

埃夫特智能装备股份有限公司

C30 快速使用手册

版本号：v0.1

软件二组

2019-6-26

修订历史

[illegible]

目录

第一章 Rpl 代码编写.....	3
1.1 登录.....	3
1.2 加载 rpl 程序文件.....	3
1.2.1 新建 rpl 程序文件.....	3
1.2.2 加载已有 rpl 程序文件.....	4
1.3 代码编辑.....	5
1.3.1 注意事项.....	5
1.3.2 添加指令.....	5
1.3.3 修改指令.....	9
1.3.4 删除指令.....	11
1.3.5 注销指令.....	11
1.3.6 添加注释.....	12
1.4 变量相关操作.....	13
1.4.1 添加变量.....	13
1.4.2 修改变量.....	14
1.5 子程序相关操作.....	14
1.5.1 新建子程序.....	14
1.5.2 程序代码切换.....	15
1.5.3 删除子程序.....	16
第二章 码垛功能使用.....	17
2.1 垛盘信息设置.....	17
2.2 rpl 程序中调用码垛功能.....	19
第三章 固定视觉.....	20
3.1 固定视觉相机设置.....	20
3.2 固定视觉手眼标定.....	21
3.3 固定视觉 rpl 程序编程.....	23
3.3.1 指令说明.....	23
3.3.2 代码示例.....	23

第一章 Rpl 代码编写

1.1 登录

进行 rpl 编程需要管理员权限，因此需要登陆后才能进行相关操作，点击快捷方式栏“登录”（图 1 中标记 1），进入登录界面（如图 1），点击密码输入框（图 1 中标记 2），输入 666666，点击“登录”按钮（图 1 中标记 3）进行登录。



图 1

1.2 加载 rpl 程序文件

加载 rpl 程序文件的方式一共有两种：

- （1）新建 rpl 程序文件
- （2）加载已有 rpl 程序文件

注意：如果机器人当前已经加载程序，记载其他程序前，请确认当前加载程序已经被终止，或已保存并退出编辑状态。

1.2.1 新建 rpl 程序文件

快捷方式栏点击“文件”快捷按钮（图 2 中标记 1），系统会进入文件管理页面。点击左下角“新建”快捷按钮（图 2 中标记 2），弹出新建列表（列表中有两个选项“文件”、“文件夹”），选择“文件”选项（图 2 中标记 3）进入文件命名对话框（如图 4）。

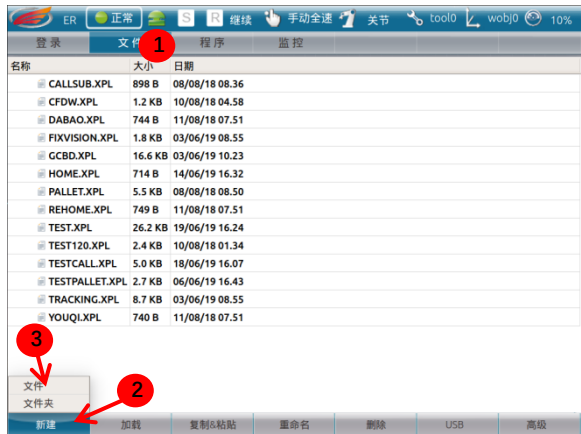


图 3

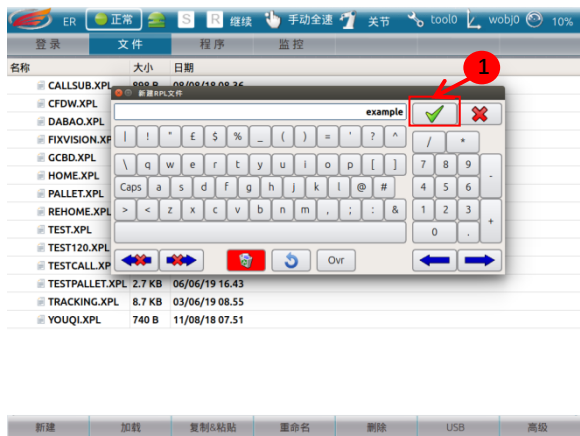


图 4

文件名输入完成后，点击虚拟键盘中“确认”按钮（即图 4 中标记 1）完成文件名输入，系统将自动创建对应文件，并自动跳转到 `rpl` 程序编辑界面（即“程序”页面）。

1.2.2 加载已有 `rpl` 程序文件

点击选中要加载的文件（如图 5 中标记 1），点击下方操作栏中的“加载”快捷按钮（如图 5 中标记 2），系统将加载该程序，并自动跳转到“程序”页面。

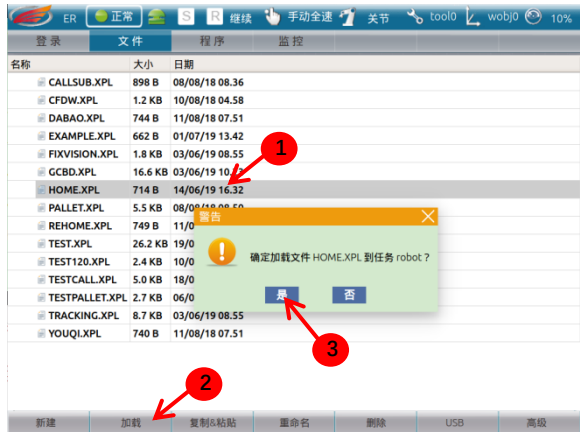


图 5

1.3 代码编辑

1.3.1 注意事项

如要进行代码编辑，请注意以下几点：

1. 机器人运动模式必须处于**手动模式**，自动模式下不允许编辑代码；
2. 完成编辑后，在**退出编辑页面**前，请先点击右下角**“保存”**按钮，否则，编辑的内容不会被保存进相应文件，机器人掉电重启之后会丢失相应内容。
3. 编辑之前，**请确保程序已被终止**，否则编辑的内容将无法被保存。

1.3.2 添加指令

假如，为 modbus 通讯总线中地址为 40071 内存值自加 1。（提前通过查询“监控”页面中“Modbus”栏中“Output”条目，得知 40071 地址存储类型为 int，对应示教变量为 fidbus.mobtxint[0]。）

1. 新建行与进入代码编辑页

新建文件的程序编辑初始页面如图 6 所示，点击图 6 中左下角的“编辑”按钮（图 6 中标记 1）进入编辑模式。

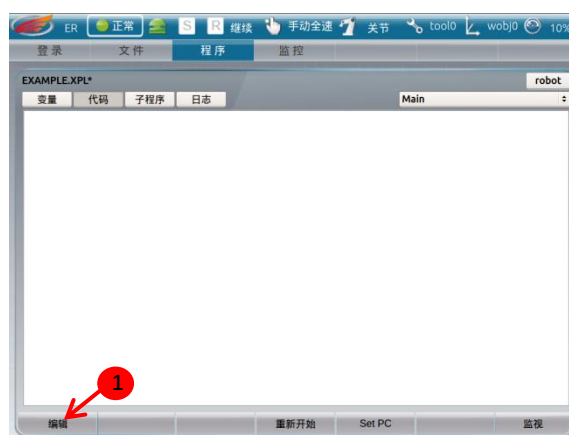


图 6

图 7 为进入编辑模式下的“代码”页面。

- 1) **在程序最后一行添加代码**：点击选中程序末尾行的“...”行（如图 7 中标记 1），如果是添加一般运动指令，可直接通过单击图 7 中标记 2 所示的快捷运动指令栏中的按钮来添加运动指令（点的位置默认为当前机器人的位置），也可通过点击标签栏“编辑”标签（图 7 中标记 3）进入指令选择页面。

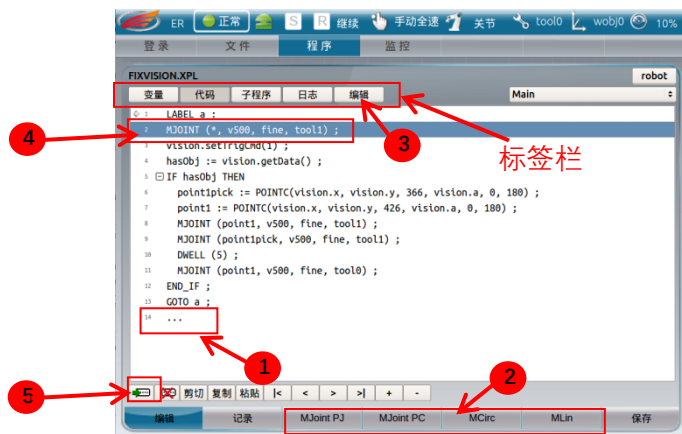


图 7

2) 在程序中的**某一行上方添加一行代码**：选中该行（图 7 中标记 4）点击新建行按钮“”，则会在该行上方新建一行“...”（与程序末尾行相同），选中新出现的行，点击标签栏“编辑”标签（图 7 中标记 3）进入指令选择页面。

2. 指令选择

图 8 为指令选择页面，这里可以选择所有 rpl 程序指令（每条指令的功能请详见《埃夫特机器人 ROBOX 控制器机器人编程语言 RPL》）。

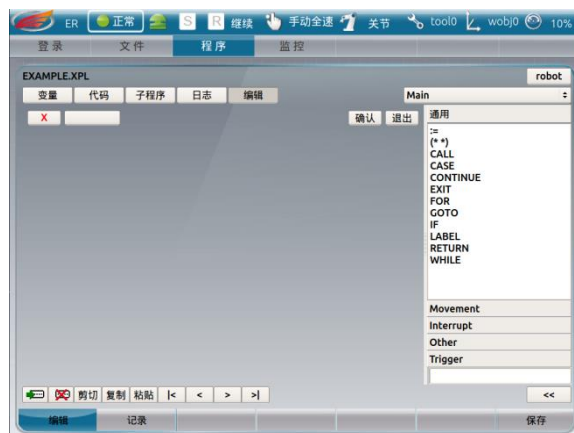


图 8

双击赋值运算符“:=”（图 9 中标记 1 处），或选中赋值运算符“:=”（图 9 中标记 1 处），单击添加按钮“<<”（图 9 中标记 2 处），完成指令选择。

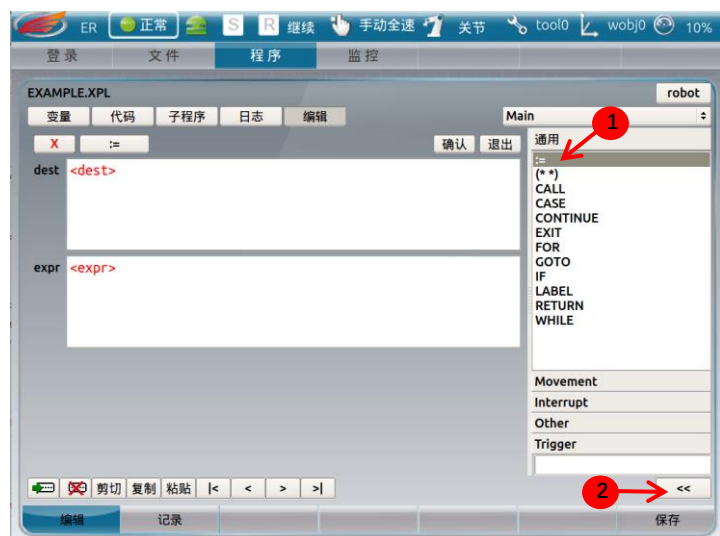


图 9

3. 指令参数填写

1) 单击红色高亮的“<dest>”字样进行被赋值变量选择，如图 10 所示，在右侧弹出候选变量列表中选择添加“fidbus.mobtxint”变量（添加操作与添加指令操作相同），结果如图 11 所示。由于“fidbus.mobtxint”变量为数组变量，因此其后缀为“[???”（图 11 中标记 1），点击右变量列表下面的附加操作按钮中的“值”按钮（图 11 中标记 2），输入“0”，点击确认按钮“√”完成值输入。

说明：对于移动指令，若要将目标点设为机器人当前点，点击点的函数名（图 15 中标记 2 位置，函数名称为 POINTJ、POINTL 等），再点击“记录”按钮（图 15 中标记 4）即可完成。

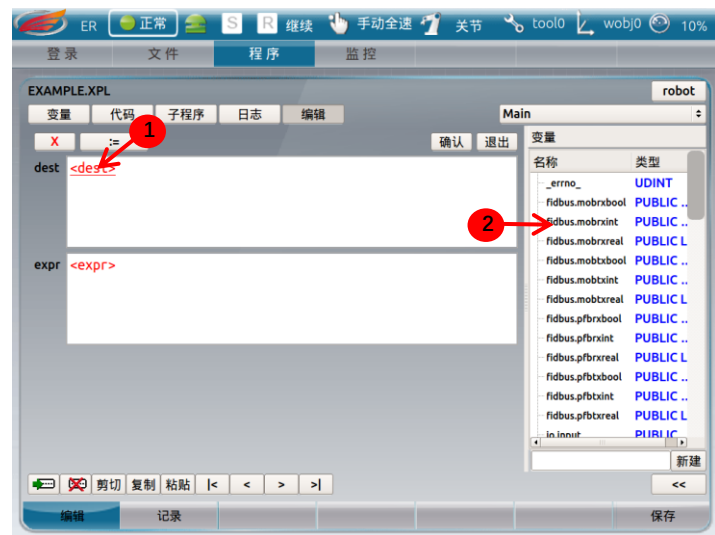


图 10

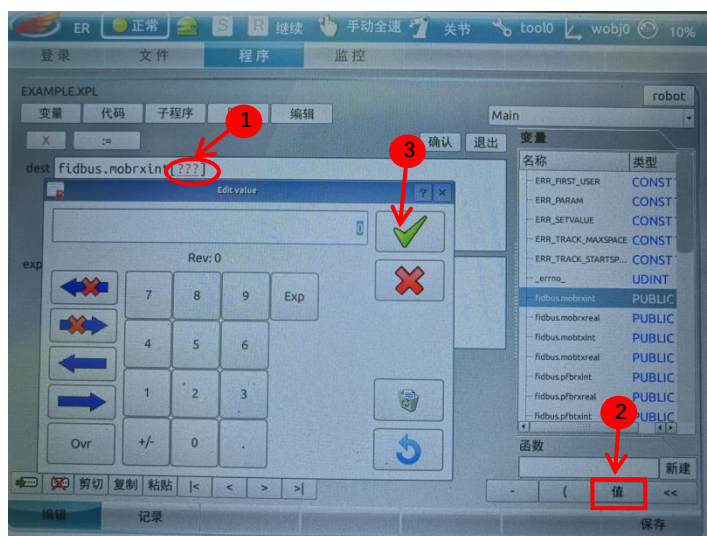


图 11

2)单击红色高亮的“<expr>”字样进行赋值表达式编辑。首先如步骤2将 fidbus.mobtxint[0] 变量，结果如图 12 所示，点击“fidbus.mobtxint[0...]”变量的变量名“fidbus.mobtxint”，“expr”输入栏表达式将变为“fidbus.mobtxint[0...]”结果如图 13 所示。

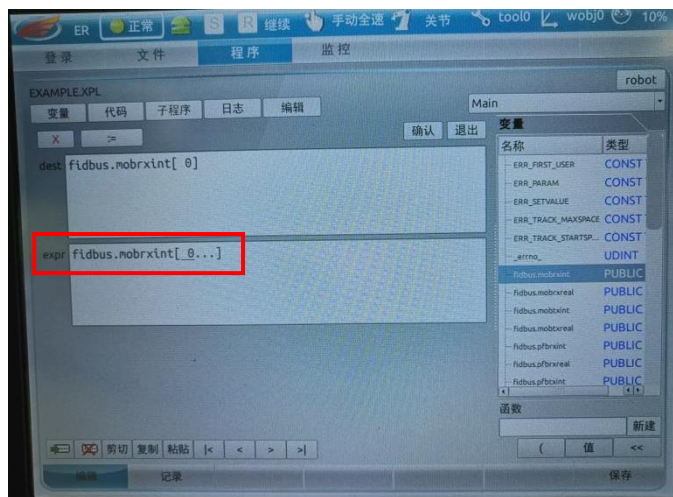


图 12

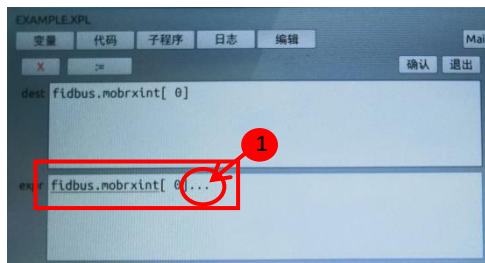


图 13

点击图 13 中标记 1 所示的“...”可进入操作符选择界面。如图 14 所示，右侧栏为操作符列表，这里选择“+”，单击“+”操作符即可，结果如图 15 所示。点击新出现的“!!!”（图 15 中的标记 1）添加值“1”。

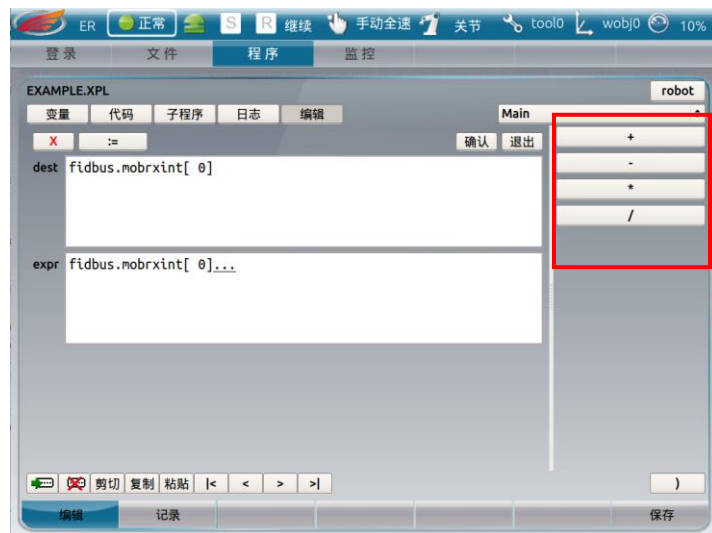


图 14

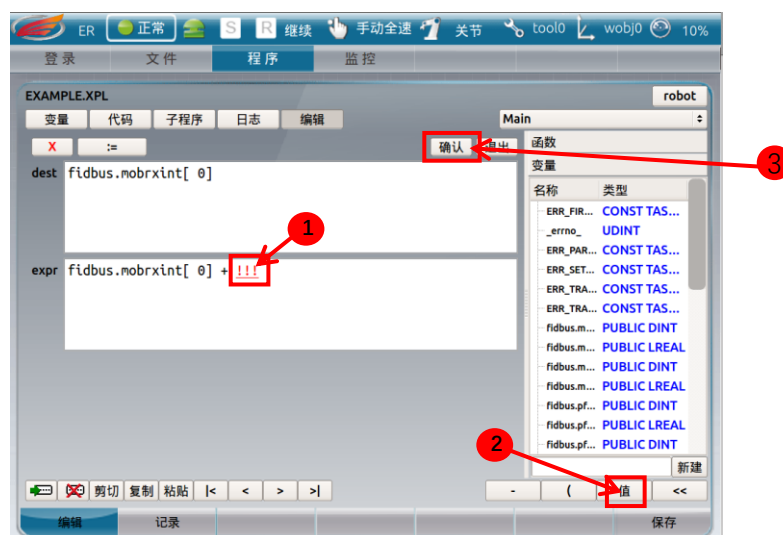


图 15

最后，点击“确认”按钮（图 15 中标记 3），将编辑好的代码添加入程序中！！！！

1.3.3 修改指令

图 16 为一 rpl 程序示例，现由于工程需要，要对第二行的运动指令 MJOINT（）进行修改，操作如下：

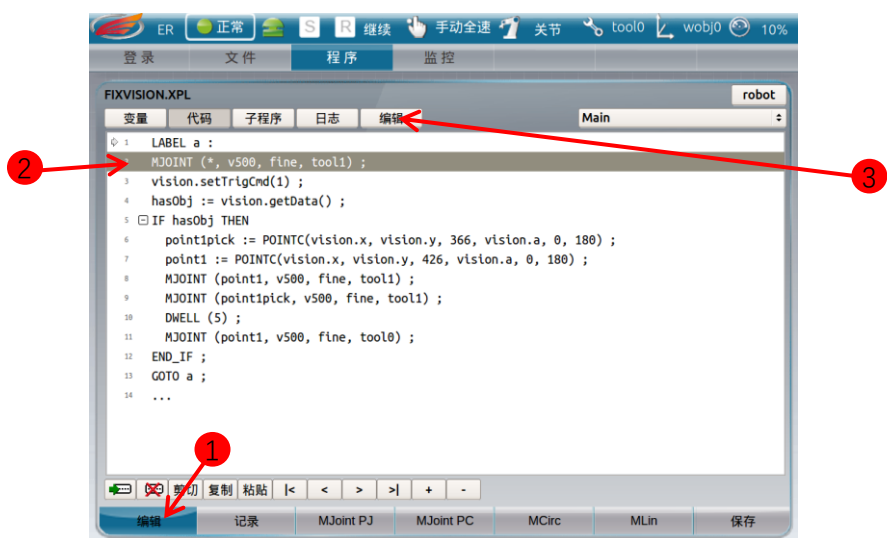


图 14

1. 点击左下角“编辑”按钮，进入编辑模式，点击第二行选中（图 14 中标记 2），点击“编辑”标签（图 14 中标记 3）进入标记页面，如图 15 所示。

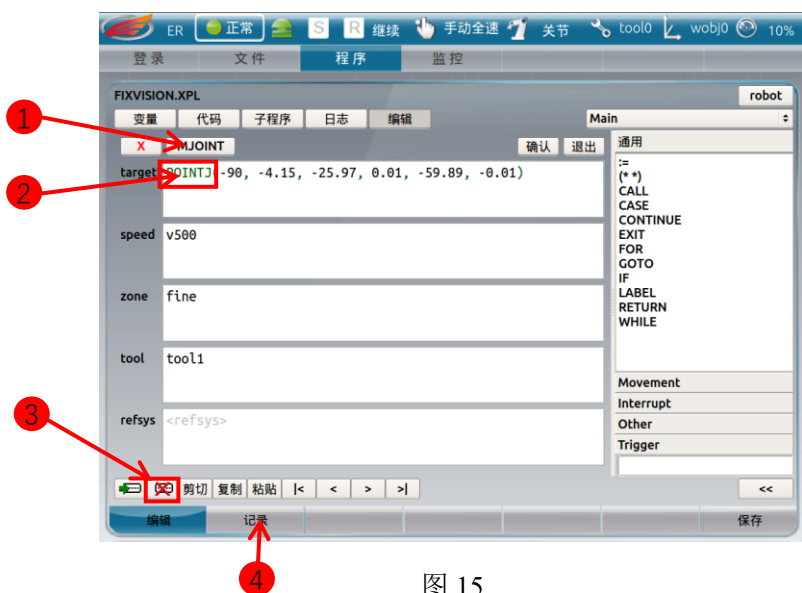


图 15

1) 修改指令

点击指令“MJOINT”（图 15 中标记 1），再点击删除按钮“

2) 修改指令参数

对于指令参数的修改，只需点击相应参数位置中变量、函数的名称或值，右侧候选列表就会相应刷新为对应的候选列表，找到目标后，双击或选中后点击右下角添加按钮“<<”，最后点击“确认”按钮完成修改，即可完成指令修改。

对于移动指令，若要将目标点修改为机器人当前点，点击点的函数名（图 15 中标记 2），再点击“记录”按钮（图 15 中标记 4）即可完成。

1.3.4 删除指令

若想删除 rpl 程序中某行代码，其操作如下：

1. 选中该行代码（如图 16 标记 1）
2. 点击删除按钮 “”（图 16 中标记 2）

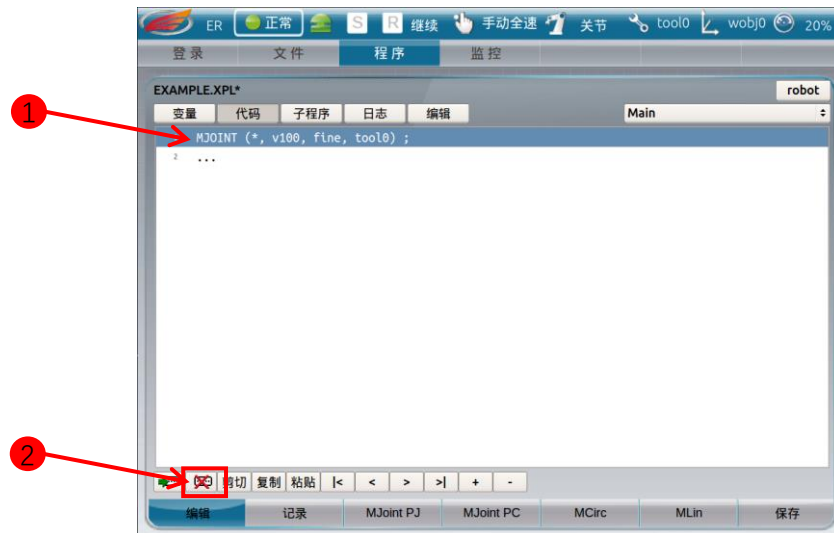


图 16

1.3.5 注销指令

若想注释某行，操作如下：

1. 选中该行（图 17 中标记 1），点击“编辑”标签（图 17 中标记 2），进入编辑页面。



图 17

2. 点击注销按钮 “”（图 18 中标记 1），再点击确认（图 18 中标记 2）即可完成对该行的注销。

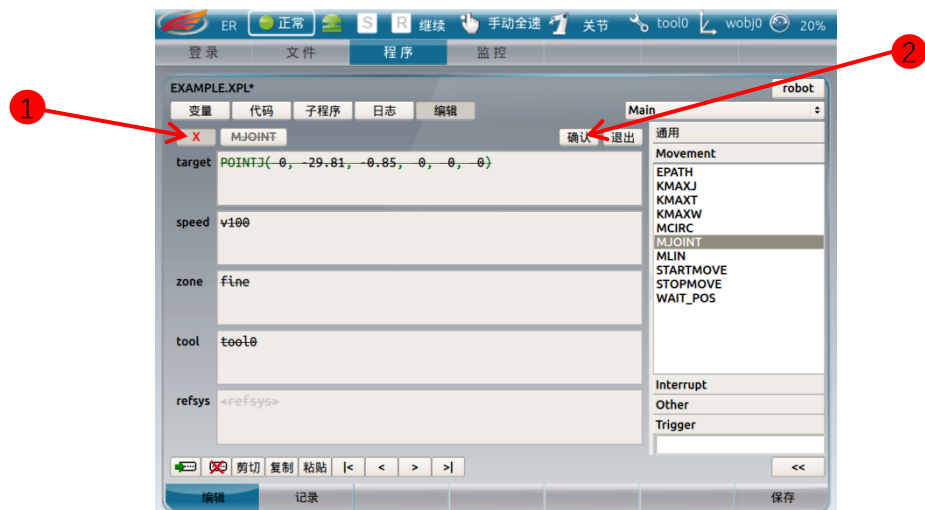


图 18

1.3.6 添加注释

Rpl 程序中可以在代码中添加注释，但是注释将占用一整行，不能和需注释代码处于同一行。添加注释的操作为：

1. 新建行，进入编辑页，
2. 在右侧指令候选栏中，点击“通用”标点栏（图 19 中标记 1），选择注释指令“(**)”（图 19 中标记 2）并添加，

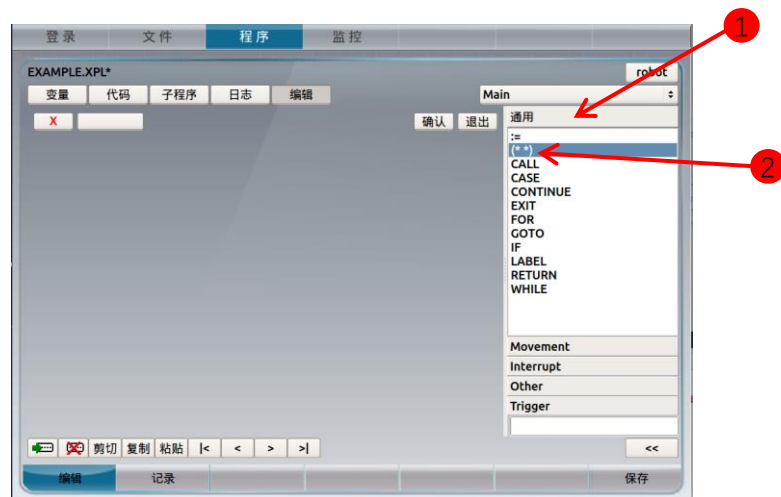


图 19

3. 点击“text”栏中的“...”（图 20 中标记 1），再点击右下角“值”按钮（图 20 中标记 2）进行注释内容填写，填写完成后点击“确认”按钮（图 20 中标记 3），将注释添加进程序。

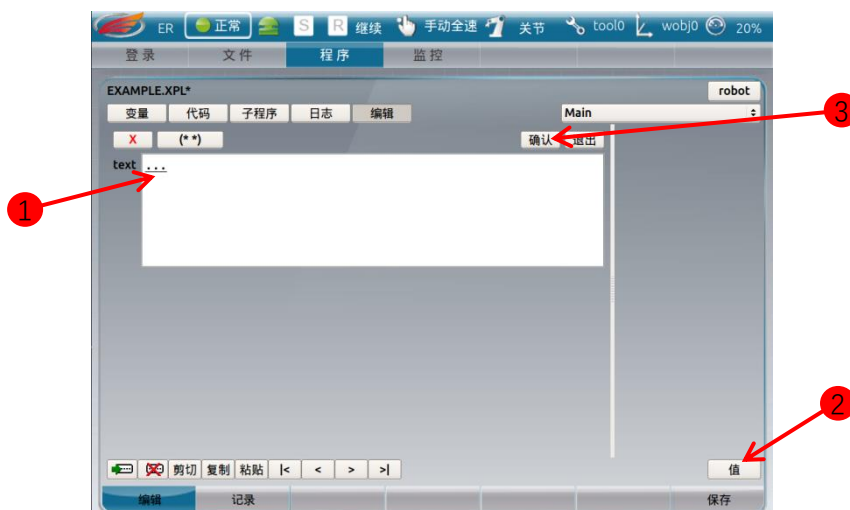


图 20

1.4 变量相关操作

在 rpl 编程中有一些预定变量，这些变量主要为用户提供某些功能，用户只能查看、引用、赋值，不能修改这些变量的名称，这些变量为图 21 中标记 1 标注的变量“_errno_”及“base_shared”、“fidbus”、“io”等折叠栏中变量（图 21 所示为 tpu1.7.0 中预设变量，后续可能有变）。

主程序 Main()函数中用户可添加的变量类型有：

1. 程序变量：只能在本程序中使用
2. 外部变量：用于和控制器交互使用
3. Module：根据所选作用域的不同，具有不同的生存周期

子程序中用户可添加的变量类型在主程序的基础上增加了：

1. 输入：程序的输入变量
2. 输出：程序的输出变量

1.4.1 添加变量

变量的添加操作：

1. 点击想要添加的变量的类型（如图 21 中添加的为 Module），再点击新建按钮“”（图 21 中标记 2），将弹出变量参数设置对话框（如图 22 所示）；

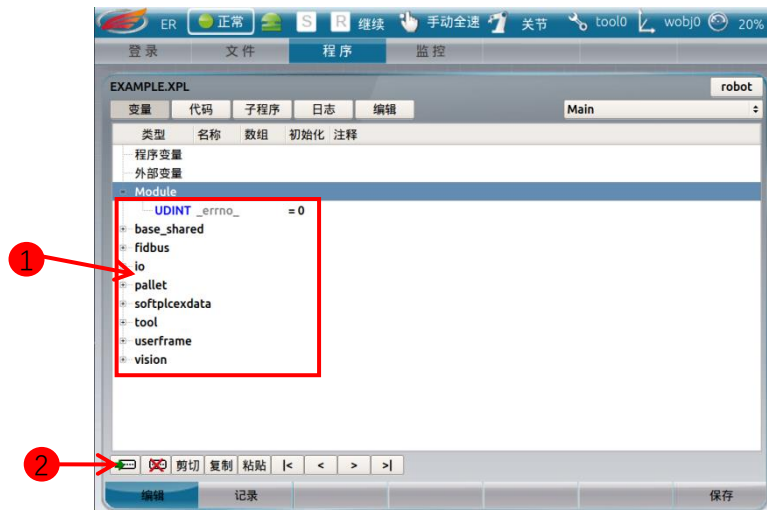


图 21

2.在变量设置页面，可以对变量的属性进行详细的设置。



图 22

Tip: 新建点变量后，选中该点变量，点击“记录”按钮可将机器人当前点赋值给该点。

1.4.2 修改变量

对变量的修改操作比较简单，只需双击对应变量，就会弹出该变量的设置窗口。

1.5 子程序相关操作

1.5.1 新建子程序

Tpu_v1.7.0 及以前的 tpu 版本不支持子程序名的修改。

新建子程序的操作步骤如下：

1. 点击“子程序”标签（图 23 中标记 1），

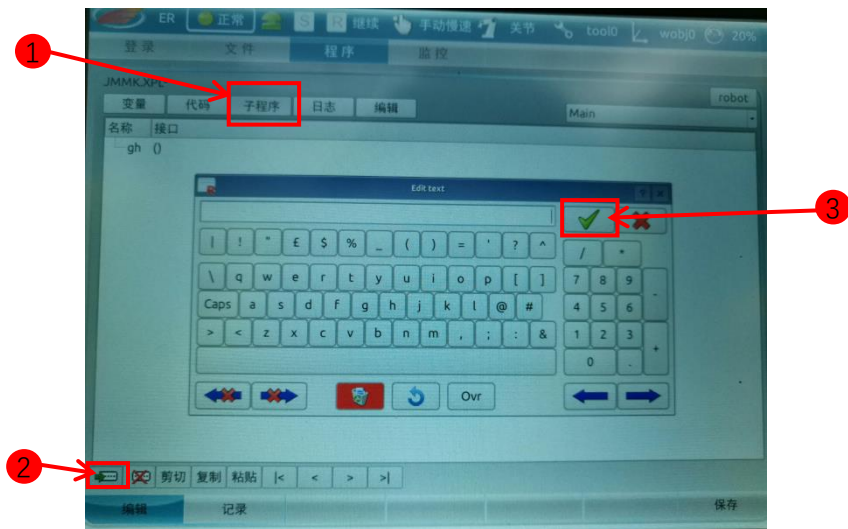
2. 点击新建按钮 “

图 23

1.5.2 程序代码切换

在编程中，如果有子程序存在时，如要对某个非当前编辑的程序的代码进行编辑，需要进行代码的切换操作，操作步骤如下：

1. 点击标签栏右侧的程序名显示栏（图 24 中标记 1），会弹出程序列表（如图 25 所示）



图 24

2. 在程序列表中，点击相应的程序，系统将会自动切换为该程序的代码、变量。

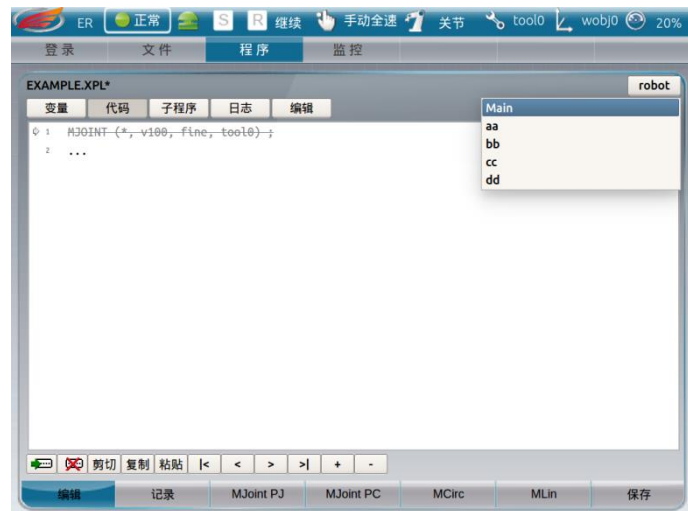


图 25

1.5.3 删除子程序

删除子程序的操作如下：

1. 点击要删除的子程序（图 26 中标记 1），
2. 点击删除按钮 “”（图 26 中标记 2），立即就可删除该子程序。

注意：删除子程序名称后，子程序的代码也将一并删除！

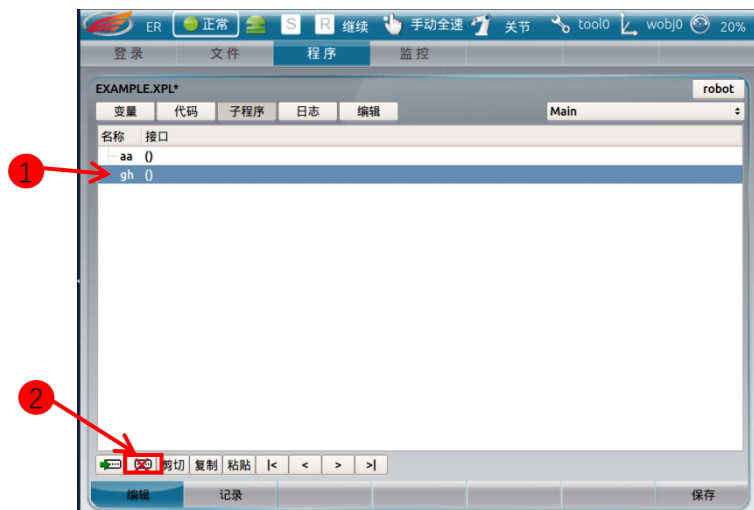


图 26

第二章 码垛功能使用

2.1 垛盘信息设置

- 1.码垛首页点击“编辑”按钮，进入编辑第一页；
- 2.编辑第一页“1.码垛盘坐标系设置”中，点击下拉框，选择码垛盘坐标系，并点击“激活”。在“2.基础设置”中，输入码垛盘中工件的排列矩阵的相关信息，完毕后点击“下一步”。

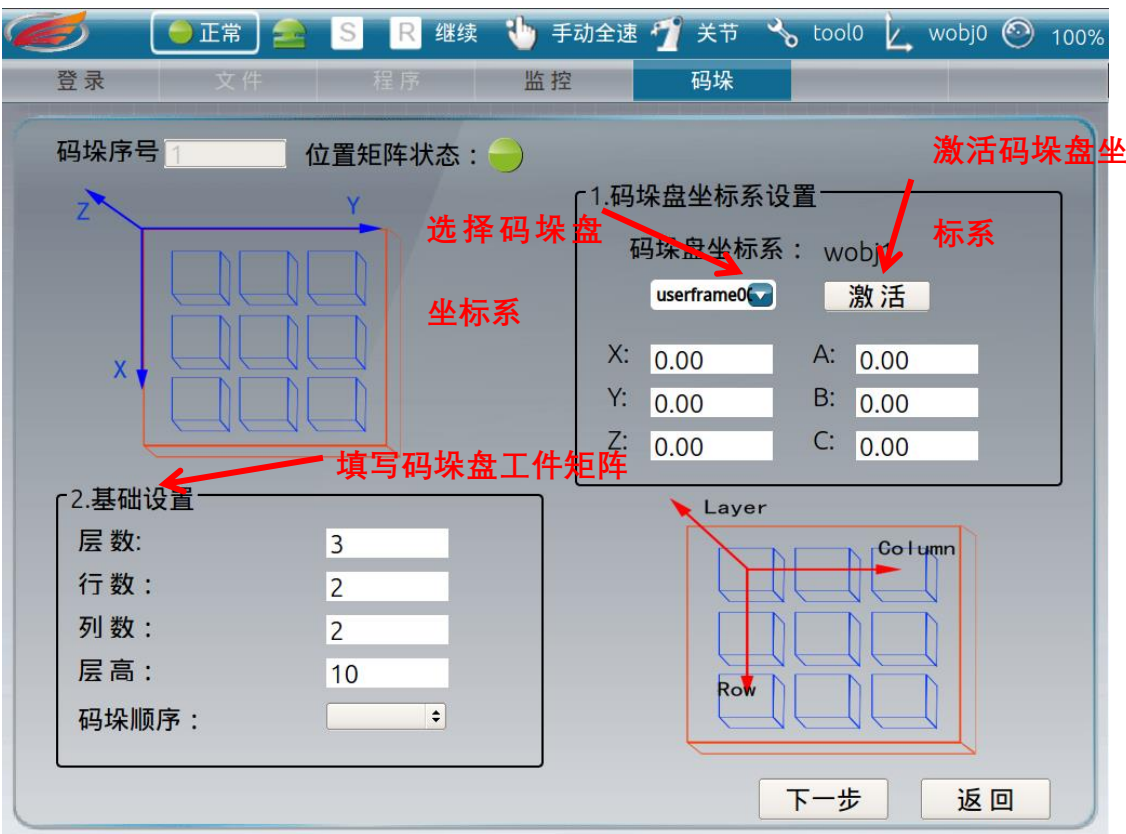


图 27

- 3. 先将机器人移动至码垛盘中第一层第一个工件的码垛进入点位置，点击“3.码垛进入点设置”中“记录”按钮；根据路径需要设置“4.过渡点设置”中的生效的点，这些点的位置是相对于工件点的偏移来确定的；设置完毕后点击“下一步”

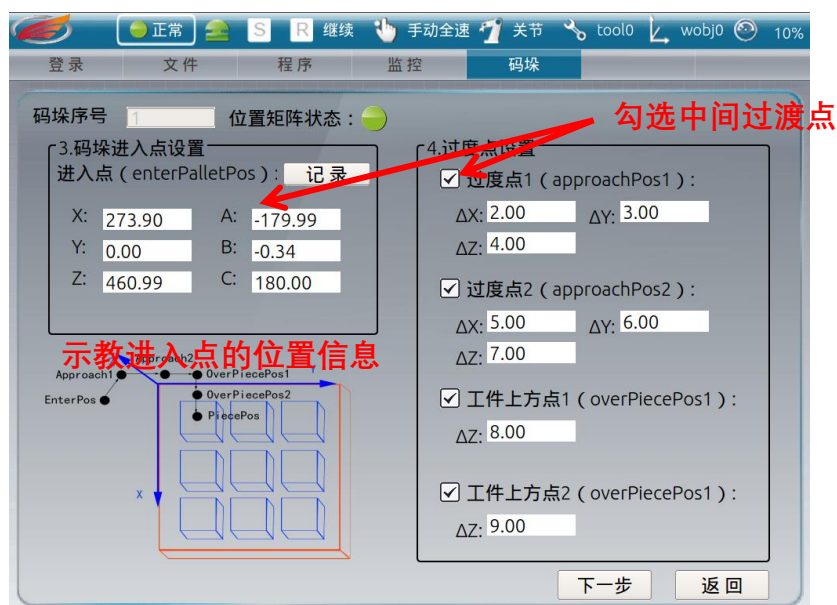


图 28

4.示教点 P1 为码垛第一层的第一工件放置点，位于工件矩阵第一层的第一行第一列，将机器人移动至该工件点后，点击“记录”进行示教；

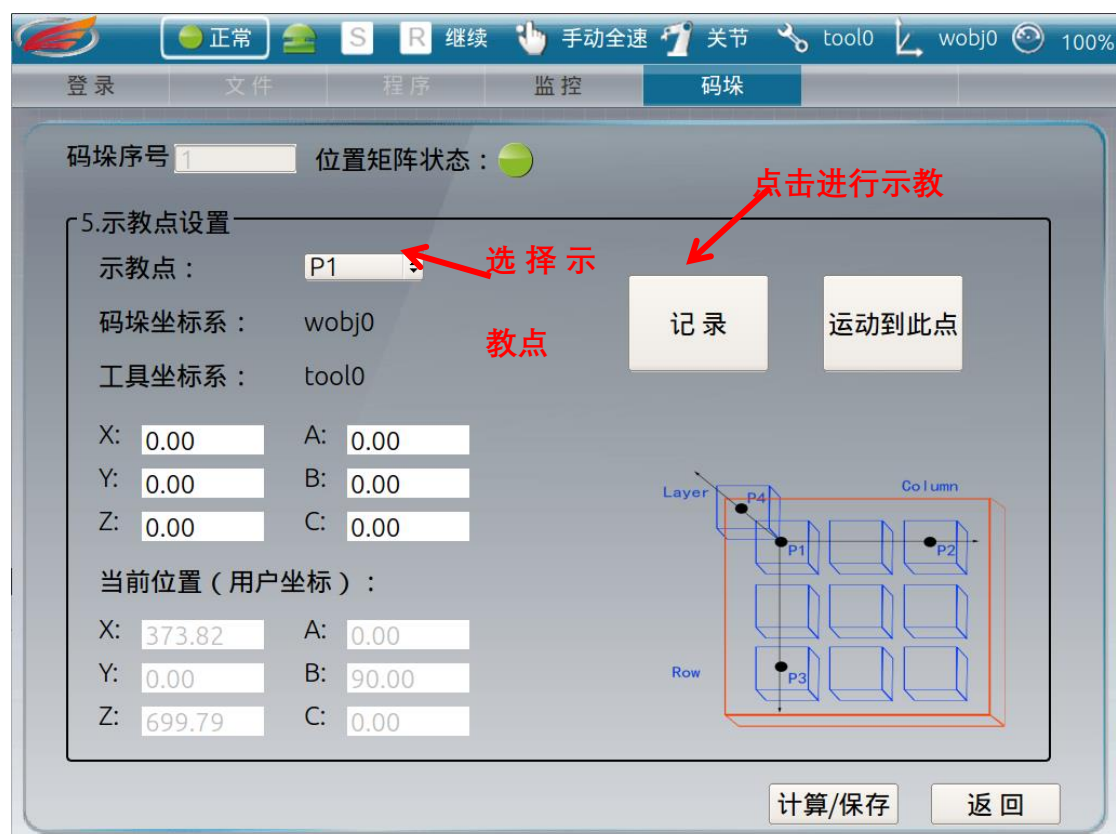


图 29

5.选取下一示教点 P2，P2 为第一行最后一个工件放置点，将机器人移动至该工件点后，点击“记录”进行示教；

- 6.选取下一示教点 P3，P3 为第一列最后一个工件放置点，将机器人移动至该工件点后，点击“记录”进行示教；
- 7.选取下一示教点 P4，P4 为第二层中位于 P1 正上方的工件放置点，将机器人移动至该工件点后，点击“记录”进行示教；
- 8.点击“计算/保存”按钮完成码垛盘信息设置。

2.2 rpl 程序中调用码垛功能

1.常用码垛指令介绍

1) 码垛函数

`pallet.update(int palletID, int palletMethod, int piecesID)`

参数说明：

- `floorID`: 码垛盘编号，
- `palletMethod`: 0 为码垛，1 为取垛
- `piecesID`: 工件的 ID，具体为（列号 - 1）+ （行号 - 1）×每行列数。

功能：用于更新当前工件点的位置信息。

3) 常用变量

变量名	说明
<code>pallet.enterPalletPos</code>	码垛的进入点
<code>pallet.approachPos1</code>	当前工件点的接近点 1
<code>pallet.approachPos2</code>	当前工件点的接近点 2
<code>pallet.overPiecePos1</code>	当前工件点的上方点 1
<code>pallet.overPiecePos2</code>	当前工件点的上方点 2
<code>pallet.piecePos</code>	当前工件点
<code>pallet.maxPieces</code>	当前垛盘中的工件点的数量
<code>pallet.maxPallets</code>	有效垛盘的数量

2.程序示例

```

1  pieceid := 1 ;
2  LABEL aaa :
3  pallet.update(1, 0, pieceid) ;
4  MLIN (pallet.enterPalletPos, v500, z0, tool0, wobj1) ;
5  MLIN (pallet.approachPos1, v500, z0, tool0, wobj1) ;
6  MLIN (pallet.piecePos, v500, z0, tool0, wobj1) ;
7  DWELL (5) ;
8  MLIN (pallet.approachPos1, v500, z0, tool0, wobj1) ;
9  MLIN (pallet.enterPalletPos, v500, z0, tool0, wobj1) ;
10 pieceid := pieceid + 1 ;
11 IF pieceid <= pallet.maxPieces THEN
12     GOTO aaa ;
13 END_IF ;

```

1. 工件编号 pieceid 为 1
2. 设置标签 aaa
3. 更新 1 号垛盘，码垛操作时工件编号为 pieceid 的相关位置信息（包括接近点、上方点、工件位置）
4. 移动到码垛进入点
5. 移动到当前码垛工件点的接近点 1
6. 移动到当前码垛工件点
7. 等待 5 秒
8. 移动到当前码垛工件点的接近点 1
9. 移动到码垛进入点
10. 工件编号 pieceid 加 1
11. 开始判断，判断工件编号是否已经超出本码垛堆的最大序号
12. 判断成立，转到标签 aaa
13. 结束判断

第三章 固定视觉

3.1 固定视觉相机设置

点击  进入桌面，点击桌面上的 ，进入固定视觉 APP 主界面，如图 30 所示：



图 30

- 1.首先打开固定视觉开关（图 30 中标记 1 为绿色代表开关已打开），
- 2.确认相机的连接状态（图 30 中标记 2），绿色对号“”代表已连接。
- 3.点击“设置”按钮（图 30 中标记 3），进入相机设置页面（如图 31 所示），对相机相关参数进行设置、修改、确认。



图 31

说明：1.当相机品牌选择康耐视相机时，需要输入相机账号和密码；2.标定方式：有相机标定和手眼标定两种方式；相机标定是在相机上完成的

3.2 固定视觉手眼标定

如果相机标定方式选择的是“手眼标定”，则需要通过示教器进行标定，其步骤如下：

1. 进入固定视觉 APP 主界面，进行相机相关设置后（详情见上节），点击“标定”按钮（图 32 中标记 1），进入手眼标定界面，



图 32

2. 点击“标定点个数”（图 33 中标记 1），选择需要标定的点数
3. 第一个点的标定，将标定件放置在相机视野区内，等待相机识别出工件，点击“相机”中“示教”按钮（如图 33 中标记 2 所示），记录当前相机坐标，
4. 将机器人移动至标定件的正上方，点击“机器人”中的“示教”按钮（图 33 中标记 3），记录当前机器人坐标，
5. 点击“下一点”按钮“>>”（图 33 中标记 4），切换到下一点，
6. 重复步骤 3、4、5 直到完成所有点的示教，
7. 点击“计算”按钮（图 33 中标记 5）进行标定计算，
8. 点击“返回”，返回主界面。



图 33

3.3 固定视觉 rpl 程序编程

3.3.1 指令说明

指令	名称	功能
Vision._Init_()	视觉功能初始化命令	该命令视觉相关功能的初始化，在正常使用中不需要调用，该函数会在程序开始自动运行
Vision.getData()	相机拍照命令	调用该命令，触发相机拍照动作并返回相应数据
Vision.setTrigCmd (int p)	设置相机触发指令	相机在指令触发的模式下，该命令可设置相机触发的指令
变量	名称	
Vision.x real	工件位置：X 方向坐标	
Vision.y real	工件位置：Y 方向坐标	
Vision.z real	工件位置：Z 方向坐标	
Vision.a real	工件姿态：绕 Z 轴角度	
Vision.b real	工件姿态：绕 Y 轴角度	
Vision.c real	工件姿态：绕 X 轴角度	
Vision.attr int	工件属性	
Vision.id int	工件 ID	

3.3.2 代码示例

```
1 LABEL a :
2 MJOINT (*, v500, fine, tool1) ;
3 vision.setTrigCmd(1) ;
4 hasObj := vision.getData() ;
5 IF hasObj THEN
6     point1pick := POINTC(vision.x, vision.y, vision.z + 5, -180 - vision.a, 180, 0) ;
7     point1 := POINTC(vision.x, vision.y, vision.z + 50, -180 - vision.a, 180, 0) ;
8     MJOINT (point1, v500, fine, tool1) ;
9     MJOINT (point1pick, v500, fine, tool1) ;
10    DWELL (5) ;
11    MJOINT (point1, v500, fine, tool1) ;
12 END_IF ;
13 GOTO a ;
```

Line1:循环开始；
Line2:机器人运动到工位 1；
Line3:设置相机触发指令为 1；（当相机设置为 IO 触发时，不需要该步骤）
Line4:触发相机拍照，当成功获取数据时变量 hasObj=true；
Line5:如果成功获取相机数据，则执行 Line6-11；

Line6:定义工位 2（抓取点）；

Line7:定义工位 3（抓取点上方位置）；

Line8:机器人运动到工位 3；

Line9:机器人向下运动到工位 2；

Line10:等待抓取工件；

Line11:机器人向上运动到工位 3；

Line13:循环结束；

注意：因为在机器人端做手眼标定，所以 vision.x,vision.y,vision.z 三个工件位置数据结果可以直接应用，vision.a,vision.b,vision.c，需要根据现场实际情况进行数值上的补偿。

根据测试时机器人的抓取点来判断标定的准确性。