ROBOTICS

应用手册
控制器软件 OmniCore

Trace back information:
Workspace 22A version a10
Checked in 2022-03-02
Skribenta version 5.4.005

应用手册

控制器软件 OmniCore

RobotWare 7.6

文档编号: 3HAC066554-010
修订: J

本手册中包含的信息如有变更，恕不另行通知，且不应视为 ABB 的承诺。 ABB 对本手册中可能出现的错误概不负责。

除本手册中有明确陈述之外，本手册中的任何内容不应解释为 ABB 对个人损失、财产损坏或具体适用性等做出的任何担保或保证。

ABB 对因使用本手册及其中所述产品而引起的意外或间接伤害概不负责。

保留以备将来参考。

可从 ABB 处获取此手册的额外复印件。

本出版物为译本。

目录

手册概述 11
RobotWare 中的开源和第三方组件 13

1 RobotWare介绍 15

2 RobotWare-OS 17

2.1 Advanced RAPID 17
2.1.1 Advanced RAPID介绍 17
2.1.2 位功能 18
2.1.2.1 概述 18
2.1.2.2 RAPID组件 19
2.1.2.3 位功能的示例 20
2.1.3 数据搜索功能 21
2.1.3.1 概述 21
2.1.3.2 RAPID组件 22
2.1.3.3 数据搜索功能的示例 23
2.1.4 别名 I/O 信号 24
2.1.4.1 概述 24
2.1.4.2 RAPID组件 25
2.1.4.3 别名I / O功能的示例 26
2.1.5 配置功能 27
2.1.5.1 概述 27
2.1.5.2 RAPID组件 28
2.1.5.3 配置功能的示例 29
2.1.6 断电功能 30
2.1.6.1 概述 30
2.1.6.2 RAPID组件与系统参数 31
2.1.6.3 断电功能的示例 32
2.1.7 工艺配套功能 33
2.1.7.1 概述 33
2.1.7.2 RAPID组件 34
2.1.7.3 工艺配套功能的示例 35
2.1.8 中断功能 37
2.1.8.1 概述 37
2.1.8.2 RAPID组件 38
2.1.8.3 中断功能的示例 39
2.1.9 用户消息功能 40
2.1.9.1 概述 40
2.1.9.2 RAPID组件 41
2.1.9.3 用户消息功能的示例 42
2.1.9.4 文本表格文件 44
2.1.10 RAPID配套功能 45
2.1.10.1 概述 45
2.1.10.2 RAPID组件 46
2.1.10.3 RAPID配套功能的示例 47
2.2 Analog Signal Interrupt 48
2.2.1 Analog Signal Interrupt介绍 48
2.2.2 RAPID组件 49
2.2.3 代码示例 50
2.3 Connected Services 51
2.3.1 概述 51
2.3.2 Connected Services 连通性 53
2.3.3 Connected Services 注册 54
2.3.4 FlexPendant 中 Connected Services 路径汇总 56
2.3.5 RobotStudio 中 Connected Services 路径汇总 57
2.3.6 配置 - 系统参数 58

2.3.7 使用 FlexPendant 配置 Connected Services 61
2.3.7.1 简介 61
2.3.7.2 使用 FlexPendant 启用或禁用 Connected Services 62
2.3.7.3 根据连接类型使用 FlexPendant 配置 Connected Services 63
2.3.7.4 使用 FlexPendant 配置公共网络 65
2.3.7.5 使用 FlexPendant 配置与代理的互联网连接 66
2.3.8 使用 RobotStudio 配置 Connected Services 67
2.3.8.1 简介 67
2.3.8.2 使用 RobotStudio 启用或禁用连接服务。 68
2.3.8.3 根据连接类型使用 RobotStudio 来配置连接服务 69
2.3.8.4 使用 RobotStudio 配置公共网络 72
2.3.8.5 使用 RobotStudio 配置与代理的互联网连接 73
2.3.9 Connected Services 信息 74
2.3.10 故障排除 85
2.3.10.1 服务器连通性故障排除 85
2.3.10.2 3G/Wi-Fi 连通性故障排除 87
2.3.10.3 如何从控制器获得 CSE（嵌入式互联服务）日志 88
2.3.10.4 嵌入式互联服务故障排除日志 89
2.3.11 网络拓扑方案 91
2.4 Cyclic bool 101
2.4.1 循环评估逻辑条件 101
2.4.2 Cyclic bool示例 103
2.4.3 系统参数 106
2.4.4 RAPID组件 107
2.5 Device Command Interface 108
2.5.1 Device Command Interface介绍 108
2.5.2 RAPID部件和系统参数 109
2.5.3 代码示例 110
2.6 File and I/O device handling 112
2.6.1 文件和 I/O 设备处理介绍 112
2.6.2 基于二进制和字符的通信 113
2.6.2.1 概述 113
2.6.2.2 RAPID组件 114
2.6.2.3 代码示例 115
2.6.3 原始数据通信 117
2.6.3.1 概述 117
2.6.3.2 RAPID组件 118
2.6.3.3 代码示例 119
2.6.4 文件与目录管理 121
2.6.4.1 概述 121
2.6.4.2 RAPID组件 122
2.6.4.3 代码示例 123
2.7 Fixed Position Events 125
2.7.1 概述 125
2.7.2 RAPID组件与系统参数 126
2.7.3 代码示例 128
2.8 Logical Cross Connections 130
2.8.1 Logical Cross Connections介绍 130
2.8.2 配置Logical Cross Connections 131
2.8.3 示例 132
2.8.4 限制 134
2.9 RAPID Message Queue 135
2.9.1 RAPID Message Queue介绍 135
2.9.2 RAPID消息队列行为 136
2.9.3 系统参数 139
2.9.4 RAPID组件 140
2.9.5 代码示例 141
2.10 Socket Messaging 145
2.10.1 Socket Messaging介绍 145

2.10.2 套接字通信的示意图 146
2.10.3 关于Socket Messaging的技术事实 147
2.10.4 RAPID组件 148
2.10.5 Socket Messaging的代码示例 150
2.11 User logs 152
2.11.1 User logs介绍 152

3 Motion Performance 153

3.1 Absolute Accuracy [3101-x] 153
3.1.1 关于Absolute Accuracy 153
3.1.2 有用工具 155
3.1.3 配置 156
3.1.4 维护 157
3.1.4.1 影响准确度的维护 157
3.1.4.2 丧失准确度 159
3.1.5 补偿理论 160
3.1.5.1 错误来源 160
3.1.5.2 Absolute Accuracy补偿 161
3.1.6 Absolute Accuracy机器人的准备 163
3.1.6.1 ABB校准进程 163
3.1.6.2 出厂证书 165
3.1.6.3 配置参数 166
3.1.7 围笼的对准 167
3.1.7.1 概述 167
3.1.7.2 测量固定装置的对准情况 168
3.1.7.3 测量机器人的对准情况 169
3.1.7.4 框架关系 170
3.1.7.5 工具校准 171
3.2 Advanced Robot Motion 3100-1 172
3.3 Advanced Shape Tuning [包含在 3100-1 中] 173
3.3.1 关于Advanced Shape Tuning 173
3.3.2 自动微调摩擦 174
3.3.3 手动微调摩擦 176
3.3.4 系统参数 177
3.3.4.1 系统参数 177
3.3.4.2 设置微调系统参数 178
3.3.5 RAPID组件 179
3.4 Motion Process Mode [包含在 3100-1 中] 180
3.4.1 关于Motion Process Mode 180
3.4.2 用户定义的模式 182
3.4.3 关于机器人微调的一般信息 184
3.4.4 附加信息 186
3.5 Wrist Move [包含在 3100-1 中] 187
3.5.1 Wrist Move介绍 187
3.5.2 切割面框架 188
3.5.3 RAPID组件 189
3.5.4 RAPID代码示例 190
3.5.5 故障排除 192

4 Motion Supervision 193

4.1 World Zones [3106-1] 193
4.1.1 概述 World Zones 193
4.1.2 RAPID组件 195
4.1.3 代码示例 197
4.2 Collision Detection [3107-1] 199
4.2.1 概述 199
4.2.2 限制 200
4.2.3 碰撞时的情况 201
4.2.4 附加信息 203

4.2.5 配置和编写设施 204
4.2.5.1 系统参数 204
4.2.5.2 RAPID组件 206
4.2.5.3 Signals 207
4.2.6 何时使用Collision Detection 208
4.2.6.1 设置系统参数 208
4.2.6.2 FlexPendant示教器的调节监控 209
4.2.6.3 用RAPID程序调节监控 210
4.2.6.4 如何避免误触发 211
4.3 Collision Avoidance 3150-1 212
4.4 SafeMove Assistant 213

5 Motor Control 215

5.1 Independent Axis [3111-1] 215
5.1.1 概述 215
5.1.2 系统参数 217
5.1.3 RAPID组件 218
5.1.4 代码示例 219

6 RAPID Program Features 221

6.1 Path Recovery [3113-1] 221
6.1.1 概述 221
6.1.2 RAPID组件 222
6.1.3 保存当前路径 223
6.1.4 路径记录 225
6.2 Multitasking [3114-1] 229
6.2.1 Multitasking介绍 229
6.2.2 系统参数 231
6.2.3 RAPID组件 232
6.2.4 各项任务间的通信 233
6.2.4.1 永久变量 233
6.2.4.2 等候其它任务 234
6.2.4.3 多项任务之间的同步 236
6.2.4.4 使用调度程序 238
6.2.5 其它编程问题 240
6.2.5.1 在各项任务之间共享资源 240
6.2.5.2 测试任务是否控制着机械单元 241
6.2.5.3 taskid 242
6.2.5.4 避免冗长环路 243

7 Communication 245

7.1 FTP&SFTP client [3116-1] 245
7.1.1 FTP&SFTP client简介 245
7.2 NFS Client [3117-1] 248
7.2.1 NFS Client介绍 248

8 User Interaction Application 251

8.1 RobotStudio Connect [3119-1] 251
8.2 FlexPendant Base Apps 252
8.3 FlexPendant Independent Apps 253

9 Engineering tools 255

9.1 RobotWare Add-In 255
9.2 Path Corrections [3123-1] 256
9.2.1 概述 256
9.2.2 RAPID组件 257
9.2.3 相关的RAPID功能 258
9.2.4 代码示例 259

9.3 Auto Acknowledge Input 260
索引 261

此页刻意留白

手册概述

关于本手册

本手册说明了何时使用及如何使用各种RobotWare的选项与函数。

手册用法

用户即可参考本手册来判断某个选项是否正好能解决某一问题，也可按本手册的说明来使用某个选项。本手册不含RAPID例程或类似程序的详细语法信息，具体请参见它们各自的参考手册。

本手册的阅读对象

本手册供机器人程序员使用。

操作前提

读者宜满足以下要求：

- 熟悉工业机器人及其术语。
- 熟悉RAPID编程语言。
- 熟悉系统参数和这些参数的配置方法。

参考信息

参考文档	文档编号
产品规格 - *OmniCore C 系列*	*3HAC065034-010*
产品规格 - *OmniCore E 系列*	*3HAC079823-010*
操作手册 - *RobotStudio*	*3HAC032104-010*
操作手册 - *OmniCore*	*3HAC065036-010*
操作手册 - *OmniCore*集成工程师指南	*3HAC065037-010*
技术参考手册 - *RAPID*指令、函数和数据类型	*3HAC065038-010*
技术参考手册 - *RAPID Overview*	*3HAC065040-010*
技术参考手册 - 系统参数	*3HAC065041-010*

修订版

版本号	描述
A	随 RobotWare 7.0 发布。
B	随RobotWare 7.01发布。 本修订中进行了如下更新： • 将整本手册中的“Cyber security”替换为“Cybersecurity”。 • 第51页的*Connected Services* 已更新。
C	随 RobotWare 7.0.2 发布。 本修订中进行了如下更新： • 整本手册更新了 FlexPendant 术语。 • 第56页的*FlexPendant 中 Connected Services* 路径汇总已更新。

下一页继续

续前页

版本号	描述
D	随 RobotWare 7.1 发布。 本修订中进行了如下更新： • 第*51*页的*Connected Services* 已更新。 • 增加了关于 YuMi 机器人和碰撞检测的信息，请参阅第*199*页的*YuMi* 机器人碰撞检测。 • 更新了 SFTP 客户端的限制，请参阅第*246*页的限制。
E	随 RobotWare 7.2 发布。 本修订中进行了如下更新： • 第*51*页的*Connected Services* 已更新。 • 章节第*207*页的*Signals*更新了关于数字输出*MotSupOn*的信息。 • 章节第*139*页的系统参数更新了有关如何调整属性值RMQ Max Message Size和RMQ Max No Of Messages的信息。 • 因远程挂载磁盘/虚拟根变更更新相关章节：第*246*页的限制（FTP 和 SFTP 客户端）和 第*248*页的限制（NFS 客户端）。
F	随 RobotWare 7.3 发布。 本修订中进行了如下更新： • 第*51*页的*Connected Services* 已更新。
G	随 RobotWare 7.4 发布。 本修订中进行了如下更新： • 新增了第*89*页的嵌入式互联服务故障排除日志 一节。 • 第*51*页的*Connected Services* 已更新。 • 对于 第*117*页的原始数据通信 中的 UTF-8 方面的信息进行更新。
H	随 RobotWare 7.5 发布。 • 更新 第*212*页的*Collision Avoidance 3150-1* 限制。
J	随 RobotWare 7.6 发布。 • 新增了第*260*页的*Auto Acknowledge Input* 一节。 • 更新了有关直通的限制，请参阅 第*193*页的概述 *World Zones*。 • 新增了第*213*页的*SafeMove Assistant* 一节。

RobotWare 中的开源和第三方组件

RobotWare 中的开源和第三方组件

ABB 产品使用第三方提供的软件，包括开源软件。以下版权声明和许可适用于 ABB 软件内分布的各种组件。每一 ABB 产品不一定使用所有列出的第三方软件组件。持证人必须完全同意并遵守这些许可条款，否则用户无权使用本产品。开始使用 ABB 软件意味着接受另外提到的许可条款。第三方许可条款仅适用于许可所属的相应软件，且第三方许可条款不适用于 ABB 产品。针对根据 GNU 通用公共许可提供的程序，GNU 通用公共许可出租许可方将根据要求向持证人提供相应源代码的机器可读副本。此服务自产品交付起有效期为三年。

ABB 软件根据 ABB 最终用户许可协议获得许可（单独提供）。

对于 RobotWare，许可信息位于 RobotWare 发布包中的文件夹**\licenses**中。

对于 OleOS，传送带跟踪模块(CTM)使用基于 Linux 的操作系统，版权声明和许可证列表可参见 CTM 板上的文件**/etc/licenses.txt**，可通过控制台端口予以访问或通过 SFTP 下载该文件。

对于 CTM 应用程序，版权声明和许可证列表可参见 CTM 板上的文件**/opt/ABB.com/ctm/licenses.txt**，可通过控制台端口予以访问或通过 SFTP 下载该文件。

此页刻意留白

1 RobotWare介绍

软件产品

RobotWare 是 ABB Robotics 的系列软件产品。此产品旨在提高生产效率以及降低机器人的拥有成本和运行成本。ABB Robotics 多年专注于这些产品的开发，它们代表了从数以千计的机器人安装中汲取的知识和经验。

产品类别

在 RobotWare 系列中，有不同的产品类别：

产品类别	描述
RobotWare-OS	这是机器人的操作系统。RobotWare-OS 为基础机器人编程和运行提供了所有必要的功能。这是机器人的固有部分，但也可以单独提供来进行升级。 有关 RobotWare-OS 的说明，请参见机器人控制器的产品规格。
RobotWare 选件	这些产品是在 RobotWare-OS 上运行的选件。它们是为需要动作控制、通信、系统工程或应用等附加功能的机器人用户准备的。 **注意** 本手册并未介绍完所有的RobotWare选项，一些更全面的选项请参见各份单独手册。
生产应用选件	这些是点焊、弧焊和分配等的特定生产应用的扩展包。它们主要是为了提升生产成果和简化应用的安装与编程而设计的。 各份单独手册介绍了相关工艺应用的所有选项。
RobotWare Add-ins	RobotWare Add-in 是自包含包，可扩展机器人系统的功能。 ABB Robotics 的部分软件产品是以Add-ins 的形式发布的，比如导轨运动 IRBT、定位器 IRBP 和独立控制器等。更多信息请参见机器人控制器的产品规格。 RobotWare Add-ins 的目的还包括让 ABB 外部的程序开发者能为 ABB 机器人系统创建选件，并将选件销售给他们的客户。有关创建 RobotWare Add-ins的更多详情请联系您当地的 ABB Robotics 代表，您可以在 *www.abb.com/contacts* 找到相关信息。

选项组

就OmniCore而言，RobotWare的各选项已根据客户的利益编成若干组，以便更好地了解客户的选项值。不过所有选项都要单独购买。这些编组如下：

选项组	描述
Motion performance	该组选项可优化您机器人的性能。
Motion coordination	该组选项可让您的机器人与外接设备或其它机器人相互协调。
Motion Events	该组选项可监管机器人的位置。
Motion functions	该组选项可控制机器人的路径。
Motion Supervision	该组选项可监管机器人的移动。
Communication	该组选项可让机器人与其它设备相互通信（外接PC等）。
Engineering tools	该组选项供高级机器人集成人员使用。
Servo motor control	该组选项可通过机器人控制器来运行独立于机器人的外部电机。

下一页继续

续前页

注意

本手册并未介绍完所有的RobotWare选项，一些更全面的选项请参见各份单独手册。

2 RobotWare-OS

2.1 Advanced RAPID

2.1.1 Advanced RAPID介绍

Advanced RAPID介绍

RobotWare的基本功能*Advanced RAPID*可供机器人程序员开发那些需要高级功能的应用程序。

Advanced RAPID包括了许多不同类型的功能，这些功能可分成下列各组：

功能组	描述
位功能	在一个字节上逐位运算。
数据搜索功能	搜索并获取 / 设置数据对象（如变量等）。
别名输入 / 输出（I/O）功能	为一个I / O信号指定一个可选的别名。
配置功能	获取 / 设置系统参数。
断电功能	在断电后恢复信号。
工艺配套功能	可用于创建工艺应用。
中断功能	包括了RobotWare基本功能中没有的其它中断功能。
用户消息功能	错误消息及其它文本。
RAPID配套功能	为程序员提供的其它支持。

2.1.2 位功能

2.1.2.1 概述

目的

位功能的目的是能够对一个字节进行操作，该字节被视为 8 个数字位。您可以获取或设置单个位，或对字节进行逻辑操作。例如，在处理一组数字 I/O 信号时，这些操作大有帮助。

其中包括

位功能包括：

- 数据类型byte。
- 用于设置一个位值的指令：BitSet和BitClear。
- 用于获取一个位值的函数：BitCheck。
- 用于在一个字节上进行逻辑运算的函数：BitAnd、BitOr、BitXOr、BitNeg、BitLSh和BitRSh。

2.1.2.2 RAPID组件

数据类型

此处简述了位功能所用的每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各种数据类型。

数据类型	描述
byte	数据类型byte代表了0到255之间的一个十进制值。

指令：

此处简述了位功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
BitSet	BitSet用于将已定义字节数据中的某个指定位设置成1。
BitClear	BitClear用于清除已定义字节数据中的某个指定位（即设置成0）。

函数

此处简述了位功能所用的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
BitAnd	BitAnd用于在数据类型字节上执行一次逻辑逐位AND运算。
BitOr	BitOr用于在数据类型字节上执行一次逻辑逐位OR运算。
BitXOr	BitXOr（位异或）用于在数据类型字节上执行一次逻辑逐位XOR运算。
BitNeg	BitNeg用于在数据类型字节上执行一次逻辑逐位非运算（一的补数）。
BitLSh	BitLSh（位左移）用于在数据类型字节上执行一次逻辑逐位左移位运算。
BitRSh	BitRSh（位右移）用于在数据类型字节上执行一次逻辑逐位右移位运算。
BitCheck	BitCheck用于检查已定义字节数据中的某个指定位是否被设置成1。

提示

字节与字符串之间的换算函数——StrToByte和ByteToStr——并非相关选项的一部分，不过人们经常将它们与位功能一同使用。

2.1.2.3 位功能的示例

程序代码

```
CONST num parity_bit := 8;

!Set data1 to 00100110
VAR byte data1 := 38;

!Set data2 to 00100010
VAR byte data2 := 34;

VAR byte data3;

!Set data3 to 00100010
data3 := BitAnd(data1, data2);

!Set data3 to 00100110
data3 := BitOr(data1, data2);

!Set data3 to 00000100
data3 := BitXOr(data1, data2);

!Set data3 to 11011001
data3 := BitNeg(data1);

!Set data3 to 10011000
data3 := BitLSh(data1, 2);

!Set data3 to 00010011
data3 := BitRSh(data1, 1);

!Set data1 to 10100110
BitSet data1, parity_bit;

!Set data1 to 00100110
BitClear data1, parity_bit;

!If parity_bit is 0, set it to 1
IF BitCheck(data1, parity_bit) = FALSE THEN
  BitSet data1, parity_bit;
ENDIF
```

2.1.3 数据搜索功能

2.1.3.1 概述

目的

数据搜索功能的用途是搜索特定类型的数据对象，并获取 / 设置这些对象的数值。

以下是数据搜索功能的一些应用示例：

- 在某变量的名称仅存在于字符串中的情况下，将某个数值设为该变量。
- 列出特定类型的所有变量。
- 为具有类似名称的一组类似变量设置一个新值。

其中包括

数据搜索功能包括：

- 数据类型datapos。
- 用于搜索一组数据对象，并获取或设置它们的数值：SetDataSearch、GetDataVal、SetDataVal和SetAllDataVal。
- 用于遍历搜索结果的一项函数：GetNextSym。

2.1.3.2 RAPID组件

数据类型

此处简述了数据搜索功能所用的每种数据类型。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各种数据类型。

数据类型	描述
datapos	datapos是用函数GetNextSym检索出的某数据对象（内部系统数据）的封闭块。

指令：

此处简述了数据搜索功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
SetDataSearch	SetDataSearch与GetNextSym一起用于从相关系统中检索出数据对象。
GetDataVal	有了GetDataVal，用户便可从某字符串变量指定的数据对象中获取一个数值，或从GetNextSym检索出的数据对象中获取一个数值。
SetDataVal	有了SetDataVal，用户便可在某字符串变量指定的数据对象中设置一个数值，或在GetNextSym检索出的数据对象中设置一个数值。
SetAllDataVal	SetAllDataVal可为了其类型符合指定语法的所有数据对象设置一个新值。

函数

此处简述了数据搜索功能所用的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
GetNextSym	GetNextSym（获取下一个符号）与SetDataSearch一起用于从相关系统中检索出数据对象。

2.1.3.3 数据搜索功能的示例

设置未知变量

此例展示了当某变量名称未知且仅出现于字符串中时，该如何在编程时设置该变量的数值。

```
VAR string my_string;
VAR num my_number;
VAR num new_value:=10;
my_string := "my_number";
!Set value to 10 for variable specified by my_string
SetDataVal my_string,new_value;
```

重置变量范围

在此例中，所有以“my”打头的数字变量都被重置为0。

```
VAR string my_string:="my.*";
VAR num zerovar:=0;
SetAllDataVal "num"\Object:=my_string,zerovar;
```

列出 / 设置特定变量

在此例中，模块“mymod”中所有以“my”打头的数字变量都被列在了FlexPendant示教器上，然后被重置为0。

```
VAR datapos block;
VAR string name;
VAR num valuevar;
VAR num zerovar:=0;

!Search for all num variables starting with "my" in the module
     "mymod"
SetDataSearch "num"\Object:="my.*"\InMod:="mymod";

!Loop through the search result
WHILE GetNextSym(name,block) DO
  !Read the value from each found variable
  GetDataVal name\Block:=block,valuevar;

  !Write name and value for each found variable
  TPWrite name+" = "\Num:=valuevar;

  !Set the value to 0 for each found variables
  SetDataVal name\Block:=block,zerovar;
ENDWHILE
```

2.1.4 别名 I/O 信号

2.1.4.1 概述

目的

别名I / O功能使程序员能为一个信号指定任何名称，并将该名称与一个配置好I / O信号联系起来。

在不同系统中重复使用RAPID程序时这一点会发挥作用：用户不用重写代码，而是利用新系统上既有的信号名称来把该程序所用的信号名称定义为一个别名。

其中包括

别名I / O功能包含了指令`AliasIO`。

2.1.4.2 RAPID组件

数据类型

没有针对别名I / O功能的RAPID数据类型。

指令：

此处简述了别名I / O功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
AliasIO	AliasIO的作用是用一个别名来定义任何类型的信号，或使用内置任务模块中的信号。其别名与一个配置好的I / O信号有关。 使用任何实际信号前都必须先运行指令AliasIO。

函数

没有针对别名I / O功能的RAPID函数。

2.1.4.3 别名I / O功能的示例

指定信号的别名

该示例展示了如何定义与“配置好的数字输出I / O信号config_do”有关的数字输出信号alias_do。

路径prog_start与“启动”事件有关。

这将确保在即使没有用上述名称配置信号的情况下，也能在RAPID代码中使用“alias_do”。

```
VAR signaldo alias_do;
PROC prog_start()
  AliasIO config_do, alias_do;
ENDPROC
```

2.1.5 配置功能

2.1.5.1 概述

目的

配置功能可让程序员在运行时访问各个系统参数，并对其进行读取和编辑。可重启相关控制器来让新的参数值生效。

其中包括

配置功能中包括了下列指令：ReadCfgData、WriteCfgData和WarmStart。

2.1.5.2 RAPID组件

数据类型

没有针对配置功能的RAPID数据类型。

指令：

此处简述了配置功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
ReadCfgData	ReadCfgData的作用是读取一个已命名系统参数（配置数据）的一项属性。
WriteCfgData	WriteCfgData的作用是写入一个已命名系统参数（配置数据）的一项属性。
WarmStart	WarmStart的作用是在运行时重启控制器。 这能在用指令WriteCfgData更改系统参数后发挥作用。

函数

没有针对配置功能的RAPID函数。

2.1.5.3 配置功能的示例

配置系统参数

在此处的示例中，用户读取了rob1_1的系统参数*cal_offset*，将其增加了0.2毫米，然后又重新写入。重启相关控制器后此更改才会生效。

```
VAR num old_offset;
VAR num new_offset;

ReadCfgData "/MOC/MOTOR_CALIB/rob1_1", "cal_offset",old_offset;
new_offset := old_offset + (0.2/1000);
WriteCfgData "/MOC/MOTOR_CALIB/rob1_1", "cal_offset",new_offset;
WarmStart;
```

2.1.6 断电功能

2.1.6.1 概述

目的

如果上电失败时机器人正在路径上移动，那么机器人恢复运动时就可能需要一些额外的行动。断电功能可帮您检测路径移动期间是否发生了上电失败。

> **注意**
>
> 更多信息请参见技术参考手册 - 系统参数中主题*I/O System*下的类型*Signal Safe Level*。

其中包括

断电功能中包括了一个检查被中断路径的函数：`PFRestart`

2.1.6.2 RAPID组件与系统参数

数据类型

没有针对断电功能的RAPID数据类型。

指令：

没有针对断电功能的RAPID指令。

函数

此处简述了断电功能所用的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
PFRestart	如果路径在断电时被中断，那么则用PFRestart（断电重启）进行检查，并可能需要采取一些特定行动。该函数会在电流等级、基本等级或中断等级上检查涉事路径。

系统参数

断电功能中没有系统参数，不过不论您是否安装了任一选项，您都能使用参数*Store signal at power fail*。

有关更多信息，请参阅 技术参考手册 - 系统参数。

2.1.6.3 断电功能的示例

对被中断路径的测试

在断电后又恢复作业的情况下，如果路径上发生过断电，那么就会像本示例一样进行测试。

```
!Test if path was interrupted
IF PFRestart() = TRUE THEN
  SetDO do5,1;
ELSE
  SetDO do5,0;
ENDIF
```

2.1.7 工艺配套功能

2.1.7.1 概述

目的

可使用工艺配套功能提供的一些RAPID指令来创建工艺应用。其用例包括：

- 可将连续工艺应用中所用的模拟输出信号设置成与机器人工具中心接触点（TCP）的速度成正比。
- 被“程序停止”或“紧急停止”所停止的连续工艺应用可从其停止处继续运行。

其中包括

工艺配套功能包括：

- 数据类型`restartdata`。
- 用于设置输出信号的指令：`TriggSpeed`。
- 与重启有关的所用指令：`TriggStopProc`和`StepBwdPath`。

限制

当您拥有基本功能*Fixed Position Events*时，才能使用指令`TriggSpeed`。

2.1.7.2 RAPID组件

数据类型

此处简述了工艺配套功能所用的每种数据类型。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各种数据类型。

数据类型	描述
restartdata	restartdata可包含机器人移动的停止序列处的指定I / O信号（工艺信号）的前值和后值。 当自主开发的工艺指令发生程序停止或紧急停止后，restartdata与TriggStopProc会被一同用于保存重启所需的数据。

指令：

此处简述了工艺配套功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
TriggSpeed	TriggSpeed的作用是定义一个模拟输出信号（该信号与TCP速度成某个数值比）的设置状况。 TriggSpeed只能搭配选项Fixed Position Events使用。
TriggStopProc	TriggStopProc的作用是保存所有已用工艺信号的前值和后值。 当自主开发的工艺指令发生程序停止或紧急停止后，TriggStopProc与数据类型restartdata会被一同用于保存重启所需的数据。
StepBwdPath	StepBwdPath的作用是从一则“重启”事件例程开始沿机器人路径向后移动TCP。

函数

没有针对工艺配套功能的RAPID函数。

2.1.7.3 工艺配套功能的示例

与速度成正比的信号

本示例将控制胶量的模拟输出信号设置成与速度成正比。

为了对机器人的任何速度骤减作出时间上的补偿，相应的模拟输出信号glue_ao会在TCP速度骤减前受到0.04秒的影响。如果glue_ao中计算出的逻辑模拟输出值溢出，那么便设置数字输出信号glue_err。

```
VAR triggdata glueflow;

!The glue flow is set to scale value 0.8 0.05 s before point p1
TriggSpeed glueflow, 0, 0.05, glue_ao, 0.8 \DipLag=:0.04,
     \ErrDO:=glue_err;
TriggL p1, v500, glueflow, z50, gun1;

!The glue flow is set to scale value 1 10 mm plus 0.05 s
! before point p2
TriggSpeed glueflow, 10, 0.05, glue_ao, 1;
TriggL p2, v500, glueflow, z10, gun1;

!The glue flow ends (scale value 0) 0.05 s before point p3
TriggSpeed glueflow, 0, 0.05, glue_ao, 0;
TriggL p3, v500, glueflow, z50, gun1;
```

提示

注意也可用NOSTEPIN例程概念来创建关于TriggSpeed的自主开发工艺指令。

在停止后恢复信号

在本例中，一个输出信号在程序停止或紧急停止后恢复了其数值。

无返回值程序supervise被定义为一则“通电”事件例程，而resume_signals则被定义为一则“重启”事件例程。

```
PERS restartdata myproc_data :=
     [FALSE,FALSE,0,0,0,0,0,0,0,0,0,0,0,0,0];
...
PROC myproc()
  MoveJ p1, vmax, fine, my_gun;
  SetDO do_close_gun, 1;
  MoveL p2,v1000,z50,my_gun;
  MoveL p3,v1000,fine,my_gun;
  SetDO do_close_gun, 0;
ENDPROC
...
PROC supervise()
  TriggStopProc myproc_data \DO1:=do_close_gun, do_close_gun;
ENDPROC
```

下一页继续

2.1.7.3 工艺配套功能的示例
续前页

```
PROC resume_signals()
  IF myproc_data.preshadowval = 1 THEN
    SetDO do_close_gun,1;
  ELSE
    SetDO do_close_gun,0;
  ENDIF
ENDPROC
```

向后移动TCP

在本例中，TCP在1秒内沿重启前的同一路径向后移动了30毫米。

无返回值程序move_backward被定义为一则“重启”事件例程。

```
PROC move_backward()
  StepBwdPath 30, 1;
ENDPROC
```

2.1.8 中断功能

2.1.8.1 概述

目的

除RAPID中始终包括的中断功件外，Advanced RAPID的中断功能还有一些额外的功件。基本中断功能方面的更多信息请参见技术参考手册 - *RAPID Overview*。

中断应用（Advanced RAPID）中的一些示例可在以下方面提供帮助：

- 在某永久变量改变数值时生成一次中断。
- 在发生一次错误时生成一次中断，然后查找该错误方面的更多信息。

其中包括

Advanced RAPID的中断功能包括：

- 错误中断的数据类型：`trapdata`、`errdomain`、`errtype`。
- 用于生成中断的指令：`IPers`和`IError`。
- 用于在错误中断方面查找更多信息的指令：`GetTrapData`和`ReadErrData`。

2.1.8.2 RAPID组件

数据类型

此处简述了中断功能所用的每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各种数据类型。

数据类型	描述
trapdata	trapdata代表了与“导致了当前待执行软中断例程的那次中断”有关的内部信息。
errdomain	errdomain的作用是指定一个错误域。系统会根据错误性质的差异而记录在不同的域内。
errtype	errtype的作用是指定一种错误类型（错误、警告、状态变化）。

指令：

此处简述了中断功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
IPers	IPers（永久中断）的作用是下令每当某永久变量发生改变时就生成一次中断。
IError	IError（中断错误）的作用是下令每当发生一次错误时就生成一次中断。
GetTrapData	GetTrapData的作用是用指令IError生成一则软中断例程。GetTrapData会获得“导致了待执行软中断例程的那次中断”方面的所有信息。
ReadErrData	ReadErrData的作用是用指令IError生成软中断例程。ReadErrData会读取GetTrapData获得的相关信息。
ErrRaise	ErrRaise的作用是在相关程序中创建一个错误，然后为相关例程的错误处理器创建一次相应的调用。也可在错误处理器中用ErrRaise来把当前错误传递给调用例程的错误处理器。

函数

没有针对中断功能的RAPID函数。

2.1.8.3 中断功能的示例

当永久变量发生变化时中断

在本例中，系统在永久变量counter的数值发生变化时调用了一则软中断例程。

```
VAR intnum int1;
PERS num counter := 0;

PROC main()
  CONNECT int1 WITH iroutine1;
  IPers counter, int1;
  ...
  counter := counter + 1;
  ...
  Idelete int1;
ENDPROC

TRAP iroutine1
  TPWrite "Current value of counter = " \Num:=counter;
ENDTRAP
```

错误中断

在本例中，系统在发生一次错误时调用了一则软中断例程。该软中断例程决定了相应的错误域和错误编号，并通过输出信号向外传输了这些内容。

```
VAR intnum err_interrupt;
VAR trapdata err_data;
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;

PROC main()
  CONNECT err_interrupt WITH trap_err;
  IError COMMON_ERR, TYPE_ERR, err_interrupt;
  ...
  a:=3;
  b:=0;
  c:=a/b;
  ...
  IDelete err_interrupt;
ENDPROC

TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number, err_type;
  SetGO go_err1, err_domain;
  SetGO go_err2, err_number;
ENDTRAP
```

2.1.9 用户消息功能

2.1.9.1 概述

目的

用户消息功能的作用一是设置事件编号，二是加快处理事件消息以及相关用户界面呈现的其它文本。

此处是一些应用示例：

- 从简化了更新和转换内容的文本表格文件中获取用户消息。
- 在“提升”指令中添加任何将被作为错误恢复常量的系统错误编号以及“错误”处理器测试所需的系统错误编号。

其中包括

用户消息功能包括：

- 文本表格操作指令`TextTabInstall`。
- 文本表格操作指令：`TextTabFreeToUse`、`TextTabGet`和`TextGet`。
- 处理错误编号的指令：`BookErrNo`.

2.1.9.2 RAPID组件

数据类型

没有针对用户消息功能的RAPID数据类型。

指令：

此处简述了用户消息功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
BookErrNo	BookErrNo的作用是定一个新的RAPID系统错误编号。
TextTabInstall	TextTabInstall的作用是在系统中安装一份文本表格。

函数

此处简述了用户消息功能所用的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
TextTabFreeToUse	TextTabFreeToUse的是测试能否自由使用相关的文本表格名称（尚未安装在系统中）。
TextTabGet	TextTabGet的作用是是获取一份由用户定义的文件表格的文件表格编号。
TextGet	TextGet的作用是是从系统文件表格中获取一段文本字符串。

2.1.9.3 用户消息功能的示例

卷册错误编号

该示例展示了如何添加一个新的错误编号。

```
VAR intnum sig1int;

!Introduce a new error number in a glue system.
!Note: The new error variable must be declared with the
! initial value -1
VAR errnum ERR_GLUEFLOW := -1;

PROC main()
  !Book the new RAPID system error number
  BookErrNo ERR_GLUEFLOW;

  !Raise glue flow error if di1=1
  IF di1=1 THEN
    RAISE ERR_GLUEFLOW;
  ENDIF
ENDPROC

!Error handling
ERROR
IF ERRNO = ERR_GLUEFLOW THEN
  ErrWrite "Glue error", "There is a problem with the glue flow";
ENDIF
```

文本表格文件的错误消息

该示例展示了如何从一份文本表格文件中获取用户消息。

名为HOME:/language/en/text_file.xml的文件中有一份名为text_table_name的文本表格。该表格包含了用英语表述的错误消息。

在“通电”事件时执行无返回值程序install_text。第一次执行时会安装文本表格文件text_file.xml，而再次执行时函数TextTabFreeToUse则会返回FALSE，且不会重复安装。

之后会用该表格来获取用户界面消息。

```
VAR num text_res_no;

PROC install_text()
  !Test if text_table_name is already installed
  IF TextTabFreeToUse("text_table_name") THEN
    !Install the table from the file HOME:/language/en/text_file.xml
    TextTabInstall "HOME:/language/en/text_file.xml";
  ENDIF
  !Assign the text table number for text_table_name to text_res_no
  text_res_no := TextTabGet("text_table_name");
ENDPROC
...
!Write error message with two strings from the table text_res_no
```

下一页继续

```
ErrWrite TextGet(text_res_no, 1), TextGet(text_res_no, 2);
```

2.1.9.4 文本表格文件

概述

文本表保存在 XML 文件中（每个文件可以包含采用一种语言的一个表）。该表可以包含任意数量的文本字符串，编码必须为 UTF-8 或ISO-8859-1。

对文本表格的说明

此处描述了相关文本表格文件中的XML标签和自变数。

标签	变元	描述
Resource		代表一份文本表格。一份文件只能包含一个Resource实例。
	Name	相关文本表格的名称。通过RAPID指令TextTabGet使用。
	Language	针对文本字符串所用语言的语言代码。 用 RAPID 指令安装的文件 TextTabInstall，用于所有语言。要使用多种语言，请使用唯一的文件路径名和唯一的 Resource，为每种语言安装一个文件。
Text		代表了一段文本字符串。
	Name	表中文本字符串的编号。
Value		待使用的文本字符串。
Comment		对相关文本字符串及其用法的意见。

文本表格文件的示例

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<Resource Name="text_table_name" Language="en">
  <Text Name="1">
    <Value>This is a text that is </Value>
    <Comment>The first part of my text</Comment>
  </Text>
  <Text Name="2">
    <Value>displayed in the user interface.</Value>
    <Comment>The second part of my text</Comment>
  </Text>
</Resource>
```

2.1.10 RAPID配套功能

2.1.10.1 概述

目的

RAPID配套功能由各式各样的例程组成，这些例程或许能为高级机器人程序员提供帮助。

此处是一些应用示例：

- 激活一件新的工具、工作对象或净负荷。
- 查找在当前例程外调用了哪个自变数。
- 测试上一次程序停止期间是否移动过程序指针。

其中包括

RAPID配套功能包括：

- 用于激活指定系统数据的指令：`SetSysData`。
- 获取原始数据对象名称的函数：`ArgName`。
- 针对程序指针移动相关信息的函数：`IsStopStateEvent`。

2.1.10.2 RAPID组件

数据类型

没有针对RAPID配套功能的数据类型。

指令

此处简述了RAPID配套功能所用的每条指令。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各条指令。

指令	描述
SetSysData	SetSysData激活（或更改当前激活的）机器人工具、机器人工作对象或机器人净负荷。

函数

此处简述了RAPID配套功能所用的每则函数。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各则函数。

功能	描述
ArgName	ArgName的作用是是获取当前自变数或当前数据所需的原始数据对象的名称。
IsStopStateEvent	IsStopStateEvent会返回程序指针的移动信息。

2.1.10.3 RAPID配套功能的示例

激活工具

此例展示了如何激活一件已知工具：

```
!Activate tool1
SetSysData tool1;
```

此例展示了当该工具的名称仅出现于字符串中时，该如何激活一件工具。

```
VAR string tool_string := "tool2";
!Activate the tool specified in tool_string
SetSysData tool0 \ObjectName := tool_string;
```

获取自变数的名称

本例中取用了par1的原始名称。输出内容将为“Argument name my_nbr with value 5”。

```
VAR num my_nbr :=5;
proc1 my_nbr;

PROC proc1 (num par1)
  VAR string name;
  name:=ArgName(par1);
  TPWrite "Argument name "+name+" with value " \Num:=par1;
ENDPROC
```

测试是否移动过程序指针

此例测试了上一次程序停止期间是否移动过程序指针。

```
IF IsStopStateEvent (\PPMoved) = TRUE THEN
  TPWrite "The program pointer has been moved.";
ENDIF
```

2.2 Analog Signal Interrupt

2.2.1 Analog Signal Interrupt介绍

目的

Analog Signal Interrupt的用途一是监管一个模拟信号，二是在达到某指定值时生成一次中断。

与各种轮询法相比，Analog Signal Interrupt更快捷，更易于执行，对计算机能力的需求也更低。

此处是一些应用示例：

- 用更好的定时来节省周期时间（在某信号达到指定值时——而非等候轮询——就准时开始移动机器人）。
- 如果某信号值超出了其允许范围，则会显示警报或错误消息。
- 如果某信号值达到了某种危险水平，则会停止相应的机器人。

其中包括

您可通过RobotWare基本功能Analog Signal Interrupt来访问以下指令：

- `ISignalAI`
- `ISignalAO`

基本方法

这是Analog Signal Interrupt的一般用法。第*50*页的代码示例用一个更详细的示例展示了其具体用法。

1. 创建一则软中断例程
2. 用指令`CONNECT`来连接相关的软中断例程。
3. 用指令`ISignalAI`或`ISignalAO`来定义相应的中断条件。

限制

如果您有一个工业网络选项（比如 DeviceNet），那么就只能使用模拟信号。

2.2.2 RAPID组件

数据类型

Analog Signal Interrupt中不包括任何数据类型。

指令：

此处简述了Analog Signal Interrupt中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
ISignalAI	为应调用的一则中断例程定义一个模拟输入信号的相关数值。 可设置成当相关信号值高于或低于某指定值时就出现一次中断，或设置成当处于或超出某指定范围时就出现一次中断。此外还可指定该中断是仅出现一次还是反复出现。
ISignalAO	为应调用的一则中断例程定义一个模拟输出信号的相关数值。 可设置成当相关信号值高于或低于某指定值时就出现一次中断，或设置成当处于或超出某指定范围时就出现一次中断。此外还可指定该中断是仅出现一次还是反复出现。

函数

Analog Signal Interrupt中不包括RAPID函数。

2.2.3 代码示例

温度监视

本例中的一部温度传感器与信号ai1相连。

含一条警告的一则中断例程被设置成“在120度到130度内，温度每上升0.5度就执行一次”。另一则停止相关机器人的软中断例程则被设置成“在温度升至130度以上后就尽快执行”。

```
VAR intnum ai1_warning;
VAR intnum ai1_exeeded;

PROC main()
  CONNECT ai1_warning WITH temp_warning;
  CONNECT ai1_exeeded WITH temp_exeeded;
  ISignalAI ai1, AIO_BETWEEN, 130, 120, 0.5, \DPos, ai1_warning;
  ISignalAI \Single, ai1, AIO_ABOVE_HIGH, 130, 120, 0, ai1_exeeded;
  ...
  IDelete ai1_warning;
  IDelete ai1_exeeded;
ENDPROC

TRAP temp_warning
  TPWrite "Warning: Temperature is "\Num:=ai1;
ENDTRAP

TRAP temp_exeeded
  TPWrite "Temperature is too high";
  Stop;
ENDTRAP
```

2.3 Connected Services

2.3.1 概述

描述

Connected Services 是一种使 ABB 机器人通过使用 3G、WiFi 或有线连接而连接到 ABB Ability™ Connected Services Cloud 的功能。Connected Services从控制器收集服务信息。

目的

Connected Services 的主要用途是从控制器收集服务信息。

这些服务信息将通过 MyRobot 用于 Connected Services 1.0、通过 Connected Services 门户用于 Connected Services 2.0 或本地推送。

其中包括

RobotWare 基本功能 Connected Services 使您能够访问：

- Connected Services 代理程序软件，以管理连通性和服务数据收集。
- 用于启用和配置连通性的系统参数。
- 状态和信息页面。
- Connected Services 关键事件的专用事件日志。
- 通过 Connected Services Gateway 的连通性。
- 通过公共端口的连通性。

操作前提

Connected Services 功能要求控制器包含在与 ABB 签署的服务协议中。请联系您当地的 ABB 办事处，以建立与 Connected Services 的服务协议，并在建立连接之后访问 MyRobot 网站以执行注册。

> **注意**
>
> MyRobot 是 ABB 旗下网站，它提供了根据服务协议可以访问的机器人控制器服务信息。

基本流程

Connected Services 可以作为 RobotWare 中的即插即连解决方案在本机上使用。这种设置理念旨在：

1. 为控制器提供互联网连通性。
2. 配置连接服务并启动连接。
3. 通过MyRobot注册页面注册控制器。

一旦接入并注册“Connected Services”，数据收集将以透明方式在后台运行。

限制

控制器识别通过控制器序列号完成，且必须与服务协议中确定的序列号匹配。

下一页继续

2.3.1 概述
续前页

开机连接

使用 Connected Services Gateway 3G 将在控制器开机后提供自动连通，而无需任何配置。

生产注册

ABB 将在生产过程中安全地预先注册 Connected Services，以避免手动注册。

必要时，用户仍可以进行手动注册。

2.3.2 Connected Services 连通性

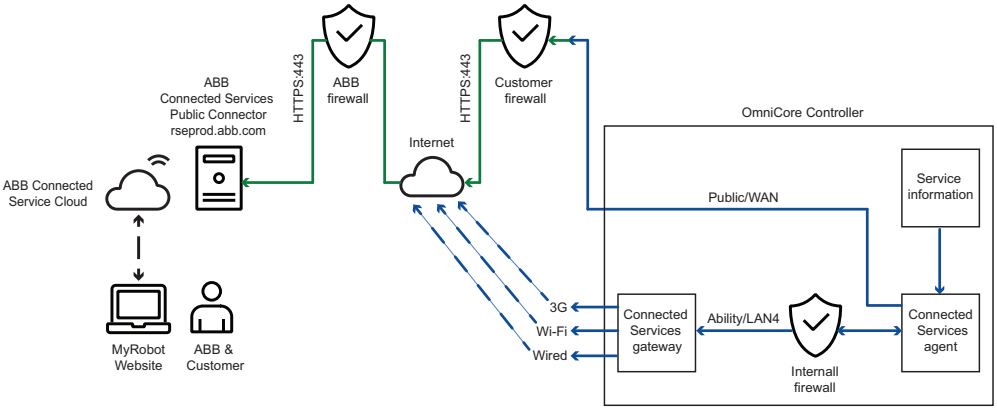
Connected Services 连接理念

Connected Services 的理念即，在控制器内部实现虚拟软件代理，且其通过互联网与 ABB Ability™ Connected Services Cloud 安全地通信。

> **注意**
>
> 控制器通过公共网络的连通性需要由客户提供的防火墙。

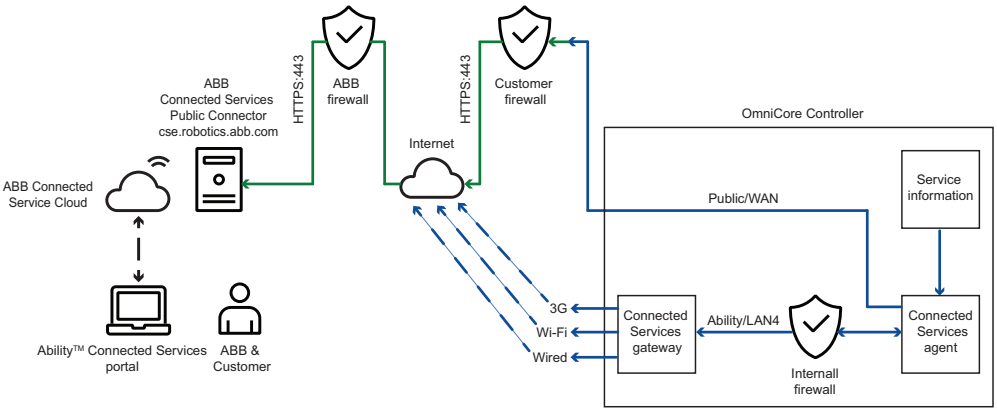
通信使用 HTTPS（安全 HTTP）加密，确保安全。只能从控制器到 ABB Ability™ Cloud 进行通信，隔离客户网络与外部互联网访问。以下各图介绍这些方案。

Connected Services 1.0：



xx1900000977

Connected Services 2.0：



xx2100000309

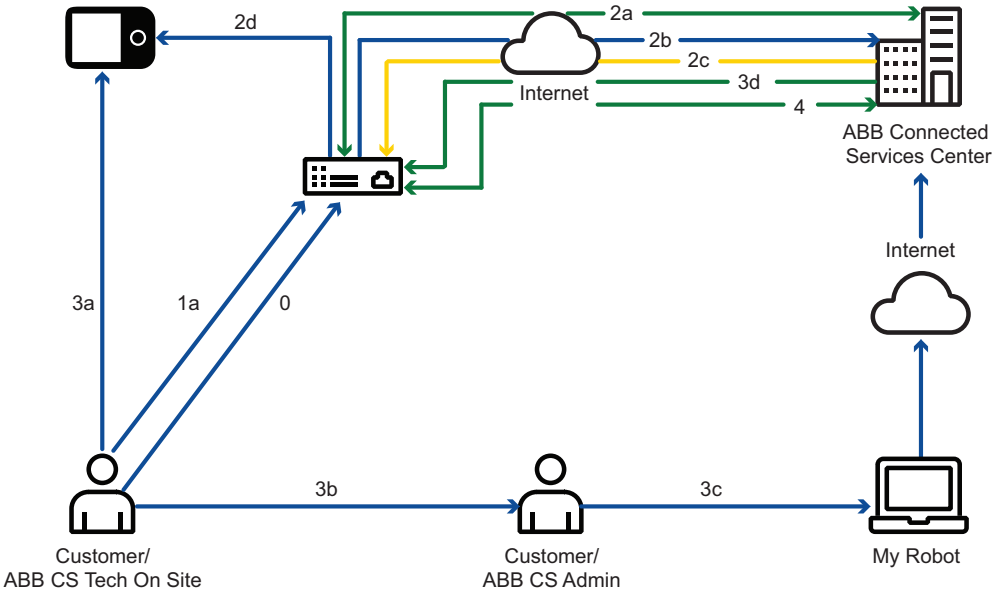
2.3.3 Connected Services 注册

Connected Services 启动

Connected Services 启动以下列步骤为基础：

- (0) Connected Services 准备
- (1) Connected Services 配置
- (2) Connected Services 连通
- (3) Connected Services 注册
- (4) Connected Services 完成连接和注册

完成上述步骤后，软件代理将安全地连接，并通过客户端证书来识别。下图描述了这些理念：



xx1500003226

步骤	描述
0	检查控制器 S/N 和网络连接
1a	启用 CSE 并设置连接配置
2a	CS 连接到位
2b	低轮询注册
2c	注册不信任（获取注册代码）
2d	显示注册代码
3a	获取注册代码
3b	提供控制器的 S/N 和注册码
3c	在 SA 中选择控制器 S/N 并使用注册代码注册
3d	注册可信任（客户端证书）
4	连接和注册安全的 CS 会话

下一页继续

Connected Services 准备工作

1 验证此控制器的服务协议是否可用于 ABB。
2 使用控制器机柜中找到的序列号验证控制器序列号。
3 验证并为机器人控制器提供互联网连接。

Connected Services 配置

1 根据连接类型配置连通性参数，请参阅 第61页的使用 *FlexPendant* 配置 *Connected Services*。

Connected Services 连通性

1 软件代理连接到 ABB Ability™ Connected Services Cloud。
2 初始注册过程基于所选的轮询速率开始。
3 初始注册不完整，尚未完全信任。
4 已收到注册码来完成信任关系。
5 注册页面上提供了 Connected Services 注册代码。

Connected Services 注册

1 客户/ABB 在现场向 Connected Services 管理员提供控制器序列号和注册代码以进行注册。
2 Connected Services 管理员在 Connected Service 1.0 的我的机器人/注册和 Connected Service 2.0 的我的 *ABB*/注册我的机器人控制器的服务协议中验证该注册代码。
3 注册信任开始，在控制器部署一个客户证书。

Connected Services 完成连接和注册

1 控制器已按照服务协议中所述经过连接、注册和识别。
2 凭客户端证书信任相关连接。

> **注意**
>
> 通过 Connected Service 2.0，还安装了第二张证书来信任 ABB Ability 云。

3 连接服务现在正在机器人控制器上运行中。

> **注意**
>
> 如果使用 3G Connected Services Gateway 和生产注册进行开机连接，则在控制器开机后，所有这些过程都将自动完成。

2.3.4 FlexPendant 中 Connected Services 路径汇总

配置

在 Flexpendant 中，Connected Services 配置可用于以下各项：

- CS Gateway 3G：**Settings（设置）** > **ABB Ability™** > **3G Connection（3G 连接)**
- CS Gateway WiFi：**Settings（设置）** > **ABB Ability™** > **WiFi Connection（WiFi 连接)**
- CS Gateway Wired：**Settings（设置）** > **ABB Ability™** > **Wired Connection（有线连接)**

状态

在 FlexPendant 中，Connected Services 状态可用于以下各项：

- Ability 网络：**Settings（设置）** > **ABB Ability™** > **Network Status（网络状态)**
- CS 网关：**Settings（设置）** > **ABB Ability™** > **Connectivity Status（连通性状态)**
- Connected Services 概要：**QuickSet** > **ABB Ability™**
- Connected Services 详细信息：**Settings** > **ABB Ability™** > **Connected Services Status（连接服务状态)**
- 重置 Connected Services：**Operate（操作）** > **Service Routine（保养例行程序)**

日志

在 Flexpendant 中，Connected Services 日志可用于以下各项：

- 连接日志 (3G/WiFi)：**Settings（设置）** > **Backup and Recovery（备份和恢复）** > **Connection Logs（连接日志)**
- 详细信息日志：**Settings（设置）** > **Backup and Recovery（备份和恢复）** > **System Diagnostic（系统诊断)**

2.3.5 RobotStudio 中 Connected Services 路径汇总

配置

在 RobotStudio 中，Connected Services 配置可用于以下各项：

- CS Gateway 3G：**Controller（控制器）** > **Configuration（配置）** > **Communication（通信）** > **CS Gateway 3G**
- CS Gateway WiFi：**Controller（控制器）** > **Configuration（配置）** > **Communication（通信）** > **CS Gateway WiFi**
- CS Gateway Wired：**Controller（控制器）** > **Configuration（配置）** > **Communication（通信）** > **CS Gateway Wired**

状态

在 RobotStudio 中，Connected Services 状态可用于以下各项：

- Connected Services汇总/详细信息：**Controller（控制器）** > **Properties（属性）** > **Device Browser（设备浏览器）** > **Software Resources（软件资源）** > **Connected Services（连接服务）**
- 重置 Connected Services：未实现

日志

在 RobotStudio 中，Connected Services 日志可用于以下各项：

- 连接日志 (3G/WiFi)：未实现
- 详细信息日志：**Controller（控制器）** > **Properties（属性）** > **Save Diagnostic（保存诊断）**

2.3.6 配置 - 系统参数

简介

本节简要介绍了用于 Connected Services 的系统参数。有关更多信息，请参阅 技术参考手册 - 系统参数。

Connected Services 连接

以下参数属于 *Communication* 主题和 *Connected Services* 类型。有关更多信息，请参阅 技术参考手册 - 系统参数 中的相应参数。

参数	描述
Enabled	启用或禁用控制器与服务器之间的 Connected Services 连接。
Connection Type	指示通信在 Ability™（ABB Connected Services Gateway 解决方案）、公共或自定义网络上完成。 **注意** 如果连接类型配置为**公共**或**自定义**，则在防火墙设置中启用 Connected Services。更多详细信息，请参阅操作手册 - *OmniCore*集成工程师指南中**防火墙设置**章节。
Internet Gateway IP	定义连接类型为私有时互联网络的互联网网关 IP，该设置对 Connection Type Custom 有效。 强制使用互联网网关连接。
Internet DNS IP	定义互联网络的互联网 DNS IP，该设置对 Connection Type Custom 有效。 强制使用 DNS 连接。
Proxy Used	定义是否使用代理来访问互联网及其名称/地址、端口和身份认证。
Proxy Auth	定义代理认证类型。Basic 将使用 HTTP 基本认证，包括用户和密码。如选择“无”，将不使用任何认证。
Server Polling	定义特定的与服务器同步的轮询频率。可用值为 `slow` 和 `fast`。 有关服务器轮询事件行为的详细信息，请参见第*83*页的服务器轮询事件的行为描述。
Debug Mode	启用广泛的日志记录以便调试问题。
Trace Level	如果启用Debug Mode，则定义日志记录的级别。
Connected Services Mode	定义不同机器人控制器的数据格式、云解决方案和特定功能的兼容性。以下是可用的模式： • 1.0 IRC5 Compatibility • 2.0 Omnicore [Preview] **注意** 默认情况下，1.0 IRC5 Compatibility 模式处于启用状态。 **注意** 2.0 Omnicore [Preview]只能通过RobotStudio进行配置。由于Connected Services 2.0仍处于预览模式，因此在使用前，请与当地ABB客户服务部门同步了解使用情况、状态和性能。

下一页继续

参数	描述
Customer Storage	定义是否需要将数据推送到外部网络磁盘（无，磁盘）以及 MQTT。磁盘路径可以单独定义。 该参数仅对 CSE 2.0 有效。
Disk Path	如果选择 “磁盘”，则定义外部磁盘的路径。 该参数仅对 CSE 2.0 有效。
Scheme	定义用于 MQTT 通信的方案或协议类型。 该参数仅对 CSE 2.0 有效。
Web Socket	定义方案是否使用 Web 套接字连接。 该参数仅对 CSE 2.0 有效。
Host	定义 MQTT 主机的IP。 该参数仅对 CSE 2.0 有效。
Port	定义 MQTT 的端口号。 该参数仅对 CSE 2.0 有效。
URL	定义要发布数据的 MQTT 主题。 该参数仅对 CSE 2.0 有效。
Auth Type	定义用于 MQTT 连接所使用的身份认证。认证类型为使用用户/密码或客户端证书的基本认证。 该参数仅对 CSE 2.0 有效。
Certificate Path	定义客户端证书路径的位置。 该参数仅对 CSE 2.0 有效。
Key Path	定义客户端私钥文件路径的位置。 该参数仅对 CSE 2.0 有效。
Server Check	定义服务器证书是否需要对 CA 进行验证。 该参数仅对 CSE 2.0 有效。
CA Path	定义证书颁发机构或根证书的路径。 该参数仅对 CSE 2.0 有效。
Force Zip	定义数据是否需要压缩以便发布。 该参数仅对 CSE 2.0 有效。（尚未实施。）
Time Out	定义 MQTT 连接超时。 该参数仅对 CSE 2.0 有效。（尚未实施。）
Keep Alive	定义 MQTT 保持活动状态的时间。 该参数仅对 CSE 2.0 有效。（尚未实施。）
Reconnect Period	定义 MQTT 重新连接间隔。 该参数仅对 CSE 2.0 有效。（尚未实施。）
Retry	定义 MQTT 连接重试计数。 该参数仅对 CSE 2.0 有效。（尚未实施。）

下一页继续

WAN 配置（公共连接）

WAN IP/掩码/网关配置在 RobotStudio 中或 FlexPendant 上完成。WAN 以太网端口配置使用户不必使用 Connected Services Gateway 即可在控制器上访问互联网。该端口由其 IP、掩码和可能的网关所定义。有关公共 (WAN) 配置的详细信息，请参阅操作手册 - *OmniCore*集成工程师指南 中的“使用 FlexPendant 配置 Connected Service Gateway”一节。

注意

如果公共/WAN 端口已连接到互联网，则客户必须安装防火墙。

DNS 配置（公共连接）

这些参数属于*Communication* 主题和 *DNS Client* 类型。如果将公共/WAN 端口用于互联网连通，则需要定义 DNS 服务器，以将 ABB Connected Services 连接器 (rseprod.abb.com) 的名称解析为其 IP 地址。有关更多详细信息，请参阅 技术参考手册 - 系统参数 中的 *Type DNS Client*。

注意

若要执行快速测试，请使用定义为 8.8.8.8 (Google DNS) 的 DNS，然后切换为向客户推荐的 DNS 服务器 IP。

IP路由配置

这些参数属于*Communication*主题和*IP Routing*类型。在某些情况下，需要定义一些路由参数指明用作客户网络网关的外部设备以便访问互联网。默认情况下，IP 路由以 WAN 端口上定义的网关为基础创建。但如果不应该使用默认路由，也可添加一个特定路由。详情请参阅技术参考手册 - 系统参数中的*Type IP Route*。

注意

如果互联网网关不是主网关，那么就必须将流向rseprod.abb.com和该DNS的信息传输定义为额外路线。

例如，如果互联网网关的 IP 地址为 100.100.100.22，rseprod.abb.com 的 IP 地址为 138.227.175.43（用 nslookup 验证），DNS 的 IP 地址为 8.8.8.8，则必须定义以下两条路线：

- 138.227.175.43/31 路由至 100.100.100.22
- 8.8.8.8/31 路由至 100.100.100.22

2.3.7 使用 FlexPendant 配置 Connected Services

2.3.7.1 简介

概述

本节说明了如何根据可用的互联网连通性使用控制器 FlexPendant 来配置Connected Services。互联网连接可通过多种方式来提供。

- Connected Services Gateway 模块（3G、Wi-Fi 或有线）
- 客户公共 (WAN) 网络上的直接互联网连接
- 在自定义网络上直接连接互联网

2.3.7.2 使用 FlexPendant 启用或禁用 Connected Services

启用或禁用 Connected Services

本节提供了有关使用 FlexPendant 启用或禁用 Connected Services 的信息。

> **注意**
>
> 在默认情况下，Connected Services 处于启用状态。

使用以下步骤可启用或禁用 FlexPendant 上的 Connected Services。

1. 在“开始”屏幕上，点击 **Settings（设置）**，然后从菜单中选择 **ABB Ability**。
2. 点击左侧窗格上的 **Connected Services**。
3. 在 **Enable Connected Services（启用 Connected Services）** 字段中点击并选择值 **Yes（是）**或 **No（否）**。
4. 点击 **Apply（应用）**。

   Restart（重新启动）确认消息随即显示。
5. 点击 **OK（确定）**。

   控制器随即重新启动，且 Connected Services 随即根据选择而启用或禁用。

2.3.7.3 根据连接类型使用 FlexPendant 配置 Connected Services

概述

Connected Services 可以通过以下方式来配置，具体取决于可用的连接类型：

- Ability
- 公共
- 自定义

配置 Ability 连接类型

连接 ABB Connected Services Gateway 解决方案后，Connected Services 将以 *Ability* 连接类型进行配置。

> **注意**
>
> 默认情况下，Ability 连接类型处于启用状态。

使用以下步骤可通过 FlexPendant 来配置 Ability 类型：

1. 在“开始”屏幕上，点击 **Settings（设置）**，然后从菜单中选择 **ABB Ability**。
2. 点击左侧窗格上的 **Connected Services**。

   Connected Services 的配置参数随即显示。
3. 在 **Connection Type（连接类型）**列表中，点击并选择 **Ability**。

   > **注意**
   >
   > Ability 网络可以根据可用的 CS Gateway（3G、Wi-Fi 或有线）来配置。有关详细信息，请参阅 操作手册 - *OmniCore*集成工程师指南。
4. 点击 **Apply（应用）**。

   Restart（重新启动）确认消息随即显示。
5. 点击**是**。

   控制器随即重新启动。

   Connected Services 将根据配置开始与服务器通信。

   检查连通性状态或事件日志。有关更多详细信息，请参阅 第74页的*Connected Services* 信息。

配置 Public 连接类型

当使用 IP、默认网关和通过 DHCP 接收的或静态配置的 DNS 在客户 Public (WAN) 网络上进行通信时，Connected Services 可以通过 *Public* 连接类型来配置。

> **注意**
>
> Public 网络和 DNS 可以静态配置，也可以（通过 DHCP）自动配置。有关更多详细信息，请参阅 第65页的使用 *FlexPendant* 配置公共网络。

下一页继续

2.3.7.3 根据连接类型使用 FlexPendant 配置 Connected Services
续前页

> **注意**
>
> 如果连接类型配置为**公共**，则在防火墙设置中启用 Connected Services。更多详细信息，请参阅操作手册 - *OmniCore*集成工程师指南中**防火墙设置**章节。

使用以下步骤可通过 FlexPendant 配置公共网络：

1. 在“开始”屏幕上，点击 **Settings（设置）**，然后从菜单中选择 **ABB Ability**。
2. 点击 **Connected Services**。
   连接服务的配置参数随即显示。
3. 在 **Connection Type（连接类型）**列表中，点击并选择 **Public**。
4. 点击 **Apply（应用）**。
   Restart（重新启动）确认消息随即显示。
5. 点击**是**。
   控制器随即重新启动。
   Connected Services 将根据配置开始与服务器通信。
   检查连通性状态或事件日志。有关更多详细信息，请参阅 第74页的*Connected Services* 信息。

配置 Custom 连接类型

当控制器必须指定在网络上可用的默认网关和 DNS 时，Connected Services 可以通过 *Custom* 连接类型来配置。

> **注意**
>
> 如果连接类型配置为**自定义**，则在防火墙设置中启用 Connected Services。更多详细信息，请参阅操作手册 - *OmniCore*集成工程师指南中**防火墙设置**章节。

使用以下步骤可通过 FlexPendant 来配置 Custom 类型：

1. 在“开始”屏幕上，点击 **Settings（设置）**，然后从菜单中选择 **ABB Ability**。
2. 点击左侧窗格上的 **Connected Services**。
   连接服务的配置参数随即显示。
3. 在 **Connection Type（连接类型）**列表中，点击并选择 **Custom**。
4. 在 **Internet Gateway IP（互联网网关 IP）**字段中，键入互联网网关的 IP 地址。
5. 在 **Internet DNS IP（互联网 DNS IP）**字段中，键入互联网 DNS 的 IP 地址。
6. 点击 **Apply（应用）**。
   Restart（重新启动）确认消息随即显示。
7. 点击 **OK（确定）**。
   控制器随即重新启动。
   Connected Services 将根据配置开始与服务器通信。
   检查连通性状态或事件日志。有关更多详细信息，请参阅 第74页的*Connected Services* 信息。

2.3.7.4 使用 FlexPendant 配置公共网络

静态配置 IP 和 DNS

使用以下步骤可通过 FlexPendant 静态地配置 IP 和 DNS：

1. 在“开始”界面，点击 **Settings（设置）**，然后从菜单中选择 **Network（网络）**。
2. 点击左侧窗格上的 **Public Network（公共网络）**。
3. 选择 **Use the following IP Address（使用以下 IP 地址）**或 **Use the following DNS server addresses（使用以下 DNS 服务器地址）**选项。
4. 在 **IP address（IP 地址）**、**Subnet mask（子网掩码）**、**Default gateway（默认网关）**、**Preferred DNS server（首选 DNS 服务器）**和 **Alternate DNS server（备用 DNS 服务器）**字段中输入值。
5. 点击 **Apply（应用）**。

   IP 和 DNS 随即静态配置。

自动配置 IP 和 DNS

使用以下步骤可通过 FlexPendant 自动地配置 IP 和 DNS：

1. 在“开始”界面，点击 **Settings（设置）**，然后从菜单中选择 **Network（网络）**。
2. 点击左侧窗格上的 **Public Network（公共网络）**。
3. 选择 **Automatically get an IP Address（自动获取 IP 地址）**或 **Automatically get DNS server addresses（自动获取 DNS 服务器地址）**选项。

   IP 地址和 DNS 服务器地址随即自动上传。
4. 点击 **Apply（应用）**。

   IP 和 DNS 随即自动配置。

2.3.7.5 使用 FlexPendant 配置与代理的互联网连接

操作步骤

以下步骤提供了有关与代理建立互联网连接时从 FlexPendant 配置 Connected Services 的信息。

1 在“开始”屏幕上，点击 **Settings（设置）**，然后从菜单中选择 **ABB Ability**。
2 点击 **Connected Services**。
Connected Services 的配置参数随即显示。
3 在 **Proxy Used（使用代理）**字段中，将值更改为 **Yes（是）**。
代理参数随即显示。
4 在 **Proxy Name（代理名称）**字段中，键入代理的名称。
5 在 **Proxy Port（代理端口）**字段中，键入代理端口编号。
6 在 **Proxy Auth（代理身份验证）**字段中，从下拉列表中选择 **Basic** 以进行基本身份验证，或选择 **None** 以不进行身份验证。

> **注意**
>
> 定义基本认证所需的代理用户名和密码。即使使用了代理，也必须定义 DNS 来进行名称解析。

7 点击 **Apply（应用）**，然后重新启动控制器以使更改生效。

2.3.8 使用 RobotStudio 配置 Connected Services

2.3.8.1 简介

概述

本节说明了如何根据可用的互联网连通性使用控制器 RobotStudio 来配置Connected Services。互联网连接可通过多种方式来提供。

- Connected Services Gateway 模块（3G、Wi-Fi 或有线）
- 客户公共 (WAN) 网络上的直接互联网连接
- 在自定义网络上直接连接互联网

2.3.8.2 使用 RobotStudio 启用或禁用连接服务。

启用或禁用连接服务。

本节提供了有关使用 RobotStudio 启用或禁用 Connected Services 的信息。

> **注意**
>
> 默认情况下，连接服务处于启用状态。

使用以下步骤可管理连接服务功能的启用和禁用：

1. 在 RobotStudio 中添加控制器。
2. 单击控制器 **Configuration（配置）**。
3. 右键单击 **Communication（通信）**，然后选择 **Configuration Editor（配置编辑器）**。

   Configuration Editor（配置编辑器）随即显示。
4. 选择 **Connected Services**。

   连接服务的配置参数随即显示。
5. 右键单击任意字段，然后选择 **Edit Connected Services（编辑 Connected Services）**。

   Instance Editor（实例编辑器）随即显示。
6. 在 **Enabled（已启用）**字段中选择值 **Yes（是）**或 **No（否）**。
7. 点击 **OK（确定）**。
8. 重启控制器.

   连接服务将根据选择而启用或禁用。

2.3.8.3 根据连接类型使用 RobotStudio 来配置连接服务

概述

连接服务可以通过以下三种方式来配置，具体取决于可用的连接类型：

- **Ability**
- **Public（公共）**
- **Custom（自定义）**

配置 Ability 连接类型

连接 ABB Connected Services Gateway 解决方案后，连接服务将以 **Ability** 连接类型进行配置。

> **注意**
>
> 默认情况下，**Ability** 连接类型处于启用状态。

使用以下步骤可通过 RobotStudio 来配置 **Ability** 连接类型：

1. 在 **Controller（控制器）**选项卡中，单击 **Configuration（配置）**。
2. 右键单击 **Communication（通信）**，然后选择 **Configuration Editor（配置编辑器）**。

   Configuration Editor（配置编辑器）随即显示。
3. 选择 **Connected Services**。

   连接服务的配置参数随即显示。
4. 右键单击任意字段，然后选择 **Edit Connected Services（编辑 Connected Services）**。

   Instance Editor（实例编辑器）随即显示。
5. 在 **Connection Type（连接类型）**字段中，选择值 **Ability Network（Ability 网络）**。

   > **注意**
   >
   > Ability 网络可以根据可用的 CS Gateway（3G、Wi-Fi 或有线）来配置。有关详细信息，请参阅 操作手册 - *OmniCore*集成工程师指南。
6. 点击 **OK（确定）**。
7. 重启控制器.

   连接服务将根据配置开始与服务器通信。

   在设备浏览器中检查连通性状态。有关更多详细信息，请参阅 第74页的 *Connected Services* 信息。

   亦请参阅生成的事件日志。有关更多详细信息，请参阅 技术参考手册 - *RobotWare 7*事件日志。

下一页继续

2.3.8.3 根据连接类型使用 RobotStudio 来配置连接服务
续前页

配置公共连接类型

当在客户 WAN 网络上进行通信且存在接收到的默认网关和 DNS 时，连接服务可以通过 **Public（公共）**连接类型来配置。

> **注意**
>
> 公共网络和 DNS 可以静态配置，也可以（通过 DHCP）自动配置。有关更多详细信息，请参阅 第72页的使用 *RobotStudio* 配置公共网络。

> **注意**
>
> 如果连接类型配置为**公共**，则在防火墙设置中启用 Connected Services。更多详细信息，请参阅操作手册 - *OmniCore*集成工程师指南中防火墙设置章节。

使用以下步骤可通过 RobotStudio 来配置 **Public（公共）**连接类型：

1. 在 **Controller（控制器）**选项卡中，单击 **Configuration（配置）**。
2. 右键单击 **Communication（通信）**，然后选择 **Configuration Editor（配置编辑器）**。

   Configuration Editor（配置编辑器）随即显示。
3. 单击 **Connected Services**。

   连接服务的配置参数随即显示。
4. 右键单击任意字段，然后选择 **Edit Connected Services（编辑 Connected Services）**。

   Instance Editor（实例编辑器）随即显示。
5. 在 **Connection Type（连接类型）**字段中，选择值 **Public Network（公共网络）**。
6. 点击 **OK（确定）**。
7. 重启控制器.

   连接服务将根据配置开始与服务器通信。

   在设备浏览器中检查连通性状态。有关更多详细信息，请参阅 第74页的 *Connected Services* 信息。

   亦请参阅生成的事件日志。有关更多详细信息，请参阅 技术参考手册 - *RobotWare 7*事件日志。

配置自定义连接类型

当控制器必须指定在网络上可用的默认网关和 DNS 时，连接服务可以通过 **Custom（自定义）**连接类型来配置。

> **注意**
>
> 如果连接类型配置为**自定义**，则在防火墙设置中启用 Connected Services。更多详细信息，请参阅操作手册 - *OmniCore*集成工程师指南中防火墙设置章节。

使用以下步骤可通过 RobotStudio 来配置 **Custom（自定义）**连接类型：

1. 在 **Controller（控制器）**选项卡中，单击 **Configuration（配置）**。

下一页继续

2 右键单击 **Communication（通信）**，然后选择 **Configuration Editor（配置编辑器）**。

Configuration Editor（配置编辑器）随即显示。

3 单击 **Connected Services**。

连接服务的配置参数随即显示。

4 右键单击任意字段，然后选择 **Edit Connected Services（编辑 Connected Services）**。

Instance Editor（实例编辑器）随即显示。

5 在 **Connection Type（连接类型）**字段中，选择值 **Private Network（专用网络）**。

6 在 **Internet Gateway IP（互联网网关 IP）**字段中，键入互联网网关的 IP 地址。

7 在 **Internet DNS IP（互联网 DNS IP）**字段中，键入互联网 DNS 的 IP 地址。

8 点击 **OK（确定）**。

9 重启控制器.

连接服务将根据配置开始与服务器通信。

在设备浏览器中检查连通性状态。有关更多详细信息，请参阅 第*74*页的 *Connected Services* 信息。

亦请参阅生成的事件日志。有关更多详细信息，请参阅 技术参考手册 - *RobotWare 7*事件日志。

2.3.8.4 使用 RobotStudio 配置公共网络

静态配置 IP

使用以下步骤可通过 RobotStudio 静态地配置 IP

1. 右键单击控制器，然后选择 **Properties（属性）** > **Network Settings（网络设置）**。

   Network settings（网络设置）窗口随即显示。
2. 选择 **Use following IP address（使用以下 IP 地址）**选项。
3. 在 **IP address（IP 地址）**、**Subnet mask（子网掩码）**和 **Default gateway（默认网关）**字段中输入值。
4. 单击 **OK（确定）**，然后重新启动控制器

   IP 随即配置。

静态配置 DNS

以下步骤提供了有关与静态 DNS 建立直接互联网连接时从 RobotStudio 配置 Connected Services 的信息。

1. 在 **Controller（控制器）**选项卡中，单击 **Configuration（配置）**。
2. 右键单击 **Communication（通信）**，然后选择 **Configuration Editor（配置编辑器）**。

   Configuration Editor（配置编辑器）随即显示。
3. 单击 **DNS Client（DNS 客户端）**。

   DNS 客户端的配置参数随即显示。
4. 右键单击任意字段，然后选择 **Edit DNS Client(s)（编辑 DNS 客户端）**。

   Instance Editor（实例编辑器）随即显示。
5. 在 **Enabled（已启用）**字段中，将值更改为 **Yes（是）**。
6. 在 **1st Name Server（第 1 名称服务器）**字段中，键入服务器 IP。
7. 单击 **OK（确定）**，然后重新启动控制器，以使更改生效。

自动配置 IP (DHCP)

使用以下步骤可通过 RobotStudio 自动地配置 IP

1. 右键单击控制器，然后选择 **Properties（属性）** > **Network Settings（网络设置）**。

   Network settings（网络设置）窗口随即显示。
2. 选择 **Obtain an IP address automatically（自动获取 IP 地址）**选项。

   IP 地址随即自动上传。
3. 单击 **OK（确定）**，然后重新启动控制器

   IP 和 DNS 应被自动接收。

> **注意**
>
> 用户仍然可以使用自动 IP 来定义手动 DNS。如果手动 DNS 与自动 DNS 之间存在冲突，则以手动 DNS 为准。

2.3.8.5 使用 RobotStudio 配置与代理的互联网连接

操作步骤

下列步骤提供了在控制器有使用代理的互联网连接时从RobotStudio配置Connected Services的信息。

1 在 **Controller（控制器）**选项卡中，单击 **Configuration（配置）**。

2 右键单击 **Communication（通信）**，然后选择 **Configuration Editor（配置编辑器）**。

Configuration Editor（配置编辑器）随即显示。

3 选择 **Connected Services**。

连接服务的配置参数随即显示。

4 右键单击任意字段，然后选择 **Edit Connected Services（编辑 Connected Services）**。

Instance Editor（实例编辑器）随即显示。

5 在 **Proxy Used（使用代理）**字段中，选择值 **Yes（是）**。

代理参数随即显示。

6 在 **Proxy Name（代理名称）**字段中，键入代理的名称。

7 在 **Proxy Port（代理端口）**字段中，键入代理端口编号。

8 在 **Proxy Auth（代理身份验证）**字段中，从下拉列表选择 *Basic* 以进行基本身份验证，或选择 *None* 以不进行身份验证。

> **注意**
>
> 为基本身份验证定义 **代理用户**和**代理密码**字段。即使使用了代理，也必须定义 DNS 来进行名称解析。

9 点击 **OK（确定）**。

控制器随即重新启动，且更改随即应用。

2.3.9 Connected Services 信息

Connected Services 页面

简介

在 FlexPendant 中，**Settings（设置）** > **ABB Ability™** > **Connected Services Status** 下提供了 Connected Services 信息页面。以下 Connected Services Status 页面可供使用：

- **概述**
- **Connectivity（连通性）**
- **注册**
- **Advanced（高级）**

> **注意**
>
> 在 RobotStudio 中，**Controller Properties（控制器属性）** > **Device Browser（设备浏览器）** > **Software resources（软件资源）** > **Communication（通信）** > **Connected Services** 提供了 Connected Services 信息页面。

> **注意**
>
> 页面上的信息可以通过更改页面或按下 **Refresh（刷新）** 按钮来刷新。如果软件代理处于等待状态（例如等待来自 MyRobot 的注册确认），**Refresh（刷新）** 按钮还可以强制建立与服务器的连接。当服务器轮询设置为 Slow 时，如果进行慢速轮询，则此按钮将大有帮助。

概述页

Overview（概览） 页面汇总了 Connected Services 状态和信息。如果它处于不活动状态，则其他页面会提供更多详细信息。

字段	描述	可能值	示例
属性			
启用	显示用于开启/关闭 Connected Services 的主配置开关的值。	Yes/No	Yes
状态	显示当前状态以查看是否有需要浏览到服务器连接页或注册页。	有关值的说明，请参阅 第81页的 *CSE* 状态。	Active
序列号	显示用于在 Connected Services 中识别控制器的标识符。	Controller Serial number	12-45678
Robot OS Version（Robot OS 版）	显示发送到服务器的 Robot OS 版。	Robot OS Version number	RobotOS_1.00.0-379
Robot Control version（Robot Control 版）	显示发送到服务器的 Robot Control 版。	Robot Control version name	RobotControl_7.0.0-405.Internal+405
服务协议	用于确认控制器是否与预期的服务协议关联。	“服务协议名称” "-"	SA_FR12_16

下一页继续

字段	描述	可能值	示例
客户名称	用于确认控制器是否与预期的服务协议关联。	“服务协议的客户名称” "-"	ABB Robotics
国家/地区	用于确认控制器是否与预期的服务协议关联。	“服务协议的国家/地区” "-"	法国
ABB 服务器	显示connected services连接的服务器类型。	机器人云	机器人云

“Connectivity（连通性）”页面

Connectivity（连通性）页面总结了 Connected Services 与服务器之间的连通性。

字段	描述	可能值	示例
状态	显示当前状态以查看是否需要导航至 **Connectivity（连通性）**页面或 **Registration（注册）**页面。	有关值的说明，请参阅 *CSE 状态*。	Active
连接状态	展示与相关服务器之间的通信状态和错误类型。	有关值的说明，请参阅 *CSE 连接状态*。	Connected
上次更新	显示自 **Connectivity（连通性）**页面生成以来经过的相对时间。		“HH:MM:SS前”
Hardware gateway（硬件网关）	显示硬件网关和连接 IP 的类型。		DSQC 1039 3G 已通过以下 IP 建立连接：192.168.126.1
服务器名称	展示配置了软件代理的服务器的名称。	"" 服务器名称	CS 1.0 - rseprod.abb.com CS 2.0 - cse.robotics.abb.com
服务器IP	显示了服务器的 IP 地址以及用于连接的端口编号。IP 地址是软件代理完成的 DNS 名称解析的结果。	"" 服务器IP	138.227.175.43 for rseprod.abb.com 13.79.129.11 for cse.robotics.abb.com
服务器证书名称	显示服务器证书的名称信息。	"" 服务器名称 不受信任（服务器）	CS 1.0 rseprod.abb.com CS 2.0 cse.robotics.abb.com
服务器证书签发方	显示服务器证书发布者的名称。	"" 签发方 不受信任（签发方）	ABB issuing CA 6 DigiCert SHA2 安全服务器 CA
服务器证书有效期开始日期	显示服务器证书的日期。	"" 签发方 签发（日期）	格林威治标准时间 2020 年 10 月 02 日 08 时 07 分 12 秒
服务器证书有效期至	显示服务器证书的日期。	"" 签发方 失效（日期）	Nov 21 07:09:28 2021 GMT
客户端证书设备	显示客户端证书设备的名称。		07-000036

下一页继续

2.3.9 Connected Services 信息

续前页

字段	描述	可能值	示例
Client Certificate issuer（客户端证书签发方）	显示服务器证书签发方的名称。		Remote-Service-PROD-Issuing-CA-1
Client Certificate valid from（客户端证书有效期开始于）	显示客户端证书从哪个日期开始有效。		格林威治标准时间 2019 年 3 月 15 日 05 时 38 分 41 秒
Client Certificate valid until（客户端证书有效期结束于）	显示客户端证书从哪个日期开始失效。		格林威治标准时间 2020 年 3 月 15 日 05 时 38 分 41 秒
Client Certificate serial number（客户证书序列号）	显示客户端证书的序列号。		15E37B17000100002D0B
Internet Ip（互联网 IP）	显示用于连接到互联网的 IP。		106.197.204.16
控制器时间	显示控制器日期和时间详情。 **注意** 很重要的一点就是在相关控制器中设置正确的时间（证书流程中需要该时间）。		16-01-08 13:52:33
Connection type（连接类型）	显示所用网络连接的类型。	Ability	
Public network information（公共网络信息）	显示公共端口的网络信息。	NoIP Static DHCP	DHCP: 10.140.198.55/ 255.255.255.0/ 10.140.198.1/ plugged
Ability network information（Ability 网络信息）	显示 Ability 端口的网络信息。	NoIP Static DHCP	Static: 192.168.126.2/ 255.255.255/ 0.0.0.0/ plugged
System DNS（系统 DNS）	显示 DNS 服务器信息。		8.8.8.8:53
Gateway used（所用网关）	显示用于创建路由的网关。		192.168.126.1
DNS used（所用 DNS）	显示当前使用的 DNS 值。		192.168.126.1:53
路由 1-8	显示 Connected Services 所创建的路由。		路由1：13.79.129.11/32 => 192.168.126.1

注册页

Registration（注册）页面总结了 Connected Services 注册。

字段	描述	可能值	示例
状态	显示当前状态以查看是否有需要浏览到**服务器连接**页或**注册**页。	有关值的说明，请参阅 *第81页的 CSE 状态*。	Active

下一页继续

字段	描述	可能值	示例
注册状态	显示注册状态。	有关值的说明，请参阅 第82页的 *CSE* 注册状态。	Register with code in MyRobot
注册码	显示注册代码。该代码可以用于通过 MyRobot 进行注册。	"-" 代码值	456735

高级页

Advanced（高级）页面提供了有关软件代理与服务器之间对话框的高级信息。

字段	描述	可能值	示例
上一个HTTP消息	显示发出的上一个消息。	Register CheckRegistered LogMessage GetMessage GetConfig SendGetSpecificCode DownLoadFile AcknowledgeMessage BoxUpload GetRSEAgreementInformation SendDeviceInformation RequestClientCertificate RenewClientCertificate periodDeviceUpdateInformation	GetMessage
上一个HTTP消息时间	显示上一个上个消息发送时的日期和时间。		Sent 00:01:28 ago
上一个HTTP错误	显示上一个消息发送时如果消息ID为4XX的HTTP错误。	Not Available -如果无错误 错误 HTTP XXX + 消息	Not Available
下一个消息	显示要发送的下一个消息以及发送消息的日期。		70秒内获取消息
上一个命令	显示从服务器接收的上一个命令。	Not Available Reboot Reset Ping Diagnostic ...	Not Available
重启计数器	显示软件代理已自动重新启动的次数。此页面用于查看监视器是否已重新启动。	0-N If not Enabled, then display 0	2
连接器版本	显示当前运行的连接器版本信息。		1.0.0

下一页继续

2.3.9 Connected Services 信息

续前页

字段	描述	可能值	示例
Data Collector status（Data Collector 状态）	显示数据收集器的状态。	Started Not Started None Downloaded Download Failed Failed Stopped Disabled	Started
最近注册	显示控制器服务器的最后注册日期和时间。		
最近连接	显示控制器检测到与服务器成功连接的最后日期和时间。		11-07-2020 08:33:35
服务器轮询	显示服务器轮询配置。服务器轮询与连接成本的计算有关。	Fast Slow	Fast
Bytes sent/received（已发送/接收的字节）	显示已发送/接收的字节数量。		17.0KB/24.17KB
Connected Service mode（Connected Service 模式）	显示连接服务模式的状态。	有关值的说明，请参阅 第82页的 *CSE* 模式。	Active Mode
Server errors（服务器错误）	展示以下服务器错误的计数值： 1 连接错误： 2 连接不可用错误 3 身份验证错误 4 请求错误： 5 超时错误： 6 代理错误 7 未知错误	0-N代表每台服务器的错误	5 0 1 0 1 0 0
Root certificate issuer（根证书签发方）	显示根证书签发方的名称。		Baltimore CyberTrust Root DigiCert Global Root CA
Root certificate subject（根证书主题）	显示根证书主题的名称。		Baltimore CyberTrust Root DigiCert Global Root CA
Root certificate valid from（根证书有效期开始于）	显示根证书从哪个日期开始有效。		May 12 18:46:00 2000 GMT May 12 18:46:00 2006 GMT
根证书有效期结束于	显示根证书从哪个日期开始失效。		May 12 18:46:00 2025 GMT May 12 18:46:00 2031 GMT

下一页继续

Ability 页面

Ability 页面提供了关于 Ability 连接的高级信息、证书信息以及数据交易的详细信息。

字段	描述	可能值	示例
Ability 连接状态	显示 Ability 云的连接状态。	关于可能的值的列表及其说明，请参阅第*82*页的*Ability 状态*。	Connected to IoT Hub
Ability 连接错误计数	显示 Ability 云连接的连接错误数。	0-N	2
Ability 设备标识	显示 Ability 云连接的设备标识。		1ca930c7-e77f-4265-8005-0ee493b84eeb
Ability 客户端证书设备	显示用于 Ability 云连接的客户端证书的通用名。		1ca930c7-e77f-4265-8005-0ee493b84eeb
Ability 客户端证书签发方	显示 Ability 客户端证书签发方的名称。		ABB Ability(tm) Issuing CA
Ability 客户端证书有效期开始日期	显示 Ability 客户端证书开始生效的日期和时间。	-1 --如果没有数据 日期和时间（如果数据已更新）	Jul 2 00:00:00 2020 GMT
Ability 客户端证书有效期结束日期	显示 Ability 客户端证书有效期的日期和时间。	-1 --如果没有数据 日期和时间--如果数据被更新	Jul 3 23:59:59 2020 GMT
Ability 客户端证书序列号	显示 Ability 客户端证书的序列号。		063433724FA28636BCB72916FF472CF6
DPS 服务器名称	显示 DNS 服务器名称。		global azure-devices-provisioning.net
DPS 服务器 IP	显示 DPS 服务器的解析 IP 地址。		52.163212.39
DPS 作用域 ID	显示云配置的 DPS 作用域 ID。		0neOOOA3934
DPS 根证书签发方	显示 DPS 根证书签发方的名称。		Microsoft IT TLS CA 4
DPS 根证书主题	显示 DPS 根证书的主题。		.azure-devices-provisioning.net
DPS 注册组 ID	显示 DPS 注册组 ID。		ABBAbilitymIssuingCA
物联网中心服务器名称	显示 Ability 物联网中心服务器的名称。		[illegible]
物联网中心服务器 IP	显示 Ability 物联网中心服务器的解析 IP 地址。	Resolved IP address IP not resolved	13.79.172.43
物联网中心 blob 存储名	显示Ability 物联网中心的 blob 存储名。		
物联网中心 blob 存储 IP	显示Ability 物联网中心 blob 存储服务器的解析 IP 地址。	Resolved IP address IP not resolved	52.239.137.68
最近发送数据类型	显示最后发送的信息模型数据类型。		[illegible]
最近发送信息	显示最后成功发送的遥测数据。		

下一页继续

2.3.9 Connected Services 信息
续前页

字段	描述	可能值	示例
最近信息发送时间	显示最后一次成功发送遥测信息的日期和时间。	-1 --如果没有数据 日期和时间（如果数据已更新）	Thu Jul 2 17:45:53 2020
最近发送所有者 ID	显示最后发送的信息模型的所有者 ID。		34554e14-71 c6-49c0-a2a8·819e 1a6c 17cd
最近发送对象 ID	显示最后发送的信息模型的对象 ID。		f827fl6f-68b3-4 785-adfd-f1e8f7f06bda
最近发送租户 ID	显示最后发送的信息模型的租户 ID。		8b5a949fe69b-41 a5·b198-1cbd5caad30c
最近尝试发送信息	显示试图发送的最后一条遥测信息。		
最近信息发送失败时间	显示最后一次发送遥测信息失败的日期和时间。	-1 --如果没有数据 日期和时间（如果数据已更新）	-1
遥测信息数量	显示成功发送遥测信息的次数。	0-N	12
已发送/接收遥测字节	显示上传和下载的遥测数据量。		4146/1086 B
文件上传数	显示成功上传至物联网 blob 存储的文件数量。	0-N	0
已发送/接收文件字节	显示用于上传和下载文件的数据量。		0/0 B
遥测失败次数	显示发送遥测信息失败的次数。	0-N	1
文件上传失败次数	显示上传到物联网 blob 存储失败的文件数量。	0-N	0

数据收集器页面

数据收集器是用于收集 ABB Ability™ Connected Services Cloud 所需数据的组件。数据收集器可从 ABB Ability™ Cloud 进行 OTA 更新。

数据收集器页面提供了关于不同数据收集器的状态和版本细节的信息。

字段	描述	可能值	示例
IRC5 兼容性	显示旧数据收集器的状态和版本信息。	Disabled 版本信息	1.0.4
运动数据采集器	显示运动数据收集器的运行状态和版本信息。	Disabled 版本信息	1.0.1
机器人系统数据收集器	显示机器人系统数据收集器的运行状态和版本信息。	Disabled 版本信息	1.0.0

注意

IRC5 兼容性是用于 CSE 1.0，而运动数据收集器和机器人系统数据收集器是用于 CSE 2.0。

下一页继续

Connected Services 信息中值的说明

CSE 状态

下表给出了 CSE 状态的信息：

代码	值	描述
BASE_FAILED	失败	Connected Services 状态为失败。
BASE_INITIALIZING	Initializing	Connected Services 状态为正在初始化。
BASE_ACTIVE	活动	Connected Services 状态为活动。
BASE_CONNECT	尝试连接	Connected Services 状态为正在尝试连接。
BASE_SHUTDOWN	关机模式	Connected Services 状态为关机模式。
UNKNOWN_STATUS	未知	Connected Services 状态为未知。
BASE_SLEEP	睡眠	Connected Services 状态为睡眠模式。

CSE 连接状态

下表给出了 CSE 连接状态的信息：

代码	值	描述
SERVER_REQUEST_TIMEOUT_ERROR	请求已超时	连接状态请求已超时。
SERVER_CONNECTED	已连接	连接状态为已连接。
SERVER_NETWORK_ERROR	服务器无法访问	连接状态服务器无法访问。
SERVER_AUTH_ERROR	服务器未经身份验证	连接状态服务器未经身份验证。
SERVER_CERT_ERROR	服务器认证验证错误	连接状态服务器显示为认证验证错误。
SERVER_HTTP_ERROR	服务器错误 (HTTP)	连接状态服务器显示为 HTTP 错误。
SERVER_PROXY_AUTH_ERROR	需要代理身份验证	连接状态显示为需要代理身份验证错误。
SERVER_REQUEST_ERROR	请求错误	连接状态服务器显示为请求错误。
SERVER_PROXY_CONN_ERROR	代理连接错误	连接状态显示为代理连接错误。
SERVER_GATEWAY_DISABLED	CS Gateway 已禁用	连接状态显示为网关已禁用。
SERVER_GATEWAY_WAITING	识别 CS Gateway	连接状态显示为网关正在识别。
SERVER_GATEWAY_WAITING_3G	等待 3G 连通	连接状态显示为网关正在等待 3G 连通。
SERVER_GATEWAY_WAITING_WIFI	等待 Wi-Fi 连通	连接状态显示为网关正在等待 Wi-Fi 连通。
SERVER_GATEWAY_CONNECTED	CS Gateway 已连接	连接状态显示为网关已连接。
SERVER_LOOKUP_IN_PROGRESS	主机名查找进行中	连接状态显示为网关主机名查找正在进行中。
WAITING_GATEWAY_IP	等待网关 IP	连接状态显示为网关正在等待网关 IP。
SERVER_DNS_RESOLUTION_FAIL	DNS 解析失败	连接状态显示为网关遭遇服务器 DNS 解析失败。
SERVER_GW_PING_TIMEOUT	网关 ping 超时	连接状态显示为网关服务器 ping 已超时。

下一页继续

2.3.9 Connected Services 信息
续前页

代码	值	描述
DNS_NOT_PINGABLE	DNS 无法执行 ping	连接状态显示为网关具有无法执行 ping 的 DNS。
AUTHN_ERROR	身份验证错误	连接状态显示为网关遭遇身份验证错误。
RESTART_ERROR	发生重新启动错误后等待启动	连接状态显示为发生重新启动错误后网关正在等待启动。
RESET_ERROR	发生重置启动错误后等待启动	连接状态显示为发生重置错误后网关正在等待启动。
CSE_SLEEP_MODE	CSE 处于配置的睡眠模式	连接状态显示为网关处于睡眠模式。

CSE 注册状态

下表给出了 CSE 注册状态的信息：

代码	值	描述
REG_REGISTERED	已注册	注册状态为已注册。
REG_IN_PROGRESS	注册进行中	注册状态为进行中。
REG_REGISTER	在 MyRobot 中使用代码注册	注册状态为已在 MyRobot 中使用代码注册。
REG_DISABLED	注册已禁用	注册状态为已禁用。
REG_FAILED	失败	注册状态为失败。

CSE 模式

下表给出了 CSE 模式的信息：

代码	值	描述
CS_MODE_BOOT	启动模式	Connected Services 处于初始化期间。
CS_MODE_REGISTRATION	注册模式	Connected Services 处于注册模式下。
CS_MODE_ACTIVE	活动模式	Connected Services 处于注册成功后时期。
CS_MODE_RESET	重置模式	Connected Services 处于重置连接服务期间。
CS_MODE_SLEEP	睡眠模式	当执行操作存在延迟时。
CS_MODE_SHUTDOWN	关机模式	当撤销之前挂起连接服务时。

Ability 状态

下表给出了关于 Ability 状态的信息：

代码	值	描述
STATUS_CONFIG_PEND	配置待处理	Ability 配置待处理。
STATUS_CONFIG_DONE	配置更新	Ability 配置更新。
STATUS_CLIENT_CERT_PEND	客户端证书待处理	等待领取新的 Ability 证书。
STATUS_CLIENT_CERT_UPDATED	客户端证书更新	Ability 客户端证书更新。
STATUS_DPS_INITIATED	启动 DPS	启动 DPS 连接。
STATUS_IOT_HUB_ASSIGNED	指定的物联网中心	物联网中枢被指定。
STATUS_DPS_FAILED	DPS 失败	DPS 连接失败。
STATUS_IOT_HUB_CONNECTED	连接到物联网中心	连接到物联网中心。

下一页继续

代码	值	描述
STATUS_IOT_HUB_DISCONNECTED	与物联网中心断开连接	与物联网中心断开连接。
STATUS_UNAUTHORIZED	未经授权	未经授权的连接。
STATUS_DISABLED	禁用（如果 CSE 模式为 CSE 1.0 IRC5 Legacy）	禁用。

数据收集状态

下表给出了数据收集状态的信息：

代码	值	描述
SPECIFIC_STATUS_NONE	无	数据收集状态为无。
SPECIFIC_STATUS_DOWNLOADED	下载	数据收集状态为已下载。
SPECIFIC_STATUS_DOWNLOAD_FAILED	下载失败	数据收集状态为下载失败。
SPECIFIC_STATUS_STARTED	已启动	数据收集状态为已启动。
SPECIFIC_STATUS_STOPPED	Stopped	数据收集状态为已停止。
SPECIFIC_STATUS_FAILED	失败	数据收集状态为失败。

服务器轮询事件的行为描述

下表给出了关于服务器轮询事件行为的信息：

事件	慢	快
检查注册	30 分钟	10 分钟
发送硬件信息（注册前） （仅在有变化的情况下才会发送定期轮询。）	30 分钟	10 分钟
发送硬件信息（注册后）	24 小时	24 小时
发送存活信息	50 分钟	50 分钟
检查模块更新	8 小时	4 小时
发送动态信息	4 小时	4 小时
内部待办命令	10 分钟	1 分钟
取消注册后重新启动	重新启动监察（4 分钟）	

Connected Services 事件日志

软件代理会在中央控制器事件日志中生成事件日志。事件日志会在启动、注册、注销、失去连接期间以及其他重要事件期间生成。

事件日志处于 170XXX 的范围内，并与所有其他控制器事件日志文档一起描述。有关更多详细信息，请参阅 技术参考手册 - *RobotWare 7*事件日志。

强制重置软件“代理程序”

用户可以重置软件代理。当您重置软件代理时，软件代理将擦除其所有内部信息，包括注册信息、数据收集器脚本和所有本地存储的服务信息。虽然配置将不会被重置，但必须使用新注册才能重新激活连接服务。

使用以下步骤可通过 FlexPendant 来重置软件代理：

1 打开 **Operate（操作）**。
2 点击 **Service Routines（保养例行程序）**。

下一页继续

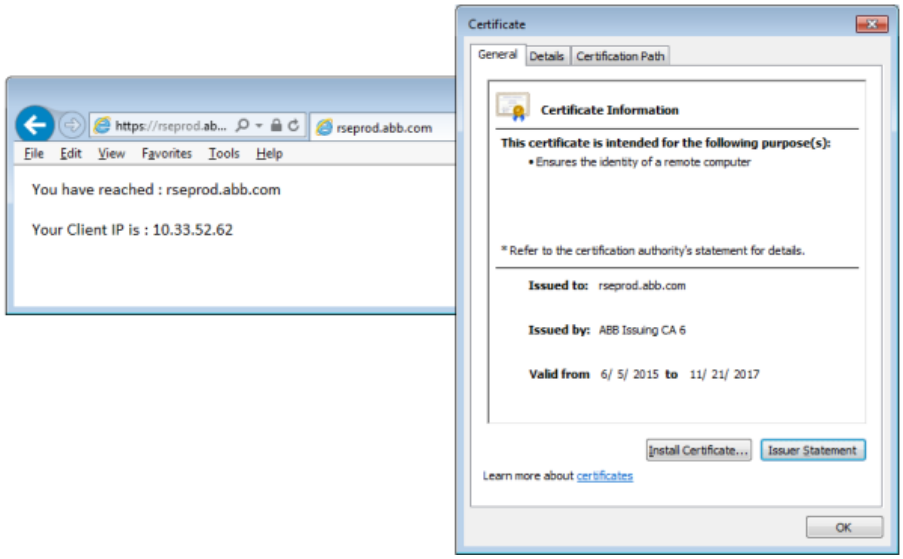
3 点击 **Connected Services Reset（所连服务重置）**。
ConnectedServiceReset（连接服务重置）窗口随即显示。
4 点击**是**。
5 按下 FlexPendant 上的 **Start（启动）**硬按钮。
伴有操作员消息的确认页面随即显示。
6 点击 **Reset（重置）**。
软件代理随即重置。

2.3.10 故障排除

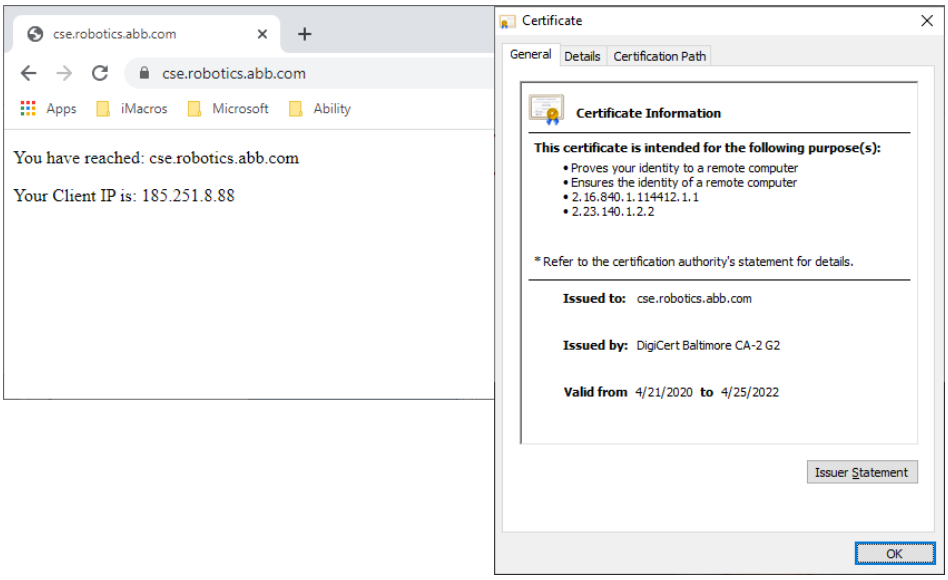
2.3.10.1 服务器连通性故障排除

概述

您可从您所在之处验证控制器到 Connected Services 公共连接服务器的连通性。连接具有相同网络配置（WAN IP/掩码、DNS、路由）的 PC（替代控制器），然后在浏览器中打开服务器根目录路径 （*https://rseprod.abb.com* (CSE 1.0) 或 *https://cse.abbrobotics.abb.com* (CSE 2.0)），即可完成验证。如果 DNS 名称已解析，则连通性验证通过，浏览器将显示指示 CS 服务器的页面并使用 ABB 证书进行保护，如下图所示。



xx1500003225



xx2000001953

下一页继续

2.3.10.1 服务器连通性故障排除
续前页

注意

在 CSE 2.0 的情况下，为了连接到 ABB Ability，微软 Azure 上的以下服务器需要通过端口 443 (HTTPS/MQTT) 可访问：

- DPS: global.azure-devices-provisioning.net
- IoTHub: *.azure-devices.net
- BlobStorage: *.blob.core.windows.net

连接服务网关

有关更多详细信息，请参阅 操作手册 - *OmniCore*集成工程师指南 中的“ABB Ability™ Connected Service 配置”一节。

网络安全

更多详细信息，请参阅操作手册 - *OmniCore*集成工程师指南中的“OmniCore网络安全”一节。

时间精度

在控制器中正确设置时间（包括时区）是非常重要的，可以手动设置或使用 NTP 设置。有关详细信息，请参阅操作手册 - *OmniCore*中的**更改日期和时间**一节。

2.3.10.2 3G/Wi-Fi 连通性故障排除

概述

此选项用于检查连通性模块的当前连接状态以进行故障排除。

> **注意**
>
> 连接日志仅适用于 Connected Services Gateway 3G 和 Wi-Fi 连接（不适用于 Wired 连接）。

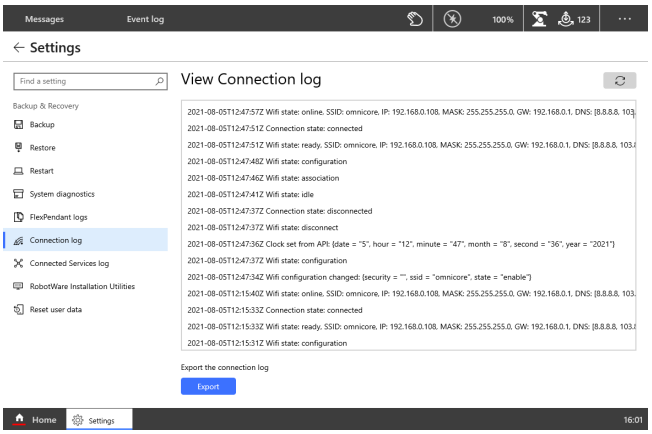
操作步骤

使用以下步骤可检查连通性模块的当前连接状态：

1. 打开 **Settings**（设置）。
2. 点击 **Backup**（备份） **& Recovery**（恢复）。
3. 在左侧边栏上点击 **Connection log**（连接日志）。

   Connection log（连接日志）页面随即显示，且日志将显示在窗口上。

   > **注意**
   >
   > 点击“Refresh（刷新）”按钮以更新日志。



xx1900000976

4. 点击 **Export**（导出）。
5. 如果需要，在 **File Name**（文件名称）字段中编辑文件的名称。
6. 如果需要，为了更改存储路径，请在 **Folder Name**（文件名称）字段中点击 **Browse**（浏览），然后选择所需的路径。
7. 点击 **Create**（创建）。

   连通性模块的当前连接状态随即保存在所选的路径中。

2.3.10.3 如何从控制器获得 CSE（嵌入式互联服务）日志

操作步骤

使用以下程序从控制器中检索 CSE 日志:

1 打开 RobotStudio，单击 **Controller（控制器）**选项卡，然后添加控制器。

> **注意**
>
> 有关添加控制器的更多详细信息，请参阅 操作手册 - *RobotStudio*。

2 在 **Configuration（配置）**组中，单击 **Properties（属性）**，然后选择 **Save diagnostics（保存诊断）**。

Save As（另存为）窗口随即显示。

3 单击 **Save（保存）**。

文件随即保存在所选的位置。

4 将完整的诊断文件发送至 ABB 支持部门，以便进行进一步处理。

2.3.10.4 嵌入式互联服务故障排除日志

嵌入式互联服务故障排除日志和描述

嵌入式互联服务 (CSE) 将一些日志信息发送给连接性管理安全服务器 (CMSS)。这些日志信息被分类为两种类型，即定期日志和非定期日志。定期日志含有 CSE 健康状态。在 CSE 进入特定的状态时，比如已注册、数据收集器已启用以及其他状态等，就会发送出非定期日志。这些日志信息有助于 CSE 的故障排除工作。可以从内部的 ABB 互联服务支持工具访问这些日志。

嵌入式互联服务基础日志

在以下表格中给出嵌入式互联服务基础日志的列表及其描述：

日志编号 日志名称	描述
3000 BASE_MODE_UNKNOWN	控制器从外部来源接收到不被支持的重置命令。
3170 BASE_MODE_FAIL_UPDATE	IRC5 兼容性数据收集器更新已经失败。
8000 GLOBAL_INIT_OK	嵌入式互联服务 (CSE) 在 CMSS 服务器成功注册。
8033 INFO_JAVA_SPECIFIC_LOAD_NOK	嵌入式互联服务 (CSE) 无法启用数据收集器。
8123 INFO_EXT_JAVARESET	嵌入式互联服务 (CSE) 接收到来自于外部来源的重置命令。
8210 INFO_STAY_ALIVE **注意** INFO_STAY_ALIVE 是一种定期日志。	嵌入式互联服务(CSE) 以预先规定好的间隔（50 分钟）将保持激活信息发送给 CMSS 服务器，以表明互联服务处于激活状态。激活信息格式如下。 **示例：** message_count Alive Bytes:xx/yy HTTP:p/q/r/s/t/u/v Mem:7069164 Run:746 RC:0 LHE: • message_count：当前激活消息计数。 • Bytes（字节）：控制器已发送和已接收的内存字节的数目。 • HTTP：在控制器与 CMSS 服务器通讯时出现的不同 http 错误。 错误类型： - p：连接错误计数。 - q：连接不可用错误计数。 - r：验证相关错误计数。 - s：请求错误计数。 - t：超时错误计数。 - u：代理错误计数。 - v：未知错误计数。 • Mem：可供使用的闲置内存，单位为字节。 • Run：控制器的正常运行时间，单位为秒。 • RC：自从上次启动之后的 CSE 重启的次数。 • LHE：有关最新 http 错误的信息。
8213 INFO_ALIVE_STARTED	在 CSE 注册之后，嵌入式互联服务 (CSE) 将发送保持激活信息。信息格式与 8210 INFO_STAY_ALIVE相同。

下一页继续

2.3.10.4 嵌入式互联服务故障排除日志

续前页

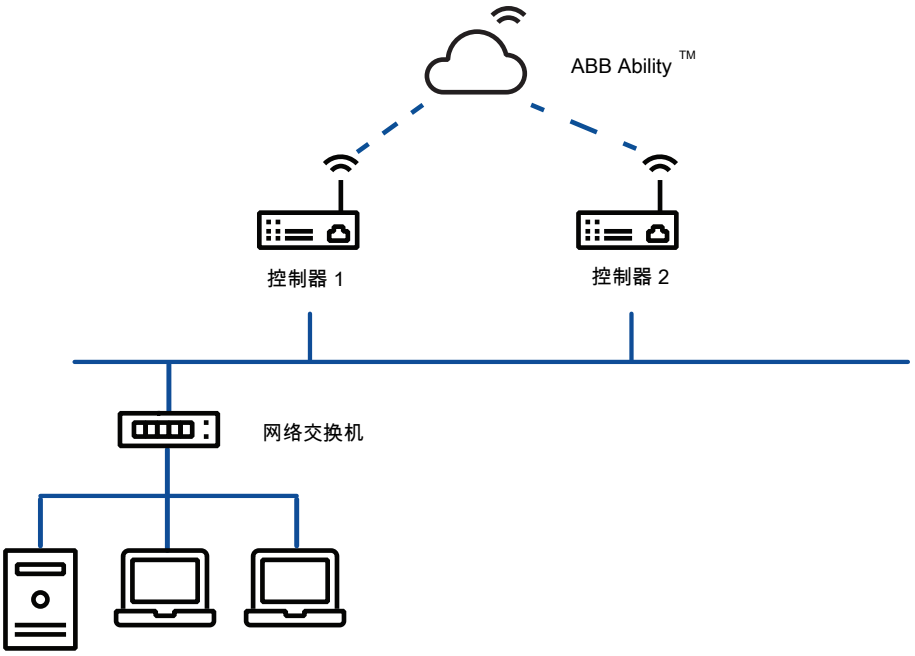
日志编号 日志名称	描述
8214 INFO_ALIVE_ENDED	在嵌入式互联服务停止的时候，嵌入式互联服务 (CSE) 将发送这种激活信息。信息格式与 8210 INFO_STAY_ALIVE相同。
8700 INFO_MODULE_UPDATE_OK	嵌入式互联服务 (CSE) 模块已成功更新。
8701 INFO_MODULE_UPDATE_NOK	嵌入式互联服务 (CSE) 模块未能更新。
8702 INFO_MODULE_UPDATE_ERROR	嵌入式互联服务 (CSE) 模块更新失败。
8801 INFO_S24_STARTED	嵌入式互联服务 (CSE) 数据收集器模块启用。
8803 INFO_S2301_STARTED	嵌入式互联服务 (CSE) Ability 连接器 (S2301) 的连接器已启动。
9003 INFO_SPECIFIC_STOP	嵌入式互联服务 (CSE) 数据收集器停用。

2.3.11 网络拓扑方案

连接类型 – 配备 3G 模块的 Ability

在以下方案中，控制器 1 和控制器 2 将安装并配置有 ABB SIM。有关详细配置，请参阅下图中的“连接设置”表。根据此配置，Connected Services Gateway 3G 模块将连接到网络。

Connected Services 以 Ability 连接类型进行配置，这意味着与 ABB Ability™ Cloud 的所有通信都将通过 Connected Services Gateway 3G 模块传递并路由。



xx1900001182

控制器 1 和 2 的 3G 连接设置	
状态	启用
APN（接入点）	abbbrootics.com
Operator（操作员）	自动
Band（频带）	自动
作者	自动
Roaming（漫游）	启用
Idle（空闲）	0
Delay（延迟）	0

Connected Services 设置	
状态	启用
Connection type（连接类型）	Ability
使用的代理	否
服务器轮询	缓慢或快速

下一页继续

2.3.11 网络拓扑方案

续前页

Connected Services 设置	
Connected Services 模式	2.0 Omnicore [预览] 1.0 IRC5 兼容性

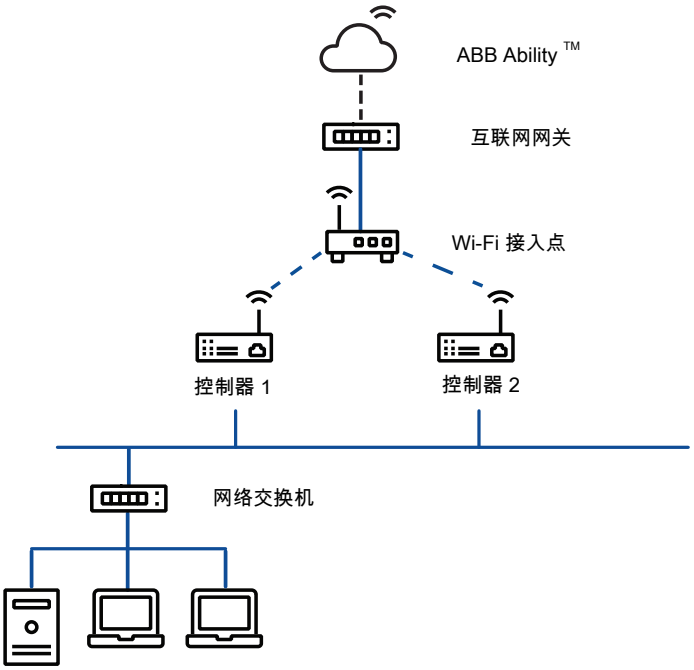
下一页继续

连接类型 – 配备 Wi-Fi 模块的 Ability

在以下方案中，控制器 1 和控制 2 与 Connected Services Gateway Wi-Fi 模块连接。Wi-Fi 模块可配置为连接任何可用的 Wi-Fi 连接点。这些接入点必须通过互联网接入启用后才能访问 ABB Ability™ Cloud。

有关详细配置，请参阅下图中的“Connected Services 设置”表。根据此配置，Connected Services Gateway Wi-Fi 模块将连接到互联网启用的 Wi-Fi 网络并访问 ABB Ability™ Cloud。

Connected Services 以 Ability 连接类型进行配置，这意味着与 ABB Ability™ Cloud 的所有通信都将通过 Connected Services Gateway Wi-Fi 模块传递并路由。



xx1900001183

控制器 1 和 2 的 Wi-Fi 连接设置	
状态	启用
SSID	SSID-123
密钥	1234567890
安全	自动

Connected Services 设置	
状态	启用
Connection type（连接类型）	Ability
使用的代理	是或否
服务器轮询	快速或缓慢
Connected Services 模式	2.0 Omnicore [预览] 1.0 IRC5 兼容性

下一页继续

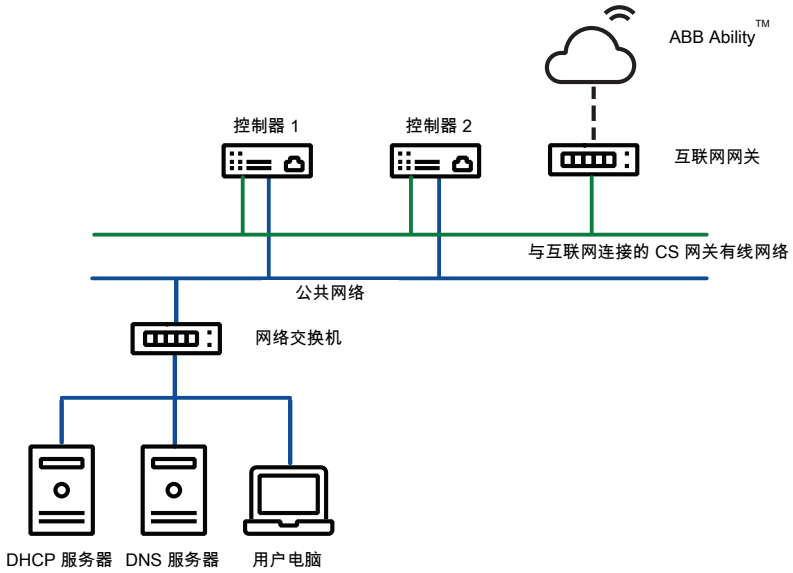
2.3.11 网络拓扑方案
续前页

连接类型 – 配备有线模块的 Ability

在以下方案中，控制器 1 和控制 2 与 Connected Services Gateway Wired 模块连接。由于该模块是有线模块，因此它需要具有互联网访问权限的有线连接才能访问 ABB Ability™ Cloud。控制器 1 和 2 也连接到可以是工厂网络的公共网络。

有线模块应始终以静态 IP 进行配置。有关简单网络配置，请参阅“Connected Services Gateway Wired 设置”表。

Connected Services 以 Ability 连接类型进行配置，这意味着与 ABB Ability™ Cloud 的所有通信都将通过 Connected Services Gateway Wired 模块而非在公共网络上传递并路由。



xx1900001184

有线连接设置	控制器 1 (NW1)	控制器 2
状态	启用	启用
IP 地址	192.168.200.20	192.168.200.21
子网掩码	255.255.255.0	255.255.255.0
默认网关	192.168.200.1	192.168.200.1
首选 DNS	192.168.200.1	192.168.200.1
备用 DNS	0.0.0.0	0.0.0.0

	控制器 1	控制器 2	互联网网关	DHCP 服务器	DNS 服务器	用户电脑
IP 地址	172.16.16.100	172.16.16.101	192.168.200.1	172.16.16.3	172.16.16.2	172.16.16.110
子网掩码	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0
默认网关	172.16.16.1	172.16.16.1		172.16.16.1	172.16.16.1	172.16.16.1

下一页继续

Connected Services 设置	
状态	启用
Connection type（连接类型)	Ability
使用的代理	否
服务器轮询	快
Connected Services 模式	2.0 Omnicore [预览] 1.0 IRC5 兼容性

下一页继续

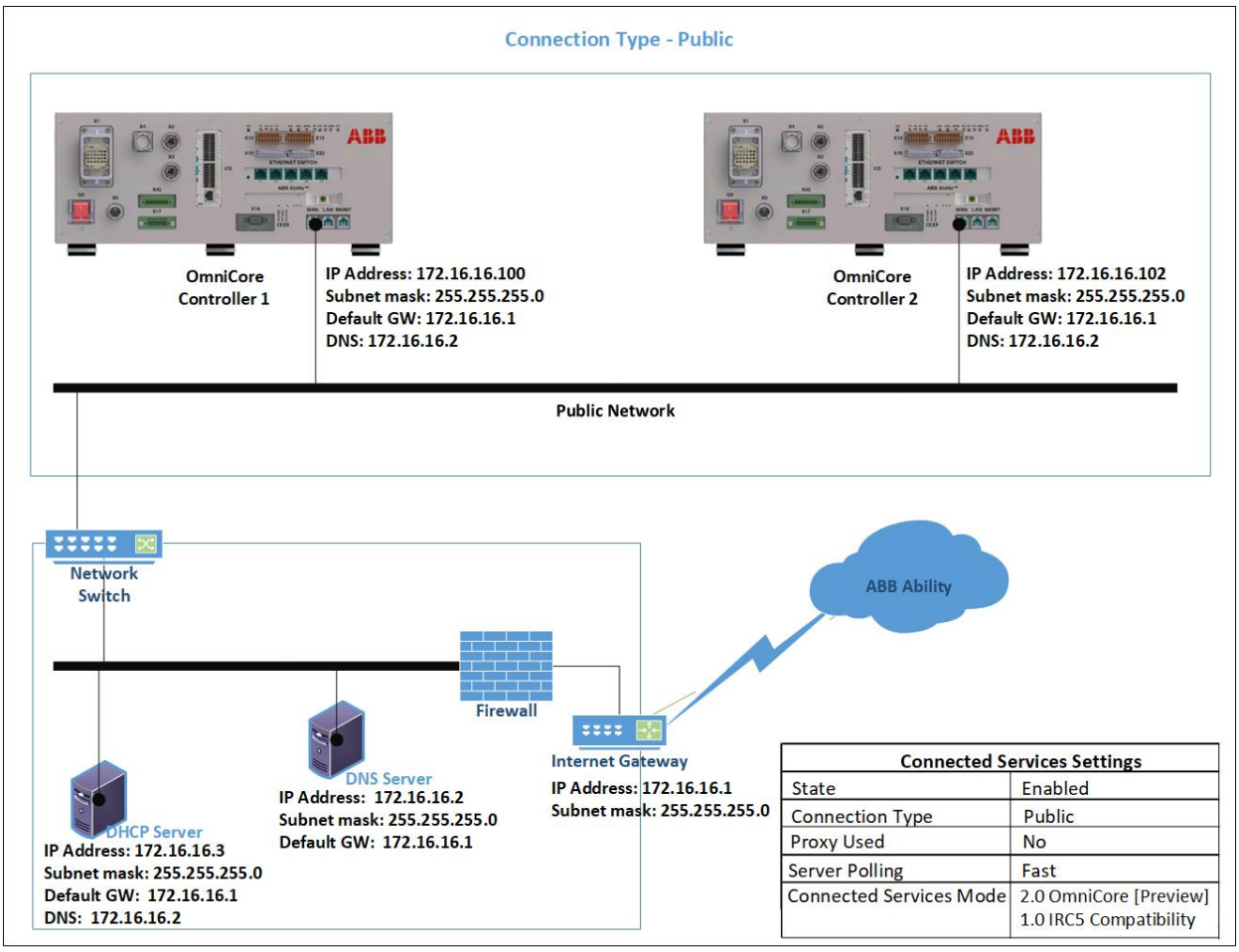
2.3.11 网络拓扑方案
续前页

连接类型 – 公共

如果工厂网络已通过互联网访问权限启用设置了防火墙，则可以使用连接到控制器公共端口的同一网络来配置 Connected Services。

在以下方案中，控制器 1 和控制器 2 通过使用控制器的公共端口连接到公共网络，该端口通过互联网由工厂网络启用。作为一种良好的做法，如果工厂网络连接到互联网，则必须使用防火墙保护工厂网络，以防止任何不需要的入站和出站访问。

公共连接类型的 Connected Services 采用 Connected Services 设置来配置，如下图所示。



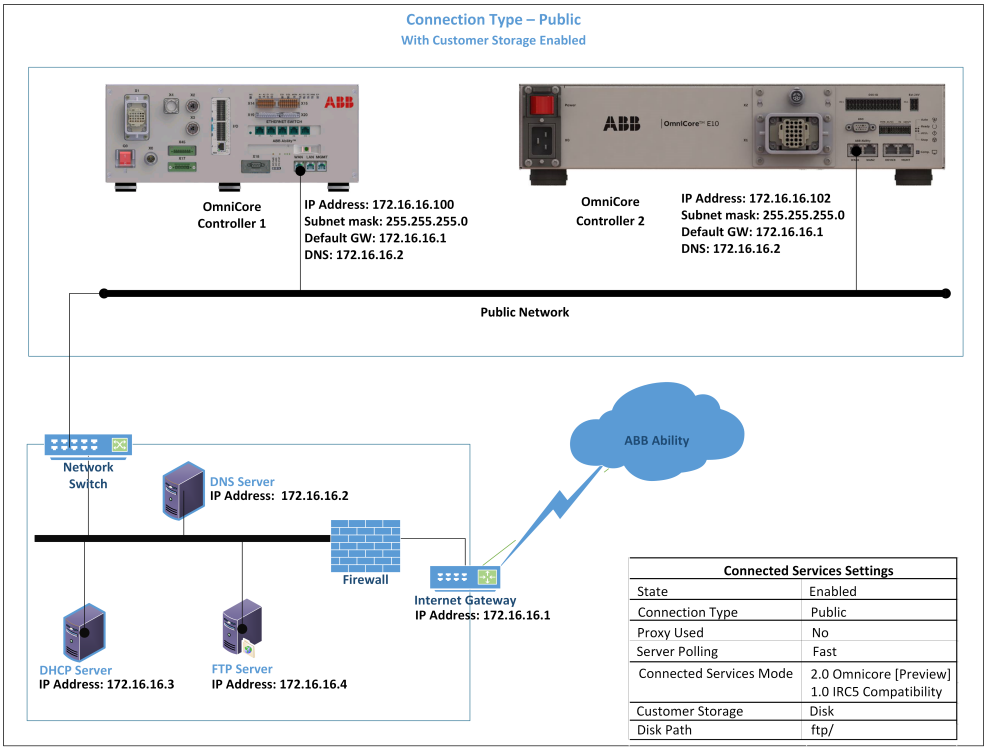
xx1900001185

下一页继续

连接类型 – 公共 - 当客户存储器启用时

下图显示了使用客户存储器来存储控制器数据的方案。客户存储器可以是控制器本地磁盘、映射网络磁盘（在 ftp/sftp 或 nfs 服务器上）。在以下方案中，我们将 ftp/sftp 服务器用作工厂网络中的客户存储器。该 ftp/sftp 服务器可以作为本地磁盘安装在控制器上。因此，在 Connected Services 配置期间，磁盘路径应称为已挂载的 ftp/sftp 磁盘。对于已挂载的 ftp/sftp 磁盘，磁盘路径为 ftp/sftp；而对于控制器上已装置的网络磁盘，磁盘路径为 pc:。

公共连接类型的互联服务采用互联服务设置来配置，如下图所示:



xx1900001188

使用 ABB 网关服务盒的 Connected Services

概述

本节说明了如何使用在控制器中未被定义为默认网关的外部互联网网关（3G/4G、Wi-Fi 等）来配置 Connected Services。在这种情况下，连接服务应通过自定义连接类型来配置。

网关服务盒可以连接在客户 WAN 端口、管理端口或 Connected Services Gateway Wired 端口上。

使用DHCP的控制器

如果存在具有 DHCP 的控制器，则使用以下步骤可从 FlexPendant 配置 Connected Services。

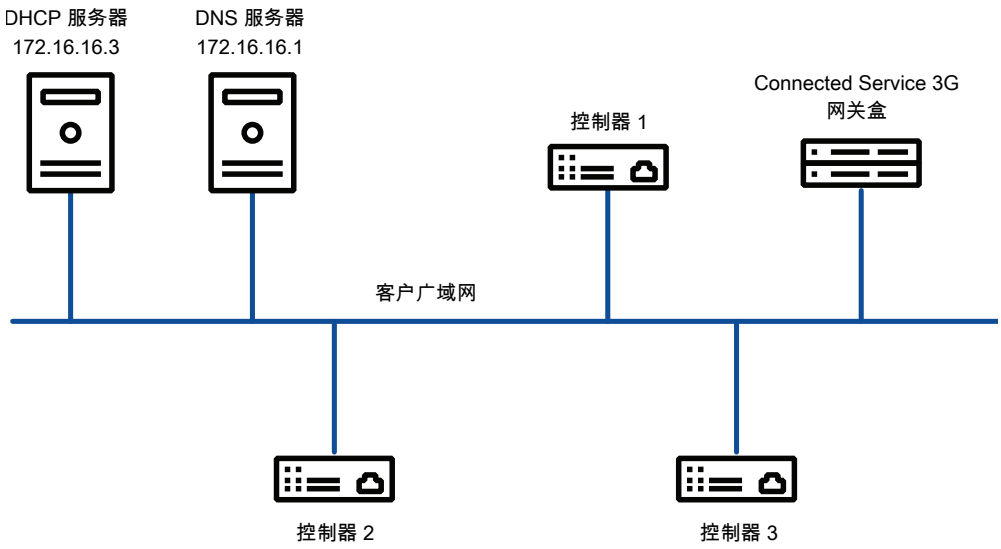
1 打开 **Settings**（设置）。
2 点击 **ABB Ability**。
3 点击左侧窗格上的 **Connected Services**。
连接服务的配置参数随即显示。

下一页继续

4 在 Connection Type（连接类型）列表中，点击并选择 Custom。
5 在 Internet Gateway IP（互联网网关 IP）字段中，键入互联网网关的 IP 地址。
6 在 Internet DNS IP（互联网 DNS IP）字段中，键入互联网 DNS 的 IP 地址。
7 点击 Apply（应用）。
Restart（重新启动）确认消息随即显示。
8 点击 OK（确定）。
控制器随即重新启动。
连接服务将根据配置开始与服务器通信。
检查事件日志中的连通性状态。有关更多详细信息，请参阅 第74页的*Connected Services* 信息。

客户网络上的网关盒

如果为多个控制器配置了网关盒，则必须根据 WAN IP 网段更改网关盒的 LAN IP。
网关盒应连接到客户网络。LAN IP 应修改以匹配客户网络的IP网段。
下图和表格显示了典型的网络基础设施：



xx1800002942

DHCP 配置	控制器 1	控制器 2	控制器 3
IP	172.16.16.58	172.16.16.59	172.16.16.60
掩码	255.255.255.0	255.255.255.0	255.255.255.0
CSE 已启用	是	是	是
Connection type（连接类型）	自定义	自定义	自定义
互联网网关 IP	172.16.16.25	172.16.16.25	172.16.16.25
互联网 DNS IP	172.16.16.25	172.16.16.25	172.16.16.25

下一页继续

有关如何为多个控制器的网关盒进行设置，请参阅产品手册 - 连接服务。

注意

该网络架构是展示网络拓扑的示例。

小心

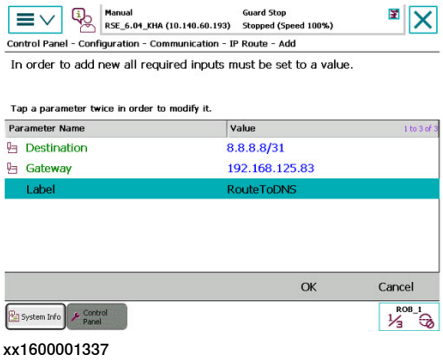
请确保您始终具备受防火墙保护的互联网接入。

注意

使用 ABB 服务盒将可以启用远程访问功能。标配 4G 路由器也可以按照上述原则来使用。

使用DHCP和手动DNS的控制器

下列步骤提供的信息为从FlexPendant配置带有DHCP和手动DNS的控制器的Connected Services。

	操作	图示
1	在ABB菜单中选择控制面板。	
2	选择配置。	
3	从主题，选择通信。	
4	选择IP路由并点击添加。	
5	输入目的地、网关和标签。 • 如果手动输入了DNS IP，请添加该DNS IP的路由。 • 在此例中，**Destination:** 8.8.8.8/31是Google的DNS地址	 xx1600001337
6	点击确定并重启控制器让修改生效。	

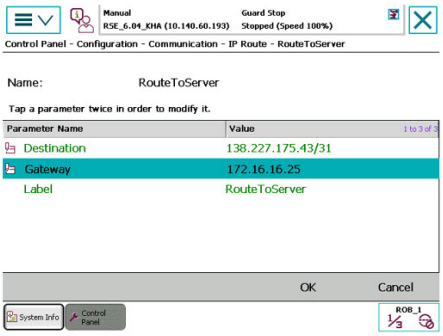
手动配置DNS的步骤

	操作	图示
1	在ABB菜单中选择控制面板。	
2	选择配置。	
3	从主题，选择通信。	
4	选择IP路由并点击添加。	

下一页继续

2.3.11 网络拓扑方案

续前页

	操作	图示
5	输入目的地、网关和标签。 • 输入网关IP作为盒子IP。 在此例中为172.16.16.25。	 xx1600001339
6	点击确定并重启控制器让修改生效。	

注意

若系统未自动提供DNS，则手动定义该DNS。此外也为该DNS IP定义一条经过网关盒的路线。

2.4 Cyclic bool

2.4.1 循环评估逻辑条件

目的

循环评估逻辑条件*Cyclic bool*的目的是允许RAPID程序员能将逻辑条件与永久布尔变量关联起来。逻辑条件每12 ms评估一次，结果将写入关联的变量。

其中包括

RobotWare基础功能*Cyclic bool*包括：

- 设置*Cyclic bool*的指令：`SetupCyclicBool`、`RemoveCyclicBool`、`RemoveAllCyclicBool`
- 获取*Cyclic bool*状态的函数：`GetMaxNumberOfCyclicBool`、`GetNextCyclicBool`、`GetNumberOfCyclicBool`。

基本方法

这是使用*Cyclic bool*的常规方法。有关如何执行的详细示例，请参阅第*103*页的*Cyclic bool*示例。

1 声明一个永久布尔变量，例如：

```
PERS bool cyclicbool1;
```

2 将一个逻辑条件关联到变量，例如：

```
SetupCyclicBool cyclicbool1, doSafetyIsOk = 1;
```

3 编程时使用此变量，例如：

```
WHILE cyclicbool1 = 1 DO
  ! Do what’s only allowed when all safety is ok
  ...
ENDWHILE
```

4 当不再使用时取消关联，例如：

```
RemoveCyclicBool cyclicbool1;
```

重启和重置操作

下表介绍了程序指针移动或控制器重启时的Cyclic bool的工作情况。

操作	描述
程序指针指向`main`	当程序指针设置到`main`时的行为是可配置的，请参阅第*102*页的配置。
重启或电源故障。	这将不会有效果。 所有关联的*Cyclic bool*条件都将保持，评估将立即重新开始。
重置 RAPID	这将会删除所有关联的*Cyclic bool*条件。
重置系统	

下一页继续

2.4.1 循环评估逻辑条件
续前页

配置

Cyclic bool功能的下列行为可以配置：

参数	描述
RemoveAtPpToMain	可以配置在程序指针设置到main时，是否删除循环评估的逻辑条件。 • *On* - 删除。 • *Off* - 不删除（默认行为）。
ErrorMode	可以配置当Cyclic bool的估值失败时要使用的错误模式。 • *SysStopError*[i] - 停止RAPID执行并生成错误日志（默认行为）。 • *Warning* - 生成错误日志。 • *None* - 不执行任何操作。
RecoveryMode	可以配置是否恢复失败的Cyclic bool。 • *On* - 尝试恢复失败的Cyclic bool估值（默认行为）。 • *Off* - 不尝试恢复失败的Cyclic bool估值。

[i] 错误模式*SysStopError*只能配合*RecoveryMode* - *"On"*使用。

有关详细信息，请参阅第*106*页的系统参数。

语法

```
SetupCyclicBool Flag Cond [\Signal]
```

Flag应为：

- 数据类型：bool
  - 对象类型：PERS或TASK PERS

Cond应为一个bool表达式，其中可以包含：

- 数据类型：num、dnum和bool
  - 对象类型：PERS、TASK PERS或CONST
- 数据类型：signaldi、signaldo或物理数字输入(DI)和数字输出(DO)
  - 对象类型：VAR
- 运算符：‘NOT’、‘AND’、‘OR’、‘XOR’、‘=’、‘(’、‘)’

\Signal应为：

- 对象类型：signaldo

```
RemoveCyclicBool Flag
```

Flag应为：

- 数据类型：bool
  - 对象类型：PERS或TASK PERS

限制

- 逻辑条件中不允许使用记录和数组。
- 同时可以关联最多60个条件。
- 在某一条件中使用的任何PERS num或dnum、CONST num或dnum或字面上的num或dnum都必须是整数型。若使用任何小数值，则会造成致命错误。

2.4.2 Cyclic bool示例

使用数字输入和输出信号

```
! Wait until all signals are set
PERS bool cyclicbool1 := FALSE;

PROC main()
  SetupCyclicBool cyclicbool1, di1=1 AND do2=1;
  WaitUntil cyclicbool1=TRUE;
  ! All is ok
  ...
  ! Remove connection when no longer in use
  RemoveCyclicBool cyclicbool1;
ENDPROC
```

使用bool变量

```
! Wait until all flags are TRUE
PERS bool cyclicbool1 := FALSE;
TASK PERS bool flag1 := FALSE;
PERS bool flag2 := FALSE;

PROC main()
  SetupCyclicBool cyclicbool1, flag1=TRUE AND flag2=TRUE;
  WaitUntil cyclicbool1=TRUE;
  ! All is ok
  ...
  ! Remove connection when no longer in use
  RemoveCyclicBool cyclicbool1;
ENDPROC
```

使用num和dnum变量

```
! Wait until all conditions are met
PERS bool cyclicbool1 := FALSE;
PERS bool cyclicbool2 := FALSE;
PERS num num1 := 0;
PERS dnum1 := 0;

PROC main()
  SetupCyclicBool cyclicbool1, num1=7 OR dnum1=10000000;
  SetupCyclicBool cyclicbool2, num1=8 OR dnum1=11000000;
  WaitUntil cyclicbool1=TRUE;
  ...
  WaitUntil cyclicbool2=TRUE;
  ...
  ! Remove all connections when no longer in use
  RemoveAllCyclicBool;
ENDPROC
```

下一页继续

2.4.2 Cyclic bool示例
续前页

使用alias变量

```
! Wait until all conditions are met
ALIAS bool aliasBool;
ALIAS num aliasNum;
ALIAS dnum aliasDnum;

PERS bool cyclicbool1 := FALSE;
PERS aliasBool flag1 := FALSE;
PERS aliasNum num1 := 0;
PERS aliasDnum dnum1 := 0;

PROC main()
  SetupCyclicBool cyclicbool1, flag1=TRUE AND (num1=7 OR
       dnum1=10000000);
  WaitUntil cyclicbool1=TRUE;
  ! All is ok
  ...
  ! Remove connection when no longer in use
  RemoveCyclicBool cyclicbool1;
ENDPROC
```

使用用户定义的常量进行比较

```
! Wait until all conditions are met
PERS bool cyclicbool1;
PERS bool flag1 := FALSE;
PERS num num1 := 0;
PERS dnum dnum1 := 0;
CONST bool MYTRUE := TRUE;
CONST num NUMLIMIT := 10;
CONST dnum DNUMLIMIT := 10000000;

PROC main()
  SetupCyclicBool cyclicbool1, flag1=MYTRUE AND num1=NUMLIMIT AND
       dnum1=DNUMLIMIT;
  WaitUntil cyclicbool1=TRUE;
  ! All is ok
  ...
  ! Remove connection when no longer in use
  RemoveCyclicBool cyclicbool1;
ENDPROC
```

下一页继续

通过引用传递参数

如果在一个调用的无返回值程序中使用了指令SetupCyclicBool，则有可能将条件作为该无返回值程序的参数传递。

使用通过引用传递的条件仅对SetupCyclicBool有效。通过引用传递的条件与SetupCyclicBool的条件有相同限制。.

无论模块是Nostepin或有任何其他模块属性，此功能均有效。

```
MODULE MainModule
CONST robtarget p10 := [[600,500,225.3], [1,0,0,0], [1,1,0,0],
     [11,12.3,9E9,9E9,9E9,9E9]];
PERS bool m1;
PERS bool Flag2 := FALSE;

PROC main()
  ! The Expression (di_1 = 1) OR Flag2 = TRUE shall be
  ! used by SetupCyclicBool
  my_routine (di_1 = 1) OR Flag2 = TRUE;
ENDPROC

PROC my_routine(bool X)
  ! It is possible to pass arguments between several procedures
  MySetCyclicBool X;
ENDPROC

PROC MySetCyclicBool (bool Y)
  RemoveCyclicBool m1;
  ! Only SetupCyclicBool can pass arguments
  SetupCyclicBool m1, Y;
  ! If conditions passed by reference shall be used by any other
  ! instruction, the condition must be setup with SetupCyclicBool
  ! before it can be used.
  WaitUntil m1;
  MoveL p10, v1000, z30, tool2;
ENDPROC
ENDMODULE
```

2.4.3 系统参数

关于系统参数

此处简述了*Cyclic bool*所用的系统参数。关于这些参数的更多信息请参见技术参考手册 - 系统参数。

类型*Cyclic bool settings*

*Cyclic bool*使用的系统参数水域类型*Cyclic bool settings*，主题*Controller*。

参数	描述
Name	每个允许值只能有一个实例，也就是系统中最多只能有三个实例。所有三个实例都将（默认）安装在系统中且不能删除。 • *RemoveAtPpToMain* • *ErrorMode* • *RecoveryMode*
RemoveAtPpToMain	操作值*RemoveAtPpToMain*用于配置在程序指针设置到Main时是否删除关联的Cyclic bool。
ErrorMode	操作值*ErrorMode*用于配置在估值失败时使用哪个错误模式。
RecoveryMode	操作值*RecoveryMode*用于配置在估值失败时使用哪个恢复模式。

2.4.4 RAPID组件

关于RAPID组件

这是在*Cyclic bool*中所有RAPID指令、函数与数据类型的概述。

有关详细信息，请参见技术参考手册 - *RAPID*指令、函数和数据类型

指令：

指令	描述
SetupCyclicBool	SetupCyclicBool关联一个逻辑条件到一个布尔变量。
RemoveCyclicBool	RemoveCyclicBool去除一个关联的逻辑条件。
RemoveAllCyclicBool	RemoveAllCyclicBool去除全部关联的逻辑条件。

函数

功能	描述
GetMaxNumberOfCyclicBool	GetMaxNumberOfCyclicBool获取可以同时关联的循环评估逻辑条件的最大数量。
GetNextCyclicBool	GetNextCyclicBool获取一个关联的循环评估逻辑条件的名称。
GetNumberOfCyclicBool	GetNumberOfCyclicBool获取一个关联的循环评估逻辑条件的编号。
IsCyclicBool	IsCyclicBool用于测试某个持续布尔变量是否为Cyclic bool，即是否用指令SetupCyclicBool将一个逻辑条件关联到了该持续布尔变量。

数据类型

*Cyclic bool*不包含数据类型。

2.5 Device Command Interface

2.5.1 Device Command Interface介绍

目的

Device Command Interface提供了一个与工业网络上的I / O装置进行通信的接口。请配合原始数据通信来使用该接口，具体请参见第*117*页的原始数据通信。

其中包括

您可通过RobotWare基本功能Device Command Interface来访问：

- 用于创建一个DeviceNet标题的指令。

基本方法

这是Device Command Interface的一般用法。第*110*页的在*DeviceNet*中写入*rawbytes*用一个更详细的示例展示了其具体用法。

1. 在一个rawbytes变量中添加一个DeviceNet标题。
2. 在rawbytes变量中添加相关数据。
3. 向eviceNet I / O写入rawbytes变量。
4. 从DeviceNet I / O的数据中读取一个rawbytes变量。
5. 从rawbytes变量中提取相关数据。

限制

装置命令通信需要所论工业网络的选项。

以下类型的工业网络支持Device Command Interface：

- DeviceNet
- EtherNet/IP

2.5.2 RAPID部件和系统参数

数据类型

没有针对Device Command Interface的RAPID数据类型。

指令：

此处简述了Device Command Interface中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
PackDNHeader	PackDNHeader在一个rawbytes变量中添加了一个DeviceNet标题，而该标题则指定了将开展的一种服务（如设置或获取）和DeviceNetI / O装置上的一个参数。

函数

没有Device Command Interface所需的RAPID函数。

系统参数

没有Device Command Interface中的特定系统参数。至于系统函数方面的一般信息，则请参见技术参考手册 - 系统参数。

2.5.3 代码示例

在DeviceNet中写入rawbytes

此例把已打包的数据作为一个rawbytes变量写入了一部DeviceNetI / O装置。
rawbytes方面的更多信息请参见第117页的原始数据通信。

```
PROC set_filter_value()
  VAR iodev dev;
  VAR rawbytes rawdata_out;
  VAR rawbytes rawdata_in;
  VAR num input_int;
  VAR byte return_status;
  VAR byte return_info;
  VAR byte return_errcode;
  VAR byte return_errcode2;

  ! Empty contents of rawdata_out and rawdata_in
  ClearRawBytes rawdata_out;
  ClearRawBytes rawdata_in;

  ! Add DeviceNet header to rawdata_out with service
  ! "SET_ATTRIBUTE_SINGLE" and path to filter attribute on
  ! DeviceNet I/O device
  PackDNHeader "10", "6,20 1D 24 01 30 64,8,1", rawdata_out;

  ! Add filter value to send to DeviceNet I/O device
  input_int:= 5;
  PackRawBytes input_int, rawdata_out,(RawBytesLen(rawdata_out) +
       1) \IntX := USINT;

  ! Open I/O device
  Open "/FCI1:" \File:="board328", dev \Bin;

  ! Write the contents of rawdata_out to the I/O device
  WriteRawBytes dev, rawdata_out \NoOfBytes :=
       RawBytesLen(rawdata_out);

  ! Read the answer from the I/O device
  ReadRawBytes dev, rawdata_in;

  ! Close the I/O device
  Close dev;

  ! Unpack rawdata_in to the variable return_status
  UnpackRawBytes rawdata_in, 1, return_status \Hex1;

  IF return_status = 144 THEN
    TPWrite "Status OK from device. Status code:
         "\Num:=return_status;
  ELSE
    ! Unpack error codes from device answer
```

下一页继续

```
    UnpackRawBytes rawdata_in, 2, return_errcode \Hex1;
    UnpackRawBytes rawdata_in, 3, return_errcode2 \Hex1;
    TPWrite "Error code from device: " \Num:=return_errcode;
    TPWrite "Additional error code from device: "
         \Num:=return_errcode2;
  ENDIF
ENDPROC
```

2.6 File and I/O device handling

2.6.1 文件和 I/O 设备处理介绍

关于文件和 I/O 设备处理

RobotWare 文件和 I/O 设备处理使机器人程序员可以通过 RAPID 代码来管控文件和现场总线。例如，这可在以下方面发挥作用：

- 用条码读取器读取。
- 将生产统计写入一份日志文件或写入一台打印机。
- 在机器人与一台个人电脑之间传输数据。

可将文件和 I/O 设备处理中的这一功能划分为若干组：

功能组	描述
基于二进制和字符的通信	基本通信功能。采用基于二进制或字符的文件或I/O设备的通信。
原始数据通信	包装在容器中的数据。尤其适用于现场总线通信。
文件与目录管理	浏览并编辑文件结构。

2.6.2 基于二进制和字符的通信

2.6.2.1 概述

目的

基于二进制和字符的通信的作用是：

- 将信息保存在远程内存或远程硬盘中
- 让机器人与其它装置进行通信

其中包括

为了处理基于二进制和字符的通信，您可通过 RobotWare 来访问：

- 操纵一份文件或一个 I/O 设备的指令
- 写入一份文件或一个 I/O 设备的指令
- 读取一份文件或一个 I/O 设备的指令
- 用于读取文件或 I/O 设备的函数。

基本方法

这是基于二进制和字符的通信的一般用法。第*115*页的代码示例用一个更详细的示例展示了其具体用法。

1 打开文件或 I/O 设备。
2 读取或写入相应的文件或 I/O 设备。
3 关闭文件或 I/O 设备。

限制

不同的 RAPID 任务不能同时访问各文件和 I/O 设备。得通过基于二进制和字符的通信中的所有指令以及`WriteRawBytes`和`ReadRawBytes`来进行此类访问。举例来说，如果要在一项任务中执行一条`ReadBin`指令，那么就必须在另一项任务中可以执行`WriteRawBytes`前作好准备。

2.6.2.2 RAPID组件

数据类型

此处简述了基于二进制和字符的通信中所用的每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各种数据类型。

数据类型	描述
iodev	iodev包含了一份文件或一个 I/O 设备的引用项。可通过指令Open将其与相关物理单元链接起来，然后用其进行读取和写入。

指令：

此处简述了基于二进制和字符的通信中使用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
Open	Open的作用是打开一份文件或一个 I/O 设备来进行读取或写入。
Close	Close的作用是关闭一份文件或一个 I/O 设备。
Rewind	Rewind将该文件的位置设置在文件开头。
Write	Write的作用是写入一份基于字符的文件或一个 I/O 设备。
WriteBin	WriteBin的作用是在一个二进制 I/O 设备或一份文件中写入大量字节。
WriteStrBin	WriteStrBin的作用是在一个二进制 I/O 设备或一份文件中写入一段字符串。
WriteAnyBin	WriteAnyBin的作用是在一个二进制 I/O 设备或一份文件中写入任何类型数据。
ReadAnyBin	ReadAnyBin的作用是通过一个二进制 I/O 设备或一份文件读取任何类型数据。

函数

此处简述了基于二进制和字符的通信中使用的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
ReadNum	ReadNum的作用是从一份基于字符的文件或一个 I/O 设备中读取一个数字。
ReadStr	ReadStr的作用是从一份基于字符的文件或一个 I/O 设备中读取一段字符串。
ReadBin	ReadBin的作用是从一份文件或一个 I/O 设备上读取一个字节（8 位）。该函数可作用于基于二进制和字符两者的文件或 I/O 设备。
ReadStrBin	ReadStrBin的作用是从一个二进制 I/O 设备或一份文件中读取一段字符串。

2.6.2.3 代码示例

用基于字符的文件进行通信

此例展示了如何读取和写入一份基于字符的文件。在 FILE1.DOC 中写入行“The number is :8”，然后读取 FILE1.DOC 的内容。先向 FlexPendant 示教器输出“The number is :8”，然后是“The number is 8”。

```
PROC write_to_file()
  VAR iodev file;
  VAR num number:= 8;
  Open "HOME:" \File:= "FILE1.DOC", file;
  Write file, "The number is :"\Num:=number;
  Close file;
ENDPROC

PROC read_from_file()
  VAR iodev file;
  VAR num number;
  VAR string text;

  Open "HOME:" \File:= "FILE1.DOC", file \Read;
  TPWrite ReadStr(file);
  Rewind file;
  text := ReadStr(file\Delim:=":");
  number := ReadNum(file);
  Close file;
  TPWrite text \Num:=number;
ENDPROC
```

采用二进制文件的通信

此例把字符串“Hello”、机器人当前位置和字符串“Hi”写入二进制文件。

```
PROC write_bin_chan()
  VAR iodev file1;
  VAR num out_buffer{20};
  VAR num input;
  VAR robtarget target;

  Open "HOME:" \File:= "FILE1.DOC", file1 \Bin;

  ! Write control character enq
  out_buffer{1} := 5;
  WriteBin file1, out_buffer, 1;

  ! Wait for control character ack
  input := ReadBin (file1 \Time:= 0.1);
  IF input = 6 THEN
    ! Write "Hello" followed by new line
    WriteStrBin file1, "Hello\0A";

    ! Write current robot position
```

下一页继续

2.6.2.3 代码示例

续前页

```
    target := CRobT(\Tool:= tool1\WObj:= wobj1);
    WriteAnyBin file1, target;

    ! Set start text character (2=start text)
    out_buffer{1} := 2;

    ! Set character "H" (72="H")
    out_buffer{2} := 72;

    ! Set character "i"
    out_buffer{3} := StrToByte("i"\Char);

    ! Set new line character (10=new line)
    out_buffer{4} := 10;

    ! Set end text character (3=end text)
    out_buffer{5} := 3;

    ! Write the buffer with the line "Hi"
    ! to the file
    WriteBin file1, out_buffer, 5;
  ENDIF
  Close file1;
ENDPROC
```

2.6.3 原始数据通信

2.6.3.1 概述

目的

原始数据通信的作用是将数据的不同类型打包到一个容器中，然后发送至一份文件或一个 I/O 设备，最后进行数据读取和解压。这在应用现场总线（如 DeviceNet）进行通信时尤为有用。

其中包括

为了处理原始数据通信，您可通过 RobotWare 来访问：

- 处理一个rawbytes变量之内容的指令
- 用于读取和写入原始数据的指令
- 用于获取一个rawbytes变量的有效数据长度的函数。

基本方法

这是原始数据通信的一般用法。第119页的写入和读取rawbytes用一个更详细的示例展示了其具体用法。

1. 将数据打包成一个rawbytes变量（num、byte或string类型的数据）。
2. 将rawbytes变量写入一份文件或一个 I/O 设备。
3. 从一份文件或一个 I/O 设备中读取一个rawbytes变量。
4. 将rawbytes变量解压成num、byte或string。

限制

装置命令通信也需要基本功能Device Command Interface和所论工业网络的选项。

不同的 RAPID 任务不能同时访问各文件和 I/O 设备。得通过基于二进制和字符的通信中所有指令以及WriteRawBytes和ReadRawBytes来进行此类访问。举例来说，如果要在一项任务中执行一条ReadBin指令，那么就必须在另一项任务中能执行WriteRawBytes指令前作好准备。

2.6.3.2 RAPID组件

数据类型

此处简述了原始数据通信所用的每种数据类型。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各种数据类型。

数据类型	描述
rawbytes	rawbytes 用作为一个通用数据容器，其可装入 num、byte 或是 string 类型的任何数据，还能保存有效数据的长度（以字节计）。 rawbytes 可以包含最多 1024 字节的数据。所支持的数据格式列在说明书 PackRawBytes之中，该说明书在 技术参考手册 - *RAPID*指令、函数和数据类型里。

指令：

此处简述了原始数据通信所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
ClearRawBytes	ClearRawBytes的作用是将一个rawbytes变量的所有内容均设置成0。rawbytes变量中的有效数据长度会被设置成0。 也可用ClearRawBytes来仅仅清除一个rawbytes变量的最终部分。
PackRawBytes	PackRawBytes的作用是把类型为num、byte或string的变量的内容打包成一个类型为rawbytes的变量。
UnpackRawBytes	UnpackRawBytes的作用是把类型为rawbytes的变量的内容解压成类型为byte、num或string的变量。
CopyRawBytes	CopyRawBytes的作用是是把一个rawbytes变量的全部或部分内容复制到另一个同类变量中。
WriteRawBytes	WriteRawBytes的作用是在任何二进制文件或 I/O 设备中写入rawbytes类型的数据。
ReadRawBytes	ReadRawBytes的作用是从任何二进制文件或 I/O 设备中读取rawbytes类型的数据。

函数

此处简述了原始数据通信所用的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
RawBytesLen	RawBytesLen的作用是获取一个rawbytes变量中的有效数据长度。

2.6.3.3 代码示例

关于示例

这些示例简要示范了如何使用rawbytes。若想通过更真实的示例来了解如何使用DeviceNet通信中的rawbytes，则请参见第110页的在DeviceNet中写入rawbytes。

写入和读取rawbytes

此例展示了如何将数据打包成一个rawbytes变量以及如何将其写入一部装置中。此外还展示了如何读取和解压一个rawbytes变量。

```
VAR iodev io_device;
VAR rawbytes raw_data;

PROC write_rawbytes()
  VAR num length := 0.2;
  VAR string length_unit := "meters";

  ! Empty contents of raw_data
  ClearRawBytes raw_data;

  ! Add contents of length as a 4 byte float
  PackRawBytes length, raw_data,(RawBytesLen(raw_data)+1) \Float4;

  ! Add the string length_unit
  PackRawBytes length_unit, raw_data,(RawBytesLen(raw_data)+1)
       \ISOLatin1Encoding;

  Open "HOME:" \File:= "FILE1.DOC", io_device \Bin;

  ! Write the contents of raw_data to io_device
  WriteRawBytes io_device, raw_data;

  Close io_device;
ENDPROC

PROC read_rawbytes()
  VAR string answer;

  ! Empty contents of raw_data
  ClearRawBytes raw_data;

  Open "HOME:" \File:= "FILE1.DOC", io_device \Bin;

  ! Read from io_device into raw_data
  ReadRawBytes io_device, raw_data \Time:=1;

  Close io_device;

  ! Unpack raw_data to the string answer
  UnpackRawBytes raw_data, 1, answer \ISOLatin1Encoding:=10;
```

下一页继续

2.6.3.3 代码示例
续前页

```
ENDPROC
```

复制rawbytes

在此例中，raw_data_1和raw_data_2的所有数据都被复制到了raw_data_3。

```
VAR rawbytes raw_data_1;
VAR rawbytes raw_data_2;
VAR rawbytes raw_data_3;
VAR num my_length:=0.2;
VAR string my_unit:=" meters";

PackRawBytes my_length, raw_data_1, 1 \Float4;
PackRawBytes my_unit, raw_data_2, 1 \ISOLatin1Encoding;

! Copy all data from raw_data_1 to raw_data_3
CopyRawBytes raw_data_1, 1, raw_data_3, 1;

! Append all data from raw_data_2 to raw_data_3
CopyRawBytes raw_data_2, 1, raw_data_3,(RawBytesLen(raw_data_3)+1);
```

2.6.4 文件与目录管理

2.6.4.1 概述

目的

文件与目录管理的目的是便于浏览和编辑文件结构（目录和文件）。

其中包括

为了处理文件与目录管理事项，您可通过 RobotWare 来访问：

- 处理目录的指令
- 用于读取目录的一则函数
- 在某文件结构等级上处理文件的指令
- 检索大小信息和类型信息的函数。

基本方法

这是文件与目录管理的一般方式。第123页的代码示例用一个更详细的示例展示了其具体做法。

1 打开一个目录。
2 从该目录中进行读取，并一直搜索到发现您的搜索对象为止。
3 关闭该目录。

2.6.4.2 RAPID组件

数据类型

此处简述了文件与目录管理所用的每种数据类型。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各种数据类型。

数据类型	描述
dir	dir包含了硬盘或网络上的某目录的引用项。可用指令OpenDir将其与物理目录链接起来。

指令：

此处简述了文件与目录管理所用的每条指令。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各条指令。

指令	描述
OpenDir	OpenDir的作用是打开一个目录。
CloseDir	CloseDir的作用是关闭一个目录。
MakeDir	MakeDir的作用是创建一个新目录。
RemoveDir	RemoveDir的作用是移除一个空目录。
CopyFile	CopyFile的作用是复制一份现有文件。
RenameFile	RenameFile的作用是为一份现有文件提供一个新的名称，此外也可在该目录结构下将一份文件从一处移到另一处。
RemoveFile	RemoveFile的作用是移除一份文件。

函数

此处简述了文件与目录管理所用的每则函数。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各则函数。

功能	描述
ReadDir	ReadDir的作用，是在已用指令OpenDir打开的一个目录中检索下一份文件或子目录的名称。 请注意，ReadDir读取的第一批条目是.（全停符）..（双重全停符），即当前目录及其上层目录的符号表达。
FileSize	FileSize的作用是检索指定文件的大小（以字节计）。
FSSize	FSSize（文件系统大小）的作用是检索指定文件所在文件系统的大小（以字节计）。FSSize可检索该系统的总大小或自由大小。
IsFile	IsFile的作用是测试指定文件的类型是否为指定类型，同时也可用于测试相关文件是否存在。

2.6.4.3 代码示例

列出文件

此例展示了如何列出一个目录中的相关文件，但该目录本身及其上层目录不在此列（.和..）。

```
PROC lsdir(string dirname)
  VAR dir directory;
  VAR string filename;

  ! Check that dirname really is a directory
  IF IsFile(dirname \Directory) THEN
    ! Open the directory
    OpenDir directory, dirname;

    ! Loop though the files in the directory
    WHILE ReadDir(directory, filename) DO
      IF (filename <> "." AND filename <> ".." THEN
        TPWrite filename;
      ENDIF
    ENDWHILE

    ! Close the directory
    CloseDir directory;
  ENDIF
ENDPROC
```

将文件移至新目录

此例创建了一个新文件，重命名了一份文件并将其移至新目录，此外还移除了旧目录。

```
VAR dir directory;
VAR string filename;

! Create the directory newdir
MakeDir "HOME:/newdir";

! Rename and move the file
RenameFile "HOME:/olddir/myfile", "HOME:/newdir/yourfile";

! Remove all files in olddir
OpenDir directory, "HOME:/olddir";
WHILE ReadDir(directory, filename) DO
  IF (filename <> "." AND filename <> ".." THEN
    RemoveFile "HOME:/olddir/" + filename;
  ENDIF
ENDWHILE
CloseDir directory;

! Remove the directory olddir (which must be empty)
RemoveDir "HOME:/olddir";
```

下一页继续

2.6.4.3 代码示例
续前页

检查大小

此例将相关文件的大小与文件系统中的剩余自由空间作了对比。若还有足够空间，系统便会复制该文件。

```
VAR num freefsyssize;
VAR num f_size;

! Get the size of the file
f_size := FileSize("HOME:/myfile");

! Get the free size on the file system
freefsyssize := FSSize("HOME:/myfile" \Free);

! Copy file if enough space free
IF f_size < freefsyssize THEN
  CopyFile "HOME:/myfile", "HOME:/yourfile";
ENDIF
```

2.7 Fixed Position Events

2.7.1 概述

目的

Fixed Position Events的作用是是确保在明确定义TCP位置的情况下执行一项程序例程。

如果用设置成fine的区域自变数来调用一条移动指令，那么一旦TCP抵达其目标点，就会执行下一项例程。如果用设置成距离（如z20）的区域自变数来调用一条移动指令，那么在TCP还未靠近目标点前就可能会执行下一项例程。出现这种差别的原因是执行RAPID指令和执行机器人移动之间存在延时。

若用设置成fine的区域来调用移动指令，则会减慢移动速度。而若Fixed Position Events，那么只要TCP位于TCP路径上的任一指定位置，便能在不减慢移动速度的情况下执行一则例程。

其中包括

您可通过RobotWare基本功能Fixed Position Events来访问：

- 用于定义一起位置事件的指令
- 移动机器人并同时执行位置事件的指令
- 在未首先定义位置事件前移动机器人并在通过目标点时调用某则无返回值程序的指令

基本方法

即可通过一则调用某则无返回值程序的简化指令来使用Fixed Position Events，也可按下列通用步骤来设置此类事件。至于更为详细的设置示例，则请参见第*128*页的代码示例。

1 声明该位置事件。
2 定义该位置事件：
   - 应发生之时（与目标点位置进行对比）
   - 应做之事
3 调用一条使用位置事件的移动指令。当TCP尽可能靠近所定义的目标点时，便会发生这一事件。

2.7.2 RAPID组件与系统参数

数据类型

此处简述了Fixed Position Events中的每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各种数据类型。

数据类型	描述
triggdata	triggdata的作用是保存一起位置事件的数据。 一起位置事件的具体形式既可以是设置一个输出信号，也可以是在机器人移动路径上的某特定位置处运行一则中断例程。 triggdata也包含何时应出现行动的信息，比如TCP何时处在已定义的距离（与目标点之间）上。 triggdata是一种非数值的数据类型。
triggios	triggios的作用是保存指令TriggLIOs所用位置事件的数据。 triggios会用一个num值来设置一个输出信号的值。
triggiosdnum	triggiosdnum的作用是保存指令TriggLIOs所用位置事件的数据。 triggiosdnum会用一个dnum值来设置一个输出信号的值。
triggstrgo	triggstrgo的作用是保存指令TriggLIOs所用位置事件的数据。 triggstrgo会用一个stringdig值（含有一个数字的字符串）来设置一个输出信号的值。

指令：

此处简述了Fixed Position Events中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
TriggIO	TriggIO定义了一个输出信号的设置以及设置该信号的时间。其定义被保存在类型triggdata的一个变量中。 TriggIO不但能把定义“在距离目标点特定距离（以毫米为单位）处或离开目标点特定时间后发出相关信号”，还能把信号设置在已定义的距离或时间点上（指与起始位置之间的距离或时间）。 若将距离设置成0（零），那么信号将被设置在TCP尽可能靠近目标点（拐角路径中间）之时。
TriggEquip	TriggEquip的作用类似于TriggIO，差别在于TriggEquip无法补偿外部设备的内部延时。 举例来说，涂胶枪的信号就必须设置一个极短的时间，且该时间必须在压出胶水并开始胶合之前。
TriggInt	TriggInt定义了何时运行一则中断例程。其定义被保存在类型triggdata的一个变量中。 TriggInt定义了应在与距离目标点（或起始位置）多远处（以毫米为单位）调用中断例程。若将爱那个距离设置成0（零），那么将在TCO尽可能靠近目标点（拐角路径中间）时发生中断。
TriggCheckIO	TriggCheckIO为一个输入或输出信号定义了一项测试以及测试时间。其定义被保存在类型triggdata的一个变量中。 TriggCheckIO定义了一项测试，以便将一个输入或输出信号与某一数值进行对比。调用了一则中断例程。作为一种选项，用户可以在发生中断时停止机器人的移动。 TriggCheckIO不但能把定义“在距离目标点特定距离（以毫米为单位）处或离开目标点特定时间后进行测试”，还能在已定义的距离或时间点上（指与起始位置之间的距离或时间）进行测试。 若将距离设置成0（零），那么将在TCP尽可能靠近目标点（拐角路径中间）时调用中断例程。

下一页继续

指令	描述
TriggRampAO	TriggRampAO定义了一个模拟输出信号的增减率以及增减时间。其定义被保存在类型triggdata的一个变量中。 TriggRampIO定义了何时开始信号增减以及增减长度。
TriggL	TriggL是一条类似于MoveL的移动指令。除移动外，TriggL指令还能设置输出信号、运行中断例程以及在固定位置检查输入或输出信号。 TriggL最多会执行8起保存为triggdata的位置事件。必须在调用TriggL前就对此进行定义。
TriggC	TriggC是一条类似于MoveC的移动指令。除移动外，TriggC指令还能设置输出信号、运行中断例程以及在固定位置检查输入或输出信号。 TriggC最多会执行8起保存为triggdata的位置事件。必须在调用TriggC前就对此进行定义。
TriggJ	TriggJ是一条类似于MoveJ的移动指令。除移动外，TriggJ指令还能设置输出信号、运行中断例程以及在固定位置检查输入或输出信号。 TriggJ最多会执行8起保存为triggdata的位置事件。必须在调用TriggJ前就对此进行定义。
TriggLIOs	TriggLIOs是一条类似于MoveL的移动指令。除移动外，TriggLIOs指令还能在固定位置设置输出信号。 TriggLIOs形同TriggEquip与TriggL的组合，差别在于TriggLIOs最多能处理50起保存为数组（数据类型为triggios、triggiosdnum或triggstrgo）的位置事件。
MoveLSync	MoveLSync是一条直线移动指令，其作用是在拐角路径中间处调用一则无返回值程序。
MoveCSync	MoveCSync是一条环形移动指令，其作用是在拐角路径中间处调用一则无返回值程序。
MoveJSync	MoveJSync是一条联合移动指令，其作用是在拐角路径中间处调用一则无返回值程序。

函数

Fixed Position Events中不包括RAPID函数。

系统参数

此处简述了Fixed Position Events中的每个参数。更多信息请参见技术参考手册 - 系统参数中的各个参数。

参数	描述
Event Preset Time	TriggEquip利用了RAPID执行与机器人移动之间的延时（约70毫秒）。如果设备延时超过70毫秒，那么就以配置*Event preset time*的方式来延长机器人移动的延时。 *Event preset time*属于主题*Motion*下的类型*Motion System*。

2.7.3 代码示例

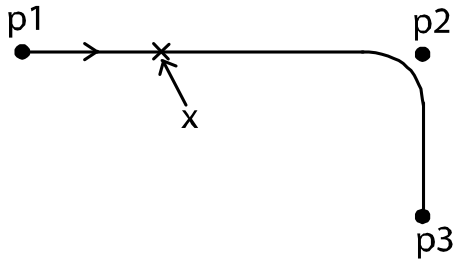
无Fixed Position Events的示例

在不使用Fixed Position Events的情况下，该代码可表达成：

```
MoveJ p1, vmax, fine, tool1;
MoveL p2, v1000, z20, tool1;
SetDO do1, 1;
MoveL p3, v1000, fine, tool1;
```

结果

该代码规定了TCP宜在设置do1前先抵达p2。由于机器人路径与执行指令之间存在延时，因此当TCP位于用X标记的位置时（参见插图），系统会设置do1。



xx0300000151

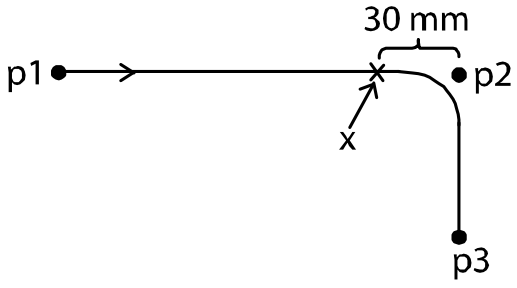
涉及TriggIO和TriggL指令的示例

可通过先定义位置事件、然后在系统执行该位置事件时移动机器人的方式来设置距离目标点30毫米处的输出信号。

```
VAR triggdata do_set;
!Define that do1 shall be set when 30 mm from target
TriggIO do_set, 30 \DOp:=do1, 1;
MoveJ p1, vmax, fine, tool1;
!Move to p2 and let system execute do_set
TriggL p2, v1000, do_set, z20, tool1;
MoveL p3, v1000, fine, tool1;
```

结果

将在TCP距离p230毫米处时设置信号do1。当TCP位于用X标记的位置时，系统会设置do1。



xx0300000158

下一页继续

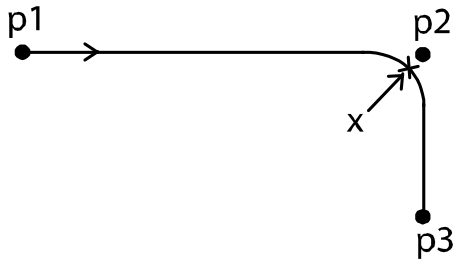
涉及MoveLSync指令的示例

可通过一次指令调用来实现“在机器人路径尽量靠近目标点时调用一则无返回值程序”。

```
MoveJ p1, vmax, fine, tool1;
!Move to p2 while calling a procedure
MoveLSync p2, v1000, z20, tool1, "proc1";
MoveL p3, v1000, fine, tool1;
```

结果

当TCP位于用X标记的位置时（参见插图），系统便会调用该无返回值程序。



xx0300000165

2.8 Logical Cross Connections

2.8.1 Logical Cross Connections介绍

目的

Logical Cross Connections的作用是检查和影响各数字I / O信号（DO、DI）或编组I / O信号（GO、GI）的组合。可由此验证或控制相关机器人之外的工艺设备。此项功能相当于一个简单的PLC。

若令I / O系统用I / O信号处理逻辑运算，则可避免执行许多RAPID代码。Logical Cross Connections可取代进程“读取I / O信号值、计算新值，并在I / O信号中写入数值”。

此处是一些应用示例：

- 当三个输入信号中的任意一个被设置成1时，系统便会中断程序执行过程。
- 若两个输入信号都被设置成1，那么就把一个输出信号设置成1。

描述

Logical Cross Connections的作用是定义一个I / O信号与其它I / O信号之间的依赖性。可用逻辑运算符AND和OR以及反信号值来配置更为复杂的依赖性。

若I / O信号由相应的逻辑表达式（执行I / O信号）和该表达式所得I / O信号（合成I / O信号）构成，那么该信号就可以是数字I / O信号（DO、DI）或编组I / O信号（GO、GI）。

其中包括

通过Logical Cross Connections，您最多可用5个执行I / O信号、逻辑运算AND和OR以及反信号值来构建逻辑表达式。

2.8.2 配置Logical Cross Connections

系统参数

此处简述了交叉连接所用的相关参数。更多信息请参见第131页的配置*Logical Cross Connections*中的各个参数。

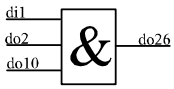
这些参数属于主题*I/O System*下的类型*Cross Connection*。

参数	描述
Name	指定该交叉连接的名称。
Resultant	把交叉连接的结果作为自身新值来接收的I / O信号。
Actor 1	求取*Resultant*时将使用的第一个I / O信号。
Invert actor 1	如果*Invert actor 1*被设置成*Yes*，那么求取*Resultant*时就使用*Actor 1*的反值。
Operator 1	*Actor 1*与*Actor 2*之间的运算元。 可以是这些运算元之一： • AND——如果两个输入值都为1，则得出该值为1。 • OR——如果至少有一个输入值为1，则得出该值为1。 **注意** 从左到右计算相关的运算符（即从*Operator 1*开始，到*Operator 4*结束）。
Actor 2	求取*Resultant*时将使用的第二个I / O信号（若此类信号超过一个）。
Invert actor 2	如果*Invert actor 2*被设置成*Yes*，那么求取*Resultant*时就使用*Actor 2*的反值。
Operator 2	*Actor 2*与*Actor 3*之间的运算元。 参见*Operator 1*。
Actor 3	求取*Resultant*时将使用的第三个I / O信号（若此类信号超过两个）。
Invert actor 3	如果*Invert actor 3*被设置成*Yes*，那么求取*Resultant*时就使用*Actor 3*的反值。
Operator 3	*Actor 3*与*Actor 4*之间的运算元。 参见*Operator 1*。
Actor 4	求取*Resultant*时将使用的第四个I / O信号（若此类信号超过三个）。
Invert actor 4	如果*Invert actor 4*被设置成*Yes*，那么求取*Resultant*时就使用*Actor 4*的反值。
Operator 4	*Actor 4*与*Actor 5*之间的运算元。 参见*Operator 1*。
Actor 5	求取*Resultant*时将使用的第五个I / O信号（若此类信号超过四个）。
Invert actor 5	如果*Invert actor 5*被设置成*Yes*，那么求取*Resultant*时就使用*Actor 5*的反值。

2.8.3 示例

逻辑AND

以下逻辑结构……



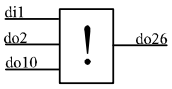
xx0300000457

……创建如下。

Resultant	Actor 1	Invert actor 1	Operator 1	Actor 2	Invert actor 2	Operator 2	Actor 3	Invert actor 3
do26	di1	No	AND	do2	No	AND	do10	No

逻辑OR

以下逻辑结构……



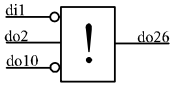
xx0300000459

……创建如下。

Resultant	Actor 1	Invert actor 1	Operator 1	Actor 2	Invert actor 2	Operator 2	Actor 3	Invert actor 3
do26	di1	No	OR	do2	No	OR	do10	No

反信号

以下逻辑结构（一个环形代表一个反信号）……



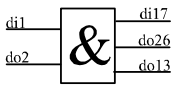
xx0300000460

……创建如下。

Resultant	Actor 1	Invert actor 1	Operator 1	Actor 2	Invert actor 2	Operator 2	Actor 3	Invert actor 3
do26	di1	Yes	OR	do2	No	OR	do10	Yes

若干结果

无法用一条交叉连接来执行以下逻辑结构……



xx0300000462

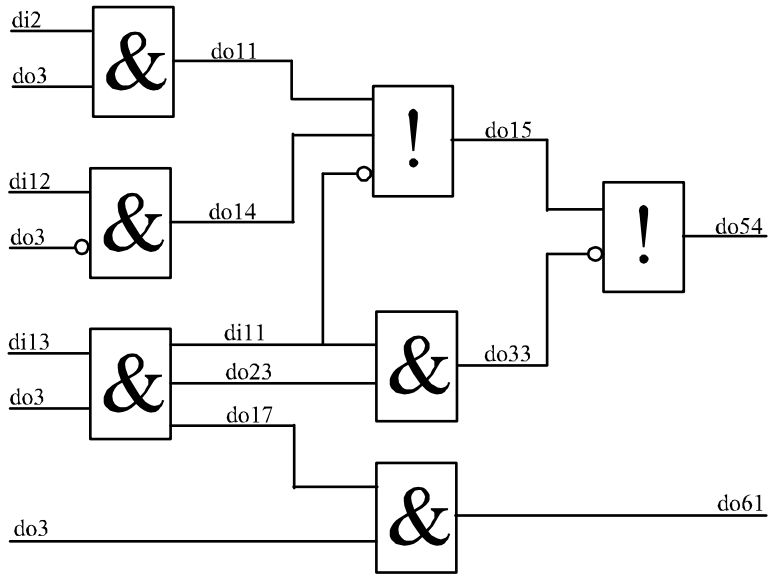
下一页继续

……但若有三条交叉连接，则能执行如下。

Resultant	Actor 1	Invert actor 1	Operator 1	Actor 2	Invert actor 2
di17	di1	No	AND	do2	No
do26	di1	No	AND	do2	No
do13	di1	No	AND	do2	No

复杂条件

以下逻辑结构……



xx0300000461

……创建如下。

Resultant	Actor 1	Invert actor 1	Operator 1	Actor 2	Invert actor 2	Operator 2	Actor 3	Invert actor 3
do11	di2	No	AND	do3	No			
do14	di12	No	AND	do3	Yes			
di11	di13	No	AND	do3	No			
do23	di13	No	AND	do3	No			
do17	di13	No	AND	do3	No			
do15	do11	No	OR	do14	No	OR	di11	Yes
do33	di11	No	AND	do23	No			
do61	do17	No	AND	do3	No			
do54	do15	No	OR	do33	Yes			

2.8.4 限制

求值顺序

如果在一条交叉连接中使用了两个以上的执行I / O信号，那么从左到右依次求值。这意味着先对*Actor 1*与*Actor 2*之间的运算进行求值，然后将其求取结果用在与*Actor 3*有关的运算中。

如果一条交叉连接中的所有运算符均属同一类型（仅为AND或仅为OR），那么求值顺序没什么影响。不过若在不考虑求值顺序的情况下混用AND和OR运算符，则可能产生意料之外的结果。

提示

采用若干条交叉连接，而不是在同一条交叉连接中混用AND和OR。

执行I / O信号的最大数目

一提案交叉连接的执行I / O信号可能不会超过五个。若需更多执行I / O信号，则请采用多条交叉连接。

交叉连接的最大次数

本机器人系统最多处理300条交叉连接。

最大深度

交叉连接求值的最大允许深度为20。

可将一条交叉连接的结果作为另一条交叉连接的执行器，而后者的结果则可依序作为下一条交叉连接的执行器。不过这种依赖性交叉连接链的深度不能超过20步。

勿创建环路

交叉连接不得出自闭合链，否则会导致无限次的求值和振荡。闭合链会出现在各交叉连接相互关联之时，并使交叉连接链形成一个循环。

相同的结果勿出现一次以上

不得使用模棱两可的合成I / O信号，否则相关结果将取决于求值顺序（无法控制这种顺序）。当若干条交叉连接均得出同一I / O信号后时，便会出现模棱两可的合成I / O信号。

重叠各装备映射

对一条交叉连接中的合成I / O信号而言，其装置映射不得与该交叉连接所定义的任何反向执行I / O信号重叠。使用交叉连接中存在重叠装置映射的I / O信号会导致无限次信号设定环路。

2.9 RAPID Message Queue

2.9.1 RAPID Message Queue介绍

目的

RAPID Message Queue的作用是与另一项RAPID任务或使用PC SDK的PC应用进行通信。

此处是一些应用示例：

- 两项RAPID tasks之间的发送数据。
- 一项RAPID task与一项PC应用之间的发送数据。

可针对中断模式或同步模式来定义RAPID Message Queue。默认设置为中断模式。

其中包括

RobotWare选项包括了RAPID Message Queue功能：

- Multitasking
- RobotStudio Connect

您可通过RAPID Message Queue来访问用于发送和接收数据的RAPID指令、函数和数据类型。

基本方法

这是RAPID Message Queue的一般用法。第*141*页的代码示例用一个更详细的示例展示了其具体用法。

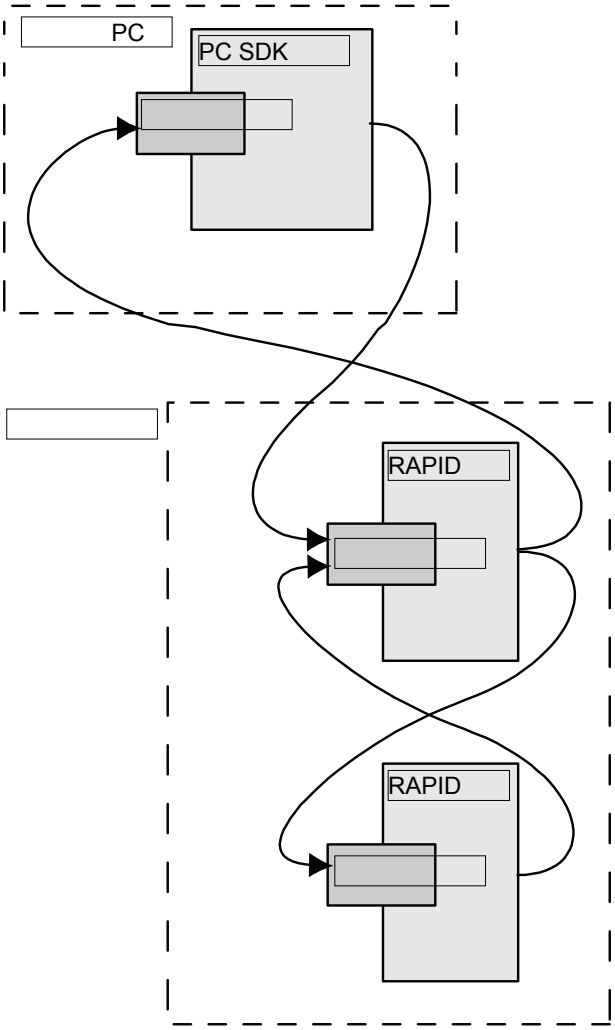
1. 对于中断模式：接收方设置了一则软中断例程来读取一则消息和关联一次中断，因此系统会在出现一则新消息时调用该软中断例程。

   对于同步模式：系统会通过一次等候或接下来执行的`RMQReadWait`指令来处理该消息。
2. 发送方会在接收方任务中的队列查询相应的槽标识。
3. 发送方会发送该消息。

2.9.2 RAPID消息队列行为

通信图

下图展示了可能出现在本系统中的各种发送方、接收方和队列。每个箭头都例举了一种向队列粘贴一则消息的途径。



en0700000430

创建一个PC SDK客户端

本手册只描述了如何用RAPID Message Queue来让一项RAPID task与其它RAPID tasks和PC SDK进行通信。至于如何在一个PC SDK客户端上设置相关通信，则请参见*http://developercenter.robotstudio.com*。

一则消息所能发送的内容

除以下内容外，一则消息中的数据可以属于RAPID中的任何数据类型：

- 非值
- 半值
- `motsetdata`

下一页继续

一则消息中的数据可以是属于某种数据类型的一个数组。

允许使用用户定义的记录，但发送方和接收方都必须对该记录作出一模一样的声明。

提示

为了保持向后兼容性，一旦在发布的某款产品中使用了用户定义的记录，就请不要更改该记录。更好的做法是创建一份新的纪录，这样就既能接收旧应用的消息，也能接收新应用的消息。

队列名称

为RAPID task配置的队列名称就是带有前缀“RMQ_”的任务名称，比如RMQ_T_ROB1这样。指令RMQFindSlot会使用该名称。

队列处理

系统会按接收消息的顺序来处理队列中的消息，这种特性被称做FIFO，即“先进先出”。如果还在处理前一则消息时就接收到了新的消息，那么系统会把该新消息置于队列中。一旦第一则消息处理完毕，系统便会立即处理队列中的下一则消息。

队列模式

用系统参数*RMQ Mode*来定义相应的队列模式。默认行为为中断模式。

中断模式

中断模式下的消息处理情况取决于数据类型。系统只会处理已关联了其数据类型的消息。

必须为宜由接收方处理的每种数据类型设置一种循环中断。可在不同的中断中调用同一则软中断例程，即是说数据类型不止一种。

系统会抛弃数据类型未关联到中断上的消息，并仅在事件日志中留下一则警告消息。

接收对指令RMQSendWait的答复不会导致一次中断。无需设置中断便可接收这一答复。

同步模式

在同步模式下，该项任务会执行一条RMQReadWait指令来接收任何数据类型的一则消息。系统会将所有消息列队，然后按其抵达顺序处理。

若有一条等候RMQReadWait指令，则系统会立即处理该消息。

若没有等候RMQReadWait指令，则接下来执行的RMQReadWait指令将会处理该消息。

消息内容

RAPID Message Queue消息由包含接收方标识的一条标题和一则RAPID消息组成。该RAPID消息为一段优质打印的字符串，其数据类型名称（以及数组维度）后面是实际数据值。

RAPID消息示例：

```
"robtarget;[[930,0,1455],[1,0,0,0],[0,0,0,0],
     [9E9,9E9,9E9,9E9,9E9,9E9]]"
"string;"A message string""
"msgrec;[100,200]"
"bool{2,2};[[TRUE,TRUE],[FALSE,FALSE]]"
```

下一页继续

2.9.2 RAPID消息队列行为
续前页

未执行RAPID任务

即使当前并未在执行包含了某RAPID任务队列的RAPID任务，用户也或可向该队列粘贴消息。到再次执行相关RAPID任务为止都不会执行相应的中断。

消息大小限制

系统发送一则消息前会先计算其最大大小（其具体数据类型和维度的最大大小）。如果其大小超过了5000字节，那么系统将抛弃该消息，并产生一项错误；如果接收方是一个最大消息大小不到400字节的PC SDK客户端，那么发送方也会得到相同的错误。对拥有具体数据类型和维度的一则消息而言，系统要么始终都能发送该消息，要么始终无法发送该消息。

当收到一则消息（调用指令RMQGetMsgData）时，系统会计算其最大大小（其具体数据类型和维度的最大大小）。如果其大小超过了为该任务的队列配置的最大消息大小，那么系统将抛弃该消息，并记录一项错误。对拥有具体数据类型和维度的一则消息而言，系统要么始终都能接收该消息，要么始终无法接收该消息。

消息丢失

在中断模式下，系统会抛弃RAPID任务无法接收的任何消息。这些消息将会丢失，而事件日志中则会加入一条警告。

抛弃一则消息的部分原因：

- 接收任务不支持所发送的数据类型。
- 接收任务并未为所发送的数据类型设置中断，也没有任何RMQSendWait指令在等候该数据类型。
- 接收任务的中断队列已满

队列丢失

会在上电失败时清除该队列。

当丢失了RAPID task中的执行文本（比如将相应的程序指针移到主例程处）时，系统便会清空相应队列。

相关信息

队列与消息方面的更多信息请参见技术参考手册 - *RAPID*语言内核。

2.9.3 系统参数

关于系统参数

此处简述了*RAPID Message Queue*功能中的每个参数。更多信息请参见技术参考手册 - 系统参数中的各个参数。

类型Task

这些参数属于主题*Controller*下的类型*Task*。

参数	描述
RMQ Type	可以拥有下列数值之一： • *None* – 就RAPID task来禁用与RAPID Message Queue之间的一切通信。 • *Internal* – 启用从相关控制器上的其它任务处——而不是从外部客户端（FlexPendant示教器和各种PC应用）处——接收RAPID Message Queue消息。该任务仍能向外部客户端发送消息。 • *Remote* – 就该项任务启用与RAPID Message Queue之间的通信，这既包括与相关控制器上的其它任务之间的通信，也包括与外部客户端（FlexPendant示教器和各种PC应用）之间的通信。 默认值为*None*。
RMQ Mode	定义相关队列的模式。 可以拥有下列数值之一： • *Interrupt* – 只有将一则软中断例程与一种指定的消息类型关联在一起后，系统才能接收该消息。 • *Synchronous* – 只有在执行了一条`RMQReadWait`指令后，系统才能接收该消息。 默认值为*Interrupt*。
RMQ Max Message Size	一则*RAPID Message Queue*消息的最大数据大小（以字节计）。 400到3000之间的一个整数。默认值为400。 **注意** 该数值无法在 RobotStudio 中或 FlexPendant 上更改。更改该数值的唯一方法是为 *RmqMaxMsgSize* 属性添加所需数值编辑 sys.cfg 文件。
RMQ Max No Of Messages	*RAPID Message Queue*消息在该任务队列中的最大数目。 1到10之间的一个整数。默认值为5。 **注意** 该数值无法在 RobotStudio 中或 FlexPendant 上更改。更改该数值的唯一方法是为 *RmqMaxNoOfMsg* 属性添加所需数值编辑 sys.cfg 文件。

2.9.4 RAPID组件

关于RAPID组件

此处简述了RAPID Message Queue中的每条指令、每则函数和每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各个参数。

指令：

指令	描述
RMQFindSlot	查找为某一RAPID task或Robot Application Builder客户端配置的队列的槽标识号。
RMQSendMessage	向为某一RAPID task或Robot Application Builder客户端配置的队列发送数据。
IRMQMessage	命令并启用某一具体数据类型的循环中断。
RMQGetMessage	从该项任务的队列中获取第一则消息。只有在*RMQ Mode*被定义为*Interrupt*的情况下才能使用该指令。
RMQGetMsgHeader	获取一则消息中的标题部分。
RMQGetMsgData	获取一则消息中的数据部分。
RMQSendWait	发送一则消息，然后等候答复。只有在*RMQ Mode*被定义为*Interrupt*的情况下才能使用该指令。
RMQReadWait	等候消息。只有在*RMQ Mode*被定义为*Synchronous*的情况下才能使用该指令。
RMQEmptyQueue	清空相应队列。

函数

功能	描述
RMQGetSlotName	在给定槽标识号（即给定`rmqslot`）情况下获取为某一RAPID task或Robot Application Builder客户端配置的队列的名称。

数据类型

数据类型	描述
rmqslot	某一RAPID task或 Robot Application Builder客户端的槽标识。
rmqmessage	与RAPID Message Queue通信时用来保存数据的一则消息，其中包含了所发送的数据类型、发送方的槽标识和实际数据这些信息。 **注意：**`rmqmessage`是一种较大的数据类型。为该数据类型声明太多变量会导致内存问题。请尽量重复使用相同的`rmqmessage`变量。
rmqheader	`rmqheader`描述了相关消息，并可由RAPID程序加以读取。

2.9.5 代码示例

涉及RMQSendMessage和RMQGetMessage的示例

此例中的发送方创建了数据（x和y值），并将其发送给另一项任务。相关的接收任务获得了该信息，并将这些数据提取到名为data的变量中。

发送方

```
MODULE SenderMod
  RECORD msgrec
    num x;
    num y;
  ENDRECORD

  PROC main()
    VAR rmqslot destinationSlot;
    VAR msgrec data;
    VAR robtarget p_current;

    ! Connect to queue in other task
    RMQFindSlot destinationSlot "RMQ_OtherTask";

    ! Perform cycle
    WHILE TRUE DO
      ...
      p_current := CRobT(\Tool:=tool1 \WObj:=wobj0);
      data.x := p_current.trans.x;
      data.y := p_current.trans.y;
      ! Send message
      RMQSendMessage destinationSlot, data;
      ...
    ENDWHILE
  ERROR
    IF ERRNO = ERR_RMQ_INVALID THEN
      WaitTime 1;
      ! Reconnect to queue in other task
      RMQFindSlot destinationSlot "RMQ_OtherTask";
      ! Avoid execution stop due to retry count exceed
      ResetRetryCount;
      RETRY;
    ELSIF ERRNO = ERR_RMQ_FULL THEN
      WaitTime 1;
      ! Avoid execution stop due to retry count exceed
      ResetRetryCount;
      RETRY;
    ENDIF
  ENDPROC
ENDMODULE
```

PC SDK客户端

```
public void RMQReceiveRecord()
```

下一页继续

2.9.5 代码示例
续前页

```
{
  const string destination_slot = "RMQ_OtherTask";
  IpcQueue queue = Controller.Ipc.CreateQueue(destination_slot,
       16, Ipc.MaxMessageSize);

  // Till application is closed
  while (uiclose)
  {
    IpcMessage message = new IpcMessage();
    IpcReturnType retValue = IpcReturnType.Timeout;

    retValue = queue.Receive(1000, message);
    if (IpcReturnType.OK == retValue)
    {
      // PCSDK App will receive following record
      // RECORD msgrec
      // num x;
      // num y;
      // ENDRECORD

      // num data type in RAPID is 3 bytes long, hence will receive
           6 bytes for x and y
      // first byte do left shift by 16,
      // second byte do left shift by 8 and OR all three byte to
           get x
      // do similar for y
      Int32 x = (message.Data[0] << 16) | (message.Data[1] << 8)
           | message.Data[2];
      Int32 y = (message.Data[3] << 16) | (message.Data[4] << 8)
           | message.Data[5];

      // Display x and y
    }
  }

  if (Controller.Ipc.Exists(destination_slot))
    Controller.Ipc.DeleteQueue(Controller.Ipc.GetQueueId(destination_slot));
}
```

涉及RMQSendWait的示例

此例中的RAPID程序发送了一则消息，然后等候答复，最后通过获取相应的答复消息来继续执行过程。

```
MODULE SendAndReceiveMod
  VAR rmqslot destinationSlot;
  VAR rmqmessage recmsg;
  VAR string send_data := "How many units should be produced?";
  VAR num receive_data;

  PROC main()
    ! Connect to queue in other task
    RMQFindSlot destinationSlot "RMQ_OtherTask";
```

下一页继续

```
    ! Send message and wait for the answer
    RMQSendWait destinationSlot, send_data, recmsg, receive_data
         \Timeout:=30;

    ! Handle the received data
    RMQGetMsgData recmsg, receive_data;
    TPWrite "Units to produce: " \Num:=receive_data;

  ERROR
    IF ERRNO = ERR_RMQ_INVALID THEN
      WaitTime 1;
      ! Reconnect to queue in other task
      RMQFindSlot destinationSlot "RMQ_OtherTask";
      ! Avoid execution stop due to retry count exceed
      ResetRetryCount;
      RETRY;
    ELSIF ERRNO = ERR_RMQ_FULL THEN
      WaitTime 1;
      ! Avoid execution stop due to retry count exceed
      ResetRetryCount;
      RETRY;
    ELSEIF ERRNO = ERR_RMQ_TIMEOUT THEN
      ! Avoid execution stop due to retry count exceed
      ResetRetyCount;
      RETRY;
    ENDIF
  ENDPROC
ENDMODULE
```

涉及RMQReceiveSend的示例

```
public void RMQReceiveSend()
  {
    const string destination_slot = "RMQ_OtherTask";
    IpcQueue queue = Controller.Ipc.CreateQueue(destination_slot,
         16, Ipc.MaxMessageSize);

    // Till application is closed
    while (uiclose)
    {
      IpcMessage message = new IpcMessage();
      IpcReturnType retValue = IpcReturnType.Timeout;

      retValue = queue.Receive(1000, message);
      if (IpcReturnType.OK == retValue)
      {
        // Received message "How many units should be produced?"
        if (message.ToString() == "How many units should be
             produced?")
        {
          Int32 UnitsToProduce = 100;
```

下一页继续

2.9.5 代码示例
续前页

```
            // num data type in Rapid is 3 bytes long, hence will
                 send 3 bytes to Rapid Module
            byte[] @bytes = new byte[3];
            bytes[0] = (byte)(UnitsToProduce >> 16);
            bytes[1] = (byte)(UnitsToProduce >> 8);
            bytes[2] = (byte)UnitsToProduce;

            // Send UnitsToProduce to Rapid Module
            message.SetData(@bytes);
            queue.Send(message);
          }
        }
      }

      if (Controller.Ipc.Exists(destination_slot))
      Controller.Ipc.DeleteQueue(Controller.Ipc.GetQueueId(destination_slot));
    }
```

2.10 Socket Messaging

2.10.1 Socket Messaging介绍

目的

Socket Messaging的作用是允许RAPID程序员通过TCP/IP网络协议在各台计算机之间传输应用数据。一个套接字代表了一条独立于当前所用网络协议的通用通信通道。

“套接字通信”是源于Berkeley所发布软件Unix的一套标准，而除Unix外，Microsoft Windows等平台也支持该项标准。有了Socket Messaging，机器人控制器上的RAPID程序就能与另一台计算机上的C/C++程序等进行通信。

其中包括

您可通过 RobotWare Socket Messaging功能来访问各计算机之间套接字通信所需的 RAPID 数据类型、指令和函数。

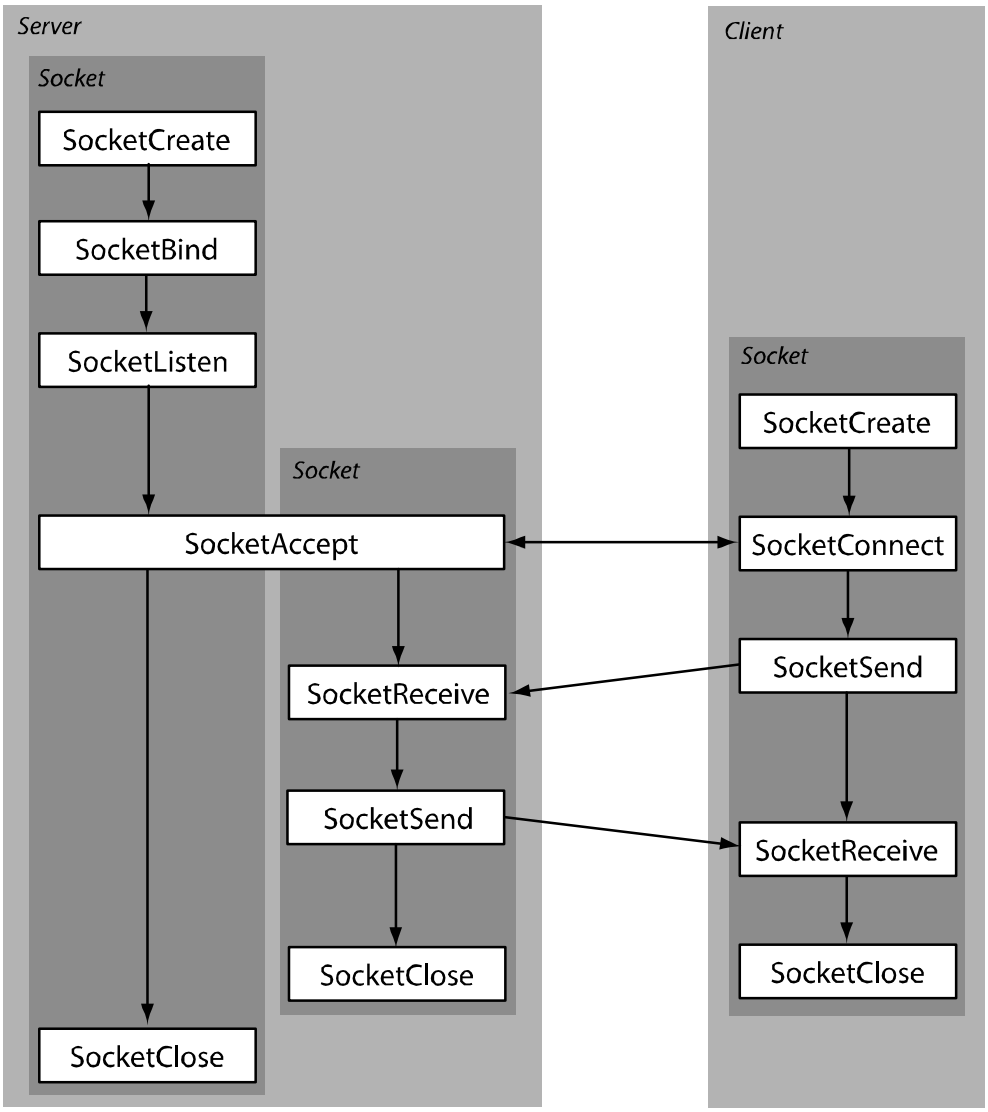
基本方法

这是Socket Messaging的一般用法。第150页的*Socket Messaging*的代码示例用一个更详细的示例展示了其具体用法。

1. 在客户端和服务器上分别创建一个套接字。机器人控制器可以是客户端，也可以是服务器。
2. 在相关服务器上使用`SocketBind`和`SocketListen`，使其对连接请求作好准备。
3. 命令相关服务器接受外来的套接字连接请求。
4. 从相关客户端提出套接字连接请求。
5. 在客户端与服务器之间发送和接收数据。

2.10.2 套接字通信的示意图

套接字通信图



en0600003224

提示

若无必要，则请勿创建和关闭套接字。通信完毕前请一直开启相应的套接字。出于TCP/IP功能之故，在`SocketClose`后，该套接字要过一段时间后才会真正关闭。

2.10.3 关于Socket Messaging的技术事实

概述

在应用Socket Messaging功能与非 RAPID 任务的客户端或服务器通信时，以下信息可能有用。

无字符串终止

当发送一则数据消息时，系统会在该消息中发送一个无字符串终止符。所发送的字节数等同于编程语言C中函数`strlen(str)`的返回之后。

消息的意外合并

如果发送了两则消息，且在它们之间没有任何延时，那么第二则消息就可能附加到第一则消息上，从而形成一则大消息而非两则消息。为避免这种情况发生，如果相关客户端 / 服务器正在接收消息，那么应使用来自数据接收方的确认消息。

不可打印字符

如果某个并非RAPID任务的客户端需要从某RAPID任务的一段字符串中接收不可打印字符（二进制数据），那么便可用RAPID按下例所示进行接收。

```
SocketSend socket1 \Str:="\0D\0A";
```

更多信息请参见技术参考手册 - *RAPID*语言内核的字符串文字一节。

2.10.4 RAPID组件

数据类型

此处简述了Socket Messaging中的每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型。

数据类型	描述
socketdev	一件套接字装置，用于与网络上的其它计算机进行通信。
socketstatus	可包含一个socketdev变量的状态信息。

用于客户端的指令

此处简述了Socket Messaging客户端所用的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型。

指令	描述
SocketCreate	创建一个新的套接字，并赋予其一个socketdev 变量。
SocketConnect	向一台远程计算机提出连接请求。客户端将用其连接相应的服务器。
SocketSend	通过套接字连接向某台远程计算机发送数据。这些数据既可以是string或rawbytes变量，也可以是byte数组。
SocketReceive	接收数据，并将其保存在一个string或rawbytes变量中，或保存在一个byte数组中。
SocketClose	关闭一个套接字，随之释放所有资源。

提示

不要在SocketSend之后直接使用SocketClose。先等候确认，然后再关闭套接字。

用于服务器的指令

除SocketConnect外，Socket Messaging服务器与客户端使用同一套指令。此外服务器还会使用下列指令：

指令	描述
SocketBind	将套接字与相关服务器上的一个指定端口号绑定起来。该服务器会用其来定义“用（该服务器上的）哪个端口监听某一连接”。 该IP地址定义了一台物理计算机，而该端口则定义了通往该计算机上某一程序的一条逻辑通道。
SocketListen	使该计算机作为一台服务器发挥作用，并接受外来的连接。其将监听SocketBind所指定端口上的某一连接。
SocketAccept	接受一项外来连接请求。服务器将用其来接受相关客户端的请求。

注意

必须先启动服务器应用，然后才启动客户端应用，这样才能在任一客户端执行SocketConnect前先执行SocketAccept指令。

下一页继续

函数

此处简述了Socket Messaging中的每则函数。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型。

功能	描述
SocketGetStatus	返回在套接字上执行的最后一条指令（已创建、已连接、边界、监听和关闭）。 `SocketGetStatus`不会检测来自外部 RAPID 的改动（比如断开的连接）。

2.10.5 Socket Messaging的代码示例

客户端 / 服务器通信的示例

此例展示了客户端和服务器彼此通信时所需的程序代码。

将在FlexPendant示教器上写入该服务器：

```
Client wrote - Hello server
Client wrote - Shutdown connection
```

将在FlexPendant示教器上写入该客户端：

```
Server wrote - Message acknowledged
Server wrote - Shutdown acknowledged
```

此例中的客户端和服务器都使用了RAPID程序，而在实际中，人们往往会在一台PC（或类似的计算机）上运行其中一段程序，并将该程序写为另一种程序语言。

客户端的代码示例，用 IP 地址192.168.0.2联系服务器：

```
! WaitTime to delay start of client.
! Server application should start first.
WaitTime 5;
VAR socketdev socket1;
VAR string received_string;
PROC main()
  SocketCreate socket1;
  SocketConnect socket1, "192.168.0.2", 1025;
  ! Communication
  SocketSend socket1 \Str:="Hello server";
  SocketReceive socket1 \Str:=received_string;
  TPWrite "Server wrote - " + received_string;
  received_string := "";
  ! Continue sending and receiving
  ...
  ! Shutdown the connection
  SocketSend socket1 \Str:="Shutdown connection";
  SocketReceive socket1 \Str:=received_string;
  TPWrite "Server wrote - " + received_string;
  SocketClose socket1;
ENDPROC
```

服务器的代码示例（带 IP 地址192.168.0.2）：

```
VAR socketdev temp_socket;
VAR socketdev client_socket;
VAR string received_string;
VAR bool keep_listening := TRUE;
PROC main()
  SocketCreate temp_socket;
  SocketBind temp_socket, "192.168.0.2", 1025;
  SocketListen temp_socket;
  WHILE keep_listening DO
    ! Waiting for a connection request
    SocketAccept temp_socket, client_socket;
    ! Communication
    SocketReceive client_socket \Str:=received_string;
```

下一页继续

```
    TPWrite "Client wrote - " + received_string;
    received_string := "";
    SocketSend client_socket \Str:="Message acknowledged";
    ! Shutdown the connection
    SocketReceive client_socket \Str:=received_string;
    TPWrite "Client wrote - " + received_string;
    SocketSend client_socket \Str:="Shutdown acknowledged";
    SocketClose client_socket;
  ENDWHILE
  SocketClose temp_socket;
ENDPROC
```

错误处理器示例

下列错误处理器将对上电失败或断开连接进行处理。

针对上例所用客户端的错误处理器：

```
! Error handler to make it possible to handle power fail
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
    SocketClose socket1;
    ! WaitTime to delay start of client.
    ! Server application should start first.
    WaitTime 10;
    SocketCreate socket1;
    SocketConnect socket1, "192.168.0.2", 1025;
    RETRY;
  ELSE
    TPWrite "ERRNO = "\Num:=ERRNO;
    Stop;
  ENDIF
```

针对上例所用服务器的错误处理器：

```
! Error handler for power fail and connection lost
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
    SocketClose temp_socket;
    SocketClose client_socket;
    SocketCreate temp_socket;
    SocketBind temp_socket, "192.168.0.2", 1025;
    SocketListen temp_socket;
    SocketAccept temp_socket, client_socket;
    RETRY;
  ELSE
    TPWrite "ERRNO = "\Num:=ERRNO;
    Stop;
  ENDIF
```

2.11 User logs

2.11.1 User logs介绍

描述

RobotWare 基本功能*User logs*会就大部分普通用户行为生成事件日志。这些事件日志会生成在运行事件组内，其序列号为*1 xxxx*。

有关事件日志处理的更多信息，请参见操作手册 - *OmniCore* 和 技术参考手册 - *RobotWare 7*事件日志。

目的

*User logs*的用途是跟踪机器人控制器中与用户行为有关的变化。举例来说，如果生产出现停止，那么就能借助该功能来寻找根本原因。

其中包括

RobotWare 基本功能*User logs*会就涉及用户行为的以下变化生成事件日志：

主题	用户行为
程序执行	更改速度或运行模式（单循环 / 连续）。在任务选择面板上做出更改。设置或重置无运动的执行模式。
模拟等待指令	模拟各种等待指令，比如`WaitTime`、`WaitUntil`和`WaitDx`等。
更改RAPID	打开或关闭RAPID程序或模块、编辑RAPID代码或修改机器人的位置。
移动程序指针	将程序指针移动到主程序、某个例程、某个位置或某个服务器例程（调用例程）。
更改机械单元	更新转速计数器或进行校准。
微动控制	更改工具、工件、有效负载、坐标系或前往某个位置。
监控	设置或重置点动或路径监控。设置监控等级。
更改配置	载入配置数据或更改某一配置属性。
更改系统	清空事件日志或更改日期和时间。
串行测量电路板	更改串行测量板中的数据，或更改机器人内存中的数据。

3 Motion Performance

3.1 Absolute Accuracy [3101-x]

3.1.1 关于Absolute Accuracy

目的

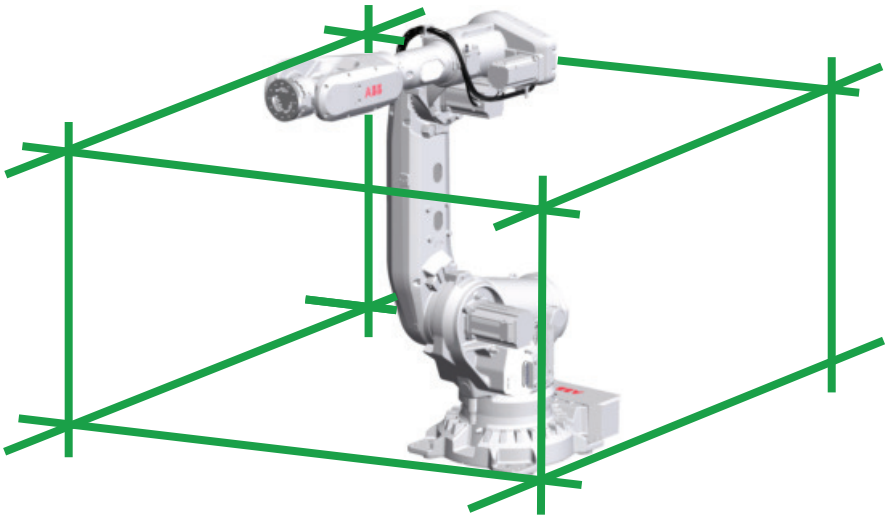
*Absolute Accuracy*是提高TCP精度的校准概念。理想机器人与真实机器人之间可能存在几毫米的差异，这是机器人结构中的机械公差和偏转引起的。*Absolute Accuracy*可以补偿这些差异。

这里有一些示例说明了这种准确度在何时意义重大：

- 机器人的可交换性。
- 无需修整或者最低程度修整下的离线编程。
- 通过工具的精确移动和重新定向进行的在线编程
- 通过与图像系统或偏移量编程等有关的精确偏移移动来进行的编程
- 重新使用各应用之间的程序

*Absolute Accuracy*选件集成在控制器算法中，并且不需要外部设备或计算。

> **注意**
>
> 性能数据适用于单台机器人的相应 RobotWare 版本。



xx1300002177

包括哪些

每台*Absolute Accuracy*机器人在交付时，均具有：

- 该机器人系列测量板上保存的补偿参数
- 一份出厂证书，代表了校准与验证序列所用的*Absolute Accuracy*测量协议。

在具有*Absolute Accuracy*校准功能机械臂的操纵器上有一个带有该信息的标签。

Absolute Accuracy支持落地式、壁挂式和吸顶式安装。机器人串行测量板上保存的补偿参数取决于选定的Absolute Accuracy选件。

下一页继续

3.1.1 关于Absolute Accuracy
续前页

何时使用*Absolute Accuracy*

Absolute Accuracy 的作用对象是笛卡尔坐标上的一个机器人目标点，而并非单个关节，因此基于关节的移动（如 MoveAbsJ）将不受影响。

如果机器人倒置安装，必须在倒置机器人时进行 Absolute Accuracy 校准。

Absolute Accuracy处于激活状态

下列情况将会激活Absolute Accuracy：

- 机器人目标点上有任何基于函数的运动（如 MoveL），或对机器人目标点进行了ModPos
- 重定方位点动
- 线性点动
- 工具定义（4、5、6点工具定义、房间固定点TCP、固定工具）
- 工件定义

Absolute Accuracy未处于激活状态

以下示例说明了Absolute Accuracy何时不会处于激活状态：

- 关节目标点上任何基于函数的运动 (MoveAbsJ)
- 独立关节
- 基于关节的点动
- 附加轴
- 动作跟踪

注意

例如，在具有附加轴或轨道运动的机器人系统中，Absolute Accuracy 为机械臂激活，但没有为附加轴或轨道运动无效。

RAPID指令

该选项中不包含RAPID指令。

3.1.2 有用工具

概述

运行和维护Absolute Accurate机器人时建议使用以下产品：

- Load Identification
- CalibWare（Absolute Accuracy校准工具）

Load Identification

Absolute Accuracy根据有效负载计算了机器人的偏移。准确描述相关负载可以说非常重要。

Load Identification是一种工具，其作用是确定有效负载的质量、重心和惯量。

有关更多信息，请参见操作手册 - *OmniCore*。

CalibWare

ABB公司提供了CalibWare作为校准Absolute Accuracy的工具。CalibWare的文件详细描述了Absolute Accuracy无返回值校准程序。

在首次校准和保养机器人时使用CalibWare。

3.1.3 配置

激活Absolute Accuracy

使用RobotStudio，然后遵守下列步骤（更多信息请参见操作手册 - *RobotStudio*）：

1 如果您尚无写入权限，则点击**请求写入权限**，然后等待FlexPendant示教器的授权。
2 点击**配置编辑器**，然后选择**运动**。
3 点击类型**机器人**。
4 对于参数*Use Robot Calibration*，将值更改为*r1_calib*。
5 对于 MultiMove 系统，配置各机器人的参数*Use Robot Calibration*。机器人 2 应设置为*r2_calib*，机器人 3 应设置为*r3_calib*，机器人 4 应设置为*r4_calib*。
6 重启控制器，从而使所做改动生效。

停用Absolute Accuracy

使用RobotStudio，然后遵守下列步骤（更多信息请参见操作手册 - *RobotStudio*）：

1 如果您尚无写入权限，则点击**请求写入权限**，然后等待FlexPendant示教器的授权。
2 单击 **Configuration Editor（配置编辑器）**，然后选择主题 **Motion（运动）**。
3 点击类型**机器人**。
4 配置参数*Use Robot Calibration*，然后将数值更改为“r1_uncalib”。
5 至于MultiMove系统，则在每台机器人上重复步骤3和4，然后将机器人2、机器人3和机器人4的使用机器人校准分别设置成“r2_uncalib”、“r3_uncalib”和“r4_uncalib”。
6 重启控制器，从而使所做改动生效。

更改校准数据

若您交换了机械臂，则必须载入新机械臂的calibration数据，也就是把该机器人系统测量板中的校准数据复制到相应的机器人控制器上。

使用FlexPendant示教器，然后遵守下列步骤（更多信息请参见操作手册 - *OmniCore*）：

1 在“开始”屏幕上，点击 **Calibrate（校准）**，然后从菜单中选择 **Calibration（校准）**。
2 轻击想要更新的机器人。
3 点击**机器人内存**选项卡。
4 轻击**高级**。
5 点击**清除控制器内存**。
6 轻击**清除**，然后通过轻击**是**加以确认。
7 轻击**关闭**。
8 点击 **Update**（更新）。
9 点击**机柜或机器人已交换**并点**是**确认。

3.1.4 维护

3.1.4.1 影响准确度的维护

概述

本节将关注那些会直接影响到机器人准确度的维护活动，并总结如下：

- 工具的重新校准
- 电机的更换
- 腕的更换（大型机器人）
- 臂的更换（低臂、高臂、齿轮箱和足）
- 机械臂的更换
- 丧失准确度

> **注意**
>
> 如果控制器上的 RobotWare 版本必须降级，请与您当地的 ABB 联系，以获得 Absolute Accuracy兼容版本的支持。

工具的重新校准

重新校准工具方面的更多信息请参见第*171*页的工具校准。

电机的更换

更换所有电机需要使用各个机器人的标准校准方法来重新校准相应的旋转变压器偏移参数。在机器人产品手册中对此作了说明。

如果更换电机需要拆卸臂，则请参阅第*157*页的臂的更换或拆卸。

腕的更换

更换腕部单元需要使用各个机器人的标准校准方法，重新校准轴 5 和轴 6 旋转变压器的偏移。

臂的更换或拆卸

更换任何机器人臂或其它机械结构（腕除外）都会使机器人的结构发生显著变化，因而需要重新校准机器人。建议更换臂后最好重新校准整台机器人，宜确保Absolute Accuracy功能处于最佳状态。重新校准时通常采用CalibWare和一套单独的测量系统。CalibWare可与任何通用的3D测量系统搭配使用。

至于此校准进程方面的更多信息，则请参见CalibWare的文件。

此校准进程总结如下：

	操作
1	更换受影响的部件。
2	对所有轴进行旋转变压器校准。参见各台机器人的产品手册。
3	重新校准 TCP。
4	通过比较围笼中的固定参考点来检查相应的准确度。

下一页继续

3.1.4.1 影响准确度的维护

续前页

	操作
5	检查相关工件的准确度。  **注意** 更新已定义的工件将会减少定位时的偏差。
6	检查当前应用中的位置准确度。
7	如果对准确度仍不满意，那么就对整台机器人进行一次Absolute Accuracy校准。具体请参见CalibWare的文件。

机械臂的更换

若不更换控制柜就更换机器人的机械臂，那么就需更新该控制柜中的Absolute Accuracy参数，并将机器人重新对准围笼。正如第*156*页的更改校准数据所述，将新换机器人的校准参数载入相应的控制器后，便可更新Absolute Accuracy参数。确保既载入了相应的相应的校准数据，又激活了Absolute Accuracy。

至于如何将新换机器人对准围笼，则取决于安装时选择的机器人对准技术。如果新换机器人的安装销对准了围笼，那么无需作进一步对准，仅需将该机器人放在这些销上即可。如果用某种机器人程序来对准新换机器人，那么就需测量围笼的固定装置以及在数个位置上对该机器人进行测量（为达到最好效果，请使用原机器人所用的程序）。具体请参见第*169*页的测量机器人的对准情况。

3.1.4.2 丧失准确度

原因和行动

机器人发生碰撞或遭遇明显的温度波动后往往会丧失准确度。

需确定相关错误的原因，并采取适当的行动。

如果	...请...
未恰当校准此工具。	如果 TCP 已被改变，则重新进行校准。
未正确定义此工具负载	执行Load Identification，以确保所激活工具的质量、重心和惯量无误。
旋转变压器偏移不再有效	1 通过检查轴刻度来查看机器人是否正确地立在起始位置上。 2 如果指示器未对准，则将机器人移到正确的位置上，然后更新转数计数器。 3 如果指示器已近乎对准但存在误差，则应用机器人标准校准法进行重新校准。
机器人与固定装置间的关系已发生变化	1 将机器人移到相应固定装置的某个预定义位置上，以便开展检查。 2 目测评估偏差是否过大。 3 如果过大，则将机器人重新对准固定装置。
该机器人的结构已发生变化	1 目测评估该机器人是否受损。 2 如果受损，则更换整支机械臂，或更换受影响的臂，又或重新校准受影响的臂。

3.1.5 补偿理论

3.1.5.1 错误来源

错误类型

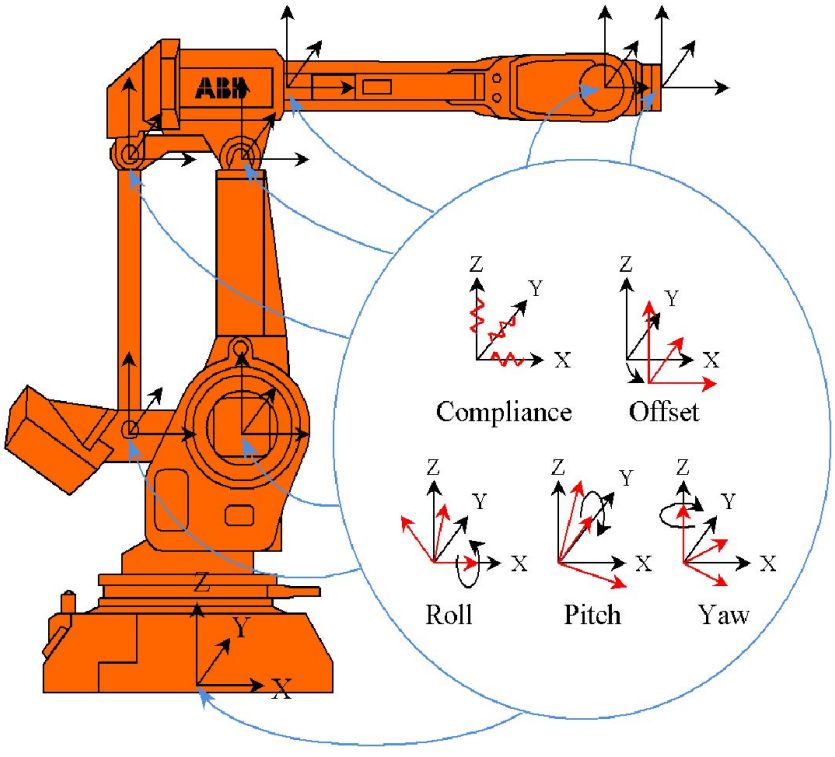
控制器中的已补偿错误源自机器人组成部分的机械容限。下文的插图详细展示了它们的一个子集。

机器人的自重以及当前有效负载导致了各种合规性错误。这些错误取决于相关负载的重力和特点。使用Load Identification可最高效地补偿这些错误（具体请参见操作手册 - *OmniCore*）。

机器人轴的位置或方位偏差导致了各种运动学错误。此类错误与负载无关。

图示

每个关节都可能发生若干种错误。



en0300000232

3.1.5.2 Absolute Accuracy补偿

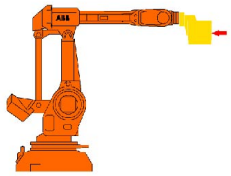
简介

用“虚假目标点”来补偿合规性错误和运动学错误。若了解机器人的偏移（即与预定位置之间的偏移程度），则可通过“下令该机器人前往某个虚假目标点”的方式来补偿 *Absolute Accuracy*。

补偿会作用在笛卡儿坐标上的某个机器人目标点上，而不是作用在单个关节上。这意味着该点就是TCP（标有下图中的箭头）在获得正确补偿后的位置。

所需位置

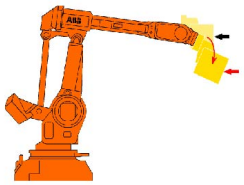
下图展示了您想设置的机器人位置。



xx0300000225

偏移所致位置

下图展示了机器人在没有*Absolute Accuracy*的情况下能实现的位置。机器人臂和负载的重量会使机器人发生偏移。注意图中的偏移有所夸大。



xx0300000227

虚假目标点

为了获得所需位置，*Absolute Accuracy*会计算一个虚假目标点。当您输入一个所需位置时，系统会将其重新计算成一个虚假目标点，而该目标点经偏移后便能处于所需位置。

xx0300000226

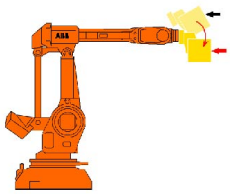
下一页继续

3.1.5.2 Absolute Accuracy补偿

续前页

已补偿的位置

实际位置将与您所需的位置如出一辙。从用户角度来说，您既不会注意到虚假目标点，也不会注意到偏移。机器人的一举一动将如同完全没有偏移一样。



xx0300000224

3.1.6 Absolute Accuracy机器人的准备

3.1.6.1 ABB校准进程

概述

本节描述了交付机器人前ABB公司在每台Absolute Accuracy机器人（不论是哪种类型或哪一系列的机器人）上执行的校准进程。

此进程可分为四步：

1 旋转变压器偏移校准
2 Absolute Accuracy校准
3 保存在系列测量板上的校准数据
4 Absolute Accuracy验证
5 生成出厂证书

旋转变压器偏移校准

通过此旋转变压器偏移校准进程来校准相应的旋转变压器偏移参数。

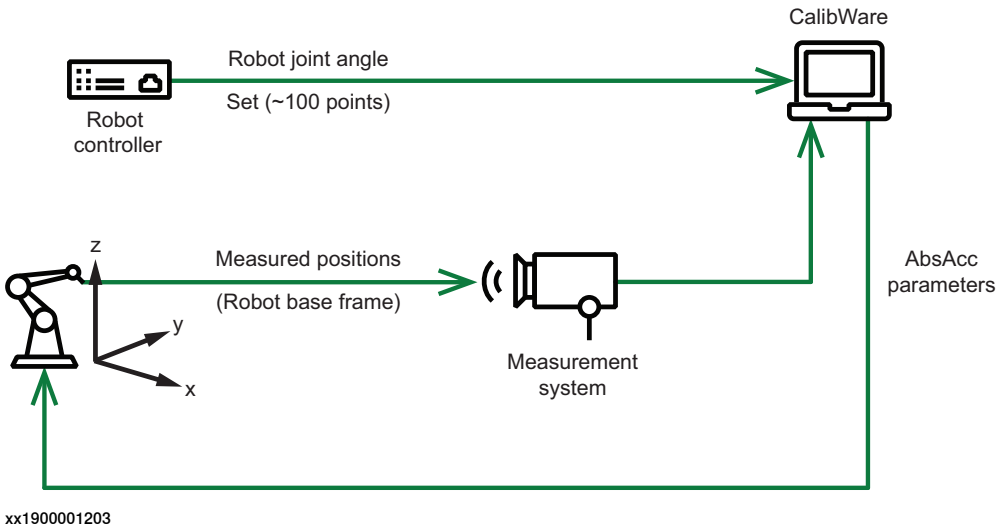
各台机器人的产品手册描述了具体做法。

Absolute Accuracy校准

由于方法的可重复性对两种进程都具有重要意义，因此在旋转变压器校准时要首先进行Absolute Accuracy校准。

按最大负载来校准每台机器人，从而确保检测到正确的补偿参数（若按低负载校准，则可能无法恰当确定机器人的灵活性参数）该进程会让机器人运行100种联合目标点姿态，并测量每个对应测量点的坐标。请向CalibWare校准核心输入各种姿态和测量的清单，然后创建一套机器人补偿参数。

CalibWare的文件描述了这方面的具体做法。



xx1900001203

Absolute Accuracy验证

将这些参数载入控制器，然后将它们激活。机器人随后会运行一套共 50 种的机器人目标点姿态。系统会测量每种姿态，然后标定相应偏差。

下一页继续

3.1.6.1 ABB校准进程
续前页

CalibWare的文件描述了这方面的具体做法。

验收要求因机器人类型而异，请参阅各个机器人产品规范中的典型性能数据。

补偿参数与出厂证书

这些补偿参数会保存在该机器人的系列测量板上（具体请参见第*166*页的配置参数）。

创建一份出厂证书，其代表了校准与验证序列所用的Absolute Accuracy测量协议（具体请参见第*165*页的出厂证书）。

3.1.6.2 出厂证书

关于出厂证书

所有Absolute Accuracy机器人都随附了一份出厂证书，其代表了校准与验证序列所用的Absolute Accuracy测量协议。

出厂证明包含以下信息：

- 机器人信息（机器人类型、序列号、Absolute Accuracy的版本）
- 准确度信息（精确点误差分布的最大偏差、平均偏差和标准偏差）
- 工具信息（TCP、质量和重心）
- 对测量协议的描述（测量与校准系统、点数和测量点的位置）

3.1.6.3 配置参数

关于补偿参数

所有Absolute Accuracy机器人都随附一套补偿参数，作为系统参数（配置）的一部分。由于Absolute Accuracy校准中的旋转变压器偏移校准浑然一体，因此旋转变压器偏移参数也保存在该机器人的系列测量板上。

补偿参数

按下列配置类型来定义补偿参数：

- ROBOT_CALIB
- ARM_CALIB
- JOINT_CALIB
- PARALLEL_ARM_CALIB
- TOOL_INTERFACE
- MOTOR_CALIB

ROBOT_CALIB类型定义了校准结构的顶层。*r1_calib*实例通过指定标志*-absacc*来激活Absolute Accuracy功能。请参阅第*156*页的激活*Absolute Accuracy*。

系统保留ARM_CALIB、JOINT_CALIB、PARALLEL_ARM_CALIB和MOTOR_CALIB类型，它们只有在安装管理器中选定Absolute Accuracy选项时才会显示。可以通过导入一个新配置文件来更改参数值。

补偿参数包含在*moc.cfg*文件中的备份中。

3.1.7 围笼的对准

3.1.7.1 概述

关于围笼的对准

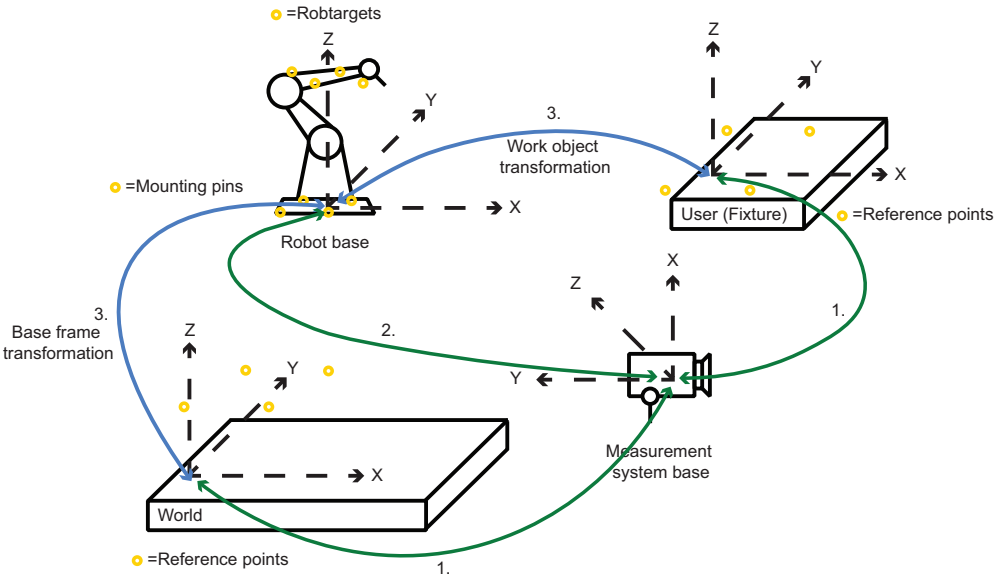
系统会确定Absolute Accuracy机器人的相关补偿参数（从物理基板到机器人工具）。对许多应用而言，这已足以像其它机器人那样来使用该机器人。不过Absolute Accuracy机器人往往会对准其围笼中的坐标，而本节就描述了这种无返回值对准程序。更详细的说明则请参见CalibWare的文件。

无返回值对准程序

要想让机器人在整个机器人围笼内都保持准确，就得正确安装该机器人。简而言之，这涉及到：

	操作	描述
1	测量固定装置的对准情况	确定相关测量系统与相关固定装置之间的关系。具体请参见第*168*页的测量固定装置的对准情况。
2	测量机器人的对准情况	确定相关测量系统与相关机器人之间的关系。具体请参见第*169*页的测量机器人的对准情况。
3	计算框架关系	确定相应的关系（比如相关机器人与相关固定装置之间的关系。具体请参见第*170*页的框架关系。
4	校准工具	确定相关机器人工具与其它围笼部件之间的关系。具体请参见第*171*页的工具校准。

图示



en0300000239

3.1.7.2 测量固定装置的对准情况

关于固定装置的对准情况

将一个固定装置定义为与特定坐标系有关的一种围笼部件。为了确保Absolute Accuracy，需要有一种准确的关系来实现机器人与固定装置之间的相互作用。

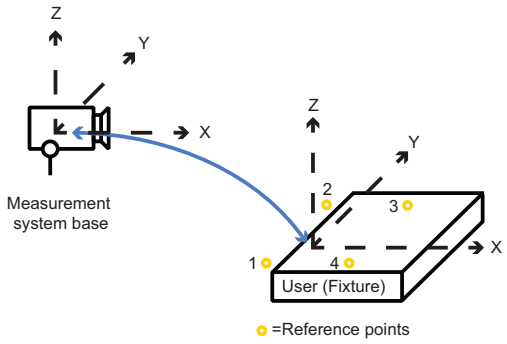
Absolute Accuracy固定装置必须至少有三个（四个更好）参考点，且明确标注每个点的位置信息。

固定装置的无返回值测量程序

按下列步骤来对准固定装置：

1 在对准软件中输入参考点的名称和位置。
2 测量相关参考点，并赋予相同的名称。
3 用对准软件来匹配相关引用项和所测点，然后确定关系框架。所有测量系统都支持这种转变格式。

图示



en0300000237

测量位置	参考位置	框架关系
Pos1: 100, 100, 200	Pos1: 100, 100, 100	1）RobotStudio 工作对象 (0,0,-100,0,0,0) (x,y,z,roll,pitch,yaw)
Pos2: 100, 200, 200	Pos2: 100, 200, 100	
Pos3: 200, 200, 200	Pos3: 200, 200, 100	
Pos4: 200, 100, 200	Pos4: 200, 100, 100	

3.1.7.3 测量机器人的对准情况

选择方法

可通过以下途径来确定测量系统与机器人之间的关系：

无返回值对准程序	描述
对准物理基座	相当于按“根据各个机器人产品手册中详述的参考位置来测量和对准物理基座销”这一方式来对准此固定装置。
对准理论基座	测量若干种机器人姿态，并用对准软件来确定该机器人的对准情况。

对准物理基座

像固定装置那样对准机器人的优点在于简单易行——将相关机器人视为围笼中的另一个固定装置，然后相应测量其基点即可。而这种做法的缺点在于：之后将机器人放到各销上时若存在小错，则可能因机器人的伸展范围而形成较大的TCP错误（即未校准机器人的放置情况）。

若要确定参考点的坐标，则需查阅此类机器人的产品手册。

一旦测得正确的点，系统就会用对准软件来确定相关测量系统与机器人基座之间的框架关系。

对准理论基座

让机器人对准理论基座的优点在于：可消除安装机器人时产生的一切错误，此外对准进程中还会详述所测点的机器人准确度，从而确认正确的Absolute Accuracy功能。这种做法的缺点在于：必须创建一则机器人程序（用CalibWare手动或自动创建），并对机器人进行测量（理想情况是使用正确工具进行测量，但也可把校准TCP作为这则无返回值程序的一部分）。

一旦测得正确的点，系统就会用对准软件来确定相关测量系统与机器人基座之间的框架关系。

3.1.7.4 框架关系

关于框架关系

一旦测得相关测量系统与其它所有围笼部件之间的关系，就能确定各围笼部件之间的关系。

全局坐标系和相关机器人之间的关系应保存在机器人基座中。相关机器人与相关固定装置之间的关系则应保存在workobject数据类型中。

由于以相对于测量系统的方式测量了全局和机器人，因此该测量系统最初是处于激活状态的相应坐标系。

确定机器人基座

用一套标准测量系统软件来确定全局坐标中的机器人基座：

1 将全局坐标系设置成激活状态（原点）。

2 读取机器人基本框架的坐标（此时是相对于全局的坐标）。

将该机器人设置成激活状态，然后读取固定装置框架的坐标，从而以类似方式确定固定装置的关系。

3.1.7.5 工具校准

关于工具校准

Absolute Accuracy机器人补偿参数会被计算为独立工具，这将使所有正确预定义了TCP的工具都与机器人法兰连接起来，且无需重新校准它们就能使用这些工具。不过由于存在未考虑到工具与机器人间的连接或工具灵活性等问题，因此实际上难以用Coordinate Measurement Machine (CMM)等来进行正确的TCP校准。

为了确保机器人具有最佳准确度，用户最好定期校准每件工具。

无返回值工具校准程序

下文详述了建议采用的无返回值工具重新校准程序：

- SBCU (Single Beam Calibration Unit）比如针对电弧焊应用或点焊应用的ABBBullsEye。
- 4、5或6点（控制器上可用的工具中心点校准例程）等几何校准。可用一套测量系统来确保所用的单个点是准确的。
- RAPID工具校准例程：MToolTCPCalib（校准移动工具的TCP）、SToolTCPCalib（校准固定工具的TCP）、MToolRotCalib（校准移动工具的旋度）以及SToolRotCalib（校准固定工具的TCP和旋度）。
- 使用理论数据，比如来自CAD模型的理论数据。

提示

由于Absolute Accuracy型号使用了相应的工具负载特点，因此至关重要的一点就是所有参数都要尽量准确。可用Load Identification来高效地确定各项工具负载特点。

3.2 Advanced Robot Motion 3100-1

关于Advanced Robot Motion

您可通过*Advanced Robot Motion*选项来访问：

- *Advanced Shape Tuning*方面的信息请参见第*173*页的*Advanced Shape Tuning [包含在 3100-1 中]*。
- 用RAPID更改*Motion Process Mode*，具体请参见第*180*页的*Motion Process Mode [包含在 3100-1 中]*。
- *Wrist Move*方面的信息请参见第*187*页的*Wrist Move [包含在 3100-1 中]*。

3.3 Advanced Shape Tuning [包含在 3100-1 中]

3.3.1 关于Advanced Shape Tuning

目的

*Advanced Shape Tuning*的作用是减少机器人关节摩擦产生的路径偏差。

*Advanced Shape Tuning*有助于小圆环等的低速切割（10到100毫米 / 秒）——这种情况下机器人的关节摩擦通常会带来0.5毫米的路径偏差。通过微调控制器中摩擦模型的参数，系统可将这种路径偏差减少到该机器人的可重复性水平上（比如中型机器人为0.1毫米）。

其中包括

RobotWare选项和*Advanced robot motion*中都包括了*Advanced Shape Tuning*，您可由此访问：

- 指令`FricIdInit`、`FricIdEvaluate`和`FricIdSetFricLevels`，这些指令会自动优化已编程路径所需的关节摩擦模型参数。
- 用于手动微调关节摩擦参数的系统参数*Friction FFW On*, *Friction FFW level*和*Friction FFW Ramp*。
- 微调类型`tune_fric_lev`和`tune_fric_ramp`，可搭配指令`TuneServo`使用。

基本方法

此处简述了最常见的Advanced Shape Tuning用法：

1 将系统参数*Friction FFW On*设置成TRUE。具体请参见第*177*页的系统参数。

2 用指令`FricIdInit`和`FricIdEvaluate`进行关节摩擦等级的自动微调。具体请参见第*174*页的自动微调摩擦。

3 用指令`FricIdSetFricLevels`来补偿相关摩擦。

3.3.2 自动微调摩擦

关于自动微调摩擦

用指令FricIdInit和FricIdEvaluate可自动微调一台机器人的关节摩擦等级。这些指令会微调每个关节在特定移动序列中的摩擦等级。

通过指令FricIdSetFricLevels把自动微调等级应用到摩擦补偿上。

程序执行

若要对一段移动序列做自动微调，该序列就必须从指令FricIdInit开始，到指令FricIdEvaluate结束。当程序执行到FricIdEvaluate时，机器人将重复该移动序列，直至找到每个关节轴的最佳摩擦等级为止。每次反复都包括一次后退运动和一次前进运动（均沿所编程的路径运动）。为了得出正确的关节摩擦等级，该序列通常必须重复20到30次才行。

当程序指针处在指令FricIdEvaluate上时，若相关程序因故停止执行并随之重启，那么所得结果就无效。这样一来，停止后就必须从开头处重启摩擦识别。

一旦找到正确的摩擦等级，就必须用指令FricIdSetFricLevels来设置它们，否则系统就不会使用它们。注意是针对FricIdInit与FricIdEvaluate之间的特定移动来微调相应的摩擦等级。至于机器人工作区域中的其它地方，则需另做一次微调来获取正确的摩擦等级。

这些指令方面的详细说明请参见技术参考手册 - *RAPID*指令、函数和数据类型。

限制

摩擦微调存在以下限制：

- 摩擦微调无法与同步移动组合起来，即是说不允许在FricIdInit与FricIdEvaluate之间实现SyncMoveOn。
- 针对所作摩擦微调的移动序列必须从一个精确点开始，到一个精确点结束。若非如此，系统则会在微调期间自动插入精确点。
- 自动微调摩擦仅对TCP机器人有用。
- 一次只能对一台机器人进行自动微调关节摩擦。
- 最多可微调至500%。如果这还不够，那么就为参数*Friction FFW Level*设置更大的值。具体请参见第*178*页的用估计值启动。
- 自动微调摩擦时将无法用TuneMaster看到任何测试信号。
- FricIdInit与FricIdEvaluate之间的移动序列不能超过10秒。

> **注意**
>
> 若要使用Advanced Shape Tuning，就必须将参数*Friction FFW On*设置成TRUE。

示例

此例展示了如何编写一条封装了摩擦微调的切割指令。系统会在首次执行该指令时（没有计算出的摩擦参数）微调摩擦。在微调期间，机器人将沿着所编程的路径来回移动。这种动作大约需要重复25次。

下一页继续

在所有后续执行过程中，相关的摩擦等级都会被设置成第一次执行时确认的微调值。
可用指令CutHole来单独微调每个孔的摩擦。

```
PERS num friction_levels1{6} := [9E9,9E9,9E9,9E9,9E9,9E9];
PERS num friction_levels2{6} := [9E9,9E9,9E9,9E9,9E9,9E9];

CutHole p1,20,v50,tool1,friction_levels1;
CutHole p2,15,v50,tool1,friction_levels2;

PROC CutHole(robtarget Center, num Radius, speeddata Speed, PERS
     tooldata Tool, PERS num FricLevels{*})
  VAR bool DoTuning := FALSE;

  IF (FricLevels{1} >= 9E9) THEN
    ! Variable is uninitialized, do tuning
    DoTuning := TRUE;
    FricIdInit;
  ELSE
    FricIdSetFricLevels FricLevels;
  ENDIF

  ! Execute the move sequence
  MoveC p10, p20, Speed, z0, Tool;
  MoveC p30, p40, Speed, z0, Tool;

  IF DoTuning THEN
    FricIdEvaluate FricLevels;
  ENDIF
ENDPROC
```

注意

真实的程序中将包括“在微调阶段前停用切割设备”。

3.3.3 手动微调摩擦

概述

可以手动微调一台机器人的关节摩擦（而不是自动微调摩擦）。可用指令TuneServo来微调每个关节的摩擦等级。本节描述了其具体做法。

通常无需更改摩擦增减率。

> **注意**
>
> 若要使用Advanced Shape Tuning，就必须将参数*Friction FFW On*设置成TRUE。

微调类型

将一种微调类型作为指令TuneServo的一个自变数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的微调类型。

Advanced Shape Tuning显然使用了两种微调类型：

微调类型	描述
TUNE_FRIC_LEV	在用自变数TUNE_FRIC_LEV调用指令TuneServo后，便可在执行程序时调节一个机器人关节的摩擦等级。为参数*Friction FFW Level*所定义的摩擦等级给出一个百分数（1到500之间）。
TUNE_FRIC_RAMP	在用自变数TUNE_FRIC_RAMP调用指令TuneServo后，便可在执行程序时调节完全摩擦补偿下所能达到的电机轴速。为参数*Friction FFW Ramp*所定义的摩擦等级给出一个百分数（1到500之间）。 通常无需微调摩擦坡。

配置摩擦等级

设置每个机器人关节的摩擦等级。一次对一个关节采取以下步骤：

	操作
1	通过让机器人运行其任务最急需的部分（最高级的形状）来测试该机器人。如果应使用机器人进行切割，则在测试中使用与生产时相同的工具来进行切割。 观测路径偏差，并测试是否需要增加或减少各个关节摩擦等级。
2	用RAPID指令TuneServo和微调类型TUNE_FRIC_LEV来微调相应的摩擦等级。*Friction FFW Level*值提供了相关等级的百分数。 示例：将摩擦等级增大20%的指令如下： `TuneServo MHA160R1, 1, 120 \Type:= TUNE_FRIC_LEV;`
3	重复步骤1和2，直到您对路径偏差满意为止。
4	可将最终微调值传输给相应的系统参数。 示例：*Friction FFW Level*为0.5，最终微调值（TUNE_FRIC_LEV）为120%。将*Friction FFW Level*设置成0.6，将微调值设置成100%（默认值）——两者是等同的。

> **提示**
>
> 最多可微调至500%。如果这还不够，那么就为参数*Friction FFW Level*设置更大的值。具体请参见第*178*页的设置微调系统参数。

3.3.4 系统参数

3.3.4.1 系统参数

关于系统参数

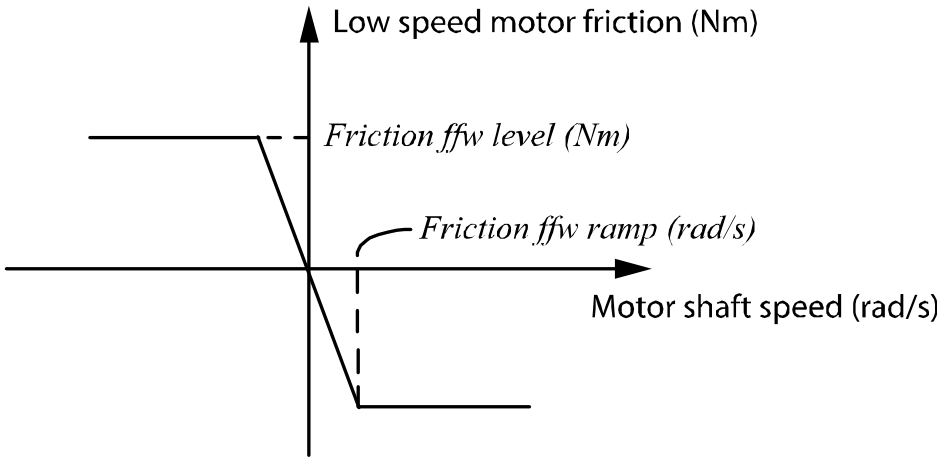
此处简述了*Advanced Shape Tuning*选项中的每个参数。更多信息请参见技术参考手册 - 系统参数中的各个参数。

Friction Compensation / Control Parameters

除机器人IRB 1400与IRB 1410（它们属于主题*Motion*下的类型*Control Parameters*）外，这些参数均属于主题*Motion*下的类型*Friction Compensation*。

参数	描述
Friction FFW On	当*Friction FFW On*被设置成TRUE时，Advanced Shape Tuning便会处于激活状态。
Friction FFW Level	*Friction FFW Level*是相关机器人关节的摩擦等级。参见下图。
Friction FFW Ramp	*Friction FFW Ramp*为摩擦力达到*Friction FFW Level*定义的摩擦等级时的机器人电机轴速度。参见下图。 通常无需更改*Friction FFW Ramp*。

图示



en0900000117

3.3.4.2 设置微调系统参数

自动微调时很少需要更改系统参数

对自动微调而言，如果将相应的摩擦等级保存在永久数组中，则可在上电失败后保留微调项。此外自动微调还能为不同的机器人移动序列设置不同的微调等级（系统参数则做不到这一点）。在使用自动微调时，除非默认值几乎都被“关闭”，否则无需更改相关系统参数。具体请参见第178页的用估计值启动。

将微调项传输给系统参数

使用手动微调时，相关微调值会被重置为上电失败时的默认项（100%）。不过系统参数设定是永久性的。

如果采用临时微调，则仅在执行一部分程序时才会有效，且不宜向外传输。

若要将摩擦等级微调值（TUNE_FRIC_LEV）传输给参数*Friction FFW Level*，则请遵循下列步骤：

	操作
1	在RobotStudio中打开配置编辑器（Motion主题），然后选择类型Friction comp（机器人IRB 1400与IRB 1410除外——它们属于类型Control parameters）。
2	用*Friction FFW Level*乘以相应的微调值。将该值设置成新的*Friction FFW Level*，然后将该微调值（TUNE_FRIC_LEV）设置成100%。 示例：*Friction FFW Level*为0.5，最终微调值（TUNE_FRIC_LEV）为120%。将*Friction FFW Level*设置成0.6（1.20×0.5），将微调值设置成100%（默认值）——两者是等同的。
3	重启控制器，从而使所做改动生效。

用估计值启动

参数*Friction FFW Level*将作为微调启动值。如果该值与正确值相距甚远，那么可能就无法将其微调成正确值。不过*Friction FFW Level*大多数情况下都会被默认设置成一个基本正确的数值，因此不太可能发生这种情况。

如果*Friction FFW Level*出于某种原因而与正确值相距甚远，则可将其改为一个新估计值。

	操作
1	在RobotStudio中打开配置编辑器（Motion主题），然后选择类型Friction comp（机器人IRB 1400与IRB 1410除外——它们属于类型Control parameters）。
2	将参数*Friction FFW Level*设置成一个估计值。请勿将该数值设置成0（零），不然将无法进行微调。
3	重启控制器，从而使所做改动生效。

3.3.5 RAPID组件

关于RAPID部件

此处概述了*Advanced Shape Tuning*中的所有指令、函数和数据类型。
有关更多信息，请参见技术参考手册 - *RAPID*指令、函数和数据类型。

指令：

指令：	描述
FricIdInit	发起摩擦识别
FricIdEvaluate	评价摩擦识别
FricIdSetFricLevels	在摩擦识别后设置摩擦等级

函数

*Advanced Shape Tuning*中不包括任何函数。

数据类型

*Advanced Shape Tuning*中不包括任何数据类型。

3.4 Motion Process Mode [包含在 3100-1 中]

3.4.1 关于Motion Process Mode

目的

Motion Process Mode的作用是简化专用微调，也就是优化特定应用中的机器人性能。对大多数应用而言，默认模式就是最佳模式。

可用的运动进程模式

运动进程模式由一套特定的机器人微调参数组成。每套微调参数（也就是每种模式）都会针对特定的应用等级来优化机器人的微调。

预定义了下列模式：

- *Optimal cycle time mode* – 此模式产生可能的最短循环时间，通常是默认模式。
- *Accuracy mode* – 此模式提高了路径准确度。相较*Optimal cycle time mode*，循环时间将稍稍增加。这是提高小型和中型机器人（比如，IRB 2400和IRB 2600）路径准确度的建议选项。
- *Low speed accuracy mode* – 此模式提高了路径准确度。相较*Accuracy mode*，循环时间将稍稍增加。这是提高大型机器人（比如，IRB 4600）路径准确度的建议选项。
- *Low speed stiff mode* – 建议在最大伺服器刚性具有重要意义的接触应用中使用该模式。此外也可用于某些想尽量减少路径波动的低速应用。该模式的周期时间要久于*Low speed accuracy mode*。
- *Press tending mode* – 更改*Kv Factor*、*Kp Factor*和*Ti Factor*，从而减轻工具振动。此模式主要用于按压应用，在该应用中会用到沿y方向大幅扩展的灵活夹具。
- *Collaborative mode* – 对于机器人应平稳运行的协作应用程序，建议使用此模式。与最佳循环时间模式相比，循环时间将增加。这只会对 GoFA CRB 15000 产生任何影响。

也有四种模式可供应用程序特定用户调整使用：

- *MPM User mode 1 – 4*

模式选择

系统会自动选择相应的的默认模式，不过用户可在类型*Robot*的系统参数*Use Motion Process Mode*中更改此模式。

只有当安装了选项*Advanced Robot Motion*时，才能用RAPID来更改*Motion Process Mode*。只有当机器人直立不动时才能更改该模式，否则就会强制使用一个精确点。

下例展示了RAPID指令MotionProcessModeSet的典型用法。

```
MotionProcessModeSet OPTIMAL_CYCLE_TIME_MODE;
! Do cycle-time critical movement
MoveL *, vmax, ...;
...

MotionProcessModeSet ACCURACY_MODE;
! Do cutting with high accuracy
```

下一页继续

```
MoveL *, v50, ...;
...
```

限制

- *Motion Process Mode*方案目前可用于所有六轴和七轴机器人，但上漆机器人除外。
- *Mounting Stiffness Factor*仅能用于以下机器人：

  IRB 120, IRB 140, IRB 1200, IRB 1520, IRB 1600, IRB 2600, IRB 4600, IRB 6620 （非LX）, IRB 6640, IRB 6700.
- 对于IRB 1410，只有*Accset*和几何准确度参数可用。
- 下列机器人模型不支持使用*World Acc Factor*（即，只允许*World Acc Factor = -1*）：

  IRB 340, IRB 360, IRB 540, IRB 1400, IRB 1410

3.4.2 用户定义的模式

可用的微调参数

如果需要更多特定微调，则可修改每种运动进程模式中的一些微调参数。可以修改预定义模型和用户模型。用户可按这种方式为特定应用创建特定微调。

下表对可用的微调参数作了简短说明。

- *Use Motion Process Mode Type* - 选择用户模式下的预定义参数。
- *Accset Acc Factor* – 更改加速度
- *Accset Ramp Factor* – 更改加速度增减率
- *Accset Fine Point Ramp Factor* – 更改精确点的减速度增减率
- *Joint Acc Factor* - 修改特定关节的加速。
- *World Acc Factor* - 如为正，则激活动态世界加速，典型值为1，如为-1则停止。
- *Geometric Accuracy Factor* - 如果减少，则增加geometric精确度。
- *Dh Factor* – 更改路径的平顺度（有效的系统带宽）
- *Df Factor* – 更改某一根轴的预测共振频率
- *Kp Factor* – 更改某一根轴的位置控制器的等效增益
- *Kv Factor* – 更改某一根轴的速度控制器的等效增益
- *Ti Factor* – 更改某一根轴的积分时间
- *Mounting Stiffness Factor X* – I / O描述了x方向上的机器人底座刚度
- *Mounting Stiffness Factor Y* – I / O描述了y方向上的机器人底座刚度
- *Mounting Stiffness Factor Z* – I / O描述了z方向上的机器人底座刚度

详细说明请参阅技术参考手册 - 系统参数中的*Motion Process Mode*。

用RAPID微调各个参数

大部分参数也可以使用TuneServo和AccSet指令修改。

> **注意**
>
> 所有参数设定都是在相对于预定义参数值进行调节。虽然运动进程模式和TuneServo/Accset指令可以一起使用，但我们还是建议要么选择运动进程模式，要么选择TuneServo/AccSet。

例 1

加速度的相对调节 = 【预定义*AccSet Acc Factor*】*【*AccSet Acc Factor*】*【*AccSet* 指令加速度系数 / *100*】

例 2

Kv的相对调节值 = 【预定义*Kv Factor*】*【*Kv Factor*】*【*TuneServo(TYPE_KV)* 指令的微调值 / *100*】

预定义参数值

如果机器人的类型不同，那么每种模式的预定义参数值也不相同。

对*Optimal cycle time mode*而言，通常所有预定义参数都会被设置成1.0。

下一页继续

对*Low speed accuracy mode*和*Low speed stiff mode*而言，系统会以降低AccSet和Dh参数的方式来提高移动的平顺度和路径的准确度，同时以更改*Kv Factor*、*Kp Factor*和*Ti Factor*的方式来提高伺服器的刚度。

某些机器人可能无法增大*Low speed accuracy mode*和*Low speed stiff mode*中的*Kv Factor*。在调节*Kv Factor*时，请始终小心行事和观察增大后的电机噪声等级，且采用的数值请勿超过达到相关应用要求时所需的数值。若*Kp Factor*太高或*Ti Factor*太低，则都会因机械共振而使振动加剧。

*Accuracy Mode*使用动态世界加速限制(*World Acc Factor*)且增加了地理准确性(*Geometric Accuracy Factor*)以提升路径准确性。

由于*Df Factor*和*Mounting Stiffness Factors*的最佳值取决于具体的安装情况（比如安装机器人的底座的刚度），因此预定义模式下的这些参数会始终被设置成1.0。可用*TuneMaster*来优化这些参数。用户可在*TuneMaster*应用中找到更多信息，此外还要注意*Mounting Stiffness Factor*的限制。

警告

若*Motion Process Mode*参数设定有误，则可能造成振荡移动或扭矩，从而对机器人造成损伤。

3.4.3 关于机器人微调的一般信息

尽量缩短周期时间

若要获得最好的周期时间，则宜采用运动进程模式*Optimal cycle time mode*。该模式通常为默认模式，用户仅需定义工具负载、有效负载和臂负载（若有）即可。一旦编写好机器人路径，*ABB QuickMove*运动技术就会自动计算该路径上的最佳加速度和最佳速度，从而得出周期时间最短的时间优化型路径，于是便无需对加速度进行微调。改善周期时间的唯一途径是更改相关路径的几何结构或处理工作空间的其它区域。若需进行此类优化，可通过RobotStudio中的模拟来开展此类优化。

增加路径准确度和减少振动

对大多数应用来说，*Optimal cycle time mode*可令路径准确度和振动方面的行为达到令人满意的程度，而其中依靠的就是*ABB TrueMove*运动技术。不过某些应用的确需通过修改机器人的微调来改进准确度。本文之前已用RAPID程序中的`TuneServo`和`AccSet`指令做了这种微调。

动作处理模式的概念将简化此应用程序的微调，且四个预定义模式在很多情况下应该都会很有用，无需进一步调整。

下文就如何解决准确度问题提出了一些通用建议（假设用户已测试过默认选择*Optimal cycle time mode*，并注意到了各种准确度问题）：

1. 验证是否恰当定义了工具负载、有效负载和臂负载。
2. 将检查工具和工艺设备连接在机器人的臂上。确保所有东西均已固定牢靠，且相关工具具有足够的硬度。
3. 检查安装机器人的底座，具体请参见第*184*页的补偿底座的灵活性。

补偿底座的灵活性

如果相关底座未达到机器人产品手册中的刚度要求，那么就宜对底座灵活性进行补偿。参见机器人产品手册中的节“底座要求，最小共振频率”。

要么通过轴1和轴2的*Df Factor*实现，要么通过*Mounting Stiffness Factor*实现（取决于机器人的类型）。具体请参见第*186*页的限制。

TuneMaster的作用是查找*Df Factor* / *Mounting Stiffness Factor*的最佳值。系统随后会为使用的*Motion Process Modes*定义所得的*Df Factor* / *Mounting Stiffness Factor*。

注意

如果底座达不到相关要求，那么即使做了所述补偿，也仍不免在一定程度上削弱准确度。如果底座硬度很差，那么可能就无法用*Df Factor* / *Mounting Stiffness Factor*来解决相关问题。

这种情况下必须改善底座或采用以下解决方案之一，比如ith a low 低*Dh Factor*的*Optimal cycle time mode*、*Accset Acc Factor*或*Accset Fine Point Ramp Factor*（具体请参见取决于相关应用）。

警告

在安装刚度极低的情况下，若微调不当，则可能导致移动或扭矩方面的振荡，进而对机器人造成损伤。

下一页继续

若准确度仍需改善

- 对于，例如切割，应该使用*Advanced Shape Tuning* and *Accuracy mode/Low speed accuracy mode*。动作模式同时取决于机器人类型与具体的应用。总的来说，对小型和中型机器人推荐使用*Accuracy mode IRB 2400/2600*)，而*Low speed accuracy mode* 则推荐对大机器人使用。
- 如果路径准确性仍需提高，则可以用微调参数来调节准确性模式，以下为部分示例：
  - 调节*Accuracy mode*实现更好的准确性：

    1) 减少*World Acc Factor*，例如从1到0.5。

    2) 减少*Dh Factor*到0.5或更低。注意低值*Dh factor*可能会改变高速下的角区域。
  - 调节*Low speed accuracy mode*实现更好的准确性：

    1) 设置*World Acc Factor*为1，并设置*Geometric Accuracy Factor*为0.1。

    2) 减少*Dh Factor*到0.5或更低。
- 编程设定的速度有时必须降低，以实现可能范围内的最好准确性（如在切削应用中）。例如，半径1 mm的圆圈不应编程设定高于20 mm/s的速度。
- 对于接触类应用，例如铣削和预压，推荐使用*Low speed stiff mode*此模式也对某些低速应用中的大机器人适用（最大100 mm/s），其中对最小路径波动有要求（如小于0.1 mm）。注意此模式的伺服调节非常刚性，且有些情况下*Kv Factor*可能由于电机震动和噪音需要降低。
- 如果需要减少精确点的过界和振动，则请采用*Optimal cycle time mode*，同时减少*Accset Fine Point Ramp Factor*或*Dh Factor*的值，直至问题得到解决为止。
- 如果在开始或结束重定方位时发生了准确度问题，则用更大的`pzone_ori`和`pzone_eax`来定义一个新区。即使系统中没有外轴，其数值也最好始终相同。同时也增大`zone_ori`。编程时请始终力求能平顺地重定方位。
- 最后要说的是，结束准确度微调后若需减少周期时间，则在RAPID程序的不同段中使用不同的运动进程模式。

3.4.4 附加信息

与TuneServo和AccSet相比的Motion Process Mode

运动进程模式简化了专用微调，并使用户能通过系统参数——而不是RAPID程序——来定义相关微调项。

总的来说，用户宜把运动进程模式作为解决准确度问题的首选。不过用户仍然能用RAPID程序中的TuneServo和AccSet指令来进行专用微调。

TuneServo和AccSet通常都不是最好的选择，不过也有例外，比如如果减少RAPID程序中某段的加速度就能准确度问题，并使也对周期时间做了优化，那么它们就是明智之选。在这种情况下，由于需要一个精确点来更改运动进程模式，因此更好的做法或许是使用无需精确点就能更改的AccSet。

限制

- *Motion Process Mode*方案目前可用于所有六轴和七轴机器人，但上漆机器人除外。
- *Mounting Stiffness Factor*仅能用于以下机器人：

  IRB 120, IRB 140, IRB 1200, IRB 1520, IRB 1600, IRB 2600, IRB 4600, IRB 6620 （非LX）, IRB 6640, IRB 6700.
- 对于IRB 1410，只有*Accset*和几何准确度参数可用。
- 下列机器人模型不支持使用*World Acc Factor*（即，只允许*World Acc Factor = -1*）：

  IRB 340, IRB 360, IRB 540, IRB 1400, IRB 1410

相关信息

信息，关于	请参阅
*Motion Process Mode*参数的配置。	技术参考手册 - 系统参数
RAPID指令： • AccSet – 减少加速度 • MotionProcessModeSet – 设置运动进程模式 • TuneServo – 微调伺服器	技术参考手册 - *RAPID*指令、函数和数据类型

3.5 Wrist Move [包含在 3100-1 中]

3.5.1 Wrist Move介绍

目的

*Wrist Move*的作用是改善切割小尺寸几何结构时的路径准确度。对小孔等几何形状而言，机器人主轴（1到3）产生的摩擦效应往往会削弱其形状的视觉外观。一条关键思路是不控制机器人的TCP，而是通过腕的移动来控制激光束（或水射流、布线芯轴等）与切割面之间的交叉点。用户只需两根腕轴即可控制该交叉点（而不是用上所有的机器人轴），于是便尽量减少了相关路径上的摩擦效应。由程序员决定使用哪一对腕轴。

使用Wrist Move

RobotWare选项*Advanced robot motion*中包括了*Wrist Move*。

Wrist Move和RAPID指令`CirPathMode`与各种移动指令一同用于圆弧作业，即`MoveC`、`TrigC`和`CapC`等。由指令`CirPathMode`配合旗标`Wrist45`、`Wrist46`或`Wrist56`之一来激活腕移动模式。在激活这种模式后，所有后续`MoveC`指令都将产生一次腕移动。若要返回正常的`MoveC`行为，则必须用`Wrist45`、`Wrist46`和`Wrist56`之外的一个旗标（比如`PathFrame`）来设置`CirPathMode`。

> **注意**
>
> 由于仅使用了两根轴，因此在腕移动期间，位于表面之上的TCP高度将不可避免地起伏不定。这种高度起伏将取决于机器人位置、工具定义和圆弧半径等等。半径越大，高度波动就越大。考虑到这种高度波动，这里建议一开始以极低速度来执行这种移动，以此验证高度波动是否会变得过大，否则就可能导致切割工具与被切割表面相撞。

限制

如果存在以下情况，则无法使用Wrist Move选项：

- 工件正在移动
- 机器人被安装在导轨上或其它正在移动的机械臂上

只有运行QuickMove的第二代机器人才支持Wrist Move选项。

切割时该工具便不会与相关表面保持正确夹角，所以用这种方法切割出的孔洞将稍微成圆锥形。这对薄板来说通常无碍，但厚板就可能呈现出明显的锥度。

在切割期间，位于表面之上的TCP高度将起伏不定。待切割形状的尺寸越大，这种高度波动就越大，从而限制了相关形状的潜在大小，进而——除碰撞风险外——又限制了激光束焦距或水射流等工艺特点。

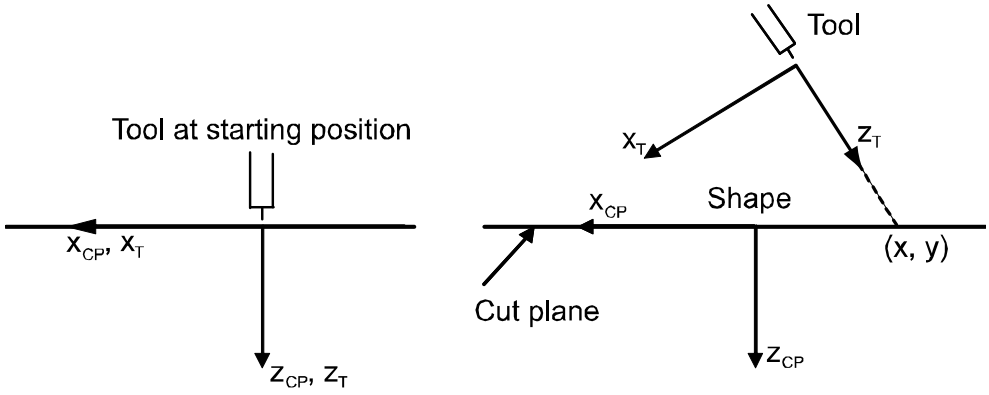
3.5.2 切割面框架

定义相关切割面框架

腕移动概念的关键在于切割面框架的定义（这种框架会提供对象表面的位置与方位信息）。在执行一条MoveC指令时，系统会按机器人的启动位置来定义该切割面框架，且该框架将被定义为“等同于启动位置处的工具框架”。注意对MoveC指令序列而言，整个序列中的切割面框架将始终都是相同的。

切割面插图

左图展示了如何定义相关切割面，右图则展示了切割期间的工具框架和切割面框架。



en0900000118

操作前提

鉴于定义切割面框架的方式，用户必须让启动位置满足以下条件：

- 该工具必须与相关表面保持正确夹角
- 该工具的z轴必须与激光束或水射流相互重合
- TCP必须尽量靠近相关表面

如果未满足头两项要求，那么切割轮廓的形状就会受到影响，比如让一个圆形孔洞更像是椭圆形。TCP往往会被定义在水射流喷口等的前方数毫米处，因此第三项要求通常易于满足；不过若未满足第三项要求，则只会影响到所得圆弧的半径，即是说切割弧的半径不符合编程半径（若是直线段，则会影响到其长度）。

提示

在FlexPendant示教器的点动窗口中，有一个按钮能来让相关工具自动对准选定坐标框架。在开始腕移动时，用户可用此功能来确保相关工具与相关表面之间的夹角无误。

提示

腕移动不仅仅限于圆弧：如果MoveC的各目标点同为一轴，那么就能做出一条直线。

3.5.3 RAPID组件

指令

此处简述了Wrist Move中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型介绍中的相关描述。

指令	描述
CirPathMode	`CirPathMode`使用户能选择不同模式来重定圆形移动期间的工具方位。 自变数`Wrist45`、`Wrist46`和`Wrist56`专供Wrist Move选项使用。

3.5.4 RAPID代码示例

基本示例

此例展示了如何现用轴4和轴5、再用轴5和轴6做出两条圆弧。在做好两条圆弧后，CirPathMode便会停用腕移动。

```
! This position will define the cut plane frame
MoveJ p10, v100, fine, tWaterJet;

CirPathMode \Wrist45;
MoveC p20, p30, v50, z0, tWaterJet;

! The cut-plane frame remains the same in a sequence of MoveC
CirPathMode \Wrist56;
MoveC p40, p50, v50, fine, tWaterJet;

! Deactivate Wrist Movement, could use \ObjectFrame
! or \CirPointOri as well
CirPathMode \PathFrame;
```

高级示例

此例展示了如何用终点半径R和长度L+2R在腕移动中切出一道槽。具体请参见第191页的插图，*pSlot*和*wSlot*。该槽的起点和终点都在pSlot这一位置（左半圆的中心）。为了避免在机器人中引发振荡，系统会沿着半圆引入路径和半圆引出路径——两者与槽的轮廓平顺相连——来开始和结束切割过程。给出的所有坐标都是相对于工件wSlot的坐标。

```
! Set the dimensions of the slot
R := 5;
L := 30;

! This position defines the cut plane frame, it must be normal
! to the surface
MoveJ pSlot, v100, z1, tLaser, \wobj := wSlot;
CirPathMode \Wrist45;

! Lead-in curve
MoveC Offs(pSlot, R/2, R/2, 0), Offs(pSlot, 0, R, 0), v50, z0,
     tLaser, \wobj := wSlot;

! Left semi-circle
MoveC Offs(pSlot, -R, 0, 0), Offs(pSlot, 0, -R, 0), v50, z0, tLaser,
     \wobj := wSlot;

! Lower straight line, circle point passes through the mid-point
! of the line
MoveC Offs(pSlot, L/2, -R, 0), Offs(pSlot, L, -R, 0), v50, z0,
     tLaser, \wobj := wSlot;

! Right semi-circle
```

下一页继续

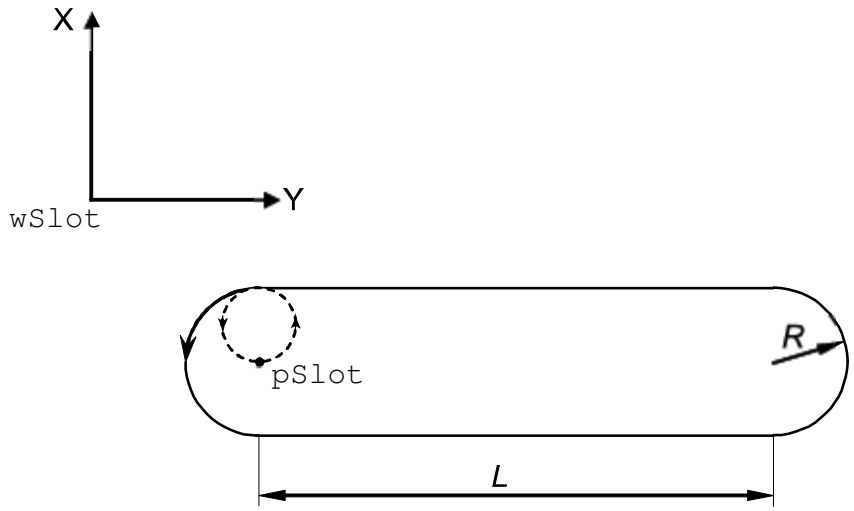
```
MoveC Offs(pSlot, L+R, 0, 0), Offs(pSlot, L, R, 0), v50, z0, tLaser,
     \wobj := wSlot;

! Upper straight line, circle point passes through the mid-point
! of the line
MoveC Offs(pSlot, L/2, R, 0), Offs(pSlot, 0, R, 0), v50, z0, tLaser,
     \wobj := wSlot;

! Lead-out curve back to the starting point
MoveC Offs(pSlot, -R/2, R/2, 0), pSlot, v50, z1, tLaser, \wobj :=
     wSlot;

Deactivate Wrist Movement
CirPathMode \ObjectFrame;
```

插图，pSlot和wSlot



xx0900000111

3.5.5 故障排除

意料之外的切割形状

如果得到的切割形状出乎意料，那么就检查以下事项：

- 该工具的z轴与激光束或水射流相互重合
- 在首次MoveC的启动位置处，该工具的z轴与相关表面之间的夹角无误
- 如果您拥有选项Advanced Shape Tuning，则请尝试对所涉腕轴进行摩擦微调。

错配半径

如果圆弧半径不符合编程半径，那么就检查启动位置处的TCP是否尽量靠近了相关表面。

无法靠选定的轴对移动

如果无法用选定轴对移动，那么就尝试用旗标Wrist45、Wrist46或Wrist56之一来激活另一轴对。若实在万不得已，则可尝试用另一套机器人配置来抵达启动位置。

4 Motion Supervision

4.1 World Zones [3106-1]

4.1.1 概述 World Zones

目的

World Zones的作用是在机器人位于用户专门定义的区域内时停止该机器人或设置一个输出信号。以下是一些应用示例：

- 当两台机器人的工作区域部分重叠时。可通过World Zones监控来安全地消除这两台机器人相撞的可能性。
- 当该机器人的工作区域内有某种永久性障碍或某些临时外部设备时。可创建一个禁区来防止机器人与此类设备相撞。
- 指明相关机器人正处在一个“允许用可编程逻辑控制器（PLC）来开始执行程序”的位置。

在程序执行期间和点动期间监控机器人移动时的一个全局区域。如果相关机器人的TCP触及该全局区域，或相关的轴触及了关节上的全局区域，那么就停止相关移动，并设置一个数字输出信号。

警告

出于安全考虑，用户不得使用本软件来保护人员——请使用硬件保护装备来保护人员。

其中包括

您可通过RobotWare选项World Zones来访问：

- 定义各种形状之体积的指令
- 在各轴坐标中定义关节区域的指令
- 定义和启用全局区域的指令

基本方法

这是设置World Zones的一般方式。第197页的代码示例用一个更详细的示例展示了其具体做法。

1. 声明该全局区域是固定的还是临时的。
2. 声明该形状变量。
3. 定义该全局区域应具有的形状。
4. 定义相应的全局区域（当达到相应体积时，系统就应停止相关机器人，或设置一个输出信号）。

限制

体积监控仅作用在TCP上，而机器人的任何其它部分都可能不知不觉地穿过这一体积。为了确保防止这种情况，您可对一个关节全局区域（由`WZLimJointDef`或`WZHomeJointDef`定义）进行监控。

下一页继续

4.1.1 概述 World Zones
续前页

无法重新定义类型为wzstationary或wztemporary的变量。这些变量只能定义一次（用WZLimSup或WZDOSet进行定义）。

当直通于活动状态时，World Zones 监督将无法访问。

4.1.2 RAPID组件

数据类型

此处简述了World Zones中的每种数据类型。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各种数据类型。

数据类型	描述
wztemporary	wztemporary的作用是识别临时全局区域，并可用在RAPID程序中的任何位置。 可通过RAPID指令来禁用、重新启用或擦除临时全局区域。当载入一段新程序时，或当从MAIN例程的起点处开始执行程序时，系统便会自动擦除临时全局区域。
wzstationary	wzstationary的作用是识别固定全局区域，并仅能用在与事件“通电”相关联的一则事件例程中。定义事件例程方面的信息请参见操作手册 - *OmniCore*。 固定全局区域会始终处于激活状态，而重启（先关闭电源然后再打开电源，或更改系统参数）则会再次激活此类区域。无法通过RAPID指令来禁用、启用或擦除固定全局区域。 如果涉及到安全问题，则应使用固定全局区域。
shapedata	shapedata的作用是描述一个全局区域的几何形状。 可将全局区域定义为4种不同的几何形状： • 一个方盒，所有侧面都与全局坐标系平行 • 一个圆柱体，与全局坐标系的z轴平行 • 一个球体 • 针对机器人轴和 / 或外轴的一个关节角区

指令：

此处简述了World Zones中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
WZBoxDef	WZBoxDef的作用是定义这样一种体积：该体积呈方盒形，且此“盒子”的每一侧都与全局坐标系的各轴平行。其定义会保存在一个类型为shapedata的变量中。 也可将该体积定义为此“盒子”的相反部分（所有体积都位于此“盒子”之外）。
WZCylDef	WZCylDef的作用是定义这样一种体积：该体积呈圆柱形，且此“圆柱体”的轴与全局坐标系的z轴平行。其定义会保存在一个类型为shapedata的变量中。 也可将该体积定义为此“圆柱体”的相反部分（所有体积都位于此“圆柱体”之外）。
WZSphDef	WZSphDef的作用是定义一种球形体积。其定义会保存在一个类型为shapedata的变量中。 也可将该体积定义为此“球体”的相反部分（所有体积都位于此“球体”之外）。
WZLimJointDef	WZLimJointDef的作用是定义各轴的关节坐标，从而对相关工作区域进行限制。机器人轴和外轴都可设置坐标限值。 WZLimJointDef定义了每根轴的上限和下限。对旋转轴来说，这些限值的单位为“度”；对线性轴来说，这些限值的单位为“毫米”。 其定义保存在一个类型为shapedata的变量中。

下一页继续

4.1.2 RAPID组件

续前页

指令	描述
WZHomeJointDef	WZHomeJointDef的作用是定义各轴的关节坐标，从而识别相关关节空间中的一个位置。机器人轴和外轴都可设置坐标限值。 就每根轴而言，WZHomeJointDef定义了该区域中点的关节坐标以及距离该中点的区域Δ偏差。对旋转轴来说，这些坐标的单位为“度”；对线性轴来说，这些坐标的单位为“毫米”。 其定义保存在一个类型为shapedata的变量中。
WZLimSup	WZLimSup的作用是定义和启用“在TCP抵达相应全局区域时停止相关机器人”，并届时附上一则错误消息。执行程序时和点动时都会激活这种监控。 在调用WZLimSup时，您可指定其是一个保存在wzstationary变量中的固定全局区域，还是一个保存在wztemporary变量中的临时全局区域。
WZDOSet	WZDOSet的作用是定义和启用“在TCP抵达相应全局区域时设置一个数字输出信号”。 在调用WZDOSet时，您可指定其是一个保存在wzstationary变量中的固定全局区域，还是一个保存在wztemporary变量中的临时全局区域。
WZDisable	WZDisable的作用是禁用对某个临时全局区域的监控。
WZEnable	WZEnable的作用是重新启用对某个临时全局区域的监控。 当创建一个全局区域时，系统会自动启用该区域。只有在用WZDisable禁用此类区域的情况下，才有必要使用“启用”。
WZFree	WZFree的作用是禁用和擦除对某个临时全局区域的监控。

函数

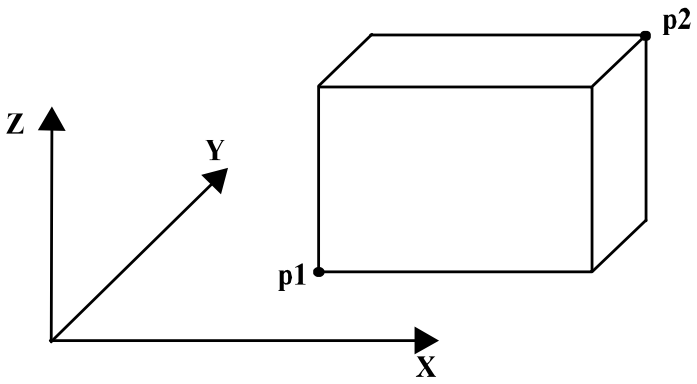
World Zones不包括任何RAPID函数。

4.1.3 代码示例

创建受保护的盒子

为了避免相关的机器人TCP移入固定设备中，请围绕该设备设置一个固定全局区域。

然后则宜把例程my_power_on与事件“通电”关联起来。至于具体的做法，则请阅读操作手册 - *OmniCore*中的事件例程定义。



xx0300000178

```
VAR wzstationary obstacle;
PROC my_power_on()
  VAR shapedata volume;
  CONST pos p1 := [200, 100, 100];
  CONST pos p2 := [600, 400, 400];

  !Define a box between the corners p1 and p2
  WZBoxDef \Inside, volume, p1, p2;

  !Define and enable supervision of the box
  WZLimSup \Stat, obstacle, volume;
ENDPROC
```

机器人就位时的信号

当两台机器人共享一个工作区域时，很重要的一点就是了解一台机器人何时会让出道路、使另一台机器人能自由移动。

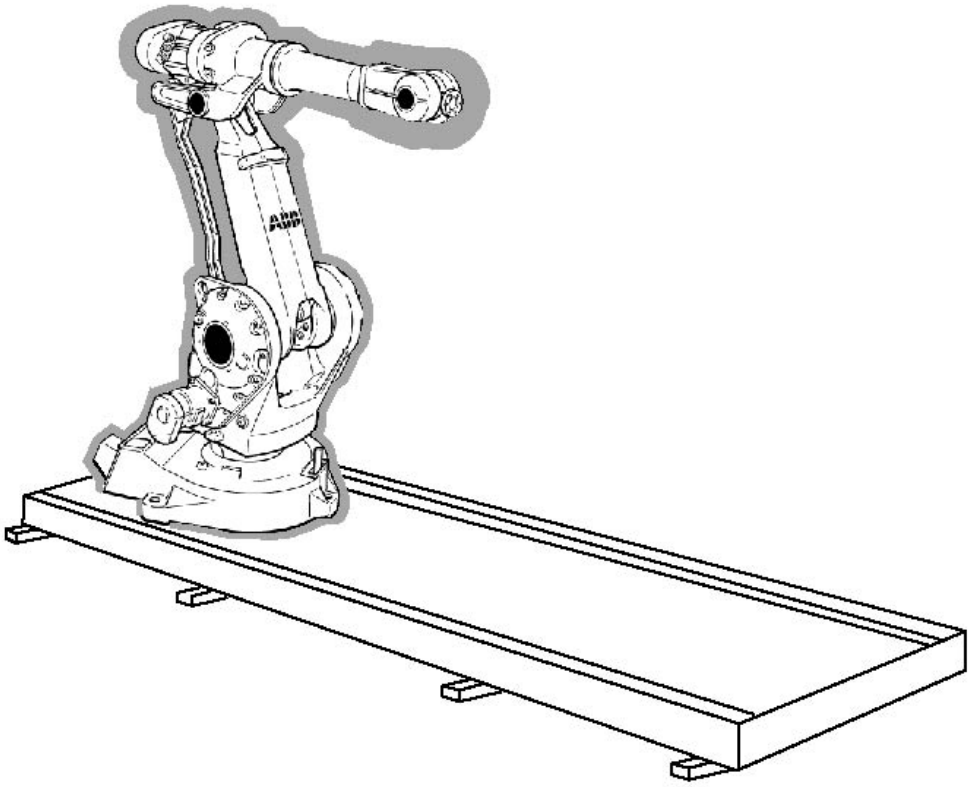
此例定义了一个相关机器人处于安全位置时的起始位置，并设置了一个机器人处于其起始位置时的输出信号。相关机器人会立在一条往返导轨上，而系统则会将其作为外轴1进行处理。其它外轴均未激活。

下一页继续

4.1.3 代码示例

续前页

插图中的阴影区域展示了相应的全局区域。



xx0300000206

```
VAR wztemporary home;
PROC zone_output()
  VAR shapedata joint_space;

  !Define the home position
  CONST jointtarget home_pos := [[0, -20, 0, 0, 0, 0], [0, 9E9,
       9E9, 9E9, 9E9, 9E9]];

  !Define accepted deviation from the home position
  CONST jointtarget delta_pos := [[2, 2, 2, 2, 2, 2], [10, 9E9,
       9E9, 9E9, 9E9, 9E9]];

  !Define the shape of the world zone
  WZHomeJointDef \Inside, joint_space, home_pos, delta_pos;

  !Define the world zone, setting the
  !signal do_home to 1 when in zone
  WZDOSet \Temp, home \Inside, joint_space, do_home, 1;
ENDPROC
```

4.2 Collision Detection [3107-1]

4.2.1 概述

目的

“碰撞检测”是一种软件选项，可以减少相关机器人承受的碰撞力度。这有助于避免机器人和外部设备受到严重损伤。

> **警告**
>
> “碰撞检测”无法在全速碰撞下保护设备。

描述

软件选项“碰撞检测”会对相关机器人实施基于型号的高敏感度监控，从而对碰撞进行识别。根据在相关机器人上刻意施加的力的状况，您既可对这种敏感度进行微调，也可以开启和关闭这种敏感度。由于相关机器人上的力会在程序执行过程中波动，因此您可在程序代码中将该敏感度设置成“在线”。

碰撞检测比普通的监控更为敏感，同时还有一些额外的特性。当检测到一次碰撞时，相关机器人会立即停止，并沿其路径反向移动一小段距离来释放余力。当接受了一则碰撞错误消息后，系统便能继续执行相应的移动，而无需按下相关控制器上的“**电机开启**”。

其中包括

您可通过RobotWare选项“碰撞检测”来访问：

- 这些系统参数的作用是定义是否宜激活“碰撞检测”，以及宜采用何种“碰撞检测”敏感度（若没有该选项，您就只能在自动模式下打开和关闭检测）
- 针对敏感度在线变化的指令：`MotionSup`

基本方法

机器人移动时默认始终激活“碰撞检测”，这意味着许多情况下您无需采取任何主动措施就能使用“碰撞检测”。

必要时您可打开和关闭“碰撞检测”，或通过两种途径更改其敏感度：

- 可通过RAPID指令`MotionSup`进行在线临时更改
- 通过系统参数进行永久性更改。

YuMi 机器人碰撞检测

YuMi 静止时默认开启碰撞检测，与其他机器人相比还额外配备停止斜坡，以便释放夹紧力。

> **注意**
>
> 如果工具数据错误，可能触发误碰撞，机器人手臂可能会在停止斜坡期间下降一小段距离。

4.2.2 限制

负载定义

为了恰当地检测出碰撞，用户必须正确定义相关机器人的有效负载。

提示

使用“负载识别”来定义有效负载。更多信息请参见操作手册 - *OmniCore*。

仅用于机器人轴

只有机器人轴才能使用“碰撞检测”，导轨运动、轨道站或其它任何外轴均无法使用这一功能。

独立关节

当至少有一根轴在独立关节模式下运行时，系统便会停用碰撞检测（即使是作为独立关节而运行的外轴也不例外）。

软伺服器

如果在软伺服器模式下使用相关机器人，那么就可能在无碰撞的情况下触发碰撞检测，因此我们建议在这种模式下使用机器人时关闭碰撞检测。

机器人移动前不作改动

如果使用RAPID指令`MotionSup`来关闭碰撞检测，那么只有当相关机器人开始移动时，这一操作才会真正生效，因此在相关机器人移动之前，数字输出项*MotSupOn*在程序开始时的数值可能会出乎意料。

反向移动距离

相关机器人在碰撞后的反向移动距离与碰撞前的运动速度成正比。如果反复发生低速碰撞，那么相关机器人就可能倒退足够远的距离来释放碰撞应力，所以在未触发监控的情况下可能无法点动机器人。此时要临时关闭“碰撞检测”，然后点动该机器人以远离障碍。

反向移动前的延时

如果在程序执行期间发生刚性碰撞，那么相关机器人可能要等待几秒才能反向移动。

导轨上的机器人

当把相关机器人安装在一条导轨上时，如果该导轨正在移动，那么就宜停用碰撞检测；如果不停用，那么即使当时没有碰撞，也可能在导轨移动时触发碰撞检测。

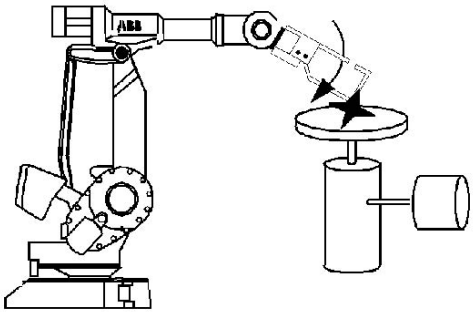
4.2.3 碰撞时的情况

概述

当触发碰撞检测时，相关机器人会尽快停止，然后通过反向移动来消除余力。相关程序会停止，并随之出现一则错误消息。相关机器人仍会停留在状态电机开启，因此待接受了该碰撞错误消息后，便能继续执行相关程序。

下图展示了一次典型的碰撞。

碰撞图



xx0300000361

一次碰撞后的机器人行为

该表展示了一次碰撞后的事件顺序。其序列说明请详见下图。

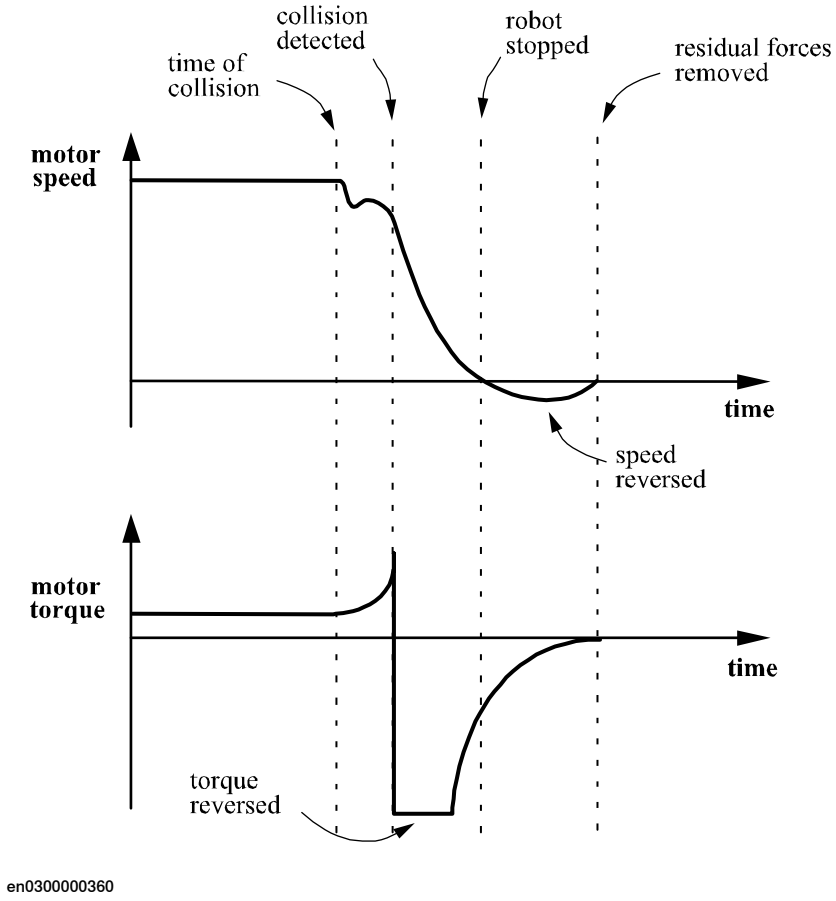
当……	那么……
检测到碰撞	为了停止相关机器人，系统会反向旋转相应电机，并采用机械制动
相关机器人已停止	为了消除碰撞或卡滞时可能存在的任何余力，相关机器人会沿相关路径反向一小段距离
消除了余力	相关机器人会再次停止，但仍然处于电机开启状态

下一页继续

4.2.3 碰撞时的情况

续前页

速度与扭矩图



en0300000360

4.2.4 附加信息

运动错误处理

碰撞错误处理方面的更多信息请参见技术参考手册 - *RAPID*语言内核。

4.2.5 配置和编写设施

4.2.5.1 系统参数

关于系统参数

这些碰撞检测参数 **无** 需重启便可生效。

有关这些参数的更多信息，请参见技术参考手册 - 系统参数。

Motion Supervision

这些参数属于主题*Motion*下的类型*Motion Supervision*。

参数	描述
Path Collision Detection	按执行程序时的需要来打开或关闭碰撞检测。 *Path Collision Detection*被默认设置成On。
Jog Collision Detection	按点动时的需要来打开或关闭碰撞检测。 *Jog Collision Detection*被默认设置成On。
Path Collision Detection Level	按指定百分数修改执行程序时所需的碰撞检测监控等级。较大的百分数意味着该函数的敏感度较低。 *Path Collision Detection Level*被默认设置成100%。
Jog Collision Detection Level	按指定百分数修改点动时所需的碰撞检测监控等级。较大的百分数意味着该函数的敏感度较低。 *Jog Collision Detection Level*被默认设置成100%。
Collision Detection Memory	定义碰撞后的机器人要沿相关路径反向移动多远（以秒为单位）。与低速机器人相比，碰撞前移动较快的机器人会移动得更远。 *Collision Detection Memory*被默认设置成75毫秒。
Manipulator Supervision	按IRB 340和IRB 360的需要来打开或关闭松臂检测监控。 *Manipulator Supervision*被默认设置成On。
Manipulator Supervision Level	按机械臂IRB 340和IRB 360的需要来修改松臂检测的监控等级。较大的数值意味着该函数的敏感度较低。 *Manipulator Supervision Level*的数值被默认设置成100%。

Motion Planner

这些参数属于主题*Motion*下的类型*Motion Planner*。

参数	描述
Motion Supervision Max Level	将最大等级设置成总的可可更改碰撞检测微调等级。其默认设置成300%。

Motion System

此参数属于 *Motion* 主题下的 *Motion System* 类型。

参数	描述
Ind collision stop without brake	此参数仅对使用 MultiMove 选项的系统有效。如果将此参数设置为 TRUE，检测到的碰撞将在独立执行的 RAPID 任务中独立处理。 此参数需要重新启动才能生效。

下一页继续

General RAPID

这些参数属于主题*Controller*下的类型*General RAPID*。

参数	描述
Collision Error Handler	启用针对碰撞的RAPID错误处理。*Collision Error Handler*被默认设置成Off。 碰撞错误处理方面的更多信息请参见技术参考手册 - *RAPID*语言内核。

4.2.5.2 RAPID组件

指令：

此处简述了“碰撞检测”中的各项指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
MotionSup	`MotionSup`的作用是： • 激活或停用“碰撞检测”。只有当路径碰撞检测被设置成“开启”时，用户才能进行这种操作。 • 按指定百分数（1%到300%）修改监控等级。较大的百分数意味着该函数的敏感度较低。

4.2.5.3 Signals

数字输出

此处简述了“碰撞检测”中的数字输出项。更多信息请参见技术参考手册 - 系统参数中的各个数字输出项。

数字输出	描述
MotSupOn	当“碰撞检测”处于激活状态时，*MotSupOn*会偏高；而当“碰撞检测”处于未激活状态时，则会偏低。 请注意，状态变化会在开始运动时生效，因此如果激活了“碰撞检测”且相关机器人正在移动，那么*MotSupOn*就会偏高。如果停止相关机器人并关闭“碰撞检测”，那么*MotSupOn*仍会偏高；而当该机器人开始移动时，*MotSupOn*则会切换为低值。 在发出机器人控制器重启后的第一个Motors On命令之前，*MotSupOn*将反映相应的系统参数*Path Collision Detection*的值： • 如果*Path Collision Detection*设置为开，*MotSupOn*将为高。 • 如果*Path Collision Detection*设置为关，*MotSupOn*将为低。
MotSupTrigg	触发碰撞检测时*MotSupTrigg*会偏高，并将高值一直保持到通过 FlexPendant 示教器认可了相应的错误代码为止。

4.2.6 何时使用Collision Detection

4.2.6.1 设置系统参数

激活监控

为了能在程序执行期间使用“碰撞检测”，用户必须将参数路径碰撞检测设置成开启。
为了能在点动期间使用“碰撞检测”，用户必须将参数点动碰撞检测设置成开启。

定义监控等级

将参数路径碰撞检测等级设置成你想要的百分数，以作为程序执行期间的默认值。
将参数点动碰撞检测等级设置成你想要的百分数，以作为点动期间的默认值。

4.2.6.2 FlexPendant示教器的调节监控

调节速度监控等级

“碰撞检测”采用了一种可变监控等级，且低速时比高速时更为敏感。出于这一缘故，用户在正常操作期间宜无需微调这一函数，不过可以通过打开和关闭该函数的方式来微调监控等级。

点动和程序执行期间可单独微调各个参数。第*204*页的系统参数对这些参数做了描述。

在FlexPendant示教器上设置点动监控

在 FlexPendant 上，从 **QuickSet 窗口选择 Control（控制）**，然后点击 **Jog（微动）**。在 **Jog Settings（微动设置）**上，点击 **Jog Supervision（微动控制监视）**。

不论是编程路径还是点动，用户均既能打开或关闭监控，又能调节相应的敏感度。按百分数来设置敏感度等级，较大的数值意味着该函数的敏感度较低。

如果在对话框中关闭了点动运动监控并执行了一段程序，那么执行该程序时“碰撞检测”仍会处于激活状态。

> **注意**
>
> 这些监控设定对应着类型为*Motion Supervision*的系统参数。正如上文所述，用户可用FlexPendant示教器上的监控设定来设置它们，此外也可用RobotStudio或FlexPendant示教器的配置编辑器或“快速设置机械单元”菜单来更改它们。

4.2.6.3 用RAPID程序调节监控

默认值

如果用相关系统参数来激活“碰撞检测”，那么程序执行期间的“碰撞检测”将默认处于激活状态（微调值100%）。系统会自动设置这些数值：

- 当使用重启模式重置系统时。
- 当载入一段新程序时。
- 当从起点开始执行程序时。

> **注意**
>
> 如果在相关系统参数和RAPID指令中设置了微调值，那么两者的数值都要纳入考虑。
>
> 示例：如果在相关系统参数中将该微调值设置成150%，在相关RAPID指令中将该微调值设置成200%，那么由此得出的微调等级就是300%。

临时停用监控

如果执行程序时有外力会影响到相关机器人，那么就用以下指令临时停用相关监控：

```
MotionSup \Off;
```

重新激活监控

如果已临时停用了相关监控，那么可用以下指令来激活监控：

```
MotionSup \On;
```

> **注意**
>
> 如果已用系统参数停用了相关监控，那么就无法用RAPID指令来激活监控。

微调

执行程序时可用指令*MotionSup*来微调监控等级。此处是按基本微调的百分数来设置微调值，即是说100%对应着基本值。更高的百分数意味着系统敏感度更低。

此例中的一条指令将相关监控等级增加到了200%。

```
MotionSup \On \TuneValue:=200;
```

4.2.6.4 如何避免误触发

关于误触发

由于相关监控被设置得十分敏感，因此如果负载数据有误，或相关机器人会承受较大的加工力，那么就可能会触发监控。

待采取的行动

如果……	那么……
有效负载的定义有误	使用“负载识别”来定义它。更多信息请参见操作手册 - *OmniCore*。
有效负载具有较大的质量或惯量	提高监控等级
因臂负载（电缆之类的物品）而导致触发	手动定义臂负载，或提高监控等级
相关应用涉及到许多外部加工力	按30%的幅度来逐步提高点动时和执行程序时的监控等级，直至您不再收到相应的错误代码为止。
外部加工力都只是暂时的	用指令MotionSup来提高相应的监控等级，或临时关闭相应的函数。
其它全部失效	关闭“碰撞检测”。

4.3 Collision Avoidance 3150-1

简介

功能*Collision Avoidance*用于监测机器人的一种详细几何模型。通过定义机器人工作区域中各主体的其它几何模型，当两个主体过于接近对方时，控制器就预测碰撞发出警告，并停止机器人。系统参数*Coll-Pred Safety Distance*决定将两物体视为会发生碰撞的距离。

功能*Collision Avoidance*可用于下列情况：例如，当设置和测试程序时，或者当程序位置不固定，但从传感器（如摄像头（非确定性程序））创建时。

本功能将监测经由RobotStudio中配置程序创建的10个对象（包括机器人在内）。典型监测对象包括安装在机器人法兰上的工具、安装在机器人臂（通常为轴3）上的其它设备或机器人的静态容积。

采用RobotStudio建立几何模型。

该功能由系统输入 *Collision Avoidance* 激活。高信号将激活该功能，低信号将停用该功能。如果没有为系统输入 *Collision Avoidance* 分配信号，该功能默认为激活。

*Collision Avoidance*在操纵和运行程序时都是活动的。同时，RAPID 功能的 `IsCollFree` 提供了一种在移动到某个位置前检查可能碰撞的方法。

> **小心**
>
> 由于碰撞可能损坏手臂机械结构，因此，务必小心以避免同外部设备发生碰撞。
>
> *Collision Avoidance*不能保证避免碰撞。

限制

不支持喷漆机械臂 IRB 6620LX 和 IRB 360

Collision Avoidance 不能与响应式微动控制一同使用。必须将系统参数 *Jog Mode* 更改为 *Standard*。

仅当运用 MultiMove 系统时，方可实现 2 个（或更多）机器人之间的*Collision Avoidance*功能。

> **小心**
>
> *Collision Avoidance*不得用于人身安全。

虚假碰撞警告

有不同的方式来降低功能*Collision Avoidance*的灵敏度，以避免发出错误警告。

- 暂时禁用*Collision Avoidance*，请参见第*212*页的禁用*Collision Avoidance*。
- 减小一般安全距离以及系统参数*Coll-Pred Safety Distance*。

禁用*Collision Avoidance*

若机器人已发生碰撞或处在默认安全距离以内，或者当机械臂需要相互靠近且碰撞风险可被接受时，有可能暂时禁用功能*Collision Avoidance*。

设置系统输入信号`Collision Avoidance`至0，以便禁用*Collision Avoidance*。建议立即重新启用此功能（将`Collision Avoidance`设置为1），只要在需要禁用*Collision Avoidance*的工作完成后即可。

4.4 SafeMove Assistant

目的

SafeMove Assistant 是 RoboTware 中的一项功能，可帮助用户在当前的 SafeMove 配置时对其应用程序进行编程。这一助手将读取活动配置并根据该配置中的限制和设置规划轨迹。助手将设置速度，以便 SafeMove 不会触发违规等。如果机器人被编程为进入禁区等，它还会停止并显示错误消息。

SafeMove Assistant 将自动调整机器人的行为以适应当前的 SafeMove 配置，机器人将采用限速区域并在进入禁区之前停止。

注意

如果发生 SafeMove Assistant 故障，SafeMove 监督将触发紧急停止。

描述

SafeMove Assistant 将检查笛卡尔速度检查点（TCP、工具点和弯头）是否有任何 SafeMove 速度限值处于活动状态。如果是这种情况，将在路径规划程序中应用相应的速度限值。由于技术上的原因，只有 TCP、手腕中心点 (WCP) 和肘部的速度受路径规划程序的限制。因此，如果其他工具点的移动速度快于 TCP，SafeMove 可能会触发工具速度违规。为避免这种情况，请更改程序或减小参数 *SafeMove assistance speed factor* 的值（见下文）。

手动模式下的 SafeMove Assistant 处于未激活状态。

SafeMove Assistant 不考虑在较低级别生成的路径修正。因此，在运行诸如外部引导运动或传送带跟踪之类的应用程序时，违反 SafeMove 的风险会增加。

系统参数

SafeMove Assistant 可被禁用以进行 SafeMove 验证等。这通过在 *Motion System* 中键入的参数 *Disable SafeMove Assistance*而实现。

如果机器人系统出现轻微过冲或以任何其他方式触发 SafeMove 违规，则可以更改某些参数。

参数	描述
SafeMove Assistance Speed Factor	它有一个默认设置 0.96，对应于速度监督的 96 % 将是路径规划程序会使用的速度。该参数可以减小以降低风险，但在大多数情况下可以保持默认值。
SafeMove assistance zone margin	当机器人在区域边界上运行时，SafeMove 在进出区域时触发违规的风险很小。可增加此参数以降低风险，但在大多数情况下可保留为默认值。

有关更多信息，请参阅 技术参考手册 - 系统参数 中 所述类型 *Motion System* 的参数。

此页刻意留白

5 Motor Control

5.1 Independent Axis [3111-1]

5.1.1 概述

目的

Independent Axis的作用是在独立于本机器人系统中其它各轴的情况下移动一根轴。以下是一些应用示例：

- 移动夹持有一个对象的一根外轴（比如在机器人为该对象喷漆时旋转该对象）。
- 在外轴执行其它任务的同时执行一项机器人任务，从而节省周期时间。
- 连续旋转机器人轴6（用于抛光任务或类似任务）。
- 当某根轴朝同一方向旋转多圈后重置相应的测量系统。保存相对于物理回卷的周期时间。

如果某根轴被设置成独立模式，它便可独立移动。用户可以将一根轴先改为独立模式，然后又恢复成正常模式。

其中包括

您可通过 RobotWare 选项Independent Axis来访问：

- 设置独立模式和指定某轴移动情况的指令
- 改回正常模式和 / 或重置相关测量系统的一条指令
- 用于验证一根独立轴之状态的函数
- 用于配置的系统参数。

基本方法

这是独立移动一根轴的一般方式。第*219*页的代码示例用一个更详细的示例展示了其具体做法。

1 调用一条独立移动指令来把该轴设置成独立模式并移动该轴。
2 让相关机器人在独立轴移动时执行另一条指令。
3 当机器人和独立轴均已停止后，请将其中的独立轴重置为正常状态。

重置轴

即使不在独立模式下，轴也可能仅朝一个方向旋转，并最终丧失精确度。此时可用指令`IndReset`来重置相关测量系统。

建议当一根轴的电机朝同一方向旋转10000圈以上后，便重置该轴的测量系统。

限制

如果某个机械单元有一根轴正处于独立模式，则可能无法停用该机械单元。

无法点动独立模式下的各轴。

唯一能用作独立轴的机器人轴就是6号轴。在IRB 1600、2600和4600型号（ID版本除外）上，用户也可对轴4使用指令`IndReset`。

内部和客户电缆及设备可能会限制在轴 4 和轴 6 上应用独立轴功能的能力。

下一页继续

5.1.1 概述
续前页

该选项不能与以下部件结合使用：

- SafeMove[I]
- 轨道运动系统（IRBT）
- 互换轴上的定位器（IRBP）

[I] 如果更多的轴并不移动机器人，而且不是由 SafeMove 对更多的轴进行监测，在某些情况下可将 *Independent Axis* 与 SafeMove2 合并。更多信息请联络您当地的 ABB 销售处。

应用“独立轴”选项时，以下功能停用：

- 碰撞检测

> **注意**
>
> 如果运动规划器中有一个轴以独立模式运行，则所有轴上的碰撞检测均将停用。

5.1.2 系统参数

关于系统参数

此处简述了*Independent Axis*选项中的每个参数。更多信息请参见技术参考手册 - 系统参数中的各个参数。

Arm

这些参数属于主题*Motion*下的类型*Arm*。

参数	描述
Independent Joint	决定了是否允许该轴使用独立模式的旗标。
Independent Upper Joint Bound	定义该关节在独立模式下运行时的工作区域上限。
Independent Lower Joint Bound	定义该关节在独立模式下运行时的工作区域下限。

Transmission

这些参数属于主题*Motion*下的类型*Transmission*。

参数	描述
Transmission Gear High	独立轴需要分辨率较高的传输齿轮比，因此该齿轮比被定义为*Transmission Gear High*除以*Transmission Gear Low*。如果*Transmission Gear High*被设置成相关机器人轴侧的嵌齿数目，而*Transmission Gear Low*被设置成电机侧的嵌齿数目，且无法使用更小的数字，那么相应的传输齿轮比就是正确的。
Transmission Gear Low	参见*Transmission Gear High*。

5.1.3 RAPID组件

数据类型

没有针对Independent Axis的数据类型。

指令：

此处简述了Independent Axis中的每条指令。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各条指令。

即使该轴此时正在移动，系统也会立即执行一条独立移动指令。如果在上一条独立移动指令尚未结束时就执行一条新的独立移动指令，那么新的指令会立即超驰旧的指令。

指令	描述
IndAMove	`IndAMove`（独立绝对位置移动）会把一根轴改为独立模式，然后将该轴移到指定位置。
IndCMove	`IndCMove`（独立连续移动）会把一根轴改为独立模式，然后开始按指定速度来连续移动该轴。
IndDMove	`IndDMove`（独立绝对Δ位置移动）会把一根轴改为独立模式，然后使该轴移动指定距离。
IndRMove	`IndRMove`（独立相对位置移动）会把一根轴改为独立模式，然后在一圈内将该轴移到一个特定位置。 由于忽略了相关位置的旋转信息，因此`IndRMove`永远不会旋转一轴圈以上。
IndReset	`IndReset`的作用是将一根独立轴改回正常模式。 `IndReset`能将旋转轴的测量系统转动许多轴圈。当逐渐远离逻辑位置0时，各位置的分辨率将有所下降，而回卷相应的轴则要耗费一定时间。通过移动相关测量系统，用户可在不对轴进行物理回卷的前提下维持相应的分辨率。 在调用`IndReset`，相关的独立轴和机器人都必须直立不动。

函数

此处简述了Independent Axis中的每则函数。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各则函数。

功能	描述
IndInpos	`IndInpos`指明了某根轴是否已抵达选定位置。
IndSpeed	`IndSpeed`指明了某根轴是否已达到选定速度。

5.1.4 代码示例

节省周期时间

A站中的对象有两处需要焊接。当相关机器人正在焊接另一个对象时，A站的外轴可将自身中的对象转动到合适位置，从而节省周期时间（而不是在移动外轴时让机器人白白等候）。

```
!Perform first welding in station A
!Call subroutine for welding
weld_stationA_1;

!Move the object in station A, axis 1, with
!independent movement to position 90 degrees
!at the speed 20 degrees/second
IndAMove Station_A,1\ToAbsNum:=90,20;

!Let the robot perform another task while waiting
!Call subroutine for welding
weld_stationB_1;

!Wait until the independent axis is in position
WaitUntil IndInpos(Station_A,1 ) = TRUE;
WaitTime 0.2;

!Perform second welding in station A
!Call subroutine for welding
weld_stationA_2;
```

用旋转轴6抛光

若要抛光一个对象，则可将机器人轴6设置成连续旋转。

将机器人轴6设置成独立模式，然后连续旋转该轴。将相关机器人移到您想抛光的区域上方，然后停止该机器人和该独立轴的移动，将模式改回正常模式。为了维持相应的分辨率，当该轴旋转了许多圈后请重置相关的测量系统。

请注意，若要使本例真正起到作用，就必须把rob1_6的参数*Independent Joint*设置成Yes。

```
PROC Polish()
  !Change axis 6 of ROB_1 to independent mode and
  !rotate it with 180 degrees/second
  IndCMove ROB_1, 6, 180;

  !Wait until axis 6 is up to speed
  WaitUntil IndSpeed(ROB_1,6\InSpeed);
  WaitTime 0.2;

  !Move robot where you want to polish
  MoveL p1,v10, z50, tool1;
  MoveL p2,v10, fine, tool1;

  !Stop axis 6 and wait until it's still
  IndCMove ROB_1, 6, 0;
```

下一页继续

5.1.4 代码示例

续前页

```
  WaitUntil IndSpeed(ROB_1,6\ZeroSpeed);
  WaitTime 0.2;

  !Change axis 6 back to normal mode and
  !reset measurement system (close to 0)
  IndReset ROB_1, 6 \RefNum:=0 \Short;
ENDPROC
```

重置一根轴

此例展示了如何重置A站中轴1的测量系统。该测量系统会以整圈为单位来更改圈数，因此其靠近零位（±180°）。

```
IndReset Station_A, 1 \RefNum:=0 \Short;
```

6 RAPID Program Features

6.1 Path Recovery [3113-1]

6.1.1 概述

目的

Path Recovery的作用是保存当前移动路径，执行某些机器人移动，然后恢复被中断的路径。一旦在路径移动期间发生错误或中断，这项功能就将发挥作用。可由一个错误处理器或一则中断例程来执行一项任务，然后重新创建相应的路径。

对电弧焊和胶合等应用而言，很重要的一点就是从相关机器人离开的位点处继续工作。如果相关机器人从起点处重新开始，那么就只能废弃相关工件了。

如果当机器人在工件内部时发生了一项进程错误，那么直接移动该机器人就可能导致碰撞。而采用了路径记录器后，该机器人便能沿着进来时的同一路径移出工件。

其中包括

您可通过RobotWare选项Path Recovery来访问：

- 在相应的错误或中断等级上暂停和继续协调同步移动模式的各条指令。
- 一件路径记录器，它能让TCP沿着移入时的同一路径移出。

限制

指令StorePath和RestoPath只会处理移动路径数据。必须保存相应的停止位置。

如果某次移动采用了路径记录器，那么就必须在软中断等级上执行此次移动，即是说必须在PathRecMoveBwd之前执行StorePath。

6.1.2 RAPID组件

数据类型

此处简述了Path Recovery中的每种数据类型。更多信息请参见技术参考手册 - *RAPID* 指令、函数和数据类型中的各种数据类型。

数据类型	描述
pathrecid	pathrecid的作用是识别路径记录器的断点。

指令：

此处简述了Path Recovery中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
StorePath	StorePath的作用是保存当发生错误或中断时正在执行的移动路径。 RobotWare基座包括了StorePath。
RestoPath	RestoPath的作用是恢复StorePath所保存的路径。 RobotWare基座包括了RestoPath。
PathRecStart	PathRecStart的作用是开始记录相关机器人的路径。路径记录器将保存执行相关机器人程序时的路径信息。
PathRecStop	PathRecStop的作用是停止记录相关机器人的路径。
PathRecMoveBwd	PathRecMoveBwd的作用是沿所记录的路径撤回相关机器人。
PathRecMoveFwd	PathRecMoveFwd的作用是将相关机器人撤回曾执行PathRecMoveBwd的位置。 也可提供一个后撤移动中被跳过的标识符，从而让相关机器人部分前移。
SyncMoveSuspend	SyncMoveSuspend的作用是暂停同步移动模式，并将系统设置成独立移动模式。
SyncMoveResume	SyncmoveResume的作用是从独立移动模式回到同步移动。

函数

此处简述了Path Recovery中的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
PathRecValidBwd	PathRecValidBwd的作用是检查相关路径记录器是否处于激活状态，以及是否有一条已记录的后撤路径可用。
PathRecValidFwd	PathRecValidFwd的作用是检查是否能用相关路径记录器向前移动。若能用路径记录器向前移动，则意味着之前肯定已经下令该路径记录器后撤。

6.1.3 保存当前路径

为何保存该路径？

使用Path Recovery的最简单方式就是仅保存“能在解决一项错误或类似行动后得以恢复”的当前路径。

假设电弧焊期间发生了一项错误。为了解决这一错误，用户可能必须从相关零件处移开相关机器人。在解决了这一错误后，又最好从机器人离开处继续焊接。而要做到这一点，就要在机器人离开相关路径前保存该机器人的路径信息和位置，然后便可在处理完错误后恢复这一路径并继续焊接。

基本方法

这是保存当前路径的一般方式：

1 在启动一个错误处理器或中断例程时：
   停止相关运动
   保存相关移动路径
   保存相应的停止位置
2 在启动一个错误处理器或中断例程时：
   移到已保存的停止位置处
   恢复相关移动路径
   开始相关移动

示例

此例展示了如何在处理错误时使用Path Recovery。首先保存相应的路径和位置，再校正相关错误，然后将相关机器人撤至适当位置，最后恢复相应路径。

```
MoveL p100, v100, z10, gun1;
...
ERROR
  IF ERRNO=MY_GUN_ERR THEN
    gun_cleaning();
  ENDIF
...
PROC gun_cleaning()
  VAR robtarget p1;

  !Stop the robot movement, if not already stopped.
  StopMove;

  !Store the movement path and current position
  StorePath;
  p1 := CRobT(\Tool:=gun1\WObj:=wobj0);

  !Correct the error
  MoveL pclean, v100, fine, gun1;
  ...
  !Move the robot back to the stored position
  MoveL p1, v100, fine, gun1;
```

下一页继续

6.1.3 保存当前路径

续前页

```
    !Restore the path and start the movement
    RestoPath;
    StartMove;
    RETRY;
  ENDPROC
```

6.1.4 路径记录

什么是路径记录器

路径记录器能储存大量移动指令，而利用这些指令，用户便能沿同一路径撤回相关机器人。

如何使用路径记录器

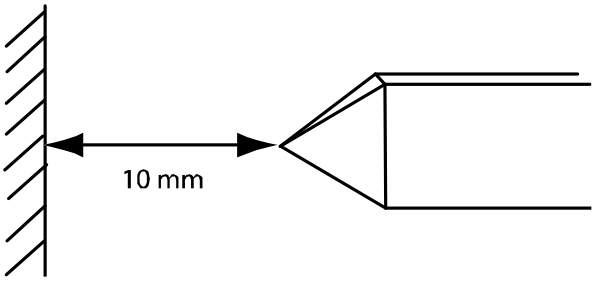
这是路径记录器的一般用法：

1 启动路径记录器
2 用常规移动、进程或指令来移动机器人
3 保存当前路径
4 沿已记录的路径后撤
5 解决错误
6 沿已记录的路径前进
7 恢复被中断的路径

抬高工具

当机器人撤回其自身导轨后，您可能希望避免工具擦刮到相关工件，比如想在电弧焊过程中避开焊缝等。

通过使用指令PathRecMoveBwd和PathRecMoveFwd中的自变数ToolOffs，您可为TCP设置一段偏移。该偏移会被设置在工具坐标上，这意味着如果将其设置成【0，0，10】，那么当相关工具沿已记录的路径后撤时，其将与相应工件将相距10毫米。



xx0400000828

注意

当一套MultiMove系统处于同步模式时，如果正要抬高一件工具，那么所有任务都必须使用ToolOffs。

话说回来，如果您仅想抬高一件工具，那么就在其它任务中设置ToolOffs=[0,0,0]。

简单示例

如果p1与p4之间发生了一项错误，那么相关机器人就会返回p1来解决这一错误。当错误得到解决后，该机器人便会从发生错误处继续运行。

当没有任何错误就抵达p4时，系统便会关闭相应的路径记录器，然后相关机器人会在没有路径记录器的情况下从p4移到p5。

```
...
```

下一页继续

6.1.4 路径记录
续前页

```
VAR pathrecid start_id;
...
  MoveL p1, vmax, fine, tool1;
  PathRecStart start_id;
  MoveL p2, vmax, z50, tool1;
  MoveL p3, vmax, z50, tool1;
  MoveL p4, vmax, fine, tool1;
  PathRecStop \Clear;
  MoveL p5, vmax, fine, tool1;

ERROR
  StorePath;
  PathRecMoveBwd;
  ! Fix the problem
  PathRecMoveFwd;
  RestoPath;
  StartMove;
  RETRY;
ENDIF
...
```

复杂示例

本例中路径记录器有两种用途：

- 如果发生了一项错误，那么操作员可以选择退回p1或p2。一旦解决了相关错误，系统就会恢复被中断的移动过程。
- 如果未发生任何错误，那么系统就会用路径记录器将相关机器人从p4移到p1。当相关机器人处在一个难以移出的狭窄位置时，这种技巧会有所帮助。

请注意，如果在第一条移动指令期间（p1与p2之间）发生错误，那么就无法返回p2。如果操作员选择退回p2，那么就用PathRecValidBwd来查看这种做法是否可行。当相关机器人前进到其之前中断的位置时，则用PathRecValidFwd来查看这种做法是否可行（如果相关机器人始终不回退，那么就是它已经就位了）。

```
...
VAR pathrecid origin_id;
VAR pathrecid corner_id;
VAR num choice;
...
  MoveJ p1, vmax, z50, tool1;
  PathRecStart origin_id;
  MoveJ p2, vmax, z50, tool1;
  PathRecStart corner_id;
  MoveL p3, vmax, z50, tool1;
  MoveL p4, vmax, fine, tool1;

  ! Use path record to move safely to p1
  StorePath;
  PathRecMoveBwd \ID:=origin_id
      \ToolOffs:=[0,0,10];
  RestoPath;
  PathRecStop \Clear;
```

下一页继续

```
  Clear Path;
  Start Move;

ERROR
  StorePath;

  ! Ask operator how far to back up
  TPReadFK choice,"Extract to:", stEmpty, stEmpty,
      stEmpty, "Origin", "Corner";

  IF choice=4 THEN
    ! Back up to p1
    PathRecMoveBwd \ID:=origin_id
        \ToolOffs:=[0,0,10];
  ELSEIF choice=5 THEN
    ! Verify that it is possible to back to p2,
    IF PathRecValidBwd(\ID:=corner_id) THEN
      ! Back up to p2
      PathRecMoveBwd \ID:=corner_id
          \ToolOffs:=[0,0,10];
    ENDIF
  ENDIF

  ! Fix the problem

  ! Verify that there is a path record forward
  IF PathRecValidFwd() THEN
    ! Return to where the path was interrupted
    PathRecMoveFwd \ToolOffs:=[0,0,10];
  ENDIF

  ! Restore the path and resume movement
  RestoPath;
  StartMove;
  RETRY;
  ...
```

恢复路径记录器

如果停止了该路径记录器，那么就能在不丢失其历史记录的情况下从同一位置重新启动该记录器。

下例中的PathRecMoveBwd指令使相关机器人退回到了p1。在重启路径记录器时，如果相关机器人正处在p2之外的其它位置，那么它就无法退回到p1。

更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中关于PathRecStop的章节。

```
...
MoveL p1, vmax, z50, tool1;
PathRecStart id1;
MoveL p2, vmax, z50, tool1;
PathRecStop;
MoveL p3, vmax, z50, tool1;
```

下一页继续

6.1.4 路径记录

续前页

```
MoveL p4, vmax, z50, tool1;
MoveL p2, vmax, z50, tool1;
PathRecStart id2;
MoveL p5, vmax, z50, tool1;
StorePath;
PathRecMoveBwd \ID:=id1;
RestoPath;
...
```

6.2 Multitasking [3114-1]

6.2.1 Multitasking介绍

目的

选项*Multitasking*的作用是能够同时执行多段程序。

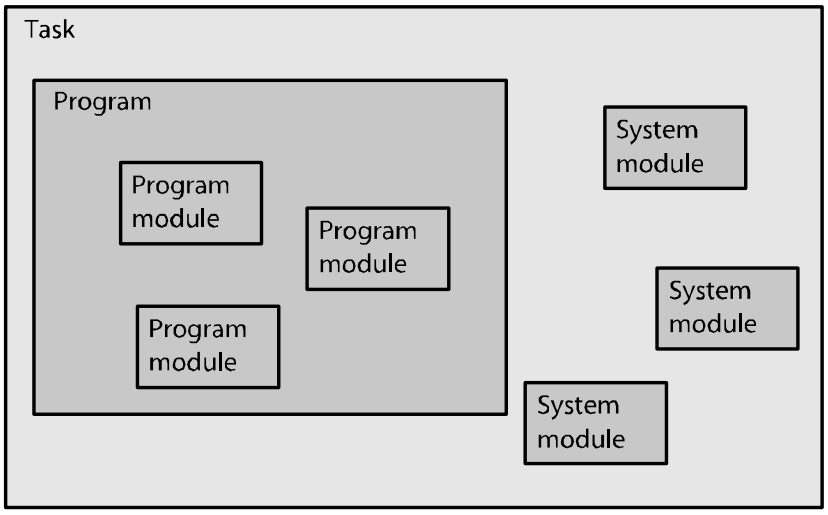
让各项应用与主程序并行运行的示例：

- 即使主程序已经停止，也仍继续进行信号监控。有时这会接手PLC的工作，不过其响应时间与PLC并不相符。
- 相关机器人正在工作时来自FlexPendant示教器的操作员输入项。
- 外部设备的控制和激活 / 停用。

基本说明

同一时间最多运行20项任务。

每项任务都由一段程序（含若干程序模块）和若干个系统模块组成。这些模块是各项任务中的本地模块。



en0300000517

变量和常量都是各项任务中的本地量，但持续变量并不是本地量。

每项任务都有自身的软中断处理事项，且只有其自身的任务系统状态才能触发事件例程。

其中包括

您可通过RobotWare选项Multitasking来访问：

- 最多能并行运行20段程序（每项任务运行一段）。
- 系统参数：类型*Task*及其所有参数。
- 数据类型：`taskid`、`syncident`和`tasks`。
- 指令：`WaitSyncTask`.
- 函数：`TestAndSet`、`TaskRunMec`和`TaskRunRob`。

下一页继续

6.2.1 Multitasking介绍
续前页

> **注意**
>
> 没有选项Multitasking也能使用TestAndSet、TaskRunMec和TaskRunRob，但搭配Multitasking时会大大提升它们的作用。

基本方法

这是设置 Multitasking 的基本方式。有关详细信息，请参阅 第232页的*RAPID*组件。

1 定义您需要的任务。
2 写入每项任务的RAPID代码。
3 指定载入各项任务的模块。

6.2.2 系统参数

关于系统参数

此处简述了*Multitasking*选项中的每个参数。更多信息请参见技术参考手册 - 系统参数中的各个参数。

Task

这些参数属于主题*Controller*下的类型*Task*。

参数	描述
Task	任务名称。 注意该任务的名称必须是唯一的，这意味着不能是相关机械单元的名称，相应的RAPID程序中也不能有相同名称的变量。 请注意，若在配置编辑器中编辑该任务条目并更改任务名称，则系统会移除旧任务并添加一项新任务。这意味着一旦做出此类改动，该任务中的所有程序或模块就会在重启后消失。
Task in foreground	用于设置各项任务的优先级。 *Task in foreground*包含了宜在该任务前台运行的任务名称。这意味着只有在前台任务程序空闲时，系统才会执行该参数设置的任务程序。 如果*Task in foreground* 被设置成某项任务的空字符串，那么它就会在最高等级上运行。
Type	控制启动 / 停止和系统重启行为： • 正常 (NORMAL) - 手动启动和停止任务程序（比如，从 FlexPendant 示教器进行启动和停止）。在紧急停止时，任务将停止。 • 静态 (STATIC) - 重启时该任务程序会从所处位置继续执行。正常情况下，不论是 FlexPendant 示教器还是紧急停止都不会停止该任务程序。 • 半静态 (SEMISTATIC) - 重启时该任务程序会从起点处重启。正常情况下，不论是 FlexPendant 示教器还是紧急停止都不会停止该任务程序。 凡是控制着机械单元的任务，其类型都必须是 *NORMAL*（正常）才行。
Main entry	该任务程序的启动例程的名称。
Check unresolved references	如果系统在链接有一个模块的情况下接受了相关程序中的未决引用项，那么就宜把该参数设置成NO，反之则设置成YES。
TrustLevel	*TrustLevel* 定义了一项静态或半静态任务被停止（比如因错误而停止）时的系统行为： • SysFail - 若该项任务的程序停止，则系统会被设置成 SYS_FAIL。这将导致所有“正常”任务的相应程序全部停止（不过系统会尽量继续执行静态和半静态任务）。此时既不能进行任何点动，也不能启动任何程序。用户需要重启一次。 • SysHalt - 若该项任务的程序停止，则系统会停止所有正常任务的相应程序。如果设置了“电机开启”，那么就可以进行点动，但不能启动任何程序。用户需要重启一次。 • SysStop - 若该项任务的程序停止，则系统会停止所有正常任务的相应程序，但可以重启这些任务。用户亦可以进行点动。 • NoSafety − 系统仅会停止该项任务的程序。
MotionTask	指明该任务程序能否用RAPID移动指令来控制机器人的移动行为。 除非使用了选项MultiMove，否则就只能把一项任务的*MotionTask*设置成YES。

6.2.3 RAPID组件

数据类型

此处简述了Multitasking中的每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各种数据类型。

数据类型	描述
taskid	`taskid`会识别本系统中的可用任务。 系统参数*Task*定义了这一标识，而RAPID程序则无法定义该标识。不过声明例程时可将数据类型`taskid`作为一个参数。 代码方面的示例请参见第*242*页的*taskid*。
syncident	`syncident`的作用是在使用指令`WaitSyncTask`时识别相关程序中的等候点。 所有任务程序中的`syncident`变量都必须具有同一个名称。 代码方面的示例请参见第*236*页的*WaitSyncTask*示例。
tasks	数据类型`tasks`的一个变量中包含了将通过指令`WaitSyncTask`实现同步的各项任务的名称。 代码方面的示例请参见第*236*页的*WaitSyncTask*示例。

指令

此处简述了Multitasking中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
WaitSyncTask	`WaitSyncTask`的作用是在相关程序的某个特殊点处实现若干任务程序的同步。 一条`WaitSyncTask`指令会延缓程序的执行过程，并等候其它任务程序。当所有任务程序都抵达指定点时，该程序才会继续执行。 代码方面的示例请参见第*236*页的*WaitSyncTask*示例。

函数

此处简述了Multitasking中的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
TestAndSet	`TestAndSet`的作用，是和一个布尔（boolean）旗标共同确保同一时间只有一段任务程序在使用某一具体的RAPID代码区域或系统资源。 代码方面的示例请参见第*240*页的涉及旗标的示例与*TestAndSet*。
TaskRunMec	检查该任务程序是否控制了任何机械单元（机器人或其它单元）。 代码方面的示例请参见第*241*页的测试任务是否控制着机械单元。
TaskRunRob	检查该任务程序是否控制了任何带有TCP的机器人。 代码方面的示例请参见第*241*页的测试任务是否控制着机械单元。

6.2.4 各项任务间的通信

6.2.4.1 永久变量

关于永久变量

若要在各项任务之间共享数据，则请使用永久变量。

永久变量是声明了该变量的所有任务的一个全局变量。必须在所有任务中为相关永久变量声明相同的类型和大小（数组维度），否则将会发生执行时间错误。

其足以为一项任务中的永久变量指定一个初始值。如果在若干项任务中指定了初始值，那么系统仅会使用最先载入的模块的初始值。

提示

在保存一段程序时，系统会把某个永久变量的当前值作为日后的初始值。如果不需要这种设定，那么就在通信后直接重置相应的永久变量。

涉及永久变量的示例

此例中的两项任务都访问了永久变量startsync和stringtosend，所以可将这些变量用于相关任务程序之间的通信。

主任务程序：

```
MODULE module1
  PERS bool startsync:=FALSE;
  PERS string stringtosend:="";
  PROC main()
    stringtosend:="this is a test";
    startsync:= TRUE
  ENDPROC
ENDMODULE
```

后台任务程序：

```
MODULE module2
  PERS bool startsync;
  PERS string stringtosend;
  PROC main()
    WaitUntil startsync;
    IF stringtosend = "this is a test" THEN
      ...
    ENDIF
    !reset persistent variables
    startsync:=FALSE;
    stringtosend:="";
  ENDPROC
ENDMODULE
```

公用数据模块

当在若干项任务中使用永久变量时，最好在所有这些任务中都加以声明。而为了避免类型错误或忘了在某处声明，最好的做法就是在一个系统模块中声明所有公用变量，然后将该系统模块载入所有需要这些变量的任务中。

6.2.4.2 等候其它任务

两种技巧

某些应用会在独立于其它任务情况下执行任务程序，但各任务程序通常还是需要了解其它任务正处在何种状态的。

可让某一任务程序等候其它任务程序，具体来说，要么设置一个其它任务程序可以调用的永久变量，要么设置一个可让其它任务程序关联到中断上的信号。

轮询

这是让任务程序等候其它任务程序的最简单方式，但却是执行最慢的方式。搭配指令WaitUntil或WHILE来使用永久变量。

如果使用了指令WaitUntil，那么系统将每100毫秒实施一次内部轮询。

> **小心**
>
> 轮询频率不要超过100毫秒一次。如果某环路的轮询没有等候指令，那么就会导致过载，进而丧失与FlexPendant示教器之间的联系。

轮询示例

主任务程序：

```
MODULE module1
  PERS bool startsync:=FALSE;
  PROC main()
    startsync:= TRUE;
    ...
  ENDPROC
ENDMODULE
```

后台任务程序：

```
MODULE module2
  PERS bool startsync:=FALSE;
  PROC main()

    WaitUntil startsync;
    ! This is the point where the execution
    ! continues after startsync is set to TRUE
    ...
  ENDPROC
ENDMODULE
```

中断

如果在某一任务程序中设置一个信号，并使用另一段任务程序中的一次中断，那么就既能快速响应，又没有轮询带来的工作负载。

其缺点是“必须把在此次中断后执行的代码放置在一则软中断例程中”。

下一页继续

中断示例

主任务程序：

```
MODULE module1
  PROC main()
    SetDO do1,1;
    ...
  ENDPROC
ENDMODULE
```

后台任务程序：

```
MODULE module2
  VAR intnum intno1;

  PROC main()
    CONNECT intno1 WITH wait_trap;
    ISignalDO do1, 1, intno1;
    WHILE TRUE DO
      WaitTime 10;
    ENDWHILE
  ENDPROC

  TRAP wait_trap
    ! This is the point where the execution
    ! continues after do1 is set in main task
    ...
    IDelete intno1;
  ENDTRAP
ENDMODULE
```

6.2.4.3 多项任务之间的同步

用WaitSyncTask实现同步

当各任务程序取决于彼此时，同步就会发挥作用。除非所有任务程序都抵达了各自程序代码中的同步点，否则这些任务程序都不会继续执行到其程序代码的同步点以外。

指令WaitSyncTask的作用是同步各任务程序。除非所有任务程序都抵达了同一WaitSyncTask指令处，否则这些任务程序都不会继续执行。

WaitSyncTask示例

在此例中，当相关的后台任务程序在计算下一个对象的位置时，主任务程序正在处理涉及当前对象的机器人工作。

该后台任务程序可能不得不等候操作员输入项或I / O信号，但在计算出相应的新位置前，主任务程序不会用下一个对象继续执行。与此类似，除非已用一个对象执行了主任务程序，且该主任务程序已准备好接收新值，否则相应的后台任务程序不得开始下一次计算。

主任务程序：

```
MODULE module1
  PERS pos object_position:=[0,0,0];
  PERS tasks task_list{2} := [["MAIN"], ["BACK1"]];
  VAR syncident sync1;

  PROC main()
    VAR pos position;
    WHILE TRUE DO
      !Wait for calculation of next object_position
      WaitSyncTask sync1, task_list;
      position:=object_position;
      !Call routine to handle object
      handle_object(position);
    ENDWHILE
  ENDPROC

  PROC handle_object(pos position)
    ...
  ENDPROC
ENDMODULE
```

后台任务程序：

```
MODULE module2
  PERS pos object_position:=[0,0,0];
  PERS tasks task_list{2} := [["MAIN"], ["BACK1"]];
  VAR syncident sync1;

  PROC main()
    WHILE TRUE DO
      !Call routine to calculate object_position
      calculate_position;
```

下一页继续

```
      !Wait for handling of current object
      WaitSyncTask sync1, task_list;
    ENDWHILE
  ENDPROC

  PROC calculate_position()
    ...
    object_position:= ...
  ENDPROC
ENDMODULE
```

6.2.4.4 使用调度程序

什么是调度程序？

可用一个数字信号来指明宜于何时让另一项任务来开展工作。不过该信号无法包含工作内容方面的信息。

用户可用一段调度程序——而不是使用一个信号——来决定调用哪一则例程。调度程序可以是一个永久字符串变量，其中包含了将在另一项任务中执行的例程之名称。

调度程序示例

在此例中，我们将routine_string设置成相应的例程名称，然后设置成从5进行到1，从而让主任务程序调用了相关后台任务中的多则例程。通过这种方式，主任务程序可初始化成以下状态：相关后台任务程序宜首先执行例程clean_gun，然后执行routine1。

主任务程序：

```
MODULE module1
  PERS string routine_string:="";

  PROC main()
    !Call clean_gun in background task
    routine_string:="clean_gun";
    SetDO do5,1;
    WaitDO do5,0;

    !Call routine1 in background task
    routine_string:="routine1";
    SetDO do5,1;
    WaitDO do5,0;
    ...
  ENDPROC
ENDMODULE
```

后台任务程序：

```
MODULE module2
  PERS string routine_string:="";

  PROC main()
    WaitDO do5,1;
    %routine_string%;
    SetDO do5,0;
  ENDPROC

  PROC clean_gun()
    ...
  ENDPROC

  PROC routine1()
    ...
  ENDPROC
```

下一页继续

```
ENDMODULE
```

6.2.5 其它编程问题

6.2.5.1 在各项任务之间共享资源

指明资源已被占用的旗标

所有任务都能使用FlexPendant示教器、文件系统和I / O信号等系统资源，不过若有多段任务程序在使用同一资源，那么请确保它们是在轮流使用该资源，而不是在同时使用该资源。

若要避免两段任务程序同时使用同一资源，就用一个旗标来指明该资源已被使用。当任务程序正在使用该资源时，用户可将一个布尔变量设置成“真（true）”。

若要加快这一处理过程，则请使用指令TestAndSet。该指令将首先测试相关旗标，如果此旗标为“假（false）”，那么该指令就会将此旗标设置成“真”，并返回“真”；否则就返回“假”。

涉及旗标的示例与TestAndSet

在此例中，有两段任务程序试图在FlexPendant示教器上各写入三行。如果未使用旗标，那么这些行就存在彼此混合的风险；而在使用了旗标后，最先执行TestAndSet指令的任务程序就会最先写入所有三行，而另一段任务程序则会等到相关旗标被设置成“假”为止，然后写入自己的所有三行。

主任务程序：

```
PERS bool tproutine_inuse := FALSE;
...
WaitUntil TestAndSet(tproutine_inuse);
TPWrite "First line from MAIN";
TPWrite "Second line from MAIN";
TPWrite "Third line from MAIN";
tproutine_inuse := FALSE;
```

后台任务程序：

```
PERS bool tproutine_inuse := FALSE;
...
WaitUntil TestAndSet(tproutine_inuse);
TPWrite "First line from BACK1";
TPWrite "Second line from BACK1";
TPWrite "Third line from BACK1";
tproutine_inuse := FALSE;
```

6.2.5.2 测试任务是否控制着机械单元

用于询问的两则函数

某些函数会检查相关任务程序是否已控制了任何机械单元（TaskRunMec）或控制了一台机器人（TaskRunRob）。

如果相关任务程序控制了一台机器人或其它机械单元，那么TaskRunMec将返回“真”；如果相关任务程序控制了一台带TCP的机器人，那么TaskRunRob仅会返回“真”。

使用MultiMove时TaskRunMec和TaskRunRob 会有所帮助。若拥有MultiMove，您就能让若干项任务控制多个机械单元。具体请参见。

> **注意**
>
> 就控制了一台机器人的某项任务而言，参数 *Type* 必须被设置成正常，而且类型 *MotionTask*必须被设置成 YES。具体请参见 第231页的系统参数。

涉及TaskRunMec和TaskRunRob的示例

此例中设置了外部设备的最大速度。如果相关任务程序控制了一台机器人，那么外部设备的最大速度就会被设置成该机器人的最大速度。如果相关任务程序控制了一部非机器人的外部设备，那么其最大速度就会被设置成5000毫米 / 秒。

```
IF TaskRunMec() THEN
  IF TaskRunRob() THEN
    !If task controls a robot
    MaxExtSpeed := MaxRobSpeed();
  ELSE
    !If task controls other mech unit than robot
    MaxExtSpeed := 5000;
  ENDIF
ENDIF
```

6.2.5.3 taskid

taskid的语法

一项任务始终有一个类型为taskid的预定义变量（由该任务的名称和后缀“Id”组成），比如MAIN任务的变量名称就是MAINId。

代码示例

在此例中，尽管有另一项任务执行了Save指令，但还是把模块PART_A保存在了任务BACK1中。

BACK1Id是一个类型为taskid的变量。系统会自动声明该变量。

```
Save \TaskRef:=BACK1Id, "PART_A"
  \FilePath:="HOME:/DOORDIR/PART_A.MOD";
```

6.2.5.4 避免冗长环路

后台任务继续形成环路

继续正常执行一段任务程序。这意味着某段后台任务程序不但是有效的，还形成了一段永恒环路。如果该程序没有任何等候指令，那么所述后台任务就可能占用过多的计算机能力，并导致相关控制器无法处理其它任务。

示例

```
MODULE background_module
  PROC main()
    WaitTime 1;
    IF di1=1 THEN
      ...
    ENDIF
  ENDPROC
ENDMODULE
```

如果此例中没有等候指令且di1为0，那么该后台任务的环路就会在一无所获的情况下完全占用相关计算机的能力。

此页刻意留白

7 Communication

7.1 FTP&SFTP client [3116-1]

7.1.1 FTP&SFTP client简介

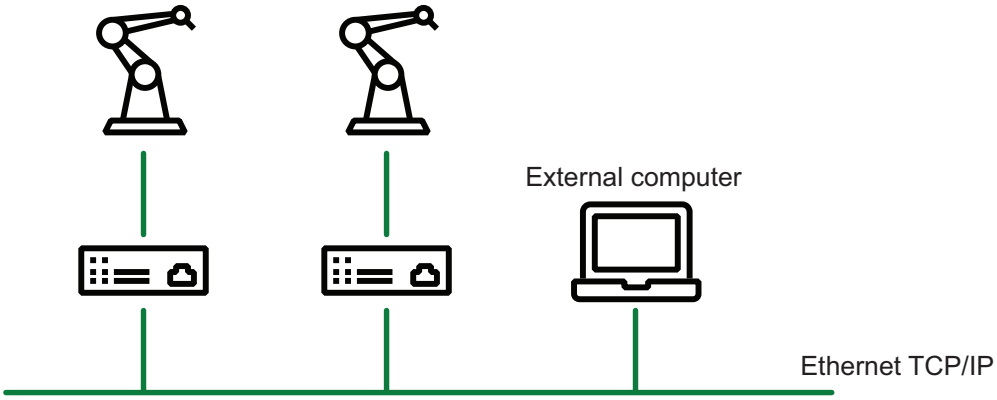
目的

FTP&SFTP Client 旨在使机器人可以访问远程挂载磁盘，例如 PC 上的硬盘驱动器。

此处是一些应用示例：

- 备份到一台远程计算机中。
- 从一台远程计算机载入程序。

网络图



xx1900001202

描述

多台机器人可通过网络来访问同一台计算机。

一旦配置了 FTP/SFTP 协议，用户就能像访问控制器的内部硬盘那样来访问远程计算机。

其中包括

RobotWare 选项 *FTP&SFTP client* 使您能够访问 *FTP Client* 和 *SFTP Client* 类型的系统参数。

基本方法

这是使用 FTP&SFTP client 的常规方法。

1. 请配置 FTP/SFTP 协议，以便在远程计算机上指明可从机器人访问的磁盘或目录。
2. 像读取和写入控制器内部硬盘那样来读取和写入相应的远程计算机。

SFTP支持下列服务器：

- Rebex第1.0.3版
- 完整的FTP第11.0.0版
- Cerberus第9.0.4.0版

下一页继续

7.1.1 FTP&SFTP client简介
续前页

在特定SFTP服务器（如完整的SFTP服务器）中，存在一种配置设置，即空闲连接超时，该设置定义了某一连接的最长空闲时间。若在此时间间隔期间未发出客户端请求，则连接闭合。将参数设定为无超时将有助于维持连接，即使并未发出客户端请求。

要求

该外部计算机必须具有：

- TCP/IP堆栈
- FTP 服务器或 SFTP 服务器

FTP服务器上的目录列表样式

FTP服务器必须以UNIX样式列出目录。

例子：

```
drwxrwxrwx 1 owner group 25 May 18 16:39 backups
```

MS-DOS样式无效。

提示

对于Windows中的互联网信息服务（IIS），目录列表样式可以进行配置。

限制

- 当使用 FTP client 时，文件名的最大长度为 99 个字符。
- 当使用 FTP client 时，包括文件名称在内的一条文件路径最多长达 200 个字符。整条路径——而不是仅仅是服务器路径——都要包括在这 200 个字符中。当命令备份到挂载磁盘上时，相关最大路径必须包括此次备份中创建的所有目录。
- 当使用 SFTP Client 时，包括文件名称在内的一条文件路径最多长达 248 个字符。整条路径——而不是仅仅是服务器路径——都要包括在这 248 个字符中。当命令备份到挂载磁盘上时，相关最大路径必须包括此次备份中创建的所有目录。

FTP 示例

参数	值
Name	myFTP
Server path	C:\robot_1

- 备份保存在 *myFTP/Backups/Backup_20130109*中
  （27个字符）
- 个人电脑上的路径为 *C:\robot_1\Backups\Backup_20130109*
  （34个字符）
- 该备份中的最长文件路径为
  C:\robot_1\Backups\Backup_20130109\RAPID\TASK1\PROGMOD\myprogram.mod
  （54 + 13个字符）

此例中的最大路径长度乍眼一看似乎是27个字符，单实际上是67个字符。

下一页继续

SFTP 示例

参数	值
Name	mySFTP

- 备份保存在 *mySFTP/Backups/Backup_20130109*中
  （27个字符）
- 个人电脑上的路径为 *\Backups\Backup_20130109*
  （24个字符）
- 该备份中的最长文件路径为
  \Backups\Backup_20130109\RAPID\TASK1\PROGMOD\myprogram.mod
  （44 + 13个字符）

此例中的最大路径长度乍眼一看似乎是27个字符，但实际上是57个字符。

系统参数

请参见技术参考手册 - 系统参数。

7.2 NFS Client [3117-1]

7.2.1 NFS Client介绍

目的

NFS Client的作用是启用相关机器人来访问远程挂载磁盘，比如一台PC上的某个硬盘驱动器。

此处是一些应用示例：

- 备份到一台远程计算机中。
- 从一台远程计算机载入程序。

描述

多台机器人可通过网络来访问同一台计算机。

一旦配置了NFS应用层协议，用户就能像访问控制器的内部硬盘那样来访问相应的远程计算机。

其中包括

RobotWare 选项 *NFS Client* 使您能够访问 *Application protocol* 类型的系统参数以及它的下列参数：*Name*、*Type*、*Transmission protocol*、*Server address*、*Server type*、*Trusted*、*Local path*、*Server path*、*User ID*、*Group ID* 和 *Show Device*.

基本方法

这是使用 NFS Client 的常规方法。

1 请配置 NFS 协议，以便在远程计算机上指明可从机器人访问的磁盘或目录。
2 像读取和写入控制器内部硬盘那样来读取和写入相应的远程计算机。

操作前提

该外部计算机必须具有：

- TCP/IP堆栈
- NFS服务器

限制

当使用NFS Client时，包括文件名称在内的一条文件路径最多长达248个字符。整条路径——而不是仅仅是服务器路径——都要包括在这248个字符中。当命令备份到挂载磁盘上时，相关最大路径必须包括此次备份中创建的所有目录。

示例

参数	值
Name	myNFS
Server path	C:\robot_1

- 备份保存在myNFS/Backups/Backup_20130109中
  （27个字符）
- 该PC上的路径将是C:\robot_1\Backups\Backup_20130109
  （34个字符）

下一页继续

- 该备份中的最长文件路径为
  C:\robot_1\Backups\Backup_20130109\RAPID\TASK1\PROGMOD\myprogram.mod
  （54 + 13个字符）

此例中的最大路径长度乍眼一看似乎是27个字符，单实际上是67个字符。

此页刻意留白

8 User Interaction Application

8.1 RobotStudio Connect [3119-1]

概述

RobotStudio 是用于 OmniCore 控制器的编程、配置和调试工具。RobotStudio 直接对控制器中的活动数据产生作用，可实现 RAPID 编程、系统软件更新/启动和系统配置等活动。在默认情况下，将 RobotStudio 连接到本地管理端口处于启用状态，但通过公共网络连接 RobotStudio 则需要采用 *RobotStudio Connect*.选项。

8.2 FlexPendant Base Apps

Limited App Package [3120-1]

Limited App Package 选项包含了操作机器人系统所需的基本功能。FlexPendant 的此基础版本软件可实现大部分重要功能，比如微动机器人、校准机器人、基本操作（启动、停止、加载程序）、读取和写入 I/O 信号、事件日志和操作员消息。

Essential App Package [3120-2]

Essential App Package 选项包含了将使利用机器人进行作业变得更轻松、更高效的多项功能。微动功能已通过 3D 图进行了改进，仪表盘可以使您轻松地一眼洞悉系统状态。这包括 *Limited App Package* 选项。

8.3 FlexPendant Independent Apps

Program Package [3151-1]

若要在 FlexPendant 上创建新 RAPID 程序和编辑现有 RAPID 程序，则必须采用 *Program Package* 选项。如果没有为 FlexPendant 选择该程序包选项，则必须在单独的 PC 上运行 RobotStudio 软件，然后才可以创建和编辑 RAPID 程序。

FlexPendant 选项没有与 FlexPendant 硬件绑定，而是与 OmniCore 控制器绑定。这意味着，FlexPendant 可以运行已授权给与其相连之控制器的应用程序。因此，共享的同一个 FlexPendant 可以在不同的机器人上具有不同的应用程序。

此页刻意留白

9 Engineering tools

9.1 RobotWare Add-In

对已授权加载项而言为必需项。

9.2 Path Corrections [3123-1]

9.2.1 概述

目的

Path Corrections 旨在实现根据来自传感器的输入在线地调整机器人路径。借助 Path Corrections 提供的指令集，用户可以对机器人路线进行比较，并利用来自传感器的输入对其进行调整。

其中包括

您可通过 RobotWare 选项Path Corrections来访问：

- 数据类型corrdescr
- 指令CorrCon、CorrDiscon、CorrClear和CorrWrite
- 函数CorrRead

基本方法

这是设置 Path Corrections 的一般方式。有关完成此设置的详细示例，请参阅 第259页的代码示例。

1. 声明相应的校正发生器。
2. 关联相应的校正发生器。
3. 定义一则决定了相关偏移量的软中断例程（该例程会被写入相应的校正发生器中）。
4. 定义一项用来频繁调用该软中断例程的中断。
5. 用该校正项调用一条移动指令。系统会反复校正相关路径。

> **注意**
>
> 说明书 CorrWrite 针对低速，而且针对适中的修正数值。太剧烈的数值可能造成卡顿。应在 RobotStudio 中对修正数值进行测试，以确认其性能。

> **注意**
>
> 如果用 \Corr 开关相互颠倒地调用了两条或更多条移动指令，那么很重要的一点就是了解相关机器人每次从一个精确点启动时是否重置了所有 \Corr 偏移量。所以在使用精确点时，相关控制器在第二条 Move 指令期间并不清楚相关路径是否已有偏移。为避免行为失常，我们建议 \Corr 开关仅与区域一同使用，而不要使用精确点。

限制

有可能同时关联上若干个校正发生器（比如一个沿Z轴校正，另一个沿Y轴校正），但最多只能同时关联5个校正发生器。

若重启了控制器，则必须重新定义相应的校正发生器。当控制器重启后，系统不会保留之前的定义和关联。

这些指令仅能用于运动任务。

9.2.2 RAPID组件

数据类型

此处简述了*Path Corrections*选项中的每种数据类型。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各种数据类型。

数据类型	描述
corrdescr	corrdescr是一个校正发生器描述符，其用途是作为相应校正发生器的引用项。

指令：

此处简述了*Path Corrections*选项中的每条指令。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

指令	描述
CorrCon	CorrCon会激活路径校正。调用CorrCon将会关联上一个校正发生器。一旦关联成功，系统就能用新的偏移输入项（比如来自传感器的偏移输入项）来不断校正相关路径。
CorrDiscon	CorrDiscon会停用路径校正。调用CorrDiscon将会断开一个校正发生器。
CorrClear	CorrClear会停用路径校正。调用CorrClear将会断开所有校正发生器。
CorrWrite	CorrWrite会设置相应的路径校正值。调用CorrWrite将会为一个校正发生器设置相应的偏移值。

函数

此处简述了*Path Corrections*选项中的每则函数。更多信息请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各则函数。

功能	描述
CorrRead	CorrRead会读取一个校正发生器所作的总校正。

9.2.3 相关的RAPID功能

自变数\Corr

可为部分移动指令设置\Corr。当执行这些移动指令时，系统便会启用路径校正。

下列指令拥有可选自变数\Corr：

- MoveL
- MoveC
- SearchL
- SearchC
- TriggL（仅当相关控制器配备了基本功能Fixed Position Events时）
- TriggC（仅当相关控制器配备了基本功能Fixed Position Events时）
- CapL（仅当相关控制器配备了选项Continuous Application Platform时）
- CapC（仅当相关控制器配备了选项Continuous Application Platform时）
- ArcL（仅当相关控制器配备了选项RobotWare Arc时）
- ArcC（仅当相关控制器配备了选项RobotWare Arc时）

至于这些指令方面的更多信息，则请参见技术参考手册 - *RAPID*指令、函数和数据类型中的各条指令。

中断

若要使用 Path Corrections 来创建程序，您需要具备处理中断的能力。有关中断的更多信息，请参阅 技术参考手册 - *RAPID Overview*。

9.2.4 代码示例

带校正的直线移动

此例展示了如何编写一条带在线路径校正的线性路径。这里的做法是每秒中断5次，并调用一则软中断例程来校正偏移量。

程序代码

```
VAR intnum int_no1;
VAR corrdescr id;
VAR pos sens_val;
PROC PathRoutine()
  !Connect to the correction generator
  CorrCon id;

  !Setup a 5 Hz timer interrupt.
  CONNECT int_no1 WITH UpdateCorr;
  ITimer\Single, 0.2, int_no1

  !Position for start of contour tracking
  MoveJ p10,v100,z10,tool1;

  !Run MoveL with correction.
  MoveL p20,v100,z10,tool1\Corr;

  !Remove the correction generator.
  CorrDiscon id;

  !Remove the timer interrupt.
  IDelete int_no1;
ENDPROC
TRAP UpdateCorr
  !Call a routine that read the sensor
  ReadSensor sens_val.x, sens_val.y, sens_val.z;

  !Execute correction
  CorrWrite id, sens_val;

  !Setup interrupt again
  IDelete int_no1;
  CONNECT int_no1 WITH UpdateCorr;
  ITimer\Single, 0.2, int_no1;
ENDTRAP
```

9.3 Auto Acknowledge Input

描述

RobotWare 基本功能 *Auto Acknowledge Input* 是一个启用系统输入的选项。当用机器人控制器上的钥匙开关从操作员手动模式切换为自动模式时，该选项会启用一条系统输入信息，以确认 FlexPendant 上出现的对话框。

警告

请注意，使用此类输入项将违反相关安全标准ISO 10218-1的5.3.5 Single point of control一章中的规定。其规定内容如下：

"*The robot control system shall be designed and constructed so that when the robot is placed under local pendant control or other teaching device control, initiation of robot motion or change of local control selection from any other source shall be prevented.*"

由此可见，用户绝对有必要采用其它安全手段来继续满足相关标准与机械指令的各项要求，同时也绝对有必要对整个围笼进行风险评估。系统集成人员将负责此类额外安排和风险评估，另外除非已完成了这些行动，否则不得启用相关系统。

限制

不能用FlexPendant或RobotStudio来定义该系统参数（只能用I / O配置文件中的一段文本字符串）。

激活Auto Acknowledge Input

必须为机器人系统安装*Auto Acknowledge Input*选项（在Installation Manager中启用）。

用以下无返回值程序来激活*Auto Acknowledge Input*所需的系统输入项。

	操作
1	用FlexPendant或RobotStudio保存I / O配置文件*eio.cfg*的一份副本。
2	用一个文本编辑器编辑I / O配置文件*eio.cfg*在组SYSSIG_IN中添加以下行： `-Signal "my_signal_name" -Action "AckAutoMode"` 宜作为系统输入项的已配置数字输入信号被命名为`my_signal_name`。
3	保存该文件，然后在控制器上重新加载该文件。
4	重启系统，以此激活该信号。

索引

A
Absolute Accuracy, 153
Absolute Accuracy补偿, 161
Absolute Accuracy验证, 163
Advanced RAPID, 17
Advanced Shape Tuning, 173
AliasIO, 25
Analog Signal Interrupt, 48
ArgName, 46
Auto acknowledge input, 12, 260

B
BitAnd, 19
BitCheck, 19
BitClear, 19
BitLSh, 19
BitNeg, 19
BitOr, 19
BitRSh, 19
BitSet, 19
BitXOr, 19
BookErrNo, 41
byte, 19
ByteToStr, 19

C
CalibWare, 155
Check unresolved references, Task type, 231
CirPathMode, 189
ClearRawBytes, 118
Close, 114
CloseDir, 122
Collision Avoidance, 212
Collision Detection Memory, 204
Collision Error Handler, 205
CopyFile, 122
CopyRawBytes, 118
CorrClear, 257
CorrCon, 257
corrdescr, 257
CorrDiscon, 257
CorrRead, 257
CorrWrite, 257
Cyclic bool, 101
Cyclic bool settings, 106
Cyclic bool系统参数, 106

D
datapos, 22
dir, 122

E
errdomain, 38
ErrRaise, 38
errtype, 38
Event Preset Time, 127

F
Fieldbus Command Interface, 108
FIFO, 137
FileSize, 122
FlexPendant示教器, 240
FricIdEvaluate, 179
FricIdInit, 179
FricIdSetFricLevels, 179
Friction FFW Level, 177
Friction FFW On, 177
Friction FFW Ramp, 177
FSSize, 122

G
General RAPID, 205
GetDataVal, 22
GetMaxNumberOfCyclicBool, 107
GetNextCyclicBool, 107
GetNextSym, 22
GetNumberOfCyclicBool, 107
GetTrapData, 38

I
IError, 38
IndAMove, 218
IndCMove, 218
Ind collision stop without brake, 204
IndDMove, 218
Independent Joint, 217
Independent Lower Joint Bound, 217
Independent Upper Joint Bound, 217
IndInpos, 218
IndReset, 218
IndRMove, 218
IndSpeed, 218
iodev, 114
IPers, 38
IRMQMessage, 140
IsCyclicBool, 107
IsFile, 122
ISignalAI, 49
ISignalAO, 49
IsStopStateEvent, 46

J
Jog Collision Detection, 204
Jog Collision Detection Level, 204
joint zones, 193

L
Logical Cross Connections, 130

M
Main entry, Task type, 231
MakeDir, 122
Manipulator Supervision, 204
Manipulator Supervision Level, 204
Motion Planner, 204
Motion Process Mode, 180
MotionSup, 206, 210
Motion Supervision, 204
Motion Supervision Max Level, 204
Motion System, 204
MotionTask, Task type, 231
MotSupOn, 207
MotSupTrigg, 207
MoveCSync, 127
MoveJSync, 127
MoveLSync, 127
Multitasking, 229

N
NFS Client, 248
NORMAL, 231
NoSafety, 231

NOT, 132

O
Open, 114
OpenDir, 122

P
PackDNHeader, 109
PackRawBytes, 118
Path Collision Detection, 204
Path Collision Detection Level, 204
pathrecid, 222
PathRecMoveBwd, 222
PathRecMoveFwd, 222
Path Recovery, 221
PathRecStart, 222
PathRecStop, 222
PathRecValidBwd, 222
PathRecValidFwd, 222
PC SDK客户端, 136
PFRestart, 31

R
r1_calib, 156
RAPID消息队列, 135
RAPID组件
  多任务化, 232
  高级RAPID语言, 46
RAPID配套功能, 45
rawbytes, 118
RawBytesLen, 118
ReadAnyBin, 114
ReadBin, 114
ReadCfgData, 28
ReadDir, 122
ReadErrData, 38
ReadNum, 114
ReadRawBytes, 118
ReadStr, 114
ReadStrBin, 114
RemoveAllCyclicBool, 107
RemoveCyclicBool, 107
RemoveDir, 122
RemoveFile, 122
RenameFile, 122
restartdata, 34
RestoPath, 222
Rewind, 114
RMQFindSlot, 140
RMQGetMessage, 140
RMQGetMsgData, 140
RMQGetMsgHeader, 140
RMQGetSlotName, 140
rmqheader, 140
rmqmessage, 140
RMQSendMessage, 140
RMQSendWait, 140
rmqslot, 140
RMQ最大消息大小, 139
RMQ最大消息数目, 139
RMQ模式, 139
RMQ清空队列, 140
RMQ类型, 139
RMQ读取等候, 140

S
SafeMove Assistant, 213
SEMISTATIC, 231
SetAllDataVal, 22
SetDataSearch, 22
SetDataVal, 22
SetSysData, 46
SetupCyclicBool, 107
shapedata, 195
SocketAccept, 148
SocketBind, 148
SocketClose, 148
SocketConnect, 148
SocketCreate, 148
socketdev, 148
SocketGetStatus, 149
SocketListen, 148
Socket Messaging, 145
SocketReceive, 148
SocketSend, 148
socketstatus, 148
STATIC, 231
StepBwdPath, 34
StorePath, 222
StrToByte, 19
syncident, 236
syncident, data type, 232
SyncMoveResume, 222
SyncMoveSuspend, 222
SysFail, 231
SysHalt, 231
SysStop, 231

T
Task, Task type, 231
Task, type, 231
taskid, 232, 242
taskid, data type, 232
Task in foreground, Task type, 231
TaskRunMec, 241
TaskRunMec, function, 232
TaskRunRob, 241
TaskRunRob, function, 232
tasks, 236
  data type, 232
tasks, data type, 232
TestAndSet, 240
TestAndSet, function, 232
TextGet, 41
TextTabFreeToUse, 41
TextTabGet, 41
TextTabInstall, 41
Transmission Gear High, 217
Transmission Gear Low, 217
trapdata, 38
TriggC, 127
TriggCheckIO, 126
triggdata, 126
TriggEquip, 126
TriggInt, 126
TriggIO, 126
triggios, 126
triggiosdnum, 126
TriggJ, 127
TriggL, 127
TriggLIOs, 127
TriggRampAO, 127
TriggSpeed, 34
TriggStopProc, 34

triggstrgo, 126
TrustLevel, Task type, 231
TUNE_FRIC_LEV, 176
TUNE_FRIC_RAMP, 176
TuneServo, 176
Type, Task type, 231

U
UnpackRawBytes, 118

W
WaitSyncTask, 236
WaitSyncTask, instruction, 232
WaitUntil, 234
WarmStart, 28
world zones, 193
Wrist Move, 187
Write, 114
WriteAnyBin, 114
WriteBin, 114
WriteCfgData, 28
WriteRawBytes, 118
WriteStrBin, 114
WZBoxDef, 195
WZCylDef, 195
WZDisable, 196
WZDOSet, 196
WZEnable, 196
WZFree, 196
WZHomeJointDef, 196
WZLimJointDef, 195
WZLimSup, 196
WZSphDef, 195
wzstationary, 195
wztemporary, 195

Z
zones, 193

不
不可打印字符, 147

与
与（AND）, 131

丢
丢失消息, 138
丢失队列, 138

丧
丧失准确度, 159

中
中断, 48, 137, 234
中断功能, 37

临
临时全局区域, 195

事
事件消息, 40
事件编号, 40

二
二进制数据, 147
二进制通信, 113

交
交叉连接, 130

以
以太网, 245, 248

任
任务, 229

传
传感器, 256

位
位功能, 18
位置事件, 125

使
使用机器人校准, 156

例
例程调用, 238

信
信号, 234, 238

俯
俯仰, 160

偏
偏移, 161
偏转, 160

停
停用监控, 210

公
公用数据, 233

共
共享资源, 240

准
准确度的错误来源, 160

出
出厂证书，Absolute Accuracy, 165

函
函数
  多任务化, 232
  高级RAPID语言, 46

切
切割形状, 192
切割面, 188

别
别名IO, 24

原
原始数据, 117

参
参数
  准确度补偿, 166

合
合成信号, 130–131
合规性错误, 160

同
同步多项任务, 236

围
围笼的对准, 167

固
固定位置事件, 125
固定全局区域, 195
固定装置的对准情况, 168

坐
坐标系, 167

基
基于字符的通信, 113

声
声明, 233

外
外轴, 200, 215

字
字符串终止, 147

对
对准, 167

导
导轨, 200

工
工具, 155
工具校准, 171
工艺配套功能, 33

已
已被抛弃的消息, 138
已记录的路径, 225

开
开源软件、OSS, 13

微
微调, 210
微调，手动, 176
微调，自动, 174

恢
恢复信号, 35
恢复路径, 221

成
成比例信号, 34

或
或（OR）, 131

手
手动微调摩擦, 176

执
执行信号, 130–131

扭
扭矩, 202

指
指令
　多任务化, 232
　高级RAPID语言, 46

摩
摩擦等级微, 174
摩擦补偿, 173

数
数字I / O信号, 130
数据, 136
数据搜索功能, 21
数据类型
　多任务化, 232

文
文件管理, 121
文件结构, 121
文件通信, 112
文本表格文件, 40

断
断电功能, 31

旋
旋转变压器偏移校准, 163

更
更换, 157
更改校准数据, 156

未
未校准, 156

机
机器人的对准情况, 169
机械单元, 241
机械臂的更换, 158

校
校准工具, 155, 171
校准数据, 156
校准进程, 163
校正发生器, 256
校正自变数, 258

框
框架, 167
框架关系, 170

模
模拟信号, 48

横
横滚, 160

永
永久变量, 233

测
测量系统, 218

消
消息合并, 147

激
激活Absolute Accuracy, 156
激活监控, 210

点
点动碰撞检测, 208
点动碰撞检测等级, 208

独
独立关节, 200
独立移动, 215
独立轴, 215

用
用户消息功能, 40

电
电机的更换, 157

监
监控等级, 204, 206, 210

目
目录管理, 121

确
确认消息, 147

碰
碰撞, 201
碰撞检测
　YuMi 机器人, 199

移
移动, 201

程
程序指针, 46

第
第三方软件, 13

等
等候若干任务, 236

系
系统参数
　多任务化, 231
　配置功能, 27
系统资源, 240

绝
绝对准确度校准, 163

维
维护, 157

编
编组I / O信号, 130

腕
腕的更换, 157

臂
臂的更换, 157

自
自动微调摩擦, 174
自变数名称, 46

虚
虚假目标点, 161

补
补偿, 161
补偿参数, 153

触
触发, 211

记
记录, 136

许
许可证, 13

设
设置“碰撞检测”, 208

证
证书， Absolute Accuary, 165

误
误触发, 211

调
调度程序, 238

负
负载识别, 155

路
路径, 31
路径偏移, 256
路径校正, 256
路径碰撞检测, 208
路径碰撞检测等级, 208
路径记录, 225

轮
轮询, 234

软
软中断例程, 137
软件许可证, 13
软伺服器, 200

轴
轴, 215
轴重置, 215

载
载入校准数据, 156

运
运动学错误, 160

通
通信, 112

速
速度, 202

逻
逻辑AND, 132
逻辑OR, 132
逻辑运算, 130

配
配置
　Absolute Accuracy, 156
配置“碰撞检测”, 208

配置功能, 27
配置参数, 166

重
重置, 218
重置轴, 215

错
错误中断, 37

队
队列名称, 137
队列处理, 137

验
验证, 163

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS
Robotics & Discrete Automation
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics & Discrete Automation
No. 4528 Kangxin Highway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics & Discrete Automation
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics

3HAC066554-010, Rev J, zh-cn