EPSON

Epson RC+ 8.0扩展功能
Library Builder 8.0

翻译版

Rev.2
SCM256S7321R

目录

1. 简介

1.1 简介

感谢您购买本公司的机器人系统。

本手册记载了正确使用Library Builder的所需事项。

安装该机器人系统前，请仔细阅读本手册与其他相关手册。

阅读之后请妥善保管，以便随时取阅，如有不明之处，请再次阅读。

本公司的产品均通过严格的测试和检查，以确保机器人系统的性能符合本公司的标准。但是如果在超出本手册所描述的环境中使用本产品，则可能会影响产品的基本性能。

本手册阐述了本公司可以预见的危险和问题。请务必遵守本手册中的安全注意事项，安全正确地使用机器人系统。

1.2 商标

Microsoft、Windows、Windows标识、Visual Basic、及Visual C++为美国Microsoft Corporation在美国或其它国家的注册商标或商标。

其它品牌与产品名称均为各公司的注册商标或商标。

1.3 关于标记

Microsoft® Windows® 10 Operating system
Microsoft® Windows® 11 Operating system
本使用说明书将上述操作系统分别标记为Windows 10、Windows 11。另外，有时可能将Windows 10、Windows 11统一标记为Windows。

1.4 注意

禁止擅自复印或转载本手册的部分或全部内容。
本手册记载的内容将来可能会随时变更，恕不事先通告。
如您发现本手册的内容有误或需要改进之处，请不吝斧正。

1.5 制造商

SEIKO EPSON CORPORATION

1.6 联系方式

联系方式的详细内容登载于以下手册中的“销售商”处。
各地区的咨询处有所不同，敬请注意。
"安全手册”- 联系方式"

从以下网站也可浏览安全手册。
URL： https://download.epson.biz/robots/

1.7 操作入门

本节介绍了您在阅读本手册之前应了解的事项。

关于Epson RC+ 8.0的安装文件夹

Epson RC+ 8.0可安装在任意指定的路径。本手册中是默认Epson RC+ 8.0被安装C:\EpsonRC80中进行说明。

符号含义

使用下述标记来记载安全注意事项。请务必阅读。

⚠ 警告

如果用户忽视该指示或处理不当，可能会导致死亡或重伤。

⚠ 警告

如果用户忽略该指示或处理不当，可能会因触电而受伤。

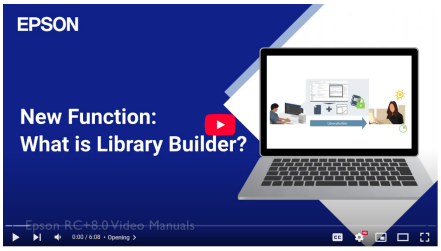
⚠ 注意

如果用户忽略该指示或处理不当，可能会导致人身伤害或财产损失。

2. 概述

2.1 概述

Library Builder在视频中也有说明。

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	

要点

- 若要浏览视频，需要连接互联网。
- 会播放声音。
- 如果要以母语显示，请使用YouTube的自动翻译字幕功能。

Library Builde为以下所有功能的总称。

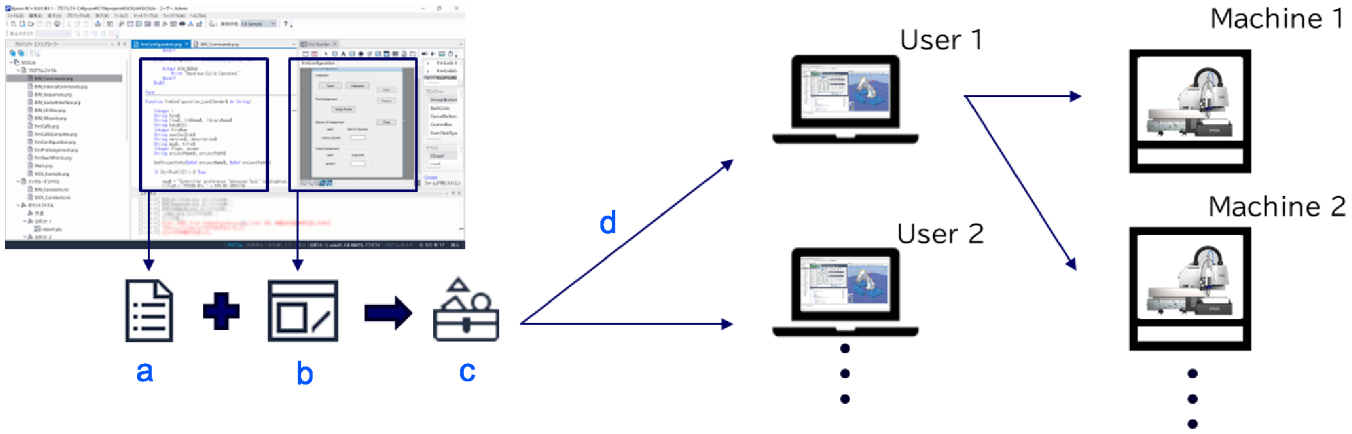
- 将SPEL+项目的配置数据和配置文件打包成库的功能
- 可将创建的库添加至另外的SPEL+项目并使用的功能

做成库后的SPEL+项目组件中，以下组件可从添加了库的SPEL+项目使用。

- 全局变量
- 备份变量(Global Preserve)
- SPEL+函数

- 在GUI Builder中创建的窗体
- 用户错误
- 程序文件
- 包含文件
- 视觉顺序

通过使用Library Builder，可自行创建对制造设备和第三方机器设置进行辅助的库，提供给用户。



符号	描述
a	• 程序 • 函数 • 变量
b	• 设置画面 • 显示画面
c	库
d	提供

要点

创建、使用库均需要Epson RC+ 8.0 Ver8.1.0.0或更高版本。而且，创建库需要Epson RC+ Premium Edition许可证。
此外，创建库时有用的SPEL+命令可在以下控制器固件版本或更高版本使用。

- RC800系列：8.1.0.0
- RC90、RC700系列：7.5.5.0
- T、VT系列：7.5.55.0

有关SPEL+命令的详细信息，请参阅以下章节。

- **SPEL+命令参考**
- “SPEL+语言参考 - Appendix C: Epson RC+ 8.0的命令 - C-2: Epson RC+ 8.0各版本添加的命令列表”

3. 开发库

3.1 库的组件

- 做成库的SPEL+项目组件分为公用组件和专用组件。
- 公用组件可从添加了库的SPEL+项目直接使用。
- 专用组件无法从添加了库的SPEL+项目直接使用。
- 公用组件和专用组件根据有无前缀区分。
- 公用程序文件和包含文件称为公开文件，可在添加了库的SPEL+项目中编辑。

功能	公用组件	专用组件
从使用库的SPEL+项目使用	可	不可
使用库的用户进行编辑	可	不可

- 公用组件的用途如下所示。

组件	指定为公用时的用途
函数	使用目的是为库用户提供作为函数的功能。
全局变量	用于以变量值表示库的动作状态和指定动作所需的值。
备份变量	用于与全局变量相同的用途。要将值保留在控制器内并在下次启动时继续使用，设为备份变量。
程序文件	・用于提供库的公共函数使用示例。 ・在希望库用户描述从库内所调用公共函数（回调函数）的处理时使用。
包含文件	关于公共函数的参数和返回值，使用时库用户可作为常数处理。通过设为常数，使库用户更容易理解值的含义。
视觉顺序	使用目的是为库用户提供视觉顺序。

3.2 创建用于库的项目

1. 创建库需要Epson RC+ Premium Edition许可证。请从销售商购买许可证后激活。
   有关许可证激活，请参考以下手册。

   “Epson RC+ 8.0用户指南 - 系统操作 - Epson RC+ 8.0的启动 - 许可证激活”

2. 新建用于库的项目。

3. 在想对使用库的用户公开的信息在添加前缀（前置字符）。包括末尾的下划线在内，前缀请指定最多10个字符。
   可公开的信息有以下内容。

信息	公开条件
函数	函数名应以前缀开头
全局变量	变量名应以前缀开头
备份变量	变量名应以前缀开头

信息	公开条件
程序文件	应满足以下所有条件 ・文件名以前缀开头 ・文件内列出的函数、常数、全局变量、备份变量均以前缀开头 ・文件内列出的仅有函数、全局变量、备份变量、注释 ・未包含不满足公开条件的包含文件
包含文件	应满足以下所有条件 ・文件名以前缀开头 ・文件内列出的常数均以前缀开头 ・文件内列出仅有常数和注释・未包含不满足公开条件的包含文件
视觉顺序	视觉名应以前缀开头

例：前缀为“MyLib_”时

```
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：公开程序文件

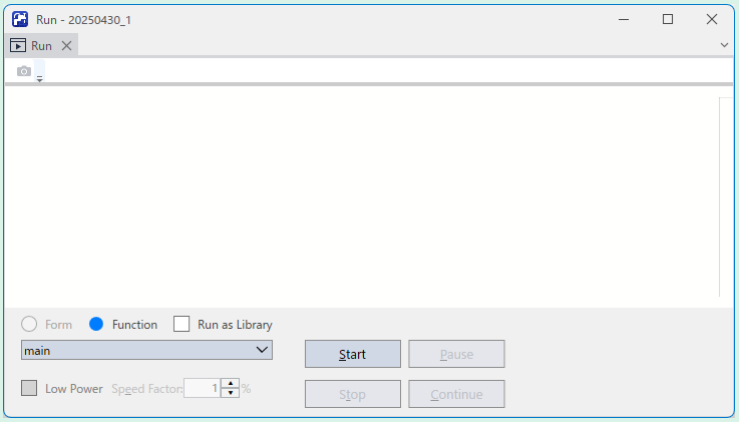
要点

使用多个库时，请加上库特有的前缀，以免函数名和变量名等重复。

1. 请开发包括上述公开信息在内的项目。

要点

在Epson RC+ Premium Edition中，运行窗口上会显示[作为库运行]复选框。启用后，可从项目执行仅在库中可使用的命令。确认动作时请充分利用。



2. 提供库工具时，请通过[工具]-[GUI Builder]，创建作为库工具显示的窗体，并设置为启动窗体。
   有关库工具，请参考以下内容。

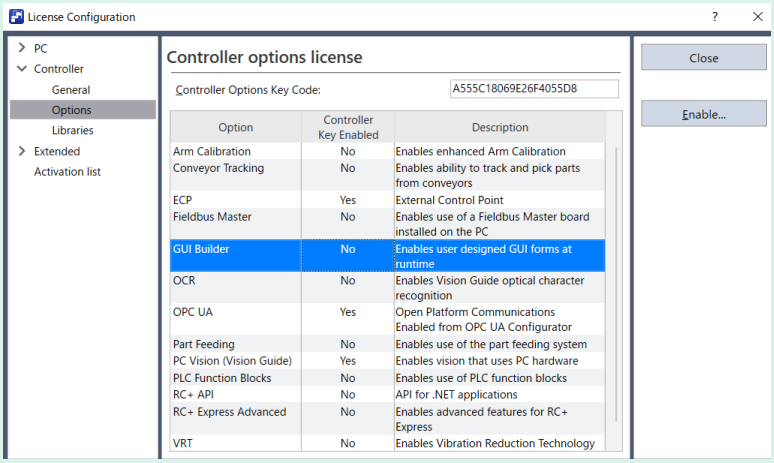
   使用库工具

   有关GUI Builder的使用方法，请参考以下手册。

“Epson RC+ 8.0选件GUI Builder 8.0”

要点

在Epson RC+ Premium Edition中，即使GUI Builder选件无效，也可执行窗体和GUI Builder相关命令。



3. 根据需要，创建作为库手册显示的PDF文件、作为库工具图标显示的图像文件、设备设置以及手册等添加文件。

要点

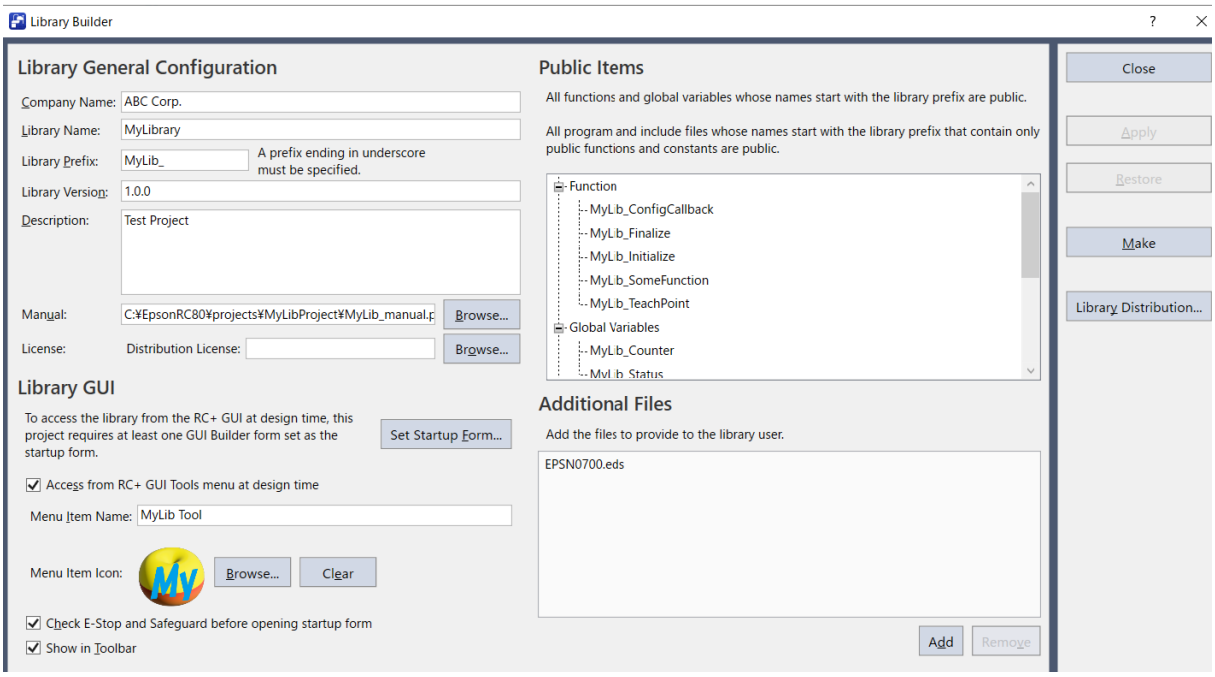
- 在Epson RC+ 8.0 Ver8.1.0.0中，请勿在用于库的项目中登记另外的库。具体来说，在库A的内部无法使用另外的库B创建库A。Library Builder启动时将显示错误。
- 可将库内部使用的设置值作为使用库的用户的项目数据[任意名称].libcfg保留。
  例如，提供库工具时，将用户使用GUI设置的值保留在[任意名称].libcfg中，下次启动时从该文件加载上次的设置值，可重现上次的设置状态。
  [任意名称]中请指定与其他库不重复的名称。

3.3 创建库

1. 打开用于库的项目。
2. 从[扩展]菜单中选择[Library Builder]。
3. 如果项目创建成功，[Library Builder]对话框将打开。

要点

根据TP4的操作模式，如下所示执行动作。
・AUTO时，可使用Epson RC+，但在Epson RC+ 8.0 Ver8.1.0.0中无法创建库（无法启动Library Builder）。
・TEACH、TEST时，无法使用Epson RC+，因此也无法创建库。
・无论TP4的操作模式如何，可在项目中使用已有的库。
有关TP4的操作模式，请参考以下手册。
“机器人控制器选件 示教器TP4手册 - 操作模式（TEACH、AUTO、TEST）”



项目	描述
公司名	输入公司名。 最多为32个单字节字符。
库名	输入库名。 最多为32个单字节字符。
前缀	输入前缀名。 最多为10个单字节字符，需要以下划线结束。
版本	输入库的版本。 最多为32个单字节字符。
注释	输入库的说明。 最多为255个字符。
手册	设定要创建的库的手册（PDF文件）。
Distribution License	输入EPSON分发的Distribution License。 也可从[浏览]按钮加载包含许可证的PDF文件。
启动窗体设置	显示启动窗体设置画面。将作为库GUI（库工具）初始显示画面使用的窗体设置为启动窗体。
开发时从RC+的工具菜单使用库GUI	将在GUI Builder中创建的窗体作为从工具菜单启动的库工具嵌入库。项目内存在已设置为启动窗体的窗体时可勾选。 默认为关闭。 有关库工具，请参考以下内容。 使用库工具
菜单显示名	输入在工具菜单中作为库工具显示的名称。 最多为32个单字节字符。
菜单显示图标	设置在工具菜单中作为库工具显示的图标。

项目	描述
启动窗体启动时检查紧急停止、安全门状态	启动窗体（库工具）启动时，检查紧急停止、安全门状态。 默认为开启。
显示在工具栏上	在工具栏上显示启动库工具的图标。
公开信息	显示已添加前缀的函数、全局变量、备份变量、程序文件、包含文件和视觉序列。
添加文件	关于库中使用的设备，添加设备的设置文件（现场总线设置用文件(eds)等）和手册等，以便用户设置、引用。 复制到C:\EpsonRC80\Libraries文件夹中该库文件夹中的AdditionalFiles文件夹。
关闭	关闭对话框。
适用	保存更改。
撤销	返回之前的设置。
创建	创建库。
库分发	库分发向导启动。输出分发给用户的库和激活密钥。 有关详细信息，请参阅以下章节。 创建用于分发的库

4. 指定前缀。包含指定前缀的函数、全局变量、备份变量、程序文件、包含文件和视觉序列自动添加至公开信息区。

要点

对于程序文件、包含文件，即使文件名中添加了前缀，如果文件内包含未添加前缀的函数、常数，将不作为公开信息显示。

5. 指定公司名、库名、版本、注释、手册文件、添加文件。

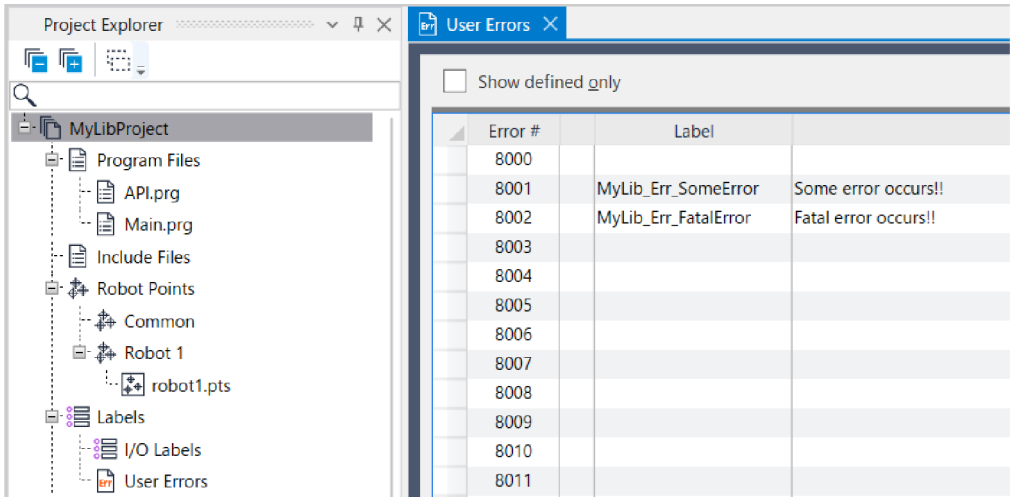
6. 提供库工具时，勾选[开发时从RC+的工具菜单使用库GUI]，并指定在Epson RC+工具菜单中显示的菜单显示名以及菜单显示图标。
   如果勾选[启动窗体启动时检查紧急停止、安全门状态]，控制器将在紧急停止状态、安全门打开状态时显示警告消息，在这些状态解除前，库工具不会启动。

7. 单击[应用]按钮以保存设置更改。

8. 单击[创建]按钮以创建库。
   在C:\EpsonRC80\Libraries文件夹中生成创建的库文件和zip文件。

3.4 报告库内定义的用户错误

1. 定义用户错误。
   此时不仅要定义消息，标签也务必定义。



2. 创建用户错误报告用函数。

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

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

有关UserErrorNumber、Error，请参考以下手册。

“SPEL+语言参考”

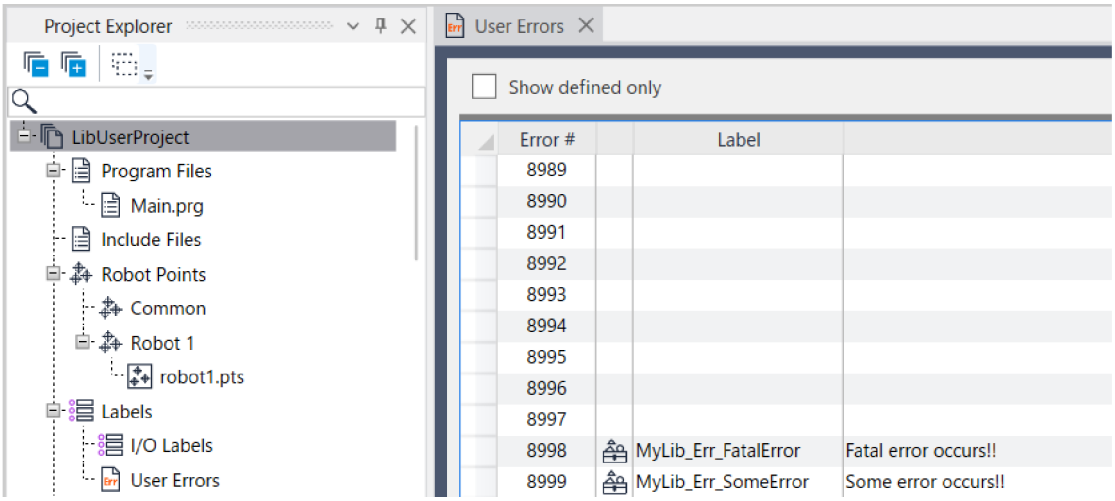
3. 实施错误处理。
用于库的项目代码内的错误处理会调用上述用户错误报告用函数。

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

4. 按照下述步骤创建库。

创建库

5. 使用库时，在项目中登记上述步骤中创建的库后，将从用户错误的末尾开始，按顺序登记至空的错误编号。



与创建库时的错误编号不同，将在库侧根据用户错误标签报告错误，因此会显示正确消息。

3.5 使用回调函数

回调函数是可在库的公用函数处理中放入项目处理的功能。
库的开发者在公开程序文件中通过库指定的函数名、参数和返回值描述回调函数雏形（骨架代码）和说明。
库用户在项目中添加库时会添加公开程序文件，因此可查看文件内的说明并实施必要处理。

例：回调函数描述示例

```
Function Lib_Callback(num As Integer) As Boolean
   'XXX处理之后从库调用。
   '参数: 检查num的值是否恰当，如果恰当，返回True，否则返回False。
Fend
```

具体有以下用途。

・在库处理中，根据用户程序改变后续处理。
【具体示例】

- 库监视输入位的值，如果Off状态持续一定时间以上，将调用返回值为Boolean型的回调函数。返回值为False将继续处理，为True则结束处理。
- 库用户通过回调函数如下实施操作，决定继续/结束后续处理。
  - 另一输入位为On时：让操作员通过MsgBox命令选择继续/结束，结果作为回调函数的返回值。
  - 另一输入位为Off时：返回值为True，无需操作员选择即结束处理。

・公用函数的处理需要较长时间时的进度显示和中断。
【具体示例】

- 库定期调用通过参数表示进度百分比、返回值为Boolean型的回调函数。返回值为True时，结束处理。
- 库用户在GUI Builder的窗体中，用GSet提取通过回调函数获得的进度百分比后显示。此外，窗体上的中止按钮被按下时，在调用回调函数之时将回调函数的返回值置为True。

3.6 使用libcfg文件的同步

SPEL项目中具有像点数据那样，可将程序动作所需的设定和计算值保留在文件中的功能。
动作中保存至控制器，也随时同步至运行RC+的PC上的项目中，因此可顺利进行SPEL项目的开发、维护以及移植到其他环境中。

使用库时，根据作为操作对象的设备和功能，可能会产生必需的信息，因此对可在任意格式下使用的文件提供相同功能。
要保留以下信息时请考虑。

- 功能和设备参照的设定文件
- 功能和设备保留・参照计算结果的文件
- 保留・监视功能和设备动作状态的日志文件

如果在控制器内操作扩展名为“libcfg”的文件，RC+连接时将进行同步确认并导入PC侧。格式无限制。

例：libcfg文件使用示例：文件导出

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

   iFileID = FreeFile
   ChDisk FLASH                        '指定控制器的项目文件夹
   WOpen "Lib1.libcfg" As #iFileID  '在所需模式下打开

   Print #iFileID, "Name: ", name$  '写入设定的种类和值

   '按参数数量重复操作。

   Print #iFileID, "ParamB: ", paramB  '写入设定的种类和值
   Close #iFileID
Fend
```

例：libcfg文件使用示例：文件读出

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

      iFileID = FreeFile
      ChDisk FLASH                        '指定控制器的项目文件夹
      ROpen "Lib1.libcfg" As #iFileID  '在所需模式下打开

      Line Input #iFileID, Buf$
      iPos = InStr(Buf$, ": ")      '将设定的种类和值细分后加载。
      name$ = Mid$(Buf$, iPos + 2)

      '按参数数量加载。

      Line Input #iFileID, Buf$
      iPos = InStr(Buf$, ": ")      '将设定的种类和值细分后加载。
      paramB = Val(Mid$(Buf$, iPos + 2))

      Close #iFileID
   Fend
```

因所用设备规格等需要使用特殊扩展名时，请使用“UploadFileAfterStop”命令。通过在库内列出，结束任务后会从控制器内导入指定的文件。但执行动作需连接RC+。

4. 创建用于分发的库

4.1 创建用于分发的库

介绍如何创建库，用于分发给希望使用库的用户（以下称为库用户）。

可以在分发库时指定可使用库的用户。请在根据分发形式设定库的价格等情况下使用。

符号	可使用库的用户	用途	必需的信息
a	仅指定控制器的使用者	允许通过特定控制器使用。 库用户必须按要执行动作的控制器台数购买相应数量的库并启用。 库开发者按台数得到相应数量的序列号后，发布激活密钥。	Distribution License, 库用户的控制器序列号（也可从库用户的RC+输出。请事先获取）
b	仅指定PC的使用者	允许通过特定PC上运行的RC+的虚拟控制器使用。 不同于实际控制器。 库用户必须按要执行动作的PC台数购买相应数量的库并启用。 库开发者按台数得到相应数量的PC硬件ID后，发布激活密钥。	Distribution License, 库用户的PC硬件ID （可从库用户的RC+输出。请事先获取）
c	获得激活密钥的所有使用者	库用户使用时不受控制器、PC限制。 库用户需要进行启用。	Distribution License, 库开发者指定的任意字符串（仅用于设定。启用库用户时不需要）
d	获得库的所有使用者	库用户使用时不受控制器、PC限制。 库用户不需要进行启用即可使用。	Distribution License

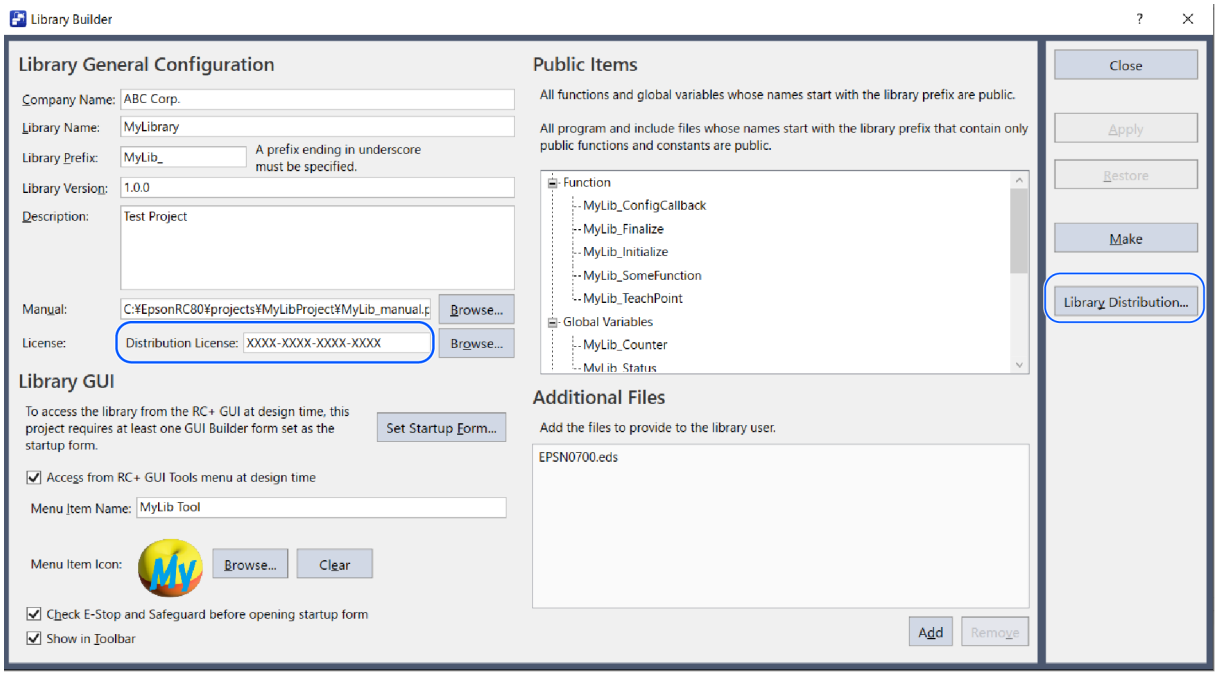
1. 打开要分发库的项目。
2. 从[扩展]菜单中选择[Library Builder]。
3. 如果项目创建成功，[Library Builder]对话框将打开。

要点

在Epson RC+ 8.0 Ver8.1.0.0中，无法通过TP4（操作模式：AUTO）使用Library Builder。
有关TP4（操作模式：AUTO），请参考以下手册。

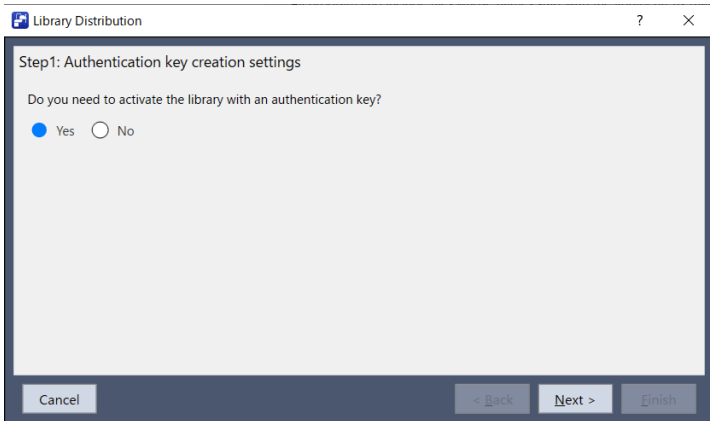
“机器人控制器选件 示教器TP4手册 - 操作模式（TEACH、AUTO、TEST）”

4. 输入EPSON分发的Distribution License。



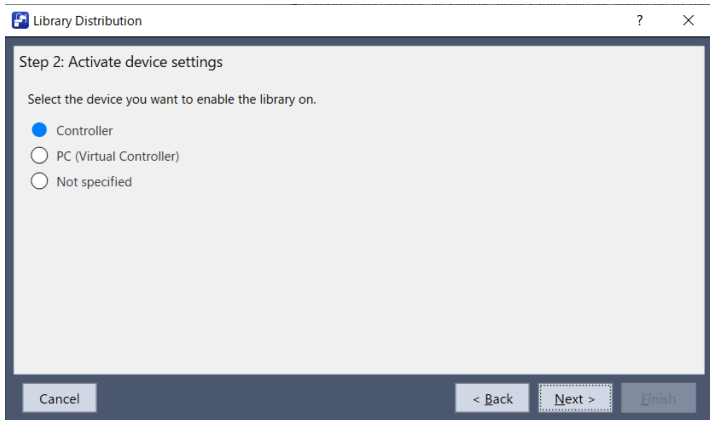
5. 单击[库分发]按钮。

6. 在显示的[库分发]向导中选择启用所分发的库是否需要激活密钥。



- [是]：可使用库的用户除d以外时选择。请进入下一步。
- [否]：可使用库的用户为d时选择。请进入步骤9。

7. 根据想指定的库用户，选择要启用库的设备。

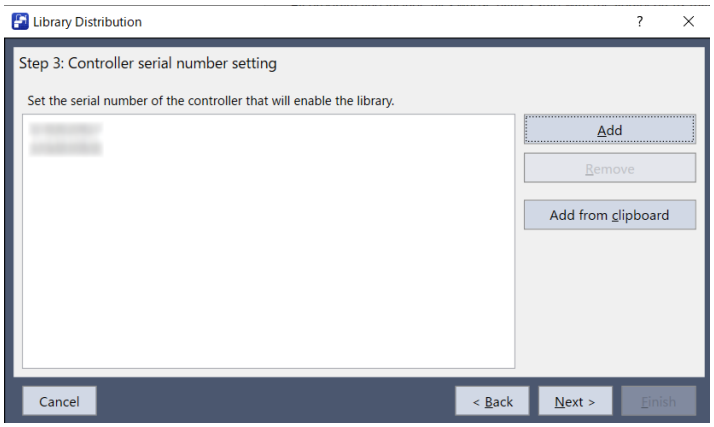


- [控制器]：可使用库的用户为a时选择。
- [PC(虚拟控制器)]：可使用库的用户为b时选择。

- [不指定]：可使用库的用户为c时选择。

8. 根据所选设备，输入以下信息。

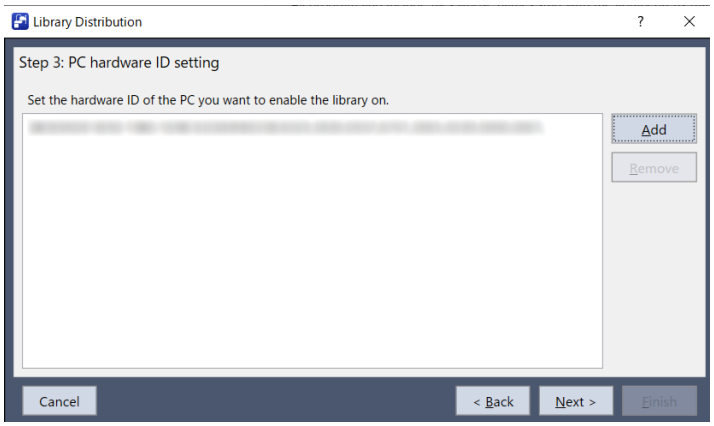
- **选择[控制器]时：**
  通过[添加]按钮，加载从库用户处获取、写有控制器序列号的csv文件。
  单击[从剪贴板添加]按钮后，将添加剪贴板中的字符串。请在复制粘贴序列号时使用。



要点

可加载多个序列号，一次性生成多个控制器的激活密钥。

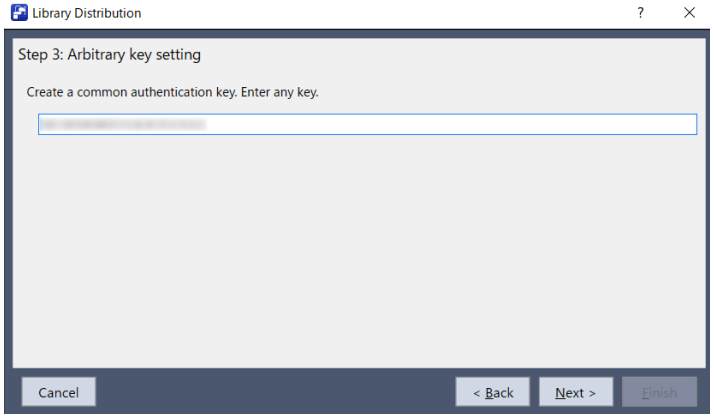
- **选择[PC(虚拟控制器)]时：**
  通过[添加]按钮，加载从库用户处获取、写有PC硬件ID的csv文件。



要点

可加载多个PC硬件ID，一次性生成多个PC的激活密钥。

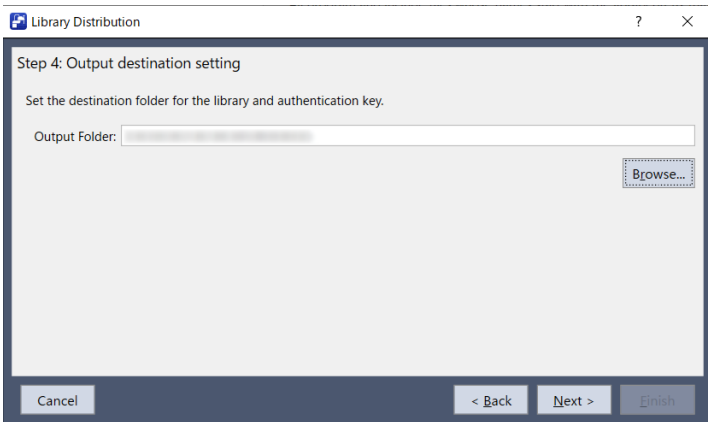
- **选择[不指定]时：**
  输入任意密钥。可使用最多32个字符的字母数字和下划线、连字符。
  即使不输入密钥，也可创建“c. 仅登记后的使用者可使用”的库进行分发。



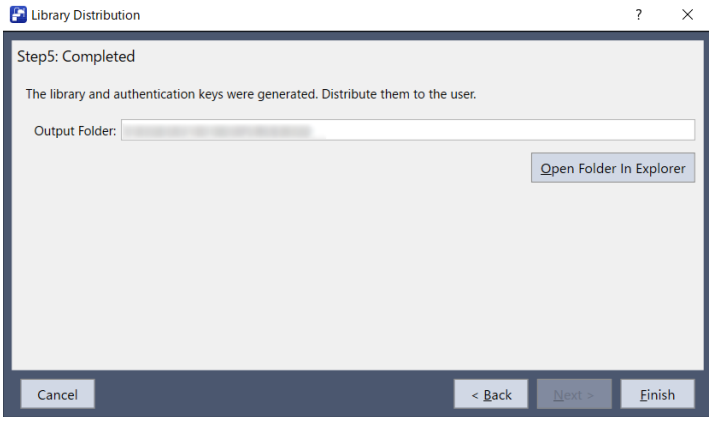
要点

生成的激活密钥无论在哪个设备均可启用库。

9. 指定库和激活密钥的输出目标。



10. 输出库和激活密钥（如果在步骤6中选择[否]，则不输出激活密钥）。
单击[使用资源管理器打开文件夹]按钮后，将显示输出目标文件夹。



5. 使用库

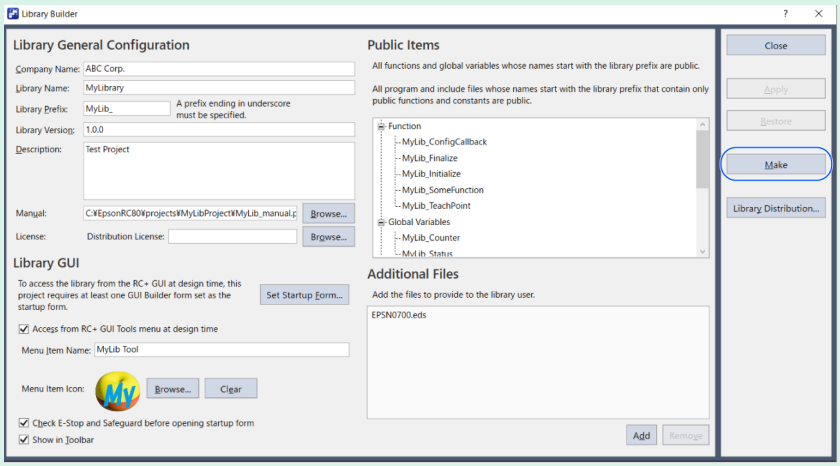
5.1 导入库

介绍将获得的库引入Epson RC+的方法。

1. 库以zip文件的形式分发。
2. 从[扩展]菜单中选择[导入库]。
3. 从显示的文件选择对话框中选择要导入的库。
4. 单击[打开]。在C:\EpsonRC80\Libraries文件夹中生成导入的库文件和zip文件。如果已有同名的库时，将显示是否覆盖的确认消息。

要点

在同一个Epson RC+中通过Library Builder的[创建]按钮创建的库无需导入。



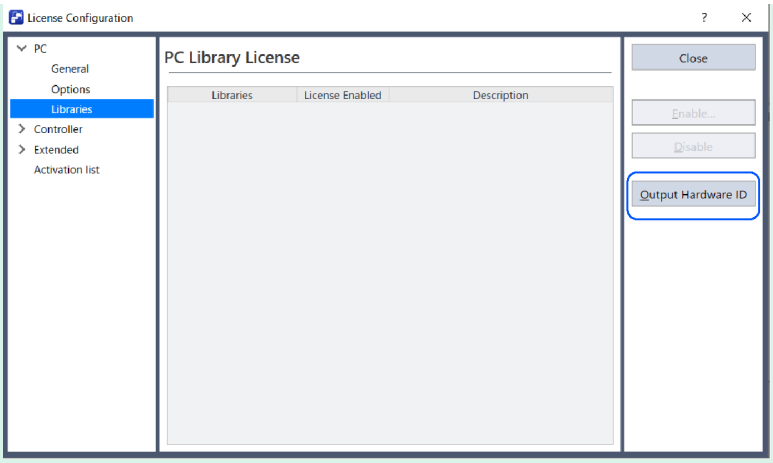
5.2 启用库

对于需要激活的库，需要启用库。介绍如何启用导入的库。
要启用库，需要库的激活密钥。
如果不启用，则无法使用该库。无需激活的库则不需要启用就可使用。

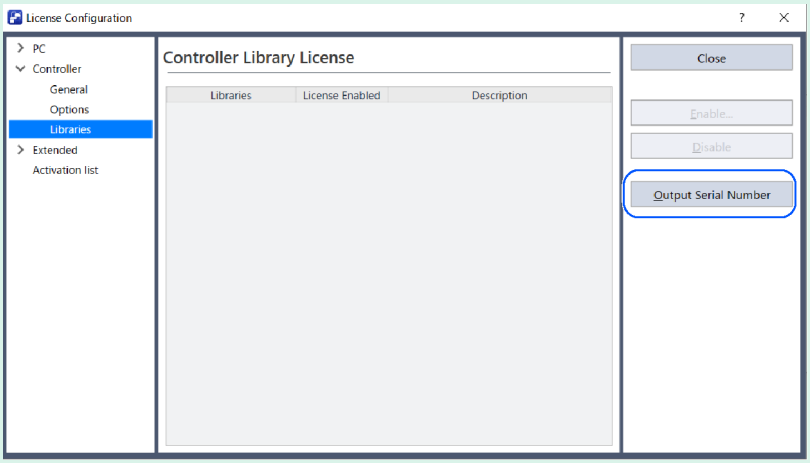
要点

激活密钥由库的创建者发行。请根据需要，将以下任意一项或两项信息发送给库的创建者。

- **PC的硬件ID**
  可在[设置] - [许可证设置]画面中，通过使用[PC] - [库]显示的[输出硬件ID]按钮保存为csv文件。



- **控制器的序列号**
  可在[设置] - [许可证设置]画面中，通过使用[控制器] - [库]显示的[输出序列号]按钮，将所连接控制器的序列号保存为csv文件。



要点

如果在未启用状态下使用库，创建项目时或启动库工具时将显示错误。

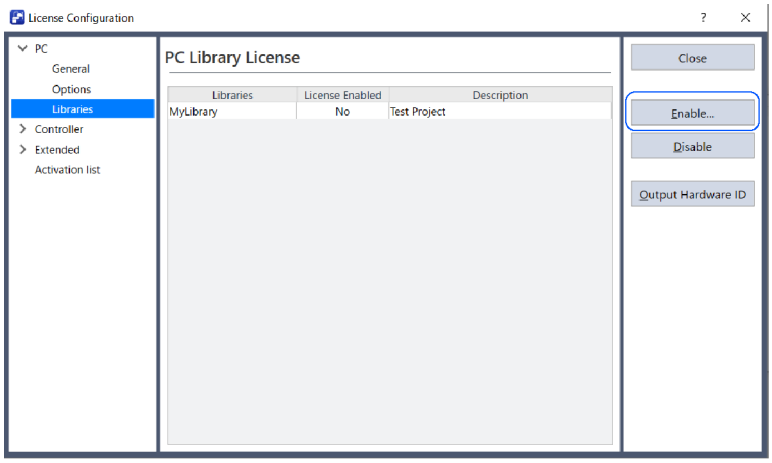
1. 单击[设置] - [许可证设置]。
2. 在[许可证设置]画面的树视图中，选择[PC] - [库]或[控制器] - [库]。

要点

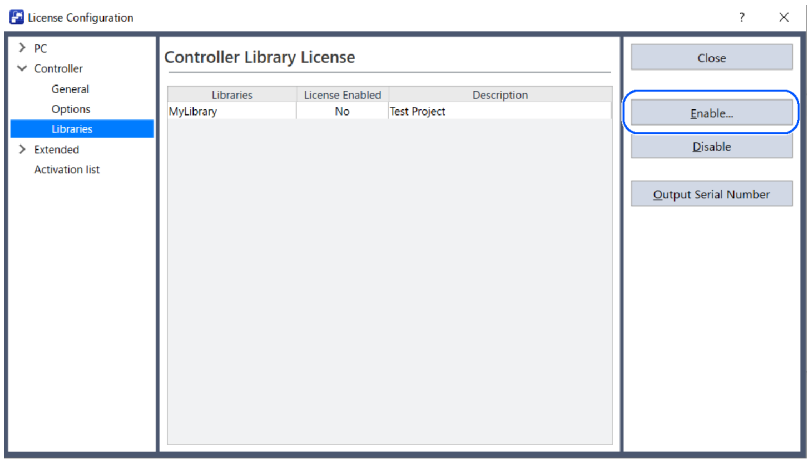
[控制器] - [库]节点仅在连接至实际控制器时显示。

3. 确认显示目标库后，单击[启用]按钮。

- 为PC库许可证时



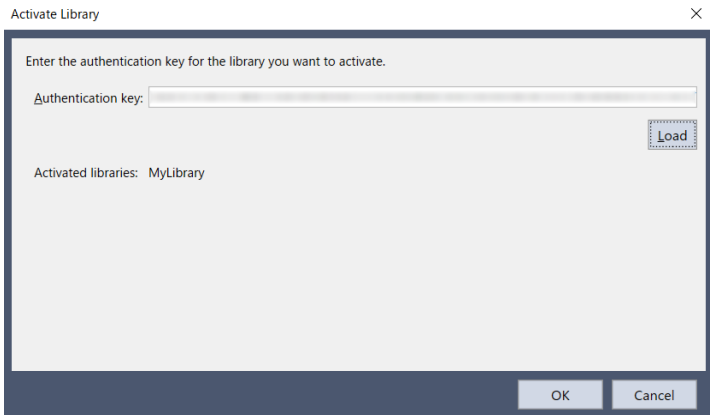
- 为控制器库许可证时



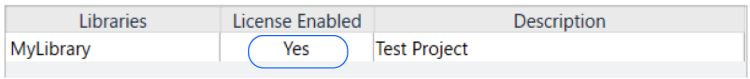
要点

C:\EpsonRC80\Libraries文件夹中的库以列表显示。

4. 在显示的[启用库]对话框中，输入激活密钥或通过[加载]按钮加载csv文件。如果激活密钥正确，将显示使用激活密钥启用的库。

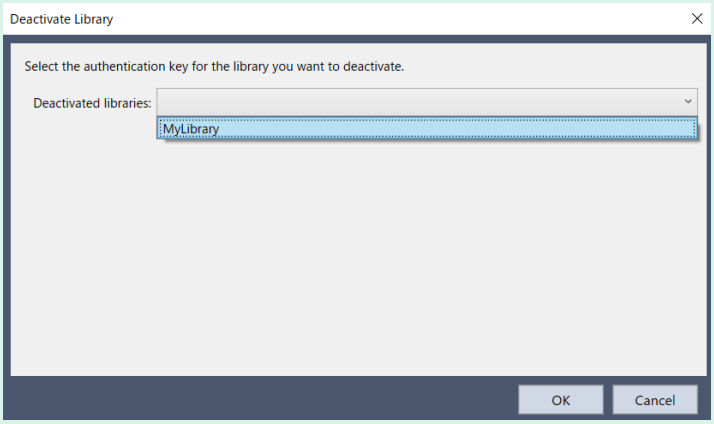


5. 单击[OK]按钮，然后确认目标库已启用。



要点

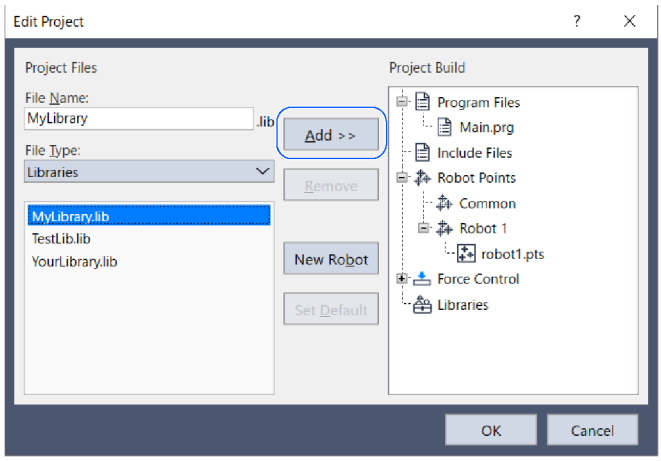
要停用时，单击[停用]按钮后，从[停用库]对话框的下拉列表中选择要停用的库，然后单击[OK]按钮。



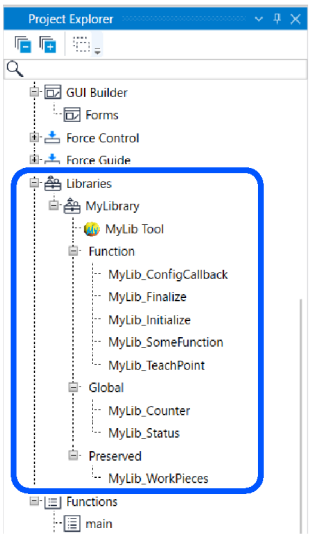
5.3 将库注册到项目

介绍将库注册到项目的方法。

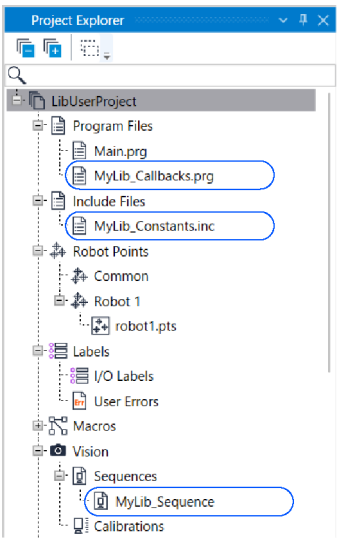
1. 创建要使用库的项目。
2. 选择[项目]-[编辑项目]，或者右击项目资源管理器的库后选择添加库，打开项目编辑对话框。



3. 在文件类型中选择库文件。
4. 显示C:\EpsonRC80\Libraries文件夹中所有库的列表。
5. 选择库，然后单击[注册]按钮。
6. 单击[OK]按钮。
7. 项目资源管理器中显示可使用的公开函数和公开变量。



8. 库和库的公开文件添加至项目。公开文件的名称从库定义的前缀开始。有无公开文件因库而异。



9. 要添加多个库时，重复此项操作。
   1个项目最多可添加5个库。

10. 将要执行的库函数写入程序。

11. 请根据需要编辑库的公开文件。

要点

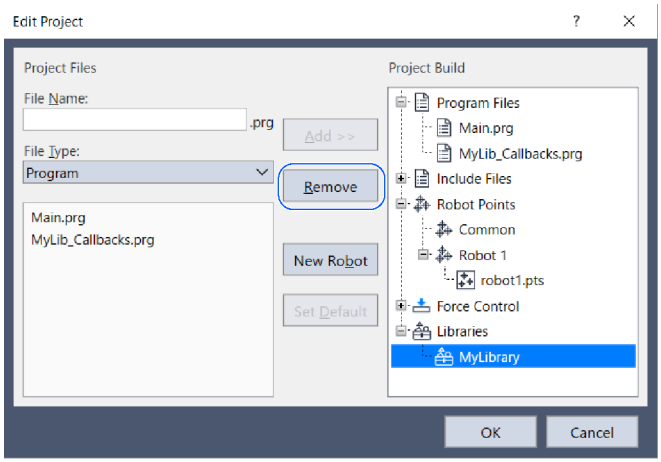
请勿注册拥有相同前缀的多个库。可能会导致函数和变量重复并发生错误。

5.4 将库从项目取消登记

介绍将已登记的库从项目取消的方法。

1. 右击项目资源管理器中要取消登记的库后，如果选择[从项目中排除]，库将取消登记。
   或者选择[项目]-[编辑项目]，打开项目编辑对话框。
   i. 选择项目结构树中要取消登记的库，然后单击[取消]按钮。

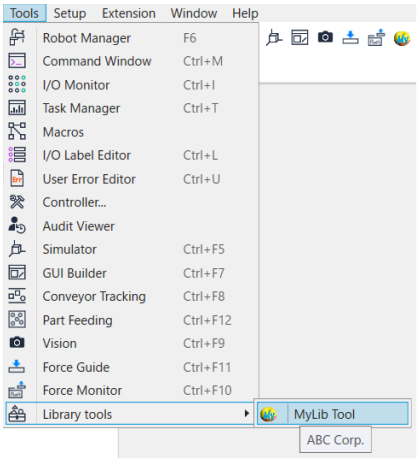
ii. 单击[OK]按钮后，项目编辑对话框将关闭，库被取消登记。



2. 请根据需要将与库一起添加的库公开文件取消登记。不会与库同时取消登记。

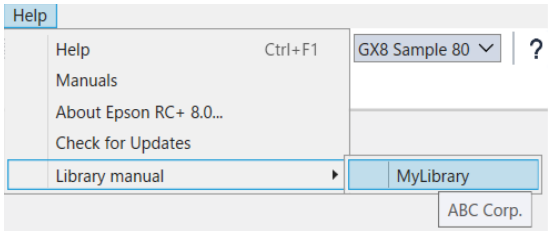
5.5 使用库工具

库可能会支持通过GUI设置或执行。在这种情况下，[工具]-[库工具]下会添加菜单，请从此处执行。有关库工具的详细信息，请参考各库的手册或咨询各库的开发者。



5.6 显示库的手册

如果库中提供了手册，可从[帮助]-[库手册]浏览。



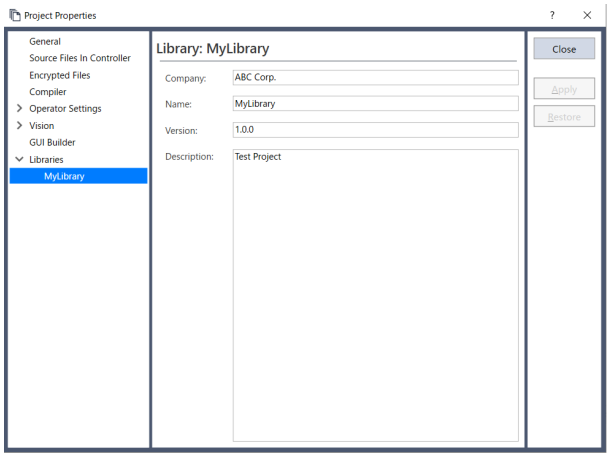
5.7 设置库的属性

可查看项目中所登记的库的属性。

1. 从[项目]-[属性]显示项目属性对话框。

2. 库已登记时，库会添加至树视图，将此展开后，显示已登记的库。

3. 选择各库名后，将显示库的创建者、库名、版本和说明。

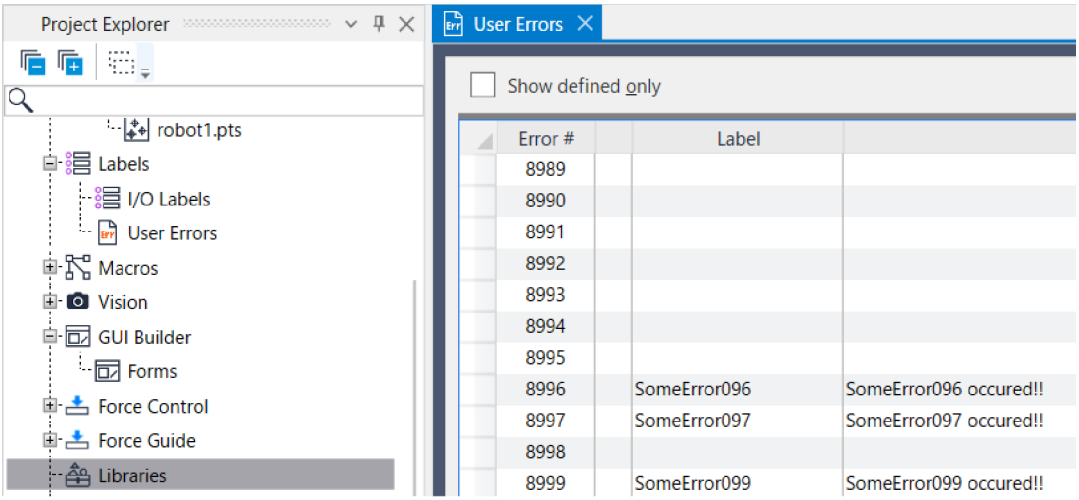


5.8 关于库的用户错误定义

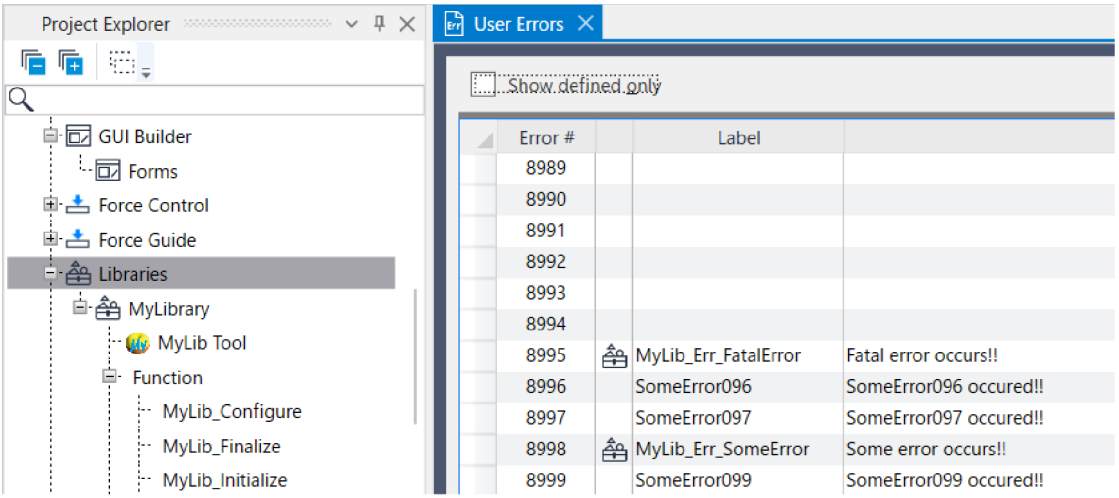
已登记的库的用户错误从错误编号8999开始，按降序使用未定义的编号。请注意，库的用户错误定义无法更改。

库添加前后的示例：

添加前



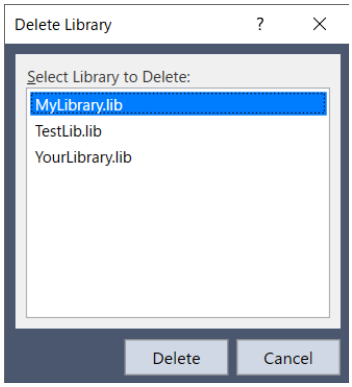
添加后



5.9 删除库

介绍将不再需要的库从Epson RC+删除的方法。

1. 从[扩展]菜单中选择[删除库]。
2. 从所显示对话框的列表中选择要删除的库。
   列表为当前Epson RC+保留的库。
3. 单击[删除]按钮。C:\EpsonRC80\Libraries文件夹中相应的库文件将被删除。



6. SPEL+命令参考

6.1 SPEL+命令列表

从除库以外的SPEL项目、公开程序文件无法使用。
仅在库中可使用的SPEL+命令如下所示。

命令/函数	描述/用途
ArchReserve	动态获取库内可独占使用的拱形编号。
ArmLib	选择动态获取的机械臂编号。
ArmReserve	动态获取库内可独占使用的机械臂编号。
LibGetInfo	获取SPEL+项目中包含的库信息。
LocalReserve	动态获取库内可独占使用的本地编号。
PointReserve	动态获取库内可独占使用的点编号。
SignalReserve	动态获取库内可独占使用的信号编号。
SyncLockReserve	动态获取库内可独占使用的SyncLock编号。
TaskReserve	动态获取库内可独占使用的任务编号。
TimerReserve	动态获取库内可独占使用的定时器编号。
TLReserve	动态获取库内可独占使用的工具编号。
ToolLib	选择动态获取的工具编号。
UploadFileAfterStop	所有任务停止后，将指定的文件从控制器复制到PC的项目文件夹。

6.2 ArchReserve函数

动态获取当前库内可使用的拱形编号。

格式
ArchReserve

参数
无

返回值
预约好的Arch编号

描述
通过预约确保拱形编号可使用，而不被其他库使用。
在7至13中，按从大到小的顺序，预约还未预约的编号。
定义至已预约的拱形编号的信息为临时值，在所有任务终止时预约取消，值也消失。通过在ArchClr指定拱形编号也可取消预约。

另见
Arch, ArchClr

ArchReserve函数使用示例

```
Integer i

i = ArchReserve

Arch i,20,20
Jump3 P2, P3-TLZ(100), P3 C(i) '使用已预约的Arch编号执行动作。

ArchClr i
```

6.3 ArmLib函数

选择动态获取的机械臂编号。

格式

ArmLib 机械臂编号

参数

机械臂编号 以整数值或表达式指定。有效值范围为16至31。可选择此库已预约的机械臂。

描述

在执行机器人命令的机械臂中指定已预约的机械臂。
可选择调用此命令的库已预约的机械臂编号。
要显示选中的机械臂编号时，请执行Arm。
通过所选机械臂编号的动作也与Arm类似。
调用此命令的用户函数结束后，将返回执行函数前的机械臂编号。

另见

Arm, ArmClr, ArmDef, ArmSet, ArmReserve函数

ArmLib函数使用示例

```
Integer i

i = ArmReserve

ArmSet i, X, Y, Z, U
ArmLib i
Move P1     '使用已预约的机械臂编号执行动作，而非当前选中的编号。
```

6.4 ArmReserve函数

动态获取当前库内可使用的机械臂编号。

格式
ArmReserve

参数
无

返回值
预约好的机械臂编号

描述
通过预约确保可使用，而不被其他库使用。在16至31中，按从大到小的顺序，预约还未预约的机械臂编号。
定义至已预约的机械臂编号的信息为临时值，在所有任务终止时，值也将随预约一起消失。或者可通过在ArmClr指定机械臂编号以取消预约。

另见
Arm, ArmLib函数, ArmClr, ArmDef, ArmSet

ArmReserve函数使用示例

```
Integer i

i = ArmReserve

ArmSet i, X, Y, Z, U
ArmLib i
Move P1 '使用已预约的机械臂编号执行动作。
ArmClr i
```

6.5 LibGetInfo

获取SPEL+项目中包含的库信息。

格式

LibGetInfo 库名, ByRef 库ID, ByRef 版本, ByRef 注释, ByRef 库路径

参数

- 库名：使用字符串指定库的名称（最多32个字符）。
- 库ID：指定获得库ID的字符串变量。库ID是每次创建库时会改变的值。
- 版本：指定获得相应库版本的字符串变量。
- 注释：指定获得相应库注释的字符串变量。
- 库路径：指定获得库保存路径的字符串变量。用“[数据文件夹]\Libraries[库名]”表示。

另见

GetProjectInfo

LibGetInfo使用示例

```
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函数

动态获取当前库内可使用的本地编号。

格式
LocalReserve

参数
无

返回值
预约好的本地编号

描述
通过预约确保可使用，而不被其他库使用。在16至31中，按从大到小的顺序，预约还未预约的本地编号。
定义至已预约的本地编号的信息为临时值，在所有任务终止时，值将随预约一起消失。或者可通过在LocalClr指定本地编号以取消预约。
已预约的本地编号中定义的点位作为本地编号0保存。

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

LocalReserve函数使用示例

```
Integer i

i = LocalReserve

Local i, X, Y, Z, U
P1 = Here /(i)
Move P1 '使用指定的本地编号执行动作。
```

6.7 PointReserve函数

动态获取当前库内可使用的点编号。

格式
PointReserve

参数
无

返回值
预约好的点编号

描述
通过预约确保可使用，而不被其他库使用。
定义至已预约的点编号的信息为临时值，在所有任务终止时，值也将随预约一起消失。通过在PDel指定点编号也可取消预约。

另见
P#, PDel

PointReserve函数使用示例

```
Integer i

i = PointReserve

P(i) = Here
Move P(i) +Z50 '使用指定的点编号执行动作。
PDel 1
```

6.8 SignalReserve函数

预约可指定至SigWait的信号编号。

格式
SignalReserve

参数
无

返回值
预约好的信号编号

描述
通过预约确保可使用，而不被其他库使用。为临时值，在所有任务终止时，值将随预约一起消失。保留预约专用的信号编号，因此可降低使用库的用户所使用的信号编号被改写的风险。

另见
WaitSig

SignalReserve函数使用示例

```
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函数

预约指定至SyncLock的信号编号。

格式
SyncLockReserve

参数
无

返回值
预约好的信号编号

描述
通过指定至SyncLock和SyncUnLock，可独占进行文件访问等。
通过预约信号编号确保可使用，而不被其他库使用。为临时值，在所有任务终止时，值将随预约一起消失。保留预约专用的信号编号，因此可降低使用库的用户所使用的信号编号被改写的风险。

另见
SyncLock

SyncLockReserve函数使用示例

```
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函数

预约执行任务时可指定的任务编号。

格式
TaskReserve

参数
无

返回值
预约好的任务编号

描述
通过预约确保可使用，而不被其他库使用。为临时值，在所有任务终止时，值将随预约一起消失。
按从大到小的顺序，预约还未预约的任务编号。

另见
WaitSig

TaskReserve函数使用示例

```
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函数

预约指定至Timer的定时器编号。

格式
TimerReserve

参数
无

返回值
预约好的定时器编号

描述
通过指定至Tmr函数，可独占进行循环时间的测量。
通过预约定时器编号确保可使用，而不被其他库使用。为临时值，在所有任务终止时，值将随预约一起消失。保留预约专用的定时器编号，因此可降低使用库的用户所使用的定时器编号被改写的风险。

另见
Tmr, TmReset

TimerReserve函数使用示例

```
Function N4_1_Call_Hidden

  t1 = TimerReserve

  TmReset t1              '重置定时器0

  Xqt ...

  Print Tmr(t1) / 10      '计算循环时间并显示
Fend
```

6.12 TLReserve函数

动态获取当前库内可使用的工具编号。

格式
TLReserve

参数
无

返回值
预约好的工具编号

描述
通过预约确保可使用，而不被其他库使用。为临时值，在所有任务终止时，值将随预约一起消失。或者可通过将工具编号指定至TLClr以取消预约。
在16至31中，按从大到小的顺序，预约还未预约的工具编号。

另见
TLClr, TLDef, TLSet, Tool, ToolLib函数

TLReserve函数使用示例

```
Integer i

i = TLReserve

TLSet i, X, Y, Z, U
ToolLib i
Move P1        '使用指定的工具编号执行动作。
TLClr i
```

6.13 ToolLib函数

选择动态获取的工具编号。

格式
ToolLib 工具编号

参数
工具编号
以整数值或表达式指定。有效值范围为16至31。可选择此库已预约的工具。

描述
在执行机器人命令的工具中指定已预约的工具。
可选择调用此命令的库的已预约工具编号。
要显示选中的工具编号时，请执行Tool。
通过所选工具编号的动作也与Tool类似。
调用此命令的用户函数结束后，将返回执行函数前的工具编号。

另见
TLClr, TLDef, TLSet, TLReserve函数, Tool

ToolLib函数使用示例

```
Integer i

i = TLReserve

TLSet i, X, Y, Z, U
ToolLib i
Move P1              '使用已预约的工具编号执行动作。
```

6.14 UploadFileAfterStop函数

所有任务停止后，将指定的文件从控制器复制到PC的项目文件夹。

格式

UploadFileAfterStop 文件名

参数

- 文件名：指定项目内包含的文件名。

另见

GetProjectInfo

UploadFileAfterStop使用示例

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