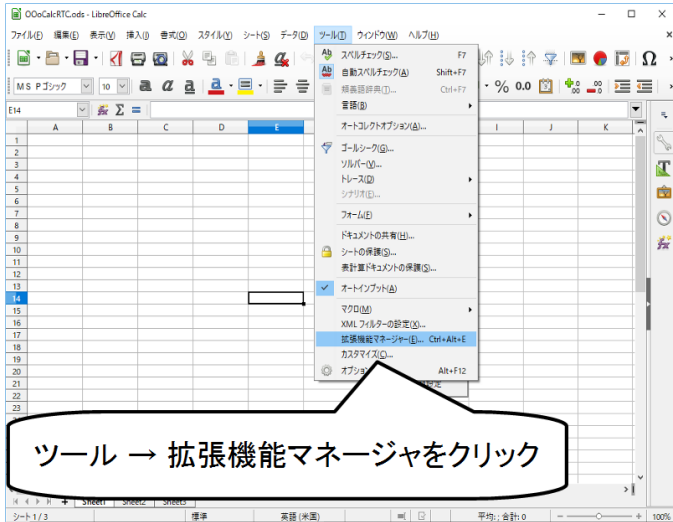
Calc用RTCを起動

- LibreOffice Calcの「RTC起動」ボタンをクリックする



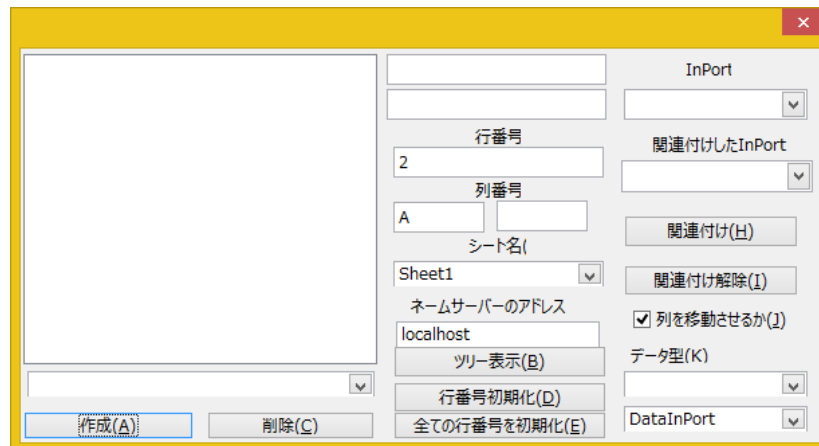
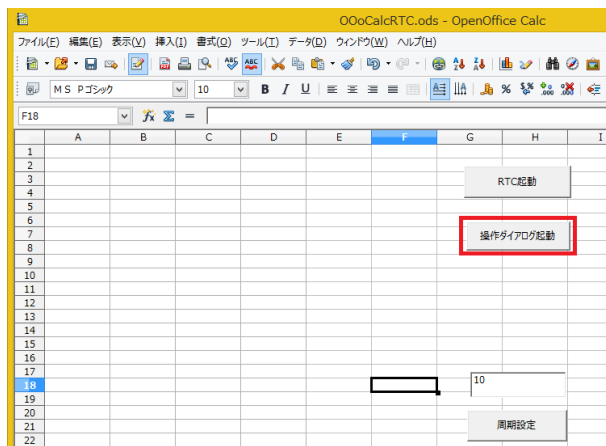
ネームサービスビューにOOoCalcControl0
が表示されていれば起動成功

起動に失敗する場合

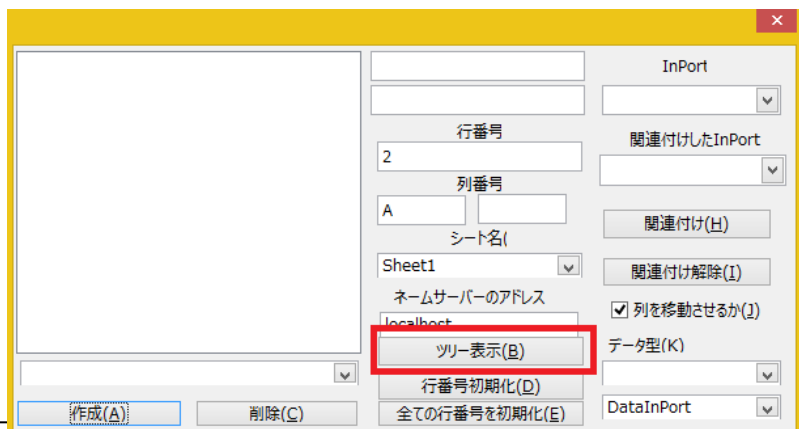


OutPortの接続

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

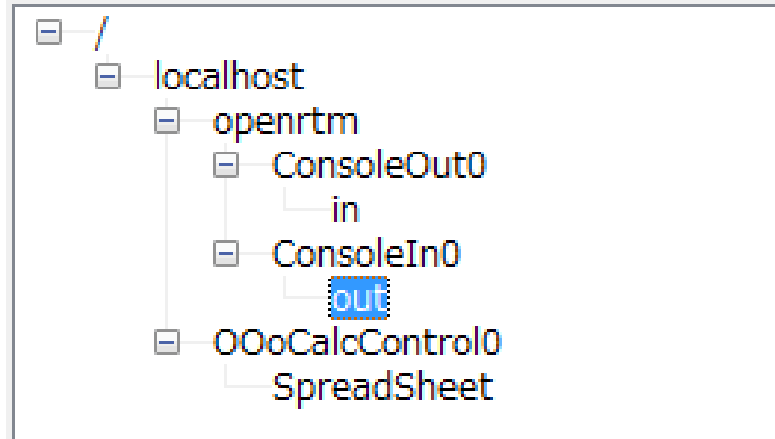


- 「ツリー表示」ボタンをクリックする

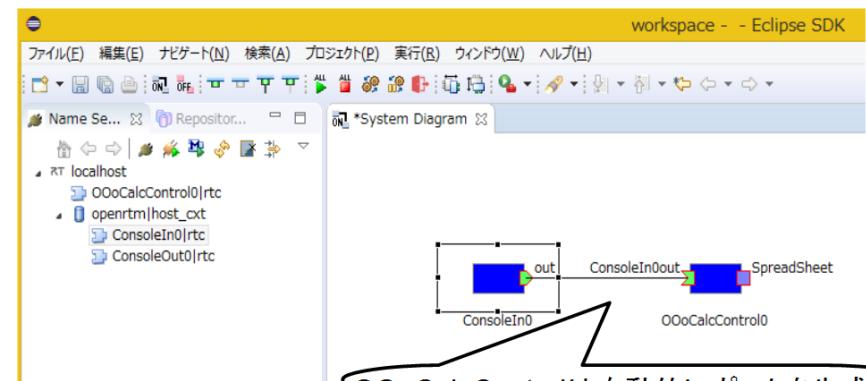
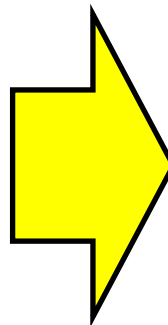
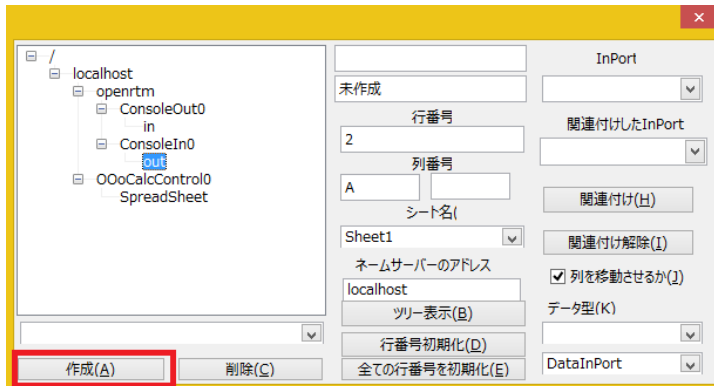


OutPortの接続

- ツリーからConsoleInのoutを選択



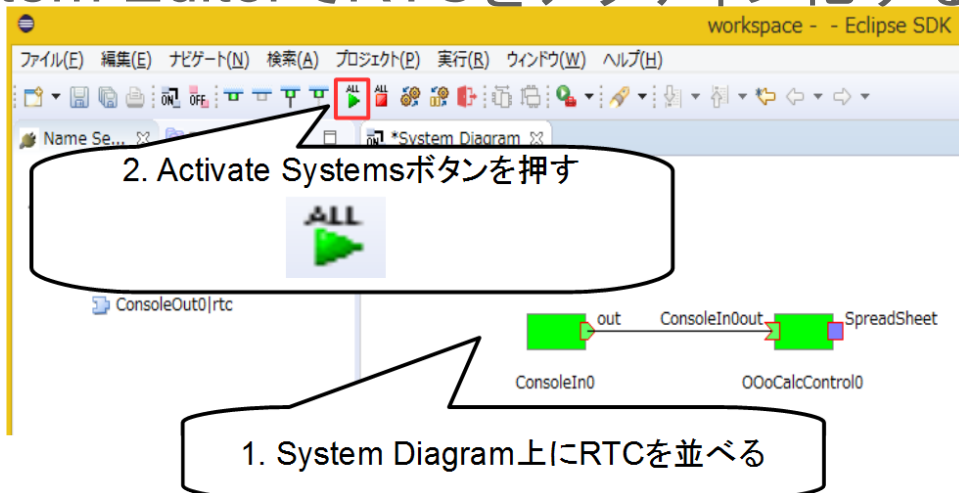
- 「作成」ボタンをクリックする



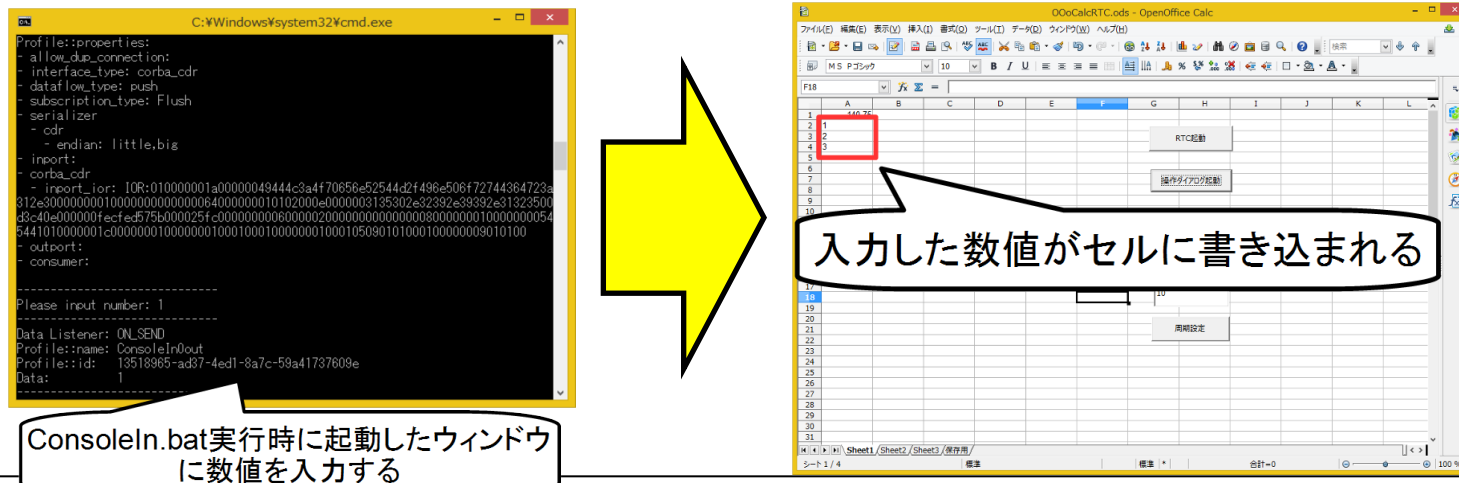
OOoCalcControl0は自動的にポートを生成し、ConsoleInと接続する

OutPortの動作確認

- RT System EditorでRTCをアクティブ化する

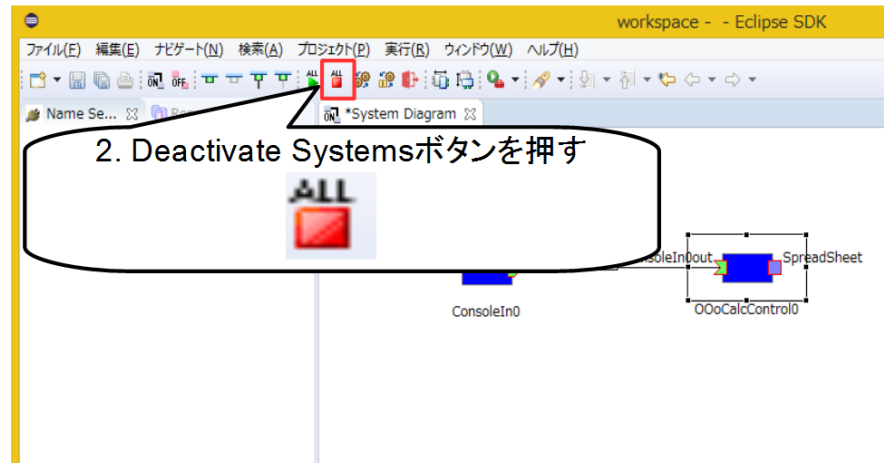


- ConsoleInのウィンドウに数値を入力する



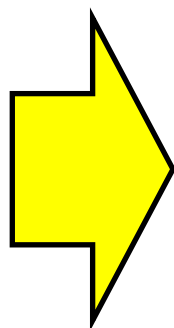
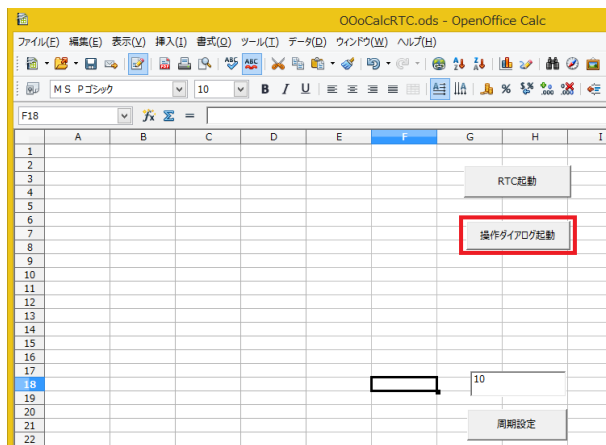
OutPortの動作確認

- 一旦、RTCを非アクティブ化する

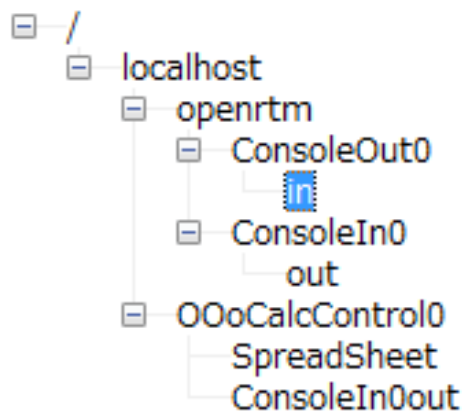


InPortの接続

- 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

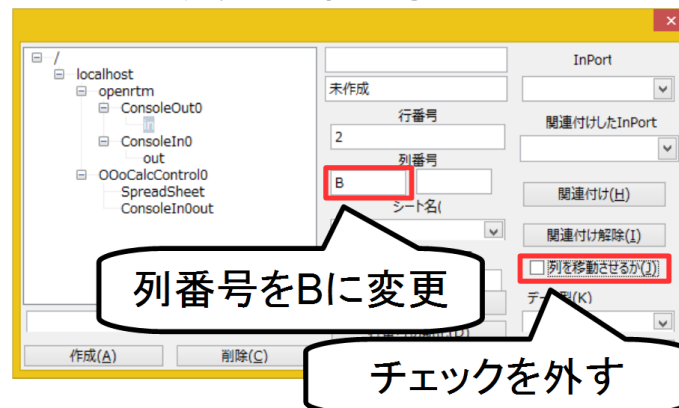


- ツリーから**ConsoleOut**の**in**を選択

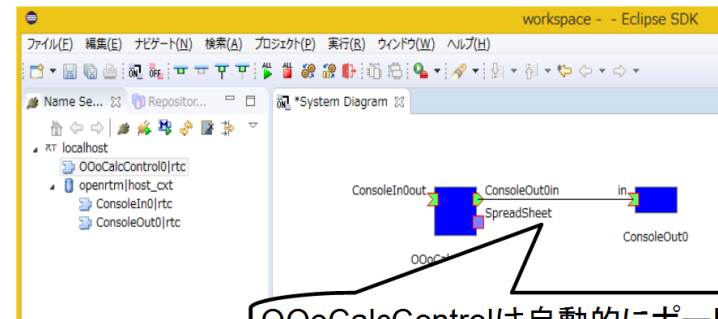
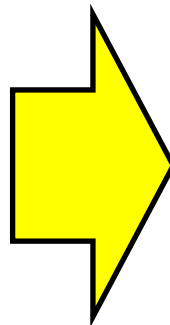
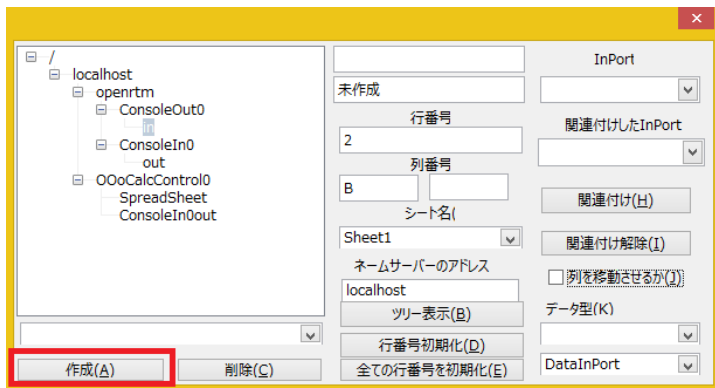


InPortの接続

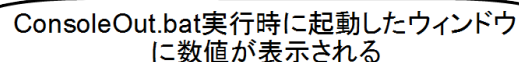
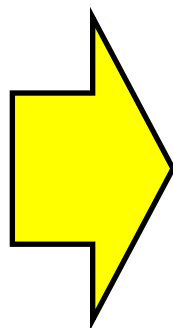
- 列番号をBに設定
- 「列を移動させるか」のチェックを外す
 - ※2回クリックしないとチェックが外れない



- 「作成」ボタンをクリックする

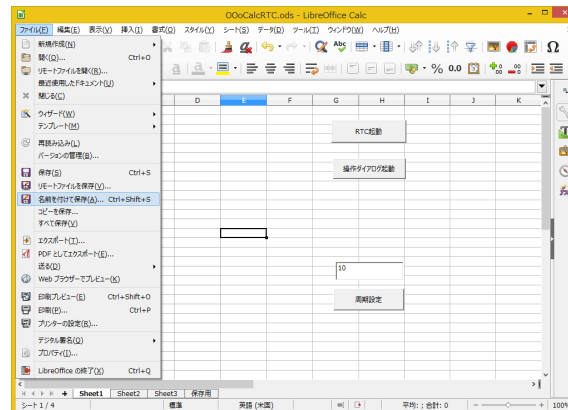


- RT System EditorでRTCをアクティブ化する

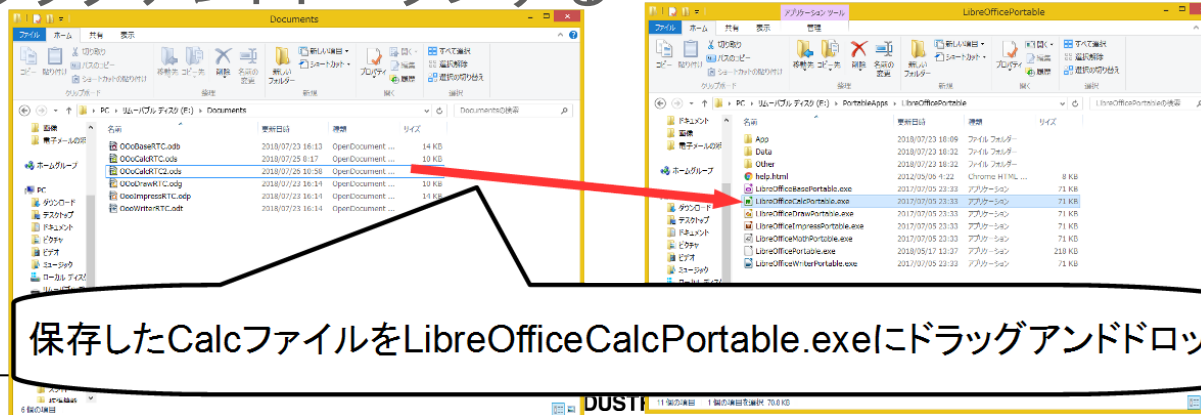


保存

- 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

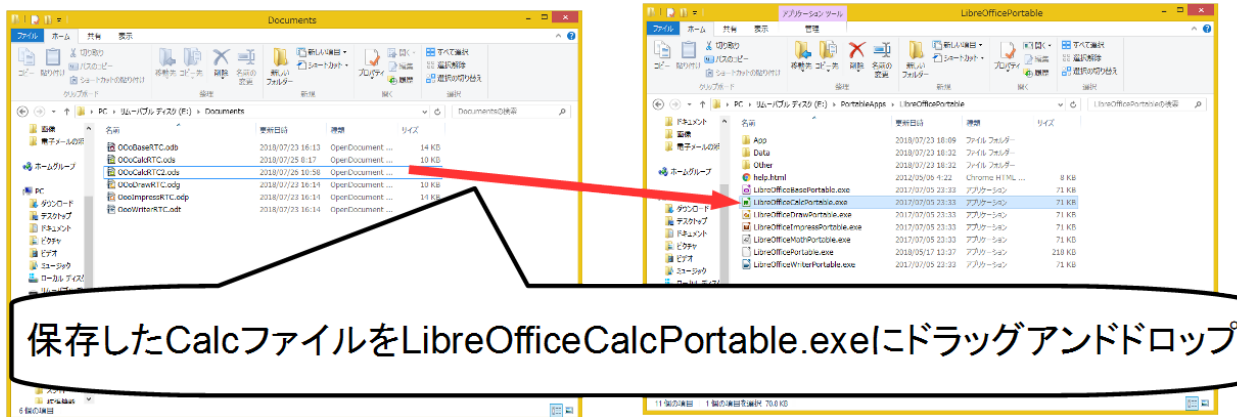


- 保存したCalcファイルをUSBメモリ内の
PortableApps¥LibreOfficePortable¥**LibreOfficeCalcPortable.exe**
にドラッグアンドドロップする

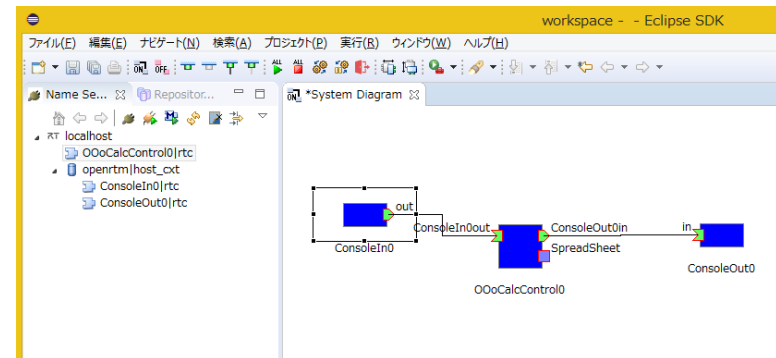
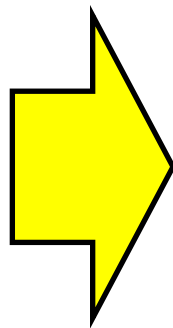
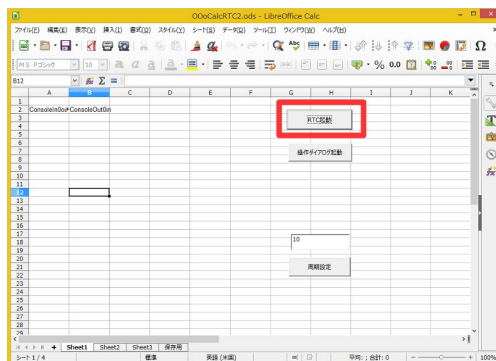


復元

- 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



まとめ

RTCの動作確認、テストで有用なツールを紹介した

- **RtStorage**

- 出力データの保存と再生が簡単
- 一度保存したデータを再利用できるため、例えばセンサRTCのデータを保存しておいて再生することで、InPort側のRTCのテストだけを実行できる(テストでセンサRTCを実行する必要がない)
- rtshellでも同じことはできるので、好きな方を使ってください

- **表計算ソフトとRTCを連携させるためのツール**

- 指定のデータの入力が簡単
- 出力データが視覚的に分かりやすい
- グラフ表示も併用するなど様々な使い方ができる

Lua版RTミドルウェア (OpenRTM Lua)

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**、**LuneScript**で開発可能にする

Lua

```

32 ConsoleIn.new = function(manager)
33     local obj = {}
34     -- RTObjectをメタオブジェクトに設定する
35     setmetatable(obj, {__index=openrtm.RTObject.new(manager)})
36     -- データ格納変数
37     obj._d_out = openrtm.RTCUtil.instantiateDataType("::RTC::TimedLong")
38     -- アウトポート生成
39     obj._outOut = openrtm.OutPort.new("out",obj._d_out,"::RTC::TimedLong")
40
41     -- 初期化時のコールバック関数
42     -- @return リターンコード
43     function obj:onInitialize()
44         -- ポート追加
45         self:addOutPort("out",self._outOut)
46
47         return self._ReturnCode_t.RTC_OK
48     end
49     -- アクティブ状態の時の実行関数
50     -- @param ec_id 実行コンテキストのID
51     -- @return リターンコード
52     function obj:onExecute(ec_id)
53         io.write("Please input number: ")
54         local data = tonumber(io.read())
55         -- 出力データ格納
56         self._d_out.data = data
57         -- 出力データにタイムスタンプ設定
58         openrtm.OutPort.setTimestamp(self._d_out)
59         -- データ書き込み
60         self._outOut.write()
61         return self._ReturnCode_t.RTC_OK
62     end
63
64     return obj
65 end

```

MoonScript

```

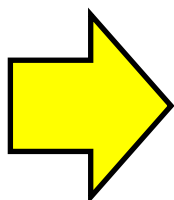
27 class ConsoleIn extends openrtm_ms.RTObject
28     -- コンストラクタ
29     -- @param manager マネージャ
30     new: (manager) =>
31         super manager
32         -- データ格納変数
33         self._d_out = openrtm_ms.RTCUtil.instantiateDataType("::RTC::TimedLong")
34         -- アウトポート生成
35         self._outOut = openrtm_ms.OutPort("out",self._d_out,"::RTC::TimedLong")
36
37     -- 初期化時のコールバック関数
38     -- @return リターンコード
39     onInitialize: =>
40         -- ポート追加
41         @addOutPort("out",self._outOut)
42
43         return self._ReturnCode_t.RTC_OK
44
45     -- アクティブ状態の時の実行関数
46     -- @param ec_id 実行コンテキストのID
47     -- @return リターンコード
48     onExecute: (ec_id) =>
49         io.write("Please input number: ")
50         data = tonumber(io.read())
51         -- 出力データ格納
52         self._d_out.data = data
53         -- 出力データにタイムスタンプ設定
54         openrtm_ms.setTimestamp(self._d_out)
55         -- データ書き込み
56         self._outOut.write()
57         return self._ReturnCode_t.RTC_OK
58
59

```

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

OpenRTM Lua版 デモ動画

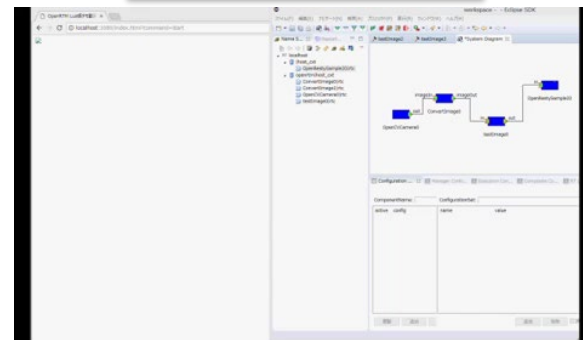
ゲームエミュレータ

OpenRTM Lua版 デモ動画
Laputan Blueprints編

物理シミュレータ

OpenRTM Lua版 デモ動画
V-REP編

ロボットシミュレータ

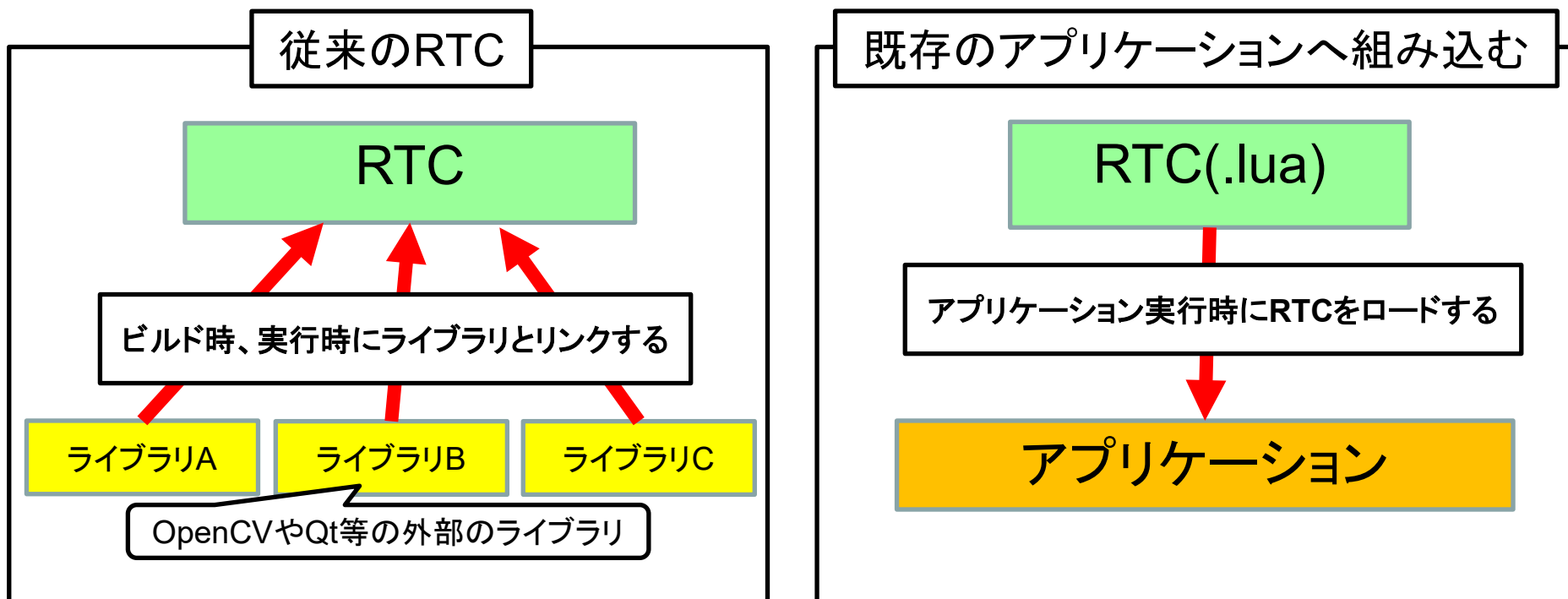


OpenResty上で起動したRTCと
OpenCVCameraコンポーネントを接続します

WEBアプリサーバー

既存のアプリケーションへの組み込み

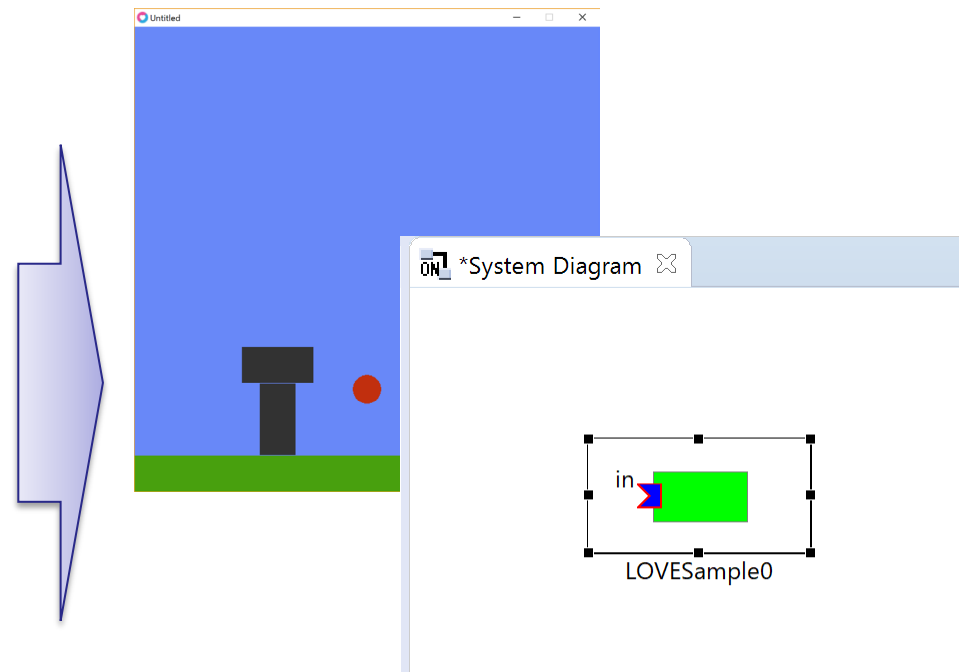
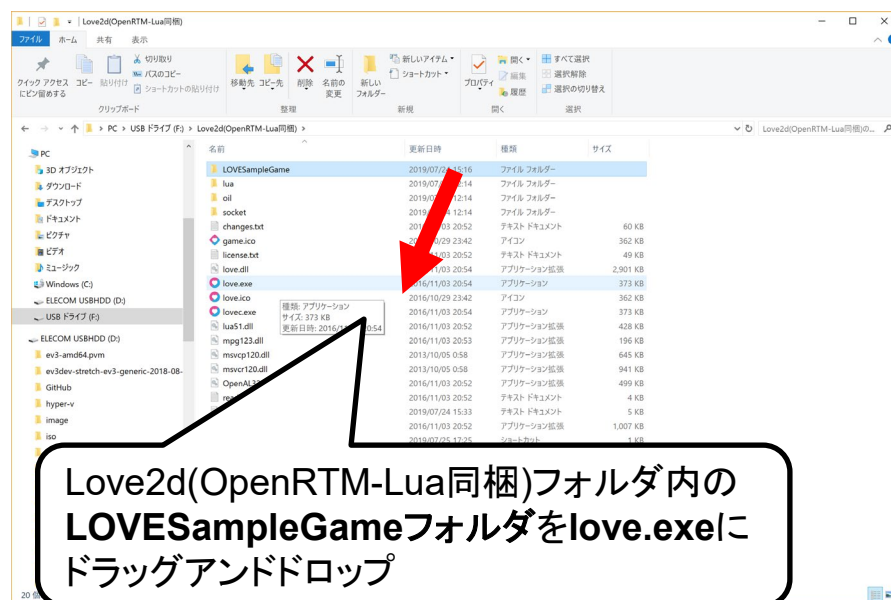
- OpenRTM-Luaは既存のアプリケーションへ組み込むことを得意としている



- V-REP等は自分でビルドすると大変だが、Luaにより容易に機能を追加可能になっている

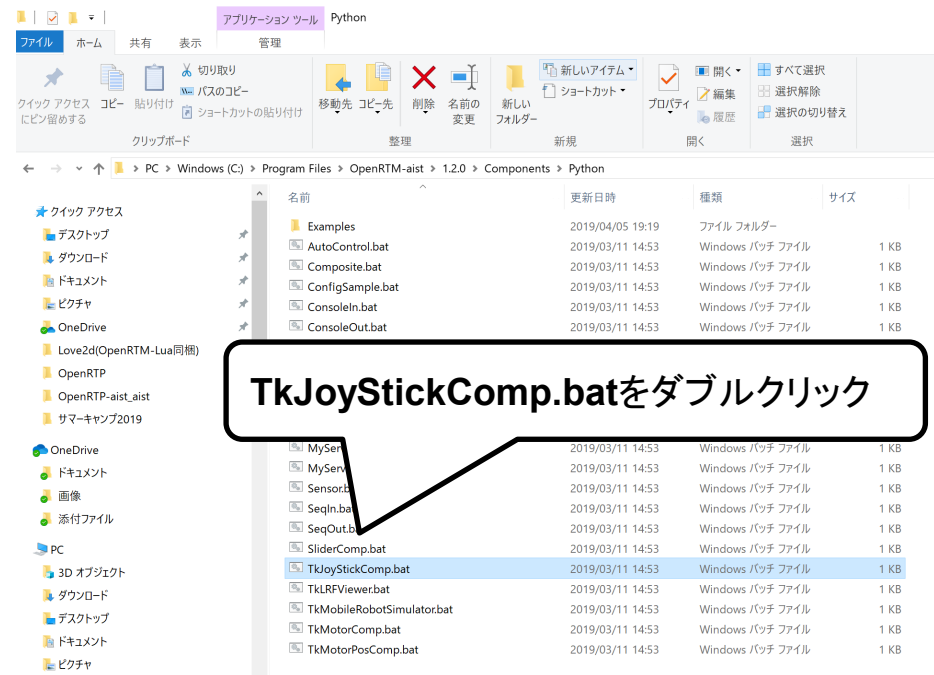
2Dゲームエンジン LÖVEでの応用例

- 配布のUSBメモリに以下のソフトウェアを同梱
 - LÖVE(ゲームエンジン)
 - OpenRTM-Lua
 - LÖVE用RTCのサンプル



2Dゲームエンジン LÖVEでの応用例

- ジョイスティックコンポーネント(OpenRTM-aist Python版 サンプル)を起動して接続

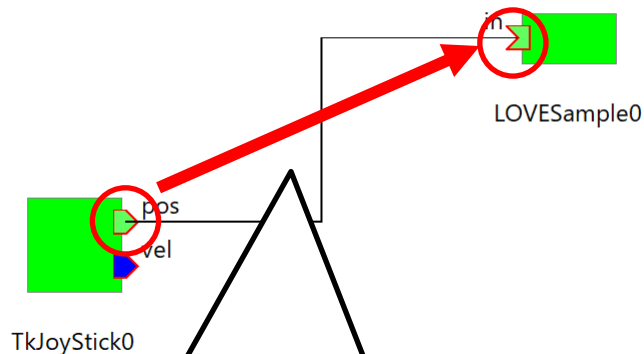


TkJoyStickComp.batをダブルクリック

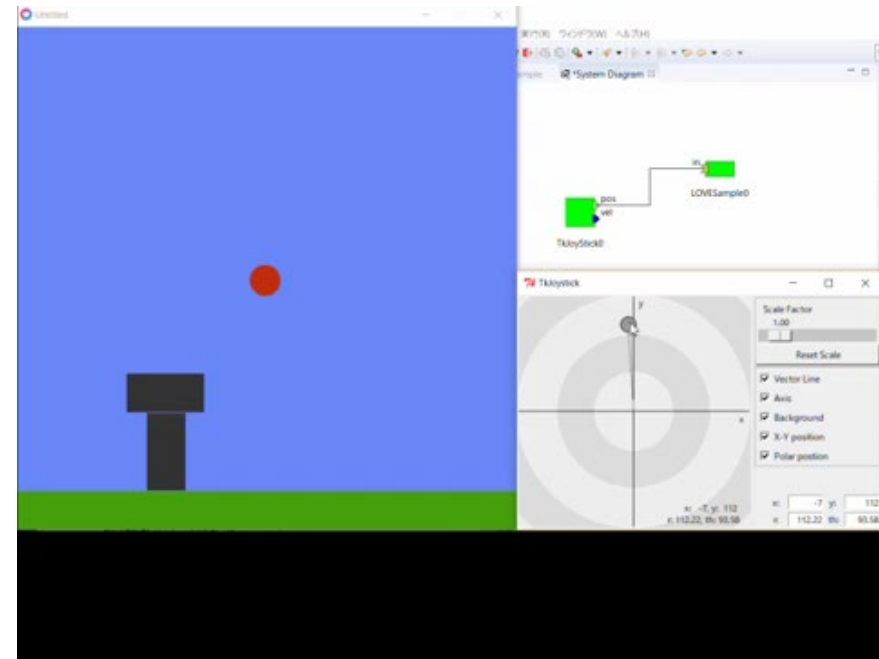
2Dゲームエンジン LÖVEでの応用例

- ジョイスティックコンポーネント(OpenRTM-aist Python版サンプル)を起動して接続

System Diagram



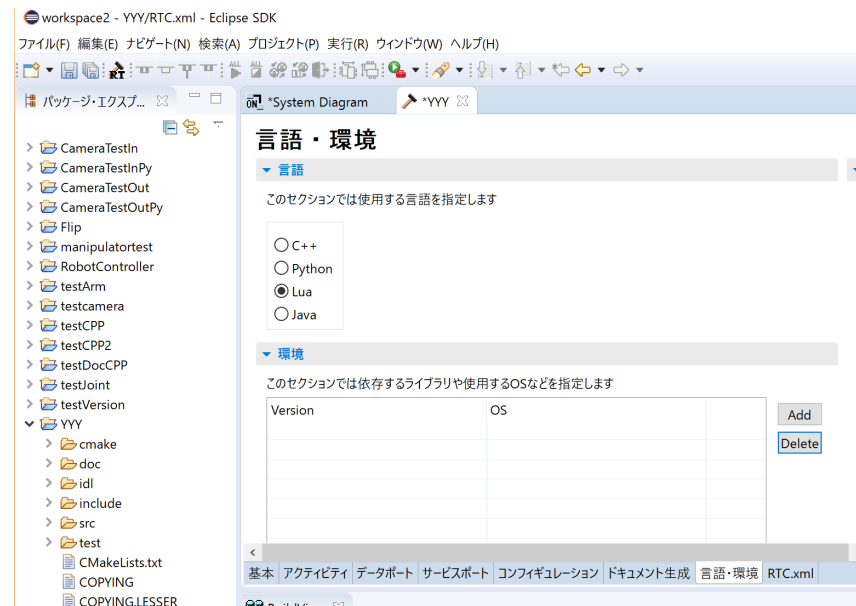
TkJoyStick0のposからLoveSample0のinにドラッグアンドドロップする。
※OpenRTM Luaの仕様上、LOVESample0からTkJoyStick0にドラッグアンドドロップするとエラーになります。



Luaスクリプティング機能をサポートするアプリケーション
(今回はゲームエンジン)とOpenRTM-aistの連携が可能

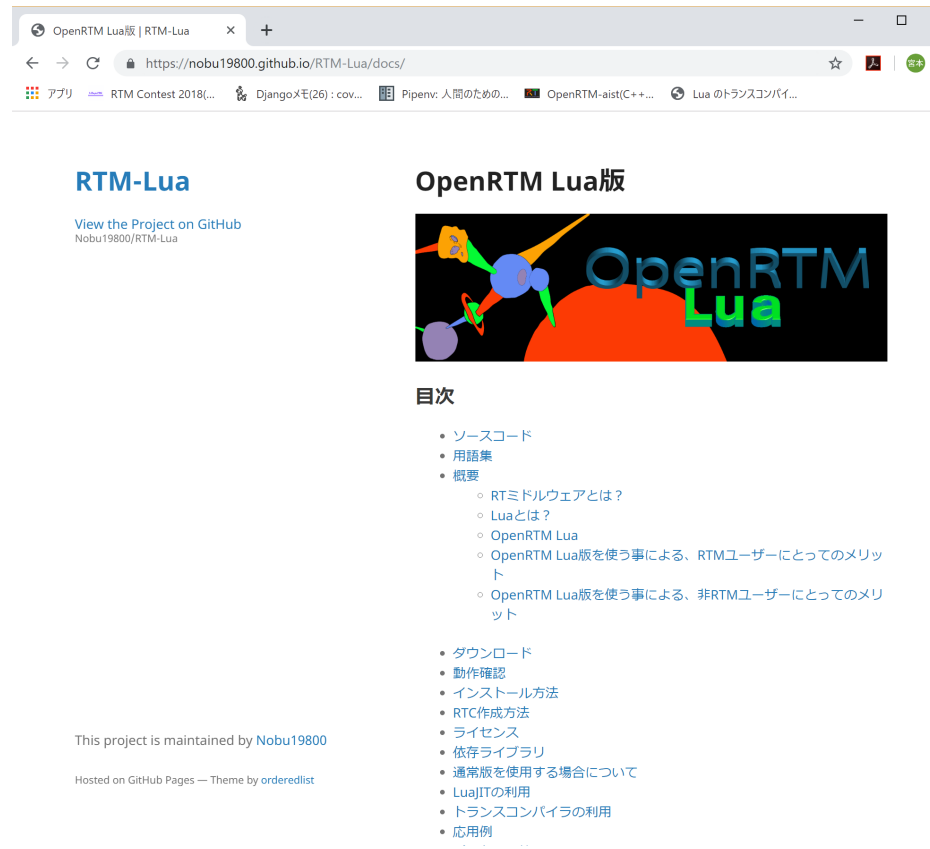
LÖVE用RTCの作成手順

- 以下のファイルが必要
 - LOVESampleGameフォルダ(名前は任意)
 - **main.lua**
 - LÖVEがロードするスクリプトファイル
 - LÖVEがmain.luaの以下の関数を呼び出す
 - » load関数: main.luaのロード時に呼び出し
 - » draw関数: 描画更新時に呼び出し
 - » update関数: フレーム更新時に呼び出し
 - 以下のLOVESample.luaをロードしてRTCを起動する
 - **LOVESample.lua** (名前は任意)
 - RTCの実装
 - RTC Builderで生成できる
 - onExecute関数等を実装する



OpenRTM Lua

- 詳細な説明は以下に記載
 - 「OpenRTM Lua」で検索すると出る



The screenshot shows a web browser displaying the GitHub repository page for OpenRTM Lua. The browser's address bar shows the URL <https://nobu19800.github.io/RTM-Lua/docs/>. The page content is divided into two main sections. The left section, titled "RTM-Lua", includes a link to "View the Project on GitHub" and the text "Nobu19800/RTM-Lua". The right section, titled "OpenRTM Lua版", features a colorful graphic with the text "OpenRTM Lua" and a list of contents (目次). The list includes links to source code, terminology, and a summary, with the summary further detailing RT middleware, Lua, OpenRTM Lua, and the benefits of using OpenRTM Lua for both RTM and non-RTM users. At the bottom of the right section, there are links for downloading, action confirmation, installation methods, RTC creation methods, licenses, dependency libraries, and usage examples. The footer of the page states "This project is maintained by Nobu19800" and "Hosted on GitHub Pages — Theme by orderedlist".

RTM-Lua

[View the Project on GitHub](#)
Nobu19800/RTM-Lua

OpenRTM Lua版

目次

- ソースコード
- 用語集
- 概要
 - RTミドルウェアとは？
 - Luaとは？
 - OpenRTM Lua
 - OpenRTM Lua版を使う事による、RTMユーザーにとってのメリット
 - OpenRTM Lua版を使う事による、非RTMユーザーにとってのメリット
- ダウンロード
- 動作確認
- インストール方法
- RTC作成方法
- ライセンス
- 依存ライブラリ
- 通常版を使用する場合について
- LuaJITの利用
- トランスコンパイラの利用
- 応用例

This project is maintained by [Nobu19800](#)

Hosted on GitHub Pages — Theme by [orderedlist](#)

<https://nobu19800.github.io/RTM-Lua/docs/>