EPSON

Epson RC+ 8.0 Extensions
Library Builder 8.0

Original instructions

Rev.2
ENM256S7320F

Table of Contents

1. FOREWORD

1.1 FOREWORD

Thank you for purchasing our robot system.

This manual contains information necessary to correctly use Library Builder.

Before using the system, read this manual and related manuals to ensure correct use.

After reading the manual, keep it in a place where it can be easily accessed at any time, and reread it if you have any questions.

Epson conducts rigorous testing and inspections to ensure that the performance of our robot systems meets our company standards. Note that the basic performance of the product will not be achieved if the robot system is used outside the conditions of use described in the manual.

This manual describes dangers and problems foreseen by Epson. To use our robot system safely and correctly, be sure to observe the safety precautions described in this manual.

1.2 Trademarks

Microsoft, Windows, the Windows logo, Visual Basic, and Visual C++ are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other company names, brand names, and product names are either registered trademarks or trademarks of their respective companies.

1.3 Notation

Microsoft® Windows® 10 operating system
Microsoft® Windows® 11 operating system
In this manual, the above operating systems are referred to as Windows 10 and Windows 11, respectively. Windows 10 and Windows 11 are sometimes collectively referred to as Windows.

1.4 Terms of Use

No part of this instruction manual may be reproduced or reprinted in any form without express written permission.
The information in this document is subject to change without notice.
Please contact us if you find any errors in this document or if you have any questions about the information in this document.

1.5 Manufacturer

SEIKO EPSON CORPORATION

1.6 Contact Information

Contact information details are listed in the "Supplier" section in the following manual.
Note that the contact information may vary depending on your region.
"Safety Manual - Contact Information"

The Safety Manual is also available at the following site.
URL: https://download.epson.biz/robots/

1.7 Getting Started

Before using this manual, be sure that you understand the following information.

The installation folder for Epson RC+ 8.0

You can change the path for the installation folder for Epson RC+ 8.0 anywhere. This manual assumes that Epson RC+ 8.0 is installed in `C:\EpsonRC80`.

Meaning of symbols

The following symbols are used in this manual to indicate important safety information. Be sure to read the descriptions shown with each symbol.

⚠ WARNING

This symbol indicates an imminently hazardous situation which, if operation is not performed properly, will result in death or serious injury.

⚠ WARNING

This symbol indicates a potentially hazardous situation which, if operation is not performed properly, could result in an injury due to electric shock.

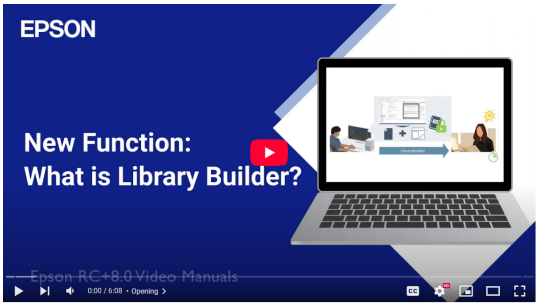
⚠ CAUTION

This symbol indicates a potentially hazardous situation which, if operation is not performed properly, may result in a minor or moderate injury or in property damage only.

2. Overview

2.1 Overview

Library Builder is also explained in a video.

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	

KEY POINTS

- An internet connection is required to view the videos.
- The video include an audio track.
- To view in your native language, use YouTube's auto-translated subtitles feature.

Library Builder is the generic name for the following functions.

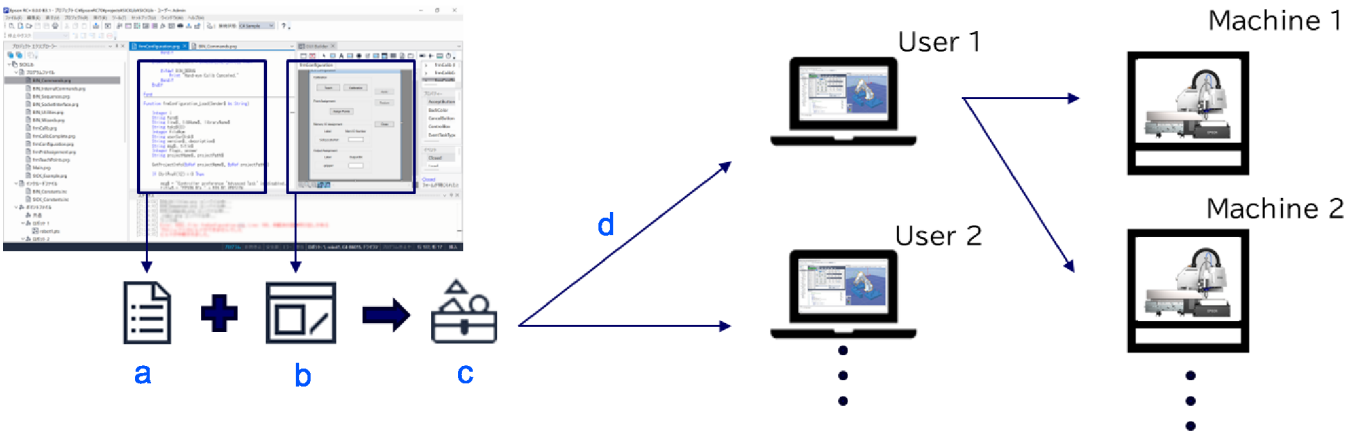
- Function to package configuration data and files from a SPEL+ project as libraries

- Function to add and use created libraries in other SPEL+ projects

Of the SPEL+ project components that are organized into libraries, the following components can be used from the SPEL+ project with library added.

- Global variables
- Backup variables (Global Preserve)
- SPEL+ function
- Form created with GUI Builder
- User error
- Program files
- Include files
- Vision sequence

Library Builder can be used to create your own libraries to assist in the configuration of your production equipment or third-party devices and provide them to your users.



Symbol	Description
a	• Program • Function • Variable
b	• Setting screen • Display screen
c	Library
d	Supply

KEY POINTS

RC+ 8.0 Ver. 8.1.0.0 or later is required to create and use libraries. Additionally, an RC+ Premium Edition license is required to create a library.
The SPEL+ commands, which are useful for creating libraries, are available in the following controller firmware versions and later.

- RC800 Series: 8.1.0.0
- RC90, RC700 Series: 7.5.5.0
- T, VT Series: 7.5.55.0

For details about the SPEL+ commands, refer to the following.

- **SPEL+ Command Reference**
- "SPEL+ Language Reference - Appendix C: Epson RC+ 8.0 Commands - C-2: List of commands added by each version of Epson RC+ 8.0"

3. Developing a Library

3.1 Library Components

- SPEL+ project components that are organized into libraries are divided into public components and private components.
- Public components are directly available to any SPEL+ project with library added.
- Private components are not directly available to a SPEL+ project with library added.
- Public and private components are distinguished by the presence or absence of a prefix.
- Public program files, include files, and vision sequences are called public files and can be edited in the SPEL+ project with library added.

Functions	Public Components	Private Components
Using from a SPEL+ project that uses a library	available	unavailable
Editing by users using a library	available	unavailable

- Public components can be used as described below.

Components	Usage when specified as public
Functions	Used to provide functions as functions to library users.
Global variables	Used to indicate the library's operation as variables, or specify the values necessary for the library to operate.
Backup variables	Used in the same way as the global variable. Convert into backup variables when you want to continue to use the same value in the next operation by storing the value inside the Controller.
Program file	・Used when displaying the usage example of the library's public function. ・It is also used when you want to have the library user to write the processing of the public function called from the library.
Include file	Used in a way that allows the library user to use the argument and return value of a public function as constants. By using them as constants, it will make it easier for the library user to understand the meaning of the value.
Vision sequence	Used to provide vision sequence to the library user.

3.2 Creating a Project for the Library

1. An RC+ Premium Edition license is required to create a library. Purchase a license key from the supplier and authenticate it.
   For details about authenticating the license, refer to the following manual.

   "Epson RC+ 8.0 User's Guide - System Operations - Starting Epson RC+ 8.0 - License Authentication"

2. Create a new project for the library.

3. Add a prefix to any information you want to publish to users of your library. The prefix must be a maximum of 10 characters, including the trailing underscore.
   The following information can be published:

Information	Public conditions
Functions	The function name starts with a prefix.

Information	Public conditions
Global variables	The variable name starts with a prefix.
Backup variables	The variable name starts with a prefix.
Program file	Meets all of the following conditions: ・The file name starts with a prefix. ・All functions, constants, global variables and backup variables within the file starts with a prefix. ・Functions, global variables, backup variables, and comments are the only ones written in the file. ・An include file that does not meet the public condition is not included.
Include file	Meets all of the following conditions: ・The file name starts with a prefix. ・All constants written in the file starts with a prefix. ・Constants and comments are the only ones written in the file.・An include file that does not meet the public condition is not included.
Vision sequence	The sequence name starts with a prefix.

Example) Using the "MyLib_" prefix

```
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
```

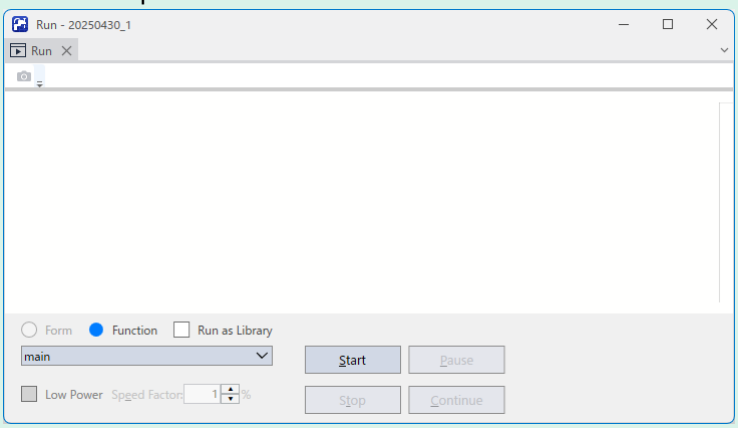
MyLib_Callbacks.prg: Public program files

> ⚠ NOTE
>
> When using multiple libraries, add a unique prefix to each library to avoid duplication of function names or variable names, etc.

1. Include the above public information and develop your project.

KEY POINTS

In Epson RC+ Premium Edition, the [Run as library] checkbox will be displayed in the command window. When enabled, commands that can only be used in the library can be executed from a project. Use this to check the operation.



2. To provide a library tool, use [Tools] - [GUI Builder] to create a form to be displayed as the library tool and set it as the startup form.
   For details about the library tools, refer to the following.

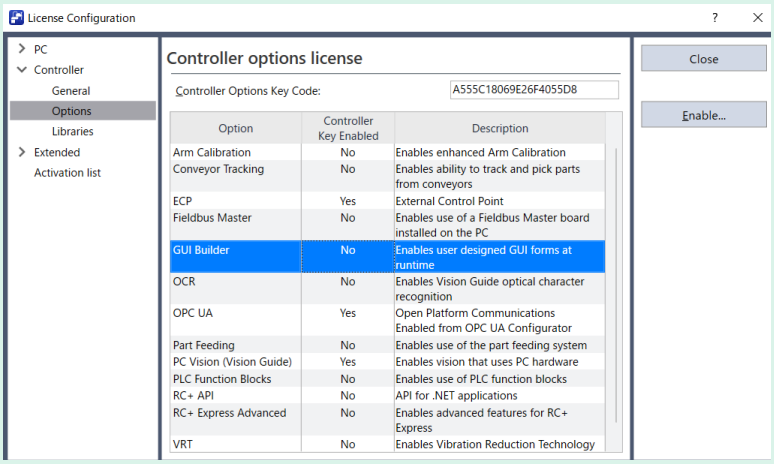
   Using Library Tools

   For more information on how to use GUI Builder, refer to the following manual.

   "Epson RC+ 8.0 Option, GUI Builder 8.0"

KEY POINTS

In RC+ Premium Edition, you can run forms and execute GUI Builder-related commands even if the GUI Builder option is disabled.



3. If necessary, create additional files such as PDF files to display as library manuals, image files to display as library tool icons, device settings and manuals, etc.

KEY POINTS

- In RC+ 8.0 Ver. 8.1.0.0, do not register other libraries in the library project. You cannot create library A that uses another library B inside it (library nesting is not allowed). This will cause an error when starting Library Builder.
- The settings used in the library can be saved as project data [arbitrary name].libcfg for the users who use the library.
  For example, when providing a library tool, you can implement it so that the values set by the user in the GUI are saved in [arbitrary name].libcfg, and the previous setting values are then read from this file the next time the tool is started to restore the previous setting state.
  Specify a name in [arbitrary name] that is not duplicated in other libraries.

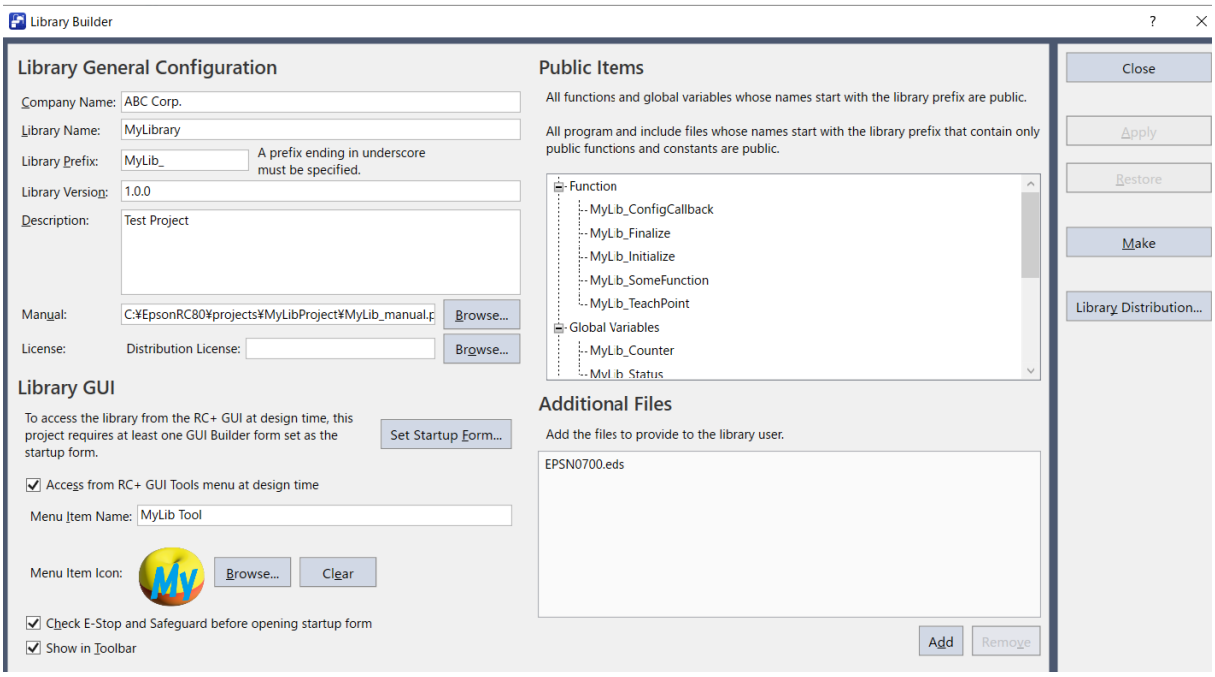
3.3 Creating a Library

1. Open the library project.
2. Select Extensions | Library Builder.
3. The Library Builder dialog box appears if the project build is successful.

KEY POINTS

The following operations are performed according to the TP4 operation mode:
・In AUTO mode, you can use the Epson RC+. However, you cannot create a library in Epson RC+ 8.0 Ver8.1.0.0.
(Library Builder will not start)
・In TEACH and TEST mode, you cannot create a library because Epson RC+ cannot be used.
・Regardless of the TP4 operation mode, existing libraries can be used in projects.
In RC+ 8.0 Ver. 8.1.0.0, Library Builder cannot be used with TP4 (Operation Mode: AUTO).
For details about TP4 Operation Mode, refer to the following manual.

"Robot Controller Option, Teaching Pendant TP4 Manual - Operation Mode (TEACH, AUTO, TEST)"



Item	Description
Company Name	Enter the name of the company. Up to 32 single-byte characters.
Library Name	Enter the name of the library. Up to 32 single-byte characters.
Library Prefix	Enter the prefix. Up to 10 single-byte characters but must end in an underscore.
Library Version	Enter the library version. Up to 32 single-byte characters.
Description	Enter a description of the library. Up to 255 characters.
Manual	Set the manual (PDF file) for the library you are creating.
Distribution License	Enter the Distribution License sent by Epson. You can also use the Browse button to load a PDF file providing the license.
Set Startup Form	Displays the startup form setup screen. Set the form that will be used as the initial display screen of the library GUI (library tool) in the startup form.
Access from RC+ GUI Tools menu at design time	Forms created with GUI Builder can be incorporated into a library as library tools that can be launched from the Tools menu. This can be selected if the project contains a form set as the startup form. Deselected by default. For details about the library tools, refer to the following. **Using Library Tools**
Menu Item Name	Enter the name that will be displayed for the library tool in the Tools menu. Up to 32 single-byte characters.
Menu Item Icon	Sets the icon that will be displayed for the library tool in the Tools menu.

Item	Description
Check E-Stop and Safeguard before opening startup form	Checks the status of emergency stops and safeguards when the startup form (library tool) is launched. Selected by default.
Show in Toolbar	Displays an icon on the toolbar to launch the library tool.
Public Items	Displays functions, global variables, backup variables, program files, include files, and vision sequences with prefixes applied.
Additional Files	For devices used in the library, device setting files (such as fieldbus setting files (eds)) and manuals are added so that users can easily make settings and refer to the manuals. They are copied to the AdditionalFiles folder in the corresponding library folder under the `C:\EpsonRC80\Libraries` folder.
Close	Close the designer.
Apply	Saves the changes.
Restore	Reverts to the previous settings.
Make	Creates the library.
Library Distribution	Opens the library distribution wizard. Outputs the library and authentication key to distribute to users. See the following chapter for details. **Creating a Library for Distribution**

4. Specify the prefix. Functions, global variables, backup variables, program files, include files, and vision sequences that contain the specified prefix are automatically added to the public information area.

> **KEY POINTS**
>
> Even if a prefix is added to the file name of a program file or include file, the file will not be displayed as public items if the file contains functions or constants without a prefix.

5. Specify the company name, library name, version, comments, manual files, and additional files.

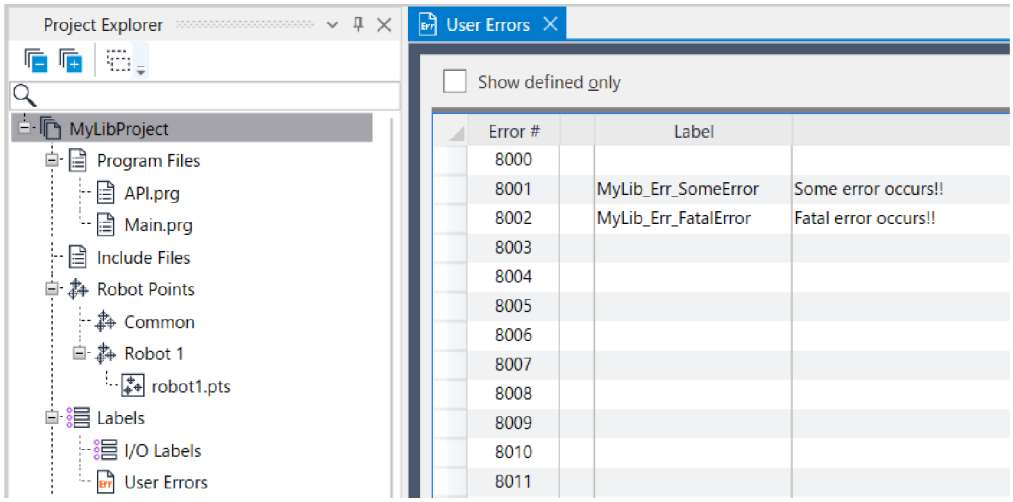
6. If you provide a library tool, select the [Access from RC+ GUI Tools menu at design time] check box and specify the menu item name and menu item icon to be displayed in the RC+ tools menu.
   If the [Check E-Stop and Safeguard before opening startup form] check box is selected, a warning message is displayed if the controller is in the emergency stop state or the safeguard is open, and the library tool will not start until these conditions are resolved.

7. Click the Apply button to save the changed settings.

8. Click the Make button to create the library.
   The library file and zip file created are generated in the `C:\EpsonRC80\Libraries` folder.

3.4 How to Report User Errors Defined in the Library

1. Define user errors.
   Be sure to define not only the message but also the label.



2. Create a function to report a user error.

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

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

   For more information on UserErrorNumber and Error, refer to the following manual.

   "SPEL+ Language Reference"

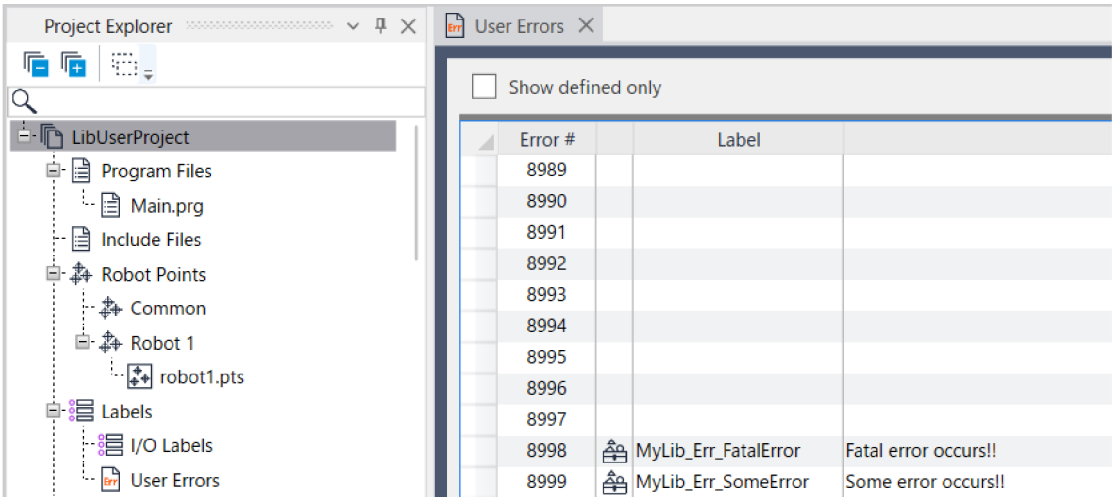
3. Implement error handling.
   The error handler in the library project's code calls the user error reporting function shown above.

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

4. Follow the procedure below to create the library.

   Creating a Library

5. When using a library, if the library created above is registered to a project, it is registered to the next available error number starting from the end of the user errors.



Although the error number differs from the error number at the time the library was created, errors are issued based on the user error label in the library, so that the correct message is displayed.

3.5 How to Use the Callback Function

The callback function is a function that allows you to include project processing during the processing of a library's public function.
The library developer writes a callback function template (skeleton code) and description in the public program file, using the function name, arguments, and return value specified by the library.
The library user can implement necessary processing by looking at the description within the file because the public program file will be added when a library is added to a project.

Example: Example of a callback function description

```
Function Lib_Callback(num As Integer) As Boolean
   'Called from the library after XXX is processed.
   'Parameter: Checks whether the num values are appropriate. True is returned when
appropriate and False when it is not appropriate.
Fend
```

It can be used in the following way.

・A user program can change the subsequent processing during library processing.
【Example】

- The library monitors the value of the input bit and calls for the callback function with the Boolean type return value if it has been OFF for a certain period of time. If the return value is False, the processing continues and ends when it is returned as True.
- The library user can implement the callback function in the following ways and choose to continue/end the subsequent processing.
  - When a different input bit is On: With the MsgBox command, have the operator select continue/end and use the result as a return value of the call back function.
    -When a different input bit is Off: Set the return value to True and terminate the process without having the operator select it.

・Progress indicator and termination when it takes a long time to process the public function.
【Example】

- The library periodically calls a callback function with a Boolean return value, with the progress rate as an argument. When the return value is True, the process will be terminated.

- The library user displays the progress rate obtained by the callback function as a GSet in the GUI builder form.When the terminate button on the form is pressed, the return value of the callback function will be set to True the moment the callback function is called.

3.6 Synchronize using libcfg file

In SPEL project, there is a function where you can store settings that are necessary for operating a program and calculated values in a file like a point data.
During operation, the data is stored in the Controller, but is also synchronized with the project on the PC running the RC+ at any time, allowing you to smoothly develop and maintain SPEL projects, and port them to other environments.

When using the library, the required information may vary depending on the device or function being operated. Therefore, we provide the same functionality for files available in any format.
Consider if you wish to retain the following information:

- Setting files that referenced by functions and devices.
- Files that stores and references the calculated results of functions and devices.
- Log files that stores and monitors the operation status of a function or device.

When operating a file with the extension "libcfg" in the Controller, it will be downloaded to the PC side after checking the synchronization when connecting to RC+. There is no limitation to the format.

Example: Example using the libcfg file: Exporting files

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

   iFileID = FreeFile
   ChDisk FLASH                     'Specify the project folder of the Controller
   WOpen "Lib1.libcfg" As #iFileID  'Open with the necessary mode

   Print #iFileID, "Name: ", name$  'Export the setting type and value

   'Repeat for each parameter.

   Print #iFileID, "ParamB: ", paramB  'Export the setting type and value
   Close #iFileID
Fend
```

Example：Example using the libcfg file: Reading files

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

      iFileID = FreeFile
      ChDisk FLASH                      'Specify the project folder of the
Controller
      ROpen "Lib1.libcfg" As #iFileID  'Open with the necessary mode

      Line Input #iFileID, Buf$
      iPos = InStr(Buf$, ": ")     'Reads the setting type and value separately.
      name$ = Mid$(Buf$, iPos + 2)

      'Read for each parameter.

      Line Input #iFileID, Buf$
      iPos = InStr(Buf$, ": ")     'Reads the setting type and value separately.
```

```
    paramB = Val(Mid$(Buf$, iPos + 2))

    Close #iFileID
Fend
```

Use the "UploadFileAfterStop" command when you want to use a specific extension due to the specification of the device you are using. By writing this in the library, the specified file will be read from the Controller after the task is completed. However, the RC+ must be connected for this operation.

4. Creating a Library for Distribution

4.1 Creating a Library for Distribution

This section explains how to create a library for distribution to users who wish to use the library.
When distributing libraries, you can identify which library user is available. Use this when setting prices for libraries according to the distribution format.

Signal	Available library user	Usage	Necessary information
a	Only users of specified Controllers	Permits the usage on a specific Controller. The library user must buy a library for each Controller they wish to operate and enable them. The library creator must obtain a serial number for each Controller and issue an authentication key.	Distribution License, Serial number of the library user's Controller (It can also be output from the library user's RC+. Obtain beforehand.)
b	Only users of the specified PC	Permits the usage on the RC+ virtual controller running on a specific PC. This is different from the actual Controller. The library user must buy a library for each PC they wish to operate and enable them. The library creator must obtain a PC hardware ID for each PC and issue an authentication key.	Distribution License, PC hardware ID of the library user (It can also be output from the library user's RC+. Obtain beforehand.)
c	All users who have received an authentication key	Library users can use Controllers and PC without limitations. Library users must enable the library.	Distribution License, A optional string defined by the creator of the library (Used only for settings. Not needed in enabling the library user.)
d	All users who have received a library	Library users can use Controllers and PC without limitations. Library users do not have to enable the library.	Distribution License

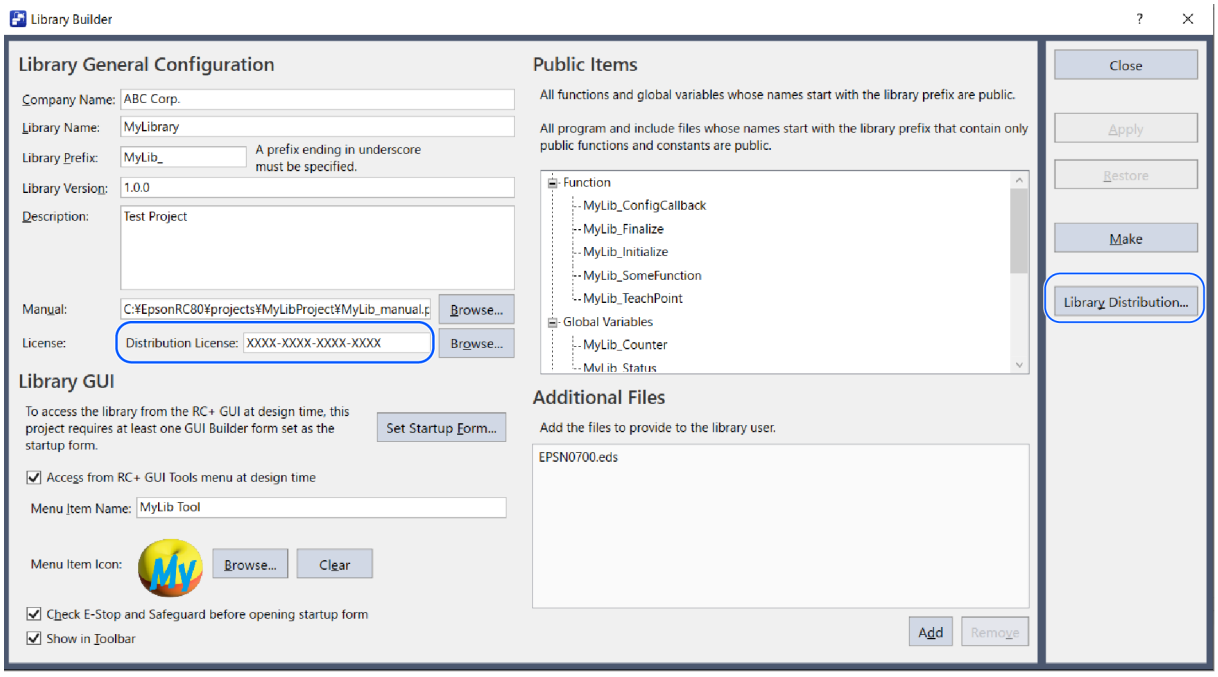
1. Open the project containing the library you want to distribute.

2. Select Extensions | Library Builder.

3. The Library Builder dialog box appears if the project build is successful.

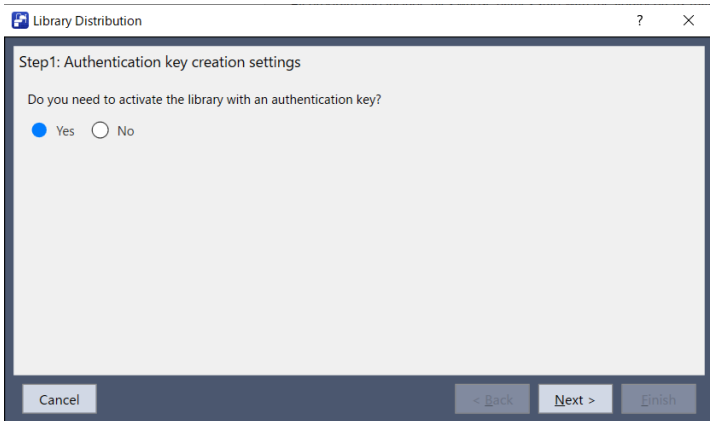
> **KEY POINTS**
>
> In RC+ 8.0 Ver. 8.1.0.0, Library Builder cannot be used with TP4 (Operation Mode: AUTO).
> For details about TP4 (Operation Mode: AUTO), refer to the following manual.
>
> "Robot Controller Option, Teaching Pendant TP4 Manual - Operation Mode (TEACH, AUTO, TEST)"

4. Enter the Distribution License sent by Epson.



5. Click the Library Distribution button.

6. When the library distribution wizard appears, select whether or not the library to be distributed requires activation with an authentication key.



- [Yes] : Select when a library user other than d is available. Move onto the next step.
- [No] : Select when d is the available library user. Move onto step 9.

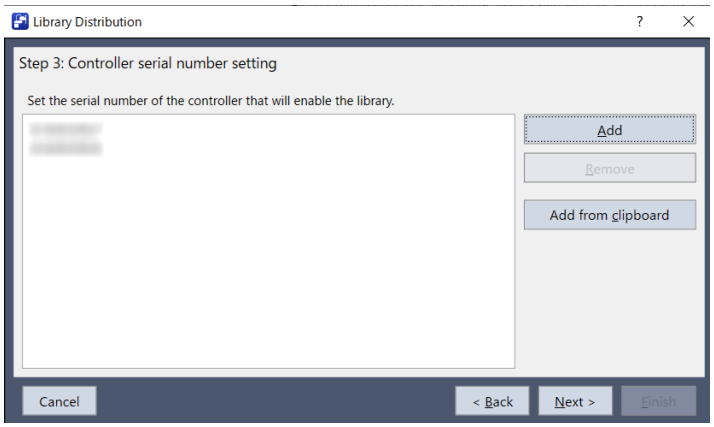
7. Depending on the library user you want to identify its usage for, select the device you want to enable the library on.

- [Controller] : Select when the available library user is a.
- [PC(Virtual Controller)] : Select when the available library user is b.

- [Do not specify] : Select when the available library user is c.

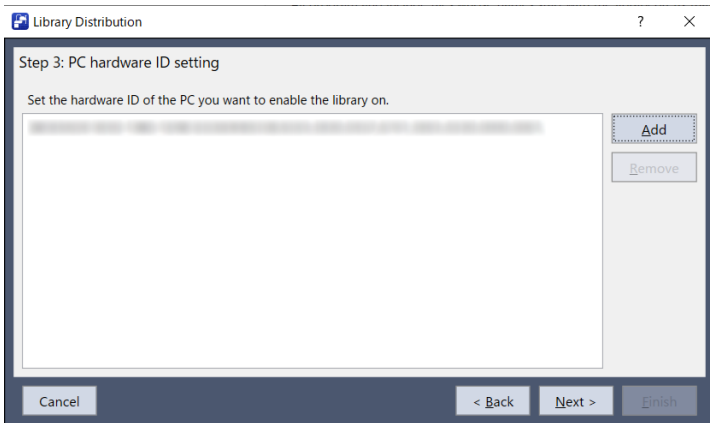
8. Enter the following information depending on the device that is selected.

- **If Controller is selected:**
  Click the Add button to load the CSV file containing the controller serial number obtained from the library user.
  Click the Add from clipboard button to add the character string from the clipboard. Use this to copy and paste the serial number.



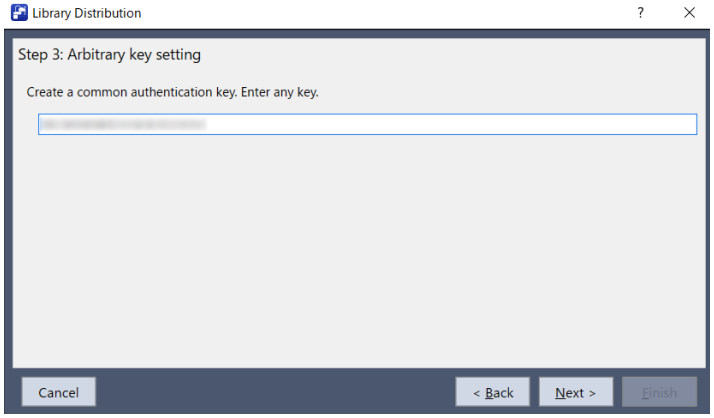
> **KEY POINTS**
>
> You can load multiple serial numbers and generate authentication keys for multiple controllers at once.

- **If PC (Virtual Controller) is selected:**
  Click the Add button to load the CSV file containing the PC hardware ID obtained from the library user.



> **KEY POINTS**
>
> You can load multiple PC hardware IDs and generate authentication keys for multiple PCs at once.

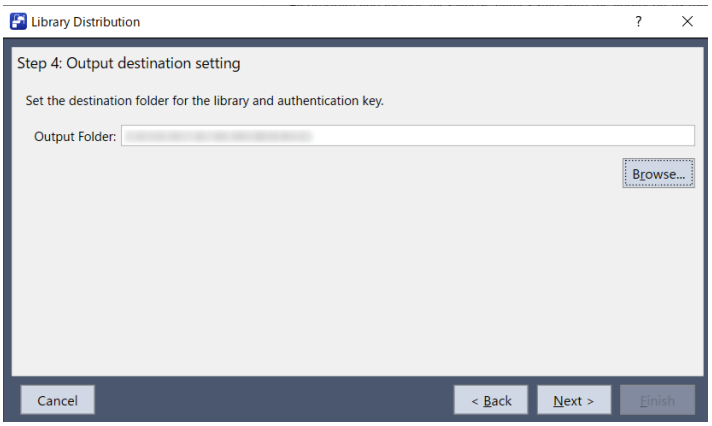
- **If Not Specified is selected:**
  Enter the random key. Up to 32 alphanumeric characters, underscores, and hyphens can be used.
  It is possible to create a distribution library that is "c. Available only to registered users" without entering a key.



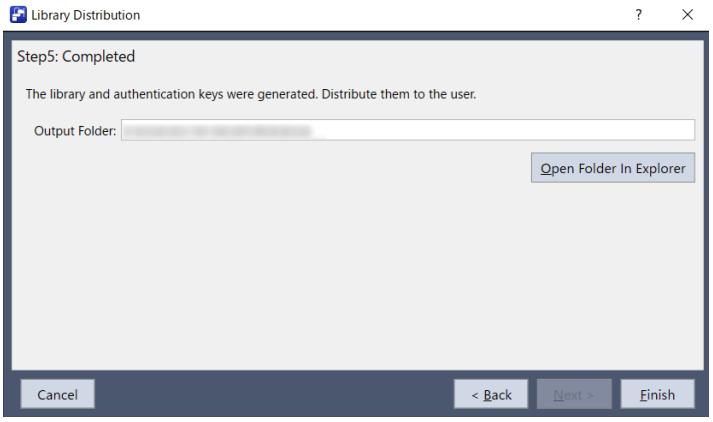
KEY POINTS

The generated authentication key allows you to activate the library regardless of the device.

9. Specify the destination to output the library and authentication key.



10. The library and authentication key are output. (If you selected No at step 6, the authentication key is not output.)
Click the Open Folder In Explorer button to display the output folder.



5. Using a Library

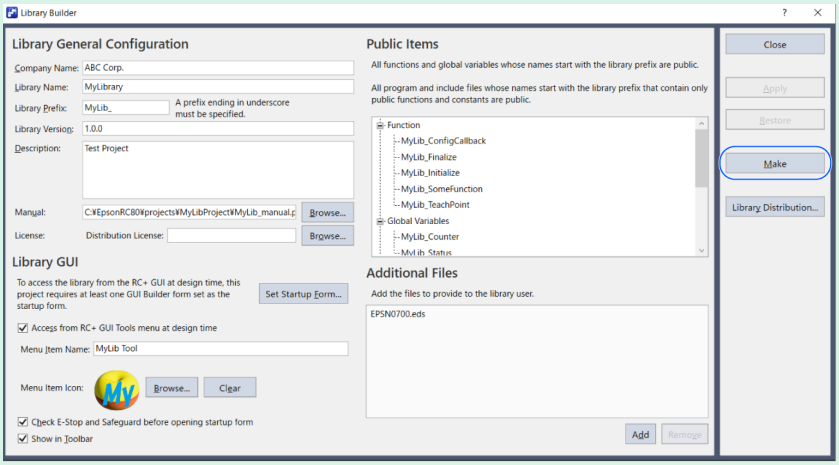
5.1 Importing a Library

This section explains how to import the acquired libraries into RC+.

1. The library is distributed as a zip file.
2. Select Extensions | Import Library.
3. Select the library you want to import from the file selection dialog box that appears.
4. Click Open. The imported library file and zip files are generated under the `C:\EpsonRC80\Libraries` folder. If a library with the same name already exists, a message prompts you to confirm whether you want to overwrite it.

KEY POINTS

Libraries created with the Make button in Library Builder in the same RC+ do not need to be imported.



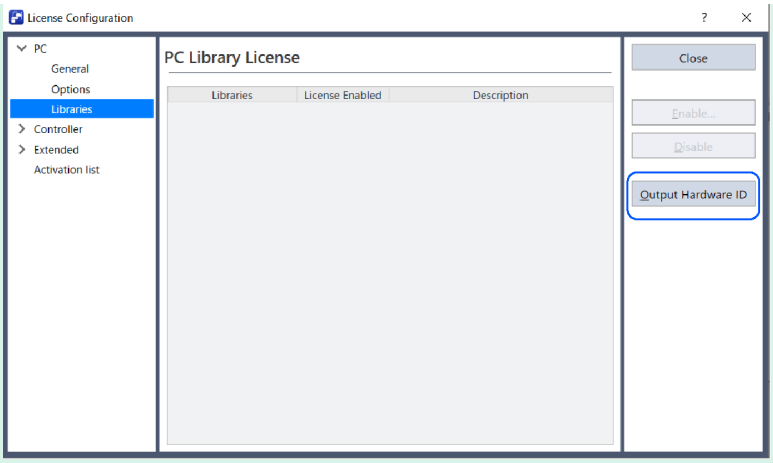
5.2 Activating a Library

If a library requires authentication, it must be activated. This section explains how to activate an imported library.
A library authentication key is required to activate a library.
A library cannot be used if it has not been activated. A library that does not require authentication can be used as is without activation.

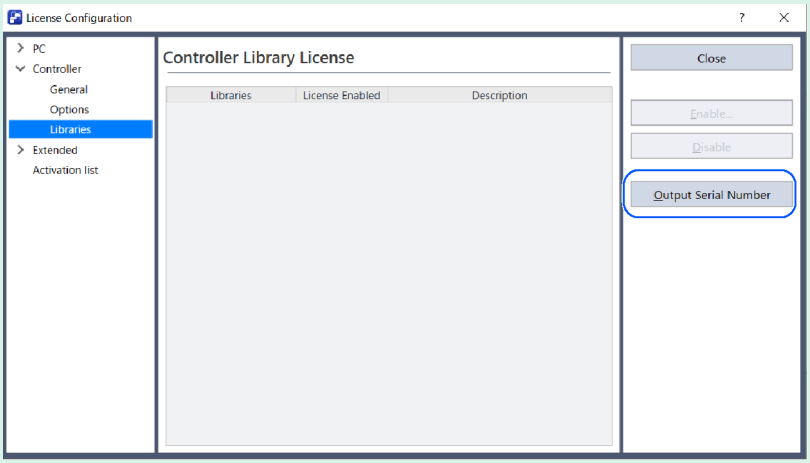
KEY POINTS

The authentication key is issued by the library creator. If necessary, send one or both of the following pieces of information to the library creator.

- **PC hardware ID**
  On the Setup | License Settings screen, select PC | Libraries and click the Output Hardware ID button to save the information as a CSV file.



- **Controller serial number**
  On the Setup | License Settings screen, select Controller | Libraries and click the Output Serial Number button to output the serial number of the connected controller as a CSV file.



KEY POINTS

If a library is used without activating it, an error occurs when you build the project or start the library tool.

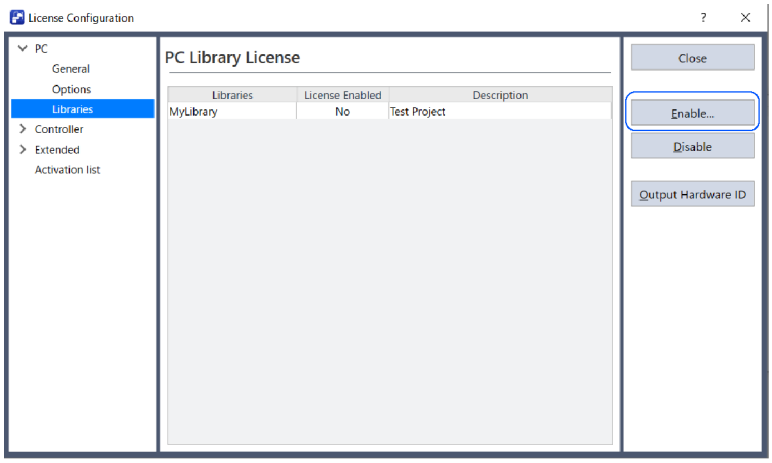
1. Click Setup | Licenses.
2. In the License Settings screen tree view, select PC | Libraries or Controller | Libraries.

KEY POINTS

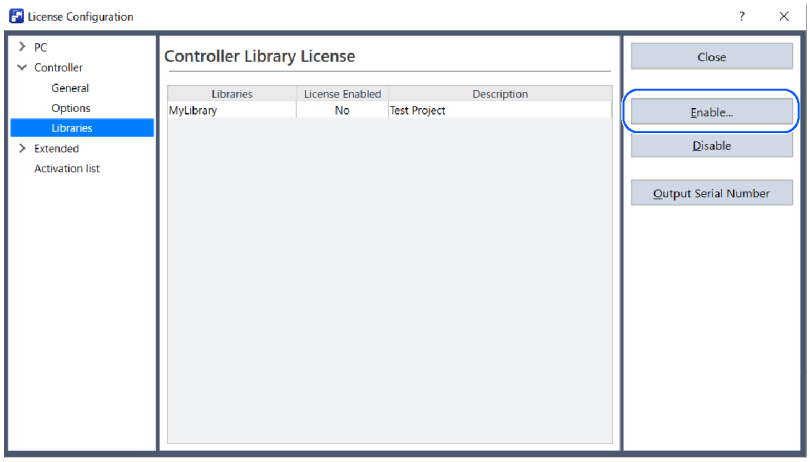
The Controller | Libraries node is only displayed when connected to an actual controller.

3. Make sure that the target library is displayed and click the Enable button.

- PC library license



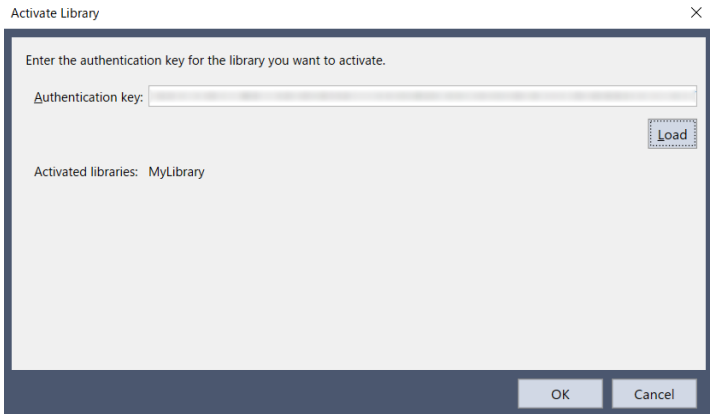
- Controller library license



KEY POINTS

A list of all the libraries in the `C:\EpsonRC80\Libraries` folder is displayed.

4. Enter the authentication key in the Library Activation dialog box that appears, or use the Load button to load a CSV file. If the authentication key is correct, the library activated by the authentication key is displayed.

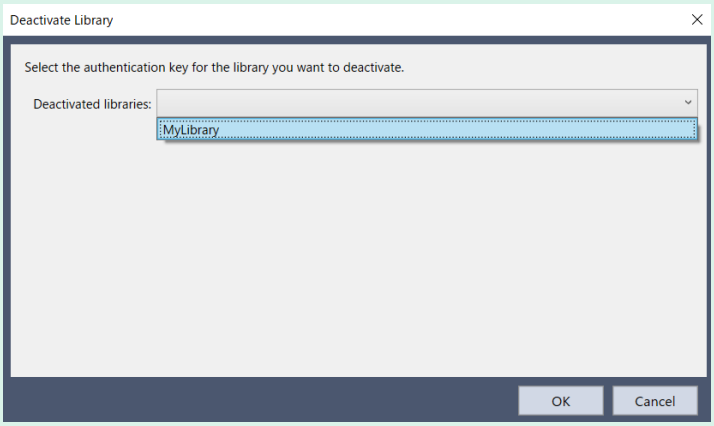


5. Click the OK button and make sure that the target library is activated.

Libraries	License Enabled	Description
MyLibrary	Yes	Test Project

KEY POINTS

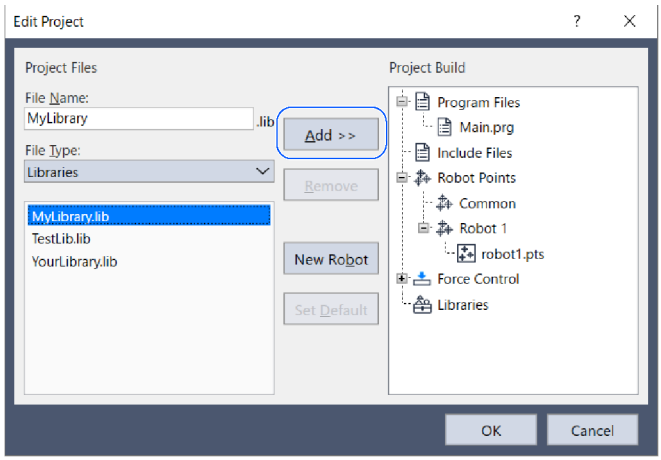
To deactivate a library, click the Disable button, select the library you want to deactivate from the drop-down list in the Deactivate Library dialog box, and click the OK button.



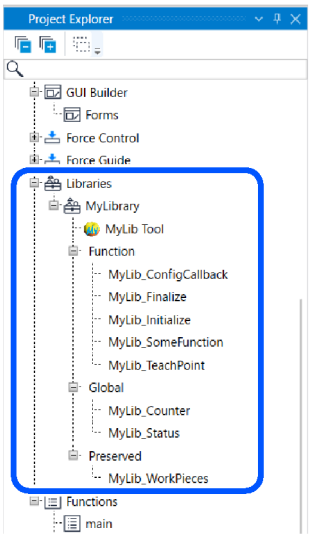
5.3 How to Register a Library to a Project

The procedure for registering a library to a project is described below.

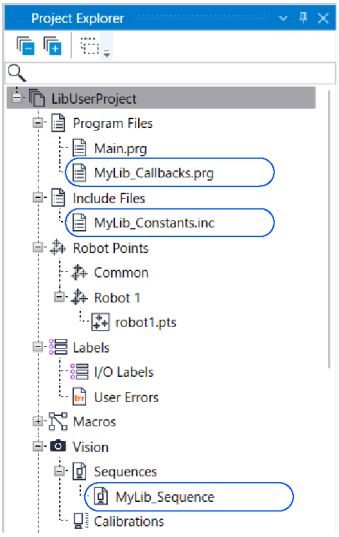
1. Create a project that uses the library.
2. Select Project | Edit, or right-click on Libraries in the Project Explorer and select Add Library to open the Edit Project dialog box.



3. Select Library File as the file type.
4. A list of all the libraries in the `C:\EpsonRC80\Libraries` folder appears.
5. Select a library and click the Add button.
6. Click the OK button.
7. The project explorer displays the available public functions and variables.



8. The library and its public files are added to the project. The names of the public files start with a library-defined prefix. The existence of public files varies depending on the library.



9. Repeat this process to add multiple libraries.
   You can add up to five libraries to one project.

10. Write the library functions you want to execute into your program.

11. Edit the public files in the library as needed.

> ⚠ NOTE
>
> Do not register multiple libraries with the same prefix. Duplicate functions or variables may cause an error.

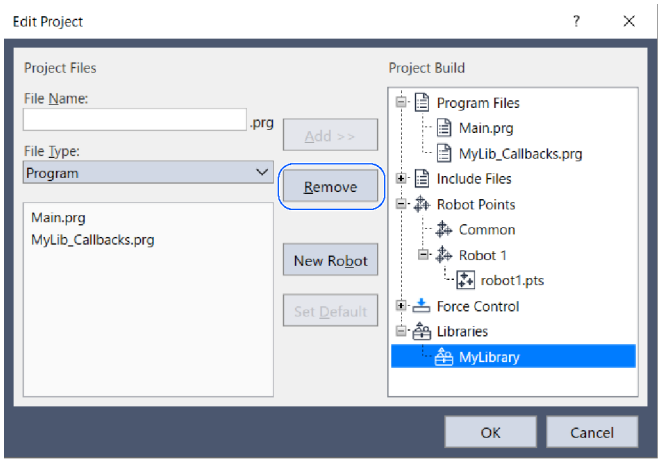
5.4 Releasing Registered Libraries from a Project

This section explains how to release a registered library from a project.

1. Right-click on the library you want to release in project explorer and select [Exclude From Project] to release the library.
   Alternatively, select Project | Edit Project to open the Edit Project dialog box.

i. Select the library you want to release in the project configuration tree and click the Remove button.

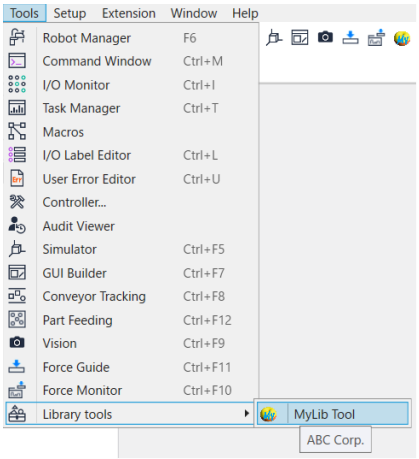
ii. Clicking the OK button closes the Edit Project dialog box and releases the library.



2. If necessary, also release any public files that were added along with the library. They are not released at the same time as the library.

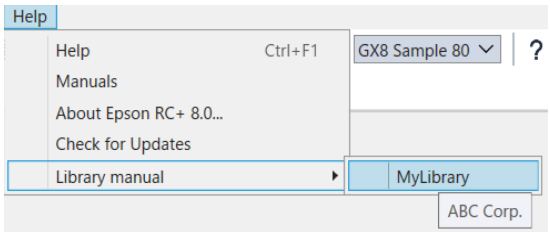
5.5 Using Library Tools

The library may support configuration and execution via a GUI. A menu is added under Tools | Library Tools at this time. Run the library tool from there. For details about the library tools, refer to the manual for each library or contact the developer of the library.



5.6 Displaying the Library Manual

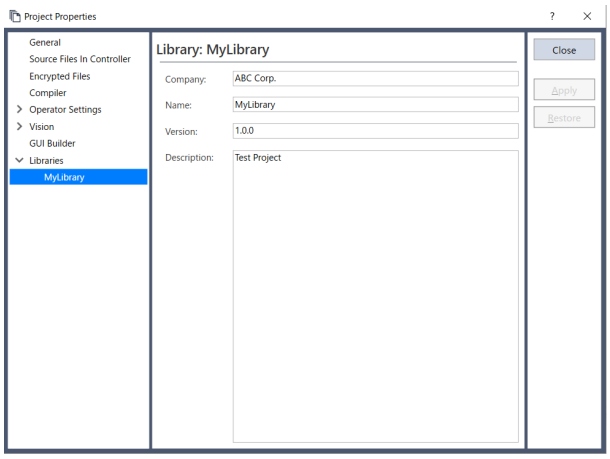
If a manual is provided for the library, you can view it by going to Help | Library Manual.



5.7 Setting Library Properties

You can check the properties of the libraries registered in your project.

1. Select Project | Properties to open the project properties dialog box.
2. If a library is registered, it is added to the tree view. It can be expanded to display the registered libraries.
3. Selecting a library name displays the library creator, library name, version, and description.

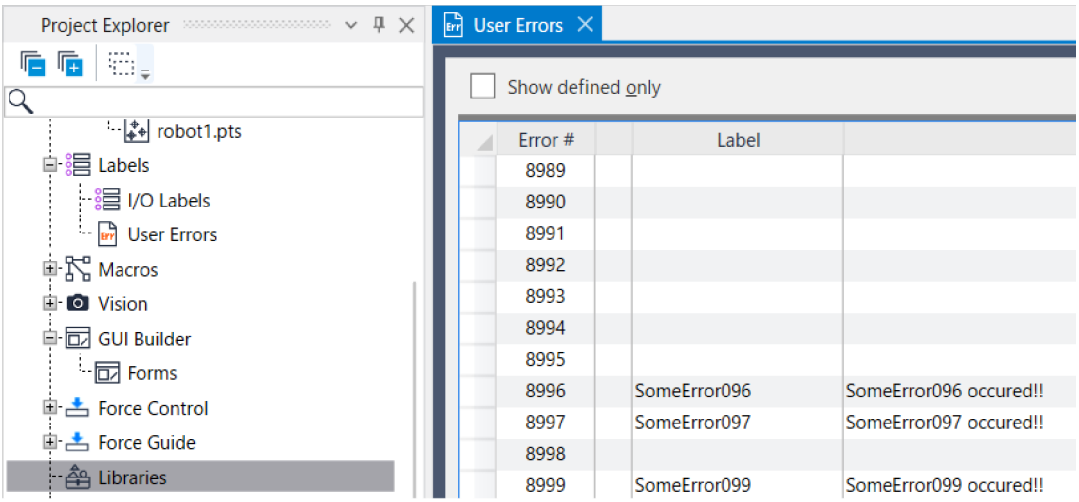


5.8 Defining User Errors in Libraries

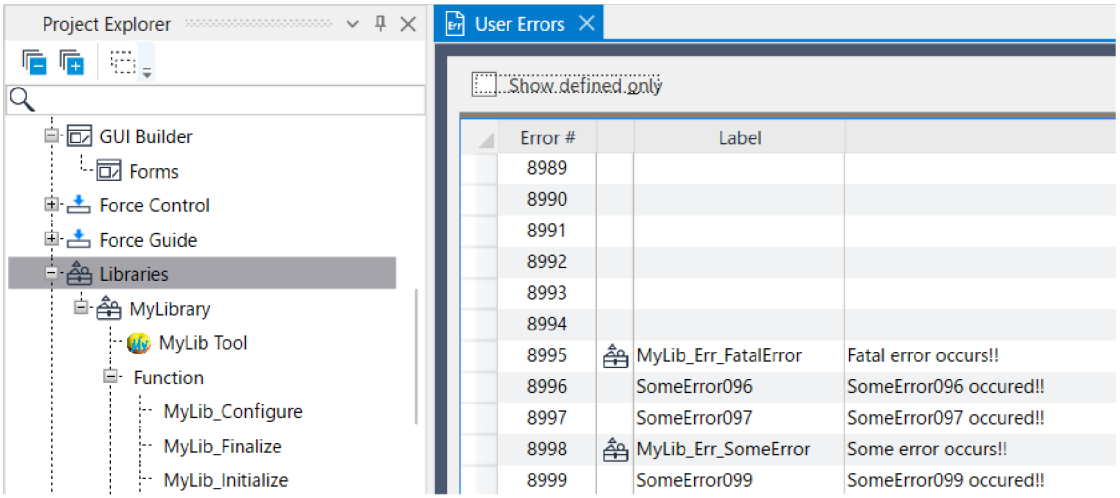
User errors in registered libraries use undefined error numbers in descending order starting from error number 8999. Note that the user error definitions in the library cannot be changed.

Example before and after adding the library:

Before addition



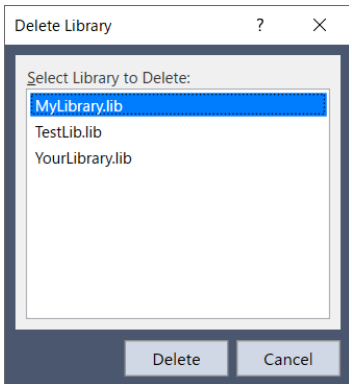
After addition



5.9 Deleting a Library

This section explains how to delete unnecessary libraries from RC+.

1. Select Extensions | Delete Library.
2. Select the library you want to delete from the list in the dialog box that appears.
   This list contains the libraries currently held by RC+.
3. Click the Delete button. The corresponding library is deleted from the `C:\EpsonRC80\Libraries` folder.



6. SPEL+ Command Reference

6.1 SPEL+ Command List

The following SPEL+ commands are available only in the library:

Command/Function	Description/Application
ArchReserve	Dynamically acquires arch numbers exclusively available in the library.
ArmLib	Selects the dynamically acquired arm number.
ArmReserve	Dynamically acquires the arm numbers exclusively available in the library.
LibGetInfo	Acquires the library information included in SPEL+ project.
LocalReserve	Dynamically acquires the local numbers exclusively available in the library.
PointReserve	Dynamically acquires point numbers available in the library.
SignalReserve	Dynamically acquires the signal numbers exclusively available in the library.
SyncLockReserve	Dynamically acquires the SyncLock numbers exclusively available in the library.
TaskReserve	Dynamically acquires the task numbers exclusively available in the library.
TimerReserve	Dynamically acquires the timer numbers exclusively available in the library.
TLReserve	Dynamically acquires the tool numbers exclusively available in the library.
ToolLib	Selects the dynamically acquired tool number.
UploadFileAfterStop	Copies the specified file from the controller to the PC project folder after all tasks stop.

6.2 ArchReserve Function

Dynamically acquires the arch numbers available in the current library.

Format

ArchReserve

Parameters

None

Returns

Reserved arch numbers

Description

By reserving arch numbers, you can use it without it being used by other libraries.
Unreserved numbers from 7 to 13, are reserved in descending order.
The information defined for the reserved numbers are temporary values that are cleared along with reservation when all tasks are completed. Alternatively, you can clear the reservation by specifying arch numbers in ArchClr.

See Also

Arch, ArchClr

ArchReserve Function Usage Example

```
Integer i

i = ArchReserve

Arch i,20,20
Jump3 P2, P3-TLZ(100), P3 C(i) 'Operation using the reserved arch numbers.

ArchClr i
```

6.3 ArmLib Function

Selects the dynamically acquired arm number.

Format

ArmLib ArmNumber

Parameters

ArmNumber: Specify as an integer value or an expression. The effective range is 16 to 31. You can select an arm reserved by this library.

Description

Specifies a reserved arm as the arm for executing robot commands.
You can select an arm number that was reserved by the library that called this command.
Run Arm to display the currently selected arm number.
The operation based on the selected arm number is the same as for Arm.
When the user function that called this command ends, the arm number returns to the number before the function was executed.

See Also

Arm, ArmClr, ArmDef, ArmSet, **ArmReserve Function**

ArmLib Function Usage Example

```
Integer i

i = ArmReserve

ArmSet i, X, Y, Z, U
ArmLib i
Move P1 'Operation using the reserved arm number instead of the currently selected
number.
```

6.4 ArmReserve Function

Dynamically acquires the arm numbers available in the current library.

Format

ArmReserve

Parameters

None

Returns

Reserved arm number

Description

By reserving a number, you can use it without it being used by other libraries. Unreserved arm numbers from 16 to 31 are reserved in descending order.
The information defined for the reserved arm number is temporary values that are cleared along with the reservation when all tasks are completed. Alternatively, you can clear the reservation by specifying the arm number in ArmClr.

See Also

Arm, **ArmLib Function**, ArmClr, ArmDef, ArmSet

ArmReserve Function Usage Example

```
Integer i

i = ArmReserve

ArmSet i, X, Y, Z, U
ArmLib i
Move P1 'Operation using the reserved arm number.
ArmClr i
```

6.5 LibGetInfo

Acquire the library information included in SPEL+ project.
LibGetInfo libraryName, ByRef libraryID, ByRef version, ByRef comment, ByRef libraryPass

Syntax

LibGetInfo libraryName, ByRef version, ByRef comment

Parameters

- libraryName: Specifies the library name as a string (up to 32 characters).
- libraryID: Specifies the string variable that gets the library ID. The library ID is a value that is changed every time a library is created.
- version: Specifies the string variable that gets the version of the corresponding library.
- comment: Specifies the string variable that gets the comment of the corresponding library.
- libraryPass: Specifies the string variable that gets the path where the library is saved. Displayed as "[Data folder]\Libraries[Library name]".

See Also

GetProjectInfo

LibGetInfo Example

```
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 Function

Dynamically acquires the local numbers available in the current library.

Format

LocalReserve

Parameters

None

Returns

Reserved local number

Description

By reserving a number, you can use it without it being used by other libraries. Unreserved local numbers from 16 to 31 are reserved in descending order.
The information defined for the reserved local number is temporary values that are cleared along with the reservation when all tasks are completed. Alternatively, you can clear the reservation by specifying the local number in LocalClr.
The point defined for the reserved local number is stored as local number 0.

See Also

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

LocalReserve Function Usage Example

```
Integer i

i = LocalReserve

Local i, X, Y, Z, U
P1 = Here /(i)
Move P1 'Operation using the specified local number.
```

6.7 PointReserve Function

Dynamically acquires the point numbers available in the current library.

Format

PointReserve

Parameters

None

Returns

Reserved point number

Description

By reserving a number, you can use it without it being used by other libraries.
The information defined for the reserved point number is temporary values that are cleared along with the reservation when all tasks are completed. Alternatively, you can clear the reservation by specifying the point number in PDel.

See Also

P#, PDel

PointReserve Function Usage Example

```
Integer i

i = PointReserve

P(i) = Here
Move P(i) +Z50 'Operation using the specified point number.
PDel 1
```

6.8 SignalReserve Function

Reserves the signal number that can be specified for SigWait.

Format

SignalReserve

Parameters

None

Returns

Reserved signal number

Description

By reserving a signal number, you can use it without it being used by other libraries. This is a temporary value that is cleared when all tasks are completed. A dedicated reservation signal number is reserved to reduce the risk of signal numbers used by library users being overwritten.

See Also

WaitSig

SignalReserve Function Usage Example

```
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 Function

Reserves the signal number to be specified for SyncLock.

Format

SyncLockReserve

Parameters

None

Returns

Reserved signal number

Description

Specifying SyncLock and SyncUnLock offers exclusive file access, etc.
By reserving a signal number, you can use it without it being used by other libraries. This is a temporary value that is cleared when all tasks are completed. A dedicated reservation signal number is reserved to reduce the risk of signal numbers used by library users being overwritten.

See Also

SyncLock

SyncLockReserve Function Usage Example

```
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 Function

Reserves the task number that can be specified for task execution.

Format

TaskReserve

Parameters

None

Returns

Reserved task number

Description

By reserving a number, you can use it without it being used by other libraries. This is a temporary value that is cleared when all tasks are completed.
Unreserved task numbers are reserved in descending order.

See Also

WaitSig

TaskReserve Function Usage Example

```
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 Function

Reserves the timer number to be specified for Timer.

Format

TimerReserve

Parameters

None

Returns

Reserved timer number

Description

You can measure the cycle time exclusively by specifying it in the Tmr function.
By reserving a timer number, you can use it without it being used by other libraries. This is a temporary value that is cleared when all tasks are completed. A dedicated reservation timer number is reserved to reduce the risk of timer numbers used by library users being overwritten.

See Also

Tmr, TmReset

TimerReserve Function Usage Example

```
Function N4_1_Call_Hidden

  t1 = TimerReserve

  TmReset t1              'Reset Timer 0

  Xqt ...

  Print Tmr(t1) / 10      'Calculate and display the cycle time
Fend
```

6.12 TLReserve Function

Dynamically acquires the tool numbers available in the current library.

Format

TLReserve

Parameters

None

Returns

Reserved tool number

Description

By reserving a number, you can use it without it being used by other libraries. This is a temporary value that is cleared when all tasks are completed. Alternatively, you can clear the reservation by specifying the tool number in TLClr.
Unreserved tool numbers from 16 to 31 are reserved in descending order.

See Also

TLClr, TLDef, TLSet, Tool, **ToolLib Function**

TLReserve Function Usage Example

```
Integer i

i = TLReserve

TLSet i, X, Y, Z, U
ToolLib i
Move P1 'Operation using the specified tool number.
TLClr i
```

6.13 ToolLib Function

Selects the dynamically acquired tool number.

Format

ToolLib ToolNumber

Parameters

ToolNumber
Specify as an integer value or an expression. The effective range is 16 to 31. You can select a tool reserved by this library.

Description

Specifies a reserved tool as the tool for executing robot commands.
You can select a tool number that was reserved by the library that called this command.
Run Tool to display the currently selected tool number.
The operation based on the selected tool number is the same as for Tool.
When the user function that called this command ends, the tool number returns to the number before the function was executed.

See Also

TLClr, TLDef, TLSet, **TLReserve Function**, Tool

ToolLib Function Usage Example

```
Integer i

i = TLReserve

TLSet i, X, Y, Z, U
ToolLib i
Move P1 'Operation using the reserved tool number.
```

6.14 UploadFileAfterStop Function

Copies the specified file from the controller to the PC project folder after all tasks stop.

Syntax

UploadFileAfterStop fileName

Parameters

- fileName: Species the file names contained in the project.

See Also

GetProjectInfo

UploadFileAfterStop Example

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