EPSON

Epson RC+ 8.0 拡張機能
Library Builder 8.0

翻訳版

Rev.2
JAM256S7319F

目次

1. はじめに

1.1 はじめに

このたびは当社のロボットシステムをお求めいただきましてありがとうございます。

本マニュアルは、Library Builder を正しくお使いいただくために必要な事項を記載したものです。

システムをご使用になる前に、本マニュアルおよび関連マニュアルをお読みいただき、正しくお使いください。

お読みになった後は、いつでも取り出せる所に保管し、不明な点があったら再読してください。

当社は、厳密な試験や検査を行い、当社のロボットシステムの性能が、当社規格に満足していることを確認しております。マニュアルに記載されている使用条件を超えて、当社ロボットシステムを使用した場合は、製品の基本性能は発揮されませんのでご注意ください。

本書の内容は、当社が予見する範囲の、危険やトラブルについて記載しています。当社のロボットシステムを、安全に正しくお使いいただくため、本書に記載されている安全に関するご注意は、必ず守ってください。

1.2 商標

Microsoft, Windows, Windowsロゴ, Visual Basic, Visual C++は、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

その他の社名, ブランド名, 製品名は、各社の登録商標または商標です。

1.3 表記について

Microsoft® Windows® 10 operating system 日本語版
Microsoft® Windows® 11 operating system 日本語版
本取扱説明書では、上記オペレーティングシステムをそれぞれ、Windows 10, Windows 11と表記しています。また、Windows 10, Windows 11を総称して、Windowsと表記することがあります。

1.4 ご注意

本取扱説明書の一部、または全部を無断で複製や転載をすることはできません。
本書に記載の内容は、将来予告なく変更することがあります。
本書の内容について、誤りや、お気づきの点がありましたら、ご連絡くださいますようお願いいたします。

1.5 製造元

セイコーエプソン株式会社

1.6 お問い合わせ先

お問い合わせ先の詳細は、以下のマニュアルの"販売元"に記載しています。
ご利用の地域によって、お問い合わせ先が異なりますのでご注意ください。
"安全マニュアル - お問い合わせ先"

安全マニュアルは、以下のサイトからも閲覧できます。
URL: https://download.epson.biz/robots/

1.7 ご使用の前に

マニュアルのご使用の前に、知っておいていただきたいことを記載しています。

Epson RC+ 8.0のインストールフォルダーについて

Epson RC+ 8.0は、インストールフォルダーパスを任意の場所に変更が可能です。本マニュアルでは、`C:\EpsonRC80` にEpson RC+ 8.0がインストールされた場合を想定して説明しています。

記号の意味

以下のマークを用いて、安全に関する注意事項を記載しています。必ずお読みください。

⚠ 警告

この表示を無視して誤った取り扱いをすると、人が死亡、または重傷を負う可能性が想定される内容を示しています。

⚠ 警告

この表示を無視して誤った取り扱いをすると、人が感電により、負傷する可能性が想定される内容を示しています。

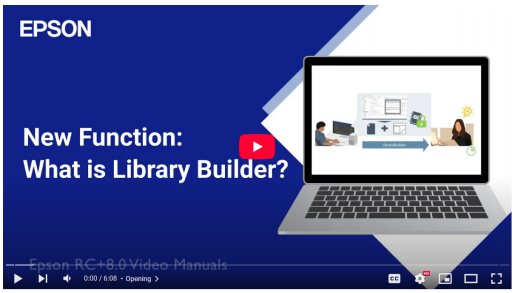
⚠ 注意

この表示を無視して誤った取り扱いをすると、人が傷害を負う可能性が想定される内容、および物的損害のみの発生が想定される内容を示しています。

2. 概要

2.1 概要

Library Builderは、動画でも説明しています。

Title	Link
1. New Function: What is Library Builder?	
2-1. Basic Operation of Library Builder Creator Edition	
2-2. Basic Operation of Library Builder User Edition	
3. Let's try it! Library Builder	

キーポイント

- 動画を閲覧する場合は、インターネット接続が必要です。
- 音声が流れます。
- 母国語の表示は、YouTubeの自動翻訳字幕機能を利用してください。

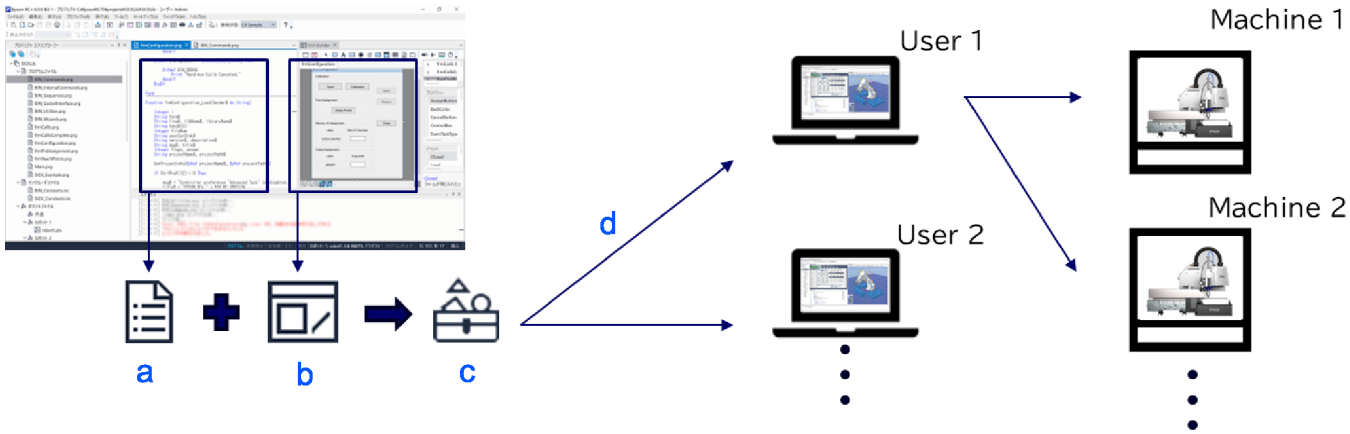
Library Builderは、以下の機能の総称となります。

- SPEL+プロジェクトの構成データ、および構成ファイルをライブラリーとしてパッケージ化する機能
- 作成したライブラリーを、他のSPEL+プロジェクトに追加し利用できる機能

ライブラリー化されたSPEL+プロジェクトの構成要素のうち、以下の構成要素は、ライブラリーを追加したSPEL+プロジェクトから利用できます。

- グローバル変数
- バックアップ変数 (Global Preserve)
- SPEL+関数
- GUI Builderで作成したフォーム
- ユーザーエラー
- プログラムファイル
- インクルードファイル
- ビジョンシーケンス

Library Builderを利用することで、製作装置やサードパーティー機器の設定を補助する独自のライブラリーを作成し、ユーザーへ提供できます。



記号	説明
a	・プログラム ・関数 ・変数
b	・設定画面 ・表示画面
c	ライブラリー
d	提供

キーポイント

ライブラリーの作成、使用には、Epson RC+ 8.0 Ver8.1.0.0 以降が必要です。さらに、ライブラリーの作成には、Epson RC+ Premium Edition のライセンスが必要です。
また、ライブラリーの作成において有用なSPEL+コマンドは、以下のコントローラーファームウェアバージョン以降で使用可能です。

- RC800シリーズ: 8.1.0.0
- RC90, RC700シリーズ： 7.5.5.0
- T, VTシリーズ： 7.5.55.0

SPEL+コマンドの詳細については、以下を参照してください。

- **SPEL+コマンドリファレンス**
- "SPEL+ランゲージリファレンス - Appendix C: Epson RC+ 8.0のコマンド - C-2: Epson RC+ 8.0のバージョンごとに追加されたコマンド一覧"

3. ライブラリーの開発

3.1 ライブラリーの構成要素

- ライブラリー化されるSPEL+プロジェクト構成要素は、パブリックな構成要素とプライベートな構成要素に分けられます。
- パブリックな構成要素は、ライブラリーを追加したSPEL+プロジェクトから直接利用できます。
- プライベートな構成要素は、ライブラリーを追加したSPEL+プロジェクトから直接利用することはできません。
- パブリックな構成要素とプライベートな構成要素は、プレフィックスの有無により区別されます。
- パブリックなプログラムファイルやインクルードファイルは、公開ファイルと呼び、ライブラリーを追加したSPEL+プロジェクトにおいて編集が可能です。

機能	パブリックな構成要素	プライベートな構成要素
ライブラリーを利用するSPEL+プロジェクトからの利用	できる	できない
ライブラリーを利用するユーザーによる編集	できる	できない

- パブリックな構成要素の用途は以下の通りです。

構成要素	パブリックに指定した場合の用途
関数	ライブラリーユーザーに関数として機能を提供する目的で使用します。
グローバル変数	ライブラリーの動作状態を変数の値で示したり、動作に必要な値を指定したりするために使用します。
バックアップ変数	グローバル変数と同じ用途で使用します。値をコントローラー内に保持し、次回起動時に続けて使用したい場合にバックアップ変数にします。
プログラムファイル	・ライブラリーの公開関数の使用例を提示する場合に使用します。 ・ライブラリーのユーザーに、ライブラリー内から呼び出される公開関数(コールバック関数)の処理を記述してほしい場合に使用します。
インクルードファイル	公開関数の引数や戻り値について、ライブラリーユーザーが定数として扱えるように使用します。定数にすることでライブラリーユーザーは値の意味を理解しやすくなります。
ビジョンシーケンス	ライブラリーユーザーにビジョンシーケンスを提供する目的で使用します。

3.2 ライブラリー用プロジェクトの作成

1. ライブラリーの作成には、Epson RC+ Premium Editionのライセンスが必要です。販売元からライセンスキーを購入し、認証を行ってください。
   ライセンスの認証については、以下のマニュアルを参照してください。

   "Epson RC+ 8.0 ユーザーズガイド - システム操作 - Epson RC+ 8.0 の起動 - ライセンス認証"

2. ライブラリー用のプロジェクトを新規作成します。

3. ライブラリーを使用するユーザーに公開したい情報には、プレフィックス (接頭辞)を追加します。プレフィックスは末尾のアンダースコア含めて最大10文字で指定してください。
   公開可能な情報は、以下になります。

情報	公開条件
関数	関数名がプレフィックスから始まること
グローバル変数	変数名がプレフィックスから始まること
バックアップ変数	変数名がプレフィックスから始まること
プログラムファイル	以下の条件をすべて満たすこと ・ファイル名がプレフィックスから始まる ・ファイル内に記載された関数、定数、グローバル変数、バックアップ変数が全てプレフィックスから始まる ・ファイル内に記載されているのが、関数、グローバル変数、バックアップ変数、コメントのみである ・公開条件を満たさないインクルードファイルがインクルードされていない
インクルードファイル	以下の条件をすべて満たすこと ・ファイル名がプレフィックスから始まる ・ファイル内に記載された定数が全てプレフィックスから始まる ・ファイル内に記載されているのが、定数とコメントのみである・公開条件を満たさないインクルードファイルがインクルードされていない
ビジョンシーケンス	シーケンス名がプレフィックスから始まること

例：プレフィックスが"MyLib_"の場合

```
Global Integer MyLib_Counter                    'public global variable
Global Preserve Integer MyLib_WorkPieces        'public global preserve variable

Function MyLib_SomeFunction                     'public function
   'some kind of processing
Fend
```

MyLib_Callbacks.prg : 公開プログラムファイル

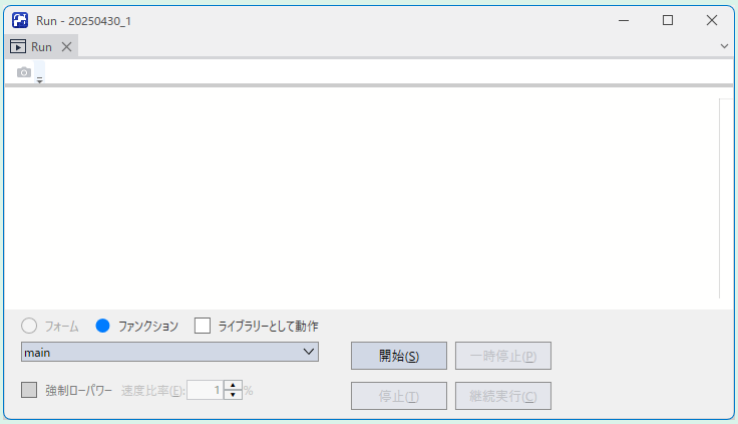
キーポイント

複数ライブラリーを使用する際に関数名や変数名などが重複しないように、ライブラリー固有のプレフィックスを付けてください。

1. 上記公開情報を含め、プロジェクトを開発してください。

キーポイント

Epson RC+ Premium Editionでは、実行ウィンドウにチェックボックス[ライブラリーとして動作]が表示されます。有効にした場合、ライブラリーでのみ使用可能なコマンドをプロジェクトから実行できます。動作確認に活用ください。



2. ライブラリーツールを提供する場合は、[ツール]-[GUI Builder]で、ライブラリーツールとして表示するフォームを作成し、スタートアップフォームに設定します。
ライブラリーツールについては以下を参照してください。

   ライブラリーツールの利用

   GUI Builderの使用方法については、以下のマニュアルを参照してください。

   "Epson RC+ 8.0オプション GUI Builder 8.0"

キーポイント

Epson RC+ Premium Edition では、GUI Builder オプションが無効の場合でも、フォームや GUI Builder 関連コマンドを実行できます。

3. 必要に応じて、ライブラリーのマニュアルとして表示するPDFファイル、ライブラリーツールのアイコンとして表示する画像ファイル、デバイスの設定やマニュアル等の追加ファイルを作成します。

キーポイント

- Epson RC+ 8.0 Ver8.1.0.0では、ライブラリー用プロジェクトに別のライブラリーを登録しないでください。具体的には、ライブラリーAの内部で、別のライブラリーBを使用し、ライブラリーAを作成することはできません。Library Builder 起動時にエラーになります。
- ライブラリー内で使用する設定値を、ライブラリーを使用するユーザーのプロジェクトデータ [任意の名称].libcfg として保持できます。
  例えば、ライブラリーツールを提供する場合に、ユーザーがGUIで設定した値を [任意の名称].libcfg に保持しておき、次回起動時に、同ファイルから前回の設定値を読み込んで、前回の設定状態を再現するように実装できます。
  [任意の名称]には、他のライブラリーと被らない名前を指定してください。

3.3 ライブラリーの作成

1. ライブラリー用プロジェクトを開きます。
2. [拡張]メニューから[Library Builder]を選択します。
3. プロジェクトのビルドに成功すると、[Library Builder]ダイアログが開きます。

キーポイント

TP4の操作モードに応じて下記の通り動作します。
・ AUTO の場合、Epson RC+を使用できますが、Epson RC+ 8.0 Ver8.1.0.0ではライブラリーを作成できません(Library Builderを起動できません)。
・ TEACH, TEST の場合、Epson RC+を使用できないため、ライブラリーも作成できません。
・ TP4の操作モードに関わらず、既存のライブラリーをプロジェクトで使用することは可能です。
TP4の操作モードについては、以下のマニュアルを参照してください。
"ロボットコントローラーオプション ティーチングペンダント TP4マニュアル - 操作モード (TEACH, AUTO,TEST)"

項目	解説
会社名	会社名を入力します。 最大文字数は、半角32文字です。
ライブラリー名	ライブラリー名を入力します。 最大文字数は、半角32文字です。
プレフィックス	プレフィックス名を入力します。 最大文字数は、半角10文字で、アンダースコアで終わる必要があります。
バージョン	ライブラリーのバージョンを入力します。 最大文字数は、半角32文字です。
コメント	ライブラリーの説明を入力します。 最大文字数は、255文字です。
マニュアル	作成するライブラリーのマニュアル (PDFファイル)を設定します。
Distribution License	エプソンから配布される Distribution License を入力します。 [参照]ボタンから、ライセンスが記載されたPDFファイルを読み込むこともできます。
スタートアップフォームの設定	スタートアップフォーム設定画面を表示します。ライブラリーGUI (ライブラリーツール)の初期表示画面として使用するフォームをスタートアップフォームに設定します。
開発時にRC+のツールメニューからライブラリーGUIを使用する	GUI Builderで作成したフォームを、ツールメニューから起動するライブラリーツールとしてライブラリーに組み込みます。プロジェクト内に、スタートアップフォームに設定されたフォームが存在する場合にチェック可能になります。 デフォルトはオフです。 ライブラリーツールについては以下を参照してください。 ライブラリーツールの利用
メニュー表示名	ツールメニューでライブラリーツールとして表示される名前を入力します。 最大文字数は、半角32文字です。
メニュー表示アイコン	ツールメニューでライブラリーツールとして表示されるアイコンを設定します。
スタートアップフォーム起動時に、非常停止や安全扉の状態をチェックする	スタートアップフォーム (ライブラリーツール)起動時に、非常停止や安全扉の状態をチェックします。 デフォルトはオンです。
ツールバーに表示する	ライブラリーツールを起動するアイコンをツールバーに表示します。
公開情報	プレフィックスが付加されたファンクション、グローバル変数、バックアップ変数、プログラムファイル、インクルードファイル、およびビジョンシーケンスを表示します。
追加ファイル	ライブラリーで使用されるデバイスについて、ユーザーが設定、参照しやすいように、デバイスの設定ファイル (フィールドバス設定用ファイル (eds)など)やマニュアルなどを追加します。 `C:\EpsonRC80\Libraries`フォルダ下の当該ライブラリーフォルダ下、AdditionalFilesフォルダにコピーされます。

項目	解説
閉じる	ダイアログを閉じます。
適用	変更を保存します。
元に戻す	前の設定に戻します。
作成	ライブラリーを作成します。
ライブラリー配布	ライブラリー配布ウィザードが起動します。ユーザーに配布するライブラリーと認証キーを出力します。 詳細は、以下を参照してください。 配布用ライブラリーの作成

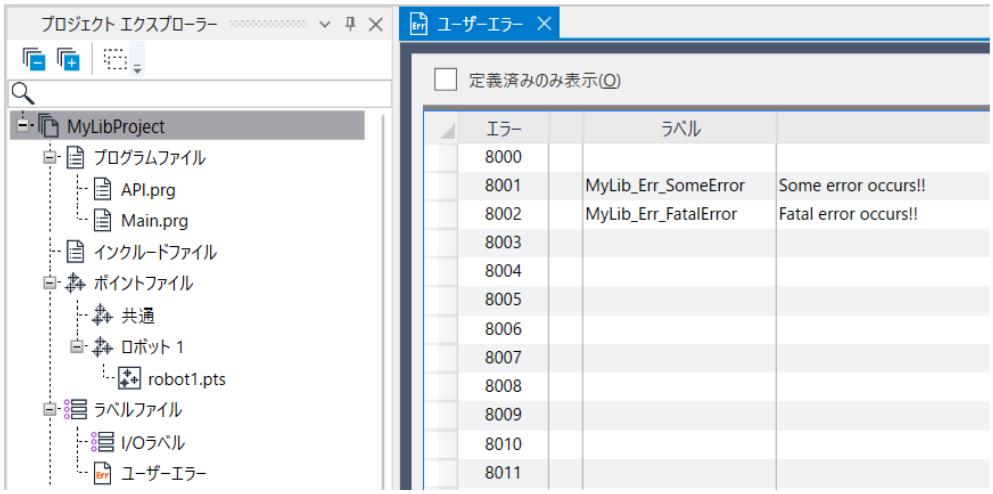
4. プレフィックスを指定します。指定したプレフィックスを含むファンクション、グローバル変数、バックアップ変数、プログラムファイル、インクルードファイル、およびビジョンシーケンスは、公開情報エリアに自動的に追加されます。

> **キーポイント**
>
> プログラムファイル、インクルードファイルは、ファイル名にプレフィックスが付加されていても、ファイル内にプレフィックスが付加されていないファンクションや定数が含まれると、公開情報として表示されません。

5. 会社名、ライブラリー名、バージョン、コメント、マニュアルファイル、追加ファイルを指定します。
6. ライブラリーツールを提供する場合は、[開発時にRC+のツールメニューからライブラリーGUIを使用する]にチェックを入れ、Epson RC+のツールメニューに表示するメニュー表示名、および、メニュー表示アイコンを指定します。
   [スタートアップフォーム起動時に、非常停止や安全扉の状態をチェックする]にチェックを入れると、コントローラーが非常停止状態、安全扉が開いている状態の場合には、警告メッセージを表示し、これらの状態が解除されるまで、ライブラリーツールは起動されません。
7. [適用]ボタンをクリックして設定の変更を保存します。
8. [作成]ボタンをクリックしてライブラリーを作成します。
   `C:\EpsonRC80\Libraries` フォルダの下に、作成したライブラリーファイルとzipファイルが生成されます。

3.4 ライブラリー内で定義したユーザーエラーの発報

1. ユーザーエラーを定義します。
   この時、メッセージだけでなく、ラベルも必ず定義してください。



2. ユーザーエラー発報用関数を作成します。

```
Function RaiseError(errLabel$ As String)
   Integer errNum

   errNum = UserErrorNumber(errLabel$)     ' Get user error number from label
   If errNum <> -1 Then
      Error errNum
   EndIf
Fend
```

UserErrorNumber, Errorについては、以下のマニュアルを参照してください。

"SPEL+ランゲージリファレンス"

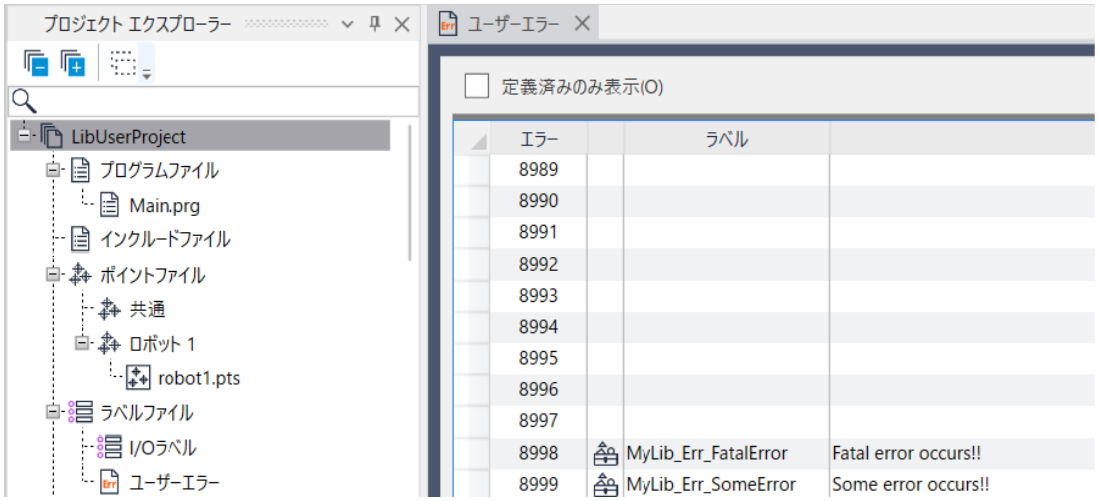
3. エラー処理を実装します。
ライブラリー用プロジェクトのコード内のエラー処理は、上記のユーザーエラー発報関数を呼び出します。

```
Function SomeInternalLibraryFunction
   ' Error occurs so throw an error
   RaiseError("MyLib_Err_SomeError")
Fend
```

4. 下記手順に従って、ライブラリーを作成します。

ライブラリーの作成

5. ライブラリーを使用する際、プロジェクトに上記で作成したライブラリーを登録すると、ユーザーエラーの末尾から順に空いているエラー番号に登録されます。



ライブラリー作成時のエラー番号とは異なりますが、ライブラリー側でユーザーエラーラベルを元にエラー発報しているため、正しいメッセージが表示されます。

3.5 コールバック関数の利用

コールバック関数は、ライブラリーの公開関数の処理中に、プロジェクトの処理を入れ込むことが出来る機能です。
ライブラリーの開発者は、公開プログラムファイルに、ライブラリーが指定する関数名、引数および戻り値で、コールバック関数の雛形(スケルトンコード)と説明を記述します。
ライブラリーユーザーは、プロジェクトにライブラリーを追加した際に公開プログラムファイルが追加されますので、ファイル内の説明を見て、必要な処理を実装します。

例：コールバック関数の記載例

```
Function Lib_Callback(num As Integer) As Boolean
    'XXXの処理の後にライブラリーから呼び出されます。
    'パラメータ: numの値が適切かどうかをチェックして、適切な場合はTrueを、そうでない場合はFalseを戻します。
Fend
```

具体的には、以下の用途があります。

・ライブラリー処理中における、ユーザープログラムによる、後続処理の変更。
【具体例】

- ライブラリーは、入力ビットの値を監視し、一定時間以上Offの状態が続いた場合に、Boolean型の戻り値のコールバック関数を呼び出す。戻り値がFalseで処理を継続し、Trueで処理を終了する。
- ライブラリーユーザーは、コールバック関数で以下のように実装し、後続処理の継続/終了を決定する。
  - 別の入力ビットがOnの場合: MsgBoxコマンドで、オペレーターに継続/終了を選択させて、結果をコールバック関数の戻り値とする。
  - 別の入力ビットがOffの場合: 戻り値をTrueにして、オペレーターによる選択無しで処理を終了させる。

・公開関数の処理が長時間かかる場合の進捗表示と中断。
【具体例】

- ライブラリーは、進捗率を引数で持つ、Boolean型の戻り値のコールバック関数を定期的に呼び出す。戻り値がTrueの場合、処理を終了する。

- ライブラリーユーザーは、GUIビルダーのフォームで、コールバック関数で取得した進捗率をGSetして表示する。また、フォーム上の中止ボタンが押された場合は、コールバック関数が呼び出されたタイミングでコールバック関数の戻り値をTrueにする。

3.6 libcfgファイルを使用した同期

SPELプロジェクトでは、ポイントデータのようにプログラムの動作に必要な設定や算出した値をファイルに保持できる機能があります。
動作中はコントローラーに保存されていますが、RC+が動作するPCのプロジェクトにも随時同期されますので、SPELプロジェクトの開発や保守、他環境への移植などをスムーズに行うことができます。

ライブラリーの利用に際し、操作対象となる機器や機能によって必要な情報が生じる可能性があるため、任意のフォーマットで利用可能なファイルに対し、同様の機能を提供します。
下記情報を保持したい場合に検討ください。

- 機能や機器が参照する設定ファイル
- 機能や機器が算出した結果を保持・参照するファイル
- 機能や機器の動作状態を保持・監視するためのログファイル

拡張子が"libcfg"のファイルをコントローラー内で操作すると、RC+接続時に同期の確認を行ってPC側に取り込みます。書式の制限はありません。

例：libcfgファイルの使用例: ファイルの書き出し

```
Function WriteSettingsToFile(fileName$ As String, … , paramB As Integer)
   Integer iFileID

   iFileID = FreeFile
   ChDisk FLASH                      'コントローラーのプロジェクトフォルダーを指定
   WOpen "Lib1.libcfg" As #iFileID  '必要なモードで開く

   Print #iFileID, "Name: ", name$  '設定の種類と値を書き込みます

   'パラメーターの数だけ繰り返します。

   Print #iFileID, "ParamB: ", paramB  '設定の種類と値を書き込みます
   Close #iFileID
Fend
```

例：libcfgファイルの使用例: ファイルの読み出し

```
   Function ReadSettingsFromFile(fileName$ As String, ByRef name$ As String, … ,
ByRef paramB As Integer)
      Integer iFileID
      Integer iPos
      String Buf$

      iFileID = FreeFile
      ChDisk FLASH                      'コントローラーのプロジェクトフォルダーを指定
      ROpen "Lib1.libcfg" As #iFileID  '必要なモードで開く

      Line Input #iFileID, Buf$
      iPos = InStr(Buf$, ": ")      '設定の種類と値を切り分けて読み込みます。
      name$ = Mid$(Buf$, iPos + 2)

      'パラメーターの数だけ読み込みます。
```

```
    Line Input #iFileID, Buf$
    iPos = InStr(Buf$, ": ")      '設定の種類と値を切り分けて読み込みます。
    paramB = Val(Mid$(Buf$, iPos + 2))

    Close #iFileID
Fend
```

使用する機器の仕様などで特定の拡張子を利用したい場合は「UploadFileAfterStop」コマンドを利用ください。
ライブラリー内に記述することでタスク終了後に指定されたファイルをコントローラー内から取り込みます。ただし、動作にはRC+の接続が必要になります。

4. 配布用ライブラリーの作成

4.1 配布用ライブラリーの作成

ライブラリーの使用を希望するユーザー (以下、ライブラリーユーザー)に配布するための、ライブラリーの作成方法を説明します。

ライブラリーの配布に際し、利用できるライブラリーユーザーを特定することができます。配布形態に応じたライブラリーの価格設定を行う場合などに利用ください。

記号	利用可能なライブラリーユーザー	用途	必要な情報
a	指定したコントローラーの利用者のみ	特定のコントローラーでの利用を許可します。 ライブラリーユーザーは動作させたいコントローラーの台数分のライブラリーを購入し、有効化する必要があります。 ライブラリー開発者は台数分のシリアル番号を入手し、認証キーを発行してください。	Distribution License, ライブラリーユーザーのコントローラーのシリアル番号(ライブラリーユーザーのRC+からも出力可能です。事前に入手してください)
b	指定したPCの利用者のみ	特定のPCで動作するRC+の仮想コントローラーでの利用を許可します。 実機のコントローラーとは異なります。 ライブラリーユーザーは動作させたいPCの台数分のライブラリーを購入し、有効化する必要があります。 ライブラリー開発者は台数分のPCハードウェアIDを入手し、認証キーを発行してください。	Distribution License, ライブラリーユーザーのPCハードウェアID (ライブラリーユーザーのRC+から出力可能です。事前に入手してください)
c	認証キーを受け取った全ての利用者	ライブラリーユーザーはコントローラーやPCの制限なく利用できます。 ライブラリーユーザーは有効化を行う必要があります。	Distribution License, ライブラリー開発者が定めた任意の文字列 (設定のみに使用しますライブラリーユーザーの有効化には不要です)
d	ライブラリーを受け取った全ての利用者	ライブラリーユーザーはコントローラーやPCの制限なく利用できます。 ライブラリーユーザーは有効化をせずに利用可能です。	Distribution License

1. 配布するライブラリーのプロジェクトを開きます。

2. [拡張]メニューから[Library Builder]を選択します。

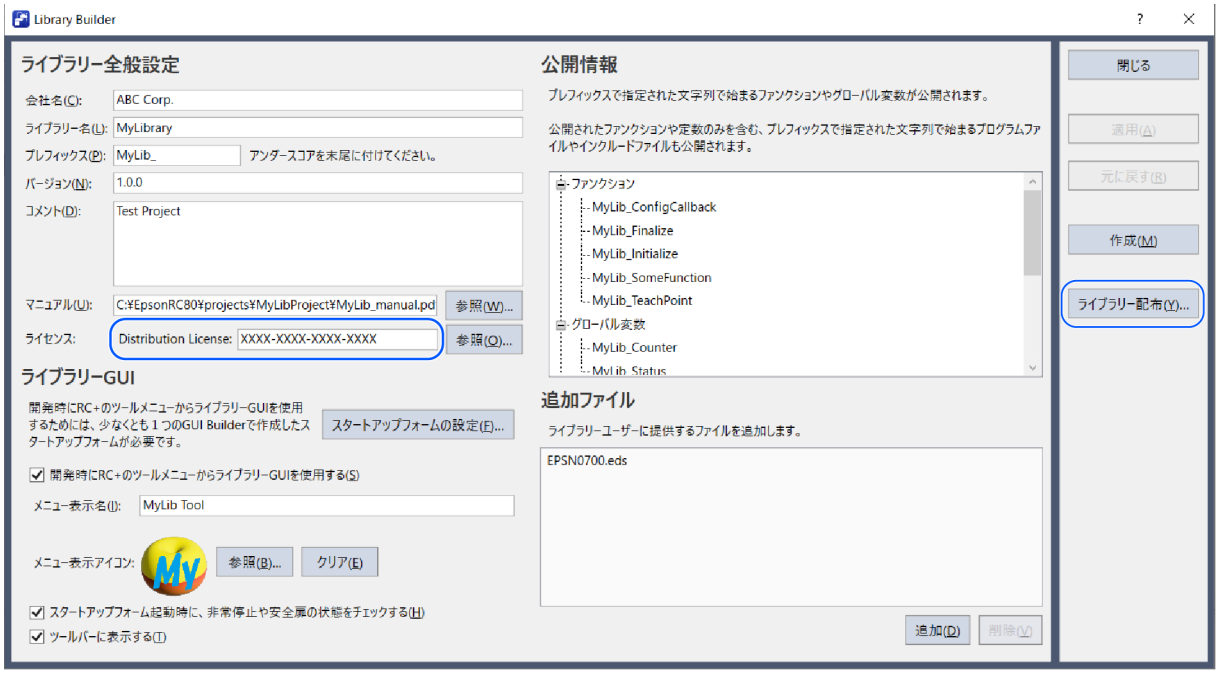
3. プロジェクトのビルドに成功すると、[Library Builder]ダイアログが開きます。

キーポイント

Epson RC+ 8.0 Ver8.1.0.0では、TP4 (操作モード：AUTO)でLibrary Builderを使用できません。
TP4 (操作モード：AUTO)については、以下のマニュアルを参照してください。

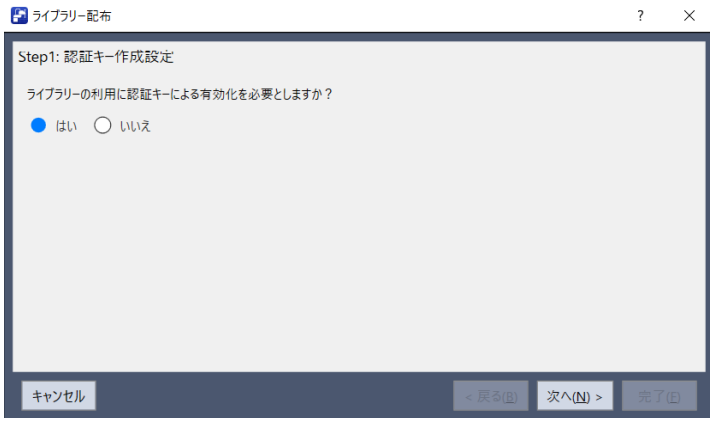
"ロボットコントローラーオプション ティーチングペンダント TP4マニュアル - 操作モード (TEACH, AUTO,TEST)"

4. エプソンから配布される Distribution License を入力します。



5. [ライブラリー配布]ボタンをクリックします。

6. 表示された[ライブラリー配布]ウィザードにおいて、配布するライブラリーに対して、認証キーによる有効化を必要とするかどうかを選択します。



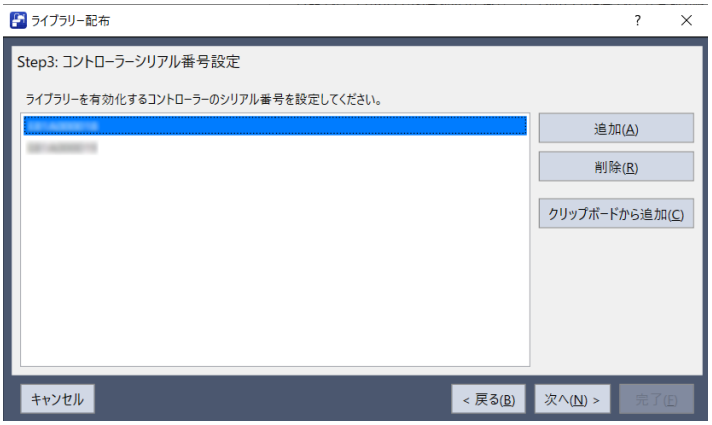
- [はい] : 利用可能なライブラリーユーザーがd以外の場合に選択します。次の手順へ進んでください。
- [いいえ] : 利用可能なライブラリーユーザーがdの場合に選択します。手順9へ進んでください。

7. 利用を特定したいライブラリーユーザーに応じて、ライブラリーを有効化するデバイスを選択します。

- [コントローラー] : 利用可能なライブラリーユーザーがaの場合に選択します。
- [PC(仮想コントローラー)] : 利用可能なライブラリーユーザーがbの場合に選択します。
- [指定しない] : 利用可能なライブラリーユーザーがcの場合に選択します。

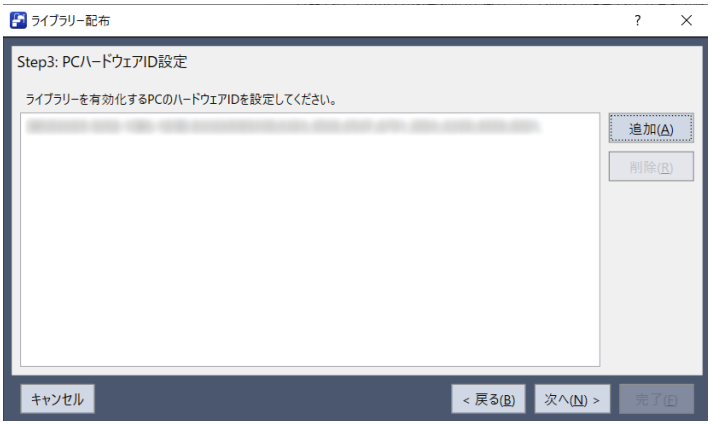
8. 選択したデバイスに応じて、下記情報を入力します。

- **[コントローラー]を選択した場合:**
  [追加]ボタンから、ライブラリーユーザーから取得した、コントローラーのシリアル番号が記載されたcsvファイルを読み込みます。
  [クリップボードから追加]ボタンをクリックすると、クリップボードの文字列が追加されます。シリアル番号をコピー＆ペーストする際にご使用ください。



> **キーポイント**
>
> 複数のシリアル番号を読み込み、一度に、複数コントローラーの認証キーを生成することができます。

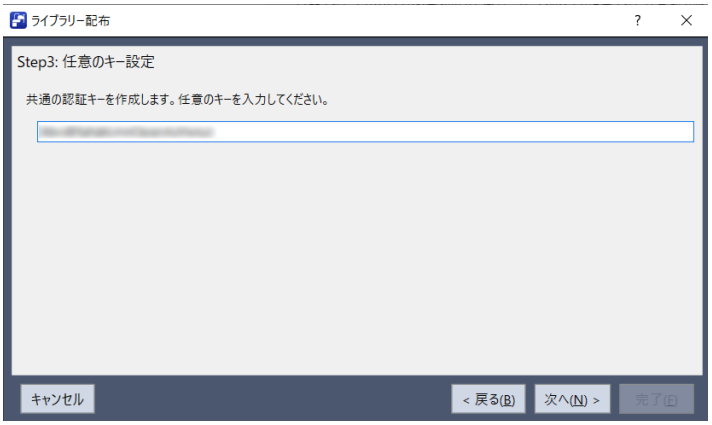
- **[PC (仮想コントローラー)]を選択した場合:**
  [追加]ボタンから、ライブラリーユーザーから取得した、PCハードウェアIDが記載されたcsvファイルを読み込みます。



キーポイント

複数のPCハードウェアIDを読み込み、一度に、複数PCの認証キーを生成することができます。

- **[指定しない]を選択した場合:**
  任意のキーを入力します。最大32文字の英数字とアンダースコア、ハイフンが使用できます。
  キーを入力しなくても「c. 登録を行った利用者のみが利用可能」な配布ライブラリーは作成可能です。

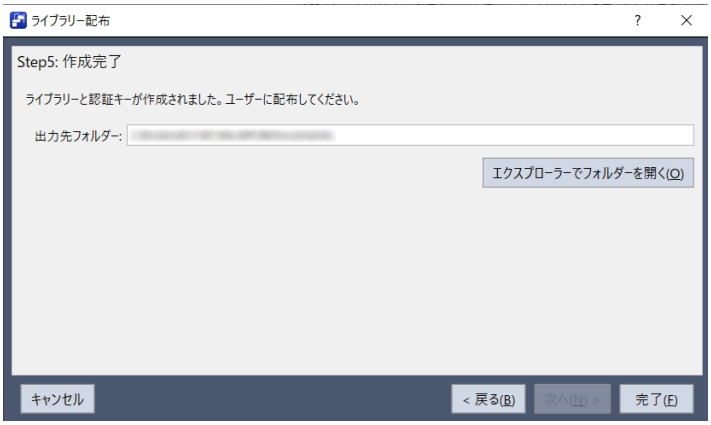


キーポイント

生成される認証キーは、デバイスに関わらず、ライブラリーを有効化できるようになります。

9. ライブラリーと認証キーの出力先を指定します。

10. ライブラリーと認証キーが出力されます (手順6で[いいえ]を選択した場合は、認証キーは出力されません)。
    [エクスプローラーでフォルダを開く]ボタンをクリックすると、出力先フォルダーが表示されます。



5. ライブラリーの使用

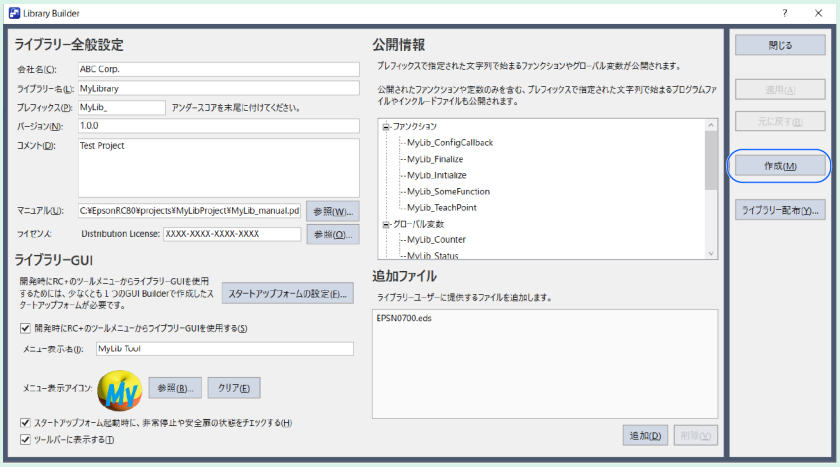
5.1 ライブラリーのインポート

入手したライブラリーをEpson RC+に取り込む方法を説明します。

1. ライブラリーは、zipファイル形式で配布されます。
2. [拡張]メニューから[ライブラリーのインポート]を選択します。
3. 表示されたファイル選択ダイアログからインポートするライブラリーを選択します。
4. [開く]をクリックします。`C:\EpsonRC80\Libraries`フォルダの下に、インポートしたライブラリーファイルとzipファイルが生成されます。既に同名のライブラリーが存在した場合は、上書きを確認するメッセージが表示されます。

キーポイント

同一のEpson RC+において、Library Builder の[作成]ボタンで作成したライブラリーは、インポート不要です。



5.2 ライブラリーの有効化

認証が必要なライブラリーでは、ライブラリーの有効化が必要です。ここでは、インポートしたライブラリーの有効化方法を説明します。
ライブラリーの有効化には、ライブラリーの認証キーが必要です。
有効化しない場合、そのライブラリーは使用できません。認証不要のライブラリーは、有効化することなく、そのまま使用できます。

キーポイント

認証キーは、ライブラリークリエーターに発行してもらいます。必要に応じて、以下のいずれか、もしくは、両方の情報を、ライブラリークリエーターに送付してください。

- **PCのハードウェアID**
  [セットアップ] - [ライセンス設定]画面において、[PC] - [ライブラリー]で表示される[ハードウェアID出力]ボタンによりcsvファイルとして保存できます。

- **コントローラーのシリアル番号**
  [セットアップ] - [ライセンス設定]画面において、[コントローラー] - [ライブラリー]で表示される[シリアル番号出力]ボタンにより、接続中のコントローラーのシリアル番号を、csvファイルとして保存できます。

キーポイント

ライブラリーを有効化せずに使用した場合、プロジェクトのビルド時、もしくは、ライブラリーツール起動時にエラーになります。

1. [セットアップ] - [ライセンス設定]をクリックします。
2. [ライセンス設定]画面のツリービューにおいて、[PC] - [ライブラリー]、もしくは、[コントローラー] - [ライブラリー]を選択します。

   ### キーポイント

   [コントローラー] - [ライブラリー]ノードは、実コントローラーに接続しているときのみ表示されます。

3. 目的のライブラリーが表示されていることを確認し、[有効]ボタンをクリックします。

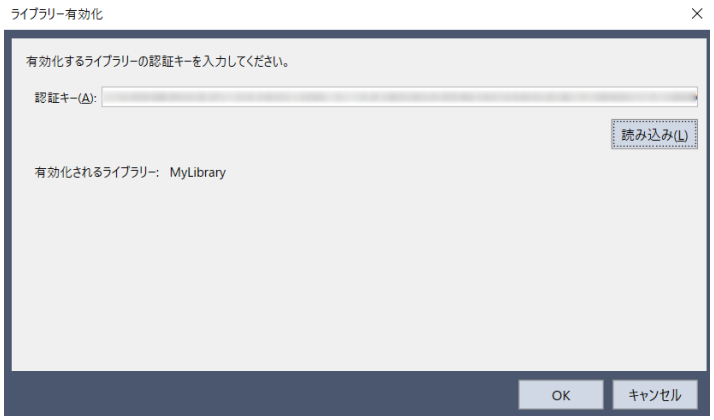
- PCライブラリーライセンスの場合

- コントローラーライブラリーライセンスの場合

キーポイント

`C:\EpsonRC80\Libraries` フォルダ下のライブラリーがリスト表示されます。

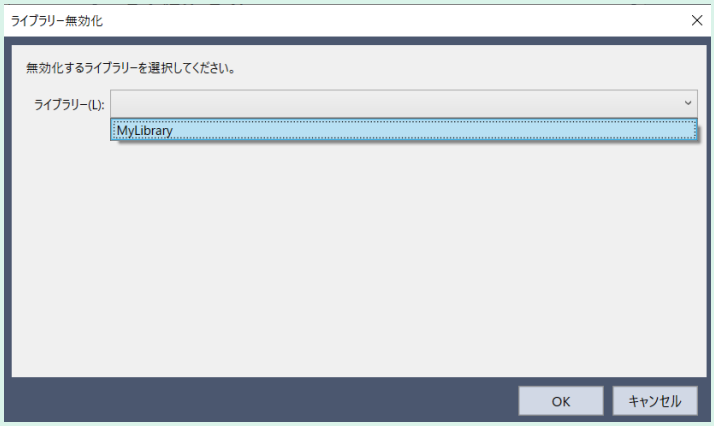
4. 表示された[ライブラリー有効化]ダイアログにおいて、認証キーを入力、もしくは、[読み込み]ボタンからcsvファイルを読み込みます。認証キーが正しい場合、認証キーにより有効化されるライブラリーが表示されます。



5. [OK]ボタンをクリックし、目的のライブラリーが有効化されていることを確認します。

キーポイント

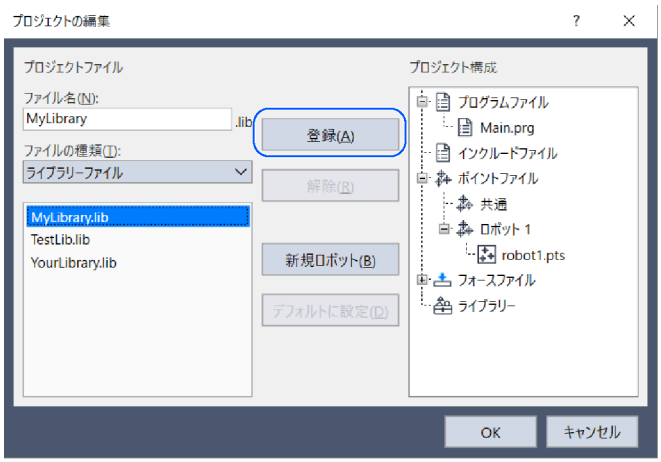
無効化する場合は、[無効]ボタンをクリックし、[ライブラリー無効化]ダイアログのドロップダウンリストから無効化したいライブラリーを選択して、[OK]ボタンをクリックします。



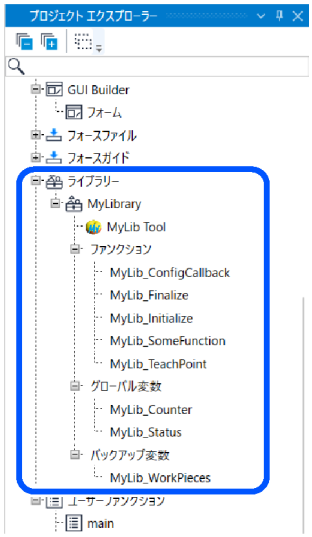
5.3 ライブラリーのプロジェクトへの登録

ライブラリーをプロジェクトに登録する方法を説明します。

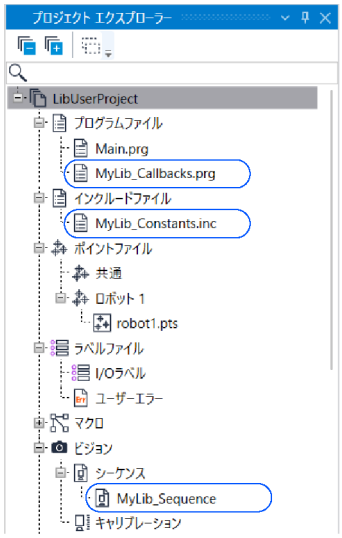
1. ライブラリーを使用するプロジェクトを作成します。
2. [プロジェクト]-[プロジェクトの編集]、もしくは、プロジェクトエクスプローラーのライブラリーを右クリックしてライブラリーの追加を選び、プロジェクトの編集ダイアログを開きます。



3. ファイルの種類において、ライブラリーファイルを選択します。
4. `C:\EpsonRC80\Libraries` フォルダー内のすべてのライブラリーのリストが表示されます。
5. ライブラリーを選択して[登録]ボタンをクリックします。
6. [OK]ボタンをクリックします。
7. プロジェクトエクスプローラーに利用可能な公開関数や公開変数が表示されます。



8. プロジェクトにライブラリー、およびライブラリーの公開ファイルが追加されます。公開ファイルの名称はライブラリーが定義したプレフィックスから始まります。公開ファイルの有無はライブラリーによって異なります。



9. ライブラリーを複数追加したい場合はこれを繰り返します。
   1つのプロジェクトに対して、ライブラリーは最大5つまで追加できます。

10. 実行したいライブラリー関数をプログラムに記述します。

11. 必要に応じてライブラリーの公開ファイルを編集してください。

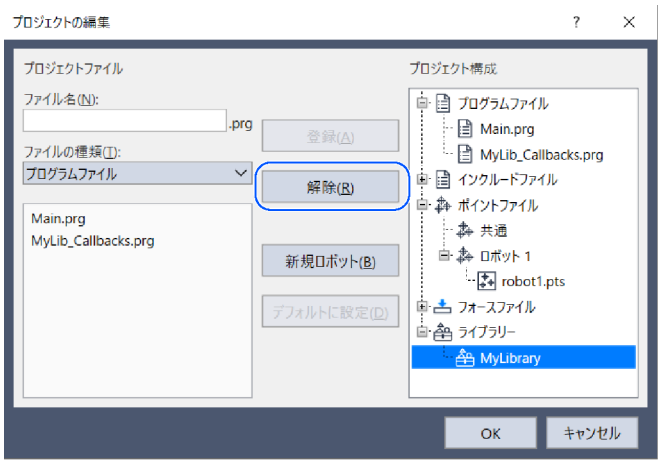
キーポイント

同じプレフィックスをもつ複数のライブラリーを登録しないでください。関数や変数が重複してエラーになる可能性があります。

5.4 ライブラリーのプロジェクトからの登録解除

登録したライブラリーをプロジェクトから解除する方法を説明します。

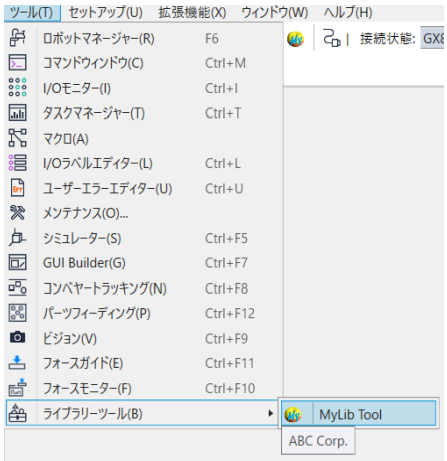
1. プロジェクトエクスプローラーの解除したいライブラリーを右クリックして [プロジェクトから除外]を選択すると、ライブラリーが解除されます。
   もしくは、[プロジェクト]-[プロジェクトの編集]を選び、プロジェクトの編集ダイアログを開きます。
   i. プロジェクト構成ツリー内の解除したいライブラリーを選択し、[解除]ボタンをクリックします。
   ii. [OK]ボタンをクリックすると、プロジェクトの編集ダイアログが閉じ、ライブラリーが解除されます。



2. ライブラリーと合わせて追加されたライブラリーの公開ファイルも必要に応じて解除してください。ライブラリーと同時には解除されません。

5.5 ライブラリーツールの利用

ライブラリーがGUIによる設定や実行をサポートしている場合があります。このとき、[ツール]-[ライブラリーツール]以下にメニューが追加されますので、こちらから実行してください。ライブラリーツールの詳細は、各ライブラリーのマニュアルを参照、または、各ライブラリーの開発元にお問い合わ せください。



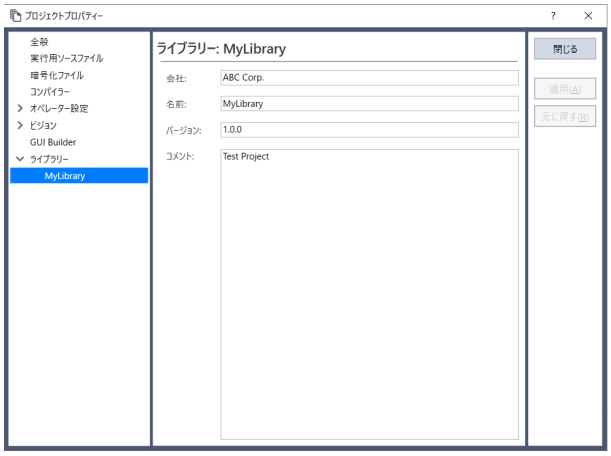
5.6 ライブラリーのマニュアル表示

ライブラリーにマニュアルが提供されている場合は、[ヘルプ]-[ライブラリーマニュアル]から参照できます。

5.7 ライブラリーのプロパティー設定

プロジェクトに登録されているライブラリーのプロパティーを確認できます。

1. [プロジェクト]-[プロパティー]からプロジェクトプロパティーダイアログを表示します。
2. ライブラリーが登録されている場合、ツリービューにライブラリーが追加されており、これを展開すると登録されたライブラリーが表示されます。
3. 各ライブラリー名を選択するとライブラリーの作成元、ライブラリー名称、バージョン、説明が表示されます。

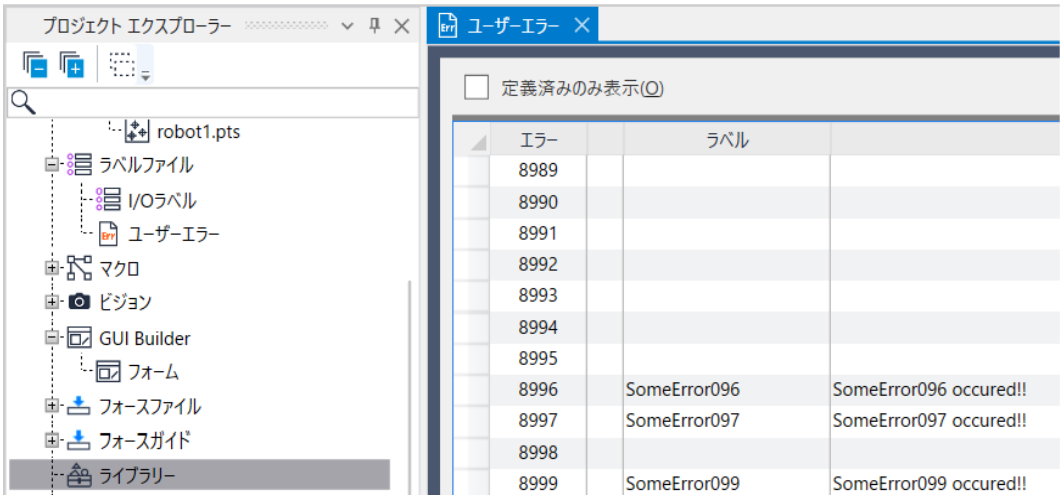


5.8 ライブラリーのユーザーエラー定義について

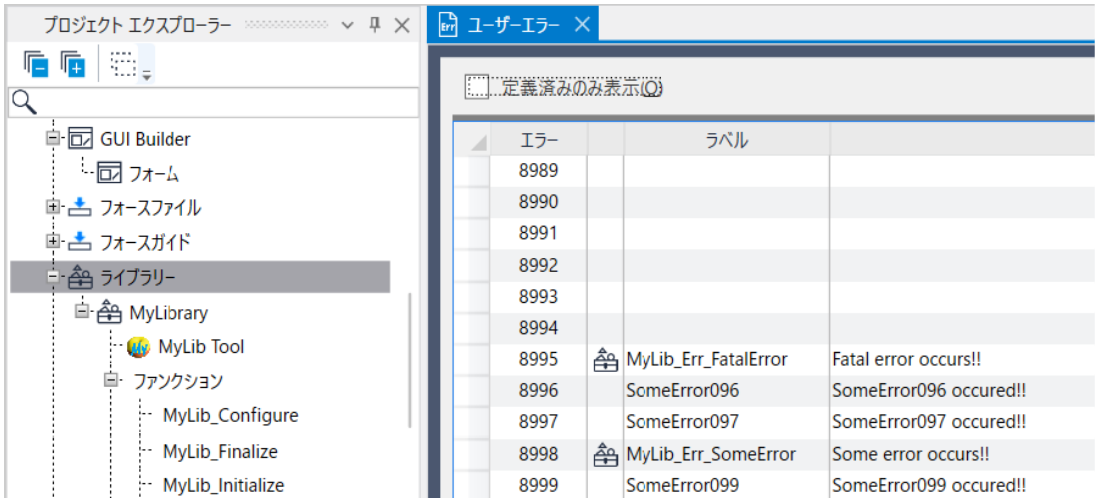
登録したライブラリーのユーザーエラーは、エラー番号の8999から降順に未定義の番号を使用します。ライブラリーのユーザーエラー定義は変更できませんので、ご注意ください。

ライブラリー追加前後の例：

追加前



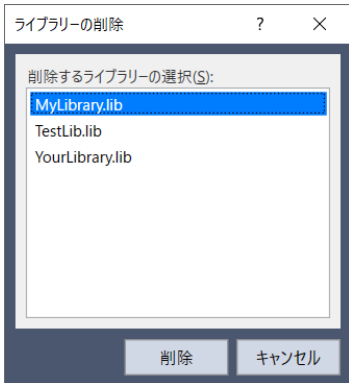
追加後



5.9 ライブラリーの削除

不要になったライブラリーを、Epson RC+から削除する方法を説明します。

1. [拡張]メニューから[ライブラリーの削除]を選択します。
2. 表示されたダイアログのリストから削除したいライブラリーを選択します。
   リストは現在Epson RC+が保持するライブラリーです。
3. [削除]ボタンをクリックします。`C:\EpsonRC80\Libraries`フォルダ下の該当ライブラリーのファイルが削除されます。


ライブラリーの削除
削除するライブラリーの選択(S):
MyLibrary.lib
TestLib.lib
YourLibrary.lib
削除
キャンセル

6. SPEL+コマンドリファレンス

6.1 SPEL+コマンド一覧

ライブラリー以外のSPEL+プロジェクト、公開プログラムファイルからは使用できません。
ライブラリーでのみ使用可能なSPEL+コマンドは、以下の通りです。

コマンド/関数	解説/用途
ArchReserve	ライブラリー内で排他的に利用可能なアーチ番号を動的に取得します。
ArmLib	動的に取得したアーム番号を選択します。
ArmReserve	ライブラリー内で排他的に利用可能なアーム番号を動的に取得します。
LibGetInfo	SPEL+プロジェクトに含まれるライブラリー情報を取得します。
LocalReserve	ライブラリー内で排他的に利用可能なローカル番号を動的に取得します。
PointReserve	ライブラリー内で排他的に利用可能なポイント番号を動的に取得します。
SignalReserve	ライブラリー内で排他的に利用可能なシグナル番号を動的に取得します。
SyncLockReserve	ライブラリー内で排他的に利用可能なSyncLock番号を動的に取得します。
TaskReserve	ライブラリー内で排他的に利用可能なタスク番号を動的に取得します。
TimerReserve	ライブラリー内で排他的に利用可能なタイマー番号を動的に取得します。
TLReserve	ライブラリー内で排他的に利用可能なツール番号を動的に取得します。
ToolLib	動的に取得したツール番号を選択します。
UploadFileAfterStop	全タスク停止後、指定されたファイルをコントローラーからPCのプロジェクトフォルダーにコピーします。

6.2 ArchReserve関数

現在のライブラリー内で利用可能なアーチ番号を動的に取得します。

書式
ArchReserve

パラメーター
なし

戻り値
予約ができたArch番号

解説
予約をすることで、他のライブラリーに使用されずにアーチ番号を使用できます。
7-13のうち、未予約の番号を大きい順に予約します。
予約したアーチ番号に定義された情報は一時的な値とし、全タスク終了時に予約が解除され、値も消えます。
ArchClrにアーチ番号を指定することでも予約を解除できます。

参照
Arch, ArchClr

ArchReserve関数使用例

```
Integer i

i = ArchReserve

Arch i,20,20
Jump3 P2, P3-TLZ(100), P3 C(i) '予約したArch番号を使用して動作する。

ArchClr i
```

6.3 ArmLib関数

動的に取得したアーム番号を選択します。

書式
ArmLib アーム番号

パラメーター
アーム番号 整数値、または式で指定します。有効値範囲は、16~31 です。このライブラリーが予約したアームを選択できます。

解説
ロボット命令を実行するアームに予約済みアームを指定します。
このコマンドを呼び出したライブラリーが予約したアーム番号を選択できます。
選択中のアーム番号を表示するためにはArmを実行してください。
選択したアーム番号による動作もArmと同様です。
このコマンドを呼び出したユーザー関数が終了すると関数実行前のアーム番号に戻ります。

参照
Arm, ArmClr, ArmDef, ArmSet, ArmReserve関数

ArmLib関数使用例

```
Integer i

i = ArmReserve

ArmSet i, X, Y, Z, U
ArmLib i
Move P1     '現在選択中の番号ではなく予約したアーム番号を使用して動作する。
```

6.4 ArmReserve関数

現在のライブラリー内で利用可能なアーム番号を動的に取得します。

書式
ArmReserve

パラメーター
なし

戻り値
予約ができたアーム番号

解説
予約を行うことで、他のライブラリーに使用されずに使用できます。16-31のうち、予約されていないアーム番号を、大きい順に予約していきます。
予約したアーム番号に定義された情報は一時的な値とし、全タスク終了時に予約とともに値も消えます。もしくはArmClrにアーム番号を指定することで予約を解除できます。

参照
Arm, ArmLib関数, ArmClr, ArmDef, ArmSet

ArmReserve関数使用例

```
Integer i

i = ArmReserve

ArmSet i, X, Y, Z, U
ArmLib i
Move P1 '予約したアーム番号を使用して動作する。
ArmClr i
```

6.5 LibGetInfo

SPEL+ プロジェクトに含まれるライブラリー情報を取得します。

書式

LibGetInfo ライブラリー名, ByRef ライブラリーID, ByRef バージョン, ByRef コメント, ByRef ライブラリーパス

パラメーター

- ライブラリー名: ライブラリー名称を文字列で指定します (最大32文字)。
- ライブラリーID: ライブラリーIDを取得する文字列変数を指定します。ライブラリーIDは、ライブラリーを作成する度に変更される値です。
- バージョン: 該当するライブラリーのバージョンを取得する文字列変数を指定します。
- コメント: 該当するライブラリーのコメントを取得する文字列変数を指定します。
- ライブラリーパス： ライブラリーの保存されているパスを取得する文字列変数を指定します。"[データフォルダ]¥Libraries[ライブラリー名]"で表します。

参照

GetProjectInfo

LibGetInfo使用例

```
String strID$
String strVer$
String strDescript$
String strPath$
LibGetInfo "test", ByRef strID$, ByRef strVer$, ByRef strDescript$, ByRef strPath$
Print "ID = " + strID$
Print "Version = " + strVer$
Print "Description = " + strDescript$
Print "Path = " + strPath$
```

6.6 LocalReserve関数

現在のライブラリー内で利用可能なローカル番号を動的に取得します。

書式
LocalReserve

パラメーター
なし

戻り値
予約ができたローカル番号

解説
予約を行うことで、他のライブラリーに使用されずに使用できます。16-31のうち、予約されていないローカル番号を、大きい順に予約していきます。
予約したローカル番号に定義された情報は一時的な値とし、全タスク終了時に予約とともに値も消えます。もしくはLocalClrにローカル番号を指定することで予約を解除できます。
予約したローカル番号で定義されたポイントはローカル番号0として保存されます。

参照
Local, LocalClr, LocalDef, P#, Arc, Go, Move, Jump

LocalRerserve関数使用例

```
Integer i

i = LocalReserve

Local i, X, Y, Z, U
P1 = Here /(i)
Move P1 '指定したローカル番号を使用して動作する。
```

6.7 PointReserve関数

現在のライブラリー内で利用可能なポイント番号を動的に取得します。

書式
PointReserve

パラメーター
なし

戻り値
予約ができたポイント番号

解説
予約をすることで、他のライブラリーに使用されずに使用できます。
予約したポイント番号に定義された情報は一時的な値とし、全タスク終了時に予約とともに値も消えます。PDelにポイント番号を指定することでも予約を解除できます。

参照
P#, PDel

PointReserve関数使用例

```
Integer i

i = PointReserve

P(i) = Here
Move P(i) +Z50 '指定したポイント番号を使用して動作する。
PDel 1
```

6.8 SignalReserve関数

SigWaitに指定可能なシグナル番号を予約します。

書式
SignalReserve

パラメーター
なし

戻り値
予約ができたシグナル番号

解説
予約を行うことで、他のライブラリーに使用されずに使用できます。一時的な値とし、全タスク終了時に予約とともに値も消えます。予約専用のシグナル番号が確保されますので、ライブラリーを利用するユーザーが利用しているシグナル番号の書き換えが発生するリスクを低減できます。

参照
WaitSig

SignalReserve関数使用例

```
Function N4_1_Call_Hidden
  Integer i1, i2, n1, n2, L1, t1

  i1 = TaskReserve
  i2 = TaskReserve

  n1 = SignalReserve
  n2 = SignalReserve

  Xqt i1, Hidden_N4_1(n1)
  Xqt i2, Hidden_N4_1(n2)

  Signal n1
  Signal n2
Fend

Function Hidden_N4_1(j As Integer)

  WaitSig j

  Print "Start Signal!"

Fend
```

6.9 SyncLockReserve関数

SyncLockに指定するシグナル番号を予約します。

書式
SyncLockReserve

パラメーター
なし

戻り値
予約ができたシグナル番号

解説
SyncLock および SyncUnLockに指定することでファイルアクセスなどを排他的に行うことができます。
シグナル番号の予約を行うことで、他のライブラリーに使用されずに使用できます。一時的な値とし、全タスク終了時に予約とともに値も消えます。予約専用のシグナル番号が確保されますので、ライブラリーを利用するユーザーが利用しているシグナル番号の書き換えが発生するリスクを低減できます。

参照
SyncLock

SyncLockReserve関数使用例

```
Function N4_1_Call_Hidden
  Integer i1, i2, n1, n2, L1, t1

  i1 = TaskReserve
  L1 = SyncLockReserve

  Xqt i1, Hidden_N4_1(L1)

Fend

Function Hidden_N4_1(j As Integer, i As Integer)
  SyncLock i

  Print "4-1 Layer Lib!"

  SyncUnlock i
Fend
```

6.10 TaskReserve関数

タスク実行時に指定可能なタスク番号を予約します。

書式
TaskReserve

パラメーター
なし

戻り値
予約ができたタスク番号

解説
予約を行うことで、他のライブラリーに使用されずに使用できます。一時的な値とし、全タスク終了時に予約とともに値も消えます。
予約されていないタスク番号を、大きい順に予約していきます。

参照
WaitSig

TaskReserve関数使用例

```
Function N4_1_Call_Hidden
  Integer i1, i2, n1, n2, L1, t1

  i1 = TaskReserve
  i2 = TaskReserve

  n1 = SignalReserve
  n2 = SignalReserve

  Xqt i1, Hidden_N4_1(n1)
  Xqt i2, Hidden_N4_1(n2)

  Signal n1
  Signal n2
Fend

Function Hidden_N4_1(j As Integer)

  WaitSig j

  Print "Start Signal!"

Fend
```

6.11 TimerReserve関数

Timerに指定するタイマー番号を予約します。

書式
TimerReserve

パラメーター
なし

戻り値
予約ができたタイマー番号

解説
Tmr関数に指定することでサイクルタイムの計測を排他的に行うことができます。
タイマー番号の予約を行うことで、他のライブラリーに使用されずに使用できます。一時的な値とし、全タスク終了時に予約とともに値も消えます。予約専用のタイマー番号が確保されますので、ライブラリーを利用するユーザーが利用しているタイマー番号の書き換えが発生するリスクを低減できます。

参照
Tmr, TmReset

TimerReserve関数使用例

```
Function N4_1_Call_Hidden

  t1 = TimerReserve

  TmReset t1              'タイマー0をリセット

  Xqt ...

  Print Tmr(t1) / 10      'サイクルタイムを計算して表示
Fend
```

6.12 TLReserve関数

現在のライブラリー内で利用可能なツール番号を動的に取得します。

書式
TLReserve

パラメーター
なし

戻り値
予約できたツール番号

解説
予約を行うことで、他のライブラリーに使用されずに使用できます。一時的な値とし、全タスク終了時に予約とともに値も消えます。もしくは、TLClrにツール番号を指定することで予約を解除できます。
16-31のうち、予約されていないツール番号を、大きい順に予約していきます。

参照
TLClr, TLDef, TLSet, Tool, ToolLib関数

TLRerserve関数使用例

```
Integer i

i = TLReserve

TLSet i, X, Y, Z, U
ToolLib i
Move P1        '指定したツール番号を使用して動作する。
TLClr i
```

6.13 ToolLib関数

動的に取得したツール番号を選択します。

書式
ToolLib ツール番号

パラメーター
ツール番号
整数値、または式で指定します。有効値範囲は、16~31 です。このライブラリーが予約したツールを選択できます。

解説
ロボット命令を実行するツールに予約済みツールを指定します。
このコマンドを呼び出したライブラリーが予約したツール番号を選択できます。
選択中のツール番号を表示するためにはToolを実行してください。
選択したツール番号による動作もToolと同様です。
このコマンドを呼び出したユーザー関数が終了すると関数実行前のツール番号に戻ります。

参照
TLClr, TLDef, TLSet, TLReserve関数, Tool

ToolLib関数使用例

```
Integer i

i = TLReserve

TLSet i, X, Y, Z, U
ToolLib i
Move P1              '予約したツール番号を使用して動作する。
```

6.14 UploadFileAfterStop関数

全タスク停止後、指定されたファイルをコントローラーからPCのプロジェクトフォルダーにコピーします。

書式
UploadFileAfterStop ファイル名

パラメーター

- ファイル名: プロジェクト内に含まれるファイル名を指定します。

参照
GetProjectInfo

UploadFileAfterStop使用例

```
UploadFileAfterStop("test.csv")
```