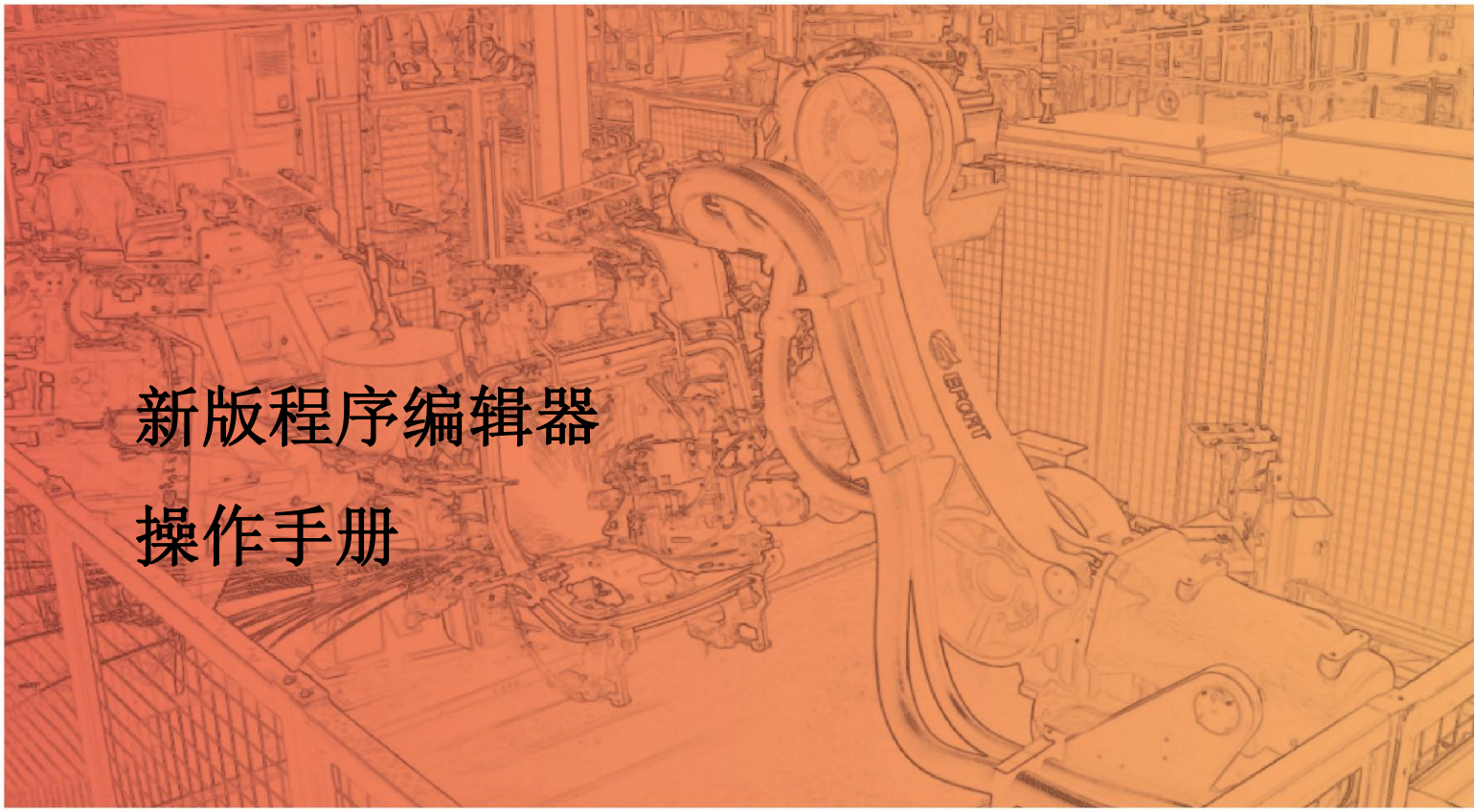
EFORT

智造专家 埃夫特



新版程序编辑器

操作手册

埃夫特智能装备股份有限公司

服务热线（Tel）：400-0528877

声 明

感谢您购买埃夫特机器人产品，为确保已对产品进行正确的设置，请您在使用本产品之前，务必仔细阅读本操作手册。本声明及手册所提及的内容涉及您的人身及财产安全，若不遵循或不按照手册的说明与警告而擅自操作，可能会给您和周围的人带来人身伤害或给埃夫特机器人或周围的其他物品造成财产损失。本声明及手册为截至本批次产品出厂前的最新版本，后续请通过访问 www.efort.com.cn 官方网站以获取更新的信息。

本手册仅作为对产品进行正常操作的指导，在产品使用过程中，埃夫特公司并不对除产品缺陷外的其他原因引发的人身伤害、财产损失承担责任。埃夫特公司郑重建议：参与机器人操作、示教、维护、维修、点检等相关活动的人员，在学习完毕埃夫特公司准备的培训课程前，请勿赋予其对机器人的操作使用权限。

版本号：V 0.1

目 录

概 述

1 关于本手册

本手册介绍了如何使用新版的程序编辑器。

2 手册使用

本手册应在操作机器人过程中使用。

3 本手册的阅读对象

本手册主要面向：产品开发与测试人员；技术服务人员；操作人员。

4 操作手册阅读指南

本操作手册分为以下几个章节。

章节	标题	内容
1	指令的功能及设置	介绍指令的功能及设置步骤
2	变量	介绍变量的数据类型、行为类型、可见性以及变量的新建、编辑、删除等操作。
3	程序树	介绍与程序树相关的操作，如新建、删除、重命名子程序，删除指令，切换显示程序，

5 操作前提

阅读本手册前，读者应当具备一定的机器人专业知识，或受过相应的机器人操作方面的技术培训。

6 修订历史

版本	日期（年/月/日）	原因（创建/修订）	创建或修订人
V0.1	2020.07.29	创建	王连兵，王乐，倪江涛，王旸

第 1 章　指令的功能及设置

1.1 本章简介

本章将介绍新版程序编辑器中所有可用指令的功能及其的设置步骤。

新版程序编辑器支持的指令如表 1-1；

表 1-1 程序编辑器支持指令列表

序号	分类	类型名称	增加版本	备注
1	基本/Basic	LABEL	V0.1	
2		GOTO	V0.1	
3		DWELL	V0.1	
4		WAIT	V0.1	
5		(* *)	V0.1	
6		SET	V0.1	
7		CALL	V0.1	
8		IF	V0.1	
9	循环/Loop	FOR	V0.4.0	
10		WHILE	V0.4.0	
11		CONTINUE	V0.4.0	
12		EXIT	V0.4.0	
13	运动/Moveme	MJOINT	V0.1	
14		MLIN	V0.1	

15	nt	MCIRC	V0.1	
16		EPATH	V0.4.0	暂不可用
17		MCIRCA	V0.4.0	
18		WAIT_POS	V0.4.0	
19		STOPMOVE	V0.4.3	
20		STARTMOVE	V0.4.3	
21		CLEARMOVE	V0.4.3	
22	运动参数 /MotionPar	KMAXT	V0.4.0	
23		KMAXJ	V0.4.0	
24		KMAXW	V0.4.0	
25	触发 /Trigger	TRIGON	V0.4.0	
26		TRIGCALL	V0.4.0	
27		TRIGSET	V0.4.0	
28	高级 /Advance	PULSE	V0.4.0	
29		LOAD	V0.4.0	
30		UNLOAD	V0.4.0	
31		EXEC	V0.4.0	
32		THREAD	V0.4.3	
33		ENDPROG	V0.4.3	
34		STOPPROG	V0.4.3	
35	时钟/Clock	CLOCKRESET	V0.4.0	

36		CLOCKSTART	V0.4.0	
37		CLOCKSTOP	V0.4.0	
38	中断 /Interrupt	INTRSET	V0.4.0	
39		INTRDIS	V0.4.0	
40		INTRENA	V0.4.0	
41		INTRCOND	V0.4.0	
42		INTRDENY	V0.4.0	
43		INTRALLOW	V0.4.0	
44		INTRERRNO	V0.4.0	
45	其他/Other	CASE	V0.4.0	
46		MESSAGE	V0.4.3	
47		ERROR	V0.4.3	
48		ALIAS	V0.4.3	
49		RETURN	V0.4.3	
50		RESTART	V0.4.3	
51		RETRY	V0.4.3	开发测试用
52		TRYNEXT	V0.4.3	开发测试用

1.2 基本 / Basic

1.2.1 标签 / LABEL

1.2.1.1 指令基本信息

指令名称：LABEL

指令形式：

LABEL labelName:

1.2.1.2 指令功能

该指令为在机器人运行程序中设置一个标签，可结合“GOTO”指令跳转到该行。

1.2.1.3 参数说明

表 1-2 指令参数列表

参数序号	参数名称	默认值	说明
1	名称	“label(i)”(i 为从 0 开始自增 1 的数)	自定义设置 label 的名称

1.2.1.4 使用用例

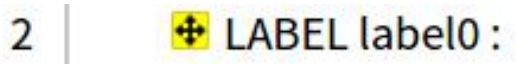
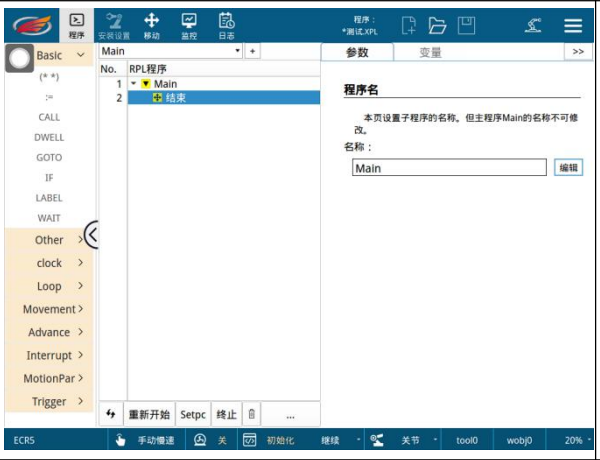
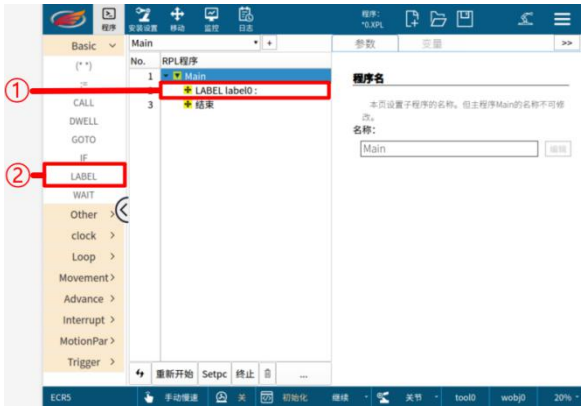
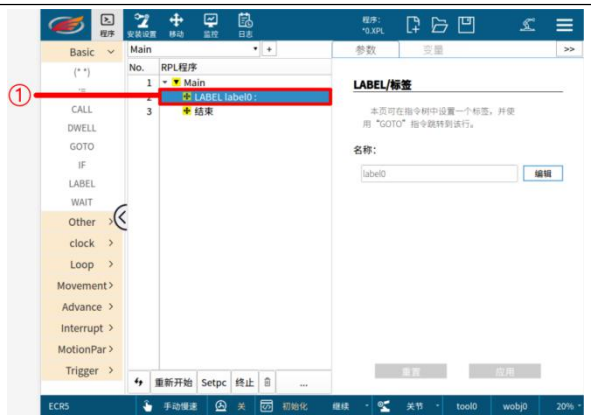
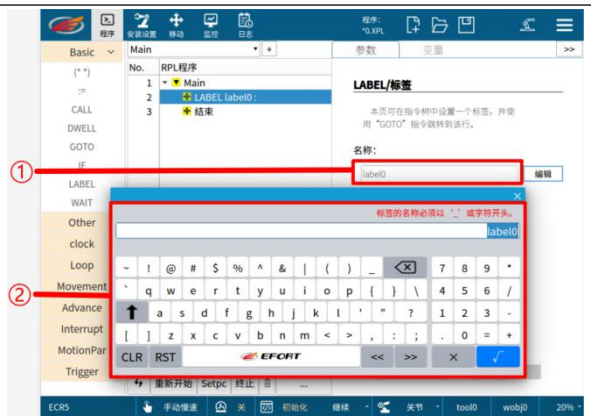
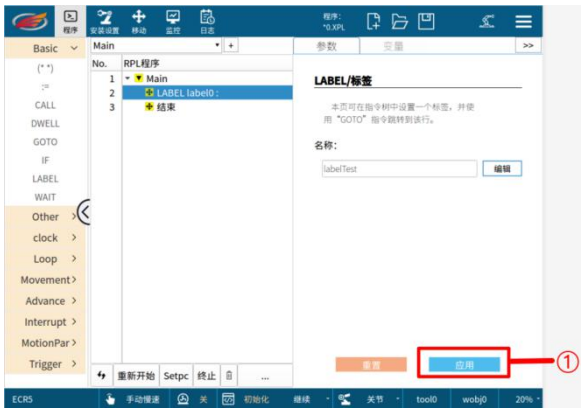
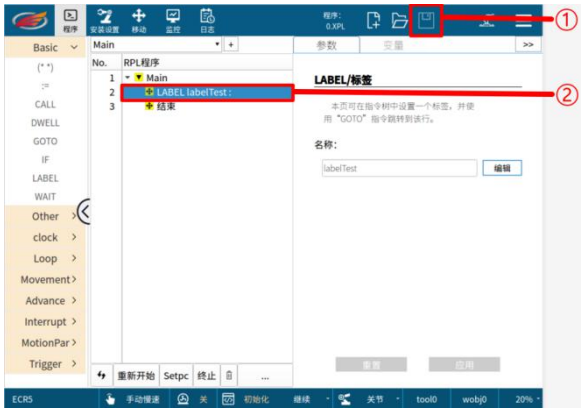


图 1-1 LABEL 指令使用用例

1.2.1.5 指令添加和设置步骤

表 1-3 LABEL 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“基本/Basic”中找到标签/ LABEL 指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”

3.点击程序树中刚刚添加的 LABEL 指令行，如右图①处。		点击程序树中的 LABEL 指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示 LABEL 指令的参数详情。
4.点击 LABEL 指令参数页中“名称”下的输入框①，并在弹出键盘②中输入自定义的名称		对自定义的名称有限制：必须以“_”或字符开头。
5.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
6.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.2.2 等待时间 / DWELL

1.2.2.1 指令基本信息

指令名称：DWELL

指令形式：

DWELL(时长);

1.2.2.2 指令功能：

该指令让机器人等待指定时长。

1.2.2.3 参数说明：

表 1-4 指令参数列表

参数序号	参数名称	默认值	说明
1	时长	1.000	时长单位为秒(s)

1.2.2.4 使用用例

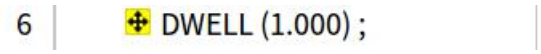
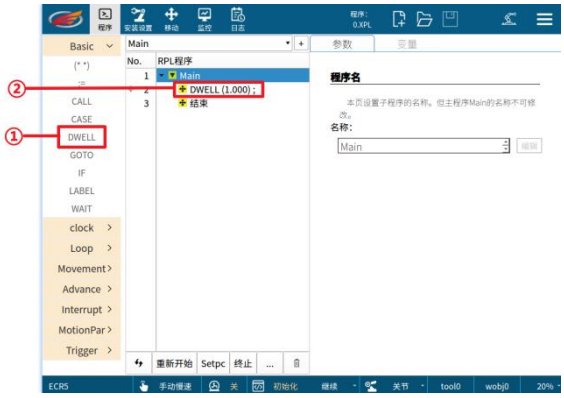
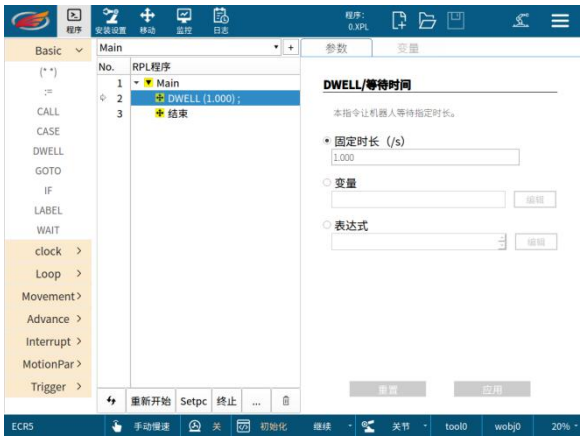
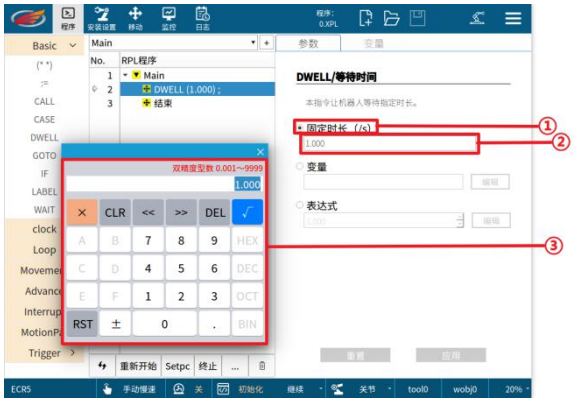


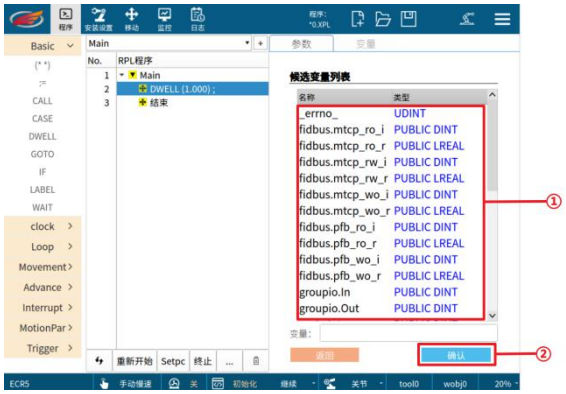
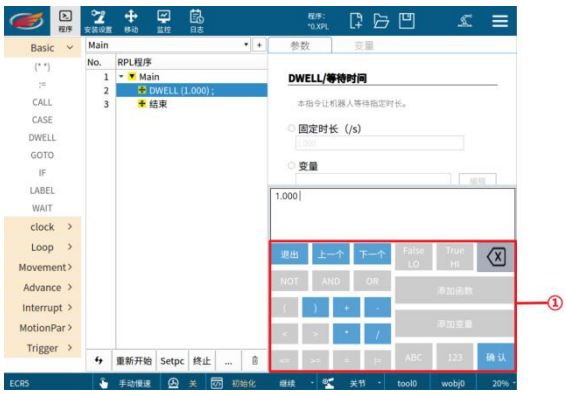
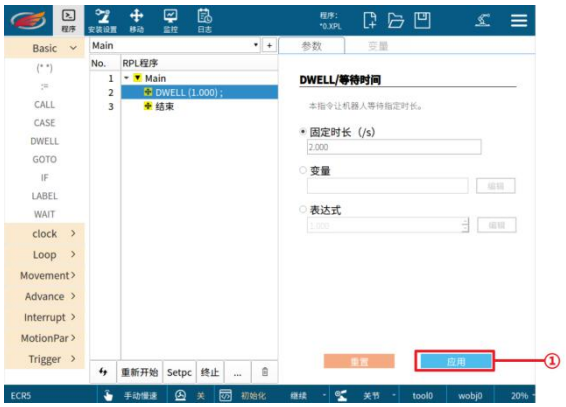
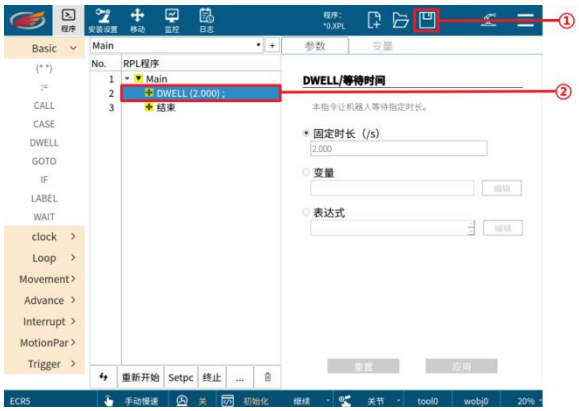
图 1-2 等待时间指令使用用例

1.2.2.5 等待时间 / DWELL 指令添加和设置步骤

表 1-5 等待时间 /DWELL 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2.在左侧指令栏中一级菜单“基本/Basic”中找到等待时间/DWELL指令，点击 DWELL 指令，向程序树中添加等待时间指令		1.若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”。
3.点击程序树中刚刚添加的等待时间指令行。		1.点击程序树中的等待时间行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示等待时间指令的参数树详情页。 2.等待时间指令拥有默认时长内容，为1.000。 3.等待时间指令时长的设置有三种方式，使用中根据实际情况选用一种即可。
4.选择等待时间指令参数页中固定时长选项并点击输入框，在弹出键盘中输入需要等待的时长		

5.选择等待时间指令参数页中变量选项，选择变量并确认		
6.选择等待时间指令参数页中表达式选项并编辑		
7.点击“应用”按钮将修改保存到程序树中		程序树中被选中的等待时间指令行的内容在点击应用后变为输入内容。
8.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.2.3 调用 / CALL

1.2.3.1 指令基本信息

指令名称：CALL

指令形式：

子程序名;

1.2.3.2 指令功能：

机器人在本节点调用子程序。

1.2.3.3 参数说明：

表 1-6 指令参数列表

参数序号	参数名称	默认值	说明
1	子程序		部分子程序本身也有参数，需要额外设置

1.2.3.4 使用用例

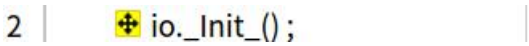
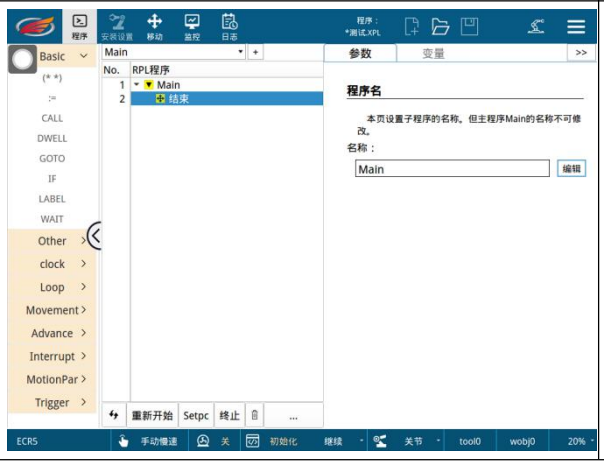
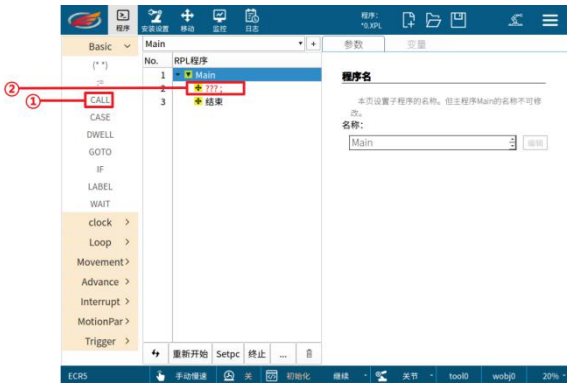
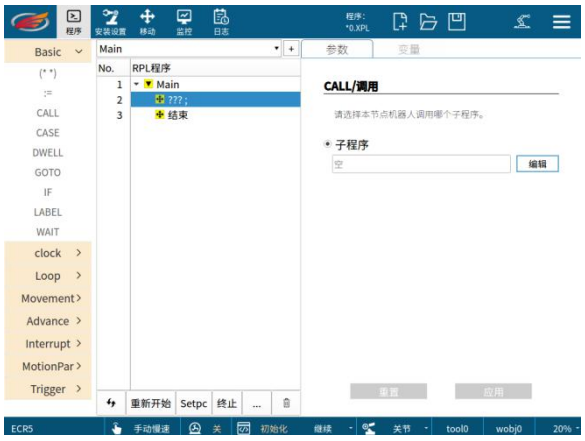
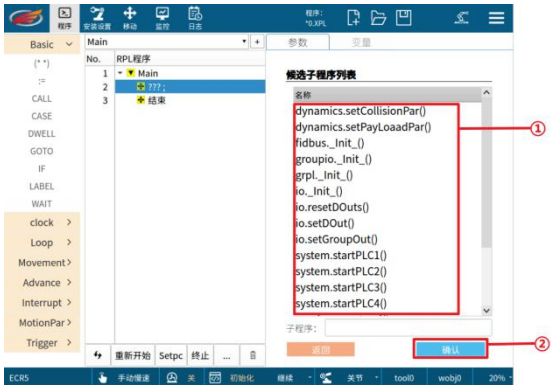
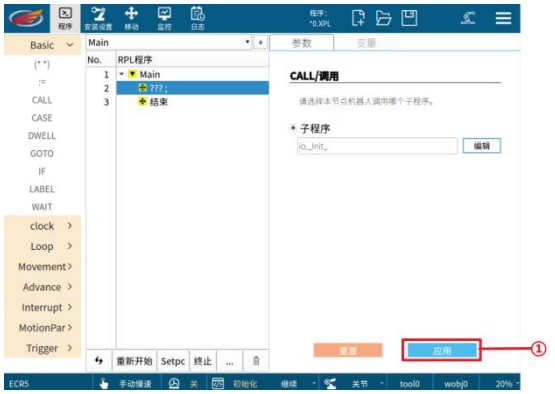


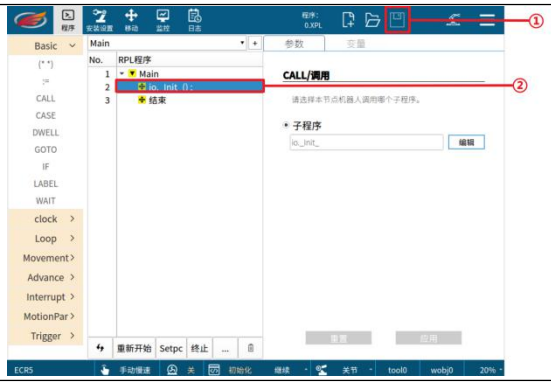
图 1-3 调用指令使用用例

1.2.3.5 调用 / CALL 指令添加和设置步骤

表 1-7 调用 /CALL 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2.在左侧指令栏中一级菜单“基本/Basic”中找到调用/CALL 指令，点击 CALL 指令，向程序树中添加调用指令		1.若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”。
3.点击程序树中刚刚添加的调用指令行。		1.点击程序树中的调用指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示调用指令的候选子程序详情页。
4.点击调用指令参数页中编辑按钮，在跳转的候选子程列表中选择子程序并确认		
5.点击“应用”按钮将修改保存到程序树中		程序树中被选中的调用指令行的内容在点击应用后变为编辑内容。

6.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.2.4 赋值 / :=

1.2.4.1 指令基本信息

指令名称：赋值

指令形式：

变量:=表达式;

1.2.4.2 指令功能：

通过此指令，可以为变量赋值。注意此指令不能用于将一个数组直接赋值给另外一个数组。如果想要把一个数组的值 复制到另外一个数组，必须针对数组中的每个元素使用此指令。

1.2.4.3 参数说明：

表 1-8 指令参数列表

参数序号	参数名称	默认值	说明
1	变量名称		需要接受赋值的变量名称
2	表达式		将此值赋值给:=前的变量

1.2.4.4 使用用例

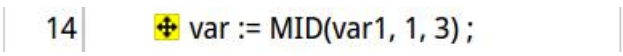
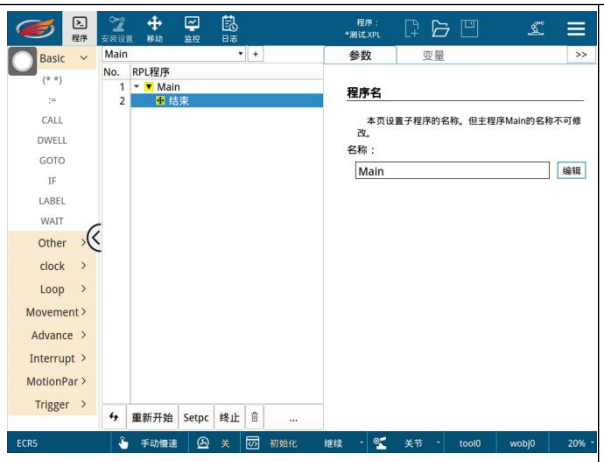
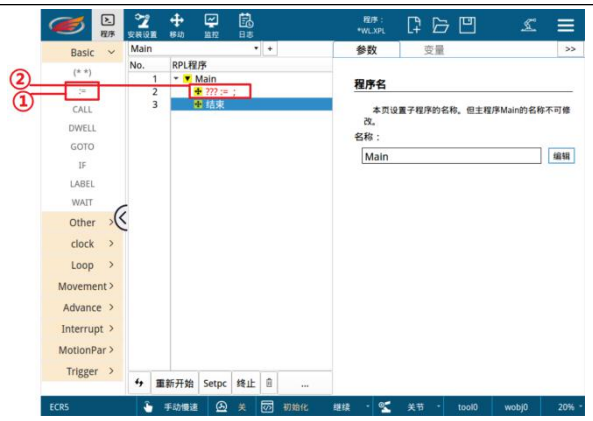
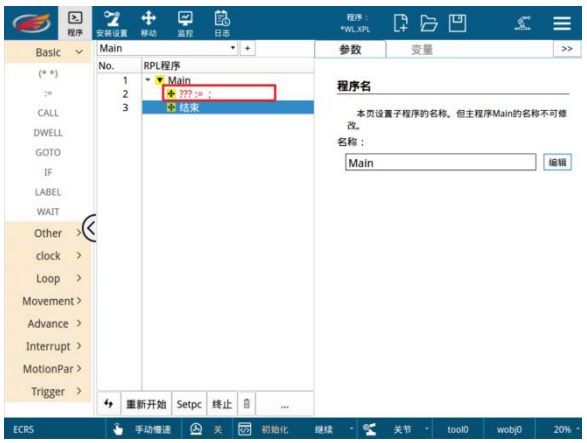


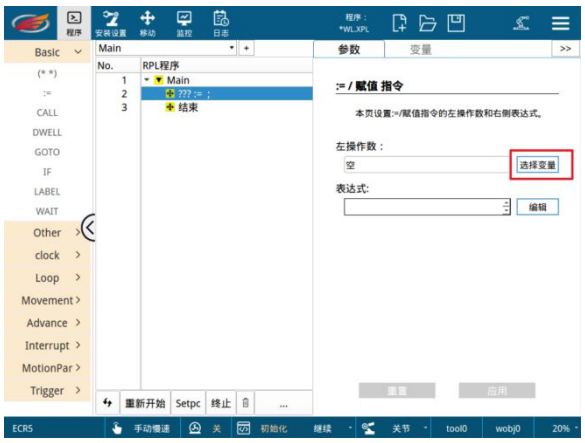
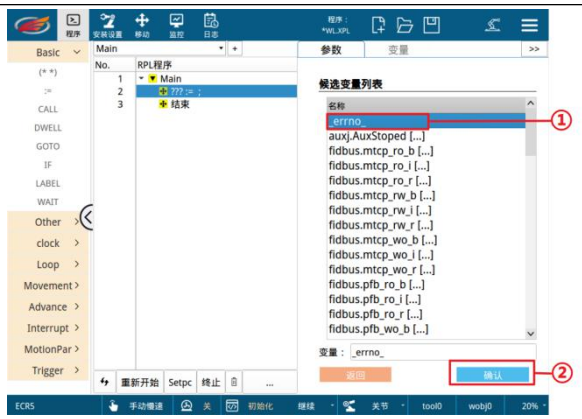
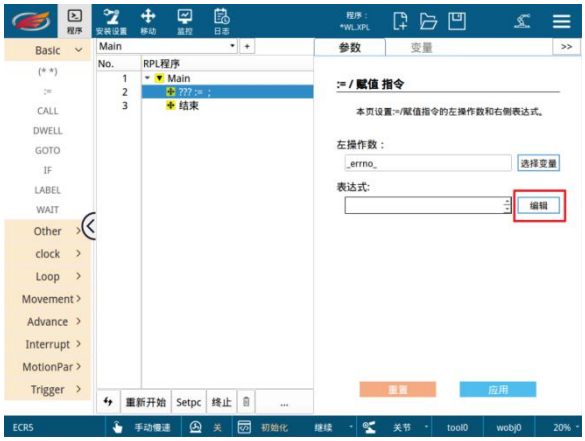
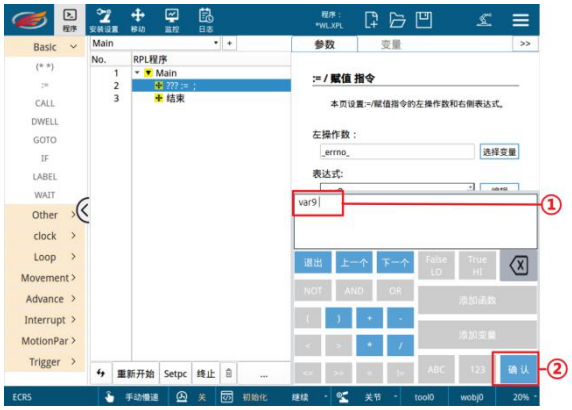
图 1-4 赋值指令使用用例

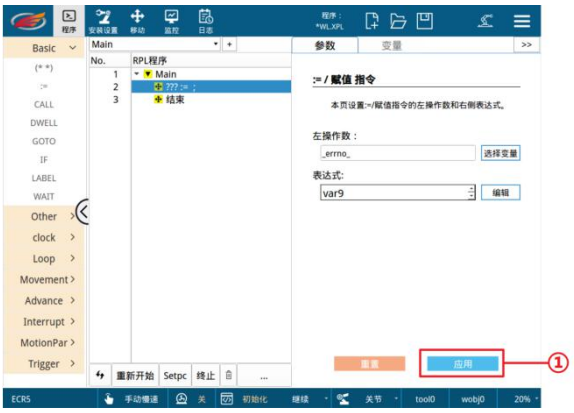
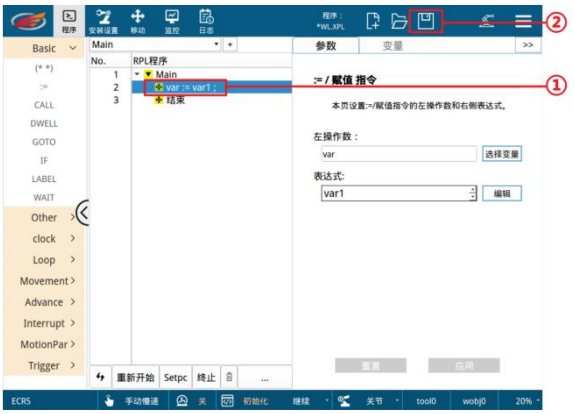
1.2.4.5 赋值 / :=指令添加和设置步骤

表 1-9 赋值 / := 指令添加和设置步骤

步骤	图示	说明

1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“基本/Basic”中找到赋值/ := 指令，点击 := 指令，向程序树中添加赋值指令		若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”
3.点击程序树中刚刚添加的赋值指令行。		点击程序树中的赋值指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示赋值指令的参数树详情页。 赋值指令两个参数默认都为空。

4.点击赋值指令参数页中第一个参数后的“选择变量”按钮，跳转到候选变量列表界面。		
5.选择一个需要接受赋值的变量，完成后点击“确认”按钮。		
6.点击表达式参数后的“编辑”按钮，弹出表达式编辑框。		
7.编辑一个表达式，操作完成后点击“确认”按钮。		表达式可以是函数，可以是变量，也可以是组合。此表达式用于传值给:=前的变量。

8.点击“应用”按钮将修改保存到程序树中		程序树中被选中的赋值指令行的参数无法点击手动输入。
9.点击任务栏中“保存”按钮，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.2.5 逻辑判断 / IF

1.2.5.1 指令基本信息

指令名称：IF

指令形式：

IF 条件名称

END IF

1.2.5.2 指令功能：

通过条件来判断是否执行判断后的事件。IF 条件为真，则执行 IF 后 END IF 之前的指令语句。当键入 IF 指令时，END IF 会自动添加。可在 IF 指令后添加 ELSE 语句，若 IF 条件不成立，则会执行 ELSE 后 END IF 前的指令。

1.2.5.3 参数说明：

表 1-10 指令参数列表

参数序号	参数名称	默认值	说明
1	判断条件	“true”	判断条件，通过该条件判断是否执行 IF 后的指令

1.2.5.4 使用用例

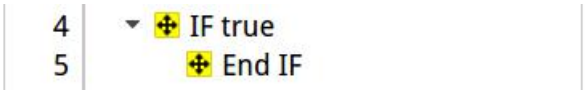
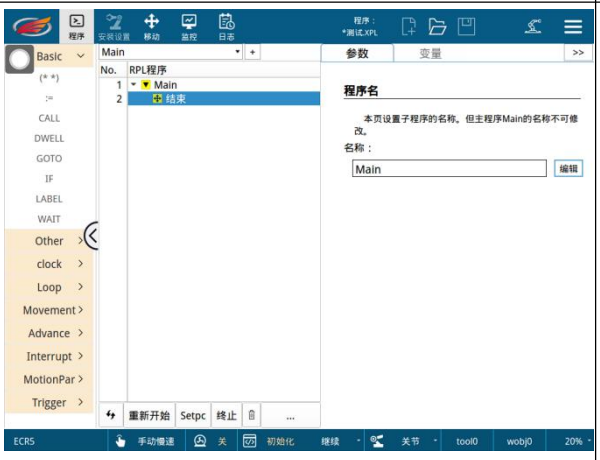
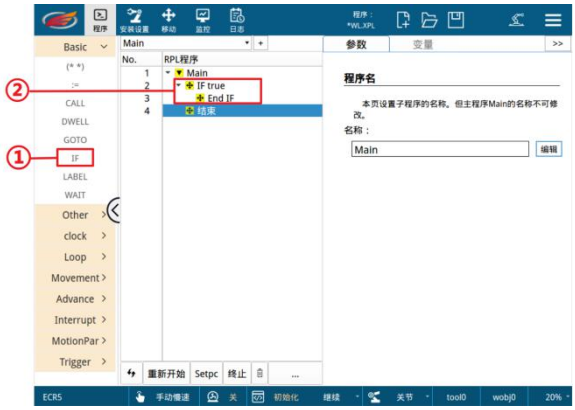
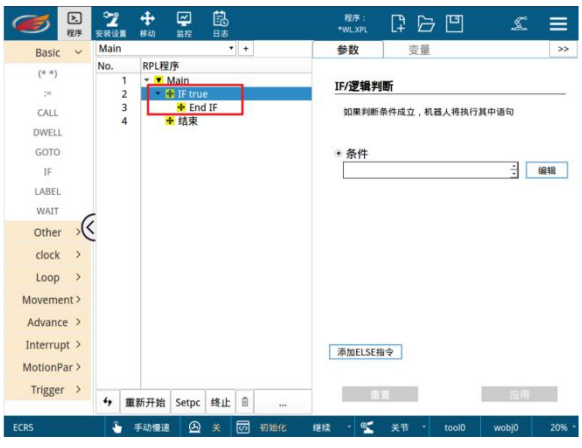
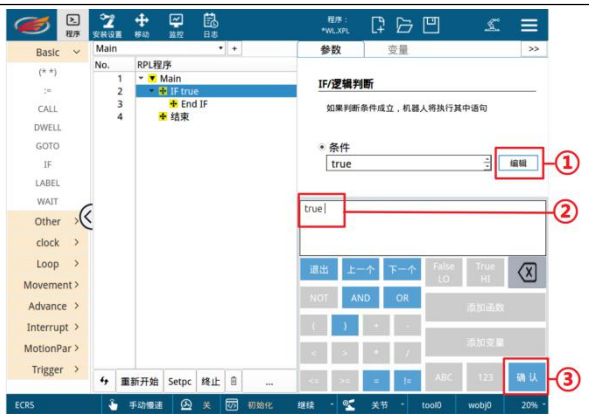
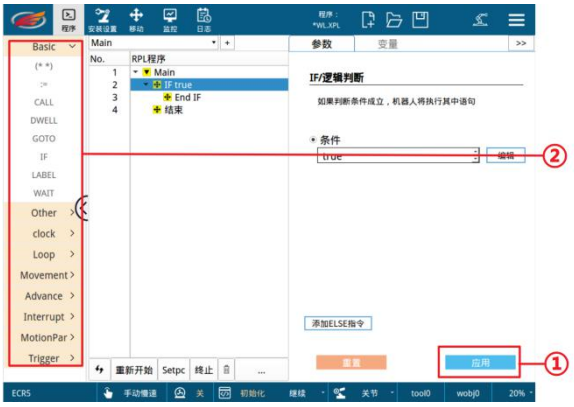
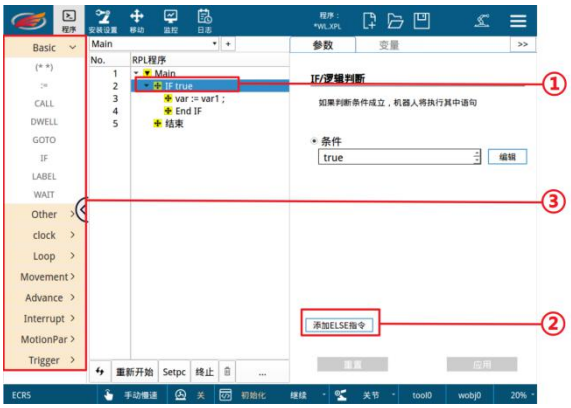
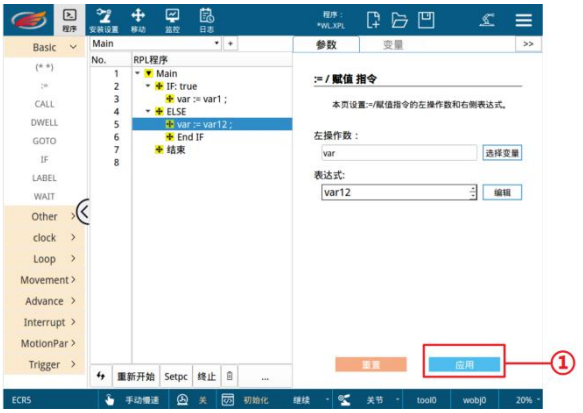


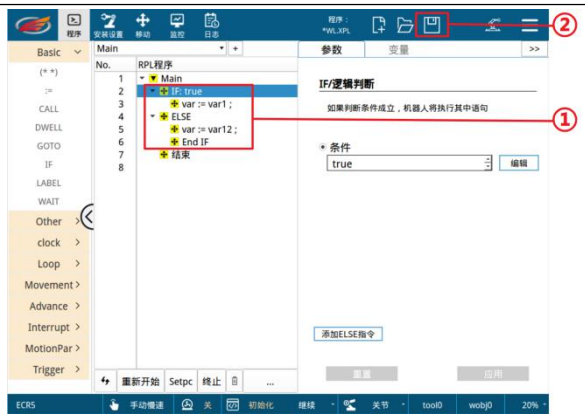
图 1-5 IF 指令使用用例

1.2.5.5 逻辑判断 /IF 指令添加和设置步骤

表 1-11 逻辑判断 /IF 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“基本/Basic”中找到逻辑判断 /IF 指令，点击 IF 指令，向程序树中添加 IF 指令		1.若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”
3.点击程序树中刚刚添加的 IF 指令行。		点击程序树中的 IF 行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示 IF 指令的参数树详情页。 IF 指令默认条件为 true。

4.点击 IF 指令参数页中内容后的编辑按钮弹出一个条件输入键盘，编辑插入到程序树中的 IF 判断条件的内容。 5.输入完成后点击确认按钮。		
6 点击应用。 7.在 IF 指令后插入一条想要在判断条件成立后执行的指令，完成编辑。		
8.若想要使用 ELSE 功能，则点击 IF 参数页中的添加 ELSE 指令按钮。 9.然后在 ELSE 指令后添加并完成编辑你想要执行的指令。		
10.点击“应用”按钮将修改保存到程序树中		程序树中被选中的 IF 指令行的参数无法点击手动输入。

11.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.2.6 事件等待 / WAIT

1.2.6.1 指令基本信息

指令名称：WAIT

指令形式：

WAIT(condition,[timeout],[var]);

1.2.6.2 指令功能

等待条件的执行，如果设置了超时，则可以选择中断等待。

1.2.6.3 参数说明

表 1-12 指令参数列表

参数序号	参数名称	默认值	说明
1	条件	空	设置需要等待的条件。
2	等待时长	空	可选项。设置等待的时长
3	变量	空	可选项。设置结果变量，如果 WAIT 正确结束该变量为 0，否则在超时发生时该变量将不等于 0。

1.2.6.4 使用用例

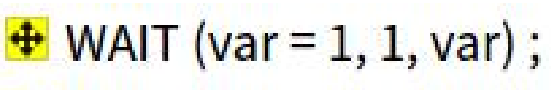
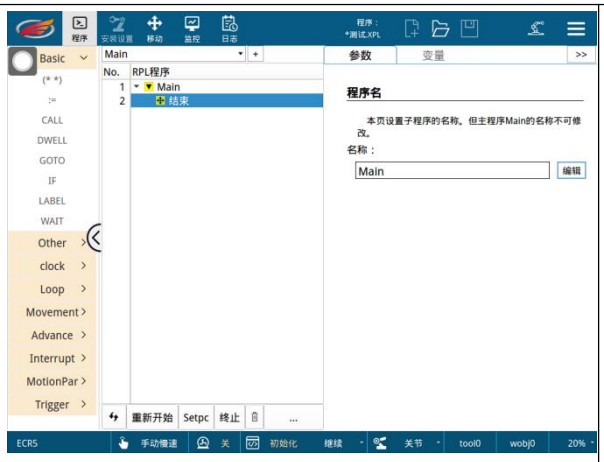
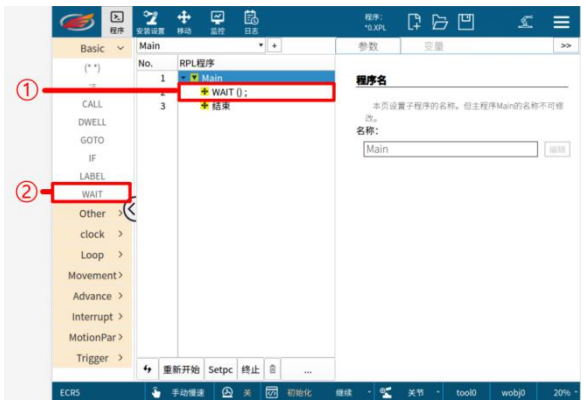
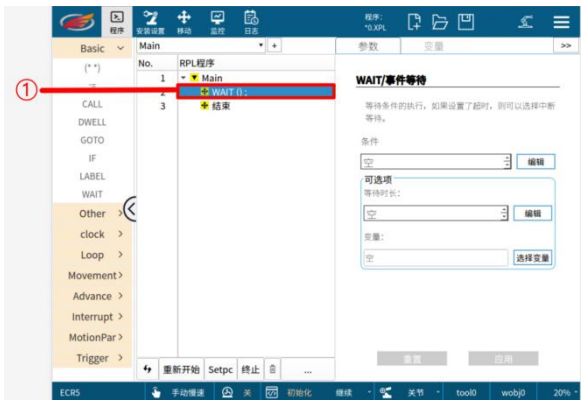
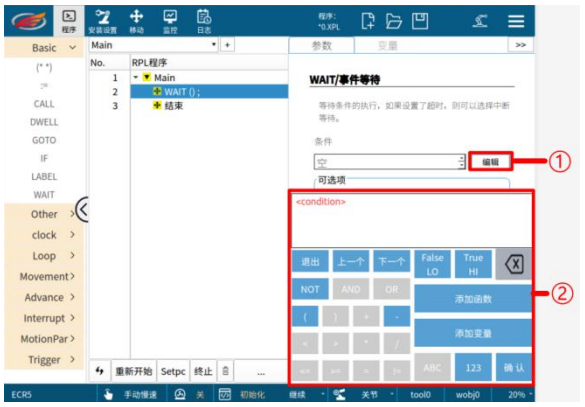


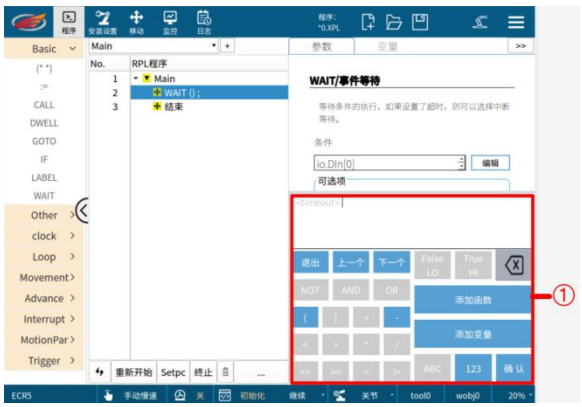
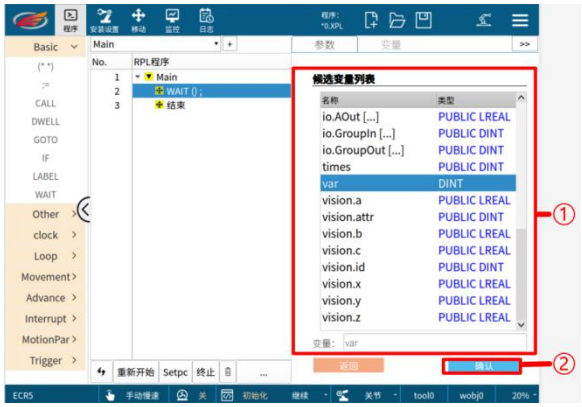
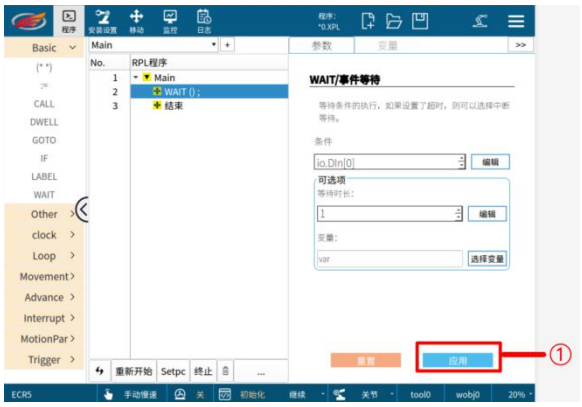
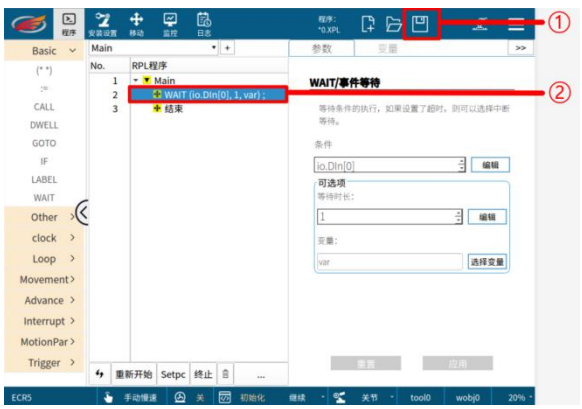
图 1-6 WAIT 指令使用用例

1.2.6.5 指令添加和设置步骤

表 1-13 WAIT 指令添加和设置步骤

步骤	图示	说明

1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“基本/Basic”中找到WAIT指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		**若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”**
3.点击程序树中刚刚添加的WAIT指令行，如右图①处。		点击程序树中的WAIT指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示WAIT指令的参数详情。
4.点击WAIT指令参数页中“条件”参数后的“编辑”按钮①，并在弹出的表达式编辑器②中输入一个条件，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时“条件”参数下的输入框文本将被设置为在表达式编辑器上编辑的文本。

5.(可选项)点击WAIT指令参数页中“等待时长”参数后的“编辑”按钮，并在弹出的表达式编辑器①中输入需要等待的时长，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时“等待时长”参数下的输入框文本将被设置为在表达式编辑器上编辑的文本。
6.(可选项)点击WAIT指令参数页中“变量”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“变量”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
7.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
8.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.2.7 跳转 / GOTO

1.2.7.1 指令基本信息

指令名称：GOTO

指令形式：

GOTO(标签名称)

1.2.7.2 指令功能：

该指令可以跳到某一个特定的标签，并执行标签后的程序指令。

1.2.7.3 参数说明：

表 1-14 指令参数列表

参数序号	参数名称	默认值	说明
1	标签名称		指令标签名称，跳转到该指令处继续执行

1.2.7.4 使用用例

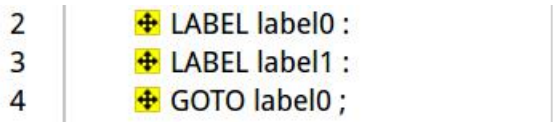
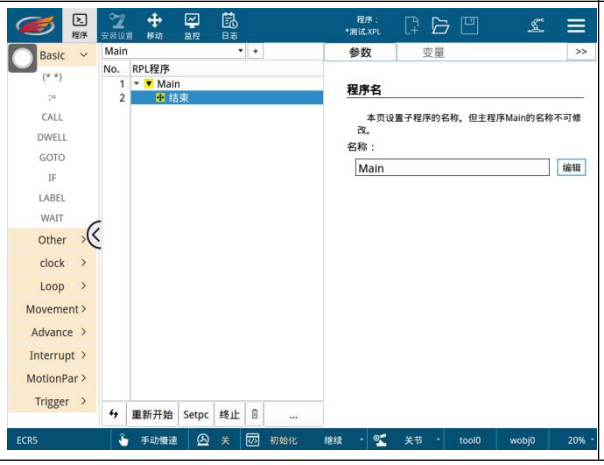
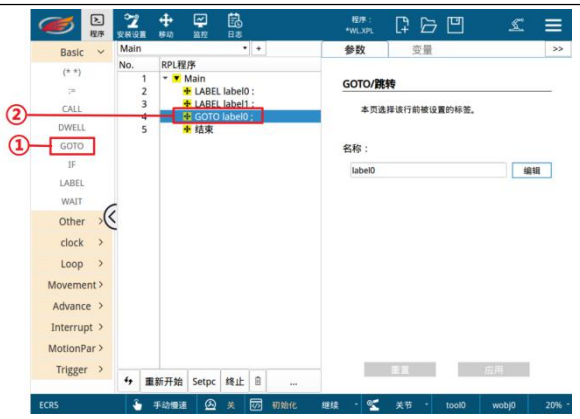
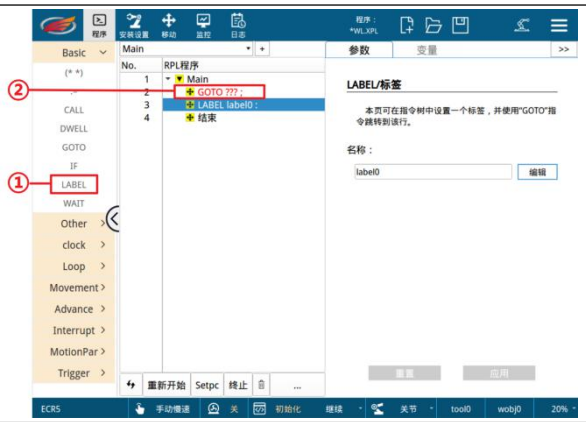
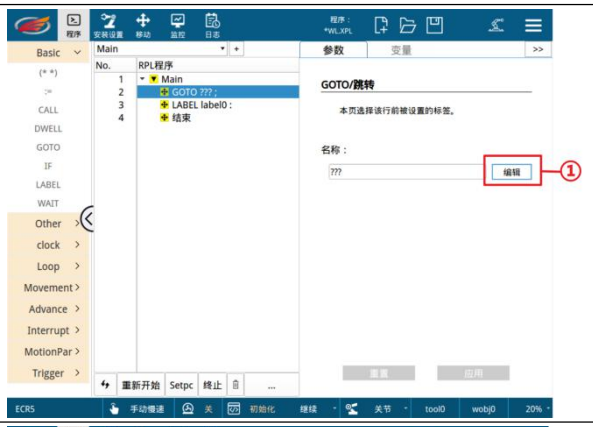
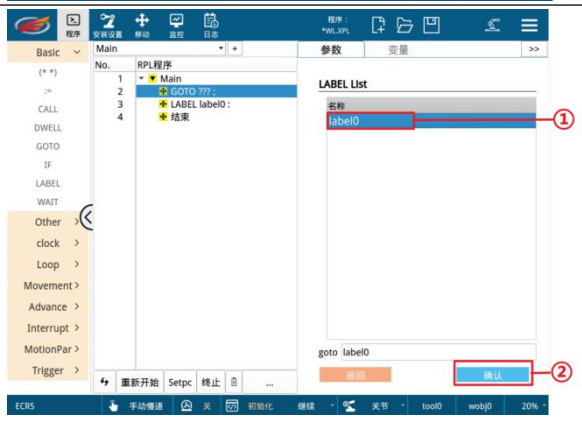


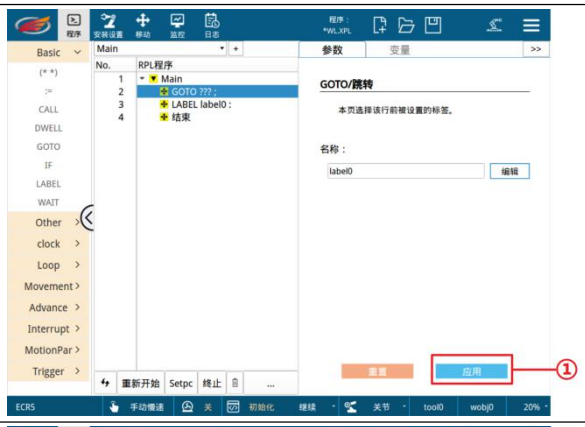
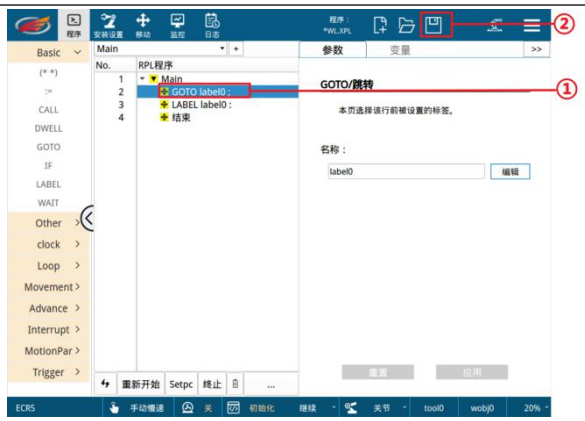
图 1-7 GOTO 指令使用用

1.2.7.5 跳转 / GOTO 指令添加和设置步骤

表 1-15 跳转 / GOTO 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2. 在左侧指令栏中一级菜单“基本/Basic”中找到跳转/GOTO 指令，点击 GOTO 指令，向程序树中添加 GOTO 指令		若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”
3.点击一级菜单下的“LABEL”指令在程序树中需要的地方插入一个 LABEL 标签。 4.点击程序树中刚刚添加的跳转指令行。		点击程序树中的跳转行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示跳转指令的参数树详情页。 跳转指令默认标签内容为空。
5.点击跳转指令参数页中内容后的编辑按钮，跳转到 label list 界面，选择插入到程序树中的 LABEL 标签。		
6.选择一个想要跳转到的标签，并点击确认按钮。		

7.点击“应用”按钮将修改保存到程序树中		程序树中被选中的跳转指令行的参数无法点击手动输入。
8.点击任务栏中“保存”按钮，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.2.8 注释 / (* *)

1.2.8.1 指令基本信息

指令名称：(* *)

指令形式：

(*内容*)

1.2.8.2 指令功能：

该指令为在机器人运行程序中添加一行注释，注释指令行在程序树中字体颜色为绿色，并使用“(**)”包裹注释的说明。

1.2.8.3 参数说明：

表 1-16 指令参数列表

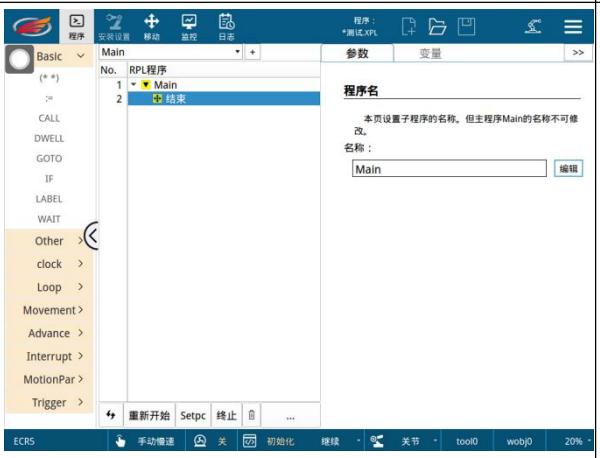
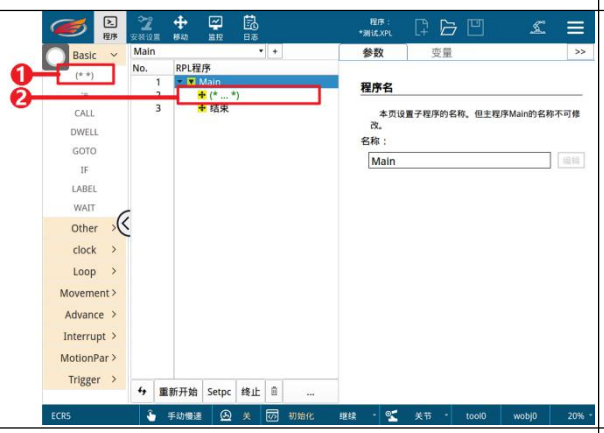
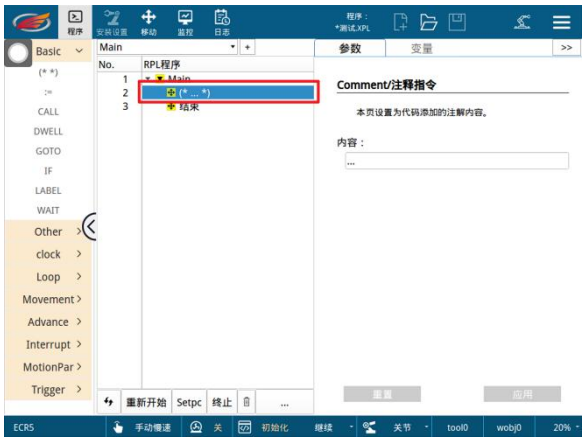
参数序号	参数名称	默认值	说明
1	内容	“...”	具体的注释内容，根据需求输入的对程序的说明

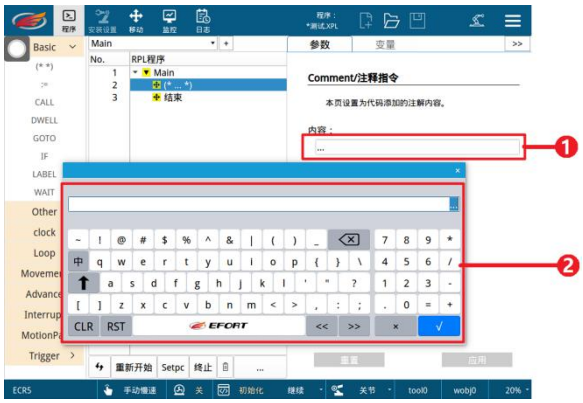
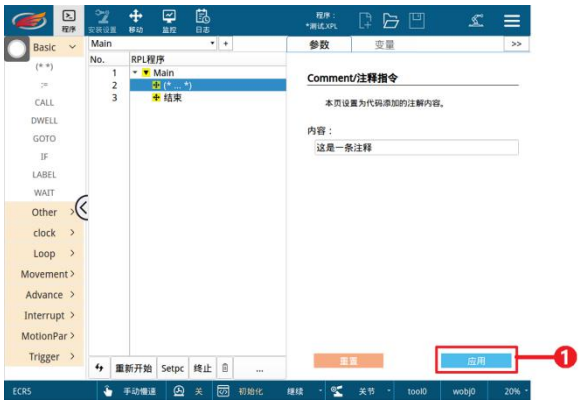
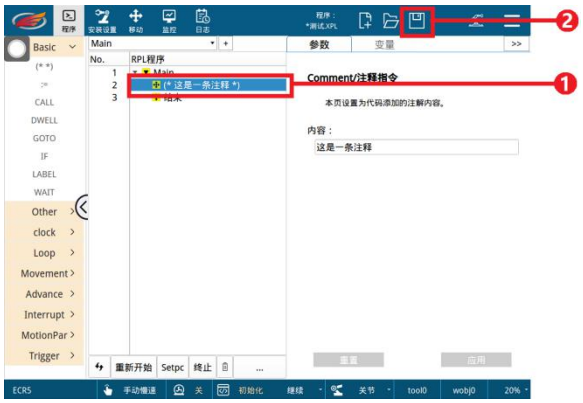
1.2.8.4 使用用例

图 1-8 注释指令使用用例

1.2.8.5 注释 / (* *)指令添加和设置步骤

表 1-17 注释 / (* *) 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“基本/Basic”中找到注释 / (* *)指令，点击 (* *) 指令，向程序树中添加注释指令		1.若一级菜单“基本/Basic”未展开，点击一级菜单“基本/Basic”
3.点击程序树中刚刚添加的注释指令行。		1.点击程序树中的注释行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示注释指令的参数树详情页。 2.注释指令拥有默认注释内容，为“...”。

4.点击注释指令参数页中内容下的输入框，并在弹出键盘中输入注释的具体内容		
5.点击“应用”按钮将修改保存到程序树中		程序树中被选中的注释指令行的内容在点击应用后变为输入输入内容。
6.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.3 循环 / Loop

1.3.1 For 循环 / FOR

1.3.1.1 指令基本信息

指令名称：FOR

指令形式：

FOR variable

…

END FOR ;

1.3.1.2 指令功能

循环执行某个代码块几次。首先需要设置使用起始值来初始化变量 variable，在代码块执行每次执行后，变量 variable 将会以步进的值更新；如果更新后的变量 variable 值在起始值和终止值之间指定范围内，代码块将会再次执行，否则执行 FOR 循环体外的后续指令。若步进不进行任何指定，默认变量 variable 的步进为 1。FOR 循环体重可以使用 EXIT 和 CONTIN UE 指令。

1.3.1.3 参数说明

表 1-18 指令参数列表

参数序号	参数名称	默认值	说明
1	变量	空	设置循环中的变量
2	起始值	空	设置变量的起始值
3	终止值	空	设置变量的终止值
4	步进	1	设置变量的步进值

1.3.1.4 使用用例

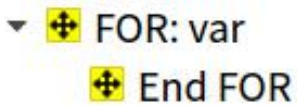
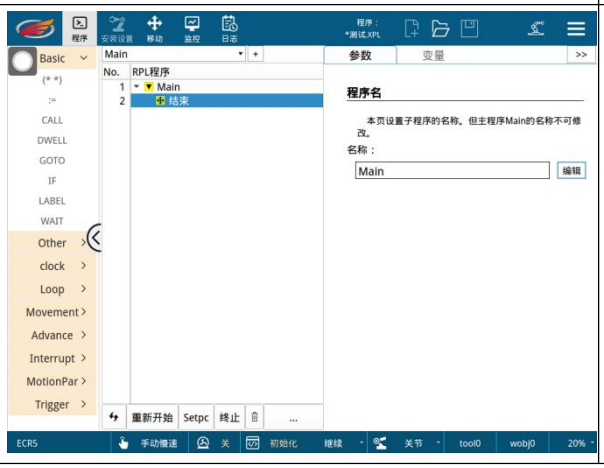
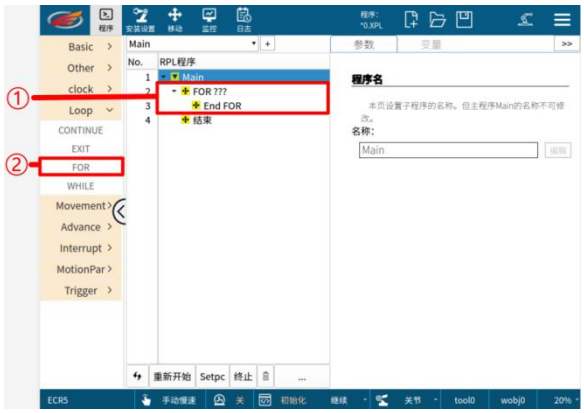
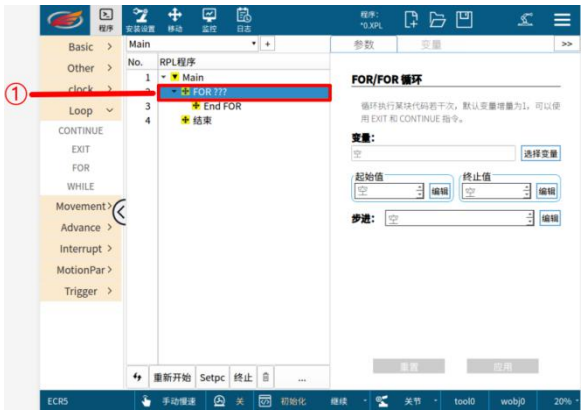
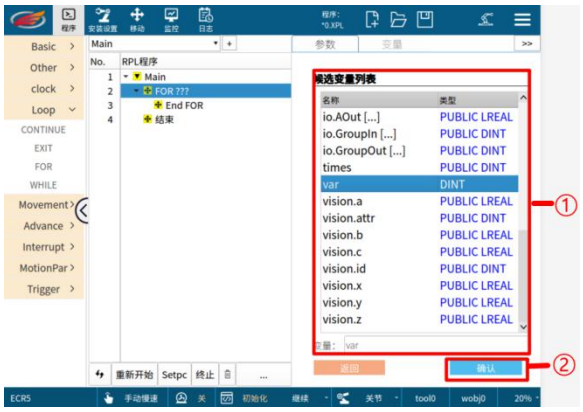
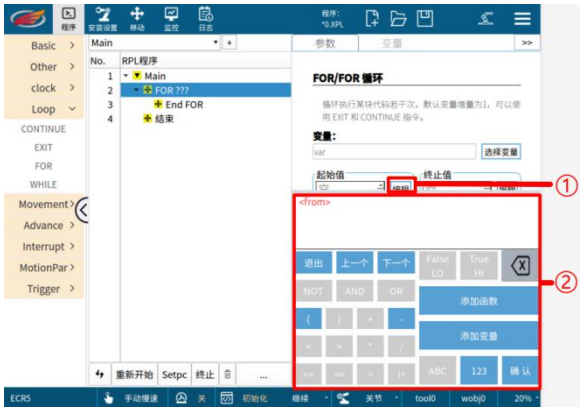


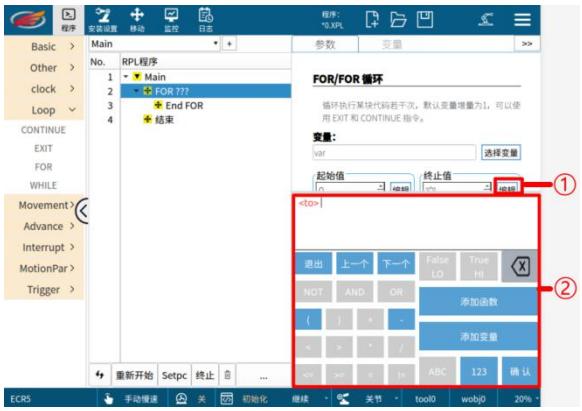
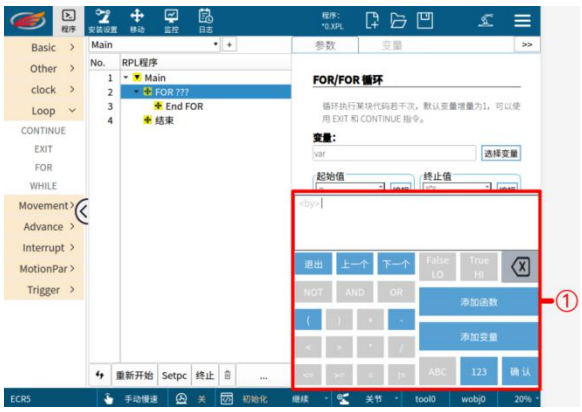
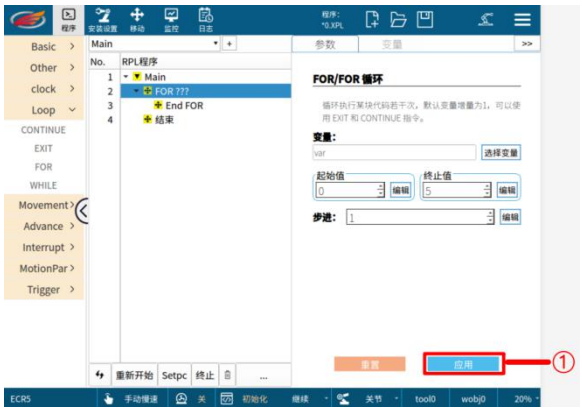
图 1-9 FOR 指令使用用例

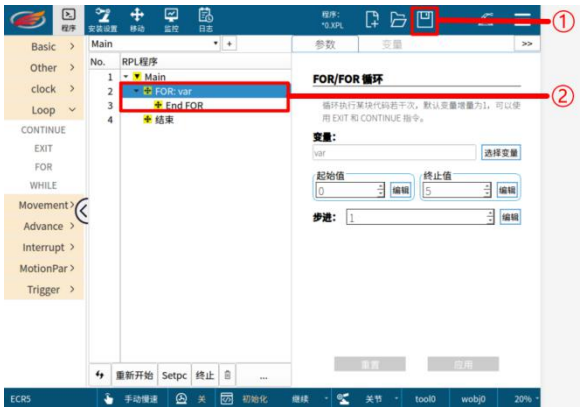
1.3.1.5 指令添加和设置步骤

表 1-19 FOR 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2．在左侧指令栏中一级菜单“循环/Loop”中找到 FOR 指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“循环/Loop”未展开，点击一级菜单“循环/Loop”
3.点击程序树中刚刚添加的 FOR 指令行，如右图①处。		点击程序树中的 FOR 指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示 FOR 指令的参数详情。
4.点击 FOR 指令参数页中“变量”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“变量”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
5.点击 FOR 指令参数页中“起始值”参数后的“编辑”按钮①，并在弹出的表达式编辑器②中输入起始值，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时“起始值”参数下的输入框文本将被设置为在表达式编辑器上编辑的文本。

6.点击 FOR 指令参数页中“终止值”参数后的“编辑”按钮①，并在弹出的表达式编辑器②中输入终止值，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时“终止值”参数下的输入框文本将被设置为在表达式编辑器上编辑的文本。
7.点击 FOR 指令参数页中“步进”参数后的“编辑”按钮后并在弹出的表达式编辑器①中输入步进，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时“步进”参数下的输入框文本将被设置为在表达式编辑器上编辑的文本。
8.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。

9.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.3.2 继续 /CONTINUE

1.3.2.1 指令基本信息

指令名称：CONTINUE

指令形式：

CONTINUE;

1.3.2.2 指令功能：

中断 WHILE 或 FOR 循环，并强制重新评估目前的循环条件。

1.3.2.3 参数说明：

表 1-20 指令参数列表

参数序号	参数名称	默认值	说明

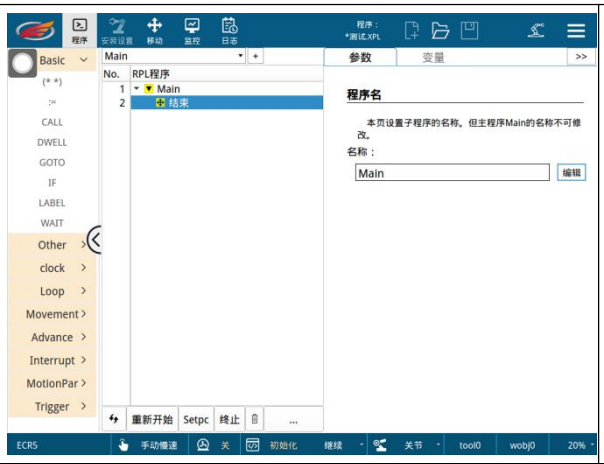
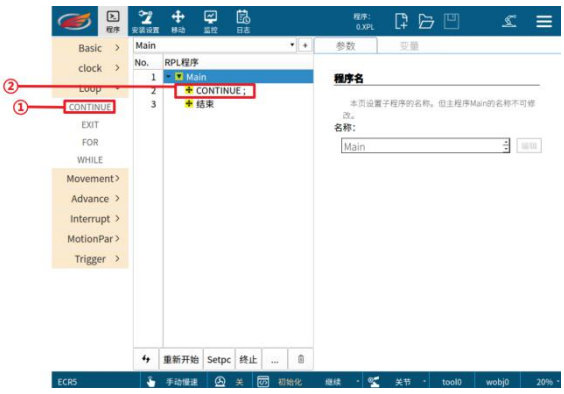
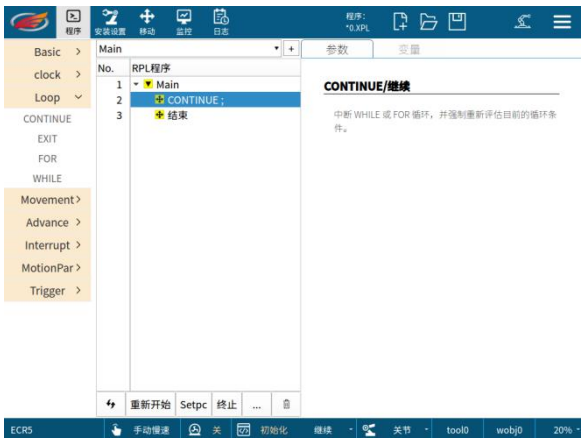
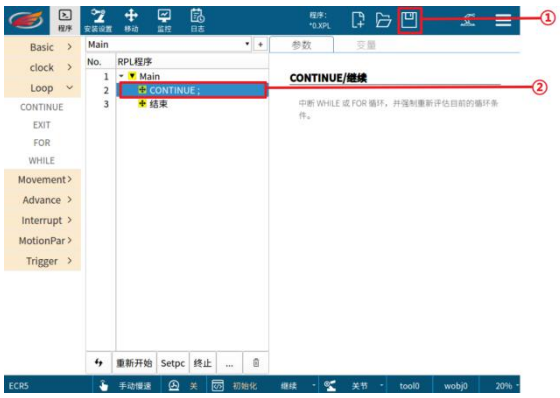
1.3.2.4 使用用例

图 1-10 继续指令使用用例

1.3.2.5 继续 / CONTINUE 指令添加和设置步骤

表 1-21 继续 /CONTINUE 指令添加和设置步骤

步骤	图示	说明

1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“Loop”中找到继续/CONTINUE 指令，点击 CONTINUE 指令，向程序树中添加继续指令		1.若一级菜单“Loop”未展开，点击一级菜单“Loop”。
3.点击程序树中刚刚添加的继续指令行。		1.点击程序树中的继续指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。
4.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.3.3 结束循环 /EXIT

1.3.3.1 指令基本信息

指令名称：EXIT

指令形式：

EXIT;

1.3.3.2 指令功能：

终止 WHILE 或 FOR 循环。

1.3.3.3 参数说明：

表 1-22 指令参数列表

参数序号	参数名称	默认值	说明

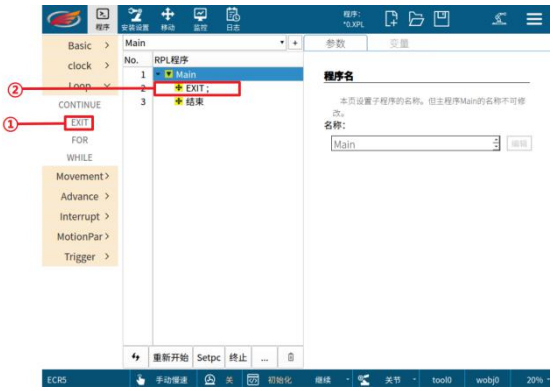
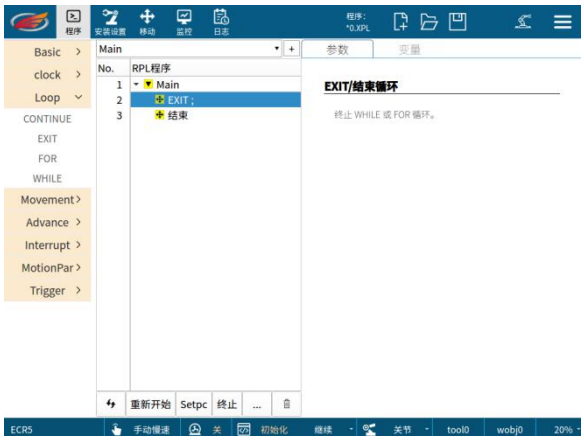
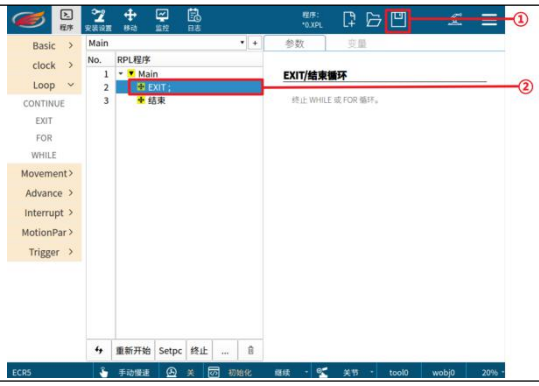
1.3.3.4 使用用例

图 1-11 结束循环指令使用用例

1.3.3.5 结束循环 / EXIT 指令添加和设置步骤

表 1-23 结束循环 /EXIT 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2.在左侧指令栏中一级菜单“Loop”中找到结束循环/EXIT 指令，点击 EXIT 指令，向程序树中添加结束循环指令		1.若一级菜单“Loop”未展开，点击一级菜单“Loop”。
3.点击程序树中刚刚添加的结束循环指令行。		1.点击程序树中的结束循环指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。
4.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.3.4 WHILE 循环 / WHILE

1.3.4.1 指令基本信息

指令名称：WHILE

指令形式：

WHILE true

END WHILE

1.3.4.2 指令功能：

如果 WHILE 后的条件为真，则执行 WHILE 后 END WHILE 之前的指令。END WHILE 之后，会再次回到 WHILE 判断条件，若判断条件依然成立，则会继续执行 WHILE 后 END WHILE 之前的指令，直到 WHILE 不成立，循环结束。当键入 WHILE 指令时，END_WHILE 会自动添加。

1.3.4.3 参数说明：

表 1-24 指令参数列表

参数序号	参数名称	默认值	说明
1	判断条件	“true”	判断条件，如果成立则执行后面的指令。

1.3.4.4 使用用例

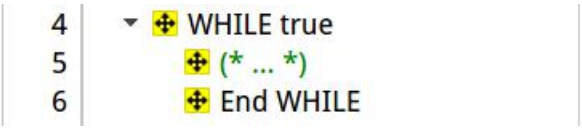
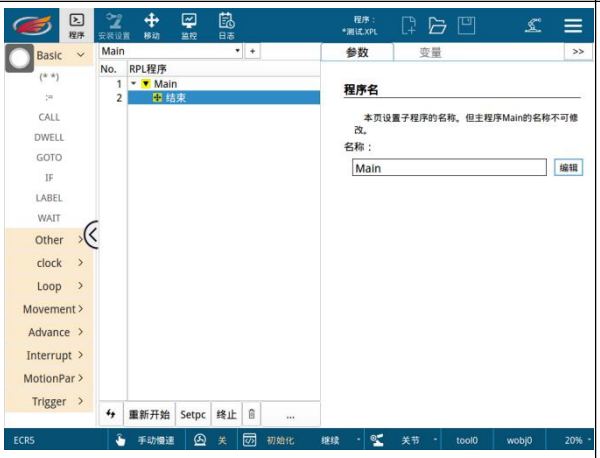
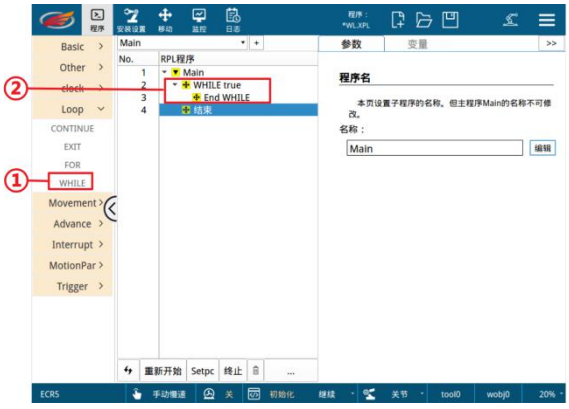
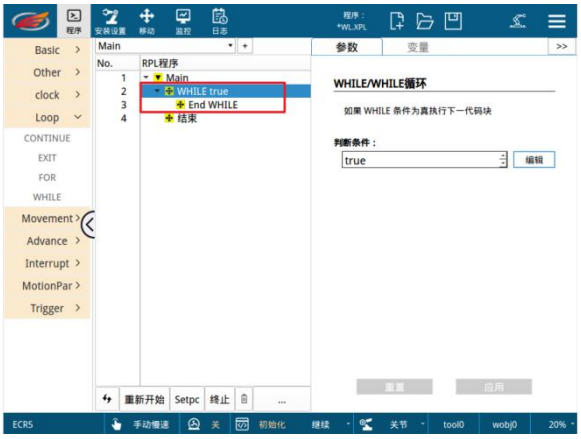
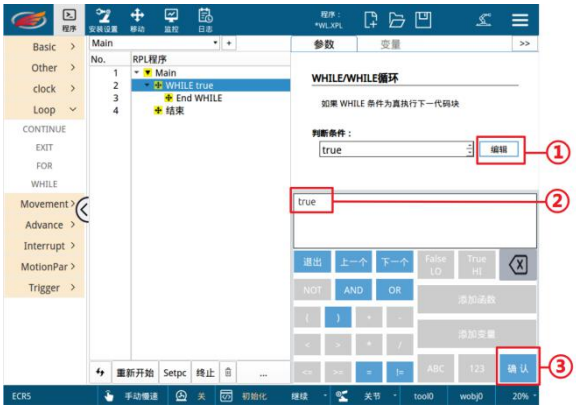
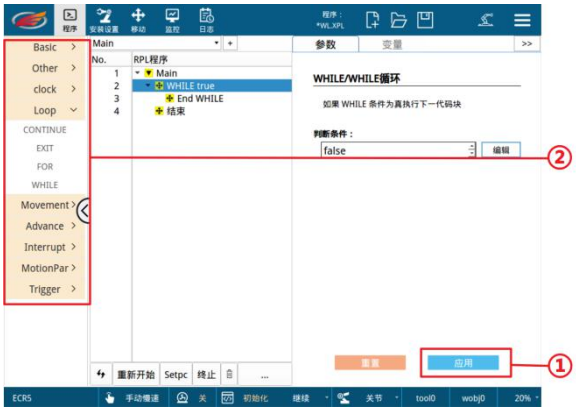
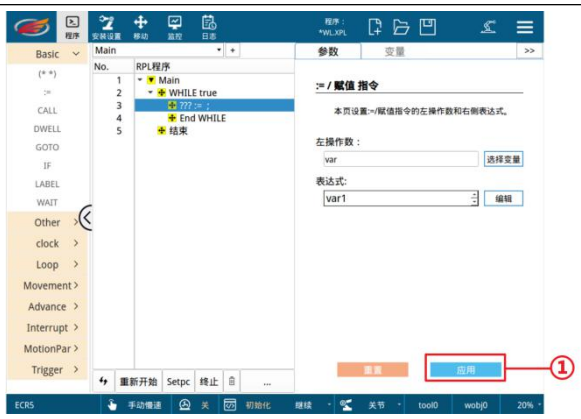


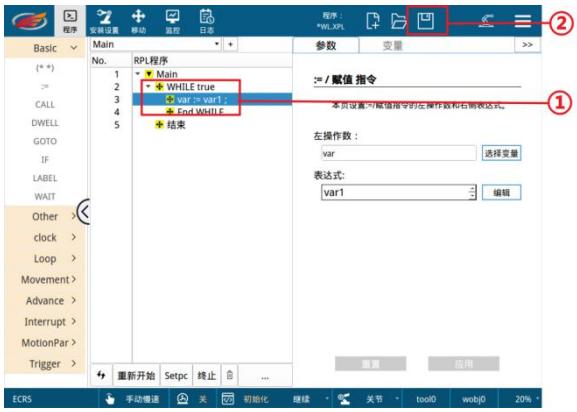
图 1-12 WHILE 指令使用用例

1.3.4.5 WHILE 循环 / WHILE 指令添加和设置步骤

表 1-25 WHILE 循环 / WHILE 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“循环/Loop”中找到 WHILE 循环 / WHILE 指令，点击 WHILE 指令，向程序树中添加 WHILE 指令		若一级菜单“循环/Loop”未展开，点击一级菜单“循环/Loop”

3.点击程序树中刚添加的 WHILE 指令行。		点击程序树中的 WHILE 行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示 WHILE 指令的参数树详情页。 WHILE 指令默认条件为 true。
4.点击 WHILE 指令参数页中判断条件框后的编辑按钮，弹出一个条件输入键盘，编辑插入到程序树中的 WHILE 判断条件。 5.输入完成后点击确认按钮。		
6 点击应用。 7.在 WHILE 指令后插入若干条想要在判断条件成立时循环执行的指令，完成编辑。		
8.点击“应用”按钮将修改保存到程序树中		程序树中被选中的跳转指令行的参数无法点击手动输入。

9.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.4 运动 / Movement

1.4.1 等待位姿 / WAIT_POS

1.4.1.1 指令基本信息

指令名称：WAIT_POS

指令形式：

WAIT_POS(时长,变量);

1.4.1.2 指令功能：

等待运动指令的执行结束，如果设置了等待时长，可以选择停止等待。

1.4.1.3 参数说明：

表 1-26 指令参数列表

参数序号	参数名称	默认值	说明
1	等待时长		
2	变量		正确终止情况下该值被置为 0

1.4.1.4 使用用例

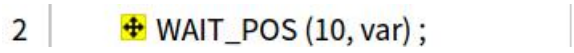
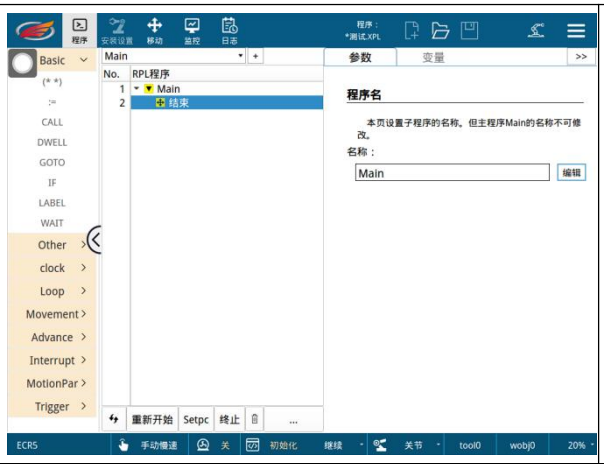
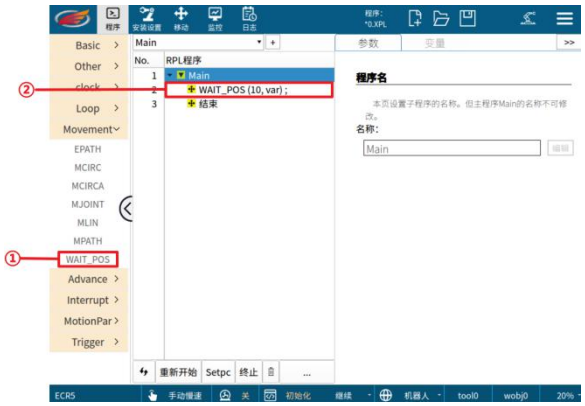
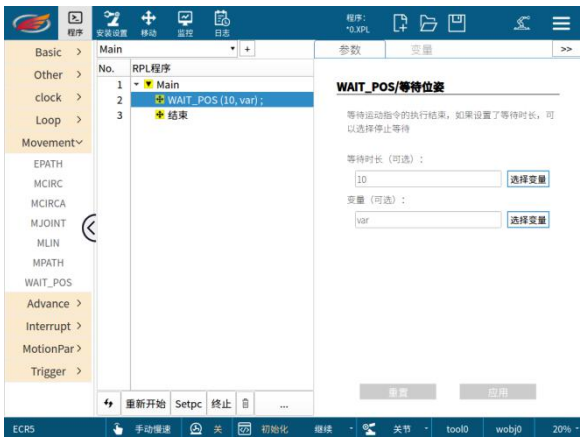
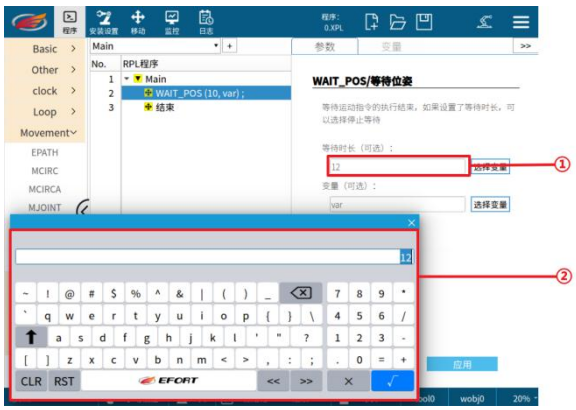


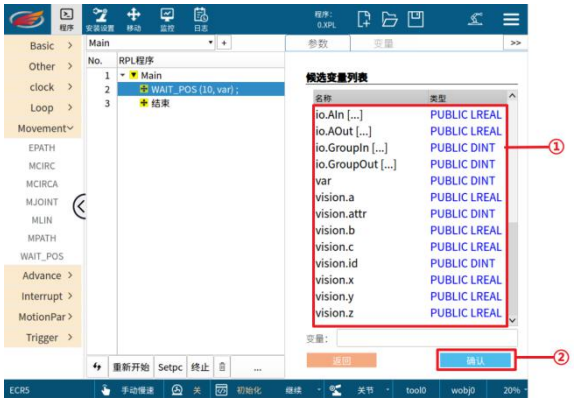
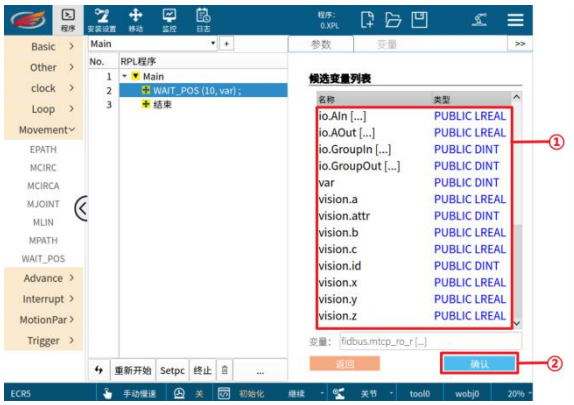
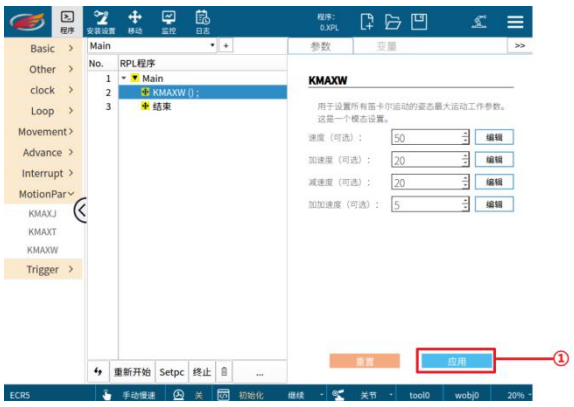
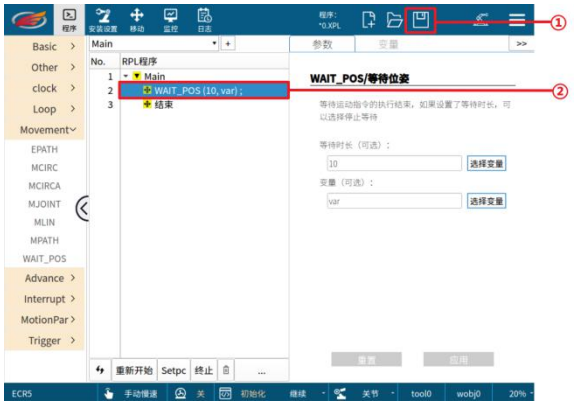
图 1-13 等待位姿指令使用用例

1.4.1.5 等待位姿 / WAIT_POS 指令添加和设置步骤

表 1-27 等待位姿 /WAIT_POS 指令添加和设置步骤

步骤	图示	说明

1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“运动/Movement”中找到等待位姿/wait_pos指令，点击 WAIT_POS 指令，向程序树中添加等待位姿指令		1. 若一级菜单“运动/Movement”未展开，点击一级菜单“运动/Movement”。
3.点击程序树中刚刚添加的等待位姿指令行。		1.点击程序树中的等待位姿指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。
4.对于等待位姿的第一个参数设置，可以直接点击输入框进行设置		

5.第一个参数的设置还可以通过点击选择变量按钮进行		点击确认使变量选择生效
6.点击参数二的选择变量按钮，进入候选变量列表选择变量		点击确认使变量选择生效
7.点击“应用”按钮将修改保存到程序树中		程序树中被选中的等待位姿指令行的内容在点击应用后变为编辑内容。
8.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.4.2 关节运动 / MJOINT

1.4.2.1 指令基本信息

指令名称：关节运动

指令形式：

MJOINT(目标点,速度,区域,工具,[参考坐标系]);

1.4.2.2 指令功能：

该开始关节运动，从上个点到达此目标点。目标点的位置可以是 POINTC 或者 POINTJ。所有轴开始运动并同时到达目标点。TCP 运动是机器人运动各轴的组合。如果参考坐标系未指定，默认在世界坐标系下运动。

警告 :只有在前一个动作结束且第二个动作开始之前 ,才有可能在两个连续动作之间更改工具。否则将发生错误。有关的更多细节，请参见 MLIN 指令中的示例。

1.4.2.3 参数说明：

表 1-28 指令参数列表

参数序号	参数名称	默认值	说明
1	速度	v500	下拉选择速度
2	圆滑过渡	fine	下拉选择圆滑过渡
3	工具坐标系	tool0	下拉选择工具坐标系
4	用户坐标系	wobj0	下拉选择用户坐标系
5	目标点	target	关节运动的目标点

1.4.2.4 使用用例

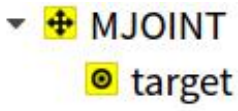
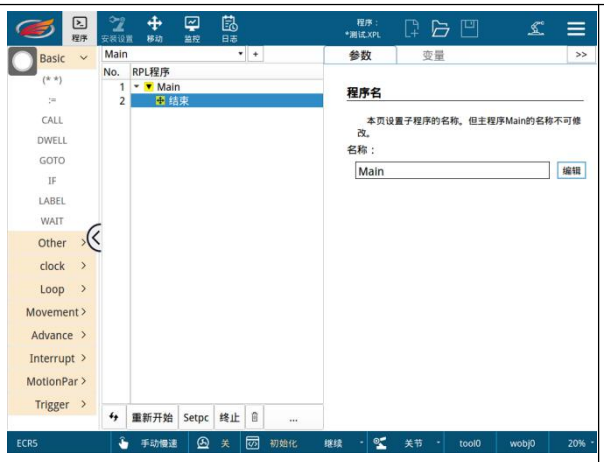
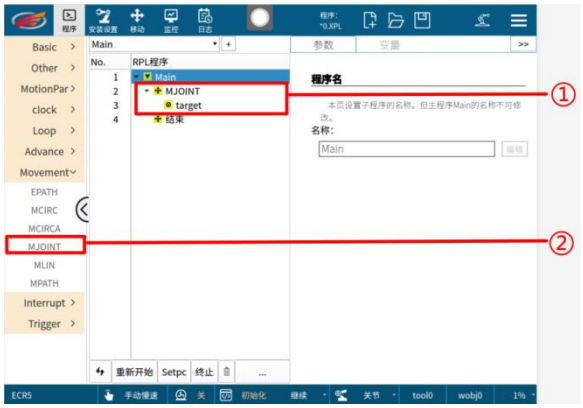
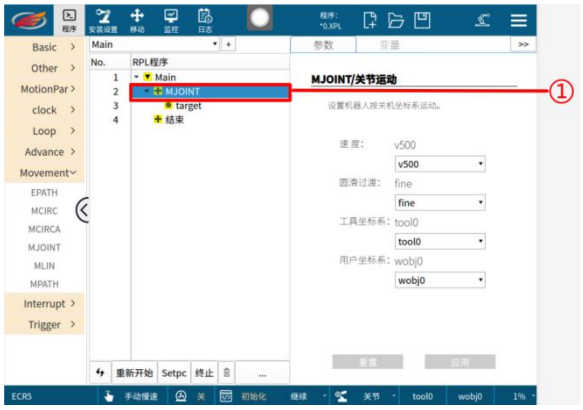
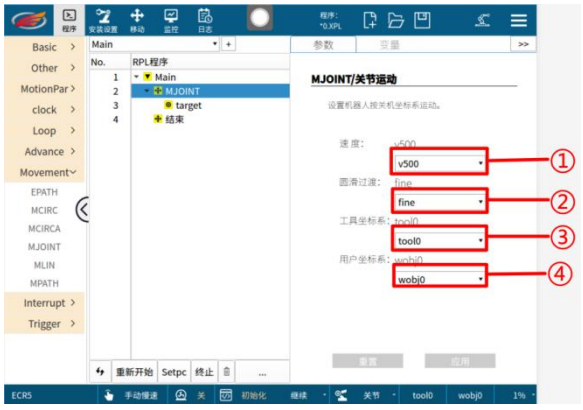


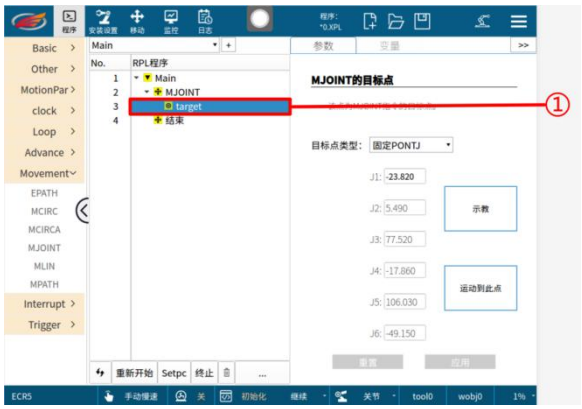
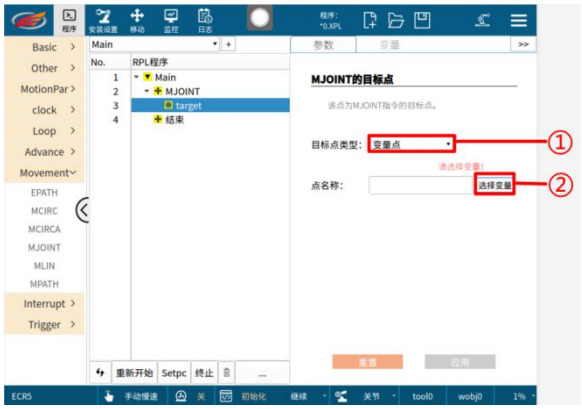
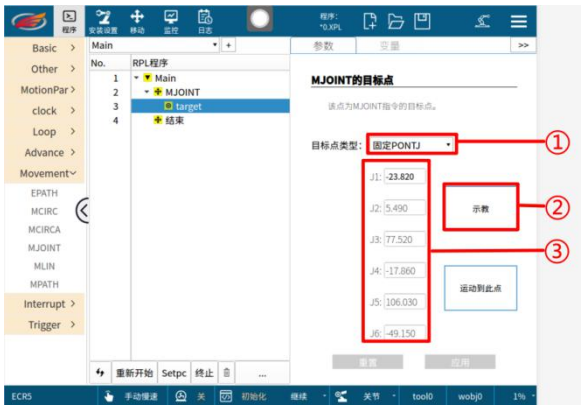
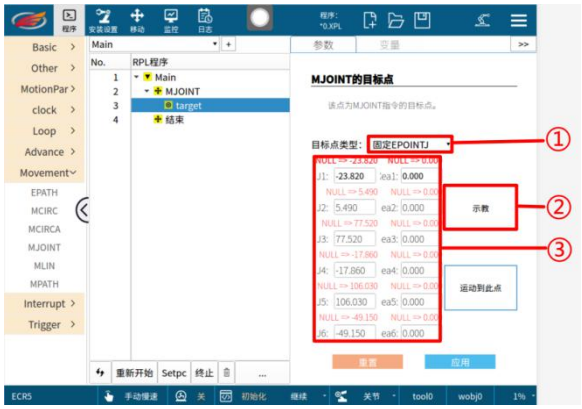
图 1-14 MJOINT 指令使用用例

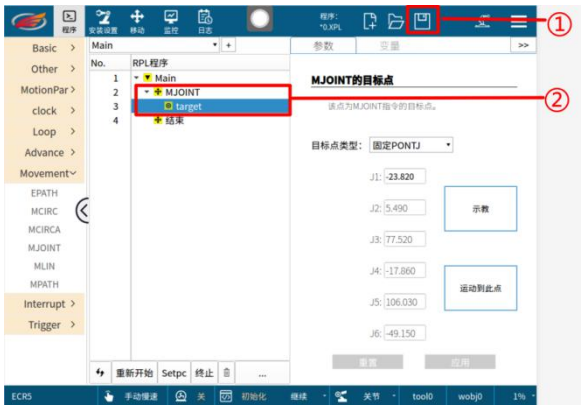
1.4.2.5 指令添加和设置步骤

表 1-29 MJOINT 指令添加和设置步骤

步骤	图示	说明

1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“运动/Movement”中找到MJOINT指令，点击指令，向程序树中添加MJOINT指令		若一级菜单“运动/Movement”未展开，点击一级菜单“运动/Movement”
3.点击程序树中刚添加的MJOINT指令行。		点击程序树中的MJOINT行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示MJOINT指令的参数树详情页。
4. 参数页中的四个参数分别为速度①，圆滑过渡②，工具坐标系③，用户坐标系④,可通过下拉选择的方式进行设置。 5. 选择完成后点击“应用”按钮保存当前设置。		

6.点击程序树中添加的关节运动指令的目标点①处。		目标点的类型共有 5 种（变量点，固定 POINTJ，固定 POINTC，固定 EPOINTJ，固定 EPOINTC）。
7 目标点类型为“变量点”时，点击选择变量		右侧参数页面展示候选参数列表页，选择合适的变量后点击确认保存。
8.目标点类型①为“固定 POINTC”或“固定 POINTJ”时，点击示教按钮②将当前点数值记录进输入框③内，或点击输入框③手动编辑数值。		输入框的数值全部设置完成后，点击应用按钮④保存此设置。
9.目标点类型①为“固定 EPOINTC”或“固定 EPOINTJ”时，点击示教按钮②将当前点数值记录进输入框③内，或点击输入框③手动编辑数值。		输入框的数值全部设置完成后，点击应用按钮保存此设置。

10.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.4.3 角圆弧运动 / MCIRCA

1.4.3.1 指令基本信息

指令名称：MCIRCA

指令形式：

MCIRCA(中间点,目标点,速度,圆滑过渡,工具坐标系,用户坐标系)

1.4.3.2 指令功能：

执行以确定的移动角度执行 TCP 的圆周(笛卡尔)运动。圆周由三个点定义:圆周运动前的最终位置、目标位置、中间点。当机器人走完期望角时，运动结束。正角度是指运动方向为从中间点到目标点的圆周运动。负角度是指运动方向为从目标点到中间点的圆周运动。

注意：如果圆周运动的起始点和目标点是同一个点，圆周运动将是不可知的。

警告: 因为实际位置是用来描述从这个指令开始执行圆周运动，如果开始时 TCP 远离了程序中的位置，可能会有危险。

警告：只有在前一个动作结束且第二个动作开始之前，才有可能在两个连续动作之间更改工具。否则将发生错误。有关的更多细节，请参见 MLIN 指令中的示例。

在运动过程中，相对于圆弧的方向一直保持为初始设置的方向。此操作模式与 MCRIC 指令中使用的方向不同，如果角度参数较大，则有可能到达一个或多个机器人轴的行程终点。

如果省略 refsys 参数，则使用世界坐标系进行移动。

1.4.3.3 参数说明：

表 1-30 指令参数列表

参数序号	参数名称	默认值	说明
1	角度	空	
2	速度	V500	
3	圆滑过渡	fine	

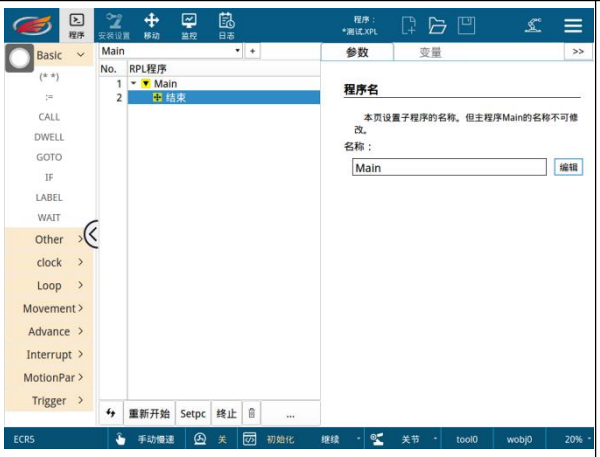
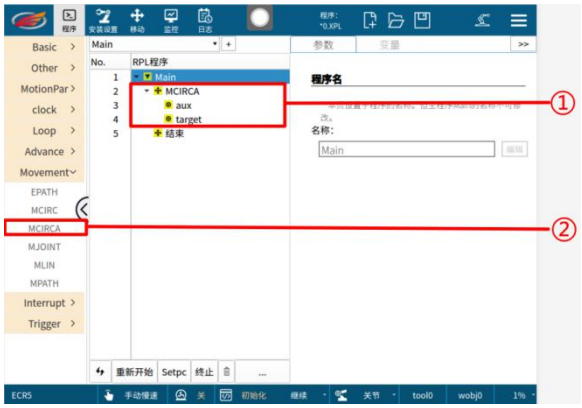
4	工具坐标系	tool0	
5	世界坐标系	wobj0	

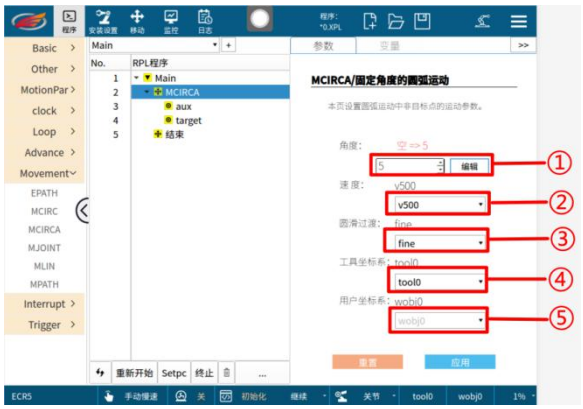
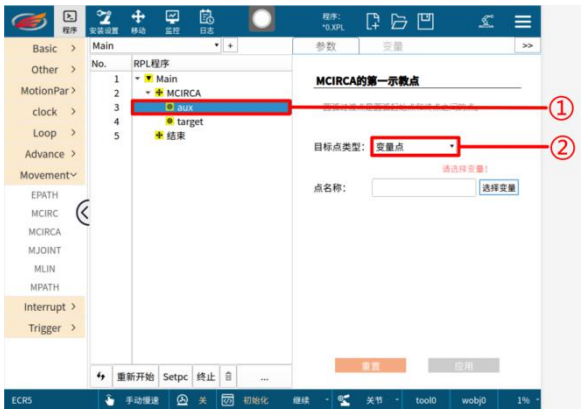
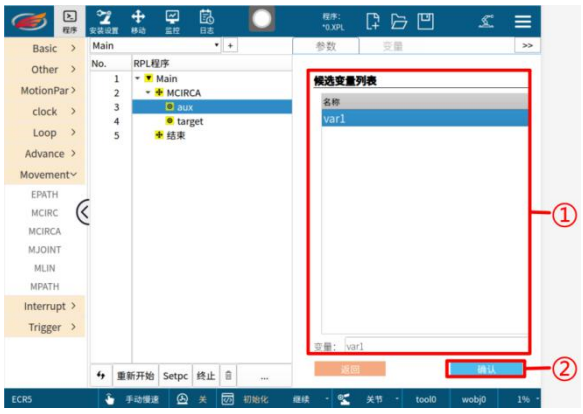
1.4.3.4 使用用例

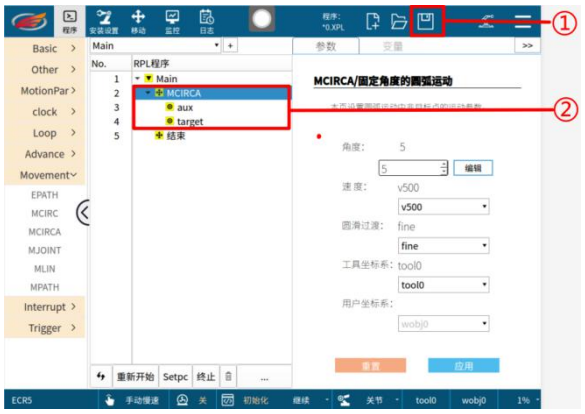
图 1-15 MCIRCA 指令使用用例

1.4.3.5 指令添加和设置步骤

表 1-31 MCIRCA 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“运动/Movement”中找到MCIRCA指令，点击指令，向程序树中添加该指令		1.若一级菜单“运动/Movement”未展开，点击一级菜单“运动/Movement”。

3.点击程序树中刚刚添加的 MCIRCA 指令行。		1. 点击程序树中的 MCIRCA 指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.此处可以设置 MCIRCA 的后 5 个参数，设置后需要点击应用以保存修改。
4.点击程序树中添加的 MCIRCA 指令的中间过渡点(或目标点)		中间过渡点(或目标点)的类型共有三种（变量点，固定 POINTC，固定 EPOINTC）。
5.目标点类型为“变量点”时，点击选择变量		右侧参数页面展示候选参数列表页，选择合适的变量后点击确认保存。
6.目标点类型为“固定 POINTC”时，点击示教按钮将当前点数值填入输入框内，或点击输入框手动编辑数值。		X,Y,Z,A,B,C 的数值全部设置完成后，点击应用保存此设置。

7.目标点类型为“固定 EPOINTC”时，点击示教按钮将当前点数值填入输入框内，或点击输入框手动编辑数值。		X,Y,Z,A,B,C 的数值全部设置完成后，点击应用保存此设置。
8.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.4.4 路径执行 / EPATH

1.4.4.1 指令基本信息

指令名称：路径执行

指令形式：

EPATH

name

1.4.4.2 指令功能：

用于启动一条路径的执行。如果想要执行路径指令，必须在键入此指令前在 plib 文件下导入期望的.pth 路径文件，并将 path 路径作为外部变量。

1.4.4.3 参数说明：

表 1-32 指令参数列表

参数序号	参数名称	默认值	说明
1	速度	v500	
2	圆滑过渡	fine	
3	工具坐标系	tool0	
4	用户坐标系	wobj0	

5	目标路径		EPATH 指令中将要执行的指令

1.4.4.4 使用用例

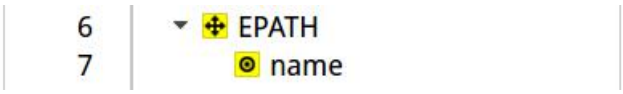
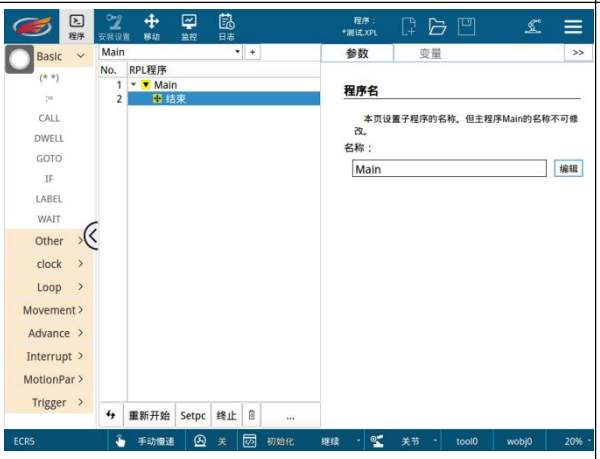
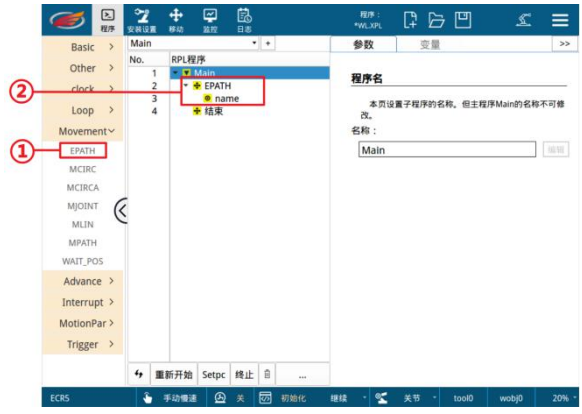
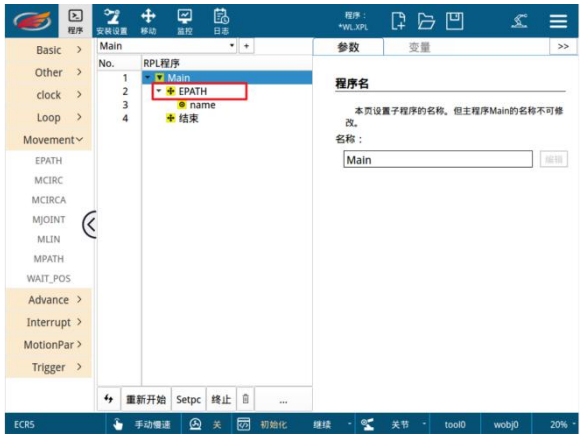
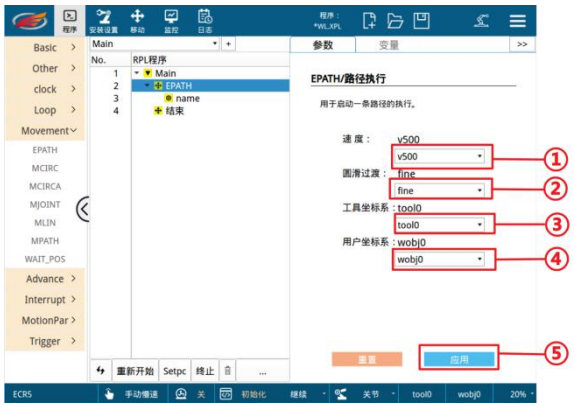
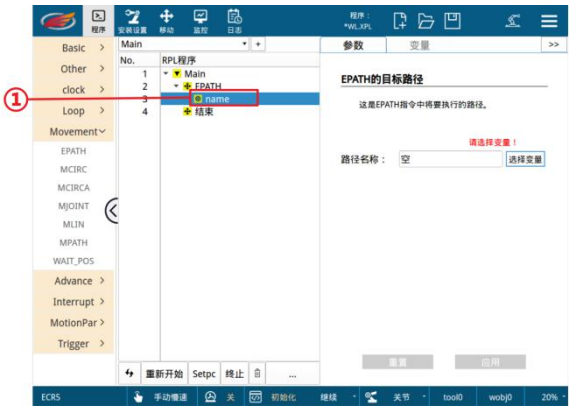
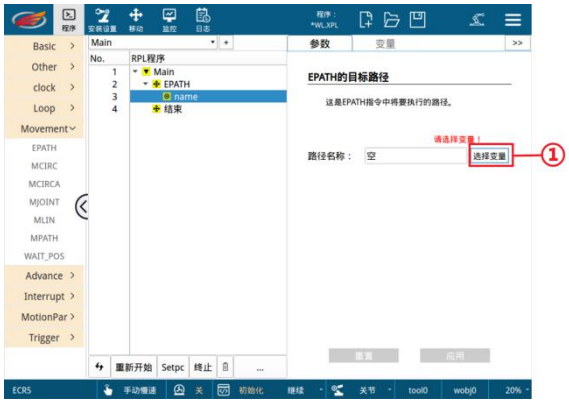
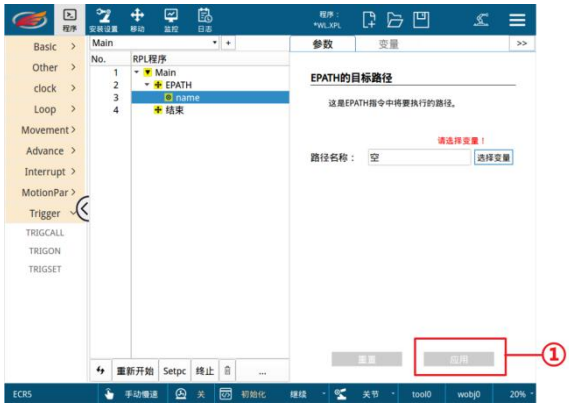


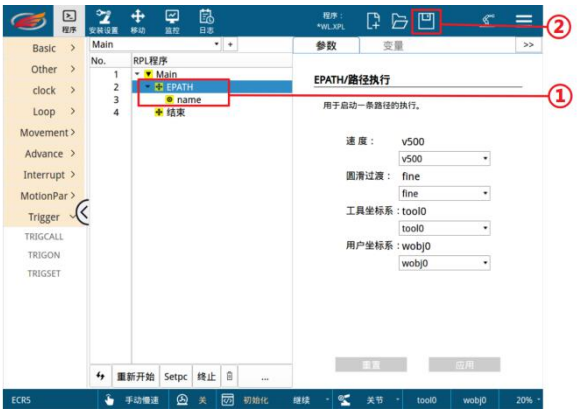
图 1-16 EPATH 指令使用用例

1.4.4.5 路径执行 / EPATH 指令添加和设置步骤

表 1-33 路径执行/EPATH 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“运动/Movement”中找到路径执行/EPATH 指令，点击 EPATH 指令，向程序树中添加 EPATH 指令		若一级菜单“运动/Movement”未展开，点击一级菜单“运动/Movement”
3.点击程序树中刚添加的 EPATH 指令行。		点击程序树中的 EPATH 行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示 EPATH 指令的参数树详情页。 EPATH 指令默认参数分别为速度：v500；圆滑过渡：fine；工具坐标系：

		tool0；用户坐标系：wobj0
4.编辑参数页中的四个参数，分别为速度，圆滑过渡，工具坐标系，用户坐标系。 5.编辑完成后点击“应用”按钮。		
6.点击程序树中添加的路径执行指令的目标路径（name）		
7.点击右边参数页的“选择变量”按钮，跳转到候选变量列表		右侧参数页面展示候选参数列表页，选择合适的变量后点击确认保存。
8.点击“应用”按钮将修改保存到程序树中		程序树中被选中的跳转指令行的参数无法点击手动输入。

9.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.4.5 圆弧运动 / MCIRC

1.4.5.1 指令基本信息

指令名称：MCIRC

指令形式：

MCIRC(中间点,目标点,速度,圆滑过渡,工具坐标系,世界坐标系)

1.4.5.2 指令功能：

执行从圆弧运动开始前的最后一个位置经过一个中间点到目标位置的圆弧运动。

1.4.5.3 参数说明：

表 1-34 指令参数列表

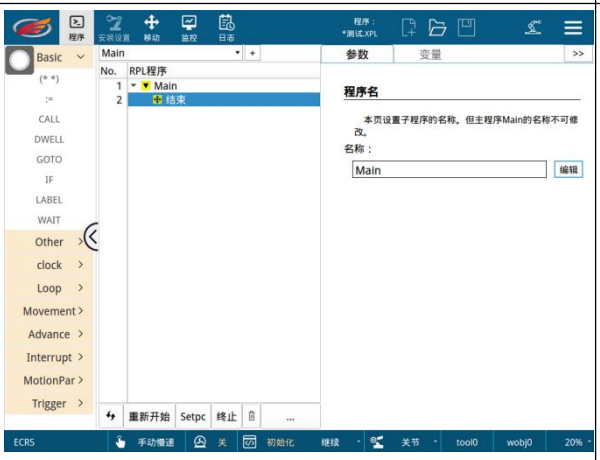
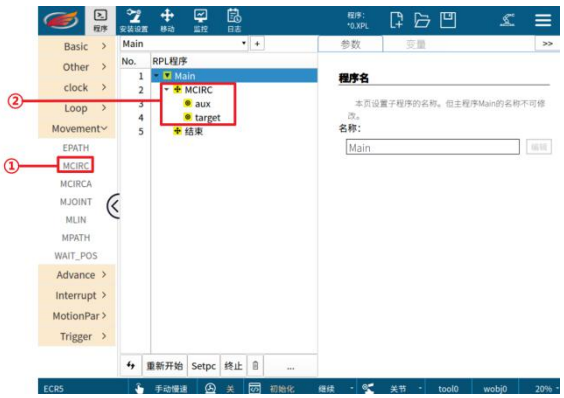
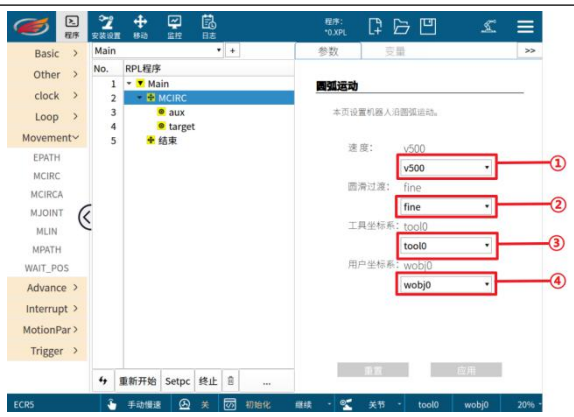
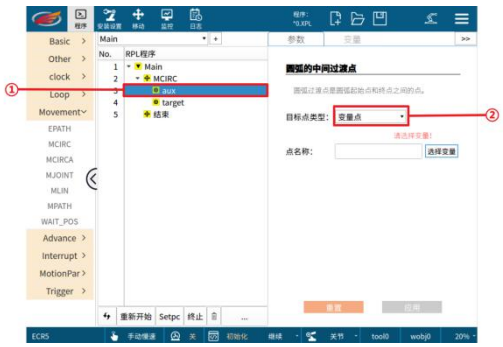
参数序号	参数名称	默认值	说明
1	中间点	当前点	
2	目标点	当前点	起始点与目标点不能是同一个点
3	速度	V500	
4	圆滑过渡	fine	
5	工具坐标系	tool0	
6	世界坐标系	wobj0	

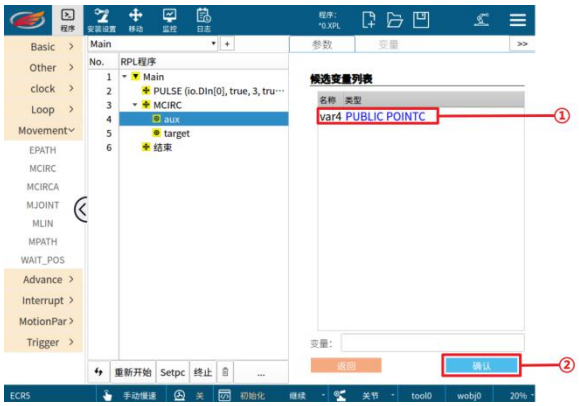
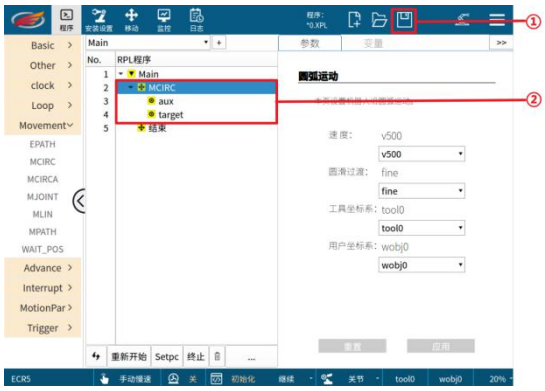
1.4.5.4 使用用例

图 1-17 圆弧运动指令使用用例

1.4.5.5 圆弧运动 / MCIRC 指令添加和设置步骤

表 1-35 圆弧运动 /MCIRC 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“运动/Movement”中找到圆弧运动/MCIRC指令，点击 MCIRC 指令，向程序树中添加圆弧运动指令		1.若一级菜单“运动/Movement”未展开，点击一级菜单“运动/Movement”。
3.点击程序树中刚刚添加的圆弧运动指令行。		1.点击程序树中的圆弧运动指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.此处可以设置圆弧运动的后四个参数，设置后需要点击应用以保存修改。
4.点击程序树中添加的圆弧运动指令的中间过渡点(或目标点)		中间过渡点(或目标点)的类型共有三种（变量点，固定 POINTC，固定 EPOINTC）。

5.目标点类型为“变量点”时，点击选择变量		右侧参数页面展示候选参数列表页，选择合适的变量后点击确认保存。
6.目标点类型为“固定 POINTC”时，点击示教按钮将当前点数值填入输入框内，或点击输入框手动编辑数值。		X,Y,Z,A,B,C 的数值全部设置完成后，点击应用保存此设置。
7.目标点类型为“固定 EPOINTC”时，点击示教按钮将当前点数值填入输入框内，或点击输入框手动编辑数值。		X,Y,Z,A,B,C 的数值全部设置完成后，点击应用保存此设置。
8.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.4.6 直线运动 / MLIN

1.4.6.1 指令基本信息

指令名称：直线运动

指令形式：

MLIN

target

1.4.6.2 指令功能：

该指令可以使机器人从上一个位姿点直线运动到目标位姿点。如果在运动过程中要求改变姿态，姿态轴 A，B，C 将会被插补，并且与位置点保持同时插补。如果参考坐标系未指定，默认在世界坐标系下运动。

只有在前一个动作结束且第二个动作开始之前，才有可能在两个连续动作之间更改工具。否则将发生错误。

1.4.6.3 参数说明：

表 1-36 指令参数列表

参数序号	参数名称	默认值	说明
1	速度	v500	
2	圆滑过渡	fine	
3	工具坐标系	tool0	
4	用户坐标系	wobj0	
5	目标点	target	直线运动的目标点

1.4.6.4 使用用例

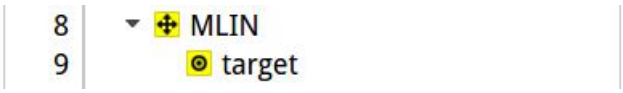
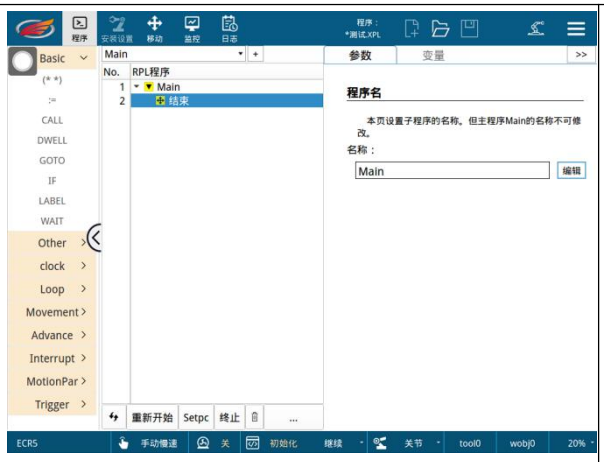
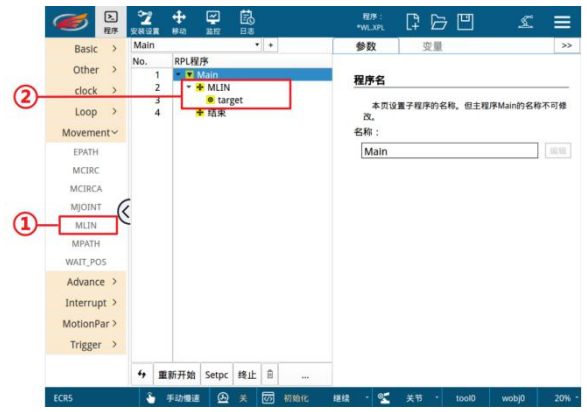
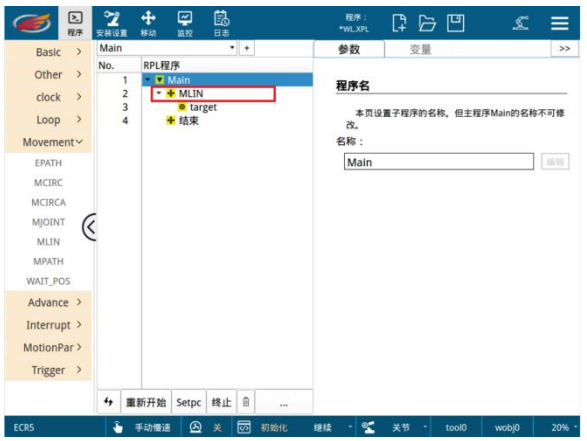
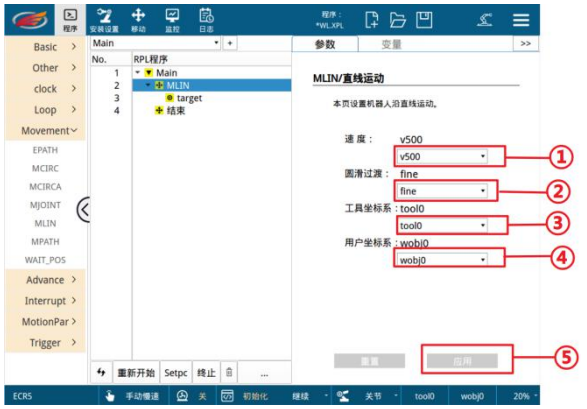


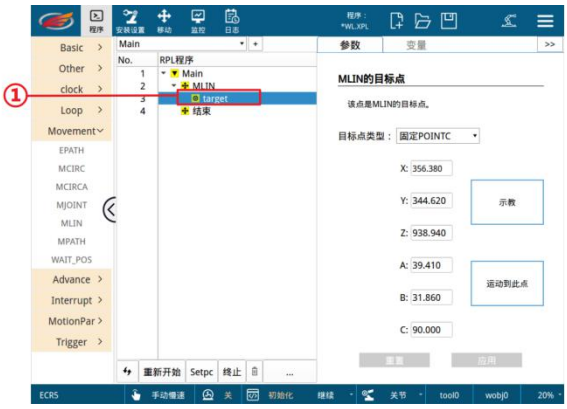
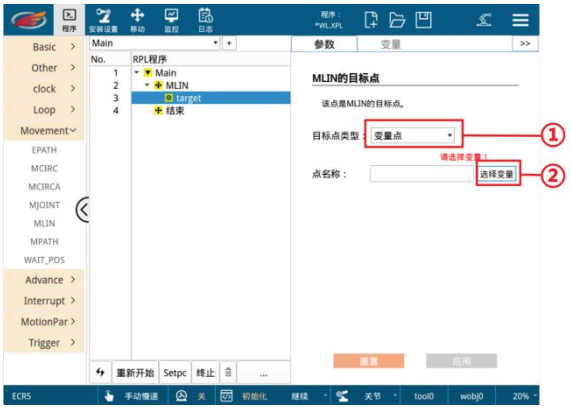
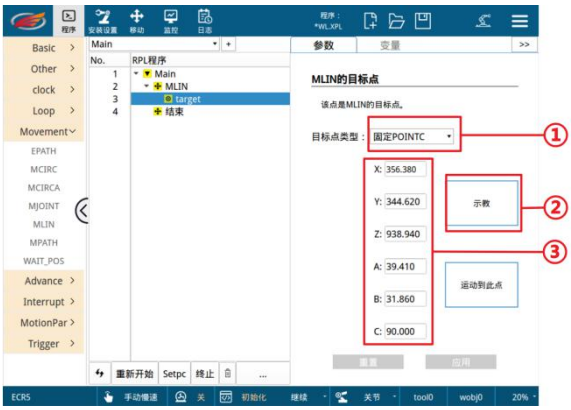
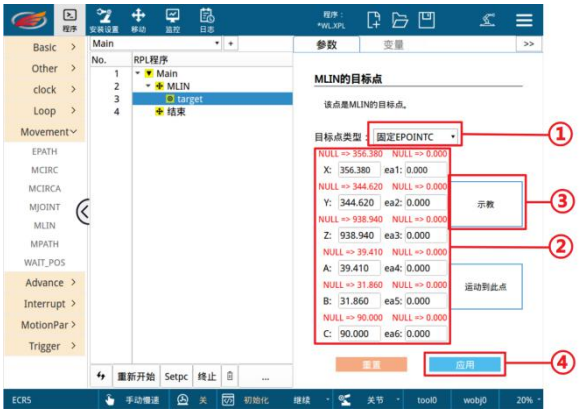
图 1-18 MLIN 指令使用用例

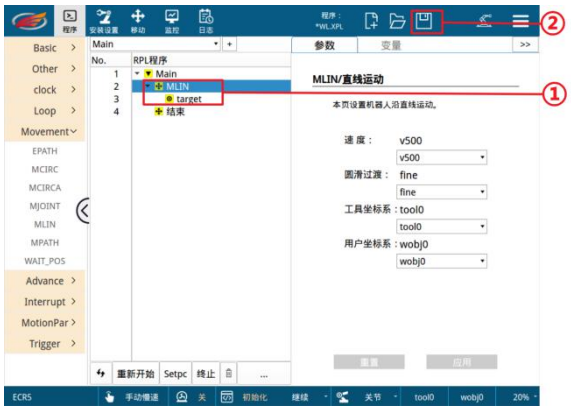
1.4.6.5 直线运动 / MLIN 指令添加和设置步骤

表 1-37 直线运动/ MLIN 指令添加和设置步骤

步骤	图示	说明

1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“运动/Movement”中找到直线运动/ MLIN 指令，点击 MLIN 指令，向程序树中添加 MLIN 指令		若一级菜单“运动/Movement”未展开，点击一级菜单“运动/Movement”
3.点击程序树中刚添加的 MLIN 指令行。		点击程序树中的 MLIN 行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页，并显示 MLIN 指令的参数树详情页。 MLIN 指令默认参数分别为速度：v500；圆滑过渡：fine；工具坐标系：tool0；用户坐标系：wobj0
6. 编辑参数页中的四个参数，分别为速度，圆滑过渡，工具坐标系，用户坐标系。 7. 编辑完成后点击“应用”按钮。		

6.点击程序树中添加的直线运动指令的目标点		目标点的类型共有三种（变量点，固定POINTC，固定EPOINTC）。
7 目标点类型为“变量点”时，点击选择变量		右侧参数页面展示候选参数列表页，选择合适的变量后点击确认保存。
8.目标点类型为“固定 POINTC”时，点击示教按钮将当前点数值填入输入框内，或点击输入框手动编辑数值。		X,Y,Z,A,B,C 的数值全部设置完成后，点击应用保存此设置。
9.目标点类型为“固定 EPOINTC”时，点击示教按钮将当前点数值填入输入框内，或点击输入框手动编辑数值。		X,Y,Z,A,B,C 的数值全部设置完成后，点击应用保存此设置。

10.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.5 运动参数/ MotionPar

1.5.1 最大关节运动 / KMAXJ

1.5.1.1 指令基本信息

指令名称：KMAXJ

指令形式：

KMAXJ(速度,加速度,减速度,加加速度)

1.5.1.2 指令功能：

设置关节运动参数的最大值。此指令是一个模态设置。

关节空间的运动参数以百分比的形式设置，参考机器人配置的最大关节速度 MAX_SPE_J，最大关节加速度 MAX_ACC_J 和最大关节加加速度 MAX_JER_J。所有参数是可选的。

1.5.1.3 参数说明：

表 1-38 指令参数列表

参数序号	参数名称	默认值	说明
1	速度		
2	加速度		
3	减速度		
4	加加速度		

1.5.1.4 使用用例

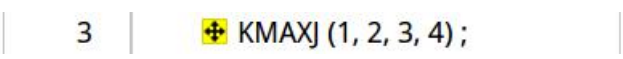
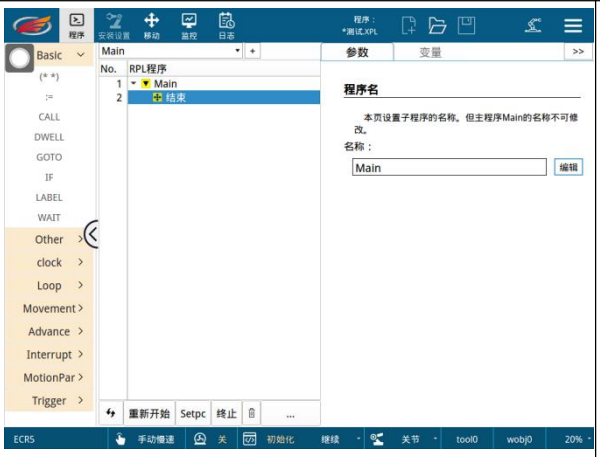
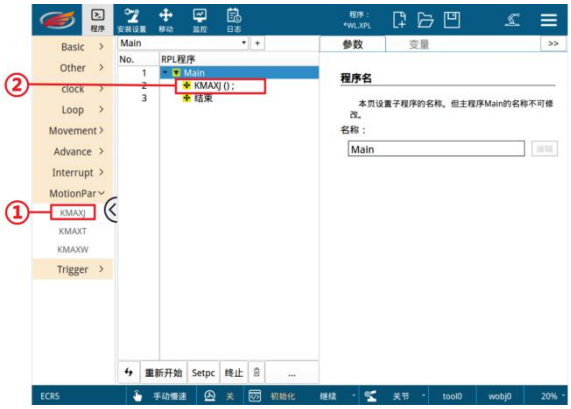
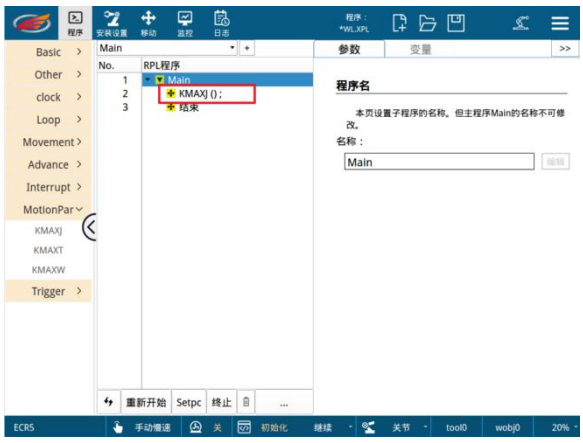
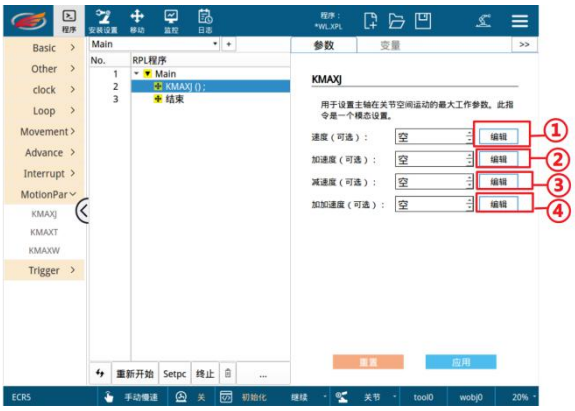
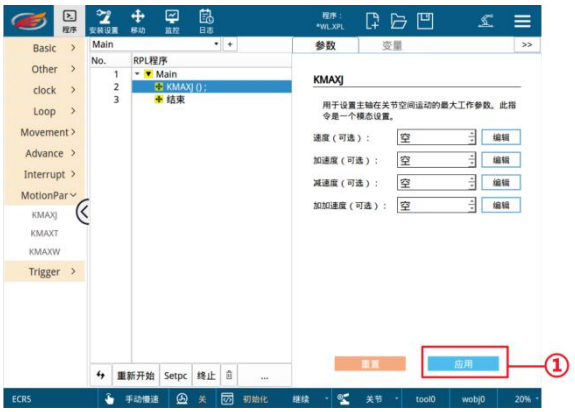
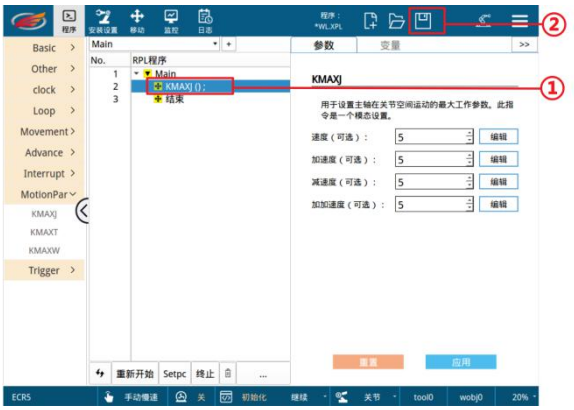


图 1-19 KMAXJ 指令使用用例

1.5.1.5 关节运动最大参数设置/KMAXJ 指令添加和设置步骤

表 1-39 关节运动最大参数设置/KMAXJ 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“运动参数/MotionPar”中找到 KMAXJ 指令，点击 KMAXJ 指令，向程序树中添加 KMAXJ 指令		若一级菜单“运动参数/MotionPar”未展开，点击一级菜单“运动参数/MotionPar”。
3.点击程序树中刚刚添加的 KMAXJ 指令行。		点击程序树中的调用指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 此处可以设置 KMAXJ 的四个参数，设置后需要点击应用以保存修改。

4.点击编辑按钮设置参数，在弹出的表达式编辑框根据实际情况设置具体参数		加速度，减速度与加加速度与速度的设置相同。
5.点击“应用”按钮将修改保存到程序树中		程序树中被选中的KMAXJ 指令行的内容在点击应用后变为输入输入内容
6.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.5.2 最大姿态变化 / KMAXW

1.5.2.1 指令基本信息

指令名称：KMAXW

指令形式：

KMAXW(速度,加速度,减速度,加加速度)

1.5.2.2 指令功能：

用于设置腕部轴在笛卡尔空间运动的最大参数。

1.5.2.3 参数说明：

表 1-40 指令参数列表

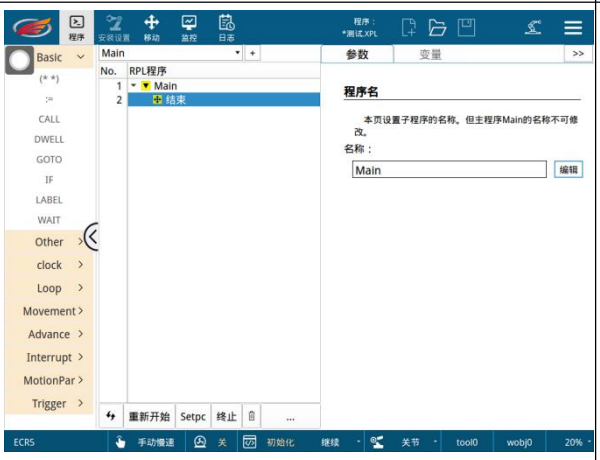
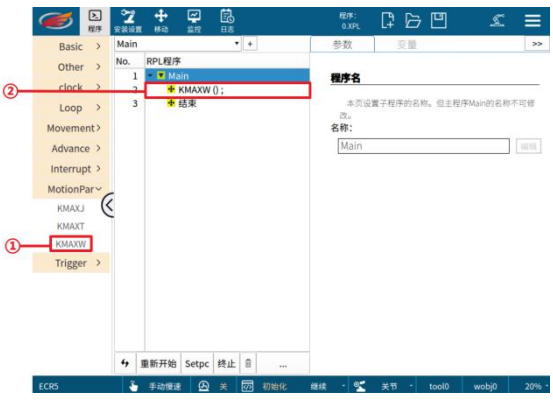
参数序号	参数名称	默认值	说明
1	速度		
2	加速度		
3	减速度		
4	加加速度		

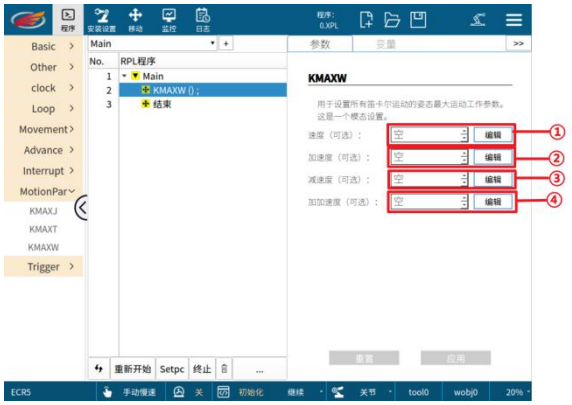
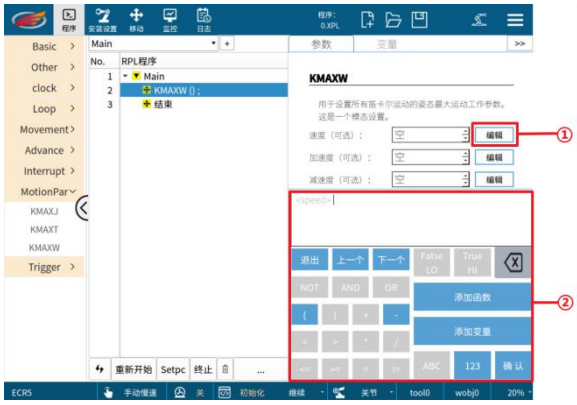
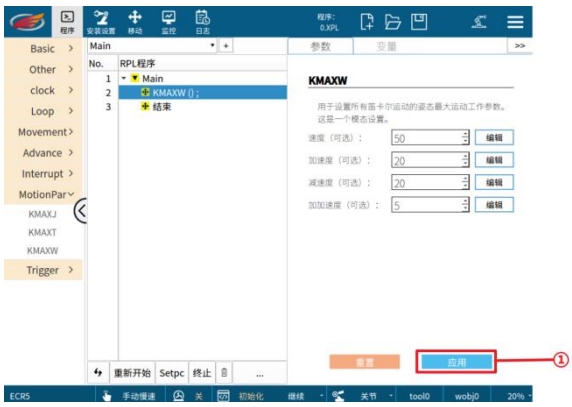
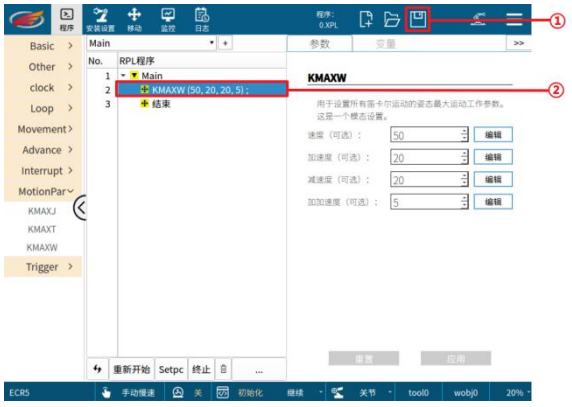
1.5.2.4 使用用例

图 1-20 KMAXW 指令使用用例

1.5.2.5 KMAXW 指令添加和设置步骤

表 1-41 KMAXW 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“运动参数/MotionPar”中找到 KMAXW 指令，点击 KMAXW 指令，向程序树中添加 KMAXW 指令		1.若一级菜单“运动参数/MotionPar”未展开，点击一级菜单“运动参数/MotionPar”。

3.点击程序树中刚刚添加的 KMAXW 指令行。		1. 点击程序树中的 KMAXW 指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.此处可以设置 KMAXW 的四个参数，设置后需要点击应用以保存修改。
4.点击编辑按钮设置参数，在弹出的表达式编辑框根据实际情况设置具体参数		加速度，减速度与加加速度与速度的设置相同。
5.点击“应用”按钮将修改保存到程序树中		程序树中被选中的 KMAXW 指令行的内容在点击应用后变为编辑内容
6.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.5.3 最大坐标运动 / KMAXT

1.5.3.1 指令基本信息

指令名称：KMAXT

指令形式：

KMAXT([speed],[Acceleration],[deceleration],[Jerk]);

1.5.3.2 指令功能

该指令是一个模态设置，用于设置主轴在笛卡尔空间运动的最大运动参数。

1.5.3.3 参数说明

表 1-42 指令参数列表

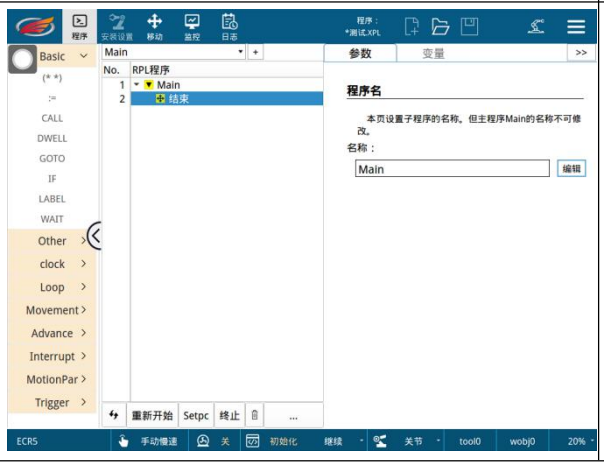
参数序号	参数名称	默认值	说明
1	速度	空	可选项。设置速度
2	加速度	空	可选项。设置加速度
3	减速度	空	可选项。设置减速度
4	加加速度	空	可选项。设置加加速度

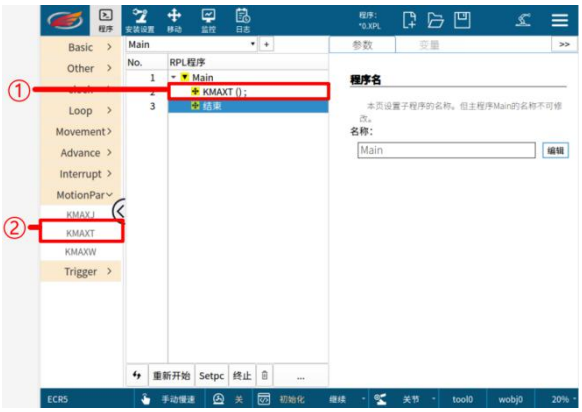
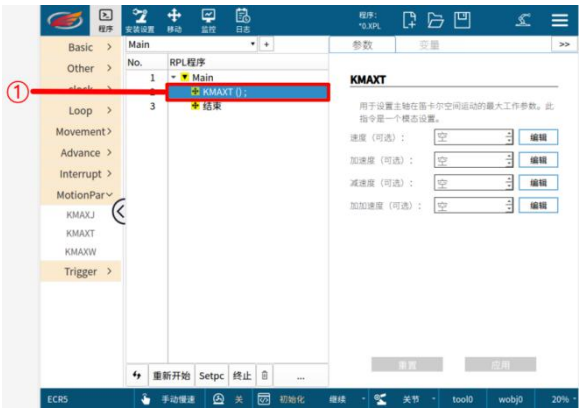
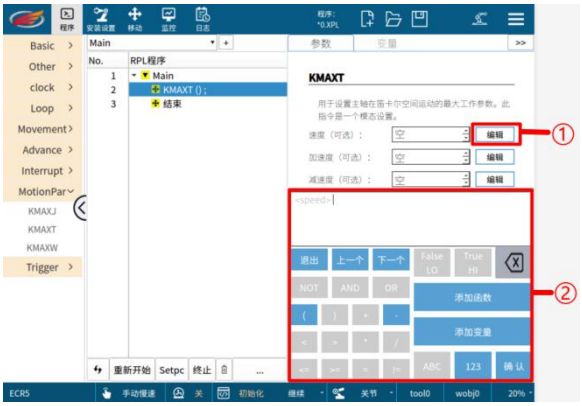
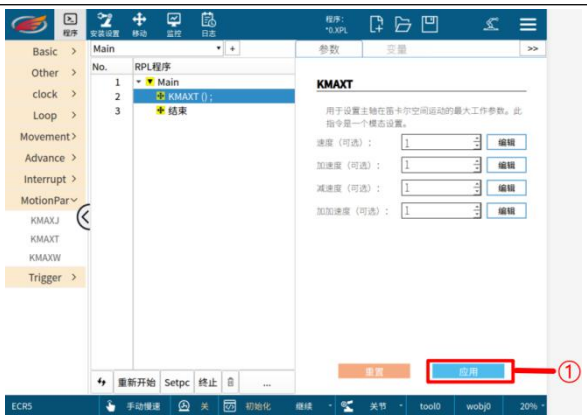
1.5.3.4 使用用例

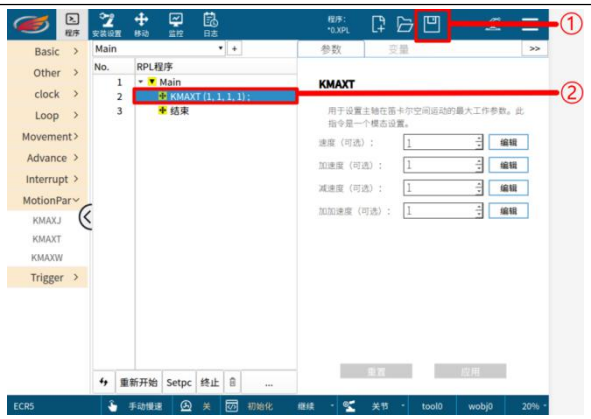
图 1-21 KMAXT 指令使用用例

1.5.3.5 指令添加和设置步骤

表 1-43 KMAXT 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2．在左侧指令栏中一级菜单“运动参数/MotionPar”中找到 KMAXT 指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“运动参数/MotionPar”未展开，点击一级菜单“运动参数/MotionPar”
3.点击程序树中刚刚添加的 KMAXT 指令行，如右图①处。		点击程序树中的 KMAXT 指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示 KMAXT 指令的参数详情。
4.（可选项）点击 KMAXT 指令参数页中“速度”/“加速度”/“减速度”/“加加速度”参数后的“编辑”按钮①，并在弹出的表达式编辑器②中输入对应的值，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时参数后的输入框文本将被设置为在表达式编辑器上编辑的文本。
5.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。

6.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.6 触发/ Trigger

1.6.1 触发调用 /TRIGCALL

1.6.1.1 指令基本信息

指令名称：TRIGCALL

指令形式：

TRIGCALL 触发器名，类型，参考值，子程序（）；

1.6.1.2 指令功能：

设置触发器类型的变量，当触发触发器的条件时，调用特定的程序。被调用子程序不能含有输入或输出参数。

1.6.1.3 参数说明：

表 1-44 指令参数列表

参数序号	参数名称	默认值	说明
1	变量		数据类型：TRIGGER
2	类型	错误	数据类型：枚举，如果是距离触发，单位为毫米。如果是时间触发，单位为秒。
3	条件表达式		
4	子程序		必须没有输入和输出

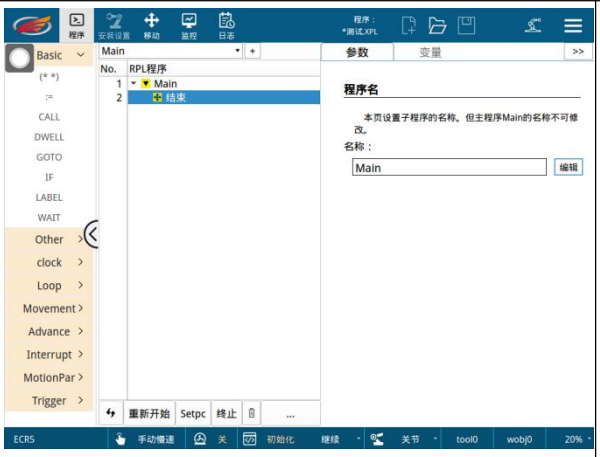
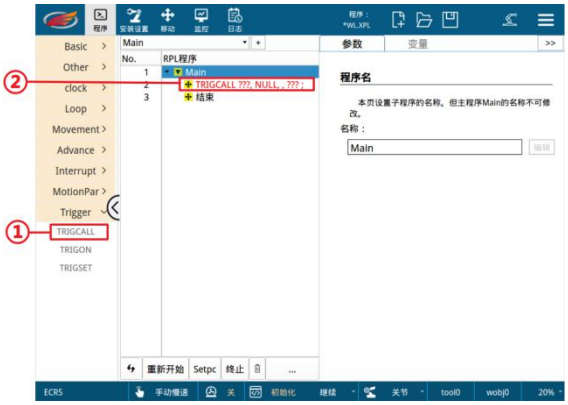
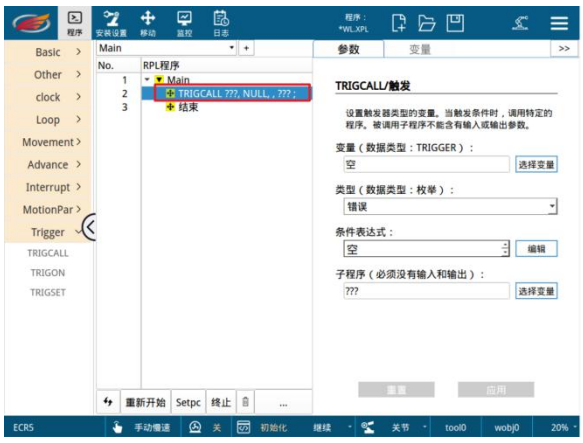
1.6.1.4 使用用例

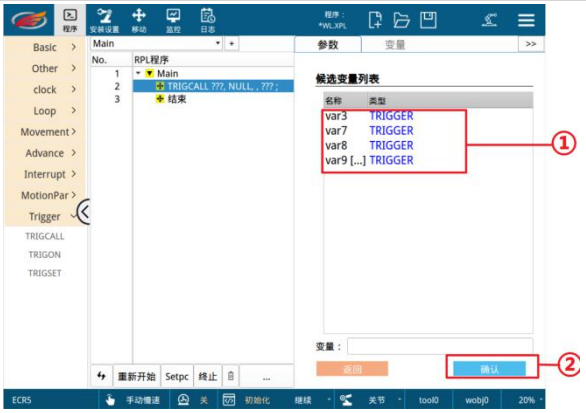
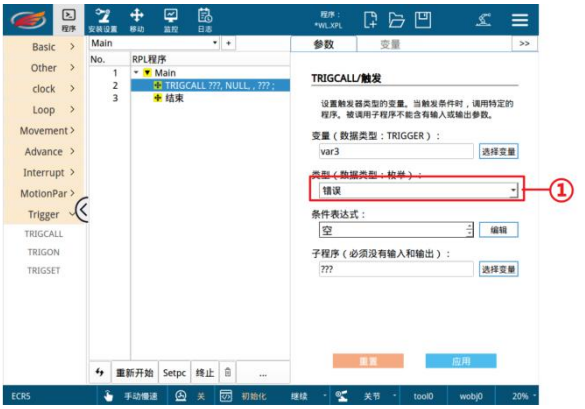
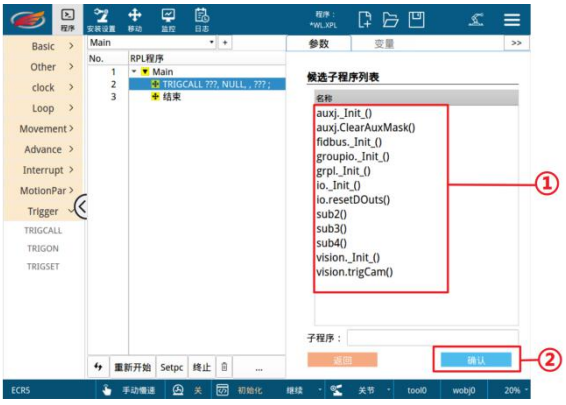
```
4   TRIGCALL var3, TimeToEnd, var9, sub2() ;
5   TRIGON (var3) ;
6   MLIN
7     target
```

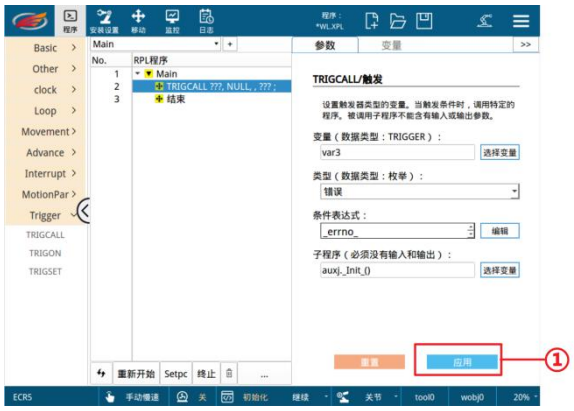
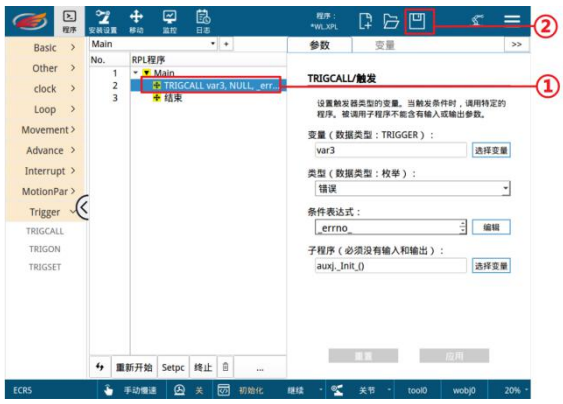
图 1-22 触发指令使用用例

1.6.1.5 触发 /TRIGCALL 指令添加和设置步骤

表 1-45 触发 /TRIGCALL 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“触发/Trigger”中找到TRIGCALL指令，点击 TRIGCALL 指令，向程序树中添加 TRIGCALL 指令		若一级菜单“触发/Trigger”未展开，点击一级菜单“触发/Trigger”。
3.点击程序树中刚刚添加的TRIGCALL指令行。		点击程序树中的调用指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 此处可以设置TRIGCALL的四个参数，设置后需要点击应用以保存修改。

4. 点击“选择变量”按钮设置第一个参数，在弹出的候选变量列表根据实际情况设置具体参数 5. 设置完成后点击“确认”按钮		该指令四个参数均不可以直接点击参数框手动输入值
6.点击第二个参数的下拉框选择数据类型		
7.点击第三个参数后的“编辑”按钮编辑条件表达式 8.编辑完成后点击“确认”按钮		
9.点击第四个参数后的“选择变量”按钮选择子程序 10.选择完成后点击“确认”按钮		

11.设置完所以参数后，点击“应用”按钮将修改保存到程序树中		
12.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.6.2 触发设置/ TRIGSET

1.6.2.1 指令基本信息

指令名称：TRIGSET

指令形式：

trigger_var:=TRIGSET when type isref_value do(var_to_set:=expr_to_set)

1.6.2.2 指令功能：

设置触发器类型的变量，当触发触发器的条件时，将使用表达式的值设置指定的变量。

1.6.2.3 参数说明：

表 1-46 指令参数列表

参数序号	参数名称	默认值	说明
1	触发变量		必须选用 TRIGGER 类型
2	条件类型		有 TimeToEnd(时间)和 Diatance(距离)两种
3	条件值		条件类型为时间时，单位为秒；类型为距离时，单位为毫米
4	待设置变量		
5	待设置值		

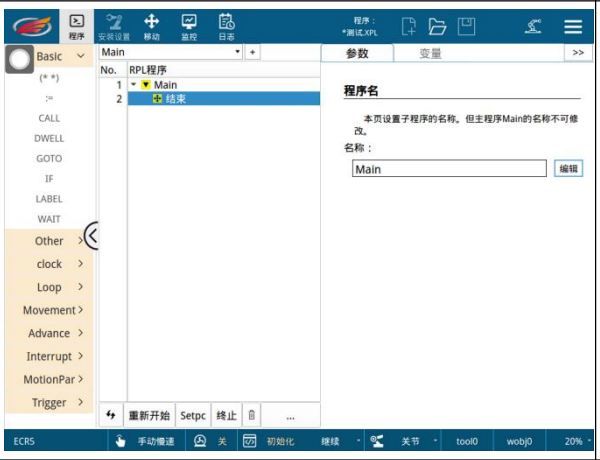
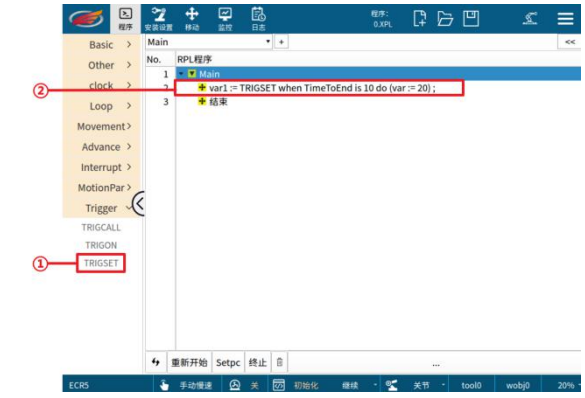
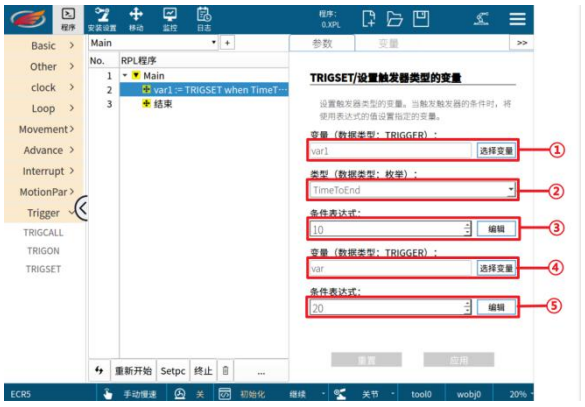
1.6.2.4 使用用例

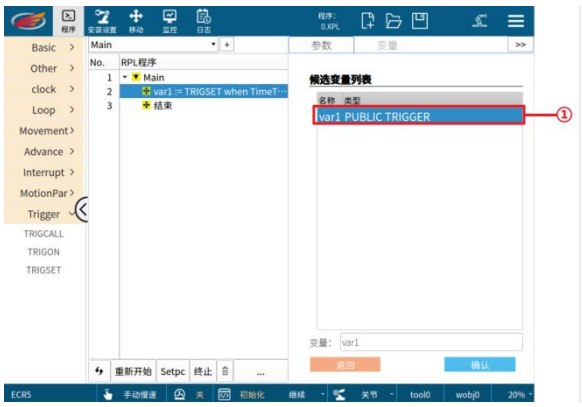
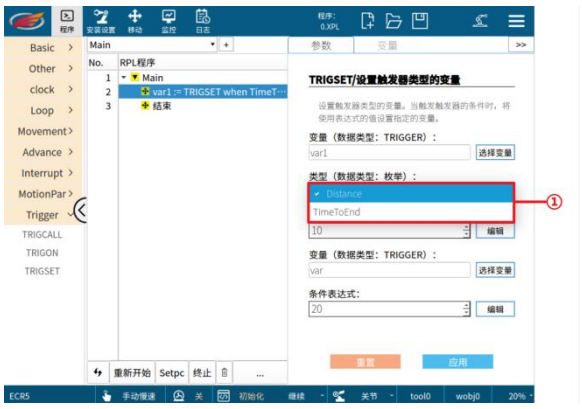
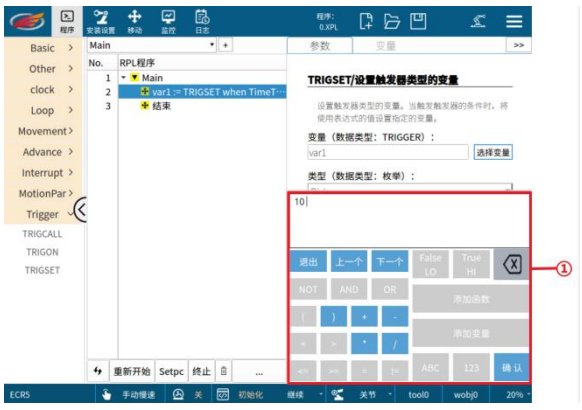
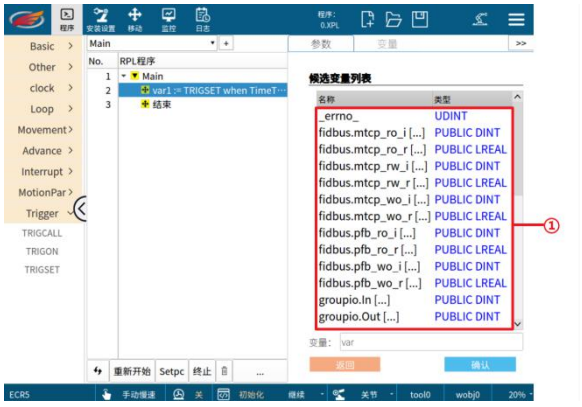
```
2    var1 := TRIGSET when TimeToEnd is 10 do (var := 20) ;
```

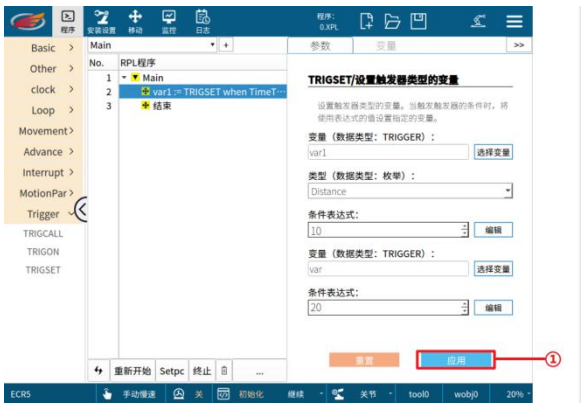
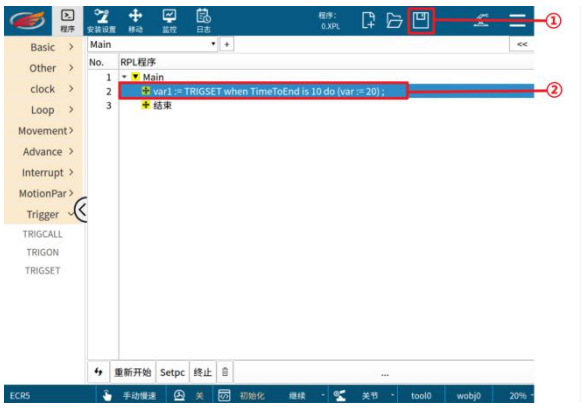
图 1-23 触发设置指令使用用例

1.6.2.5 触发设置 /TRIGSET 指令添加和设置步骤

表 1-47 触发设置 /TRIGSET 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“触发/Trigger”中找到TRIGSET指令，点击 TRIGSET 指令，向程序树中添加TRIGSET指令		1.若一级菜单“触发/Trigger”未展开，点击一级菜单“触发/Trigger”。
3.点击程序树中刚刚添加的触发设置指令行。		1.点击程序树中的触发设置指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.此处可以设置触发设置指令的五个参数，设置后需要点击应用以保存修改。

4.参数一设置：点击选择变量，进入候选变量列表		只能选择 TRIGGER 类型的变量。选择后点击确认以保存。
5.参数二设置：点击下拉框，选择触发条件类型为距离或时间		
6.参数三设置：点击编辑便可弹出表达式编辑框，可根据实际情况选择		参数五的设置类似（条件表达式编辑）
7.参数四设置：点击选择变量按钮，选择需要设置的变量		选择完成后点击确认以保存。

8.点击“应用”按钮将修改保存到程序树中		程序树中被选中的触发设置指令行的内容在点击应用后变为设置内容
9.点击任务栏中“保存”按钮，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.6.3 激活触发/ TRIGON

1.6.3.1 指令基本信息

指令名称：TRIGON

指令形式：

TRIGON(variable1,[variable2],[variable3],[variable4])；

1.6.3.2 指令功能

在下一个运动中激活触发变量。使用 TRIGON 指令最多可以设置 4 个事件。如果多次调用 TRIGON，所有的触发变量将会在运动前的最后一个 TRIGON 指令激活。调用运动指令后，将没有触发变量被设置，因此必须使用多个 TRIGON 指令在下一个运动中来激活触发变量。激活变量只能为 MLIN 和 MCIRC 运动指令使用。其他运动指令 MJOINT 或者 EPATH 不能用于处理事件。

1.6.3.3 参数说明

表 1-48 指令参数列表

参数序号	参数名称	默认值	说明
1	变量 1	空	选择变量 1
2	变量 2	空	可选项。选择变量 2
3	变量 3	空	可选项。选择变量 3
4	变量 4	空	可选项。选择变量 4

1.6.3.4 使用用例

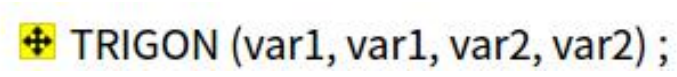
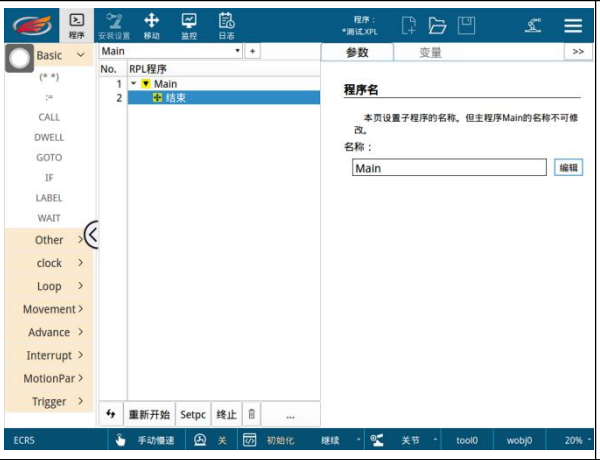
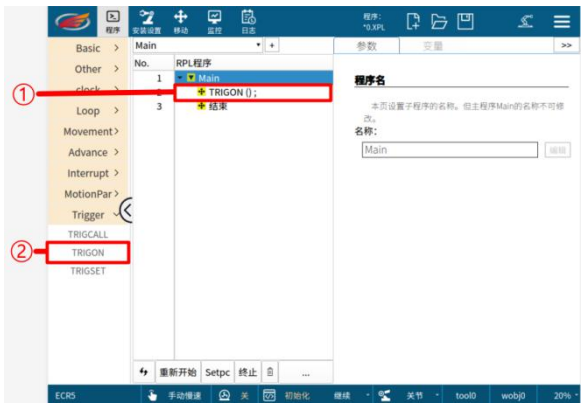
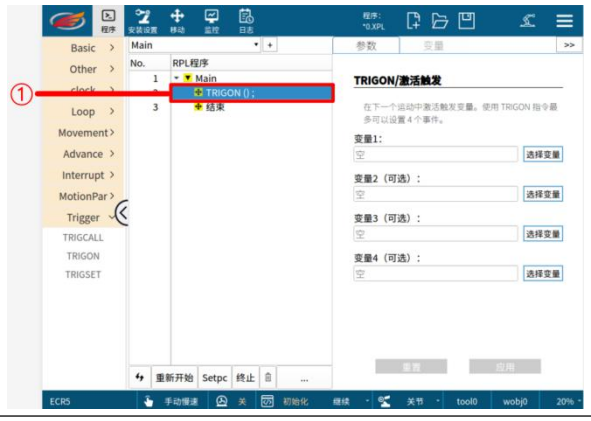
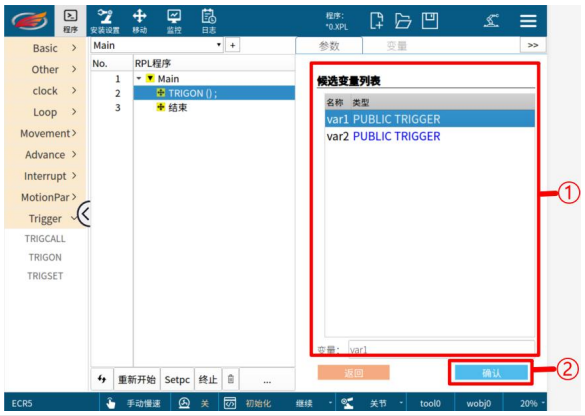
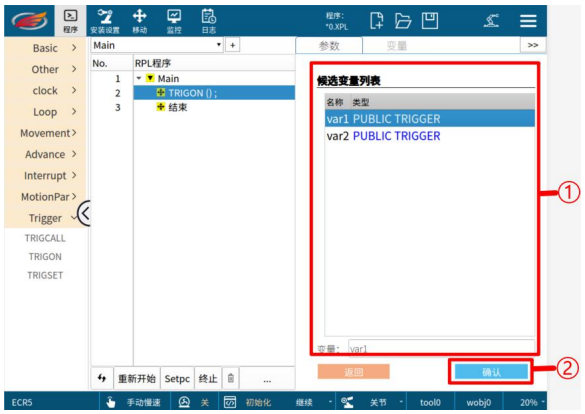
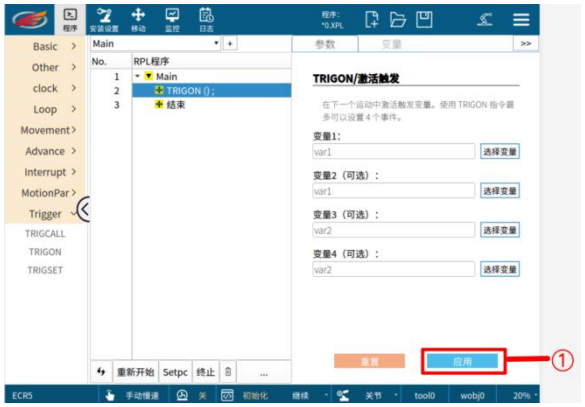


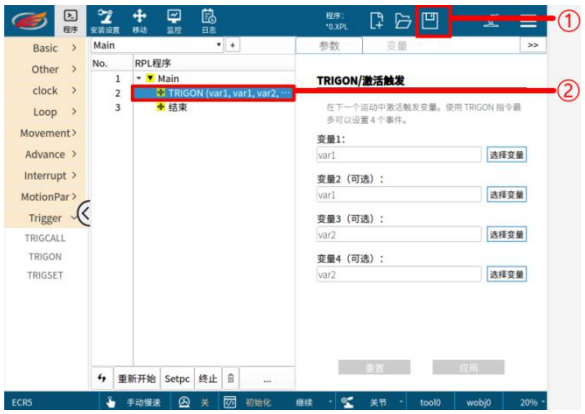
图 1-24 TRIGON 指令使用用例

1.6.3.5 指令添加和设置步骤

表 1-49 TRIGON 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“触发/Trigger”中找到TRIGON指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“触发/Trigger”未展开，点击一级菜单“触发/Trigger”

3.点击程序树中刚刚添加的 TRIGON 指令行，如右图①处。		点击程序树中的 TRIGON 指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示 TRIGON 指令的参数详情。
4.点击 TRIGON 指令参数页中“变量 1”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“变量 1”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
5.（可选项）点击 TRIGON 指令参数页中“变量 2”/“变量 3”/“变量 4”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“变量 2”/“变量 3”/“变量 4”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
6.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。

7.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.7 Advance

1.7.1 脉冲函数/ PULSE

1.7.1.1 指令基本信息

指令名称：PULSE

指令形式：

PULSE(variable, Value, time, advance) ;

1.7.1.2 指令功能

设置一个布尔变量为特殊值，并且维持预设定时间，时间单位为秒。预设定时间达到时，布尔变量将会被设置为相反值。设置 advance 为真时，后续指令则可以继续执行，无需等待设置时间达到。

1.7.1.3 参数说明

表 1-50 指令参数列表

参数序号	参数名称	默认值	说明
1	目标变量	空	设置目标变量
2	目标值	空	设置目标值
3	持续时间	空	设置持续时间
4	是否重复	空	设置是否重复

1.7.1.4 使用用例

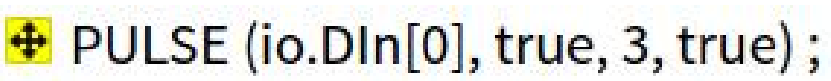
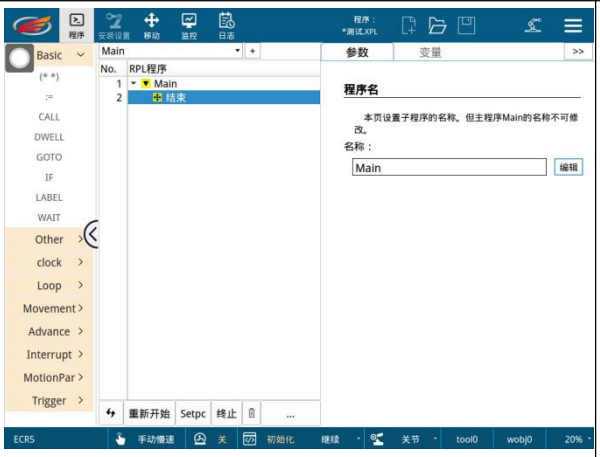
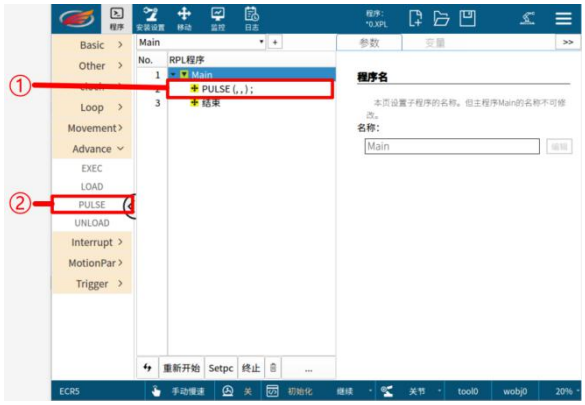
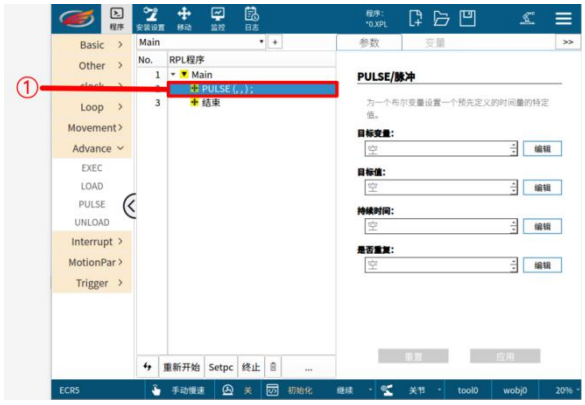
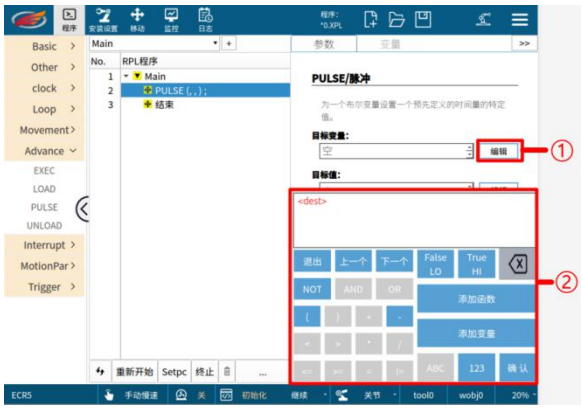
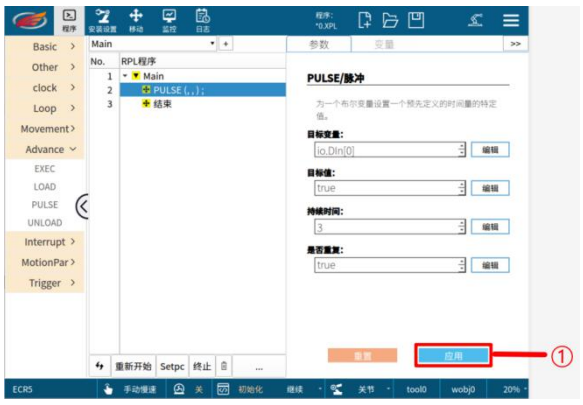
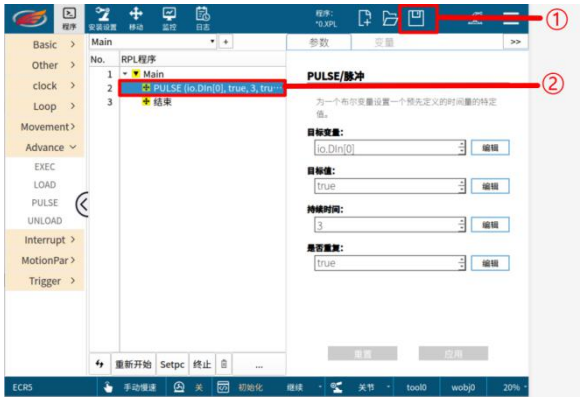


图 1-25 PULSE 指令使用用例

1.7.1.5 指令添加和设置步骤

表 1-51 PULSE 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“高级/Advance”中找到PULSE指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“高级/Advance”未展开，点击一级菜单“高级/Advance”
3.点击程序树中刚刚添加的PULSE指令行，如右图①处。		点击程序树中的PULSE指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示PULSE指令的参数详情。

4.点击 PULSE 指令参数页中“目标变量”/“目标值”/“持续时间”/“是否重复”参数后的“编辑”按钮①，并在弹出的表达式编辑器②中输入对应的值，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时参数后的输入框文本将被设置为在表达式编辑器上编辑的文本。
5.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
6.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.7.2 加载文件 / LOAD

1.7.2.1 指令基本信息

指令名称：文件加载 /LOAD

指令形式：

Load（文件名）；

1.7.2.2 指令功能：

将指定文件中的程序加载到内存中，作为当前程序的模块。

1.7.2.3 参数说明：

表 1-52 指令参数列表

参数序号	参数名称	默认值	说明
1	文件名		

1.7.2.4 使用用例

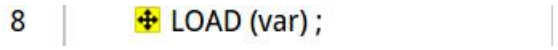
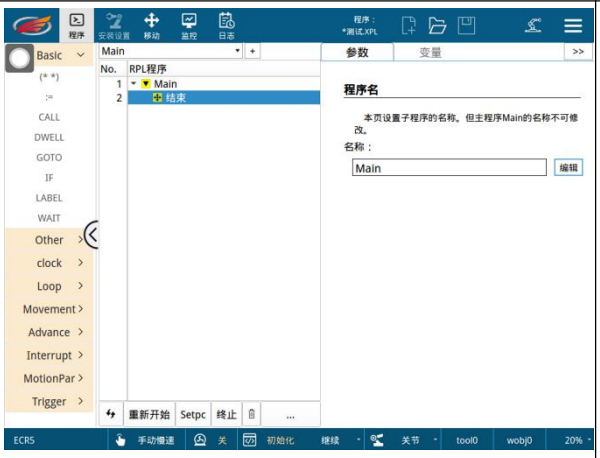
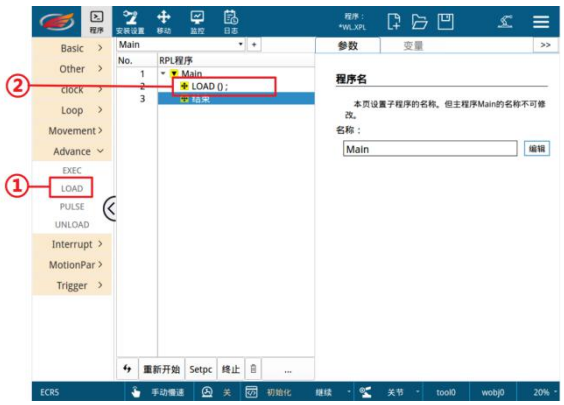
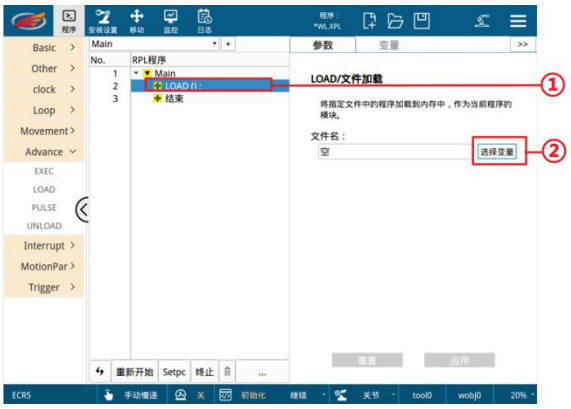
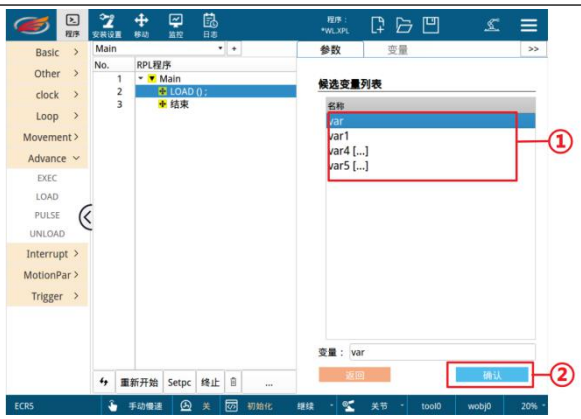
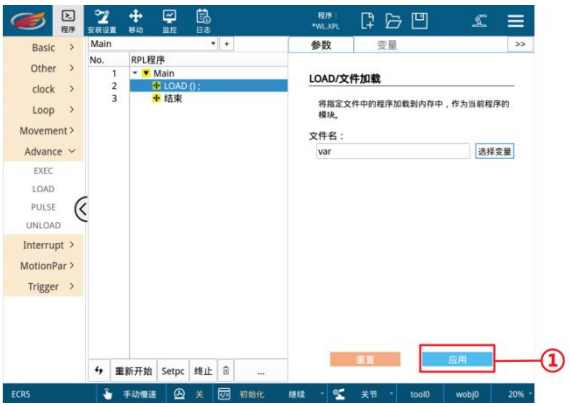
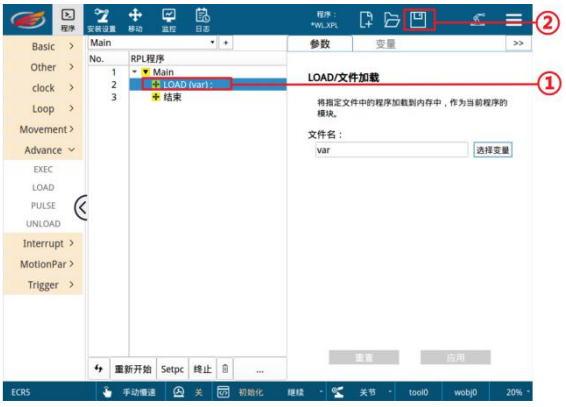


图 1-26 文件加载指令使用用例

1.7.2.5 文件加载 /LOAD 指令添加和设置步骤

表 1-53 文件加载 /LOAD 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“高级/Advance”中找到 LOAD 指令，点击 LOAD 指令，向程序树中添加 LOAD 指令		若一级菜单“高级/Advance”未展开，点击一级菜单“高级/Advance”。

3.点击程序树中刚刚添加的 LOAD 指令行。		点击程序树中的 LOAD 指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 此处可以设置 LOAD 的参数，设置后需要点击应用以保存修改。
4.点击“选择变量”按钮设置第一个参数，在弹出的候选变量列表根据实际情况设置具体参数，设置完成后点击“确认”按钮 5.也可以直接点击输入框使用键盘输入文件名		该指令参数可以直接点击参数框手动输入值
6.设置完参数后，点击“应用”按钮将修改保存到程序树中		
7.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.7.3 卸载文件/ UNLOAD

1.7.3.1 指令基本信息

指令名称：UNLOAD

指令形式：

UNLOAD(文件名);

1.7.3.2 指令功能：

从之前加载指令 LOAD 的存储器中卸载模块。

1.7.3.3 参数说明：

表 1-54 指令参数列表

参数序号	参数名称	默认值	说明
1	文件名		

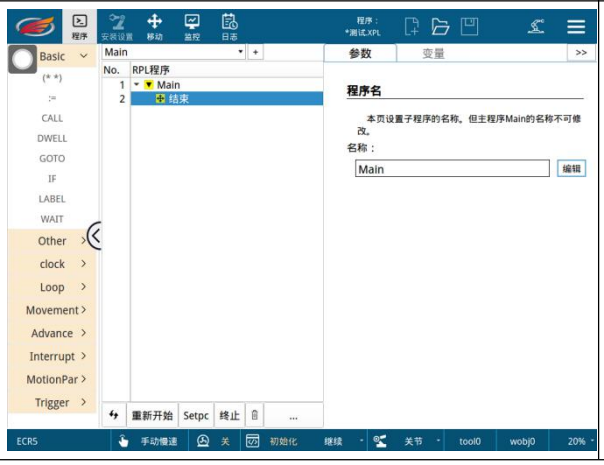
1.7.3.4 使用用例

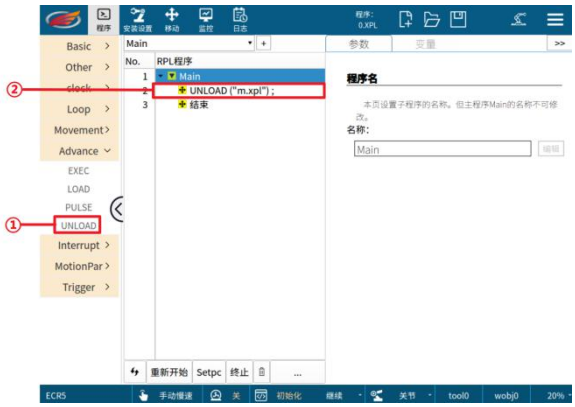
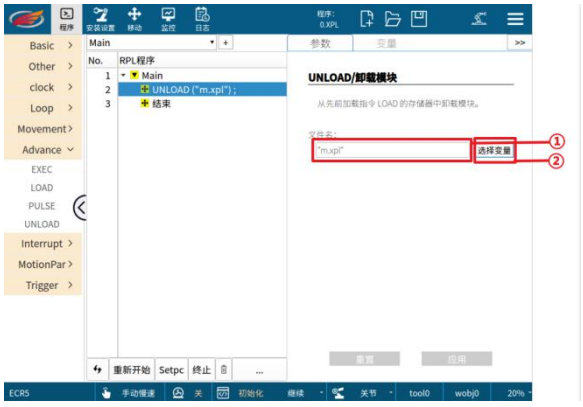
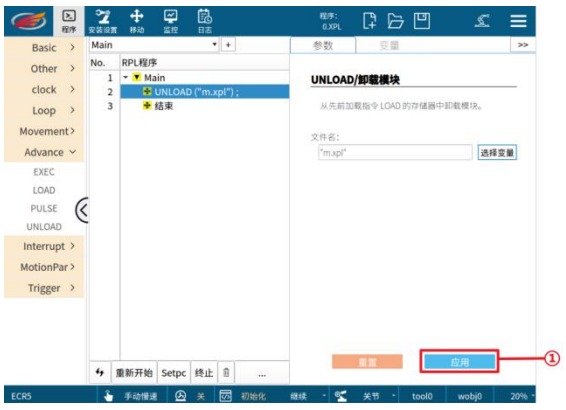
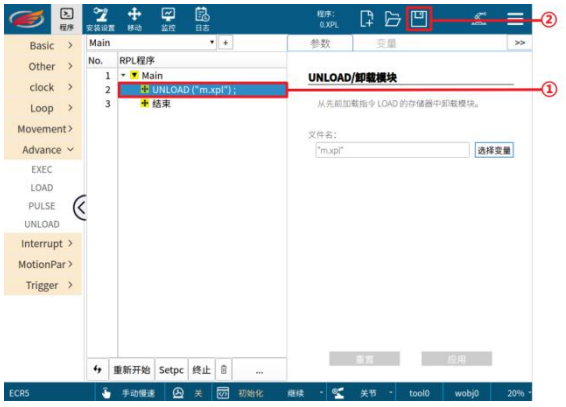
```
2   UNLOAD ("m.xpl") ;
```

图 1-27 卸载指令使用用例

1.7.3.5 卸载 / UNLOAD 指令添加和设置步骤

表 1-55 卸载 /UNLOAD 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2.在左侧指令栏中一级菜单“Advance”中找到UNLOAD指令，点击 UNLOAD 指令，向程序树中添加UNLOAD指令		1.若一级菜单“Advance”未展开，点击一级菜单“Advance”。
3.点击程序树中刚刚添加的卸载指令行。		1.点击程序树中的卸载指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.点击输入框直接输入文件名或点击选择变量按钮设置。
4.点击“应用”按钮将修改保存到程序树中		程序树中被选中的卸载指令行的内容在点击应用后变为设置内容
5.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.7.4 执行外部程序 / EXEC

1.7.4.1 指令基本信息

指令名称：EXEC

指令形式：

EXEC [returnvalue]=programname(Input parameters)；

1.7.4.2 指令功能

调用使用“LOAD”指令加载的文件中的函数。

1.7.4.3 参数说明

表 1-56 指令参数列表

参数序号	参数名称	默认值	说明
1	返回值	空	可选项。设置调用程序函数的返回值
2	程序名	空	设置调用的程序名
3	输入参数	空	可选项。设置输入参数

1.7.4.4 使用用例

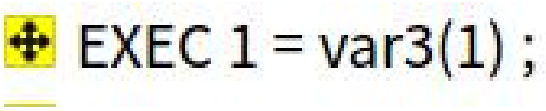
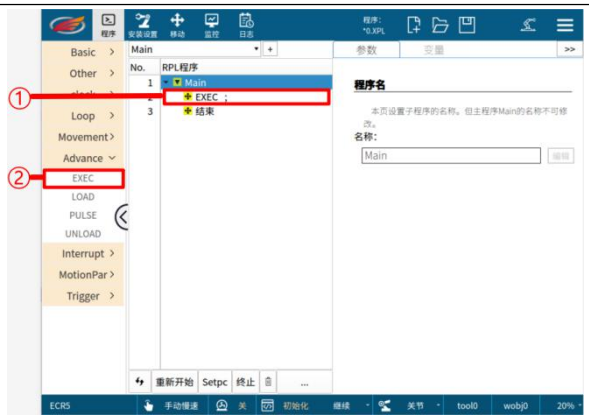
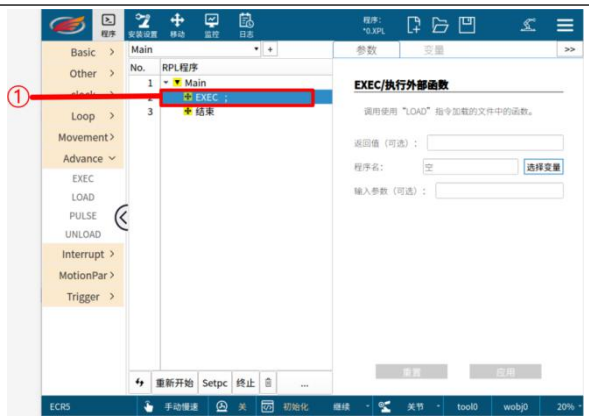
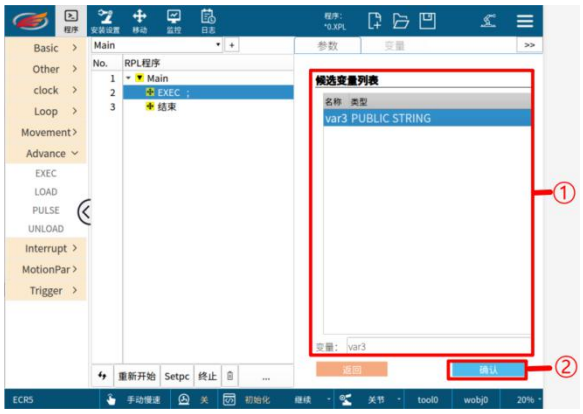


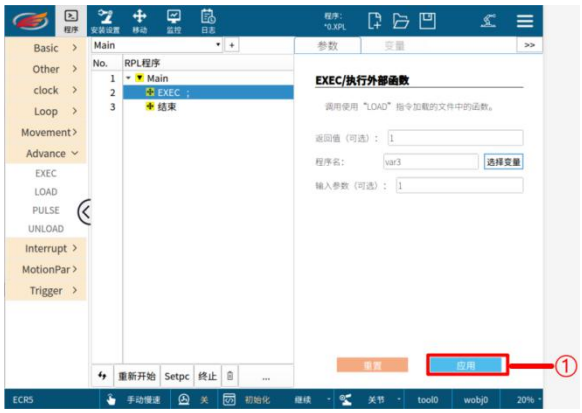
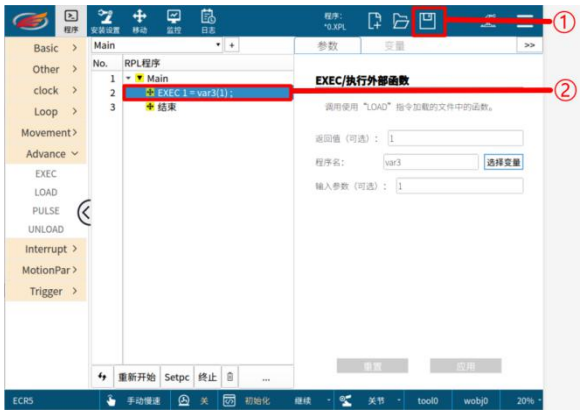
图 1-28 EXEC 指令使用用例

1.7.4.5 指令添加和设置步骤

表 1-57 EXEC 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2．在左侧指令栏中一级菜单“高级/Advance”中找到EXEC指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“高级/Advance”未展开，点击一级菜单“高级/Advance”
3.点击程序树中刚刚添加的EXEC指令行，如右图①处。		点击程序树中的EXEC指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示EXEC指令的参数详情。
4.点击EXEC指令参数页中“程序名”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“程序名”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
5.（可选项）点击EXEC指令参数页中“返回值”/“输入参数”下的输入框①/③，并在弹出键盘②中输入文本		输入完成后点击确认。

6.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
7.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.8 计时/ Clock

1.8.1 重置计时 / CLOCKRESET

1.8.1.1 指令基本信息

指令名称：CLOCKRESET

指令形式：

CLOCKRESET(name);

1.8.1.2 指令功能

重置用于计时的 CLOCK 数据类型的变量。

1.8.1.3 参数说明

表 1-58 指令参数列表

参数序号	参数名称	默认值	说明
1	名称	空	选择需要重置的变量名称

1.8.1.4 使用用例

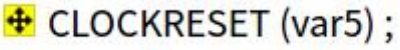
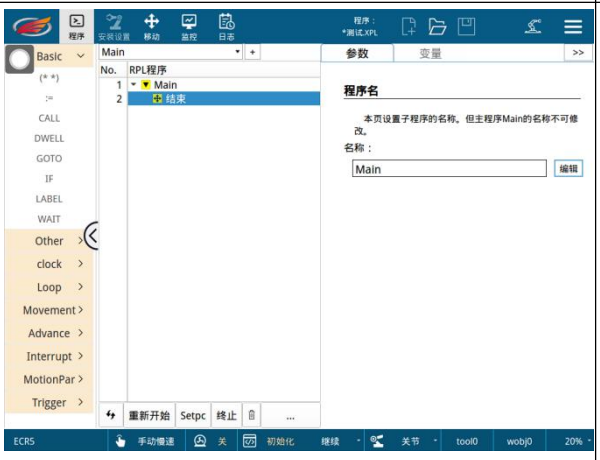
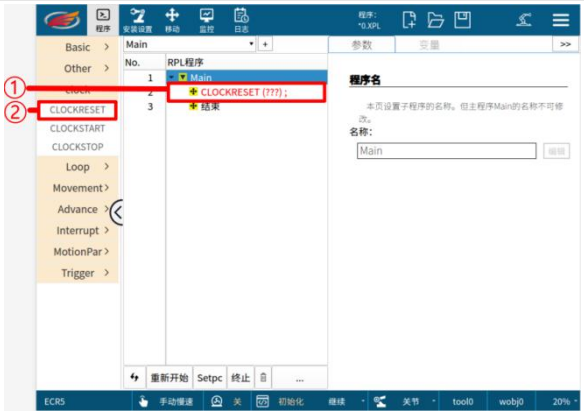
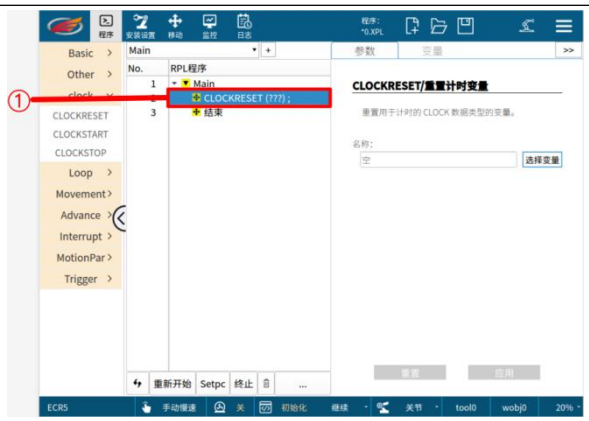
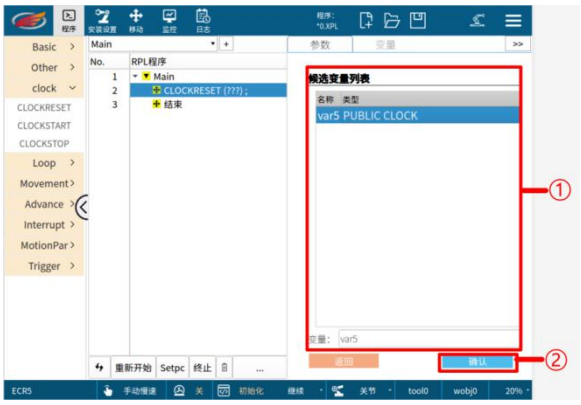
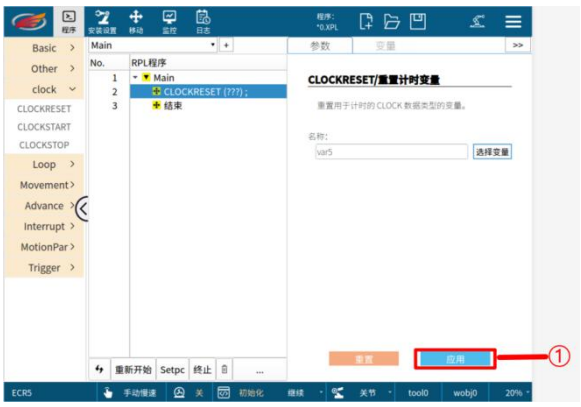
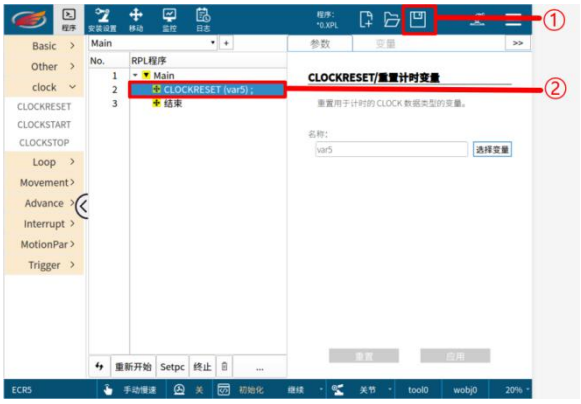


图 1-29 CLOCKRESET 指令使用用例

1.8.1.5 指令添加和设置步骤

表 1-59 CLOCKRESET 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“时钟/Clock”中找到CLOCKRESET指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“时钟/Clock”未展开，点击一级菜单“时钟/Clock”
3.点击程序树中刚刚添加的CLOCKRESET指令行，如右图①处。		点击程序树中的CLOCKRESET指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示CLOCKRESET指令的参数详情。

4. 点击CLOCKRESET指令参数页中“名称”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“名称”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
5.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
6.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.8.2 开始计时 /CLOCKSTART

1.8.2.1 指令基本信息

指令名称：CLOCKSTART

指令形式：

CLOCKSTART（变量名）；

1.8.2.2 指令功能：

通过 CLOCK 类型变量开始计时。

1.8.2.3 参数说明：

表 1-60 指令参数列表

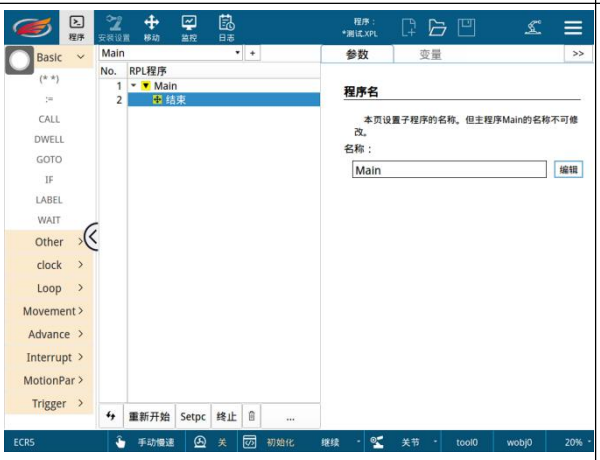
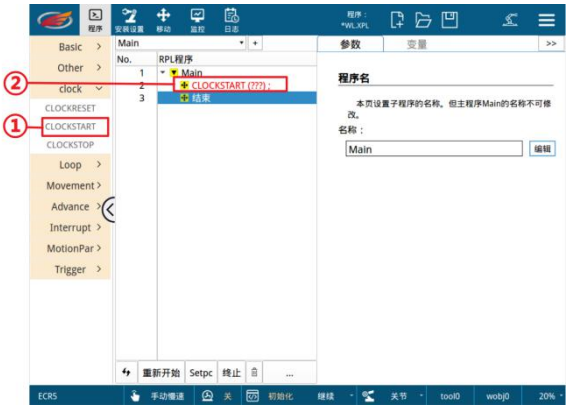
参数序号	参数名称	默认值	说明
1	变量名		

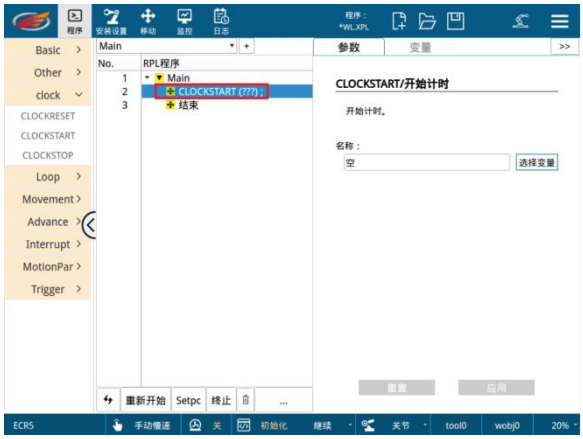
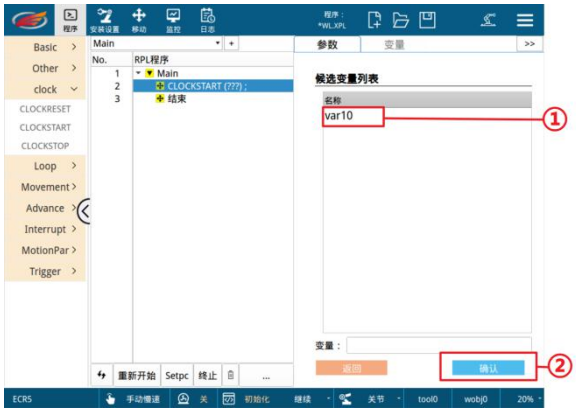
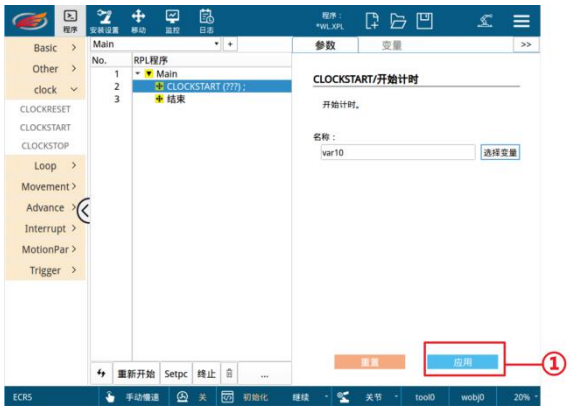
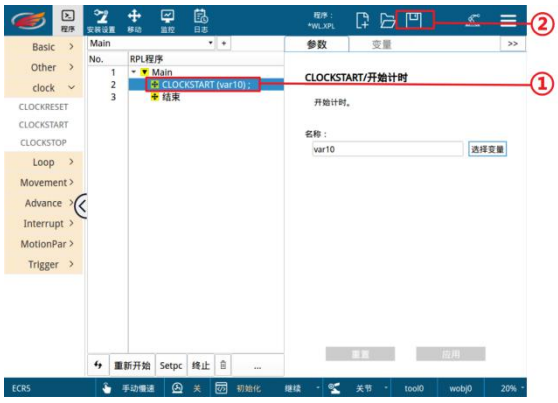
1.8.2.4 使用用例

图 1-30 开始计时指令使用用例

1.8.2.5 开始计时 /CLOCKSTART 指令添加和设置步骤

表 1-61 开始计时 /CLOCKSTART 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“时钟/clock”中找到CLOCKSTART指令，点击  指令，向程序树中添加CLOCKSTART指令		若一级菜单“时钟/clock”未展开，点击一级菜单“时钟/clock”。

3.点击程序树中刚刚添加的CLOCKSTART指令行。		点击程序树中的CLOCKSTART指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 此处可以设置CLOCKSTART的参数，设置后需要点击应用以保存修改。
4.点击“选择变量”按钮设置第一个参数，在弹出的候选变量列表根据实际情况设置具体参数，设置完成后点击“确认”按钮		该指令参数不可以直接点击参数框手动输入值。该参数类型为clock
5.设置完参数后，点击“应用”按钮将修改保存到程序树中		
6.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.8.3 停止计时/ CLOCKSTOP

1.8.3.1 指令基本信息

指令名称：CLOCKSTOP

指令形式：

CLOCKSTOP(变量);

1.8.3.2 指令功能：

停止计时。

1.8.3.3 参数说明：

表 1-62 指令参数列表

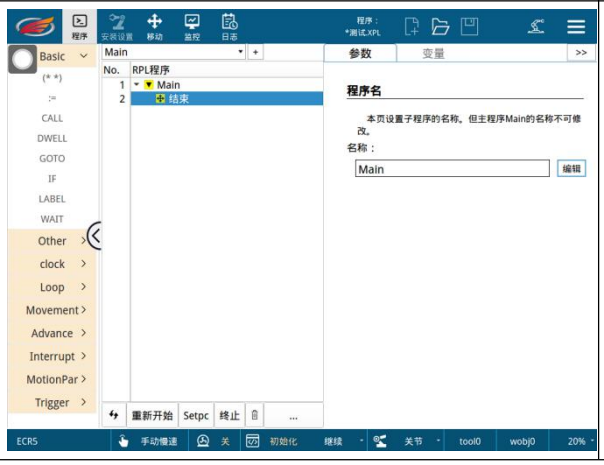
参数序号	参数名称	默认值	说明
1	变量		

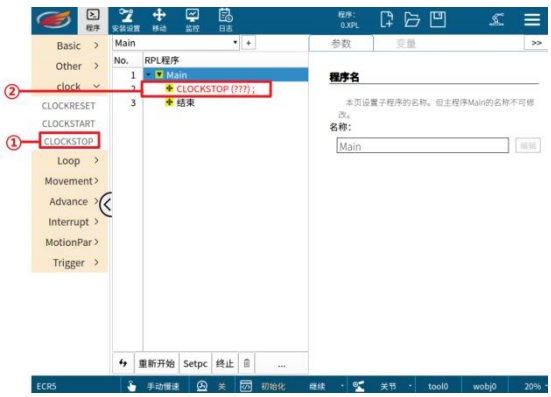
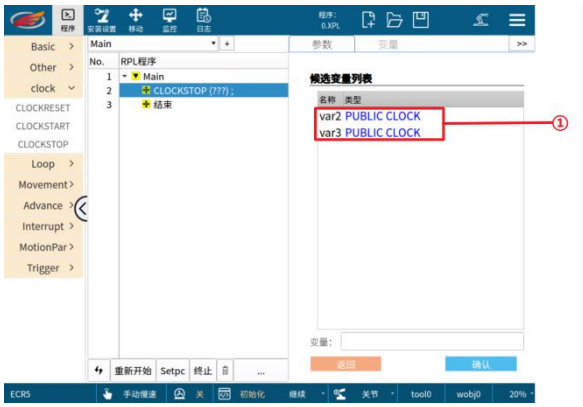
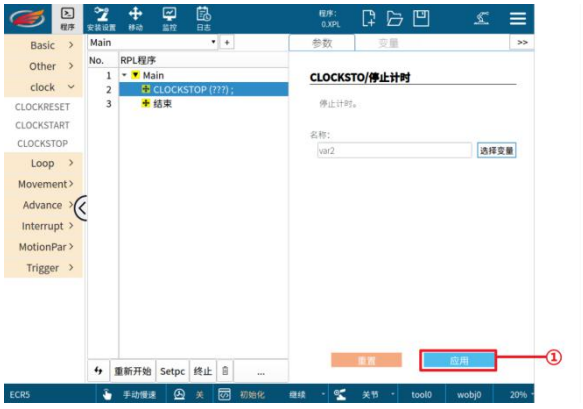
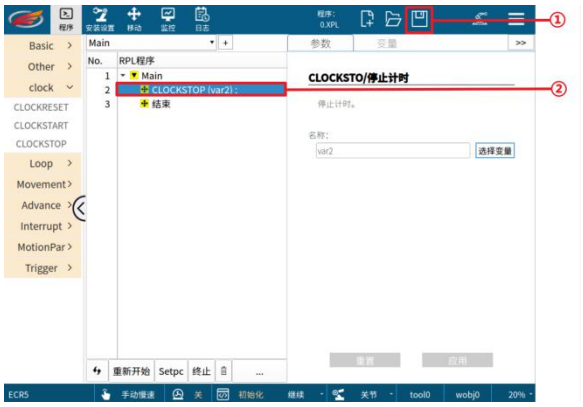
1.8.3.4 使用用例

图 1-31 停止计时指令使用用例

1.8.3.5 停止计时 / CLOCKSTOP 指令添加和设置步骤

表 1-63 停止计时 /CLOCKSTOP 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2.在左侧指令栏中一级菜单“计时/Clock”中找到CLOCKSTOP指令，点击 CLOCKSTOP 指令，向程序树中添加 CLOCKSTOP 指令		1.若一级菜单“计时/Clock”未展开，点击一级菜单“计时/Clock”。
3.点击程序树中刚刚添加的停止计时指令行并点击选择变量。		1.点击程序树中的停止计时指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.选择合适变量后点击确认以保存此次选择。
4.点击“应用”按钮将修改保存到程序树中		程序树中被选中的停止计时指令行的内容在点击应用后变为设置内容
5.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.9 中断/ Interrupt

1.9.1 错误编号中断 / INTRERRNO

1.9.1.1 指令基本信息

指令名称：INTRERRNO

指令形式：

INTRERRNO**(Intr variable,[integer value]);**

1.9.1.2 指令功能

当系统变量_errno_被设置成一个特定值或非零值，激活该触发变量。

1.9.1.3 参数说明

表 1-64 指令参数列表

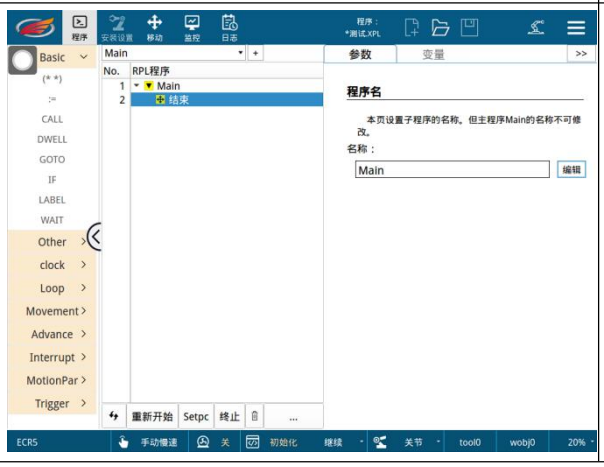
参数序号	参数名称	默认值	说明
1	名称	空	选择变量名称
2	目标值	空	设置目标值

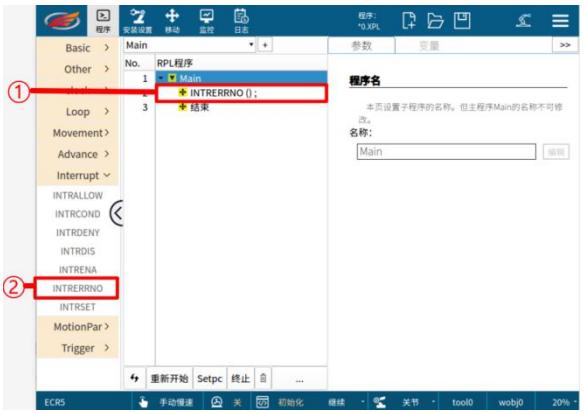
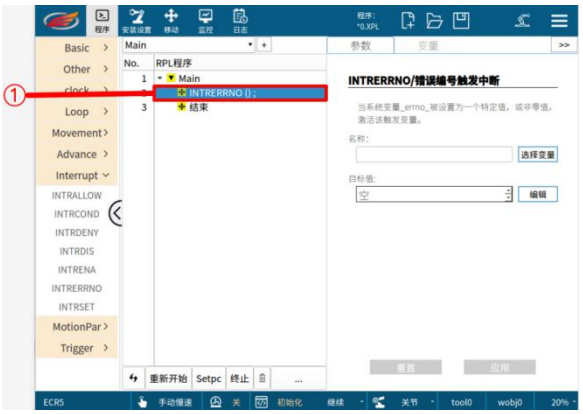
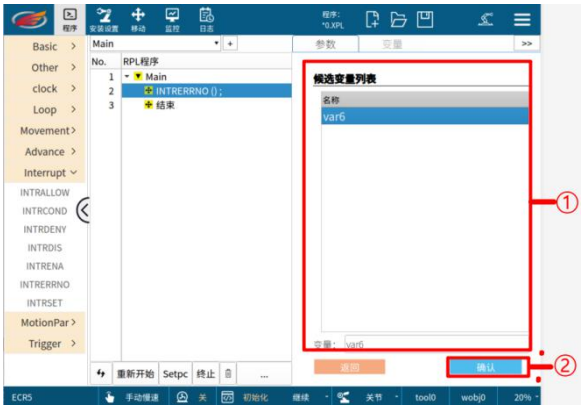
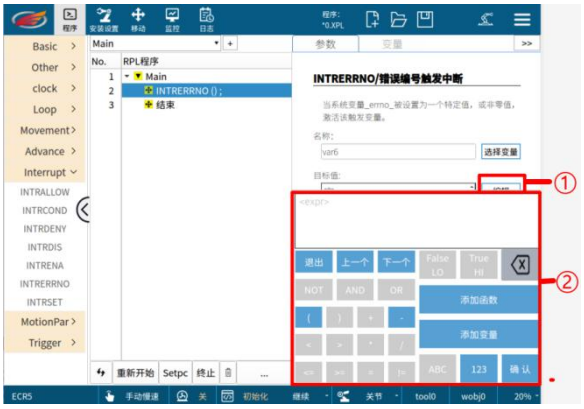
1.9.1.4 使用用例

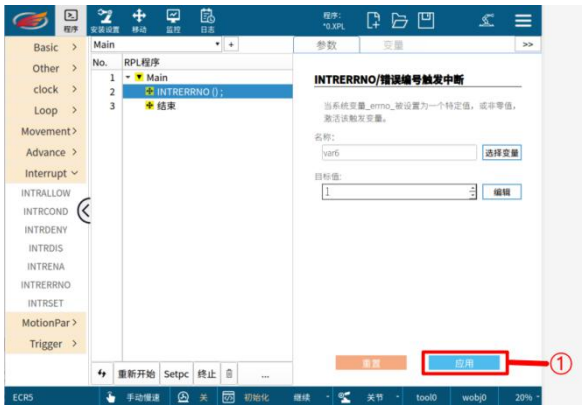
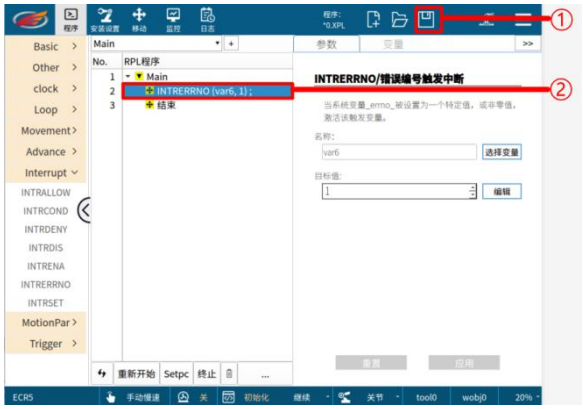
图 1-32 INTRERRNO 指令使用用例

1.9.1.5 指令添加和设置步骤

表 1-65 INTRERRNO 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2．在左侧指令栏中一级菜单“中断/Interrupt”中找到INTRERRNO指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“中断/Interrupt”未展开，点击一级菜单“中断/Interrupt”
3.点击程序树中刚刚添加的INTRERRNO指令行，如右图①处。		点击程序树中的INTRERRNO指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示INTRERRNO指令的参数详情。
4. 点击INTRERRNO指令参数页中“名称”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“名称”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
5. 点击INTRERRNO指令参数页中“目标值”参数后的“编辑”按钮①，并在弹出的表达式编辑器②中输入一个条件，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时“目标值”参数下的输入框文本将被设置为在表达式编辑器上编辑的文本。

6.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
7.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.9.2 激活中断 / INTRENA

1.9.2.1 指令基本信息

指令名称：INTRENA

指令形式：

INTRENA(变量);

1.9.2.2 指令功能：

启用所有中断或某个中断。

1.9.2.3 参数说明：

表 1-66 指令参数列表

参数序号	参数名称	默认值	说明
1	变量		

1.9.2.4 使用用例

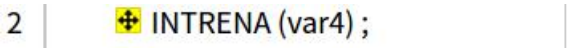
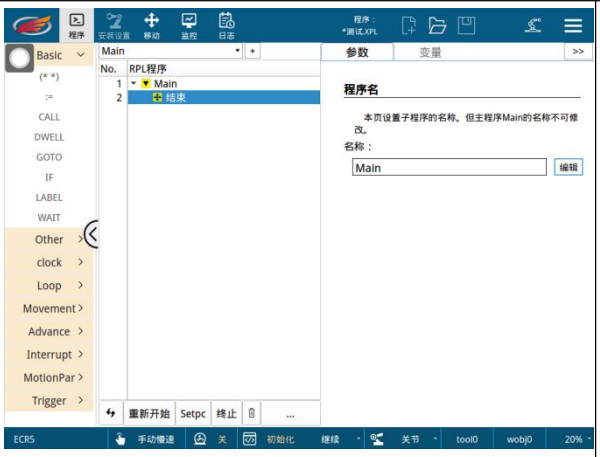
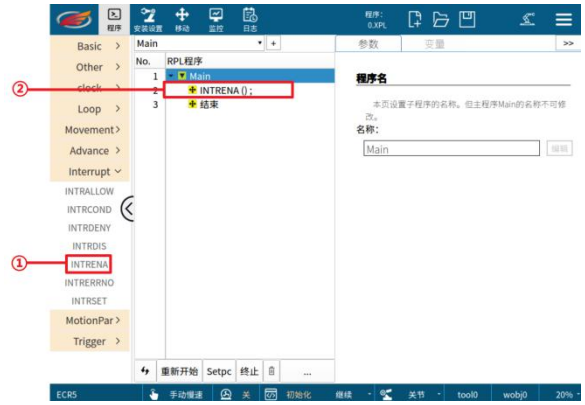
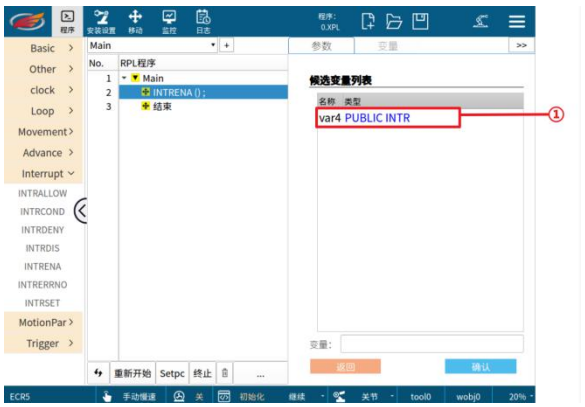
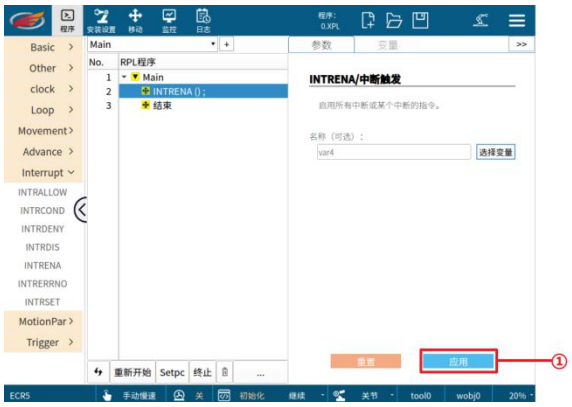
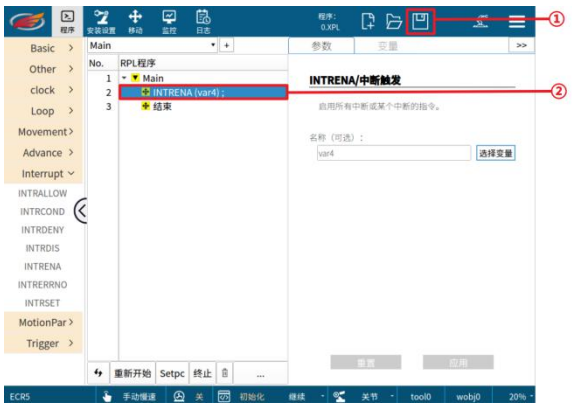


图 1-33 中断触发指令使用用例

1.9.2.5 中断触发 / INTRENA 指令添加和设置步骤

表 1-67 中断触发 /INTRENA 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“中断/Interupt”中找到INTRENA 指令，点击 INTRENA 指令，向程序树中添加 INTRENA 指令		1.若一级菜单“中断/Interupt”未展开，点击一级菜单“中断/Interupt”。
3.点击程序树中刚刚添加的中断触发指令行并点击选择变量。		1.点击程序树中的中断触发指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.选择合适变量后点击确认以保存此次选择。

4.点击“应用”按钮将修改保存到程序树中		程序树中被选中的中断触发指令行的内容在点击应用后变为设置内容
5.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.9.3 中断否决/INTRDENY

1.9.3.1 指令基本信息

指令名称：INTRDENY

指令形式：

INTRDENY（变量名）；

1.9.3.2 指令功能：

禁用特定中断触发的指令。此指令之后该指定的任何触发条件都将失效。在使用该指令前，INTR变量必须要与一个中断关联，否则报告错误。

1.9.3.3 参数说明：

表 1-68 指令参数列表

参数序号	参数名称	默认值	说明
1	变量名		

1.9.3.4 使用用例

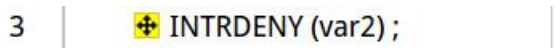
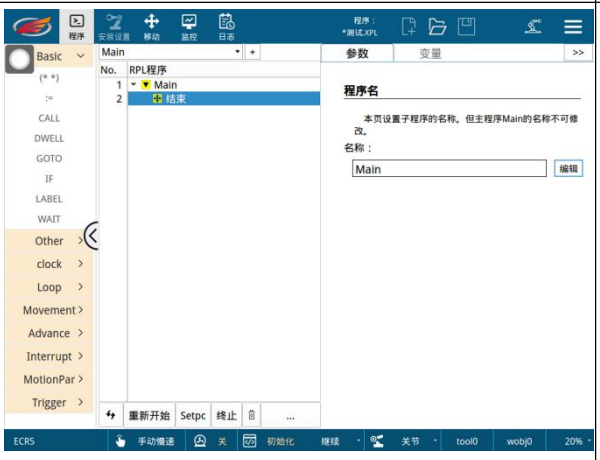
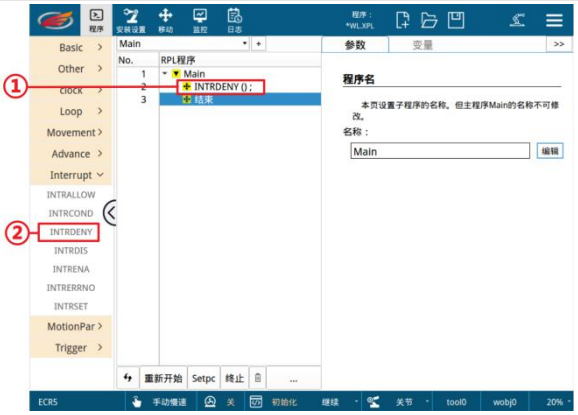
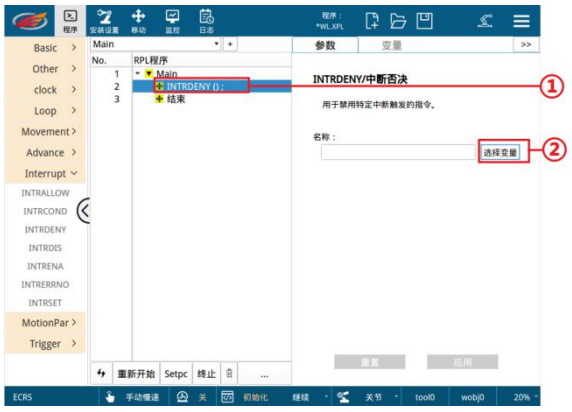
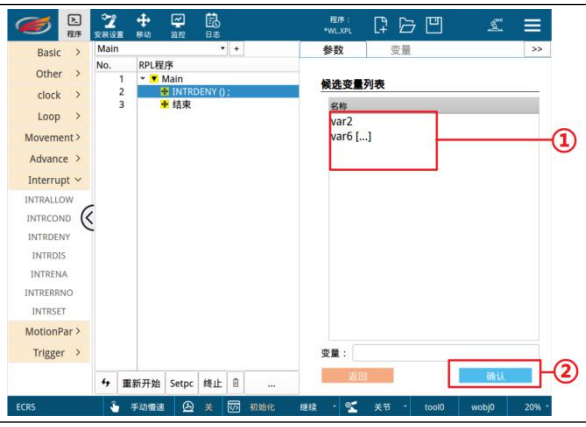
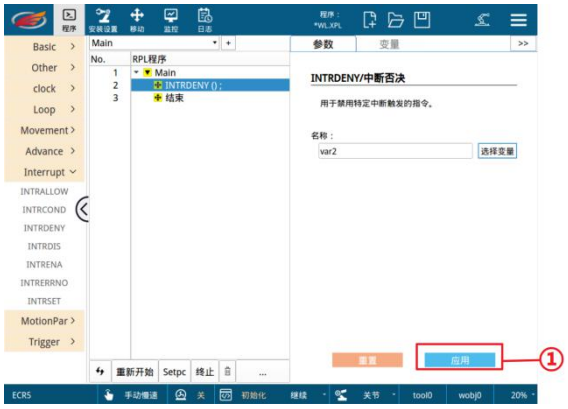
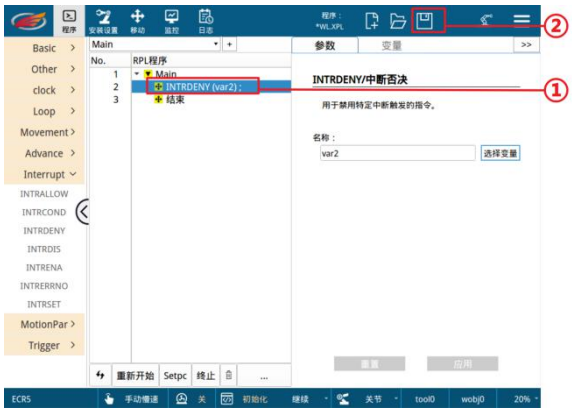


图 1-34 中断否决指令使用用例

1.9.3.5 中断否决 /INTRDENY 指令添加和设置步骤

表 1-69 中断否决 /INTRDENY 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“中断/Interrupt”中找到INTRDENY 指令，点击 INTRDENY 指令，向程序树中添加INTRDENY 指令		若一级菜单“中断/Interrupt”未展开，点击一级菜单“中断/Interrupt”。
3.点击程序树中刚刚添加的INTRDENY 指令行。		点击程序树中的INTRDENY 指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 此处可以设置INTRDENY 的参数，设置后需要点击应用以保存修改。

4.点击“选择变量”按钮设置第一个参数，在弹出的候选变量列表根据实际情况设置具体参数，设置完成后点击“确认”按钮		该指令参数不可以直接点击参数框手动输入值,该参数类型为 INTR
5.设置完参数后，点击“应用”按钮将修改保存到程序树中		
6.点击任务栏中“保存”按钮，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.9.4 中断禁用 /INTRDIS

1.9.4.1 指令基本信息

指令名称：INTRDIS

指令形式：

INTRDIS（变量名）；

1.9.4.2 指令功能：

禁用所有中断或某个中断的指令。。如果省略了指令后的可选参数，所有中断将会被禁用。 当

中断被禁用了，中断条件仍会被检测；如果满足了中断条件，则一旦中断被重新启用，就 会调用该中断。 如果没用省略指令后的可选参数，则 INTR 变量必须要与一个中断关联，否则报告错误。

1.9.4.3 参数说明：

表 1-70 指令参数列表

参数序号	参数名称	默认值	说明
1	变量名		该变量可为空

1.9.4.4 使用用例

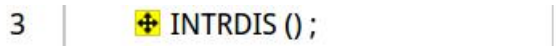
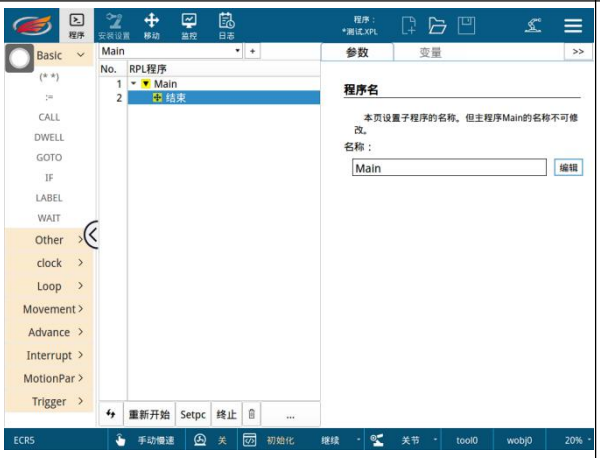
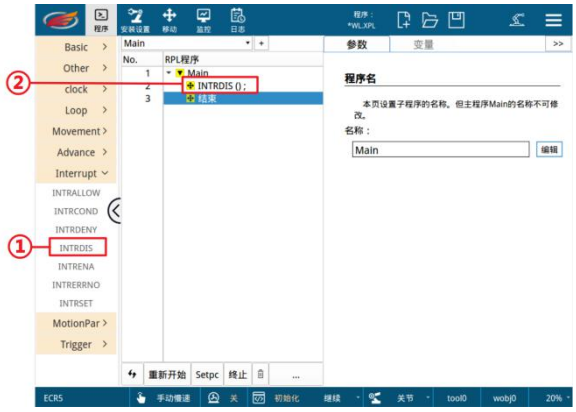
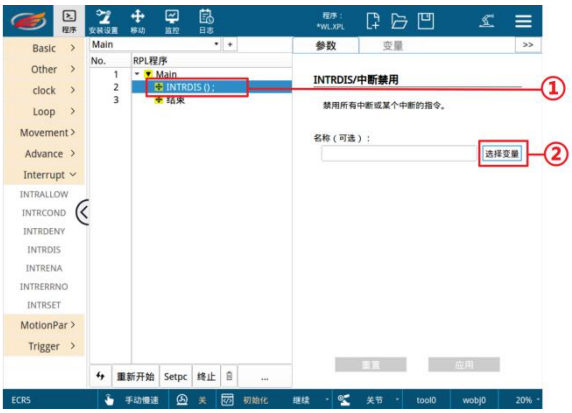
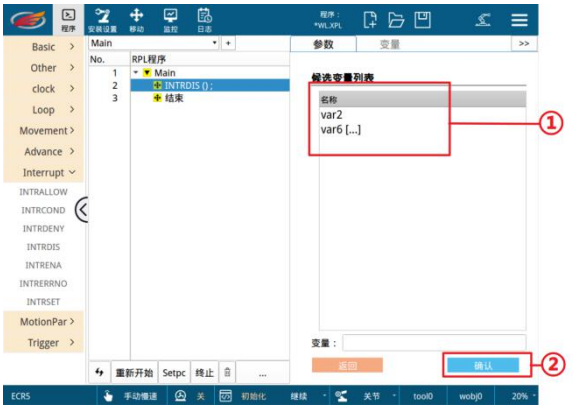
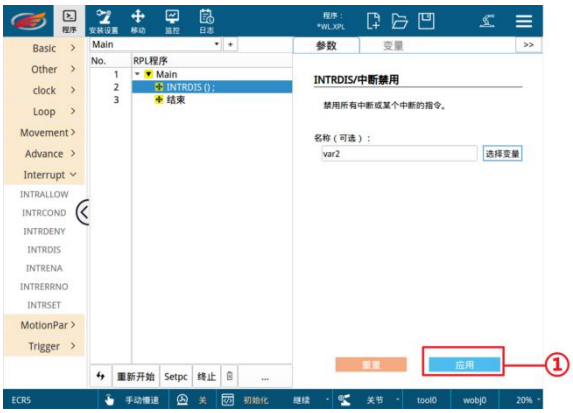
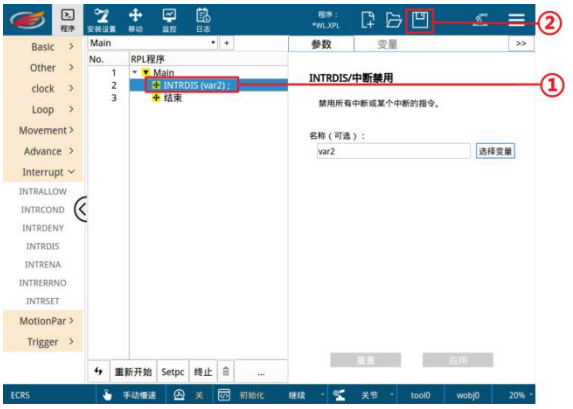


图 1-35 中断禁用指令使用用例

1.9.4.5 中断禁用 /INTRDIS 指令添加和设置步骤

表 1-71 中断禁用 /INTRDIS 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“中断/Interrupt”中找到 INTRDIS 指令，点击 INTRDIS 指令，向程序树中添加 INTRDIS 指令		若一级菜单“中断/Interrupt”未展开，点击一级菜单“中断/Interrupt”。

3.点击程序树中刚刚添加的 INTRDIS 指令行。		点击程序树中的 INTRDIS 指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 此处可以设置 INTRDIS 的参数，设置后需要点击应用以保存修改。
4.点击“选择变量”按钮设置第一个参数，在弹出的候选变量列表根据实际情况设置具体参数，设置完成后点击“确认”按钮		该指令参数不可以直接点击参数框手动输入值,该参数类型为 INTR
5.设置完参数后，点击“应用”按钮将修改保存到程序树中		
6.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.9.5 中断条件 / INTRCOND

1.9.5.1 指令基本信息

指令名称：INTRCOND

指令形式：

INTRCOND(Intr variable,Bool expression);

1.9.5.2 指令功能

用于设置关联中断 INTR 变量的触发中断的条件，变量必须是布尔类型。在设置与中断相关的条件后，中断被使能，即当中断触发条件从假转化为真时，中断被触发。

1.9.5.3 参数说明

表 1-72 指令参数列表

参数序号	参数名称	默认值	说明
1	名称	空	选择变量名称
2	条件表达式	空	可选项。设置条件表达式

1.9.5.4 使用用例

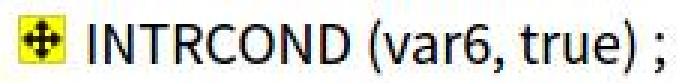
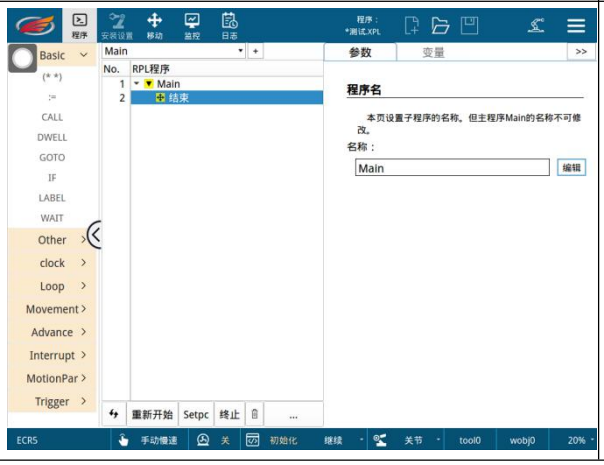
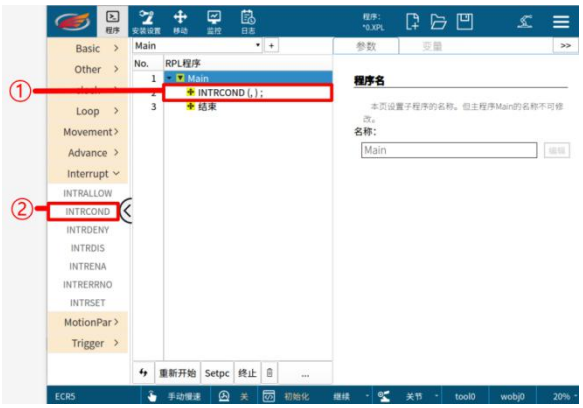
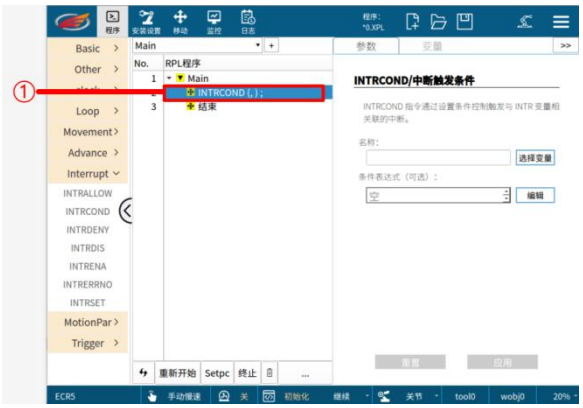
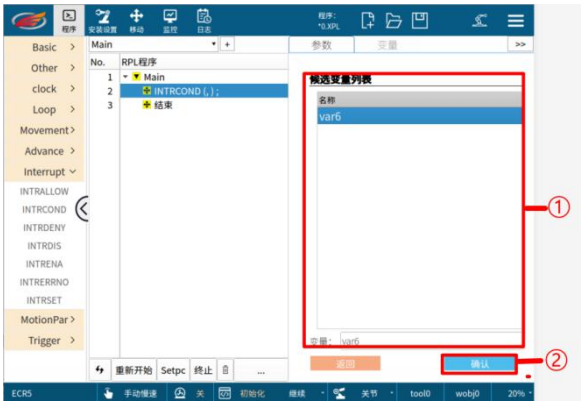
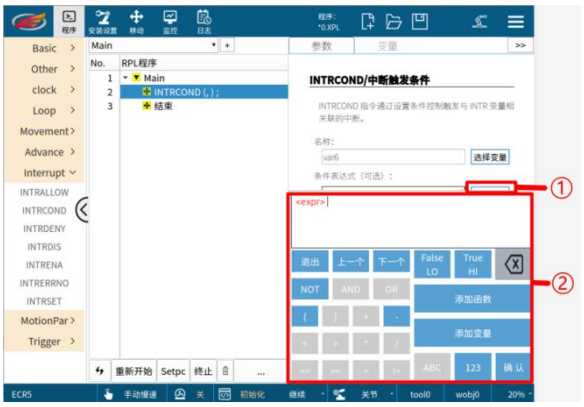


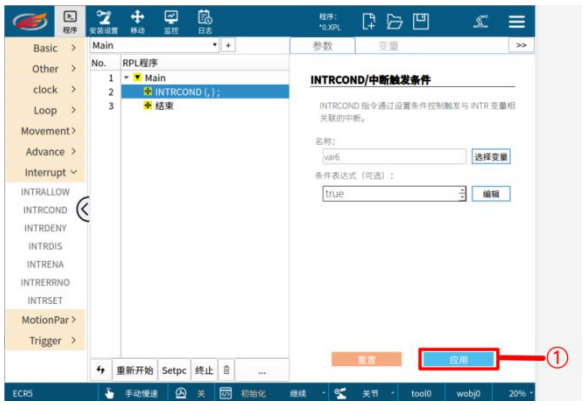
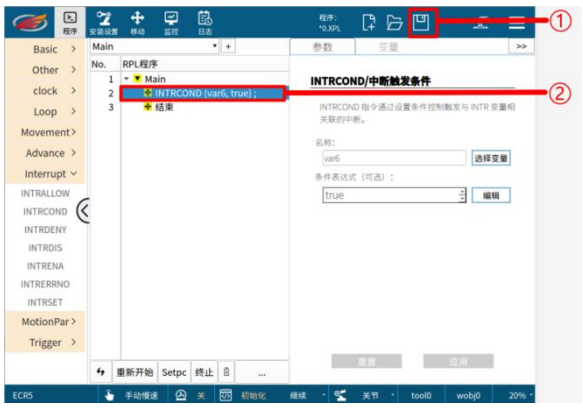
图 1-36 INTRCOND 指令使用用例

1.9.5.5 指令添加和设置步骤

表 1-73 INTRCOND 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2．在左侧指令栏中一级菜单“中断/Interrupt”中找到INTRCOND指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“中断/Interrupt”未展开，点击一级菜单“中断/Interrupt”
3.点击程序树中刚刚添加的INTRCOND指令行，如右图①处。		点击程序树中的INTRCOND指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示INTRCOND指令的参数详情。
4.点击INTRCOND指令参数页中“名称”参数后的“选择变量”按钮后在候选变量列表中①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“名称”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
5.点击INTRCOND指令参数页中“条件表达式”参数后的“编辑”按钮①，并在弹出的表达式编辑器②中输入一个条件，完成后点击确认。		点击表达式编辑器上确认按钮后，表达式编辑器关闭，同时“条件表达式”参数下的输入框文本将被设置为在表达式编辑器上编辑的文本。

6.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
7.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.9.6 中断设置 / INTRSET

1.9.6.1 指令基本信息

指令名称：INTRSET

指令形式：

INTRSET(Intr variable,Subroutine);

1.9.6.2 指令功能

将一个 INTR 变量与子程序连接的指令。无法设置已由先前的 INTRSET 指令设置的 INTR 变量，否则将会报错。请注意，不要循环且重复定义一个中断（参见 INTR 变量类型）。

1.9.6.3 参数说明

表 1-74 指令参数列表

参数序号	参数名称	默认值	说明
1	名称	空	选择变量名称
2	子程序	空	设置子程序

1.9.6.4 使用用例

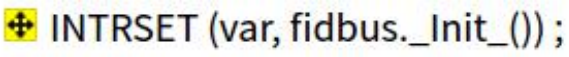
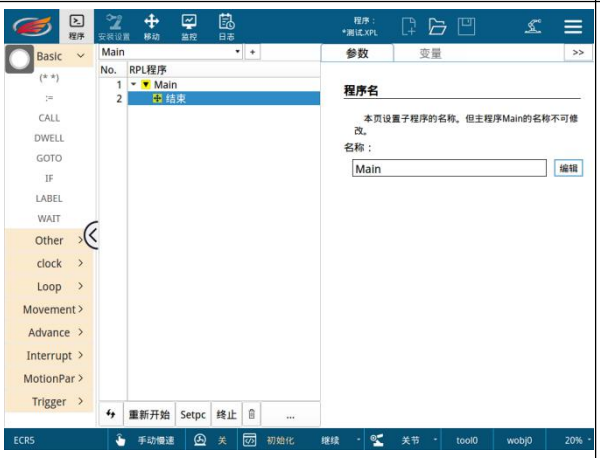
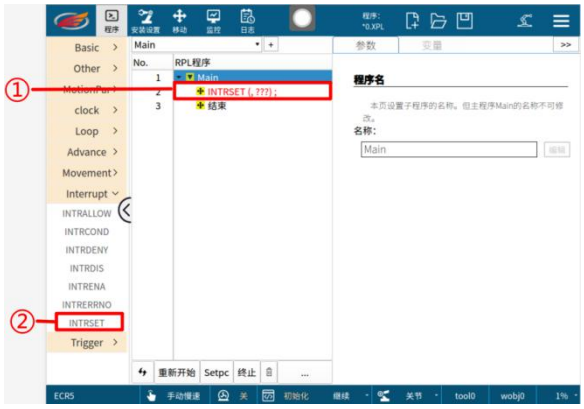
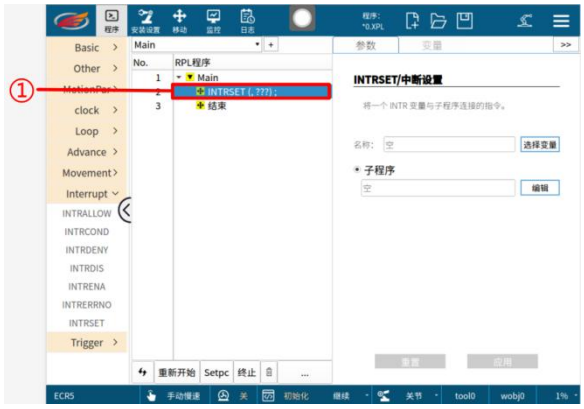
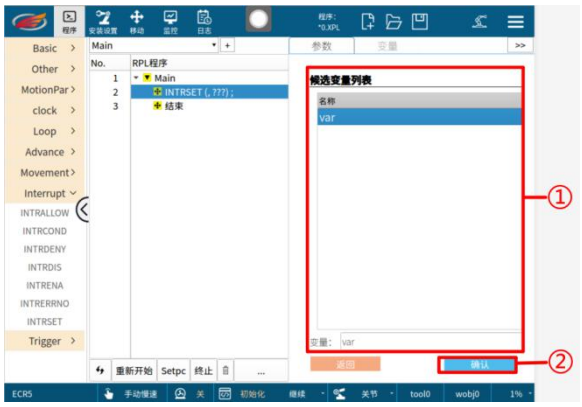
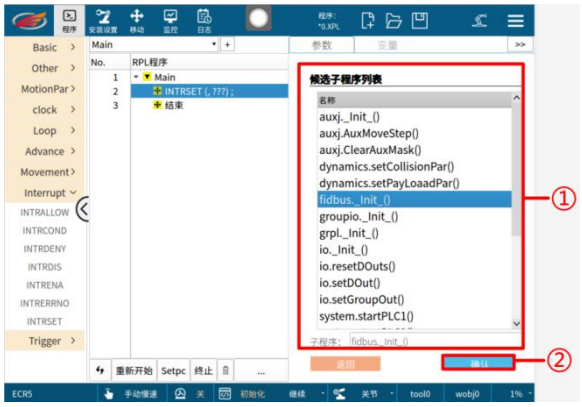
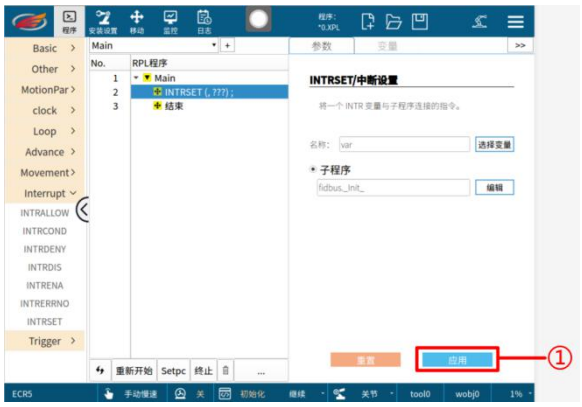
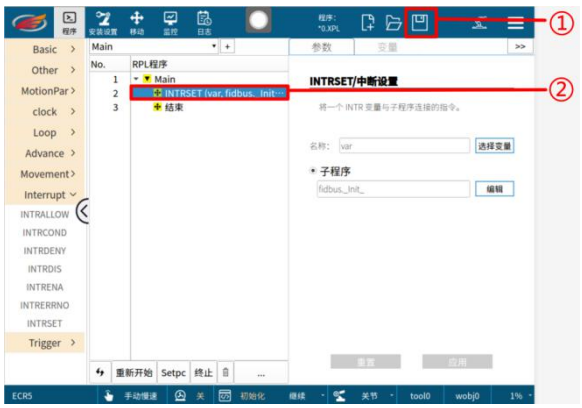


图 1-37 INTRSET 指令使用用例

1.9.6.5 指令添加和设置步骤

表 1-75 INTRSET 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2．在左侧指令栏中一级菜单“中断/Interrupt”中找到INTRSET指令，如右图所示②处，点击该处向程序树中添加该指令，添加成功后程序树中会显示该指令如右图①处		若一级菜单“中断/Interrupt”未展开，点击一级菜单“中断/Interrupt”
3.点击程序树中刚刚添加的INTRSET指令行，如右图①处。		点击程序树中的INTRSET指令行后，程序树会选中该行，同时，右侧的参数显示区页面将跳转至“参数”标签页，并显示INTRSET指令的参数详情。

4.点击 INTRSET 指令参数页中“名称”参数后的“选择变量”按钮后在候选变量列表①中选择需要设置的变量，完成后点击确认按钮②。		在候选变量列表中，选中某一个变量行后，该变量将在下方的变量输入框中显示，点击确认按钮后，候选变量列表页面关闭，同时“名称”参数下的输入框文本将被设置为在候选变量列表页面变量输入框中的文本。
5.点击 INTRSET 指令参数页中“名称”参数后的“选择变量”按钮后在候选子程序列表①中选择需要设置的子程序，完成后点击确认按钮②。		在候选子程序列表中，选中某一个子程序行后，该子程序将在下方的输入框中显示，点击确认按钮后，候选子程序列表页面关闭，同时“子程序”参数下的输入框文本将被设置为在候选子程序列表页面输入框中的文本。
6.点击“应用”按钮①将修改保存到程序树中		保存到程序树后，程序树中被选中的指令行参数的详细信息将变成文本框的输入内容。
7.点击任务栏中“保存”按钮①，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.9.7 中断允许/ INTRALLOW

1.9.7.1 指令基本信息

指令名称：INTRALLOW

指令形式：

INTRALLOW(变量);

1.9.7.2 指令功能：

启用先前被 INTRENY 指令禁用的特定中断指令。

1.9.7.3 参数说明：

表 1-76 指令参数列表

参数序号	参数名称	默认值	说明
1	变量		

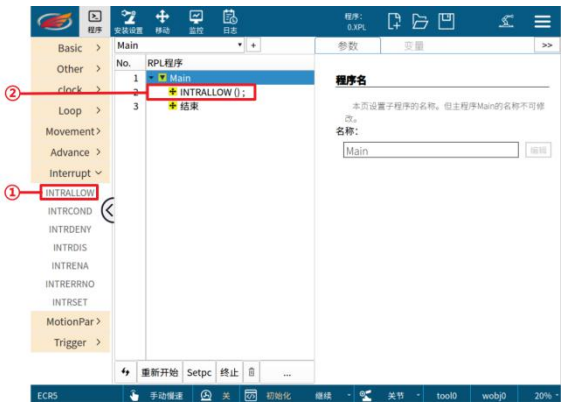
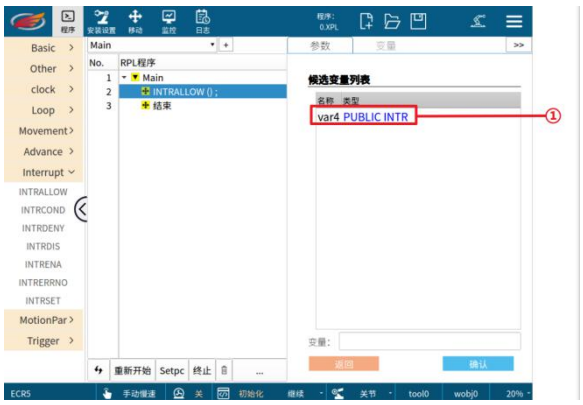
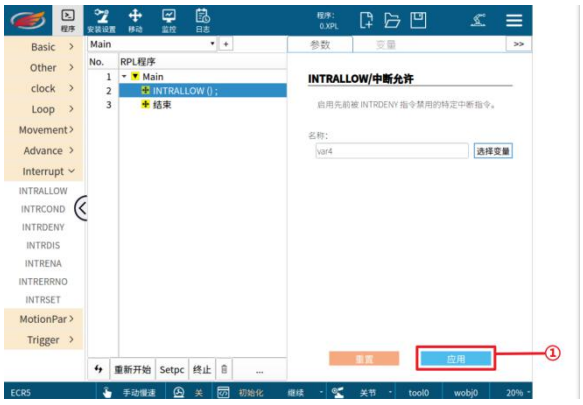
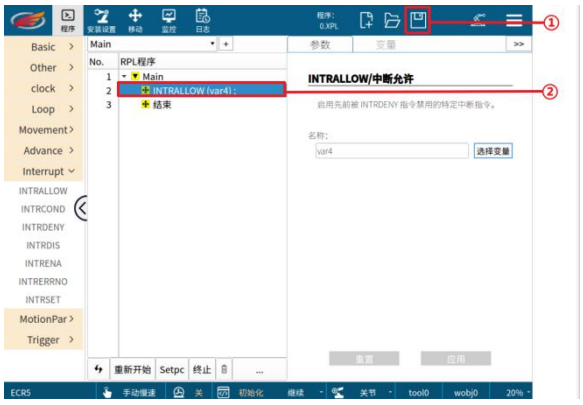
1.9.7.4 使用用例

图 1-38 中断允许指令使用用例

1.9.7.5 中断允许 / INTRALLOW 指令添加和设置步骤

表 1-77 中断允许 /INTRALLOW 指令添加和设置步骤

步骤	图示	说明
1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。

2.在左侧指令栏中一级菜单“中断/Interupt”中找到INTRALLOW指令，点击 INTRALLOW 指令，向程序树中添加INTRALLOW指令		1.若一级菜单“中断/Interupt”未展开，点击一级菜单“中断/Interupt”。
3.点击程序树中刚刚添加的中断允许指令行并点击选择变量。		1.点击程序树中的中断允许指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 2.选择合适变量后点击确认以保存此次选择。
4.点击“应用”按钮将修改保存到程序树中		程序树中被选中的中断允许指令行的内容在点击应用后变为设置内容
5.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

1.10 其他/Other

1.10.1 开关语句/CASE

1.10.1.1 指令基本信息

指令名称：CASE

指令形式：

CASE：var

VALUE_LIST:值

...

EXIT;

END CASE

1.10.1.2 指令功能：

可设置 case/开关语句的跳转条件。依赖于表达式的值而后执行 CASE 语句中一个 case 程序块。与 CASE 表达式匹配的 CASE 指令将会被执行；ELSE 指令必须在放置在 CASE 块中的最后，如果 CASE 中没有匹配的值，ELSE 段将会被执行。

1.10.1.3 参数说明：

表 1-78 指令参数列表

参数序号	参数名称	默认值	说明
1	变量名		该变量可为表达式，函数或值

1.10.1.4 使用用例

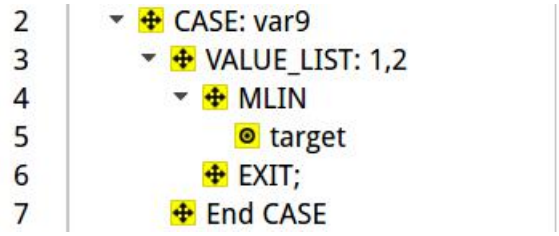
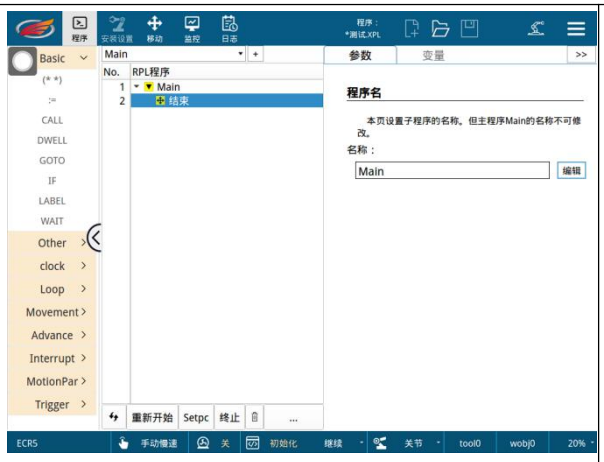
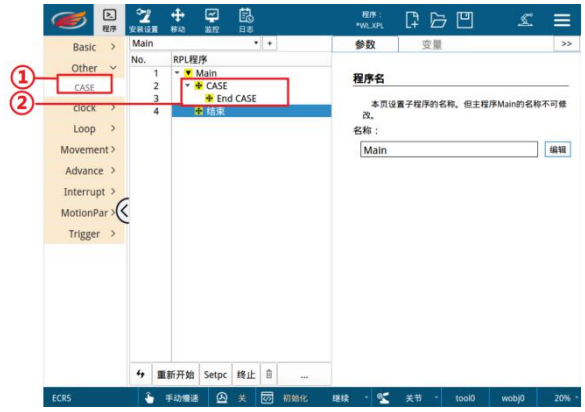
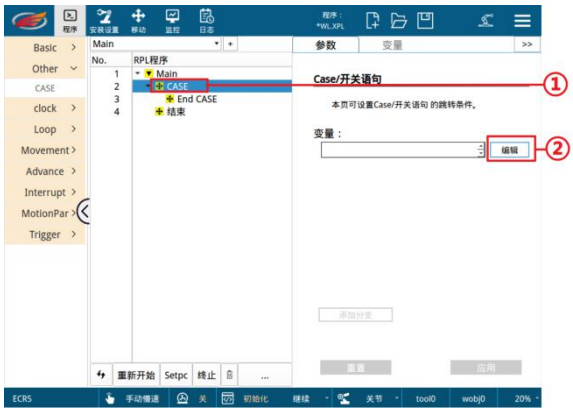
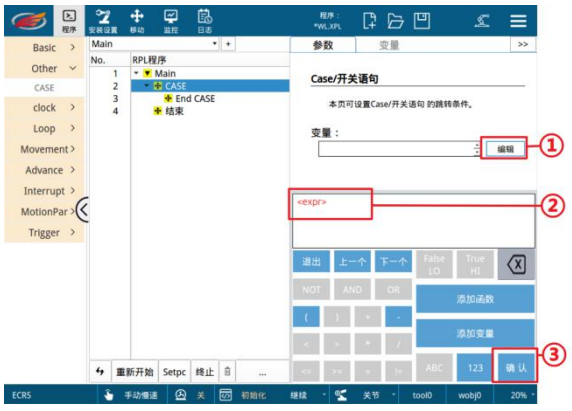


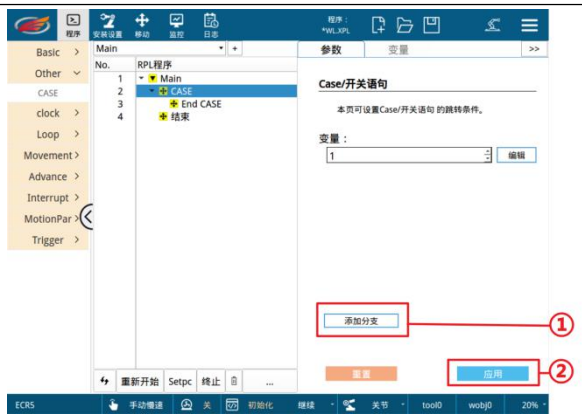
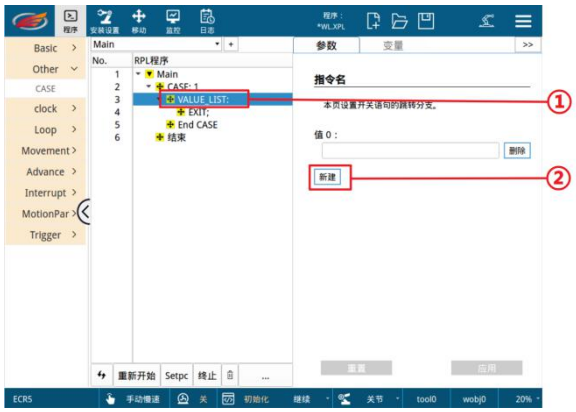
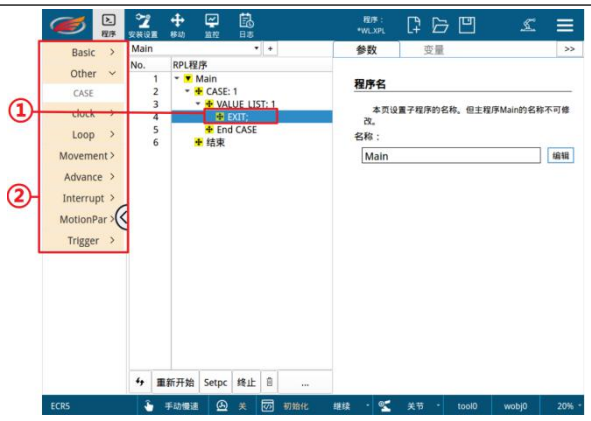
图 1-39 开关语句指令使用用例

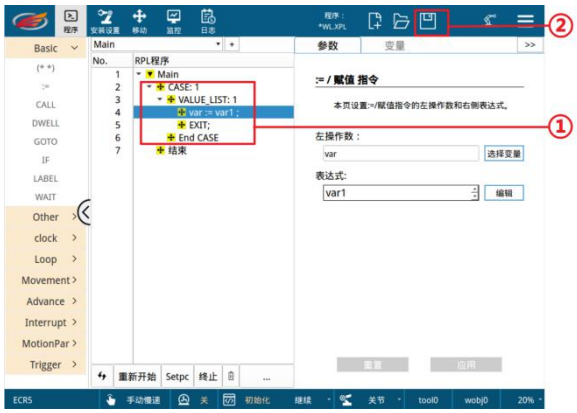
1.10.1.5 开关语句 /CASE 指令添加和设置步骤

表 1-79 开关语句 /CASE 指令添加和设置步骤

步骤	图示	说明

1.点击任务栏程序进入程序界面		如若编辑程序内容请确保已登录用户具有编程权限。
2.在左侧指令栏中一级菜单“其他/Other”中找到CASE指令，点击指令 CASE ，向程序树中添加CASE指令		若一级菜单“其他/Other”未展开，点击一级菜单“其他/Other”。
3.点击程序树中刚刚添加的CASE指令行。		点击程序树中的CASE指令行后，程序树会选中该行，同时，右侧的参数显示区，页面将跳转至“参数”标签页。 此处可以设置CASE的参数，设置后需要点击应用以保存修改。
4.点击“编辑”按钮设置第一个参数，在弹出的表达式输入框根据实际情况设置具体参数，设置完成后点击“确认”按钮		该指令参数不可以直接点击参数框手动输入值,该参数类型为INTR

5.编辑完成第一个参数后，“添加分支”按钮变为可操作状态，点击“添加分支”按钮为CASE指令添加一个新的分支。并点击“应用”按钮。		
6.点击程序树上添加的新分支，右边会显示相应参数页。 7.点击参数页中的“新建”按钮，新建一个值。		
8.点击第一个参数，输入一个与CASE变量表达式相匹配的值。输入完成后点击“应用”按钮。		当该值与CASE表达式的值相等时，则执行该分支内的指令
9.在第一个分支内添加需要执行的指令，完成配置。		

10.点击任务栏中“保存”按钮  ，将当前对程序的修改保存到文件。		如不保存到文件，对程序的修改将是暂时的。

第 2 章　变量

2.1 本章简介

本章将介绍新版程序编辑器中与变量相关的功能及其的操作步骤。编译器共支持 19 中变量类型，表 2-1 是编译器支持的变量类型及其相关说明。本版编辑器将变量的可见性分为：系统（对应老版编译器 module 中 task 和 global）、全局（对应老版编译器 module 中 public）、工程（对应老版编译器 module 中 Local）、Main、子程序的输入、子程序的输出以及子程序的本地 7 个不同的作用域类型。变量的行为类型有普通变量行为模式、CONST 行为模式和 RETAIN 行为模式。

表 2-1 变量类型

序号	分类	类型名称	增加版本	备注
1	数字	VECT3	V0.1	
2		UDINT	V0.1	
3		LREAL	V0.1	
4		BOOL	V0.1	
5		DINT	V0.1	
6		STRING	V0.1	
7	目标点	POINTC	V0.1	
8		EPOINTC	V0.1	
9		EPOINTJ	V0.1	
10		POINTJ	V0.1	
11	运动参数	REFSYS	V0.1	
12		TOOL	V0.1	不可设置初始值
13		SPEED	V0.1	不可设置初始值
14		ZONE	V0.1	不可设置初始值
15	其他	TRIGGER	V0.1	不可设置初始值

16		INTR	V0.1	不可设置初始值
17		CLOCK	V0.1	不可设置初始值
18		TRACKING	V0.1	不可设置初始值
19		ROBOT	V0.1	不可设置初始值

2.2 变量可见性

2.2.1 工程

该可见性对应老版编译器 module 中 Local，此类变量可以被所有程序和子程序使用。不能使用相同的名字定义多个 工程变量。这些变量对于其他程序文件不可见。

2.2.2 全局

该可见性对应老版编译器 module 中 public。同工程变量相似，但是这个变量可以从其他程序文件中的程序看到。在其他程序文件中，调用改类型的格式为“程序文件名称.变量名”，例如 moduleName.variableName。

备注：只有当开发人员将本程序文件声明系统功能模块，并做相应处理后，全局变量才能发挥其全部作用，在非系统功能模块的程序文件中，全局变量可见性与工程变量相同。

2.2.3 系统

系统变量中显示两类可见性的变量，分别是：TASK 和 GLOBAL，对应老版编译器 module 中 TASK 和 GLOBAL。

备注：只有当开发人员将本程序文件声明系统功能模块，并做相应处理后，系统变量才能发挥其全部作用，在非系统功能模块的程序文件中，系统变量可见性与工程变量相同。

2.2.3.1 TASK

类似全局公共变量，在其他程序文件中，可以使用这种变量，而无需添加程序文件名称，例如 variableName。

2.2.3.2 GLOBAL

这种变量对于所有 TASK 系统都是通用的。在不同的 TASK 之间共享数据很有用。如果对同一 GLOBAL 有不同的定义，则报告错误。使用 GLOBAL 变量时，无需添加程序文件名称。

2.2.4 Main

此类型的变量只能在主程序 Main 中可见和使用，不能被其他程序和子程序可见和使用。可以在多个子程序中定义同一个变量名的变量，子程序中定义的变量只能被自己使用。

2.2.5 子程序的输入

输入变量用于定义子程序的输入参数。输入变量的使用类似局部变量但是其初始值来源于调用的程序。输入变量定义顺序与调用指令传递参数顺序一致。主程序中不能够使用此类变量。

2.2.6 子程序的输出

输出变量用于定义子程序的输出参数。输入变量的使用类似局部变量但是其初始值来源于调用的程序。输出变量定义顺序应与调用指令设置的参数顺序一致。主程序中不能够使用此类变量。

2.2.7 子程序的本地

此类型的变量只能在其定义的子程序中可见和使用，不能被其他程序和子程序可见和使用。可以在多个子程序中定义同一个变量名的变量。

2.3 变量行为类型

2.3.1 普通变量行为模式

这是变量的默认行为。可以在程序中设置变量，并在重新启动程序时丢失其值。

2.3.2 CONST 行为模式

具有此行为的变量无法在程序中更改，必须使用 Init 值进行设置。

2.3.3 RETAIN 行为模式

当程序从内存中卸载时重新启动并存储程序时，将保留变量的值。对于“工程”和“系统中的 TASK”变量，该值保存在程序中。

2.4 变量类型

2.4.1 VECT3 三维实向量

该类型的数据用于表示三维向量。该类型变量使用赋值指令调用函数 VECT3()对其进行赋值操作。

2.4.1.1 子元素

表 2-2 VECT3 三维实向量子元素表

序号	子元素名称	类型	默认值	说明
1	x	LREAL	0.0	
2	y	LREAL	0.0	
3	z	LREAL	0.0	

2.4.1.2 使用用例

```
VECT3 pose
...
pose := VECT3( 100, 200, 300) ;
```

2.4.2 UDINT 无符号整数

此类数据用于仅为正数的整数值。UDINT 值可以在 0 到 4294967295 之间。初始化后的默认值为 0。

表 2-3 UDINT 无符号整数

类型	范围	默认值	说明
UDINT	0～4294967295	0	

2.4.2.1 使用用例

```
DINT counter
...
counter := counter + 1 ;
```

2.4.3 LREAL 长实数

此类数据用于具有双精度的十进制数。其值是有符号值，在-1.7976931348623158e +308 和 1.7976931348623158e +308 之间，初始化后的默认值为 0.0。

表 2-4 LREAL 长实数

类型	范围	默认值	说明
LREAL	-1.7976931348623158e +308 ～1.7976931348623158e +308	0.0	

2.4.3.1 使用用例

```
LREAL width
...
width := 88.506 ;
```

2.4.4 BOOL 布尔

此类数据用于逻辑值。BOOL 值可以是 true 或 false。初始化后的默认值为 false。

表 2-5 BOOL 布尔

类型	范围	默认值	说明
BOOL	true 或 false	false	

2.4.4.1 使用用例

```
BOOL firstRun
...
IF firstRun = false THEN
firstRun:= true ;
...
END_IF ;
```

当 firstRun 为 false 时，执行后，被置为 true。

2.4.5 DINT 整数

此类数据用于整数值，可以是正数或负数。DINT 值可以在-2147483648 和 2147483647 之间。初始化后的默认值为 0。

表 2-6 DINT 整数

类型	范围	默认值	说明
DINT	-2147483648~2147483647	0	

2.4.5.1 使用用例

例子:

```
DINT counter
DINT diff
…
counter := counter + 1 ;
```

2.4.6 STRING 字符串

此类数据用于存储字符串。STRING 最多可包含 128 个字符。初始化后的默认值为空字符串。

2.4.6.1 使用用例

例子:

```
STRING str
…
str := “Hello world” ;
MESSAGE (“%1”, str) ;
```

在用户日志中包含内容为“Hello world”的字符串 str。

2.4.7 POINTC 笛卡尔空间位姿

这种类型的数据用于表示笛卡尔点。数据具有 LREAL 类型的 x，y，z，a，b，c 分量。分量 x，y，z 表示位置，a，b，c 表示具有 Euler 约定的方向。分量 a 是指 z 轴上的方向，b 是指 y 轴上的方向，c 是指 x 轴。要使用 6 个 LREAL 值组合此类数据才可以使用 POINTC 功能。POINTC 可用于笛卡尔和关节运动。初始化后的默认值是无效点（因此，如果在移动指令中使用则发出错误），所有分量 x，y，z，a，b，c 都设置为 0.0。

2.4.7.1 子元素

表 2-7 POINTC 子元素表

序号	子元素名称	类型	默认值	说明
1	x	LREAL	0.0	
2	y	LREAL	0.0	
3	z	LREAL	0.0	
4	a	LREAL	0.0	
5	b	LREAL	0.0	

6	c	LREAL	0.0	

2.4.7.2 使用用例

例子:

```
POINTC target
...
target := POINTC( 100, 200, 300, 10, 20, 30) ;
MLIN (target, v250, fine, tool0, wobj0) ;
```

2.4.8 EPOINTC

这种类型的数据是 POINTC 类型的 x，y，z，a，b，c 分量加上系统配置的附加轴的坐标。所有分量的默认值都为 0.0。

2.4.9 POINTJ 关节位置类型

这种类型的数据用于确定机器人关节的位置。数据具有 LREAL 类型的 j1，j2，j3，j4，j5，j6 组件。要设置此类数据，可以使用 6 个 LREAL 值的 POINTJ 函数。POINTJ 用于 MJOINT 指令，将机器人移动到关节位置定义的特定位置。POINTJ 不能用作笛卡儿运动的目标。初始化后的默认值是无效点（因此，如果在移动指令中使用则发出错误），所有组件 j1，j2，j3，j4，j5，j6 都设置为 0.0。

2.4.9.1 子元素

表 2-8 POINTJ 子元素表

序号	子元素名称	类型	默认值	说明
1	j1	LREAL	0.0	
2	j2	LREAL	0.0	
3	j3	LREAL	0.0	
4	j4	LREAL	0.0	
5	j5	LREAL	0.0	
6	j6	LREAL	0.0	

2.4.9.2 使用用例

```
POINTJ startpos
...
startpos:= POINTJ( 0, 0, 0, 0, -90, 0) ;
MJOINT(startpos, v500, fine, tool0) ;
```

2.4.10 EPOINTJ

这种类型的数据是 POINTJ 类型的 j1，j2，j3，j4，j5，j6 分量加上系统配置的附加轴的坐标。所有分量的默认值都为 0.0。

2.4.11 REFSYS 参考坐标系

这种类型的数据用于定义笛卡尔运动的坐标参考系。如果在移动指令中使用，则位置将基于指定坐标系。REFSYS 可以基于分层链，可以是固定的或可移动的。例如，可以使用需要父 REFSYS 和六个 LREAL 值的 REFSYS 函数。初始化后的默认值是 REFSYS，其中所有数据都设置为 0.0，与世界参考系统相同。

2.4.11.1 子元素

表 2-9 REFSYS 子元素表

序号	子元素名称	类型	默认值	说明
1	parent	LREAL	0.0	
2	x	LREAL	0.0	
3	y	LREAL	0.0	
4	z	LREAL	0.0	
5	a	LREAL	0.0	
6	b	LREAL	0.0	
7	c	LREAL	0.0	

2.4.11.2 使用用例

```
REFSYS wobj
POINTC p0
…
wobj:= REFSYS( wobj0, 100, 200, 50, 0, 0, 0) ;
p0 := POINTC(0, 0, 0, 0, 0, 0) ;
MLIN (p0, v500, fine, tool0, wobj) ;
```

在此示例中，机器人将移动到参考系统 wobj 的原点。

2.4.12 TOOL 工具类型

此类数据用于描述工具的参数。TOOL 数据类型用于移动指令。要设置此类数据，可以使用 STOOL 函数。初始化后的默认值是 TOOL，所有参数都设置为 0.0。

2.4.12.1 子元素

表 2-10 TOOL 子元素表

序号	子元素名称	类型	默认值	说明
1	x	LREAL	0.0	
2	y	LREAL	0.0	
3	z	LREAL	0.0	
4	a	LREAL	0.0	

5	b	LREAL	0.0	
6	c	LREAL	0.0	

2.4.12.2 使用用例

```
TOOL gun
…
gun := STOOL(0, 0, 100, 0, 0, 0) ;
…
MLIN (target, v250, fine, tool0) ;
MLIN (target, v250, fine, gun) ;
```

此示例可用于使用相同目标和两个不同工具检查机器人位置的不同。

2.4.13 SPEED 速度类型

SPEED 数据类型用于指定移动指令中的目标速度。它包含以 mm / s 表示的切向速度和以度/ s 表示的定向速度。要将数据存储在 SPEED 类型的变量中，可以使用 SSPEED 函数。初始化后的默认值为 SPEED，所有参数均设置为 0.0。

2.4.13.1 子元素

表 2-11 SPEED 子元素表

序号	子元素名称	类型	默认值	说明
1	v_tcp	LREAL	0.0	
2	v_ori	LREAL	0.0	

2.4.13.2 使用用例

```
SPEED vel
…
vel := SSPEED( 250, 5) ;
MLIN (target, vel, fine, tool0) ;
```

在此示例中，机器人将以 250 mm / s 和 5 度/ s 的目标速度移动。

例子:

```
SPEED vel
…
vel := v250 ;
…
MLIN (target1, vel, fine, tool0) ;
MLIN (target2, vel, fine, tool0) ;
MLIN (target3, vel, fine, tool0) ;
```

此示例显示如何使用变量 vel 中定义的预定义速度设置为一组运动定义目标速度。

2.4.14 ZONE 过渡混成类型

ZONE 数据类型用于指定两个连续移动指令之间的混合参数。它包含以 mm 为单位的线性距离和以度为单位的重定向角距离。要设置此类型的数据，可以使用 SZONE 功能。初始化后的默认值为 ZONE，所有参数均设置为 0.0。

2.4.14.1 子元素

表 2-12 ZONE 子元素表

序号	子元素名称	类型	默认值	说明
1	mov.tcp	LREAL	0.0	
2	reori.ori	LREAL	0.0	

2.4.14.2 使用用例

```
ZONE blend
…
blend := SZONE(100, 5) ;
…
MLIN (target1, v250, blend, tool0) ;
MLIN (target2, v250, fine, tool0) ;
```

在此示例中，target2 的移动在机器人到达目标 1 之前从 100 mm 开始。

2.4.15 TRIGGER 触发类型

TRIGGER 数据类型用于存储与移动指令相关的事件中涉及的数据。通过指令，TRIGSET 可以将数据存储在 TRIGGER 变量中，指令 TRIGON 可以在执行运动指令之前激活触发器。初始化后的默认值是无效触发器。

2.4.15.1 子元素

表 2-13 TRIGGER 子元素表

序号	子元素名称	类型	默认值	说明
1	Type	STRING		
2	ParType	LREAL	0.0	
3	Var	STRING		
4	Value	LREAL	0.0	

2.4.15.2 使用用例

```
TRIGGER trig
…
trig := TRIGSET when Distance is 20 do (io.output[5] := true) ;
…
TRIGON (trig) ;
MLIN (target, v500, fine, tool1, wobj1) ;
```

在此示例中，当机器人在目标之前 20 mm 时，io.output [5]设置为 true。

2.4.16 INTR 中断处理类型

这种类型的数据用于中断处理。要将此类型的变量与特定中断例程绑定，必须使用 INTRSET 指令。可以使用指令 INTRCOND 或 INTRERRNO 指定导致中断的条件。请注意，此类数据必须在程序执行时只设置一次，否则将发出错误。有关其他信息，请参阅以 INTR 开头的指令并指导“如何使用中断”。初始化后的默认值是无效的 INTR。

2.4.16.1 子元素

表 2-14 INTR 子元素表

序号	子元素名称	类型	默认值	说明
1	Subroutine	STRING		
2	Num	UDINT	0	

2.4.16.2 使用用例

```
INTR intr1
BOOL firstRun
EFORT 机器人 C30 系统指令说明手册
11
All rights reserved. @Efort Intelligent Equipment Ltd. Company
…
(* set interrupt only once *)
IF NOT firstRun THEN
firstRun := true ;
INTRSET (intr1, intHnd()) ;
INTRCOND (intr1, io.input[8]) ;
END_IF ;
…
```

此示例显示如何绑定中断以及如何设置触发该中断的条件。

2.4.17 CLOCK 时钟类型

CLOCK 数据类型用于时间测量。时间测量以秒表示。无法直接设置或读取此类数据。可以使用 CLOCKRESET 指令重置时间测量。要开始测量必须使用指令 CLOCKSTART 和 CLOCKSTOP 来停止测量。对于读取必须使用 CLOCKREAD 函数返回 LREAL 值。初始化后的默认值是 0.0 秒的测量值。

2.4.17.1 子元素

表 2-15 CLOCK 子元素表

序号	子元素名称	类型	默认值	说明
1	elapsed	LREAL	0	

2.4.17.2 使用用例

```
CLOCK clk1
…
CLOCKRESET (clk1) ;
CLOCKSTART (clk1) ;
…
CLOCKSTOP (clk1) ;
MESSAGE (“Time elapsed %1”, CLOCKREAD(clk1)) ;
```

此示例显示如何测量执行指令期间所经过的时间。

2.4.18 TRACKING

TRACKING 数据类型用于存储跟踪应用程序的数据。要设置此类数据必须使用函数 TRACKING。有关参数的说明，请参阅跟踪应用说明文档。初始化后的默认值是无效的跟踪。

2.4.18.1 子元素

表 2-16 TRACKING 子元素表

序号	子元素名称	类型	默认值	说明
1	maxspace	LREAL	0.0	
2	startspace	LREAL	0.0	
3	maxspe	LREAL	0.0	
4	maxacc	LREAL	0.0	
5	type	STRING		
6	par1	LREAL	0.0	

2.4.18.2 使用用例

```
TRACKING cvy
REFSYS wobj
REFSYS rsPhoto
…
cvy := TRACKING(1000, 700, 100, 1000, Linear, 0.1) ;
…
wobj := GETTRKWOBJ(rsPhoto, cvy) ;
…
MLIN (target, v500, fine, tool1, wobj) ;
…
```

此示例显示如何为跟踪应用程序设置跟踪参数。

2.4.19 ROBOT 机器人

此类数据用于与系统中定义的轴组进行交互。例如，可以设置轴组的参考系。要设置此类型的数据，必须使用函数 ROBOT，该函数需要作为参数的字符串，其名称为 axis group。初始化后的默认值是无效的 ROBOT。

2.4.19.1 子元素

表 2-17 ROBOT 子元素表

序号	子元素名称	类型	默认值	说明
1	name	STRING		
2	kmaxt_speed	LREAL	0.0	
3	kmaxt_acc	LREAL	0.0	
4	kmaxt_dec	LREAL	0.0	
5	kmaxt_jerk	LREAL	0.0	
6	kmaxw_speed	LREAL	0.0	
7	kmaxw_acc	LREAL	0.0	
8	kmaxw_dec	LREAL	0.0	
9	kmaxw_jerk	LREAL	0.0	
10	kmaxj_speed	LREAL	1.0	
11	kmaxj_acc	LREAL	1.0	
12	kmaxj_dec	LREAL	1.0	
13	kmaxj_jerk	LREAL	1.0	

2.4.19.2 使用用例

```
ROBOT conveyor
REFSYS cvywobj
POINTC cvyOrigin
...
conveyor := ROBOT(“conveyor”) ;
cvywobj:= SETROBOTWOBJ(conveyor, “”, cvyOrigin) ;
...
```

在这个例子中，它设置了传送带的参考系统。

完成此设置是为了在程序中指定传送带的参考系统，例如可以在跟踪应用程序中使用。

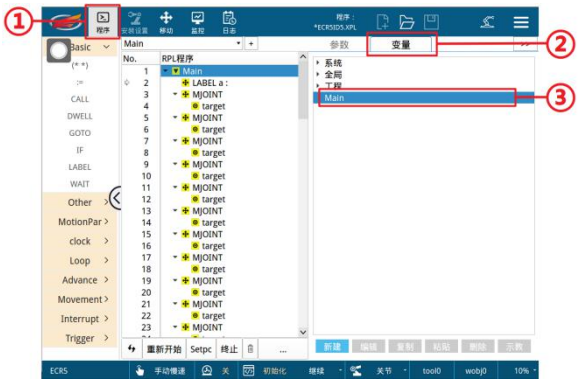
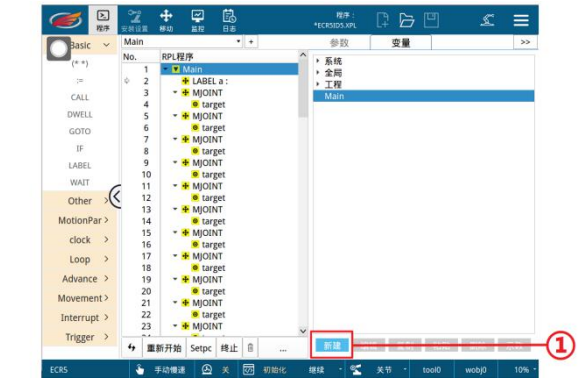
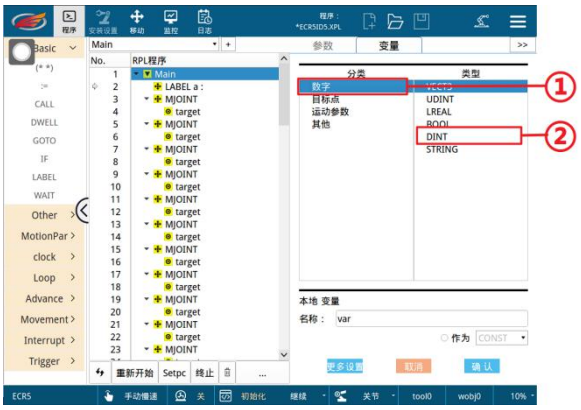
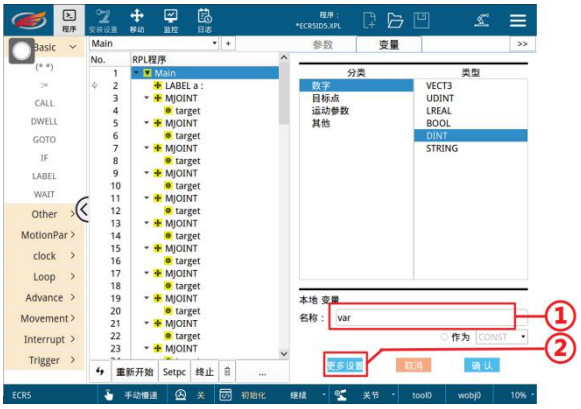
2.5 变量操作

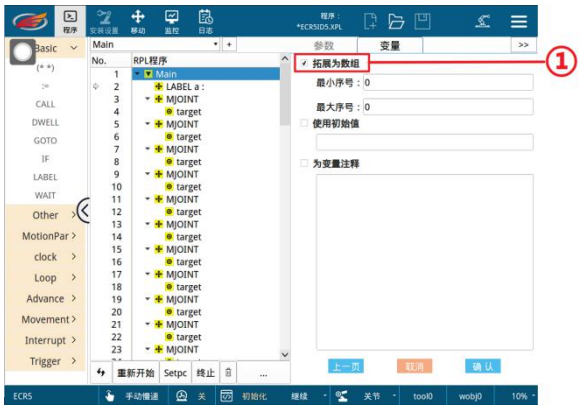
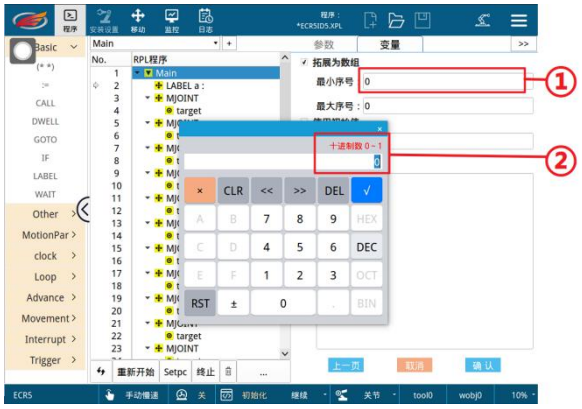
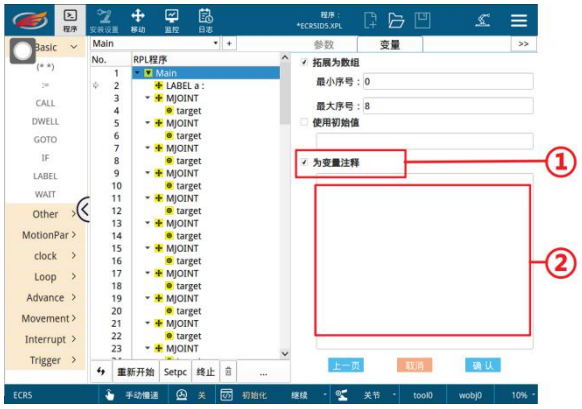
2.5.1 新建变量

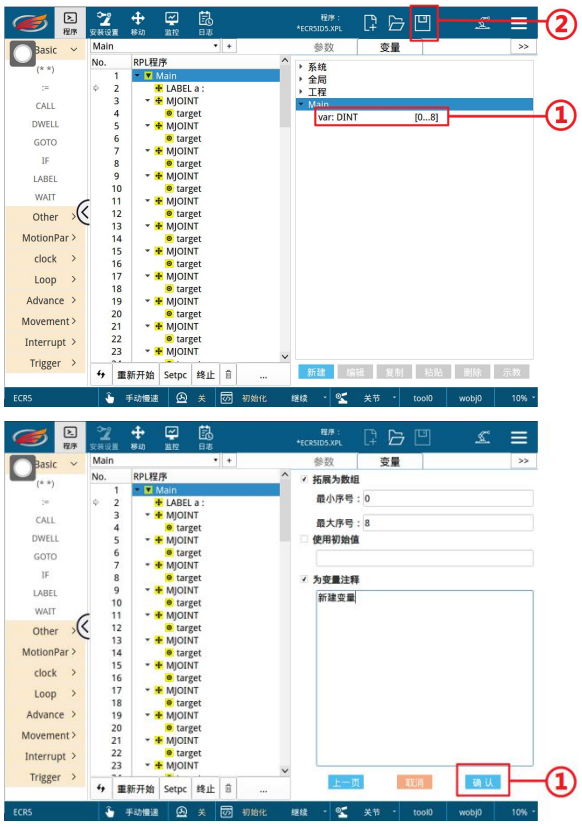
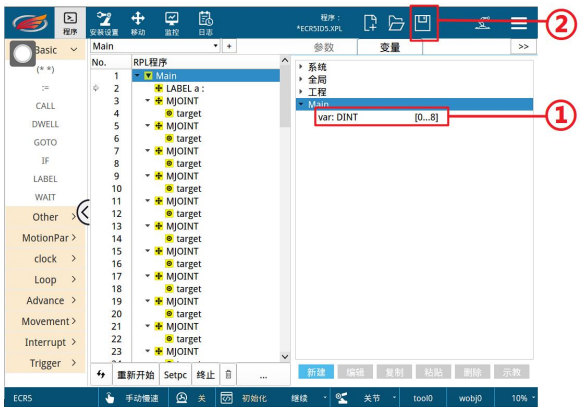
以在 Main 中添加 DINT 类型变量为例，介绍添加变量的步骤。

表 2-18 新建变量步骤

步骤	图示	说明

1.点击任务栏程序（图中①）进入程序界面，点击右侧参数区标签栏中的变量标签（图中②）进入变量管理页，点击 Main 节点（图中③），选中该节点。		1. 如若编辑程序内容请确保已登录用户具有编程权限。 2. 当选中 Main 节点（图中③）后，如果当前登录用户拥有编程权限，功能按钮栏中的“新建”按钮将被激活
2．点击变量功能按钮栏中的“新建”按钮（图中①），进入新建变量设置页面。		
3.先点击选中变量类型分类中的“数字”节点（图中①），然后点击选中右侧类型栏中的“DINT”节点（图中②）		
4.点击变量名输入框（图中①）键入新的变量名称（本例默认），完成后点击“更多设置”按钮（图中②）进入变量额外信息设置页。		新建变量时，系统会自动生成一个可用的变量名 var**，其中**为数字或没有。

5.点击勾选拓展为数组复选框（图中①），激活数组序号范围输入框。		
6.点击最小序号输入框（图中①），并在弹出键盘（图中②）中输入序号的最小值。		数组序号的最小值只能在 0 和 1 之间选择。
7.点击最大序号输入框（图中①），并在弹出键盘（图中②）中输入数组序号的最大值。		数组序号的最大值，必须大于数组序号的最小值，不然设置无效。 数组序号的最大值必须小于 99。
8. 点击勾选“为变量注释”复选框，激活注释输入框（图中②）。点击注释输入框在弹出键盘中输入注释内容。		

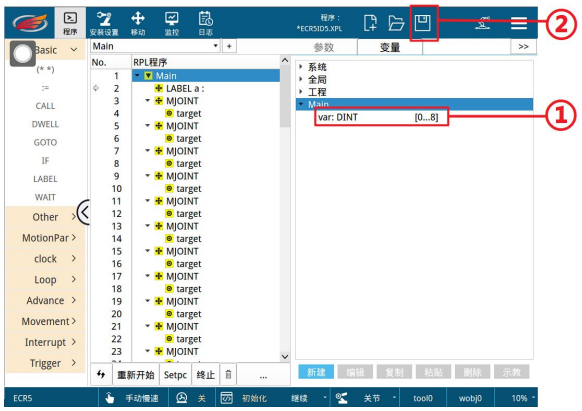
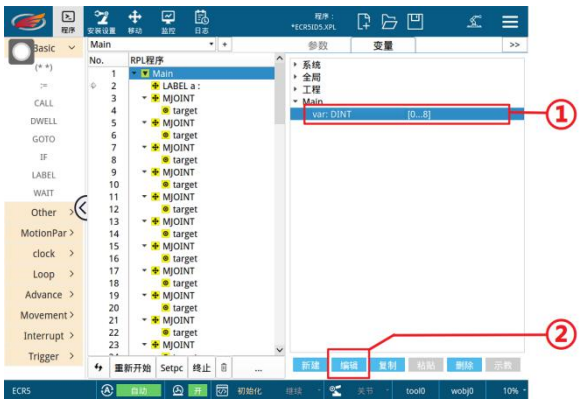
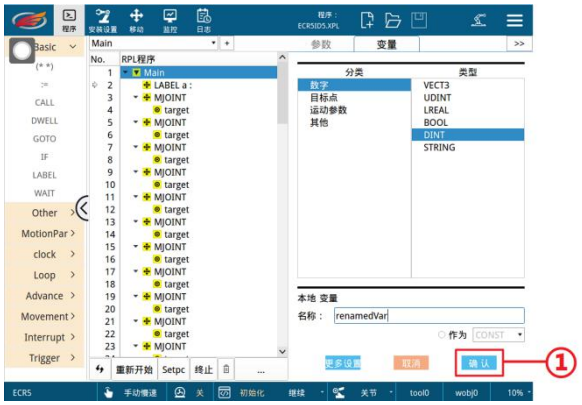
9.点击“确定”按钮，完成新建变量的操作。		
10.图中①即为新建变量。点击任务栏“保存”按钮，将当前修改保存至文件程序文件中。		如不保存程序文件，机器人掉电后，修改将丢失。

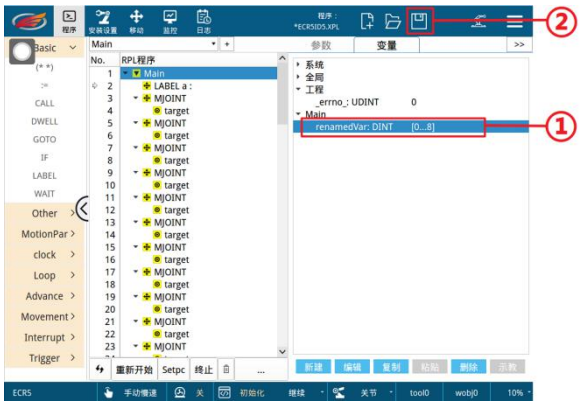
2.5.2 编辑变量

编辑变量的步骤，以修改 2.5.1 节中添加的 DINT var[0-8]变量的名称为例。

表 2-19 编辑变量步骤

步骤	图示	说明

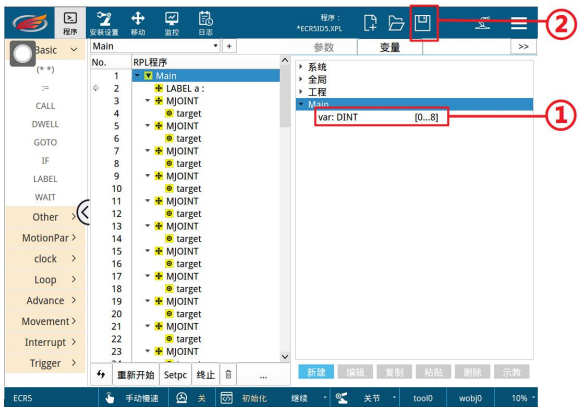
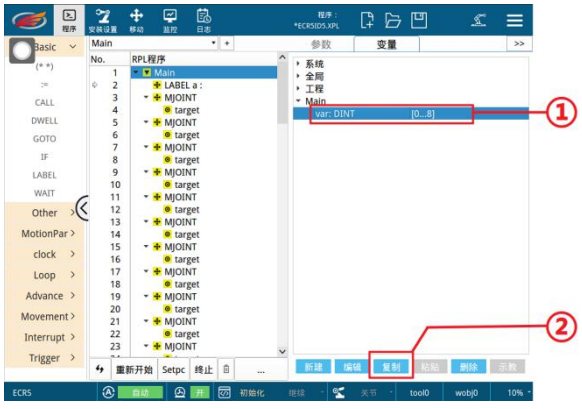
1.按照 2.5.1 节中步骤添加变量 DINT var[0-8]		如若编辑程序内容请确保已登录用户具有编程权限
2．点击选中新建的变量 var 行（图中①），点击“编辑”按钮（图中②）进入变量编辑页面。		
3.点击变量名输入框（图中①），在弹出键盘中输入新的变量名称，本例中输入“renamedVar”。		
4.点击页面下方的“确认按钮”（图中①），确认对变量 var 的修改。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

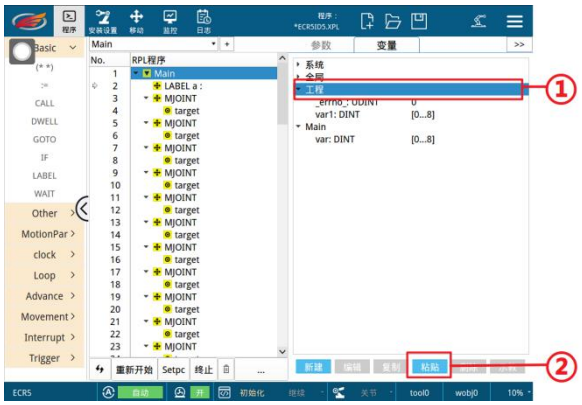
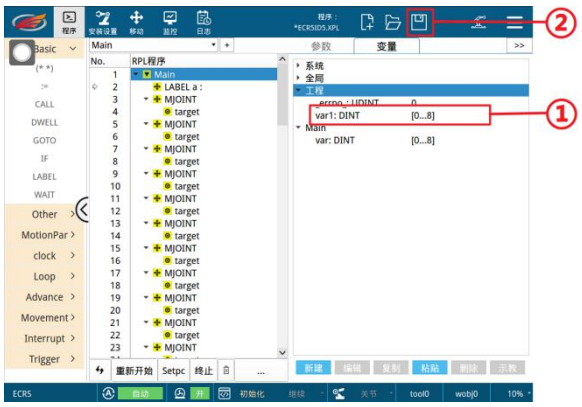
5.图中①即为修改后的变量，点击任务栏的“保存”按钮（图中②）将修改保存至程序文件。		

2.5.3 复制变量

本节以章节 2.5.1 为基础，将以复制 2.5.1 节新建的变量 var 到工程中为例，介绍变量的复制操作。

表 2-20 复制变量步骤

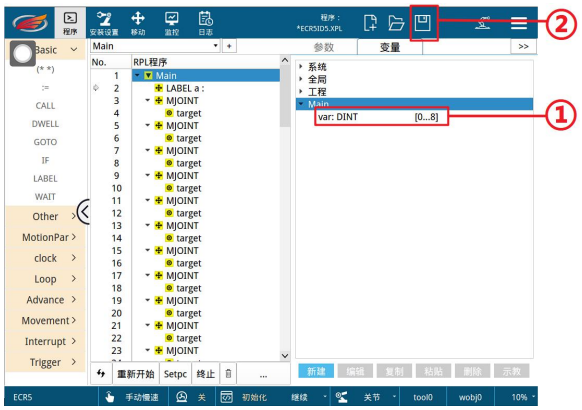
步骤	图示	说明
1.按照 2.5.1 节中步骤添加变量 DINT var[0-8]		如若编辑程序内容请确保已登录用户具有编程权限
2．点击选中新建的变量 var 行（图中①），点击“复制”按钮（图中②）将选中变量的信息保存至内存。		点击“复制”按钮后，如果操作正确，“粘贴”按钮将被激活。

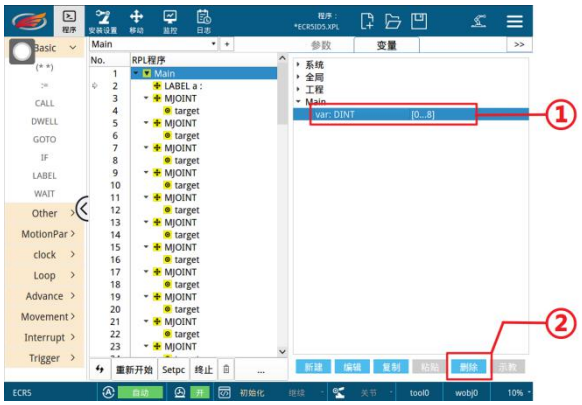
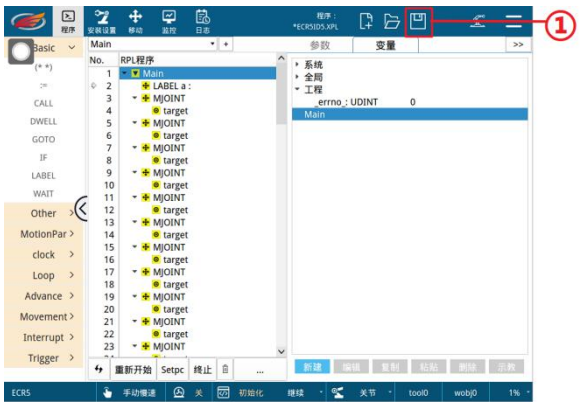
3.选中“工程”节点（图中①），点击“粘贴”按钮（图中②）处，将复制的变量复制到该处。		复制变量时，新变量的名称为原变量的左侧非数字部分+(原变量结尾数字+1)，如：原变量名为：var12，新的变量名为：var13（若可用）；若原来名称结尾无数字，则自动在原变量名结尾添加数字1，如本例中的结果。
4.图中①即为粘贴得到的变量，点击任务栏的“保存”按钮（图中②）将修改保存至程序文件。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

2.5.4 删除变量

本节以章节 2.5.1 为基础，将以删除 2.5.1 节新建的变量 var 到工程中为例，介绍变量的删除操作。

表 2-21 删除变量步骤

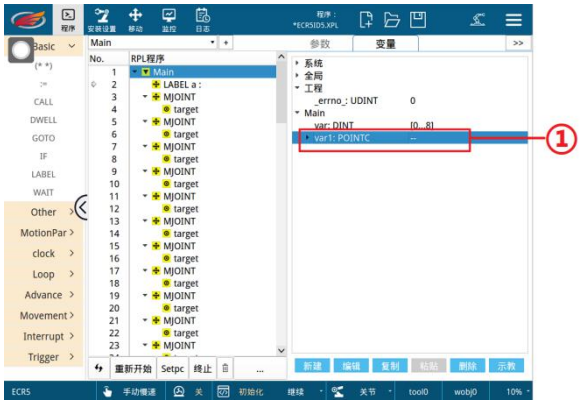
步骤	图示	说明
1.按照 2.5.1 节中步骤添加变量 DINT var[0-8]		如若编辑程序内容请确保已登录用户具有编程权限

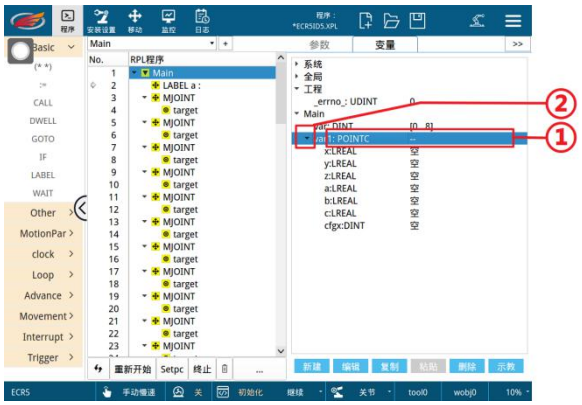
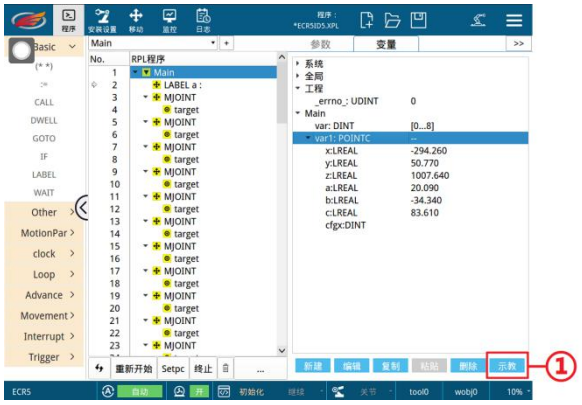
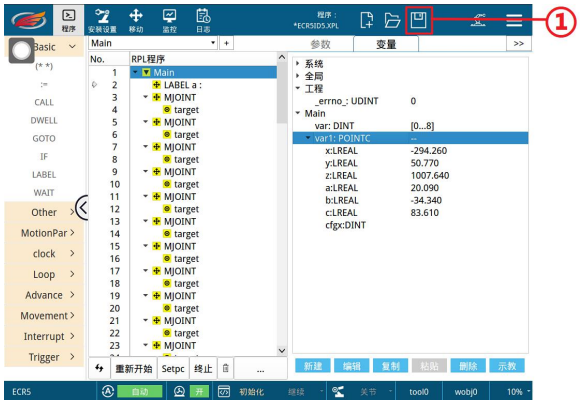
2．点击选中新建的变量 var 行（图中①），点击“删除”按钮（图中②）将选中变量删除。		变量被删除时，将自动选择其上一个仍存在的节点。
3.点击任务栏的“保存”按钮（图中①）将修改保存至程序文件。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

2.5.5 示教目标点

本节以章节 2.5.1 为基础，首先按照章节 2.5.1 步骤添加 POINTC 类型变量 var1，并以示教 var1 为例介绍示教目标定类型变量的步骤。

表 2-22 示教目标点类型变量步骤

步骤	图示	说明
1.按照 2.5.1 节中步骤添加变量 PONTC var1（图中①）		如若编辑程序内容请确保已登录用户具有编程权限。 不要新建数组类型的 POINTC，目前编辑器不支持示教数组类型的目标点类型变量。

2．点击选中新建的变量 var1 行（图中①），点击行左侧的折叠标签（图中②），将节点 var1 展开。		未为变量设置初始值时，在程序处于初始化状态下，变量的值将为空。
3.点击下方功能按钮栏中的“示教”按钮（图中①），就可以为 POINTC 点 var1 完成示教。		
4.点击任务栏的“保存”按钮（图中①）将修改保存至程序文件。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

第 3 章　程序树

3.1 本章简介

本章主要介绍程序树显示区域为用户提供的功能，及实现操作。

程序树显示区域为左侧指令栏和右侧参数显示区中间的显示区域，如图 3-1 中所示，图中①区域为程序树显示区，图中②为程序树。在程序树显示区，用户可以进行查看当前程序的所有指令，切换、新建、删除子程序，刷新编程界面，设置程序开始执行位置，终止程序，以及删除指令等操作。

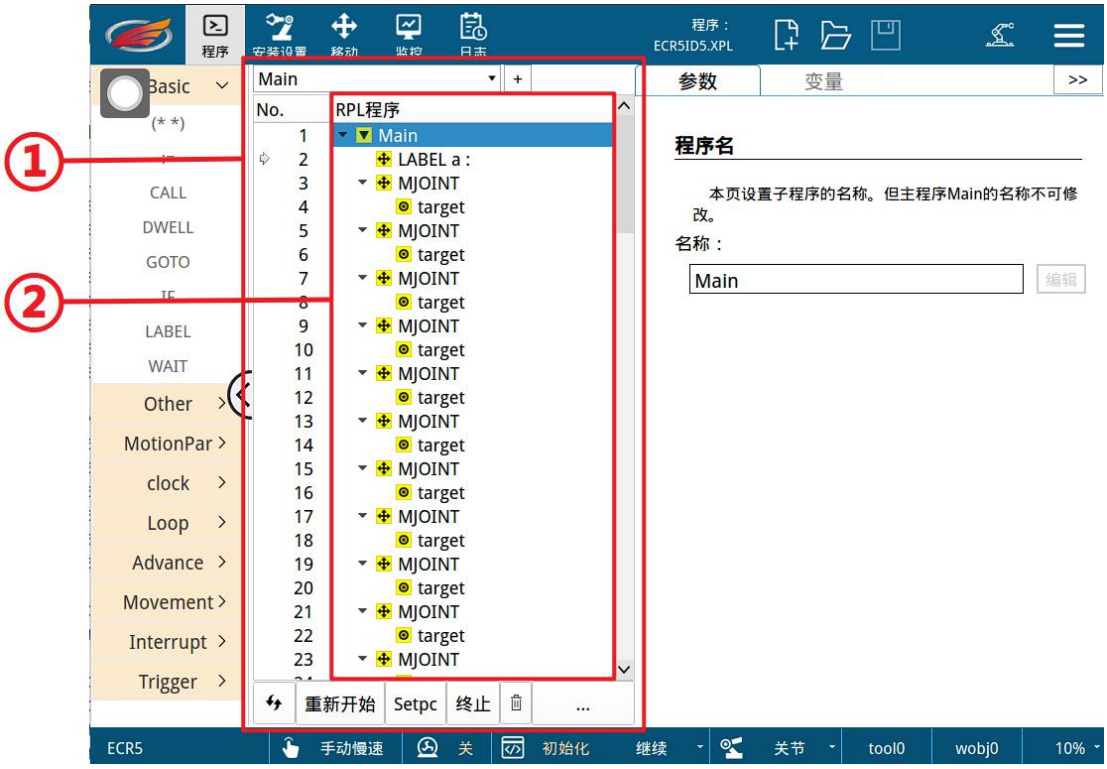


图 3-1 程序树显示区域

3.2 新建子程序

本节介绍如何在程序树显示区中，通过程序名右侧的新建子程序按钮  来新建子程序。新建子程序按钮所在位置如图 3-2 中①所示。

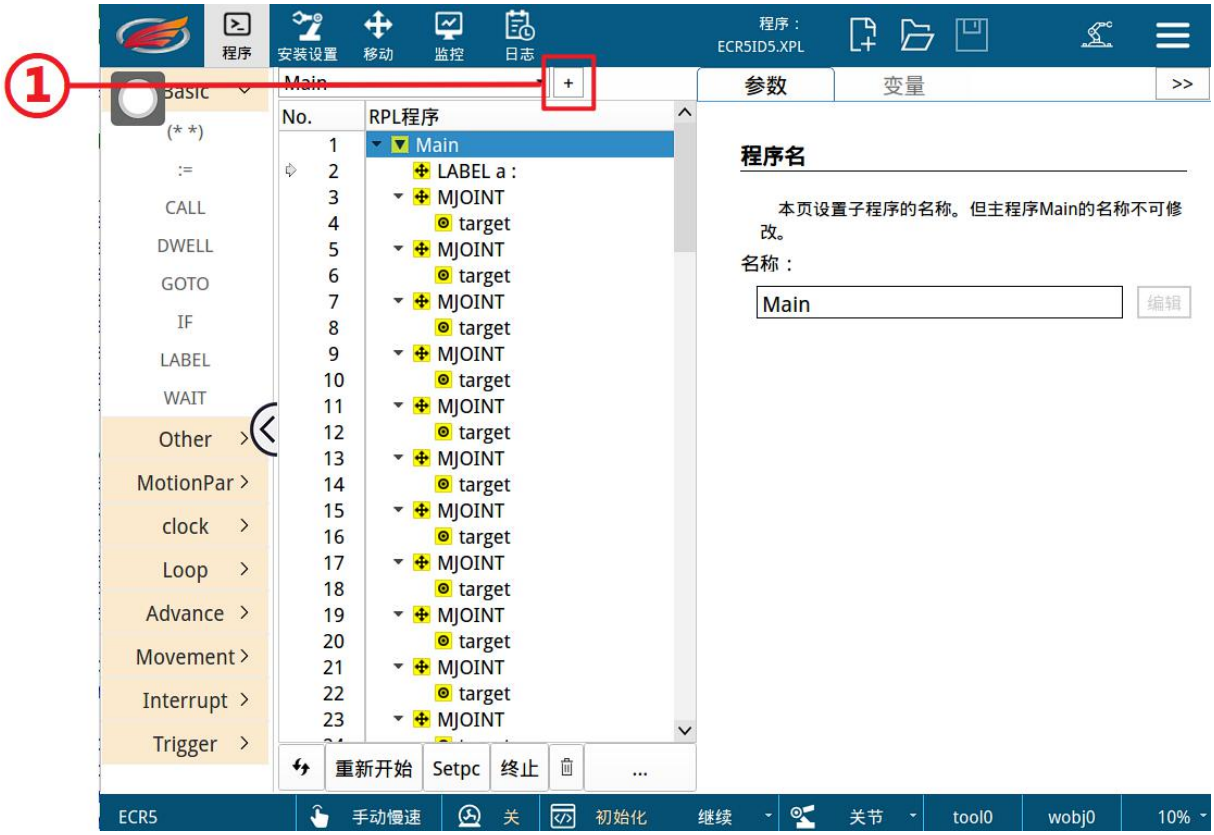
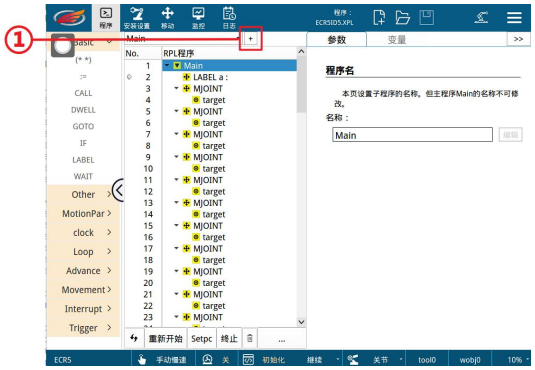
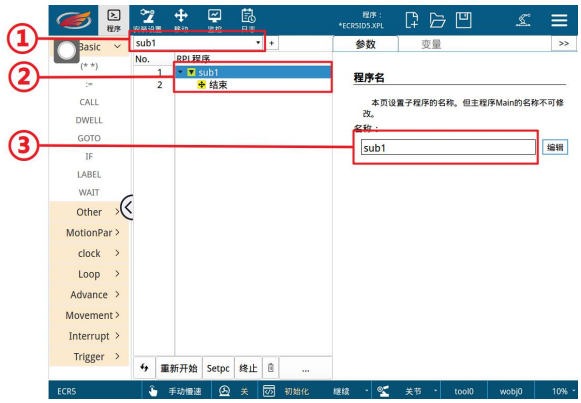
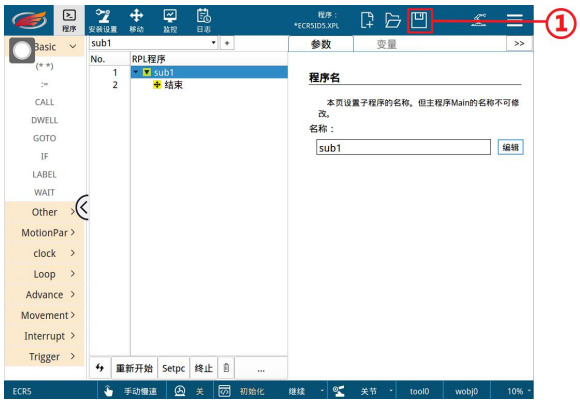


图 3-2 新建子程序按钮位置

表 3-1 新建子程序步骤

步骤	图示	说明
1.用户登录，点击任务栏中的“程序”快捷键，进入编程界面。		如若编辑程序内容请确保已登录用户具有编程权限。
2．点击新建子程序按钮（图中①），就可新建子程序。		点击“新建子程序”按钮后，系统会自动添加子程序，并自动生成可用的子程序名称，其命名格式为：sub+n。

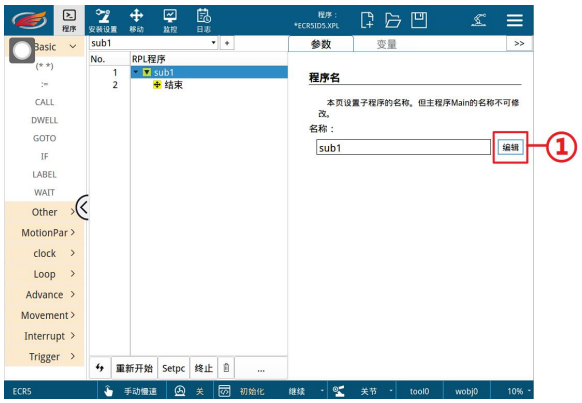
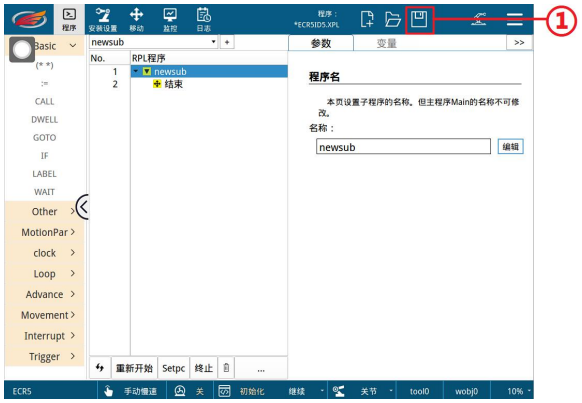
3.新建子程序后，程序选择下拉框（图中①）会自动切换到刚刚新建的子程序，程序树（图中②）也会显示新建子程序的空白指令树，参数页会显示程序名页（图中③）。		
4.点击任务栏的“保存”按钮（图中①）将修改保存至程序文件。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

3.3 重命名子程序

本节将在章节 3.2 的基础上，以重命名章节 3.2 中新建的子程序 sub1 为 newSub 为例，介绍如何重新命名子程序。

表 3-2 重命名子程序步骤

步骤	图示	说明
1.用户登录，按照章节 3.2 新建 sub1 子程序。		如若编辑程序内容请确保已登录用户具有编程权限。

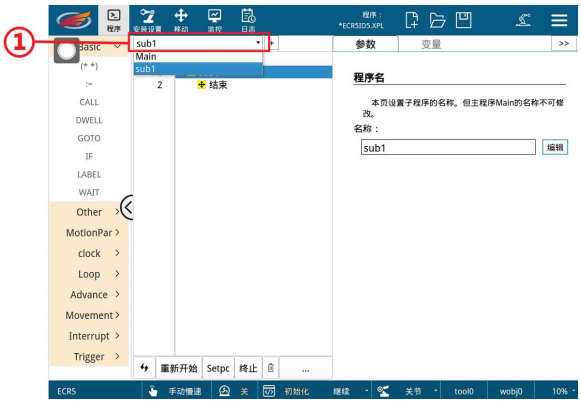
2．点击程序名参数页中的“编辑”按钮（图中①），在弹出键盘中输入新的程序名“newSub”并确定。		键盘确定输入后，程序名如果可用，则程序名选择框和程序树中的子程序会同步修改。
3.点击任务栏的“保存”按钮（图中①）将修改保存至程序文件。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

3.4 切换当前程序

本章节将在章节 3.2 的基础上，新建子程序 sub1 后，以将程序树有 sub1 切换到 Main 为例，介绍程序的切换操作。

表 3-3 切换当前程序步骤

步骤	图示	说明
1.用户登录，按照章节 3.2 新建 sub1 子程序。		如若编辑程序内容请确保已登录用户具有编程权限。

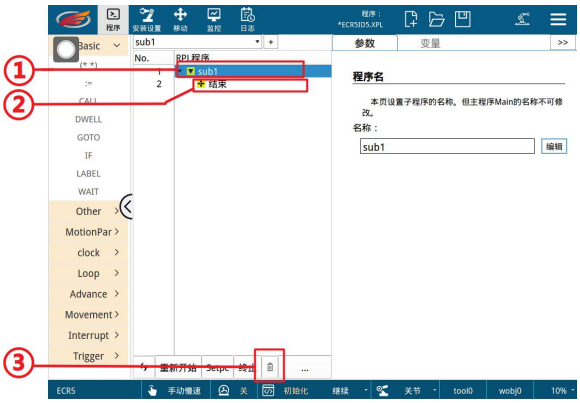
2.点击程序名程序名选择下拉框（图中①），在弹出程序名候选列表中点击 Main。		
3.程序树会显示 Main 的所有指令。		

3.5 删除子程序

本章节将在章节 3.2 的基础上，新建子程序 sub1 后，以新建的子程序 sub1 删除为例，介绍子程序的删除操作。

表 3-4 删除子程序步骤

步骤	图示	说明
1.用户登录，按照章节 3.2 新建 sub1 子程序。		如若编辑程序内容请确保已登录用户具有编程权限。

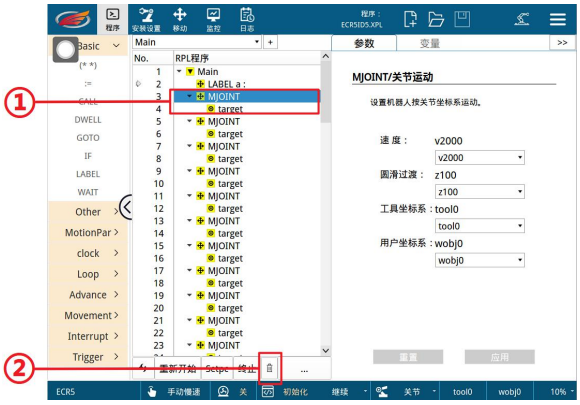
2.点击选中程序树中的程序名行（图中①）或程序树的最后一行--结束行（图中②），点击程序树下方的“删除”按钮（图中③）来删除子程序sub1。		
3.程序树会自动切换至程序下拉框中排序前一位的程序，本例中是Main。		
4.点击任务栏的“保存”按钮（图中①）将修改保存至程序文件。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

3.6 删除指令

删除指令的操作步骤与删除子程序的操作步骤相同，区别在于：删除子程序时，选中的是程序名行或者程序的结束行，而删除指令选中的将要被删除的指令行。

表 3-5 删除指令步骤

步骤	图示	说明

1.用户登录，打开非空程序文件。		如若编辑程序内容请确保已登录用户具有编程权限。
2.假如要删除第三行的 MJOINT 指令（图中①），则首先点击选中第三行的 MJOINT，然后点击程序树下方的“删除”按钮（图中②）。		系统删除指令后，将自动选择被删除指令的上一行。
3.点击任务栏的“保存”按钮（图中①）将修改保存至程序文件。		点击确定后，变量的名称会被改变，页面自动跳转到变量的展示页。

服务热线：400-0528877

本产品的额定功率、规格、外部尺寸等如需改良而进行变更，恕不另行通告。技术数据和插图仅作为供货参考，保留更改权利。

埃夫特智能装备股份有限公司

EFORT INTELLIGENT EQUIPMENT CO.,LTD
中国安徽省芜湖市鸠江经济开发区万春东路 96 号
No 96,Wanchun Road,Jiujiang Economic Development Zone,
Wuhu, Anhui,China