SF512x: Kinematics

<SF512x: Kinematics>

作者	
日期	
版本	V1.0.0

中科时代 SINSEGYE | 基于PC技术的工智机新时代

深圳市南山区粤海街道百度国际大厦西塔楼

官网：www.sinsegye.com.cn

Email		邮箱：Sales@sinsegye.com.cn 热线电话：400-013-2158

中科时代 SINSEGYE | 基于PC技术的工智机新时代

SF512x运动学变换器使用介绍

前言

一、文件说明

本说明专为熟悉相关国家标准且经过专业培训的控制与自动化技术专家而制定。

在安装与调试部件时，务必仔细审阅所有相关文件及以下说明。

合格人员应始终采用最新的有效文档进行操作。

责任人员必须确保所述产品的应用或使用完全符合所有安全要求，涵盖所有相关法律法规、指导原则及标准。

1、免责声明

本文件经过精心编制，但鉴于所描述产品处于持续的开发与升级过程中，中科时代（深圳）计算机系统有限公司保留随时对文件进行修改和更新的权利，且无需事先通知。请注意，禁止依据数据图及本文件描述对已交付的产品进行任何改动。

对于因使用或信赖本手册所载明或未明示的信息而造成的任何损失或损害，中科时代计算机系统有限公司不承担任何责任。

2、版权所有

二、安全声明

1、安全规程

为了您的安全，请阅读以下说明。始终遵守产品特定的安全说明，您可以在本文档的适当位置找到这些说明。

2、责任免除

所有组件都提供了硬件和软件配置。不允许对文件中所述以外的硬件或软件配置进行修改，中科时代不对文件所述外的硬件或软件负责。

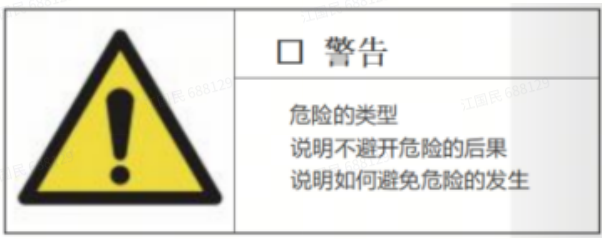
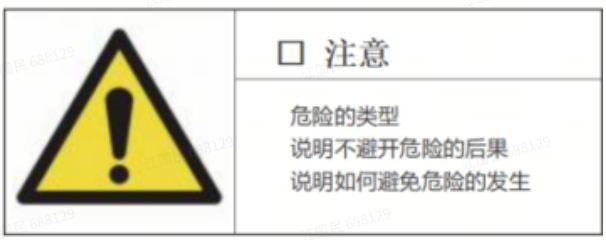
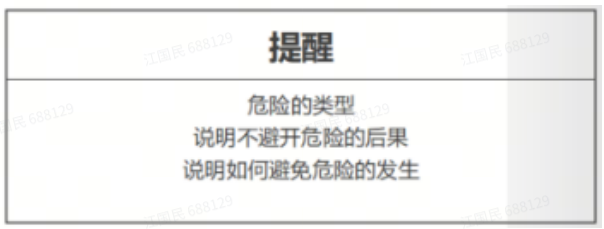
3、人员资格

本说明仅适用于熟悉适用国家标准的经过培训的控制、自动化和驱动技术专家。

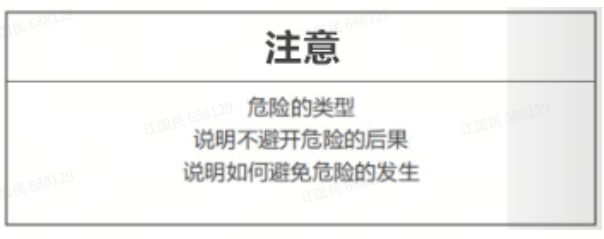
4、信号词

文档中使用的信号词分类如下。为了防止人员和财产受到伤害和损害，请阅读并遵守安全和警告通知。

5、个人伤害警示

	警告表示一种潜在的危险情况，如果不加以避免，可能会导致严重的伤害或死亡。
	注意表示潜在的危险情况，如果不避免，可能会导致轻度受伤或中度受伤，或导致设备损坏。
	注意表示一种潜在的危险情况，如果不加以避免，可能只导致设备的损坏。

6、对财产或环境造成损坏的警告

	环境、设备或数据可能会被损坏。

7、产品处理信息

例如，这些信息包括：行动建议、援助或有关产品的进一步信息。

概述

一、SF512x运动学变换器

本产品是一款专为工业自动化应用设计的高性能运动学变换工具，将机器人控制与传统PLC集成于单一系统。通过在同一系统内实现整个控制功能，消除了PLC、运动控制和机器人控制之间不同CPU接口导致的性能损失。

本产品提供高精度的正向运动学与逆向运动学计算功能，支持4D-SCARA机器人在复杂工作环境中的精确位置和姿态控制。通过参数化设计，支持多种机器人配置，包括臂长、角度范围、负载能力等，以适应不同工作需求。

二、坐标系概述

在描述系统的运动和位置行为时，坐标系是必不可少的。以下是常用的几种坐标系及其特点：

- 机械坐标系(MCS - Machine Coordinate System)是以机器人为基础的笛卡尔坐标系，通常原点位于机器人的基座上。
- 世界坐标系(WCS - World Coordinate System)是描述整个建模“世界”的笛卡尔坐标系，与特定机器人无直接关联。机器人机械坐标系 (MCS) 的原点位于世界坐标系中的某一特定点，用户可以定义机器人在“世界”中的位置和朝向。一个WCS可包含多个机器人。使用机器人时，世界坐标系与机械坐标系可以重合，以提高透明度。
- 工件坐标系(PCS - Program Coordinate System)可以在世界坐标系内的任意位置和任意方向进行设置。适合于复杂工作环境，比如多个工作区域或多个工件操作的场景。
- 轴坐标系 (ACS - Axis Coordinate System)描述机器人物理轴的运动位置，通常不是笛卡尔坐标系。许多机器人关节轴以旋转运动为主，因此轴坐标系更适合描述旋转角度、速度和加速度的极限值。轴坐标系通常用于参考点定位（参考/归位操作）。

三、运动学中的坐标变换

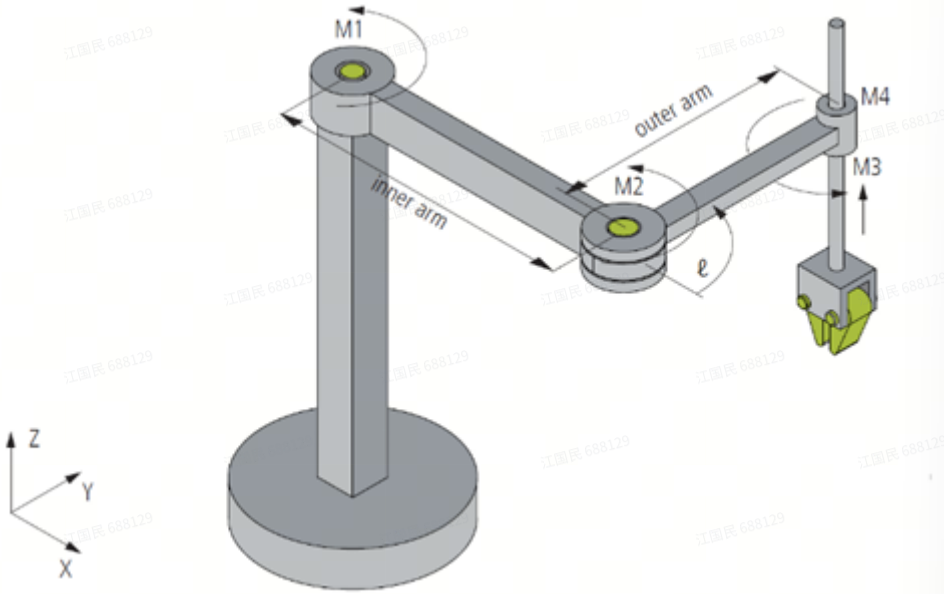
在工业机器人编程中，通常以机械坐标系 (MCS)为基础。由于人类更直观地使用笛卡尔坐标系（如X、Y、Z轴）进行思考和规划，因此在执行运动时，必须在轴坐标系 (ACS) 和笛卡尔空间之间进行坐标转换。

运动学中的变换描述了从一个坐标系切换到另一个坐标系所需的计算过程。机器人运动学中的坐标变换主要涉及以下两种情况：

- 正向变换是指从轴坐标系 (ACS) 到笛卡尔坐标系的转换。刀具中心点 （TCP） 的笛卡尔位置是根据机器人的轴特定关节坐标计算得出的。

- 逆向变换是指将笛卡尔坐标系中的TCP位置转换到轴坐标系 (ACS)。为了驱动机器人实际运动，需要将期望的TCP位置和姿态转化为每个关节的轴坐标参数。

四、4D-SCARA机器人(S_CCZC)



4D-SCARA（选择性顺应装配机器人臂）是一种具有四自由度（DOF）的机器人，其运动学结构和电机轴配置如图所示。电机轴M1、M2和M4为旋转轴，单位为角度（°），箭头指示其正向旋转方向。第三个电机轴M3为线性轴，单位为毫米（mm），用于实现垂直方向的运动。

机械坐标系（MCS）的原点位于第一个关节（M1）处。当所有旋转电机轴（M1、M2、M4）均处于0°时，SCARA机械臂指向的方向即为X轴。此配置定义了MCS的X轴方向，Y轴与X轴垂直，形成一个右手笛卡尔坐标系，Z轴则为垂直方向，与M3的线性运动一致。

这种配置方式确保了高精度的路径规划和运动控制，并为机器人在工业任务中的高效表现提供了基础。

安装卸载

一、安装要求

本节描述工程和/或运行时系统所需的最低要求。

1、硬件要求

支持 MetaFacture 平台的设备（ PLC 或嵌入式系统）。

2、软件要求

已安装 MetaFacture 的开发环境。

操作系统支持：MetaOS或其他兼容平台。

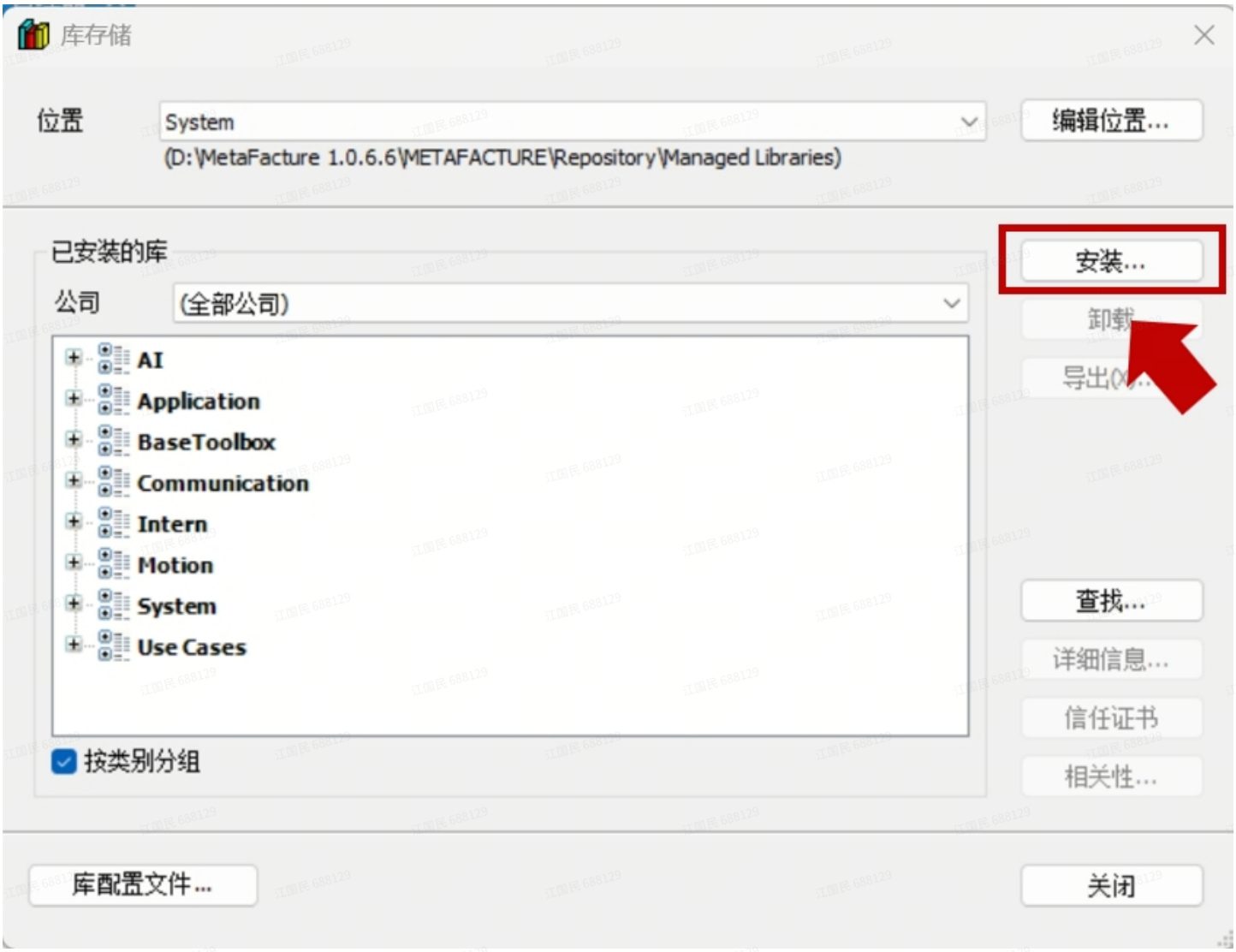
必须具备相关的库或模块依赖（如基础 I/O 库）

二、安装过程

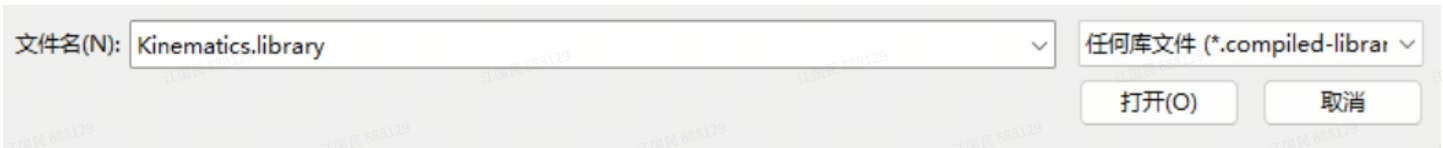
1、通过库管理器安装

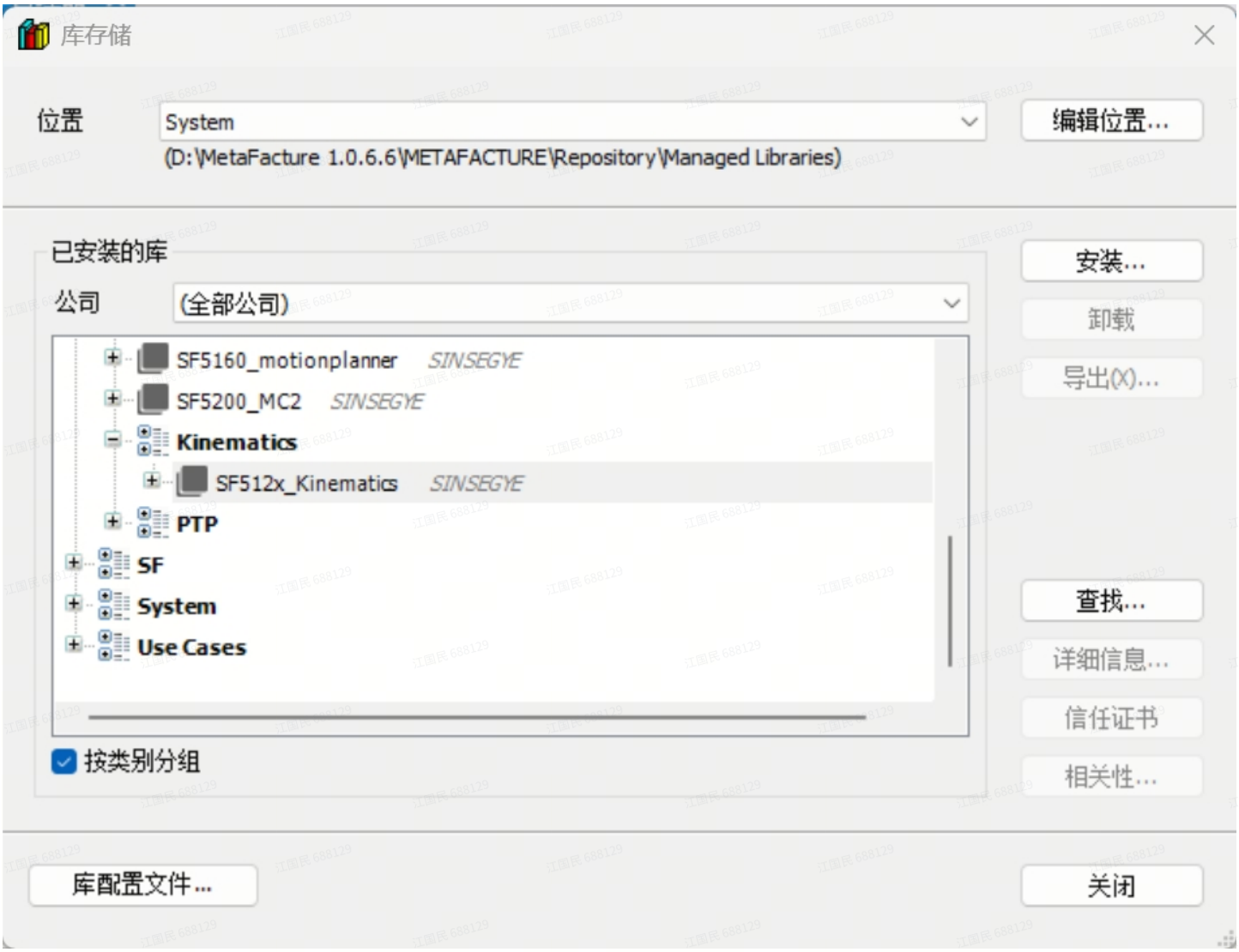
在 MetaFacture 中打开工程项目。

通过菜单选择“工具”->“库存储”。



点击安装，打开 **Kinematics** 库文件。

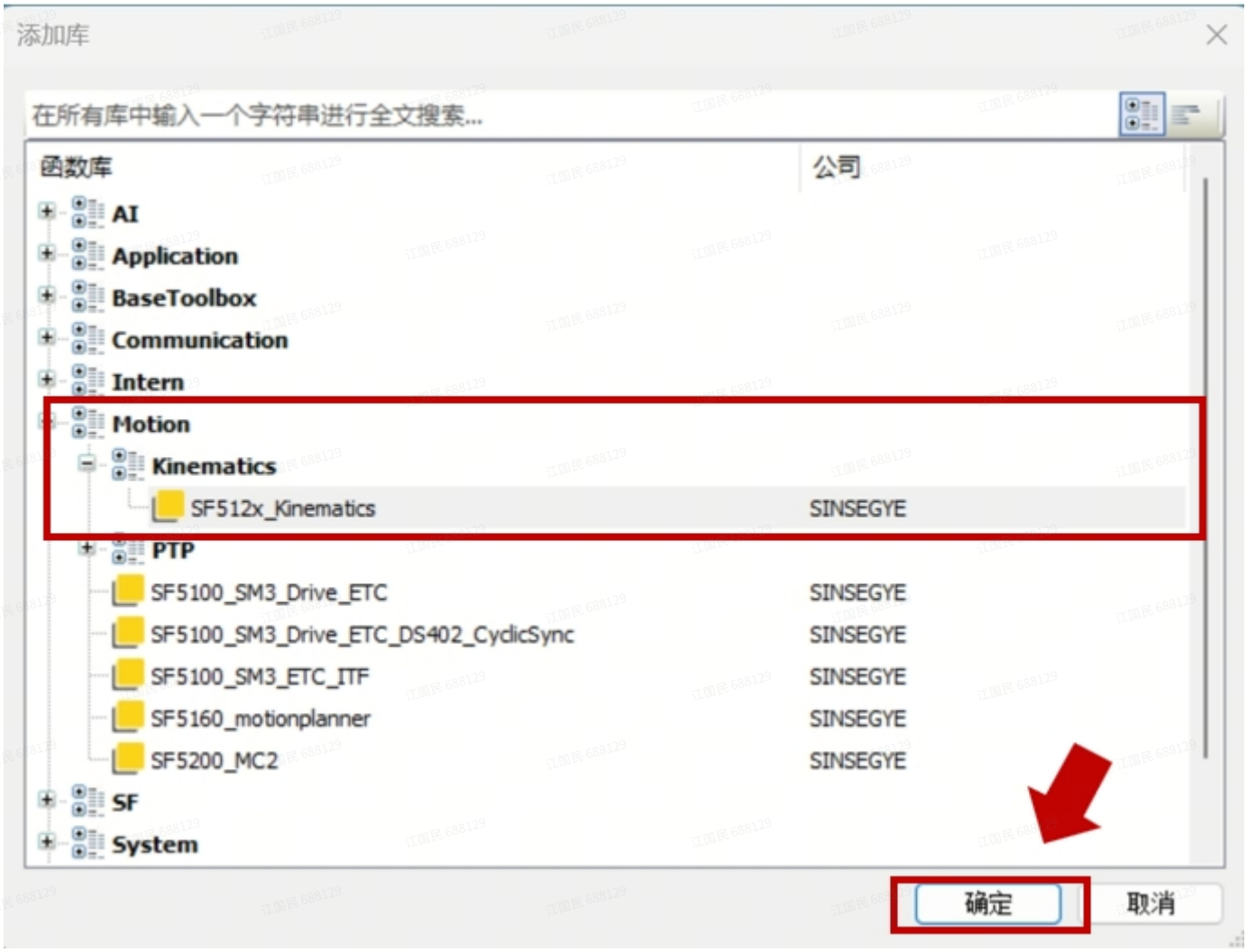




打开成功，关闭窗口。

点击”添加库“

选中 **Kinematics** ，点击确定。



三、更新安装

1、覆盖安装

下载最新版本的 Kinematics 库文件。

打开 MetaFacture 工程，在“库管理器”中删除旧版本的库，导入新版本库文件。

保存工程并重新编译。

2、版本兼容性验证

检查新版本是否与现有工程兼容。

根据文档调整旧版本与新版本可能的差异。

四、卸载过程

1、通过库管理器卸载

打开 MetaFacture 项目，进入“库管理器”。

选择 Kinematics 库，点击“删除库”卸载库文件。

2、**清理项目依赖**

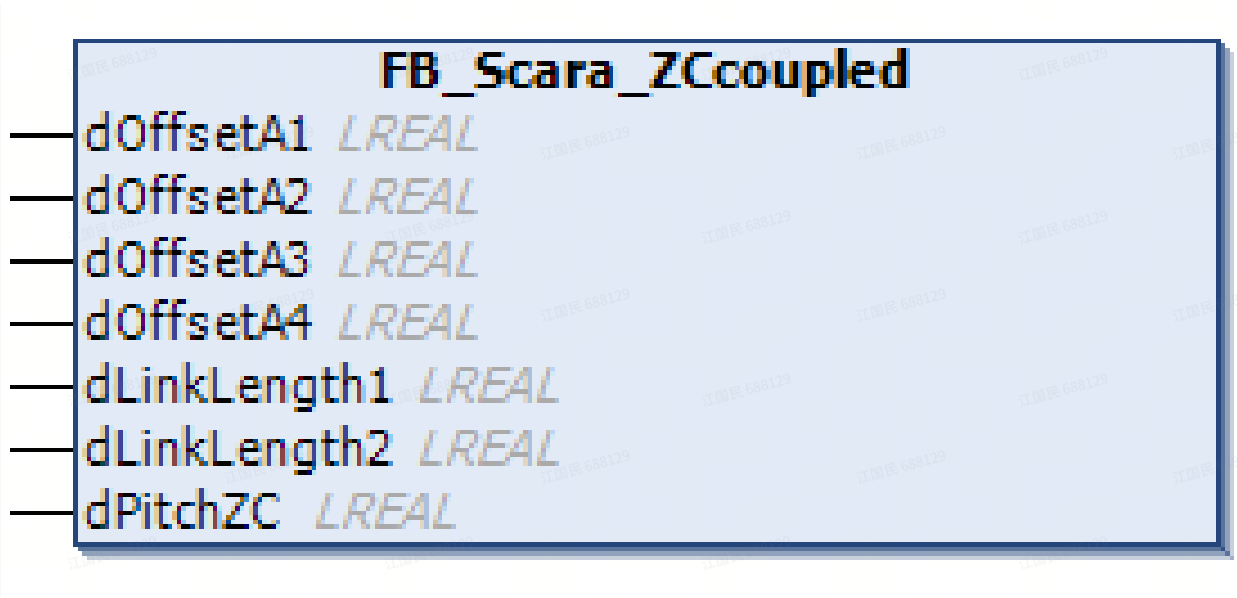
打开工程项目，检查并移除所有使用 Kinematics 库的功能块或接口。

确保工程重新编译时无报错。

技术说明

一、功能块

FB_Scara_ZCcoupled



SCARA机器人运动学变换模块**FB_Scara_ZCcoupled**有多个输入，具体描述如下。

接口

VAR_INPUT

```
VAR_INPUT
  dOffsetA1              : LREAL;
  dOffsetA2              : LREAL;
  dOffsetA3              : LREAL;
  dOffsetA4              : LREAL;
  dLinkLength1           : LREAL;
  dLinkLength2           : LREAL;
  dPitchZC               : LREAL;
END_VAR
```

名称	类型	描述
dOffsetA1	LREAL	A1轴，运动学零位时关节旋转角度
dOffsetA2	LREAL	A2轴，运动学零位时关节旋转角度
dOffsetA3	LREAL	A3轴，运动学零位时关节移动距离

dOffsetA4	LREAL	A4轴，运动学零位时关节旋转角度
dLinkLength1	LREAL	连杆参数1，关节1和关节2轴线的公垂线长度X
dLinkLength2	LREAL	连杆参数2，关节2和关节3轴线的公垂线长度X
dPitchZC	LREAL	关节4正旋转360度，TCP向上移动的高度，若TCP实际向下运动则为负值

二、测试代码

以下示例代码用于 4D-SCARA 机器人的运动学变换测试。运行代码后，用户可以通过网页访问机器人测试的可视化界面，在虚拟环境中对轴运动进行测试和验证。

```
FUNCTION_BLOCK test_scara_with_axis
VAR_INPUT
        robPower : BOOL;
        robEnable : BOOL;
        robReset : BOOL;
        robToolSet : BOOL;
        robWObjSet : BOOL;
END_VAR
VAR_OUTPUT
        robState : DINT;
        bPowerDone : BOOL;
END_VAR
VAR_IN_OUT
        ToolOffset : MC_COORD_REF;
        WobjOffset : MC_COORD_REF;
END_VAR
VAR

        bRegulatorOn :BOOL :=FALSE;
        bDriveStart  :BOOL :=FALSE;
        jogAxis : SMC_GroupJog2;
        jogCartesian : SMC_GroupJog2;
        moveAbsJ : MC_MoveDirectAbsolute;
        moveJ : MC_MoveDirectAbsolute;
        moveL : MC_MoveLinearAbsolute;
        gStop : MC_GroupStop;
        read_Axis_position : MC_GroupReadActualPosition;
        read_Pose_position : MC_GroupReadActualPosition;
        set_Tool : SMC_GroupSetTool;
        gmcs : MC_SetCoordinateTransform ;
        set_PathTolerance : SMC_GroupSetPathTolerance;

```

```
33
34
35          gPower : SMC_GroupPower ;
36          gEnable : MC_GroupEnable ;
37          gReadStatus : MC_GroupReadStatus ;
38          gReadError : MC_GroupReadError ;
39          gReset : MC_GroupReset ;
40          gSetTool : SMC_GroupSetTool ;
41          gSetWobj : MC_SetCoordinateTransform;
42          state : DINT;
43
44          statusText : STRING;
45          jogJointEnable: BOOL := FALSE;
46          jogCartEnable: BOOL := FALSE;
47          jogSpeedFactor: LREAL := 0.5;
48
49          jogForward : ARRAY[0..5] OF BOOL ;
50          jogBackward : ARRAY[0..5] OF BOOL ;
51          bJogAxisActive : BOOL;
52          bJogAxisBusy : BOOL;
53          bJogAxisError : BOOL;
54          bJogAxisErrorID : SMC_ERROR;
55          jogAxisPosition: SMC_POS_REF;
56          fbGroupSetPathTolerance : SMC_GroupSetPathTolerance;
57
58          iCoordSystem : UDINT := 0;
59          select_coord_system : SMC_COORD_SYSTEM;
60          jogCartesianForward : ARRAY[0..5] OF BOOL ;
61          jogCartesianBackward : ARRAY[0..5] OF BOOL ;
62          bJogCartesianActive : BOOL;
63          bJogCartesianBusy : BOOL;
64          bJogCartesianError : BOOL;
65          bJogCartesianErrorID : SMC_ERROR;
66          jogCartesianPosition: SMC_POS_REF;
67
68          moveAbsJ_Enable : BOOL := FALSE;
69          moveAbsJ_Target : SMC_POS_REF;
70          moveAbsJ_BufferMode: SM3_Robotics.MC_BUFFER_MODE;
71          moveAbsJ_TransitionMode: SM3_Robotics.MC_TRANSITION_MODE;
72          moveAbsJ_TransitioParameter: ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS -
    1)] OF LREAL;
73          moveAbsJ_Done: BOOL;
74          moveAbsJ_Busy: BOOL;
75          moveAbsJ_Active: BOOL;
76          moveAbsJ_CommandAborted: BOOL;
77          moveAbsJ_CommandAccepted: BOOL;
78          moveAbsJ_Error: BOOL;
```

```
        moveAbsJ_ErrorID: SM3_Robotics.SMC_ERROR;
        moveAbsJ_MovementId: SM3_Robotics.SMC_Movement_Id;


        moveJ_Enable: BOOL := FALSE;
        moveJ_Target: SM3_Robotics.SMC_POS_REF;
        moveJ_Done: BOOL;
        moveJ_Busy: BOOL;
        moveJ_Active: BOOL;
        moveJ_CommandAborted: BOOL;
        moveJ_CommandAccepted: BOOL;
        moveJ_Error: BOOL;
        moveJ_ErrorID: SM3_Robotics.SMC_ERROR;
        moveJ_MovementId: SM3_Robotics.SMC_Movement_Id;


        moveL_Enable: BOOL := FALSE;
        moveL_Target: SM3_Robotics.SMC_POS_REF;
        moveL_Done: BOOL;
        moveL_Busy: BOOL;
        moveL_Active: BOOL;
        moveL_CommandAborted: BOOL;
        moveL_CommandAccepted: BOOL;
        moveL_Error: BOOL;
        moveL_ErrorID: SM3_Robotics.SMC_ERROR;
        moveL_MovementId: SM3_Robotics.SMC_Movement_Id;


        bStop : BOOL;
        stop_Done: BOOL;
        stop_Busy: BOOL;
        stop_Active: BOOL;
        stop_CommandAborted: BOOL;
        stop_CommandAccepted: BOOL;
        stop_Error: BOOL;
        stop_ErrorID: SM3_Robotics.SMC_ERROR;
        stop_MovementId: SM3_Robotics.SMC_Movement_Id;

        read_Axis_position_enable: BOOL := TRUE;
        read_Axis_position_Valid: BOOL;
        read_Axis_position_Busy: BOOL;
        read_Axis_position_Error: BOOL;
        read_Axis_position_ErrorID: SM3_Robotics.SMC_ERROR;
        read_Axis_position_Position: SM3_Robotics.SMC_POS_REF;
        read_Actual_Axis_position_Position: SM3_Robotics.SMC_POS_REF;
        read_Axis_position_KinematicConfig: SM3_Robotics.TRAFO.CONFIGDATA;

```

```
126         read_Pose_position_enable: BOOL := TRUE;
127         read_Pose_position_Valid: BOOL;
128         read_Pose_position_Busy: BOOL;
129         read_Pose_position_Error: BOOL;
130         read_Pose_position_ErrorID: SM3_Robotics.SMC_ERROR;
131         read_Pose_position_Position: SM3_Robotics.SMC_POS_REF;
132         read_Pose_position_KinematicConfig: SM3_Robotics.TRAFO.CONFIGDATA;
133
134         set_Tool_Excute: BOOL;
135         set_Tool_ToolOffset: SM3_Robotics.MC_COORD_REF;
136         set_Tool_Done: BOOL;
137         set_Tool_Error: BOOL;
138         set_Tool_ErrorID: SM3_Robotics.SMC_ERROR;
139
140         bGMCS : BOOL := FALSE;
141         gmcs_coord: SM3_Robotics.MC_COORD_REF;
142
143         set_PathTolerance_Execute: BOOL := FALSE;
144         PathTolerance_AxisLag: SM3_Robotics.TRAFO.AXISPOS_REF;
145         PathTolerance_MaxPositionLag: LREAL := 10000;
146         PathTolerance_MaxOrientationLag: LREAL := 10000;
147         PathTolerance_Done: BOOL;
148         PathTolerance_Error: BOOL;
149         PathTolerance_ErrorID: SM3_Robotics.SMC_ERROR;
150
151         axispos : TRAFO.AXISPOS_REF;
152
153         robStatus: DINT;
154         bPower : BOOL := FALSE;
155         bEnable : BOOL := FALSE;
156         bReset : BOOL := FALSE;
157         bToolSet : BOOL := FALSE;
158         bWobjSet : BOOL := FALSE;
159
160         ToolOffset1 : MC_COORD_REF;
161         WobjOffset1 : MC_COORD_REF;
162
163         config1_enable :BOOL :=FALSE;
164         config2_enable :BOOL :=FALSE;
165         xElbowRight :BOOL :=FALSE;
166         config_to_set : TRAFO.Kin_Scara3_Z_Config;
167         config_read: TRAFO.Kin_Scara3_Z_ReadConfig;
168         skc: SMC_SetKinConfiguration;
169         skc_execure :BOOL :=FALSE;
170         nP1 :DINT:=1;  nP2 :DINT:=1;  nP3 :DINT:=1;  nP4 :DINT:=1;
171         setConfig : SMC_SetKinConfiguration;
172         setConfig_Execute :BOOL :=FALSE;
```

```
173         Scara3ZConfig : TRAFO.Kin_Scara3_Z_Config;
174         periodConfig :SM3_Robotics.TRAFO.CONFIGDATA ;
175
176 END_VAR
177
178
179 bRegulatorOn := TRUE;  bDriveStart := TRUE;
180 bPower:= robPower;     bEnable:= robEnable;
181 gPower(AxisGroup:= scara,bRegulatorOn:= bRegulatorOn,bDriveStart:=
    bDriveStart,Status=>,Busy=>,Error=>,ErrorID=>) ;
182 gEnable(AxisGroup:= scara,Done=>,Busy=>,Error=>,ErrorID=>) ;
183 CASE state OF
184 0:
185         gPower.Enable := bPower;
186         IF gPower.Status THEN
187                 state := state + 10;
188         ELSE
189                 state := 0;
190                 RETURN;
191         END_IF
192 10:
193         gEnable.Execute := bEnable;
194         IF gEnable.Done THEN
195                 state := state + 10;
196         END_IF
197 20:
198         gReadStatus(AxisGroup:= scara);
199         gReadError(AxisGroup:= scara);
200         gSetTool(AxisGroup:= scara);
201         gSetWobj(AxisGroup:= scara);
202         gReset(AxisGroup := scara);
203         IF gPower.Status AND gEnable.Done THEN
204                 gReadStatus.Enable :=TRUE;
205                 gReadError.Enable := TRUE;
206                 state := state + 10;
207         END_IF
208 END_CASE
209
210 IF   robReset   THEN
211         gReset(AxisGroup := scara, Execute:=robReset,
    Done=>,Busy=>,Error=>,ErrorID=>);
212 END_IF
213 IF state = 30 THEN
214         bPowerDone := TRUE;
215 ELSE
216         bPowerDone := FALSE;
217 END_IF
```

```
robStatus := state;
robState := robStatus;
IF robState <> 30 THEN
        RETURN;
END_IF
// >>>> robStatus
IF gReadStatus.GroupErrorStop THEN
        statusText :=CONCAT('Error :
',SMC_ErrorString(gReadError.GroupErrorID, SMC_LANGUAGE_TYPE.english));
ELSE
        statusText := 'No Error';
END_IF

jogAxis(
        AxisGroup:= scara,
        Enable:= jogJointEnable, Forward:= jogForward, Backward:=
jogBackward,
        MaxLinearDistance:= 100, MaxAngularDistance:= 100,
        Velocity:= 1, Acceleration:= 10, Deceleration:= 10, Jerk:= 100,
        VelFactor:= jogSpeedFactor, AccFactor:= 1, JerkFactor:= 1,
TorqueFactor:= 1,
        CoordSystem:= SMC_COORD_SYSTEM.ACS, ABC_as_ACS:= FALSE,
        Active=> bJogAxisActive, Busy=> bJogAxisBusy,
    Error=> bJogAxisError, ErrorID=> bJogAxisErrorID, CurrentPosition=>
jogAxisPosition);

jogCartesian(
        AxisGroup                       := scara,
        Enable                          := jogCartEnable, Forward :=
jogCartesianForward, Backward := jogCartesianBackward,
        MaxLinearDistance       := 20, MaxAngularDistance       := 20,
        Velocity                        := 1, Acceleration      := 10,
Deceleration       := 10, Jerk       := 100,
        VelFactor                       := jogSpeedFactor, AccFactor:= 1,
JerkFactor:= 1, TorqueFactor:= 1,
        CoordSystem                     := SMC_COORD_SYSTEM.MCS,
ABC_as_ACS                      := FALSE,
        Active                          => bJogCartesianActive,   Busy
=> bJogCartesianBusy,
    Error => bJogCartesianError, ErrorID => bJogCartesianErrorID,
CurrentPosition=> jogCartesianPosition);

IF TRUE THEN
        read_Axis_position(
                AxisGroup:= scara, Enable:= read_Axis_position_enable,
                CoordSystem:= SMC_COORD_SYSTEM.ACS,
```

```
254                 Valid=> read_Axis_position_Valid, Busy=>
    read_Axis_position_Busy,
255         Error=> read_Axis_position_Error, ErrorID=>
    read_Axis_position_ErrorID,
256                 Position=> read_Axis_position_Position, KinematicConfig=>
    read_Axis_position_KinematicConfig);
257 END_IF
258
259
260 select_coord_system := SMC_COORD_SYSTEM.MCS ;
261 IF TRUE THEN
262         read_Pose_position(
263                 AxisGroup:= scara,
264                 Enable:= read_Pose_position_enable, CoordSystem:=
    select_coord_system,
265                 Valid=> read_Pose_position_Valid, Busy=>
    read_Pose_position_Busy,
266         Error=> read_Pose_position_Error, ErrorID=>
    read_Pose_position_ErrorID,
267                 Position=> read_Pose_position_Position, KinematicConfig=>
    read_Pose_position_KinematicConfig);
268
269         config_to_set(xElbowRight := xElbowRight, nPeriodA1:=nP1,
    nPeriodA2:=nP2, nPeriodA3:=nP3, Config=>periodConfig);
270         IF  config1_enable THEN
271                 skc(AxisGroup := scara, Execute := skc_execure, ConfigData :=
    periodConfig, Done =>, Busy =>, Error =>, ErrorId=>);
272         END_IF
273         IF config2_enable AND read_Pose_position.Valid THEN
274                 setConfig(AxisGroup:= scara, Execute:= setConfig_Execute,
    ConfigData:= read_Pose_position_KinematicConfig,
275                Done=>, Busy=>, Error=>, ErrorID=>);
276                 IF setConfig.Done THEN
277                         setConfig_Execute := FALSE ;
278                 ELSIF setConfig.Error THEN
279                         setConfig_Execute := FALSE ;
280                 END_IF
281     END_IF
282 END_IF
283
284 CASE iCoordSystem OF
285 0:
286     select_coord_system := SMC_COORD_SYSTEM.MCS ;
287 1:
288     select_coord_system := SMC_COORD_SYSTEM.PCS_1;
289 ELSE
290     select_coord_system := SMC_COORD_SYSTEM.MCS ;
```

```
END_CASE

axispos.a0 := 100000000; axispos.a1 := 100000000; axispos.a2 := 100000000;
axispos.a3 := 100000000; axispos.a4 := 100000000; axispos.a5 := 100000000;
fbGroupSetPathTolerance(
        AxisGroup := scara,
        Execute  := set_PathTolerance_Execute,
        MaxPositionLag := 1000000,
        MaxOrientationLag := 360000,
        MaxAxisLag := axispos
);

moveAbsJ(
        AxisGroup:= scara,
        Execute:= moveAbsJ_Enable,
        Position:= moveAbsJ_Target, MovementType:= ,
        CoordSystem:= SMC_COORD_SYSTEM.ACS,
        BufferMode:= SM3_Robotics.MC_BUFFER_MODE.Aborting,
        TransitionMode:=  SM3_Robotics.MC_TRANSITION_MODE.TMNone,
        TransitionParameter:= moveAbsJ_TransitioParameter,
        VelFactor:= 1, AccFactor:= 1, JerkFactor:= 1, TorqueFactor:= 1,
        Done=> moveAbsJ_Done, Busy=> moveAbsJ_Busy, Active=> moveAbsJ_Active,
        CommandAborted=> moveAbsJ_CommandAborted, CommandAccepted=>
moveAbsJ_CommandAccepted,
        Error=> moveAbsJ_Error, ErrorID=> moveAbsJ_ErrorID,
        MovementId=>moveAbsJ_MovementId);
moveJ(
        AxisGroup:= scara,
        Execute:= moveJ_Enable, Position:= moveJ_Target, MovementType:= ,
        CoordSystem:= SMC_COORD_SYSTEM.PCS_1,
        BufferMode:= SM3_Robotics.MC_BUFFER_MODE.Aborting,
        TransitionMode:= SM3_Robotics.MC_TRANSITION_MODE.TMNone,
        TransitionParameter:= moveAbsJ_TransitioParameter,
        VelFactor:= 1, AccFactor:= 1, JerkFactor:= 1, TorqueFactor:= 1,
        Done=> moveJ_Done, Busy=> moveJ_Busy, Active=> moveJ_Active,
        CommandAborted=> moveJ_CommandAborted, CommandAccepted=>
moveJ_CommandAccepted,
        Error=> moveJ_Error, ErrorID=> moveJ_ErrorID, MovementId=>
moveJ_MovementId);
moveL(
        AxisGroup:= scara, Execute:= moveL_Enable, Position:= moveL_Target,
        Velocity:= 100, Acceleration:= 1000, Deceleration:= 1000, Jerk:=
1000,
        CoordSystem:= SMC_COORD_SYSTEM.MCS,
        BufferMode:= SM3_Robotics.MC_BUFFER_MODE.Aborting,
        TransitionMode:= SM3_Robotics.MC_TRANSITION_MODE.TMNone,
        TransitionParameter:= moveAbsJ_TransitioParameter,
```

```
334         OrientationMode:= SMC_ORIENTATION_MODE.GreatCircle,
335         VelFactor:= 1, AccFactor:= 1, JerkFactor:= 1, TorqueFactor:= 1,
336         Done=> moveL_Done, Busy=> moveL_Busy, Active=> moveL_Active,
337         CommandAborted=> moveL_CommandAborted, CommandAccepted=>
    moveL_CommandAccepted,
338         Error=> moveL_Error, ErrorID=> moveL_ErrorID, MovementId=>
    moveL_MovementId);
339 gStop(
340         AxisGroup:= scara,
341         Execute:= bStop,
342         Deceleration:= 1000, Jerk:= 1000, AccFactor:= 1, JerkFactor:= 1,
    TorqueFactor:= 1,
343         ClearMovements:=TRUE,
344         Done=> stop_Done, Busy=> stop_Busy, Active=> stop_Active,
345         CommandAborted=> stop_CommandAborted, CommandAccepted=>
    stop_CommandAccepted,
346         Error=> stop_Error, ErrorID=> stop_ErrorID, MovementId=>
    stop_MovementId);
```