OMRON

オートメーションソフトウェア

Sysmac Studio

3D シミュレーション機能
オペレーションマニュアル

形 SYSMAC-SE2□□□
形 SYSMAC-SA4□□L-64



SYSMAC
always in control

SBCA-482L

おことわり

(1) 本マニュアルの内容の一部または全部を無断で複写、複製、転載することを禁じます。

(2) 本マニュアルの内容に関しては、改良のため予告なしに仕様などを変更することがあります。
あらかじめご了承ください。

(3) 本マニュアルの内容に関しては、万全を期しておりますが、万一ご不審な点や誤りなど、お気づきのことがありましたら、お手数ですが、弊社支店または営業所までご連絡ください。
その際、巻末記載の Man.No. （マニュアル No.）も併せてお知らせください。

商標

- Sysmac は、オムロン株式会社 FA 機器製品の日本およびその他の国における商標または登録商標です。
- Microsoft、Windows、Excel、Visual Basic、Microsoft Edge は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
- EtherCAT®は、ドイツ Beckhoff Automation GmbH によりライセンスされた特許取得済み技術であり登録商標です。
- ODVA、CIP、CompoNet、DeviceNet、EtherNet/IP は ODVA の商標です。
- SD、SDHC ロゴは、SD-3C,LLC の商標です。
- NVIDIA、NVIDIA ロゴ、GeForce、GeForce ロゴは、米国およびその他の国における NVIDIA Corporation の登録商標または商標です。
- AMD Radeon™は、Advanced Micro Devices, Inc. の商標です。
- Celeron、Intel、Intel Core は、米国およびその他の国における Intel Corporation の商標です。
- Git and the Git logo are either registered trademarks or trademarks of Software Freedom Conservancy, Inc., corporate home of the Git Project, in the United States and/or other countries.
- JavaScript®は、オラクルおよびその関連会社の登録商標です。
- EPLAN および EPLAN Electric P8 は EPLAN GmbH & Co.KG 社の登録商標です。

その他、本文中に掲載している会社名および製品名は、それぞれ各社の商標または登録商標です。

著作権

スクリーンショットはマイクロソフトの許諾を得て使用しています。

はじめに

このたびは、Sysmac Studio 3D シミュレーションオプションをお買い上げいただきまして、ありがとうございます。
本マニュアルは、Sysmac Studio 3D シミュレーションオプションで 3D シミュレーション機能を使用するうえで、必要な情報を記載しています。お使いになる前に本マニュアルをよく読んで、機能、性能などを十分ご理解いただき、システムの構築にご活用ください。
また、お読みになったあとも本マニュアルは大切に保管して、いつも手元においてお使いください。

対象となる読者の方々

本マニュアルは、次の方を対象に記述しています。電気の知識（電気工事士あるいは同等の知識）を有する方で

- FA 機器の導入を担当される方
- FA システムを設計される方
- FA 機器を設置、接続される方
- FA 現場を管理される方

また、プログラミング言語に関して、国際標準規格 IEC 61131-3 もしくは国内標準規格 JIS B 3503 に規定される内容をご理解いただいている方を対象に記述しています。

対象となる製品

本マニュアルは以下の製品を対象にしています。

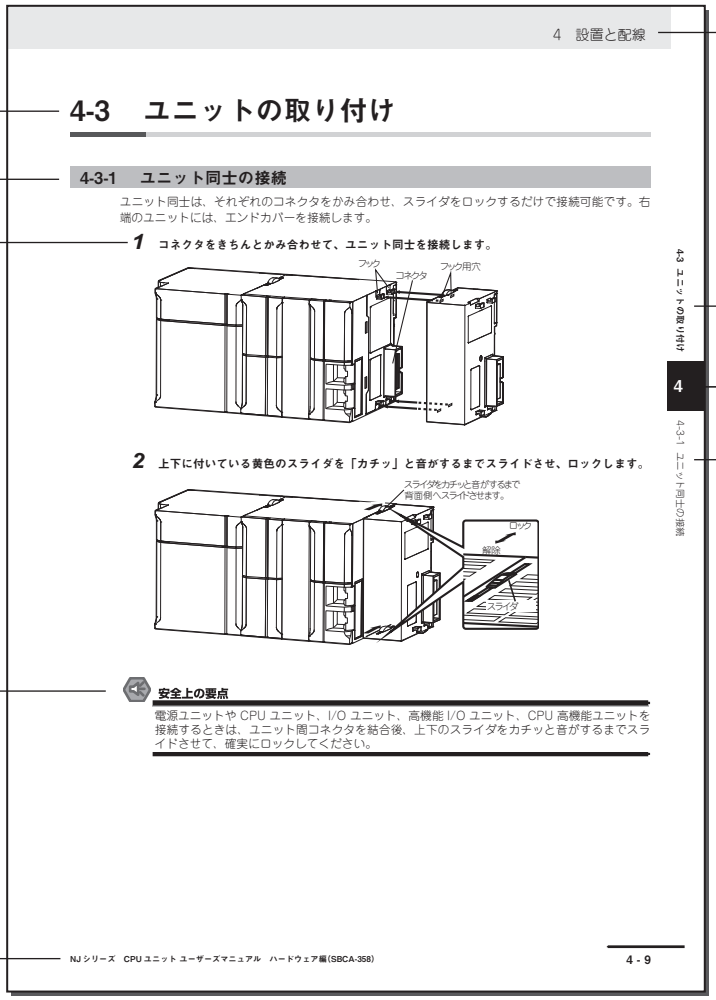
- Sysmac Studio 3D シミュレーションオプション

各製品の仕様の一部または制約事項が、他のマニュアルに記載されている場合があります。
「関連マニュアル (P.17)」を確認してください。

マニュアルの読み方

ページ構成

このマニュアルの各ページの構成は次のとおりです。



節タイトル

項タイトル

操作手順

操作手順を示します。

注記、補足、参照先

注記、補足、参照先などの各項目を、アイコンによって示しています。

マニュアル名称

章タイトル

節タイトル

項タイトル

現在のページの章／節／項タイトルをそれぞれ示します。

章番号

現在のページの章番号を示します。

このページは、説明用の見本です。実際の内容とは異なります。

アイコン

このマニュアルで使われているアイコンには、次のような意味があります。

安全上の要点

製品を安全に使用するために実施または回避すべきことを示します。

使用上の注意

製品の動作不能、誤動作、または性能・機能への悪影響を予防するために実施または回避すべきことを示します。



参考

必要に応じて読んでいただきたい項目です。
知っておくと便利な情報や、使用する上で参考となる内容について説明しています。



バージョン関連情報

コントローラやユニット、サポートソフトの、バージョンによる性能や機能の違いについて説明しています。

表記についての留意事項

本マニュアルでは、Sysmac Studio から実機コントローラにデータを転送することを「ダウンロード」、実機コントローラからデータを Sysmac Studio に転送することを「アップロード」と記述しています。
Sysmac Studio では、「ダウンロード」および「アップロード」は、いずれも「同期」機能を使用します。Sysmac Studio の「同期」機能は、パソコン上のデータと実機コントローラのデータを自動照合し、どちらの方向に転送するかをユーザが選択する機能です。

目次構成

1 3Dシミュレーション機能の特長と仕様

2 ソフトウェアのセットアップ

3 3Dシミュレーション実行の流れ

4 3D形状データの作成

5 3D表示画面、3D編集エリアの操作

6 3D形状データを動作させる設定およびスクリプトの作成

7 ワークシミュレーション簡単設定

8 3Dシミュレーションの実行

9 ロボット統合システムの3Dシミュレーション

10 便利機能

A 付録

I 索引

目次

はじめに **1**
対象となる読者の方々 1
対象となる製品 1

マニュアルの読み方 **2**
ページ構成 2
アイコン 2
表記についての留意事項 3

目次構成 **5**

ご承諾事項 **11**

安全上の注意 **12**

安全上の要点 **13**

使用上の注意 **14**

法規と規格 **15**
海外でのご使用 15
ソフトウェアのライセンスと著作権 15

バージョン **16**

関連マニュアル **17**

用語説明 **18**

マニュアル改訂履歴 **20**

第 1 章　3D シミュレーション機能の特長と仕様

1-1　3D シミュレーションの特長 **1-2**

1-2　仕様 **1-3**
1-2-1　商品形式 1-3
1-2-2　対応言語 1-3
1-2-3　使用可能な機種 1-3
1-2-4　使用可能なパソコン 1-4

第 2 章　ソフトウェアのセットアップ

2-1　Sysmac Studio のインストール **2-2**

2-2　Sysmac Studio オプションライセンスの登録 **2-3**

第 3 章　3D シミュレーション実行の流れ

3-1　3D シミュレーションの実施概要 **3-2**

3-2　3D シミュレーションの実施工程 3-4
3-2-1　シミュレーション設定方法の選択 3-5
3-2-2　スクリプトを使用する場合の 3D シミュレーションの設定 3-6
3-2-3　ワークシミュレーション簡単設定を使用する場合の 3D シミュレーションの設定 3-7

第 4 章　3D 形状データの作成

4-1　3D 形状データの概要 4-3
4-1-1　3D 形状データのロケーション 4-3
4-1-2　3D 形状データの親子関係 4-6

4-2　アプリケーションマネージャの追加 4-7

4-3　メカニカル機構の 3D 形状データの作成 4-9
4-3-1　サポートする CAD データファイルの種類 4-9
4-3-2　CAD データファイルの準備 4-9
4-3-3　CAD データのインポート手順 4-11
4-3-4　メカニカル機構モデルの種類 4-17
4-3-5　メカニカル機構の設定項目 4-25

4-4　カスタムメカニカル機構の 3D 形状データの作成 4-30
4-4-1　カスタムメカニカル機構で実現できる機構 4-30
4-4-2　カスタムメカニカル機構の 3D 形状データの作成の流れ 4-31
4-4-3　サポートする OpenUSD ファイルについて 4-32
4-4-4　カスタムメカニカル機構の追加手順 4-33
4-4-5　可動部品の追加／CAD データのインポート 4-36
4-4-6　USDPhysicsJoint を含む OpenUSD ファイルのインポート 4-40
4-4-7　TCP の設定 4-45
4-4-8　結合部の動作設定 4-47
4-4-9　可動部品の動作方向設定 4-55
4-4-10　カスタムメカニカル機構の動作確認 4-60
4-4-11　カスタムメカニカル機構の動作設定 4-62
4-4-12　CAD データの更新 4-69
4-4-13　OpenUSD ファイルのエクスポート 4-70
4-4-14　カスタムメカニカル機構の作成例 4-71

4-5　パラレルリンク機構の 3D 形状データの作成 4-78
4-5-1　パラレルリンク機構の追加手順 4-78
4-5-2　パラレルリンク機構の設定項目 4-79
4-5-3　CAD データのインポート 4-86

4-6　ワークの 3D 形状データの追加 4-89
4-6-1　CAD データのインポート 4-89
4-6-2　直方体、円柱の追加 4-93

4-7　ワーク検出センサの追加 4-98
4-7-1　仮想ワーク検出センサの追加 4-98
4-7-2　仮想ワーク検出センサの設定項目 4-98

4-8　3D 形状データの配置方法 4-101
4-8-1　座標の直接入力 4-101
4-8-2　3D 表示画面上でのドラッグ&ドロップ 4-101
4-8-3　リンクポイントへのスナップ 4-101

4-9　3D 表示オブジェクトのエクスポート／インポート 4-102
4-9-1　エクスポート／インポート可能な 3D 表示オブジェクトの種類 4-102
4-9-2　エクスポート／インポート操作可能なアクセスレベル 4-102
4-9-3　3D 表示オブジェクトのエクスポート 4-102
4-9-4　3D 表示オブジェクトの一括エクスポート 4-103
4-9-5　3D 表示オブジェクトのインポート手順 4-105

第 5 章　3D 表示画面、3D 編集エリアの操作

5-1　3D 表示画面、3D 編集エリアの表示 5-2
5-1-1　3D 表示画面の表示 5-2

5-1-2 3D 編集エリアの表示 5-3

5-2 3D 表示画面、3D 編集エリアの項目 5-5
5-2-1 画面分割 5-6
5-2-2 選択と編集 5-6
5-2-3 平行移動 5-7
5-2-4 回転 5-8
5-2-5 ズーム 5-8
5-2-6 投影切り替え 5-9
5-2-7 シーングラフ 5-10
5-2-8 測定ルーラ 5-12
5-2-9 スナップ 5-13
5-2-10 記録 5-14
5-2-11 3D ビュー切替ツール 5-17

5-3 3D 表示画面の 3D 形状データ操作 5-19
5-3-1 3D 形状データの移動 5-19
5-3-2 3D 形状データの回転 5-20
5-3-3 3D 形状データの簡易編集 5-21

5-4 マウントポイント／リンクポイントを使用した位置設定 5-24
5-4-1 マウントポイントとリンクポイントの概要 5-24
5-4-2 マウントポイントの設定 5-26
5-4-3 リンクポイントの設定 5-27
5-4-4 オフセットの設定方法 5-29
5-4-5 マウントポイントとリンクポイントを使用した 3D 形状データの配置 5-33

第 6 章 3D 形状データを動作させる設定およびスクリプトの作成

6-1 3D 形状データを動作させる設定およびスクリプトの概要 6-2
6-1-1 シェイプスクリプトの概要 6-2
6-1-2 シェイプスクリプトとプログラム実行のタイミングと周期 6-3

6-2 メカニカル機構の設定 6-10
6-2-1 リミットスイッチの仮想出力用スクリプト生成 6-10

6-3 ワークの動作スクリプトの作成 6-12
6-3-1 シェイプスクリプトの追加 6-12
6-3-2 シェイプスクリプトの実行設定 6-12

6-4 ワーク検出センサの動作設定 6-16

6-5 シェイプスクリプトエディタ 6-18
6-5-1 シェイプスクリプトエディタ画面 6-18
6-5-2 シェイプスクリプトのプログラミング 6-21

6-6 シェイプスクリプトファンクション 6-24
6-6-1 シェイプスクリプトファンクションの追加 6-24
6-6-2 デフォルトファンクションおよびユーザファンクションの参照 6-25

第 7 章 ワークシミュレーション簡単設定

7-1 ワークシミュレーション簡単設定の概要 7-2

7-2 メカニカル機構の設定 7-3

7-3 ワーク動作設定 7-4
7-3-1 ワーク動作設定の一覧 7-6
7-3-2 生成 7-7
7-3-3 把持 7-12
7-3-4 搬送 7-14
7-3-5 消去 7-18
7-3-6 押出 7-20
7-3-7 センサ 7-22
7-3-8 物理シミュレーション 7-25
7-3-9 ワーク動作設定共通の設定 7-28

第 8 章　3D シミュレーションの実行

8-1　3D シミュレーションの操作手順 8-2
8-1-1　コントローラのシミュレーションを実行する 8-2
8-1-2　仮想ワーク検出センサの動作スクリプト、およびメカニカル機構の仮想出力用スクリプトを実行する 8-3
8-1-3　ワークのシェイプスクリプトを実行する 8-3
8-1-4　ワーク動作設定のシミュレーションを実行する 8-4
8-1-5　ワーク動作設定によるシミュレーションを停止する 8-4
8-1-6　3D 表示画面で動作を確認する 8-4
8-1-7　3D シミュレーションを一括で実行する 8-5

8-2　シェイプスクリプトのデバッグ方法 8-8
8-2-1　ブレークポイント 8-8
8-2-2　ステップ実行 8-9
8-2-3　トレース文 8-10
8-2-4　タクトタイムの計測 8-10
8-2-5　シェイプスクリプト実行時のエラー表示 8-13

8-3　衝突検知機能 8-15
8-3-1　衝突検知対象 8-15
8-3-2　衝突検知設定手順 8-15
8-3-3　衝突検知の確認方法 8-17
8-3-4　シェイプスクリプトで衝突を検知する方法 8-18
8-3-5　衝突検知用モデルの設定 8-24

第 9 章　ロボット統合システムの 3D シミュレーション

9-1　アプリケーションコントローラによるロボットシステムの 3D シミュレーションの概要 9-2
9-1-1　3D シミュレーションの種類 9-2
9-1-2　IEC61131-3 言語で制御するメカニカル機構の 3D シミュレーション 9-2
9-1-3　ロボットアプリケーションの 3D シミュレーション 9-2

9-2　ロボットシステムの 3D シミュレーション 9-4
9-2-1　ロボットで操作したワークを周辺機器で操作する 9-4
9-2-2　周辺機器で操作したワークをロボットで操作する 9-5
9-2-3　ロボットだけで操作するワークの衝突検知を行う 9-6

第 10 章　便利機能

10-1　CAD データの一括置換 10-2
10-1-1　一括置換手順 10-2

付録

A-1　シェイプスクリプトで使用する関数 A-2
A-1-1　関数一覧 A-2
A-1-2　ワーク、パレットの生成、表示 A-7
A-1-3　ワーク、パレットの搬送 A-19
A-1-4　コンベア A-21
A-1-5　ワーク、パレットのピック A-23
A-1-6　ワーク、パレットのプレース A-28
A-1-7　状態の遷移、検知 A-34
A-1-8　コントローラとの変数共有 A-38
A-1-9　プロセスマネージャとの連携 A-58
A-1-10　シェイプスクリプト実行時間計測 A-62
A-1-11　シェイプスクリプトで使用する関数の更新 A-63

A-2　シミュレータと実機コントローラの差異 A-66
A-2-1　各機能の動作 A-66

A-3　シェイプスクリプトの実行時間 A-67
A-3-1　システムモニタによる実行時間の確認 A-67
A-3-2　Stopwatch 構文を使用した実行時間の計測 A-68
A-3-3　シェイプスクリプトの処理の高速化 A-69

A-4　ワーク動作設定 A-74
A-4-1　3D 形状データごとのワーク動作設定可否一覧 A-74
A-4-2　シェイプスクリプトとワーク動作設定の同時実行を実現する方法 A-76
A-4-3　ロボットハンドのツールチェンジャーを実現する方法 A-77
A-4-4　パレットを実現する方法 A-78

索引

ご承諾事項

● 保証内容

(1)保証期間

本ソフトウェアの保証期間は、ご購入後 1 年といたします。
（ただし「カタログ等」に別途記載がある場合を除きます。）

(2)保証範囲

a）本ソフトウェアの使用許諾を受けたお客様が、上記保証期間中にコンピュータ・プログラムの瑕疵（マニュアルとの重要な不一致）を発見し、当社に返却した場合は、当社は瑕疵（当該マニュアルとの重要な不一致）のないコンピュータ・プログラムを記録した媒体と交換いたします。もしくは、当社の選択により、当社ホームページより瑕疵（当該マニュアルとの重要な不一致）のないコンピュータ・プログラムをダウンロードしていただく方法により提供いたします。また、当社の責任によるコンピュータ・プログラムの記録媒体の不良を発見し、当社に返却した場合、当社は、無償で、良品の媒体に記録したコンピュータ・プログラムと交換いたします。

b）万一、当社がコンピュータ・プログラムの瑕疵（マニュアルとの重要な不一致）を除去できないと判断した場合は、お客様が本ソフトウェア購入代金として支払った金額をお返しいたします。

● 責任の制限

(1)前条に定める交換または購入代金の返金は、本ソフトウェアの保証責任のすべてを定めるものであり、当社は本ソフトウェアの瑕疵により発生した、お客様の直接的、間接的あるいは波及効果による損害等いかなる損害に対しても一切の責任を負いません。

(2)当社は、本ソフトウェアを当社以外の第三者が変更、改造することにより生じた瑕疵や、本ソフトウェアのインストールその他の使用によって当社商品、インストールされたソフトウェア、またはすべてのコンピュータ機器、コンピュータプログラム、ネットワーク、データベースが DDoS 攻撃（分散型 DoS 攻撃）、コンピュータウイルスその他の技術的な有害プログラム、不正アクセスを受けたことにより生じた損害につきましては、一切責任を負いません。

(3)当社は、本ソフトウェアに基づき、当社以外の第三者が開発したソフトウェアおよびそれにより生じた結果について一切の責任を負いません。

● 本ソフトウェアの用途

本ソフトウェアを本マニュアルに記載の用途以外の用途で使用しないでください。

● 仕様の変更

本ソフトウェアの仕様および付属品は改善またはその他の事由により、必要に応じて、変更される場合があります。

● 適用範囲

以上の内容は、日本国内での取引および使用を前提としております。日本国外での取引および使用に関しては、当社営業担当者までご相談ください。

安全上の注意

安全上の注意の内容については、『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

安全上の要点

安全上の要点の内容については、『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

使用上の注意

使用上の注意の内容については、『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

法規と規格

海外でのご使用

本製品のうち、外国為替および外国貿易管理法に定める輸出許可、承認対象貨物（または技術）に該当するものを輸出（または非居住者に提供）する場合は、同法に基づく輸出許可、承認（または役務取引許可）が必要です。

ソフトウェアのライセンスと著作権

本製品にはサードパーティ製のソフトウェアが組み込まれています。このソフトウェアに関連するライセンスと著作権については、http://www.fa.omron.co.jp/nj_info_j/ をご覧ください。

バージョン

NJ/NX シリーズの各ユニット、NY シリーズ産業用 PC および各 EtherCAT スレーブは、ハードウェアやソフトウェアが、ハードウェアリビジョンやユニットバージョンという改変番号でバージョン管理されています。
バージョンの詳細は、『Sysmac Studio Version 1 オペレーションマニュアル(SBCA-470)』を参照してください。

関連マニュアル

本マニュアルに関連するマニュアルは、下表のとおりです。あわせてご覧ください。

マニュアル名称	Man.No.	形式	用途	内容
Sysmac Studio 3D シミュレーション機能 オペレーションマニュアル	SBCA-482	形 SYSMAC-SE2□ □□ 形 SYSMAC-SA4□ □□-64	Sysmac Studio の 3D シミュレーション機能の概要、使用方法について知りたいとき。	Sysmac Studio の 3D シミュレーション機能の概要、実行手順、操作方法について説明します。
Sysmac Studio Version 1 オペレーション マニュアル	SBCA-470	形 SYSMAC-SE2□ □□	Sysmac Studio の操作方法、機能について知りたいとき。	Sysmac Studio の操作方法について説明します。
Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編	SBCA-464	形 SYSMAC-SE2□ □□ 形 SYSMAC- SE200D-64	NJ シリーズ ロボット統合 CPU ユニットを使用してロボット統合システムを構築するとき。	Sysmac Studio で NJ シリーズ ロボット統合 CPU ユニットを使用するための機能の概要、操作方法について説明します。
Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル IPC Application Controller 編	SBCA-483	形 SYSMAC-SE2□ □□ 形 SYSMAC- SE200D-64	IPC Application Controller を使用してロボット統合システムを構築するとき。	Sysmac Studio で IPC Application Controller を使用するための機能の概要、操作方法について説明します。
Sysmac Studio ドライブ機能 オペレーションマニュアル	SBCE-401	形 SYSMAC-SE2□ □□ 形 SYSMAC-DE□□ □L	Sysmac Studio のサーボドライバに関する機能について知りたい時	Sysmac Studio の操作方法、機能のうち、サーボドライバに関する機能の操作方法について説明します。
NJ/NX シリーズ CPU ユニット ユーザーズマニュアル ソフトウェア編	SBCA-467	形 NX701-□□□□ 形 NX502-□□□□ 形 NX102-□□□□ 形 NX1P2-□□□□ 形 NJ501-□□□□ 形 NJ301-□□□□ 形 NJ101-□□□□	NJ/NX シリーズ CPU ユニットのプログラミング／システムの立ち上げについて知りたいとき。 おもにソフトウェアに関する情報。	NJ/NX シリーズ CPU ユニットに関して、以下の内容を説明します。 • CPU ユニットの動作 • CPU ユニットの機能 • 初期設定 • IEC 61131-3 ベースの言語仕様とプログラミング
NJ シリーズ NJ Robotics CPU ユニット ユーザーズマニュアル	SBCA-421	形 NJ501-4□□□ 形 NJ501-R□□□	NJ シリーズで Robotics 機能を使用するとき。	NJ Robotics 機能について説明します。
NJ シリーズ ロボット統合 CPU ユニット ユーザーズマニュアル	SBCE-436	形 NJ501-R□□□	NJ シリーズ ロボット統合 CPU ユニットを使用するとき。	オムロン製ロボットを制御するための CPU ユニットの設定や動作、プログラミングの考え方について説明します。
プログラマブルターミナル NA シリーズ ユーザーズマニュアル ソフトウェア編	SBSA-546	形 NA5-□W□□□□	プログラマブルターミナル NA シリーズのページや各オブジェクトの機能について知りたいとき。	プログラマブルターミナル NA シリーズのページや各オブジェクトの機能について説明します。

用語説明

本マニュアルで使用する用語について説明します。

用語	説明
仮想装置モデル	3D シミュレーションを行うために Sysmac Studio 上に作成される装置のモデルです。仮想装置モデルは、メカニカル機構、カスタムメカニカル機構、パラレルリンク機構、仮想ワーク検出センサ、ワークの 3D 形状データと、これらの 3D 形状データの動作を定義する設定やスクリプトから構成されます。
3D 形状データ	Sysmac Studio 上に作成される仮想装置モデルの、3D 空間上での形状、サイズ、位置を表すデータです。CAD データ、直方体、円柱から作成します。
3D 表示オブジェクト	3D シミュレーションで使用する仮想装置モデルの 3D 形状データ、メカニカル機構や仮想ワーク検出センサを動作させるための設定、ワークの動作を定義するスクリプトを管理するデータの総称です。 マルチビューエクスプローラの［構成・設定］-［3D 表示］項目下に追加する「直方体」、「仮想ワーク検出センサ」、「シェイプスクリプト」などがあります。
メカニカル機構	XY テーブルなどの、サーボで駆動する機構です。装置を構成する可動部品の動作を設定し、コントローラの軸変数や BOOL 型変数を割り付けて動かすことができます。
カスタムメカニカル機構	任意の機構を組み合わせて作成する機構です。標準のメカニカル機構では表現できない動作を実現する場合に使用します。電動チャック、電動シリンダの動作を実現することができます。
パラレルリンク機構	パラレルリンクロボットを表す機構です。NJ シリーズ NJ Robotics CPU ユニットでサポートする 3 種類のパラレルリンク機構を使用することができます。
ワーク	3D シミュレーションにおいてメカニカル機構やカスタムメカニカル機構、パラレルリンク機構によって移動される対象物です。ワークの表示、移動といったワークの動作は、C#言語でスクリプトを作成することで実現します。
仮想ワーク検出センサ	3D シミュレーションにおいてワークを検出する仮想的なセンサです。コントローラの BOOL 型変数を割り付けて、コントローラへの入力とすることができます。
アプリケーションマネージャ	3D シミュレーション機能を使用するために必要なデータや設定、およびロボットシステムを制御する IPC Application Controller の設定やプログラムを管理するデバイスです。3D シミュレーションでは、仮想装置モデルの 3D 形状データ、メカニカル機構や仮想ワーク検出センサを動作させるための設定、ワークの動作を定義するスクリプトを管理します。 ロボットシステムの制御に使用するアプリケーションマネージャの機能については、『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル IPC Application Controller 編(SBCA-483)』を参照してください。
シェイプスクリプト	ワークや可動部品などの 3D 形状データの動作を定義するプログラムです。C#言語で記述します。
シェイプスクリプトシーケンス	シェイプスクリプトの実行順序を定義する設定です。シェイプスクリプトの実行もここから行います。
CAD データ	3D 形状データの元となる、装置やワークの 3DCAD データです。サードパーティ製の 3DCAD ソフトウェアで作成します。読み込み可能な CAD データファイルの拡張子は、.stp、.step、.igs、.iges、.usd、.usda、.usdc、.usdz です。
直方体	3D 形状データの元となる、直方体形状の部品です。Sysmac Studio に標準で用意されており、高さ、幅、奥行き、色を設定できます。
円柱	3D 形状データの元となる、円柱形状の部品です。Sysmac Studio に標準で用意されており、半径、高さ、色を設定できます。
ロケーション	3D 形状データの位置と姿勢を表す情報です。位置は右手座標系の座標成分（X、Y、Z）、姿勢はローカル座標系の原点を中心とする軸回りの角度（ヨー、ピッチ、ロール）で表されます。 位置と姿勢を表す値（X、Y、Z、ヨー、ピッチ、ロール）を、ロケーションの要素と呼びます。ヨー、ピッチ、ロールの詳細な定義は「3D 形状データの姿勢 (P.4-4)」を参照してください。
ワールド座標系	3D 表示画面での絶対座標を表す座標系です。ワールド座標系における 3D 形状データの位置は、X、Y、Z で表されます。

用語	説明
ローカル座標系	個々の 3D 形状データが持つ座標系です。3D 形状データの回転の中心を決めるオフセットや、2 つの 3D 形状データ間の相対位置を設定するときに使用します。
親／子	結合された 2 つの 3D 形状データの間の関係を示す用語です。3D 形状データが動作するとき動作の主体となる 3D 形状データを「親」、親の動作に追従する 3D 形状データを「子」と呼びます。子となる 3D 形状データの設定画面で、親となる 3D 形状データを選択することで、2 つの 3D 形状データの親子関係が設定されます。
マウントポイント／リンクポイント	3D 表示画面で、親子関係のある 2 つの 3D 形状データを接合するときに使用する点です。子の 3D 形状データにマウントポイントを設定し、親の 3D 形状データにリンクポイントを設定します。子の 3D 形状データのマウントポイントを親の 3D 形状データのリンクポイントに接合することで、2 つの 3D 形状データ間の位置合わせを簡単に行うことができます。
TCP(Tool Center Point)	メカニカル機構やカスタムメカニカル機構、パラレルリンク機構の 3D 形状データ上にロボットハンドなどのツールを取り付ける位置を示す点です。
OpenUSD(Universal Scene Description)	異なるソフトウェアアプリケーション間で 3D シーンやデータを転送するために Pixar Animation Studio が開発したオープンソースのファイルフォーマットです。
USDPhysicsJoint	OpenUSD に定義された物理スキーマの物理結合情報です。物理シミュレーションにおける 2 つ以上の剛体間の関係を表現するために使用され、3D 環境における複雑な相互接続された構造や動きの作成を可能にします。
Prim	OpenUSD の基本的な構成要素です。シーン内のオブジェクトやエンティティを表現し、ジオメトリ、ライト、カメラ、グループなど、さまざまな種類があります。
Prim Path	シーン グラフ階層内の Prim の位置を特定する文字列です。パスの各要素は階層内の入れ子のレベルを表します。ルート（"/"）で始まり、スラッシュで区切られた親子 Prim の名前が続きます。

マニュアル改訂履歴

マニュアル改訂記号は、表表紙・裏表紙に記載されている Man.No. の後尾に付記されます。



改訂記号	改訂年月	改訂理由・改訂ページ
A	2020 年 4 月	初版
B	2020 年 8 月	Sysmac Studio Version 1.42 へのバージョンアップに伴う改訂
C	2020 年 10 月	Sysmac Studio Version 1.43 へのバージョンアップに伴う改訂
D	2021 年 1 月	Sysmac Studio Version 1.44 へのバージョンアップに伴う改訂
E	2021 年 4 月	Sysmac Studio Version 1.45 へのバージョンアップに伴う改訂
F	2021 年 10 月	Sysmac Studio Version 1.47 へのバージョンアップに伴う改訂
G	2022 年 4 月	Sysmac Studio Version 1.49 へのバージョンアップに伴う改訂
H	2022 年 7 月	Sysmac Studio Version 1.50 へのバージョンアップに伴う改訂
J	2022 年 10 月	Sysmac Studio Version 1.52 へのバージョンアップに伴う改訂
K	2023 年 4 月	Sysmac Studio Version 1.54 へのバージョンアップに伴う改訂
L	2025 年 4 月	Sysmac Studio Version 1.62 へのバージョンアップに伴う改訂

1

3D シミュレーション機能の特長と仕様

本章では、Sysmac Studio の「3D シミュレーション」機能の概要と特長について説明します。

1-1　3D シミュレーションの特長 1-2

1-2　仕様 1-3

1-2-1　商品形式 1-3
1-2-2　対応言語 1-3
1-2-3　使用可能な機種 1-3
1-2-4　使用可能なパソコン 1-4

1-1　3D シミュレーションの特長

Sysmac Studio の 3D シミュレーション機能は、NJ/NX/NY シリーズコントローラで制御する装置の動作と、加工や組立の対象となるワークの搬送動作を、パソコン上で視覚的に確認できる機能です。ベルトコンベアや XY テーブルによるワークの搬送、ガントリクレーンなどの動作を、実際の機器を組み立てる前にパソコン上で確認できます。
3D シミュレーション機能には、次の特長があります。

ROI（費用対効果）の確認時間の短縮

装置全体の動作を視覚的に確認でき、実機を揃えなくても短期間で装置への要求の実現性を検証することができます。
これにより、装置導入の投資判断を早期に行うことができます。

設計、立ち上げ時間の短縮

装置全体の動作やワークの移動を同じ画面で確認しながらプログラムの作成やデバッグを行うことができ、設計時間を短縮できます。
また、装置の CAD データを使用して装置部品の干渉の有無を確認することができます。
これにより、装置の部品やワークの配置を事前に調整することで、現場での装置立ち上げの時間を短縮できます。

品種切り替え検証時間の短縮

装置で取り扱うワークの品種切り替えが必要な場合、稼働中の装置を止めることなく切り替えに必要なプログラムや設定の変更と検証を行うことができます。
装置の実機を用いた検証や調整を短時間で行うことができ、装置の停止時間を短縮できます。

1-2 仕様

1-2-1 商品形式

Sysmac Studio の 3D シミュレーション機能は、Sysmac Studio 64 ビット版 Ver.1.40 以降でサポートしています。
Sysmac Studio の 3D シミュレーション機能を使用するためには、次の Sysmac Studio ライセンスが必要です。また、メカニカル機構の 3D シミュレーションを行うためには、次の Sysmac Studio オプション ライセンスが必要です。
Sysmac Studio 64 ビット版をインストールするには、次の DVD メディアが必要です。

● Sysmac Studio ライセンス

品名	ライセンス数	形式
Sysmac Studio スタンダードエディション Ver.1.□□	1 ライセンス版	形 SYSMAC-SE201L
	3 ライセンス版	形 SYSMAC-SE203L
	10 ライセンス版	形 SYSMAC-SE210L
	30 ライセンス版	形 SYSMAC-SE230L
	50 ライセンス版	形 SYSMAC-SE250L

● Sysmac Studio オプションライセンス

品名	ライセンス数	形式
Sysmac Studio 3D シミュレーションオプション	1 ライセンス版	形 SYSMAC-SA401L-64
	3 ライセンス版	形 SYSMAC-SA403L-64
	10 ライセンス版	形 SYSMAC-SA410L-64
	30 ライセンス版	形 SYSMAC-SA430L-64
	50 ライセンス版	形 SYSMAC-SA450L-64

● DVD メディア

品名	メディア	対応形式
Sysmac Studio スタンダードエディション Ver.1.□□	64 ビット版 DVD	形 SYSMAC-SE200D-64

1-2-2 対応言語

対応言語は、Sysmac Studio に準じます。詳細は『Sysmac Studio Version 1 オペレーションマニュアル(SBCA-470)』を参照してください。

1-2-3 使用可能な機種

Sysmac Studio の 3D シミュレーション機能を使用できる機種は、Sysmac Studio で使用可能なコントローラのすべての機種です。詳細は『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

1-2-4 使用可能なパソコン

使用可能なパソコンは、Sysmac Studio 64 ビット版を使用可能なパソコンです。詳細は『Sysmac Studio Version 1 オペレーションマニュアル(SBCA-470)』を参照してください。
また、3D シミュレーションオプションの機能を使用する場合は、次の動作環境が必要です。

システム要件	詳細
CPU	Intel® Core™ i5 8250U processor(1.60-3.40GHz)、または相当以上のプロセッサを搭載した DOS/V（IBM AT 互換機）パーソナルコンピュータ Intel® Core™ i7 9750H 以上を推奨
RAM	8GB 以上 16GB 以上を推奨
ディスプレイ	Full HD 1920×1080 1600 万色以上
ビデオカード	DirectX11 をサポートするビデオカード（NVIDIA® GeForce®、AMD Radeon™ など)。 NVIDIA を推奨。

2

ソフトウェアのセットアップ

本章では、Sysmac Studio の 3D シミュレーション機能を使用するためのソフトウェアのセットアップ手順を説明します。

2-1 Sysmac Studio のインストール 2-2
2-2 Sysmac Studio オプションライセンスの登録 2-3

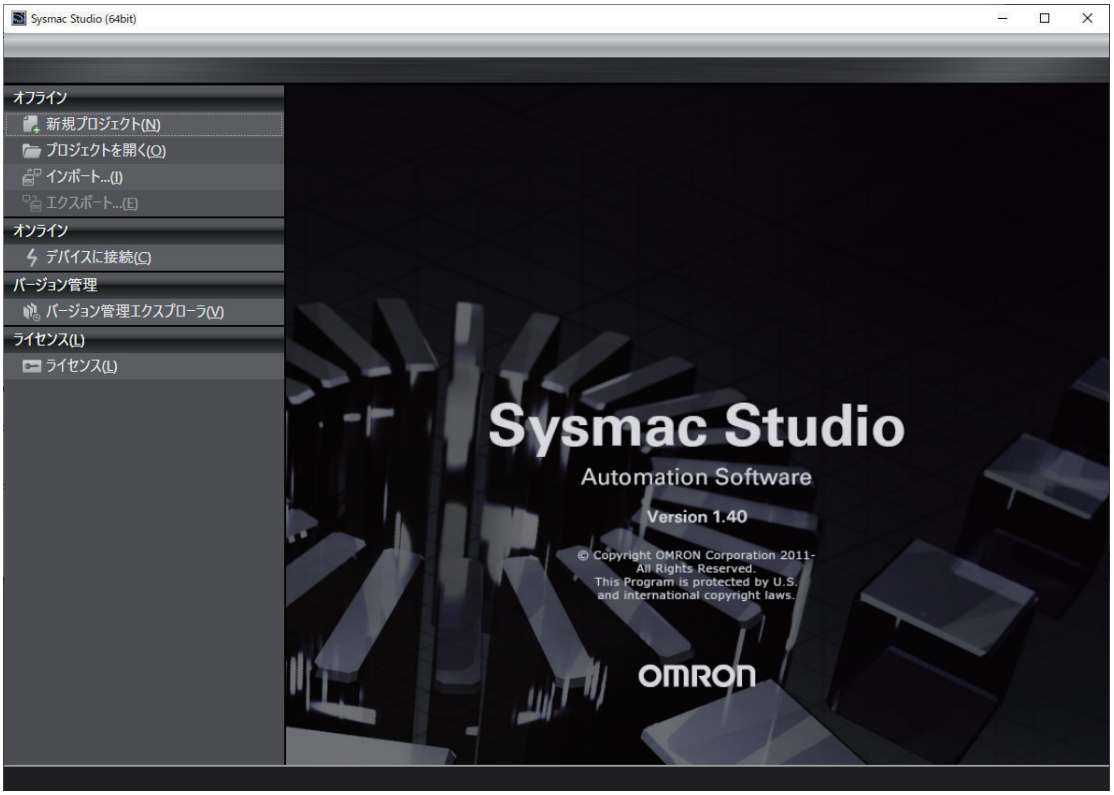
2-1 Sysmac Studio のインストール

Sysmac Studio を DVD メディアからインストールします。インストール手順の詳細は、『Sysmac Studio Version 1 オペレーションマニュアル(SBCA-470)』を参照してください。

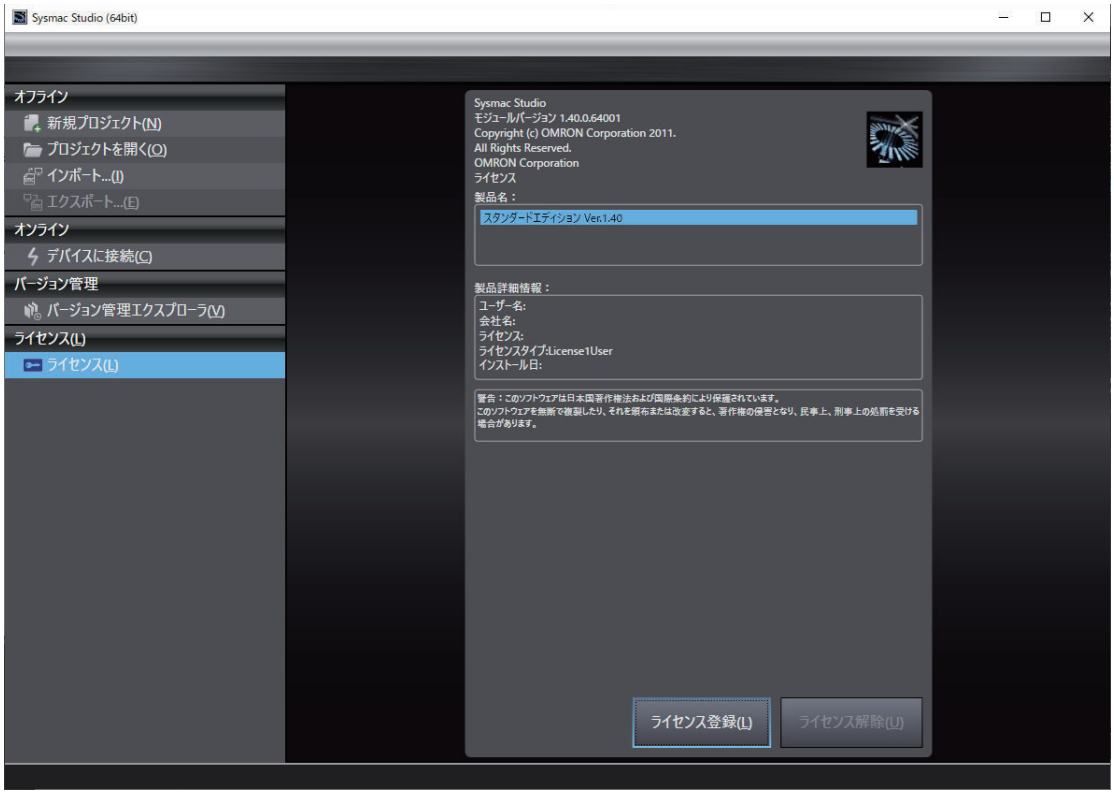
2-2 Sysmac Studio オプションライセンスの登録

メカニカル機構の 3D シミュレーションを行うためには、Sysmac Studio スタンダードエディションに「Sysmac Studio 3D シミュレーションオプション」のライセンスを登録する必要があります。
オプションライセンスの登録手順を説明します。

1 **Windows のスタートメニューから［すべてのプログラム］-［OMRON］-［Sysmac Studio］-［Sysmac Studio］を選択します。**
Sysmac Studio が起動し、スタートページが表示されます。

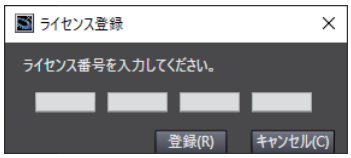


2 **スタートページで［ライセンス］をクリックします。**
現在登録されているライセンスが表示されます。



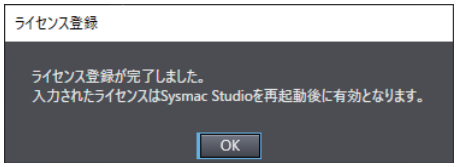
3 **［ライセンス登録］ボタンをクリックします。**

［ライセンス登録］ダイアログボックスが表示されます。



4 **Sysmac Studio 3D シミュレーションオプションのライセンス番号を入力して、［登録］ボタンをクリックします。**

登録が正常に完了すると再起動を促すメッセージが表示されます。



Sysmac Studio を再起動すると、登録完了です。

3

3D シミュレーション実行の流れ

本章では、3D シミュレーションを実行するために必要な準備とおおまかな流れを説明します。

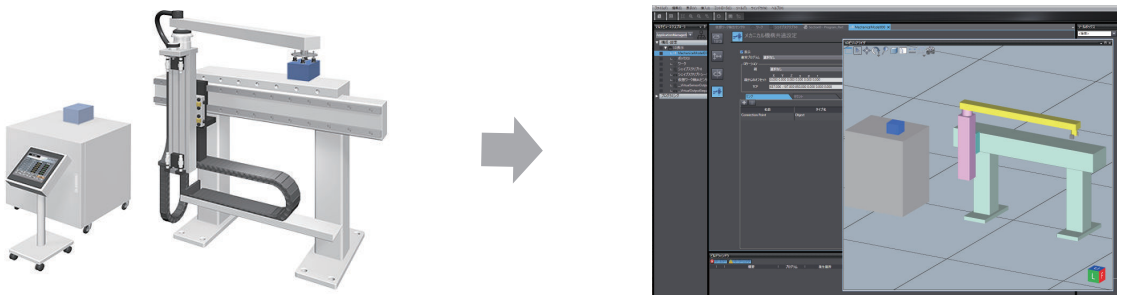
3-1　3D シミュレーションの実施概要 3-2

3-2　3D シミュレーションの実施工程 3-4

3-2-1　シミュレーション設定方法の選択 3-5
3-2-2　スクリプトを使用する場合の 3D シミュレーションの設定 3-6
3-2-3　ワークシミュレーション簡単設定を使用する場合の 3D シミュレーションの設定 3-7

3-1　3D シミュレーションの実施概要

Sysmac Studio の 3D シミュレーションは、Sysmac Studio 上に装置やワークを表す仮想装置モデルを作成し、仮想装置モデルをシミュレーションで動作させて行います。

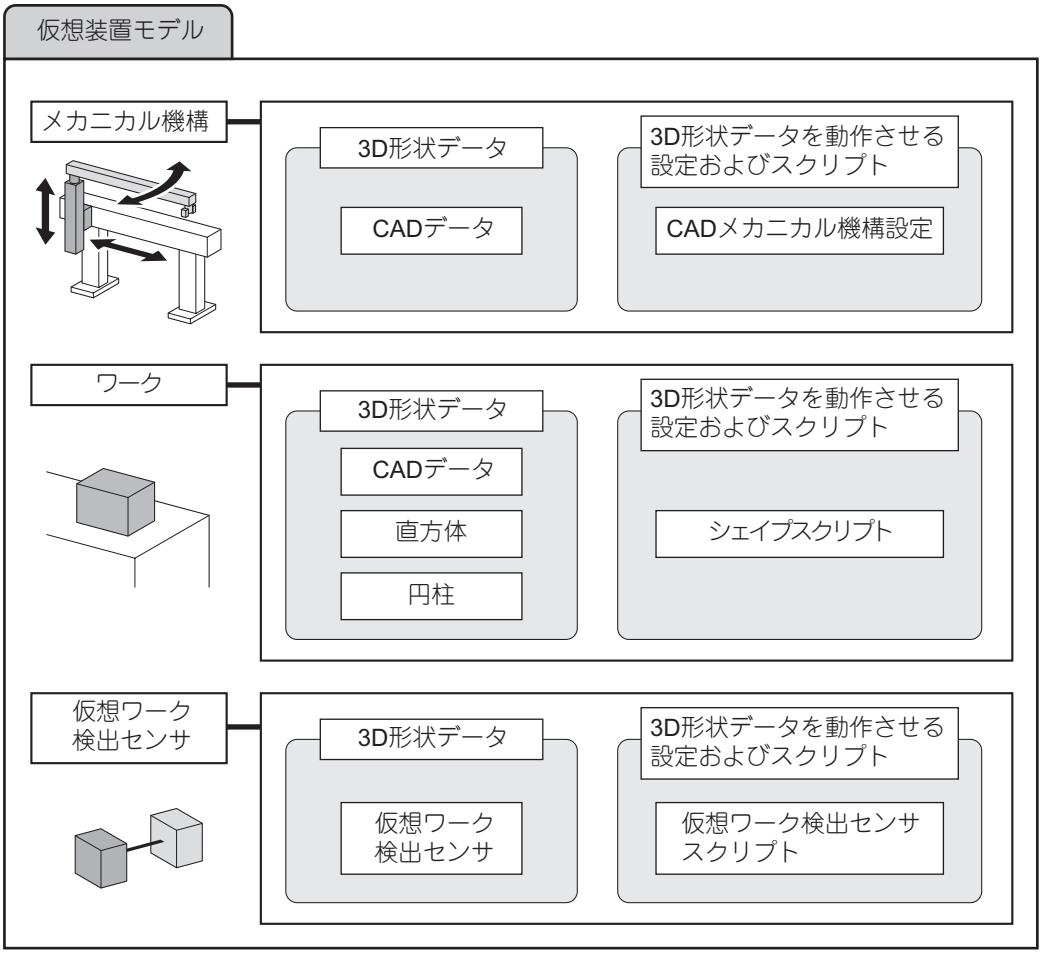


仮想装置モデルは、装置の可動部分の「メカニカル機構*1」、搬送対象である「ワーク」、ワークを検出する「仮想ワーク検出センサ」で構成されます。

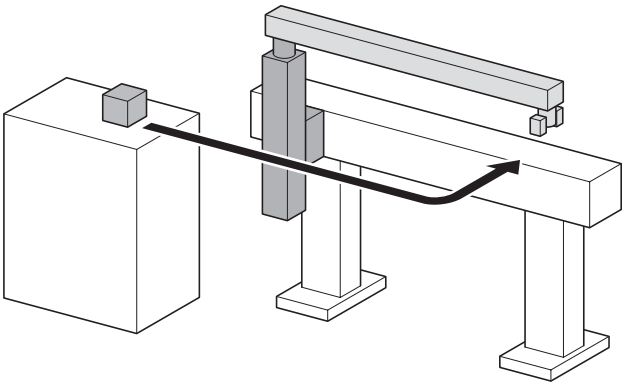
*1. カスタムメカニカル機構、パラレルリンク機構を含みます。以降、メカニカル機構、カスタムメカニカル機構、パラレルリンク機構を代表して「メカニカル機構」と呼ぶことがあります。

メカニカル機構、ワーク、仮想ワーク検出センサは、それぞれ 3D 形状データと、3D 形状データを動作させる設定およびスクリプトからなります。
3D 形状データを作成し、3D 形状データを動作させる設定およびスクリプトを設定、作成します。

仮想装置モデル、3D 形状データ、3D 形状データを動作させる設定およびスクリプトの関係は、次のとおりです。



以降では、ワークを搬送する装置を例に、3D シミュレーションの操作手順と機能について説明します。



3-2　3D シミュレーションの実施工程

Sysmac Studio を用いた 3D シミュレーションの実施工程について説明します。
それぞれの工程の詳細は、後出の章を参照してください。
3D シミュレーションを行うには、あらかじめプロジェクトを作成し、プログラムの作成や軸の登録を行います。プロジェクトの作成、プログラムの作成や軸の登録の手順は、『Sysmac Studio Version 1 オペレーションマニュアル(SBCA-470)』を参照してください。

メカニカル機構の 3D シミュレーションを例に実施工程について説明します。

工程	説明	参照先
プロジェクトの作成 プログラムの作成、軸の登録	Sysmac Studio のプロジェクトを作成し、可動部品（メカニカル機構）を制御するプログラムを作成します。また、可動部品を制御する軸の登録を行います。	『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』
↓		
3D 形状データの作成	プロジェクトにアプリケーションマネージャを追加し、装置のメカニカル機構、ワーク、仮想ワーク検出センサの 3D 形状データを作成します。	-
アプリケーションマネージャの追加	3D シミュレーション機能を使用するために必要なデータや設定を管理するアプリケーションマネージャを追加します。	「4-2 アプリケーションマネージャの追加 (P.4-7)」
↓		
メカニカル機構の追加	対象装置の CAD データを読み込み、メカニカル機構の 3D 形状データを作成します。 対象装置の機構に応じて、「メカニカル機構」、「カスタムメカニカル機構」、「パラレルリンク機構」から選択します。	「4-3 メカニカル機構の 3D 形状データの作成 (P.4-9)」 「4-4 カスタムメカニカル機構の 3D 形状データの作成 (P.4-30)」 「4-5 パラレルリンク機構の 3D 形状データの作成 (P.4-78)」
↓		
ワークの追加	ワークの CAD データを読み込み、3D 形状データを追加します。CAD データの代わりに、Sysmac Studio に標準で用意されている直方体や円柱を追加することでワークの 3D 形状データを作成することもできます。	「4-6 ワークの 3D 形状データの追加 (P.4-89)」
↓		
ワーク検出センサの追加	ワークを検出する仮想のセンサ「仮想ワーク検出センサ」の 3D 形状データを追加します。	「4-7 ワーク検出センサの追加 (P.4-98)」
↓		
シミュレーションの設定	3D シミュレーションの実施方法として、スクリプトを用いる方法またはワークシミュレーション簡単設定を用いる方法を選択します。	「3-2-1 シミュレーション設定方法の選択 (P.3-5)」

工程	説明	参照先
	スクリプトを用いる方法を選択した場合の設定方法。	「3-2-2 スクリプトを使用する場合の 3D シミュレーションの設定 (P.3-6)」
	ワークシミュレーション簡単設定を用いる方法を選択した場合の設定方法。	「3-2-3 ワークシミュレーション簡単設定を使用する場合の 3D シミュレーションの設定 (P.3-7)」
シミュレーションの実行	3D シミュレーションを実行し、仮想装置モデルの動作を確認します。	-
	スクリプトを用いた場合の 3D シミュレーションの実行方法。	「8-1-2 仮想ワーク検出センサの動作スクリプト、およびメカニカル機構の仮想出力用スクリプトを実行する (P.8-3)」
	ワークシミュレーション簡単設定を用いた場合の 3D シミュレーションの実行方法。	「8-1-4 ワーク動作設定のシミュレーションを実行する (P.8-4)」

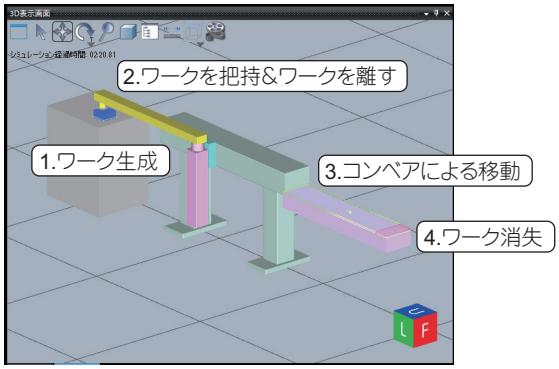
3-2-1 シミュレーション設定方法の選択

3D シミュレーションを行う方法は、スクリプトを用いる方法と、ワークシミュレーション簡単設定を用いる方法があります。以下の特徴を確認し、3D シミュレーションの実施方法を選択してください。

3D シミュレーションの実施方法	特徴
スクリプトを使用	装置の設計段階で検討されたメカニカル機構の動作仕様を元に、ワークの動きをスクリプトの関数に置き換えて、スクリプト（C#言語ベースのプログラム）を入力します。 3D 表示画面で表示されている 3D 形状データはすべてスクリプトで取り扱うことができるため、柔軟性があります。
ワークシミュレーション簡単設定を使用	装置の設計段階で検討されたメカニカル機構の動作仕様を元に、各メカニカル機構がワークにどのような動きを与えるかをメカニカル機構の設定として入力します。スクリプトを使用する場合とは異なり、スクリプトを作成するための検討やスクリプトの入力が不要なため、単時間でワークシミュレーションを行うことができます。 ワークのシミュレーションのための動作に特化しているため、複雑な 3D 表示を行う場合はスクリプトの併用も必要となる場合があります。

3-2-2　スクリプトを使用する場合の 3D シミュレーションの設定

スクリプトを使用して 3D シミュレーションを実施する場合の概要は、以下のとおりです。



装置の設計段階で検討されたメカニカル機構の動作仕様

No.	関連する機構のふるまい	使用する関数	状態(StepID)
1	ワーク生成	LoadPart()	-
2	ワークを把持	ClampPart()	1
	ワークを離す	ReleasePart()	2
3	移動	MoveObjectOnBelt()	-
4	ワーク消失	UnloadCollidingPart()	-

ワークの動きをスクリプトの関数に置き換え

```
        // Please assign your shape.
        IBox partModel = (IBox) ace["/ApplicationManager0/Part"];
        this.CreateMultipleObjects(partModel, "Part", "PartGroup", 2, renderInfoList);

        return renderInfoList;
    }

    /// <summary>
    /// Called to render the Shape Script object
    /// </summary>
    /// <returns>The list of shapes to render</returns>
    public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
        this.InitializeObject(renderInfoList);

        IBox partModel = (IBox) ace["/ApplicationManager0/Part"];
        IBox handModel = (IBox) ace["/ApplicationManager0/HandBox"];
        IBox beltModel = (IBox) ace["/ApplicationManager0/Belt"];
        IBox pickPlaceModel = (IBox) ace["/ApplicationManager0/PickPlace"];
        IBox stopperModel = (IBox) ace["/ApplicationManager0/Stopper"];
        var controllerName = "new_Controller_0";
```

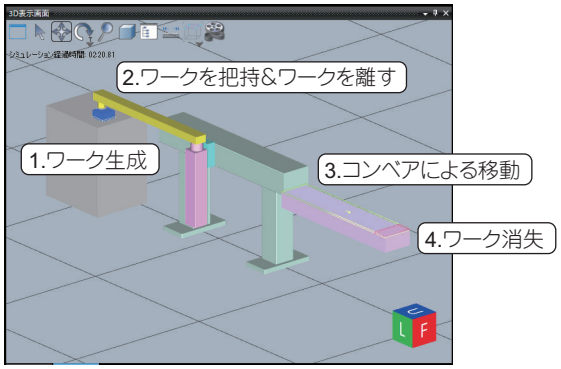
スクリプト（C#言語ベースのプログラム）を入力

以下に、スクリプトを用いた 3D シミュレーションの設定の流れについて説明します。

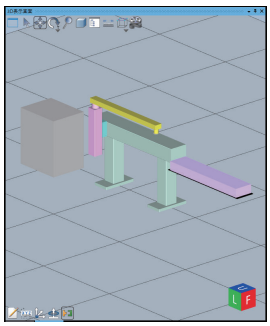
工程	説明	参照先
3D 形状データを動作させる設定およびスクリプトの作成	装置の 3D 形状データを動作させる設定やスクリプトの作成を行います。	-
メカニカル機構の設定	メカニカル機構の動作設定を行います。	「6-2 メカニカル機構の設定 (P.6-10)」
↓		
ワークの動作スクリプトの作成	ワークの動作を定義するスクリプトを作成します。	「6-3 ワークの動作スクリプトの作成 (P.6-12)」
↓		
ワーク検出センサの動作設定	仮想ワーク検出センサの動作設定を行います。動作設定から、動作を定義するスクリプトを生成します。	「6-4 ワーク検出センサの動作設定 (P.6-16)」

3-2-3 ワークシミュレーション簡単設定を使用する場合の 3D シミュレーションの設定

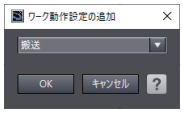
ワークシミュレーション簡単設定を使用して 3D シミュレーションを実施する場合の概要は、以下のとおりです。



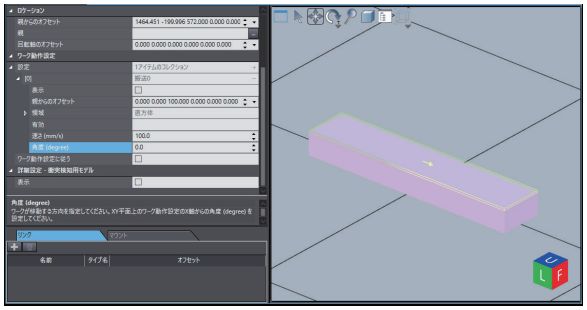
装置の設計段階で検討されたメカニカル機構の動作仕様



(1) 3D表示画面で、「搬送」を設定したい3D形状データをダブルクリックします。



(2)［ワーク動作設定の追加］ダイアログボックスで［搬送］を選択します。



(3) ワークを移動する領域、方向、スピードを設定します。

各メカニカル機構がワークにどのように動きを与えるかをメカニカル機構の設定として入力

以下に、ワークシミュレーション簡単設定を用いた 3D シミュレーションの設定の流れについて説明します。

工程	説明	参照先
ワークの 3D 形状データを動作させる設定	メカニカル機構など仮想装置モデルを構成する 3D 形状データの動作により、ワークを動作させるための設定を行います。	-
メカニカル機構の設定	メカニカル機構の動作設定を行います。	「7-2 メカニカル機構の設定 (P.7-3)」
↓		
ワークを動作させる設定	メカニカル機構など仮想装置モデルを構成する 3D 形状データに、ワークを動作させる設定を行います。	「7-3 ワーク動作設定 (P.7-4)」

4

3D 形状データの作成

Sysmac Studio で 3D シミュレーションを実行するために、アプリケーションマネージャを追加し、メカニカル機構やワーク、仮想ワーク検出センサの 3D 形状データを作成します。

4-1　3D 形状データの概要 4-3
4-1-1　3D 形状データのロケーション 4-3
4-1-2　3D 形状データの親子関係 4-6
4-2　アプリケーションマネージャの追加 4-7
4-3　メカニカル機構の 3D 形状データの作成 4-9
4-3-1　サポートする CAD データファイルの種類 4-9
4-3-2　CAD データファイルの準備 4-9
4-3-3　CAD データのインポート手順 4-11
4-3-4　メカニカル機構モデルの種類 4-17
4-3-5　メカニカル機構の設定項目 4-25
4-4　カスタムメカニカル機構の 3D 形状データの作成 4-30
4-4-1　カスタムメカニカル機構で実現できる機構 4-30
4-4-2　カスタムメカニカル機構の 3D 形状データの作成の流れ 4-31
4-4-3　サポートする OpenUSD ファイルについて 4-32
4-4-4　カスタムメカニカル機構の追加手順 4-33
4-4-5　可動部品の追加／CAD データのインポート 4-36
4-4-6　USDPhysicsJoint を含む OpenUSD ファイルのインポート 4-40
4-4-7　TCP の設定 4-45
4-4-8　結合部の動作設定 4-47
4-4-9　可動部品の動作方向設定 4-55
4-4-10　カスタムメカニカル機構の動作確認 4-60
4-4-11　カスタムメカニカル機構の動作設定 4-62
4-4-12　CAD データの更新 4-69
4-4-13　OpenUSD ファイルのエクスポート 4-70
4-4-14　カスタムメカニカル機構の作成例 4-71
4-5　パラレルリンク機構の 3D 形状データの作成 4-78
4-5-1　パラレルリンク機構の追加手順 4-78
4-5-2　パラレルリンク機構の設定項目 4-79
4-5-3　CAD データのインポート 4-86
4-6　ワークの 3D 形状データの追加 4-89
4-6-1　CAD データのインポート 4-89
4-6-2　直方体、円柱の追加 4-93
4-7　ワーク検出センサの追加 4-98
4-7-1　仮想ワーク検出センサの追加 4-98

4-7-2　仮想ワーク検出センサの設定項目 4-98

4-8　3D 形状データの配置方法 4-101
4-8-1　座標の直接入力 4-101
4-8-2　3D 表示画面上でのドラッグ&ドロップ 4-101
4-8-3　リンクポイントへのスナップ 4-101

4-9　3D 表示オブジェクトのエクスポート／インポート 4-102
4-9-1　エクスポート／インポート可能な 3D 表示オブジェクトの種類 4-102
4-9-2　エクスポート／インポート操作可能なアクセスレベル 4-102
4-9-3　3D 表示オブジェクトのエクスポート 4-102
4-9-4　3D 表示オブジェクトの一括エクスポート 4-103
4-9-5　3D 表示オブジェクトのインポート手順 4-105

4-1 3D 形状データの概要

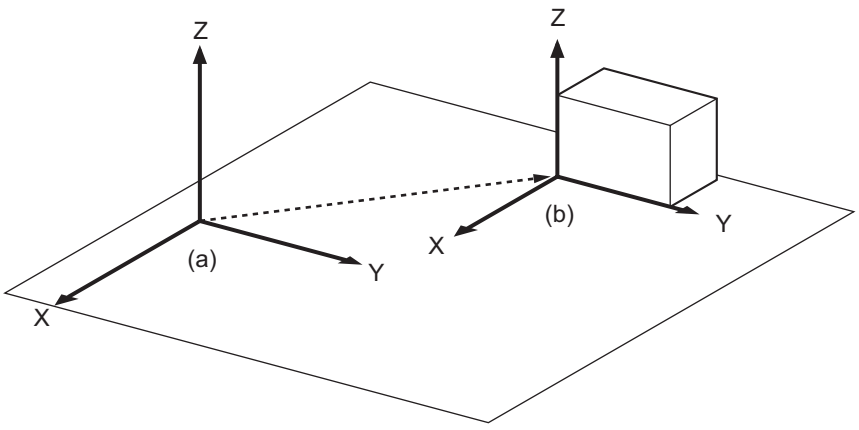
3D 形状データは、3D シミュレーション機能で使用する、装置の可動部品やワーク、ワーク検出センサの、形状、サイズ、位置、姿勢を表すデータです。3DCAD ソフトウェアで作成された CAD データや、Sysmac Studio に標準で用意されている直方体や円柱のデータを使用して作成します。
Sysmac Studio の 3D 表示画面および 3D 編集エリア上では、3D 形状データの位置は座標値で表されます。
2 つの 3D 形状データの間に主従関係（親子関係）を設定し、ワークの搬送動作を実現することができます。

4-1-1 3D 形状データのロケーション

3D 形状データの位置と姿勢は、ロケーションの要素（X、Y、Z、ヨー、ピッチ、ロール）で表されます。
位置、姿勢の定義は次のとおりです。

3D 形状データの位置

3D 形状データの位置を表す座標系には、ワールド座標系とローカル座標系の 2 つの座標系があります。
3D 形状データの位置は、ワールド座標系またはローカル座標系の座標成分「X、Y、Z」で表されます。

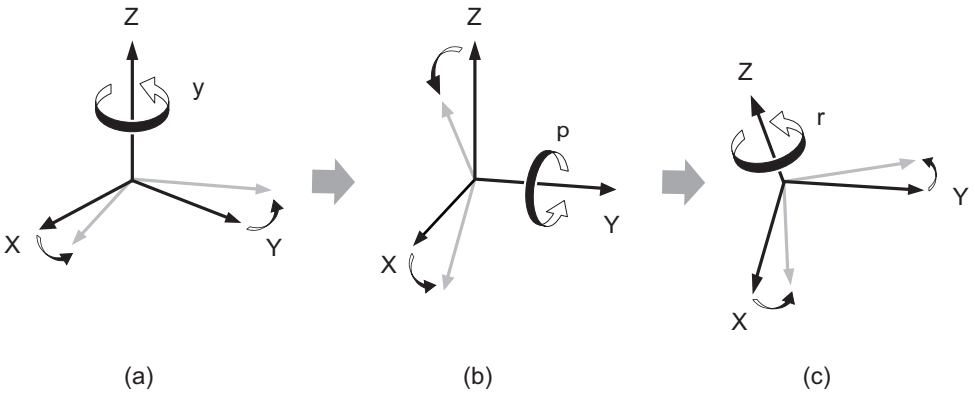


	座標系	説明
(a)	ワールド座標系	3D 表示画面上の絶対位置を表す座標系です。
(b)	ローカル座標系	各 3D 形状データが持つ座標系です。3D 形状データの回転の中心を決めるオフセットや、2 つの 3D 形状データ間の相対位置を設定するときに使用します。

座標成分	説明	単位
X	X 軸方向の位置を示します。	mm
Y	Y 軸方向の位置を示します。	mm
Z	Z 軸方向の位置を示します。	mm

3D 形状データの姿勢

3D 形状データの姿勢は、ローカル座標系の原点を中心とした軸回りの角度「y（ヨー）、p（ピッチ）、r（ロール）」で表します。本 3D シミュレーション機能においては、「y（ヨー）、p（ピッチ）、r（ロール）」を次のように定義します。



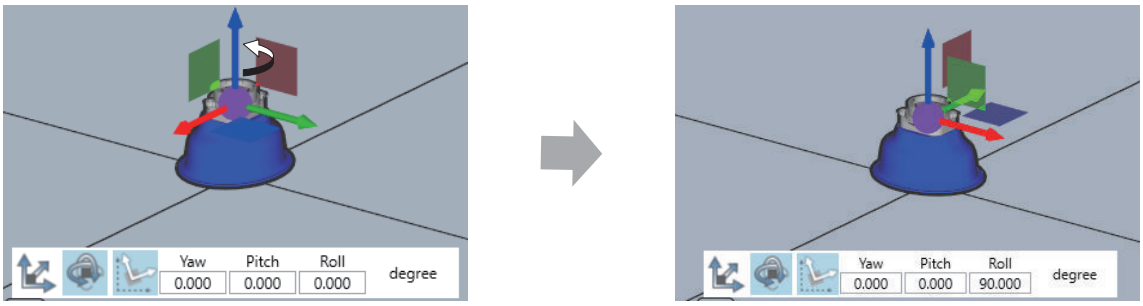
(a)　(b)　(c)

	回転	説明	単位
(a)	y（ヨー）	ローカル座標の Z 軸まわりの回転角を表します。	°（度）
(b)	p（ピッチ）	ヨーを計算したあとの、ローカル座標の Y 軸まわりの回転角を表します。	°（度）
(c)	r（ロール）	ヨーおよびピッチを計算したあとの、ローカル座標系の Z 軸まわりの回転角を表します。	°（度）

これは、Z-Y-Z オイラー角の考え方と同様です。Z 軸まわり、Y 軸まわり、Z 軸まわりの順番に y（ヨー）、p（ピッチ）、r（ロール）の値を回転させます。
下記に例を示します。

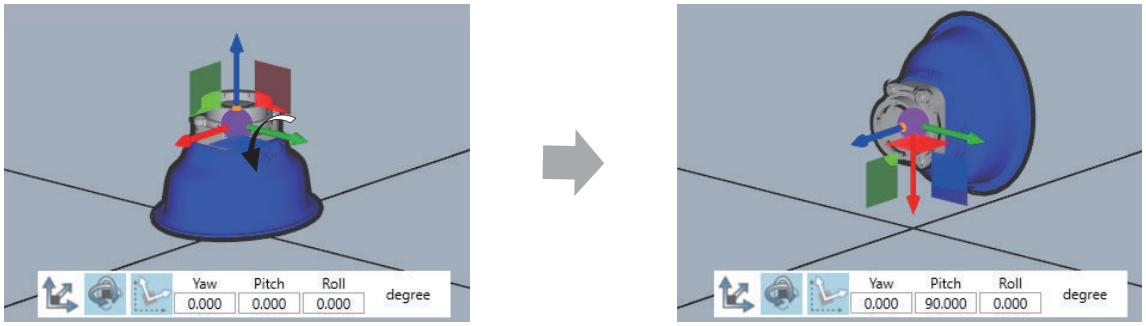
● Z 軸（青色の矢印）方向の回転

Z 軸方向に 90° 回転させる場合、「y（ヨー）」あるいは「r（ロール）」の値を「90」に設定します。
ただし、「y（ヨー）」の値を「90」に設定した場合、内部での計算により「y（ヨー）」の値が「0」、「r（ロール）」の値が「90」に自動で変換されます。



● Y 軸（緑色の矢印）方向の回転

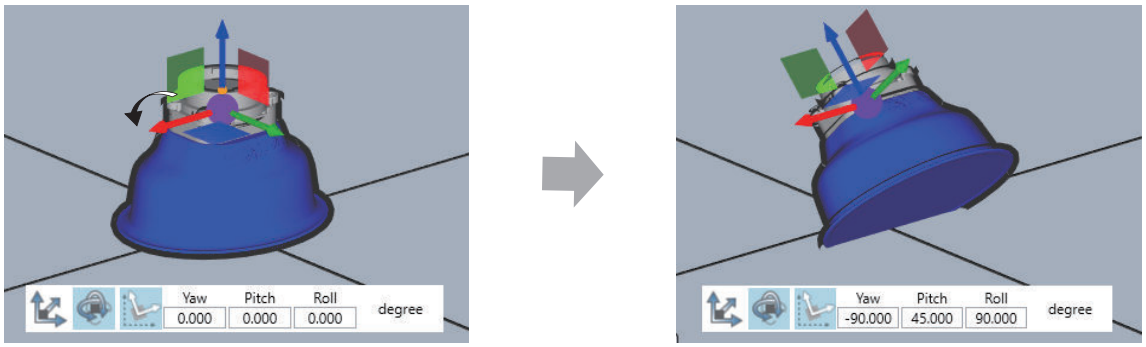
Y 軸方向に 90° 回転させる場合、「p（ピッチ）」の値を「90」に設定します。



● X 軸（赤色の矢印）方向の回転

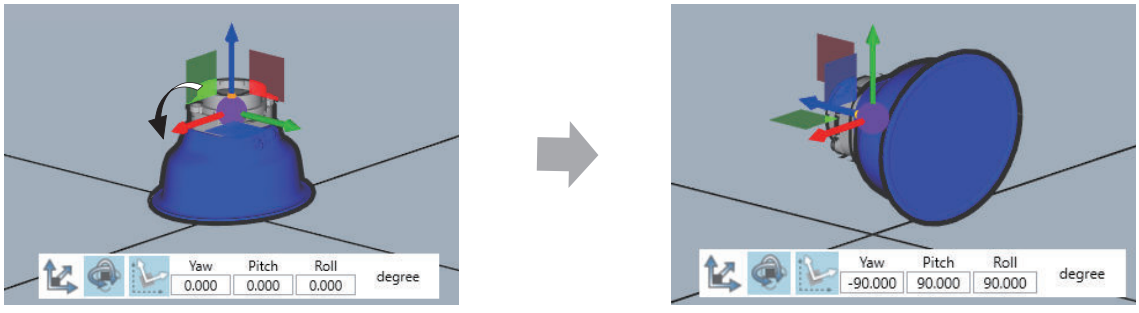
X 軸方向の場合は、Y 軸方向、Z 軸方向とは違い、二つ以上のパラメータに数値を設定する必要があります。

- X 軸方向に 45° 回転させる場合



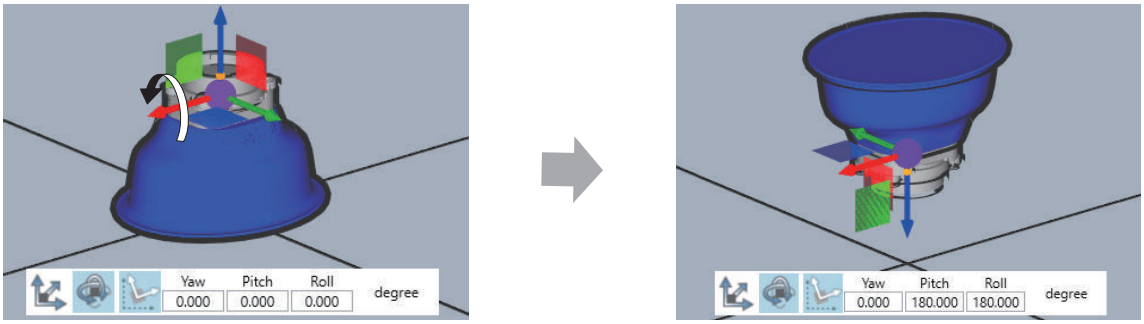
「y（ヨー）」の値を「-90」、「p（ピッチ）」の値を「45」、「r（ロール）」の値を「90」に設定します。

- X 軸方向に 90° 回転させる場合



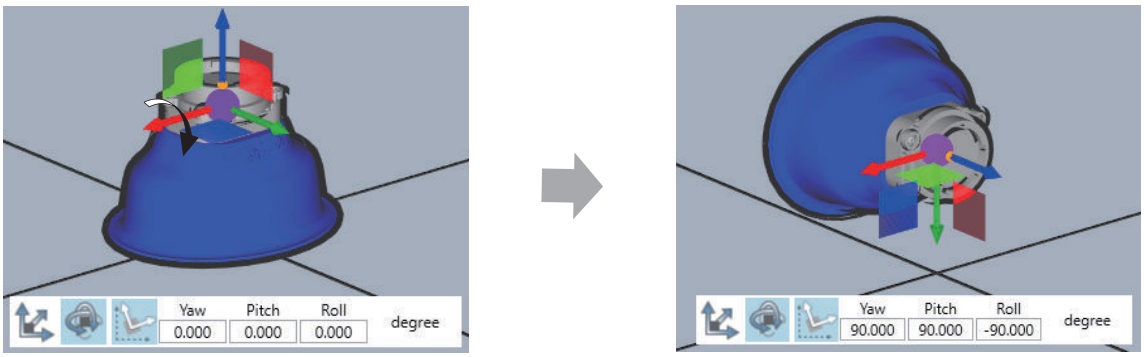
「y（ヨー）」の値を「-90」、「p（ピッチ）」の値を「90」、「r（ロール）」の値を「90」に設定します。

- X 軸方向に 180° 回転させる場合



「p（ピッチ）」の値を「180」、「r（ロール）」の値を「180」に設定します。

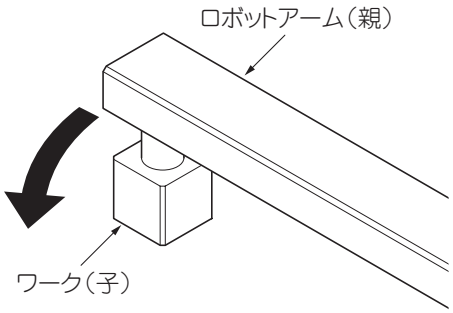
- X 軸方向に-90° 回転させる場合



「y（ヨー）」の値を「90」、「p（ピッチ）」の値を「90」、「r（ロール）」の値を「-90」に設定します。

4-1-2 3D 形状データの親子関係

ロボットアームでワークを吸着して移動させる場合のように、一方の 3D 形状データがもう一方の 3D 形状データに追従する動作は、2 つの 3D 形状データの親子関係を設定して実現します。
3D 形状データが動作するとき、動作の主体となる 3D 形状データを「親」、親の動作に追従する 3D 形状データを「子」と呼びます。

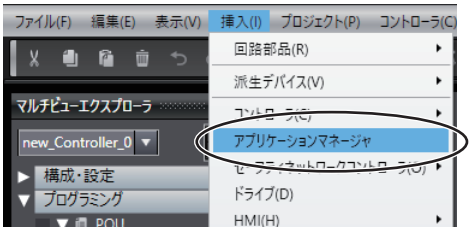


親子関係を設定するには、子となる 3D 形状データの設定画面で、親となる 3D 形状データを選択します。

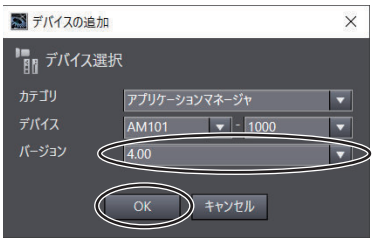
4-2　アプリケーションマネージャの追加

アプリケーションマネージャは 3D シミュレーション機能を使用するために必要なデータや設定を管理する論理デバイスです。
3D 形状データを作成するために、アプリケーションマネージャをプロジェクトに追加します。
アプリケーションマネージャの追加手順は以下のとおりです。

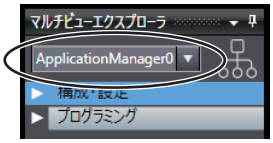
1　メインメニューから［挿入］-［アプリケーションマネージャ］を選択します。



［デバイスの追加］ダイアログボックスが表示されます。その画面の［バージョン］のプルダウンメニューで、追加したいバージョンを選択し、［OK］ボタンをクリックします。

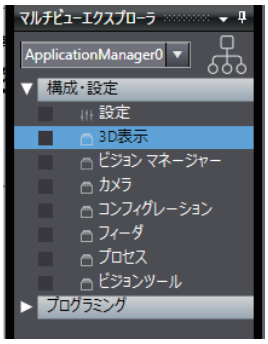


マルチビューエクスプローラのデバイスリストに［アプリケーションマネージャ］が追加されます。



参考

アプリケーションマネージャには、3D シミュレーション機能を使用するために必要なデータや設定を管理する項目の他に、ロボットシステムを制御する IPC Application Controller の設定やプログラムを管理する項目があります。

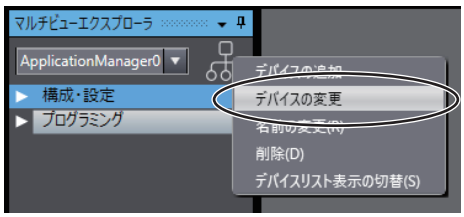


［構成・設定］の［3D 表示］では、3D シミュレーションで使用する仮想装置モデルの 3D 形状データ、メカニカル機構や仮想ワーク検出センサを動作させるための設定、ワークの動作を定義するスクリプトを管理します。
以降の説明では、簡単化のため［3D 表示］だけを表示した画面を用います。
IPC Application Controller の設定やプログラムを管理する項目の詳細については、『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル IPC Application Controller 編(SBCA-483)』を参照してください。

アプリケーションマネージャのバージョンの確認、および変更

追加したアプリケーションマネージャのバージョンの確認、およびバージョンの変更手順は下記のとおりです。

1 プロジェクトのデバイスアイコンの右クリックメニューの［デバイスの変更］を選択します。



［デバイスの変更］ダイアログボックスが表示されます。そのダイアログボックスの［バージョン］の値がアプリケーションマネージャのバージョンとなります。
また、［デバイスの変更］ダイアログボックスにより、バージョンを変更することができます。ただし、変更前のバージョンの値より小さい値を設定することはできません。

2 ［デバイスの変更］ダイアログボックスの［バージョン］のプルダウンリストから変更対象のバージョンを選択し、［OK］ボタンをクリックします。



4-3 メカニカル機構の 3D 形状データの作成

メカニカル機構の 3D 形状データは、装置の可動部品の CAD データをインポートして作成します。インポートする際に、Sysmac Studio に用意されている機構モデルから該当する機構を選択し、可動部の動作を設定します。

参考

CAD データは、あらかじめ装置メーカや部品メーカの提供する CAD データを入手するか、3DCAD ソフトウェアで作成しておきます。

4-3-1 サポートする CAD データファイルの種類

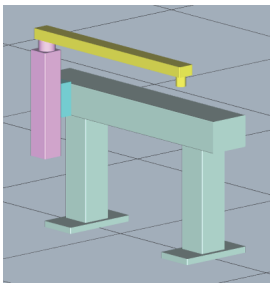
Sysmac Studio で読み込み可能な CAD データのファイルの種類は以下のとおりです。

CAD データの形式	対応バージョン
STEP	AP203、AP214、AP242
IGES	Ver.5.3 以降
OpenUSD	OpenUSD 23.11 SDK をサポートしている OpenUSD

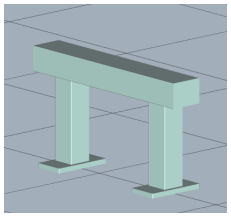
4-3-2 CAD データファイルの準備

メカニカル機構部品の CAD データファイルをインポートするために、対応するメカニカル機構に合わせて、機構を構成する部品ごとに CAD データファイルを作成します。
3D シミュレーション機能がサポートする「メカニカル機構」の種類の詳細については「4-3-4 メカニカル機構モデルの種類 (P.4-17)」を参照してください。

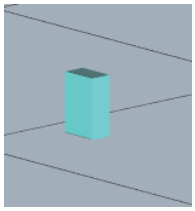
例: XY テーブル(XYθ)の機構を持つ可動部品



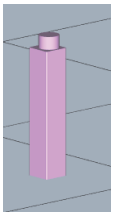
ベース、X 軸方向のスライダ、Z 軸方向のスライダ、回転軸を持つアームで構成される可動部品の CAD データを、メカニカル機構「XY テーブル（XYθ）」としてインポートする場合、ベース、X 軸方向のスライダ、Z 軸方向のスライダ、回転軸を持つアームの CAD データファイルをそれぞれ作成しておきます。（この例では、Z 軸を XY テーブル(XYθ)の Y 軸に当てはめています）



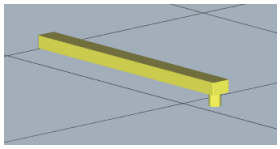
ベース
(XYThetaTable_base.step)



X軸方向のスライダ
(XYThetaTable_X.step)



Z軸方向のスライダ
(XYThetaTable_Y.step)



回転軸を持つアーム
(XYThetaTable_theta.step)

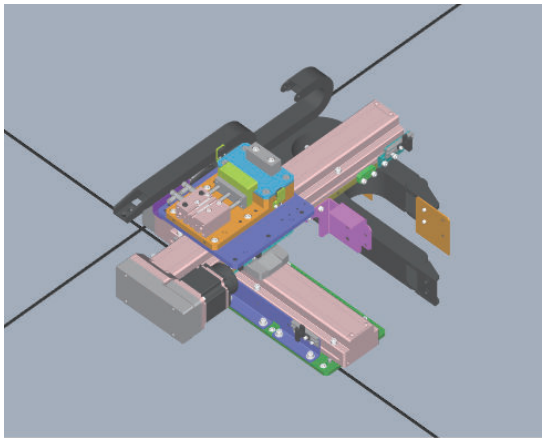
3DCAD ソフトウェアによる CAD データファイルの出力手順

メカニカル機構は、土台となるベース部品、軸または I/O の出力にあわせて動作する可動部品で構成されます。メカニカル機構を 3D 表示画面にインポートするには、3DCAD ソフトウェアを利用して出力されたベース部品と可動部品の CAD データファイルを準備する必要があります。

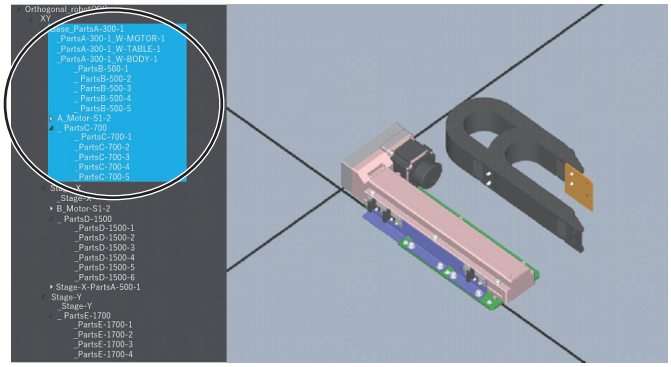
参考

- 機械設計で作成した CAD データファイルが存在する場合は、CAD データファイルをもとに 3DCAD ソフトウェアを使用し、メカニカル機構の CAD データファイルを作成してください。次に、ベース部品および各可動部品の CAD データファイルを作成してください。
- CAD データファイルを 3DCAD ソフトウェアで開くと、アセンブリ情報がツリー構造として表示されます。3DCAD ソフトウェアの多くは、ツリー構造からベース部品または各可動部品に必要なアセンブリ情報を選択することで、選択したパーツの CAD データファイルを出力することができます。これらの CAD データファイルを、メカニカル機構のベース部品または可動部品としてインポートしてください。

3DCAD ソフトウェアを使用して、直交ロボット(XY)の CAD データファイルを、ベース部品、可動部品「X ステージ」、可動部品「Y ステージ」の 3 つの CAD データファイルに出力する例を説明します。

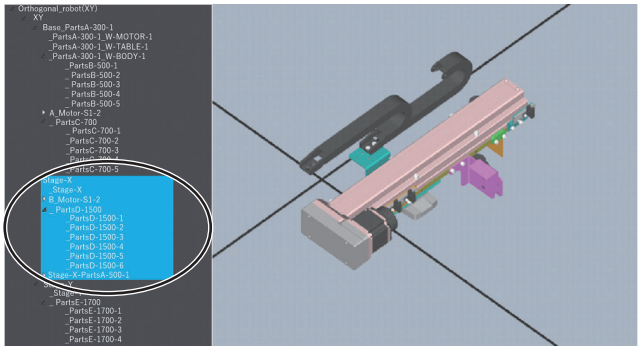


1 **3DCAD ソフトウェアに表示されるアセンブリ情報から、ベース部品に対応するアセンブリ情報を選択し、その部分だけを CAD データファイルに出力します。**
以下に、アセンブリ情報を選択した例と、選択したアセンブリ情報のみを 3D 表示画面に表示した例を示します。



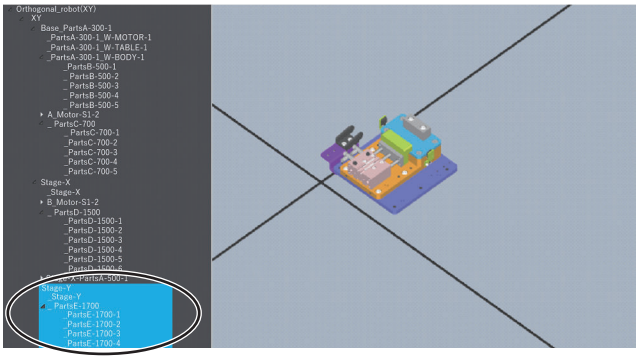
2 **可動部品「X ステージ」に対応するアセンブリ情報を選択し、その部分だけを CAD データファイルに出力します。**

以下に、アセンブリ情報を選択した例と、選択したアセンブリ情報のみを 3D 表示画面に表示した例を示します。



3 **可動部品「Y ステージ」に対応するアセンブリ情報を選択し、その部分だけを CAD データファイルに出力します。**

以下に、アセンブリ情報を選択した例と、選択したアセンブリ情報のみを 3D 表示画面に表示した例を示します。



使用上の注意

CAD データファイルのアセンブリ構造によっては、例のようにアセンブリ情報とベース部品または可動部品が適切に対応しているとは限りません。3DCAD ソフトウェアを使用して、可動する部分を正確に選択したうえで、ベース部品または各可動部品の CAD データファイルを出力してください。

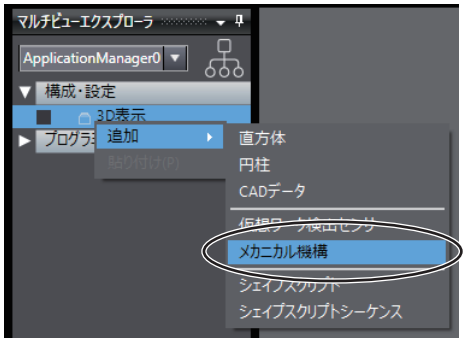
4-3-3 CAD データのインポート手順

準備した CAD データファイルをインポートします。インポート手順は以下のとおりです。

参考

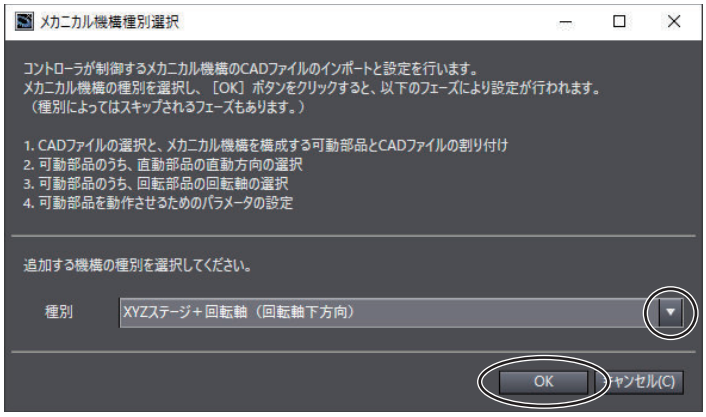
インポート時 CAD データのメッシュの粗さをオプションで設定することができます。オプション設定の詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

1 **［構成・設定］の［3D 表示］を右クリックし、メニューから［追加］-［メカニカル機構］を選択します。**



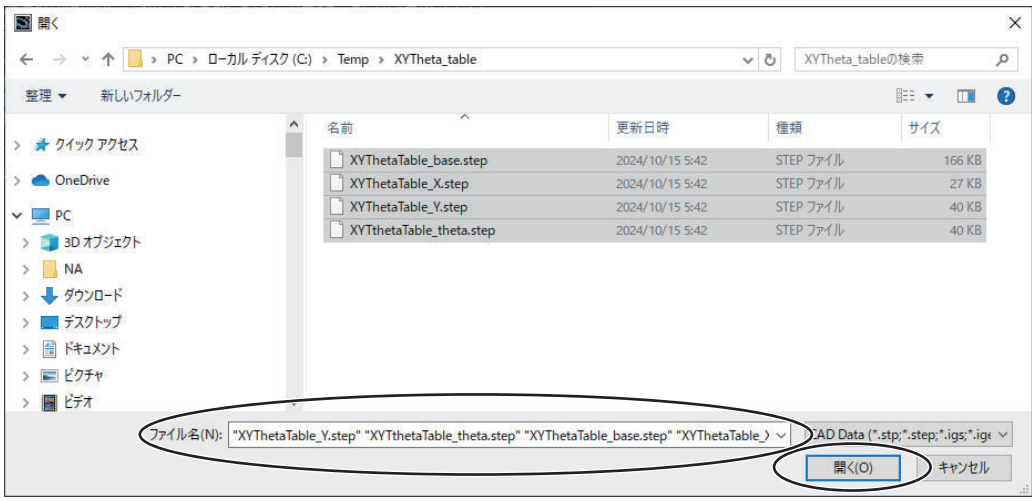
［メカニカル機構種別選択］ダイアログボックスが表示されます。

2 **メカニカル機構の種別を選択し、［OK］ボタンをクリックします。**



種別に［コンベア］以外を指定した場合、［開く］ダイアログボックスが表示されます。

3 **拡張子.stp、.step、.igs、.iges、.usd、.usda、.usdc、.usdz のいずれかの CAD ファイルを選択し、［開く］ボタンをクリックします。**



［メカニカル機構追加ウィザード］が表示されます。ウィザードのステップは次のとおりです。

ステップ	ナビゲーション内容	内容
1	可動部品選択	各 CAD ファイルとメカニカル機構を構成する可動部品の割り付けを行います。
2	直動方向選択	メカニカル機構を構成する直動部品の移動方向を設定します。
3	回転軸選択	メカニカル機構を構成する回転軸の回転方向を設定します。

ステップ	ナビゲーション内容	内容
4	パラメータ設定	メカニカル機構を構成する可動部品のパラメータを設定します。

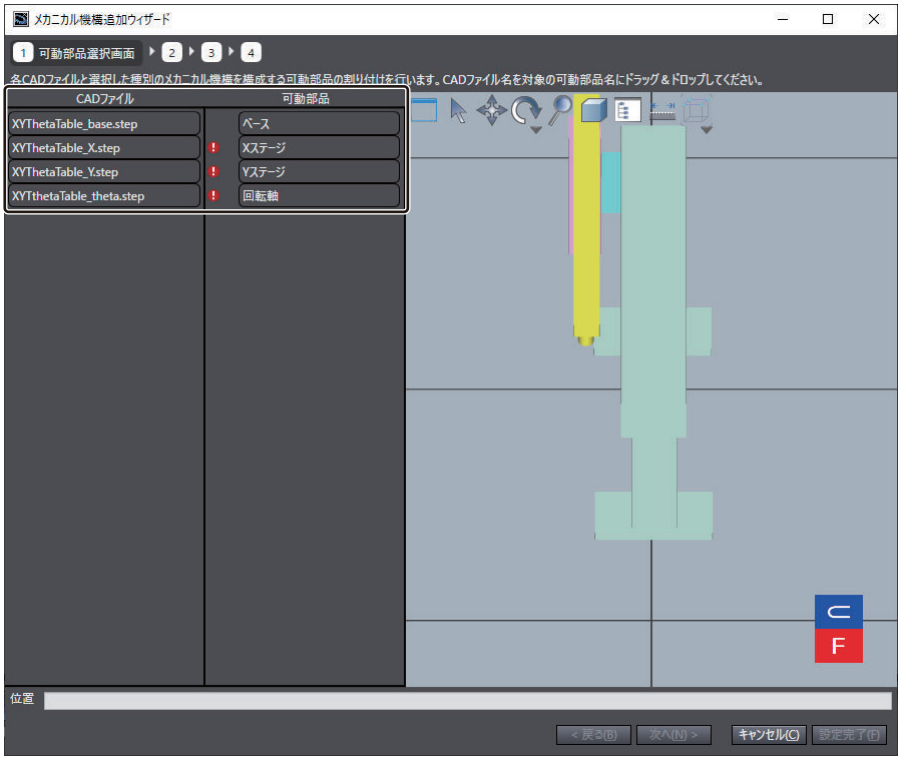
メカニカル機構の種別により設定不要なステップは省略されます。省略されるステップはグレーアウトされます。

ステップ	ナビゲーション内容	ステップが省略されるメカニカル機構の種別
1	可動部品選択	コンベア
2	直動方向選択	回転機構 1 軸
3	回転軸選択	• 直動機構 1 軸 • エアシリンダ（シングルソレノイドタイプ） • エアシリンダ（ダブルソレノイドタイプ） • ロボットツール（平行開閉型 2 爪タイプチャック／シングルソレノイドタイプ） • ロボットツール（平行開閉型 2 爪タイプチャック／ダブルソレノイドタイプ） • 直交ロボット(XY) • 直交ロボット(XYZ) • コンベア
4	パラメータ設定	なし

以降では XY テーブル(XY θ)を例に各ステップの操作手順を説明します。

ステップ 1: 可動部品選択

各 CAD ファイルとメカニカル機構を構成する可動部品の割り付けを行います。

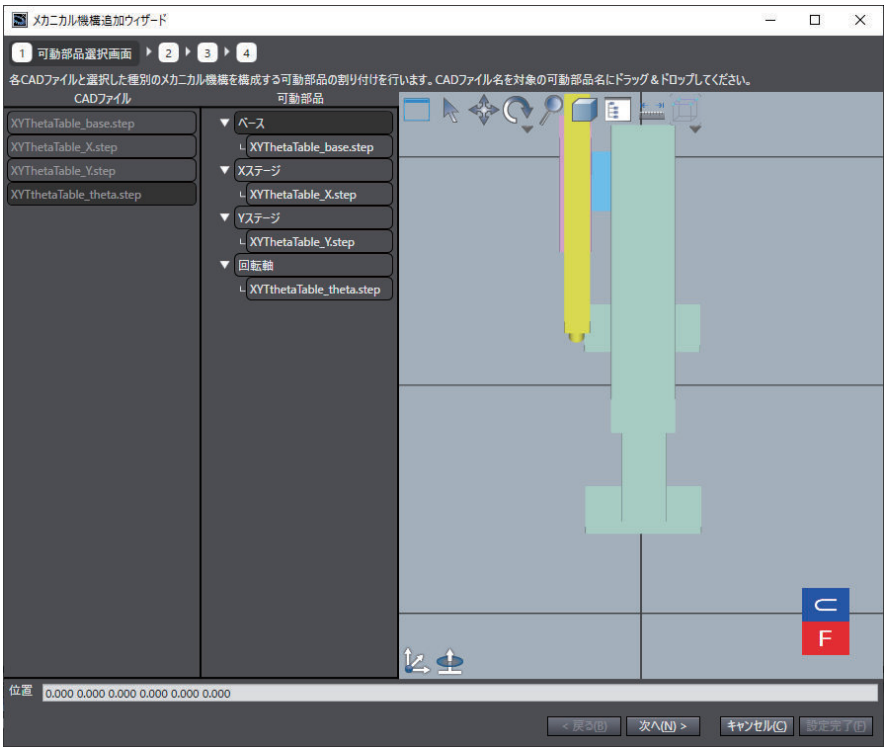


● 手順 1: 可動部品に CAD ファイルを割り付ける

各 CAD ファイルをドラッグし、各可動部品にドロップします。



可動部品下に CAD ファイルが割り付けられます。



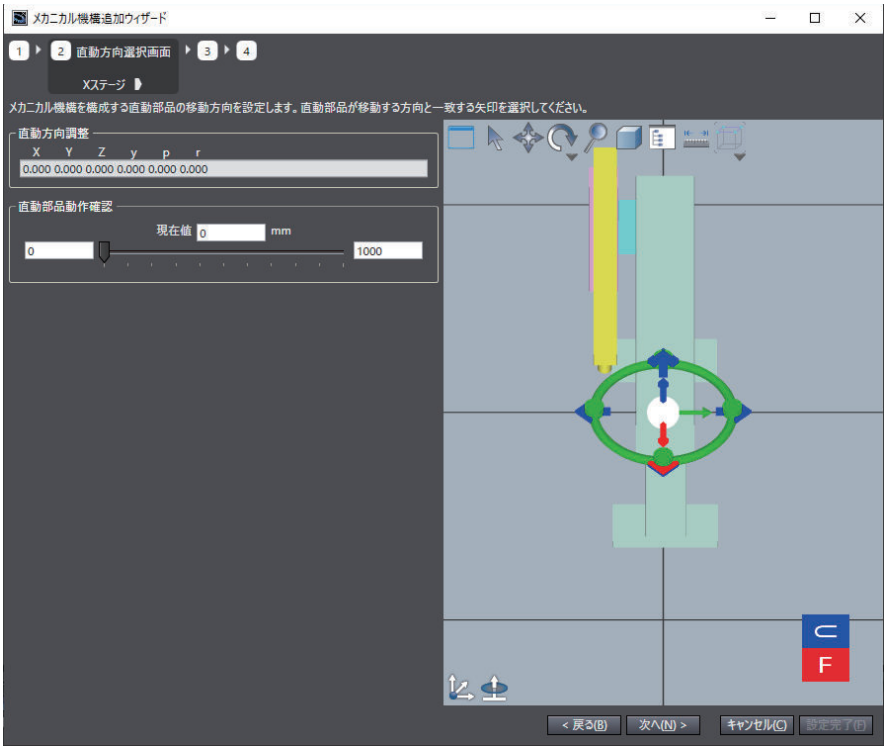
- **手順 2: 可動部品の位置を設定する**

  ［位置］にて各可動部品の位置と姿勢を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。

  すべての設定が完了したら、［次へ］ボタンをクリックします。

ステップ 2: 直動方向選択

メカニカル機構を構成する直動部品の移動方向を設定します。



- **手順 1: 可動部品の直動方向を選択する**

  3D 表示画面上の四つの青い矢印アイコンから直動する可動部品の直動方向をクリックして、選択します。選択されたアイコンは赤くなります。
  コンベアの場合、X 方向の矢印だけが表示されます。

- **手順 2: 可動部品のローカル座標原点を設定する**

  3D 表示画面上の白い球体をドラッグして移動させることで、可動部品のローカル座標原点を設定します。移動させると、［直動方向調整］の座標値も更新されます。また、［直動方向調整］の座標を設定することもできます。

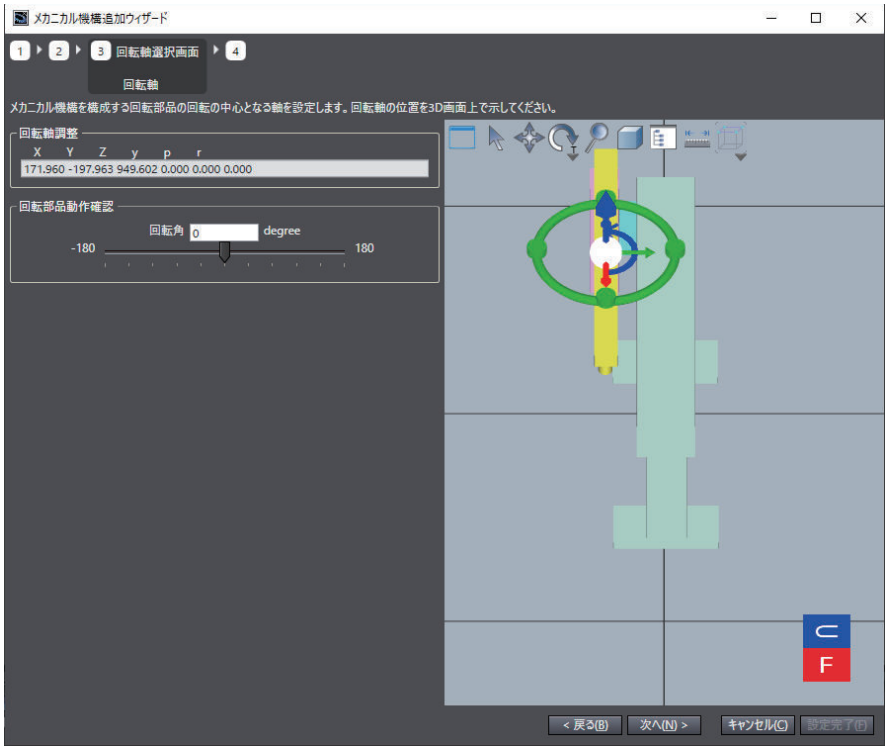
- **手順 3: 直動部品の動作を確認する**

  ［直動部品動作確認］のスライダをスライドさせて、3D 表示画面上の直動部品の動作を確認します。
  コンベアの場合、［直動部品動作確認］の項目は表示されません。

  すべての設定が完了したら、［次へ］ボタンをクリックします。

ステップ 3: 回転軸選択

メカニカル機構を構成する回転軸の回転方向を設定します。



- **手順 1: 可動部品の回転方向を設定する**

  ［回転軸調整］の y、p、r の値を設定することで、回転軸の方向を設定します。設定すると、3D 表示画面上の回転軸の方向編集アイコンの向きが変更されます。また、3D 表示画面上の回転軸の方向編集アイコンをドラックして、設定することもできます。

- **手順 2: 可動部品の回転軸の中心を設定する**

  3D 表示画面上の白い球体をドラッグして移動させることで、回転部品の回転軸の中心を設定します。移動させると、［回転軸調整］の座標値も更新されます。また、［回転軸調整］の座標を直接変更して設定することもできます。

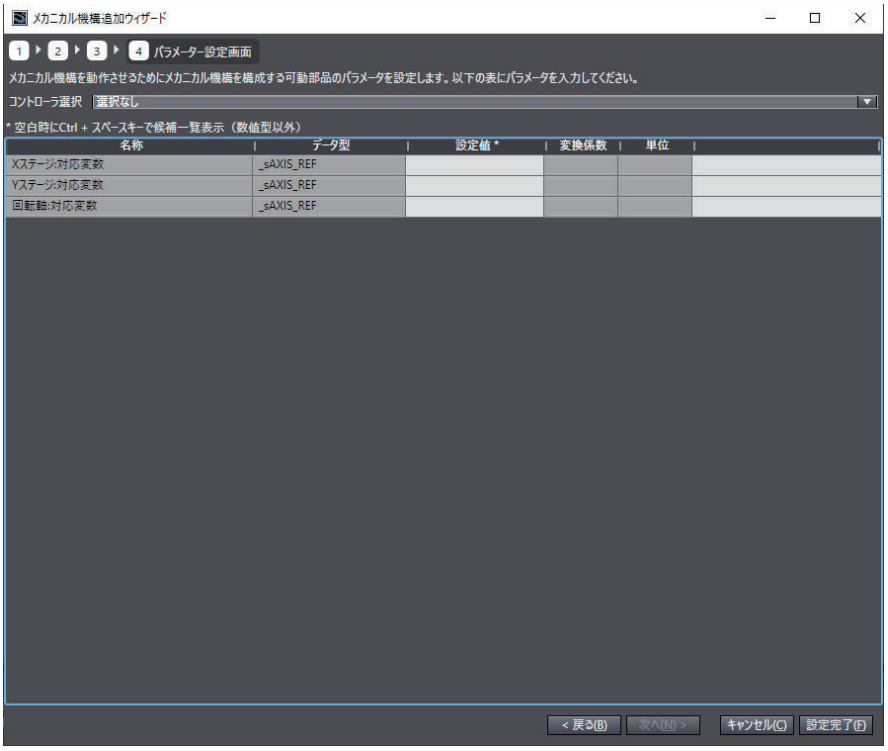
- **手順 3: 直動部品の動作を確認する**

  ［回転部品動作確認］のスライダをスライドさせて、3D 表示画面上の回転部品の動作を確認します。

  すべての設定が完了したら、［次へ］ボタンをクリックします。

ステップ 4: パラメータ設定

メカニカル機構を動作させるためにメカニカル機構を構成する可動部品のパラメータを設定します。
可動部品のパラメータはメカニカル機構の種別により異なります。
各メカニカル機構のパラメータの詳細は「4-3-4 メカニカル機構モデルの種類 (P.4-17)」を参照してください。



- **手順 1: データ型に従って変数を設定する**

  ［コントローラ選択］で、可動部品を制御するコントローラを選択し、各変数の［設定値］に、対応する軸変数を入力します。変数名をテキストボックスに直接入力するか、入力可能な状態で Ctrl + Space キーを押して表示される候補リストから選択します。

- **手順 2: 変換係数を設定する**

  変換係数が設定可能な場合は［変換係数］に倍率を入力します。
  設定が完了したら、［設定完了］ボタンをクリックします。
  ［3D 表示］の下に［MechanicalModel000］が登録されます。



4-3-4 メカニカル機構モデルの種類

Sysmac Studio でサポートしているメカニカル機構は次のとおりです。メカニカル機構はメカニカル機構追加ウィザードで設定します。詳細は、「4-3-3 CAD データのインポート手順 (P.4-11)」を参照してください。

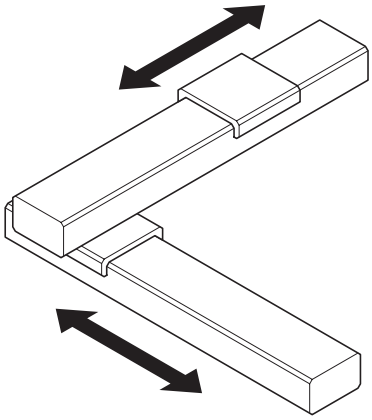
- 直交ロボット(XY)
- 直交ロボット(XYZ)
- XYZ ステージ+回転軸（回転軸上方向）
- XYZ ステージ+回転軸（回転軸下方向）
- XY テーブル(XYθ)
- XY テーブル(θXY)

- 直動機構 1 軸
- 回転機構 1 軸
- エアシリンダ（シングルソレノイドタイプ）
- エアシリンダ（ダブルソレノイドタイプ）
- ロボットツール（平行開閉型 2 爪タイプチャック／シングルソレノイドタイプ）
- ロボットツール（平行開閉型 2 爪タイプチャック／ダブルソレノイドタイプ）
- コンベア

以降に各機構で設定するパラメータを示します。

直交ロボット(XY)

直交ロボット(XY)は平面上の任意の位置へ移動可能な機構です。



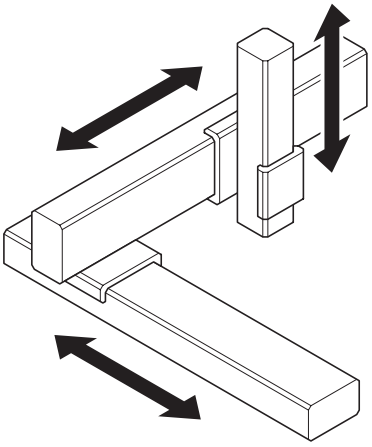
直交ロボット(XY)で設定するパラメータは次のとおりです。
軸グループの対応変数を設定するか、X ステージおよび Y ステージの対応変数をそれぞれ設定します。

設定項目	説明	設定値
軸グループ: 対応変数*1*2	直交ロボット(XY)に割り付ける軸グループの設定です。	データ型 _sGROUP_REF の変数
X ステージ: 対応変数	X ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
Y ステージ: 対応変数	Y ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数

*1. 軸グループの軸グループ使用が［未使用軸グループ］の場合、エラーとなります。
*2. 設定すると、軸グループの軸 A0 が X ステージ、軸 A1 が Y ステージに割り付けられます。

直交ロボット(XYZ)

直交ロボット(XYZ)は 3 次元空間上の任意の位置へ移動可能な機構です。



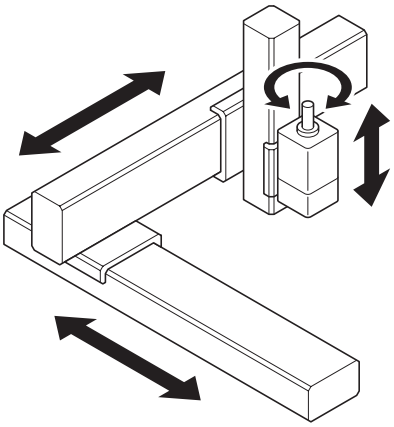
直交ロボット(XYZ)で設定するパラメータは次のとおりです。
軸グループの対応変数を設定するか、X ステージ、Y ステージ、および Z ステージの対応変数をそれぞれ設定します。

設定項目	説明	設定値
軸グループ: 対応変数*1*2	直交ロボット(XYZ)に割り付ける軸グループの設定です。	データ型 _sGROUP_REF の変数
X ステージ: 対応変数	X ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
Y ステージ: 対応変数	Y ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
Z ステージ: 対応変数	Z ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数

*1. 軸グループの軸グループ使用が［未使用軸グループ］の場合、エラーとなります。
*2. 設定すると、軸グループの軸 A0 が X ステージ、軸 A1 が Y ステージ、軸 A2 が Z ステージに割り付けられます。

XYZ ステージ＋回転軸（回転軸上方向）

XYZ ステージ＋回転軸（回転軸上方向）は XYZ ステージの先端に上方向の回転軸が付いている機構です。

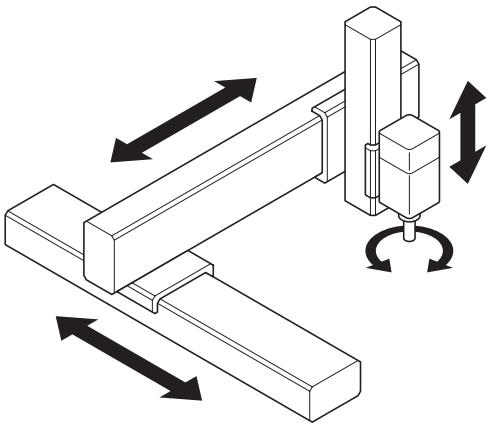


XYZ ステージ＋回転軸（回転軸上方向）で設定するパラメータは次のとおりです。

設定項目	説明	設定値
X ステージ: 対応変数	X ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
Y ステージ: 対応変数	Y ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
Z ステージ: 対応変数	Z ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
回転軸: 対応変数	回転軸の対応軸を設定します。	データ型 _sAXIS_REF の変数

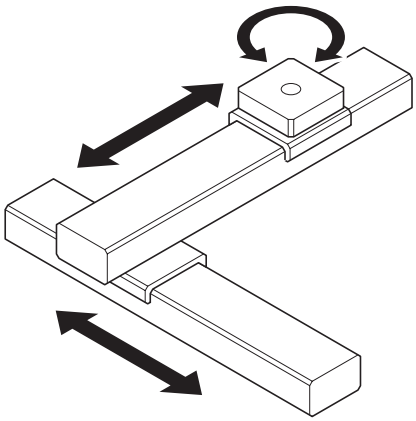
XYZ ステージ＋回転軸（回転軸下方向）

XYZ ステージ＋回転軸（回転軸下方向）は XYZ ステージの先端に下方向の回転軸が付いている機構です。
設定するパラメータは XYZ ステージ＋回転軸（回転軸上方向）と同じです。「XYZ ステージ＋回転軸（回転軸上方向）（P.4-19)」を参照してください。



XY テーブル(XYθ)

XY テーブル(XYθ)は XY テーブル上に回転機構が付いている機構です。

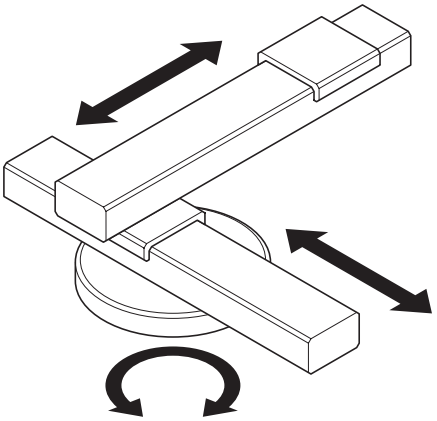


XY テーブル(XYθ)で設定するパラメータは次のとおりです。

設定項目	説明	設定値
X ステージ: 対応変数	X ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
Y ステージ: 対応変数	Y ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数
回転軸: 対応変数	回転軸の対応軸を設定します。	データ型 _sAXIS_REF の変数

XY テーブル(θXY)

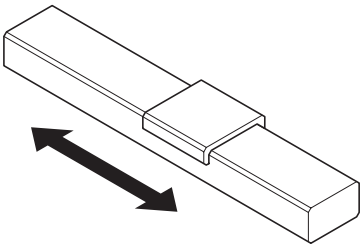
XY テーブル(θXY)は回転機構上に XY テーブルが付いている機構です。



設定するパラメータは XY テーブル(XYθ)と同じです。「XY テーブル(XYθ) (P.4-20)」を参照してください。

直動機構 1 軸

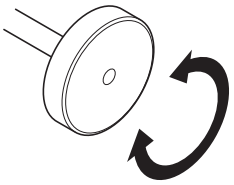
直動機構 1 軸は直線上の任意の位置へ移動可能な機構です。
直動機構 1 軸で設定するパラメータは次のとおりです。



設定項目	説明	設定値
Y ステージ: 対応変数	Y ステージの対応軸を設定します。	データ型 _sAXIS_REF の変数

回転機構 1 軸

回転機構 1 軸はモータの回転方向を視覚化する機構です。

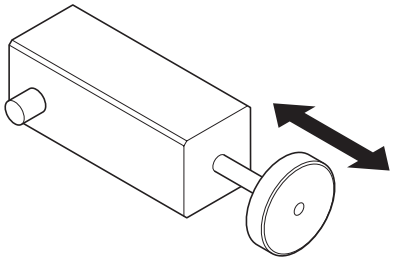


回転機構 1 軸で設定するパラメータは次のとおりです。

設定項目	説明	設定値
モータ: 対応変数	モータの対応軸を設定します。	データ型 _sAXIS_REF の変数

エアシリンダ（シングルソレノイドタイプ）

エアシリンダ（シングルソレノイドタイプ）は、ひとつの入力値により、ピストン部分が移動する機構です。

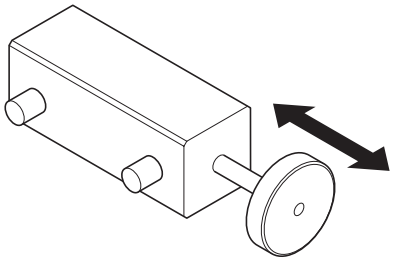


エアシリンダ（シングルソレノイドタイプ）で設定するパラメータは次のとおりです。

設定項目	説明	設定値	初期値
エアシリンダ: 対応変数	エアシリンダのエア注入、注入停止を行う I/O の変数を設定します。	BOOL 型のグローバル変数	-
仮想出力（押出位置検出）: 対応変数	ピストンが完全に押し出されたときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
仮想出力（引込位置検出）: 対応変数	ピストンが完全に引き込んだときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
エアシリンダ: シリンダの移動時間	円柱の移動時間を設定します。（単位: s）	0.0～100.0	0.5
エアシリンダ: シリンダの長さ	対応変数の ON/OFF が変化したときの移動量を設定します。（単位: mm）	0～35,000,000,000,000	100
エアシリンダ: シリンダの型	エアシリンダの作動方式を設定します。	通電時押出型／通電時引込型	通電時押出型

エアシリンダ（ダブルソレノイドタイプ）

エアシリンダ（ダブルソレノイドタイプ）は二つの入力値により、ピストン部分が移動する機構です。

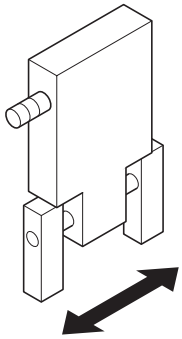


エアシリンダ（ダブルソレノイドタイプ）で設定するパラメータは次のとおりです。

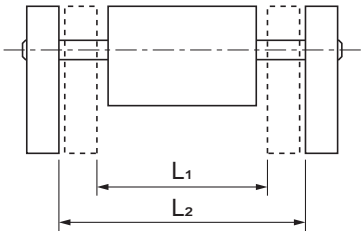
設定項目	説明	設定値	初期値
押出方向の入力: 対応変数	エアシリンダ引き込み時にピストンを押し出す変数を設定します。	BOOL 型のグローバル変数	-
引込方向の入力: 対応変数	エアシリンダ押し出し時にピストンを引き込む変数を設定します。	BOOL 型のグローバル変数	-
仮想出力（押出位置検出）: 対応変数	ピストンが完全に押し出されたときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
仮想出力（引込位置検出）: 対応変数	ピストンが完全に引き込んだときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
エアシリンダ: シリンダの移動時間	円柱の移動時間を設定します。（単位: s）	0.0～100.0	0.5
エアシリンダ: シリンダの長さ	対応変数の ON/OFF が変化したときの移動量を設定します。（単位: mm）	0～35,000,000,000,000	100
エアシリンダ: ピストンの初期状態	エアシリンダのピストンの初期状態を設定します。	戻／出	戻

ロボットツール（平行開閉型 2 爪タイプチャック／シングルソレノイドタイプ）

ロボットツール（平行開閉型 2 爪タイプチャック／シングルソレノイドタイプ）は、ロボットの先端にロボットがワークを把持するためなどに付けるチャックを、エア注入することで動作させる機構です。

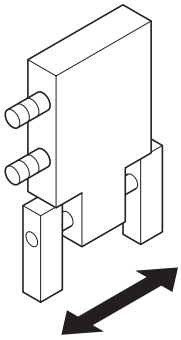


ロボットツール（平行開閉型 2 爪タイプチャック／シングルソレノイドタイプ）で設定するパラメータは、次のとおりです。

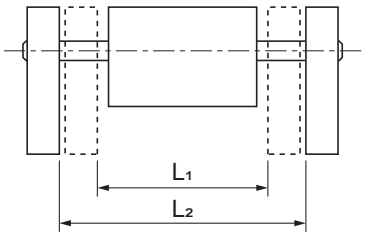
設定項目	説明	設定値	初期値
チャック: 対応変数	チャックのエア注入、注入停止を行う I/O の変数を設定します。	BOOL 型のグローバル変数	-
仮想出力(開位置検出) : 対応変数	チャックが完全に開いたときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
仮想出力(閉位置検出) : 対応変数	チャックが完全に閉じたときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
チャック: ストローク時間	チャックが完全に開閉するまでの時間を設定します。(単位: s)	0.0～100.0	0.5
チャック: 開閉ストローク幅	チャックの開閉ストローク幅を設定します。(単位: mm) 下図 L2-L1 の値を設定します。 	0～100	10
チャック: 動作	チャックの作動方式を設定します。	通電時開／通電時閉	通電時閉

ロボットツール（平行開閉型 2 爪タイプチャック／ダブルソレノイドタイプ）

ロボットツール（平行開閉型 2 爪タイプチャック／ダブルソレノイドタイプ）は、ロボットの先端にロボットがワークを把持するためなどに付けるチャックをエア注入し、チャックを開閉させる機構です。

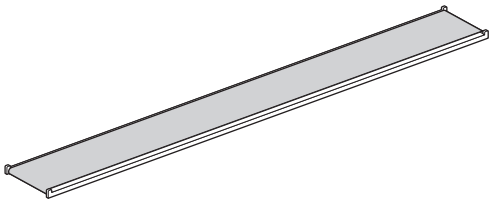


ロボットツール（平行開閉型 2 爪タイプチャック／ダブルソレノイドタイプ）で設定するパラメータは、次のとおりです。

設定項目	説明	設定値	初期値
開方向の入力: 対応変数	チャックの開方向に対応するエア注入、注入停止を行う I/O の変数を設定します。	BOOL 型のグローバル変数	-
閉方向の入力: 対応変数	チャックの閉方向に対応するエア注入、注入停止を行う I/O の変数を設定します。	BOOL 型のグローバル変数	-
仮想出力（開位置検出）: 対応変数	チャックが完全に開いたときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
仮想出力（閉位置検出）: 対応変数	チャックが完全に閉じたときに ON となる I/O 変数を設定します。	BOOL 型のグローバル変数	-
チャック: ストローク時間	チャックが完全に開閉するまでの時間を設定します。（単位: s）	0.0～100.0	0.5
チャック: 開閉ストローク幅	チャックの開閉ストローク幅を設定します。（単位: mm） 下図 L2-L1 の値を設定します。 	0～100	10
チャック: 初期状態	チャックの初期状態を設定します。	開／閉	開

コンベア

コンベアはモーション軸と関連付けることにより、ワークを運搬することができる機構です。



バージョン関連情報

コンベアは、アプリケーションマネージャ Ver.5.0 以降でサポートしています。
コンベアで設定するパラメータは次のとおりです。

設定項目	説明	設定値	初期値
コンベア: 対応変数	コンベアに対応する軸を設定します。	_sAXIS_REF 型のグローバル変数	-
コンベア: 長さ	コンベアの長さを設定します。(単位: mm)	100～1000000	1800
コンベア: 幅	コンベアの幅を設定します。(単位: mm)	100～1000000	100

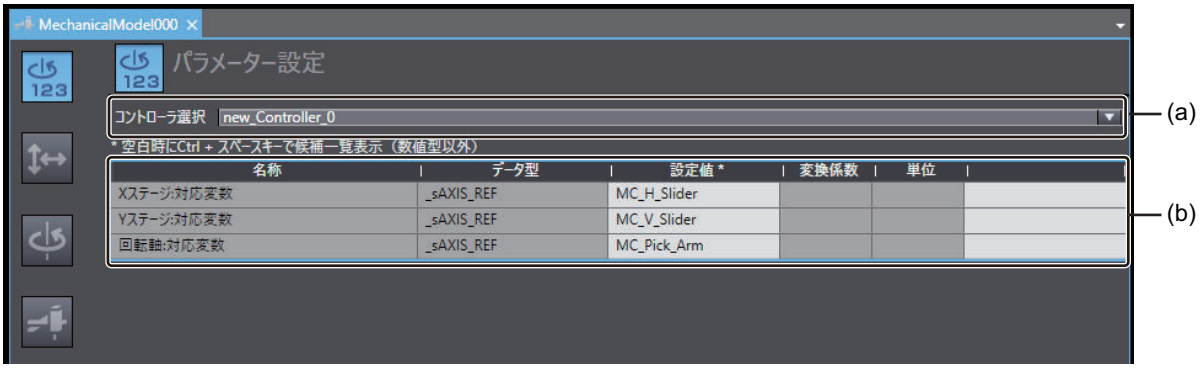
4-3-5 メカニカル機構の設定項目

追加したメカニカル機構をダブルクリックすると、メカニカル機構の設定画面が表示されます。
メカニカル機構には次の設定項目があります。

- パラメーター設定
- 直動方向設定
- 回転軸設定
- メカニカル機構共通設定

パラメーター設定

メカニカル機構を制御するコントローラと、軸変数を割り付けます。

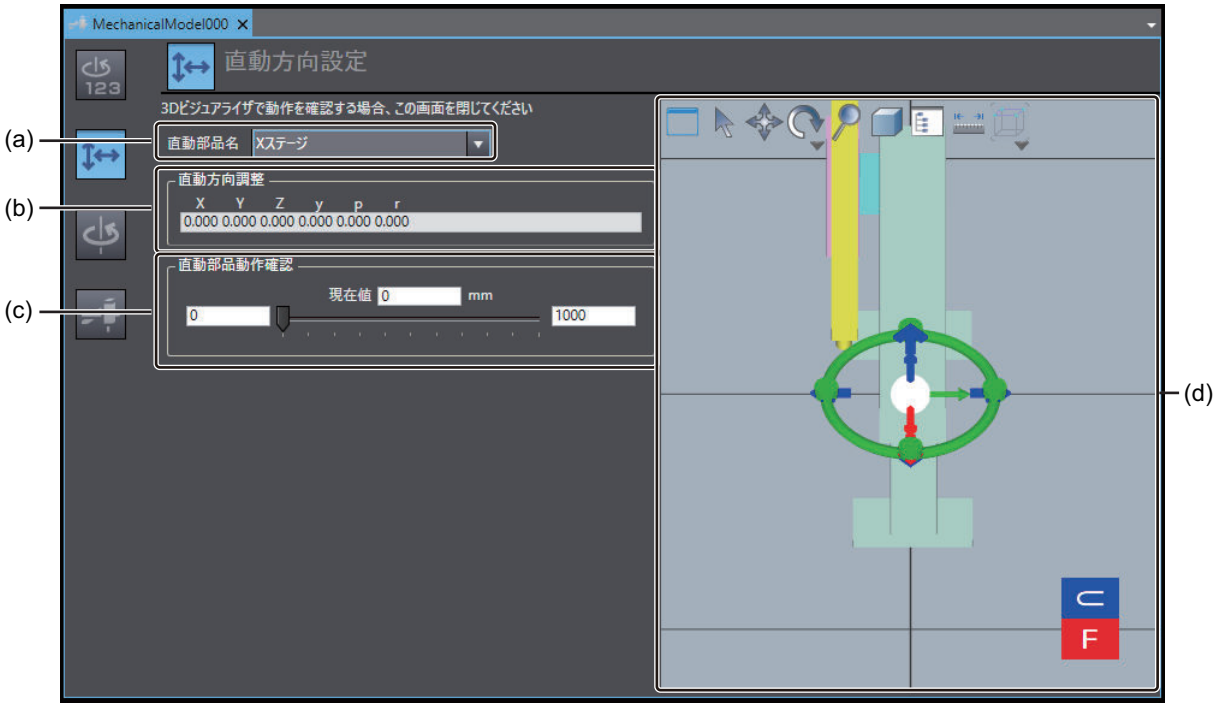


	設定項目	説明	設定値	初期値*1
(a)	コントローラ選択	メカニカル機構を制御するコントローラを選択します。	コントローラ名	選択なし
(b)	軸変数の設定	メカニカル機構の各可動部品を制御する軸変数を割り付けます。	選択したコントローラの軸変数	なし

*1. ［メカニカル機構追加ウィザード］のステップ 4 で設定した値が表示されます。

直動方向設定

メカニカル機構の可動部品のうち、直動動作をする部品の位置、姿勢、動作範囲を設定します。

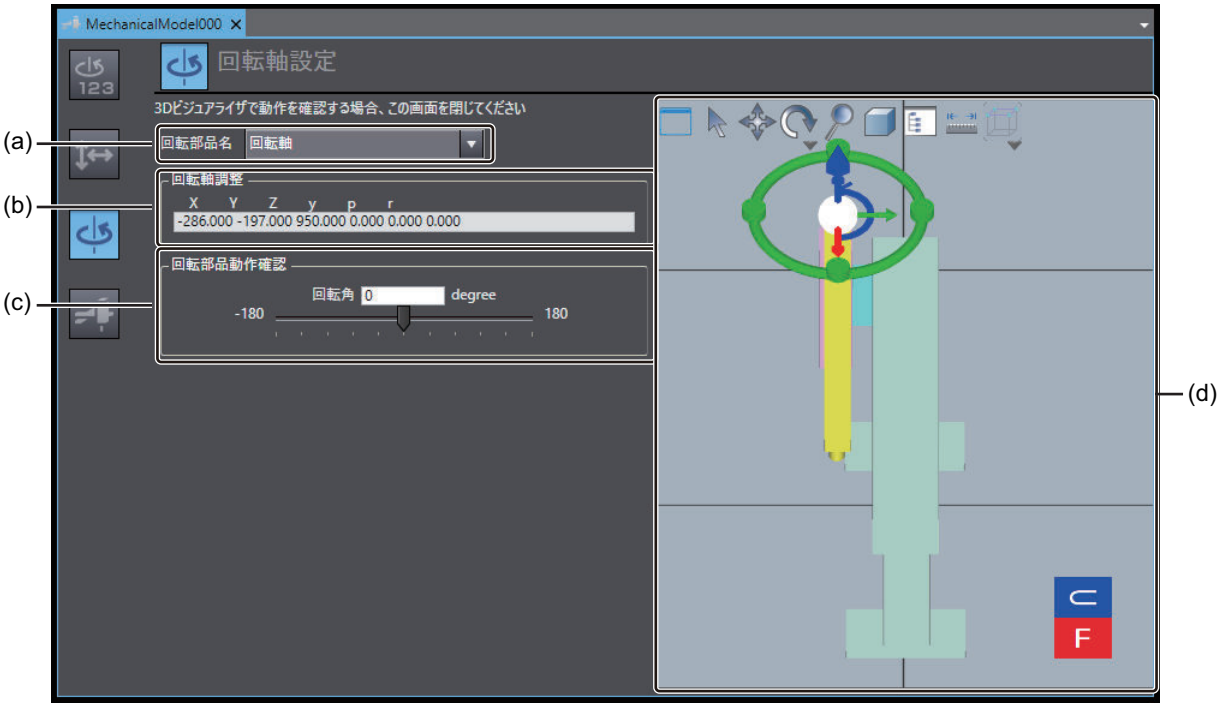


	設定項目*1	説明	設定値	初期値*2
(a)	直動部品名	メカニカル機構の可動部品のうち、直動動作をする部品を選択します。	各可動部品	-
(b)	直動方向調整	直動部品の動作の開始位置と、直動部品の姿勢を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.000
(c)	直動部品動作確認	スライダを操作して、直動部品の動作を確認します。	-10,000,000,000,000～ +10,000,000,000,000	最小値: 0 最大値: 1000
(d)	3D 編集エリア	直動部品の位置や姿勢を設定します。また、［直動部品動作確認］での操作に応じて直動部品が動作します。	-	-

*1. 直動動作をする部品がないメカニカル機構では、各項目の値は表示されません。
*2. ［メカニカル機構追加ウィザード］のステップ 2 で設定した値が表示されます。

回転方向設定

メカニカル機構の可動部品のうち、回転動作をする部品の位置、姿勢、動作範囲を設定します。

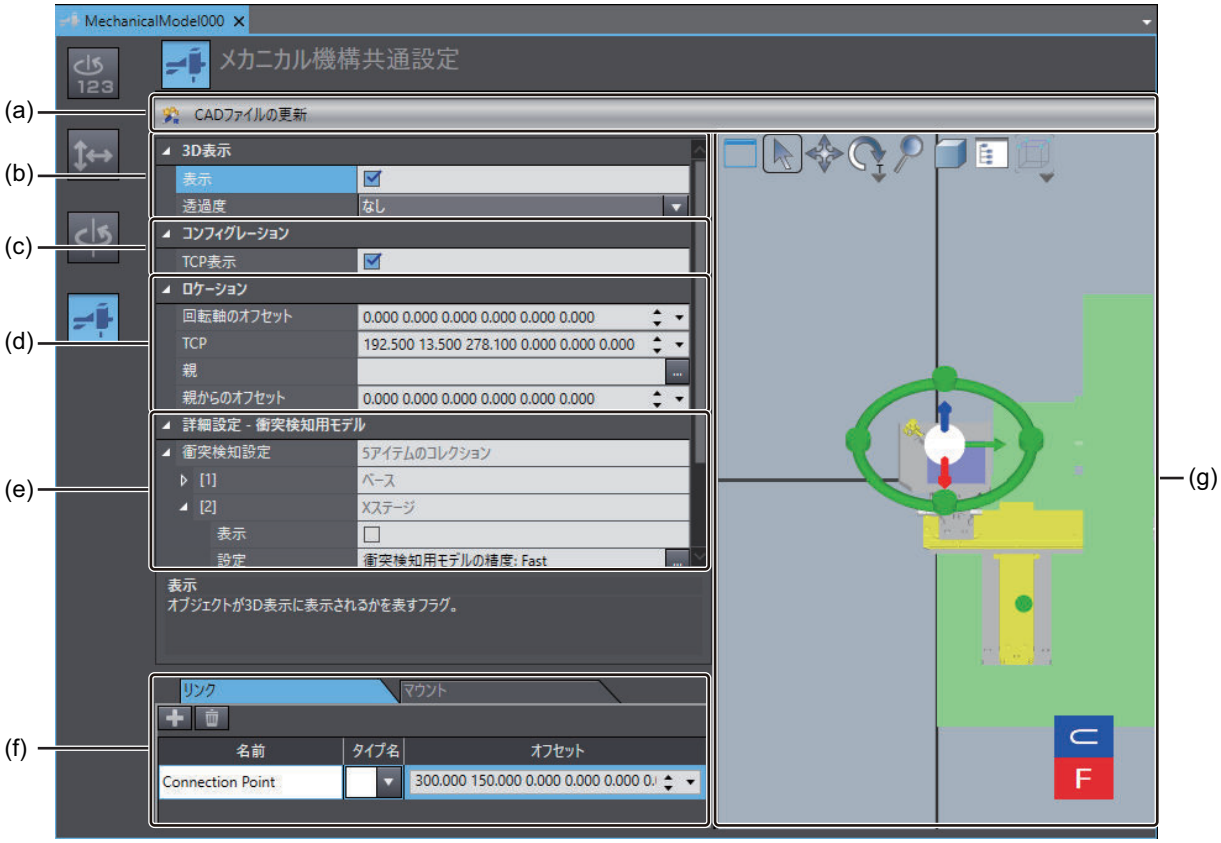


	設定項目*1	説明	設定値	初期値*2
(a)	回転部品名	メカニカル機構の可動部品のうち、回転動作をする部品を選択します。	各可動部品	-
(b)	回転方向調整	回転部品の動作の開始位置と、回転部品の姿勢を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	それぞれ -1,000,000.000～+1,000,000.000	すべて 0.000
(c)	回転部品動作確認	スライダを操作して、回転部品の動作を確認します。	-	-
(d)	3D 編集エリア	回転部品の位置や姿勢を設定します。また、[回転部品動作確認] での操作に応じて回転部品が動作します。	-	-

*1. 回転動作をする部品がないメカニカル機構では、各項目の値は表示されません。
*2. [メカニカル機構追加ウィザード] のステップ 3 で設定した値が表示されます。

メカニカル機構共通設定

3D 表示画面での表示や配置、マウントポイント／リンクポイントを設定します。
メカニカル機構の 3D 形状データの原点は、インポートした CAD データファイルに定義されている原点の位置に設定されます。



	設定項目		説明	設定値	初期値
(a)	CAD ファイルの更新		メカニカル機構部品の CAD データファイルを更新します。	-	-
(b)	3D 表示	表示	メカニカル機構の 3D 形状データを 3D 表示画面に表示する／しないを切り替えます。チェック ON で 3D 表示画面に表示されます。	チェック ON／チェック OFF	チェック ON
		透過度	3D 形状データの透過度を設定します。	なし／低／中／高	なし
(c)	コンフィグレーション	TCP 表示	TCP を 3D 編集エリアに表示する／しないを切り替えます。チェック ON で表示されます。	チェック ON／チェック OFF	チェック ON
(d)	ロケーション	回転軸のオフセット	この 3D 形状データのローカル座標系の原点に対する回転軸のオフセットを設定します。オフセットを元に、回転軸の位置が変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.000
		TCP	メカニカル機構の 3D 形状データ上にロボットハンドなどのツールを取り付ける位置を示す点の座標です。設定する座標は、本メカニカル機構の 3D 形状データのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.000
		親	メカニカル機構の 3D 形状データの親となる 3D 形状データを設定します。	3D 形状データの名称	選択なし

	設定項目		説明	設定値	初期値
		親からのオフセット	メカニカル機構の 3D 形状データの位置と姿勢を設定します。親が設定されていない場合、設定する座標は 3D 表示画面のワールド座標系における座標です。親が設定されている場合、設定する座標は親の 3D 形状データのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(e)	詳細設定 - 衝突検知モデル	表示	衝突検知用モデルを表示するかどうかを設定します。	チェック ON／チェック OFF	チェック OFF
		設定	衝突検知用モデルの設定を行います。右側のボタンから、衝突検知用モデルの詳細な設定を行います。設定方法の詳細は「衝突検知用モデルの詳細設定 (P.8-25)」を参照してください。	-	衝突検知用モデルの簡略化: True | 分割して衝突検知用モデルを生成: False
(f)	マウントポイント、リンクポイント設定エリア		メカニカル機構の 3D 形状データのマウントポイント、リンクポイントを設定するエリアです。マウントポイント、リンクポイントの設定方法の詳細については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。	-	-
(g)	3D 編集エリア		メカニカル機構の 3D 形状データの位置や姿勢、マウントポイント、リンクポイントを表示する画面です。3D 編集エリアでの表示や操作方法の詳細は、「第 5 章 3D 表示画面、3D 編集エリアの操作 (P.5-1)」を参照してください。	-	-

4-4 カスタムメカニカル機構の 3D 形状データの作成

カスタムメカニカル機構は、任意の可動部品の動作を設定できる機構です。
直動、回転、直動／回転の複合動作をする可動部品を組み合わせて、Sysmac Studio であらかじめ用意されているメカニカル機構ではサポートしていない機構を実現することができます。

4-4-1 カスタムメカニカル機構で実現できる機構

カスタムメカニカル機構では、直動、回転、直動／回転の複合動作をする可動部品の組み合わせにより、次の機構を実現することができます。

機構	説明	組み合わせる動作
電動シリンダ	電動シリンダは、ひとつの入力値により、ピストン部分が電動で移動する機構です。	直動
電動チャック（2 爪タイプ、3 爪タイプ）	電動チャックは、ワークを把持するためにロボットの先端に取り付けたチャックを、電動で動作させる機構です。	直動
エアチャック（2 爪タイプ、3 爪タイプ）	エアチャックは、ワークを把持するためにロボットの先端に取り付けたチャックを、エアを注入することで動作させる機構です。	直動
直動／回転複合アクチュエータ	ひとつの入力値により直動動作と回転動作を行う機構です。入力があると、まず直動動作を行い、直動動作が終わると回転動作を行います。	直動／回転

使用上の注意

次の機構は実現できません。
- ガントリ
- H-Bot
- T-Bot
- メカカム、電子カム
- ギア
- Screw Joint で構成される機構

4-4-2 カスタムメカニカル機構の 3D 形状データの作成の流れ

カスタムメカニカル機構の 3D 形状データは、次の流れで作成します。

	項目	内容	参照先	
1	カスタムメカニカル機構の追加	アプリケーションマネージャにカスタムメカニカル機構を追加します。	「4-4-4 カスタムメカニカル機構の追加手順 (P.4-33)」	
↓				
2	可動部品の追加／CAD データのインポート	カスタムメカニカル機構を構成する可動部品の CAD データをインポートして、3D 形状データを作成します。可動部品は複数個登録できます。 インポートする CAD データの種類により、操作の手順が異なります。	USDPhysicsJoint を含む OpenUSD ファイルの場合	「4-4-6 USDPhysicsJoint を含む OpenUSD ファイルのインポート (P.4-40)」
			上記以外の場合	「4-4-5 可動部品の追加／CAD データのインポート (P.4-36)」
↓				
3	TCP の設定	カスタムメカニカル機構の先端となる可動部品を選択し、TCP を設定します。	「4-4-7 TCP の設定 (P.4-45)」	
↓				
4	結合部の動作設定	可動部品どうしの結合部の接続方法を設定します。「固定ジョイント」、「ヒンジジョイント」、「ボールジョイント」、「スライダージョイント」、「回転ジョイント」、「親子としての設定」の 6 種類から選択します。	「4-4-8 結合部の動作設定 (P.4-47)」	

	項目	内容	参照先
5	可動部品の動作方向設定	可動部品の動作方向を設定します。「直動」、「回転」、「直動・回転」の 3 種類から選択します。	「4-4-9 可動部品の動作方向設定 (P.4-55)」
6	カスタムメカニカル機構の動作確認	カスタムメカニカル機構の可動部品を試運転し、可動部品および結合部の設定が正しいことを確認します。	「4-4-10 カスタムメカニカル機構の動作確認 (P.4-60)」
7	カスタムメカニカル機構の動作設定	3D シミュレーション上でのカスタムメカニカル機構の動作を定義します。	「4-4-11 カスタムメカニカル機構の動作設定 (P.4-62)」

4-4-3 サポートする OpenUSD ファイルについて

Sysmac Studio でサポートする OpenUSD ファイルについて説明します。CAD データについては「4-3-1 サポートする CAD データファイルの種類 (P.4-9)」を参照してください。

● サポートする OpenUSD ファイルのフォーマット

Sysmac Studio で読み込み可能な OpenUSD ファイルのフォーマットは以下のとおりです。

ファイルフォーマット	内容
USD	ASCII ベースのテキスト形式またはバイナリ形式
USDA	ASCII ベースのテキスト形式
USDC	バイナリ形式
USDZ	アーカイブ形式

● サポートする OpenUSD ファイルのバージョン

Sysmac Studio で読み込み可能な OpenUSD ファイルの対応バージョンは以下のとおりです。

	対応バージョン
OpenUSD	OpenUSD 23.11 SDK をサポートしている OpenUSD

● サポートする USDPhysicsJoint

Sysmac Studio で読み込み可能な USDPhysicsJoint は以下のとおりです。

結合情報	種別
USDPhysicsJoint	Prismatic
	Revolute
	Fixed

参考

- OpenUSD の USDPhysicsJoint の body0 と body1 が Prim Path での親子関係になるものを Sysmac Studio ではサポートしています。
  定義の例:

```
def PhysicsPrismaticJoint "SpecialJoint" {
  rel physics:body0 = </World/Base>
  rel physics:body1 = </World/Base/X>
```

- ある USDPhysicsJoint の body1 が body0 の親となるような場合、不正な情報として判定され、その USDPhysicsJoint はインポートされません。
  定義の例:

```
def PhysicsPrismaticJoint "InvalidJoint" {
  rel physics:body0 = </World/Base/X>
  rel physics:body1 = </World/Base>
```

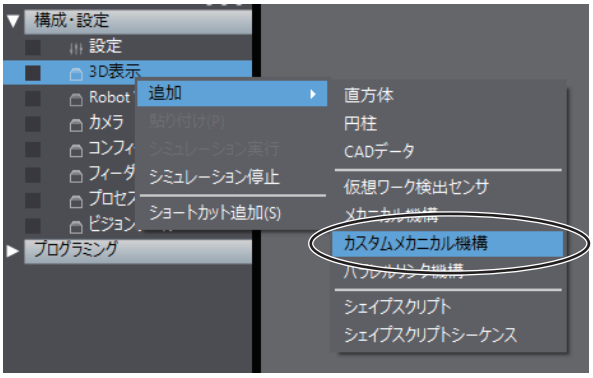
- ある USDPhysicsJoint の body1 が指定する Prim を、同様に body1 として指定する USDPhysicsJoint が別に存在する場合、不正な OpenUSD ファイルとして判定し、インポート処理を中断します。

4-4-4 カスタムメカニカル機構の追加手順

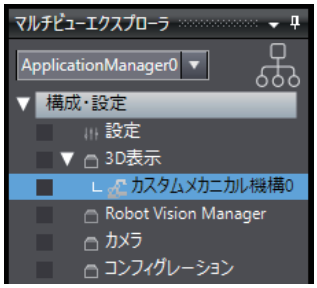
アプリケーションマネージャにカスタムメカニカル機構を追加します。カスタムメカニカル機構の追加手順は以下のとおりです。

操作手順

1 **［構成・設定］の［3D 表示］を右クリックし、メニューから［追加］-［カスタムメカニカル機構］を選択します。**

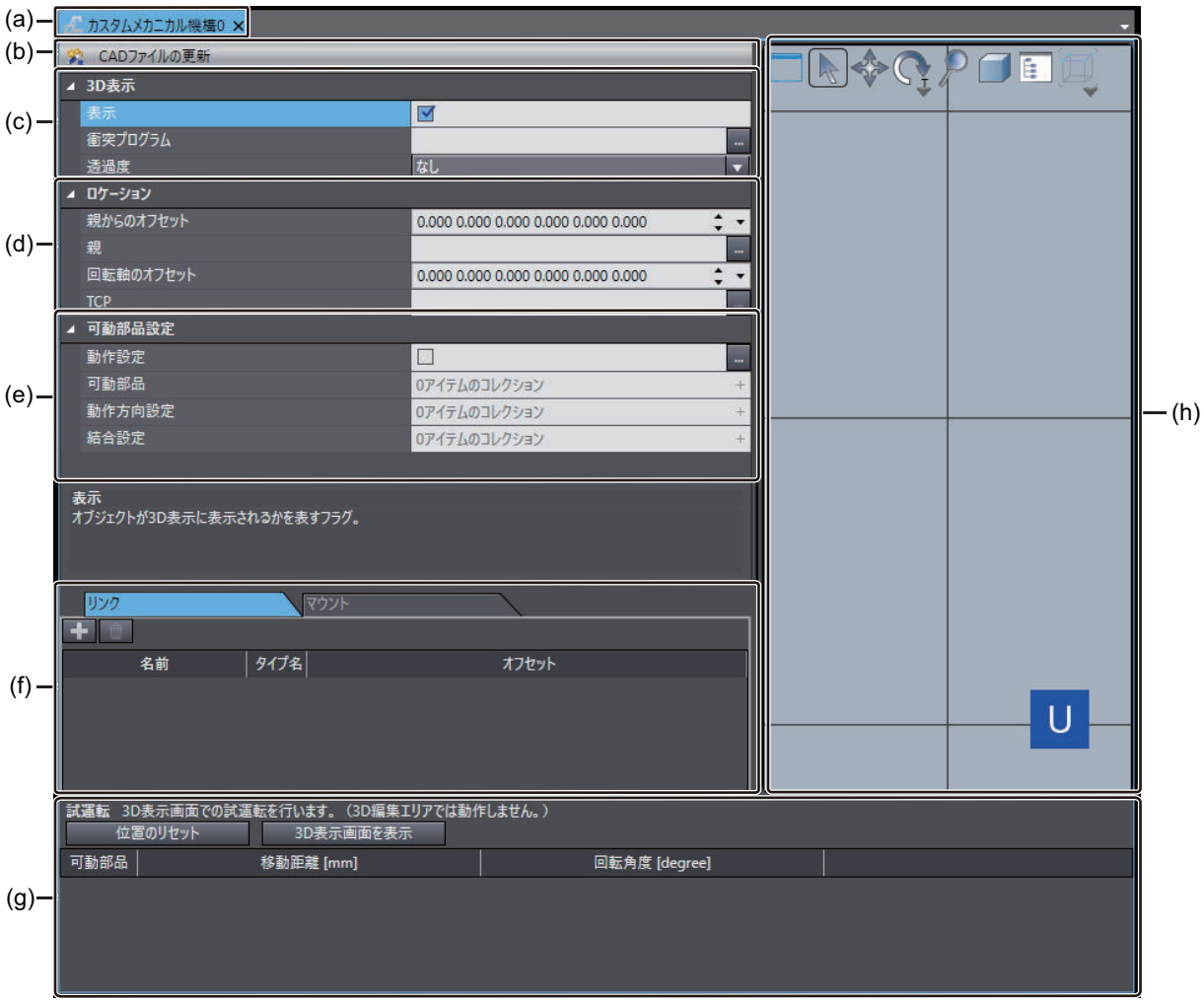


［3D 表示］の下に選択した機構が登録されます。



カスタムメカニカル機構の設定項目

追加したカスタムメカニカル機構をダブルクリックすると、カスタムメカニカル機構の設定画面が表示されます。カスタムメカニカル機構の共通の設定項目、および機構種別ごとの固有の設定項目について説明します。



	設定項目	説明	設定値	初期値
(a)	カスタムメカニカル機構名	マルチビューエクスプローラに表示されたカスタムメカニカル機構名が表示されます。	-	-

	設定項目		説明	設定値	初期値
(b)	CAD ファイルの更新		3D 形状データとして使用する CAD ファイルをインポートします。詳細は「4-4-5 可動部品の追加／CAD データのインポート (P.4-36)」を参照してください。	-	-
(c)	3D 表示	表示	カスタムメカニカル機構の 3D 形状データを 3D 表示画面に表示する／しないを切り替えます。チェック ON で 3D 表示画面に表示されます。	チェック ON／チェック OFF	チェック ON
		衝突プログラム	3D 形状データ衝突を検出したときに実行する C#プログラムを設定します。右側のボタンから、C#プログラムを選択します。C#プログラムに衝突した時刻と衝突した対象などを出力するプログラムを記述することで、衝突したときの情報を取得することができます。	C#プログラム名	なし
		透過度	3D 形状データの透過度を設定します。	なし／低／中／高	なし
(d)	ロケーション	親からのオフセット	カスタムメカニカル機構の 3D 形状データの位置と姿勢を設定します。親が設定されていない場合、設定する座標は 3D 表示画面のワールド座標系における座標です。親が設定されている場合、設定する座標は親の 3D 形状データのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
		親	カスタムメカニカル機構の 3D 形状データの親となる 3D 形状データを設定します。	3D 形状データの名称	選択なし
		回転軸のオフセット	3D 形状データのローカル座標系の原点に対する回転軸のオフセットを設定します。オフセットを元に、回転軸の位置が変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
		TCP	カスタムメカニカル機構の TCP となる可動部品とその位置を設定します。設定する位置は選択した可動部品のローカル座標系における座標です。設定方法の詳細は「4-4-7 TCP の設定 (P.4-45)」を参照してください。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	-
(e)	可動部品設定	動作設定	可動部品の動作設定を行います。設定方法の詳細は「4-4-11 カスタムメカニカル機構の動作設定 (P.4-62)」を参照してください。	-	-
		可動部品	可動部品の設定を行います。設定方法の詳細は「4-4-5 可動部品の追加／CAD データのインポート (P.4-36)」を参照してください。	-	-
		動作方向設定	可動部品の動作方向設定を行います。設定方法の詳細は「4-4-9 可動部品の動作方向設定 (P.4-55)」を参照してください。	-	-

	設定項目		説明	設定値	初期値
		結合設定	可動部品どうしの結合部の接続方法を設定します。詳細は「4-4-8 結合部の動作設定 (P.4-47)」を参照してください。	-	-
(f)	マウントポイント、リンクポイント設定エリア		カスタムメカニカル機構の 3D 形状データのマウントポイント、リンクポイントを設定するエリアです。マウントポイント、リンクポイントの設定方法の詳細については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。	-	-
(g)	可動部品の試運転		カスタムメカニカル機構の動作部品を動作させ、可動部品および結合部の動作設定を確認します。詳細は「4-4-10 カスタムメカニカル機構の動作確認 (P.4-60)」を参照してください。	-	-
(h)	3D 編集エリア		カスタムメカニカル機構の 3D 形状データの位置や姿勢、マウントポイント、リンクポイントを表示する画面です。3D 編集エリアでの表示や操作方法の詳細は、「第 5 章 3D 表示画面、3D 編集エリアの操作 (P.5-1)」を参照してください。	-	-

4-4-5 可動部品の追加／CAD データのインポート

カスタムメカニカル機構に可動部品を追加し、可動部品の CAD データをインポートして、3D 形状データを作成します。可動部品は複数個登録できます。

ここでは USDPhysicsJoint を含まない OpenUSD 形式の CAD データを読み込む例を説明します。

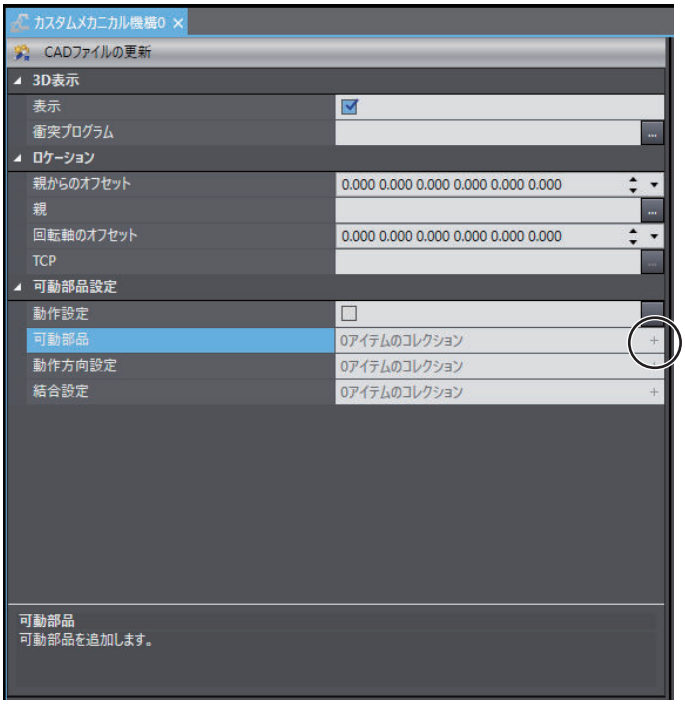
操作手順

可動部品の追加手順は以下のとおりです。

参考

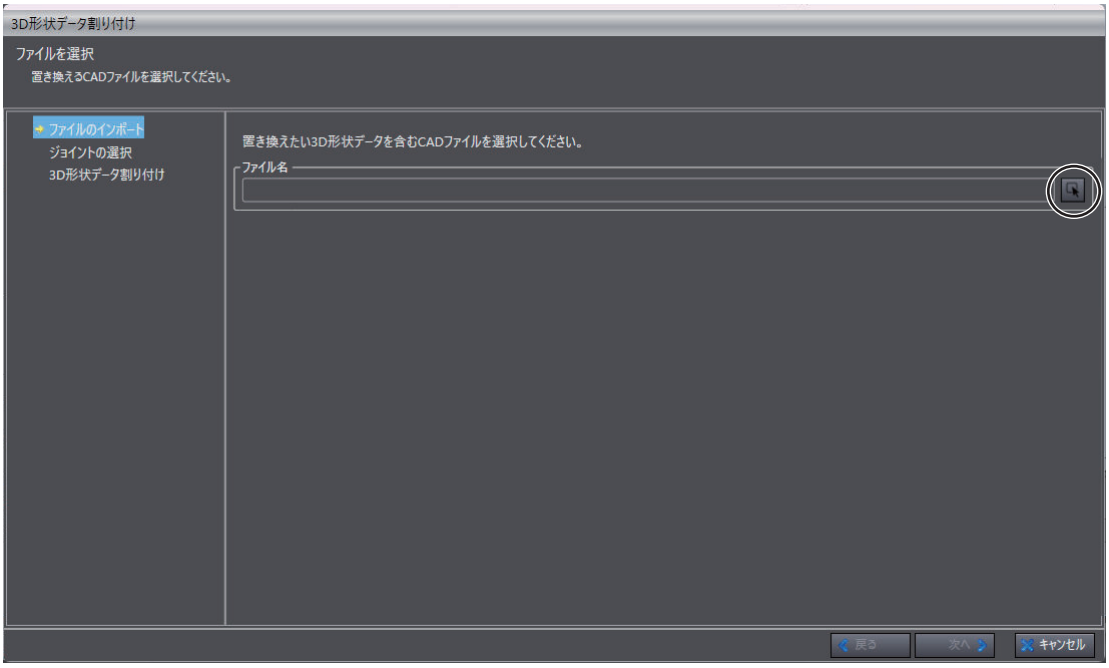
インポート時 CAD データのメッシュの粗さをオプションで設定することができます。オプション設定の詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

1 **対象のカスタムメカニカル機構の設定画面で、[可動部品] の右側の追加ボタンをクリックします。**



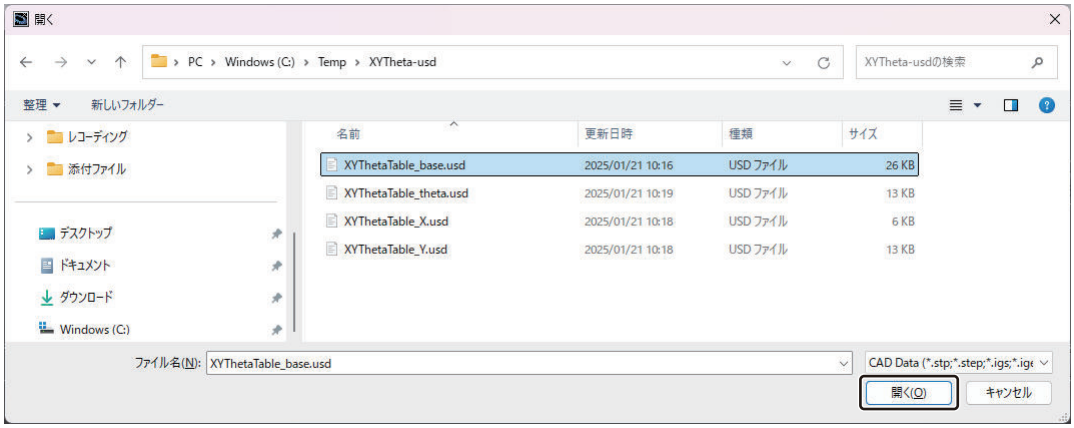
ウィザードが起動し、［3D 形状データ割り付け］画面が表示されます。

2 **［ファイルを選択］の画面で、［ファイル名］の右側のボタンをクリックします。**

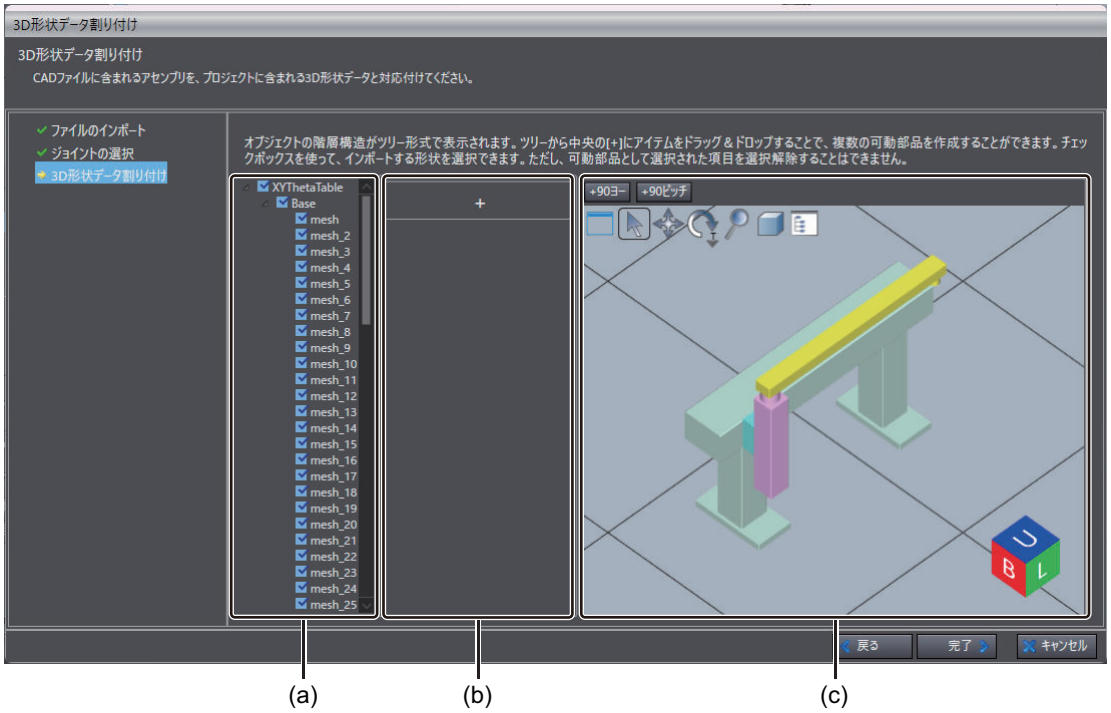


［開く］ダイアログボックスが表示されます。

3 **［開く］ダイアログボックスでインポートする STEP、IGES または OpenUSD 形式の CAD データファイルを選択し、［開く］ボタンをクリックします。**

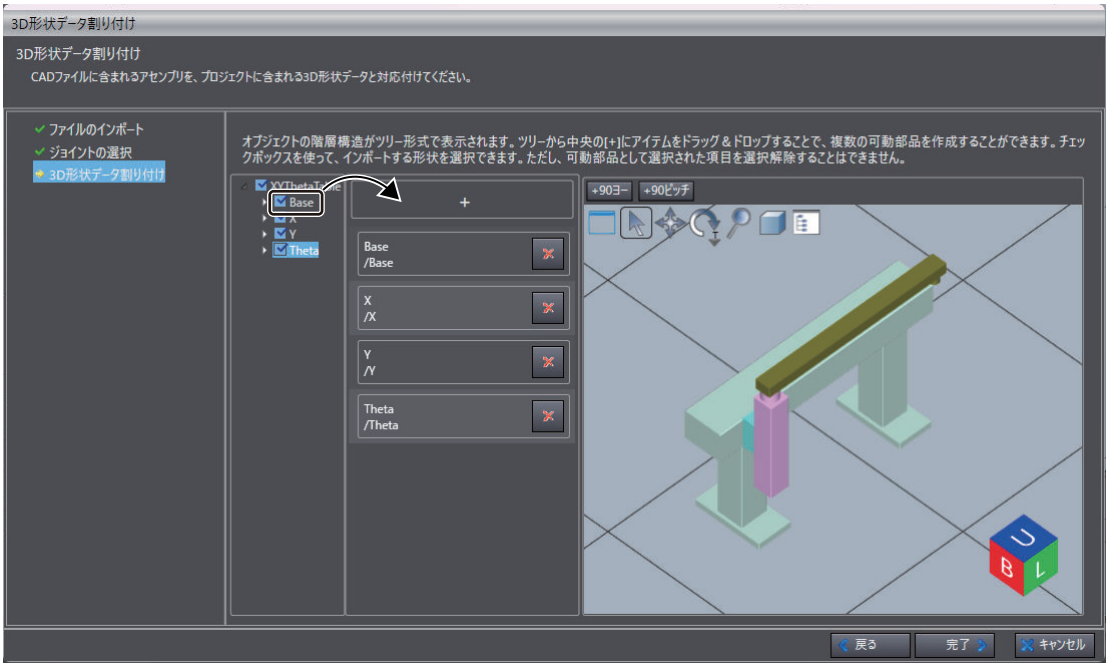


［ファイルを選択］の画面で［次へ］ボタンをクリックすると、［3D 形状データ割り付け］画面が表示されます。



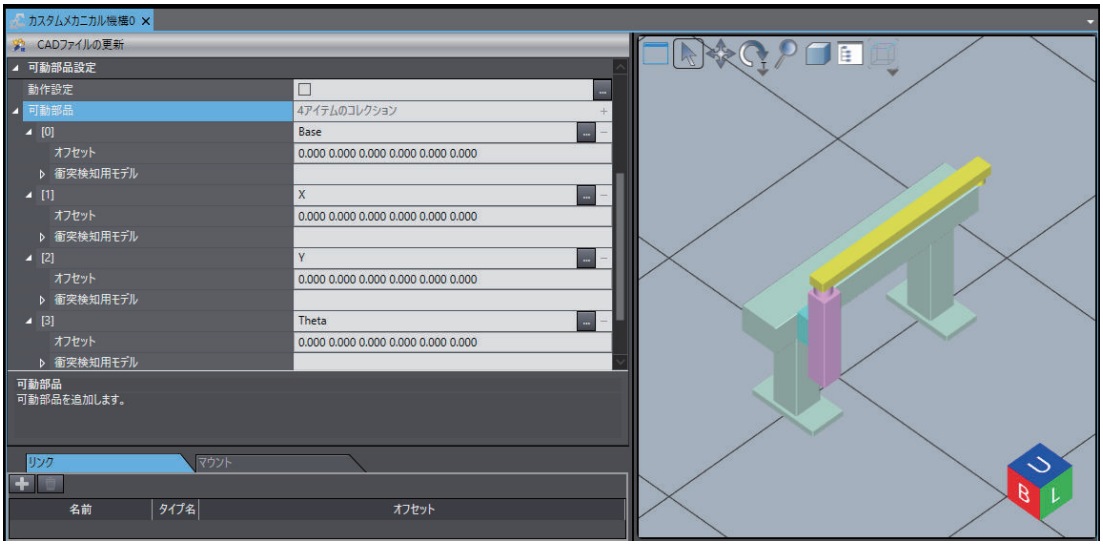
	項目	説明
(a)	インポートした OpenUSD の Prim	インポートした OpenUSD の Prim が階層表示されます。 チェックを外すことで 3D 形状データをインポート対象外とすることができます。
(b)	可動部品の一覧	(a)の各要素をドラッグし、［+］の部分にドロップすると、可動部品が作成されます。同じ要素から複数の可動部品を作成することはできません。 ［×］ボタンをクリックすることで作成した可動部品を削除することができます。
(c)	インポートした CAD データの 3D 表示画面	インポートした CAD データを 3D 表示します。

4 **可動部品として設定する要素をドラッグし［+］の部分にドロップします。**



可動部品として設定する分だけドラッグ＆ドロップを繰り返します。

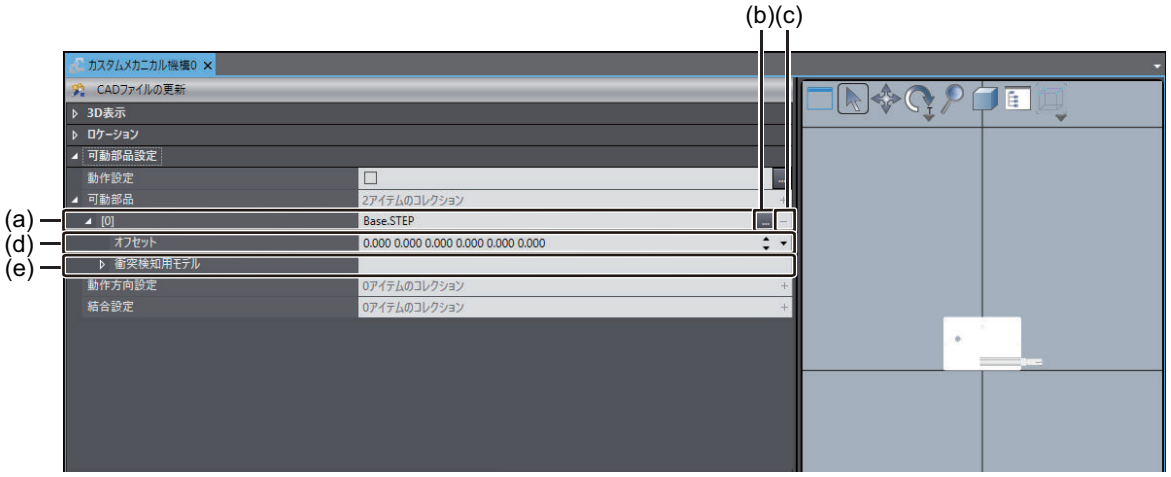
5 **可動部品への割り付けが完了したら、［完了］ボタンをクリックします。**



可動部品として設定した CAD データの項目名が追加されます。また、インポートした CAD ファイルのデータが 3D 編集エリアに表示されます。

可動部品の設定項目

可動部品の設定項目は以下のとおりです。



	設定項目	説明	設定値	初期値
(a)	CAD ファイル名	インポートした CAD ファイル名が表示されます。	-	インポートした CAD ファイル名
(b)	CAD ファイルインポートボタン	可動部品の CAD ファイルを置き換えることができます。ボタンをクリックすると、[ファイル選択] ダイアログボックスが表示されます。	-	-
(c)	可動部品削除ボタン	このボタンをクリックすると、追加した可動部品が削除されます。	-	-
(d)	オフセット	可動部品の位置を指定します。カスタムメカニカル機構の原点からの相対座標を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(e)	衝突検知用モデル	衝突検知用モデルに関する設定を行います。詳細は、「8-3-5 衝突検知用モデルの設定 (P.8-24)」を参照してください。	-	-

4-4-6 USDPhysicsJoint を含む OpenUSD ファイルのインポート

USDPhysicsJoint を含む OpenUSD ファイルをインポートします。USDPhysicsJoint を含むため、カスタムメカニカル機構の 3D 形状データ作成時の可動部品割り付け、動作方向設定にかかる工数を削減し、シミュレーション環境構築時間を短縮することができます。

インポート手順

インポート手順は以下のとおりです。

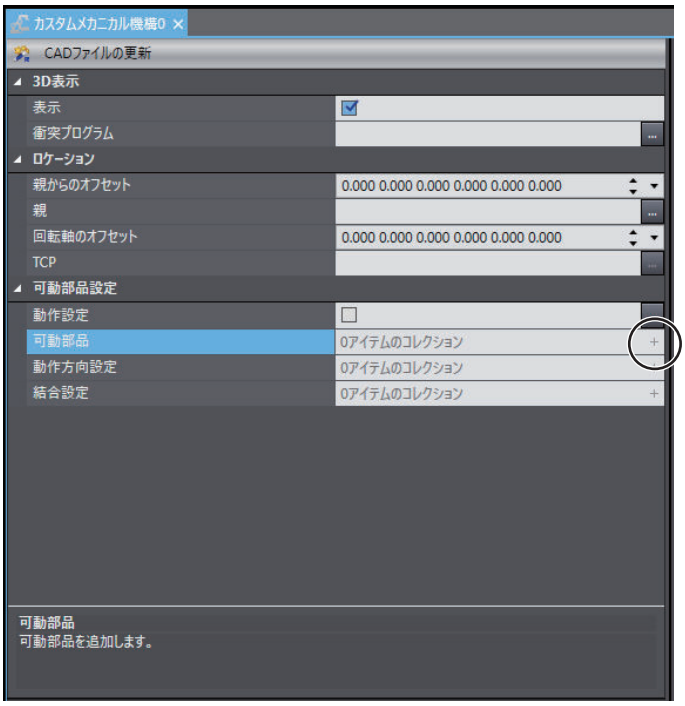
参考

インポート時 CAD データのメッシュの粗さをオプションで設定することができます。オプション設定の詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

1 カスタムメカニカル機構を追加します。

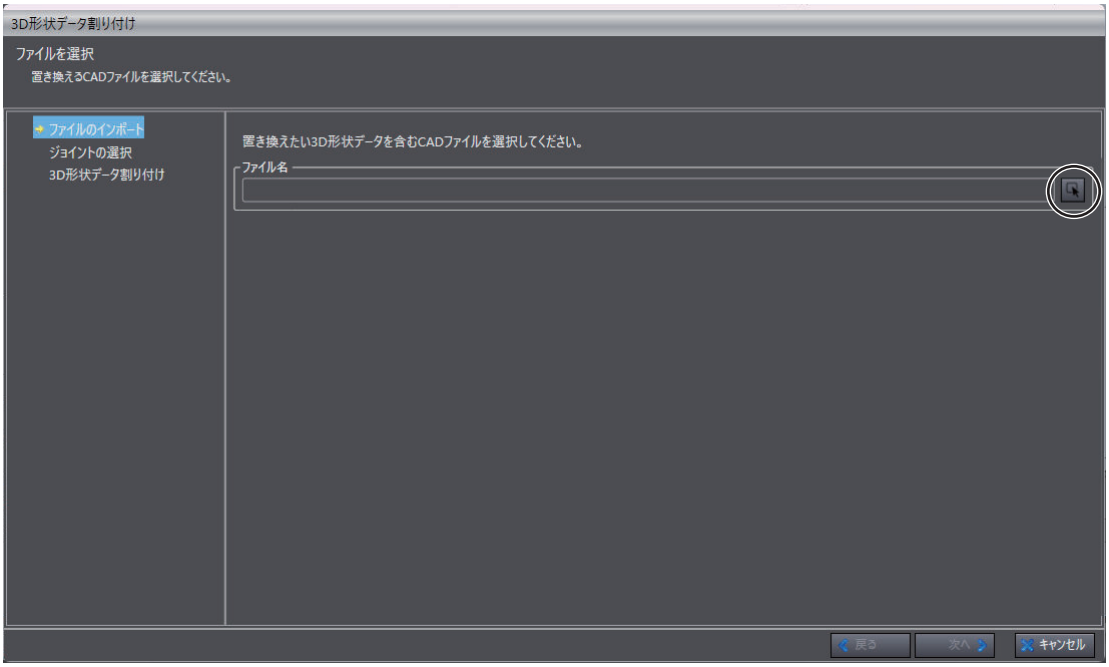
カスタムメカニカル機構の追加手順については「4-4-4 カスタムメカニカル機構の追加手順 (P.4-33)」を参照してください。

2 **対象のカスタムメカニカル機構の設定画面で、[可動部品] の右側の追加ボタンをクリックします。**



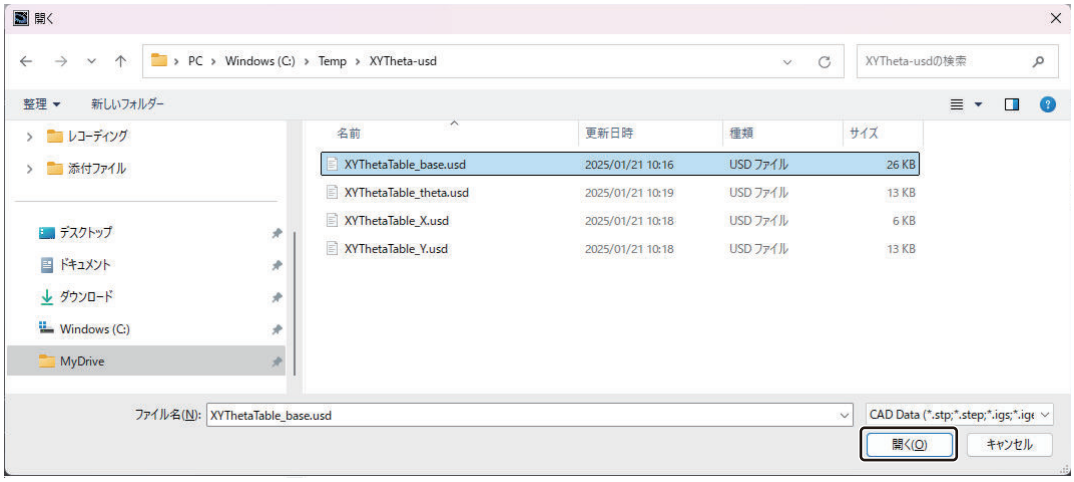
ウィザードが起動し、[3D 形状データ割り付け] 画面が表示されます。

3 **[ファイルを選択] の画面で、[ファイル名] の右側のボタンをクリックします。**

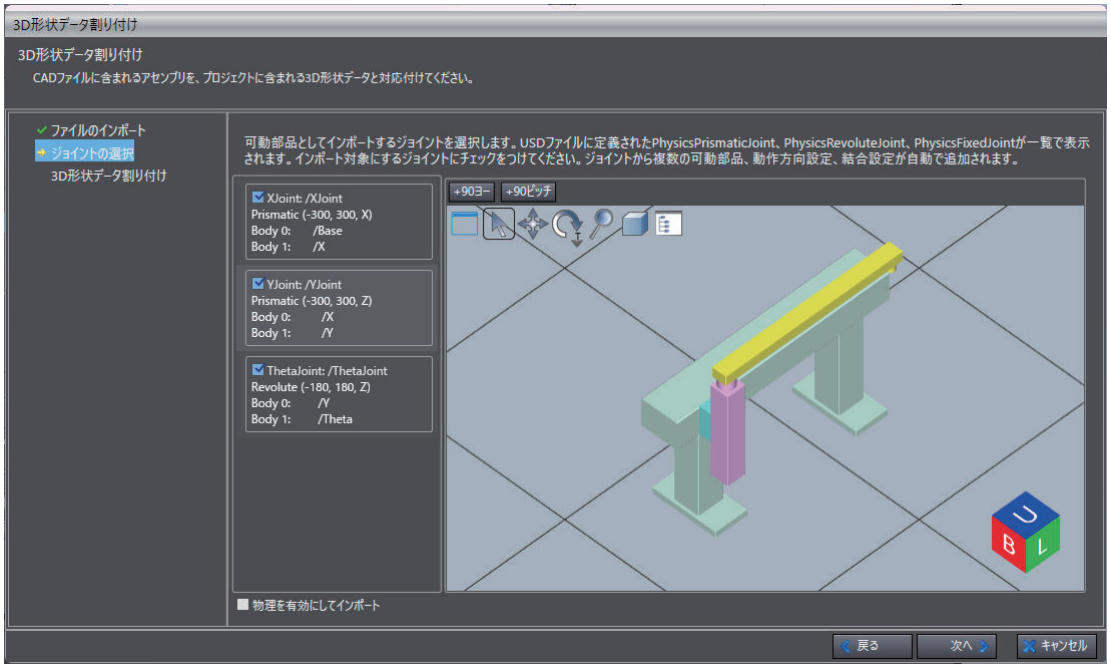


[開く] ダイアログボックスが表示されます。

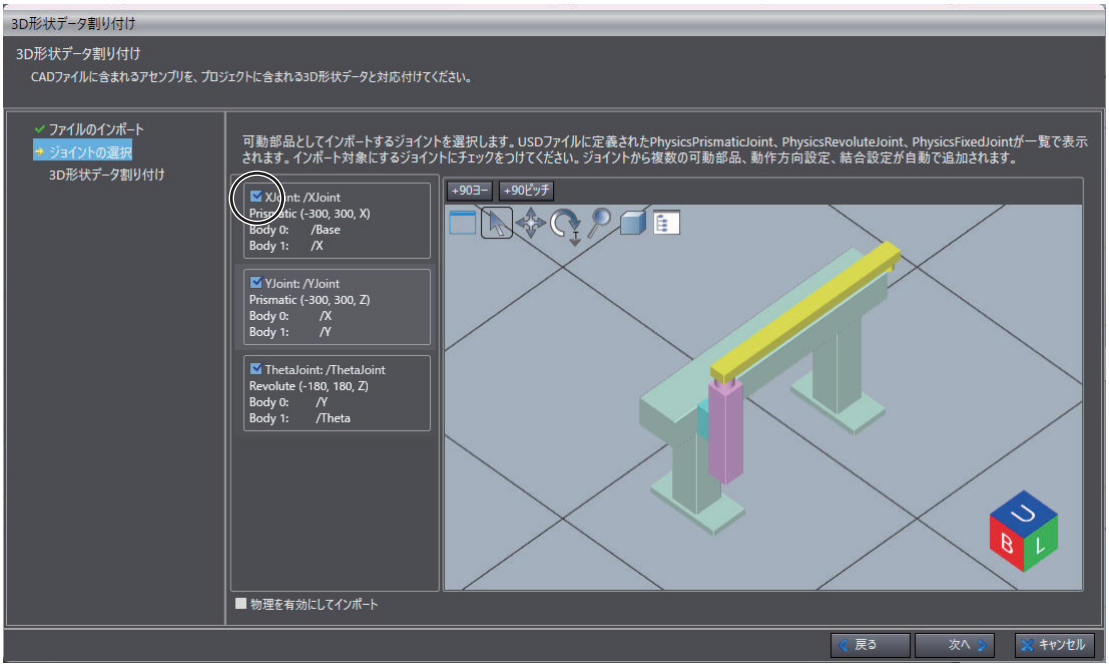
4 **[開く] ダイアログボックスでインポートする USD、USDA、USDC または USDZ のフォーマット形式の OpenUSD ファイルを選択し、[開く] ボタンをクリックします。**



［ファイルを選択］の画面で［次へ］ボタンをクリックすると、［ジョイントの選択］画面が表示されます。



5 **［ジョイントの選択］画面でインポートする USDPhysicsJoint にチェックを入れます。**

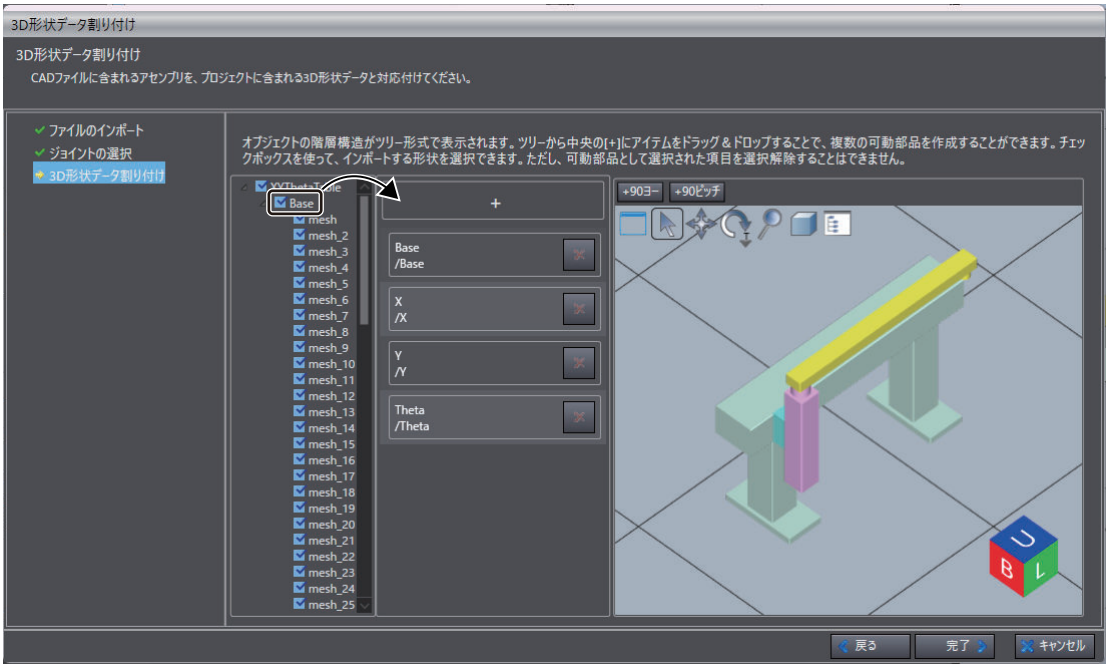


6 **必要に応じて［物理を有効にしてインポート］チェックボックスにチェックを入れ、［次へ］ボタンをクリックします。**



［物理を有効にしてインポート］についての詳細は後述の「［物理を有効にしてインポート］(P.4-44)」を参照してください。

7 **［3D 形状データ割り付け］画面で USDPhysicsJoint を元に作成された複数の可動部品が表示されます。**

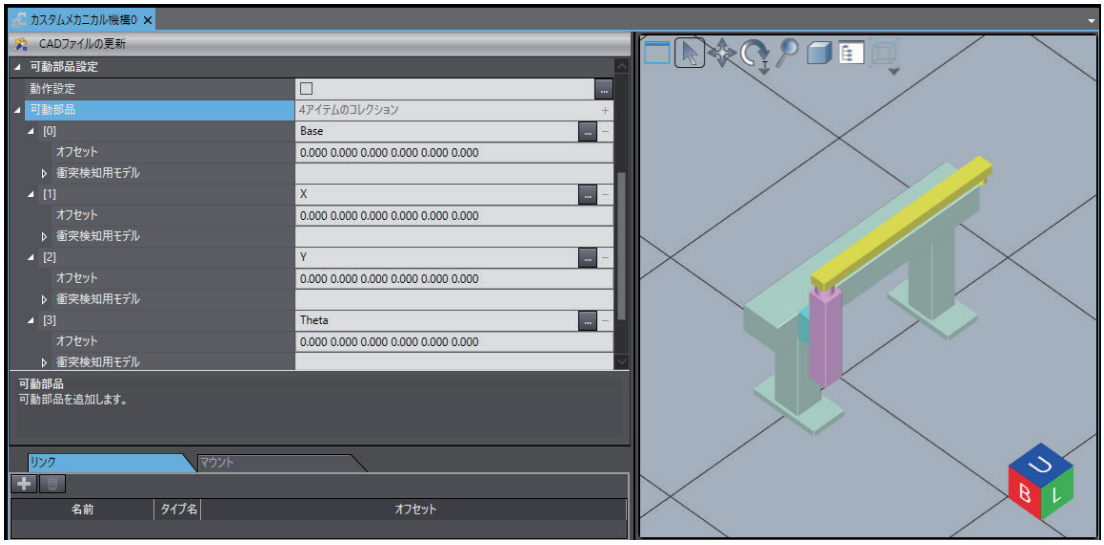


可動部品をさらに追加したい場合は、可動部品として設定する要素をドラッグし、［+］の部分にドロップします。可動部品として設定する分だけドラッグ＆ドロップを繰り返したあと、［完了］ボタンをクリックします。

参考

- USD Prim のツリーに表示される要素のレベルはオプションの設定によって決まります。また、レベルの設定によって［ジョイントの選択］画面がスキップされ、USDPhysicsJoint がインポートされない場合があります。オプション設定については『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編 (SBCA-464)』を参照してください。
- ［3D 形状データ割り付け］画面項目の詳細については「4-4-5 可動部品の追加／CAD データのインポート (P.4-36)」を参照してください。

8 **対象のカスタムメカニカル機構の設定画面に可動部品が追加されているので、各可動部品について動作設定を行います。**



動作設定については「4-4-11 カスタムメカニカル機構の動作設定 (P.4-62)」を参照してください。

［物理を有効にしてインポート］

［物理を有効にしてインポート］は USDPhysicsJoint をインポートする際、USDPhysicsJoint の種別に応じて結合設定を行うか、親子として結合設定を行うかを選択する機能です。
［物理を有効にしてインポート］を有効にした場合、USDPhysicsJoint の種別に応じて結合設定を行うため、各可動部が軸により制御されず、他の可動部等の動作と連動した動きを再現する物理シミュレーションを実行することができます。
［物理を有効にしてインポート］を無効にした場合、親子として結合設定を行うため、各可動部が軸により制御され、物理によるシミュレーションが不要な場合に適しています。

- ［物理を有効にしてインポート］を有効にした場合
  USDPhysicsJoint をインポートする際、body0 で指定される可動部品を「結合対象 1」、body1 で指定される可動部品を「結合対象 2」として、OpenUSD の USDPhysicsJoint 種別に対応して以下の結合設定を行います。

インポートする OpenUSD の USDPhysicsJoint	Sysmac Studio での結合設定
UsdPhysicsPrismaticJoint	スライダージョイント

インポートする OpenUSD の USDPhysicsJoint	Sysmac Studio での結合設定
UsdPhysicsRevoluteJoint	回転ジョイント
UsdPhysicsFixedJoint	固定ジョイント

「結合対象 2」に設定した可動部品の物理設定を有効とします。USDPhysicsJoint に記述された以下の軸方向と軸位置の定義から自動で結合位置の設定を行います。
a）physics:axis
b）physics:localPos0
c）physics:localRot0
d）physics:localPos1
e）physics:localRot1

- ［物理を有効にしてインポート］を無効にした場合
  USDPhysicsJoint をインポートする際、body0 で指定される可動部品を「親」、body1 で指定される可動部品を「子」とする親子として、結合設定を行います。

  USDPhysicsJoint により生成された可動部品が、OpenUSD 上の階層構造で親子関係を持つ場合は、親子として設定する結合設定を行います。ただし、すべての可動部品間で親子関係を設定する必要はなく、最も階層が近い可動部品間のみに設定されます。また、すでに USDPhysicsJoint により結合設定がされている場合は無視されます。

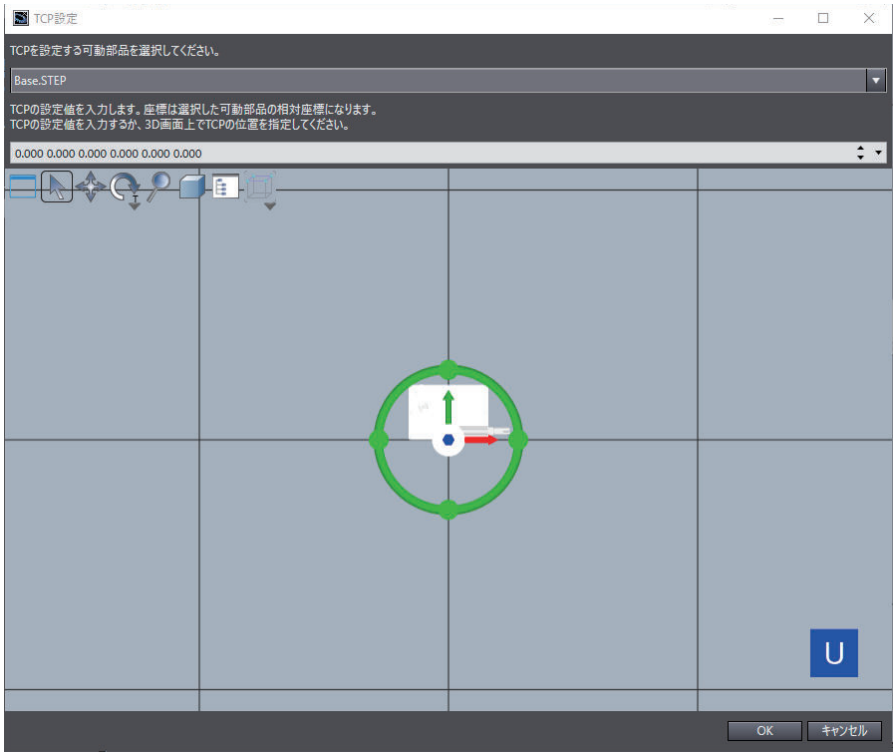
4-4-7　TCP の設定

カスタムメカニカル機構の先端となる可動部品を選択し、TCP を設定します。

設定手順

TCP の設定手順は以下のとおりです。

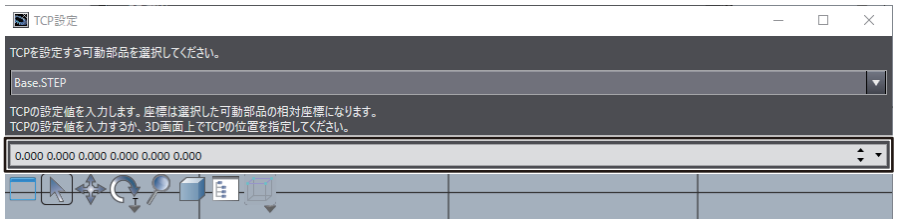
1 **カスタムメカニカル機構の設定画面で、［TCP］の右側のボタンをクリックします。**
［TCP 設定］ダイアログボックスが表示されます。



2 **カスタムメカニカル機構の先端となる可動部品を選択します。**



3 **TCP の位置を設定します。**

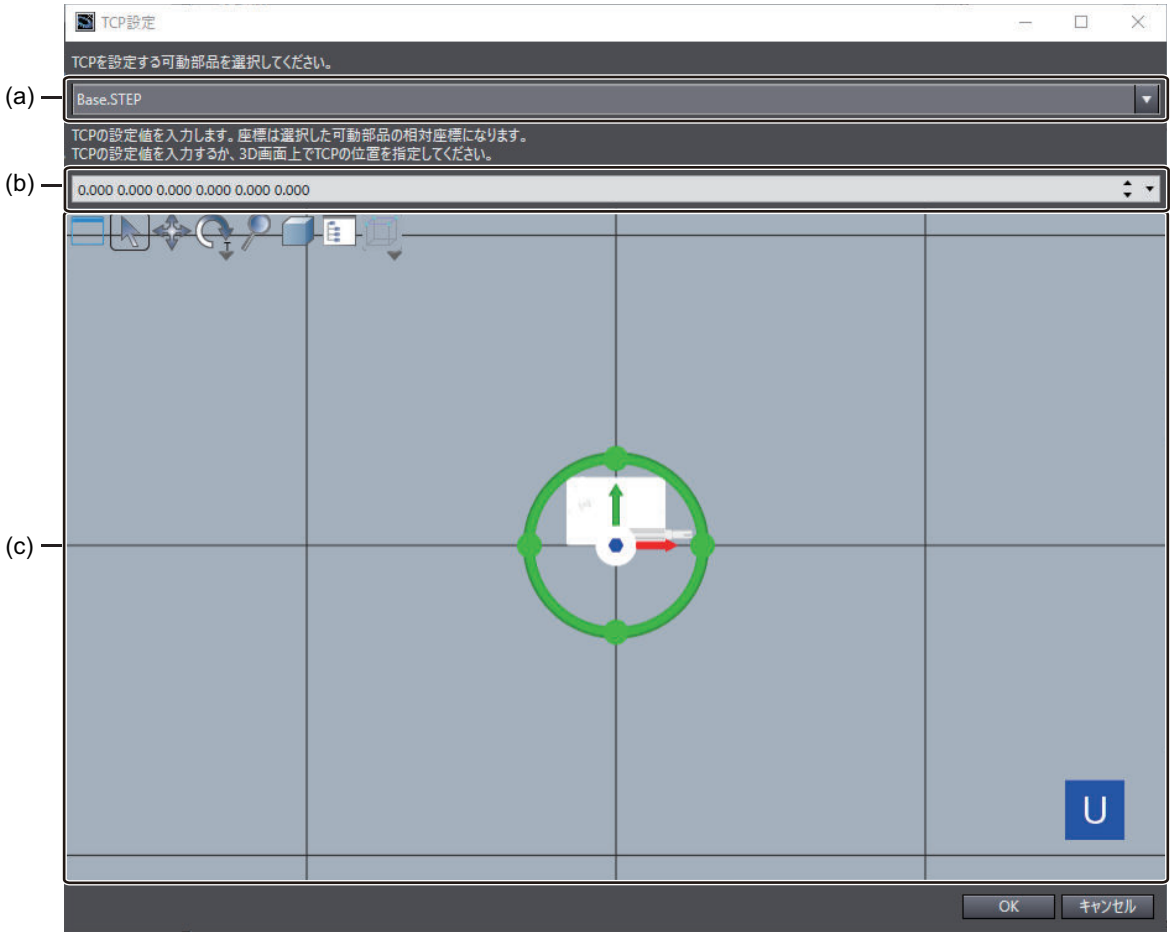


4 **[OK] ボタンをクリックします。**

カスタムメカニカル機構の TCP が保存されます。

TCP の設定項目の詳細については「TCP 設定の設定項目 (P.4-47)」を参照してください。

TCP 設定の設定項目



	設定項目	説明	設定値	初期値
(a)	可動部品選択	TCP を設定する可動部品を選択します。	-	-
(b)	TCP 設定値	TCP の位置を設定します。位置は可動部品選択コンボボックスで選択した可動部品からの相対座標です。設定値を変更すると、3D 編集エリアの TCP の位置が設定値に合わせて反映されます。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(c)	3D 編集エリア	カスタムメカニカル機構と TCP の位置が表示されます。マウス操作をして TCP の位置や向きを変更することができます。変更すると、TCP 設定値の値に反映されます。	-	-

4-4-8　結合部の動作設定

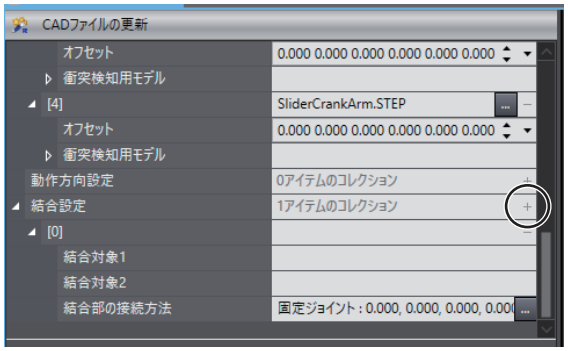
可動部品どうしの結合部の接続方法を設定します。
結合部の接続は、可動部品の物理オプションを設定したときに有効になります。
「固定ジョイント」、「ヒンジジョイント」、「ボールジョイント」、「スライダージョイント」、「回転ジョイント」、「親子として設定」の 6 種類から選択します。

接続方法	説明
固定ジョイント	2 つの可動部品を固定する接続方法です。固定された状態とは、片方の可動部品の位置および姿勢が変更されたときに、他方の可動部品が追従します。
ヒンジジョイント	2 つの可動部品をヒンジジョイント（X 軸を基準に固定角度間で回転）で接続する方法です。
ボールジョイント	2 つの可動部品をボールジョイント（X 軸を固定して Y、Z 軸を基準に回転）で接続する方法です。
スライダージョイント	2 つの可動部品を直線方向（X 軸を基準に直動）に接続する方法です。
回転ジョイント	2 つの可動部品を、X 軸を基準に回転する接続方法です。
親子として設定	2 つの可動部品の親子関係による接続方法です。子に設定された可動部品は、親に設定された可動部品に追従します。

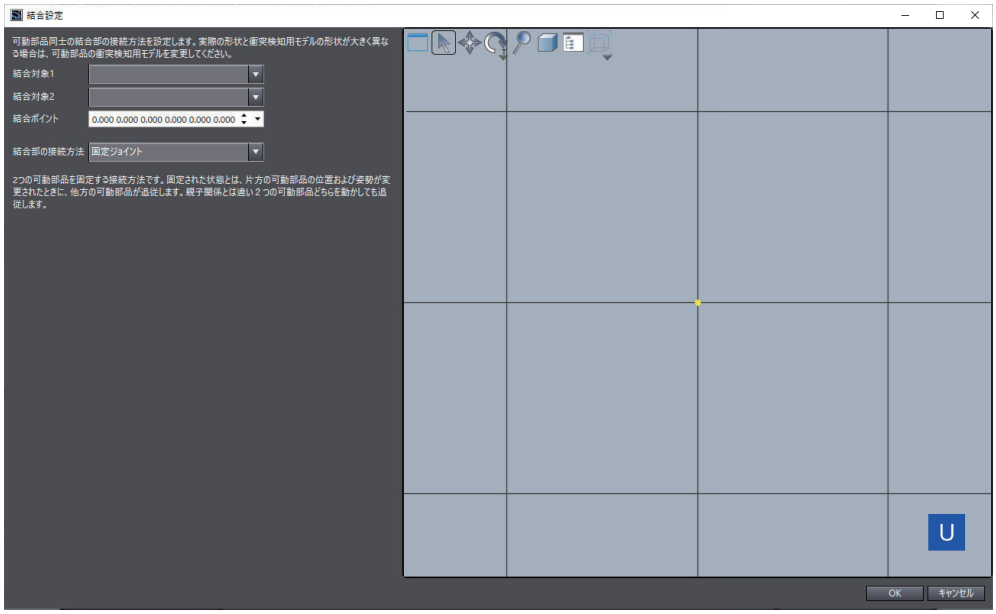
設定手順

結合部の接続設定手順は以下のとおりです。

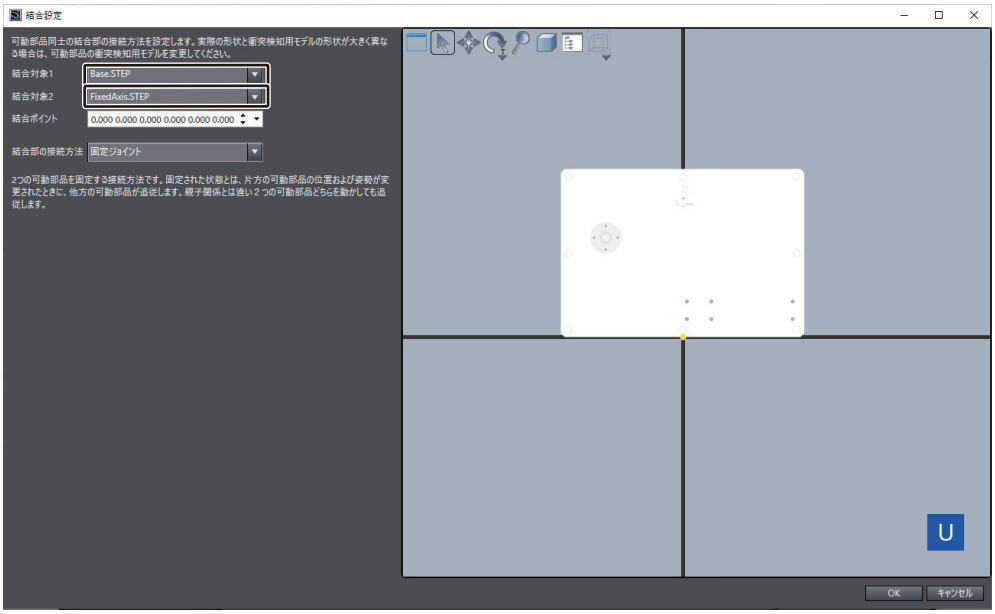
1 **カスタムメカニカル機構の設定画面で、[結合設定] の右側のボタンをクリックします。**



2 **[結合設定] ダイアログボックスが表示されます。**



3 **[結合対象 1]、[結合対象 2] で、結合する 2 つの可動部品を選択します。**



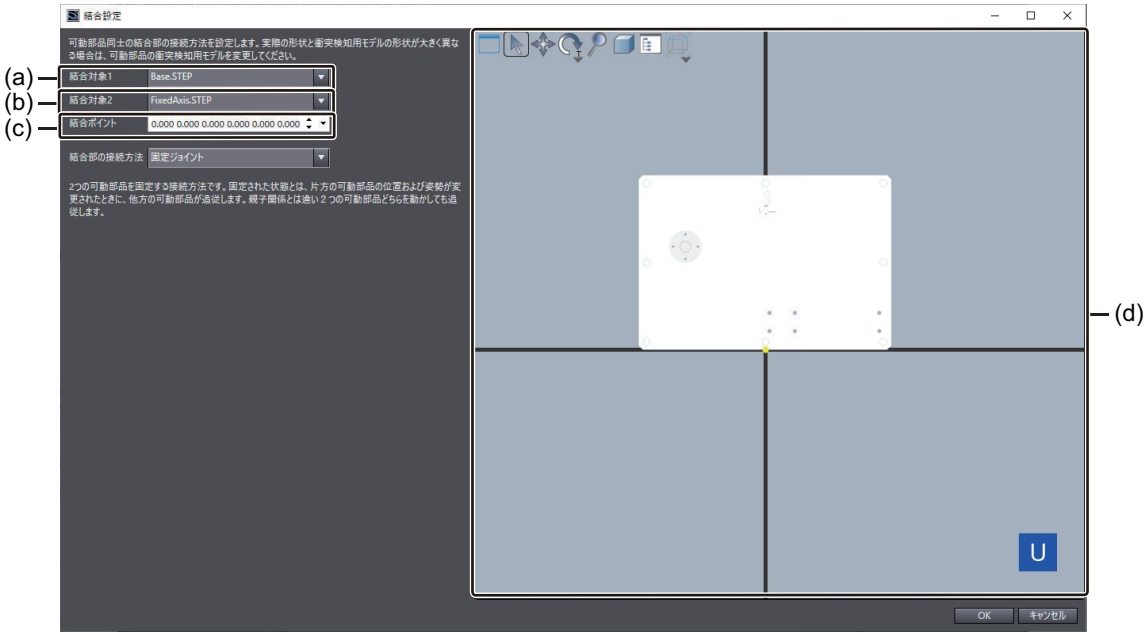
4 **［結合部の接続方法］で「固定ジョイント」、「ヒンジジョイント」、「ボールジョイント」、「スライダージョイント」、「回転ジョイント」、「親子として設定」のいずれかを選択します。**
選択した接続方法の設定項目が［結合部の接続方法］の項目の下に表示されます。表示される内容は選択した接続方法によって異なります。

5 **結合部の接続方法の詳細設定を行い、［OK］ボタンをクリックします。**
結合部の接続方法の設定が保存されます。
各接続方法の設定項目の詳細については「結合部の設定項目 (P.4-49)」を参照してください。

結合部の設定項目

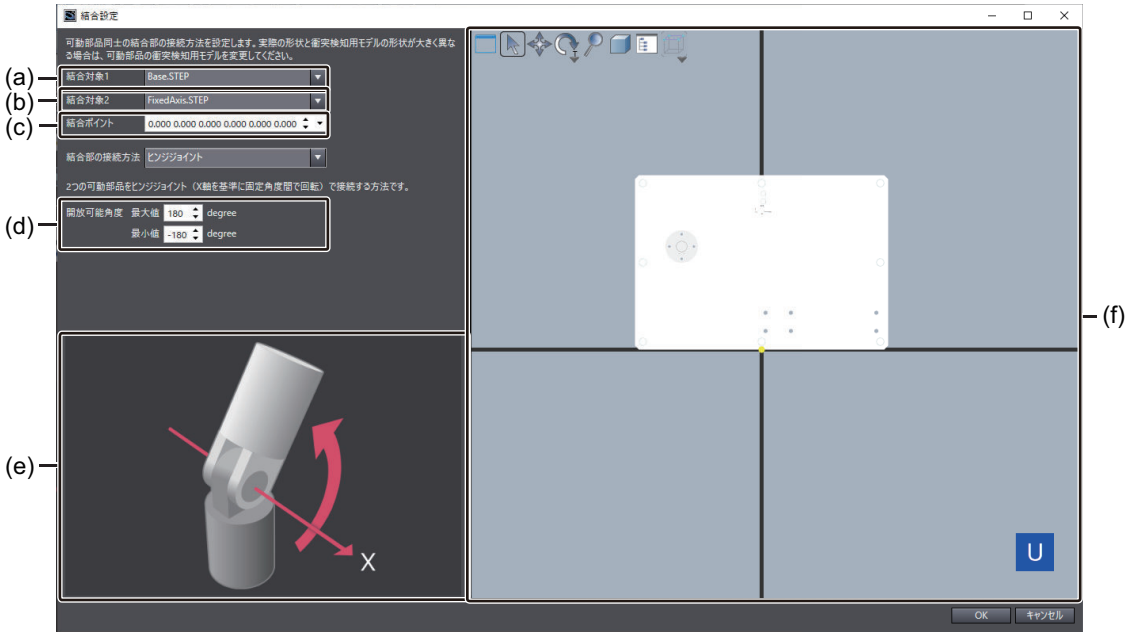
結合部ごとの設定項目は以下のとおりです。

●「固定ジョイント」



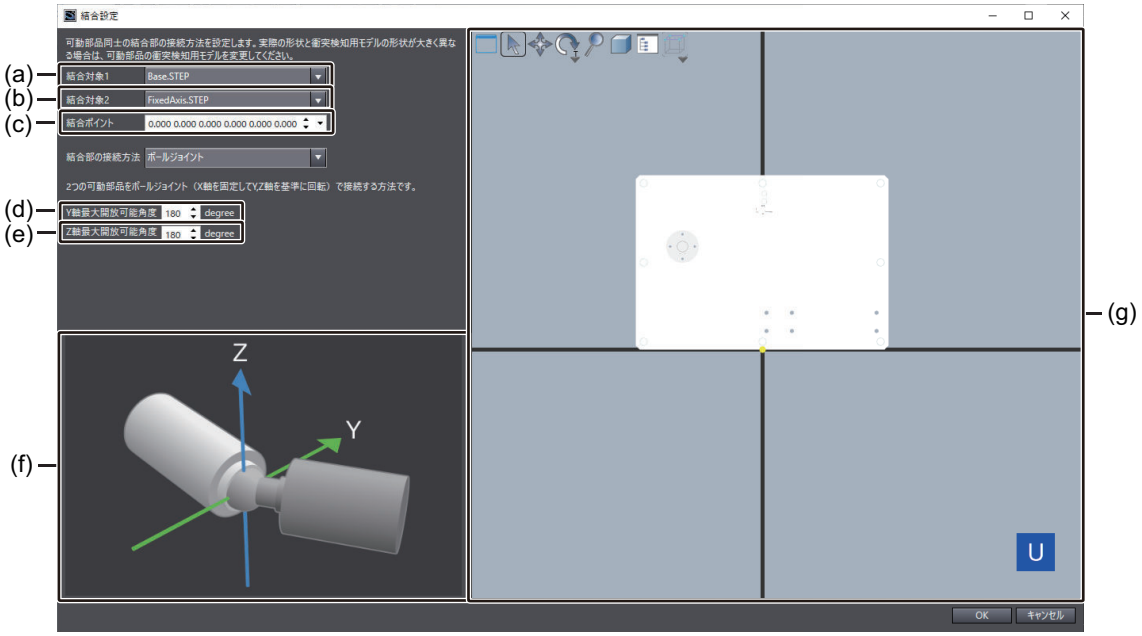
	設定項目	説明	設定値	初期値
(a)	結合対象 1	ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(b)	結合対象 2	ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(c)	結合ポイント	可動部品どうしの結合点の座標を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000 ～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(d)	3D 編集エリア	結合する可動部品と結合ポイントが表示されます。 マウス操作をして結合ポイントを移動したり可動部品にスナップしたりすることができます。	-	-

●「ヒンジジョイント」



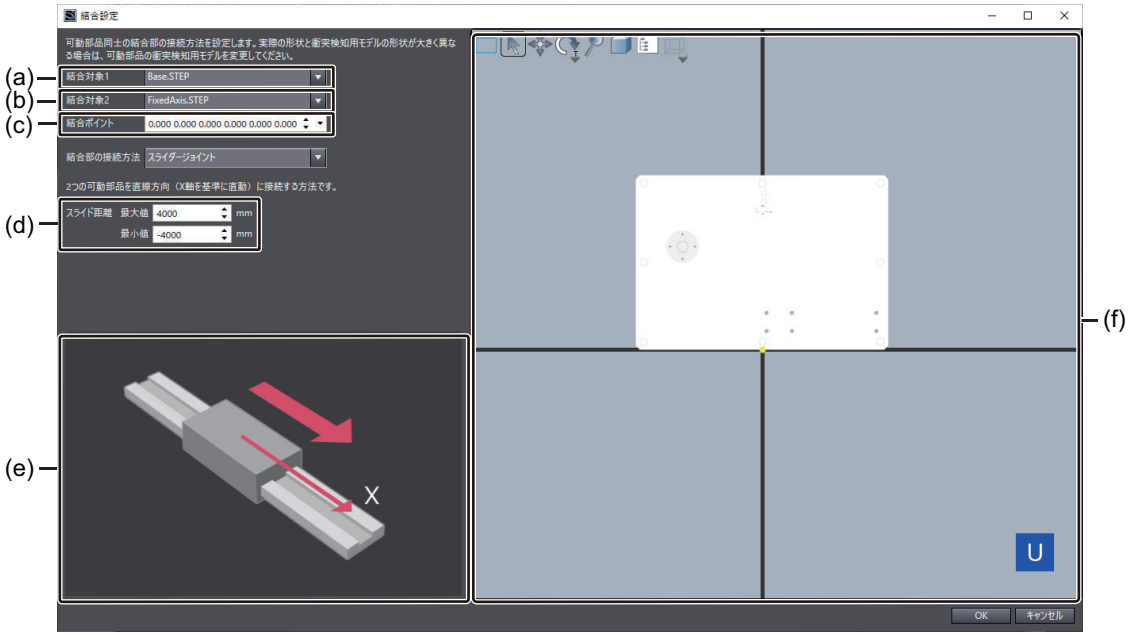
	設定項目		説明	設定値	初期値
(a)	結合対象 1		ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(b)	結合対象 2		ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(c)	結合ポイント		可動部品どうしの結合点の座標を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000 ～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(d)	開放可能角度	最大値	ヒンジジョイントが開く角度の最大値を設定します。単位は[degree]です。	-179.000～ +180.000 かつ最小値以上の値	180.000
		最小値	ヒンジジョイントが開く角度の最小値を設定します。単位は[degree]です。	-180.000～ +179.000 かつ最大値以下の値	-180.000
(e)	ヒンジジョイント説明図		ヒンジジョイントと開放可能角度について説明します。	-	-
(f)	3D 編集エリア		結合する可動部品と結合ポイントが表示されます。 マウス操作をして結合ポイントを移動したり可動部品にスナップしたりすることができます。	-	-

●「ボールジョイント」



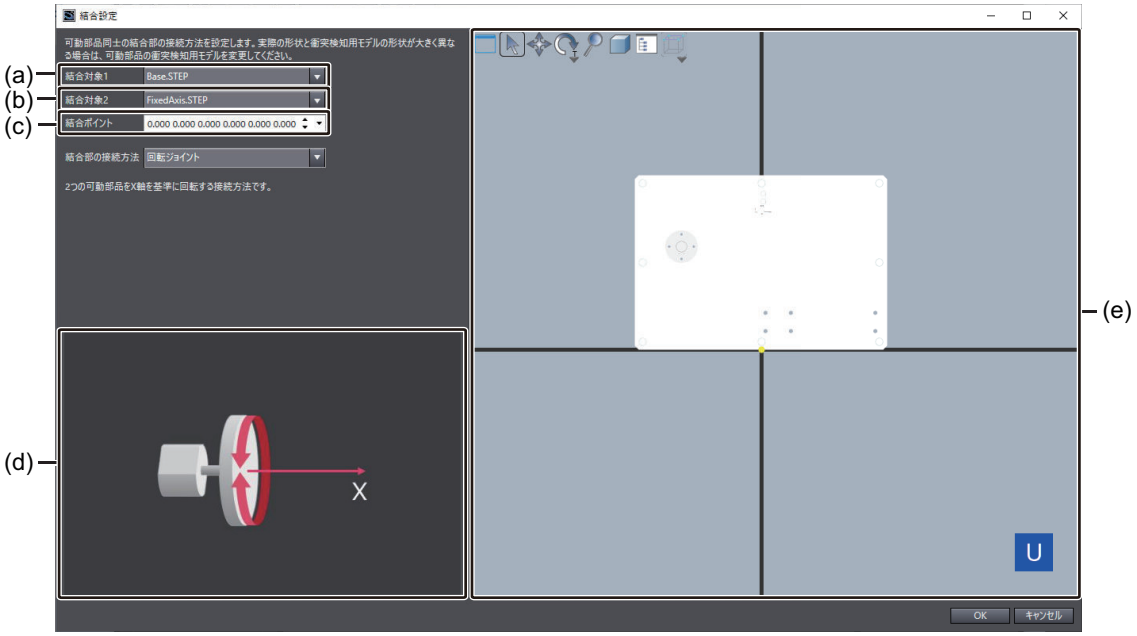
	設定項目	説明	設定値	初期値
(a)	結合対象 1	ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(b)	結合対象 2	ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(c)	結合ポイント	可動部品どうしの結合点の座標を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000 ～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(d)	Y 軸最大開放可能角度	ボールジョイントが Y 軸方向に開く角度の最大値を設定します。単位は[degree]です。	1.000～360.000	180.000
(e)	Z 軸最大開放可能角度	ボールジョイントが Z 軸方向に開く角度の最大値を設定します。単位は[degree]です。	1.000～360.000	180.000
(f)	ボールジョイント説明図	ボールジョイントと Y 軸、Z 軸の最大開放可能角度について説明します。	-	-
(g)	3D 編集エリア	結合する可動部品と結合ポイントが表示されます。 マウス操作をして結合ポイントを移動したり可動部品にスナップしたりすることができます。	-	-

●「スライダージョイント」



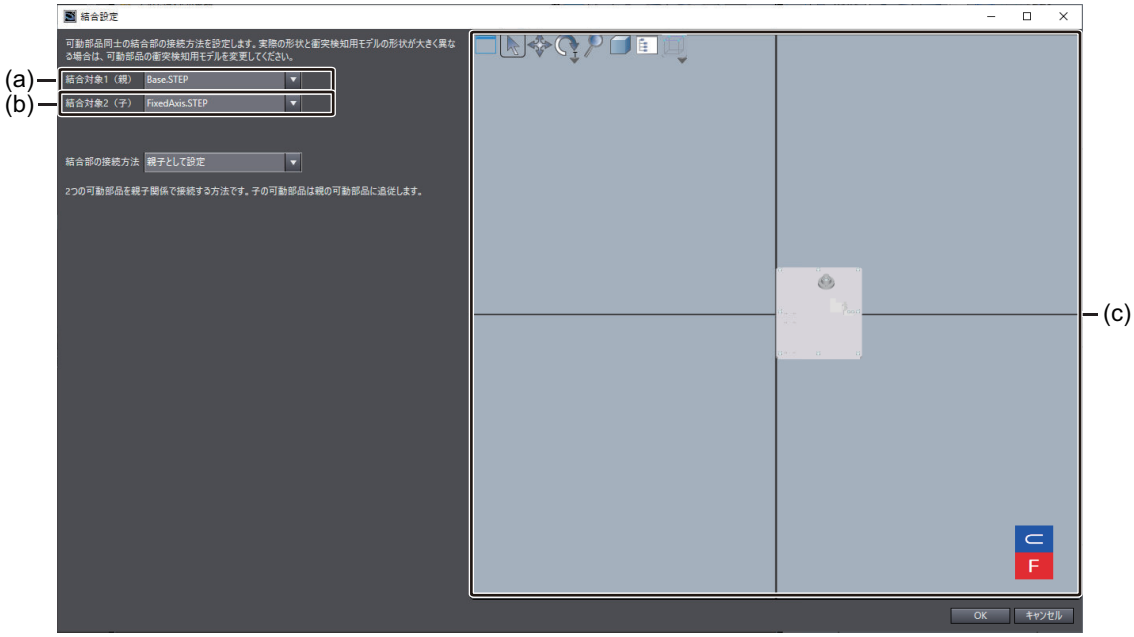
	設定項目		説明	設定値	初期値
(a)	結合対象 1		ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(b)	結合対象 2		ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(c)	結合ポイント		可動部品どうしの結合点の座標を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000 ～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(d)	スライド距離	最大値	スライダージョイントがスライドする距離の最大値を設定します。単位は[mm]です。	-3999.000～ +4000.000 かつ最小値以上の値	1000.000
		最小値	スライダージョイントがスライドする距離の最小値を設定します。単位は[mm]です。	-4000.000～ +3999.000 かつ最大値以下の値	-1000.000
(e)	スライダージョイント説明図		スライダージョイントおよびスライド距離について説明します。	-	-
(f)	3D 編集エリア		結合する可動部品と結合ポイントが表示されます。 マウス操作をして結合ポイントを移動したり可動部品にスナップしたりすることができます。	-	-

● **「回転ジョイント」**



	設定項目	説明	設定値	初期値
(a)	結合対象 1	ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(b)	結合対象 2	ジョイントで結合する可動部品のひとつを選択します。	カスタムメカニカル機構に追加されている可動部品	-
(c)	結合ポイント	可動部品どうしの結合点の座標を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000 ～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(d)	回転ジョイント説明図	回転ジョイントについて説明します。	-	-
(e)	3D 編集エリア	結合する可動部品と結合ポイントが表示されます。 マウス操作をして結合ポイントを移動したり可動部品にスナップしたりすることができます。	-	-

● 「親子として設定」



	設定項目	説明	設定値	初期値
(a)	結合対象 1（親）	親として結合する可動部品を選択します。	カスタムメカニカル機構に追加されている可動部品	-
(b)	結合対象 2（子）	子として結合する可動部品を選択します。	カスタムメカニカル機構に追加されている可動部品	-
(c)	3D 編集エリア	接続する可動部品が表示されます。	-	-

4-4-9 可動部品の動作方向設定

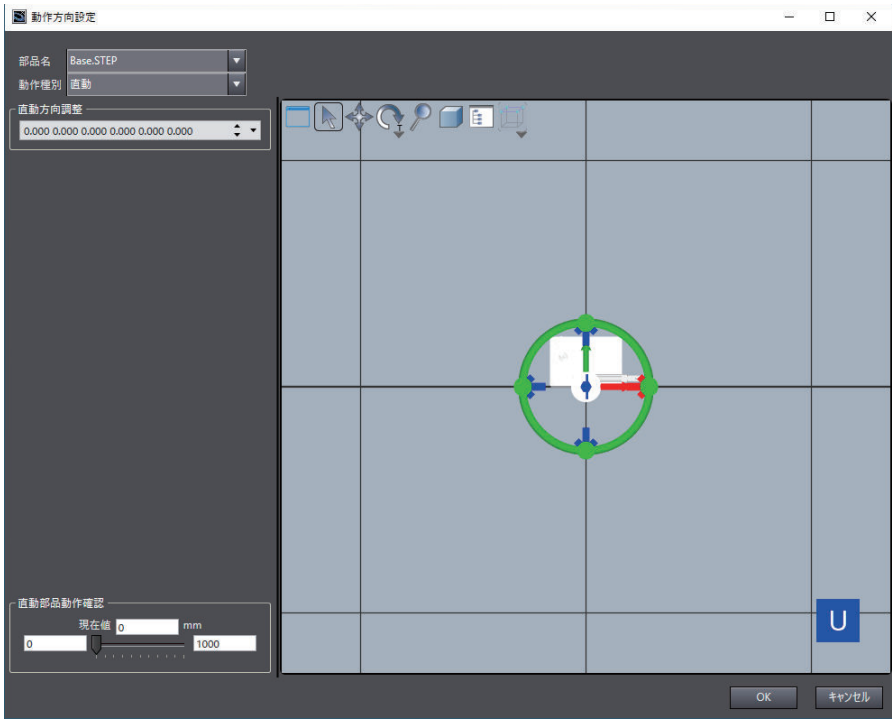
可動部品の動作方向を設定します。動作方向を設定するための動作種別を「直動」、「回転」、「直動・回転」の 3 種類から選択します。

設定手順

可動部品の動作方向の設定手順は以下のとおりです。

1 **カスタムメカニカル機構の設定画面で［動作方向設定］の右側のボタンをクリックします。**

2 **［動作方向設定］ダイアログボックスが表示されます。**



3 **[部品名] で動作方向設定を行う可動部品を選択します。**

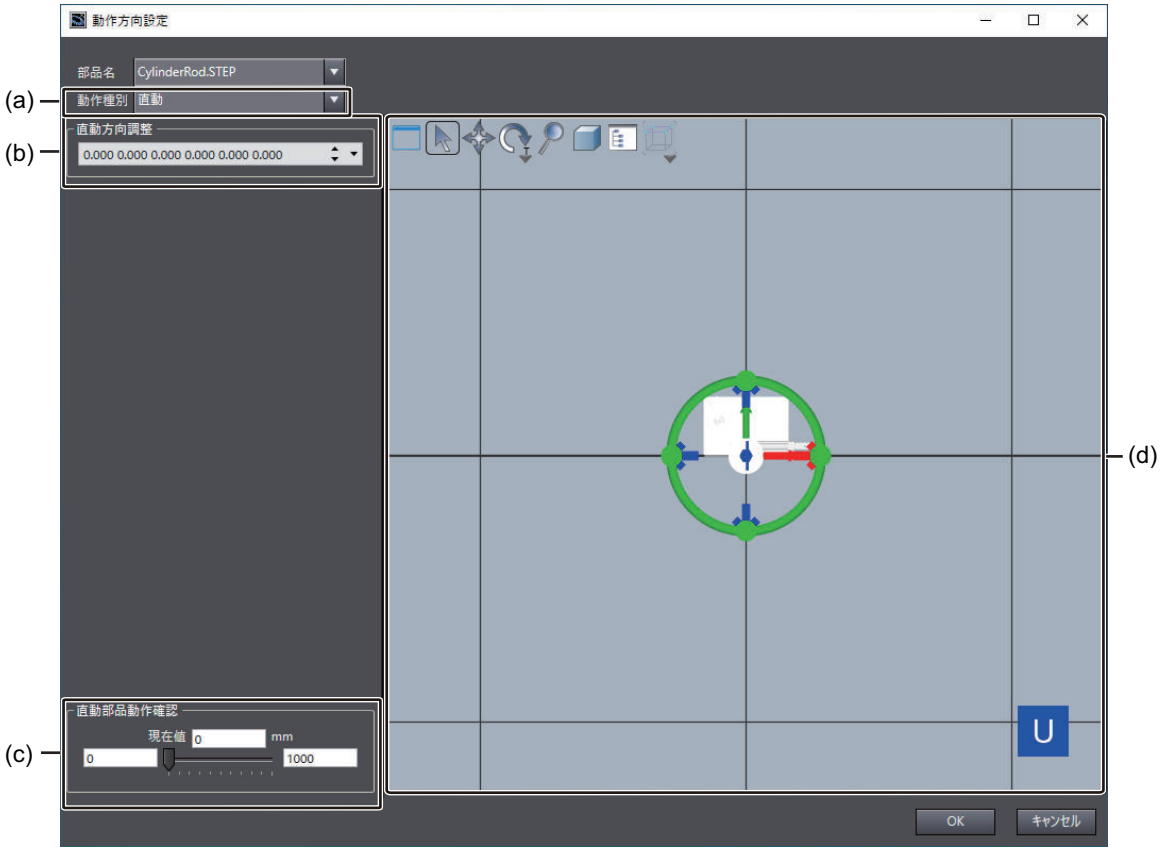
4 **[動作種別] で「直動」、「回転」、「直動・回転」のいずれかを選択します。**
選択した動作種別の設定項目が［動作種別］の項目の下に表示されます。表示される内容は動作種別によって異なります。

5 **動作方向の設定を行い、[OK] ボタンをクリックします。**
可動部品の動作方向設定が保存されます。
各動作種別の設定項目の詳細については「動作種別の設定項目 (P.4-56)」を参照してください。

動作種別の設定項目

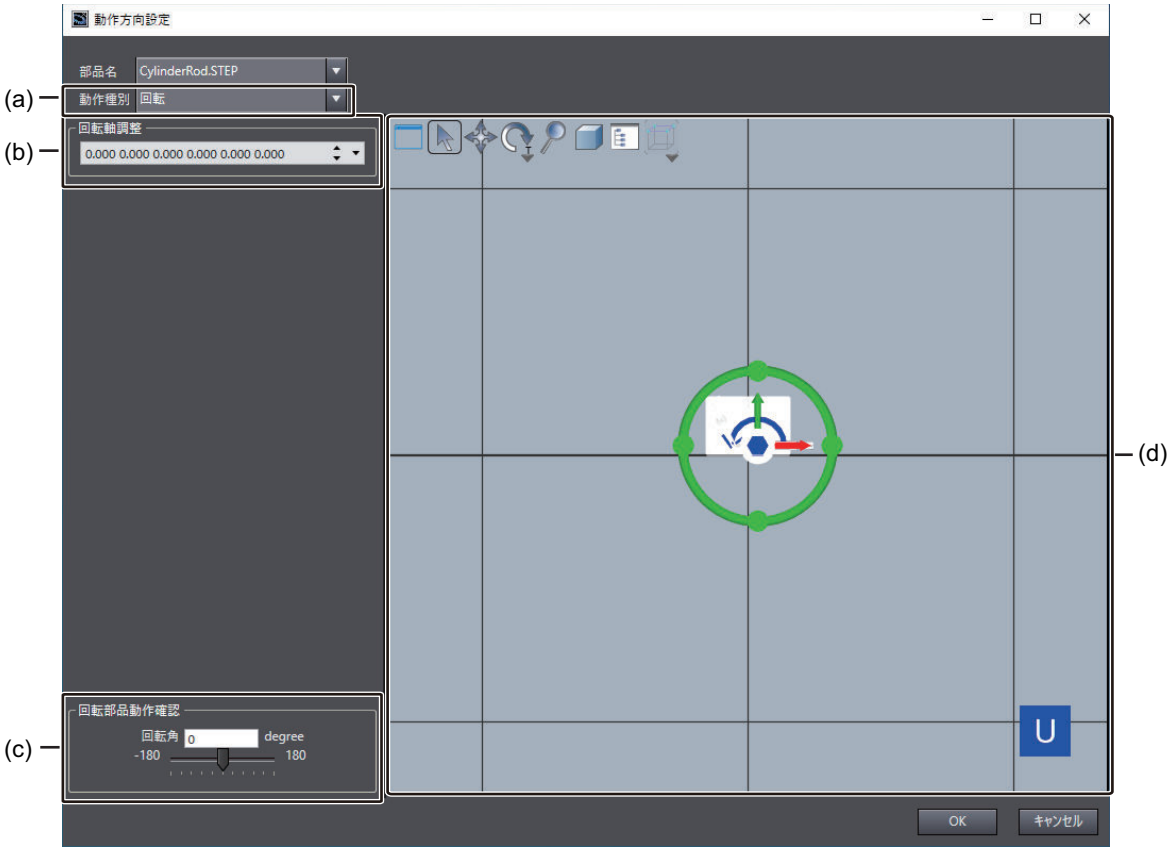
動作種別ごとの設定項目は以下のとおりです。

●「直動」



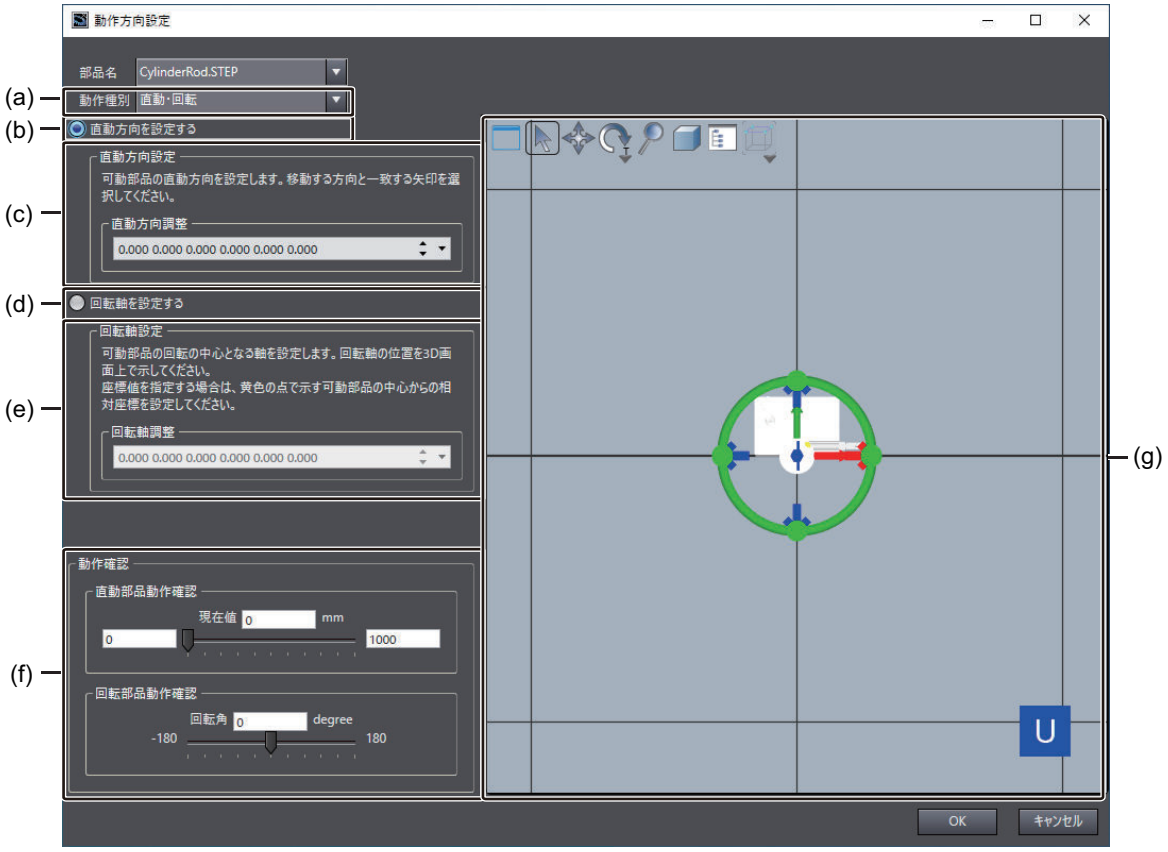
	設定項目	説明	設定値	初期値
(a)	動作種別	可動部品の動作種別を設定します。 ここでは「直動」を選択します。	直動	なし
(b)	直動方向調整	直動部品の動作の開始位置と、直動部品の姿勢を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -3.402823E+38 ～+3.402823E +38 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(c)	直動部品動作確認	スライダーを操作して、直動部品の動作を確認します。	-	最小値: 0 最大値: 1000
(d)	3D 編集エリア	直動部品の位置や姿勢を設定します。 3D 編集エリアの 6 つの青い矢印のアイコンをクリックして、可動部品の直動方向を選択します。選択された矢印のアイコンは赤くなります。 また、［直動部品動作確認］での操作に応じた直動部品の動作が表示されます。	-	-

● 「回転」



	設定項目	説明	設定値	初期値
(a)	動作種別	可動部品の動作種別を設定します。 ここでは「回転」を選択します。	回転	なし
(b)	回転軸調整	回転部品の動作の開始位置と、回転部品の姿勢を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -3.402823E+38 ～+3.402823E +38 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(c)	回転部品動作確認	スライダーを操作して、回転部品の動作を確認します。	-	-
(d)	3D 編集エリア	回転部品の位置や姿勢を設定します。［回転軸調整］の X、Y、Z、y、p、r の値を設定することで、回転軸の方向を設定します。設定すると、3D 編集エリアの回転軸の方向編集アイコンの向きが変わります。また、3D 編集エリアの回転軸の方向編集アイコンをドラッグして、方向を設定することもできます。	-	-

●「直動・回転」



	設定項目		説明	設定値	初期値
(a)	動作種別		可動部品の動作種別を設定します。 ここでは「直動・回転」を選択します。	直動・回転	なし
(b)	直動方向を設定する		直動方向を設定する場合に選択します。	-	-
(c)	直動方向設定		直動部品の動作の開始位置と、直動部品の姿勢を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -3.402823E+38 ～+3.402823E +38 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(d)	回転軸を設定する		回転軸を設定する場合に選択します。	-	-
(e)	回転軸設定		回転部品の動作の開始位置と、回転部品の姿勢を設定します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -3.402823E+38 ～+3.402823E +38 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
(f)	動作確認	直動部品動作確認	スライダーを操作して、直動部品の動作を確認します。	-	最小値: 0 最大値: 1000

	設定項目		説明	設定値	初期値
		回転部品動作確認	スライダーを操作して、回転部品の動作を確認します。	-	-
(g)	3D 編集エリア		[直動方向を設定する] を選択した場合、直動部品の位置や姿勢を設定します。また、[直動部品動作確認] での操作に応じた直動部品の動作が表示されます。 [回転軸を設定する] を選択した場合、回転部品の位置や姿勢を設定します。また、[回転部品動作確認] での操作に応じた回転部品の動作が表示されます。	-	-

4-4-10 カスタムメカニカル機構の動作確認

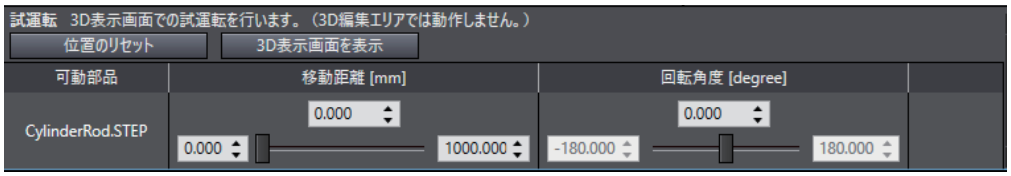
カスタムメカニカル機構の動作部品を動作させ、可動部品および結合部の動作設定が正しいことを確認します。確認は、可動部品の試運転画面で行います。

操作手順

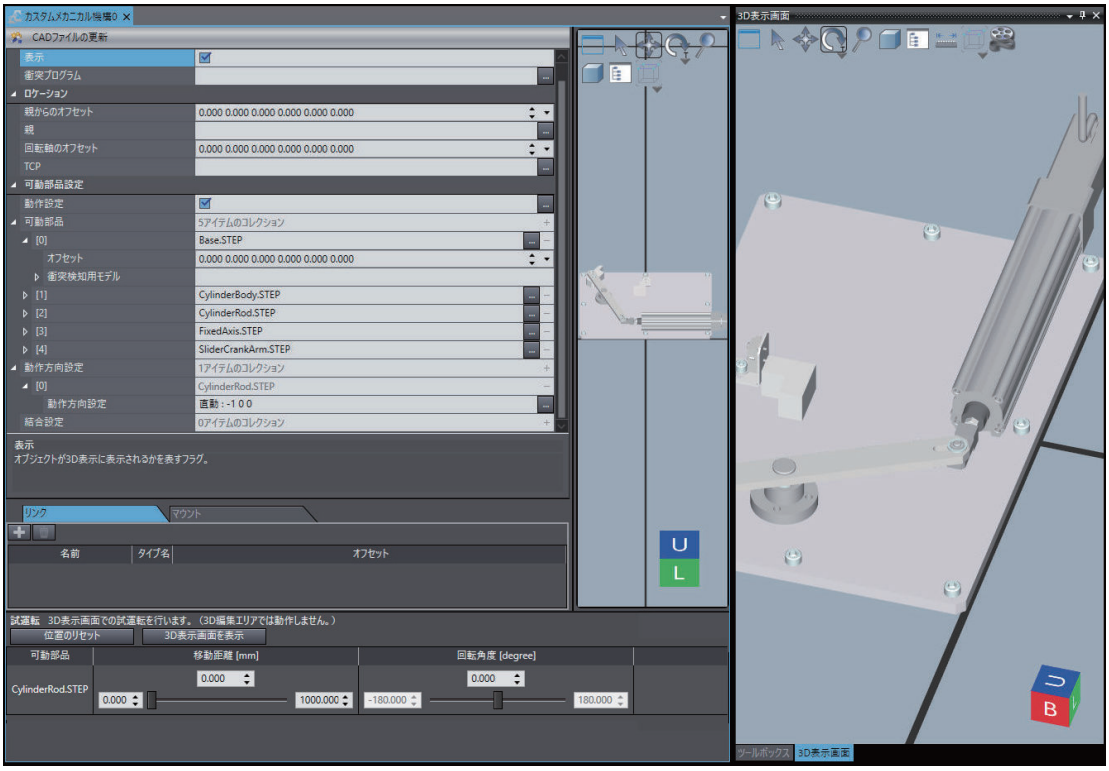
可動部品の試運転画面での操作手順は以下のとおりです。

1 **[構成・設定] の [3D 表示] から、対象のカスタムメカニカル機構をダブルクリックします。**
カスタムメカニカル機構の設定画面が表示されます。カスタムメカニカル機構の設定画面下部の [試運転] 画面に、「直動」、「回転」、「直動・回転」が設定されている可動部品が表示されます。

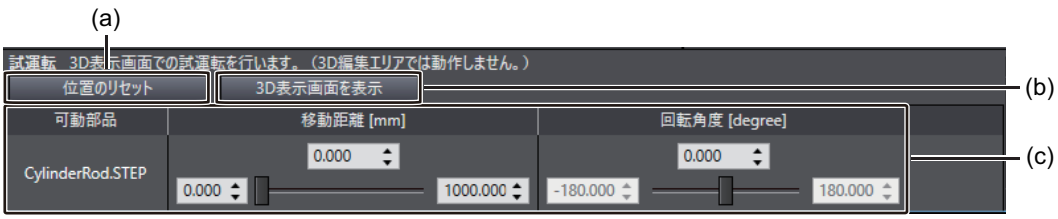
2 **[試運転] 画面で各可動部品の移動距離や回転角度を設定します。**



3 **移動距離、回転角度のスライダーをクリックします。**
3D 表示画面上の対象可動部品が、設定した移動距離、回転角度だけ移動します。



可動部品の試運転画面



	設定項目			説明	設定値	初期値
(a)	［位置のリセット］ボタン			位置のリセットボタンをクリックすると、試運転で動作した可動部品の位置がリセットされます。	-	-
(b)	［3D 表示画面を表示］ボタン			ツールボックスの位置に［3D 表示画面］を表示します。	-	-
(c)	可動部品リスト	可動部品		「直動」、「回転」、「直動・回転」が設定されている可動部品が表示されます。	-	-
		移動距離（単位: mm）		試運転中の可動部品の移動距離の表示と変更を行います。本項目は可動部品の動作種別が「直動」または「直動・回転」の場合に表示されます。	-	-
			最小値	［現在値］および［スライダー］で設定可能な値の最小値を設定します。最大値より大きい値を設定した場合は、最大値と同じ値が設定されます。	-10000.000～最大値	0.000

設定項目				説明	設定値	初期値
			最大値	［現在値］および［スライダー］で設定可能な値の最大値を設定します。最小値より小さい値が設定された場合は、最小値と同じ値が設定されます。	最小値～10000.000	1000.000
			現在値	可動部品の移動距離の現在値を表示します。テキストボックスに入力した値が［最小値］と［最大値］の範囲内の場合はその値の位置、下回る場合は［最小値］、上回る場合は［最大値］の位置に可動部品が移動します。	最小値～最大値	0.000
			スライダー	スライダーを操作して最小値と最大値の範囲内で移動距離の現在値を変更します。	最小値～最大値	0.000
		回転角度（単位: degree）		試運転中の可動部品の回転角度の表示と変更を行います。本項目は可動部品の動作種別が「直動」または「直動・回転」の場合に表示されます。	-	-
			最小値	［現在値］および［スライダー］で設定可能な値の最小値を表示します。	-	-180（固定）
			最大値	［現在値］および［スライダー］で設定可能な値の最大値を表示します。	-	180（固定）
			現在値	可動部品の回転角度の現在値を表示します。テキストボックスに入力した値が［最小値］と［最大値］の範囲内の場合はその値の角度、下回る場合は［最小値］、上回る場合は［最大値］の角度に可動部品が回転します。	最小値～最大値	0.000
			スライダー	スライダーを操作して最小値と最大値の範囲内で回転角度の現在値を変更します。	最小値～最大値	0.000

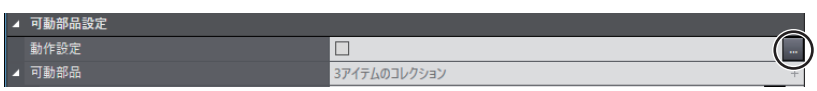
4-4-11　カスタムメカニカル機構の動作設定

カスタムメカニカル機構の可動部品の動作を定義するパラメータを設定します。位置決めは動作番号を指定して行う方法と、変数値を指定して行う方法があります。3D シミュレーションを実行すると、設定されたパラメータに合わせて可動部品を動作させることができます。

操作手順

● 動作番号を指定して位置決めを行う場合

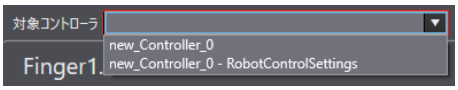
1　［動作設定］の右側のボタンをクリックします。



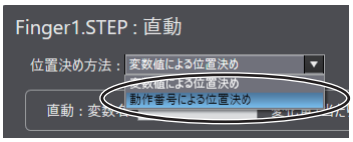
可動部品の動作設定画面が表示されます。



2 **［対象コントローラ］でカスタムメカニカル機構を動作させるコントローラを選択します。**
プロジェクトに追加されているコントローラまたはロボット統合 CPU ユニットのサブデバイスを選択することができます。



3 **［動作番号による位置決め］を選択します。**



4 **［I/O 変数名（動作開始）］、［I/O 変数名（位置決め完了）］、［変数名（動作番号指定）］、［変数名（現在位置取得）］に、可動部品を動作させる各変数名を設定します。**
設定する変数の詳細は後述の「動作番号による位置決めの場合 (P.4-65)」を参照してください。

5 **可動部品を動作させる動作パラメータを設定します。**
設定する動作パラメータの詳細は後述の「動作番号による位置決めの場合 (P.4-65)」を参照してください。

6 **手順 3～5 を動作設定画面に表示されているすべての可動部品に対して行います。**

● 変数値を指定して位置決めを行う場合

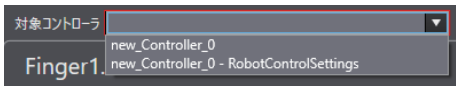
1 **［動作設定］の右側のボタンをクリックします。**



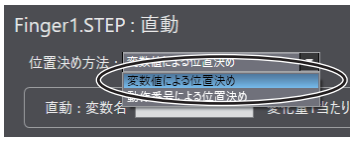
可動部品の動作設定画面が表示されます。



2 **［対象コントローラ］でカスタムメカニカル機構を動作させるコントローラを選択します。**
プロジェクトに追加されているコントローラまたはロボット統合 CPU ユニットのサブデバイスを選択することができます。



3 **［変数値による位置決め］を選択します。**

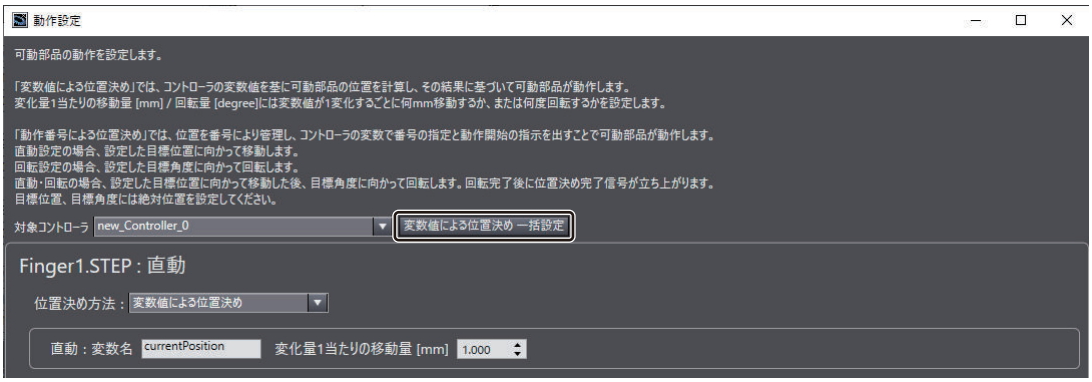


4 **［直動］および［回転］の［変数名］と［変化量 1 当たりの移動量(mm)／変化量 1 当たりの回転量(degree)］を設定します。**
パラメータの詳細は後述の「変数値による位置決めの場合 (P.4-67)」を参照してください。

5 **手順 3～4 を動作設定画面に表示されているすべての可動部品に対して行います。**

参考

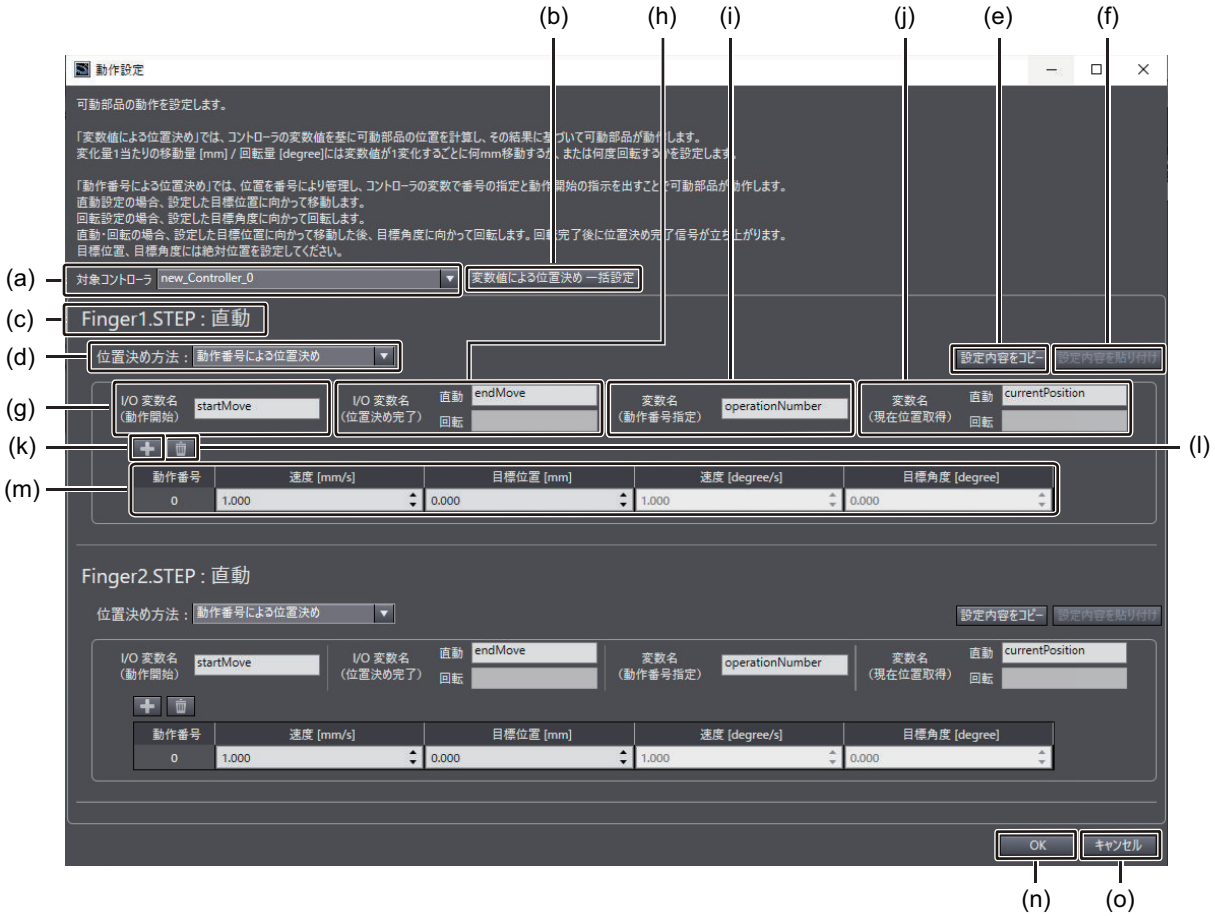
可動部品のうち、位置決め方法が「変数値による位置決め」に設定されている可動部品を一括で設定することができます。
可動部品の動作設定画面で［変数値による位置決め一括設定］ボタンをクリックします。



変数値による位置決め一括設定の各設定項目については後述の「変数値による位置決めの場合 一括設定 (P.4-68)」を参照してください。

設定項目

● 動作番号による位置決めの場合



	設定項目		説明	設定値	初期値
(a)	対象コントローラ		カスタムメカニカル機構を動作させるコントローラを選択します。	プロジェクトに追加されているコントローラまたはロボット統合 CPU ユニットのサブデバイス	-
(b)	[変数値による位置決め 一括設定] ボタン		可動部品のうち、位置決め方法が「変数値による位置決め」に設定されている可動部品を一括で設定します。設定項目については後述の「変数値による位置決めの場合 一括設定 (P.4-68)」を参照してください。	-	-
(c)	可動部品名		動作方向設定で登録した可動部品名が表示されます。	-	-
(d)	位置決め方法		可動部品の位置決め方法を選択します。	変数値による位置決め／動作番号による位置決め	変数値による位置決め
(e)	[設定内容をコピー] ボタン		I/O 変数名 (動作開始)、I/O 変数名 (位置決め完了)、変数名 (動作番号指定)、変数名 (現在位置取得)、動作パラメータの内容を一括でコピーします。	-	-
(f)	[設定内容を貼り付け] ボタン		別の可動部品名の [設定内容をコピー] ボタンでコピーした設定内容を、一括で貼り付けます。	-	-
(g)	I/O 変数名 (動作開始)		位置決め動作を開始する BOOL 型のグローバル変数を設定します。対象コントローラがロボット統合 CPU ユニットのサブデバイスの場合は V+デジタル I/O 番号の設定も可能です。	BOOL 型のグローバル変数または V+デジタル I/O 番号	-
(h)	I/O 変数名 (位置決め完了)	直動	直動位置決め完了信号の値を書き込む BOOL 型のグローバル変数を設定します。対象コントローラがロボット統合 CPU ユニットのサブデバイスの場合は V+デジタル I/O 番号の設定も可能です。	BOOL 型のグローバル変数または V+デジタル I/O 番号	-
		回転	回転位置決め完了信号の値を書き込む BOOL 型のグローバル変数を設定します。対象コントローラがロボット統合 CPU ユニットのサブデバイスの場合は V+デジタル I/O 番号の設定も可能です。	BOOL 型のグローバル変数または V+デジタル I/O 番号	-
(i)	変数名 (動作番号指定)		動作番号を指定する UDINT 型のグローバル変数を設定します。対象コントローラがロボット統合 CPU ユニットのサブデバイスの場合は REAL 型の V+変数の設定も可能です。	UDINT 型のグローバル変数または REAL 型の V+変数	-
(j)	変数名 (現在位置取得)	直動	直動現在位置の値を書き込む LREAL 型のグローバル変数を設定します。対象コントローラがロボット統合 CPU ユニットのサブデバイスの場合は REAL 型の V+変数の設定も可能です。	LREAL 型のグローバル変数または REAL 型の V+変数	-
		回転	回転現在位置の値を書き込む LREAL 型のグローバル変数を設定します。対象コントローラがロボット統合 CPU ユニットのサブデバイスの場合は REAL 型の V+変数の設定も可能です。	LREAL 型のグローバル変数または REAL 型の V+変数	-

	設定項目		説明	設定値	初期値
(k)	追加ボタン		動作パラメータ行を追加します。パラメータ行が選択されている場合は、選択されている行の下に追加されます。	-	-
(l)	削除ボタン		動作パラメータ行を削除します。	-	-
(m)	動作パラメータ	動作番号	動作パラメータを指定するための番号です。	-	-
		速度 [mm/s]	目標位置への移動速度を設定します。速度の値が正の場合、目標位置に近付く方向に動作し、負の場合は目標位置から離れる方向に動作します。	-10,000.000～+10,000.000	1.000
		目標位置 [mm]	目標位置を設定します。速度の値が正の場合、目標位置に近付く方向に動作し、負の場合は目標位置から離れる方向に動作します。	-4,000.000～+4,000.000	0.000
		速度 [degree/s]	目標角度への回転速度を設定します。速度の値が正の場合、目標角度に近付く方向に動作し、負の場合は目標角度から離れる方向に動作します。	-10,000.000～+10,000.000	1.000
		目標角度 [degree]	目標角度を設定します。速度の値が正の場合、目標角度に近付く方向に動作し、負の場合は目標角度から離れる方向に動作します。	-180.000～+180.000	0.000
(n)	[OK] ボタン		可動部品の動作設定内容を確定します。	-	-
(o)	[キャンセル] ボタン		設定をキャンセルします。	-	-

ロボット統合 CPU ユニット、V+デジタル I/O 番号、および V+変数については『NJ シリーズ ロボット統合 CPU ユニット ユーザーズマニュアル(SBCE-436)』および『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編 (SBCA-464)』を参照してください。

● 変数値による位置決めの場合



	設定項目	説明	設定値	初期値
(a)	対象コントローラ	カスタムメカニカル機構を動作させるコントローラを選択します。	プロジェクトに追加されているコントローラまたはロボット統合 CPU ユニットのサブデバイス	-
(b)	[変数値による位置決め 一括設定] ボタン	可動部品のうち、位置決め方法が「変数値による位置決め」に設定されている可動部品を一括で設定します。設定項目については後述の「変数値による位置決めの場合 一括設定 (P.4-68)」を参照してください。	-	-
(c)	可動部品名	動作方向設定で登録した可動部品名が表示されます。	-	-
(d)	位置決め方法	可動部品の位置決め方法を選択します。	変数値による位置決め／動作番号による位置決め	変数値による位置決め
(e)	変数名	現在位置として取得する LREAL 型の変数名を設定します。[直動] および [回転] についてそれぞれ設定します。	LREAL 型のグローバル変数	-
(f)	変化量 1 当たりの移動量(mm)／変化量 1 当たりの回転量(degree)	可動部品の現在位置を計算する際に使用する単位です。[変数名] で指定した変数の値に本項目で設定した値を乗算した位置に可動部品を移動します。[直動] および [回転] についてそれぞれ設定します。	-10000.000～+10000.000	-1.0

● 変数値による位置決めの場合 一括設定



	設定項目	説明
(a)	[コピー] ボタン	設定内容をクリップボードに tsv 形式でコピーします。[直動]、[回転] などの表のタイトルはヘッダとして含みません。
(b)	[貼り付け] ボタン	クリップボードから tsv 形式のテキストの内容を元にペーストします。テキストの行のうち、可動部品名が一致するもののみペーストします。
(c)	設定一覧	位置決め方法が「変数値による位置決め」の可動部の設定が一覧で表示されます。動作種別により、対応していない動作は表示しません。
(d)	[OK] ボタン	設定内容を確定し、[変数値による位置決め 一括設定] ウィンドウを閉じます。
(e)	[キャンセル] ボタン	設定内容をキャンセルし、[変数値による位置決め 一括設定] ウィンドウを閉じます。

4-4-12　CAD データの更新

登録済みの可動部品の CAD データを一括して置き換えることができます。

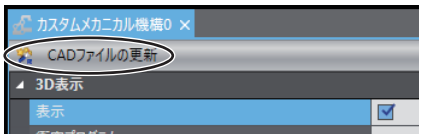
操作手順

操作手順は以下のとおりです。

参考

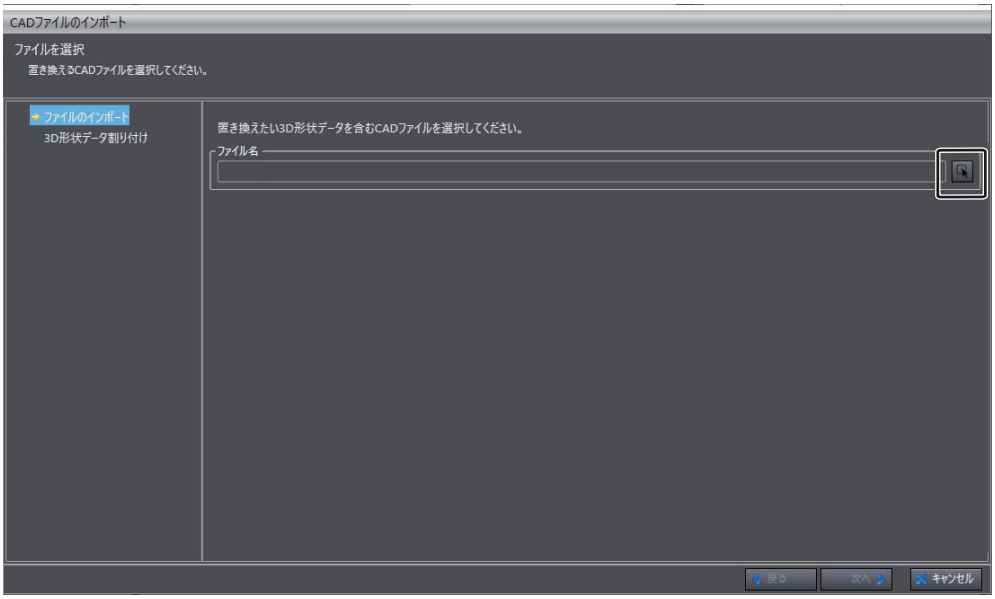
インポート時 CAD データのメッシュの粗さをオプションで設定することができます。オプション設定の詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

1　**対象のカスタムメカニカル機構の設定画面で［CAD ファイルの更新］ボタンをクリックします。**



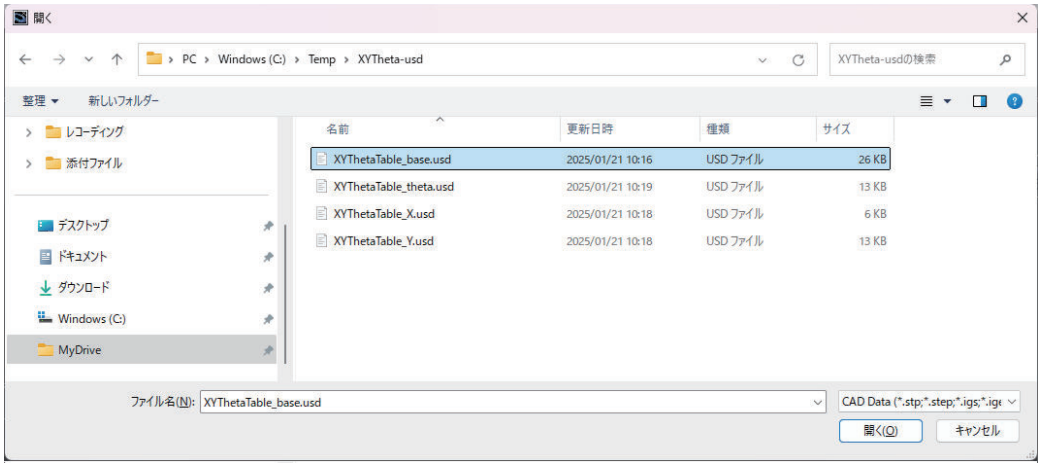
ウィザードが起動し、［ファイルを選択］画面が表示されます。

2　**［ファイルを選択］の画面で、［ファイル名］の右側のボタンをクリックします。**



［開く］ダイアログボックスが表示されます。

3　**［開く］ダイアログボックスで更新する STEP 形式、IGES 形式または OpenUSD 形式の CAD データファイルを選択し、［開く］ボタンをクリックします。**



［ファイルを選択］の画面で［次へ］ボタンをクリックすると、［3D 形状データ割り付け］画面が表示されます。

4 **［3D 形状データ割り付け］画面で更新する CAD データを 3D 形状データに割り付けます。**
画面の操作手順については、「10-1-1 一括置換手順 (P.10-2)」を参照してください。

参考

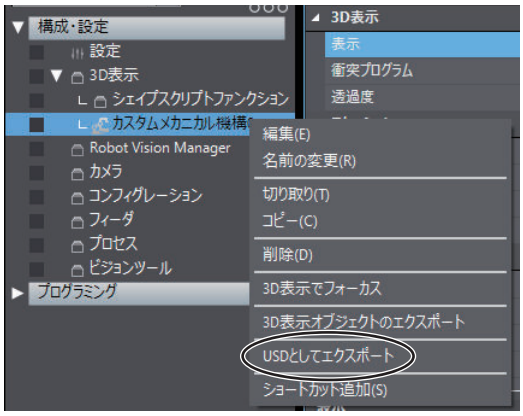
同じ PrimPath を持つ既存の 3D 形状データを更新した場合、3D 形状データの割り付けが自動で行われます。

4-4-13 OpenUSD ファイルのエクスポート

登録済みのカスタムメカニカル機構を OpenUSD ファイル形式でエクスポートすることができます。各可動部品の結合設定と動作方向設定を行っている場合、結合情報を含んだ OpenUSD ファイルをエクスポートすることができます。

エクスポート手順

1 **マルチビューエクスプローラの［構成・設定］-［3D 表示］項目下の「カスタムメカニカル機構 0」を右クリックし、メニューから［USD としてエクスポート］をクリックします。**



［ファイルをエクスポート］ダイアログボックスが表示されます。

2 **ファイル名を入力し、ファイルの種類を「USD」、「USDA」、「USDC」から選択し、[保存]ボタンをクリックします。**



エクスポート対象のカスタムメカニカル機構の可動部品の数分と、各可動部品の構成情報が、OpenUSD 形式ファイルに保存されます。

エクスポートした OpenUSD ファイルに含むデータの種類

エクスポートした OpenUSD ファイルに含むデータの種類は以下のとおりです。

- ジオメトリ(UsdGeomMesh)
- マテリアル(To define the material model)
- ジョイント(UsdPhysicsJoint)
  ジョイントの型は以下の 3 種類です。
  UsdPhysicsRevoluteJoint/UsdPhysicsPrismaticJoint/UsdPhysicsFixedJoint

以下に Sysmac Studio での結合部の動作設定と、エクスポートされる OpenUSD の USDPhysicsJoint の対応を示します。

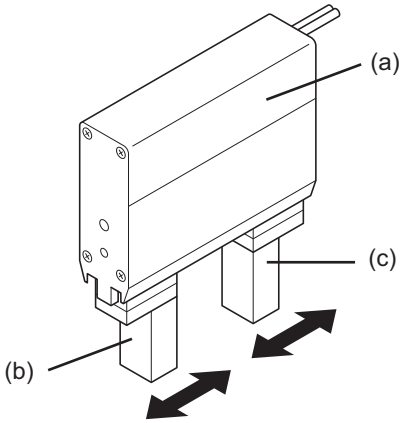
Sysmac Studio での結合部の動作設定	エクスポートされる OpenUSD の USDPhysicsJoint
スライダージョイント	UsdPhysicsPrismaticJoint
回転ジョイント	UsdPhysicsRevoluteJoint
固定ジョイント	UsdPhysicsFixedJoint

結合部の動作設定の詳細については「4-4-8 結合部の動作設定 (P.4-47)」を参照してください。

4-4-14　カスタムメカニカル機構の作成例

カスタムメカニカル機構の例として電動チャック（2 爪）、電動シリンダを作成し、動作設定をする手順を説明します。

電動チャック（2 爪）



可動部品:

	可動部品の名称（CAD データのパーツ）	動作種別
(a)	ベース	なし
(b)	爪 1	直動
(c)	爪 2	直動

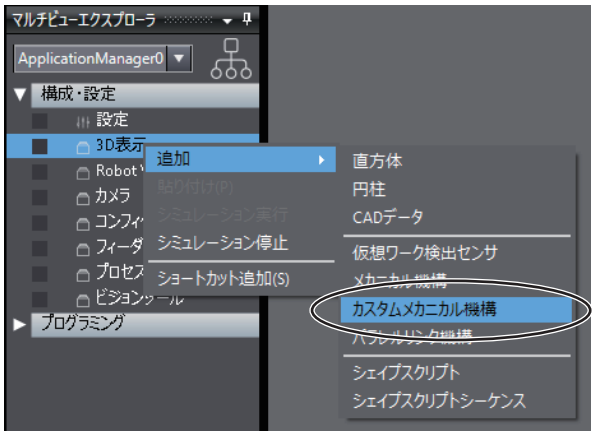
結合部: なし

● 作成手順

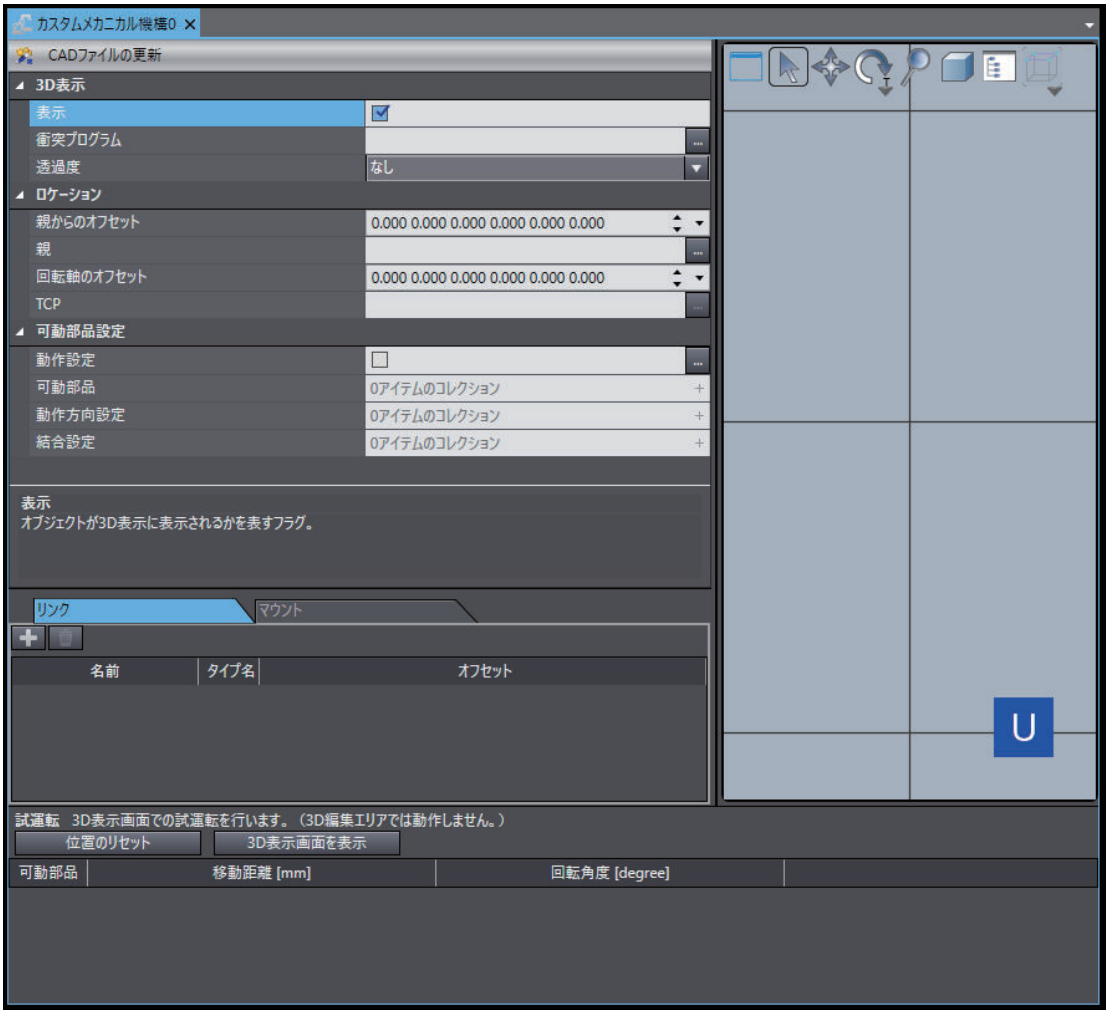
1 **3D CAD ソフトウェアを使用して、電動チャック（2 爪）を可動部品ごとにパーツに分けて保存します。**

3D CAD ソフトウェアを使用した CAD データファイルの出力方法については、「4-3-2 CAD データファイルの準備 (P.4-9)」を参照してください。

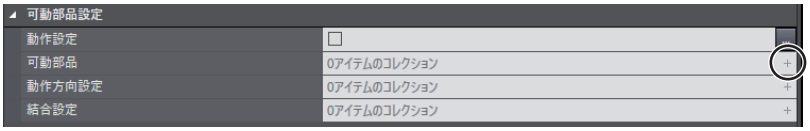
2 **［構成・設定］の［3D 表示］を右クリックし、メニューから［追加］-［カスタムメカニカル機構］を選択してカスタムメカニカル機構を登録します。**



3 **登録したカスタムメカニカル機構をダブルクリックし、カスタムメカニカル機構の設定画面を表示します。**



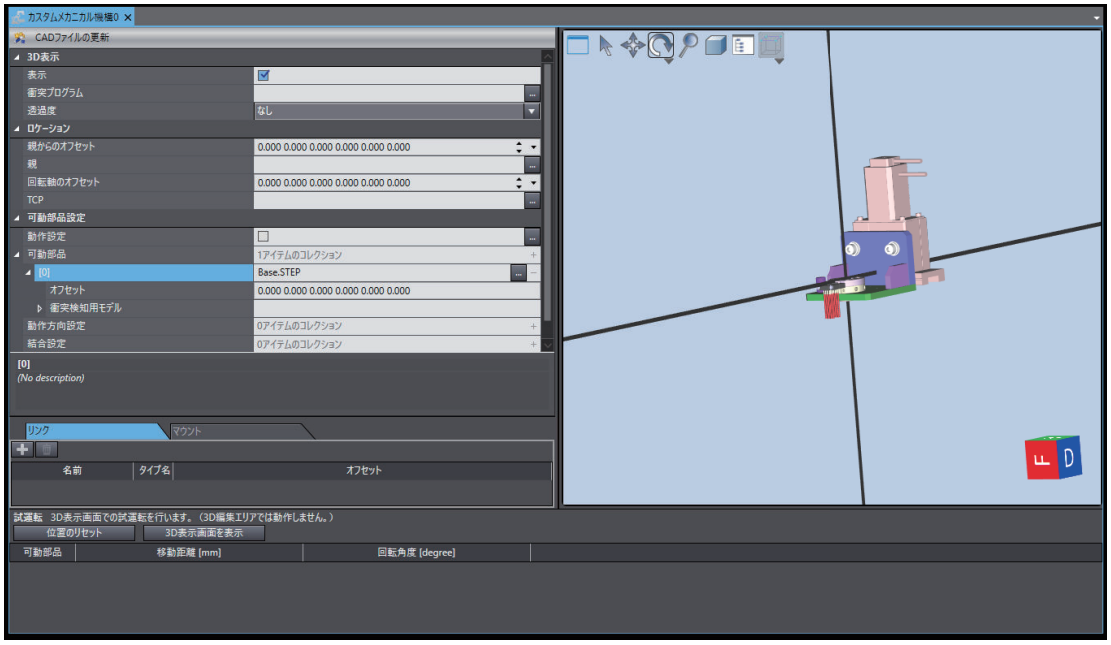
4 **［可動部品］の右側の追加ボタンをクリックします。**



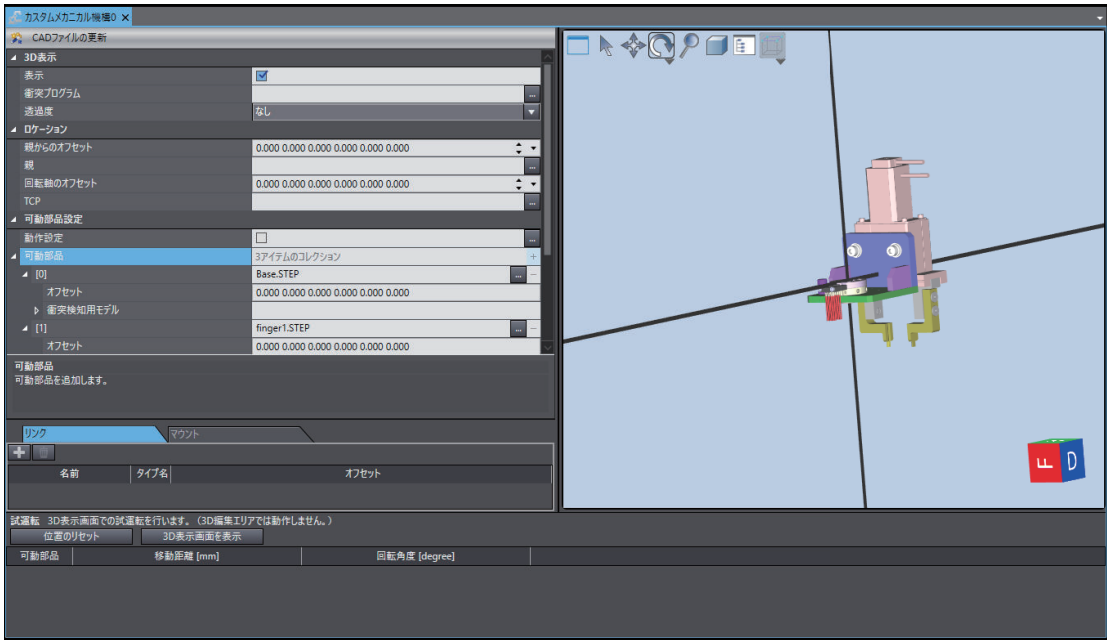
［開く］ダイアログボックスが表示されます。

5 **［開く］ダイアログボックスで、手順 1 で保存した可動部品の CAD データファイルを選択し、［開く］ボタンをクリックします。**

選択した CAD データの項目名が追加されます。また、インポートした CAD ファイルのデータが 3D 編集エリアに表示されます。



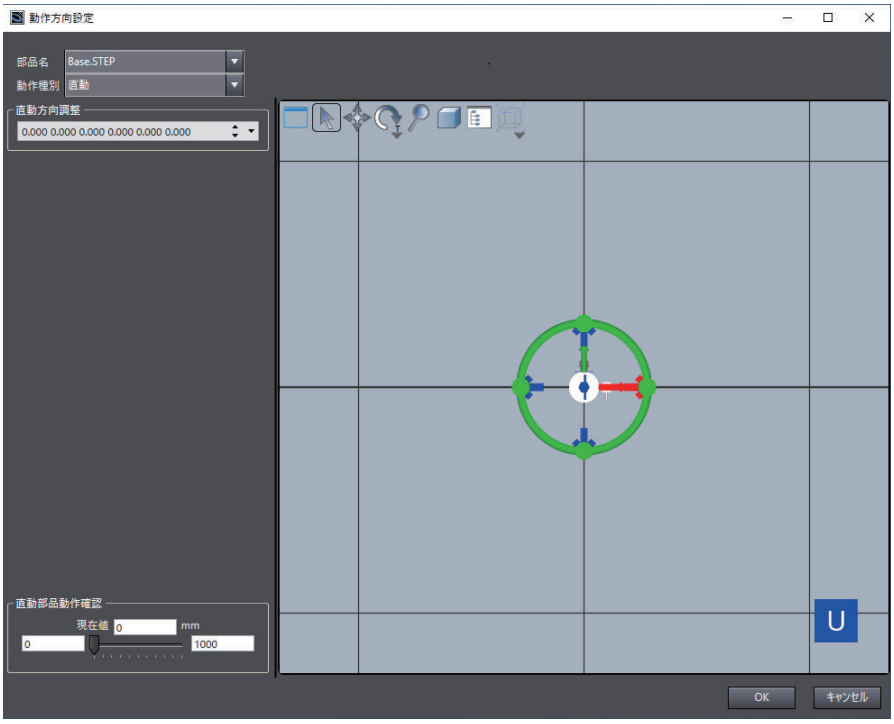
6 同様にして、すべての可動部品の CAD データを追加します。



7 [動作方向設定] の右側のボタンをクリックし、動作方向設定項目を追加します。

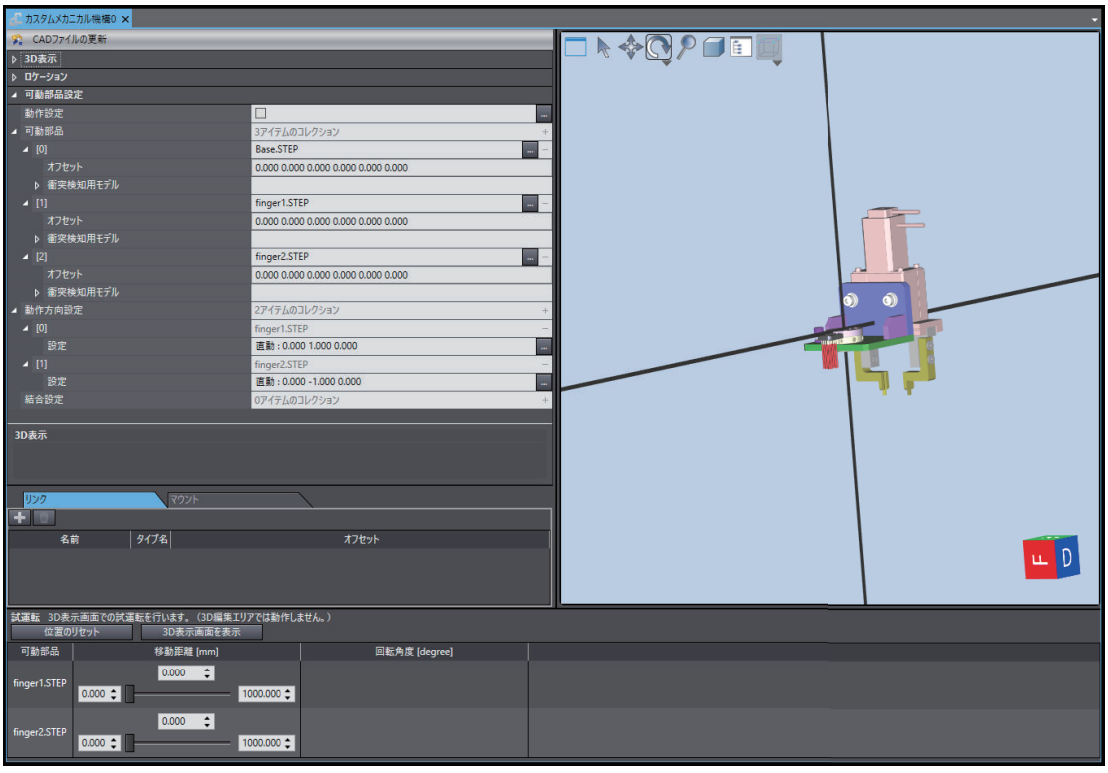


8 [動作方向設定] ダイアログボックスが表示されます。



9 **可動部品の動作種別を選択して必要な設定を行い、[OK] ボタンをクリックします。**
可動部品の動作方向設定が保存されます。

10 **同様にすべての可動部品の動作方向設定を行います。**



11 **[動作設定] の右側のボタンをクリックします。**



12 **[動作設定] ダイアログボックスが表示されます。**



設定項目の詳細については、「4-4-11 カスタムメカニカル機構の動作設定 (P.4-62)」を参照してください。

13 **設定値を入力し、[OK] ボタンをクリックします。**

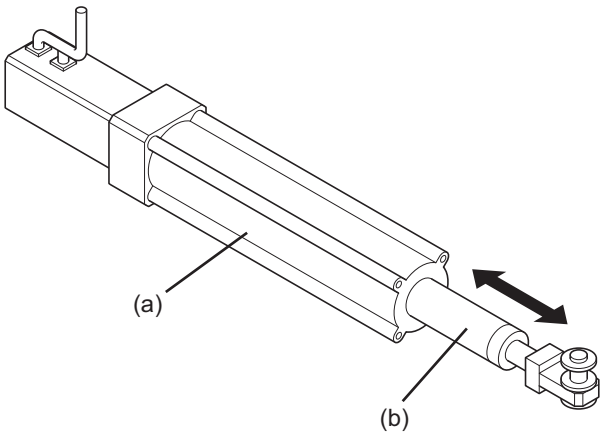
14 **[動作設定] をチェックします。**



これで電動チャック（2 爪）の作成、および動作設定は完了です。

電動シリンダ

例として次のような電動シリンダを作成します。



可動部品:

	可動部品の名称（CAD データのパーツ）	動作種別
(a)	シリンダ	動作なし
(b)	ピストンロッド	直動

● 作成手順

作成手順は、電動チャック（2 爪）と同様です。「電動チャック（2 爪）（P.4-72)」の作成手順を参照してください。

4-5 パラレルリンク機構の 3D 形状データの作成

パラレルリンク機構は、あらかじめ用意されている機構から選択して作成します。NJ シリーズ NJ Robotics CPU ユニットでサポートする次の機構から選択することができます。

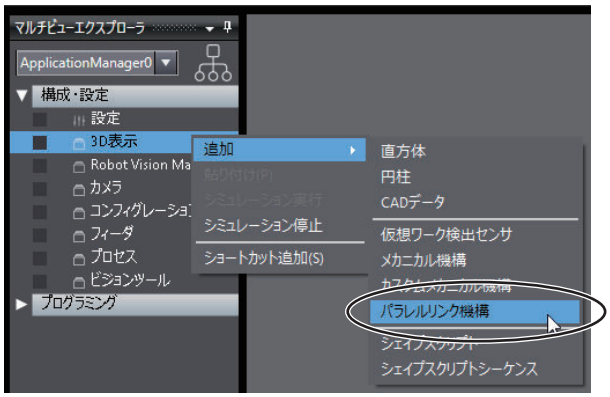
種別	説明
Delta3	3 つのアームで構成されるパラレルリンク機構です。
Delta3R	Delta3 に手首の回転軸を加えた機構です。
Delta2	2 つのアームで構成されるパラレルリンク機構です。

作成したパラレルリンク機構の 3D 形状データは、CAD データをインポートして置き換えることができます。
NJ シリーズ NJ Robotics CPU ユニットでサポートする機構の詳細については、『NJ シリーズ NJ Robotics CPU ユニット ユーザーズマニュアル(SBCA-421)』を参照してください。

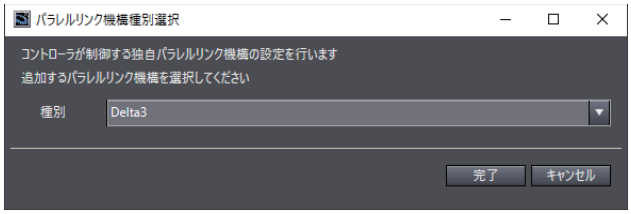
4-5-1 パラレルリンク機構の追加手順

パラレルリンク機構の追加手順は以下のとおりです。

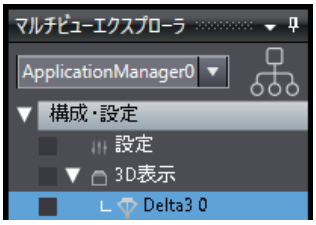
1 **［構成・設定］の［3D 表示］を右クリックし、メニューから［追加］-［パラレルリンク機構］を選択します。**



［パラレルリンク機構種別選択］ダイアログボックスが表示されます。



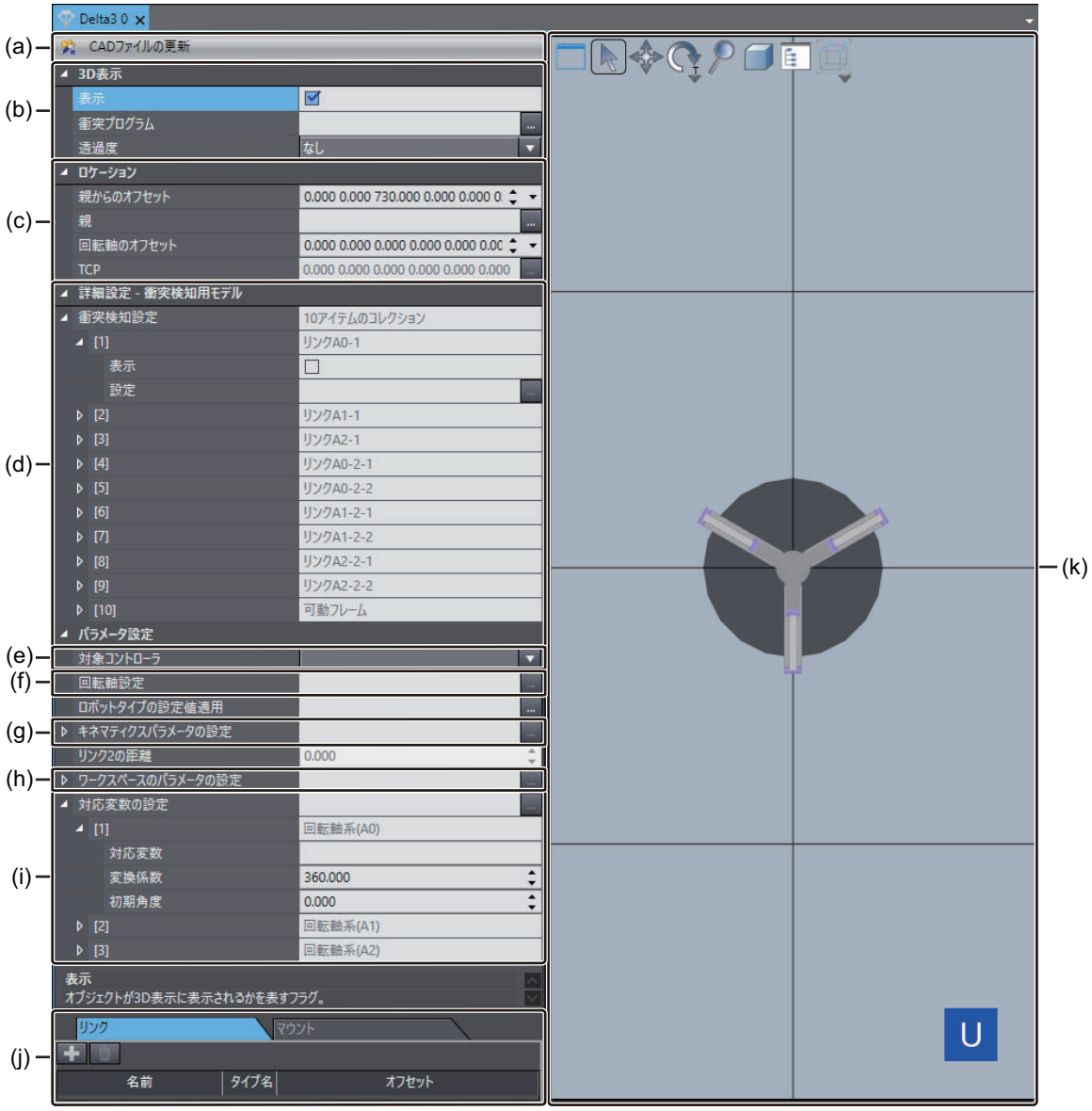
2 **［種別］から使用するパラレルリンク機構の種別を選択し、［完了］ボタンをクリックします。**
［3D 表示］の下に選択した機構が登録されます。



4-5-2 パラレルリンク機構の設定項目

追加したパラレルリンク機構をダブルクリックすると、パラレルリンク機構の設定画面が表示されます。
パラレルリンク機構の共通の設定項目、および機構種別ごとの固有の設定項目について説明します。

パラレルリンク機構の共通設定項目

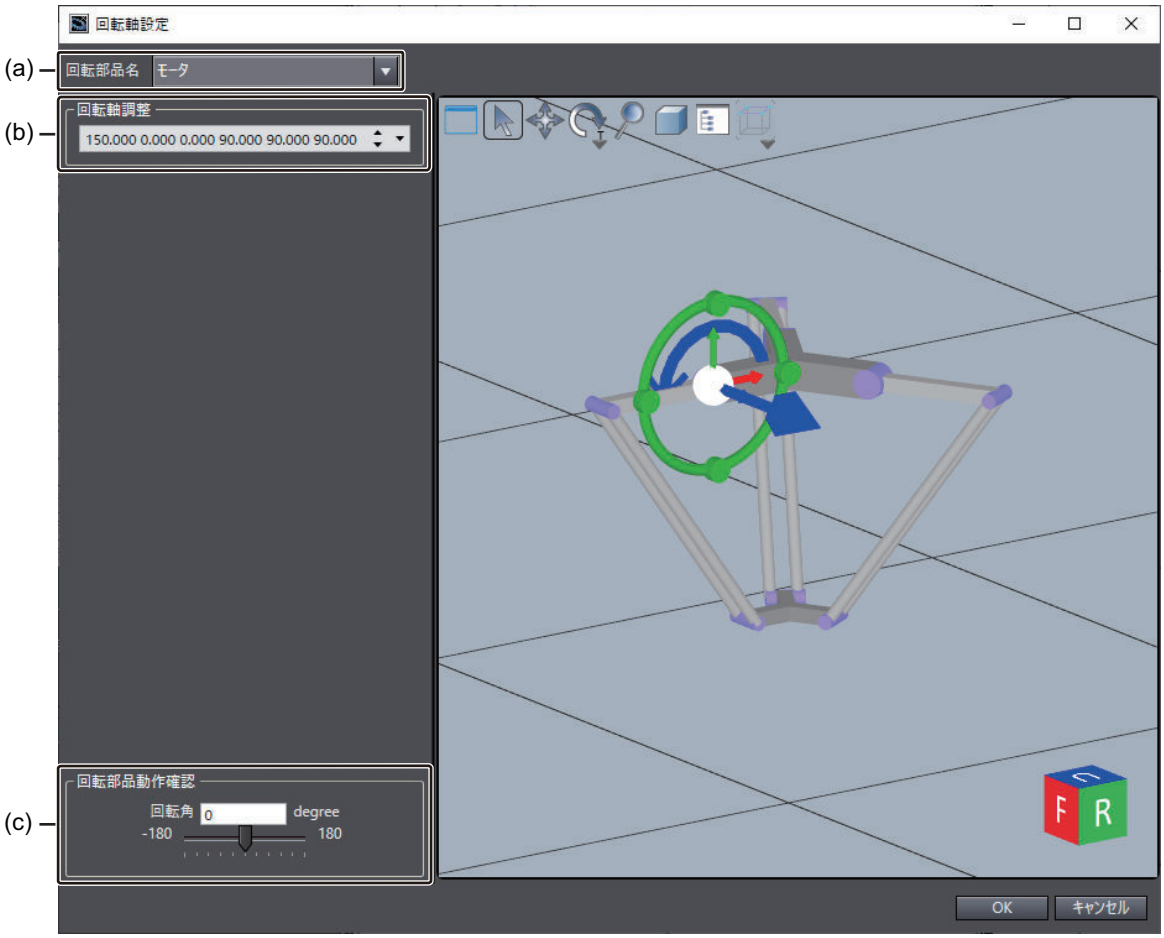


	設定項目		説明	設定値	初期値
(a)	CAD ファイルの更新		3D 形状データとして使用する CAD ファイルをインポートします。 詳細は「4-5-3 CAD データのインポート (P.4-86)」を参照してください。	-	-
(b)	3D 表示	表示	パラレルリンク機構の 3D 形状データを 3D 表示画面に表示する／しないを切り替えます。チェック ON で 3D 表示画面に表示されます。	チェック ON／チェック OFF	チェック ON
		衝突プログラム	パラレルリンク機構の衝突を検出したときに実行する C#プログラムを設定します。右側のボタンから、C#プログラムを選択します。C#プログラムに衝突した時刻と衝突した対象などを出力するプログラムを記述することで、衝突したときの情報を取得することができます。	C#プログラム名	なし
		透過度	3D 形状データの透過度を設定します。	なし／低／中／高	なし
(c)	ロケーション	親からのオフセット	パラレルリンク機構の 3D 形状データの位置と姿勢を設定します。親が設定されていない場合、設定する座標は 3D 表示画面のワールド座標系における座標です。親が設定されている場合、設定する座標は親の 3D 形状データのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	Z 以外: 0.000 Z: 730.000
		親	パラレルリンク機構の 3D 形状データの親となる 3D 形状データを設定します。	3D 形状データの名称	選択なし
		回転軸のオフセット	3D 形状データのローカル座標系の原点に対する回転軸のオフセットを設定します。オフセットを元に、回転軸の位置が変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	すべて 0.000
		TCP	パラレルリンク機構の 3D 形状データ上にロボットハンドなどのツールを取り付ける位置を示す点の座標です。3D 形状データにインポートした CAD データを使用している場合に設定可能です。右側のボタンから、座標の設定を行います。 設定方法の詳細は「TCP 設定 (P.4-83)」を参照してください。 設定する座標は、本パラレルリンク機構の可動フレームのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～ 180.000	可動フレームの座標値
(d)	詳細設定 – 衝突検知モデル	表示	衝突検知用モデルを表示するかどうかを設定します。	チェック ON／チェック OFF	チェック OFF
(e)	対象コントローラ		パラレルリンク機構を動作させる IEC プログラムが記載されているコントローラを選択します。ロボティクス命令を実行可能な形式のコントローラだけ選択可能です。	• 形 NJ501-R□□□ • 形 NJ501-4□□□（ユニット Ver.1.08 以降）	-

	設定項目		説明	設定値	初期値
(f)	回転軸設定		3D 形状データにインポートした CAD ファイルを使用している場合に、回転軸の中心位置設定を行います。右側のボタンから、回転軸の設定を行います。設定方法の詳細は、後述の「回転軸設定 (P.4-82)」を参照してください。	-	-
(g)	キネマティクスパラメータの設定		キネマティクスパラメータを設定します。対象コントローラが設定されているときに、右側のボタンから、各パラメータの設定を行います。設定方法の詳細は後述の「キネマティクスパラメータの設定 (P.4-82)」を参照してください。	-	-
(h)	ワークスペースのパラメータの設定		ワークスペースのパラメータを設定します。対象コントローラが設定されているときに、右側のボタンから、各パラメータの設定を行います。設定方法の詳細は後述の「ワークスペースのパラメータの設定 (P.4-83)」を参照してください。	-	-
(i)	対応変数の設定		軸座標系ごとに各軸の対応変数の設定を行います。それぞれ次の軸数分設定します。 Delta3: 3 軸 Delta3R: 4 軸 Delta2: 2 軸 対象コントローラが設定されているときに、右側のボタンから軸グループ変数を選択すると、各軸変数を一度に設定することができます。設定方法の詳細は後述の「対応変数の設定 (P.4-83)」を参照してください。	-	-
		対応変数	各軸の対応変数を設定します。対応する軸変数名を直接入力します。	-	-
		変換係数	[対応変数] で設定した軸変数の単位が degree 以外の場合に、degree に変換するときの係数を設定します。単位は [変数の単位／rev] です。	0.000～ 4294967295.000	360.000
		初期角度	各軸の初期角度です。リンク 1 の向きが XY 平面と平行でないときに設定します。シミュレータや実機での動作確認時、各軸の角度には、軸変数の値に初期角度を加算した値が反映されます。 Delta3R の回転軸系(A3)には本項目はありません。	-180.000～ +180.000	0.000
(j)	マウントポイント、リンクポイント設定エリア		パラレルリンク機構の 3D 形状データのマウントポイント、リンクポイントを設定するエリアです。マウントポイント、リンクポイントの設定方法の詳細については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。	-	-
(k)	3D 編集エリア		パラレルリンク機構の 3D 形状データの位置や姿勢、マウントポイント、リンクポイントを表示する画面です。3D 編集エリアでの表示や操作方法の詳細は、「第 5 章 3D 表示画面、3D 編集エリアの操作 (P.5-1)」を参照してください。	-	-

● 回転軸設定

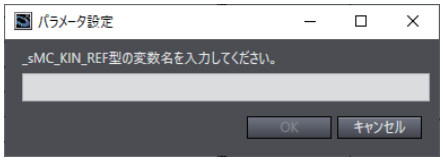
パラレルリンク機構のモータ回転部の中心位置を設定します。
［回転軸設定］の右側のボタンをクリックすると、［回転軸設定］ダイアログボックスが表示されます。



	設定項目	説明	設定値	初期値
(a)	回転部品名	回転部品名を表示します。Delta2、Delta3 の場合は「モータ」、Delta3R の場合は「モータ」または「回転軸」を選択することができます。	モータ 回転軸	モータ
(b)	回転軸調整	回転軸の位置を調整します。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.00	すべて 0.000
(c)	回転部品動作確認	回転部品の動作を確認します。スライダの位置に合わせて回転軸が回転します。	-180～+180	0

● キネマティクスパラメータの設定

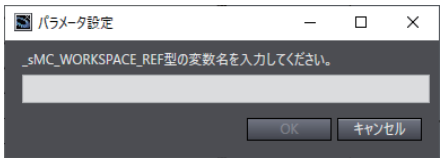
キネマティクスパラメータを設定します。
［キネマティクスパラメータの設定］の右側のボタンをクリックすると、［パラメータ設定］ダイアログボックスが表示されます。



設定値が格納されている「_sMC_KIN_REF 型」変数の変数名を入力し、[OK] ボタンをクリックすると、「_sMC_KIN_REF 型」変数のメンバ「KinParam」および「ExpansionParam」に設定されているキネマティクスパラメータが反映されます。
キネマティクスパラメータの詳細は『NJ シリーズ NJ Robotics CPU ユニット ユーザーズマニュアル(SBCA-421)』を参照してください。

● ワークスペースのパラメータの設定

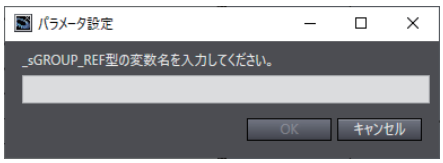
ワークスペースのパラメータを設定します。
[ワークスペースのパラメータの設定] の右側のボタンをクリックすると、[パラメータ設定] ダイアログボックスが表示されます。



設定値が格納されている「sMC_WORKSPACE_REF 型」変数の変数名を入力し、[OK] ボタンをクリックすると、「sMC_WORKSPACE_REF 型」変数のメンバ「WorkspaceParam」に設定されているワークスペースのパラメータが反映されます。
ワークスペースのパラメータの詳細は『NJ シリーズ NJ Robotics CPU ユニット ユーザーズマニュアル(SBCA-421)』を参照してください。

● 対応変数の設定

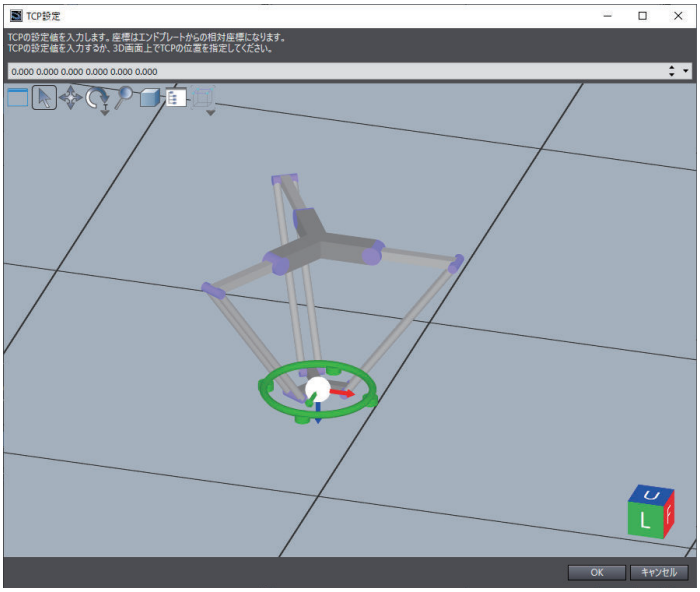
対応する軸変数を一度に割り付けます。
[対応変数の設定] の右側のボタンをクリックすると、[パラメータ設定] ダイアログボックスが表示されます。



対応する軸グループ変数名を入力し、[OK] ボタンをクリックすると、各軸変数が一度に割り付けられます。
軸変数、軸グループ変数の詳細は『NJ シリーズ NJ Robotics CPU ユニット ユーザーズマニュアル(SBCA-421)』を参照してください。

● TCP 設定

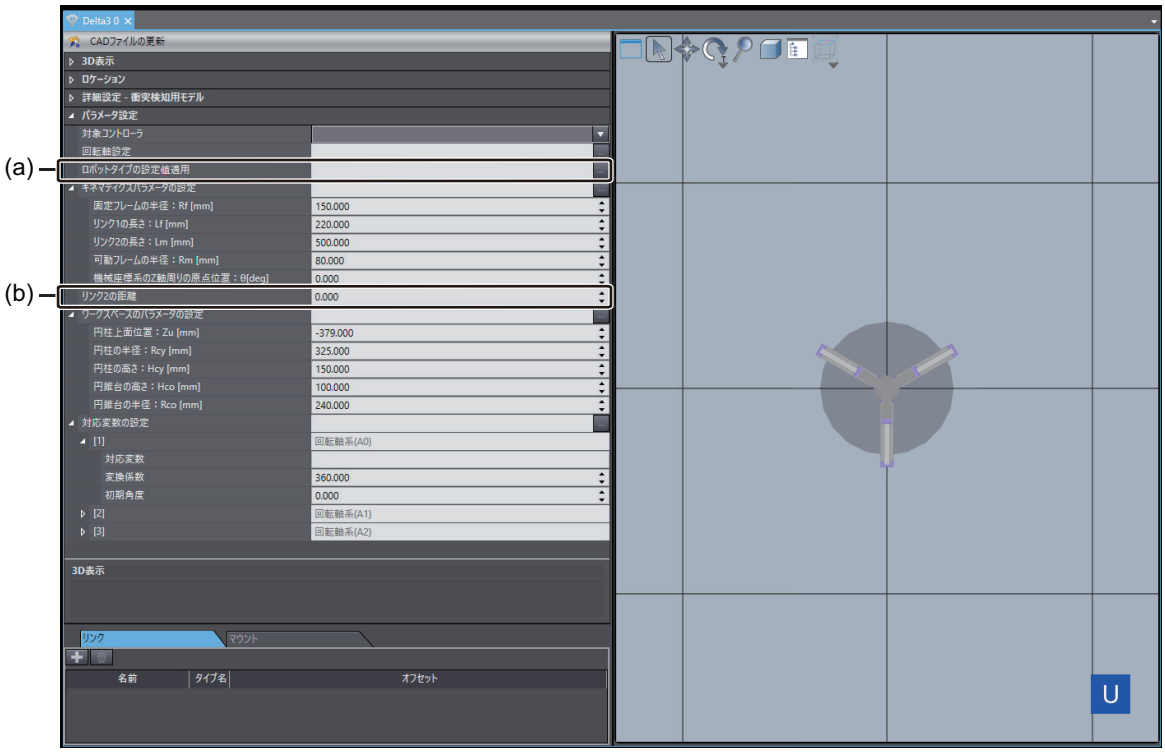
TCP の座標を設定します。
[TCP 設定] の右側のボタンをクリックすると、[TCP 設定] ダイアログボックスが表示されます。



座標値を入力するか、画面内の TCP をドラッグして位置を指定します。TCP の設定値は左から順に X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。

Delta3 の設定項目

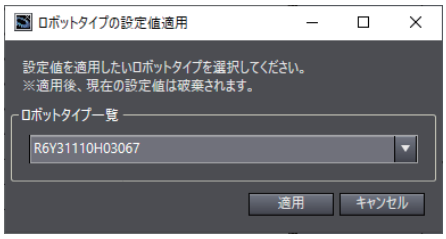
Delta3 固有のパラメータ設定の項目は以下のとおりです。



	設定項目	説明	設定値	初期値
(a)	ロボットタイプの設定値適用	キネマティクスパラメータと、ワークスペースのパラメータを自動的に設定します。右側のボタンから、設定を行います。設定方法の詳細は、後述の「ロボットタイプの設定値適用 (P.4-85)」を参照してください。	-	-
(b)	リンク 2 の距離	リンク 2 を結ぶ部品の両端にあるボールジョイントのボール部分の中心間の距離です。	0.000-1000.000	0.000

● ロボットタイプの設定値適用

ロボットタイプを選択します。
［ロボットタイプの設定値適用］の右側のボタンをクリックすると、［ロボットタイプの設定値適用］ダイアログボックスが表示されます。
設定値を適用したいロボットタイプを選択し、［適用］ボタンをクリックします。

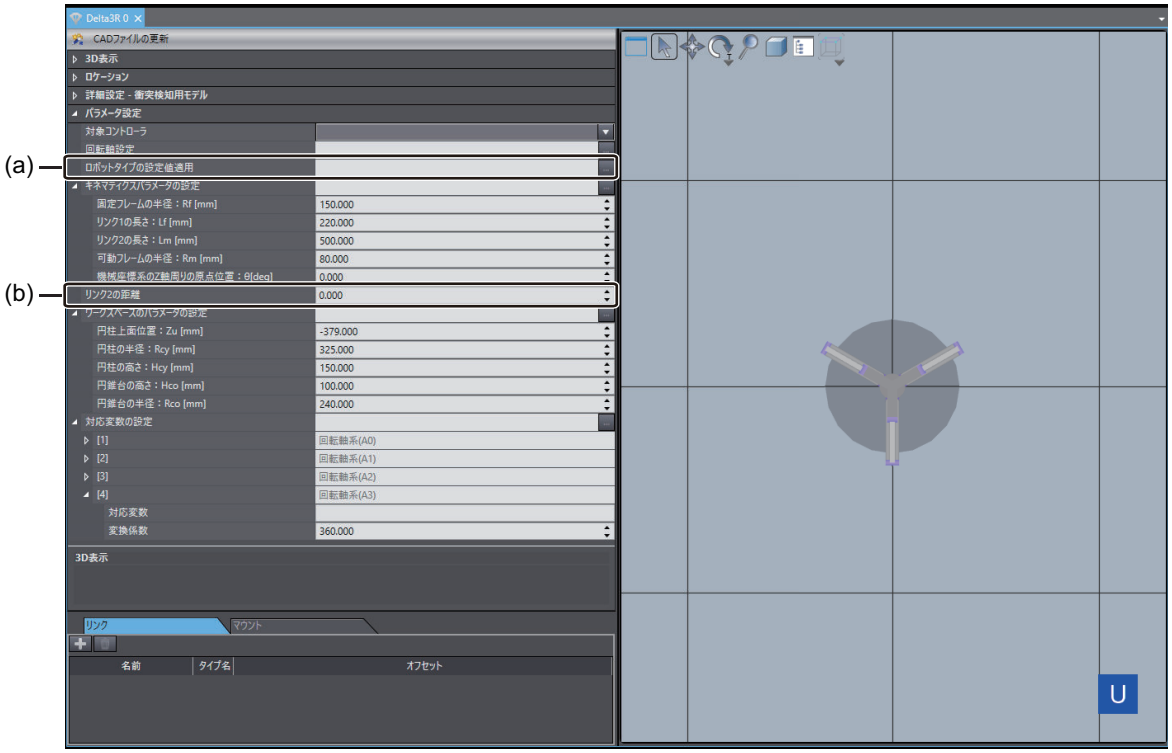


ロボットタイプ一覧に表示される項目は以下のとおりです。

ロボットタイプ	説明
R6Y31110H03067	中型パラレルリンクロボット（稼働範囲 φ1100mm、可搬重量 3kg、θ 軸付き高慣性タイプ）
R6Y31110L03067	中型パラレルリンクロボット（稼働範囲 φ1100mm、可搬重量 3kg、θ 軸付き低慣性タイプ）
R6Y30110S03067	中型パラレルリンクロボット（稼働範囲 φ1100mm、可搬重量 3kg）
R6Y31065H02067	小型パラレルリンクロボット（稼働範囲 φ650mm、可搬重量 2kg、θ 軸付き高慣性タイプ）
R6Y31065L02067	小型パラレルリンクロボット（稼働範囲 φ650mm、可搬重量 2kg、θ 軸付き低慣性タイプ）
R6Y30065S02067	小型パラレルリンクロボット（稼働範囲 φ650mm、可搬重量 2kg）

Delta3R の設定項目

Delta3R 固有のパラメータ設定の項目は以下のとおりです。



	設定項目	説明	設定値	初期値
(a)	ロボットタイプの設定値適用	キネマティクスパラメータと、ワークスペースのパラメータを自動的に設定します。右側のボタンから、設定を行います。設定方法の詳細は、Delta3 の「ロボットタイプの設定値適用 (P.4-85)」を参照してください。	-	-
(b)	リンク 2 の距離	リンク 2 を結ぶ部品の両端にあるボールジョイントのボール部分の中心間の距離です。	0.000-1000.000	0.000

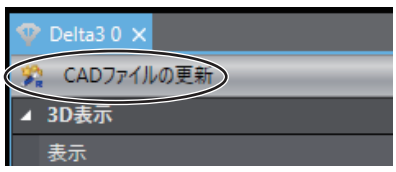
4-5-3 CAD データのインポート

パラレルリンク機構の 3D 形状データとして CAD データを使用する場合、登録済みのパラレルリンク機構に CAD データファイルをインポートします。

参考

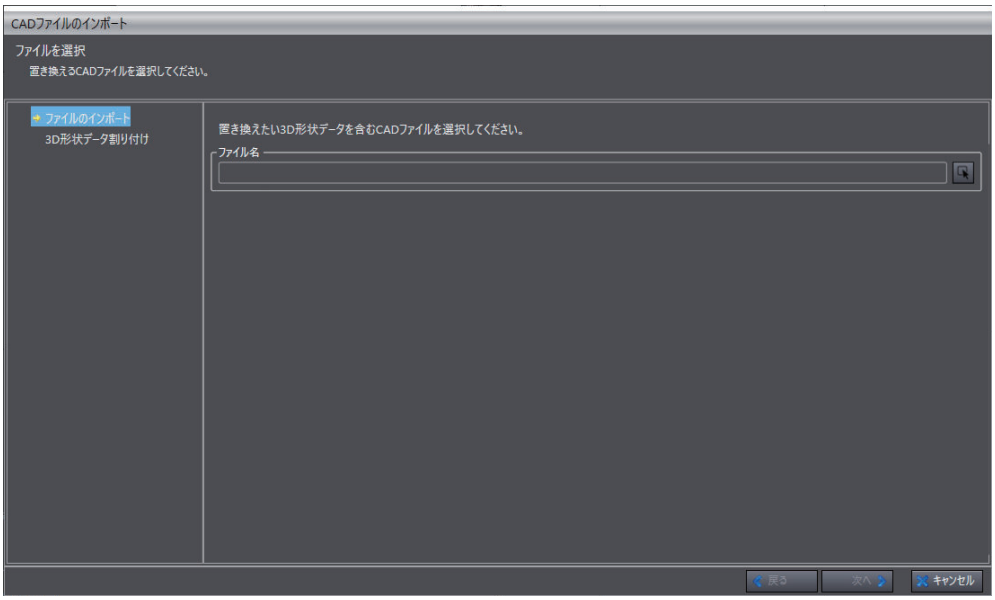
インポート時 CAD データのメッシュの粗さをオプションで設定することができます。オプション設定の詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

1 **対象のパラレルリンク機構の設定画面で、[CAD ファイルの更新] ボタンをクリックします。**



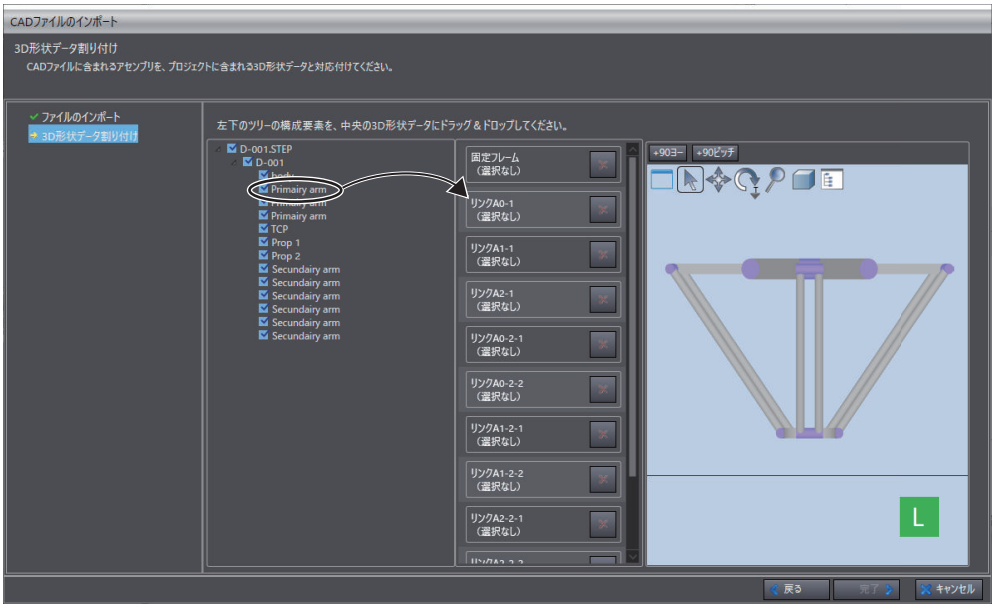
ウィザードが起動し、[ファイルを選択] 画面が表示されます。

2 **ファイル選択ボタンからインポートする CAD データファイルを選択し、[次へ] ボタンをクリックします。**



[3D 形状データ割り付け] 画面が表示されます。

3 **CAD データの構成要素をドラッグし、割り付け対象の 3D 形状データの項目の上にドロップします。**



対象の 3D 形状データに CAD データが割り付けられます。
3D 形状データは、パラレルリンク機構を構成するリンク部品ごとに分けられています。

リンク部品名	Delta3	Delta3R	Delta2
固定フレーム	○	○	○
リンク A0-1	○	○	○
リンク A1-1	○	○	○
リンク A2-1	○	○	-
リンク A0-2-1	○	○	○
リンク A0-2-2	○	○	○

リンク部品名	Delta3	Delta3R	Delta2
リンク A1-2-1	○	○	○
リンク A1-2-2	○	○	○
リンク A2-2-1	○	○	-
リンク A2-2-2	○	○	-
可動フレーム	○	○	○
回転軸リンク A3-1	-	○	-
回転軸リンク A3-2	-	○	-
回転エンドエフェクタ	-	○	-

リンク部品の詳細については『NJ シリーズ NJ Robotics CPU ユニット ユーザーズマニュアル(SBCA-421)』を参照してください。
すべての 3D 形状データへ CAD データの割り付けを行い、［完了］ボタンをクリックします。



参考

次のリンク部品は、CAD データの割り付けを省略することができます。
- 固定フレーム
- リンク 2-A-2
- リンク 2-B-2
- リンク 2-C-2（Delta3、Delta3R のみ）

［固定フレーム］へ CAD データの割り付けを行わなかった場合は、ベース部分は 3D 表示画面に表示されません。

4-6 ワークの 3D 形状データの追加

ワークの 3D 形状データを追加します。ワークの 3D 形状データとして、CAD データ、直方体、円柱を使用することができます。

4-6-1 CAD データのインポート

ワークの 3D 形状データとして使用する CAD データを 3DCAD データファイルから読み込みます。

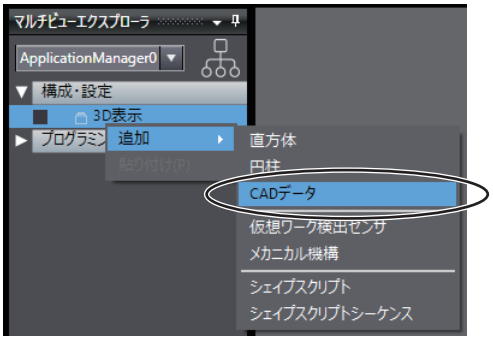
CAD データのインポート手順

ワークの 3D 形状データとして、CAD データをアプリケーションマネージャにインポートします。インポート手順は以下のとおりです。

参考

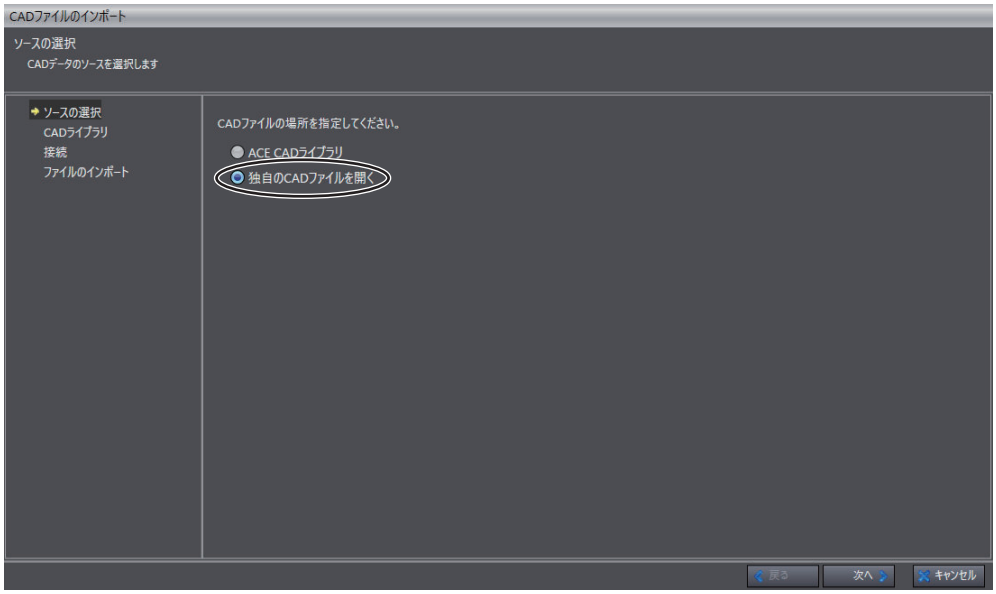
インポート時 CAD データのメッシュの粗さをオプションで設定することができます。オプション設定の詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

1 **［構成・設定］の［3D 表示］を右クリックし、メニューから［追加］-［CAD データ］を選択します。**



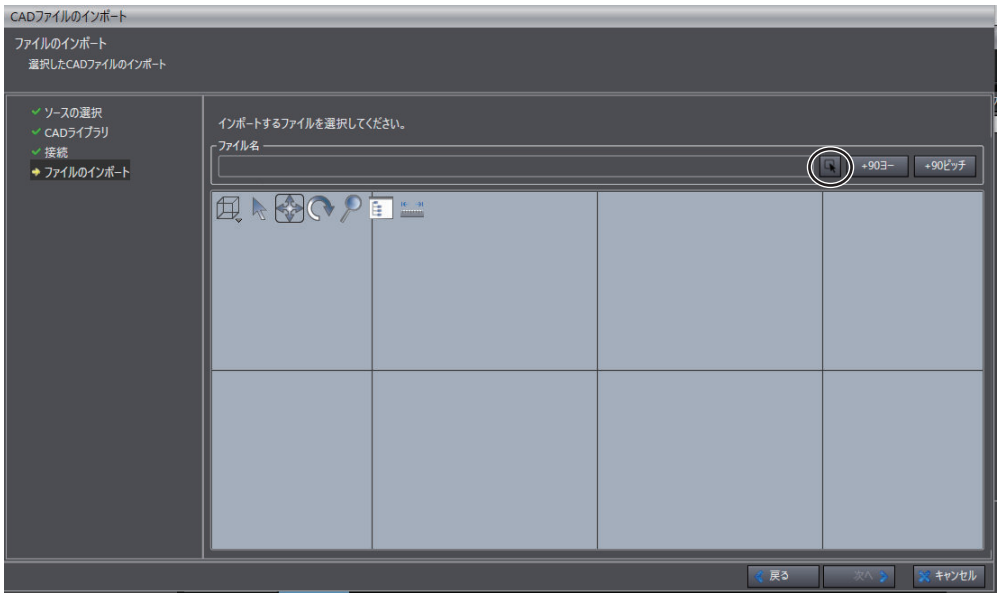
［CAD ファイルのインポート］ウィザードが表示され、［ソースの選択］画面が表示されます。

2 **［ソースの選択］画面で［独自の CAD ファイルを開く］を選択し、［次へ］ボタンをクリックします。**



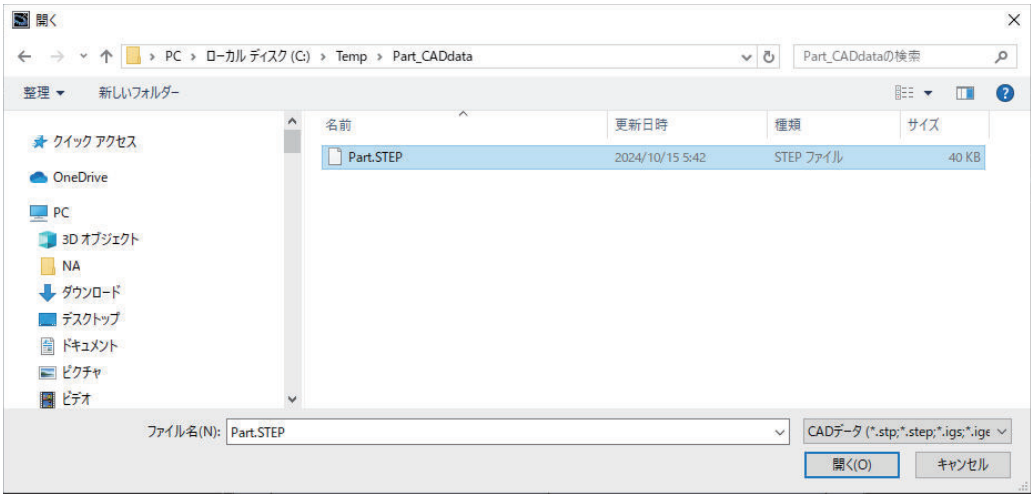
［ファイルのインポート］画面が表示されます。

3 **［ファイルのインポート］画面の［ファイルを開く］ボタンをクリックします。**



［開く］ダイアログボックスが表示されます。

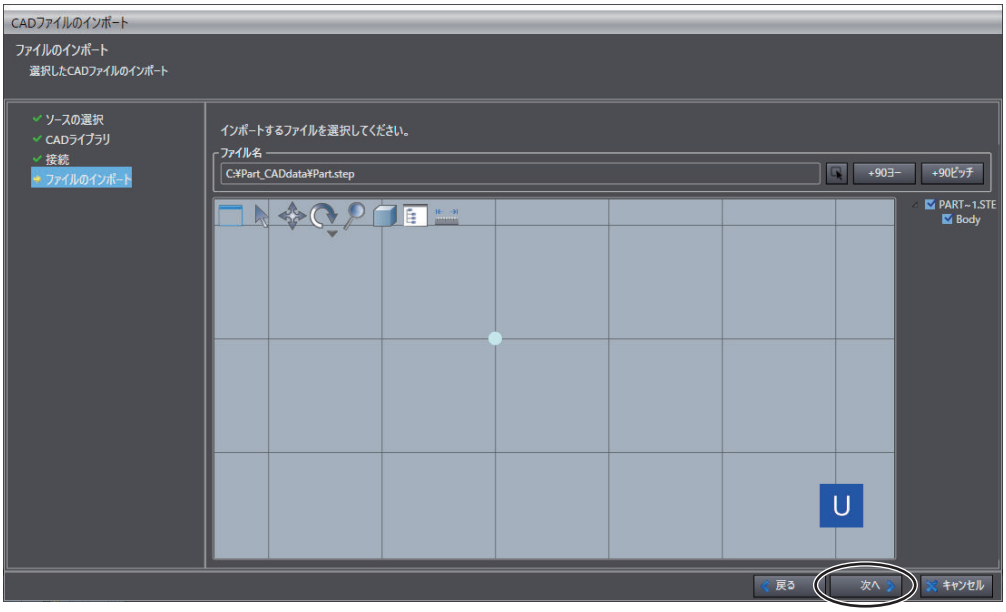
4 **拡張子.stp、.step、.igs、.iges、.usd、.usda、.usdc、.usdz のいずれかの CAD ファイルを選択し、［開く］ボタンをクリックします。**



CAD ファイルがインポートされます。
必要に応じて、［+ 90 ヨー］ボタンまたは［+ 90 ピッチ］ボタンをクリックして、インポートする CAD データの向きを変更します。

ボタン	説明
+ 90 ヨー	CAD データを、CAD データの座標系の Z 軸まわりに+ 90° 回転させます。このとき CAD データの座標系も一緒に回転します。
+ 90 ピッチ	CAD データを、CAD データの座標系の Y 軸まわりに+ 90° 回転させます。このとき CAD データの座標系も一緒に回転します。

5 **［次へ］ボタンをクリックします。**

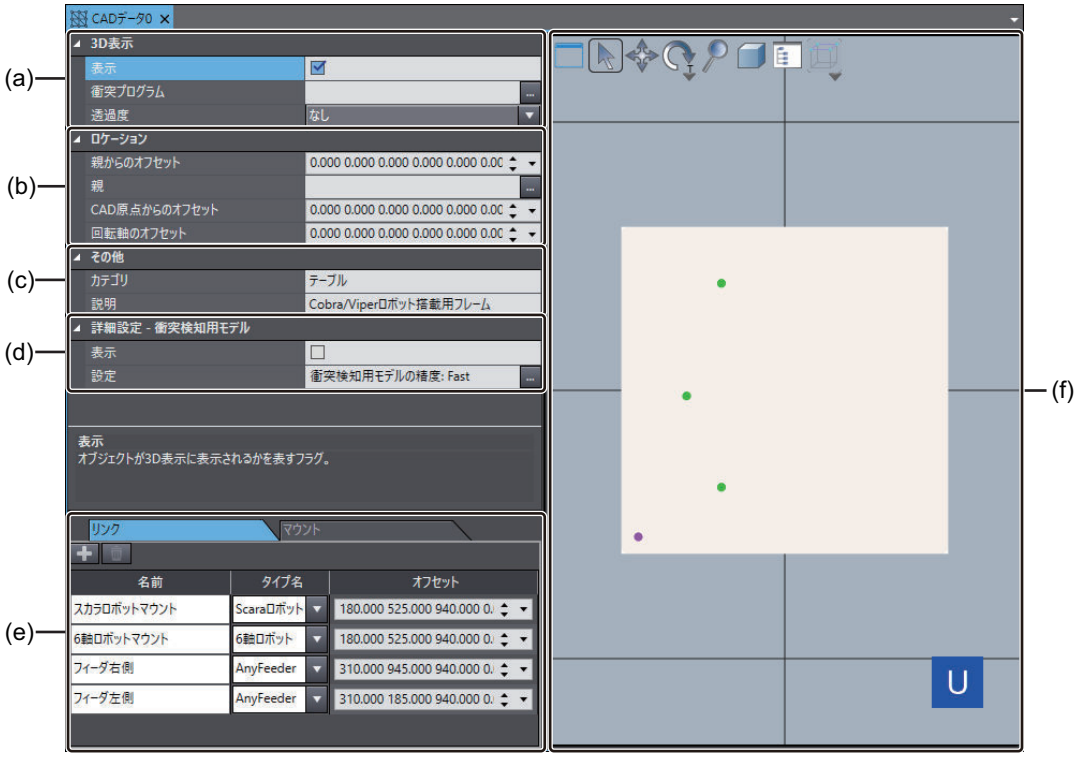


マルチビューエクスプローラの［3D 表示］項目の下にインポートした CAD データが追加されます。



CAD データの設定項目

インポートした CAD データをダブルクリックすると、CAD データの設定画面が表示されます。
CAD データの配置、マウントポイント／リンクポイントを設定します。
CAD データの設定項目は次のとおりです。



	設定項目		説明	設定値	初期値
(a)	3D 表示	表示	この 3D 形状データを 3D 表示画面に表示する／しないを切り替えます。チェック ON で 3D 表示画面に表示されます。	チェック ON／チェック OFF	チェック ON
		衝突プログラム	CAD データの衝突を検出したときに実行する C#プログラムを設定します。右側のボタンから、C#プログラムを選択します。C#プログラムに衝突した時刻と衝突した対象などを出力するプログラムを記述することで、衝突したときの情報を取得することができます。	C#プログラム名	なし
		透過度	3D 形状データの透過度を設定します。	なし／低／中／高	なし
(b)	ロケーション	親	この 3D 形状データの親となる 3D 形状データを設定します。右側のボタンから、親として設定する 3D 形状データを選択します。	3D 形状データの名称	なし

	設定項目		説明	設定値	初期値
		親からのオフセット	この 3D 形状データの位置と姿勢を設定します。 親が設定されていない場合、設定する座標は 3D 表示画面のワールド座標系における座標です。親が設定されている場合、設定する座標は親の 3D 形状データのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.000
		回転軸のオフセット	この 3D 形状データのローカル座標系の原点に対する回転軸のオフセットを設定します。 オフセットを元に、回転軸の位置が変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.000
		CAD の原点からのオフセット	この 3D 形状データのローカル座標系原点に対する CAD の原点からのオフセットを設定します。これにより、3DCAD ソフトウェアで設定された原点の位置を補正することができます。 オフセットを元に、3D 形状データの原点位置が変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.000
(c)	その他	カテゴリ／説明	必要に応じて、この 3D 形状データに関する説明を記述します。	文字列	なし
(d)	詳細設定 - 衝突検知用モデル	表示	衝突検知用モデルを表示するかどうかを設定します。	チェック ON／チェック OFF	チェック OFF
		設定	衝突検知用モデルの設定を行います。右側のボタンから、衝突検知用モデルの詳細な設定を行います。設定方法の詳細は「衝突検知用モデルの詳細設定 (P.8-25)」を参照してください。	-	衝突検知用モデルの簡略化: True | 分割して衝突検知用モデルを生成: False
(e)	マウントポイント、リンクポイント設定エリア		この 3D 形状データのマウントポイント、リンクポイントを設定するエリアです。マウントポイント、リンクポイントの設定方法の詳細については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。	-	-
(f)	3D 編集エリア		3D 形状データの位置や姿勢、マウントポイント、リンクポイントを表示する画面です。3D 編集エリアでの表示や操作方法の詳細は、「第 5 章 3D 表示画面、3D 編集エリアの操作 (P.5-1)」を参照してください。	-	-

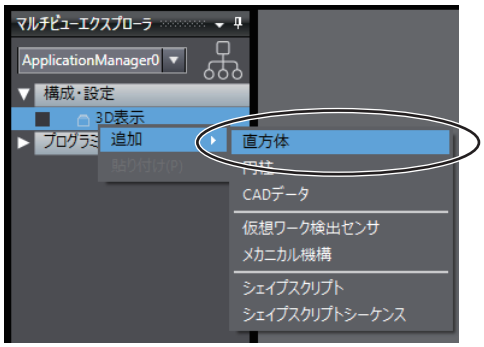
4-6-2 直方体、円柱の追加

直方体、円柱の追加手順を説明します。

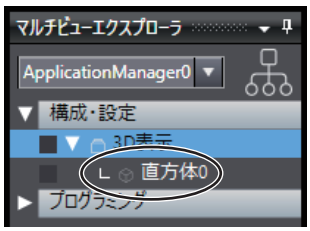
直方体、円柱の追加手順

直方体、円柱の追加手順は以下のとおりです。

1 **［構成・設定］の［3D 表示］を右クリックし、メニューから［追加］-［直方体］または［円柱］を選択します。**

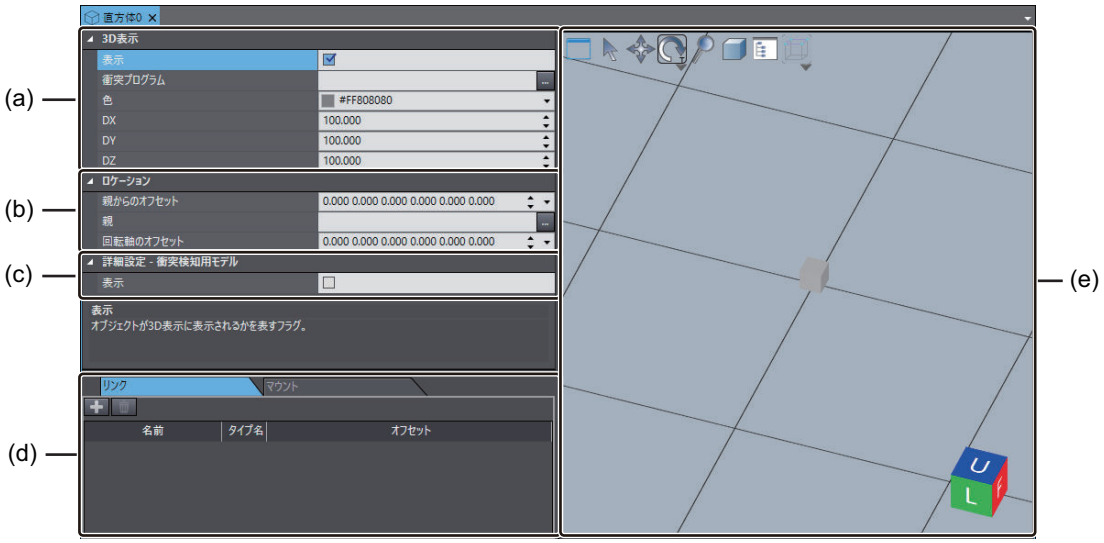


マルチビューエクスプローラの［3D 表示］項目の下に直方体、または円柱が追加されます。



直方体の設定項目

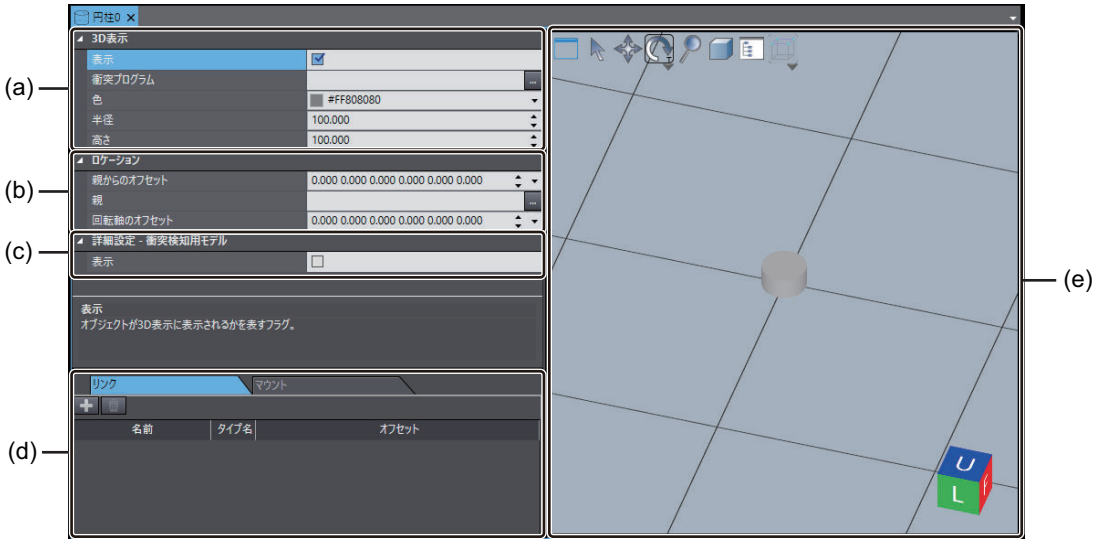
直方体のサイズ、配置、マウントポイント／リンクポイントを設定します。
直方体の 3D 形状データの原点は、直方体の底面の重心の位置に設定されます。
直方体の設定項目は次のとおりです。



	設定項目		説明	設定値	初期値
(a)	3D 表示	表示	直方体を 3D 表示画面に表示する／しないを切り替えます。チェック ON で 3D 表示画面に表示されます。	チェック ON／チェック OFF	チェック ON
		色	直方体の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#FF808080
		DX	直方体の X 軸方向の長さを設定します。（単位: mm）	0～5,000	100
		DY	直方体の Y 軸方向の長さを設定します。（単位: mm）	0～5,000	100
		DZ	直方体の Z 軸方向の長さを設定します。（単位: mm）	0～5,000	100
(b)	ロケーション	親からのオフセット	直方体の位置と姿勢を設定します。 親が設定されていない場合、設定する座標は 3D 表示画面のワールド座標系における座標です。親が設定されている場合、設定する座標は親の 3D 形状データのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.00
		親	直方体の親となる 3D 形状データを設定します。 右側のボタンから、親として設定する 3D 形状データを選択します。	3D 形状データの名称	なし
		回転軸のオフセット	直方体のローカル座標系の原点に対する回転軸のオフセットを設定します。 オフセットを元に、回転軸の位置が変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.00
(c)	詳細設定 - 衝突検知用モデル	表示	衝突検知用モデルを表示するかどうかを設定します。	チェック ON／チェック OFF	チェック OFF
(d)	マウントポイント、リンクポイント設定エリア		直方体のマウントポイント、リンクポイントを設定するエリアです。マウントポイント、リンクポイントの設定方法の詳細については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。	-	-
(e)	3D 編集エリア		直方体の位置や姿勢、マウントポイント、リンクポイントを表示する画面です。3D 編集エリアでの表示や操作方法の詳細は、「第 5 章 3D 表示画面、3D 編集エリアの操作 (P.5-1)」を参照してください。	-	-

円柱の設定項目

円柱のサイズ、配置、マウントポイント／リンクポイントを設定します。
円柱の 3D 形状データの原点は、円柱の底面の中心の位置に設定されます。
円柱の設定項目は次のとおりです。



	設定項目		説明	設定値	初期値
(a)	3D 表示	表示	円柱を 3D 表示画面に表示する／しないを切り替えます。チェック ON で 3D 表示画面に表示されます。	チェック ON／チェック OFF	チェック ON
		色	円柱の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#FF808080
		半径	円柱の半径を設定します。（単位: mm）	0～5,000	100
		高さ	円柱の高さを設定します。（単位: mm）	0～5,000	100
(b)	ロケーション	親からのオフセット	円柱の位置と姿勢を設定します。 親が設定されていない場合、設定する座標は 3D 表示画面のワールド座標系における座標です。親が設定されている場合、設定する座標は親の 3D 形状データのローカル座標系における座標です。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0.00
		親	円柱の親となる 3D 形状データを設定します。右側のボタンから、親として設定する 3D 形状データを選択します。	文字列	なし
		回転軸のオフセット	円柱のローカル座標系の原点に対する回転軸のオフセットを設定します。 オフセットを元に、回転軸の位置が変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	すべて 0
(c)	詳細設定 - 衝突検知用モデル	表示	衝突検知用モデルを表示するかどうかを設定します。	チェック ON／チェック OFF	チェック OFF
(d)	マウントポイント、リンクポイント設定エリア		円柱のマウントポイント、リンクポイントを設定するエリアです。マウントポイント、リンクポイントの設定方法の詳細については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。	-	-

	設定項目	説明	設定値	初期値
(e)	3D 編集エリア	円柱の位置や姿勢、マウントポイント、リンクポイントを表示する画面です。3D 編集エリアでの表示や操作方法の詳細は、「第 5 章 3D 表示画面、3D 編集エリアの操作 (P.5-1)」を参照してください。	-	-

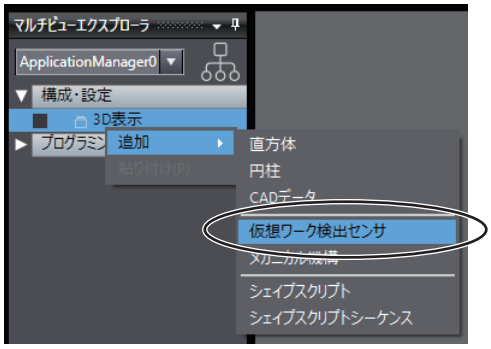
4-7　ワーク検出センサの追加

3D シミュレーションにおいてワークの位置を検出するための仮想的なセンサである「仮想ワーク検出センサ」を追加します。装置の所定の位置に仮想ワーク検出センサを設置することでワークの移動を検出できるため、ワークを検出するデバッグ用プログラムの作成が不要になります。

4-7-1　仮想ワーク検出センサの追加

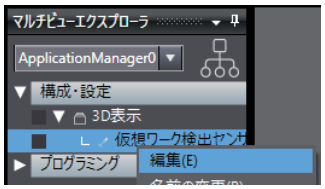
アプリケーションマネージャに仮想ワーク検出センサを追加します。

1　マルチビューエクスプローラの［構成・設定］から［3D 表示］を右クリックし、メニューから［追加］-［仮想ワーク検出センサ］を選択します。



仮想ワーク検出センサが追加されます。

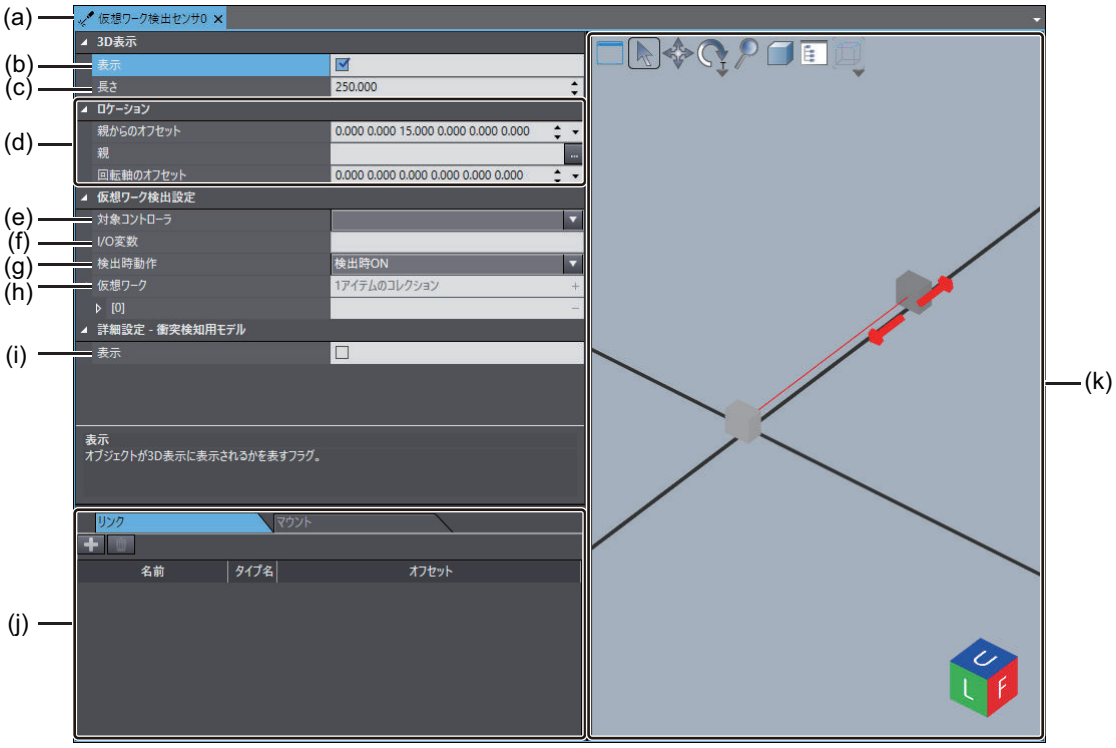
2　仮想ワーク検出センサを設定するには、マルチビューエクスプローラから対象の仮想ワーク検出センサをダブルクリックするか、または右クリックして［編集］を選択します。



仮想ワーク検出センサの設定画面が表示されます。

4-7-2　仮想ワーク検出センサの設定項目

仮想ワーク検出センサのサイズ、配置、ワーク検出時の動作、マウントポイント／リンクポイントを設定します。
仮想ワーク検出センサの設定画面で設定できる項目について説明します。



	項目		説明	設定値	初期値
(a)	仮想ワーク検出センサ名		仮想ワーク検出センサ名を表示します。	-	-
(b)	表示		仮想ワーク検出センサを 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック ON
(c)	長さ		仮想ワーク検出センサの長さを設定します。 太さは変更できません。	0.001～8,000mm	250mm
(d)	ロケーション		ロケーションの設定を行います。 設定内容は CAD データ、直方体、円柱と同じです。	-	-
(e)	仮想ワーク検出設定	対象コントローラ	ワークを検出するコントローラを選択します。 Sysmac Studio V1.42 以降、コントローラに加えて、ロボットサブデバイスの選択が可能です。	プロジェクトに登録されているコントローラ	なし
(f)		I/O 変数	仮想ワーク検出センサがワークを検出したときに ON/OFF を切り替える I/O の変数名です。 対象コントローラの BOOL 型のグローバル変数を設定します。 対象がロボットサブデバイスの場合は V+デジタル I/O を設定します。	文字列	なし
(g)		検出時動作	仮想ワーク検出センサがワークを検出したときの I/O 変数の動作を設定します。 検出した場合に TRUE にするときは「検出時 ON」、FALSE にするときは「検出時 OFF」を設定します。	検出時 ON／検出時 OFF	検出時 ON
(h)		仮想ワーク	仮想ワーク検出センサが検出する対象のワーク名を設定します。 設定できるワーク名は、同じアプリケーションマネージャに含まれるワーク名です。	文字列	なし
(i)	詳細設定 – 衝突検知用モデル	表示	衝突検知用モデルを表示するかどうかを設定します。	チェック ON／チェック OFF	チェック OFF

	項目	説明	設定値	初期値
(j)	マウントポイント、リンクポイント設定エリア	仮想ワーク検出センサのマウントポイント、リンクポイントを設定します。 マウントポイント、リンクポイントの設定方法の詳細については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。	-	-
(k)	3D 編集エリア	仮想ワーク検出センサの位置や姿勢、マウントポイント、リンクポイントを表示する画面です。3D 編集エリアでの表示や操作方法の詳細は、「第 5 章 3D 表示画面、3D 編集エリアの操作 (P.5-1)」を参照してください。	-	-

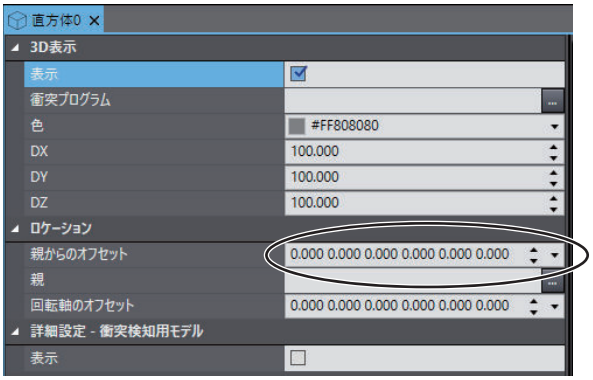
4-8　3D 形状データの配置方法

3D 形状データの配置を設定する方法には、次の方法があります。
- 座標の直接入力
- 3D 表示画面上でのドラッグ&ドロップ
- リンクポイントの位置へのスナップ

4-8-1　座標の直接入力

1　3D 形状データの設定画面を表示します。

2　［親からのオフセット］で、直交座標空間内の位置と姿勢を決める 6 つのロケーションの要素（X、Y、Z、y、p、r）を直接入力します。



4-8-2　3D 表示画面上でのドラッグ&ドロップ

3D 表示画面上で 3D 形状データをドラッグして目的の位置へドロップします。詳細は、「5-3-1 3D 形状データの移動 (P.5-19)」を参照してください。

4-8-3　リンクポイントへのスナップ

ワークをパレット上の指定した位置に配置する場合など、3D 形状データが他の 3D 形状データの指定した箇所と接触する場合にこの方法を使用します。
設定方法や操作方法については、「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」を参照してください。

4-9 3D 表示オブジェクトのエクスポート／インポート

作成した 3D 表示オブジェクトをエクスポートし、別のプロジェクトで再利用することができます。これにより 3D 表示オブジェクトの再設定が不要になり、設計、立ち上げ時間を短縮することができます。

4-9-1 エクスポート／インポート可能な 3D 表示オブジェクトの種類

エクスポート／インポート可能な 3D 表示オブジェクトは、アプリケーションマネージャのバージョンによって異なります。エクスポート／インポート可能な 3D 表示オブジェクトの種類およびアプリケーションマネージャのバージョンは、以下のとおりです。

3D 表示オブジェクトの種類	アプリケーションマネージャのバージョン
デフォルトファンクション	アプリケーションマネージャ Ver.5.0 以降でサポートしています。
ユーザファンクション	アプリケーションマネージャ Ver.5.0 以降でサポートしています。
直方体	すべてのバージョン
円柱	すべてのバージョン
CAD データ	すべてのバージョン
仮想ワーク検出センサ	すべてのバージョン
メカニカル機構	コンベア以外: すべてのバージョン コンベア: アプリケーションマネージャ Ver.5.0 以降でサポートしています。
カスタムメカニカル機構	アプリケーションマネージャ Ver.4.0 以降でサポートしています。
パラレルリンク機構	アプリケーションマネージャ Ver.4.0 以降でサポートしています。
シェイプスクリプト	すべてのバージョン
シェイプスクリプトシーケンス	すべてのバージョン

インポート先のアプリケーションマネージャのバージョンが上記に示すバージョンよりも古い場合、インポート処理は中断されます。

4-9-2 エクスポート／インポート操作可能なアクセスレベル

エクスポート／インポート操作は、ロボットシステム操作権限認証のアクセスレベルが設計技術者(Engineer)の場合、実行できます。ロボットシステム操作権限認証のアクセスレベルについては、『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

4-9-3 3D 表示オブジェクトのエクスポート

3D 表示オブジェクトをエクスポートする手順を説明します。

エクスポート対象のデータ

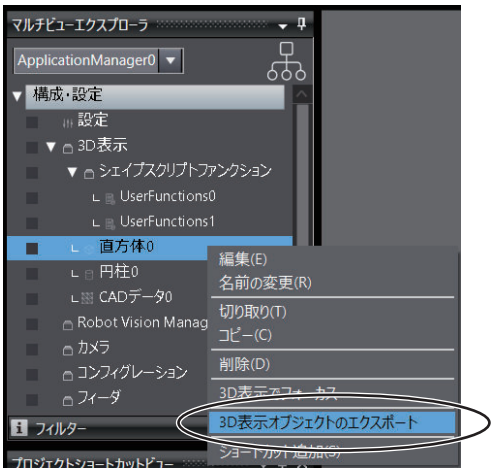
3D 表示オブジェクトのうち、下記参照情報を除くすべてのデータがエクスポートの対象となります。

- 各 3D 表示オブジェクトの親
- 仮想ワーク検出センサの仮想ワーク
- シェイプスクリプトシーケンスのシェイプスクリプト

エクスポート手順

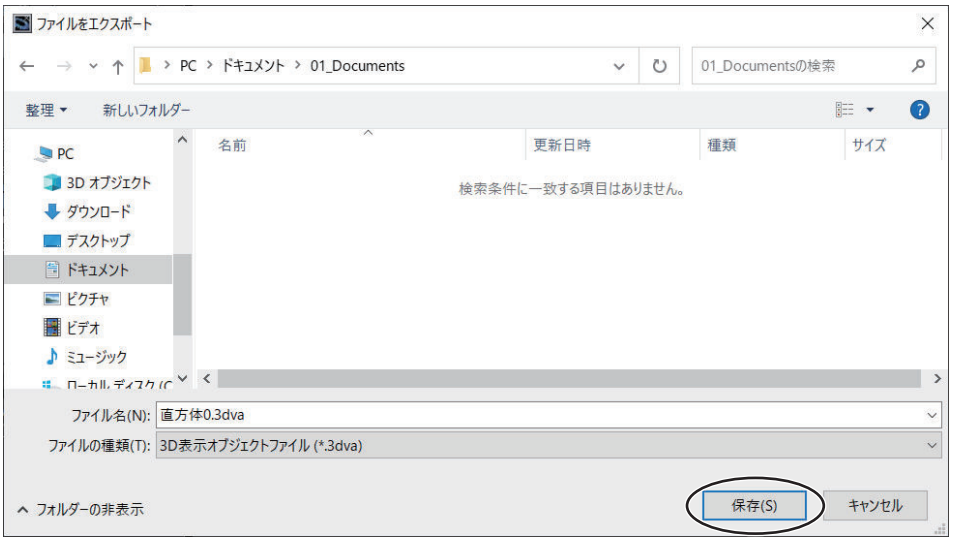
ここでは直方体を例に説明します。

1 **マルチビューエクスプローラの［構成・設定］-［3D 表示］項目下の「直方体 0」を右クリックし、メニューから［3D 表示オブジェクトのエクスポート］をクリックします。**



［ファイルをエクスポート］ダイアログボックスが表示されます。

2 **ファイル名を入力し、［保存］ボタンをクリックします。**



エクスポート対象の 3D 表示オブジェクトが、3dva 形式ファイルに保存されます。

4-9-4 3D 表示オブジェクトの一括エクスポート

3D 表示オブジェクトを一括エクスポートする手順を説明します。

エクスポート対象のデータ

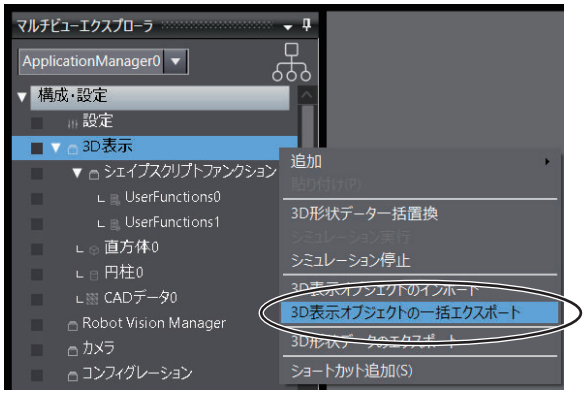
3D 表示オブジェクトのすべてのデータがエクスポートの対象となります。ただし、各 3D 表示オブジェクトが参照している下記のデータを併せて選択していない場合、下記に関する参照情報はエクスポートの対象外となります。

- 各 3D 表示オブジェクトの親
- 仮想ワーク検出センサの仮想ワーク
- シェイプスクリプトシーケンスのシェイプスクリプト

エクスポート対象外となった参照情報は、インポート後の 3D 表示オブジェクトでは設定値が空になります。

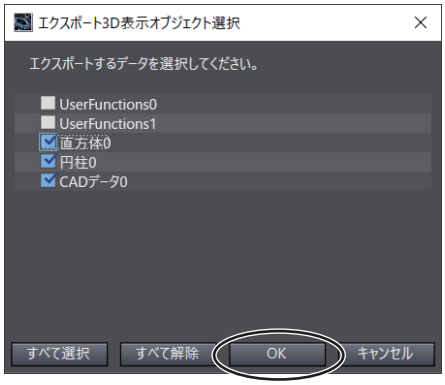
一括エクスポート手順

1 **マルチビューエクスプローラの［構成・設定］から［3D 表示］を右クリックし、メニューから［3D 表示オブジェクトの一括エクスポート］をクリックします。**



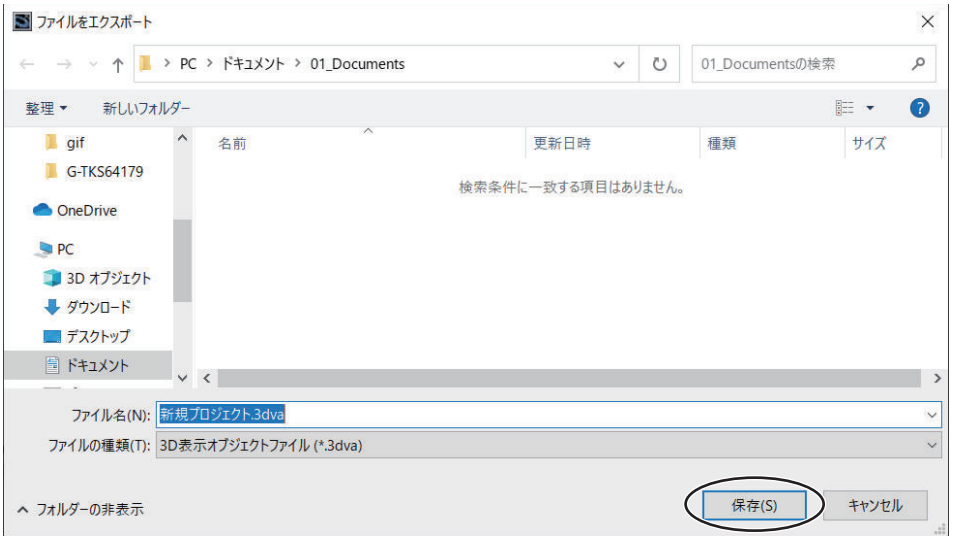
［エクスポート 3D 表示オブジェクト選択］ダイアログボックスが表示されます。

2 **エクスポートする 3D 表示オブジェクトのチェックボックスにチェックを入れ、［OK］ボタンをクリックします。**



［ファイルをエクスポート］ダイアログボックスが表示されます。

3 **ファイル名を入力し、［保存］ボタンをクリックします。**

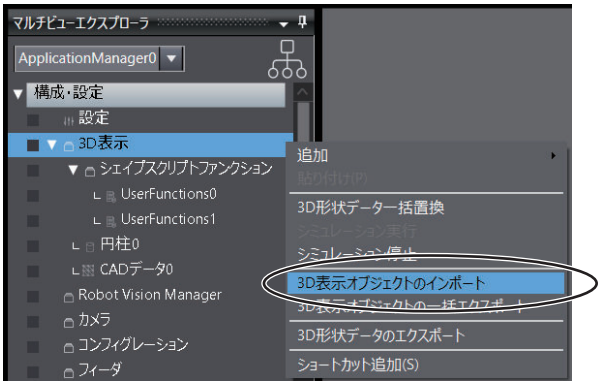


エクスポート対象の 3D 表示オブジェクトが、3dva 形式ファイルに保存されます。

4-9-5 3D 表示オブジェクトのインポート手順

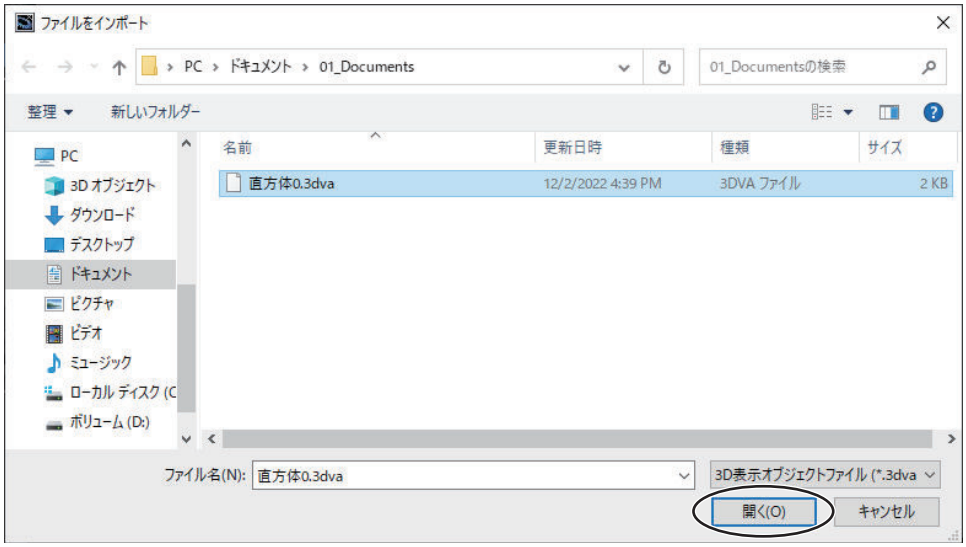
3D 表示オブジェクトをインポートする手順を説明します。
ここでは直方体を例に説明します。

1 **マルチビューエクスプローラの［構成・設定］から［3D 表示］を右クリックし、メニューから［3D 表示オブジェクトのインポート］をクリックします。**

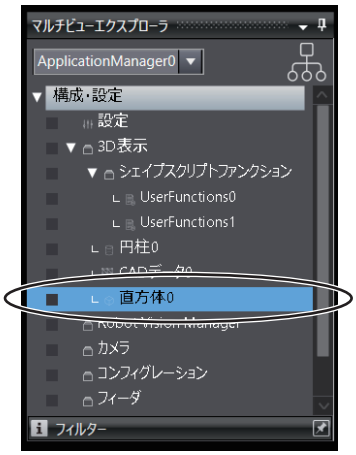


［ファイルをインポート］ダイアログボックスが表示されます。

2 **インポートする.3dva ファイルを選択し、［開く］ボタンをクリックします。**



マルチビューエクスプローラの［構成・設定］-［3D 表示］項目下に「直方体 0」が追加されます。



インポート先に同名の 3D 表示オブジェクトが存在する場合は、インポート元の 3D 表示オブジェクトが「［3D 表示オブジェクト名］_1」という名前で追加されます。

使用上の注意

エクスポート対象外となった参照情報は、インポート後の 3D 表示オブジェクトでは設定値が空になります。

参考

一括エクスポートで生成した.3dva ファイルをインポートした場合は、.3dva ファイルに含まれるすべての 3D 表示オブジェクトが追加されます。

5

3D 表示画面、3D 編集エリアの操作

本章では 3D 表示画面、3D 編集エリアに表示される項目と、操作方法について説明します。

5-1　3D 表示画面、3D 編集エリアの表示 5-2
5-1-1　3D 表示画面の表示 5-2
5-1-2　3D 編集エリアの表示 5-3

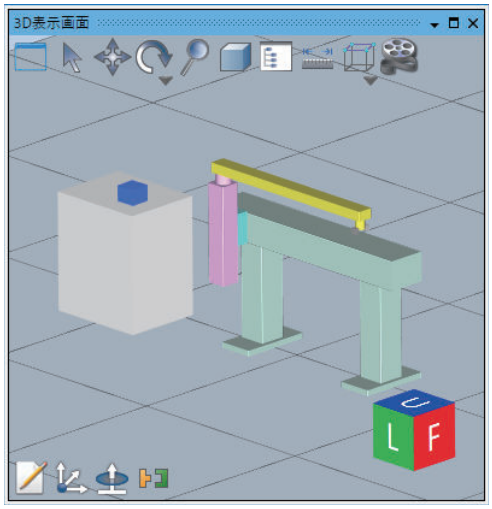
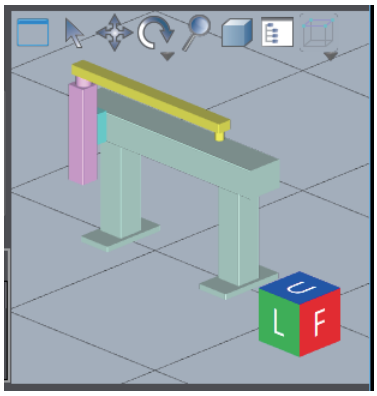
5-2　3D 表示画面、3D 編集エリアの項目 5-5
5-2-1　画面分割 5-6
5-2-2　選択と編集 5-6
5-2-3　平行移動 5-7
5-2-4　回転 5-8
5-2-5　ズーム 5-8
5-2-6　投影切り替え 5-9
5-2-7　シーングラフ 5-10
5-2-8　測定ルーラ 5-12
5-2-9　スナップ 5-13
5-2-10　記録 5-14
5-2-11　3D ビュー切替ツール 5-17

5-3　3D 表示画面の 3D 形状データ操作 5-19
5-3-1　3D 形状データの移動 5-19
5-3-2　3D 形状データの回転 5-20
5-3-3　3D 形状データの簡易編集 5-21

5-4　マウントポイント／リンクポイントを使用した位置設定 5-24
5-4-1　マウントポイントとリンクポイントの概要 5-24
5-4-2　マウントポイントの設定 5-26
5-4-3　リンクポイントの設定 5-27
5-4-4　オフセットの設定方法 5-29
5-4-5　マウントポイントとリンクポイントを使用した 3D 形状データの配置 5-33

5-1　3D 表示画面、3D 編集エリアの表示

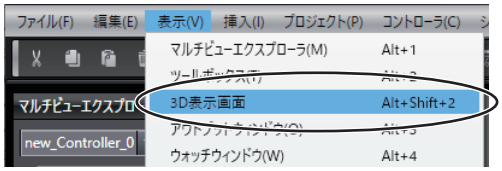
3D 表示画面、および 3D 編集エリアには追加した 3D 形状データが表示されます。
3D 表示画面と 3D 編集エリアには次の違いがあります。

	説明	
3D 表示画面	プロジェクトに登録されているすべての 3D 形状データが表示されます。3D シミュレーション実行時に、メカニカル機構、ワーク、仮想ワーク検出センサの動作を確認することができます。	
3D 編集エリア	対象の 3D 形状データの配置や姿勢を設定するときに使用する画面です。各 3D 形状データの設定画面の一部として表示されます。	

5-1-1　3D 表示画面の表示

3D 表示画面の表示手順は次のとおりです。

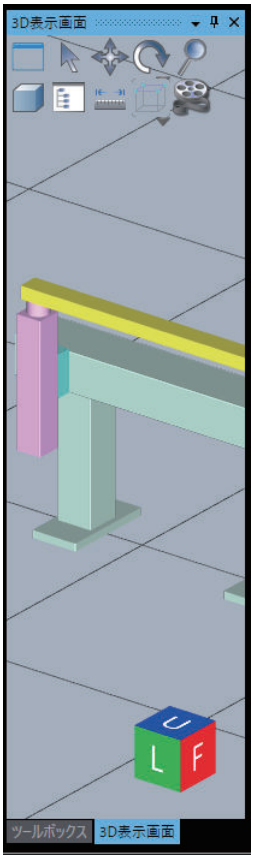
1　**メインメニューから［表示］-［3D 表示画面］を選択します。**



または、ツールバーの［3D 表示画面］ボタンをクリックします。



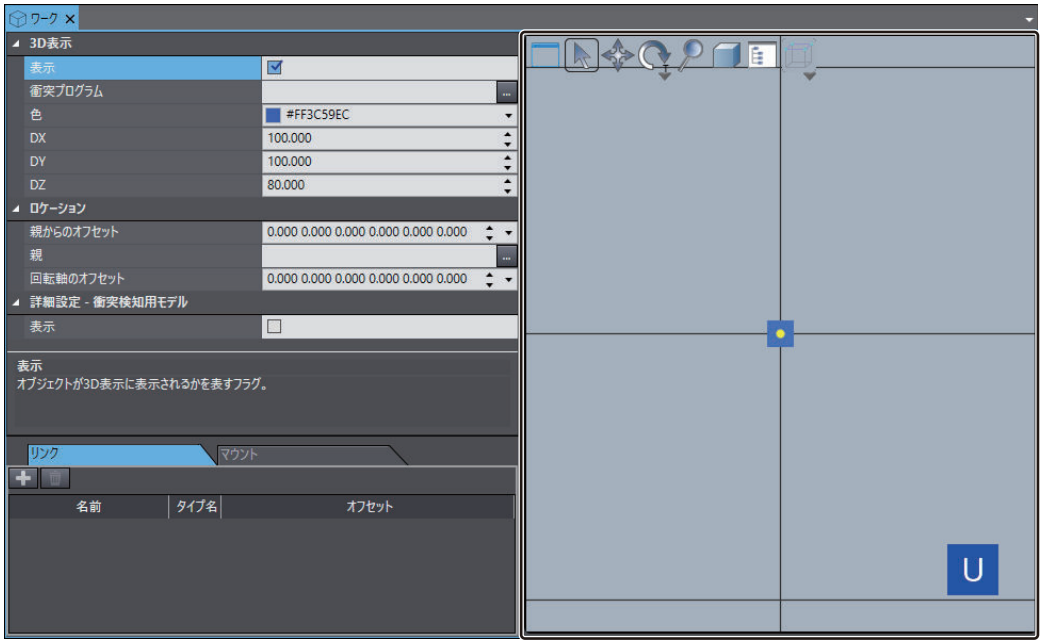
ツールボックスの位置に［3D 表示画面］が表示されます。



3D 表示画面をフローティングウィンドウとして表示し、ウィンドウのサイズを変更して見やすくすることができます。ウィンドウ配置の変更方法は『Sysmac Studio Version 1 オペレーションマニュアル(SBCA-470)』を参照してください。

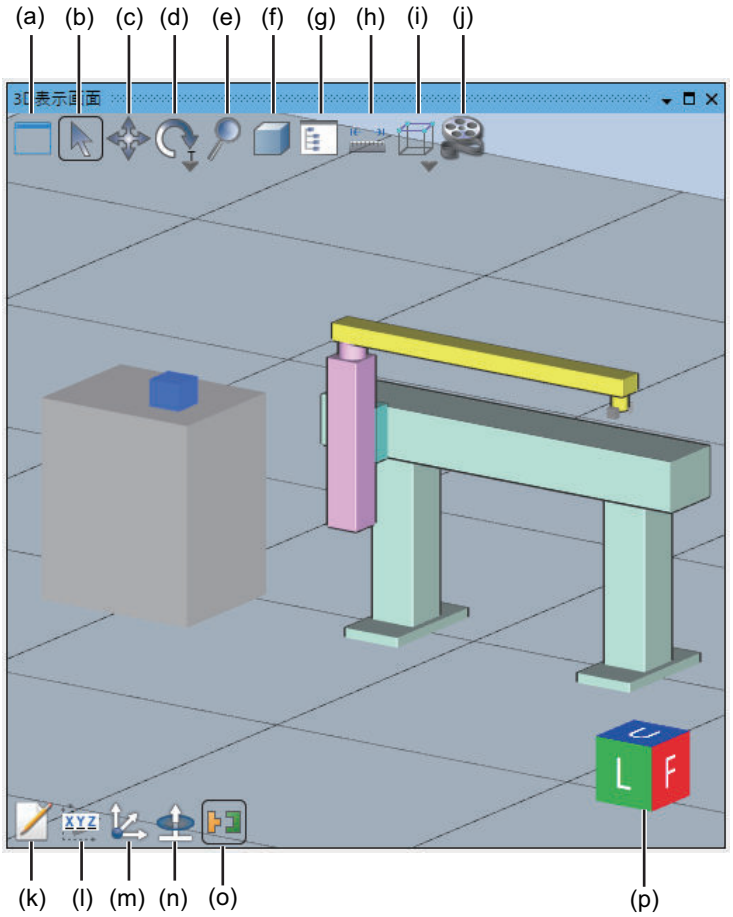
5-1-2　3D 編集エリアの表示

3D 編集エリアは、3D 形状データの設定画面の一部として表示されます。表示方法は、各 3D 形状データの追加方法の手順を参照してください。直方体の 3D 形状データの例を示します。


ワーク
3D表示
表示
衝突プログラム
色
#FF3C59EC
DX
100.000
DY
100.000
DZ
80.000
ロケーション
親からのオフセット
0.000 0.000 0.000 0.000 0.000 0.000
親
回転軸のオフセット
0.000 0.000 0.000 0.000 0.000 0.000
詳細設定 - 衝突検知用モデル
表示
表示
オブジェクトが3D表示に表示されるかを表すフラグ。
リンク
マウント
名前
タイプ名
オフセット
U

5-2 3D 表示画面、3D 編集エリアの項目

3D 表示画面、3D 編集エリアに表示される項目やアイコンについて説明します。一部の機能は、3D 表示画面でのみ使用できます。



記号	名称	概要	参照先
(a)	画面分割	3D 表示画面、3D 編集エリアの分割を行います。	「5-2-1 画面分割 (P.5-6)」
(b)	選択	編集する 3D 形状データを選択します。	「5-2-2 選択と編集 (P.5-6)」
(c)	平行移動	3D 表示画面、3D 編集エリアの視点の位置をスクロールします。	「5-2-3 平行移動 (P.5-7)」
(d)	回転	3D 表示画面、3D 編集エリアの視点の位置を回転します。	「5-2-4 回転 (P.5-8)」
(e)	ズーム	3D 表示画面、3D 編集エリアを拡大／縮小します。	「5-2-5 ズーム (P.5-8)」
(f)	投影切り替え	3D 表示画面、3D 編集エリアの投影方法を、平行投影または透視投影に切り替えます。	「5-2-6 投影切り替え (P.5-9)」
(g)	シーングラフ	3D 形状データの表示／非表示の設定や衝突検知の設定を行うシーングラフダイアログボックスを開きます。	「5-2-7 シーングラフ (P.5-10)」
(h)*1	測定ルーラ	3D 表示画面で 3D 形状データの間の距離を測定します。	「5-2-8 測定ルーラ (P.5-12)」
(i)	スナップ	3D 形状データ、および 3D 形状データのマウントポイントやリンクポイントを指定した点に移動させます。	「5-2-9 スナップ (P.5-13)」
(j)*1	記録	3D 表示画面で実行中のシミュレーション表示を動画として記録します。	「5-2-10 記録 (P.5-14)」

記号	名称	概要	参照先
(k)*1	編集	選択している 3D 形状データの設定を開きます。	「5-2-2 選択と編集 (P.5-6)」
(l)*1	簡易編集画面	選択している 3D 形状データの位置、向き、サイズを数値で入力し、変更します。	「5-3-3 3D 形状データの簡易編集 (P.5-21)」
(m)*1	ワークスペースの位置の編集	選択している 3D 形状データの位置を移動します。	「5-3-1 3D 形状データの移動 (P.5-19)」
(n)*1	ワークスペースの向きの編集	選択している 3D 形状データの向きを変更します。	「5-3-2 3D 形状データの回転 (P.5-20)」
(o)*1	マウントポイント表示	マウントポイントの表示／非表示を切り替えます。	「5-4 マウントポイント／リンクポイントを使用した位置設定 (P.5-24)」
(p)	3D ビュー切替ツール	3D 表示画面、3D 編集エリアの表示方向を切り替えます。	「5-2-11 3D ビュー切替ツール (P.5-17)」

*1. 3D 編集エリアでは表示されません。

編集、簡易編集画面、ワークスペースの位置の編集、ワークスペースの向きの編集については、「5-3 3D 表示画面の 3D 形状データ操作 (P.5-19)」を参照してください。

5-2-1 画面分割

3D 表示画面、3D 編集エリアの分割を行うときに使用します。
アイコンをクリックすると、以下のウィンドウが開きます。閉じるときは［X］をクリックします。



ウインドウで選択可能なアイコンの機能は以下のとおりです。

アイコン	名称	機能
	分割なし	3D 表示画面、3D 編集エリアを分割なしで表示します。
	2 分割（左右）	3D 表示画面、3D 編集エリアを左右に 2 分割して表示します。
	2 分割（上下）	3D 表示画面、3D 編集エリアを上下に 2 分割して表示します。
	4 分割	3D 表示画面、3D 編集エリアを 4 分割して表示します。

5-2-2 選択と編集

3D 形状データを選択し、3D 形状データ固有の設定を変更するときに、このアイコンを選択します。

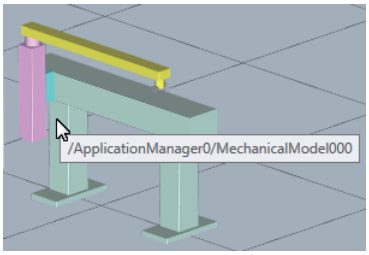
1 **3D 表示画面または 3D 編集エリアの［選択］アイコンをクリックします。または、Q キーを押します。**



カーソルが 3D 形状データを選択する状態を示す矢印になります。



2 **選択したい 3D 形状データにカーソルを移動します。**
カーソル位置の 3D 形状データの名前が表示されます。



3 **3D 表示画面から 3D 形状データを編集する場合は、3D 形状データをダブルクリックします。または［編集］アイコンをクリックします。**
該当する 3D 形状データの設定画面が開きます。



5-2-3　平行移動

3D 表示画面、3D 編集エリアの視点の位置をスクロールするときにこのアイコンを選択します。

1 **3D 表示画面または 3D 編集エリアの［平行移動］アイコンをクリックします。または、W キーを押します。**



カーソルが 3D 表示画面または 3D 編集エリアの平行移動を行う状態を示すアイコンになります。



2 **マウスの左ボタンを押したまま移動させたい方向にドラッグします。**

5-2-4 回転

3D 表示画面、3D 編集エリアの視点の位置を回転するときにこのアイコンを選択します。

1 **3D 表示画面または 3D 編集エリアの［回転］アイコンをクリックします。または、E キーを押します。**



カーソルが視点の位置の回転を行う状態を示すアイコンになります。



2 **マウスの左ボタンを押したまま回転させたい方向にドラッグします。**
上方向のドラッグ: 視点の位置を下方向に移動
下方向のドラッグ: 視点の位置を上方向に移動
右方向のドラッグ: 視点の位置を左方向に移動
左方向のドラッグ: 視点の位置を右方向に移動

回転には、次の 2 つのモードがあります。

モード	説明
タンブラーモード	任意の角度から 3D 形状データを見ることができます。
ターンテーブルモード	ワールド座標系の Z 軸を中心に左右に回転します。また、上下は±90°の範囲で見ることができます。

5-2-5 ズーム

3D 表示画面、3D 編集エリアを拡大／縮小するときにこのアイコンを選択します。

1 **3D 表示画面または 3D 編集エリアの［ズーム］アイコンをクリックします。または、R キーを押します。**



カーソルがズームを行う状態を示すアイコンになります。



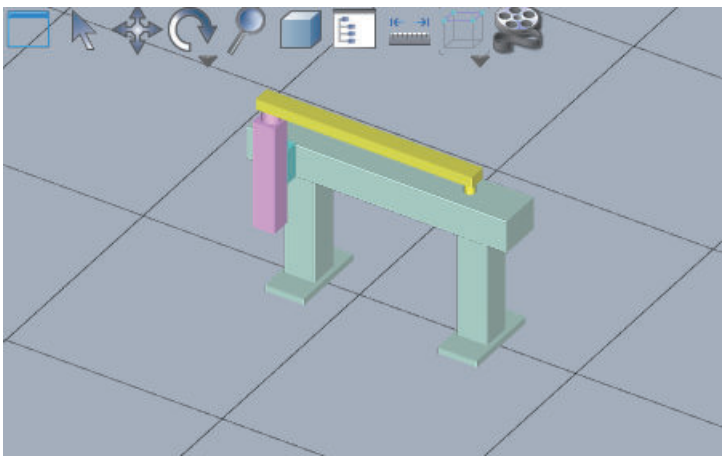
2 **マウスの左ボタンを押したままドラッグします。**
上方向にドラッグ: 拡大（ズームイン）
下方向にドラッグ: 縮小（ズームアウト）

5-2-6 投影切り替え

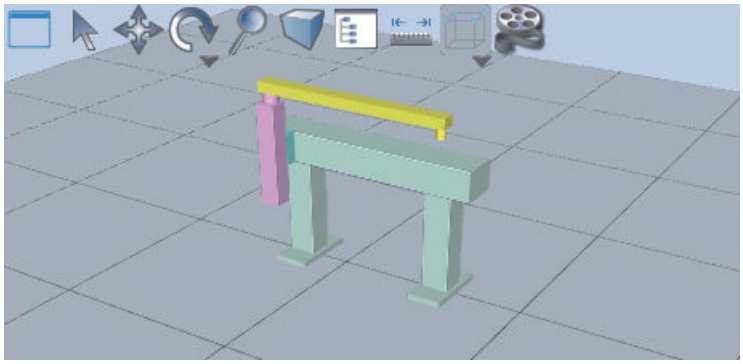
3D 表示画面、3D 編集エリアの投影方法を平行投影と透視投影に切り替えるときに使用します。

アイコン	名称	説明
	平行投影	物体上の点と視点を平行に結んだ投影線を投影する方法です。 同じ大きさの物体が、遠くにあっても近くにあっても同じ大きさで表示されるという特徴があります。配置されている物体の大きさを比較したいときは、この投影方法が適しています。
	透視投影	遠近法を適用して投影する方法です。 同じ大きさの物体が遠くにある場合は小さく、近くにある場合は大きく表示されるという特徴があります。現実の世界に近い表示をしたいときは、この投影方法が適しています。

平行投影の例



透視投影の例



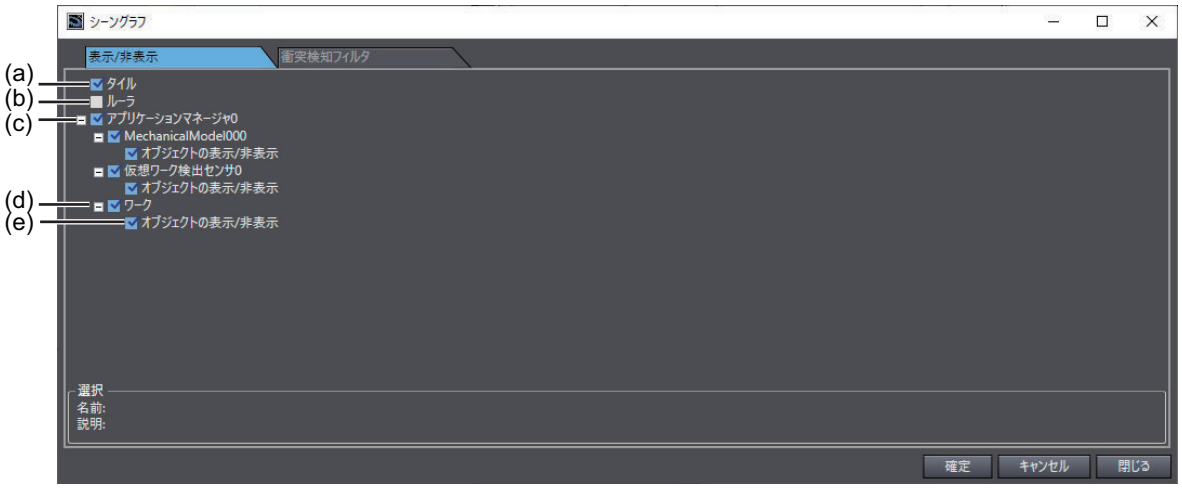
5-2-7　シーングラフ

3D 形状データの表示／非表示の設定や衝突検知の設定を行います。

設定項目は以下のとおりです。

表示／非表示タブ

3D 表示画面または 3D 編集エリアにおけるタイルやルーラ、3D 形状データの表示／非表示を一覧で設定します。
チェックボックスにチェックを入れると表示されます。



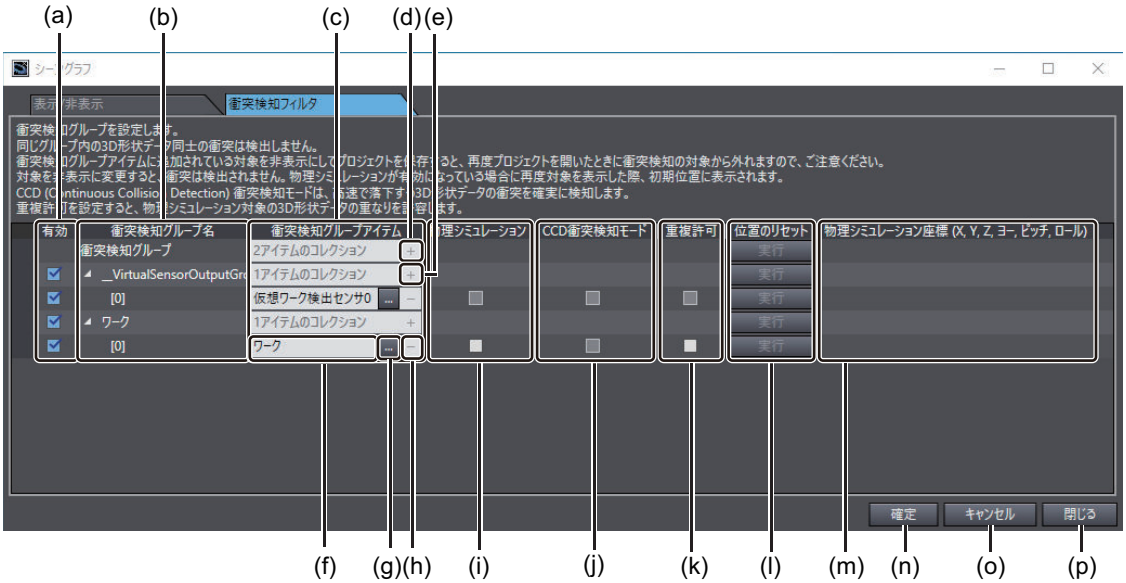
	設定項目	説明	設定値	初期値
(a)	タイル	3D 表示画面または 3D 編集エリアにおける Z 軸 0 の XY 平面を 1000mm 間隔のメッシュで表示します。	チェック ON／チェック OFF	チェック ON
(b)	ルーラ	3D 表示画面で 3D 形状データの間の距離を測定するときに使用するルーラを表示します。測定ルーラアイコンと同じです。	チェック ON／チェック OFF	チェック OFF
(c)	デバイス	プロジェクトに登録されている 3D 形状データの 3D 表示画面における表示／非表示をデバイス単位で設定します。*1	チェック ON／チェック OFF	チェック ON

	設定項目	説明	設定値	初期値
(d)	3D 形状データ	プロジェクトに登録されている 3D 形状データの 3D 表示画面における表示／非表示を 3D 形状データ単位で設定します。*1	チェック ON／チェック OFF	チェック ON
(e)	オブジェクトの表示／非表示			

*1. 3D 編集エリアにおける 3D 形状データは、チェック ON／チェック OFF にかかわらず常に表示されます。

衝突検知フィルタタブ

衝突検知フィルタタブは、3D 表示画面の場合のみ表示されます。



	設定項目	説明	設定値	初期値
(a)	有効チェックボックス	各検知グループに含まれる項目を衝突検知対象にするかどうかを切り替えます。チェック ON で衝突検知対象になります。	チェック ON／チェック OFF	チェック ON
(b)	衝突検知グループ名リスト	衝突検知グループのリストを管理します。 テキストボックスには「*-itemcollection（*は衝突検知グループ数）」と表示されます。［衝突検知グループ追加］ボタンから衝突検知グループを追加します。 追加されるグループ名は、「グループ*（*は衝突検知グループ数）」です。	文字列	0-item collection（初期状態では衝突検知グループは存在しません）
(c)	衝突検知グループアイテムリスト	衝突検知グループアイテムのリストを管理します。 テキストボックスには「*-itemcollection（*は衝突検知グループアイテム数）」と表示されます。［衝突検知グループアイテム追加］ボタンから衝突検知グループアイテムを追加します。	文字列	0-item collection（初期状態では衝突検知グループアイテムは存在しません）
(d)	衝突検知グループ追加ボタン	衝突検知グループを追加します。 このボタンをクリックすると、衝突検知グループがリストに追加されます。	-	-
(e)	衝突検知グループアイテム追加ボタン	衝突検知グループアイテムを追加します。 このボタンをクリックすると、衝突検知グループアイテムを設定する行がリストに追加されます。	-	-

	設定項目	説明	設定値	初期値
(f)	衝突検知グループアイテム名	衝突検知グループに追加する対象を設定します。 設定できる対象は、「8-3-1 衝突検知対象 (P.8-15)」を参照してください。	文字列	-
(g)	衝突検知グループアイテム選択ボタン	衝突検知グループに追加する対象を選択するためのダイアログボックスを表示します。	-	-
(h)	衝突検知グループアイテム削除ボタン	衝突検知グループアイテムを衝突検知グループから削除します。 このボタンをクリックすると、該当する項目が削除されます。	-	-
(i)	物理シミュレーション	チェックを入れ、[確定] ボタンをクリックすると、対象の 3D 形状データの物理シミュレーションが有効になります。 3D 形状データが落下したときに他の 3D 形状データへ衝突する様子や、カスタムメカニカル機構の可動部品の動作を確認できます。	チェック ON／チェック OFF	チェック OFF
(j)	CCD 衝突検知モード	CCD(Continuous Collision Detection)衝突検知モードを有効にすると、高速で落下する 3D 形状データの衝突を確実に検知します。有効にすると衝突を確実に検知しますが、3D 表示画面の描画性能に影響します。	チェック ON／チェック OFF	チェック OFF
(k)	重複許可	重複許可を設定すると、物理シミュレーション対象の 3D 形状データどうしの重なりを許容します。	チェック ON／チェック OFF	チェック OFF
(l)	位置のリセット	物理シミュレーション対象の 3D 形状データを元の位置へ戻します。 [物理シミュレーション] にチェックが入っている場合、[実行] ボタンをクリックするたびに、対象の 3D 形状データが元の位置から落下したり、カスタムメカニカル機構の可動部品が元の位置から動作したりします。	-	-
(m)	物理シミュレーション座標 (X、Y、Z、ヨー、ピッチ、ロール)	物理シミュレーション実行時の 3D 形状データの座標値が表示されます。	-	-
(n)	[確定] ボタン	変更内容を確定します。	-	-
(o)	[キャンセル] ボタン	変更をキャンセルし、シーングラフダイアログボックスを閉じます。	-	-
(p)	[閉じる] ボタン	シーングラフダイアログボックスを閉じます。	-	-

衝突検知の設定方法の詳細については、「8-3 衝突検知機能 (P.8-15)」を参照してください。

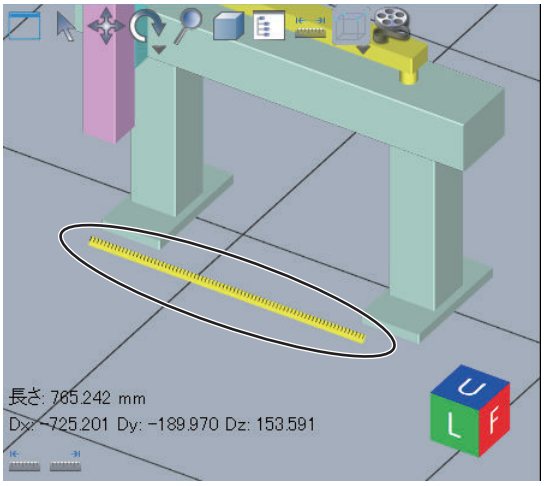
5-2-8 測定ルーラ

3D 表示画面で 3D 形状データの間の距離を測定するときに使用します。

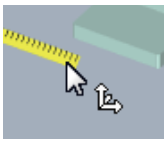
1 3D 表示画面の［測定ルーラ］アイコンをクリックします。



黄色の帯のルーラ（目盛りが黒）が表示されます。



2 **ルーラの端にマウスカーソルを移動します。**
マウスカーソルが変化します。



3 **計測したい区間の端にルーラの端をドラッグして移動させます。**

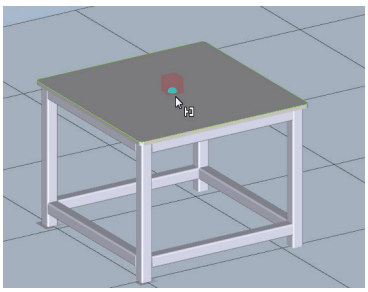
5-2-9 スナップ

3D 形状データのマウントポイントやリンクポイントのオフセットを設定するときに使用します。

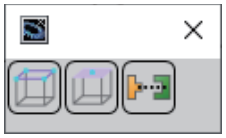
1 **対象の 3D 形状データを選択し、［スナップ］アイコンをクリックまたは T キーを押します。**



2 **カーソルをスナップ先に移動させると、スナップ先となる候補が表示されます。スナップ先となる点をクリックするとその位置にスナップできます。**

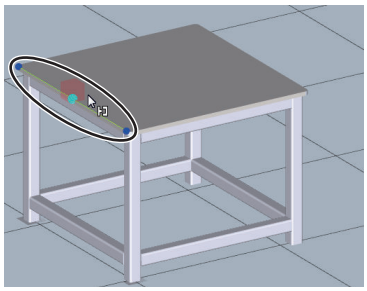


スナップのモードを変更するにはアイコンをクリックし、以下のウィンドウで選択します。または T +数字キーを押します。閉じるときはウィンドウの［×］をクリックします。

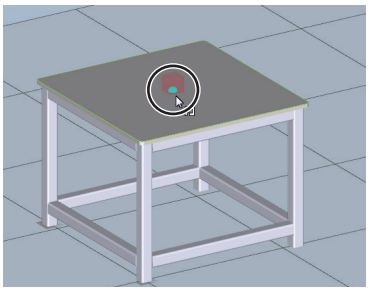


ウィンドウで選択可能なアイコンの機能は以下のとおりです。

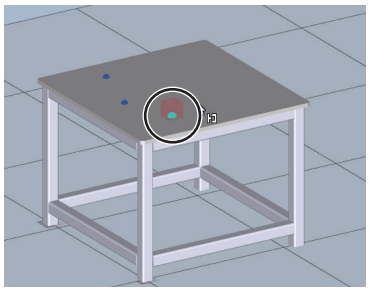
アイコン	名称	機能	ショートカット
	エッジへのスナップ	3D 形状データに設定したマウントポイントまたはリンクポイントの近くにカーソルを移動すると、スナップ位置のプレビューが表示されます。 ハイライトされた辺の両端または中心が強調され、その位置にスナップできます。	T+1
	面の重心へのスナップ	3D 形状データに設定したマウントポイントまたはリンクポイントの近くにカーソルを移動すると、スナップ位置のプレビューが表示されます。 ハイライトされた面の重心が強調され、その位置にスナップできます。	T+2
	リンクポイントへのスナップ	3D 形状データに設定したマウントポイントの近くにカーソルを移動すると、スナップ位置のプレビューが表示されます。 ハイライトされた面にあるリンクポイントが表示され、その位置にスナップできます。 3D 表示画面でのみ使用できます。	T+5



エッジへのスナップ



面の重心へのスナップ



リンクポイントへのスナップ



参考

- スナップ位置のプレビューが表示されるオブジェクトは以下のとおりです。
  a）直方体
  b）円柱
  c）CAD データ
  d）メカニカル機構
  e）カスタムメカニカル機構
  f）パラレルリンク機構
  ロボットに装着したエンドエフェクタのスナップ時、エンドエフェクタを親に設定した上記オブジェクト a)～f)のプレビューが表示されます。
- スナップ位置のハイライト表示色を変更することができます。変更はオプション設定で行います。詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

5-2-10　記録

3D 表示画面で実行中のシミュレーション表示を動画として記録するときに使用します。

動画の記録

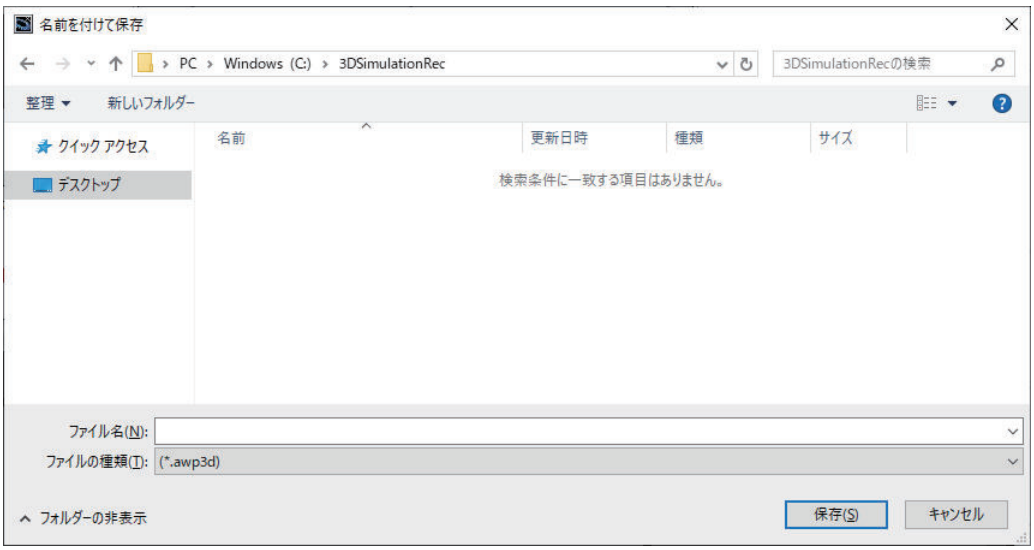
1 **3D シミュレーションを実行中に、［記録］アイコンをクリックします。**



アイコンが変化し、点滅します。動画の記録中であることを示します。



2 **記録を終了させたいときに［記録］アイコンをクリックします。**
［名前を付けて保存］ダイアログボックスが表示されます。

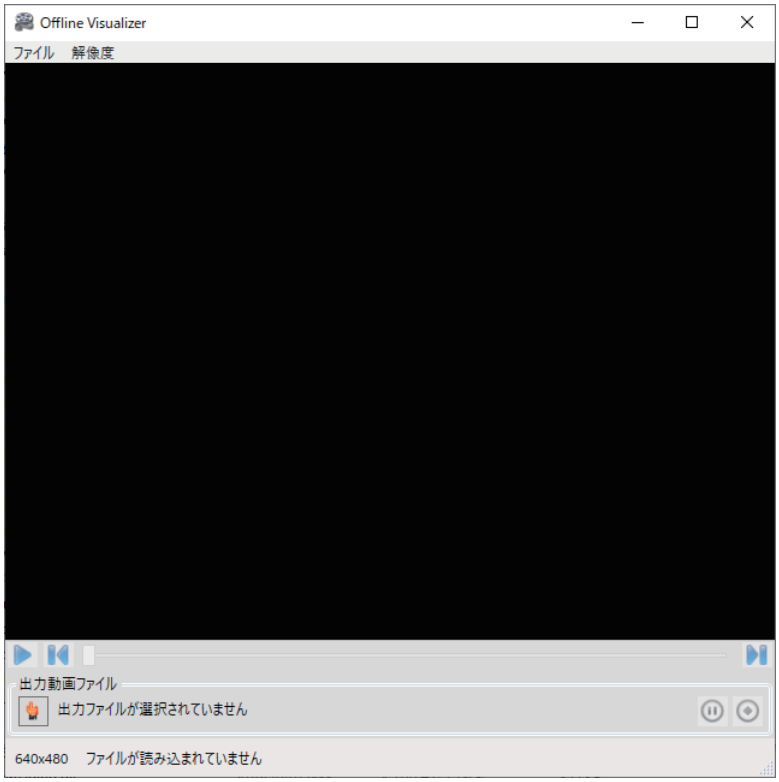


3 **ファイル名を入力し、［保存］ボタンをクリックします。**
動画がファイルに保存されます。

動画の再生

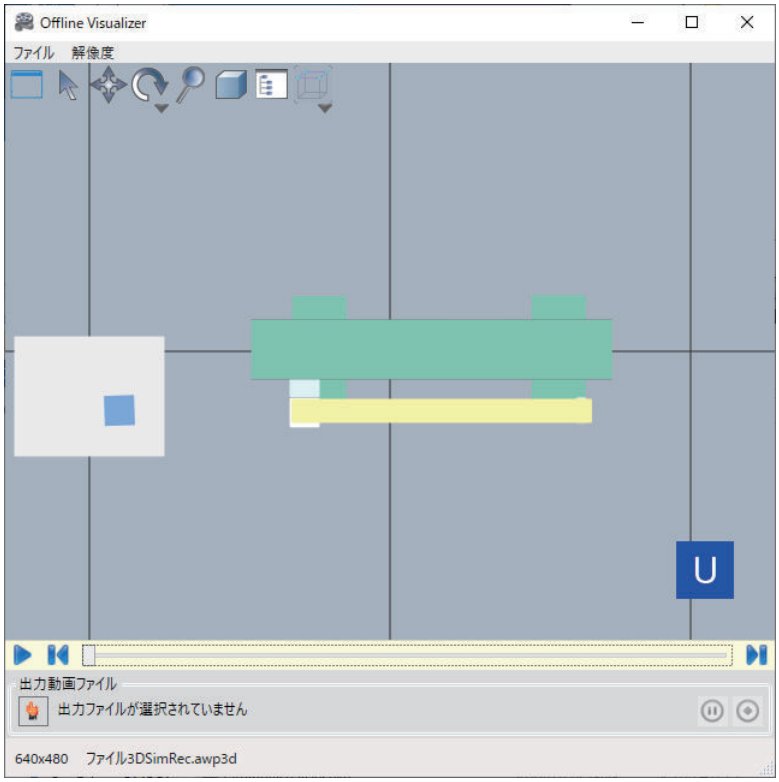
再生専用ツールで動画を再生します。

1 **Windows のスタートメニューから［すべてのプログラム］-［OMRON］-［Sysmac Studio］-［Tools］-［Offline Visualizer］を選択します。**
［Offline Visualizer］が起動します。



2 **［ファイル］メニューから［開く］を選択します。**
［開く］ダイアログボックスが表示されます。

3 **再生したい記録ファイル（拡張子.awp3d）を選択し、［開く］ボタンをクリックします。**
選択した記録ファイルが開きます。



4 **［再生］ボタンをクリックします。**
動画が再生されます。

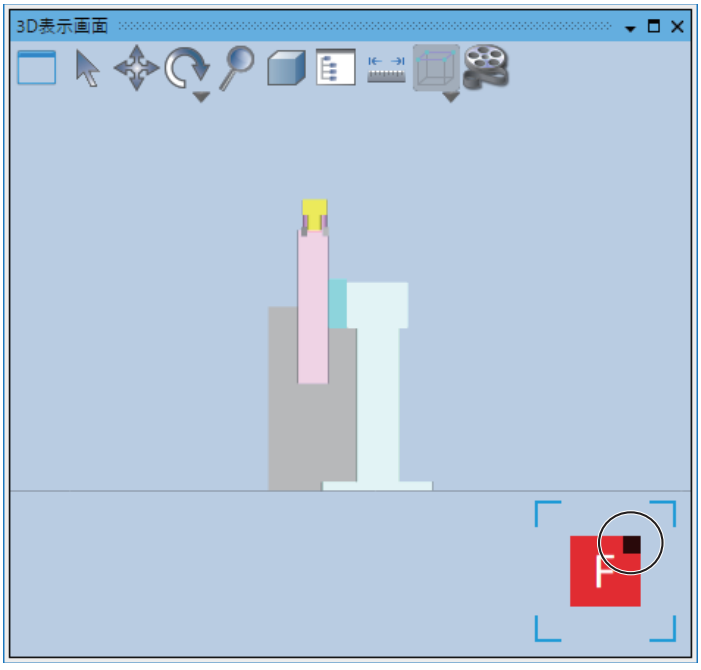
5-2-11　3D ビュー切替ツール

3D 表示画面または 3D 編集エリア上の 3D 形状データの表示方向を切り替えるツールです。
3D ビュー切替ツールは、フェース、コーナー、エッジの三つの要素で構成されます。要素にマウスカーソルをあわせると黒く表示され、そのままクリックするとクリックした部分が正面になるように表示が切り替わります。それに合わせて 3D 表示画面、3D 編集エリア上の 3D 形状データの表示方向も切り替わります。

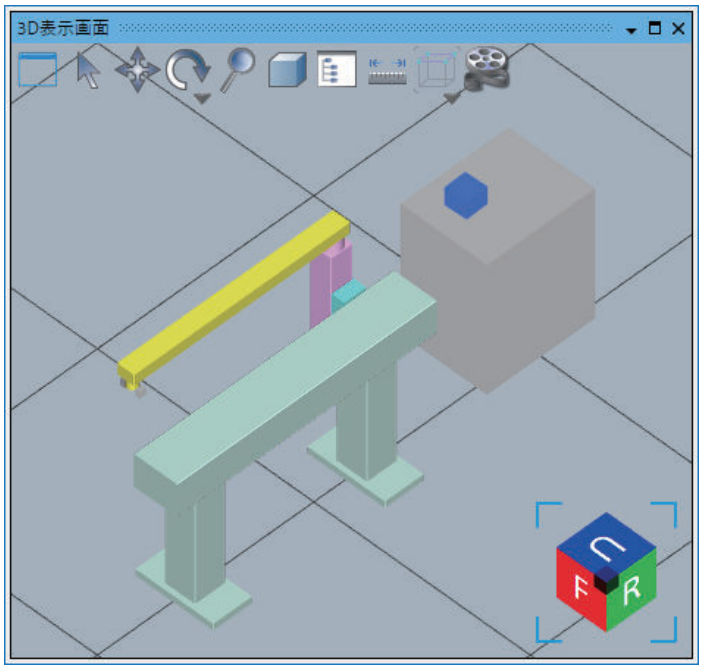
構成要素	名称	説明	ショートカット
	フェース	面を表します。面にはそれぞれ以下の記号が表示されます。	-
		F(Front): yz 平面に正対したときの、正面	Ctrl+1
		B(Back): F 面に対して裏側の面	Ctrl+2
		U(Up): F 面に対して上側の面	Ctrl+3
		D(down): F 面に対して下側の面	Ctrl+4
		L(Left): F 面に対して左側の面	Ctrl+5
		R(Right): F 面に対して右側の面	Ctrl+6
	コーナー	頂点を表します。	-
	エッジ	辺を表します。	Ctrl+7 （Up と Front の間の辺）

3D ビュー切替ツールのコーナーを選択する場合を例に、操作方法を説明します。

1 **3D 表示画面または 3D 編集エリアで、フェースの右上のコーナーをクリックします。**



選択したコーナーが正面となる表示に切り替わり、同時に 3D 表示画面、3D 編集エリアの 3D 形状データの表示も切り替わります。



参考

3D 表示の縮尺や表示位置の状態を初期状態に戻すには、マウスカーソルを 3D 表示画面または 3D 編集エリア上に置き、Ctrl ＋ 8 キーを押します。

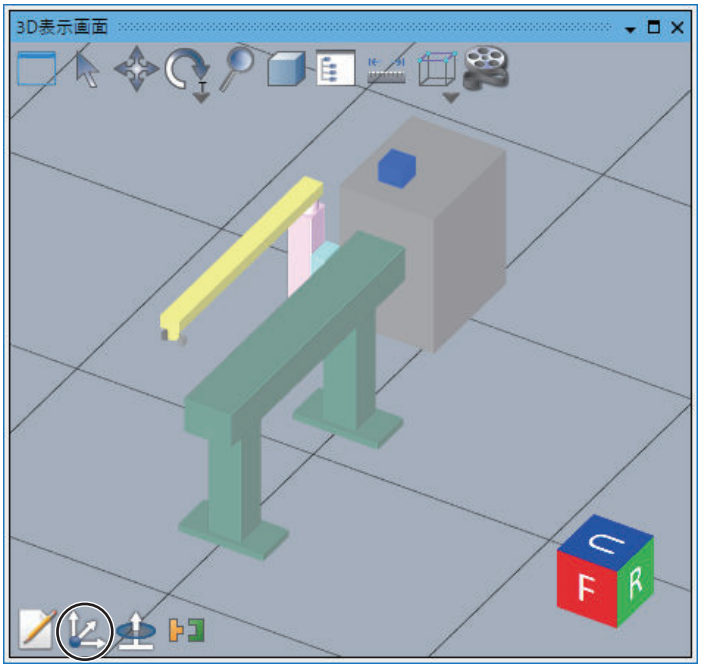
5-3　3D 表示画面の 3D 形状データ操作

3D 表示画面での 3D 形状データの移動などの操作方法を説明します。

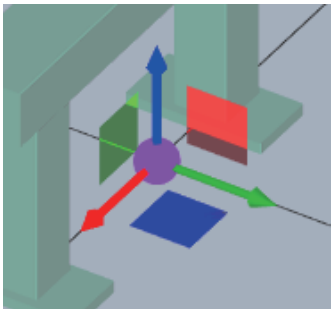
5-3-1　3D 形状データの移動

3D 表示画面で 3D 形状データ間の位置関係を確認しながら編集できます。

1　**3D 表示画面で移動する 3D 形状データをマウスカーソルで選択します。**
3D 表示画面に［ワークスペースの位置の編集］アイコンが表示されます。



2　**［ワークスペースの位置の編集］アイコンをクリックします。**
3D 形状データの上に［移動］アイコンが表示されます。



3　**アイコンをドラッグするとその 3D 形状データが移動します。**

アイコン	名称	機能
	X 方向平行移動	アイコンをドラッグすると、ローカル座標系の X 軸方向に平行に移動します。

アイコン	名称	機能
	Y 方向平行移動	アイコンをドラッグすると、ローカル座標系の Y 軸方向に平行に移動します。
	Z 方向平行移動	アイコンをドラッグすると、ローカル座標系の Z 軸方向に平行に移動します。
	YZ 平面上移動	アイコンをドラッグすると、ローカル座標系の YZ 平面上を移動します。
	XZ 平面上移動	アイコンをドラッグすると、ローカル座標系の XZ 平面上を移動します。
	XY 平面上移動	アイコンをドラッグすると、ローカル座標系の XY 平面上を移動します。

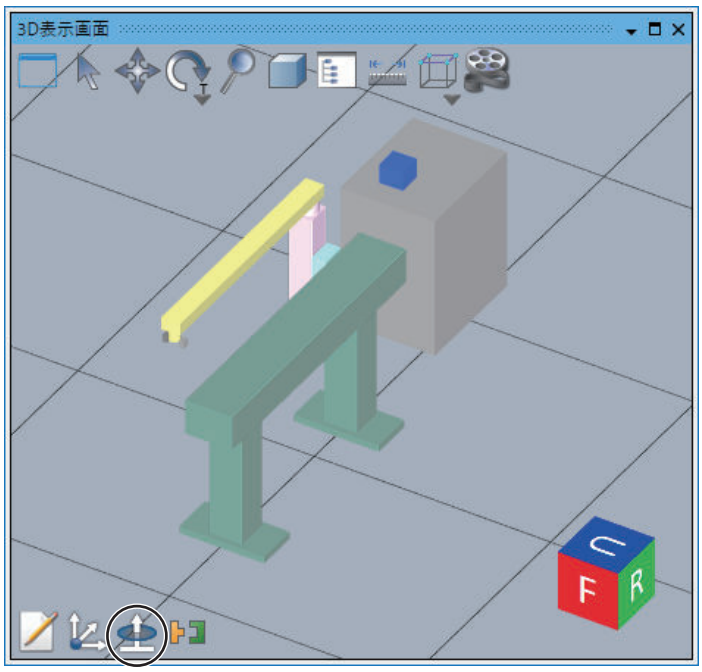
参考

3D 形状データは、対象 3D 形状データの設定画面の［ロケーション］の値を直接編集することでも移動可能です。

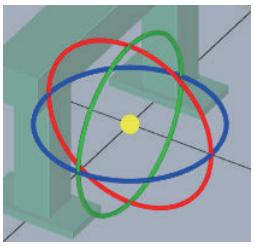
5-3-2 3D 形状データの回転

3D 表示画面で 3D 形状データ間の位置関係を確認しながら編集できます。

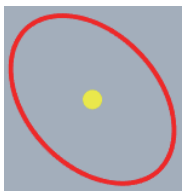
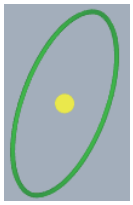
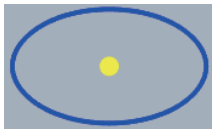
1 **3D 表示画面で回転させる 3D 形状データをマウスカーソルで選択します。**
3D 表示画面に［ワークスペースの向きの編集］アイコンが表示されます。



2 **［ワークスペースの向きの編集］アイコンをクリックします。**
3D 形状データの上に［回転］アイコンが表示されます。



3 **アイコンをドラッグするとその 3D 形状データが回転します。**

アイコン	名称	機能
	X 軸まわりの回転	アイコンのハンドルをドラッグすると、対象 3D 形状データの設定画面の［回転軸のオフセット］で設定した点を中心に、X 軸まわりに回転します。
	Y 軸まわりの回転	アイコンのハンドルをドラッグすると、対象 3D 形状データの設定画面の［回転軸のオフセット］で設定した点を中心に、Y 軸まわりに回転します。
	Z 軸まわりの回転	アイコンのハンドルをドラッグすると、対象 3D 形状データの設定画面の［回転軸のオフセット］で設定した点を中心に、Z 軸まわりに回転します。

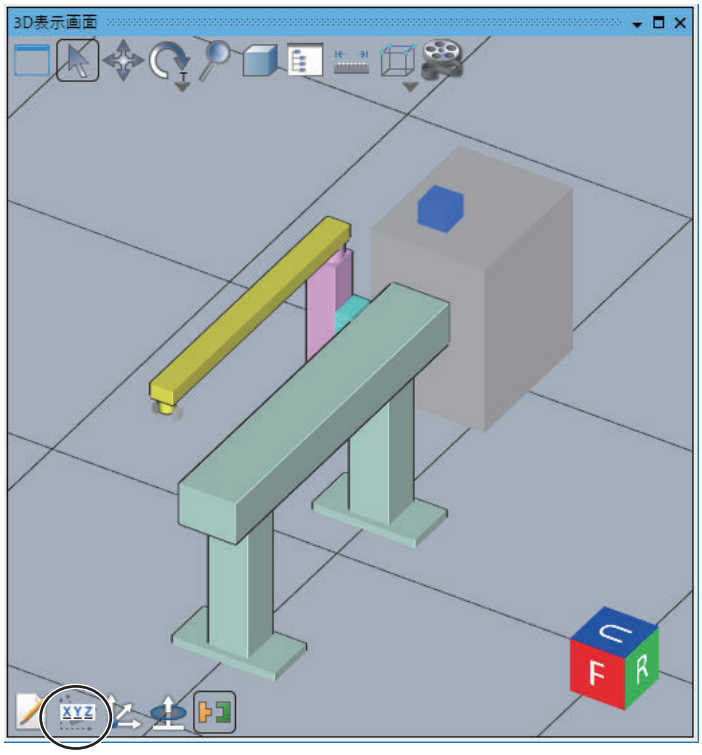
参考

- 3D 形状データは、対象 3D 形状データの設定画面の［ロケーション］の値を直接編集することでも回転可能です。
- 各軸まわりに回転するとき、3D 形状データのローカル座標系も一緒に回転します。

5-3-3 3D 形状データの簡易編集

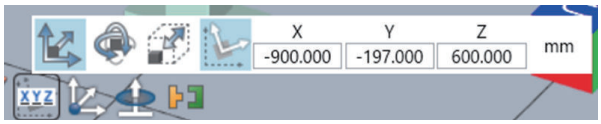
3D 表示画面で 3D 形状データ間の位置関係を確認しながら直接数値を入力することで編集できます。

1 **3D 表示画面で編集する 3D 形状データをマウスカーソルで選択します。**
3D 表示画面に［簡易編集画面］アイコンが表示されます。



2 **［簡易編集画面］アイコンをクリックします。**

3D 表示画面に 3D 形状データの数値入力画面が表示されます。



3 **［簡易編集画面］のアイコンを選択すると 3D 形状データの数値入力が可能となります。**

アイコン	名称	機能
	ワークスペースの位置の編集	3D 形状データの位置を X, Y, Z の数値で入力します。 X -900.000 Y -197.000 Z 600.000 mm
	ワークスペースの向きの編集	3D 形状データの向きを Yaw, Pitch, Roll の数値で入力します。 Yaw 0.000 Pitch 0.000 Roll 1.944 degree

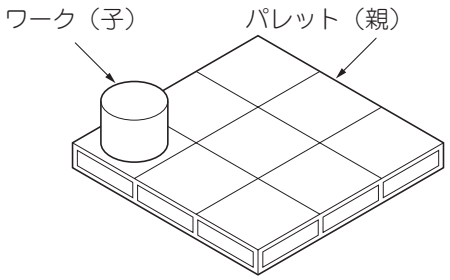
アイコン	名称	機能
	サイズの編集	3D 形状データのサイズを数値で入力します。3D 形状データにより入力する数値が変更されます。 ［直方体］  ［円柱］  ［仮想ワーク検出センサ］  ［ベルト］ 
	ローカル座標系	［ワークスペースの位置の編集］および［ワークスペースの向きの編集］の座標系を選択します。クリックするたびに座標系が交互に変更されます。 初期値はローカル座標系です。
	ワールド座標系	

5-4 マウントポイント／リンクポイントを使用した位置設定

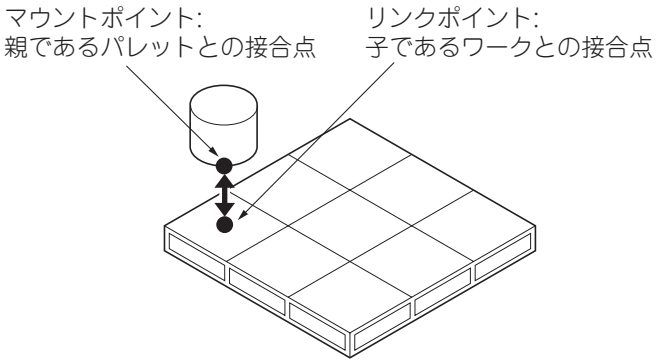
パレット上の特定の位置にワークを配置するなど、2 つの 3D 形状データを接触させて配置する場合に、マウントポイント／リンクポイントを使用すると、位置合わせを簡単に行うことができます。
ここではマウントポイント／リンクポイントの設定方法と、マウントポイント／リンクポイントを使用した 3D 形状データの配置の操作手順を説明します。

5-4-1 マウントポイントとリンクポイントの概要

マウントポイント、リンクポイントとは、親子関係のある二つの 3D 形状データを接合するときに使用する点です。子が持つ親との接合点をマウントポイント、親が持つ子との接合点をリンクポイントと呼びます。



親がパレット、子がワーク



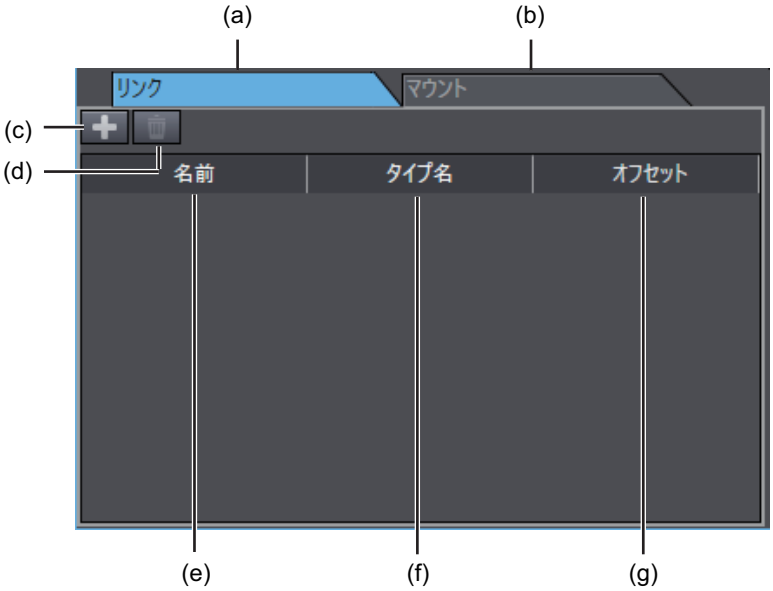
パレットにリンクポイント、
ワークにマウントポイントを設定した例

参考

3D 表示画面上で、接触する二つの 3D 形状データの位置合わせを行う際、マウスで正確に行うのは容易ではありません。
対象の 3D 形状データにマウントポイント／リンクポイントをあらかじめ設定しておくことで、3D 表示画面上で簡単に位置合わせを行うことができます。

マウントポイント／リンクポイントの設定画面

3D 形状データのマウントポイント、リンクポイントは、設定画面のマウントポイント／リンクポイント設定エリアで追加します。



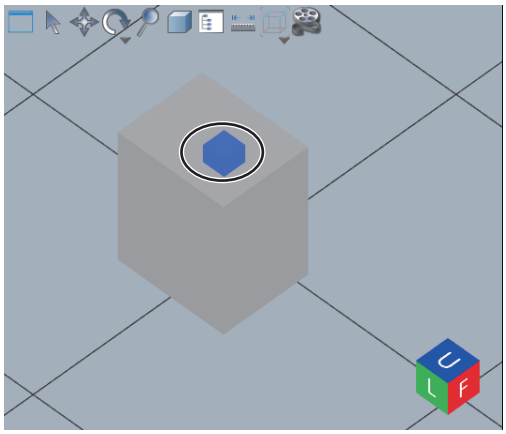
	設定項目	説明	設定値	初期値
(a)	リンクタブ	3D 形状データのリンクポイントを設定するタブです。 リンクポイントとは、親である本 3D 形状データに、子となる 3D 形状データを取り付けるポイントです。	なし	空
(b)	マウントタブ	3D 形状データのマウントポイントを設定するタブです。 マウントポイントとは、親となる 3D 形状データに、子となる本 3D 形状データを取り付けるポイントです。	なし	空
(c)	マウント／リンクポイント追加ボタン	ボタンをクリックすると、マウントポイント設定エリアの表にマウントポイント／リンクポイントが追加されます。	-	-
(d)	マウント／リンクポイント削除ボタン	削除したい行を選択し、このボタンをクリックすると、選択した行のマウントポイント／リンクポイントが削除されます。	-	-
(e)	名前	マウントポイント／リンクポイントの名称を表示します。 任意の名称を設定できます。	文字列	Connection Point
(f)	タイプ名	マウントポイント／リンクポイントのタイプを設定します。 2 つの 3D 形状データのマウントポイントとリンクポイントに同じタイプ名を設定すると、3D 表示画面で［リンクポイントへのスナップ］を選択したときに同じタイプ名のリンクポイントだけが表示されます。	文字列	Object

	設定項目	説明	設定値	初期値
(g)	オフセット	マウントポイント／リンクポイントの座標を設定します。 設定する座標は本 3D 形状データのローカル座標系における座標です。対応するマウントポイント／リンクポイントは、3D 編集エリアに表示されます。オフセットの値を変更すると、3D 編集エリアに表示されるマウントポイント／リンクポイントの位置も変わります。ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～ +180.000 p: 0.000～180.000	すべて 0.000

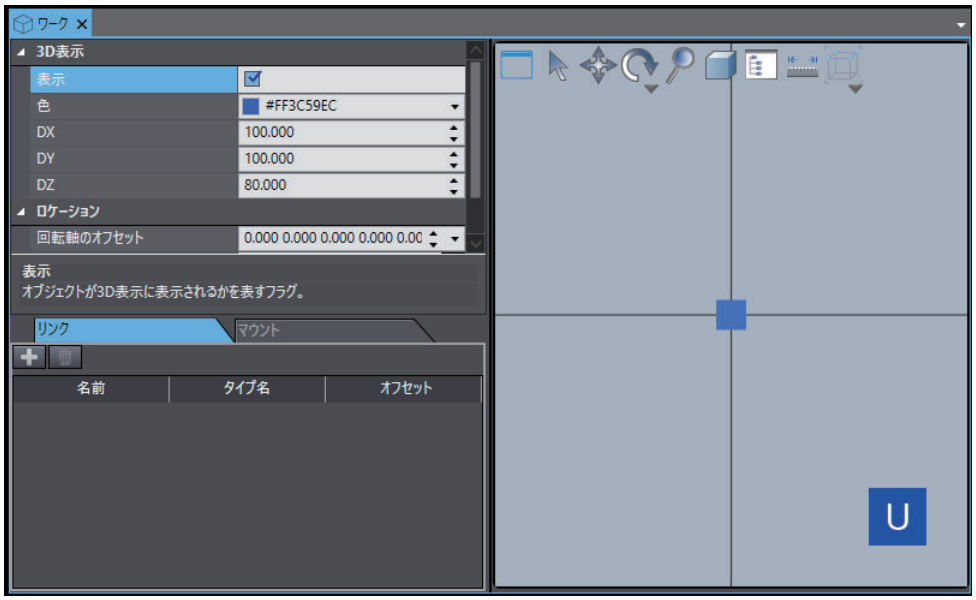
5-4-2 マウントポイントの設定

マウントポイントを設定する手順を説明します。

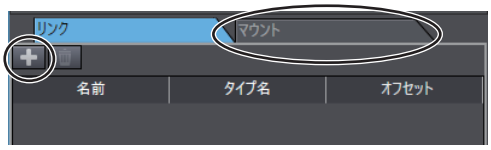
1 **マウントポイントを設定する 3D 形状データを 3D 表示画面またはマルチビューエクスプローラでダブルクリックします。**



対象 3D 形状データの設定画面が表示されます。



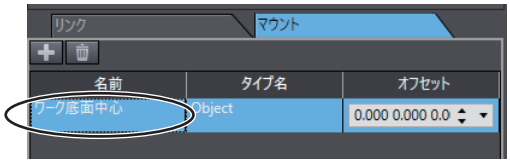
2 **［マウント］タブをクリックし、［追加］ボタンをクリックします。**



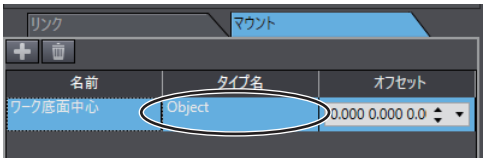
一覧に新しいマウントポイントの行が追加されます。



3 **マウントポイントの名前を入力し、Enter キーを押して確定します。**



4 **マウントポイントのタイプを入力し、Enter キーを押して確定します。**
ここでは「Object」を設定しています。

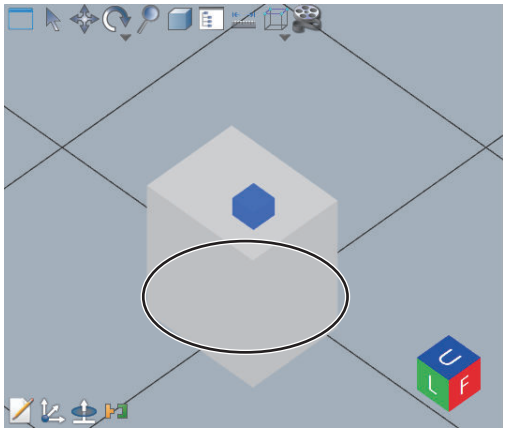


5 **マウントポイントのオフセットを設定します。**
［オフセット］の値を直接変更するか、3D 編集エリア上でマウントポイントを移動させて設定します。3D 編集エリア上での設定方法は「5-4-4 オフセットの設定方法 (P.5-29)」を参照してください。

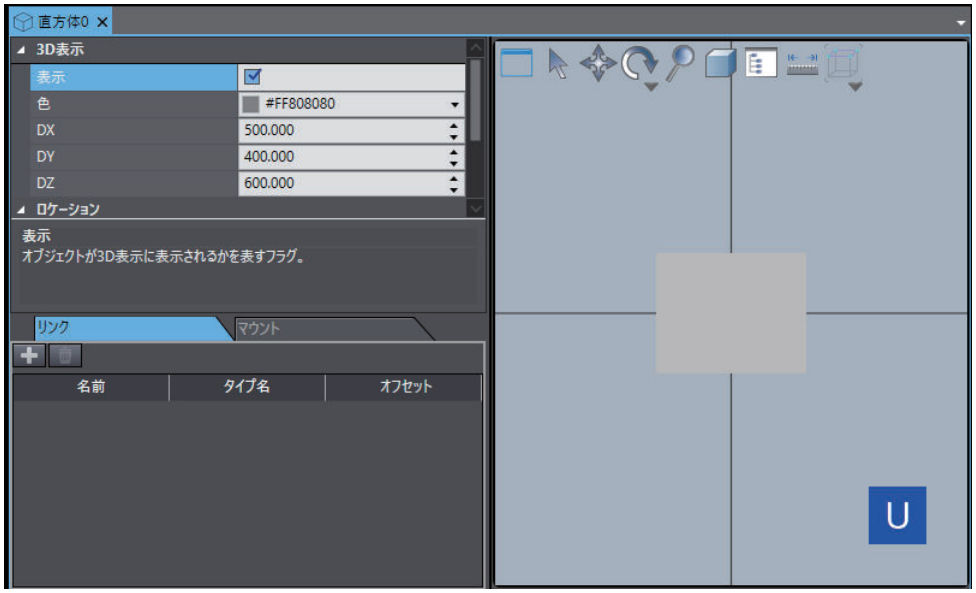
5-4-3 リンクポイントの設定

リンクポイントを設定する手順を説明します。

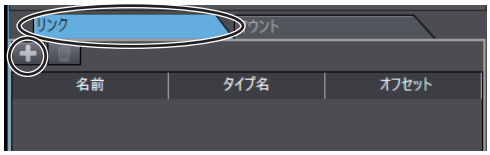
1 **リンクポイントを設定する 3D 形状データを 3D 表示画面またはマルチビューエクスプローラでダブルクリックします。**



対象 3D 形状データの設定画面が表示されます。



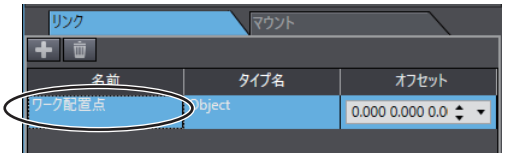
2 **リンクタブをクリックし、[追加] ボタンをクリックします。**



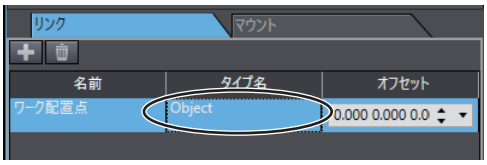
一覧に新しいリンクポイントの行が追加されます。



3 **リンクポイントの名前を入力し、Enter キーを押して確定します。**



4 **リンクポイントのタイプを入力し、Enter キーを押して確定します。**
ここでは「Object」を設定しています。



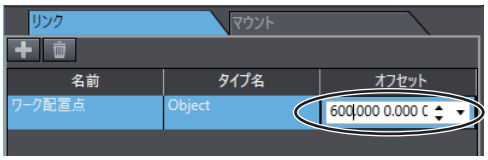
5 **リンクポイントのオフセットを設定します。**
［オフセット］の値を直接変更するか、3D 編集エリア上でリンクポイントを移動させて設定します。3D 編集エリア上での設定方法は「5-4-4 オフセットの設定方法 (P.5-29)」を参照してください。

5-4-4 オフセットの設定方法

3D 編集エリア上でマウントポイント、リンクポイントのオフセットを設定する方法は、3 とおりあります。
- オフセット座標の直接入力
- アイコンを使用する方法
- スナップ機能を使用する方法

オフセット座標の直接入力

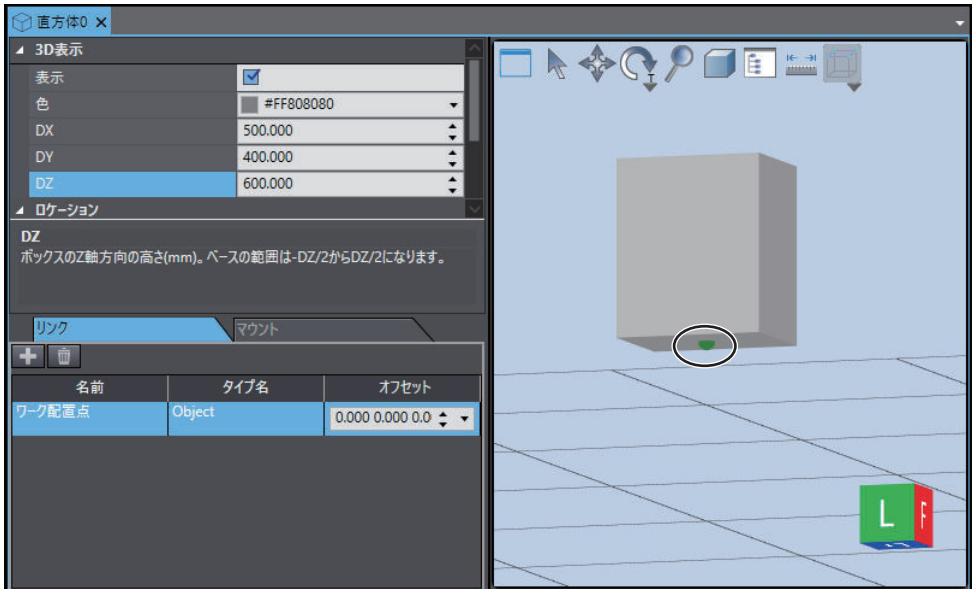
マウントポイント、リンクポイントの［オフセット］の座標値を直接入力します。



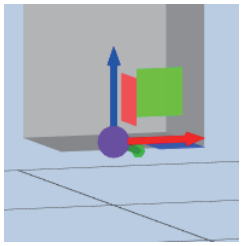
アイコンを使用したオフセット設定

リンクポイントを例に操作方法を説明します。マウントポイントの場合も同じです。

1 **緑色のリンクポイントアイコンをクリックします。**
通常は原点（X=0、Y=0、Z=0）にあります。

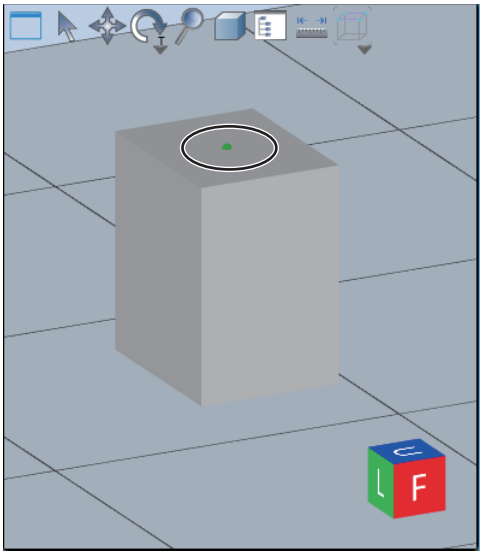


リンクポイントを移動させるためのアイコンが表示されます。



アイコン	名称	機能
	X 方向平行移動	アイコンをドラッグすると、X 方向に平行に移動します。
	Y 方向平行移動	アイコンをドラッグすると、Y 方向に平行に移動します。
	Z 方向平行移動	アイコンをドラッグすると、Z 方向に平行に移動します。
	YZ 平面上移動	アイコンをドラッグすると、YZ 平面上を移動します。
	XZ 平面上移動	アイコンをドラッグすると、XZ 平面上を移動します。
	XY 平面上移動	アイコンをドラッグすると、XY 平面上を移動します。

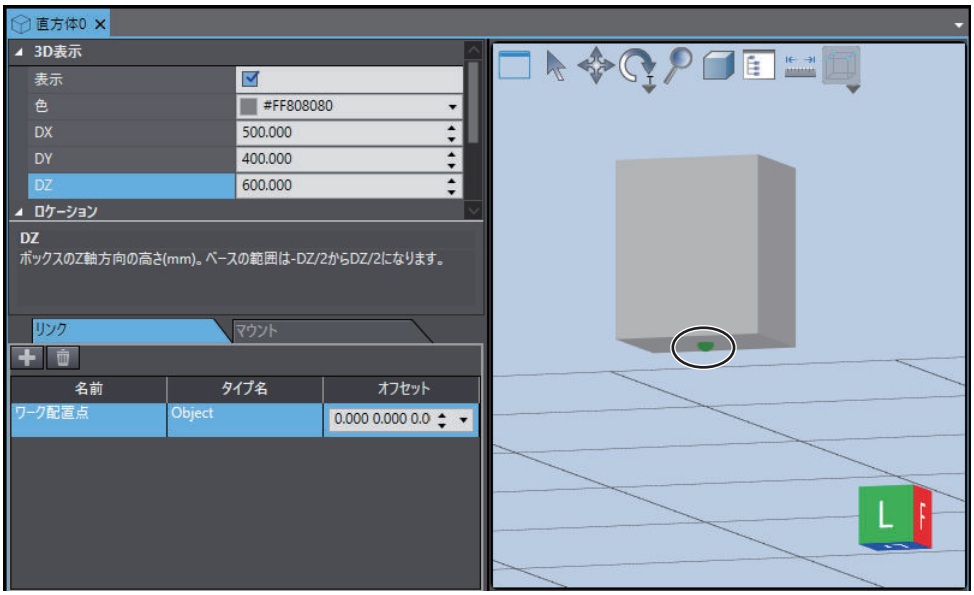
2 アイコンをドラッグしてリンクポイントを目的の位置に移動します。



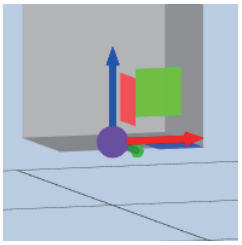
スナップ機能を使用したオフセット設定

リンクポイントを例に操作方法を説明します。マウントポイントの場合も同じです。

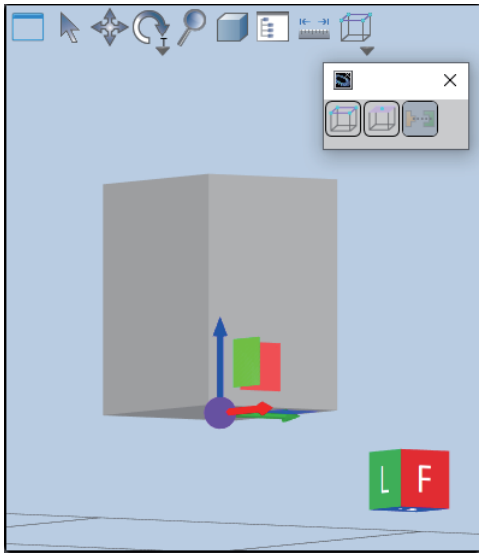
1 **緑色のリンクポイントアイコンをクリックします。**
通常は原点（X=0、Y=0、Z=0）にあります。



リンクポイントを移動させるためのアイコンが表示されます。



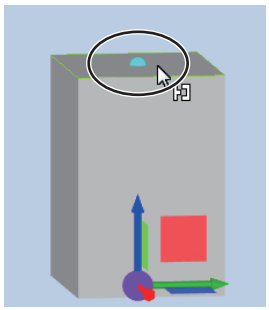
2 **スナップアイコンをクリックし、面の重心へのスナップボタンをクリックします。**



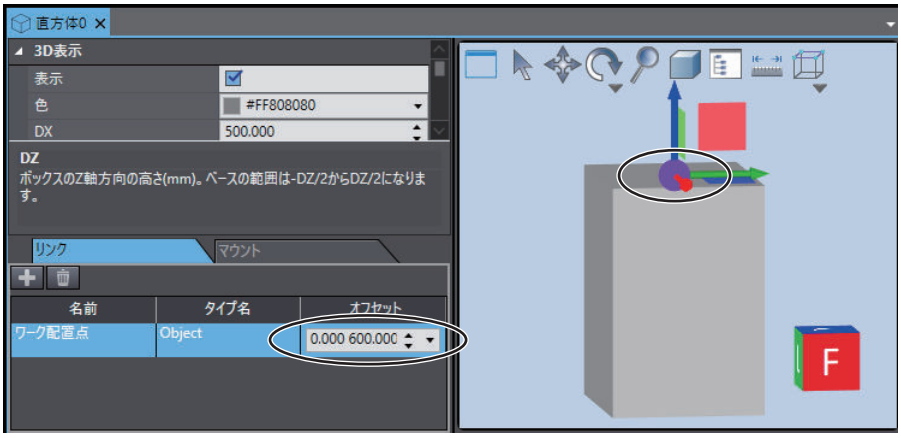
アイコンが変化します。



3 **リンクポイントを設定する面の重心近辺にカーソルを移動します。**
3D 形状データの面が青色に、3D 形状データの面の重心に水色の点が表示されます。

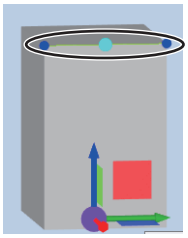


4 **水色の点をクリックします。**
3D 編集エリア上でリンクポイントが移動し、［オフセット］にも値が設定されます。



参考

エッジへのスナップボタンを使用した場合、マウスカーソル近辺の線分の端点と線分の中間の点がリンクポイントの候補として表示されます。

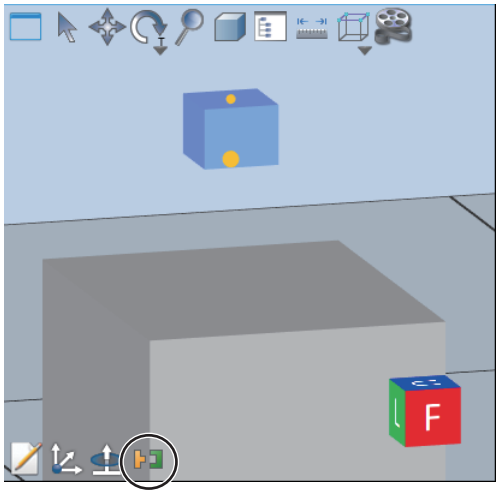


5-4-5 マウントポイントとリンクポイントを使用した 3D 形状データの配置

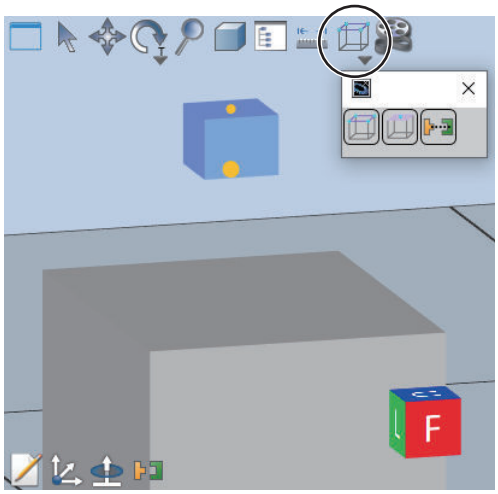
マウントポイントとリンクポイントを設定した 2 つの 3D 形状データを配置する場合は、3D 表示画面でスナップ機能を使用します。
子の 3D 形状データのマウントポイントを選択し、親の 3D 形状データのリンクポイントへスナップします。

1 **3D 表示画面で 3D 形状データを選択します。**

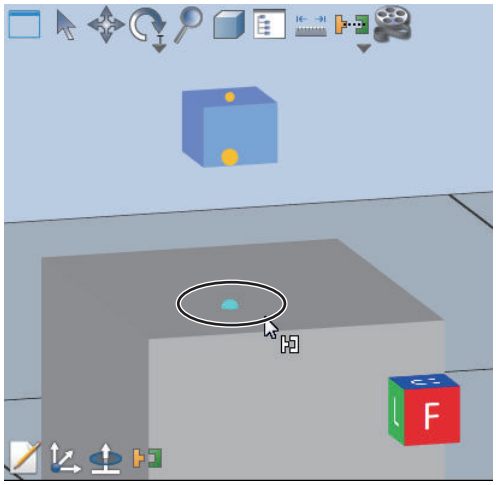
2 **［マウントポイント表示］アイコンをクリックしてマウントポイントを表示し、3D 表示画面上で選択します。**



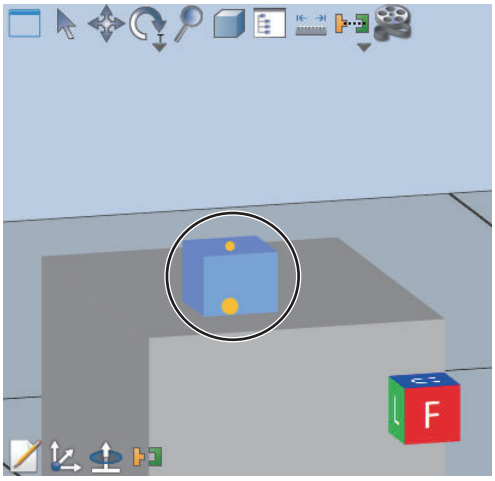
3 **［スナップ］アイコンをクリックして、［リンクポイントへのスナップ］を選択します。**



4 **リンクポイントを選択します。**



3D 形状データがリンクポイントへ移動します。


F

6

3D 形状データを動作させる設定およびスクリプトの作成

本章では、3D シミュレーション上でのワークやメカニカル機構などの 3D 形状データの動作を実現する設定およびスクリプトの作成手順について説明します。

6-1　3D 形状データを動作させる設定およびスクリプトの概要 6-2
6-1-1　シェイプスクリプトの概要 6-2
6-1-2　シェイプスクリプトとプログラム実行のタイミングと周期 6-3

6-2　メカニカル機構の設定 6-10
6-2-1　リミットスイッチの仮想出力用スクリプト生成 6-10

6-3　ワークの動作スクリプトの作成 6-12
6-3-1　シェイプスクリプトの追加 6-12
6-3-2　シェイプスクリプトの実行設定 6-12

6-4　ワーク検出センサの動作設定 6-16

6-5　シェイプスクリプトエディタ 6-18
6-5-1　シェイプスクリプトエディタ画面 6-18
6-5-2　シェイプスクリプトのプログラミング 6-21

6-6　シェイプスクリプトファンクション 6-24
6-6-1　シェイプスクリプトファンクションの追加 6-24
6-6-2　デフォルトファンクションおよびユーザファンクションの参照 6-25

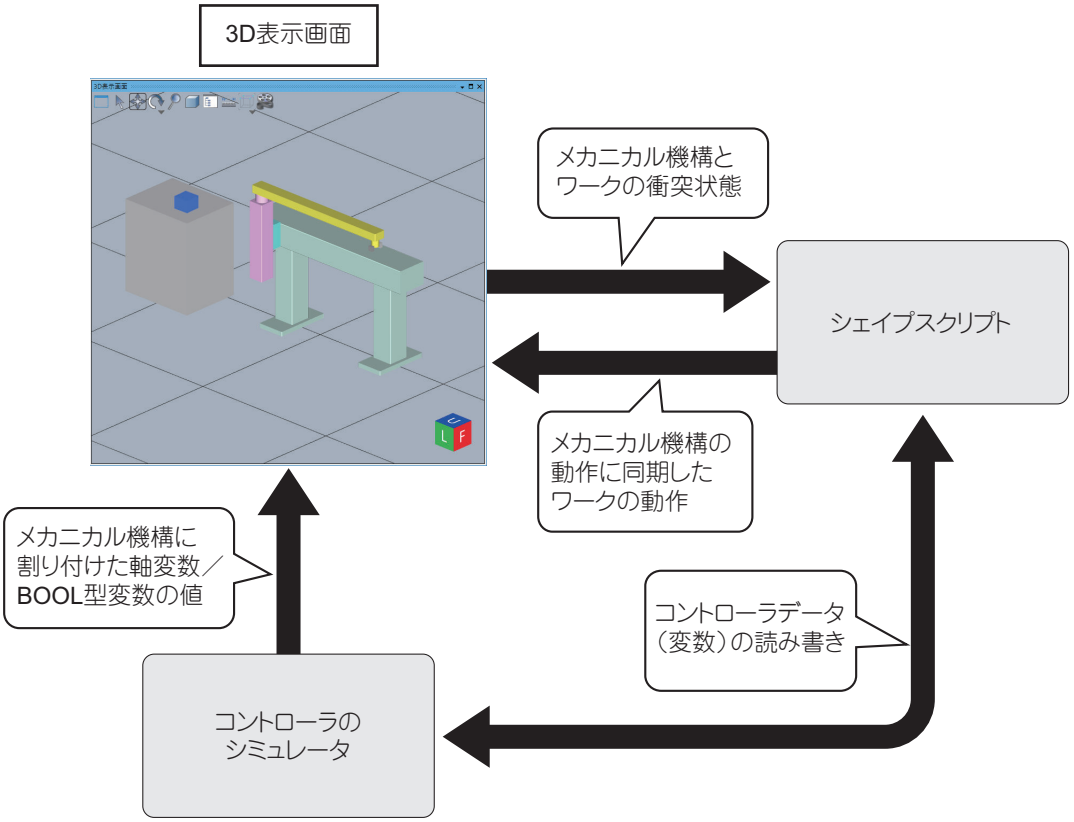
6-1 3D 形状データを動作させる設定およびスクリプトの概要

ワークや装置の動きを 3D シミュレーション上で実現するためのプログラムの作成手順を説明します。
ワークの動作プログラムは、C#言語の仕様に準拠した「シェイプスクリプト」として作成します。
また、次の項目については、動作を定義したスクリプトを自動生成することができます。

- リミットスイッチの仮想出力用スクリプト
- 仮想ワーク検出センサの動作スクリプト
- アプリケーションマネージャ Ver.5.0 以降で、シェイプスクリプトの関数ライブラリ機能として「シェイプスクリプトファンクション」が利用できます。
  シェイプスクリプトファンクションの作成方法や、シェイプスクリプトから呼び出す方法については、「6-6 シェイプスクリプトファンクション (P.6-24)」を参照してください。

6-1-1 シェイプスクリプトの概要

シェイプスクリプトとは、3D 表示画面に表示される仮想装置モデルのメカニカル機構の動作時に、メカニカル機構と連動するワークの動作を定義するプログラムです。
シミュレーション実行時にコントローラのプログラムを実行し、メカニカル機構に割り付けた軸変数や BOOL 型変数の値が変化すると、変数の値の変化に応じて、3D 表示画面上のメカニカル機構が動作します。
コントローラのシミュレーション実行と同時にシェイプスクリプトを実行させることにより、ワークの動作も 3D 表示画面に表示させることができます。
シェイプスクリプトは、メカニカル機構とワークの衝突状態を 3D 表示画面から衝突検知機能により検出し、これを条件にメカニカル機構の動作に同期したワークの動作を作り出します。これにより、3D 表示画面でメカニカル機構の動作に合わせてワークが移動する表示を実現しています。



参考

シェイプスクリプトは実際の装置の動作に影響しません。3D シミュレーションのためのプログラムです。

6-1-2 シェイプスクリプトとプログラム実行のタイミングと周期

シェイプスクリプトは、シェイプスクリプトシーケンスに割り付けて実行します。シェイプスクリプトの実行設定については、「6-3-2 シェイプスクリプトの実行設定 (P.6-12)」を参照してください。
シェイプスクリプトを実行すると、シェイプスクリプト内の Render()関数に記載したスクリプトが定期的に実行されます。実行のタイミングは、シェイプスクリプトシーケンスの実行設定から変更することができます。実行タイミングの設定には、対象コントローラを設定しない「非同期実行」と、対象コントローラを設定する「同期実行」があります。ここでは非同期実行時の実行タイミングと、同期実行時の実行周期を説明します。

非同期実行と同期実行の違いは、以下のとおりです。

比較項目	非同期実行	同期実行
コントローラのシミュレータの実行速度	速い	遅い
衝突検知を見逃す確率	同期実行と比較すると高い	実行回数が小さいほど低い

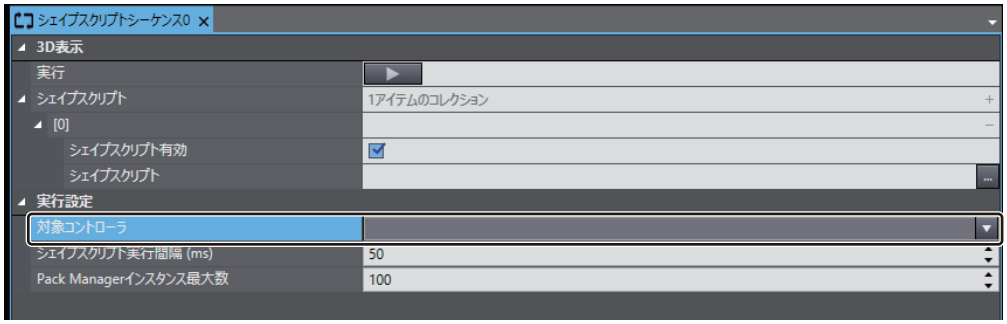
比較項目	非同期実行	同期実行
コントローラのシミュレータとシェイプスクリプト間の信号、変数の送受信のタイミング	シェイプスクリプトシーケンスの「シェイプスクリプト実行間隔」の設定に基づいた実行間隔ごとに、コントローラのシミュレータとシェイプスクリプトの間の送受信処理が実行される。 実行間隔はコントローラのシミュレータの内部時刻に関係なく計測されるため、コントローラのシミュレータとシェイプスクリプト間の信号、変数は同期しない。	シェイプスクリプトシーケンスの「シェイプスクリプト実行数」の設定に基づいた実行間隔ごとに、コントローラのシミュレータとシェイプスクリプトの間の送受信処理が実行される。 実行間隔はコントローラのシミュレータの内部時刻に基づいて計測されるため、コントローラのシミュレータとシェイプスクリプト間の信号、変数は同期する。

以降では、非同期実行時の実行タイミングと、同期実行時の実行周期について詳細を説明します。

シェイプスクリプトとプログラムの非同期実行と実行タイミング

非同期実行とは、コントローラのシミュレータとシェイプスクリプトを同期せずに互いに独立して実行することです。コントローラのシミュレータとシェイプスクリプトが互いの実行状態の待ち合わせを行わないため、シミュレータの実行速度は速くなりますが、互いの信号の送受信や 3D 表示画面上の 3D 形状データの衝突検知のタイミングは不正確です。また、コントローラのシミュレータが一時停止した場合でも、シェイプスクリプトは実行され続けます。

シェイプスクリプトを非同期で実行するためには、シェイプスクリプトシーケンスの［対象コントローラ］を［なし］に設定してください。



非同期実行の場合、シェイプスクリプトの Render()関数は、シェイプスクリプトシーケンスの［シェイプスクリプト実行間隔］で設定した間隔で実行されます。［シェイプスクリプト実行間隔］に 50ms を設定した場合、シェイプスクリプトの Render()関数は 50ms に 1 回実行されます。より細かいサイクルで実行したい場合は小さい値を、粗いサイクルで実行したい場合は大きい値を設定してください。

「シェイプスクリプト実行間隔」を 50ms に設定した場合と、200ms に設定した場合の違いについて説明します。
コンベアの上を 100mm/s の速度で動くワークをシェイプスクリプトで制御するときのスクリプトの例を、以下に示します。

```
private DateTime previousTime;

/// <summary>
/// Called to render the Shape Script object
/// </summary>
/// <returns>The list of shapes to render</returns>
public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
    int speed = 100;        // 100mm/s
    DateTime currentTime = this.GetCurrentControllerTime("new_Controller_0");

    if (this.IsInitialized)
    {
        var delta = currentTime - this.previousTime;
        var distance = delta.TotalMilliseconds * speed / 1000;
        IShapeBase workpiece = (IShapeBase) ace["/ApplicationManager0/Workpiece"];
        workpiece.OffsetFromParent = new Transform3D(
            workpiece.OffsetFromParent.DX,
            workpiece.OffsetFromParent.DY + distance,
            workpiece.OffsetFromParent.DZ);
    }

    this.previousTime = currentTime;

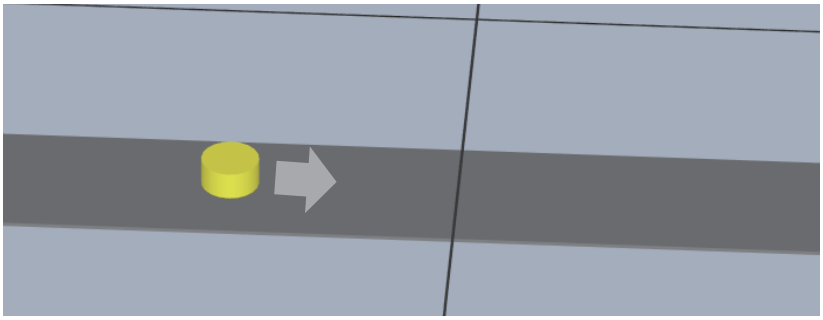
    this.InitializeObject(renderInfoList);
}
```

コントローラのシミュレータの現在時刻を取得

前回のRender()関数実行からの経過時間を計算し、ワークの移動距離を求める

ワークを移動

スクリプトを実行すると、3D 表示画面上のワークがコンベア上を移動します。



実行間隔を 50ms に設定した場合、コンベアの速度が 100mm/s のため、ワークは 3D 表示画面上で 5mm 動くと描画されます。
対して、実行間隔を 200ms に設定した場合、ワークは 3D 表示画面上で 20mm 動くと描画されます。つまり、実行間隔が 50ms に比べて描画が粗くなります。この 20mm の間でワークが障害物に衝突することがあっても、検知はされません。

したがって、実行間隔を短くした場合、描画は滑らかになり、障害物との衝突を検知できる可能性が高くなります。しかし、実行回数が多くなる分、PC に対する負荷が大きくなり、コントローラのシミュレータの実行が遅くなるおそれがあります。
実行間隔を長くした場合、実行回数が少なくなる分、PC に対する負荷は小さくなり、コントローラのシミュレータの実行が遅くなるおそれが軽減されます。しかし、描画が粗くなるため、障害物との衝突検知を見逃してしまう可能性が高くなります。

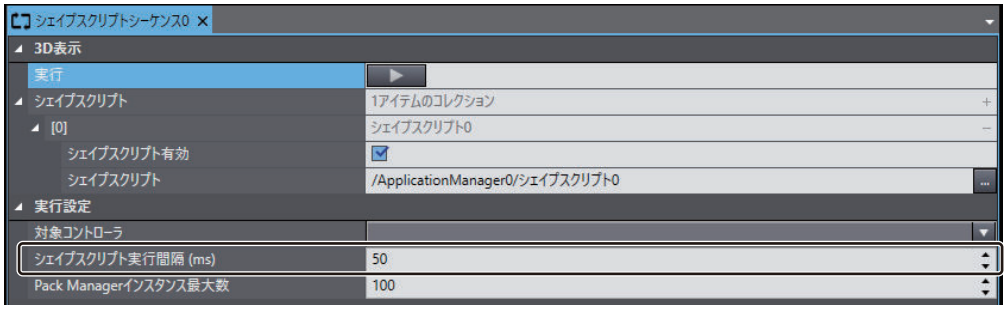
また、［シェイプスクリプト実行間隔］に適切な値が設定されていない場合や、シェイプスクリプトの実行時間が設定された実行間隔を超えていた場合は、以下の現象が発生することがあります。

- 3D 表示画面上で、ワークが仮想ワーク検出センサの位置を過ぎてから仮想ワーク検出センサに対応付けた I/O 変数の値が変化する。
- テンプレート関数の LoadPart()関数などの、コントローラの変数の値の変化に応じてワークの表示状態または位置を更新する処理を実行した場合に、変数の値が変化してからワークの表示状態または位置が更新されるまでに想定以上に時間がかかる。
- テンプレート関数の ClampPart()関数などのワークの把持を行う処理を実行した場合に、メカニカル機構とワークが接触したときではなく、接触後、メカニカル機構の位置が変わったあとにワークが追従を開始する。

- テンプレート関数の MoveObjectOnBelt()関数などの、コンベア上を流れるワークに対する処理を実行した場合に、ワークの描画が粗い。

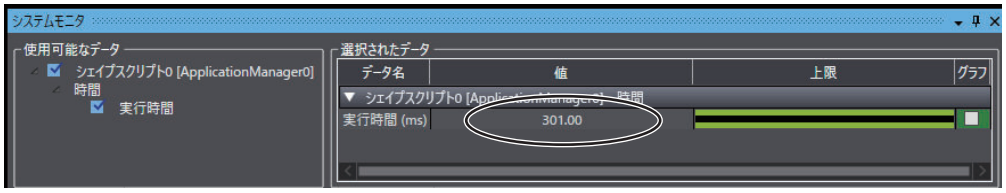
このような現象が発生したときの改善手順を以下に示します。

1 **［シェイプスクリプト実行間隔］を短く設定します。**



［シェイプスクリプト実行間隔］を短く設定すると、検知したい現象が発生してから処理を行うまでの時間が短縮されます。たとえば、ワークが仮想ワーク検出センサと接触してから、仮想ワーク検出センサに対応付けられた I/O 変数を変化させるシェイプスクリプトがより早く実行されるようになり、期待どおりのタイミングで I/O 変数の値を変化させることができます。

2 **［シェイプスクリプト実行間隔］を短くしても改善しない場合、シェイプスクリプトの実行時間が設定された実行間隔を超えている可能性があります。シェイプスクリプトの実行時間が短くなるようにシェイプスクリプトの記述を変更します。アプリケーションマネージャのシステムモニタからシェイプスクリプトの実行時間を確認します。**



シェイプスクリプトに、コントローラの変数の読み出し／書き込みなどの時間を要する処理を多数記述している場合や、パソコンのスペックや使用状態によってシェイプスクリプトの実行時間が長くなった場合は、シェイプスクリプトの実行時間が［シェイプスクリプト実行間隔］を超えることがあります。超えていた場合は、シェイプスクリプトの実行時間が短くなるようにシェイプスクリプトの記述を変更してください。
コントローラの変数の読み書き処理を多数記述している場合は、変数をひとつひとつ読み書きするのではなく、複数の変数をまとめて読み書きするように変更します。

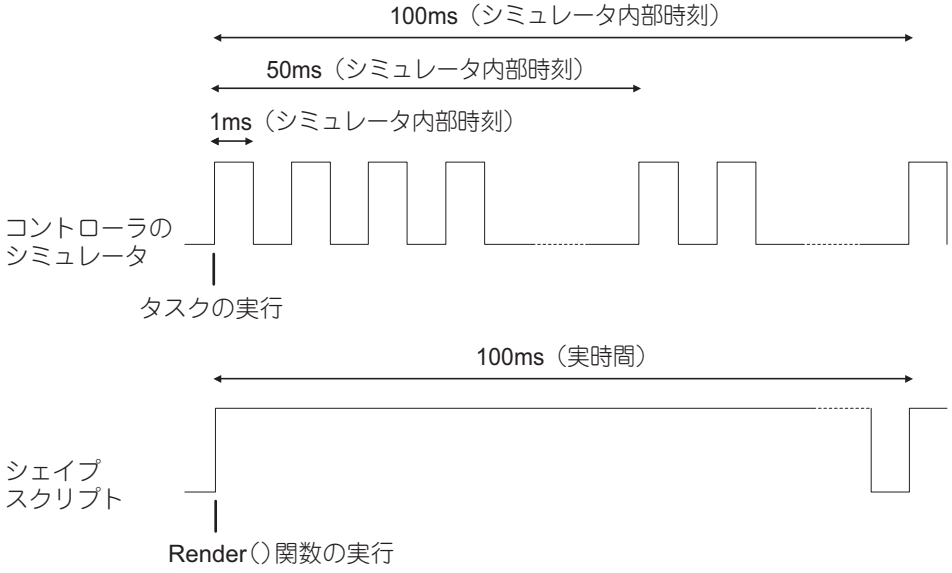
3 **シェイプスクリプトの記述を変更しても改善しない、またはシェイプスクリプトの変更自体が難しい場合は、コントローラのシミュレータの実行速度を遅くします。シミュレータの実行速度は、［シミュレーション］ウィンドウの「シミュレーション速度変更スライダ」をドラッグして変更してください。**



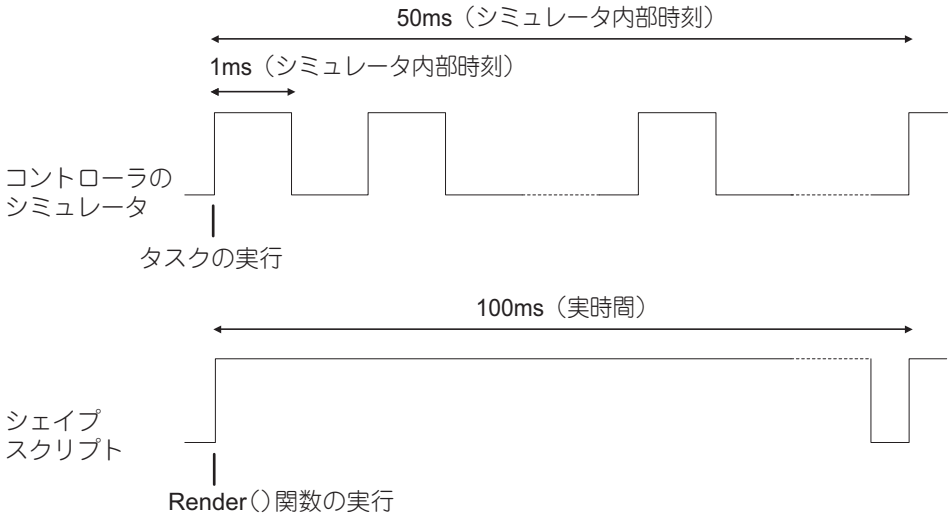
コントローラのシミュレータの実行速度を変更しても、シェイプスクリプトの実行時間は変わりません。

前述のコンベアの例において、［シェイプスクリプト実行間隔］の設定が 50ms、シェイプスクリプトの実行時間が 100ms の場合、シミュレータの実行速度が等倍のときは、ワークは 10mm 進むごとに描画されます。
これに対して、コントローラのシミュレータの実行速度を 0.5 倍に変更すると、シミュレータ内部時刻で 50ms 経過したときにシェイプスクリプトが実行されます。つまり、シミュレータの内部時刻の経過速度は半分になり、［シェイプスクリプト実行間隔］の設定と同じ 50ms の間隔でシェイプスクリプトが実行されます。このとき、ワークは 5mm 進むごとに描画されるため、3D 表示画面上のワークはシミュレータの実行速度が等倍の場合に比べて滑らかに動きます。
これらをタイミングチャートに表すと、それぞれ次のようになります。

● コントローラのシミュレータの実行速度が等倍の場合



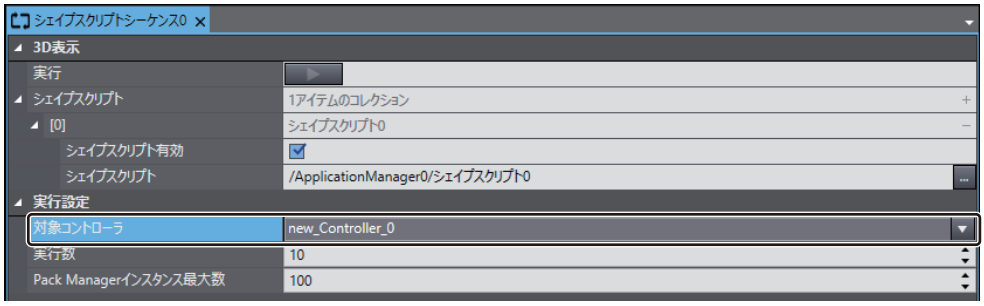
● コントローラのシミュレータの実行速度が 0.5 倍の場合



シェイプスクリプトとプログラムの同期実行と実行周期

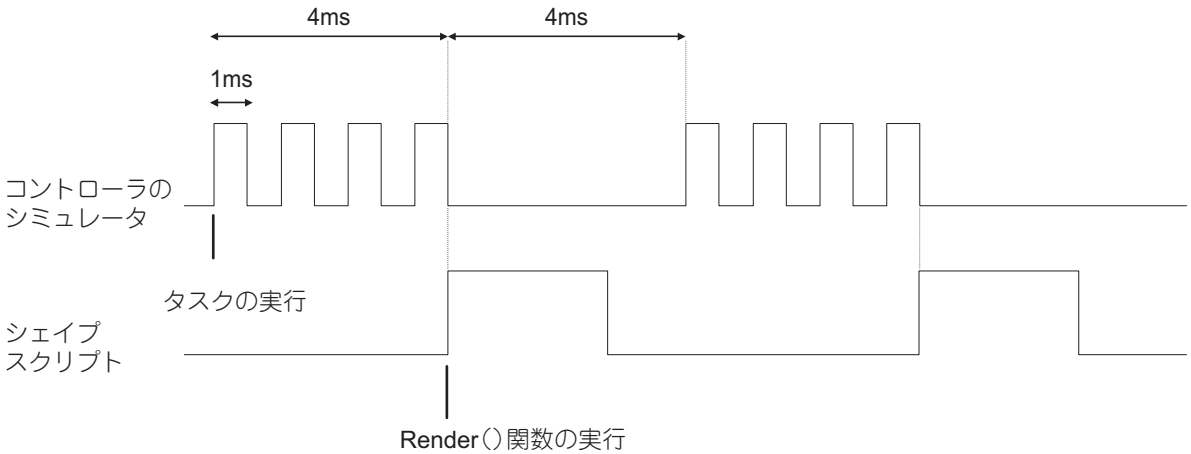
同期実行とは、コントローラのシミュレータとシェイプスクリプトの実行時間を管理して、同期させることです。コントローラのシミュレータのタスクの実行完了を待って、シェイプスクリプトが実行されます。コントローラのシミュレータもまた、シェイプスクリプトの実行完了を待って、タスクが実行されます。互いに待ち合わせて実行するため、互いの信号の送受信や 3D 表示画面上の 3D 形状データの衝突検知のタイミングは正確になりますが、待ち合わせを行う分、シミュレータの実行速度は遅くなります。また、待ち合わせを行うため、コントローラのシミュレータが一時停止した場合、シェイプスクリプトの実行も止まります。逆に、シェイプスクリプトがブレークポイントで一時停止した場合、コントローラのシミュレータも止まります。

コントローラのシミュレータとシェイプスクリプトを同期実行するためには、シェイプスクリプトシーケンスの［対象コントローラ］に、同期したいプロジェクトに登録されているコントローラを設定してください。



同期実行の場合、シェイプスクリプトの Render()関数は、シェイプスクリプトシーケンスの［実行数］で設定した間隔で実行されます。実行数は、コントローラのシミュレータの 4ms を 1 回と数えます。実行数を 1 に設定したとき、コントローラのシミュレータのタスクが 4ms 分実行されると、シェイプスクリプトの Render()関数が 1 回実行されます。実行数を 2 に設定したとき、コントローラのシミュレータのタスクが 2×4ms 分、つまり 8ms 分実行されると、シェイプスクリプトの Render()関数が 1 回実行されます。

コントローラのシミュレータのタスク周期が 1ms、シェイプスクリプトシーケンスの［実行数］が 1 のとき、コントローラのシミュレータとシェイプスクリプト実行タイミングは以下のようになります。



図のように、コントローラのシミュレータのタスクが 4ms 分実行されると、シェイプスクリプトの Render()関数が実行されます。シェイプスクリプトの Render()関数の実行時間が 4ms より短い場合であっても、コントローラのシミュレータの次の 4ms 分のタスクは、Render()関数の実行から 4ms

後に実行されます。Render()関数の実行が 4ms を超える場合は、実行の完了を待ってから、コントローラのシミュレータの次の 4ms 分のタスクが実行されます。



参考

シェイプスクリプトシーケンスの［対象コントローラ］に、ロボット統合 CPU ユニットを選択することはできません。

6-2 メカニカル機構の設定

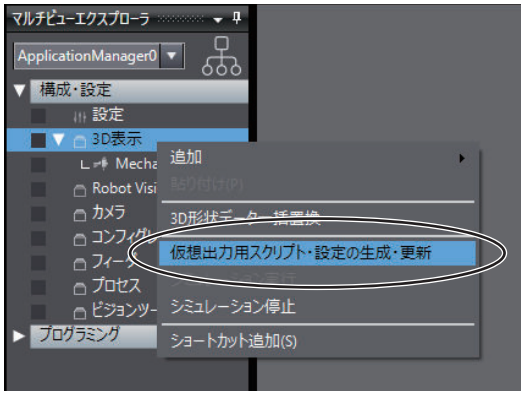
メカニカル機構の動作に関する設定を行います。
メカニカル機構の設定画面を表示し、［パラメーター設定］画面で対象のコントローラおよび変数が割り付けられていることを確認します。
項目の詳細は「4-3-5 メカニカル機構の設定項目 (P.4-25)」を参照してください。

6-2-1 リミットスイッチの仮想出力用スクリプト生成

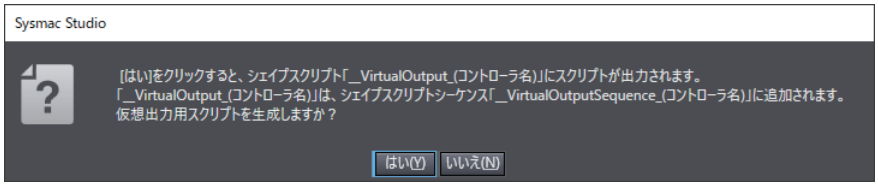
特定のメカニカル機構では、メカニカル機構が持つリミットスイッチの動きを 3D シミュレーション上で再現するための動作スクリプトを自動生成できます。

メカニカル機構	仮想出力の種類	説明
• エアシリンダ（シングルソレノイドタイプ） • エアシリンダ（ダブルソレノイドタイプ）	押出位置検出	ピストンの押し出し位置を検出します。ピストンが完全に押し出されたときに ON になります。
	引込位置検出	ピストンの引き込み位置を検出します。ピストンが完全に引き込まれたときに ON になります。
• ロボットツール（平行開閉型 2 爪タイプチャック／シングルソレノイドタイプ） • ロボットツール（平行開閉型 2 爪タイプチャック／ダブルソレノイドタイプ）	開位置検出	チャックが開くときの位置を検出します。チャックが完全に開いたときに ON になります。
	閉位置検出	チャックが閉じるときの位置を検出します。チャックが完全に閉じたときに ON になります。

1 マルチビューエクスプローラの［構成・設定］から［3D 表示］を右クリックし、メニューから［仮想出力用スクリプト・設定の生成・更新］を選択します。

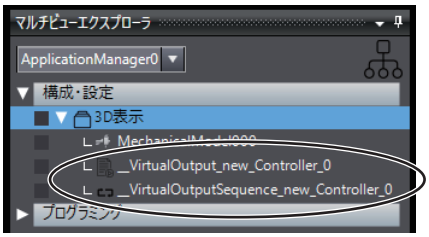


スクリプト生成・更新の確認ダイアログボックスが表示されます。



2 ［はい］ボタンをクリックします。

3D 表示の下にシェイプスクリプト［__VirtualOutput_（コントローラ名）］、およびシェイプスクリプトシーケンス［__VirtualOutputSequence_（コントローラ名）］が生成されます。すでに［__VirtualOutput_（コントローラ名）］、および［__VirtualOutputSequence_（コントローラ名）］が存在する場合、シェイプスクリプトおよびシェイプスクリプトシーケンスは更新されます。



生成されたスクリプトの実行方法は、「8-1 3D シミュレーションの操作手順 (P.8-2)」を参照してください。

使用上の注意

- メカニカル機構の種別が以下の場合、両方の仮想出力を同時に ON するとメカニカル機構が故障するため、同時に ON しないようにしてください。
  - エアシリンダ（ダブルソレノイドタイプ）：「押出位置検出」と「引込位置検出」
  - ロボットツール（平行開閉 2 爪タイプチャック／ダブルソレノイドタイプ）：「開位置検出」と「閉位置検出」
- 次の操作を行った場合は、再度、右クリックメニューから［仮想出力用スクリプト・設定の生成・更新］を選択してください。［仮想出力用スクリプト・設定の生成・更新］を選択すると、登録済みの［__VirtualOutput］、［__VirtualOutputSequence］が更新され、設定の変更が反映されます。
  - 対象のメカニカル機構の設定を変更した場合
  - シェイプスクリプトシーケンス［__VirtualOutputSequence］の、対象コントローラ、実行数またはシェイプスクリプト実行間隔を変更した場合

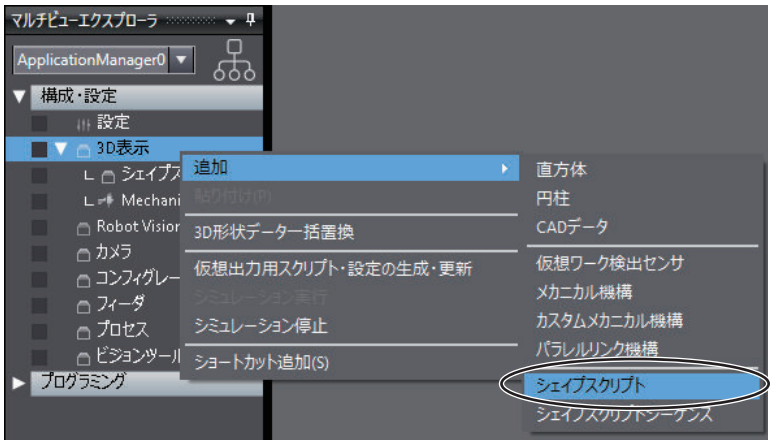
6-3　ワークの動作スクリプトの作成

ワークの動作を定義するシェイプスクリプトを追加し、C#言語でプログラムを作成します。
また、シェイプスクリプトシーケンスを追加し、シェイプスクリプトの実行順序を設定します。

6-3-1　シェイプスクリプトの追加

シェイプスクリプトの追加方法を説明します。

1　マルチビューエクスプローラから［構成・設定］の［3D 表示］を右クリックして、［追加］-［シェイプスクリプト］を選択します。



［3D 表示］の下に［シェイプスクリプト 0］が登録されます。
［シェイプスクリプトファンクション］の下に［Default Functions］が登録されます。
その登録される条件は下記のとおりです。

- アプリケーションマネージャのバージョンが 5.0 以上
- シェイプスクリプトの初回登録時

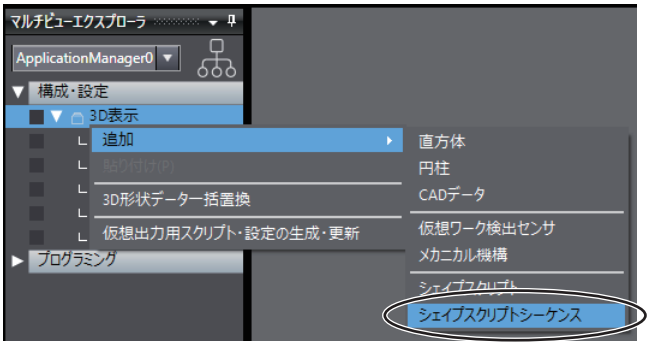
2　［シェイプスクリプト 0］をダブルクリックします。または、右クリックして［編集］を選択します。

シェイプスクリプトエディタ画面が表示されます。
シェイプスクリプトエディタで、ワークの動作を定義するプログラムを作成します。シェイプスクリプトエディタ画面の詳細は、「6-5 シェイプスクリプトエディタ (P.6-18)」を参照してください。

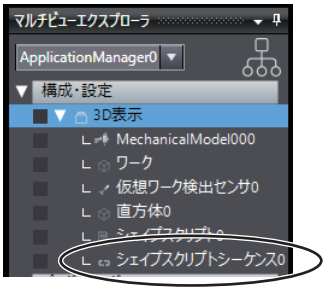
6-3-2　シェイプスクリプトの実行設定

ワークの動作を定義したシェイプスクリプトはシェイプスクリプトシーケンスに割り付けて実行します。
シェイプスクリプトの実行手順は以下のとおりです。

1　［構成・設定］の［3D 表示］を右クリックし、メニューから［追加］-［シェイプスクリプトシーケンス］を選択します。

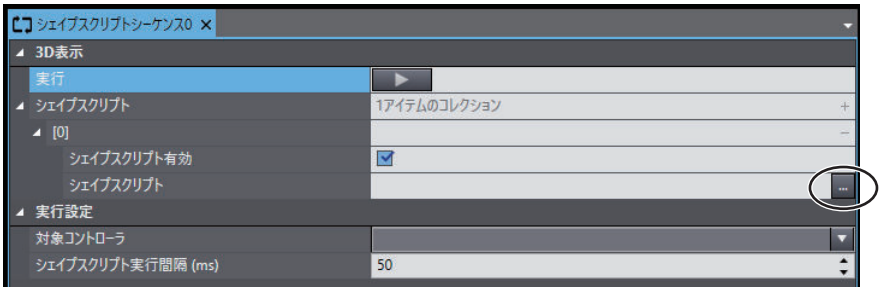


3D 表示下に［シェイプスクリプトシーケンス 0］が登録されます。



2 **シェイプスクリプトシーケンスをダブルクリックします。**
シェイプスクリプトシーケンスの設定画面が表示されます。

3 **シェイプスクリプトの参照ボタンをクリックします。**



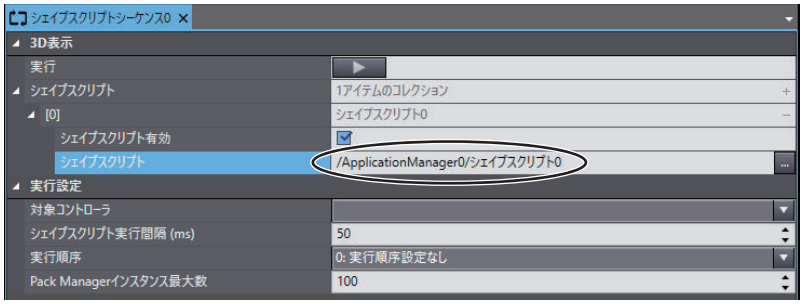
登録されているシェイプスクリプトの一覧が表示されます。



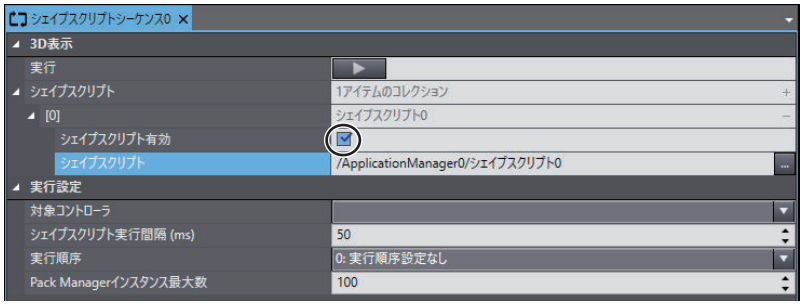
4 **動作させたいシェイプスクリプトを選択し、[選択] ボタンをクリックします。**



シェイプスクリプトがシェイプスクリプトシーケンスに割り付けられます。

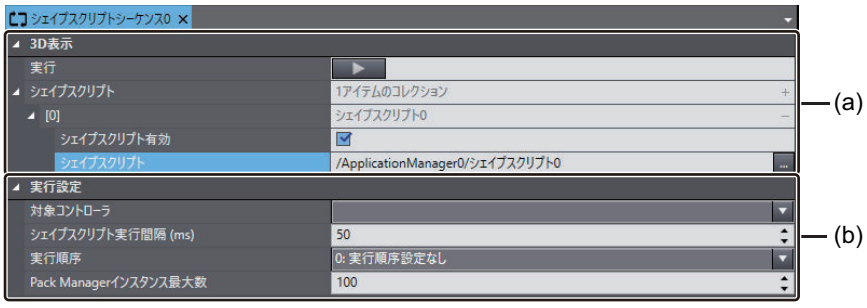


5 **[シェイプスクリプト有効] のチェックボックスにチェックを入れます。**



これでシェイプスクリプトの実行設定は完了です。

シェイプスクリプトシーケンスの設定項目は次のとおりです。



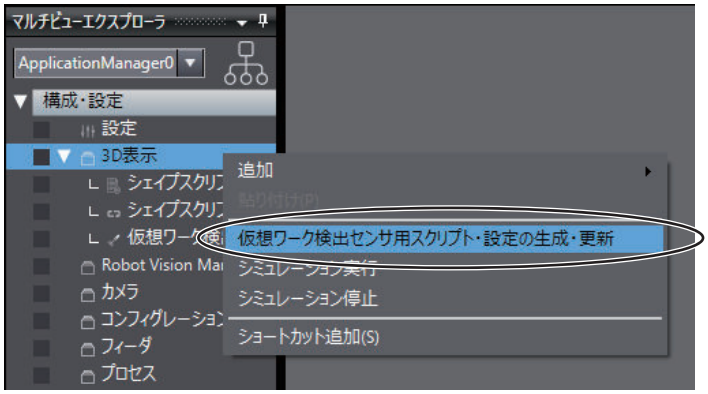
	設定項目		説明	設定値	初期値
(a)	3D 表示	実行	シェイプスクリプトの実行ボタンです。[シェイプスクリプト有効] にチェックを入れてボタンをクリックすると、登録されたシェイプスクリプトが実行状態になります。	-	-
		シェイプスクリプト	シェイプスクリプトシーケンスに割り付けられているシェイプスクリプトの数が表示されます。	-	-
		インデックス	シェイプスクリプトシーケンス実行時の実行順序を示します。*1	-	-
		+（追加）ボタン	割り付けるシェイプスクリプトを追加します。	-	-
		-（解除）ボタン	選択したシェイプスクリプトの割り付けを解除します。	-	-
		シェイプスクリプト有効	シェイプスクリプトを実行するかを切り替えます。	チェック ON／チェック OFF	チェック ON
		シェイプスクリプト	割り付けたシェイプスクリプト名が表示されます。右側のボタンから、シェイプスクリプトを選択します。	シェイプスクリプトの名称	なし
(b)	実行設定	対象コントローラ	シェイプスクリプトシーケンスを実行するコントローラを選択します。	プロジェクトに登録されているコントローラ	なし
		実行数*2	シェイプスクリプトの 1 回の実行あたりにコントローラで実行されるプライマリ定周期タスクのユーザプログラム実行回数を設定します。	1～1000	10
		シェイプスクリプト実行周期*2	シェイプスクリプトの実行周期を設定します。（単位: ms）	0 以上の整数値	50
		実行順序	同時に複数のシェイプスクリプトシーケンスが実行される場合の実行順序を選択します。	0: 順序に依存せず実行／1～プロジェクトに登録されているシェイプスクリプトシーケンスの数: 選択の順番で実行	0

*1. シェイプスクリプトの実行順序を変更する場合は、シェイプスクリプトを右クリックして［上へ移動］または［下へ移動］を選択します。
*2. 対象コントローラが選択されている場合は［実行数］が、選択されていない場合は［シェイプスクリプト実行周期］が表示されます。

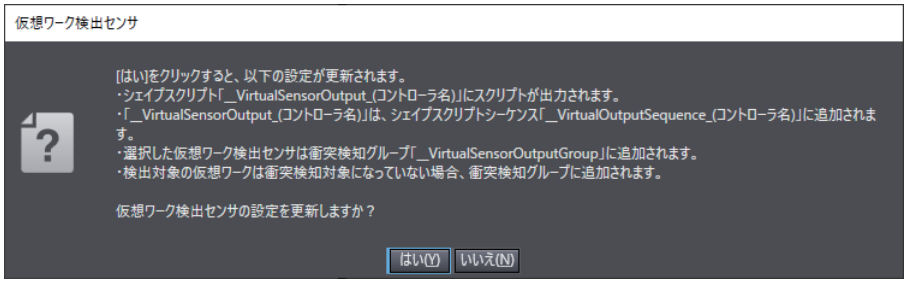
6-4　ワーク検出センサの動作設定

仮想ワーク検出センサがワークを検出する動作は、「仮想ワーク検出センサ用スクリプト」で定義します。
仮想ワーク検出センサ用スクリプトの追加手順は次のとおりです。

1　マルチビューエクスプローラの［構成・設定］から［3D 表示］を右クリックし、メニューから［仮想ワーク検出センサ用スクリプト・設定の生成・更新］を選択します。

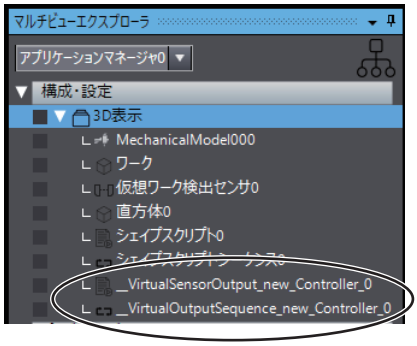


スクリプト生成・更新の確認ダイアログボックスが表示されます。



2　［はい］ボタンをクリックします。

3D 表示の下に、シェイプスクリプト［__VirtualSensorOutput_（コントローラ名）］、およびシェイプスクリプトシーケンス［__VirtualOutputSequence_（コントローラ名）］が生成されます。すでに［__VirtualSensorOutput_（コントローラ名）］、および［__VirtualOutputSequence_（コントローラ名）］が存在する場合、シェイプスクリプトおよびシェイプスクリプトシーケンスは更新されます。



生成されたスクリプトの実行方法は、「8-1 3D シミュレーションの操作手順 (P.8-2)」を参照してください。

使用上の注意

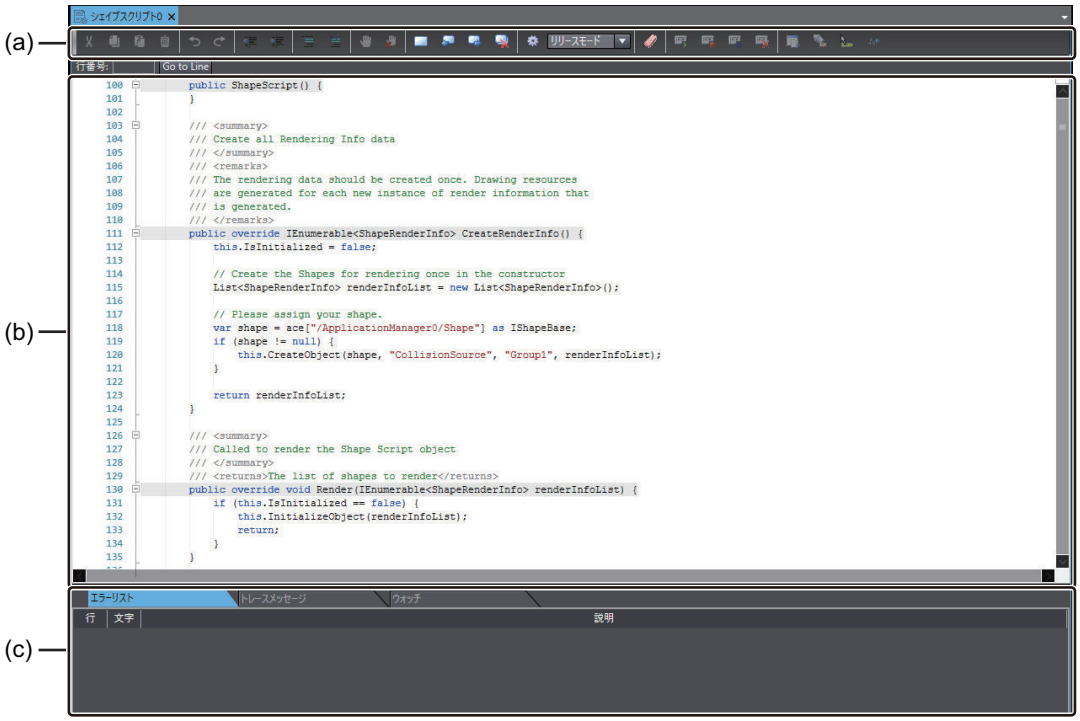
仮想ワーク検出センサの設定を変更した場合は、再度、右クリックメニューから［仮想ワーク検出センサ用スクリプト・設定の生成・更新］を選択してください。
［仮想ワーク検出センサ用スクリプト・設定の生成・更新］を選択すると、登録済みの［__VirtualSensorOutput］、［__VirtualOutputSequence］が更新され、設定の変更が反映されます。

6-5 シェイプスクリプトエディタ

シェイプスクリプトエディタは、ワークなど 3D 形状データの動作を定義する「シェイプスクリプト」を記述するエディタです。
シェイプスクリプトの言語仕様はマイクロソフト社の C#バージョン 6.0 を採用しています。変数、データ型、制御構文、関数、演算子などは C#言語の言語仕様に準拠しています。
C#言語の言語仕様については、マイクロソフト社が提供するリファレンスやガイド、または市販の C#言語の専門書を参照してください。
3D 形状データを動作させるシェイプスクリプトの関数の説明については、「A-1-1 関数一覧 (P.A-2)」を参照してください。

6-5-1 シェイプスクリプトエディタ画面

シェイプスクリプトエディタ画面の機能について説明します。

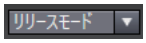


	項目	説明
(a)	ツールバー	スクリプト作成に使用する機能のボタンを表示します。詳細は、「シェイプスクリプトエディタのツールバー (P.6-18)」を参照してください。
(b)	エディタ	本エディタでスクリプトを作成します。詳細は、「6-5-2 シェイプスクリプトのプログラミング (P.6-21)」を参照してください。
(c)	タブ	エラー、トレースメッセージ、および変数の値を表示します。詳細は、「シェイプスクリプトエディタのタブ (P.6-20)」を参照してください。

シェイプスクリプトエディタのツールバー

ツールバーのボタンを使用して、以下の表の操作を実行します。

ボタン	操作（ショートカット）	機能
	切り取り(Ctrl+X)	選択したテキストを削除し、クリップボードに保存します。
	コピー(Ctrl+C)	選択したテキストをクリップボードに保存します。
	貼り付け(Ctrl+V)	クリップボードに保存されているテキストをカーソルの現在位置に貼り付けます。
	削除(Del)	選択したテキストを削除します。
	元に戻す(Ctrl+Z)	直前の編集前の状態に戻します。
	やり直す(Ctrl+Y)	元に戻す前の状態に戻します。
	インデント解除(Shift+Tab)	選択した行頭の字下げを戻します。
	インデント(Tab)	選択した行頭を字下げします。
	選択範囲のコメント	選択した行をコメントアウトします。
	選択範囲のコメントを解除	選択した行のコメントアウトを解除します。
	ブレークポイントを切り替え	デバッグモードのときだけ、カーソルの現在位置にスクリプトが停止するポイントを設定／解除します。
	ブレークポイントのクリア	デバッグモードのときだけ、すべてのブレークポイントを削除します。
	ブックマークの設定／解除	選択した行にブックマークを設定／解除します。
	前のブックマーク	前のブックマークにジャンプします。
	次のブックマーク	次のブックマークにジャンプします。
	ブックマークの解除	すべてのブックマークを削除します。
	コンパイル	スクリプトのエラーをチェックします。エラー内容をエラータブに表示します。

ボタン	操作（ショートカット）	機能
	エディタモード選択	［リリースモード］と［デバッグモード］を選択できます。デバッグモードはスクリプトのデバッグに使用します。
	トレースメッセージのクリア	トレースメッセージタブの内容を削除します。トレースメッセージについては「8-2-3 トレース文 (P.8-10)」を参照してください。
	記録したマクロの実行	記録したマクロを実行します。ボタンを押すたびに、マクロに記録したキー入力が行われます。
	マクロの記録	マクロとは、シェイプスクリプトエディタ上でのキー入力の操作を記録したものです。ボタンを押すと記録を開始し、もう一度ボタンを押すと記録を終了します。
	記録の一時停止	マクロの記録を一時停止します。もう一度ボタンを押すと記録を再開します。
	記録のキャンセル	マクロの記録をキャンセルします。
	オブジェクトメンバーリストの表示	C#オブジェクトのメンバーの候補リストを表示します。入力した C#オブジェクトの後ろにカーソルを合わせてボタンを押します。
	パラメータ情報の表示	カーソル位置の C#オブジェクトのパラメータの説明を表示します。
	クイック情報の表示	カーソル位置のエラーの情報をツールチップで表示します。
	単語補間の表示	入力途中の文字から C#オブジェクトの候補を表示します。
	行ジャンプボタン	入力された行番号にジャンプします。
	ステップオーバー	デバッグモードのときだけ表示されます。関数内に入らず、スクリプトをステップ実行します。
	ステップイン	デバッグモードのときだけ表示されます。関数内に入って、スクリプトをステップ実行します。
	開始	デバッグモードのときだけ表示されます。停止中のスクリプトを再開します。

シェイプスクリプトエディタのタブ

以下のタブを利用できます。

項目	説明
エラーリスト	コンパイルエラーのリストが表示されます。各エラーについて、行番号、エラー発生位置、およびエラーメッセージが表示されます。
トレースメッセージ	トレースメッセージ（スクリプトの Trace.WriteLine()コールに含まれるテキスト）およびワーニング情報が表示されます。

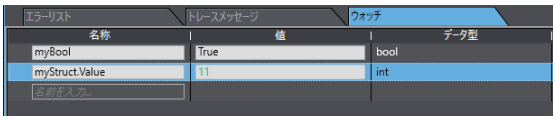
項目	説明
ウォッチ	シェイプスクリプトエディタ上の変数の値が表示されます。詳細は「シェイプスクリプトの［ウォッチ］タブ (P.6-21)」を参照してください。

● シェイプスクリプトの［ウォッチ］タブ

［ウォッチ］タブでは、シェイプスクリプトで使用している変数を登録し、一時停止中のシェイプスクリプトの変数の値を表示／変更することができます。

1 デバッグモードでブレークポイントを設定して実行、またはステップ実行を実施し、シェイプスクリプトの実行を一時停止します。

2 ［ウォッチ］タブで、変数名を登録します。



［値］列に、そのときの変数の値が表示されます。

参考

- 変数の値は、デバッグモードでブレークポイントまたはステップ実行で一時停止しているときだけ表示され、変更できます。それ以外のときは、値はグレーアウトされます。
- 値を変更する場合は、［値］列に直接値を入力し、Enter キーを押します。

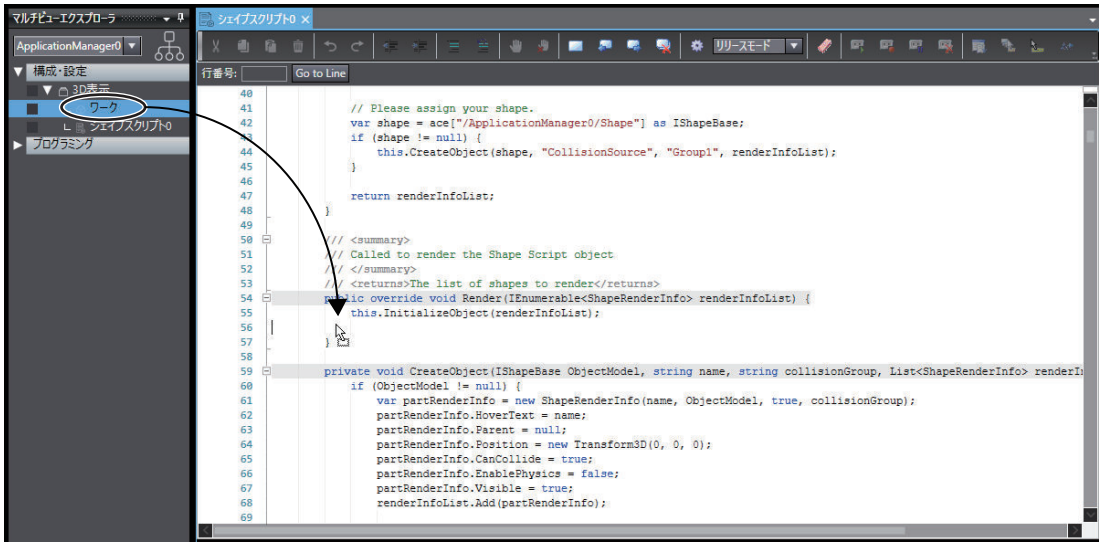
6-5-2 シェイプスクリプトのプログラミング

シェイプスクリプトエディタでプログラミングする際の操作方法について説明します。
シェイプスクリプトエディタでは、マルチビューエクスプローラのデータやツールボックスに登録されている関数をドラッグ&ドロップしてスクリプトを記述することができます。

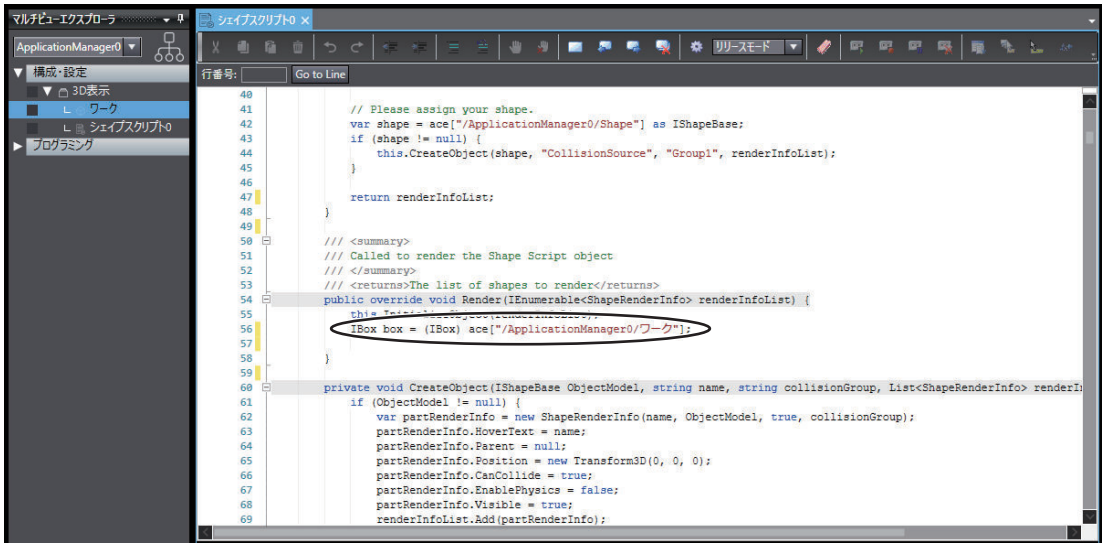
シェイプスクリプトでの 3D 形状データを表す変数宣言の作成

シェイプスクリプト上でワークなどの 3D 形状データを表す変数を使用する場合、変数宣言をする必要があります。
マルチビューエクスプローラの 3D 形状データの項目をシェイプスクリプトエディタ上にドラッグ&ドロップすることで、変数宣言を作成することができます。
ここでは、ワークの変数宣言を例として操作方法を説明します。

1 マルチビューエクスプローラの［構成・設定］から、［3D 表示］の下の［ワーク］をドラッグし、挿入する箇所にドロップします。



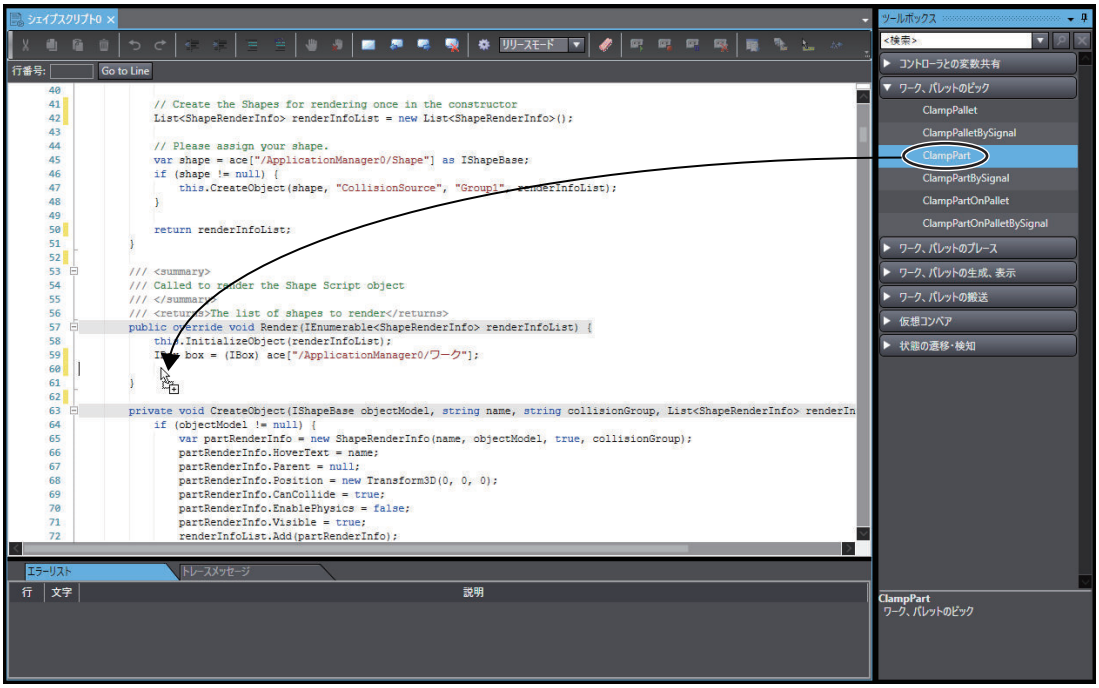
ドロップした箇所にワークの変数宣言文が挿入されます。



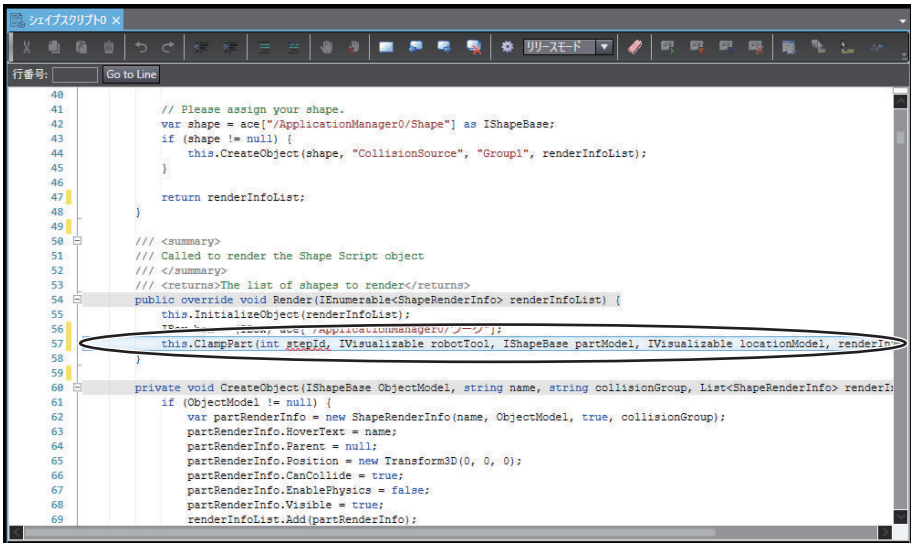
ツールボックスからシェイプスクリプトへの関数の挿入

ツールボックスから、シェイプスクリプトへ関数を挿入します。
ここでは、ワークをつかむ動作をする処理の関数の挿入を例として操作方法を説明します。

1 **ツールボックスの［Clamp part, pallet］から［ClampPart］をドラッグし、挿入する箇所にドロップします。**



ドロップした箇所に関数が挿入されます。



挿入された関数の引数に適切な変数を指定します。
関数の挿入が完了します。
Sysmac Studio に用意されているシェイプスクリプトの関数の詳細については、「A-1-1 関数一覧 (P.A-2)」を参照してください。

6-6 シェイプスクリプトファンクション

シェイプスクリプトファンクションは、シェイプスクリプト用の C#言語の関数を集めたライブラリです。
シェイプスクリプトファンクションには、下記の 2 種類があります。

(1)デフォルトファンクション
シェイプスクリプトで標準的に使用する関数を集めたライブラリです。シェイプスクリプト作成時、自動的に［Default Functions］が作成されます。また、［Default Functions］のほかに任意のデフォルトファンクションも登録することができます。Sysmac Studio のバージョンアップにより、デフォルトファンクションに新しい関数が追加されたり、既存の関数に機能が追加されたりすることがあります。これらの変更は、デフォルトファンクションのバージョンで管理されています。登録されるデフォルトファンクションのバージョンは、そのときの最新のバージョンとなります。
シェイプスクリプトで使用する関数については、「A-1 シェイプスクリプトで使用する関数 (P.A-2)」を参照してください。

(2)ユーザファンクション
ユーザが作成した関数を集めたライブラリです。複数登録することができます。

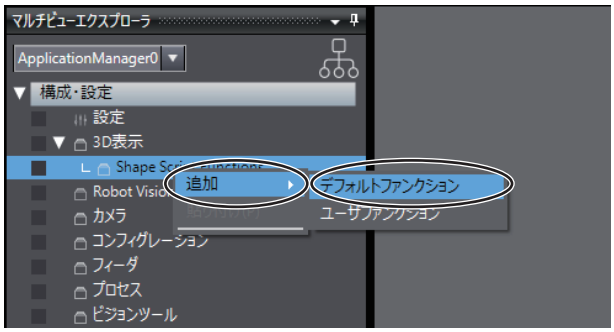
6-6-1 シェイプスクリプトファンクションの追加

デフォルトファンクションとユーザファンクションを登録する手順を説明します。

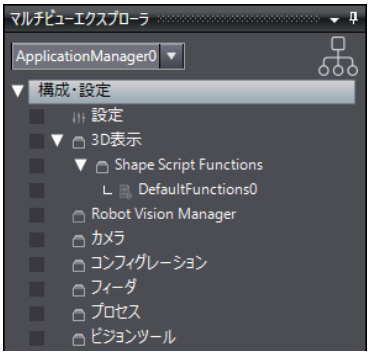
デフォルトファンクションの追加

デフォルトファンクションを追加する手順を説明します。

1 マルチビューエクスプローラの［構成・設定］から［3D 表示］-［シェイプスクリプトファンクション］を右クリックして、［追加］-［デフォルトファンクション］を選択します。



［3D 表示］の下に［シェイプスクリプトファンクション］、その下に［DefaultFunctions0］が登録されます。



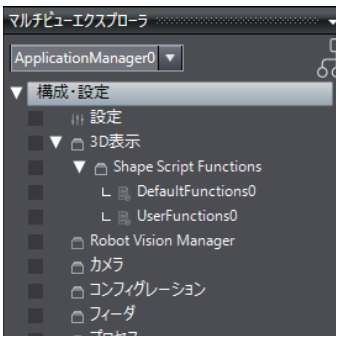
ユーザファンクションの追加

ユーザファンクションを追加する手順を説明します。

1 **マルチビューエクスプローラの［構成・設定］から［3D 表示］-［シェイプスクリプトファンクション］を右クリックして、［追加］-［ユーザファンクション］を選択します。**



［3D 表示］の下に［シェイプスクリプトファンクション］、その下に［UserFunctions0］が登録されます。



6-6-2　デフォルトファンクションおよびユーザファンクションの参照

シェイプスクリプトからデフォルトファンクション、およびユーザファンクションを参照する方法を説明します。

デフォルトファンクションの参照

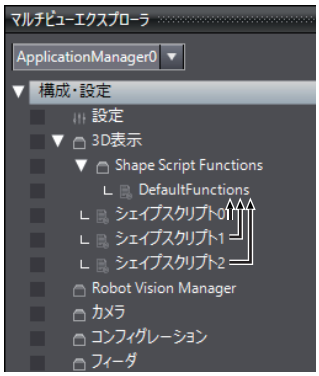
シェイプスクリプトからデフォルトファンクションを参照する方法を説明します。
追加したシェイプスクリプトは、自動的に［Default Functions］の関数を参照します。

デフォルトファンクション名.関数名（引数）;

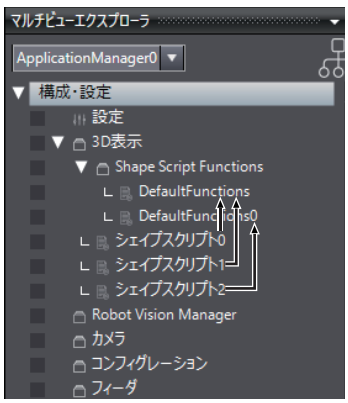
例:

```
DefaultFunctions.CreateObject(this, shape, "CollisionSource", "Group1", renderInfoList);
```

複数のシェイプスクリプトを作成した場合、［Default Functions］の関数を参照することで、複数のシェイプスクリプトに共通して使用できるようになります。



また、シェイプスクリプトで参照するデフォルトファンクションを［Default Functions］以外のデフォルトファンクションの関数を指定することができます。



ユーザファンクションの参照

シェイプスクリプトからユーザファンクションを参照する方法を説明します。
追加したユーザファンクション内の関数を指定する方法は下記のとおりです。

ユーザファンクション名.関数名（引数）;

例:

```
UserFunctions0.MySelectPalletFunction(this, stepId, controllerName, variableName, palletModel);
```

複数のシェイプスクリプトで使用する共通の関数をユーザが定義することができます。

7

ワークシミュレーション簡単設定

本章では、3D シミュレーション上でワークの動作を設定のみで実現する機能について説明します。

7-1　ワークシミュレーション簡単設定の概要 7-2

7-2　メカニカル機構の設定 7-3

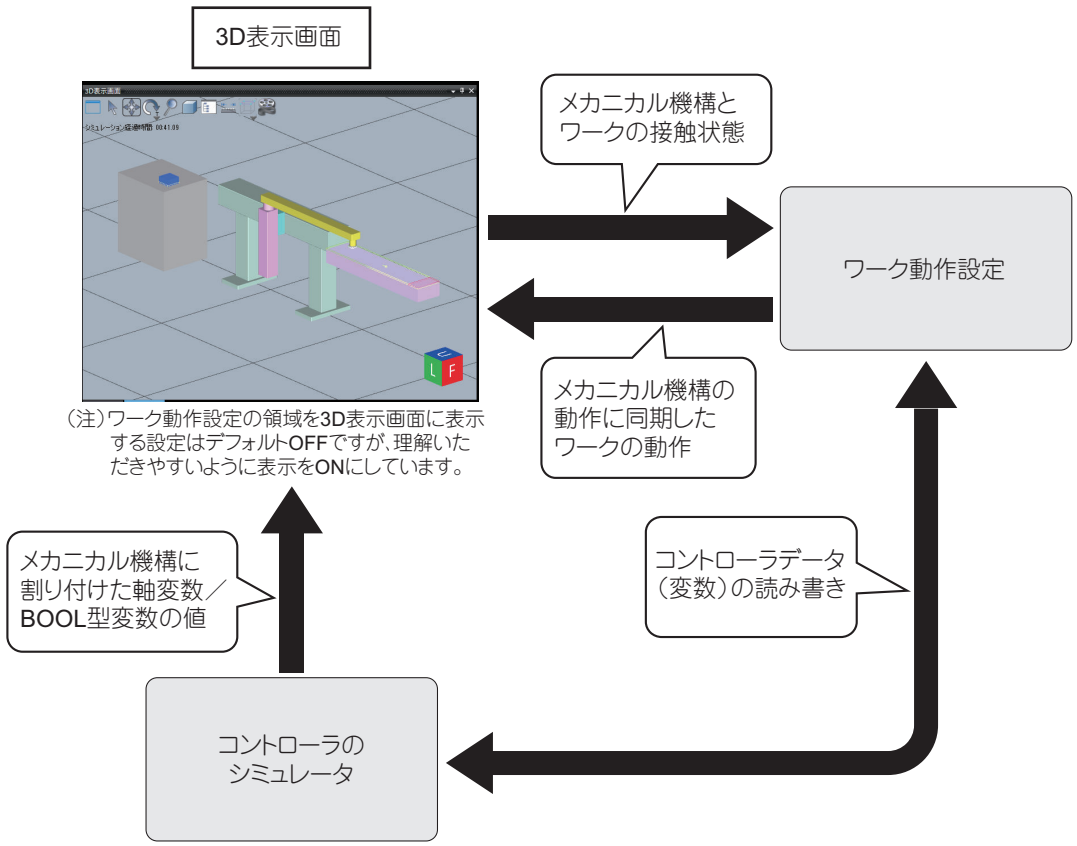
7-3　ワーク動作設定 7-4

7-3-1　ワーク動作設定の一覧 7-6
7-3-2　生成 7-7
7-3-3　把持 7-12
7-3-4　搬送 7-14
7-3-5　消去 7-18
7-3-6　押出 7-20
7-3-7　センサ 7-22
7-3-8　物理シミュレーション 7-25
7-3-9　ワーク動作設定共通の設定 7-28

7-1　ワークシミュレーション簡単設定の概要

ワークの動作を 3D シミュレーション上で実現するための設定について説明します。

ワークシミュレーション簡単設定とは、3D 表示画面に表示される仮想装置モデルのメカニカル機構の動作時に、メカニカル機構と連動するワークの動作を設定し、シミュレーション実行するものです。「第 6 章 3D 形状データを動作させる設定およびスクリプトの作成 (P.6-1)」に記載のシェイプスクリプトによるワークのシミュレーションを、プログラムレスで実現します。
メカニカル機構と連動するワークの動作設定はワークに接する各メカニカル機構に設定します。これをワーク動作設定と呼びます。
コントローラのシミュレーション実行時にはコントローラのプログラムを実行し、メカニカル機構に割り付けた軸変数や BOOL 型変数の値が変化すると、変数の値の変化に応じて 3D 表示画面上のメカニカル機構や 3D 形状データが動作します。
ワークシミュレーション簡単設定によるシミュレーション実行時にはワークと接触したメカニカル機構は、メカニカル機構が持つワーク動作設定に沿った処理を実行することで、ワークを動作させます。これにより、3D 表示画面でメカニカル機構の動作に合わせてワークが移動する表示を実現しています。



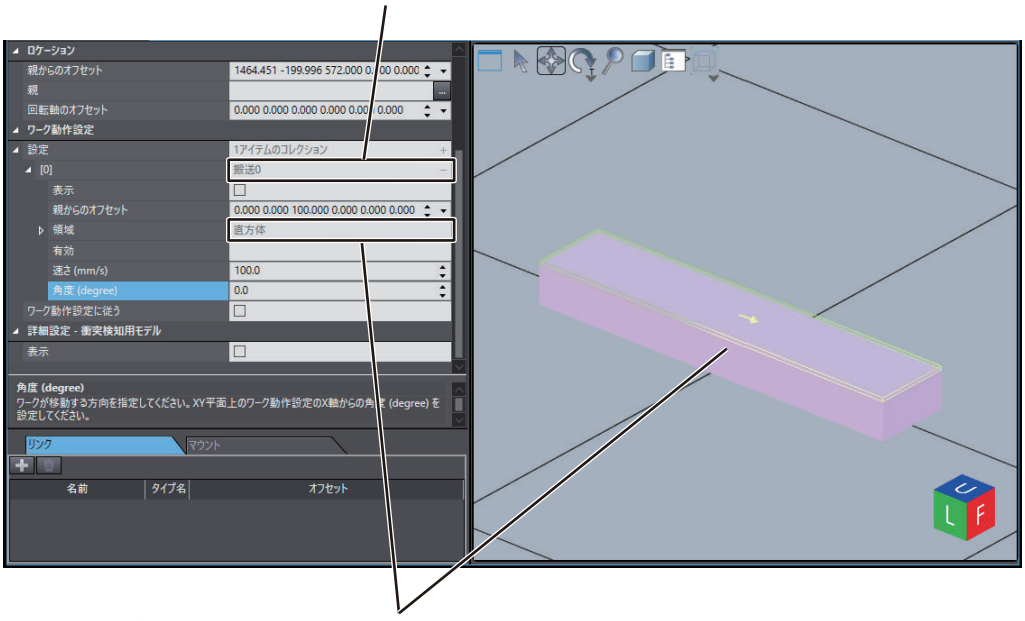
7-2　メカニカル機構の設定

メカニカル機構の動作に関する設定を行います。
メカニカル機構の設定画面を表示し、［パラメーター設定］画面で対象のコントローラおよび変数が割り付けられていることを確認します。
項目の詳細は「4-3-5 メカニカル機構の設定項目 (P.4-25)」を参照してください。

7-3　ワーク動作設定

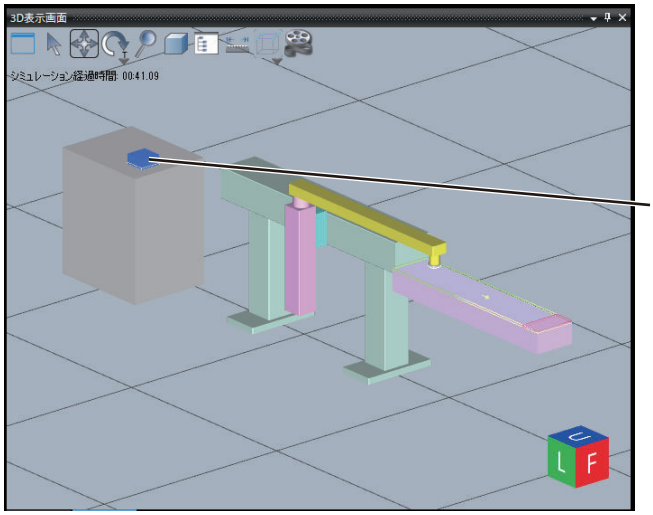
ワーク動作設定とは、メカニカル機構などの 3D 形状データがワークに接したときにワークをどのように動作させるかを設定することです。
以下は、ワークを搬送する動作の 3D 形状データの例です。
3D 形状データの設定に、ワーク動作設定という項目があり、この項目で 3D 形状データとワークが接触したときにワークを動作させる領域と、ワークが接触したときにワークをどのように動作させるかを設定します。ワーク動作設定の詳細については「7-3-1 ワーク動作設定の一覧 (P.7-6)」を参照してください。

ワークをどのように動作させるかの設定（例: 搬送）

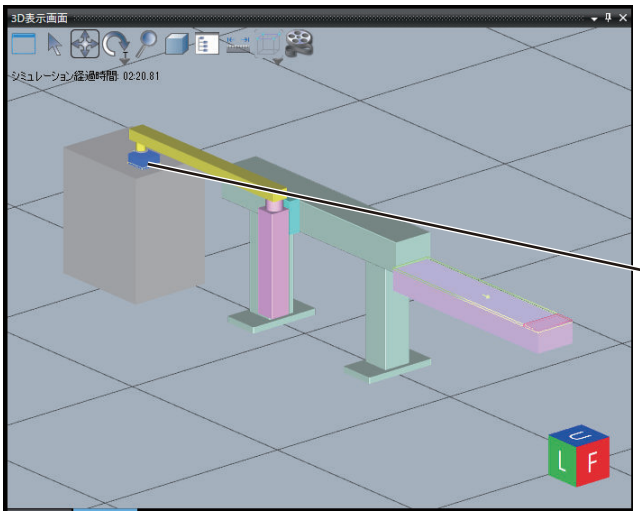


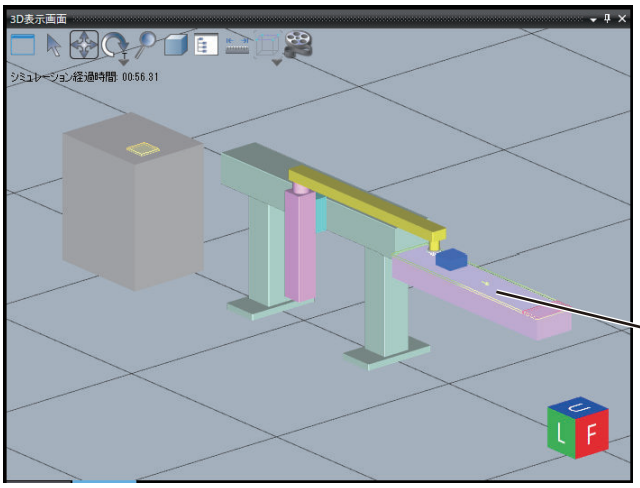
3D形状データとワークが接触したときにワークを動作させる領域

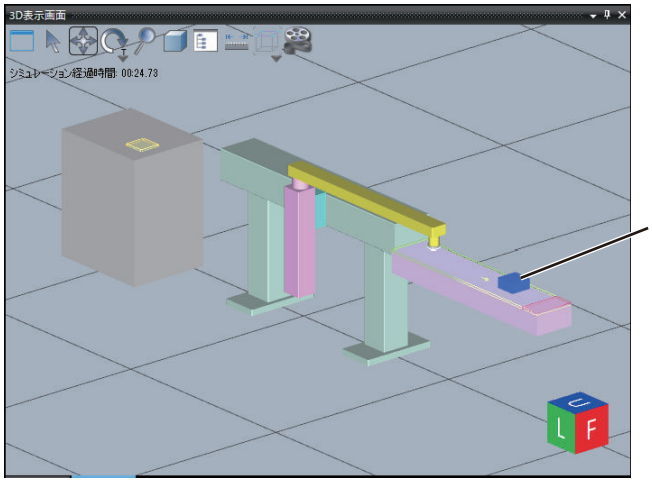
ワークシミュレーション簡単設定によるワークのシミュレーションでは、仮想装置モデルでのワークの移動をシミュレーションで実施することができます。
以下の例ではローダを模した直方体上部にワークが存在し、XYθ スライダ先端でワークを模した直方体を把持し、XYθ スライダが動作してワークをコンベアに模した直方体上でワークを離し、コンベアがワークを移動させるという動作を実現します。



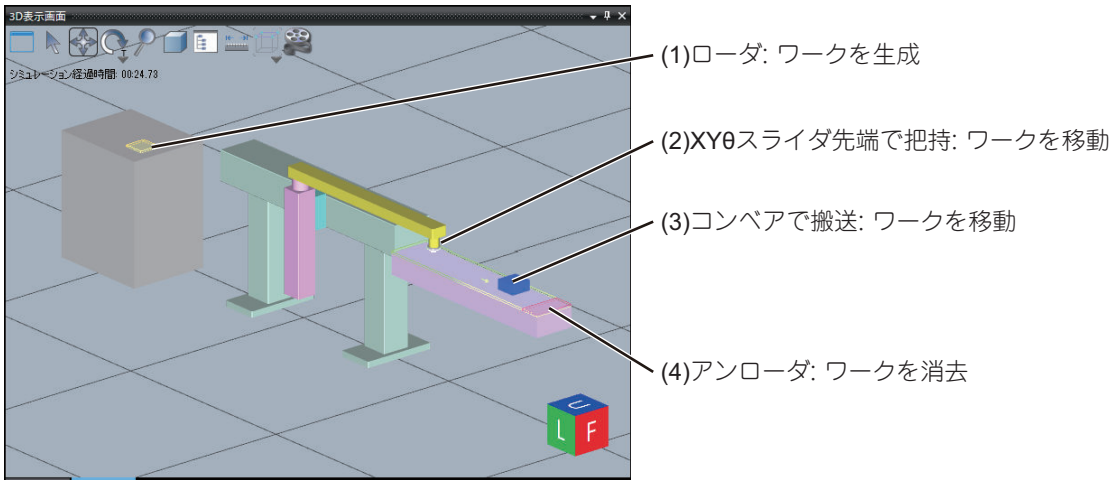
（注）ワーク動作設定の領域を3D表示画面に表示する設定はデフォルトOFFですが、理解いただきやすいように表示をONにしています。





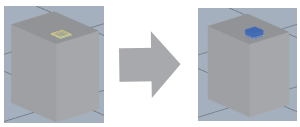
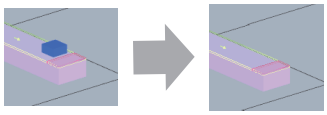


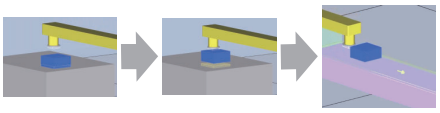
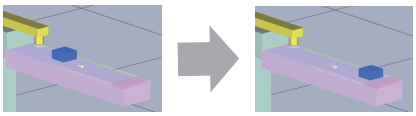
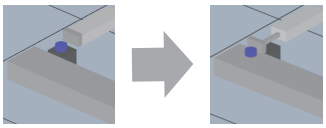
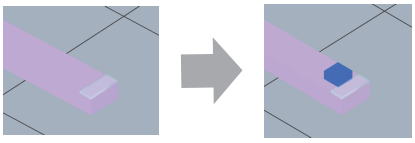
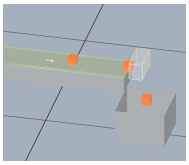
このような仮想装置モデルの動作について、ワークに接する 3D 表示データがワークにどのような動作を与えるかを検討します。
この例は、(1)ローダがワークを生成する動作、(2)XY θ スライダ先頭でワークを把持、(3)コンベアでワーク搬送、(4)アンローダがワークを消去する動作となります。



7-3-1 ワーク動作設定の一覧

ワーク動作設定を以下に一覧で示します。

カテゴリ	ワーク動作設定	動作イメージ	説明
ワーク生成／消去	生成		指定された 3D 形状データをワーク動作設定で取り扱われるワークとして表示します。 パラメータの設定により、ワークの生成間隔を変更したり生成位置をランダムに設定したりすることができます。 コントローラの BOOL 変数を指定することで、ワーク生成のタイミングをコントロールすることができます。
	消去		指定された領域にワークが接したときに、ワークを消去します。

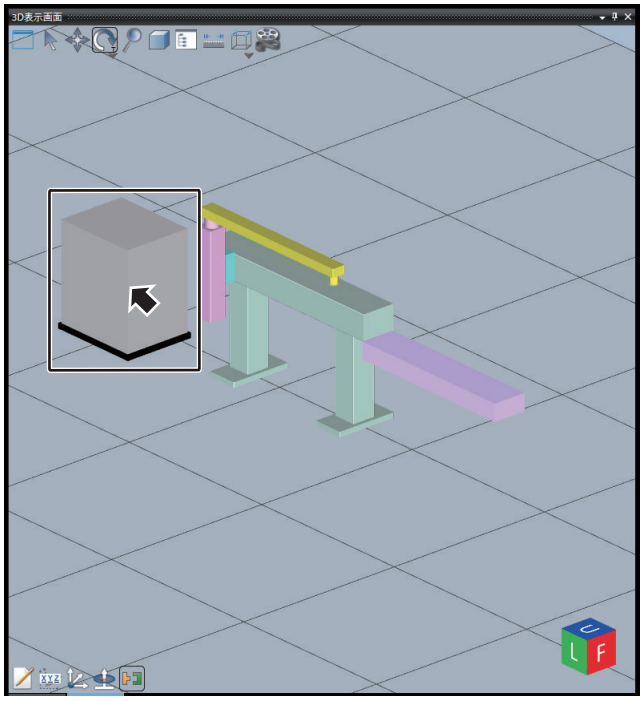
カテゴリ	ワーク動作設定	動作イメージ	説明
ワークの移動	把持		メカニカル機構の把持を模して、3D 形状データに接したワークを接合します。 以下のいずれかの条件で 3D 形状データからワークを分離します。 • ワークが他の 3D 形状データのワーク動作設定の領域に接した場合（例: 把持の設定を持つ 3D 形状データのワーク動作設定の領域に接した場合） • 指定されたコントローラの BOOL 変数が FALSE になった場合
	搬送		ワークを指定した方向と速さで移動させます。 コントローラの BOOL 変数を指定することでワークの移動や停止を変数で制御することができます。
	押出		ワークを押して移動させます。
ワークの検知	センサ		ワークに接したときに指定されたコントローラ変数（BOOL 型）を TRUE にします。 ワークに対して動きは与えません。
物理シミュレーション	物理シミュレーション		指定された領域にワークが接したときに、ワークが簡易的な物理シミュレーションに基づいた動作になります。落下やすべりを再現できます。

7-3-2　生成

アプリケーションマネージャに登録されているメカニカル機構などの 3D 形状データに、ワーク動作設定として「生成」を設定する手順は以下のとおりです。

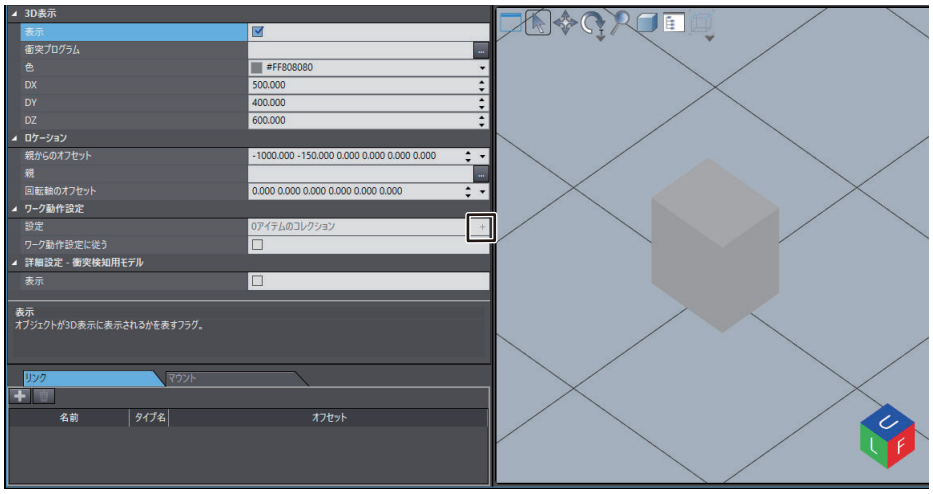
操作手順

1 **3D 表示画面で、「生成」を設定したい 3D 形状データをダブルクリックします。**



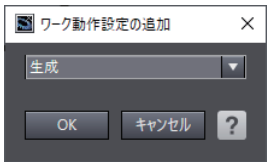
エディットウィンドウ領域に 3D 形状データの設定画面が表示されます。

2 **3D 形状データの設定画面で［ワーク動作設定］以下にある［設定］の右側のボタンをクリックします。メカニカル機構の場合、［ワーク動作設定］はメカニカル機構共通設定内にあります。**

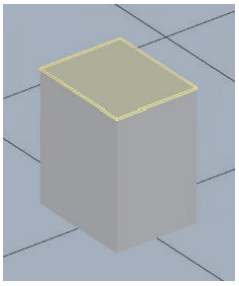


［ワーク動作設定の追加］ダイアログボックスが表示されます。

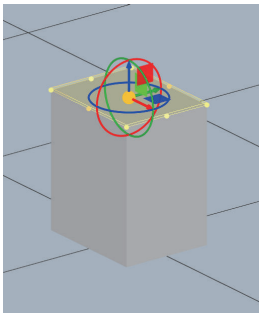
3 **［生成］を選択し、［OK］ボタンをクリックします。**



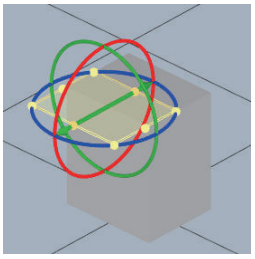
4 **3D 編集エリアでワークを生成する領域を設定します。**



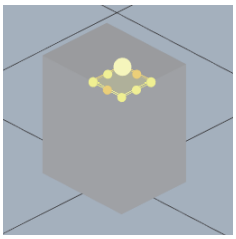
3D 形状データ上に表示される領域の中心点をクリックすると［移動］と［回転］アイコンが表示されます。



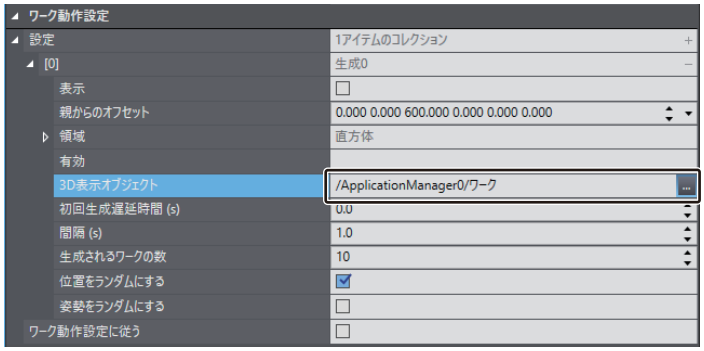
各アイコンをドラッグして領域の位置と姿勢を調整します。



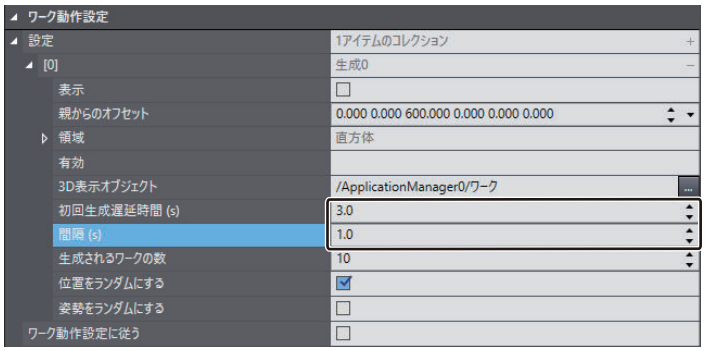
3D 形状データ上に設定した領域に表示される点をドラッグし、領域のサイズを調整します。



5 **ワークとして表示する 3D 形状データを設定します。［ワーク動作設定］の［3D 表示オブジェクト］に 3D 表示オブジェクトに登録されている 3D 形状データの名前を選択します。**

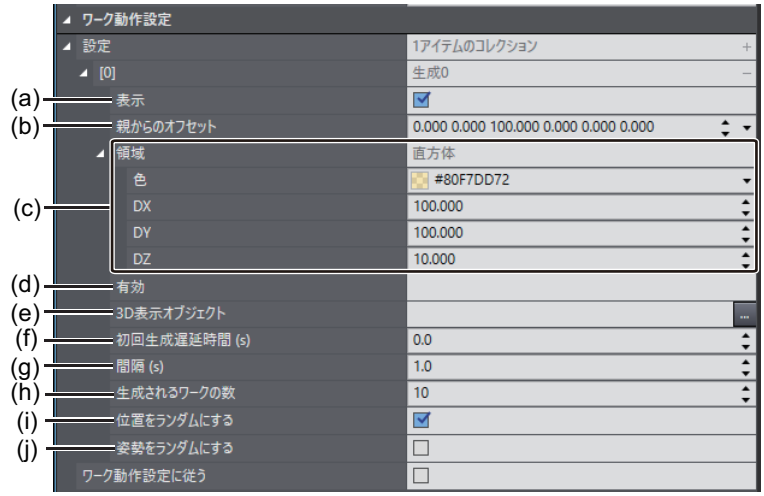


6 **ワークの生成条件を設定します。ここでは、ワークシミュレーションが開始してからワークを生成するまでの初回の時間とワークを生成する時間間隔を設定します。**



「生成」ワーク動作設定で設定可能な項目一覧

「生成」ワーク動作設定で設定可能な項目は以下のとおりです。



	項目	説明	設定値	初期値
(a)	表示	ワーク動作設定の領域を 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック OFF

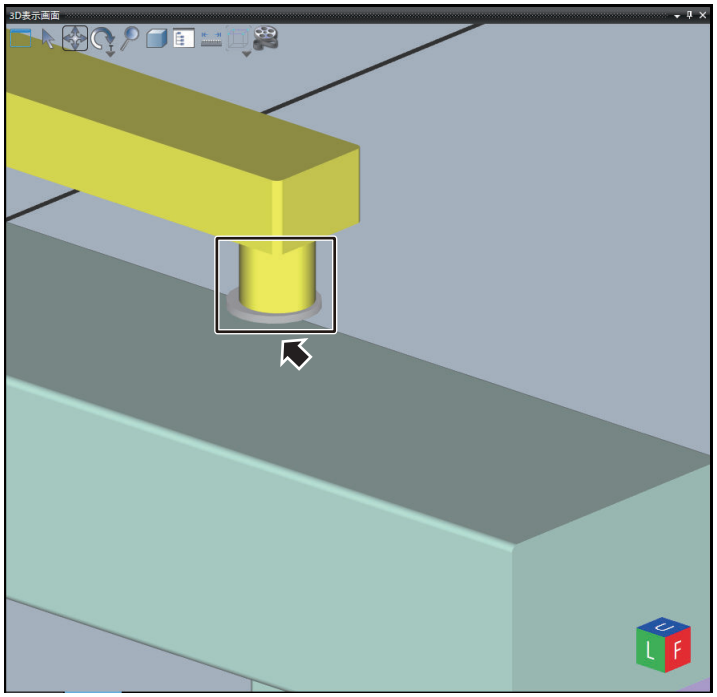
	項目		説明	設定値	初期値
(b)	親からのオフセット		領域の位置と姿勢を設定します。 設定する座標は 3D 形状データのローカル座標系における座標です。 ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～ +1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	領域設定時の操作による
(c)	領域	色	領域の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#80F7DD72
		DX	領域の X 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DY	領域の Y 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DZ	領域の Z 軸方向の長さを設定します。（単位: mm）	0.1～10,000	10
(d)	有効		「生成」ワーク動作設定を有効／無効にします。 コントローラの BOOL 型変数名を指定することで、コントローラ変数によるワークの生成タイミングを制御することができます。 空白: 「生成」ワーク動作設定が常に有効です。 True: 「生成」ワーク動作設定が常に有効です。 False:「生成」ワーク動作設定が常に無効です。 コントローラ名.BOOL 型のコントローラ変数名: コントローラ変数が True のときに「生成」ワーク動作設定が有効です。 例: 以下のように設定すると、コントローラ名「new_Controller0」の変数「PartGenerate」が True になるたびにワークが生成されます。 本設定項目: new_Controller0.PartGenerate 初回生成遅延時間(s): 0 ワークが生成される時間間隔(s): 0 生成されるワークの数: 1	True False コントローラ名.BOOL 型のコントローラ変数名	空白
(e)	3D 表示オブジェクト		ワークとして生成する 3D 形状データを指定します。	3D 形状データの名称	空白
(f)	初回生成遅延時間(s)		ワークシミュレーションが開始し、かつ「生成」ワーク動作設定が有効になってから初回のワーク生成までの時間を設定します。（単位: 秒）	0.0～1000	0
(g)	間隔(s)		ワークが生成される時間間隔を設定します。（単位: 秒）	0.1～100	1
(h)	生成されるワークの数		ワークが生成される数を設定します。	1～1000	10
(i)	位置をランダムにする		生成されるワークの位置を、領域の範囲でランダムにします。	チェック ON／チェック OFF	チェック ON
(j)	姿勢をランダムにする		生成されるワークの姿勢をランダムにします。	チェック ON／チェック OFF	チェック OFF

7-3-3　把持

アプリケーションマネージャに登録されているメカニカル機構などの 3D 形状データに、ワーク動作設定として「把持」を設定する手順は以下のとおりです。

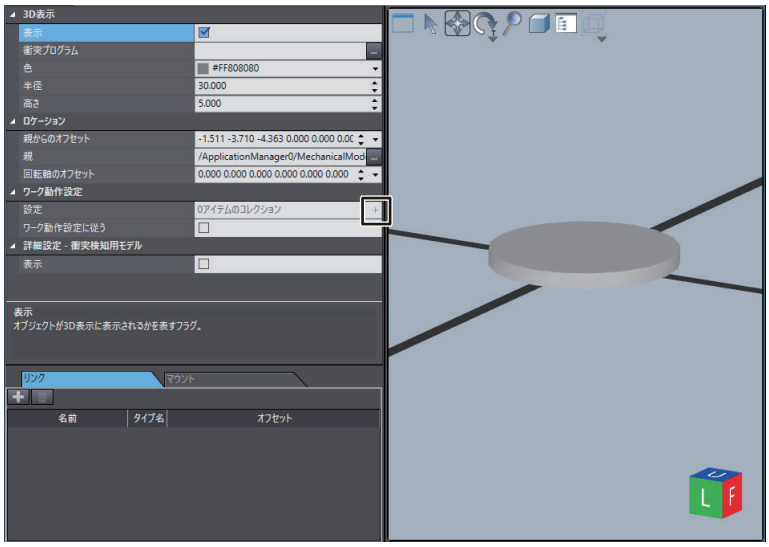
操作手順

1　**3D 表示画面で、「把持」を設定したい 3D 形状データをダブルクリックします。**



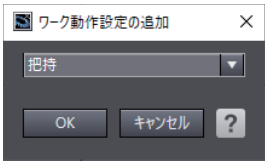
エディットウィンドウ領域に 3D 形状データの設定画面が表示されます。

2　**3D 形状データの設定画面で［ワーク動作設定］以下にある［設定］の右側のボタンをクリックします。メカニカル機構の場合、［ワーク動作設定］はメカニカル機構共通設定内にあります。**

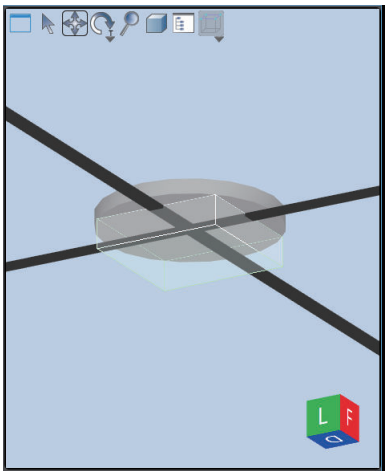


［ワーク動作設定の追加］ダイアログボックスが表示されます。

3 **［把持］を選択し、［OK］ボタンをクリックします。**

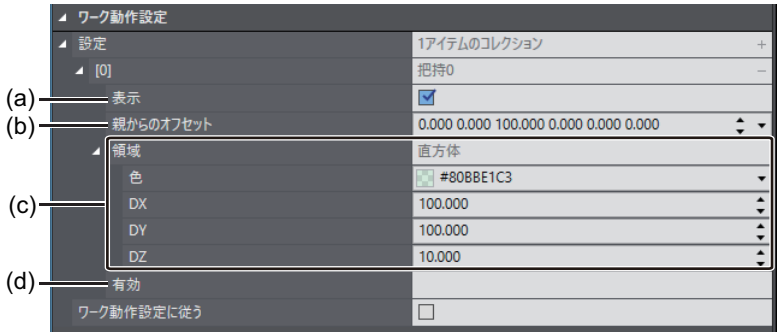


4 **3D 形状データの編集ウィンドウの右に表示されている 3D 形状データ上でワークを把持させる領域を設定します。**



「把持」ワーク動作設定で設定可能な項目一覧

「把持」ワーク動作設定で設定可能な項目は以下のとおりです。



	項目	説明	設定値	初期値
(a)	表示	ワーク動作設定の領域を 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック OFF
(b)	親からのオフセット	領域の位置と姿勢を設定します。 設定する座標は 3D 形状データのローカル座標系における座標です。 ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	領域設定時の操作による

	項目		説明	設定値	初期値
(c)	領域	色	領域の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#80BBE1C3
		DX	領域の X 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DY	領域の Y 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DZ	領域の Z 軸方向の長さを設定します。（単位: mm）	0.1～10,000	10
(d)	有効		オプションで「把持」ワーク動作設定を有効／無効にします。 コントローラの BOOL 型変数名を指定することで、コントローラ変数によるワークのピック生成タイミングを制御することができます。 空白:「把持」ワーク動作設定が常に有効です。 True:「把持」ワーク動作設定が常に有効です。 False:「把持」ワーク動作設定が常に無効です。 コントローラ名.BOOL 型のコントローラ変数名: コントローラ変数が True のときに「把持」ワーク動作設定が有効です。 例: 3D 形状データがワークに接しているときに、コントローラ変数が True のときは 3D 形状データとワークは接合します。このあとコントローラ変数が False となると、3D 形状データとワークは分離します。	True False コントローラ名.BOOL 型のコントローラ変数名	空白

参考

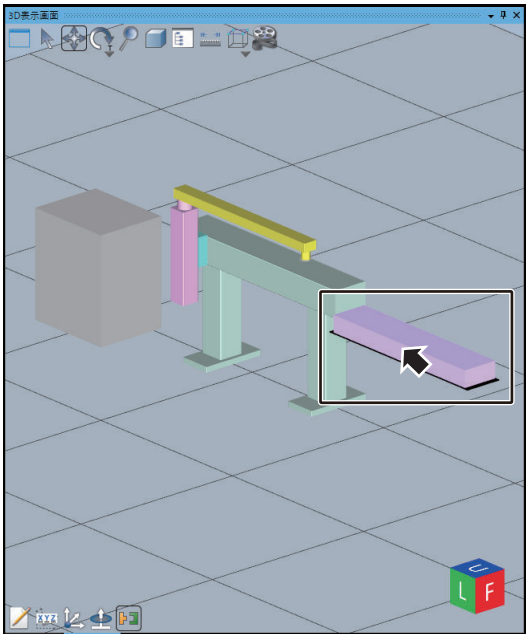
「把持」ワーク動作設定により 3D 形状データがワークを接合している場合、ワークが「搬送」ワーク動作設定を持つ 3D 形状データに接してもワークは「搬送」ワーク動作設定を持つ 3D 形状データに接合しません。このような場合は、実際の制御と同じようにワークを把持／分離を制御するコントローラ変数を準備し、「把持」ワーク動作設定の項目の「有効」にそのコントローラ変数を割り当てることで、ワークの把持／分離に相当するコントローラ変数が TRUE となると「把持」ワーク動作設定が有効となり、ワークを把持できる状態になります。また、コントローラ変数が FALSE となると「搬送」ワーク動作設定を持つ 3D 形状データに制御が移行します。

7-3-4 搬送

アプリケーションマネージャに登録されているメカニカル機構などの 3D 形状データに、ワーク動作設定として「搬送」を設定する手順は以下のとおりです。

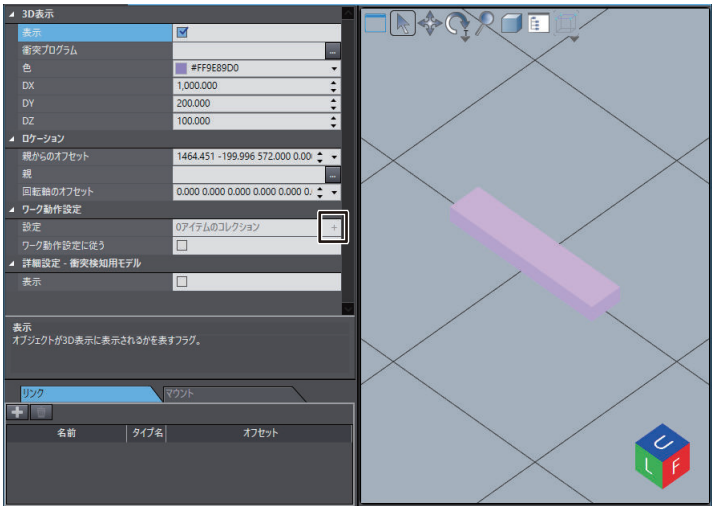
操作手順

1 **3D 表示画面で、「搬送」を設定したい 3D 形状データをダブルクリックします。**



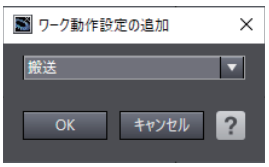
エディットウィンドウ領域に 3D 形状データの設定画面が表示されます。

2 **3D 形状データの設定画面で［ワーク動作設定］以下にある［設定］の右側のボタンをクリックします。メカニカル機構の場合、［ワーク動作設定］はメカニカル機構共通設定内にあります。**

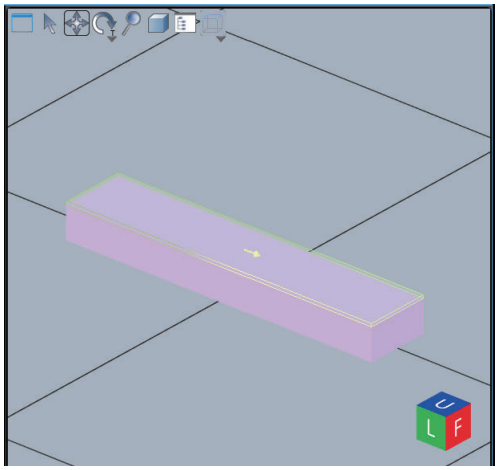


［ワーク動作設定の追加］ダイアログボックスが表示されます。

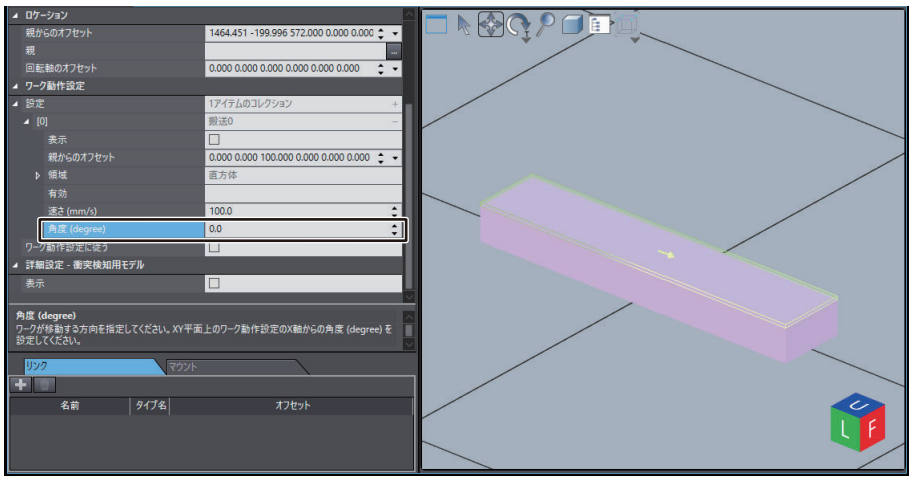
3 **［搬送］を選択し、［OK］ボタンをクリックします。**



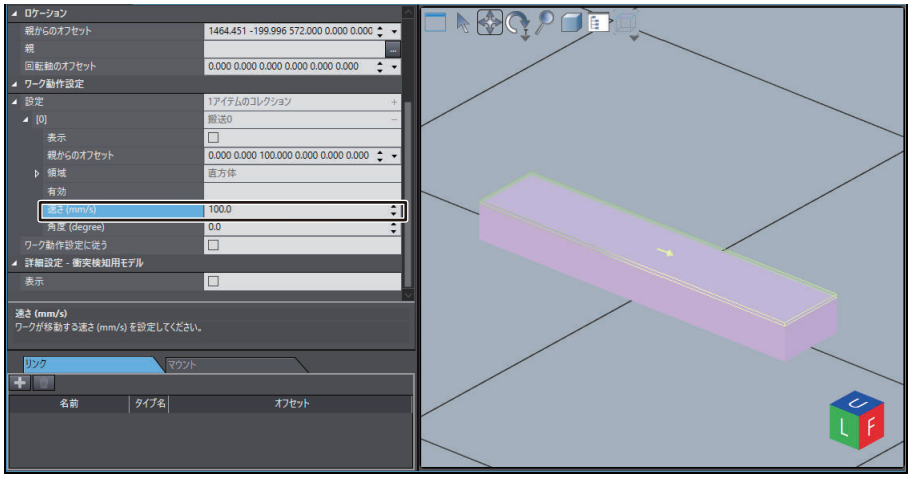
4 **3D 形状データの編集ウィンドウの右に表示されている 3D 形状データ上でワークを移動する領域を設定します。**



5 **ワークを移動させる方向を設定します。**

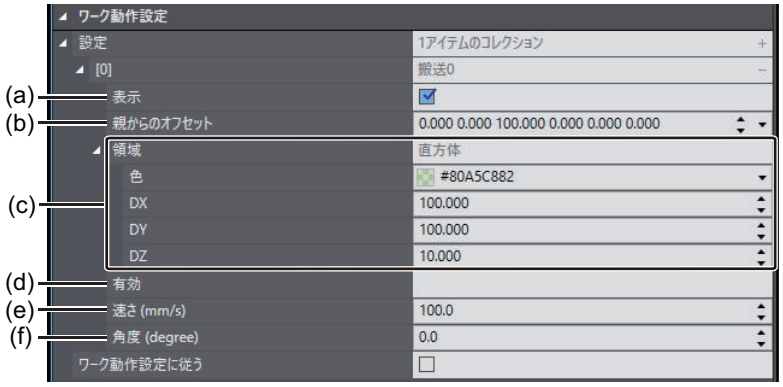


6 **ワークを移動させる速さを設定します。**



「搬送」ワーク動作設定で設定可能な項目一覧

「搬送」ワーク動作設定で設定可能な項目は以下のとおりです。



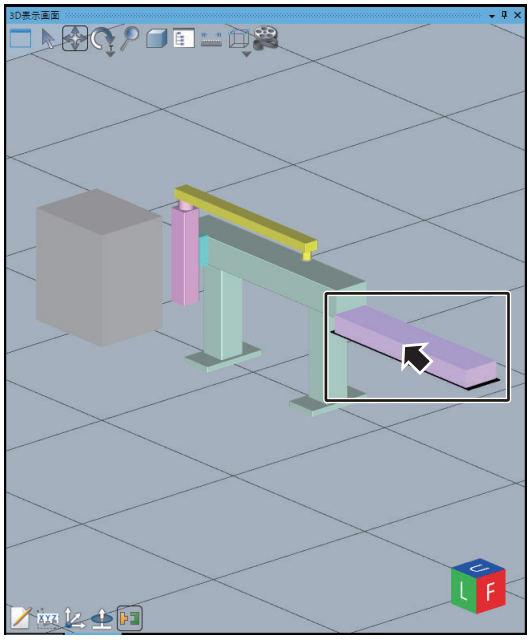
	項目		説明	設定値	初期値
(a)	表示		ワーク動作設定の領域を 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック OFF
(b)	親からのオフセット		領域の位置と姿勢を設定します。 設定する座標は 3D 形状データのローカル座標系における座標です。 ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	領域設定時の操作による
(c)	領域	色	領域の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#80A5C882
		DX	領域の X 軸方向の長さを設定します。 （単位: mm）	0.1～10,000	領域設定時の操作による
		DY	領域の Y 軸方向の長さを設定します。 （単位: mm）	0.1～10,000	領域設定時の操作による
		DZ	領域の Z 軸方向の長さを設定します。 （単位: mm）	0.1～10,000	10
(d)	有効		オプションで「搬送」ワーク動作設定を有効／無効にします。 コントローラの BOOL 型変数名を指定することで、コントローラ変数によるワークの移動タイミングを制御することができます。 空白: 「搬送」ワーク動作設定が常に有効です。 True: 「搬送」ワーク動作設定が常に有効です。 False:「搬送」ワーク動作設定が常に無効です。 コントローラ名.BOOL 型のコントローラ変数名: コントローラ変数が True のときにワーク動作設定が有効です。 例: 3D 形状データがワークに接しているときに、コントローラ変数が True のときはワークは移動します。このあとコントローラ変数が False となると、ワークは停止します。	True False コントローラ名.BOOL 型のコントローラ変数名	空白
(e)	速さ(mm/s)		ワークを移動させる速さを設定します。 （単位: mm／秒）	1～10,000	100
(f)	角度(degree)		領域の X 軸方向を 0 度として、ワークを移動させる方向を設定します。	-180～+180	0

7-3-5　消去

アプリケーションマネージャに登録されているメカニカル機構などの 3D 形状データに、ワーク動作設定として「消去」を設定する手順は以下のとおりです。

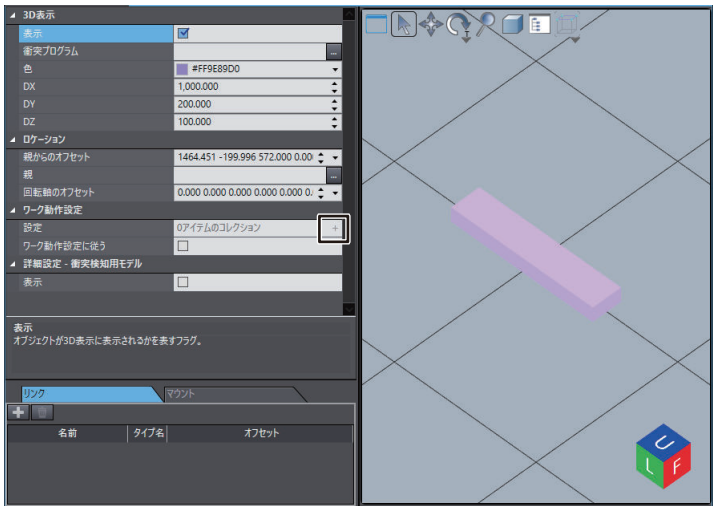
操作手順

1　**3D 表示画面で、「消去」を設定したい 3D 形状データをダブルクリックします。**



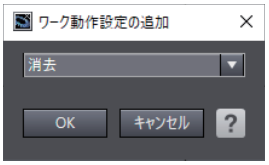
エディットウィンドウ領域に 3D 形状データの設定画面が表示されます。

2　**3D 形状データの設定画面で［ワーク動作設定］以下の［設定］の右側のボタンをクリックします。メカニカル機構の場合、［ワーク動作設定］はメカニカル機構共通設定内にあります。**

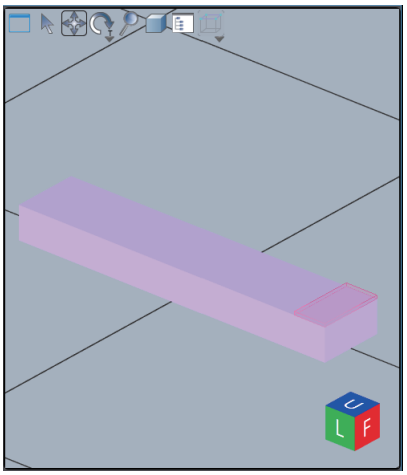


［ワーク動作設定の追加］ダイアログボックスが表示されます。

3　**［消去］を選択し、［OK］ボタンをクリックします。**

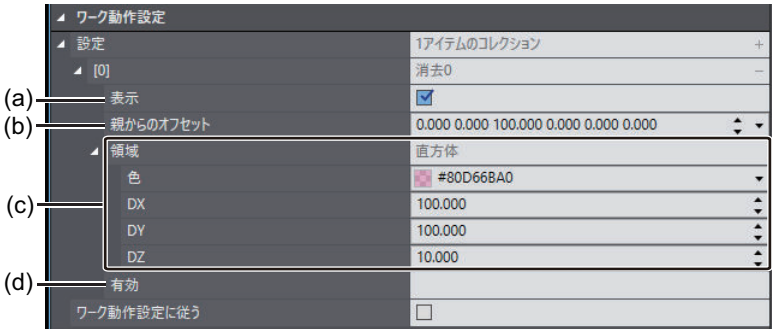


4 **3D 形状データの編集ウィンドウの右に表示されている 3D 形状データ上でワークを消去させる領域を設定します。**



「消去」ワーク動作設定で設定可能な項目一覧

「消去」ワーク動作設定で設定可能な項目は以下のとおりです。



	項目	説明	設定値	初期値
(a)	表示	ワーク動作設定の領域を 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック OFF
(b)	親からのオフセット	領域の位置と姿勢を設定します。 設定する座標は 3D 形状データのローカル座標系における座標です。 ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	領域設定時の操作による

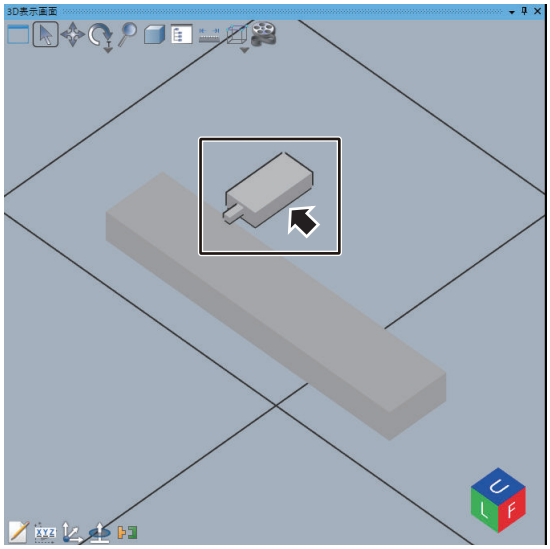
	項目		説明	設定値	初期値
(c)	領域	色	領域の色を設定します。	透過性（0～#FF）およびRGB（0～#FF、0～#FF、0～#FF）	#80D66BA0
		DX	領域の X 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DY	領域の Y 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DZ	領域の Z 軸方向の長さを設定します。（単位: mm）	0.1～10,000	10
(d)	有効		オプションで「消去」ワーク動作設定を有効／無効にします。 コントローラの BOOL 型変数名を指定することで、コントローラ変数によるワークの消去のタイミングを制御することができます。 空白:「消去」ワーク動作設定が常に有効です。 True:「消去」ワーク動作設定が常に有効です。 False:「消去」ワーク動作設定が常に無効です。 コントローラ名.BOOL 型のコントローラ変数名: コントローラ変数が True のときに「消去」ワーク動作設定が有効となります。	True False コントローラ名.BOOL 型のコントローラ変数名	空白

7-3-6 押出

アプリケーションマネージャに登録されているメカニカル機構などの 3D 形状データに、ワーク動作設定として「押出」を設定する手順は以下のとおりです。

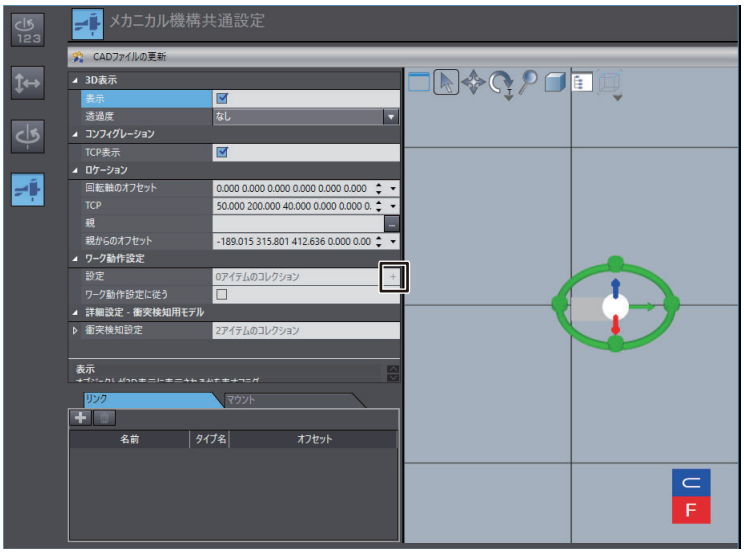
操作手順

1 **3D 表示画面で、「押出」を設定したい 3D 形状データをダブルクリックします。**



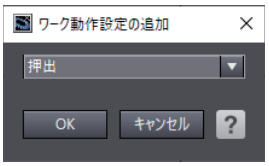
エディットウィンドウ領域に 3D 形状データの設定画面が表示されます。

2 **3D 形状データの設定画面で［ワーク動作設定］以下にある［設定］の右側のボタンをクリックします。メカニカル機構の場合、［ワーク動作設定］はメカニカル機構共通設定内にあります。**

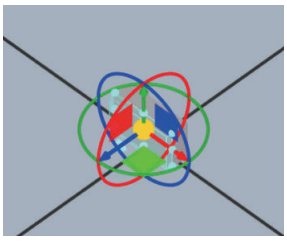


［ワーク動作設定の追加］ダイアログボックスが表示されます。

3 **［押出］を選択し、［OK］ボタンをクリックします。**

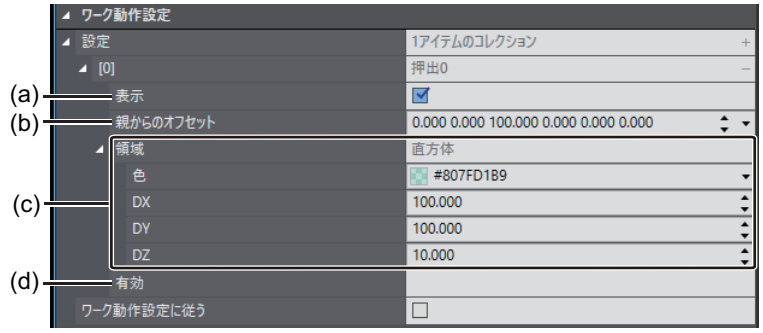


4 **3D 形状データの編集ウィンドウの右に表示されている 3D 形状データの表面上でワークを押し出す領域を設定します。**



「押出」ワーク動作設定で設定可能な項目一覧

「押出」ワーク動作設定で設定可能な項目は以下のとおりです。



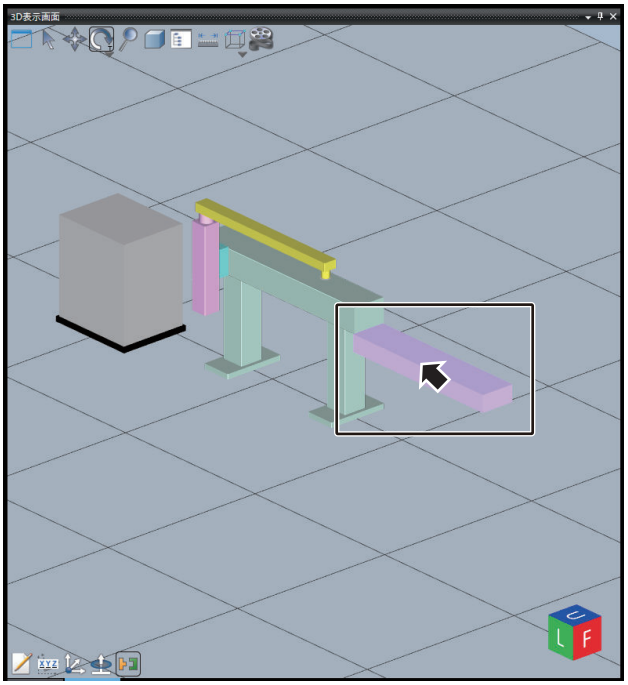
	項目		説明	設定値	初期値
(a)	表示		ワーク動作設定の領域を 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック OFF
(b)	親からのオフセット		領域の位置と姿勢を設定します。 設定する座標は 3D 形状データのローカル座標系における座標です。 ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	領域設定時の操作による
(c)	領域	色	領域の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#807FD1B9
		DX	領域の X 軸方向の長さを設定します。 （単位: mm）	0.1～10,000	領域設定時の操作による
		DY	領域の Y 軸方向の長さを設定します。 （単位: mm）	0.1～10,000	領域設定時の操作による
		DZ	領域の Z 軸方向の長さを設定します。 （単位: mm）	0.1～10,000	10
(d)	有効		オプションで「押出」ワーク動作設定を有効／無効にします。 コントローラの BOOL 型変数名を指定することで、コントローラ変数によるワークの押し出しのタイミングを制御することができます。 空白:「押出」ワーク動作設定が常に有効です。 True:「押出」ワーク動作設定が常に有効です。 False:「押出」ワーク動作設定が常に無効です。 コントローラ名.BOOL 型のコントローラ変数名: コントローラ変数が True のときに「押出」ワーク動作設定が有効となります。	True False コントローラ名.BOOL 型のコントローラ変数名	空白

7-3-7 センサ

アプリケーションマネージャに登録されているメカニカル機構などの 3D 形状データに、ワーク動作設定として「センサ」を設定する手順は以下のとおりです。

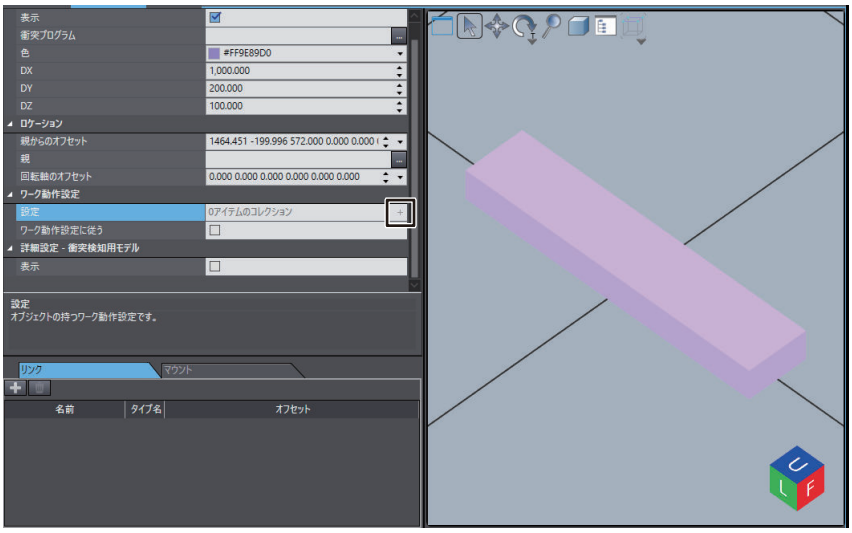
操作手順

1 **3D 表示画面で、「センサ」を設定したい 3D 形状データをダブルクリックします。**



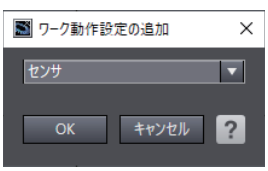
エディットウィンドウ領域に 3D 形状データの設定画面が表示されます。

2 **3D 形状データの設定画面で［ワーク動作設定］以下にある［設定］の右側のボタンをクリックします。メカニカル機構の場合は、［ワーク動作設定］はメカニカル機構共通設定内にあります。**

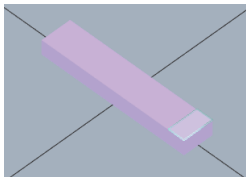


［ワーク動作設定の追加］ダイアログボックスが表示されます。

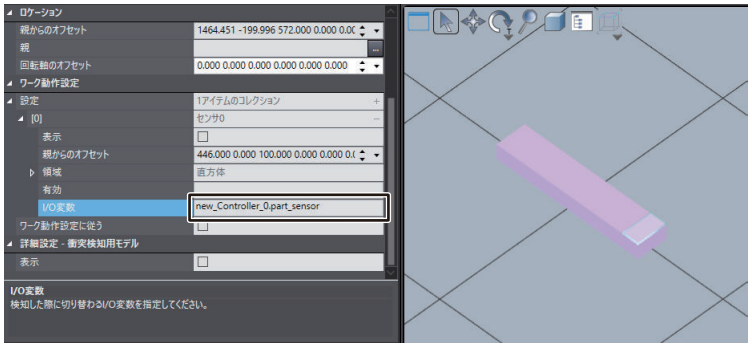
3 **［センサ］を選択し、［OK］ボタンをクリックします。**



4 **3D 形状データの編集ウィンドウの右に表示されている 3D 形状データの表面上でワークを検知する領域を設定します。**

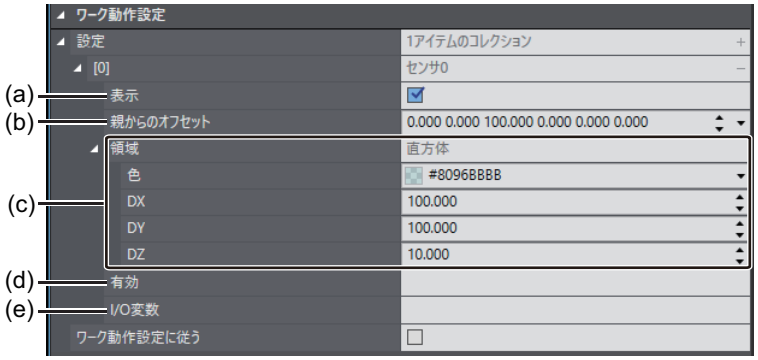


5 **ワークを検知したときに現在値を TRUE にするコントローラの変数を設定します。**



「センサ」ワーク動作設定で設定可能な項目一覧

「センサ」ワーク動作設定で設定可能な項目は以下のとおりです。



	項目		説明	設定値	初期値
(a)	表示		ワーク動作設定の領域を 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック OFF
(b)	親からのオフセット		領域の位置と姿勢を設定します。 設定する座標は 3D 形状データのローカル座標系における座標です。 ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	領域設定時の操作による
(c)	領域	色	領域の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#8096BBBB
		DX	領域の X 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DY	領域の Y 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DZ	領域の Z 軸方向の長さを設定します。（単位: mm）	0.1～10,000	10

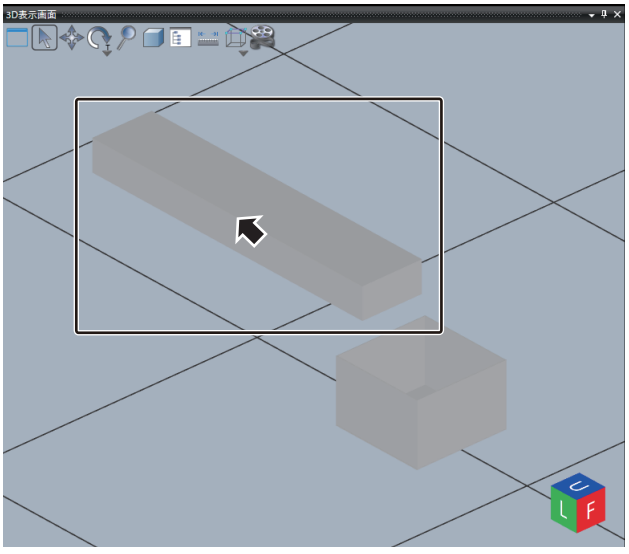
	項目	説明	設定値	初期値
(d)	有効	オプションで「センサ」ワーク動作設定を有効／無効にします。 コントローラの BOOL 型変数名を指定することで、「センサ」ワーク動作設定の有効化／無効化が可能です。 空白:「センサ」ワーク動作設定が常に有効です。 True:「センサ」ワーク動作設定が常に有効です。 False:「センサ」ワーク動作設定が常に無効です。 コントローラ名.BOOL 型のコントローラ変数名: コントローラ変数が True のときに「センサ」ワーク動作設定が有効になります。	True False コントローラ名.BOOL 型のコントローラ変数名	空白
(e)	I/O 変数	ワークが領域に入ったときに現在値を TRUE にする BOOL 型のコントローラ変数名を設定します。	コントローラ名.BOOL 型のコントローラ変数名	空白

7-3-8 物理シミュレーション

指定された領域にワークが接したときに、ワークが簡易的な物理シミュレーションに基づいた動作になります。落下やすべりを再現できます。
アプリケーションマネージャに登録されているメカニカル機構などの 3D 形状データに、ワーク動作設定として「物理シミュレーション」を設定する手順は以下のとおりです。

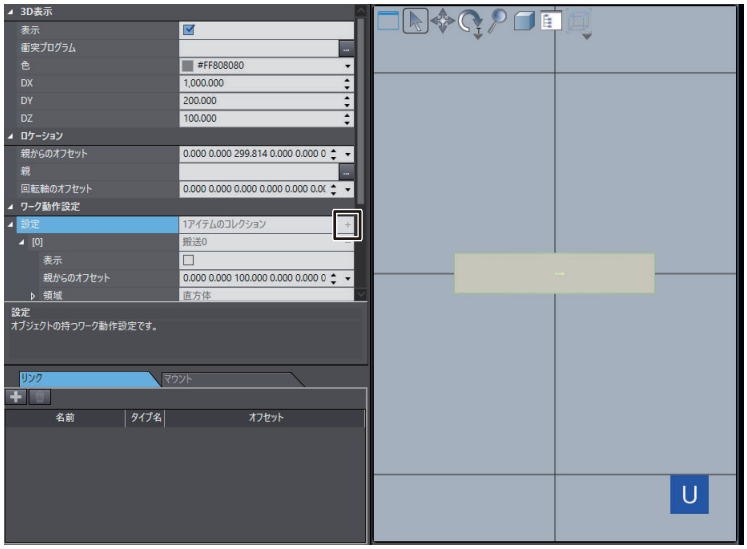
操作手順

1 **3D 表示画面で、「物理シミュレーション」を設定したい 3D 形状データをダブルクリックします。**



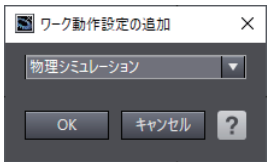
エディットウィンドウ領域に 3D 形状データの設定画面が表示されます。

2 **3D 形状データの設定画面で［ワーク動作設定］以下にある［設定］の右側のボタンをクリックします。メカニカル機構の場合、［ワーク動作設定］はメカニカル機構共通設定内にあります。**

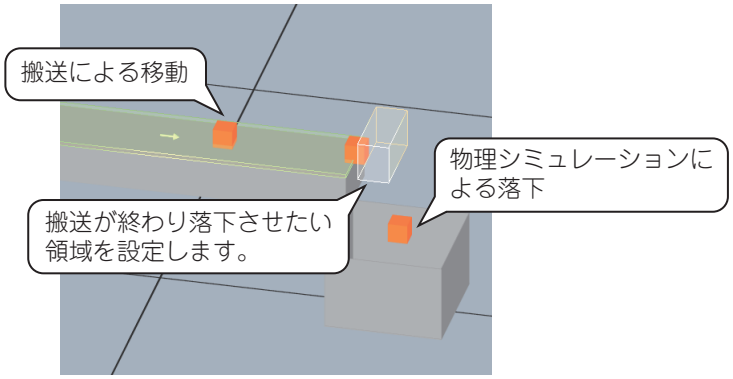


［ワーク動作設定の追加］ダイアログボックスが表示されます。

3 **［物理シミュレーション］を選択し、［OK］ボタンをクリックします。**



4 **3D 形状データの編集ウィンドウの右に表示されている 3D 形状データ上でワークを検知する領域を設定します。**

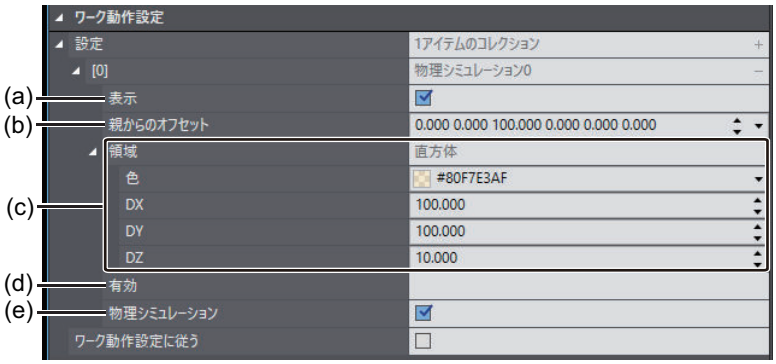


5 **ワークが物理シミュレーション動作を行ったあとに接する 3D 形状データに、ワークの動作を継続して処理をさせるために衝突検知フィルタの設定を行います。この設定を行わないと、物理シミュレーションによって落下したワークは 3D 形状データに衝突しても通過する動作になります。**



「物理シミュレーション」ワーク動作設定で設定可能な項目一覧

「物理シミュレーション」ワーク動作設定で設定可能な項目は以下のとおりです。

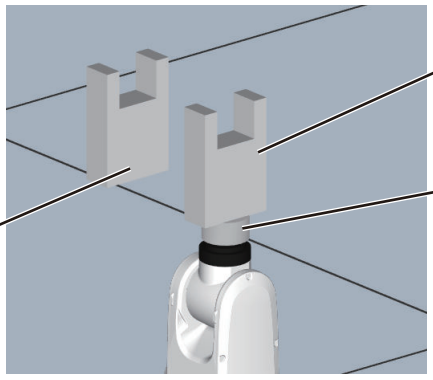


	項目		説明	設定値	初期値
(a)	表示		ワーク動作設定の領域を 3D 表示画面に表示する／しないを切り替えます。	チェック ON／チェック OFF	チェック OFF
(b)	親からのオフセット		領域の位置と姿勢を設定します。 設定する座標は 3D 形状データのローカル座標系における座標です。 ロケーションの要素は左から順に、X、Y、Z、y（ヨー）、p（ピッチ）、r（ロール）です。	X、Y、Z: それぞれ -1,000,000.000～+1,000,000.000 y、r: それぞれ -180.000～+180.000 p: 0.000～180.000	領域設定時の操作による
(c)	領域	色	領域の色を設定します。	透過性（0～#FF）および RGB（0～#FF、0～#FF、0～#FF）	#80F7E3AF
		DX	領域の X 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DY	領域の Y 軸方向の長さを設定します。（単位: mm）	0.1～10,000	領域設定時の操作による
		DZ	領域の Z 軸方向の長さを設定します。（単位: mm）	0.1～10,000	10

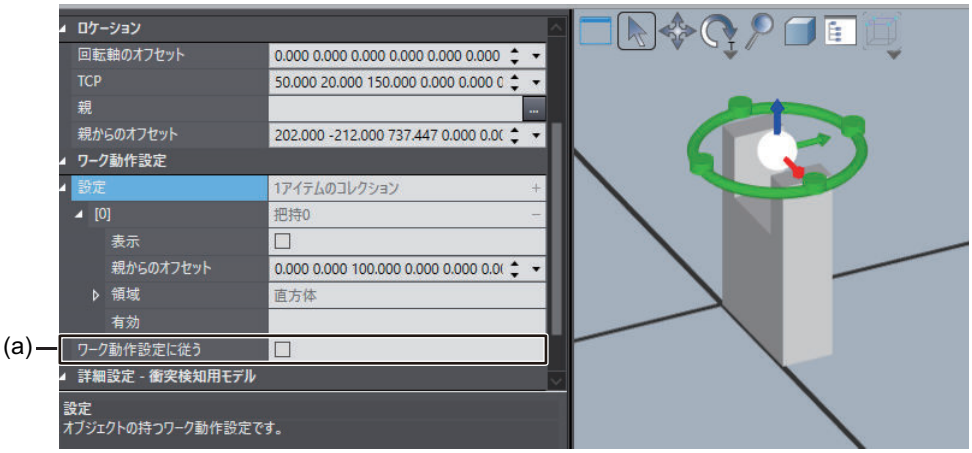
	項目	説明	設定値	初期値
(d)	有効	オプションで「物理シミュレーション」ワーク動作設定を有効／無効にします。 コントローラの BOOL 型変数名を指定することで、「物理シミュレーション」ワーク動作設定の有効化／無効化が可能です。 空白: 「物理シミュレーション」ワーク動作設定が常に有効です。 True: 「物理シミュレーション」ワーク動作設定が常に有効です。 False: 「物理シミュレーション」ワーク動作設定が常に無効です。 コントローラ名.BOOL 型のコントローラ変数名: コントローラ変数が True のときに「物理シミュレーション」ワーク動作設定が有効になります。	True False コントローラ名.BOOL 型のコントローラ変数名	空白
(e)	物理シミュレーション	ワークが領域に入ったときの物理シミュレーションの動作の有効化／無効化を設定します。	チェック ON／チェック OFF	チェック ON

7-3-9 ワーク動作設定共通の設定

ワーク動作設定は、ワーク動作設定の「生成」により生成されたワークに動きを与えるための設定ですが、「ワーク動作設定に従う」を有効にすることで、「生成」により生成されたワークではない 3D 形状データをワークと同様に取り扱うことができるようになります。以下は、ツールチェンジャーがロボットツール 1 またはロボットツール 2 を切り替える動作を実現するための構成の例です。ツールチェンジャーに「把持」の設定をし、ロボットツール 1 およびロボットツール 2 の「ワーク動作設定に従う」を有効にします。



［ワーク動作設定に従う］は 3D 形状データの設定画面で［ワーク動作設定］以下の［設定］にあります。



	項目	説明	設定値	初期値
(a)	ワーク動作設定に従う	ワークと同様に接触した 3D 形状データのワーク動作設定に沿った動きを行います。	チェック ON／チェック OFF	チェック OFF

8

3D シミュレーションの実行

本章では、3D シミュレーションの実行手順と、動作の確認方法について説明します。

8-1　3D シミュレーションの操作手順 8-2
8-1-1　コントローラのシミュレーションを実行する 8-2
8-1-2　仮想ワーク検出センサの動作スクリプト、およびメカニカル機構の仮想出力用スクリプトを実行する 8-3
8-1-3　ワークのシェイプスクリプトを実行する 8-3
8-1-4　ワーク動作設定のシミュレーションを実行する 8-4
8-1-5　ワーク動作設定によるシミュレーションを停止する 8-4
8-1-6　3D 表示画面で動作を確認する 8-4
8-1-7　3D シミュレーションを一括で実行する 8-5

8-2　シェイプスクリプトのデバッグ方法 8-8
8-2-1　ブレークポイント 8-8
8-2-2　ステップ実行 8-9
8-2-3　トレース文 8-10
8-2-4　タクトタイムの計測 8-10
8-2-5　シェイプスクリプト実行時のエラー表示 8-13

8-3　衝突検知機能 8-15
8-3-1　衝突検知対象 8-15
8-3-2　衝突検知設定手順 8-15
8-3-3　衝突検知の確認方法 8-17
8-3-4　シェイプスクリプトで衝突を検知する方法 8-18
8-3-5　衝突検知用モデルの設定 8-24

8-1　3D シミュレーションの操作手順

3D シミュレーションの実行の操作手順は、スクリプトを用いる方法とワークシミュレーション簡単設定を用いる方法で異なります。

- **スクリプトを用いる方法の場合**

  (1)「8-1-1 コントローラのシミュレーションを実行する (P.8-2)」参照

  (2)「8-1-2 仮想ワーク検出センサの動作スクリプト、およびメカニカル機構の仮想出力用スクリプトを実行する (P.8-3)」参照

  (3)「8-1-3 ワークのシェイプスクリプトを実行する (P.8-3)」参照

  (4)3D 表示画面で動作を確認する

- **ワークシミュレーション簡単設定を用いる方法の場合**

  (1)「8-1-1 コントローラのシミュレーションを実行する (P.8-2)」参照

  (2)「8-1-4 ワーク動作設定のシミュレーションを実行する (P.8-4)」参照

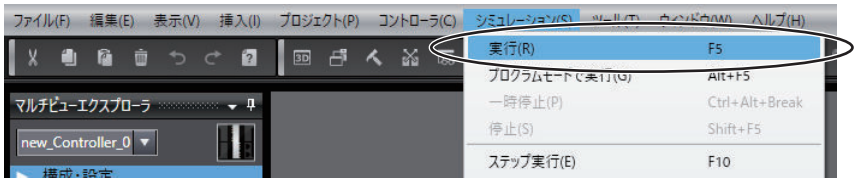
  (3)3D 表示画面で動作を確認する

8-1-1　コントローラのシミュレーションを実行する

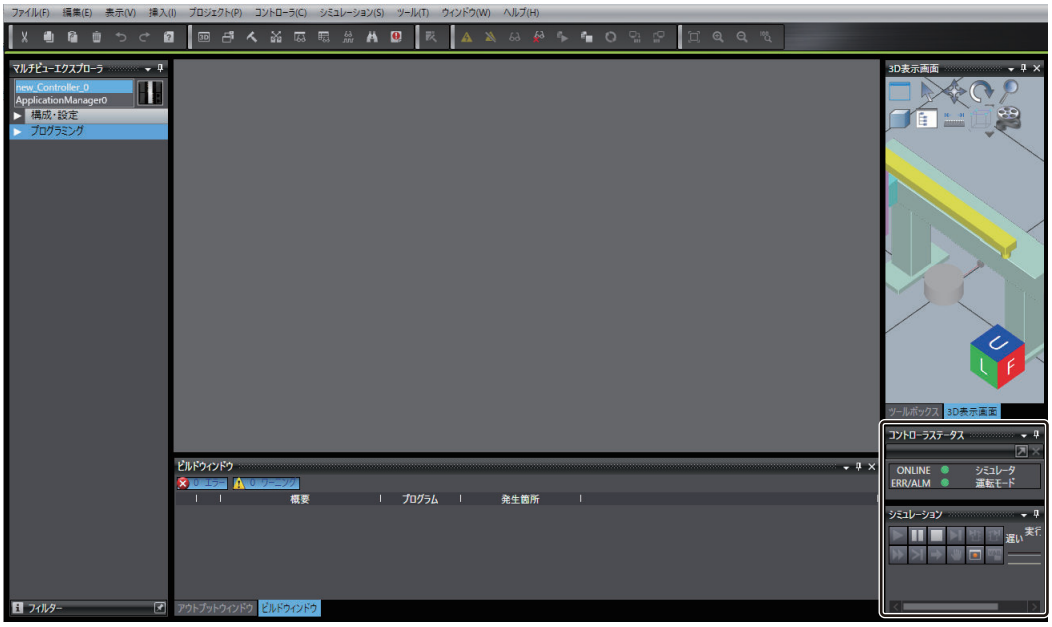
1 **デバイスリストでコントローラを選択します。**
デバイスがコントローラに切り替わります。



2 **［シミュレーション］メニューから［実行］を選択します。**



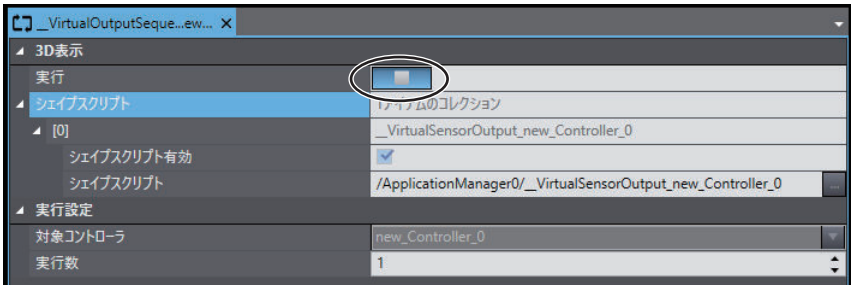
コントローラのシミュレータに接続した状態になります。



8-1-2 仮想ワーク検出センサの動作スクリプト、およびメカニカル機構の仮想出力用スクリプトを実行する

1 **デバイスリストでアプリケーションマネージャを選択し、マルチビューエクスプローラの［3D 表示］から、［__VirtualOutputSequence_（コントローラ名）］をダブルクリックします。**
［__VirtualOutputSequence_（コントローラ名）］の設定画面が表示されます。

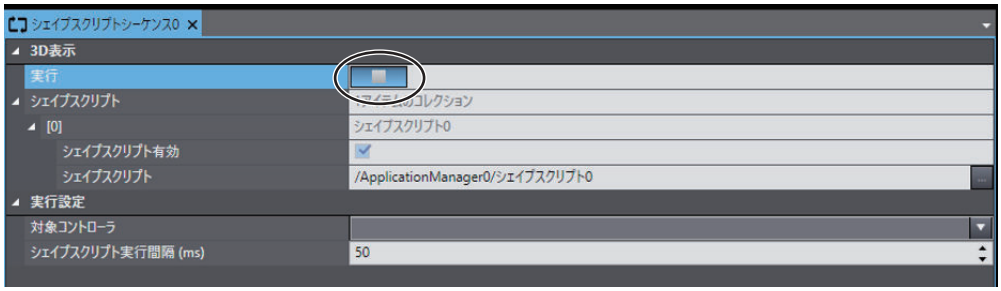
2 **［シェイプスクリプト］の［シェイプスクリプト有効］にチェックを入れ、［3D 表示］の［実行］のボタンをクリックします。**
仮想ワーク検出センサの動作スクリプト、およびメカニカル機構の仮想出力用スクリプトの実行が開始されます。



8-1-3 ワークのシェイプスクリプトを実行する

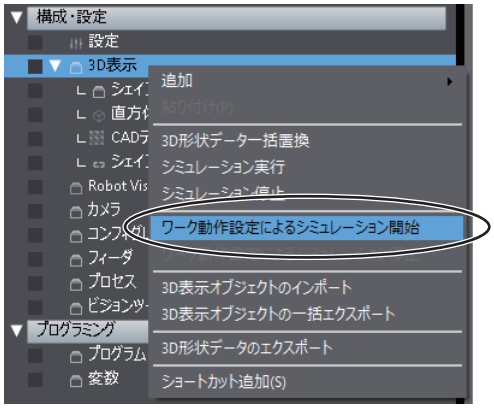
1 **デバイスリストでアプリケーションマネージャを選択し、マルチビューエクスプローラの［3D 表示］から、シェイプスクリプトシーケンスをダブルクリックします。**
シェイプスクリプトシーケンスの設定画面が表示されます。

2 **［シェイプスクリプト］の［シェイプスクリプト有効］にチェックを入れ、［3D 表示］の［実行］のボタンをクリックします。**
シェイプスクリプトの実行が開始されます。



8-1-4 ワーク動作設定のシミュレーションを実行する

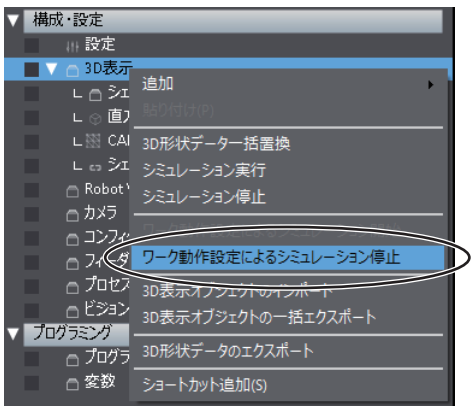
1 **デバイスリストでアプリケーションマネージャを選択し、マルチビューエクスプローラの［3D 表示］を右クリックして、［ワーク動作設定によるシミュレーション開始］を選択します。**



ワーク動作設定によるシミュレーションの実行が開始されます。

8-1-5 ワーク動作設定によるシミュレーションを停止する

1 **デバイスリストでアプリケーションマネージャを選択し、マルチビューエクスプローラの［3D 表示］を右クリックして、［ワーク動作設定によるシミュレーション停止］を選択します。**



ワーク動作設定によるシミュレーションの実行が停止されます。

8-1-6 3D 表示画面で動作を確認する

1 **3D 表示画面で、視点の変更、回転、ズームなどの操作を行い、ワークやメカニカル機構の動作を確認します。**

視点の変更、回転、ズームの操作については、「5-1 3D 表示画面、3D 編集エリアの表示 (P.5-2)」を参照してください。

ワークの動作の確認やデバッグに使用できるシェイプスクリプトエディタ機能の操作方法は、「8-2 シェイプスクリプトのデバッグ方法 (P.8-8)」を参照してください。

メカニカル機構の制御プログラムのデバッグにはラダーエディタ／ST エディタのデバッグ機能を使用します。操作方法の詳細は、Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)を参照してください。

参考

- 使用するパソコンの動作環境によっては、3D 表示画面上の仮想装置モデルの動作が実際の装置の動作よりも遅くなることがあります。3D 表示画面上の仮想装置モデルを実際の装置と同じ速度で動作させたい場合は、コントローラのシミュレーションを「実行時間予測モード」で実行してください。実行時間予測モードの機能や実行方法については、『Sysmac Studio Version 1 オペレーションマニュアル(SBCA-470)』を参照してください。
- メカニカル機構の制御プログラムのデバッグと、シェイプスクリプトのデバッグを同時に行うことができます。コントローラのデータトレースを実行して制御プログラムをデバッグしながら、デバイスをアプリケーションマネージャに切り替えてシェイプスクリプトのデバッグを行うことができます。

シミュレーション経過時間

コントローラのシミュレーションを実行すると、3D 表示画面上にコントローラのシミュレーション実行開始からの経過時間が表示されます。



項目	表示単位
シミュレーション経過時間	分 : 秒 . 1/100 秒

［シミュレーション経過時間］には、シミュレーション上の実行時間が表示されます。

使用するパソコンの負荷の変化により 3D 表示画面上の仮想装置モデルの動作が一時的に遅くなり、モデルの動作タイミングや経過時間の把握が難しいことがあります。シミュレーション経過時間は仮想装置モデルの表示上の動作と連動しているため、パソコンの負荷状況によらずモデルの動作のタイミングや経過時間を確認することができます。

参考

- シミュレーションを一時停止すると、シミュレーション経過時間の表示も一時停止します。
- シミュレーションを終了すると、シミュレーション経過時間は非表示になります。

8-1-7　3D シミュレーションを一括で実行する

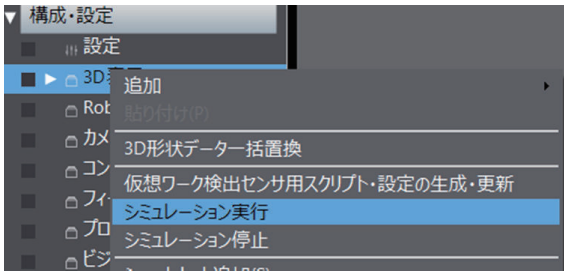
3D シミュレーションの実行は以下の操作を順番に実行する必要がありますが、これらの操作を行わずに一括で実行することが可能です。

順序	操作	参照先
1	コントローラのシミュレーションを実行する	「8-1-1 コントローラのシミュレーションを実行する (P.8-2)」
2	仮想ワーク検出センサ動作スクリプト、およびメカニカル機構の仮想出力用スクリプトを実行する	「8-1-2 仮想ワーク検出センサの動作スクリプト、およびメカニカル機構の仮想出力用スクリプトを実行する (P.8-3)」
3	ワークのシェイプスクリプトを実行する	「8-1-3 ワークのシェイプスクリプトを実行する (P.8-3)」
4	ワーク動作設定のシミュレーションを実行する	「8-1-4 ワーク動作設定のシミュレーションを実行する (P.8-4)」

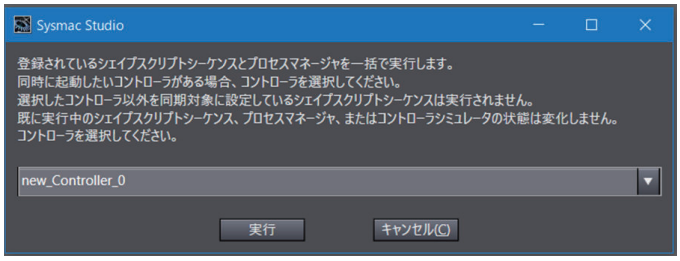
3D シミュレーションの実行を一括で実行する操作手順を説明します。

3D シミュレーションの実行

1 **マルチビューエクスプローラから［構成・設定］の［3D 表示］を右クリックして、［シミュレーション実行］を選択します。**



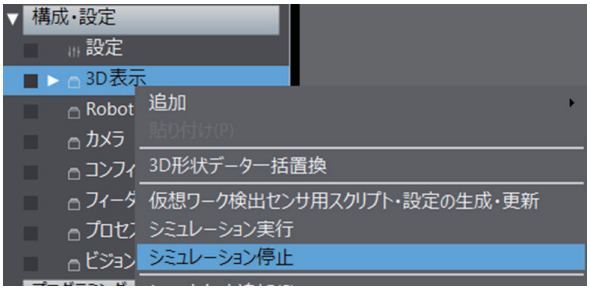
実行するコントローラの選択画面が表示されます。



2 **実行するコントローラを選択し、［実行］ボタンをクリックします。**
コントローラのシミュレータに接続した状態となり、仮想ワーク検出センサの動作スクリプト、メカニカル機構の仮想出力用スクリプト、シェイプスクリプト、およびワーク動作設定のシミュレーションの実行が開始されます。

シミュレーションの停止

1 **マルチビューエクスプローラから［構成・設定］の［3D 表示］を右クリックして、［シミュレーション停止］を選択します。**



コントローラのシミュレータが停止した状態となり、仮想ワーク検出センサの動作スクリプト、メカニカル機構の仮想出力用スクリプト、シェイプスクリプト、およびワーク動作設定のシミュレーションの実行が停止されます。

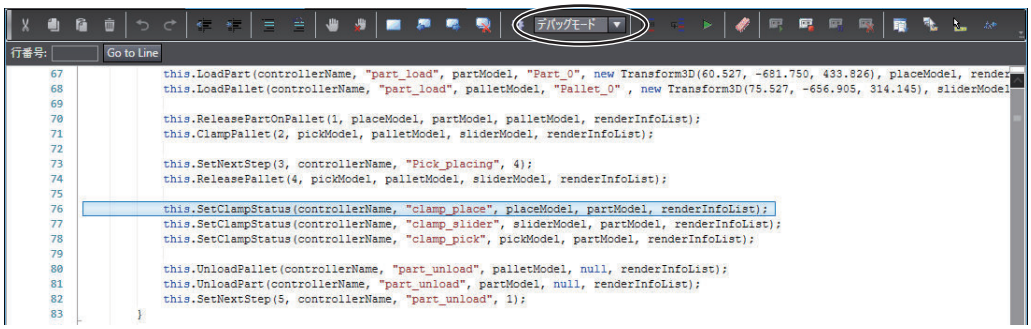
8-2 シェイプスクリプトのデバッグ方法

シェイプスクリプトのデバッグには、シェイプスクリプトエディタのデバッグ機能を使用します。あらかじめ、マルチビューエクスプローラから対象のシェイプスクリプトをダブルクリックしてシェイプスクリプトエディタを表示しておきます。

8-2-1 ブレークポイント

ブレークポイントはシェイプスクリプトの実行時、任意の行で実行を一時停止させ、その時点でのローカル変数の現在値を確認するときに使用します。
シェイプスクリプトの実行は、ブレークポイントを設定した行の先頭で一時停止します。ブレークポイントを設定した行は実行されません。

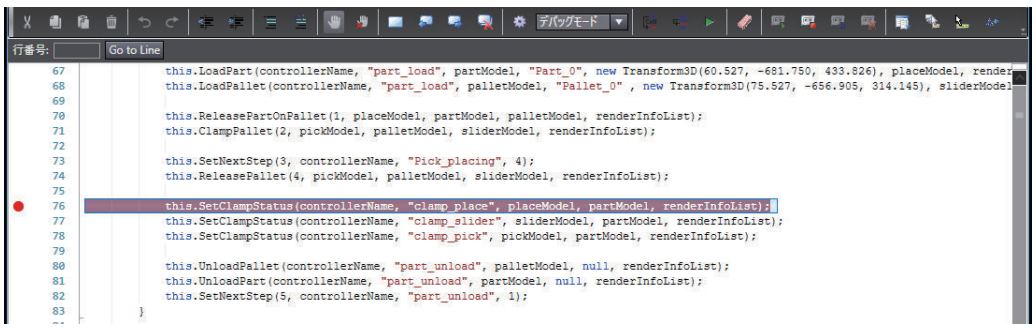
1 シェイプスクリプトエディタのツールバーの［モード選択］を「リリースモード」から「デバッグモード」に変更します。



デバッグモードに切り替わり、［ブレークポイントを切り替え］ボタンおよび［ブレークポイントのクリア］ボタンがクリックできる状態になります。



2 実行を一時停止させたい行にカーソルを移動させ、［ブレークポイントを切り替え］ボタンをクリックします。
カーソル行にブレークポイントがセットされます。



3 シェイプスクリプトシーケンスで対象のシェイプスクリプトを実行します。
ブレークポイント行の色が変化し、行番号の左に実行位置を意味する矢印のアイコンがブレークポイント行を意味するアイコンと重なって表示されます。

```
73          this.SetNextStep(3, controllerName, "Pick_placing", 4);
74          this.ReleasePallet(4, pickModel, palletModel, sliderModel, renderInfoList);
75
76          this.SetClampStatus(controllerName, "clamp_place", placeModel, partModel, renderInfoList);
77          this.SetClampStatus(controllerName, "clamp_slider", sliderModel, partModel, renderInfoList);
78          this.SetClampStatus(controllerName, "clamp_pick", pickModel, partModel, renderInfoList);
```

```
73          this.SetNextStep(3, controllerName, "Pick_placing", 4);
74          this.ReleasePallet(4, pickModel, palletModel, sliderModel, renderInfoList);
75
76          this.SetClampStatus(controllerName, "clamp_place", placeModel, partModel, renderInfoList);
77          this.SetClampStatus(controllerName, "clamp_slider", sliderModel, partModel, renderInfoList);
78          this.SetClampStatus(controllerName, "clamp_pick", pickModel, partModel, renderInfoList);
```

4 **実行位置より上の行のローカル変数にマウスカーソルを移動します。**
ローカル変数の現在値がツールチップで表示されます。

```
73          this.SetNextStep(3, controllerName, "Pick_placing", 4);
74          this.ReleasePallet(4  controllerName = "new_Controller_0"  , renderInfoList);
75
76          this.SetClampStatus(controllerName, "clamp_place", placeModel, partModel, renderInfoList);
77          this.SetClampStatus(controllerName, "clamp_slider", sliderModel, partModel, renderInfoList);
78          this.SetClampStatus(controllerName, "clamp_pick", pickModel, partModel, renderInfoList);
```

参考

［ウォッチ］タブを使用すると複数の変数の値を一度に確認することができます。［ウォッチ］タブの詳細は、「シェイプスクリプトの［ウォッチ］タブ (P.6-21)」を参照してください。

8-2-2 ステップ実行

ステップ実行はシェイプスクリプトの実行時、ブレークポイントで任意の行の先頭で実行を一時停止させたあと、行単位で実行させて変数の現在値を確認するときに使用します。

1 **ブレークポイントを設定し、シェイプスクリプトシーケンスで対象のシェイプスクリプトを実行してシェイプスクリプトのステップ実行したい箇所の前で実行を停止させます。**

```
66          var controllerName = "new_Controller_0";
67          this.LoadPart(controllerName, "part_load", partModel, "Part_0", new Transform3D(60.527, -681.750, 433.826), placeModel, renderI
68          this.LoadPallet(controllerName, "part_load", palletModel, "Pallet_0" , new Transform3D(75.527, -656.905, 314.145), sliderModel,
69
```

2 **［ステップオーバー］ボタンまたは［ステップイン］ボタンをクリックします。**



実行位置が移動します。［ステップオーバー］ボタンをクリックした場合は、実行位置が次の関数の先頭へ移動します。［ステップイン］ボタンをクリックした場合は、実行位置が対象関数内に移動します。

```
66          var controllerName = "new_Controller_0";
67          this.LoadPart(controllerName, "part_load", partModel, "Part_0", new Transform3D(60.527, -681.750, 433.826), placeModel, renderI
68          this.LoadPallet(controllerName, "part_load", palletModel, "Pallet_0" , new Transform3D(75.527, -656.905, 314.145), sliderModel,
69
```

3 **繰り返し、［ステップオーバー］ボタンまたは［ステップイン］ボタンをクリックします。**
実行結果に応じ、実行位置が変化します。

参考

［ウォッチ］タブを使用すると複数の変数の値を一度に確認することができます。［ウォッチ］タブの詳細は、「シェイプスクリプトの［ウォッチ］タブ (P.6-21)」を参照してください。

8-2-3 トレース文

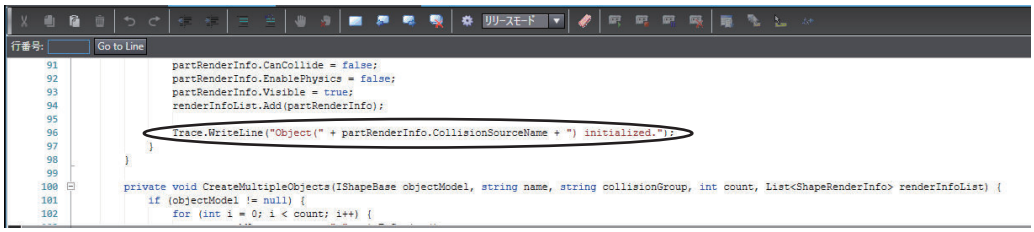
トレース文をシェイプスクリプトの任意の行に挿入することで、シェイプスクリプトのどの箇所まで処理されたかを確認することができます。シェイプスクリプトのトレース文の実行時に、トレース文で指定した文字列を［トレースメッセージ］タブにログとして出力します。

1 **シェイプスクリプトエディタを開き、任意の箇所にトレース文を挿入します。**

書式:
Trace.WriteLine（表示内容）;

表示内容には、“文字列”、またはメンバを記述します。複数の文字列、メンバを記述するときは「+」でつなぎます。

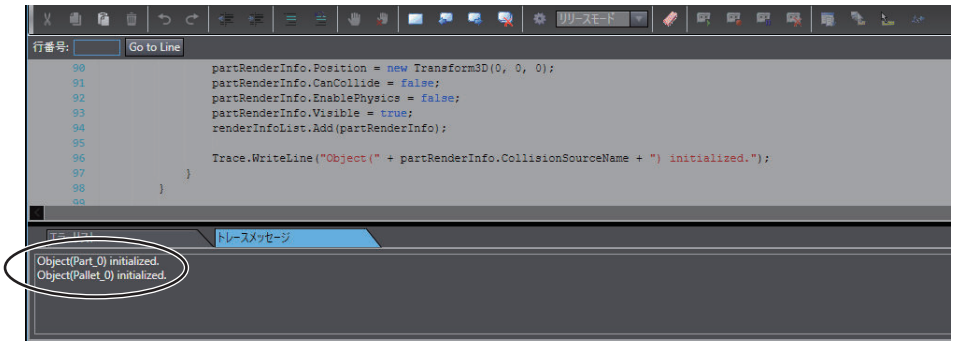
記述例: Trace.WriteLine("Object(" + partRenderInfo.CollisionSourceName + ") initialized.");
「Trace.WriteLine("Script Starting");」
「Trace.WriteLine("partDetectionSensor = "+ partDetectionSensor);」



上の例では、該当箇所が実行された結果をログに残すために、
「partRenderInfo.CollisionSourceName」の値を表示するようにしています。

2 **シェイプスクリプトシーケンスで対象のシェイプスクリプトを実行します。**

トレース文の実行結果が、シェイプスクリプトエディタの下の［トレースメッセージ］タブに表示されます。



8-2-4 タクトタイムの計測

タクトタイムは、シェイプスクリプト実行時にコントローラのシミュレーションの内部時刻を取得することで計測します。
ワークをピックし、プレースを行うアプリケーションを例に、手順を説明します。

```
行番号:  Go to Line
55      private bool isPicking;
56      private DateTime startTime;
57      private int count;
58
59      /// <summary>
60      /// Called to render the Shape Script object
61      /// </summary>
62      /// <returns>The list of shapes to render</returns>
63      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
64          if (this.IsInitialized == false) {
65              DefaultFunctions.InitializeObject(this, renderInfoList);
66              return;
67          }
68
69          var isPicked = DefaultFunctions.GetBoolVariable(this, "new_Controller_0", "IsPicked");
70          var isPlaced = DefaultFunctions.GetBoolVariable(this, "new_Controller_0", "IsPlaced");
71
72          if(isPicked && !this.isPicking){
73              // タクト開始
74              this.startTime = DefaultFunctions.GetCurrentControllerTime(this, "new_Controller_0");
75              this.isPicking = true;
76              this.count++;
77          }
78
79          if(isPlaced && this.isPicking){
80              // タクト終了
81              var currentTime = DefaultFunctions.GetCurrentControllerTime(this, "new_Controller_0");
82              var taktTime = currentTime - this.startTime;
83              Trace.WriteLine(count + " : " + taktTime.TotalMilliseconds + " ms");
84
85              this.isPicking = false;
86          }
87      }
88  }
```

1 **変数を定義します。**

- isPicking（BOOL 型）：ワークをピックしている間、TRUE となる BOOL 型の変数です。タクトの開始時あるいは終了時に処理を行うかどうか、判断する場合に使用します。
- startTime（DATETIME 型）：タクト開始時の時刻を記録する変数です。
- count（INT 型）：タクトの番号を記録する変数です。

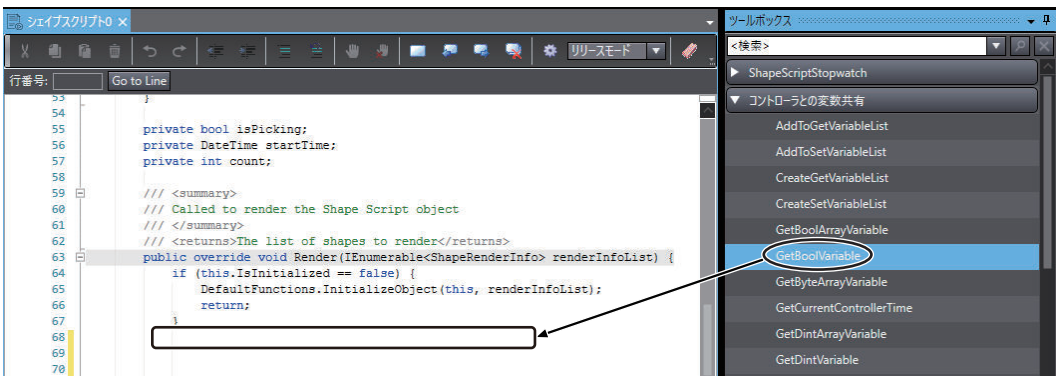
```
行番号:  Go to Line
55      private bool isPicking;
56      private DateTime startTime;
57      private int count;
58
```

2 **コントローラの変数をシェイプスクリプトの Render 関数内で取得します。**

コントローラに下記変数が存在しているものとします。

- IsPicked（BOOL 型）：ワークをピックしている間、TRUE となる BOOL 型の変数です。
- IsPlaced（BOOL 型）：ワークをプレースし、次のピックが開始されるまで TRUE となる BOOL 型の変数です。

シェイプスクリプトの Render 関数内で上記変数を取得します。
コントローラの BOOL 型変数を取得するために、GetBoolVariable 関数を追加します。
追加方法は、ツールボックスから［コントローラとの変数共有］を選択し、［GetBoolVariable］をドラッグし、シェイプスクリプトエディタ画面にドロップします。GetBoolVariable 関数の呼び出しが挿入されるので、パラメータを設定します。



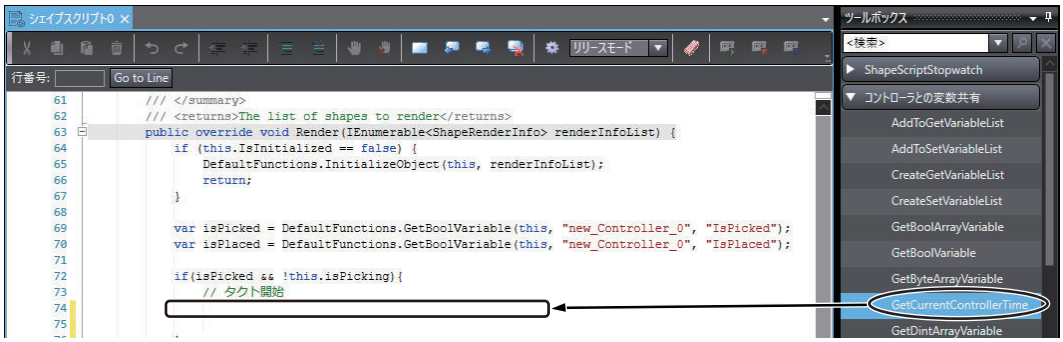
［GetBoolVariable］の、ひとつ目の引数にコントローラ名「new_Controller_0」を、二つ目の引数にコントローラの変数名「IsPicked」を指定します。
また、取得した値を代入する変数を定義します。ここでは変数「IsPicked」を定義します。
同様にコントローラの変数名「IsPlaced」を代入する「IsPlaced」を定義します。

```
63      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
64          if (this.IsInitialized == false) {
65              DefaultFunctions.InitializeObject(this, renderInfoList);
66              return;
67          }
68
69          var isPicked = DefaultFunctions.GetBoolVariable(this, "new_Controller_0", "IsPicked");
70          var isPlaced = DefaultFunctions.GetBoolVariable(this, "new_Controller_0", "IsPlaced");
71
```

3 タクト開始時の処理を記述します。

タクト開始時をピックが開始された時点とします。変数「IsPicked」が TRUE となった時点です。
その時点で、コントローラのシミュレーションの内部時刻を取得し、変数「startTime」に代入します。シミュレーションの内部時刻を取得する関数は下記となります。

コントローラのシミュレーションの内部時刻を取得するために、GetCurrentControllerTime 関数を追加します。
追加方法は、ツールボックスから［コントローラとの変数共有］を選択し、［GetCurrentControllerTime］をドラッグし、シェイプスクリプトエディタ画面にドロップします。GetCurrentControllerTime 関数の呼び出しが挿入されるので、パラメータを設定します。



タクト開始時、変数「isPicking」に TRUE を代入します。次回以降の Render 関数の呼び出しが行われても、同じ処理は通らなくなります。
変数「count」を 1 加算します。［トレースメッセージ］タブに結果が出力されます。

```
69          var isPicked = DefaultFunctions.GetBoolVariable("new_Controller_0", "IsPicked");
70          var isPlaced = DefaultFunctions.GetBoolVariable("new_Controller_0", "IsPlaced");
71
72          if(isPicked && !this.isPicking){
73              // タクト開始
74              this.startTime = DefaultFunctions.GetCurrentControllerTime(this, "new_Controller_0");
75              this.isPicking = true;
76              this.count++;
77          }
78
```

4 タクト終了時の処理を記述します。

タクト終了時をワークのプレースされた時点とします。変数「isPlaced」が TRUE となった時点です。
その時点で、コントローラのシミュレーションの内部時刻を取得し、新たな変数「currentTime」に代入します。
変数「currentTime」とタクト開始時に内部時刻を設定した変数「startTime」との差を変数「taktTime」に代入します。
Trace.WriteLine 関数で、取得したタクトタイムを［トレースメッセージ］タブに出力します。

変数「isPicking」に FALSE を代入します。これにより、次回のピック開始時点でタクト開始時の処理が実行されるようになります。

```
            this.isPicking = true;
            this.count++;
        }

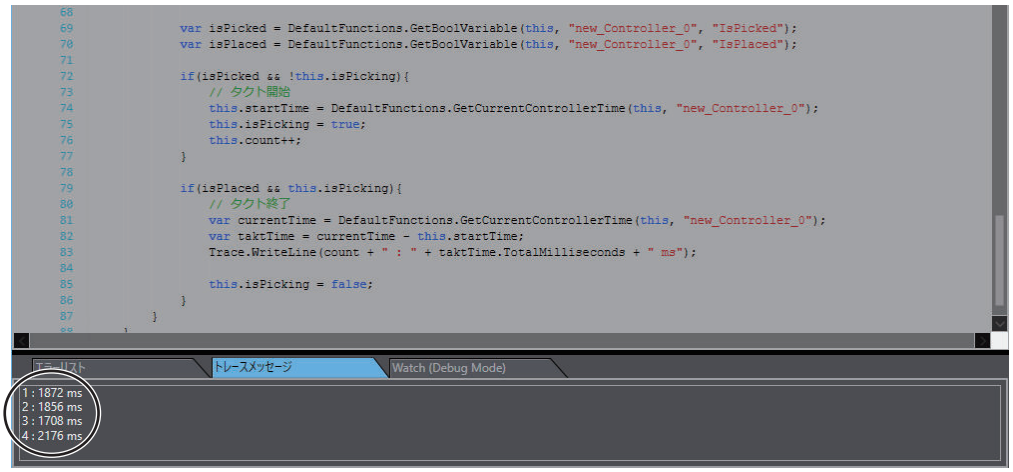
        if(isPlaced && this.isPicking){
            // タクト終了
            var currentTime = DefaultFunctions.GetCurrentControllerTime(this, "new_Controller_0");
            var taktTime = currentTime - this.startTime;
            Trace.WriteLine(count + " : " + taktTime.TotalMilliseconds + " ms");

            this.isPicking = false;
        }
    }
}
```

5 **シェイプスクリプトシーケンスで対象のシェイプスクリプトを実行します。**

［トレースメッセージ］タブに、タクトタイムが表示されます。

コントローラの変数「IsPicked」が TRUE になってから、変数「IsPlaced」が TRUE になるまでのシミュレーションの内部時刻の差が表示されます。



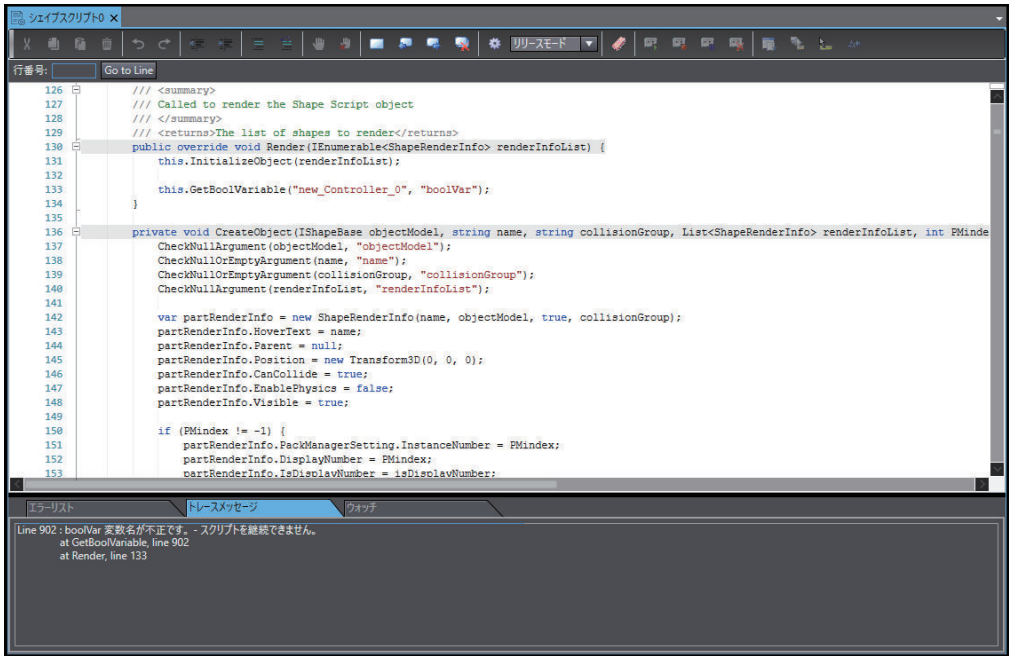
参考

Stopwatch 構文を使用すると、シェイプスクリプトの任意の処理の実行時間を計測することができます。詳細は「A-3-2 Stopwatch 構文を使用した実行時間の計測 (P.A-68)」を参照してください。

8-2-5 シェイプスクリプト実行時のエラー表示

シェイプスクリプトシーケンスから対象のシェイプスクリプトを実行した場合に、シェイプスクリプトを正しく実行できないときは、エラーの発生箇所が［トレースメッセージ］タブに表示されます。［トレースメッセージ］タブには、エラーの発生している関数と発生している箇所の行番号が表示され、その下にエラーの発生している関数を呼び出している関数と呼び出している行番号が表示されます。エラーの発生している関数が複数の関数を介して呼び出されている場合は、呼び出し履歴の順に、関数名と呼び出している箇所の行番号が表示されます。呼び出し履歴の一番下には、Render()関数と、対象の関数を呼び出している箇所の行番号が表示されます。

たとえば、コントローラの変数を取得しようとしたときに指定した変数が見つからない場合は、次のようなメッセージが表示されます。



この例では、Render()関数から GetBoolVariable()関数が実行され、GetBoolVariable()関数内でエラーが発生しています。GetBoolVariable()関数は Render()関数から指定されたパラメータを受けて実行されます。このとき、Render()関数から指定されたパラメータに誤りがあると、GetBoolVariable()関数はエラーを出力します。ここでは、Render()関数で GetBoolVariable()関数を呼び出している箇所を修正する必要があります。
この例の場合、GetBoolVariable()関数の 902 行目で失敗していることがわかります。GetBoolVariable()関数を呼び出しているのは、Render()関数の 133 行目であることが示されています。したがって、133 行目の修正が必要であるということがわかります。この例では引数のコントローラの変数名が間違っています。

8-3 衝突検知機能

装置を構成する部品やワークなどの 3D 形状データどうしが動作時に接触するかどうかを検知する機能です。
3D シミュレーションであらかじめ接触の有無を確認し、修正することで、動作時の意図しない部品どうしの衝突を防ぐことができます。

8-3-1 衝突検知対象

衝突検知が可能な 3D 形状データは以下のとおりです。

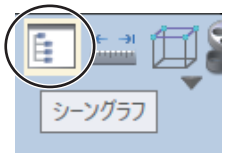
モデル種別	衝突検知の対象
メカニカル機構	各可動部品またはメカニカル機構全体
ワーク	ワークの設定に含まれるすべての 3D 形状データ

8-3-2 衝突検知設定手順

衝突検知機能の設定手順を説明します。
衝突を検知するために、衝突する可能性のある対象物を異なる「衝突検知グループ」に登録します。異なる衝突検知グループの 3D 形状データどうしの衝突が検知されます。同じ衝突検知グループの 3D 形状データどうしの衝突は検知されません。
「メカニカル機構の一部がワークに接触するケース」を例に、メカニカル機構とワークの衝突検知の設定方法を説明します。
あらかじめ次のメカニカル機構とワークの 3D 形状データを登録しておきます。

メカニカル機構の名称: MechanicalModel000
ワークの名称: Part

1 **3D 表示画面の［シーングラフ］アイコンをクリックします。**

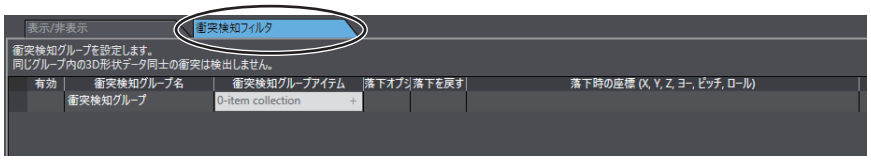


［シーングラフ］ダイアログボックスが表示されます。

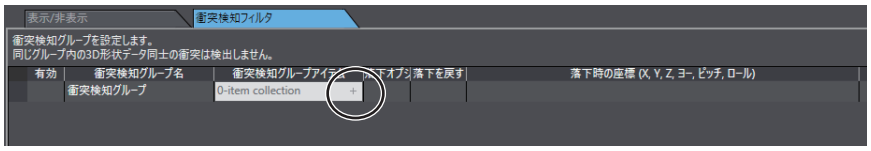


2 **［衝突検知フィルタ］タブをクリックします。**

［衝突検知フィルタ］画面に切り替わります。



3 **衝突検知グループの［+］ボタンをクリックします。**



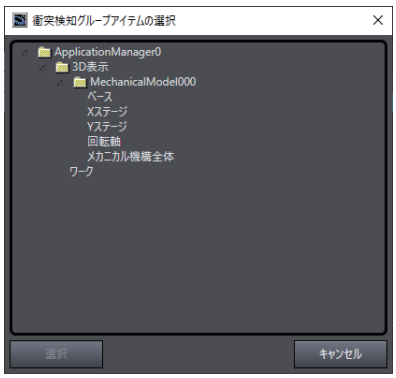
衝突検知グループが追加されます。



4 **追加された衝突検知グループのアイテム選択ボタンをクリックします。**



衝突検知グループに対象の 3D 形状データを登録するためのダイアログボックスが表示されます。



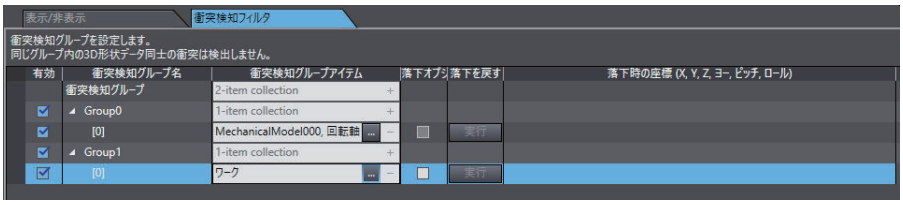
5 **衝突検知グループに登録する対象のメカニカル機構の 3D 形状データを選択し、［選択］ボタンをクリックします。**



衝突検知グループに、選択した 3D 形状データが登録されます。



6 **同様にして、ワークの 3D 形状データを登録します。**
このときメカニカル機構の 3D 形状データの衝突検知グループと異なるグループに設定します。



7 **衝突検知を行いたい 3D 形状データおよびグループの［有効］チェックボックスにチェックが入っていることを確認し、［OK］ボタンをクリックします。**

使用上の注意

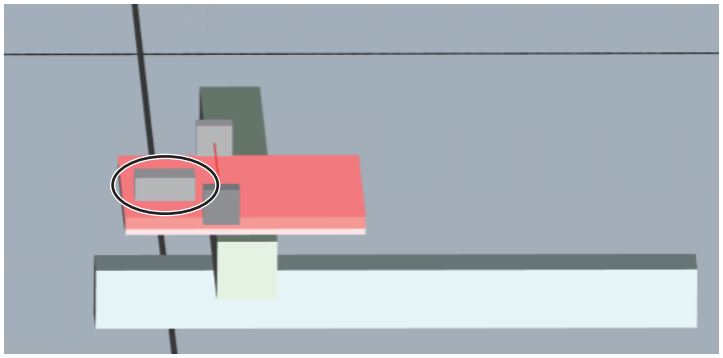
仮想ワーク検出センサを追加し、検出対象のワークを設定すると、衝突検知グループに「__VirtualSensorOutputGroup」と「ワーク名称」が登録されます。仮想ワーク検出センサの動作に必要な設定ですので、削除しないように注意してください。

8-3-3 衝突検知の確認方法

3D シミュレーション実行時の衝突検知の確認方法を説明します。
衝突検知の確認は、3D 表示画面上で行います。

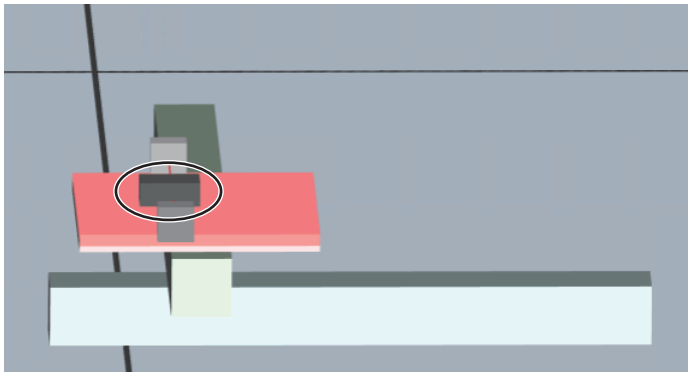
衝突していない 3D 形状データ

3D 形状データの設定で指定されている色で表示されます。



衝突した 3D 形状データ

3D 形状データの設定で指定されている色よりも濃い色で表示されます。



該当する 3D 形状データが衝突した場合は、シェイプスクリプトエディタ上の該当する 3D 形状データのプロパティの値が変化します。このプロパティの値をトレース文で出力することでも衝突したことを確認できます。

意図しない衝突が発生した場合は、衝突を回避するために以下の要素を見直してください。
- 衝突した 3D 形状データの位置
- 衝突した 3D 形状データの形状
- 衝突した 3D 形状データの動作に関係する制御プログラムとパラメータ

8-3-4　シェイプスクリプトで衝突を検知する方法

シェイプスクリプトを使用して衝突を検知する方法や、衝突した対象の 3D 形状データの名称を取得する方法を説明します。

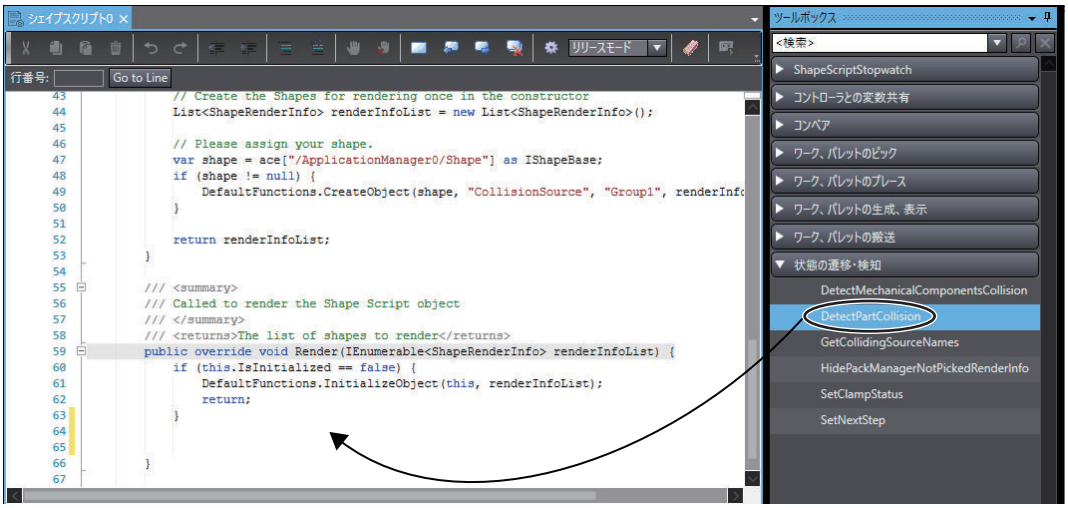
ワークまたはパレットがメカニカル機構と衝突したことを検知する方法

ワークまたはパレットが、メカニカル機構と衝突したことを検知する方法を説明します。

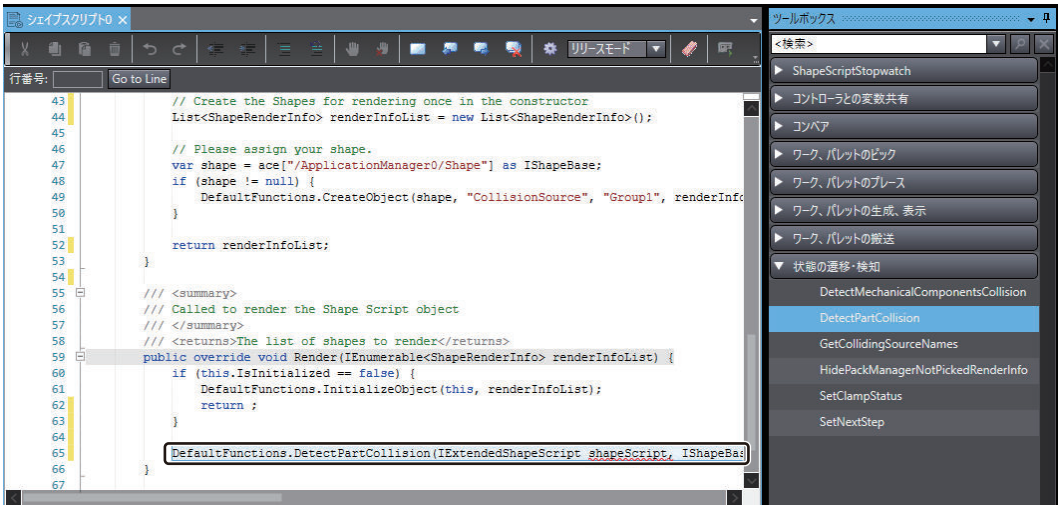
1 **衝突を検知したいメカニカル機構を衝突検知フィルタに追加します。**
衝突検知フィルタへの追加方法については、「8-3-2 衝突検知設定手順 (P.8-15)」を参照してください。

2 **シェイプスクリプトの Render()関数に DetectPartCollision()関数を追加します。**

ツールボックスの［状態の遷移・検知］から［DetectPartCollision］をドラッグし、Render() 関数にドロップします。



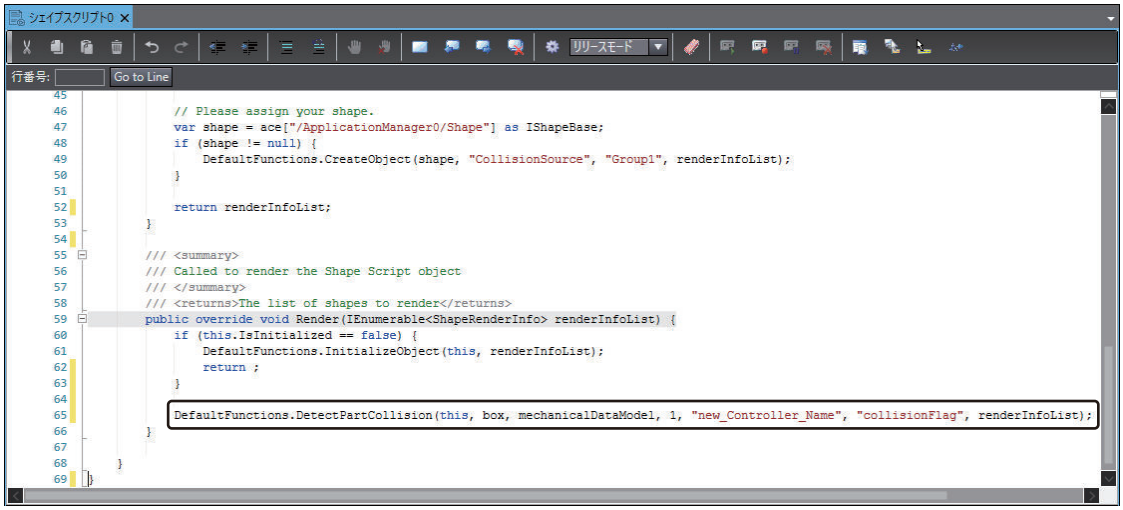
ドロップした箇所に DetectPartCollision()関数が挿入されます。



3 **DetectPartCollision()関数の引数に適切な変数を指定します。**

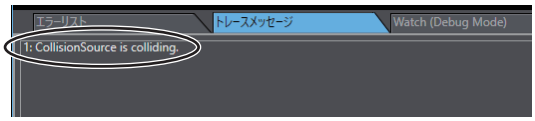
DetectPartCollision()関数の詳細については「DetectPartCollision (P.A-34)」を参照してください。

以下は、ワークのインスタンス「box」と、メカニカル機構のインスタンス「mechanicalDataModel」が接触したとき、コントローラ「new_Controller_0」に含まれる BOOL 型グローバル変数「collisionFlag」が ON する例を示します。



4 **コントローラのシミュレータを起動します。**

5 **シェイプスクリプトシーケンスに、DetectPartCollision()関数を実行するシェイプスクリプトを追加して、実行します。**
シェイプスクリプトの追加手順については「6-3-2 シェイプスクリプトの実行設定 (P.6-12)」を、シェイプスクリプトの実行方法については「8-1-3 ワークのシェイプスクリプトを実行する (P.8-3)」を参照してください。
シェイプスクリプトを実行し、衝突を検知するとシェイプスクリプトの［トレースメッセージ］タブにログが表示されます。



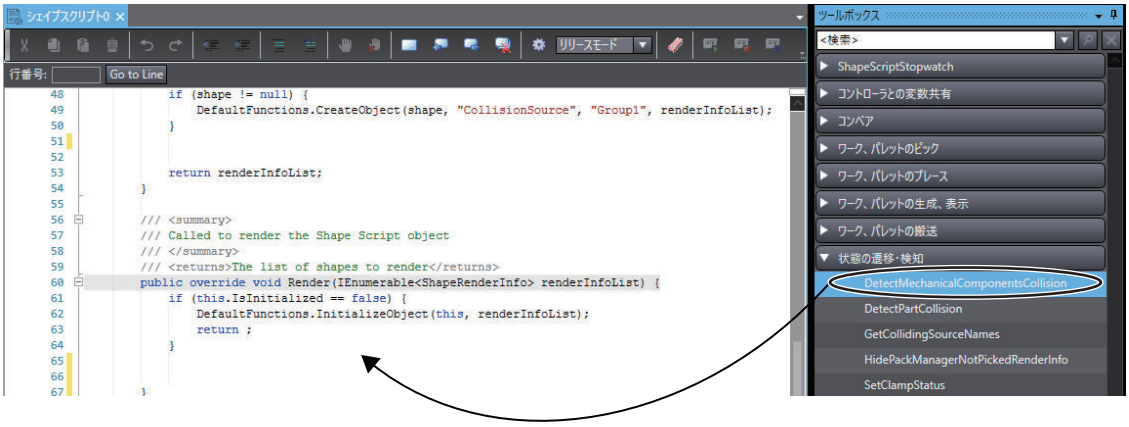
手順 3 の例では、コントローラ「new_Controller_0」に含まれる BOOL 型グローバル変数「collisionFlag」が ON に切り替わります。

メカニカル機構がほかのメカニカル機構に衝突したことを検知する方法

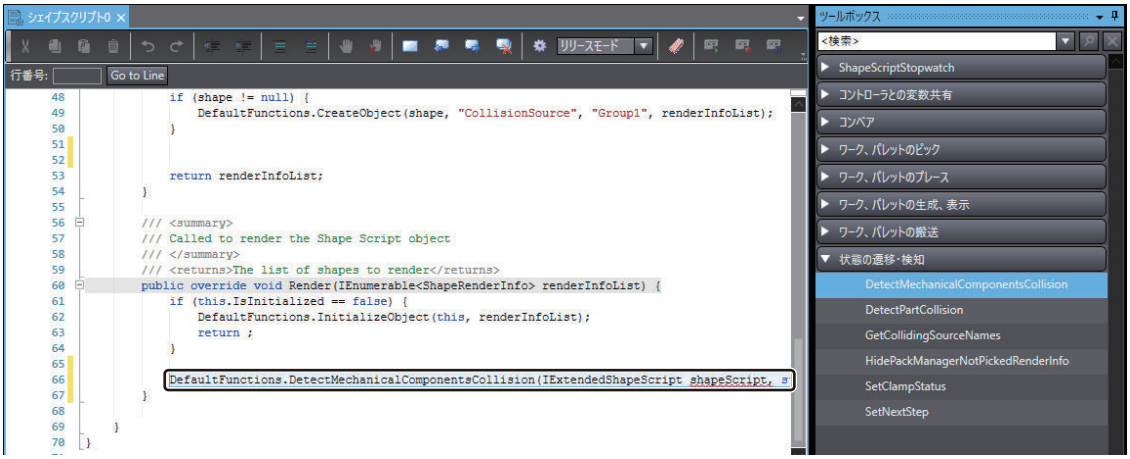
メカニカル機構がほかのメカニカル機構に衝突したことを検知する方法を説明します。

1 **衝突を検知したいメカニカル機構を、それぞれ衝突検知フィルタに追加します。**
衝突検知フィルタへの追加方法については、「8-3-2 衝突検知設定手順 (P.8-15)」を参照してください。

2 **シェイプスクリプトの Render()関数に DetectMechanicalComponentsCollision()関数を追加します。**
ツールボックスの［状態の遷移・検知］から［DetectMechanicalComponentsCollision］をドラッグし、Render()関数にドロップします。

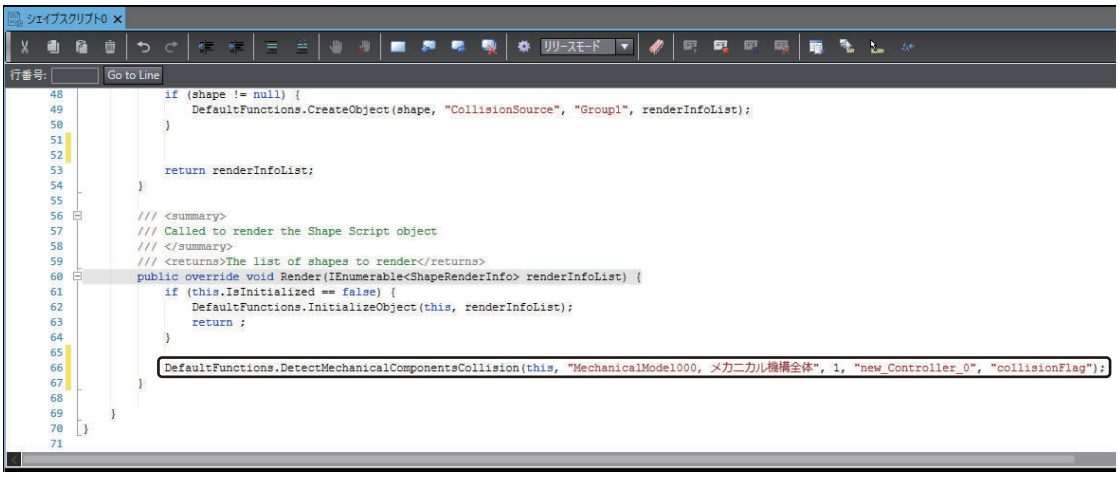


ドロップした箇所に DetectMechanicalComponentsCollision()関数が挿入されます。



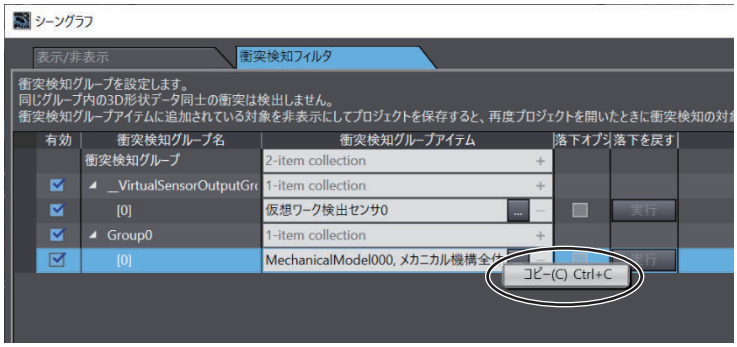
3 **DetectMechanicalComponentsCollision()関数の引数に適切な変数を指定します。**
DetectMechanicalComponentsCollision()関数の詳細については「DetectMechanicalComponentsCollision (P.A-35)」を参照してください。

以下に、メカニカル機構「MehanicalModel000」がほかのメカニカル機構に衝突したときに、コントローラ「new_Controller_0」に含まれる BOOL 型グローバル変数「collisionFlag」が ON する例を示します。



参考

DetectMechanicalComponentsCollision()関数の第一引数「衝突検知フィルタに登録されている名前」は、衝突検知フィルタから取得することができます。[衝突検知フィルタ]画面で、対象のメカニカル機構の衝突グループアイテムをコピーしてください。「衝突検知フィルタに登録されている名前」をシェイプスクリプトに貼り付けることができます。

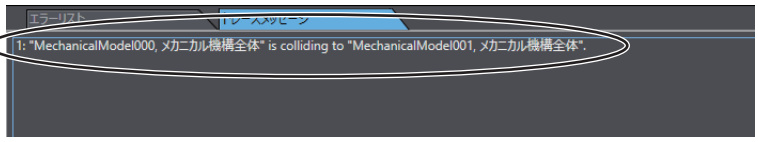


4 **コントローラのシミュレータを起動します。**

5 **シェイプスクリプトシーケンスに、DetectMechanicalComponentsCollision()関数を実行するシェイプスクリプトを追加して、実行します。**

シェイプスクリプトの追加手順については「6-3-2 シェイプスクリプトの実行設定 (P.6-12)」を、シェイプスクリプトの実行方法については「8-1-3 ワークのシェイプスクリプトを実行する (P.8-3)」を参照してください。

シェイプスクリプトを実行し、衝突を検知するとシェイプスクリプトの［トレースメッセージ］タブにログが表示されます。



参考

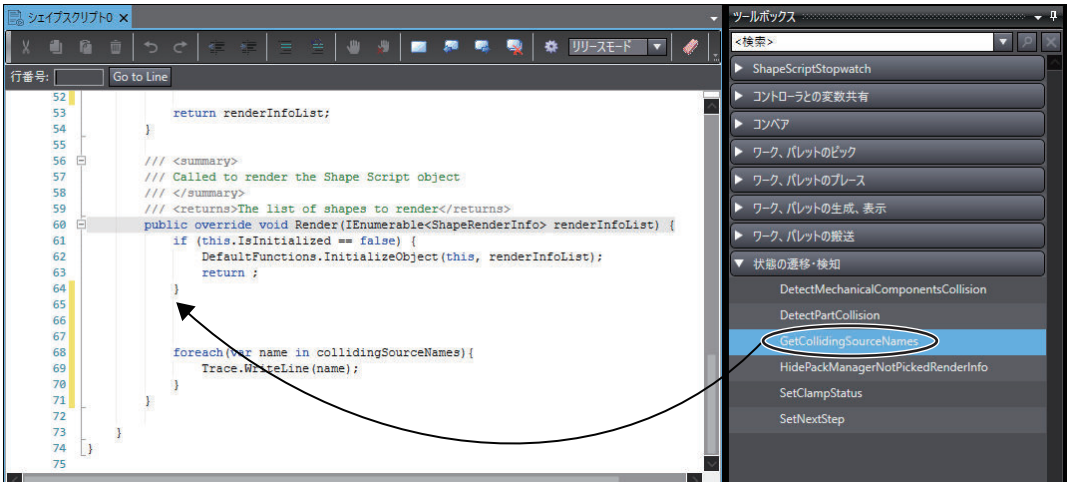
衝突が検知されたときにコントローラの変数を変更する必要がない場合は、DetectMechanicalComponentsCollision()関数の引数「コントローラ名」と「コントローラの BOOL 型グローバル変数名」に「string.Empty（空文字）」を設定してください。

衝突した対象の 3D 形状データの名称を取得する方法

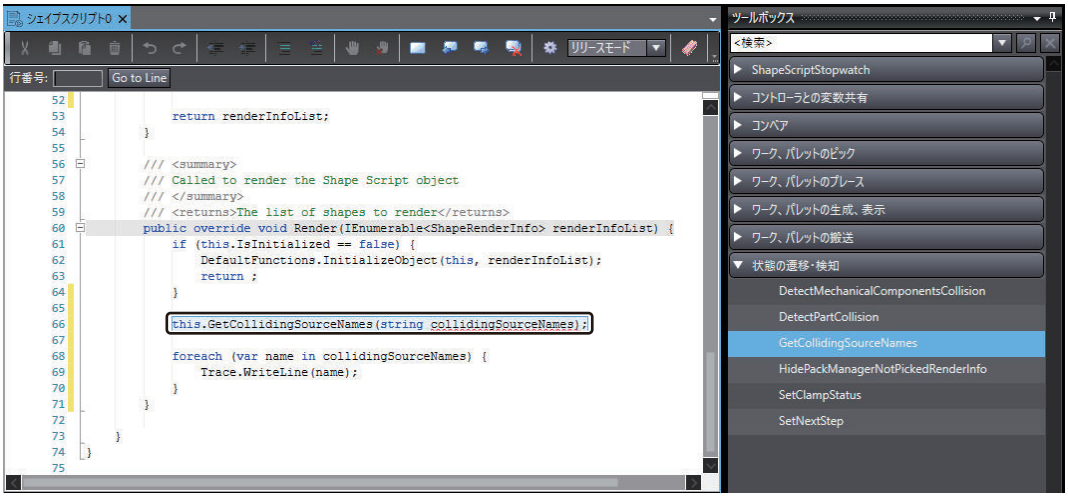
衝突した対象の 3D 形状データの名称を取得する方法を説明します。

1 **シェイプスクリプトの任意の箇所に GetCollidingSourceNames()関数を追加します。**

ツールボックスの［状態の遷移・検知］から［GetCollidingSourceNames］をドラッグし、任意の箇所にドロップします。



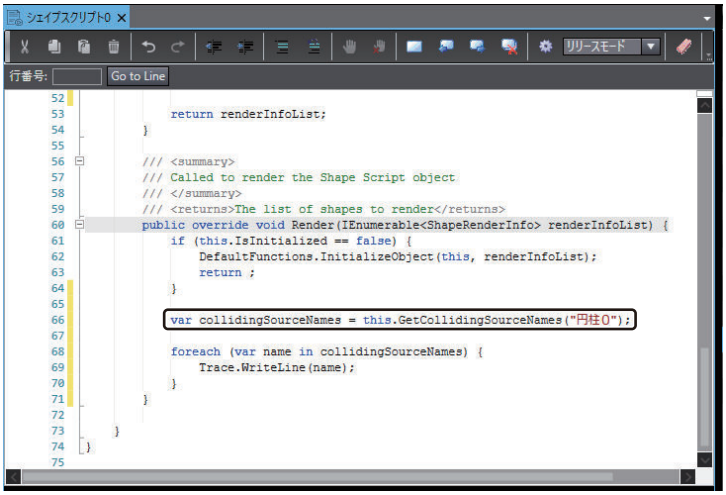
ドロップした箇所に GetCollidingSourceNames()関数が挿入されます。



2 GetCollidingSourceNames()関数の引数に適切な変数を指定します。

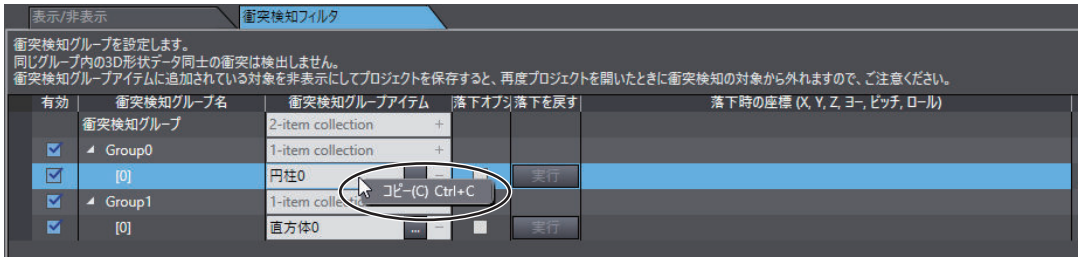
GetCollidingSourceNames()関数の詳細については「GetCollidingSourceNames (P.A-37)」を参照してください。

以下は「円柱 0」と衝突している「3D 形状データの名称」を取得し、変数「collidingSourceNames」に代入する例を示します。



参考

GetCollidingSourceNames()関数の引数「3D 形状データの名称」は衝突検知フィルタから取得することができます。［衝突検知フィルタ］画面で対象の衝突グループアイテムをコピーしてください。

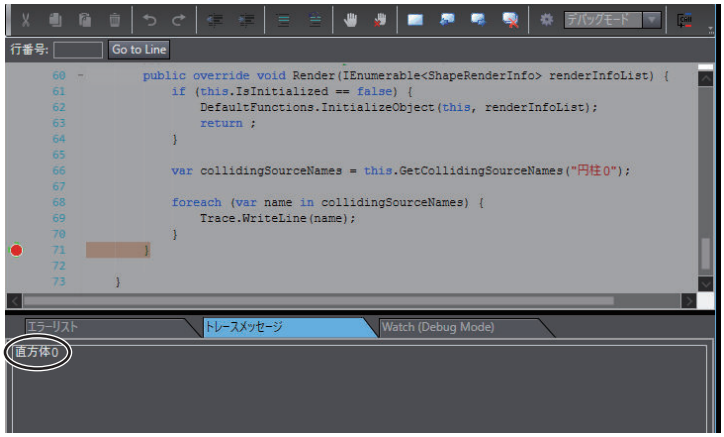


3 **コントローラのシミュレータを起動します。**

4 **シェイプスクリプトシーケンスに、GetCollidingSourceNames()関数を実行するシェイプスクリプトを追加して、実行します。**

シェイプスクリプトの追加手順については「6-3-2 シェイプスクリプトの実行設定 (P.6-12)」を、シェイプスクリプトの実行方法については「8-1-3 ワークのシェイプスクリプトを実行する (P.8-3)」を参照してください。

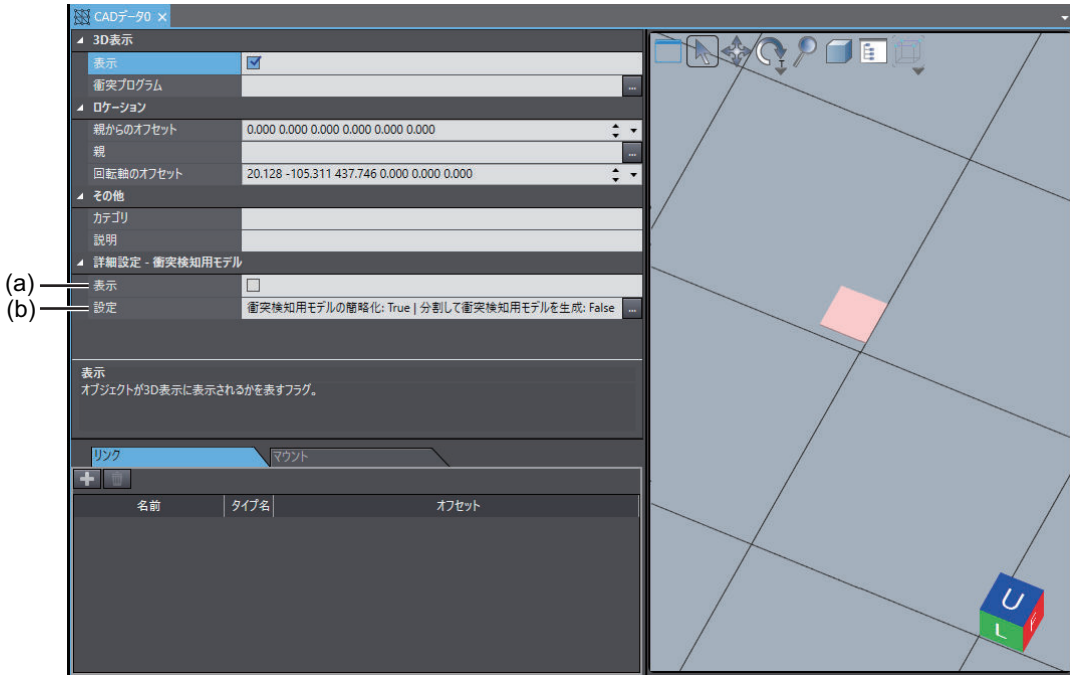
シェイプスクリプトを実行すると、「直方体 0」が「円柱 0」と衝突している対象として検出され、［トレースメッセージ］タブにログが表示されます。



8-3-5 衝突検知用モデルの設定

衝突検知用モデルとは、3D 表示画面で動作する 3D 形状データどうしの衝突を検知するための、3D 形状データを模した三角錐の集合体です。

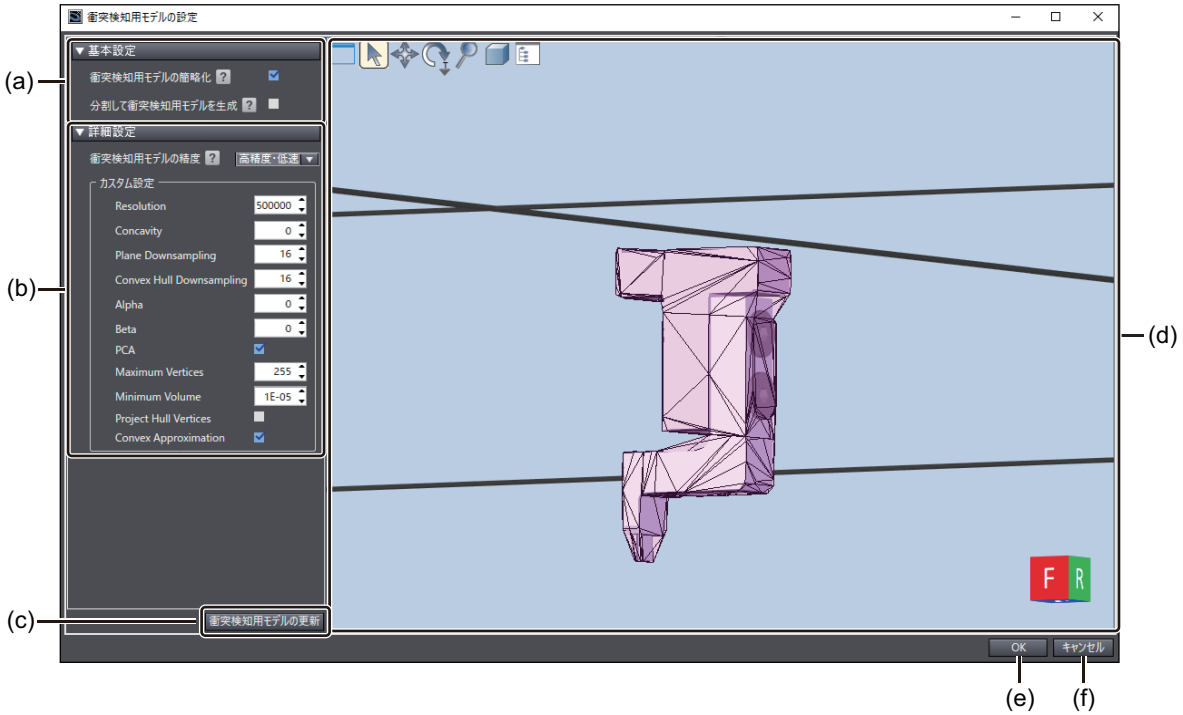
この設定を変更することで、3D 形状データの形状や複雑さに応じ、衝突検知の精度に関係する衝突検知用モデルを調整することができます。



	項目	説明	設定値	初期値
(a)	表示	衝突検知用モデルを表示するかどうかを設定します。	チェック ON／チェック OFF	チェック OFF
(b)	設定	衝突検知用モデルの設定を行います。右側のボタンから、衝突検知用モデルの詳細な設定を行います。	-	衝突検知用モデルの簡略化: True | 分割して衝突検知用モデルを生成: False

衝突検知用モデルの詳細設定

［詳細設定 - 衝突検知用モデル］の［設定］の右側のボタンをクリックすると、衝突検知用モデルの詳細設定を行う［衝突検知用モデルの設定］ダイアログボックスが表示されます。
衝突検知用モデルの設定項目は次のとおりです。



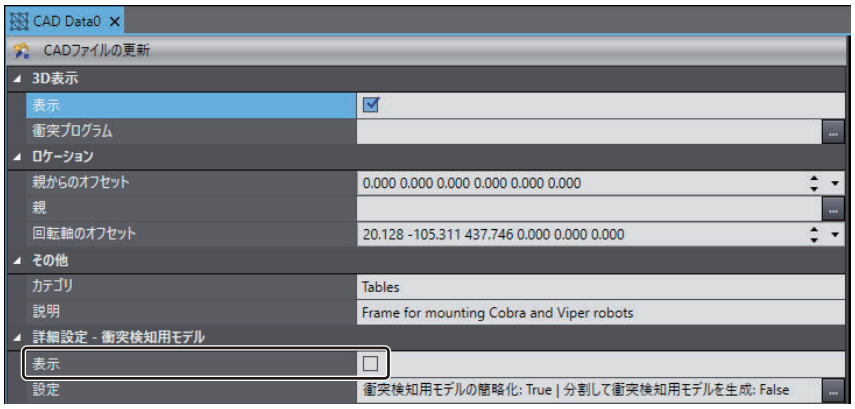
	設定項目		説明	設定値	初期値
(a)	基本設定	衝突検知用モデルの簡略化(Convex)	衝突検知用モデルの形状を簡略化する／しないを切り替えます。チェック ON で衝突検知用モデルを簡略化します。 衝突検知用モデルを簡略化した場合、衝突検知の精度は下がりますが、3D 表示画面の描画速度は向上します。ベルトやテーブルなどの静的な 3D 形状データを除き、必ず簡略化してください。簡略化しない場合、衝突検知用モデルが正しく表示されないことがあります。 チェックを外し、衝突検知用モデルの簡略化をしない場合には、以下の動作となります。 • オブジェクトどうしの衝突は検出されません。 • 衝突検知用モデルが簡略化されていないオブジェクトは、表面の衝突のみ検出します。内部の衝突は検出しません。 • 落下オプションを設定できません。	チェック ON／チェック OFF	チェック ON
		分割して衝突検知用モデルを生成(Multiple Hulls)	3D 形状データを複数に分割して衝突検知用モデルを生成するかどうかを設定します。 分割した場合、3D 表示画面の描画速度は遅くなりますが、衝突検知の精度が向上します。 以下の場合は設定できません。 • 3D 形状データが単一の部品で構成されている場合 • 衝突検知用モデルの生成をサポートしていないバージョンの Sysmac Studio で生成された 3D 形状データの場合	チェック ON／チェック OFF	チェック OFF

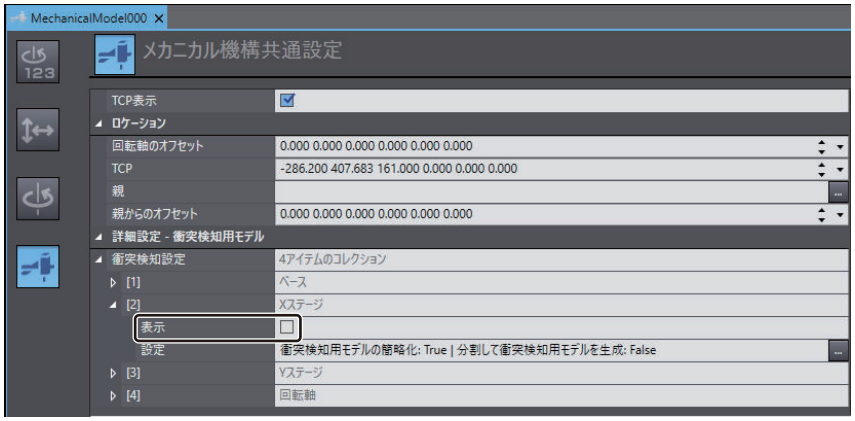
	設定項目		説明	設定値	初期値
(b)	詳細設定 (Convex Decomposition)	衝突検知用モデルの精度	衝突検知用モデルの精度を設定します。［カスタム］を選択した場合、以下のパラメータを調整できるようになります。設定を変更して下部の［衝突検知用モデルの更新］ボタンをクリックすると、右の 3D 編集エリアに衝突検知用モデルが表示され、精度を確認できます。	設定なし／低精度・高速／中精度・中速／高精度・低速／カスタム	設定なし
		Resolution	衝突検知用モデルをボクセル化する段階で生成されるボクセルの最大数を設定します。	10,000-64,000,000	100,000
		Concavity	最大凹面を設定します。	0.0-1.0	0.0
		Plane Downsampling	最も良いクリッピング平面の検索粒度を制御する値を設定します。	1-16	4
		Convex Hull Downsampling	クリッピング平面の選択段階での凸包生成プロセスの精度を制御する値を設定します。	1-16	4
		Alpha	対称面に沿ったクリッピングバイアスを制御する値を設定します。	0.0-1.0	0
		Beta	回転軸に沿ったクリッピングバイアスを制御する値を設定します。	0.0-1.0	0
		PCA	3D 形状データのメッシュを正規化する／しないを切り替えます。チェック ON で正規化します。	チェック ON／チェック OFF	チェック OFF
		Maximum Vertices	ひとつの衝突検知用モデルの頂点数の最大値を設定します。	5-1024	255
		Minimum Volume	生成された衝突検知用モデルの適応サンプリングを制御する値を設定します。	0.0-0.01	0.00001
		Project Hull Vertices	衝突検知用モデルに 3D 形状データの頂点を投影する／しないを切り替えます。チェック ON で投影します。投影すると、衝突検知用モデルの精度が向上します。	チェック ON／チェック OFF	チェック OFF
		Convex Approximation	衝突検知用モデルを近似する／しないを切り替えます。チェック ON で近似します。近似とは、浮動小数点の損失精度を意味します。	チェック ON／チェック OFF	チェック OFF
(c)	［衝突検知用モデルの更新］ボタン		設定内容を 3D 編集エリアに表示される衝突検知用モデルに反映します。設定を変更すると、ボタンは有効になります。	-	-
(d)	3D 編集エリア		突検知用モデルを表示する画面です。［衝突検知用モデルの更新］ボタンをクリックすると、衝突検知用モデルが更新されます。	-	-
(e)	［OK］ボタン		設定内容を確定し、衝突検知用モデルの設定画面を閉じます。	-	-
(f)	［キャンセル］ボタン		設定内容をキャンセルし、衝突検知用モデルの設定画面を閉じます。	-	-

衝突検知用モデルの表示

3D 形状データの衝突検知は、処理の負荷を軽減させるため、3D 形状データを簡略化して行います。このため、3D 表示画面上では衝突しているのに衝突していないと判定されること、逆に、衝突していないのに衝突していると判定されることがあります。簡略化された形状を確認する場合は、対象の 3D

形状データの設定画面を開き、［詳細設定 - 衝突検知用モデル］の［表示］にチェックを入れてください。チェックを入れると、3D 表示画面上に簡略化した衝突検知用モデルが表示されます。





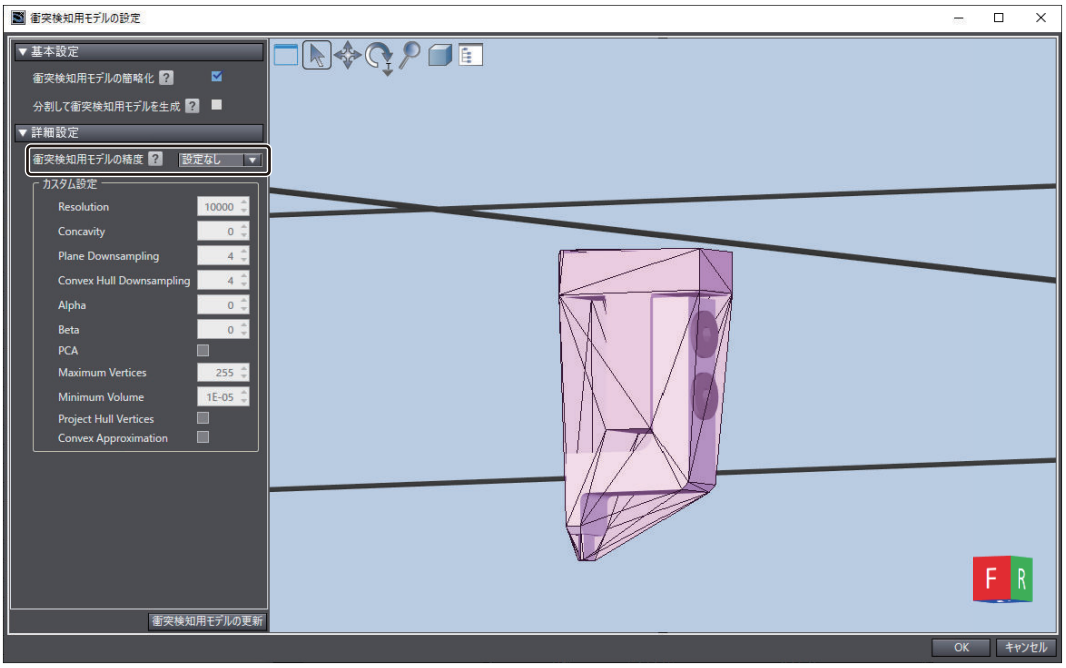
衝突検知用モデルの精度調整

表示されている 3D 形状データに、より近い衝突検知用モデルを使って衝突検知を行いたい場合、衝突検知用モデルの設定画面の［分割して衝突検知用モデルを生成］にチェックを入れてください。チェックを入れると、3D 形状データを解析し、内部的に複数のパーツに分割します。そして、分割したパーツごとに衝突検知用モデルを生成します。

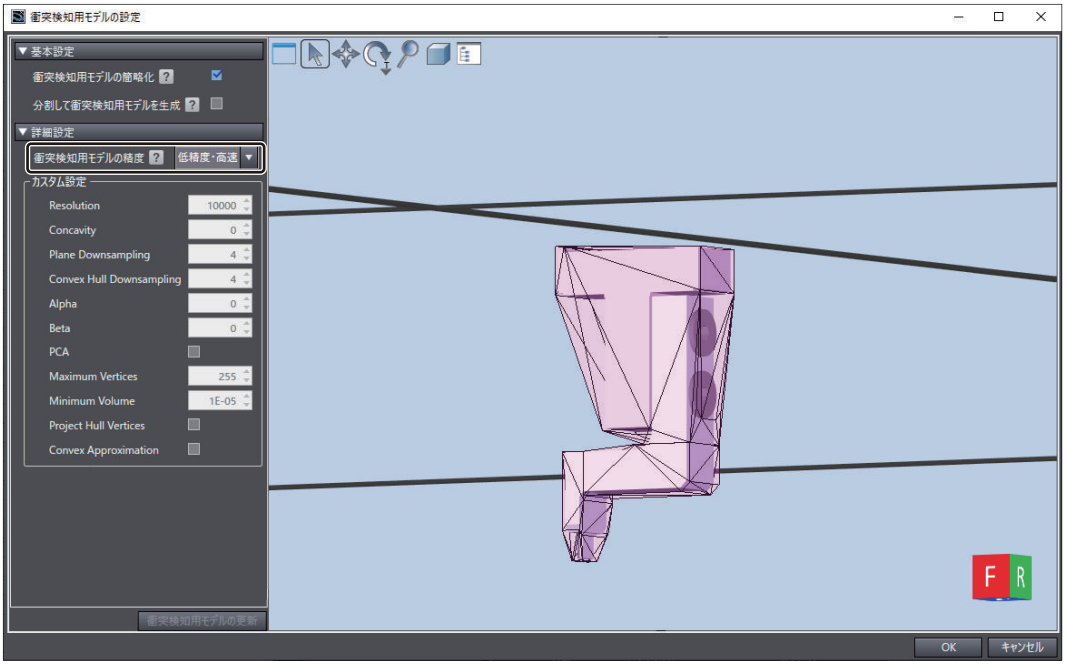


ただし、下図のように 3D 形状データを単一または数少ない部品で構成している場合、［分割して衝突検知用モデルを生成］にチェックを入れても効果はあまり期待できません。より精度の高い衝突検知を行いたい場合は、衝突検知用モデルの設定画面から衝突検知用モデルの精度を変更してください。

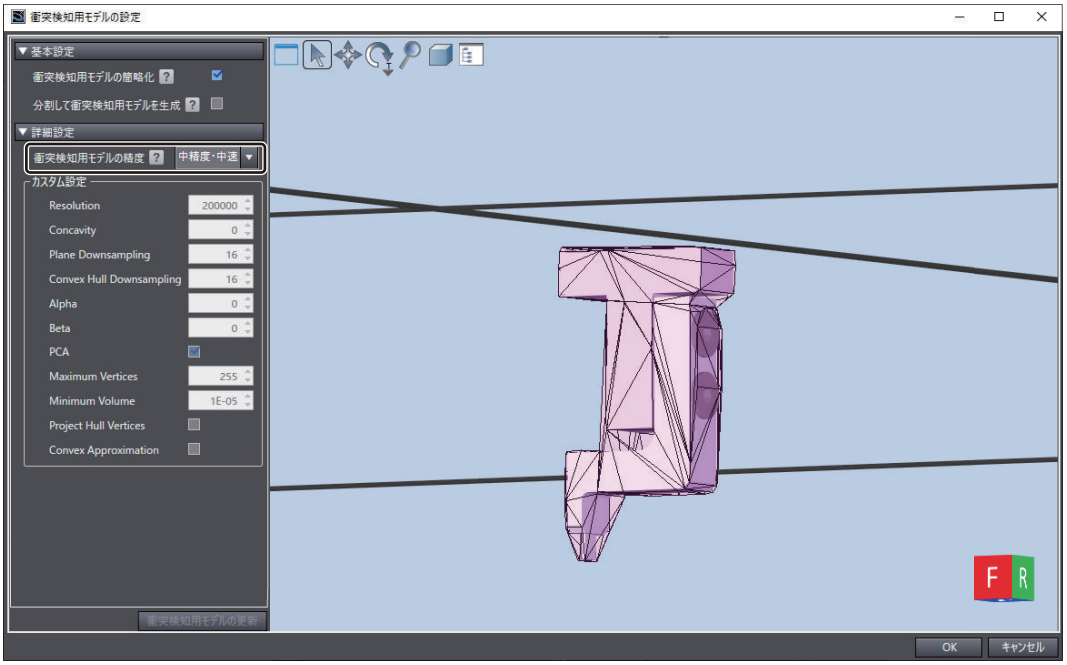
● 衝突検知用モデルの精度「設定なし」



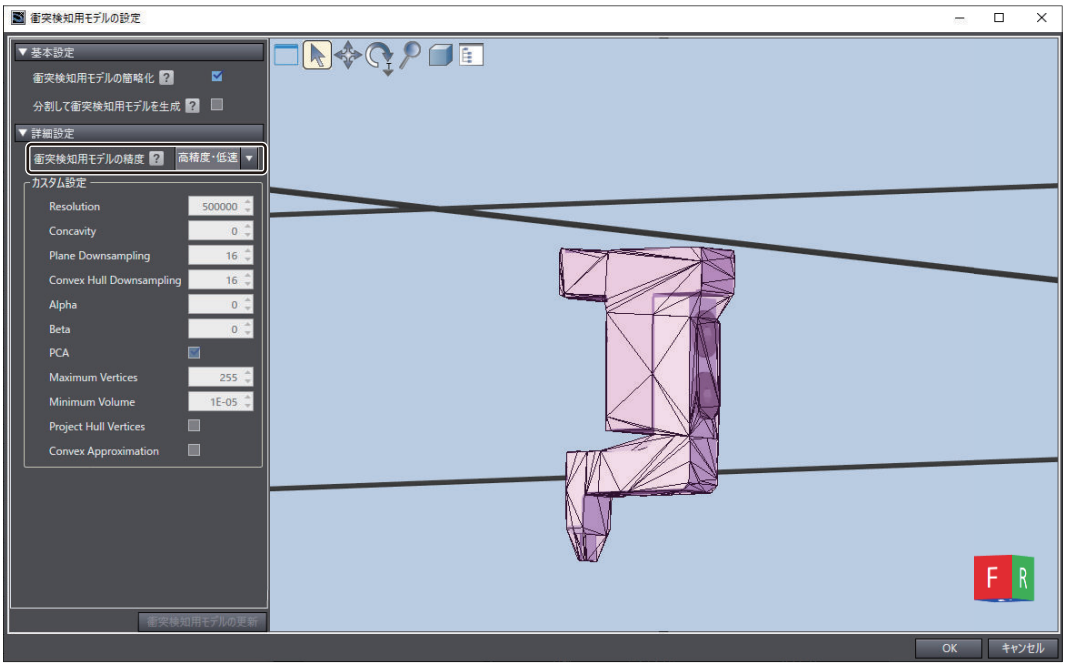
● 衝突検知用モデルの精度「低精度・高速」



● **衝突検知用モデルの精度「中精度・中速」**



● **衝突検知用モデルの精度「高精度・低速」**



9

ロボット統合システムの 3D シミュレーション

本章では、アプリケーションコントローラによるロボットシステムの 3D シミュレーションで使用する機能と使用方法について説明します。

9-1　アプリケーションコントローラによるロボットシステムの 3D シミュレーションの概要 9-2
9-1-1　3D シミュレーションの種類 9-2
9-1-2　IEC61131-3 言語で制御するメカニカル機構の 3D シミュレーション 9-2
9-1-3　ロボットアプリケーションの 3D シミュレーション 9-2

9-2　ロボットシステムの 3D シミュレーション 9-4
9-2-1　ロボットで操作したワークを周辺機器で操作する 9-4
9-2-2　周辺機器で操作したワークをロボットで操作する 9-5
9-2-3　ロボットだけで操作するワークの衝突検知を行う 9-6

9-1 アプリケーションコントローラによるロボットシステムの 3D シミュレーションの概要

ここでは、ロボットだけでなく周辺機器（メカニカル機構など）も制御するアプリケーションコントローラによるロボットシステムの 3D シミュレーションの概要について説明します。

9-1-1 3D シミュレーションの種類

3D シミュレーションには、以下の 2 種類があります。

- IEC61131-3 言語で制御するメカニカル機構の 3D シミュレーション
  「第 6 章 3D 形状データを動作させる設定およびスクリプトの作成 (P.6-1)」で説明したとおり、シェイプスクリプトを作成し、シェイプスクリプトシーケンスにより、ワーク、パレットを作成し、操作します。操作する、ワーク、パレットの形状は、シェイプスクリプトから 3D 形状データを指定します。

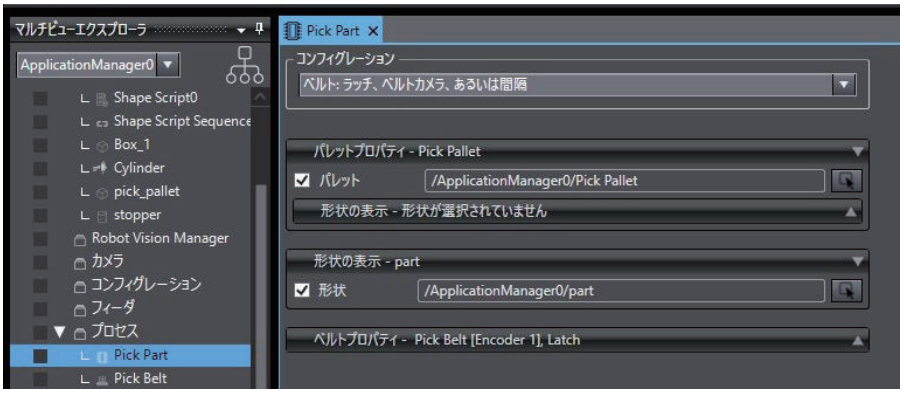
- ロボットアプリケーションの 3D シミュレーション
  アプリケーションマネージャのプロセスマネージャによるエミュレーション機能で、ロボットを動作、およびワーク、パレットを作成、操作します。操作するワーク、パレットの形状は、プロセスマネージャのパラメータから 3D 形状データを指定します。

9-1-2 IEC61131-3 言語で制御するメカニカル機構の 3D シミュレーション

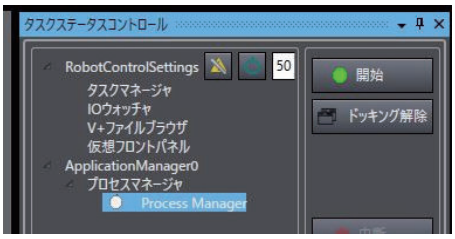
3D シミュレーションで検証したい範囲が、IEC61131-3 言語で制御するメカ機構またはロボットによるワークの操作の場合、シェイプスクリプトおよびシェイプスクリプトシーケンスにより実行します。
アプリケーションマネージャの［3D 表示］のコンテキストメニューから［追加］-［シェイプスクリプト］、および［追加］-［シェイプスクリプトシーケンス］により追加します。追加したシェイプスクリプトでプログラミングを行い、そのシェイプスクリプトをシェイプスクリプトシーケンスに登録して、シェイプスクリプトシーケンスを実行します。
操作方法の詳細は「第 6 章 3D 形状データを動作させる設定およびスクリプトの作成 (P.6-1)」を参照してください。

9-1-3 ロボットアプリケーションの 3D シミュレーション

3D シミュレーションで検証したい範囲が、アプリケーションマネージャでのロボットによるワークのピック＆プレース操作の場合、プロセスマネージャのエミュレーション機能により実行します。
アプリケーションマネージャの［3D 表示］に追加したワークやパレットの 3D 形状データを使用したい場合、アプリケーションマネージャの［プロセス］の［ピックワーク］、および［プレースワークターゲット］の［パレットプロパティ］と［形状の表示］に、それぞれ形状データを登録します。



3D シミュレーションは、メインメニューの［表示］-［タスクステータスコントロール］画面から、対象のプロセスマネージャを選択し、［開始］ボタンをクリックして開始します。



9-2 ロボットシステムの 3D シミュレーション

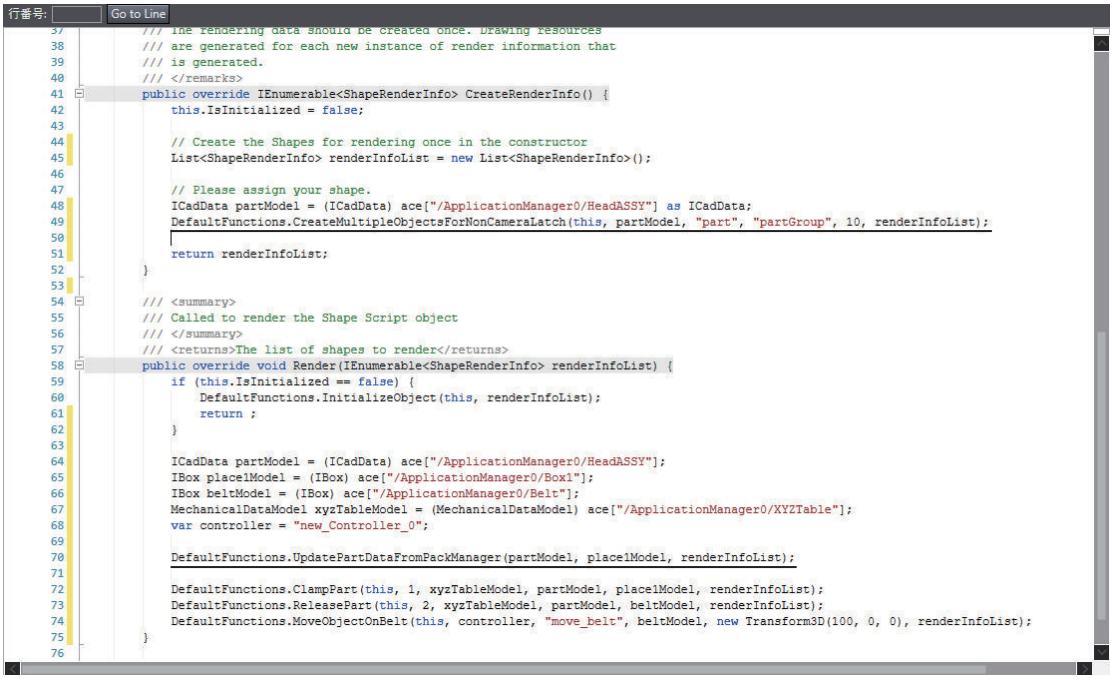
シェイプスクリプトおよびシェイプスクリプトシーケンスと、プロセスマネージャのエミュレーション機能を使用することにより、ロボットと、周辺機器（メカニカル機構など）の、ワークを含めた 3D シミュレーションが実行できます。

9-2-1 ロボットで操作したワークを周辺機器で操作する

3D シミュレーションで検証したい範囲が、アプリケーションマネージャによるロボットでワークをピック＆プレースし、そのワークを IEC61131-3 言語で制御する周辺機器（メカニカル機構など）で操作する場合や、ロボットでピックしたワークをベルトコンベアなどの上に置き、コンベア上を移動させたい場合は、以下のように行います。

- シェイプスクリプトにメソッドを追加して、プロセスマネージャによるエミュレーション機能で生成するワークを、あらかじめ必要な数だけ作成します。CreateRenderInfo メソッド内で、CreateMultipleObjectsForNonCameraLatch メソッドを使用して、プロセスマネージャで使用するワークを生成しています。
- プロセスマネージャで操作が終わったワークを、シェイプスクリプトから操作できるようにするために、「ワークの引き渡し処理」のメソッドをシェイプスクリプトに追加します。Render メソッド内では、UpdatePartDataFromPackManager メソッドを使用して、プロセスマネージャで操作の終わったワークをシェイプスクリプトから操作できるようにしています。

シェイプスクリプトには、以下のようにプログラミングします。




```
        /// The rendering data should be created once. Drawing resources
        /// are generated for each new instance of render information that
        /// is generated.
        /// </remarks>
        public override IEnumerable<ShapeRenderInfo> CreateRenderInfo() {
            this.IsInitialized = false;

            // Create the Shapes for rendering once in the constructor
            List<ShapeRenderInfo> renderInfoList = new List<ShapeRenderInfo>();

            // Please assign your shape.
            ICadData partModel = (ICadData) ace["/ApplicationManager0/HeadASSY"] as ICadData;
            DefaultFunctions.CreateMultipleObjectsForNonCameraLatch(this, partModel, "part", "partGroup", 10, renderInfoList);

            return renderInfoList;
        }

        /// <summary>
        /// Called to render the Shape Script object
        /// </summary>
        /// <returns>The list of shapes to render</returns>
        public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
            if (this.IsInitialized == false) {
                DefaultFunctions.InitializeObject(this, renderInfoList);
                return ;
            }

            ICadData partModel = (ICadData) ace["/ApplicationManager0/HeadASSY"];
            IBox place1Model = (IBox) ace["/ApplicationManager0/Box1"];
            IBox beltModel = (IBox) ace["/ApplicationManager0/Belt"];
            MechanicalDataModel xyzTableModel = (MechanicalDataModel) ace["/ApplicationManager0/XYZTable"];
            var controller = "new_Controller_0";

            DefaultFunctions.UpdatePartDataFromPackManager(partModel, place1Model, renderInfoList);

            DefaultFunctions.ClampPart(this, 1, xyzTableModel, partModel, place1Model, renderInfoList);
            DefaultFunctions.ReleasePart(this, 2, xyzTableModel, partModel, beltModel, renderInfoList);
            DefaultFunctions.MoveObjectOnBelt(this, controller, "move_belt", beltModel, new Transform3D(100, 0, 0), renderInfoList);
        }
```

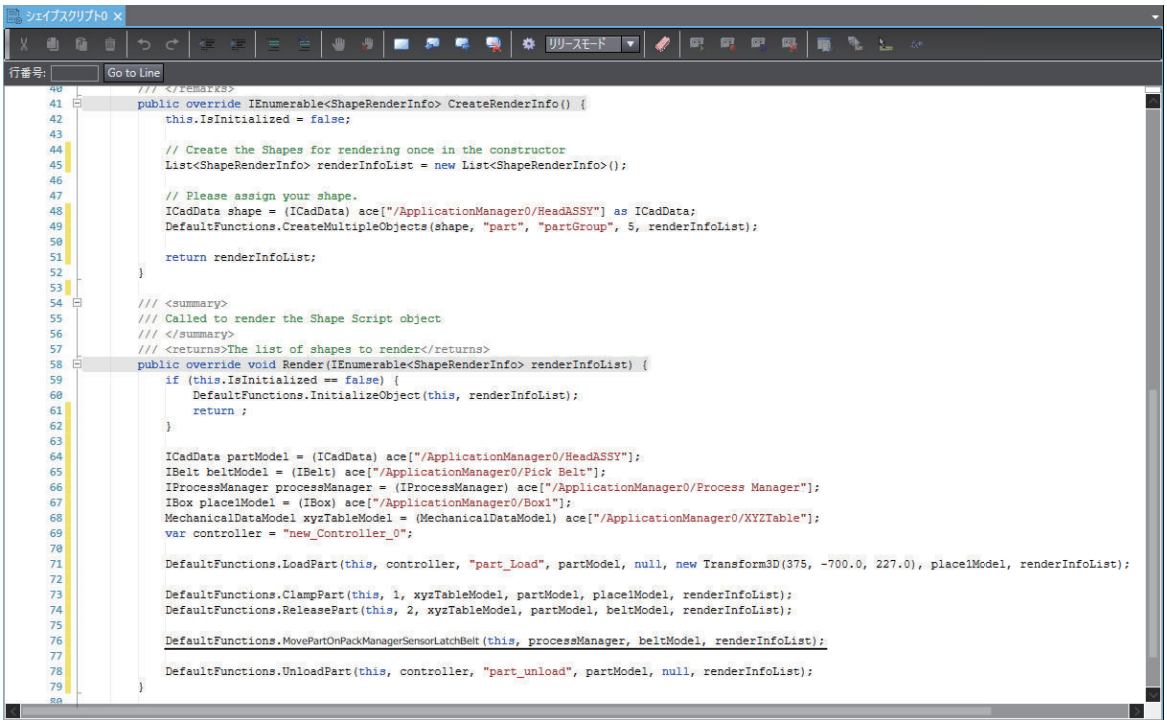
メソッドの詳細については、「A-1-1 関数一覧 (P.A-2)」を参照してください。

9-2-2　周辺機器で操作したワークをロボットで操作する

3D シミュレーションで検証したい範囲が、IEC61131-3 言語で制御する周辺機器（メカニカル機構など）でワークを操作し、プロセスマネージャによりコンベアベルトに置き、アプリケーションマネージャによるピック＆プレースを行いたい場合は、以下のように行います。

- シェイプスクリプトで作成、操作した「ワークの引き渡し処理」を、プロセスマネージャに対して行うためのメソッドを、シェイプスクリプトに追加します。Render メソッド内で、ClampPart、ReleasePart メソッドによりベルト上に置いたワークを、MovePartOnPackManagerSensorLatchBelt メソッドを使用し、ラッチ位置まで移動させて、プロセスマネージャで操作できるようにします。

シェイプスクリプトには、以下のようにプログラミングします。




```
        /// </remarks>
        public override IEnumerable<ShapeRenderInfo> CreateRenderInfo() {
            this.IsInitialized = false;

            // Create the Shapes for rendering once in the constructor
            List<ShapeRenderInfo> renderInfoList = new List<ShapeRenderInfo>();

            // Please assign your shape.
            ICadData shape = (ICadData) ace["/ApplicationManager0/HeadASSY"] as ICadData;
            DefaultFunctions.CreateMultipleObjects(shape, "part", "partGroup", 5, renderInfoList);

            return renderInfoList;
        }

        /// <summary>
        /// Called to render the Shape Script object
        /// </summary>
        /// <returns>The list of shapes to render</returns>
        public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
            if (this.IsInitialized == false) {
                DefaultFunctions.InitializeObject(this, renderInfoList);
                return ;
            }

            ICadData partModel = (ICadData) ace["/ApplicationManager0/HeadASSY"];
            IBelt beltModel = (IBelt) ace["/ApplicationManager0/Pick Belt"];
            IProcessManager processManager = (IProcessManager) ace["/ApplicationManager0/Process Manager"];
            IBox placeModel = (IBox) ace["/ApplicationManager0/Box1"];
            MechanicalDataModel xyzTableModel = (MechanicalDataModel) ace["/ApplicationManager0/XYZTable"];
            var controller = "new_Controller_0";

            DefaultFunctions.LoadPart(this, controller, "part_Load", partModel, null, new Transform3D(375, -700.0, 227.0), placeModel, renderInfoList);

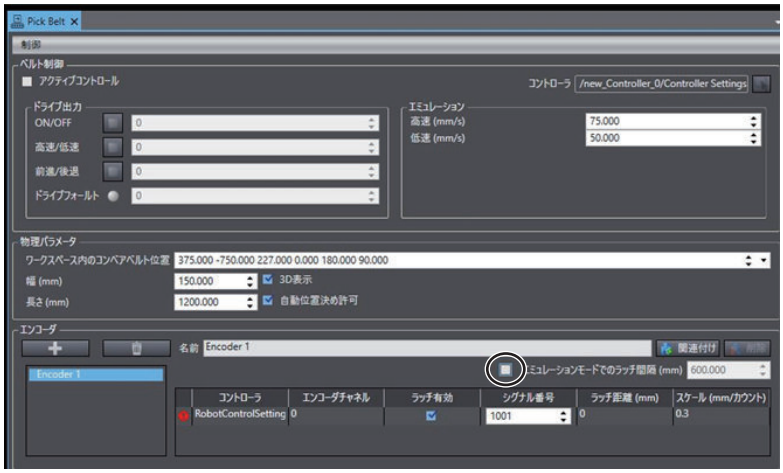
            DefaultFunctions.ClampPart(this, 1, xyzTableModel, partModel, placeModel, renderInfoList);
            DefaultFunctions.ReleasePart(this, 2, xyzTableModel, partModel, beltModel, renderInfoList);

            DefaultFunctions.MovePartOnPackManagerSensorLatchBelt(this, processManager, beltModel, renderInfoList);

            DefaultFunctions.UnloadPart(this, controller, "part_unload", partModel, null, renderInfoList);
        }
```

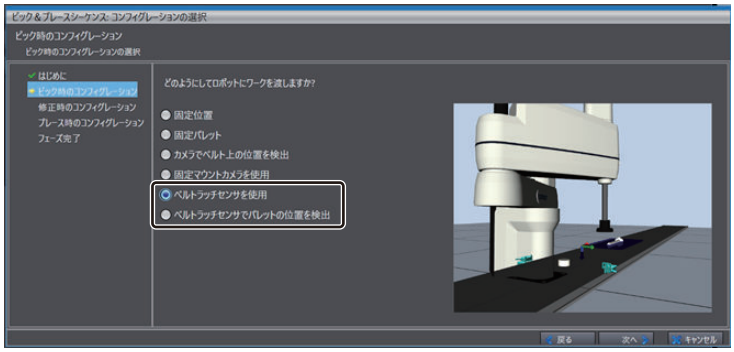
メソッドの詳細については、「A-1-1 関数一覧 (P.A-2)」を参照してください。

また、プロセスマネージャによるワークの生成を停止させるために、プロセスマネージャのベルトの設定にある、［エミュレーションモードでのラッチ間隔(mm)］のチェックを外してください。



使用上の注意

メインメニューの［挿入］-［アプリケーションサンプル］-［Pack Manager サンプル］を実行して、表示される［ピック＆プレースシーケンス: コンフィグレーションの選択］で［ピック時のコンフィグレーション］が［ベルトラッチセンサを使用］または［ベルトラッチセンサでパレットの位置を検出］を選択した場合のみ、シェイプスクリプトにより作成し、操作されたワークやパレットをロボットアプリケーションで続けて操作できます。

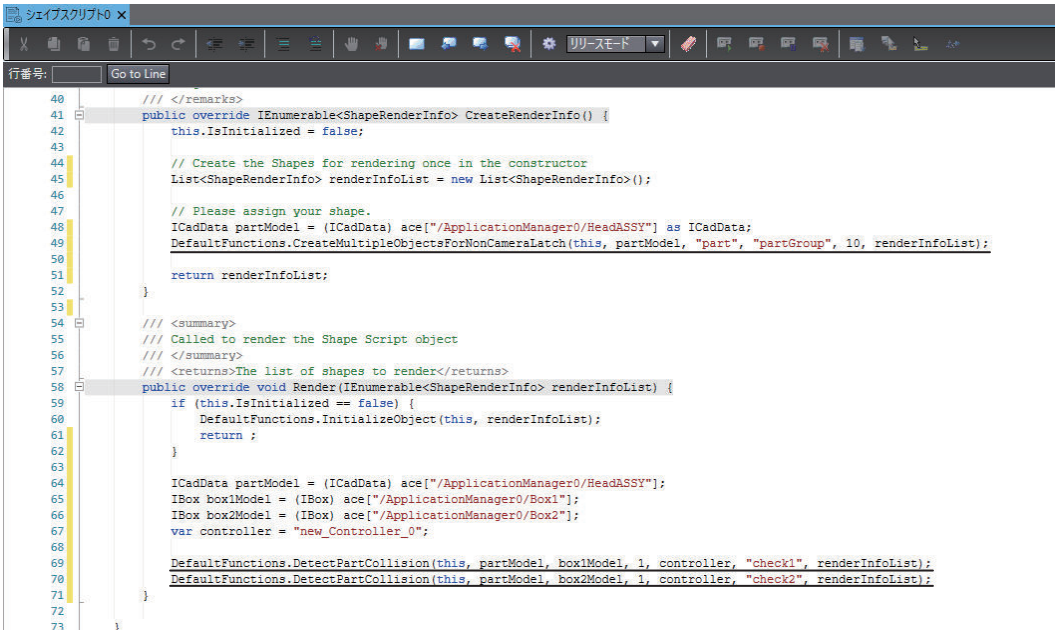


9-2-3 ロボットだけで操作するワークの衝突検知を行う

3D シミュレーションで検証したい範囲が、アプリケーションマネージャによるロボットのピック＆プレースだけで、IEC61131-3 言語で制御する周辺機器（メカニカル機構など）の 3D シミュレーションを行わない場合でも、ワークの衝突検知を行いたいときには、シェイプスクリプトを作成し、シェイプスクリプトシーケンスを実行する必要があります。

- シェイプスクリプトにメソッドを追加して、プロセスマネージャによるエミュレーション機能で生成するワークを、あらかじめ必要な数だけ作成します。CreateRenderInfo メソッド内で、CreateMultipleObjectsForNonCameraLatch メソッドを使用して、プロセスマネージャで使用するワークを生成しています。
- ワークが衝突検知を行いたい 3D 形状データに衝突したときに、トレースメッセージやコントローラの変数の値の変化で認識できるようにするためのメソッドを、シェイプスクリプトに追加します。Render メソッド内では、DetectPartCollision メソッドを使用して、プロセスマネージャで操作しているワークが対象の 3D 形状データに衝突しているかを検知できるようにしています。カメラを利用して、複数の種類のワークを扱う場合には、CreateMultipleObjectsForNonCameraLatch メソッドではなく、CreateMultipleObjectsForCameraLatch メソッドを使用します。

シェイプスクリプトには、以下のようにプログラミングします。




```
        /// </remarks>
        public override IEnumerable<ShapeRenderInfo> CreateRenderInfo() {
            this.IsInitialized = false;

            // Create the Shapes for rendering once in the constructor
            List<ShapeRenderInfo> renderInfoList = new List<ShapeRenderInfo>();

            // Please assign your shape.
            ICadData partModel = (ICadData) ace["/ApplicationManager0/HeadASSY"] as ICadData;
            DefaultFunctions.CreateMultipleObjectsForNonCameraLatch(this, partModel, "part", "partGroup", 10, renderInfoList);

            return renderInfoList;
        }

        /// <summary>
        /// Called to render the Shape Script object
        /// </summary>
        /// <returns>The list of shapes to render</returns>
        public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
            if (this.IsInitialized == false) {
                DefaultFunctions.InitializeObject(this, renderInfoList);
                return ;
            }

            ICadData partModel = (ICadData) ace["/ApplicationManager0/HeadASSY"];
            IBox box1Model = (IBox) ace["/ApplicationManager0/Box1"];
            IBox box2Model = (IBox) ace["/ApplicationManager0/Box2"];
            var controller = "new_Controller_0";

            DefaultFunctions.DetectPartCollision(this, partModel, box1Model, 1, controller, "check1", renderInfoList);
            DefaultFunctions.DetectPartCollision(this, partModel, box2Model, 1, controller, "check2", renderInfoList);
        }

    }
```

メソッドの詳細については、「A-1-1 関数一覧 (P.A-2)」を参照してください。

10

便利機能

本章では、3D シミュレーションの便利機能について説明します。

10-1 CAD データの一括置換 10-2
10-1-1 一括置換手順 10-2

10-1 CAD データの一括置換

3D 形状データの CAD データを、別の CAD データに一括して置き換えることができます。
装置の実現性確認のために作成した概略設計の CAD データを、詳細設計した CAD データに置き換えることで、作成したプログラムやスクリプトを大きく変更することなく活用できます。

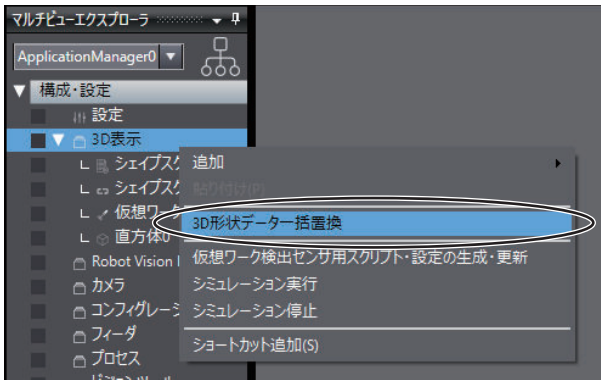
10-1-1 一括置換手順

CAD データの一括置換手順は以下のとおりです。

参考

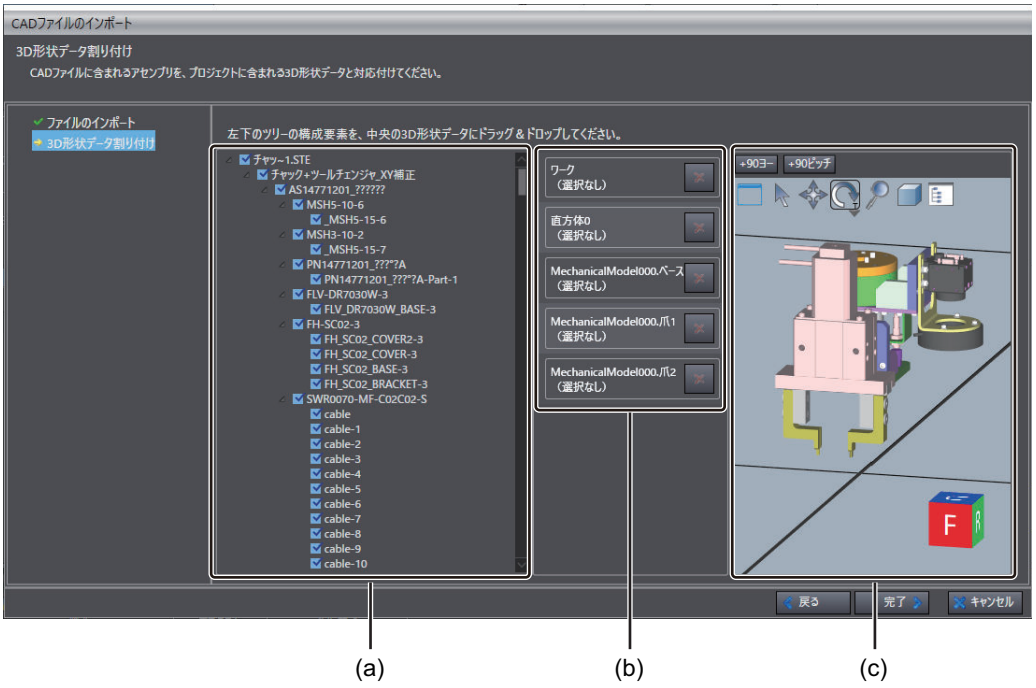
インポート時 CAD データのメッシュの粗さをオプションで設定することができます。オプション設定の詳細は『Sysmac Studio ロボット統合システム構築機能 オペレーションマニュアル ロボット統合 CPU ユニット編(SBCA-464)』を参照してください。

1 **マルチビューエクスプローラの［3D 表示］を右クリックし、［3D 形状データー括置換］を選択します。**



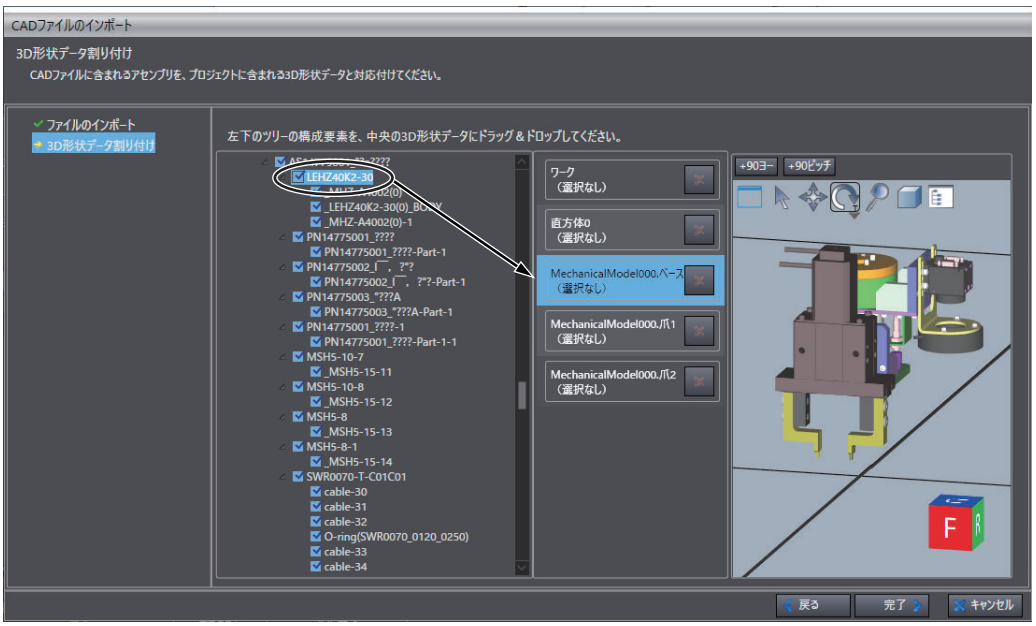
一括置換ウィザードが起動し、ファイル選択画面が表示されます。
ファイル選択画面では、CAD ファイルをひとつ選択することができます。選択可能な CAD ファイルの形式は、「4-3-1 サポートする CAD データファイルの種類 (P.4-9)」を参照してください。

2 **ファイル選択画面で置換先の CAD ファイルを選択し、［次へ］ボタンをクリックします。**
［3D 形状データ割り付け］画面が表示されます。



	項目	説明
(a)	置換先の CAD データ	選択された CAD ファイルのデータが表示されます。CAD データの部品の構成が表示されます。
(b)	置換対象の 3D 形状データ	プロジェクト上の 3D 形状データの一覧が表示されます。
(c)	置換先 CAD データの 3D 表示画面	置換先の CAD データを 3D 表示します。

3 **置換先の CAD データの部品をドラッグし、置換対象の 3D 形状データの項目の上にドロップします。**



対象の 3D 形状データに置換先の CAD データが割り付けられます。



選択を解除する場合は、割り付けられた CAD データ名の右側の［×］ボタンをクリックします。

4 **必要な置換先の CAD データの割付を行い、［完了］ボタンをクリックします。**
対象の 3D 形状データが、指定した CAD データで置き換えられます。

A

付録

本章では、シェイプスクリプトで使用する関数の説明など、本文の補足的な情報を記述します。

A-1　シェイプスクリプトで使用する関数 A-2
A-1-1　関数一覧 A-2
A-1-2　ワーク、パレットの生成、表示 A-7
A-1-3　ワーク、パレットの搬送 A-19
A-1-4　コンベア A-21
A-1-5　ワーク、パレットのピック A-23
A-1-6　ワーク、パレットのプレース A-28
A-1-7　状態の遷移、検知 A-34
A-1-8　コントローラとの変数共有 A-38
A-1-9　プロセスマネージャとの連携 A-58
A-1-10　シェイプスクリプト実行時間計測 A-62
A-1-11　シェイプスクリプトで使用する関数の更新 A-63
A-2　シミュレータと実機コントローラの差異 A-66
A-2-1　各機能の動作 A-66
A-3　シェイプスクリプトの実行時間 A-67
A-3-1　システムモニタによる実行時間の確認 A-67
A-3-2　Stopwatch 構文を使用した実行時間の計測 A-68
A-3-3　シェイプスクリプトの処理の高速化 A-69
A-4　ワーク動作設定 A-74
A-4-1　3D 形状データごとのワーク動作設定可否一覧 A-74
A-4-2　シェイプスクリプトとワーク動作設定の同時実行を実現する方法 A-76
A-4-3　ロボットハンドのツールチェンジャーを実現する方法 A-77
A-4-4　パレットを実現する方法 A-78

A-1　シェイプスクリプトで使用する関数

ワークの動作を定義するシェイプスクリプトには、ワークに対する基本的な操作や処理が関数として用意されています。
関数の入力方法については、「6-5-2 シェイプスクリプトのプログラミング (P.6-21)」を参照してください。

使用上の注意

シェイプスクリプトで使用できる関数の定義は、シェイプスクリプトに記載されています。シェイプスクリプトを新規作成したときに、シェイプスクリプト内にすべての関数の定義が記載されます。シェイプスクリプトを編集する際は以下の点に注意してください。

- シェイプスクリプト内で使用している関数の定義を削除したり、定義の内容を変更しないでください。関数が動作しなくなります。誤って定義を削除したり変更した場合は、あらためて新規作成したシェイプスクリプトに、ワークの動作を記述した行をコピーして貼り付けてください。
- Sysmac Studio のバージョンアップにより新しく追加された関数や機能が追加された関数を既存のシェイプスクリプトで使用する場合は、新しく追加された関数の定義が記載されたシェイプスクリプトを使用する必要があります。バージョンアップした Sysmac Studio でシェイプスクリプトを新規作成し、そのシェイプスクリプトに既存のシェイプスクリプトからワークの動作を記述した行をコピーして貼り付けてください。
- シェイクスクリプトファンクションのデフォルトファンクションを使用している場合は、デフォルトファンクションの更新機能を使って関数を更新することができます。手順については「A-1-11 シェイプスクリプトで使用する関数の更新 (P.A-63)」を参照してください。

A-1-1　関数一覧

シェイプスクリプトで使用できる関数は以下のとおりです。

カテゴリ	関数名	説明	参照先
ワーク、パレットの生成、表示	CreateObject	ひとつのワーク／パレットを生成します。	「CreateObject (P.A-7)」
	CreateMultipleObjects	複数のワーク／パレットを生成します。	「CreateMultipleObjects (P.A-8)」
	CreateMultipleObjectsForCameraLatch	プロセスマネージャのエミュレーション機能と連携するワークまたはパレットを複数生成します。	「CreateMultipleObjects ForCameraLatch (P.A-8)」
	CreateMultipleObjectsForNonCameraLatch	プロセスマネージャのエミュレーション機能で使用するワークまたはパレットを複数生成します。	「CreateMultipleObjects ForNonCameraLatch (P.A-11)」
	LoadPart	ワークを表示します。	「LoadPart (P.A-12)」
	LoadPallet	パレットを表示します。	「LoadPallet (P.A-14)」
	LoadPartOnPallet	ワークをパレットの上に表示します。	「LoadPartOnPallet (P.A-15)」
	UnloadPart	コントローラの BOOL 型変数の入力により、ワークを消去します。	「UnloadPart (P.A-16)」
	UnloadPallet	コントローラの BOOL 型変数の入力により、パレットを消去します。	「UnloadPallet (P.A-17)」
	UnloadCollidingPart	指定 3D 形状データと衝突時にワークを消去します。	「UnloadCollidingPart (P.A-17)」
	UnloadCollidingPallet	指定 3D データと衝突時にパレットを消去します。	「UnloadCollidingPallet (P.A-18)」
ワーク、パレットの搬送	PushPart	ワークをシリンダで押し出します。	「PushPart (P.A-19)」
	PushPallet	パレットをシリンダで押し出します。	「PushPallet (P.A-19)」
コンベア	MoveObjectOnBelt	コンベア上のワーク／パレットを移動させます。	「MoveObjectOnBelt (P.A-21)」
	InitializeMechanicalConveyor	メカニカル機構「コンベア」を初期化します。	「InitializeMechanicalConveyor (P.A-22)」
	MoveObjectOnMechanicalConveyor	メカニカル機構「コンベア」上のワーク／パレットを移動させます。	「MoveObjectOnMechanicalConveyor (P.A-22)」
ワーク、パレットのピック	ClampPart	ワークをピックします。	「ClampPart (P.A-23)」
	ClampPallet	パレットをピックします。	「ClampPallet (P.A-24)」
	ClampPartOnPallet	パレット上のワークをピックします。	「ClampPartOnPallet (P.A-24)」
	ClampPartBySignal	ワークを把持信号によりピックします。	「ClampPartBySignal (P.A-25)」
	ClampPalletBySignal	パレットを把持信号によりピックします。	「ClampPalletBySignal (P.A-26)」
	ClampPartOnPalletBySignal	パレット上のワークを把持信号によりピックします。	「ClampPartOnPalletBySignal (P.A-27)」

カテゴリ	関数名	説明	参照先
ワーク、パレットのプレース	ReleasePart	ピックしているワークをプレースします。	「ReleasePart (P.A-28)」
	ReleasePallet	ピックしているパレットをプレースします。	「ReleasePallet (P.A-29)」
	ReleasePartOnPallet	ピックしているワークをパレットにプレースします。	「ReleasePartOnPallet (P.A-30)」
	ReleasePartBySignal	ワークをロボットハンドの把持信号により放します（置きます）。	「ReleasePartBySignal (P.A-31)」
	ReleasePalletBySignal	パレットをロボットハンドの把持信号により放します（置きます）。	「ReleasePalletBySignal (P.A-32)」
	ReleasePartOnPalletBySignal	ワークをロボットハンドの把持信号によりパレットに放します（置きます）。	「ReleasePartOnPalletBySignal (P.A-33)」
状態の遷移、検知	SetNextStep	stepId の値を変更します。	「SetNextStep (P.A-34)」
	DetectPartCollision	ワーク、パレットがメカニカル機構に衝突したことを検知します。	「DetectPartCollision (P.A-34)」
	DetectMechanicalComponetsCollision	メカニカル機構がほかのメカニカル機構に衝突したことを検知します。	「DetectMechanicalComponentsCollision (P.A-35)」
	SetClampStatus	ロボットハンドの把持（衝突）状態をコントローラの変数に書き込みます。	「SetClampStatus (P.A-36)」
	GetCollidingSourceNames	指定した 3D 形状データと衝突している、すべての 3D 形状データの名称を取得します。	「GetCollidingSourceNames (P.A-37)」
コントローラとの変数共有	GetBoolVariable	指定したコントローラから BOOL 変数の値を読み出します。	「GetBoolVariable (P.A-38)」
	SetBoolVariable	指定したコントローラの BOOL 変数に値を書き込みます。	「SetBoolVariable (P.A-38)」
	GetIntegerVariable	指定したコントローラから DINT 変数の値を読み出します。	「GetIntegerVariable (P.A-39)」
	SetIntegerVariable	指定したコントローラの DINT 変数に値を書き込みます。	「SetIntegerVariable (P.A-39)」
	Get**Variable	指定したコントローラから整数型変数の値を読み出します。関数名の“**”にはコントローラの変数のデータ型を指定します。	「Get**Variable (P.A-39)」
	Set**Variable	指定したコントローラの整数型変数に値を書き込みます。関数名の“**”にはコントローラの変数のデータ型を指定します。	「Set**Variable (P.A-40)」
	GetByteArrayVariable	指定したコントローラから BYTE 型配列変数の値を読み出します。	「GetByteArrayVariable (P.A-41)」

カテゴリ	関数名	説明	参照先
	SetByteArrayVariable	指定したコントローラのBYTE 型配列変数に値を書き込みます。	「SetByteArrayVariable (P.A-42)」
	GetBoolArrayVariable	指定したコントローラからBOOL 型配列変数の値を読み出します。	「GetBoolArrayVariable (P.A-42)」
	SetBoolArrayVariable	指定したコントローラのBOOL 型配列変数に値を書き込みます。	「SetBoolArrayVariable (P.A-43)」
	GetIntegerArrayVariable	指定したコントローラからDINT 型配列変数の値を読み出します。	「GetIntegerArrayVariable (P.A-43)」
	SetIntegerArrayVariable	指定したコントローラのDINT 型配列変数に値を書き込みます。	「SetIntegerArrayVariable (P.A-44)」
	GetWordArrayVariable	指定したコントローラからWORD 型配列変数の値を読み出します。	「GetWordArrayVariable (P.A-45)」
	SetWordArrayVariable	指定したコントローラのWORD 型配列変数に値を書き込みます。	「SetWordArrayVariable (P.A-45)」
	GetDwordArrayVariable	指定したコントローラからDWORD 型配列変数の値を読み出します。	「GetDwordArrayVariable (P.A-46)」
	SetDwordArrayVariable	指定したコントローラのDWORD 型配列変数に値を書き込みます。	「SetDwordArrayVariable (P.A-46)」
	GetLwordArrayVariable	指定したコントローラからLWORD 型配列変数の値を読み出します。	「GetLwordArrayVariable (P.A-47)」
	SetLwordArrayVariable	指定したコントローラのLWORD 型配列変数に値を書き込みます。	「SetLwordArrayVariable (P.A-47)」
	GetRealArrayVariable	指定したコントローラからREAL 型配列変数の値を読み出します。	「GetRealArrayVariable (P.A-48)」
	SetRealArrayVariable	指定したコントローラのREAL 型配列変数に値を書き込みます。	「SetRealArrayVariable (P.A-48)」
	GetLrealArrayVariable	指定したコントローラからLREAL 型配列変数の値を読み出します。	「GetLrealArrayVariable (P.A-49)」
	SetLrealArrayVariable	指定したコントローラのLREAL 型配列変数に値を書き込みます。	「SetLrealArrayVariable (P.A-49)」
	GetStringArrayVariable	指定したコントローラからSTRING 型配列変数の値を読み出します。	「GetStringArrayVariable (P.A-50)」
	SetStringArrayVariable	指定したコントローラのSTRING 型配列変数に値を書き込みます。	「SetStringArrayVariable (P.A-50)」
	Get**ArrayVariable	指定したコントローラから整数型配列変数の値を読み出します。関数名の“**”にはコントローラの変数のデータ型を指定します。	「Get**ArrayVariable (P.A-51)」

カテゴリ	関数名	説明	参照先
	Set**ArrayVariable	指定したコントローラの整数型配列変数に値を書き込みます。関数名の“**”にはコントローラの変数のデータ型を指定します。	「Set**ArrayVariable (P.A-52)」
	CreateGetVariableList	コントローラから複数の変数を読み出すときに使用する読み出し変数リストを新規作成します。	「CreateGetVariableList (P.A-53)」
	AddToGetVariableList	コントローラから読み出したい変数を、読み出し変数リストに追加します。	「AddToGetVariableList (P.A-54)」
	GetVariableValues	コントローラから複数の変数の値を一度に読み出します。	「GetVariableValues (P.A-54)」
	GetValueFromVariableValueList	GetVariableValues()関数によって読み出された変数の値を取得します。	「GetValueFromVariableValueList (P.A-55)」
	CreateSetVariableList	コントローラから複数の変数の値を書き込むときに使用する書き込み変数リストを新規作成します。	「CreateSetVariableList (P.A-56)」
	AddToSetVariableList	コントローラに書き込みたい変数と値を、書き込み変数リストに追加します。	「AddToSetVariableList (P.A-56)」
	SetVariableValues	コントローラの複数の変数に値を一度に書き込みます。	「SetVariableValues (P.A-57)」
	GetCurrentControllerTime	指定したコントローラのシミュレーションの内部時刻を読み出します。	「GetCurrentControllerTime (P.A-58)」
プロセスマネージャとの連携	MovePartOnPackManagerSensorLatchBelt	プロセスマネージャのコンベア上にあるワークを、ラッチ位置まで移動させ、プロセスマネージャのエミュレーション機能で、続けてワークを操作できるようにします。	「MovePartOnPackManagerSensorLatchBelt (P.A-58)」
	MovePalletOnPackManagerSensorLatchBelt	プロセスマネージャのコンベア上にあるパレットを、ラッチ位置まで移動させ、プロセスマネージャのエミュレーション機能で、続けてパレットを操作できるようにします。	「MovePalletOnPackManagerSensorLatchBelt (P.A-59)」
	SetPackManagerPartPosition	プロセスマネージャで操作中のワークの位置を、コントローラの変数に書き込みます。	「SetPackManagerPartPosition (P.A-60)」
	UpdatePartDataFromPackManager	プロセスマネージャでピック＆プレースされたワークを、シェイプスクリプトで続けて操作できるようにします。	「UpdatePartDataFromPackManager (P.A-61)」

カテゴリ	関数名	説明	参照先
	UpdatePartDataOnPalletFromPackManager	プロセスマネージャでピックされ、パレット上にプレースされたワークをシェイプスクリプトで続けて操作できるようにします。	「UpdatePartDataOnPalletFromPackManager (P.A-61)」
シェイプスクリプト実行時間計測	StartStopwatch	シェイプスクリプトの実行時間計測を開始します。	「StartStopwatch (P.A-62)」
	StopStopwatch	シェイプスクリプトの実行時間計測を終了します。	「StopStopwatch (P.A-62)」

A-1-2　ワーク、パレットの生成、表示

CreateObject

ひとつのワーク、またはパレットを生成します。

● 関数呼び出し

関数	CreateObject(IShapeBase objectModel, string name, string collisionGroup, List<ShapeRenderInfo> renderInfoList)
使用箇所	CreateRenderInfo()関数内

● 引数

引数	説明	記述例
IShapeBase objectModel	この関数の前にあらかじめ CAD データ形状のモデル（円筒の場合 ICylinder）とプロジェクト上に登録された 3D 形状データ（デバイス名と 3D 形状データ名）を代入した変数を準備しておきます。	ICylinder objectModel = (ICylinder) ace["/ApplicationManager0/ワーク"];
string name	Render()関数内で使用する際のインスタンス名を文字列で指定します。	“Part”
string collisionGroup	登録される衝突検知グループの名前を文字列で指定します。	“PartGroup”
List<ShapeRenderInfo> renderInfoList	上記 3D 形状データ名、インスタンス名、衝突検知グループ名から構成されるリスト型構造体です。ワーク／パレットインスタンス一覧と呼びます。	-

● 説明

「objectModel」に指定している 3D 形状データを「name」の名前でインスタンス化し、ワークとして扱えるようにします。
本関数でインスタンス化したワークを Render()関数内で使用する場合、該当するワークの 3D 形状データ名とワーク／パレットインスタンス一覧の名前を指定する必要があります。関数によってはインスタンス名も指定する必要があります。

レ　バージョン関連情報

Sysmac Studio Ver.1.42 以降で新規作成したシェイプスクリプトに含まれる CreateObject()関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

CreateMultipleObjects

複数のワーク、またはパレットを生成します。

● 関数呼び出し

関数	CreateMultipleObjects(IShapeBase objectModel, string name, string collisionGroup, int count, List<ShapeRenderInfo> renderInfoList)
使用箇所	CreateRenderInfo()関数内

● 引数

引数	説明	記述例
IShapeBase objectModel	この関数の前にあらかじめ CAD データ形状のモデル（直方体の場合 IBox）とプロジェクト上に登録された 3D 形状データ（デバイス名と 3D 形状データ名）を代入した変数を準備しておきます。	ICylinder objectModel = (ICylinder) ace["/ApplicationManager0/ワーク"];
string name	Render()関数内で使用する際のインスタンス名を文字列で指定します。	“Part”
string collisionGroup	登録される衝突検知グループの名前を文字列で指定します。	“PartGroup”
int count	生成するワークまたはパレットの数を指定します。	-
List<ShapeRenderInfo> renderInfoList	上記 3D 形状データ名、インスタンス名、衝突検知グループ名から構成されるリスト型構造体です。ワーク／パレットインスタンス一覧と呼びます。	-

● 説明

「objectModel」に指定している 3D 形状データを「name」の名前でインスタンス化し、複数のワークとして扱えるようにします。
本関数でインスタンス化したワークを Render()関数内で使用する場合、該当するワークの 3D 形状データ名とワーク／パレットインスタンス一覧の名前を指定する必要があります。関数によってはインスタンス名も指定する必要があります。

ワーク／パレットインスタンス一覧に「count」で指定された数の分登録します。ワーク／パレットインスタンス一覧に登録される名前は、「name」で指定された文字列に“_N”（N=0,1,2, …, count-1)を加えた名前が登録されます。
例: 「name」が“Part”、N が 3 の場合、以下となります。
“Part_0”
“Part_1”
“Part_2”

CreateMultipleObjectsForCameraLatch

プロセスマネージャのエミュレーション機能と連携するワークまたはパレットを複数生成します。プロセスマネージャのワーク機能において画像センサが使用されているときに、この関数を使ってワークを生成すると、連携が有効になります。

● 関数呼び出し

関数	CreateMultipleObjectsForCameraLatch(IExtendedShapeScript shapeScript, IShapeBase objectModel, string name, string collisionGroup, int count, List<ShapeRenderInfo> renderInfoList, string objectName, bool isDisplayNumber = false)
使用箇所	CreateRenderInfo()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IShapeBase objectModel	この関数の前にあらかじめ CAD データ形状のモデル（直方体の場合 IBox）とプロジェクト上に登録された 3D 形状データ（デバイス名と 3D 形状データ名）を代入した変数を準備しておきます。	ICylinder objectModel = (ICylinder) ace["/ApplicationManager0/ワーク"];
string name	Render()関数内で使用する際のインスタンス名を文字列で指定します。	“Part”
string collisionGroup	登録される衝突検知グループの名前を文字列で指定します。	“PartGroup”
int count	生成するワークまたはパレットの数を指定します。	10
List<ShapeRenderInfo> renderInfoList	上記 3D 形状データ名、インスタンス名、衝突検知グループ名から構成されるリスト型構造体です。ワーク／パレットインスタンス一覧と呼びます。	-
string objectName	プロセスマネージャで制御されるワークのうち、どのワークと紐付けるかを設定します。	“ワーク 0”
bool isDisplayNumber = false	プロセスマネージャで生成されるワークのインスタンス番号を表示します。番号は、ワークの生成順とは限りません。省略することも可能です。省略した場合、false が設定されます。	true

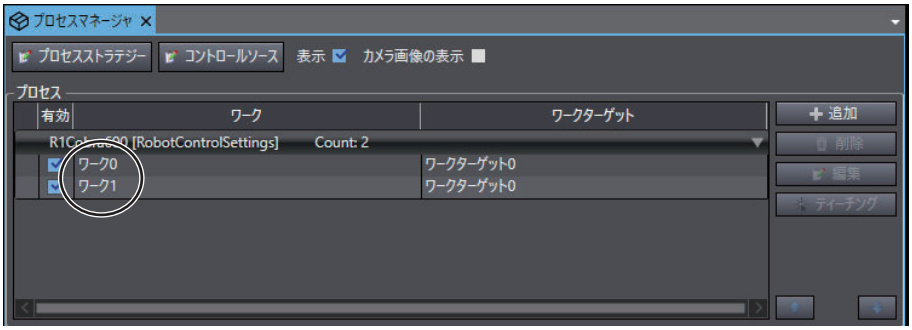
● 説明

「objectModel」に指定している 3D 形状データを「name」の名前でインスタンス化し、プロセスマネージャのワークとして、エミュレーション機能で扱えるようにします。
ワーク／パレットインスタンス一覧に「count」で指定された数の分だけ登録します。ワーク／パレットインスタンス一覧に登録される名前は、「name」で指定された文字列に“_N”(N=0,1,2, …, count-1)を加えた名前が登録されます。
例: 「name」が“Part”、N が 3 の場合、以下となります。
“Part_0”
“Part_1”
“Part_2”

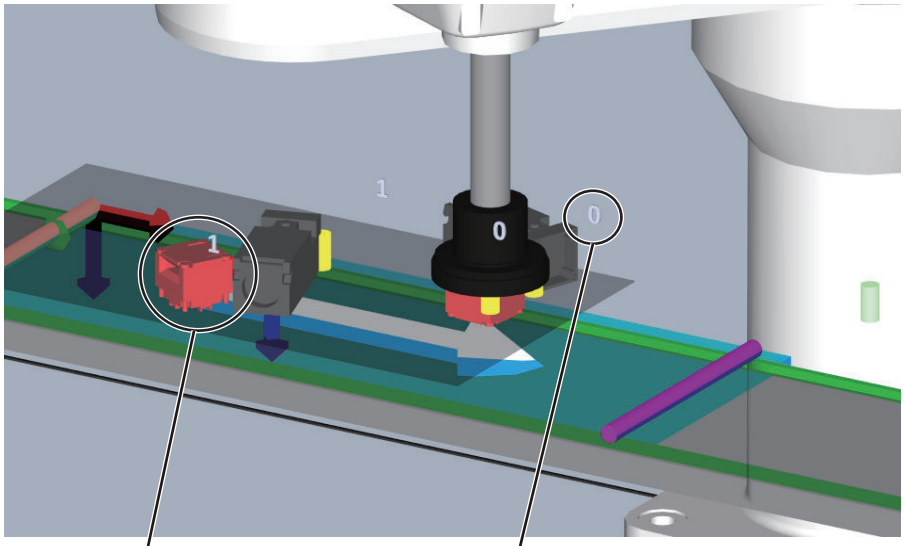
3D 形状データの異なる複数の種類のワークを生成したい場合には、それぞれの種類のワークに対して、本関数を実行します。
プロセスマネージャのワーク“ワーク 0”のコンフィグレーションに［ベルト: ラッチ、ベルトカメラ、あるいは間隔］が設定されているときの記述例を以下に示します。

```
40      public override IEnumerable<ShapeRenderInfo> CreateRenderInfo() {
41          this.IsInitialized = false;
42
43          // Create the Shapes for rendering once in the constructor
44          List<ShapeRenderInfo> renderInfoList = new List<ShapeRenderInfo>();
45
46          var caddata0 = (ICadData) ace["/ApplicationManager0/Part0"];
47          var caddata1 = (ICadData) ace["/ApplicationManager0/Part1"];
48
49          DefaultFunctions.CreateMultipleObjectsForCameraLatch(this, cadData0, "Part0", "Group1", 50, renderInfoList, "ワーク0", true);
50          DefaultFunctions.CreateMultipleObjectsForCameraLatch(this, cadData1, "Part1", "Group1", 50, renderInfoList, "ワーク1", true);
51
52          return renderInfoList;
53      }
```

この例では、2 種類のワークを表示します。まず、46～47 行目で使用するワークの 3D 形状データを定義します。次に、49～50 行目で CreateMultipleObjectsForCameraLatch()関数を使ってワークを生成しています。
引数の説明は次のとおりです。

引数	説明
caddata0	46～47 行目で定義した 3D 形状データを指定します。
“Part0”	インスタンス名を指定します。例のように複数の種類のワークを定義する場合は、重複しない名前を指定してください。名前が重複するとワークは表示されません。
“Group1”	衝突検知グループの名前を指定します。49 行目で生成するワークと 50 行目で生成するワークの間でお互いの衝突を検出したい場合は、別々の名前を指定してください。
50	生成したいワークのインスタンス数を指定します。
renderInfoList	renderInfoList を指定します。
“ワーク 0”	プロセスマネージャの設定の中に定義されているワークの名前を指定します。 
true	ワークの番号を表示するかどうかを指定します。“true”を指定すると、3D 表示画面にワークの番号が表示されます。本引数は省略可能で、省略した場合はワークの番号は表示されません。

本記述例で示したシェイプスクリプトを実行したときの 3D 表示画面の表示例を、以下に示します。生成されるインスタンスに応じて 2 種類のワークが表示されています。



使用上の注意

本関数を使用する際は、CreateMultipleObjectsForCameraLatch()関数で生成されるワークのインスタンス番号と、プロセスマネージャで生成されるワークのインスタンス番号を合わせる必要があります。インスタンス番号を合わせるために、シェイプスクリプトを実行する前に［タスクステータスコントロール］画面でプロセスマネージャのすべてのインスタンスをクリアしてください。



CreateMultipleObjectsForNonCameraLatch

プロセスマネージャのエミュレーション機能で使用するワークまたはパレットを複数生成します。カメラで複数の種類のワークを一度に認識するような場合には使用できません。

● 関数呼び出し

関数	CreateMultipleObjectsForNonCameraLatch(IExtendedShapeScript shapeScript, IShapeBase objectModel, string name, string collisionGroup, int count, List<ShapeRenderInfo> renderInfoList, bool isDisplayNumber = false)
使用箇所	CreateRenderInfo()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IShapeBase objectModel	この関数の前にあらかじめ CAD データ形状のモデル（直方体の場合 IBox）とプロジェクト上に登録された 3D 形状データ（デバイス名と 3D 形状データ名）を代入した変数を準備しておきます。	ICylinder objectModel =(ICylinder)ace["/ApplicationManager0/ワーク"];
string name	Render()関数内で使用する際のインスタンス名を文字列で指定します。	“Part”
string collisionGroup	登録される衝突検知グループの名前を文字列で指定します。	“PartGroup”
int count	生成するワークまたはパレットの数を指定します。	10
List<ShapeRenderInfo> renderInfoList	上記 3D 形状データ名、インスタンス名、衝突検知グループ名から構成されるリスト型構造体です。ワーク／パレットインスタンス一覧と呼びます。	-
bool isDisplayNumber = false	プロセスマネージャで生成されるワークのインスタンス番号を表示します。番号は、ワークの生成順とは限りません。省略することも可能です。省略した場合、false が設定されます。	true

● 説明

「objectModel」に指定している 3D 形状データを「name」の名前でインスタンス化し、複数のワークとして、アプリケーションマネージャのエミュレーション機能で扱えるようにします。
ワーク／パレットインスタンス一覧に「count」で指定された数の分だけ登録します。ワーク／パレットインスタンス一覧に登録される名前は、「name」で指定された文字列に“_N”（N=0,1,2,…, count-1)を加えた名前が登録されます。
例: 「name」が“Part”、N が 3 の場合、以下となります。
“Part_0”
“Part_1”
“Part_2”

LoadPart

ワークを表示します。

● 関数呼び出し

関数	LoadPart(IExtendedShapeScript shapeScript, string controllerName, string variableName, IShapeBase partModel, string name, Transform3D worldCoordinate, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”

引数	説明	記述例
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“LoadPart”
IShapeBase partModel	「renderInfoList」に登録されている 3D 形状データを示す変数を指定します。	“partModel”
string name	3D 形状データのインスタンス名を指定します。	“Part”
Transform3D worldCoordinate	ワークを配置する 3D 表示画面上の座標を指定します。	new Transform3D(527.211, -81.746, 135.503)
IVisualizable locationModel	この関数で表示するワークの親となる 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データ名「partModel」とインスタンス名「name」を検索し、指定したワールド座標「worldCoordinate」に配置します。
「locationModel」に指定された 3D 形状データを親とします。

● 制約事項

LoadPart()関数は、グローバル変数「variableName」が OFF から ON に変わったときにワークを表示します。引数「Name」には null または string.Empty（空文字）を設定することができますが、この場合、OFF から ON への変化を検知できるのは、「variableName」を指定した LoadPart()関数が初めて呼ばれたときだけになることがあります。

LoadPart()関数を実行してもワークが正しく表示されない例を示します。

```
59      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
60          if (this.IsInitialized == false) {
61              DefaultFunctions.InitializeObject(this, renderInfoList);
62              return ;
63          }
64
65          ICadData partModel1 = (ICadData) ace["/ApplicationManager0/CAD Data0"];
66          ICadData partModel2 = (ICadData) ace["/ApplicationManager0/CAD Data1"];
67          IShapeBase parent = (IShapeBase) ace["/ApplicationManager0/parent"];
68
69          DefaultFunctions.LoadPart("new_Controller_0", "var_LoadPart", partModel1, null, new Transform3D(0, 0, 0), parent, renderInfoList);
70          DefaultFunctions.LoadPart("new_Controller_0", "var_LoadPart", partModel2, null, new Transform3D(0, 0, 0), parent, renderInfoList);
71      }
```

69 行目が実行されたとき、変数「var_LoadPart」が OFF から ON に変わったかどうかを確認します。OFF から ON に変わっていた場合、「renderInfoList」に登録されているワークのうち、「partModel1」に対応するワークが表示されます。
この後、70 行目が実行されても、変数「var_LoadPart」は OFF から ON に切り替わり済みと判断されるため、「partModel2」に対応するワークは表示されません。

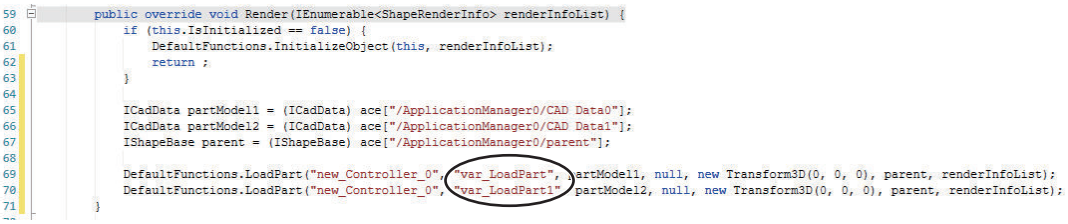
ワークを正しく表示するには、次の 2 とおりの方法があります。

(1)CreateObject()関数で引数「Name」に設定した値を LoadPart()関数の引数「Name」に設定します。以下の例では、CreateObject()関数の引数「Name」に設定された「CollisionSource1」または「CollisionSource2」を、LoadPart()関数の引数「Name」に設定しています。



```
40      public override IEnumerable<ShapeRenderInfo> CreateRenderInfo() {
41          this.IsInitialized = false;
42
43          // Create the Shapes for rendering once in the constructor
44          List<ShapeRenderInfo> renderInfoList = new List<ShapeRenderInfo>();
45
46          ICadData partModel1 = (ICadData) ace["/ApplicationManager0/CAD Data0"];
47          ICadData partModel2 = (ICadData) ace["/ApplicationManager0/CAD Data1"];
48
49          DefaultFunctions.CreateObject(partModel1, "CollisionSource1", "Group1", renderInfoList);
50          DefaultFunctions.CreateObject(partModel2, "CollisionSource2", "Group1", renderInfoList);
51
52          return renderInfoList;
53      }
54
55      /// <summary>
56      /// Called to render the Shape Script object
57      /// </summary>
58      /// <returns>The list of shapes to render</returns>
59      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
60          if (this.IsInitialized == false) {
61              DefaultFunctions.InitializeObject(this, renderInfoList);
62              return ;
63          }
64
65          ICadData partModel1 = (ICadData) ace["/ApplicationManager0/CAD Data0"];
66          ICadData partModel2 = (ICadData) ace["/ApplicationManager0/CAD Data1"];
67          IShapeBase parent = (IShapeBase) ace["/ApplicationManager0/parent"];
68
69          DefaultFunctions.LoadPart("new_Controller_0", "var_LoadPart", partModel1, "CollisionSource1", new Transform3D(0, 0, 0), parent, renderInfoList);
70          DefaultFunctions.LoadPart("new_Controller_0", "var_LoadPart", partModel2, "CollisionSource2", new Transform3D(0, 0, 0), parent, renderInfoList);
71      }
```

(2)コントローラ側で「var_LoadPart」以外の BOOL 型グローバル変数を作成し、「var_LoadPart」が OFF から ON に変わると同時に、新しく追加したグローバル変数も OFF から ON に変わるようなプログラムを作成します。70 行目に設定する引数は、新しく追加したグローバル変数を設定してください。この方法は、引数「name」に null または string.Empty（空文字）を設定してもワークを表示することができます。



```
59      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
60          if (this.IsInitialized == false) {
61              DefaultFunctions.InitializeObject(this, renderInfoList);
62              return ;
63          }
64
65          ICadData partModel1 = (ICadData) ace["/ApplicationManager0/CAD Data0"];
66          ICadData partModel2 = (ICadData) ace["/ApplicationManager0/CAD Data1"];
67          IShapeBase parent = (IShapeBase) ace["/ApplicationManager0/parent"];
68
69          DefaultFunctions.LoadPart("new_Controller_0", "var_LoadPart", partModel1, null, new Transform3D(0, 0, 0), parent, renderInfoList);
70          DefaultFunctions.LoadPart("new_Controller_0", "var_LoadPart1", partModel2, null, new Transform3D(0, 0, 0), parent, renderInfoList);
71      }
```

LoadPallet

パレットを表示します。

● 関数呼び出し

関数	LoadPallet(IExtendedShapeScript shapeScript, string controllerName, string variableName, IShapeBase palletModel, string name, Transform3D worldCoordinate, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“LoadPallet”

引数	説明	記述例
IShapeBase palletModel	「renderInfoList」に登録されている 3D 形状データを示す変数を指定します。	“palletModel”
string name	3D 形状データのインスタンス名を指定します。	“Pallet”
Transform3D worldCoordinate	パレットを配置する 3D 表示画面上の座標を指定します。	new Transform3D(527.211, -81.746, 135.503)
IVisualizable locationModel	この関数で表示するパレットの親となる 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているパレットの 3D 形状データ名「palletModel」とインスタンス名「name」を検索し、指定したワールド座標「worldCoordinate」に配置します。
「locationModel」に指定された 3D 形状データを親とします。

● 制約事項

LoadPartOnPallet()関数は、LoadPart()関数と同様に、グローバル変数「variableName」が OFF から ON に変わったときにパレットを表示します。制約事項については、「LoadPart (P.A-12)」の「制約事項」を参照してください。

LoadPartOnPallet

ワークをパレットの上に表示します。

● 関数呼び出し

関数	LoadPartOnPallet(IExtendedShapeScript shapeScript, string controllerName, string variableName, IShapeBase partModel, string partName, Transform3D worldcoordinate, IShapeBase palletModel, string palletName, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“LoadPart”

引数	説明	記述例
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	“partModel”
string partName	ワークの 3D 形状データのインスタンス名を指定します。	“Part”
Transform3D worldCoordinate	ワークを配置する 3D 表示画面上の座標を指定します。	new Transform3D(527.211, -81.746, 135.503)
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
string palletName	パレットの 3D 形状データのインスタンス名を指定します。	“Pallet”
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データ名「partModel」とインスタンス名「partName」、およびパレットの 3D 形状データ名「palletModel」とインスタンス名「palletName」を検索し、ワークを指定したワールド座標「worldCoordinate」に配置します。
3D 形状データ名「palletModel」、インスタンス名「palletName」に指定された 3D 形状データを親とします。

UnloadPart

コントローラの BOOL 型変数の入力により、ワークを消去します。

● 関数呼び出し

関数	UnloadPart(IExtendedShapeScript shapeScript, string controllerName, string variableName, IShapeBase partModel, string name, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“UnloadPart”
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
string name	ワークの 3D 形状データのインスタンス名を指定します。	“Part”
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

引数	説明	記述例
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データ名「partModel」とインスタンス名「name」を検索し、3D 表示画面から消去します。

UnloadPallet

コントローラの BOOL 型変数の入力により、パレットを消去します。

● 関数呼び出し

関数	UnloadPallet(IExtendedShapeScript shapeScript, string controllerName, string variableName, IShapeBase palletModel, string name, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“UnloadPart”
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
string name	パレットの 3D 形状データのインスタンス名を指定します。	“Pallet”
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているパレットの 3D 形状データ名「palletModel」とインスタンス名「name」を検索し、3D 表示画面から消去します。パレットの子であるワークも消去します。

UnloadCollidingPart

指定 3D 形状データと衝突時にワークを消去します

● 関数呼び出し

関数	UnloadCollidingPallet(IExtendedShapeScript shapeScript, IShapeBase palletModel, IVisualizable targetModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IVisualizable targetModel	ワークと衝突する 3D 形状データに相当する変数を指定します。	targetModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

● 説明

ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データ名「partModel」を検索し、「partModel」が「targetModel」で指定されている 3D 形状データと衝突したときに、ワーク「partModel」を消去します。

UnloadCollidingPallet

指定 3D データと衝突時にパレットを消去します。

● 関数呼び出し

関数	UnloadCollidingPallet(IExtendedShapeScript shapeScript, IShapeBase palletModel, IVisualizable targetModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IVisualizable targetModel	パレットと衝突する 3D 形状データに相当する変数を指定します。	targetModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

● 説明

ワーク／パレットインスタンス一覧の中に登録されているパレットの 3D 形状データ名「partModel」を検索し、「palletModel」が「targetModel」で指定されている 3D 形状データと衝突したときに、パレット「palletModel」を消去します。
パレットの子であるワークも消去します。

A-1-3　ワーク、パレットの搬送

PushPart

ワークをシリンダで押し出します。

● 関数呼び出し

関数	PushPart(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable pushModel, IShapeBase partModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“PushPart”
IVisualizable pushModel	この関数で表示するワークの親となる 3D 形状データに相当する変数を指定します。	pushModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IVisualizable locationModel	この関数で移動させるワークが衝突した結果、親となる 3D 形状データに相当する変数を指定します。	placeModel
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データ名「partModel」を検索し、3D 表示画面でそのワークをシリンダ「pushModel」と連動して動作させます。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

PushPallet

パレットをシリンダで押し出します。

● 関数呼び出し

関数	PushPallet(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable pushModel, IShapeBase palletModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“PushPallet”
IVisualizable pushModel	この関数で表示するワークの親となる 3D 形状データに相当する変数を指定します。	pushModel
IShapeBase palletModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	palletModel
IVisualizable locationModel	この関数で移動させるワークが衝突した結果、親となる 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データ名「palletModel」を検索し、3D 表示画面でそのパレットとパレットの上のワークをシリンダ「pushModel」と連動して動作させます。
処理の結果、パレットと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

A-1-4　コンベア

MoveObjectOnBelt

コンベア上のワーク、またはパレットを移動させます。

● 関数呼び出し

関数	MoveObjectOnBelt(IExtendedShapeScript shapeScript, string controllerName, string variableName, IVisualizable beltModel, Transform3D distancePerSecond, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“MoveBelt”
IVisualizable beltModel	コンベアの 3D 形状データに相当する変数を指定します。	beltModel
Transform3D distancePerSecond	コンベア上のワークの移動方向、移動量を 3 次元ベクトルで指定します。単位は [mm/sec]です。	Transform3D(-2, 0, 0) X 軸方向に-2[mm/sec]を意味します。
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データのうちコンベアの 3D 形状データ「beltModel」の子となっているものを「distancePerSecond」で指定された移動方向、移動量で動作させます。
「beltModel」以外に衝突している 3D 形状データは動作しません。
MoveObjectOnBelt()関数では、「variableName」で指定した変数の他に、コントローラの内部時刻を格納する DATE_AND_TIME 型のシステム変数「_CurrentTime」を取得しています。これは、ベルトの移動量を計算するために使われます。GetVariableValues()関数を使って「variableValues」を取得する場合、変数「_CurrentTime」の値も取得しておくと、シェイプスクリプトの処理が速くなります。

InitializeMechanicalConveyor

メカニカル機構「コンベア」を初期化します。

● 関数呼び出し

関数	InitializeMechanicalConveyor(IMechanicalConveyor mechanicalConveyor)
使用箇所	CreateRenderInfo()関数内、あるいは Render()関数内

● 引数

引数	説明	記述例
IMechanicalConveyor mechanicalConveyor	メカニカル機構（コンベア）のオブジェクトを指定します。	-

● 説明

「IMechanicalConveyor」の「InitializeConveyorMoveStatus」を呼び出し、内部で「IsMoveConveyor」フラグを FALSE にセットします。
なお、シェイプスクリプト実行時、一度だけ呼び出して使用します。

MoveObjectOnMechanicalConveyor

メカニカル機構「コンベア」上のワーク、またはパレットを移動させます。

● 関数呼び出し

関数	MoveObjectOnMechanicalConveyor(IExtendedShapeScript shapeScript, IMechanicalConveyor mechanicalConveyor, IEnumerable<ShapeRenderInfo>renderInfoList, IDictionary<string, object>variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IMechanicalConveyor mechanicalConveyor	メカニカル機構（コンベア）のオブジェクトを指定します。	-
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

● 説明

初回呼び出し時は、メカニカル機構のオブジェクト「IMechanicalConveyor」の「ConveyorPrevAxisValue」に、そのときのメカニカル機構に割り付けられている軸の位置「Act.Pos」を記録します。また、「IsMoveConveyor」に TRUE をセットします。

2 回目以降の呼び出し時は、メカニカル機構のオブジェクト「IMechanicalConveyor」の「ConveyorPrevAxisValue」に格納されている前回の軸の位置と、現在の軸の位置の差を計算し、その値とメカニカル機構の直動移動の方向をもとに、ワーク／パレットの表示（配置）座標を移動します。このとき、移動対象のワーク／パレットは、親が対象のメカニカル機構のものとなります。ただし、ワーク／パレットが対象のメカニカル機構以外の何かに衝突していた場合は、移動しません。

A-1-5　ワーク、パレットのピック

ClampPart

ワークをピックします。

● 関数呼び出し

関数	ClampPart(IExtendedShapeScript shapeScript, int stepId, IVisualizable robotTool , IShapeBase partModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
IVisualizable robotTool	この関数でワークをピックする 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IVisualizable locationModel	ワークが衝突する前に親となっている 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークの親が「locationModel」で指定されている 3D 形状データで、「partModel」が「robotTool」で指定されている 3D 形状データと衝突したときに、ワークの親を「robotTool」で指定されている 3D 形状データに変更します。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ClampPallet

パレットをピックします。

● 関数呼び出し

関数	ClampPallet(IExtendedShapeScript shapeScript, int stepId, IVisualizable robotTool , IShapeBase palletModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
IVisualizable robotTool	この関数でパレットをピックする 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IVisualizable locationModel	パレットが衝突する前に親となっている 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「palletModel」のパレットの親が「locationModel」で指定されている 3D 形状データで、「palletModel」が「robotTool」で指定されている 3D 形状データと衝突したときに、パレットの親を「robotTool」で指定されている 3D 形状データに変更します。
処理の結果、パレットと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ClampPartOnPallet

パレット上のワークをピックします。

● 関数呼び出し

関数	ClampPartOnPallet(IExtendedShapeScript shapeScript, int stepId, IVisualizable robotTool , IShapeBase partModel, IShapeBase palletModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
IVisualizable robotTool	この関数でワークをピックする 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IShapeBase palletModel	「renderInfoList」に登録されていて、かつワークの親となっているパレットの 3D 形状データを示す変数を指定します。	palletModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークの親が「palletModel」で指定されている 3D 形状データで、「partModel」が「robotTool」で指定されている 3D 形状データと衝突したときに、ワークの親を「robotTool」で指定されている 3D 形状データに変更します。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ClampPartBySignal

ワークを把持信号によりピックします。

● 関数呼び出し

関数	ClampPartBySignal(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable robotTool , IShapeBase partModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPart”
IVisualizable robotTool	この関数でワークをピックする 3D 形状データに相当する変数を指定します。	pickModel

引数	説明	記述例
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IVisualizable locationModel	ワークが衝突する前に親となっている 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークの親が「locationModel」で指定されている 3D 形状データで、「partModel」が「robotTool」で指定されている 3D 形状データと衝突したときに、ワークの親を「robotTool」で指定されている 3D 形状データに変更します。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ClampPalletBySignal

パレットを把持信号によりピックします。

● 関数呼び出し

関数	ClampPalletBySignal(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable robotTool , IShapeBase palletModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPallet”
IVisualizable robotTool	この関数でパレットをピックする 3D 形状データに相当する変数を指定します。	pickModel

引数	説明	記述例
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IVisualizable locationModel	パレットが衝突する前に親となっている 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「palletModel」のパレットの親が「locationModel」で指定されている 3D 形状データで、「palletModel」が「robotTool」で指定されている 3D 形状データと衝突したときに、パレットの親を「robotTool」で指定されている 3D 形状データに変更します。
処理の結果、パレットと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ClampPartOnPalletBySignal

パレット上のワークを把持信号によりピックします。

● 関数呼び出し

関数	ClampPartOnPalletBySignal(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable robotTool , IShapeBase partModel, IShapeBase palletModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPart”

引数	説明	記述例
IVisualizable robotTool	この関数でワークをピックする 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IVisualizable locationModel	ワークが衝突する前に親となっている 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークの親が「palletModel」で指定されている 3D 形状データで、「partModel」が「robotTool」で指定されている 3D 形状データと衝突したときに、ワークの親を「robotTool」で指定されている 3D 形状データに変更します。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

A-1-6　ワーク、パレットのプレース

ReleasePart

ピックしているワークをプレースします。

● 関数呼び出し

関数	ReleasePart(IExtendedShapeScript shapeScript, int stepId, IVisualizable robotTool, IShapeBase partModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-

引数	説明	記述例
int stepId	ワーク／パレット状態番号*1 を指定します。	1
IVisualizable robotTool	この関数でワークをピックしている 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IVisualizable locationModel	ワークをプレースする 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークが、「robotTool」で指定されている 3D 形状データの子となっており、「locationModel」で指定されている 3D 形状データと衝突したときに、ワークの親を「locationModel」で指定されている 3D 形状データに変更します。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ReleasePallet

ピックしているパレットをプレースします。

● 関数呼び出し

関数	ReleasePallet(IExtendedShapeScript shapeScript, int stepId, IVisualizable robotTool, IShapeBase palletModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
IVisualizable robotTool	この関数でパレットをピックしている 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IVisualizable locationModel	パレットをプレースする 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。

シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「palletModel」のパレットが、「robotTool」で指定されている 3D 形状データの子となっており、「locationModel」で指定されている 3D 形状データと衝突したときに、パレットの親を「locationModel」で指定されている 3D 形状データに変更します。
処理の結果、パレットと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ReleasePartOnPallet

ピックしているワークをパレットにプレースします。

● 関数呼び出し

関数	ReleasePartOnPallet(IExtendedShapeScript shapeScript, int stepId, IVisualizable robotTool, IShapeBase partModel, IShapeBase palletModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
IVisualizable robotTool	この関数でワークをピックしている 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークが、「robotTool」で指定されている 3D 形状データの子となっており、「palletModel」で指定されている 3D 形状データと衝突したときに、ワークの親を「palletModel」で指定されている 3D 形状データに変更します。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ReleasePartBySignal

ワークをロボットハンドの把持信号により放します（置きます）。

● 関数呼び出し

関数	ReleasePartBySignal(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable robotTool, IShapeBase partModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPart”
IVisualizable robotTool	この関数でワークをピックしている 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IVisualizable locationModel	ワークをプレースする 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が OFF の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークが、「robotTool」で指定されている 3D 形状データの子となっているときに、ワークを落下させ、ワークの親を消去します。
ワークが「locationModel」で指定されている 3D 形状データに衝突した場合、ワークの親を「locationModel」で指定されている 3D 形状データにします。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ReleasePalletBySignal

パレットをロボットハンドの把持信号により放します（置きます）。

● 関数呼び出し

関数	ReleasePalletBySignal(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable robotTool, IShapeBase palletModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPallet”
IVisualizable robotTool	この関数でパレットをピックしている 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IVisualizable locationModel	パレットをプレースする 3D 形状データに相当する変数を指定します。	placeModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が OFF の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「palletModel」のパレットが、「robotTool」で指定されている 3D 形状データの子となっているときに、パレットを落下させ、パレットの親を消去します。
パレットが「locationModel」で指定されている 3D 形状データに衝突した場合、パレットの親を「locationModel」で指定されている 3D 形状データにします。
処理の結果、パレットと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。

ReleasePartOnPalletBySignal

ワークをロボットハンドの把持信号によりパレットに放します（置きます）。

● 関数呼び出し

関数	ReleasePartOnPalletBySignal(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, IVisualizable robotTool, IShapeBase partModel, IShapeBase palletModel, IEnumerable<ShapeRenderInfo> renderInfoList, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPallet”
IVisualizable robotTool	この関数でワークをピックしている 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IShapeBase palletModel	「renderInfoList」に登録されているパレットの 3D 形状データを示す変数を指定します。	palletModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1. ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が OFF の場合、ワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」のワークが、「robotTool」で指定されている 3D 形状データの子となっているときに、ワークを落下させ、ワークの親を消去します。
ワークが「palletModel」で指定されている 3D 形状データに衝突した場合、ワークの親を「palletModel」で指定されている 3D 形状データにします。
処理の結果、ワークと 3D 形状データの親子関係が変更された場合に、ワーク／パレット状態番号を 1 インクリメントします。
ワークをパレットにプレースする場合に使用します。

A-1-7　状態の遷移、検知

SetNextStep

stepId の値を変更します。

● 関数呼び出し

関数	SetNextStep(IExtendedShapeScript shapeScript, int stepId, string controllerName, string variableName, int value, IDictionary<string, object> variableValues = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
int stepId	ワーク／パレット状態番号*1 を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“NextStep”
int value	変更したいワーク／パレット状態番号を指定します。	2
IDictionary<string, object> variableValues = null	GetVariableValues で取得した変数一覧を指定します。コントローラから複数の変数を取得する場合、あらかじめ GetVariableValues で複数の変数の値を取得しておくと、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。	-

*1.　ワーク／パレット状態番号とは、連続的に実行されるカテゴリが「ワーク、パレットの搬送」の関数の実行順序を決めるものです。
シェイプスクリプトの初回実行時に 1 がセットされ、カテゴリが「ワーク、パレットの搬送」の関数は、自関数の第 1 パラメータに指定されている「stepId」が一致しているときに処理を実行し、処理の結果ワーク／パレットと 3D 形状データの親子関係が変更された場合に 1 インクリメントします。また、SetNextStep()関数により、任意の番号を指定することができます。これにより、「ワーク、パレットの搬送」の関数の実行順序を設定することができます。

● 説明

ワークの状態番号が「stepId」、かつコントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数が ON のときに、ワークの状態番号を「value」の値に変更します。

DetectPartCollision

ワーク、またはパレットがメカニカル機構に衝突したことを検知します。

● 関数呼び出し

関数	DetectPartCollision(IExtendedShapeScript shapeScript, IShapeBase partModel, IVisualizable mechanicalModel, int targetStep , string controllerName, string variableName, IEnumerable<ShapeRenderInfo> renderInfoList, IList<Tuple<string, object>> setVariableList = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IShapeBase partModel	「renderInfoList」に登録されているワーク／パレットの 3D 形状データを示す変数を指定します。	partModel
IVisualizable mechanicalModel	メカニカル機構の 3D 形状データ名を指定します。	XYTable
int targetStep	ワーク／パレット状態番号を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPallet”
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-
IList<Tuple<string, object>> setVariableList = null	CreateSetVariableList()関数で生成した書き込み変数リストを指定します。DetectPartCollision()関数が実行されると、書き込み変数リストに書き込みたい変数名と値が追加されます。コントローラに複数の変数の値を書き込みたい場合、この書き込み変数リストに変数名と値を追加して SetVariableValues()関数の引数に書き込み変数リストを設定すると、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。null が設定されたときは、DetectPartCollision()関数の中で「variableName」に指定した値をコントローラに書き込みます。	-

● 説明

ワーク／パレット状態番号が「targetStep」、かつワーク／パレットインスタンス一覧の中に登録されている 3D 形状データ名「partModel」とインスタンス名「name」のワークが、「mechanicalModel」で指定されているメカニカル機構と衝突しているとき、トレースメッセージにワークのワールド座標を表示します。
コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数がセットされている場合、そのグローバル変数を ON します。

DetectMechanicalComponentsCollision

メカニカル機構がほかのメカニカル機構に衝突したことを検知します。

● 関数呼び出し

関数	DetectMechanicalComponentsCollision(IExtendedShapeScript shapeScript, string csName, int targetStep, string controllerName, string variableName, IList<Tuple<string, object>> setVariableList = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-

引数	説明	記述例
string csName	衝突検知フィルタに登録されている名前(CollisionSourceName)を「csName」に設定します。	“Slider, メカニカル機構全体”
int targetStep	ワーク／パレット状態番号を指定します。	1
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPallet”
IList<Tuple<string, object>> setVariableList = null	CreateSetVariableList()関数で生成した書き込み変数リストを指定します。 DetectMechanicalComponetsCollision()関数が実行されると、書き込み変数リストに書き込みたい変数名と値が追加されます。コントローラに複数の変数の値を書き込みたい場合、この書き込み変数リストに変数名と値を追加して SetVariableValues()関数の引数に書き込み変数リストを設定すると、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。null が設定されたときは、DetectMechanicalComponetsCollision()関数の中で「variableName」に指定した値をコントローラに書き込みます。	-

● 説明

ワーク／パレット状態番号が「targetStep」、かつ「csName」で指定されているメカニカル機構が衝突した場合、トレースメッセージに衝突した対象を衝突検知フィルタに登録されている名前で表示します。
コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数がセットされている場合、そのグローバル変数を ON します。

SetClampStatus

ロボットハンドの把持（衝突）状態をコントローラの変数に書き込みます。

● 関数呼び出し

関数	SetClampStatus(IExtendedShapeScript shapeScript, string controllerName, string variableName, IVisualizable robotTool, IShapeBase partModel, IEnumerable<ShapeRenderInfo> renderInfoList, IList<Tuple<string, object>> setVariableList = null)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	“GraspPart”
IVisualizable robotTool	ワークをピックしている 3D 形状データに相当する変数を指定します。	pickModel
IShapeBase partModel	「renderInfoList」に登録されているワーク／パレットの 3D 形状データを示す変数を指定します。	partModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

引数	説明	記述例
IList<Tuple<string, object>> setVariableList = null	CreateSetVariableList で生成した書き込み変数リストを指定します。 SetClampStatus()関数が実行されると、書き込み変数リストに書き込みたい変数名と値が追加されます。コントローラに複数の変数の値を書き込みたい場合、この書き込み変数リストに変数名と値を追加して SetVariableValues()関数の引数に書き込み変数リストを設定すると、処理が速くなることがあります。省略することも可能です。省略した場合、null が設定されます。null が設定されたときは、SetClampStatus()関数の中で「variableName」に指定した値をコントローラに書き込みます。	-

● 説明

「robotTool」で指定されている 3D 形状データが「partModel」で指定されている子（ワーク／パレット）を持っている場合、コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数を ON します。

GetCollidingSourceNames

指定した 3D 形状データと衝突している、すべての 3D 形状データの名称を取得します。

● 関数呼び出し

関数	IEnumerable<string>GetCollidingSourceNames(string collisionSourceName)
使用箇所	Render()関数内

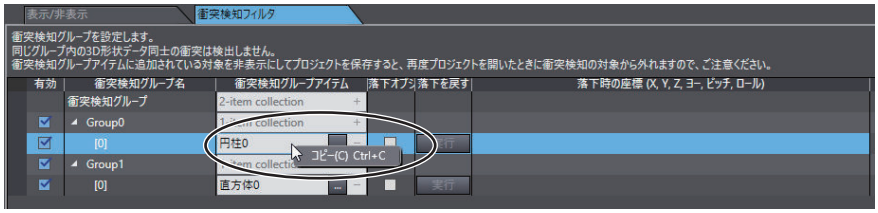
● 引数

引数	説明	記述例
string collisionSourceName	3D 形状データの名称を文字列で指定します。	“円柱 0”

● 説明

引数に指定した 3D 形状データと衝突している、すべての 3D 形状データの名称を取得します。値はこの関数の戻り値になります。

引数に指定する「3D 形状データの名称」は、衝突検知フィルタから取得することができます。[衝突検知フィルタ] 画面で対象の衝突グループアイテムを右クリックし、コピーすることができます。



A-1-8　コントローラとの変数共有

GetBoolVariable

指定したコントローラから BOOL 変数の値を読み出します。

● 関数呼び出し

関数	Bool GetBoolVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数を読み出します。値はこの関数の戻り値になります。

SetBoolVariable

指定したコントローラの BOOL 変数に値を書き込みます。

● 関数呼び出し

関数	SetBoolVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, bool value)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型グローバル変数名を文字列で指定します。	-
bool value	TRUE または FALSE を指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型グローバル変数に値を書き込みます。

GetIntegerVariable

指定したコントローラから DINT 変数の値を読み出します。

● 関数呼び出し

関数	int GetIntegerVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの DINT 型グローバル変数名を文字列で指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の DINT 型グローバル変数を読み出します。値はこの関数の戻り値になります。

SetIntegerVariable

指定したコントローラの DINT 変数に INT 型の値を書き込みます。

● 関数呼び出し

関数	SetIntegerVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, int value)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの DINT 型グローバル変数名を文字列で指定します。	-
int value	INT 型の値を指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の DINT 型グローバル変数に値を書き込みます。

Get**Variable

指定したコントローラから整数型変数の値を読み出します。関数名の“**”にはコントローラの変数のデータ型を指定します。指定できる変数のデータ型と具体的な関数名は、次の「関数呼び出し (P.A-40)」を参照してください。

● 関数呼び出し

関数	sbyte GetSintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) short GetIntVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) int GetDintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) long GetLintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) byte GetUsintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) ushort GetUintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) uint GetUdintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) ulong GetUlintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● 引数

項目	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの整数型変数名を文字列で指定します。	-

● 説明

コントローラ名「controllerName」、変数名「variableName」の整数型変数を読み出します。値はこの関数の戻り値になります。

Set**Variable

指定したコントローラの整数型変数に値を書き込みます。関数名の“**”にはコントローラの変数のデータ型を指定します。指定できる変数のデータ型と具体的な関数名は、次の「関数呼び出し (P.A-41)」を参照してください。

● 関数呼び出し

関数	SetSintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetIntVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetDintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetLintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUsintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUdintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUlintVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value)
使用箇所	Render()関数内

● 引数

項目	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの整数型変数名を文字列で指定します。	-
object value	設定する値を指定します。シェイプスクリプトの以下のデータ型に変換できる値を設定してください。 SetSintVariable: sbyte SetIntVariable: short SetDintVariable: int SetLintVariable: long SetUsintVariable: byte SetUintVariable: ushort SetUdintVariable: uint SetUlintVariable: ulong	-

● 説明

コントローラ名「controllerName」、変数名「variableName」の整数型変数に値を書き込みます。
書き込めない値が指定された場合、トレースメッセージにエラーメッセージが表示されます。

GetByteArrayVariable

指定したコントローラから BYTE 型配列変数の値を読み出します。

● 関数呼び出し

関数	byte[] GetByteArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BYTE 型配列のグローバル変数名を文字列で指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の BYTE 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetByteArrayVariable

指定したコントローラの BYTE 型配列変数に値を書き込みます。

● **関数呼び出し**

関数	SetByteArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, byte[] values)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BYTE 型配列のグローバル変数名を文字列で指定します。	-
byte[] values	BYTE 型配列の値を指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の BYTE 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetBoolArrayVariable

指定したコントローラから BOOL 型配列変数の値を読み出します。

● **関数呼び出し**

関数	bool[] GetBoolArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)

使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型配列のグローバル変数名を文字列で指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetBoolArrayVariable

指定したコントローラの BOOL 型配列変数に値を書き込みます。

● **関数呼び出し**

関数	SetBoolArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, bool[] values)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの BOOL 型配列のグローバル変数名を文字列で指定します。	-
bool[] values	BOOL 型配列の値を指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の BOOL 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetIntegerArrayVariable

指定したコントローラから DINT 型配列変数の値を読み出します。

● 関数呼び出し

関数	int[] GetIntegerArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの DINT 型配列のグローバル変数名を文字列で指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の DINT 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetIntegerArrayVariable

指定したコントローラの DINT 型配列変数に int 型配列変数の値を書き込みます。

● 関数呼び出し

関数	SetIntegerArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, int[] values)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの DINT 型配列のグローバル変数名を文字列で指定します。	-
int[] values	int 型配列の値を指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の DINT 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetWordArrayVariable

指定したコントローラから WORD 型配列変数の値を読み出します。

● 関数呼び出し

関数	ushort[]GetWordArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの WORD 型配列のグローバル変数名を文字列で指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の WORD 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetWordArrayVariable

指定したコントローラの WORD 型配列変数に ushort 型配列変数の値を書き込みます。

● 関数呼び出し

関数	SetWordArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, ushort[] values)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの WORD 型配列のグローバル変数名を文字列で指定します。	-
ushort[] values	ushort 型配列の値を指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の WORD 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetDwordArrayVariable

指定したコントローラから DWORD 型配列変数の値を読み出します。

● 関数呼び出し

関数	uint[]GetDwordArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの DWORD 型配列のグローバル変数名を文字列で指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の DWORD 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetDwordArrayVariable

指定したコントローラの DWORD 型配列変数に uint 型配列変数の値を書き込みます。

● 関数呼び出し

関数	SetDwordArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, uint[] values)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの DWORD 型配列のグローバル変数名を文字列で指定します。	-
uint[] values	uint 型配列の値を指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の DWORD 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetLwordArrayVariable

指定したコントローラから LWORD 型配列変数の値を読み出します。

● **関数呼び出し**

関数	ulong[]GetLwordArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの LWORD 型配列のグローバル変数名を文字列で指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の LWORD 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetLwordArrayVariable

指定したコントローラの LWORD 型配列変数に ulong 型配列変数の値を書き込みます。

● **関数呼び出し**

関数	SetLwordArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, ulong[] values)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの LWORD 型配列のグローバル変数名を文字列で指定します。	-

引数	説明	記述例
ulong[] values	ulong 型配列の値を指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の LWORD 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetRealArrayVariable

指定したコントローラから REAL 型配列変数の値を読み出します。

● **関数呼び出し**

関数	float[]GetRealArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの REAL 型配列のグローバル変数名を文字列で指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の REAL 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetRealArrayVariable

指定したコントローラの REAL 型配列変数に float 型配列変数の値を書き込みます。

● **関数呼び出し**

関数	SetRealArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, float[] values)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの REAL 型配列のグローバル変数名を文字列で指定します。	-
float[] values	float 型配列の値を指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の REAL 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetLrealArrayVariable

指定したコントローラから LREAL 型配列変数の値を読み出します。

● **関数呼び出し**

関数	double[]GetLrealArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの LREAL 型配列のグローバル変数名を文字列で指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の LREAL 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetLrealArrayVariable

指定したコントローラの LREAL 型配列変数に double 型配列変数の値を書き込みます。

● **関数呼び出し**

関数	SetLrealArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, double[] values)

使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの LREAL 型配列のグローバル変数名を文字列で指定します。	-
double[] values	double 型配列の値を指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の LREAL 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

GetStringArrayVariable

指定したコントローラから STRING 型配列変数の値を読み出します。

● **関数呼び出し**

関数	string[] GetStringArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの STRING 型配列のグローバル変数名を文字列で指定します。	-

● **説明**

コントローラ名「controllerName」、グローバル変数名「variableName」の STRING 型配列のグローバル変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

SetStringArrayVariable

指定したコントローラの STRING 型配列変数に値を書き込みます。

● 関数呼び出し

関数	SetStringArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, string[] values)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの STRING 型配列のグローバル変数名を文字列で指定します。	-
string[] values	string 型配列の値を指定します。	-

● 説明

コントローラ名「controllerName」、グローバル変数名「variableName」の STRING 型配列のグローバル変数に値を書き込みます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

Get**ArrayVariable

指定したコントローラから整数型配列変数の値を読み出します。関数名の“**”にはコントローラの変数のデータ型を指定します。指定できる変数のデータ型と具体的な関数名は、次の「関数呼び出し(P.A-51)」を参照してください。

● 関数呼び出し

関数	sbyte[] GetSintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) short[] GetIntArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) int[] GetDintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) long[] GetLintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) byte[] GetUsintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) ushort[] GetUintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) uint[] GetUdintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName) ulong[] GetUlintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-

引数	説明	記述例
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの整数型配列名を文字列で指定します。	-

● 説明

コントローラ名「controllerName」、変数名「variableName」の整数型配列変数を読み出します。値はこの関数の戻り値になります。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

Set**ArrayVariable

指定したコントローラの整数型配列変数に値を書き込みます。関数名の“**”にはコントローラの変数のデータ型を指定します。指定できる変数のデータ型と具体的な関数名は、次の「関数呼び出し(P.A-52)」を参照してください。

● 関数呼び出し

関数	SetSintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetIntArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetDintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetLintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUsintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUdintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value) SetUlintArrayVariable(IExtendedShapeScript shapeScript, string controllerName, string variableName, object value)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	シェイプスクリプトのインスタンス名を指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
string variableName	コントローラの整数型配列変数名を文字列で指定します。	-

引数	説明	記述例
object value	設定する値を指定します。シェイプスクリプトの以下のデータ型に変換できる値を設定してください。 SetSintArrayVariable: sbyte[] SetIntArrayVariable: short[] SetDintArrayVariable: int[] SetLintArrayVariable: long[] SetUsintArrayVariable: byte[] SetUintArrayVariable: ushort[] SetUdintArrayVariable: uint[] SetUlintArrayVariable: ulong[]	-

● 説明

コントローラ名「controllerName」、変数名「variableName」の整数型配列変数に値を書き込みます。書き込めない値が指定された場合、トレースメッセージにエラーメッセージが表示されます。

バージョン関連情報

本関数は Sysmac Studio Ver.1.51 以前では実行エラーになります。

CreateGetVariableList

コントローラから複数の変数を読み出すときに使用する読み出し変数リストを新規作成します。

● 関数呼び出し

関数	IList<Tuple<string, Type>> CreateGetVariableList()
使用箇所	Render()関数内

● 引数

なし。

● 説明

GetVariableValues()関数の引数に指定する読み出し変数リストを新規作成します。新規作成された空の読み出し変数リストが関数の戻り値になります。AddToGetVariableList()関数、GetVariableValues()関数と組み合わせて使用してください。

コントローラ“new_Controller_0”から BOOL 型の変数“boolVar1”,“boolVar2”,“boolVar3”を一度に取得したいときの記述例を以下に示します。

```
62      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
63          if (this.IsInitialized == false) {
64              DefaultFunctions.InitializeObject(this, renderInfoList);
65              return ;
66          }
67
68          var getVariableList = DefaultFunctions.CreateGetVariableList();
69          DefaultFunctions.AddToGetVariableList("boolVar1", "BOOL", getVariableList);
70          DefaultFunctions.AddToGetVariableList("boolVar2", "BOOL", getVariableList);
71          DefaultFunctions.AddToGetVariableList("boolVar3", "BOOL", getVariableList);
72
73          var variableValueList = DefaultFunctions.GetVariableValues(this, "new_Controller_0", getVariableList);
74      }
```

68 行目で読み出し変数リストを新規作成します。次に、69～71 行目で AddToGetVariableList()関数により読み出したい変数を読み出し変数リストに追加します。そして、73 行目で GetVariableValues()関数によりコントローラから読み出した変数の一覧を取得します。

AddToGetVariableList

コントローラから読み出したい変数を、読み出し変数リストに追加します。

● 関数呼び出し

関数	bool AddToGetVariableList(string variableName, string iecDataType, IList<Tuple<string, Type>> getVariableList, bool isShowErrorMessage = true)
使用箇所	Render()関数内

● 引数

項目	説明	記述例
string variableName	コントローラの変数名を文字列で指定します。	-
string iecDataType	変数のデータ型を文字列で指定します。	“BOOL”
IList<Tuple<string, Type>> getVariableList	読み出し変数リストの参照を指定します。	-
bool isShowErrorMessage	読み出し変数リストの追加に失敗したときに、エラーメッセージをトレースメッセージに表示するかどうかを設定します。この引数は省略することは可能です。省略した場合、エラーメッセージはトレースメッセージに表示されます。	true

● 説明

変数名「variableName」とデータ型「iecDataType」を読み出し変数リスト「getVariableList」に追加します。変数の追加に成功したら、戻り値は TRUE になります。「iecDataType」に不正な値を設定したとき、追加しようとした変数がすでに追加されている場合、戻り値は FALSE になります。また、「isShowErrorMessage」が TRUE であれば、エラーメッセージをトレースメッセージに表示します。

GetVariableValues

コントローラから複数の変数の値を一度に読み出します。

● 関数呼び出し

関数	IDictionary<string, object> GetVariableValues(IExtendedShapeScript shapeScript, string controllerName, IList<Tuple<string, Type>> getVariableList)
使用箇所	Render()関数内

● 引数

項目	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
IList<Tuple<string, Type>> getVariableList	読み出し変数リストの参照を指定します。	-

● 説明

読み出し変数リストに追加された変数名をもとに、コントローラの変数の値を読み出します。読み出した変数の変数名と値の組み合わせがこの関数の戻り値になります。

バージョン関連情報

本関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

GetValueFromVariableValueList

GetVariableValues()関数によって読み出された変数の値を取得します。

● 関数呼び出し

関数	object GetValueFromVariableValueList(string variableName, IDictionary<string, object> variableValueList)
使用箇所	Render()関数内

● 引数

項目	説明	記述例
string variableName	コントローラの変数名を文字列で指定します。	-
IDictionary<string, object> variableValueList	GetVariableValues()関数によって読み出された変数の値のリストを指定します。	-

● 説明

GetVariableValues()関数によって読み出された変数の値を取得します。変数の値がこの関数の戻り値となります。指定した変数名に対応する値を取得していなかった場合、戻り値は null になります。

戻り値の型は、コントローラの変数のデータ型によって異なります。コントローラの変数のデータ型とシェイプスクリプトの変数のデータ型の対応は以下のとおりです。

コントローラの変数のデータ型	シェイプスクリプトの変数のデータ型
BOOL	bool
SINT	sbyte
USINT	byte
INT	short
UINT	ushort
DINT	int
UDINT	uint
LINT	long
ULINT	ulong
BYTE	byte
WORD	ushort
DWORD	uint
LWORD	ulong
REAL	float
LREAL	double
TIME	TimeSpan
DATE	DateTime
TIME_OF_DAY	DateTime
DATE_AND_TIME	DateTime
STRING	string

読み出された変数“boolVar1”の値を取得するときの記述例を以下に示します。

```
62      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
63          if (this.IsInitialized == false) {
64              DefaultFunctions.InitializeObject(this, renderInfoList);
65              return ;
66          }
67
68          var getVariableList = DefaultFunctions.CreateGetVariableList();
69          DefaultFunctions.AddToGetVariableList("boolVar1", "BOOL", getVariableList);
70          DefaultFunctions.AddToGetVariableList("boolVar2", "BOOL", getVariableList);
71          DefaultFunctions.AddToGetVariableList("boolVar3", "BOOL", getVariableList);
72
73          var variableValueList = DefaultFunctions.GetVariableValues(this, "new_Controller_0", getVariableList);
74          var result = (bool)DefaultFunctions.GetValueFromVariableValueList("boolVar1", variableValueList);
75      }
```

73 行目で GetVariableValues()関数によりコントローラから読み出した変数の一覧を取得します。74 行目で、取得した変数の値の一覧から読み出したい変数の値のみを取得します。取得した値はコントローラの変数のデータ型に応じてシェイプスクリプトの変数のデータ型にキャストしてください。

CreateSetVariableList

コントローラから複数の変数の値を書き込むときに使用する書き込み変数リストを新規作成します。

● 関数呼び出し

関数	IList<Tuple<string, object>> CreateSetVariableList()
使用箇所	Render()関数内

● 引数

なし。

● 説明

SetVariableValues()関数の引数に指定する書き込み変数リストを新規作成します。新規作成された空の書き込み変数リストが関数の戻り値になります。AddToSetVariableList()関数、SetVariableValues()関数と組み合わせて使用してください。

コントローラ“new_Controller_0”に INT 型の変数“intVar1”，“intVar2”，“intVar3”にそれぞれ“1”，“2”，“3”を一度に書き込みたいときの記述例を以下に示します。

```
62      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
63          if (this.IsInitialized == false) {
64              DefaultFunctions.InitializeObject(this, renderInfoList);
65              return ;
66          }
67
68          var setVariableList = DefaultFunctions.CreateSetVariableList();
69          DefaultFunctions.AddToSetVariableList("intVar1", "INT", 1, setVariableList);
70          DefaultFunctions.AddToSetVariableList("intVar2", "INT", 2, setVariableList);
71          DefaultFunctions.AddToSetVariableList("intVar3", "INT", 3, setVariableList);
72
73          DefaultFunctions.SetVariableValues(this, "new_Controller_0", setVariableList);
74      }
```

68 行目で書き込み変数リストを新規作成します。次に、69～71 行目で AddToSetVariableList()関数により書き込みたい値を書き込み変数リストに追加します。そして、73 行目で SetVariableValues()関数によりコントローラの変数に値を書き込みます。

AddToSetVariableList

コントローラに書き込みたい変数と値を、書き込み変数リストに追加します。

● 関数呼び出し

関数	bool AddToSetVariableList(string variableName, string iecDataType, object setValue, IList<Tuple<string, object>> setVariableList, bool isShowErrorMessage = true)

使用箇所	Render()関数内

● **引数**

項目	説明	記述例
string variableName	コントローラの変数名を文字列で指定します。	-
string iecDataType	変数のデータ型を文字列で指定します。	“BOOL”
object setValue	コントローラの変数に書き込みたい値を指定します。	-
IList<Tuple<string, object>> setVariableList	書き込み変数リストの参照を指定します。	-
bool isShowErrorMessage	読み出し変数リストの追加に失敗したときに、エラーメッセージをトレースメッセージに表示するかどうかを設定します。この引数は省略することは可能です。省略した場合、エラーメッセージはトレースメッセージに表示されます。	true

● **説明**

変数名「variableName」と設定値「setValue」を書き込み変数リスト「setVariableList」に追加します。
変数の追加に成功したら、戻り値は TRUE になります。以下の場合、戻り値は FALSE になり、変数名と設定値は書き込み変数リストに追加されません。
- 「iecDataType」に不正な値を設定した
- 「setValue」が不正な値
- 追加しようとした変数がすでに追加されている

また、「isShowErrorMessage」が TRUE であれば、エラーメッセージをトレースメッセージに表示します。

SetVariableValues

コントローラの複数の変数に値を一度に書き込みます。

● **関数呼び出し**

関数	SetVariableValues(IExtendedShapeScript shapeScript, string controllerName, IList<Tuple<string, object>> setVariableList)
使用箇所	Render()関数内

● **引数**

項目	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”
IList<Tuple<string, object>> setVariableList	書き込み変数リストの参照を指定します。	-

● **説明**

書き込み変数リストに追加された変数名と値をもとに、コントローラの変数に値を書き込みます。

バージョン関連情報

本関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

GetCurrentControllerTime

指定したコントローラのシミュレーションの内部時刻を読み出します。

● 関数呼び出し

関数	DateTime GetCurrentControllerTime(IExtendedShapeScript shapeScript, string controllerName)
使用箇所	Render()関数内

● 引数

項目	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string controllerName	コントローラ名を文字列で指定します。	“new_controller_0”

● 説明

コントローラ名「controllerName」のシミュレーションの内部時刻を読み出します。値はこの関数の戻り値になります。

A-1-9　プロセスマネージャとの連携

MovePartOnPackManagerSensorLatchBelt

プロセスマネージャのコンベア上にあるワークを、ラッチ位置まで移動させ、プロセスマネージャのエミュレーション機能で、続けてワークを操作できるようにします。
プロセスマネージャのコンフィグレーションで、ベルトラッチセンサを使用している場合に有効です。

● 関数呼び出し

関数	MovePartOnPackManagerSensorLatchBelt(IExtendedShapeScript shapeScript, IProcessManager processManager, IBelt beltModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IProcessManager processManager	この関数の前にあらかじめアプリケーションマネージャに登録したプロセスマネージャを代入した変数を準備しておきます。	IProcessManager processManager=(IProcessManager)ace["/ApplicationManager0/Process Manager"];
IBelt beltModel	プロセスマネージャに登録したコンベア（ベルト）を代入した変数を準備しておきます。	IBelt beltModel=(IBelt)ace["/ApplicationManager0/Pick Belt"];

引数	説明	記述例
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfoで指定したワーク／パレットインスタンス一覧の名前を指定します。	-

● 説明

プロセスマネージャのエミュレーション機能で操作されているコンベア（ベルト）の動作に合わせて、ワーク／パレットインスタンス一覧の中に登録されているワークの 3D 形状データのうち、コンベアの 3D 形状データ「beltModel」の子となっているものをラッチ位置まで移動します。ラッチ位置まで移動したワークは、続けてプロセスマネージャにより操作されます。

バージョン関連情報

本関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

MovePalletOnPackManagerSensorLatchBelt

プロセスマネージャのコンベア上にあるパレットを、ラッチ位置まで移動させ、プロセスマネージャのエミュレーション機能で、続けてパレットを操作できるようにします。
プロセスマネージャのコンフィグレーションで、ベルトラッチセンサを使用している場合に有効です。

● 関数呼び出し

関数	MovePalletOnPackManagerSensorLatchBelt(IExtendedShapeScript shapeScript, IProcessManager processManager, IBelt beltModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IProcessManager processManager	この関数の前にあらかじめアプリケーションマネージャに登録し、プロセスマネージャを代入した変数を準備しておきます。	IProcessManager processManager=(IProcessManager)ace["/ApplicationManager0/Process Manager"];
IBelt beltModel	プロセスマネージャに登録したコンベア（ベルト）を代入した変数を準備しておきます。	IBelt beltModel=(IBelt)ace["/ApplicationManager0/Pick Belt"];
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfoで指定したワーク／パレットインスタンス一覧の名前を指定します。	-

● 説明

プロセスマネージャのエミュレーション機能で操作されているコンベア（ベルト）の動作に合わせて、ワーク／パレットインスタンス一覧の中に登録されているパレットの 3D 形状データのうち、コンベアの 3D 形状データ「beltModel」の子となっているものをラッチ位置まで移動します。また、パレットに配置されているワークも合わせて移動させます。ラッチ位置まで移動したパレットは、続けてプロセスマネージャにより操作されます。

バージョン関連情報

本関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

SetPackManagerPartPosition

プロセスマネージャで操作中のワークの位置を、コントローラの変数に書き込みます。

● 関数呼び出し

関数	SetPackManagerPart Position(IExtendedShapeScript shapeScript, string controllerName, string variableDX, string variableDY, string variableDZ, IShapeBase partModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
string ControllerName	コントローラ名を文字列で指定します。	“new_Controller_0”
string variableDX	ワークの X 位置を格納するコントローラの LREAL 型グローバル配列変数名を文字列で指定します。	“PositionX”
string variableDY	ワークの Y 位置を格納するコントローラの LREAL 型グローバル配列変数名を文字列で指定します。	“PositionY”
string variableDZ	ワークの Z 位置を格納するコントローラの LREAL 型グローバル配列変数名を文字列で指定します。	“PositionZ”
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

● 説明

プロセスマネージャで操作中のワークの X、Y、Z 位置を、コントローラの LREAL 型のグローバル配列変数に書き込みます。プロセスマネージャで操作中のワークには、0 から ID 番号が割り当てられており、ID 番号が配列の添え字に使用されます。つまり、ID が 0 のワークの X 位置は PositionX[0]に、ID が 1 のワークの X 位置は PositionX[1]に書き込まれます。変数名を null にした場合には、位置の値はどこにも書き込まれません。
このときのワークは、シェイプスクリプトで作成する必要があります。

バージョン関連情報

本関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

UpdatePartDataFromPackManager

プロセスマネージャでピック&プレースされたワークを、シェイプスクリプトで続けて操作できるようにします。

● 関数呼び出し

関数	UpdatePartDataFromPackManager(IShapeBase partModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel
IVisualizable locationModel	プロセスマネージャがワークをプレースした先の 3D 形状データを示す変数を指定します。	locationModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

● 説明

シェイプスクリプトで作成したワークに対して、プロセスマネージャでピック&プレースの操作を行ったあとに、再びシェイプスクリプトで操作できるようにします。

バージョン関連情報

本関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

UpdatePartDataOnPalletFromPackManager

プロセスマネージャでピックされ、パレット上にプレースされたワークをシェイプスクリプトで続けて操作できるようにします。

● 関数呼び出し

関数	UpdatePartDataOnPalletFromPackManager(IExtendedShapeScript shapeScript, IShapeBase partModel, IShapeBase palletModel, IVisualizable locationModel, IEnumerable<ShapeRenderInfo> renderInfoList)
使用箇所	Render()関数内

● 引数

引数	説明	記述例
IExtendedShapeScript shapeScript	この関数で使用するシェイプスクリプトを指定します。	-
IShapeBase partModel	「renderInfoList」に登録されているワークの 3D 形状データを示す変数を指定します。	partModel

引数	説明	記述例
IShapeBase palletModel	「renderInfoList」に登録されているパレットの3D 形状データを示す変数を指定します。	palletModel
IVisualizable locationModel	パレットの親となっている 3D 形状データを示す変数を指定します。	locationModel
IEnumerable<ShapeRenderInfo> renderInfoList	CreateRenderInfo で指定したワーク／パレットインスタンス一覧の名前を指定します。	-

● **説明**

シェイプスクリプトで作成したワーク、パレットに対して、プロセスマネージャでワークのピック、パレット上へプレースの操作を行ったあとに、再びパレットとパレット上のワークを合わせてシェイプスクリプトで操作できるようにします。

バージョン関連情報

本関数は、Sysmac Studio Ver.1.40 ではコンパイルエラーになります。

A-1-10　シェイプスクリプト実行時間計測

StartStopwatch

シェイプスクリプトの実行時間計測を開始します。StopStopwatch()関数とともに使用し、StartStopwatch()関数と StopStopwatch()関数で囲まれたシェイプスクリプトの、パソコン上の実行時間を計測します。

● **関数呼び出し**

関数	void StartStopwatch(string name)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
string name	シェイプスクリプトの実行時間計測を識別するための名前を設定します。	“Stopwatch0”

● **説明**

引数に設定した名前のシェイプスクリプトの実行時間計測を開始します。同じ引数の StopStopwatch()関数との間のシェイプスクリプトの実行時間を計測し、計測した値はシステムモニタに表示されます。システムモニタ上ではサイクルごとの実行時間を確認することができます。

StopStopwatch

シェイプスクリプトの実行時間計測を停止します。StartStopwatch()関数とともに使用し、StartStopwatch()関数と StopStopwatch()関数の間のシェイプスクリプトの、パソコン上の実行時間を計測します。

● **関数呼び出し**

関数	void StopStopwatch(string name)
使用箇所	Render()関数内

● **引数**

引数	説明	記述例
string name	シェイプスクリプトの実行時間計測を識別するための名前を設定します。	“Stopwatch0”

● **説明**

引数に設定した名前のシェイプスクリプトの実行時間計測を、停止します。

A-1-11　シェイプスクリプトで使用する関数の更新

Sysmac Studio のバージョンアップにより、シェイプスクリプトに新しい関数が追加されたり、既存の関数に機能が追加されたりすることがあります。
「デフォルトファンクションの更新」機能を使用すると、シェイプスクリプトの関数に変更があった場合にプロジェクトで使用している既存のシェイプスクリプトの関数を最新の状態に更新することができます。
デフォルトファンクションの更新の対象は、シェイプスクリプトファンクションのデフォルトファンクションの関数です。
シェイプスクリプトシーケンスによるシェイプスクリプトが未実行の場合、「デフォルトファンクションの更新」が可能です。

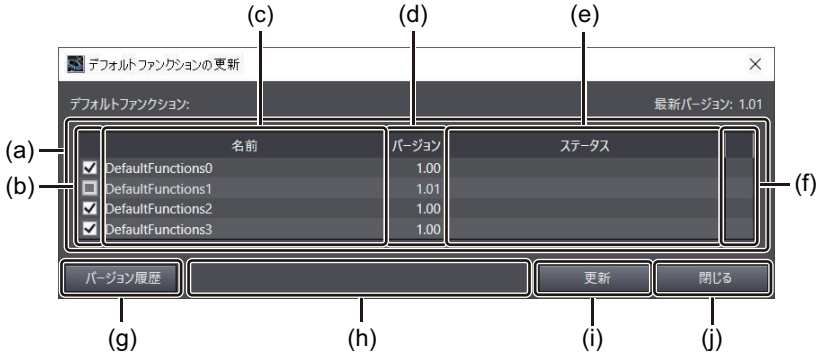
デフォルトファンクションの更新手順

デフォルトファンクションを更新する手順を説明します。

1 **マルチビューエクスプローラから［構成・設定］–［3D 表示］–［シェイプスクリプトファンクション］を右クリックして、［デフォルトファンクションの更新］を選択します。**



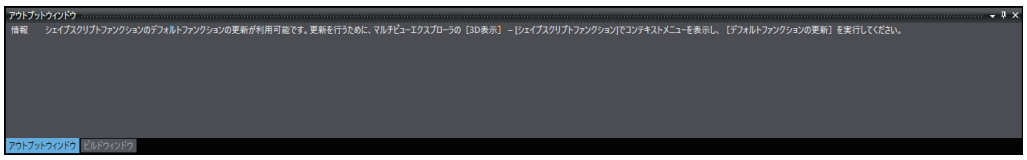
［デフォルトファンクションの更新］ダイアログボックスが表示されます。



	項目	説明
(a)	デフォルトファンクション一覧	すべてのデフォルトファンクションの一覧が表示されます。 ［バージョン］が最新バージョン以上のデフォルトファンクションはグレーアウトされます。 また、［名前］または［バージョン］のタイトルをクリックすることでソートすることができます。

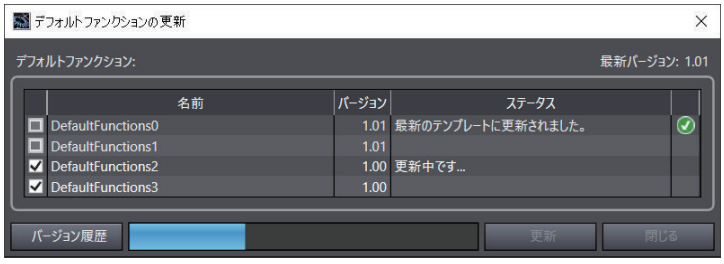
<table>
<tr><th></th><th>項目</th><th>説明</th></tr>
<tr><td>(b)</td><td>チェックボックス</td><td>グレーアウトされていないデフォルトファンクションに対して、更新する／しないを設定します。
チェック有: 更新する（デフォルト）
チェック無: 更新しない</td></tr>
<tr><td>(c)</td><td>[名前]</td><td>デフォルトファンクションの名称が表示されます。</td></tr>
<tr><td>(d)</td><td>[バージョン]</td><td>デフォルトファンクションのバージョンが表示されます。</td></tr>
<tr><td>(e)</td><td>[ステータス]</td><td>(b)チェックボックスで更新するに設定したデフォルトファンクションの更新処理の状態が表示されます。
<table>
<tr><th>表示メッセージ</th><th>状態</th></tr>
<tr><td>空白</td><td>実行待ち</td></tr>
<tr><td>更新中です</td><td>実行中</td></tr>
<tr><td>最新のテンプレートに更新されました。</td><td>実行完了（正常）</td></tr>
<tr><td>変更が検出されました。更新されていません。*1</td><td>実行完了（異常）</td></tr>
</table></td></tr>
<tr><td>(f)</td><td>ステータス（アイコン）</td><td>デフォルトファンクションの更新処理が完了すると、実行結果のアイコンが表示されます。
<table>
<tr><th>表示アイコン</th><th>状態</th></tr>
<tr><td>  </td><td>実行完了（正常）</td></tr>
<tr><td>  </td><td>実行完了（異常）</td></tr>
</table></td></tr>
<tr><td>(g)</td><td>[バージョン履歴] ボタン</td><td>ボタンをクリックすると、過去更新されたデフォルトファンクションすべてのバージョンの変更履歴の一覧が表示されます。</td></tr>
<tr><td>(h)</td><td>プログレスバー</td><td>更新処理の実行状況が表示されます。</td></tr>
<tr><td>(i)</td><td>[更新] ボタン</td><td>ボタンをクリックすると、更新処理が実行されます。*2</td></tr>
<tr><td>(j)</td><td>[閉じる] ボタン</td><td>ダイアログボックスを閉じます。</td></tr>
</table>

*1. デフォルトファンクションを手動で編集している場合は、更新されません。
*2. 本ボタンが有効になる条件は下記のとおりです。
- Sysmac Studio のプロジェクトを開いたとき、アウトプットウインドウに下記メッセージが表示された場合

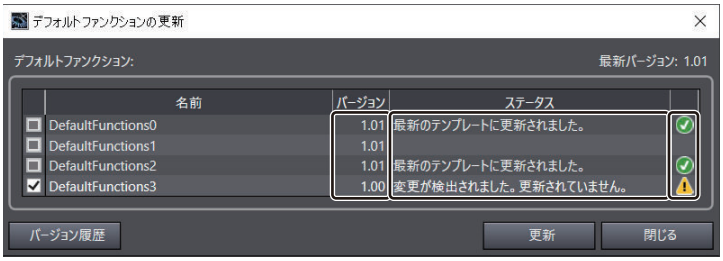


- (b)のチェックボックスにひとつ以上チェックを入れた場合

2 **［デフォルトファンクションの更新］ダイアログボックスの［更新］ボタンを選択します。**
更新処理が開始されます。



対象の全デフォルトファンクションの更新処理が順次実行されます。
［ステータス］に更新処理の進捗状況が表示されます。また、更新処理が完了すると、ステータス（アイコン）が表示されます。更新処理が正常に終了した場合、［バージョン］が更新されたバージョンに変更されます。



使用上の注意

CreateRenderInfo()関数と Render()関数は更新されません。

参考

［デフォルトファンクションの更新］ダイアログボックスの［バージョン履歴］ボタンを選択すると、過去更新されたデフォルトファンクションの全バージョンの変更履歴の一覧が表示されます。更新対象のデフォルトファンクションを選択する上での参考にしてください。



A-2 シミュレータと実機コントローラの差異

A-2-1 各機能の動作

シミュレータは、実機コントローラに比べ、以下のような機能的な差異があります。

項目		シミュレータと実機コントローラの機能的な差異
メカニカル機構	シミュレータ実行中の 3D 表示画面上の動作	シミュレータでは、メカニカル機構は各軸の指令値に合わせて動作します。質量、重力、慣性などの物理要素は考慮しません。
ロボット	シミュレータ実行中の 3D 表示画面上の動作	シミュレータでは、ロボットは各軸の指令値に合わせて動作します。質量、重力、慣性などの物理要素は考慮しません。
ベルト	ラッチ機能	プロセスマネージャのシミュレーション中、ワークのベルトプロパティに［Latch］が選択されているとき、一定間隔ごとに外部トリガを発生させることができます。シェイプスクリプトの CreateLatch()関数から発生させることも可能です。
	ベルトエンコーダ機能	シミュレータでは、ベルト制御のドライブ出力の ON/OFF を切り替えることで、エンコーダを動作させることが可能です。
ワーク検出センサ	ワーク検出センサの制約	ワーク検出センサは、3D 表示画面上でのワークの通過を検出する機能です。光電センサのような見た目をしていますが、以下の機能は備えていません。したがって、光電センサなどのレイアウト調整には使用しないでください。 • 光軸の太さ調整 • ビームの反射のシミュレーション • 応差のシミュレーション

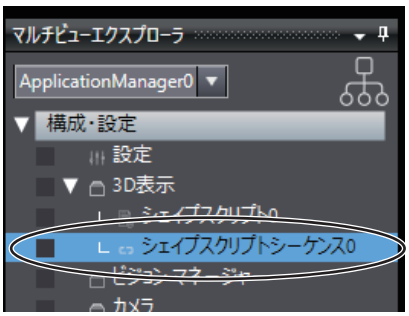
A-3　シェイプスクリプトの実行時間

シェイプスクリプトの実行時間は、スクリプトの内容によって変わります。処理に時間を要するスクリプトを記述した場合、実行時間がシェイプスクリプトシーケンスに設定した［シェイプスクリプト実行間隔］を超えることがあります。
ここでは、シェイプスクリプトの実行時間の確認方法と、実行時間を短縮する方法について説明します。

A-3-1　システムモニタによる実行時間の確認

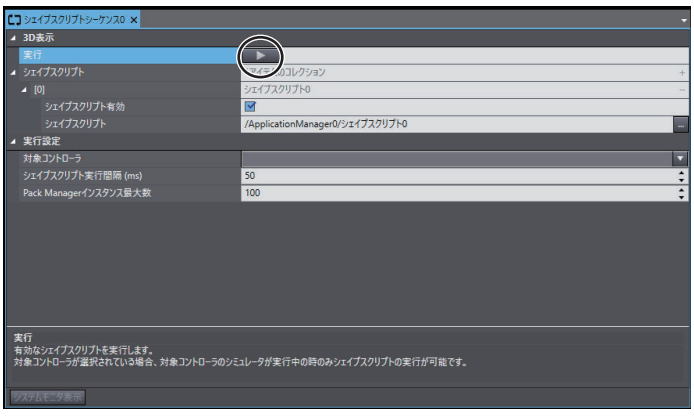
シェイプスクリプトの実行時間は、システムモニタから確認することができます。システムモニタからシェイプスクリプトの実行時間を確認する手順を説明します。

1　マルチビューエクスプローラの［構成・設定］から、実行時間を確認したいシェイプスクリプトシーケンスをダブルクリックします。



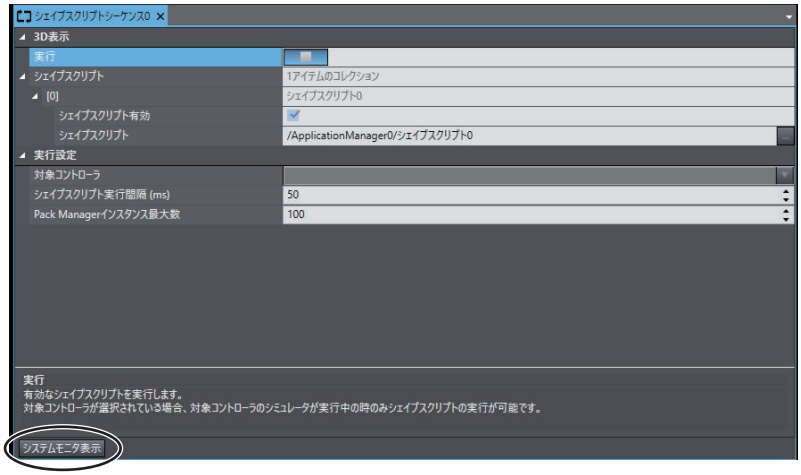
シェイプスクリプトシーケンスの設定画面が表示されます。

2　シェイプスクリプトシーケンスの［実行］ボタンをクリックします。

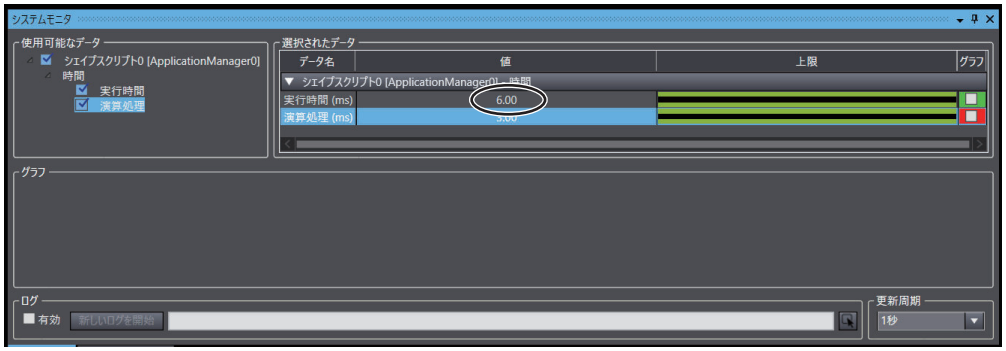


シェイプスクリプトシーケンスが実行状態になります。

3　シェイプスクリプトシーケンスの［システムモニタ表示］ボタンをクリックします。



システムモニタが表示され、シェイプスクリプトの実行時間を確認することができます。



参考

Stopwatch 構文を使用すると、シェイプスクリプトの特定の箇所の実行時間を確認することができます。
操作方法の詳細は「A-3-2 Stopwatch 構文を使用した実行時間の計測 (P.A-68)」を参照してください。

A-3-2 Stopwatch 構文を使用した実行時間の計測

シェイプスクリプトに Stopwatch 構文を記述すると、シェイプスクリプトの任意の処理について、パソコン上での実行時間を計測することができます。計測した時間はシステムモニタに表示されます。
シェイプスクリプトの実行に時間がかかる処理を特定して最適化することで、シェイプスクリプトの実行時間を短縮することができます。
算術演算を実行するシェイプスクリプトを例に、手順を説明します。

1 シェイプスクリプトの Render()関数内で、計測する対象箇所を［StartStopwatch(“演算処理”)］および［StopStopwatch(“演算処理”)］で囲みます。

「演算処理」は測定対象を識別するための名前で、任意の文字列を指定します。ここで入力した名前はシステムモニタに表示されます。

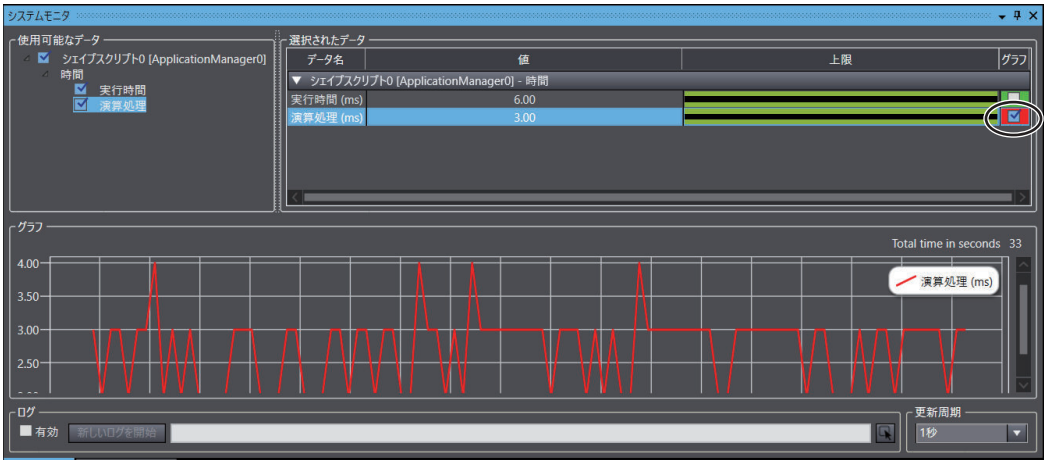
```
121         StartStopwatch("演算処理");
122         // Move in circle
123         Thread.Sleep(2);
124
125         var x = renderInfo.Position.DX;
126         var y = renderInfo.Position.DY;
127         var z = renderInfo.Position.DZ;
128         var r = Math.Sqrt(x*x + y*y);
129         double phi;
130         if (x == 0 && y == 0) phi = 0;
131         else if (x >= 0) phi = Math.Atan(y / x);
132         else phi = -Math.Asin(y / r) + Math.PI;
133
134         phi += speed;
135         x = r * Math.Cos(phi);
136         y = r * Math.Sin(phi);
137
138         renderInfo.Position = new Transform3D(x, y, z);
139         StopStopwatch("演算処理");
```

2 **対象のシェイプスクリプトを実行します。**

3 **［シェイプスクリプトシーケンス］から［システムモニタ表示］ボタンをクリックします。**
システムモニタが表示され、「演算処理」の項目に対象箇所のシェイプスクリプトの実行時間が表示されます。



「演算処理」の［グラフ］列のチェックボックスにチェックを入れると、サイクルごとの「演算処理」の実行時間がグラフに表示されます。



StartStopwatch()関数、StopStopwatch()関数の詳細については「StartStopwatch (P.A-62)」、「StopStopwatch (P.A-62)」を参照してください。

A-3-3　シェイプスクリプトの処理の高速化

時間を要するスクリプトの処理のひとつに、コントローラの変数の読み出し／書き込み処理があります。コントローラの変数の読み出し／書き込みを行うとき、パソコンの内部でネットワークを介してコントローラのシミュレータに対して読み出し／書き込みが行われており、ネットワークの通信処理に時間を要するためです。
シェイプスクリプトの Render()関数の中で複数の変数の読み出し／書き込みを行う場合、処理がひとつひとつ実行されると、処理の回数だけ通信が発生します。それに対し、複数の変数を一度にまとめ

て読み出し／書き込みを行う場合、通信回数が少なくなるため、シェイプスクリプトの実行時間を短縮することができます。
ここでは、シェイプスクリプトで複数の変数を一度に読み出す場合と書き込む場合のスクリプトの記述方法を説明します。

コントローラの複数の変数の読み出し

コントローラの複数の変数を一度に読み出すには、まず読み出し変数リストを作成します。読み出し変数リストを作成するには、CreateGetVariableList()関数と AddToGetVariableList()関数を使用してください。
次に、読み出し変数リストを使ってコントローラの複数の変数を一度に読み出します。コントローラの複数の変数を一度に読み出すには、GetVariableValues()関数を使用してください。
GetVariableValues()関数の戻り値が取得した変数とその値の一覧になります。
取得した変数と値の一覧から特定の変数の値を取得するには、GetValueFromVariableValueList()関数を使用してください。GetValueFromVariableValueList()関数の戻り値が取得したい変数の値になります。

コントローラの複数の変数を一度に読み出すシェイプスクリプトの例を、以下に示します。

```
62      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
63          if (this.IsInitialized == false) {
64              DefaultFunctions.InitializeObject(this, renderInfoList);
65              return ;
66          }
67
68          var getVariableList = DefaultFunctions.CreateGetVariableList();
69          DefaultFunctions.AddToGetVariableList("boolVar1", "BOOL", getVariableList);
70          DefaultFunctions.AddToGetVariableList("boolVar2", "BOOL", getVariableList);
71          DefaultFunctions.AddToGetVariableList("boolVar3", "BOOL", getVariableList);
72
73          var variableValueList = DefaultFunctions.GetVariableValues(this, "new_Controller_0", getVariableList);
74          var result = (bool)DefaultFunctions.GetValueFromVariableValueList("boolVar1", variableValueList);
75      }
```

この例では、コントローラ「new_Controller_0」から BOOL 型の変数「boolVar1」、「boolVar2」、「boolVar3」を一度に読み出したあと、「boolVar1」の値を取得しています。
68 行目では、CreateGetVariableList()関数を使って読み出し変数リストを新規作成しています。
次に、69～71 行目で AddToGetVariableList()関数により読み出したい変数を読み出し変数リストに追加します。
そして、73 行目で GetVariableValues()関数によりコントローラから読み出した変数の一覧を取得します。74 行目で、取得した変数の値の一覧から読み出したい変数の値のみを取得します。

バージョン関連情報

複数の変数の読み出しは、Sysmac Studio Ver1.40 では使用できません。

シェイプスクリプトで使用する関数には、関数の内部でコントローラの変数を読み出すものがあります。これらの関数を使用する際も、コントローラの複数の変数を一度に読み出すことで、シェイプスクリプトの実行時間を短縮することができます。
コントローラの変数を読み出す関数の一覧を以下に示します。

コントローラの変数を読み出す関数一覧
LoadPart
LoadPallet
LoadPartOnPallet
UnloadPart
UnloadPallet
SetNextStep

コントローラの変数を読み出す関数一覧
PushPart
PushPallet
MoveObjectOnBelt
ClampPartBySignal
ClampPalletBySignal
ClampPartOnPalletBySignal
ReleasePartBySignal
ReleasePalletBySignal
ReleasePartOnPalletBySignal

これらすべての関数に、引数「variableValues」を設定することができます。「variableValues」にGetVariableValues()関数の戻り値を設定することで、コントローラの変数の読み出しを行う際の通信処理を省略することができます。通信処理の省略によって、シェイプスクリプトの実行時間を短縮することができます。

GetVariableValues()関数の戻り値を関数の引数に設定するシェイプスクリプトの例を、以下に示します。

```
62      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
63          if (this.IsInitialized == false) {
64              DefaultFunctions.InitializeObject(this, renderInfoList);
65              return ;
66          }
67
68          var getVariableList = DefaultFunctions.CreateGetVariableList();
69          DefaultFunctions.AddToGetVariableList("boolVar1", "BOOL", getVariableList);
70          DefaultFunctions.AddToGetVariableList("boolVar2", "BOOL", getVariableList);
71          DefaultFunctions.AddToGetVariableList("boolVar3", "BOOL", getVariableList);
72
73          var variableValues = DefaultFunctions.GetVariableValues(this, "new_Controller_0", getVariableList);
74
75          ICadData partModel1 = (ICadData) ace["/ApplicationManager0/partModel1"];
76          ICadData partModel2 = (ICadData) ace["/ApplicationManager0/partModel2"];
77          ICadData table = (ICadData) ace["/ApplicationManager0/table"];
78
79          DefaultFunctions.LoadPart(this, "new_Controller_0", "boolVar1", partModel1, string.Empty, new Transform3D(0, 0, 0), table, renderInfoList, variableValues);
80          DefaultFunctions.LoadPart(this, "new_Controller_0", "boolVar2", partModel2, string.Empty, new Transform3D(100, 0, 0), table, renderInfoList, variableValues);
81      }
```

この例では、コントローラ「new_Controller_0」から BOOL 型の変数「boolVar1」、「boolVar2」、「boolVar3」を一度に読み出しています。読み出した結果は、73 行目の「variableValues」に設定されます。
79 行目では、LoadPart()関数の引数に「variableValues」を設定しています。こうすることで、LoadPart()関数内では「variableValues」から「boolVar1」の値を取得します。
同様に、80 行目の LoadPart()関数でも「variableValues」から「boolVar2」を取得します。これによって、各 LoadPart()関数内でネットワークを介さずにコントローラの変数の値を高速に取得できるようになります。

バージョン関連情報

引数「variableValues」は、Sysmac Studio Ver1.42 以前では使用できません。

コントローラの複数の変数の書き込み

コントローラの複数の変数に一度に書き込むには、まず書き込み変数リストを作成します。書き込み変数リストを作成するには、CreateSetVariableList()関数と AddToSetVariableList()関数を使用してください。
次に、書き込み変数リストを使ってコントローラの複数の変数に値を一度に書き込みます。コントローラの複数の変数をまとめて書き込むには、SetVariableValues()関数を使用してください。

コントローラの複数の変数に値を一度に書き込むシェイプスクリプトの例を、以下に示します。

```
62      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
63          if (this.IsInitialized == false) {
64              DefaultFunctions.InitializeObject(this, renderInfoList);
65              return ;
66          }
67
68          var setVariableList = DefaultFunctions.CreateSetVariableList();
69          DefaultFunctions.AddToSetVariableList("intVar1", "INT", 1, setVariableList);
70          DefaultFunctions.AddToSetVariableList("intVar2", "INT", 2, setVariableList);
71          DefaultFunctions.AddToSetVariableList("intVar3", "INT", 3, setVariableList);
72
73          DefaultFunctions.SetVariableValues(this, "new_Controller_0", setVariableList);
74      }
```

この例では、コントローラ「new_Controller_0」に INT 型の変数「intVar1」、「intVar2」、「intVar3」にそれぞれ「1」、「2」、「3」を書き込んでいます。
68 行目では、CreateSetVariableList()関数を使って書き込み変数リストを新規作成します。
次に、69～71 行目で AddToSetVariableList()関数により書き込みたい値を書き込み変数リストに追加します。
そして、73 行目で SetVariableValues()関数によりコントローラの変数に値を書き込みます。

バージョン関連情報

複数の変数の書き込みは、Sysmac Studio Ver1.40 では使用できません。

シェイプスクリプトで使用する関数には、関数の内部でコントローラの変数に値を書き込むものがあります。これらの関数を使用する際も、コントローラの複数の変数に値を一度に書き込むことでシェイプスクリプトの実行時間を短縮することができます。
コントローラの変数に値を書き込む関数の一覧を以下に示します。

コントローラの変数に書き込む関数一覧
SetClampStatus
DetectPartCollision
DetectMechanicalComponentsCollision

これらの関数の引数「setVariableList」に CreateSetVariableList()関数の戻り値を引数に設定することで、書き込みたいコントローラの変数名と値が「setVariableList」に追加されます。
「setVariableList」を処理の最後で SetVariableValues()関数の引数に設定することで、コントローラの変数に値を一度に書き込むことができます。これにより、コントローラの変数に書き込みを行う際の通信処理を省略することができます。通信処理の省略によって、シェイプスクリプトの処理時間を短縮することができます。

CreateSetVariableList()関数の戻り値を関数の引数に設定して、コントローラの変数に値を一度に書き込むシェイプスクリプトの例を、以下に示します。

```
110         public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
111             if (this.IsInitialized == false) {
112                 DefaultFunctions.InitializeObject(this, renderInfoList);
113                 return ;
114             }
115 
116             var getVariableList = DefaultFunctions.CreateGetVariableList();
117             DefaultFunctions.AddToGetVariableList("boolVar1", "BOOL", getVariableList);
118             DefaultFunctions.AddToGetVariableList("boolVar2", "BOOL", getVariableList);
119             DefaultFunctions.AddToGetVariableList("boolVar3", "BOOL", getVariableList);
120 
121             var variableValues = DefaultFunctions.GetVariableValues(this, "new_Controller_0", getVariableList);
122 
123             var setVariableList = DefaultFunctions.CreateSetVariableList();
124 
125             ICadData partModel1 = (ICadData) ace["/ApplicationManager0/partModel1"];
126             ICadData partModel2 = (ICadData) ace["/ApplicationManager0/partModel2"];
127             ICadData table = (ICadData) ace["/ApplicationManager0/table"];
128             MechanicalDataModel mechanicalDataModel = (MechanicalDataModel) ace["/ApplicationManager0/MechanicalModel"];
129 
130             DefaultFunctions.LoadPart(this, "new_Controller_0", "boolVar1", partModel1, string.Empty, new Transform3D(0, 0, 0), table, renderInfoList, variableValues);
131             DefaultFunctions.LoadPart(this, "new_Controller_0", "boolVar2", partModel2, string.Empty, new Transform3D(100, 0, 0), table, renderInfoList, variableValues);
132 
133             DefaultFunctions.DetectPartCollision(this, partModel1, mechanicalDataModel, 1, "new_Controller_0", "boolVar4", renderInfoList, setVariableList);
134             DefaultFunctions.DetectPartCollision(this, partModel2, mechanicalDataModel, 1, "new_Controller_0", "boolVar5", renderInfoList, setVariableList);
135 
136             if(setVariableList.Count > 0){
137                 DefaultFunctions.SetVariableValues(this, "new_Controller_0", setVariableList);
138             }
139         }
```

この例では、123 行目の CreateSetVariableList()関数によって書き込み変数リスト「setVariableList」を作成しています。「setVariableList」は、133～134 行目の DetectPartCollision()関数の引数に設定しています。それぞれの DetectPartCollision()関数の中で、「setVariableList」にコントローラの変数「boolVar4」と「boolVar5」に設定する値が追加されます。
136 行目では、「setVariableList」に値が設定されているかどうかをチェックしています。値が設定されていれば、137 行目でコントローラの変数に値を一度に書き込みます。これにより、各 DetectPartCollision()関数内でネットワークを介さずにコントローラの変数に値を高速で書き込みできるようになります。

バージョン関連情報

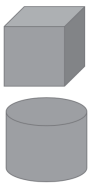
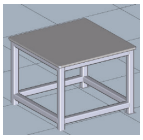
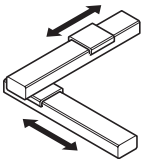
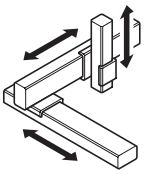
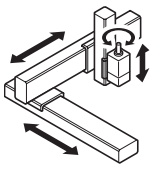
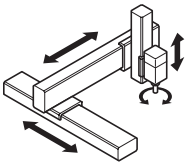
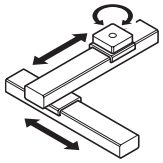
引数「setVariableList」は、Sysmac Studio Ver1.43 以前では使用できません。

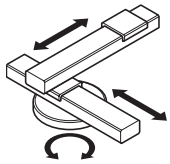
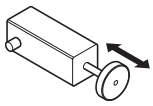
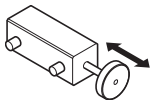
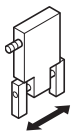
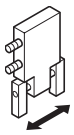
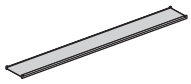
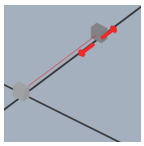
A-4　ワーク動作設定

A-4-1　3D 形状データごとのワーク動作設定可否一覧

ワーク動作設定は、3D 形状データごとに設定可能なものが異なります。以下のとおりです。

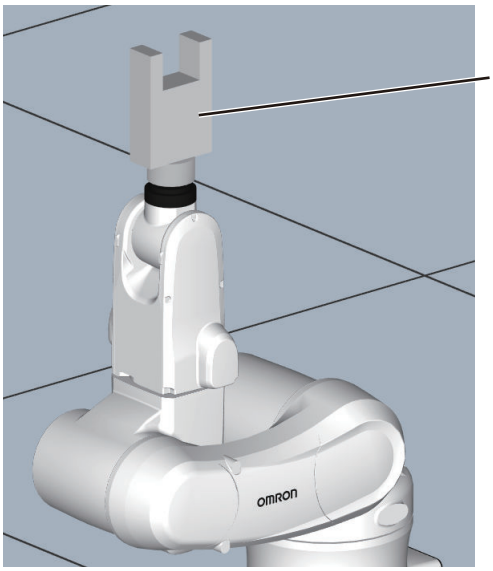
○: 設定可能、×: 設定不可

カテゴリ	イメージ	機構	ワーク動作設定						
			生成	搬送	把持	押出	センサ	物理シミュレーション	消去
基本図形		直方体、円柱	○	○	○	○	○	○	○
CAD データ		CAD データ	○	○	○	○	○	○	○
メカニカル機構		直交ロボット(XY)	×	×	○	○	○	○	×
		直交ロボット(XYZ)	×	×	○	○	○	○	×
		XYZ ステージ+回転軸（回転軸上方向）	×	×	○	○	○	○	×
		XYZ ステージ+回転軸（回転軸下方向）	×	×	○	○	○	○	×
		XY テーブル(XYθ)	×	×	○	○	○	○	×

カテゴリ	イメージ	機構	ワーク動作設定						
			生成	搬送	把持	押出	センサ	物理シミュレーション	消去
		XY テーブル(θXY)	×	×	○	○	○	○	×
		エアシリンダ（シングルソレノイドタイプ）	×	×	○	○	○	○	×
		エアシリンダ（ダブルソレノイドタイプ）	×	×	○	○	○	○	×
		ロボットツール（平行開閉型 2 爪タイプチャック／シングルソレノイドタイプ）	×	×	○	○	○	○	×
		ロボットツール（平行開閉型 2 爪タイプチャック／ダブルソレノイドタイプ）	×	×	○	○	○	○	×
		コンベア	×	○	×	×	○	○	×
センサ		仮想ワーク検出センサ	×	×	×	×	○	○	○

参考

ロボットおよびカスタムメカニカル機構にはワーク動作設定はありませんが、ワークに接する箇所に 3D 形状データ（例: ロボットハンド）を設定し、3D 形状データにワーク動作設定を設定することで目的の動作を設定することができます。



ロボットハンドの
ワーク動作設定に
「把持」を設定

A-4-2　シェイプスクリプトとワーク動作設定の同時実行を実現する方法

「生成」ワーク動作設定により生成されたワークをシェイプスクリプトで制御することが可能です。また、シェイプスクリプトで生成したワークをワーク動作設定で制御することも可能です。それぞれの設定方法は以下のとおりです。

「生成」ワーク動作設定により生成されたワークをシェイプスクリプトで制御する方法

「生成」ワーク動作設定により生成され 3D 表示画面に表示されるワークは、シェイプスクリプトで 3D 表示画面上のすべてのオブジェクトを意味する world.Objects の中のオブジェクトを RuntimePart 型にキャストし、GetComponent(“Hierarchy”)メソッドを使用して必要なオブジェクトを抽出し、プロパティを更新することでシェイプスクリプトで動作させることが可能です。
以下の例は、シェイプスクリプトの Render 関数の中でワーク動作設定により生成されるワークを Z 方向に 1mm 移動させるコードになります。

```
9  using Ace.Server.Core.Sim3d.PartDetection;
10 using Ace.Server.Core.Sim3d.MechanicalComponent;
11 using Ace.Server.Core.Sim3d.Behaviors.Runtime;
12 using Ace.Visualization.Hierarchies;
13 using Ace.Server.Xpert.PackXpert;
```

Using文で Ace.Server.Core.Sim3d.Behaviors.RuntimeとAce.Visualization.Hierarchiesの使用を宣言（新規でシェイプスクリプト登録時には不要です）

```
65      public override void Render(IEnumerable<ShapeRenderInfo> renderInfoList) {
66          if (this.IsInitialized == false) {
67              DefaultFunctions.InitializeObject(this, renderInfoList);
68
69              foreach(var obj in world.Objects){
70                  var runtimeObject = obj as RuntimePart;
71                  if (runtimeObject == null) {
72                  continue;
73                  }
74
75                  var hierarchyComponent = runtimeObject.GetComponent("Hierarchy") as HierarchyComponent;
76                  if (hierarchyComponent == null) {
77                  continue;
78                  }
79
80                  hierarchyComponent.OffsetFromParent = hierarchyComponent.OffsetFromParent.Shift(0, 0, 1);
81                  }
82
83              return;
84          }
85      }
```

RuntimePart型のオブジェクトをworld.Objectから検索してruntimeObjectに代入

上の処理で特定されたワークに相当するオブジェクトhierarchyComponentをZ方向に1mm移動

RuntimeObjectの中にワークオブジェクトが持つ属性である"Hierarchy"をキーに取得し、hierarchyComponentに代入

シェイプスクリプトで生成したワークをワーク動作設定で制御する方法

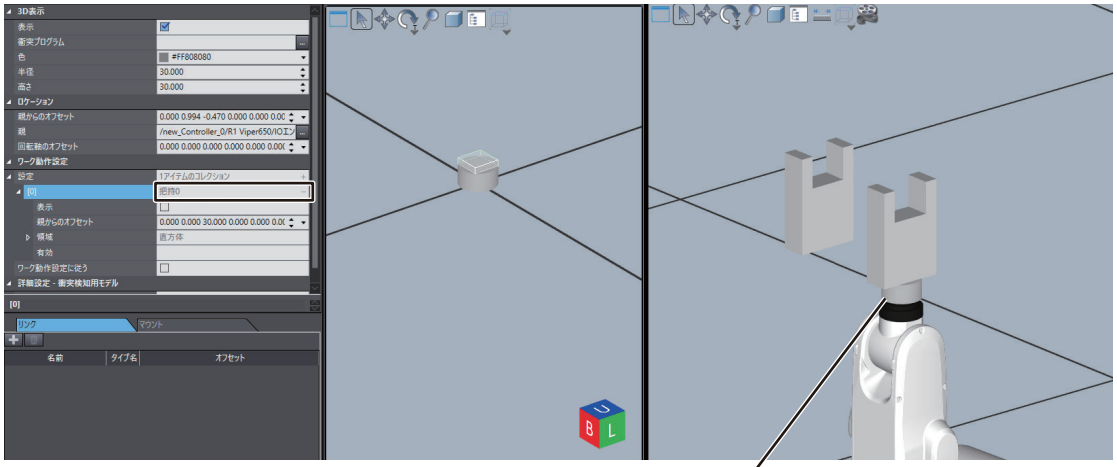
シェイプスクリプトで生成されるワークは、EnableCanBeAffectedByBehaviors()メソッドを呼び出すことで、ワーク動作設定により制御することができます。以下の例は、シェイプスクリプトのCreateRenderInfo()の中で、マルチビューエクスプローラで Part という名前の円柱をワークとしてrenderInfoList に登録する処理の中で、renderInfoList から renderInfo0 という変数に代入し、EnableCanBeAffectedByBehaviors()メソッドを呼び出し、ワーク動作設定により制御できるようにするコードです。また、DisableCanBeAffectedByBehaviors()メソッドを呼び出すことで、ワーク動作設定による制御を受けないように変更することができます。デフォルトではワーク動作設定による制御は受けないようになっています。

```
46      public override IEnumerable<ShapeRenderInfo> CreateRenderInfo() {
47          this.IsInitialized = false;
48
49          // Create the Shapes for rendering once in the constructor
50          List<ShapeRenderInfo> renderInfoList = new List<ShapeRenderInfo>();
51
52          // Please assign your shape.
53          ICylinder Part = (ICylinder) ace["/ApplicationManager0/Part"];
54          if (Part != null) {
55              DefaultFunctions.CreateObject(Part, "CollisionSource", "Group1", renderInfoList);
56
57              var renderInfo0 = renderInfoList[0];
58              renderInfo0.EnableCanBeAffectedByBehaviors();
59          }
60
61          return renderInfoList;
62      }
63
```

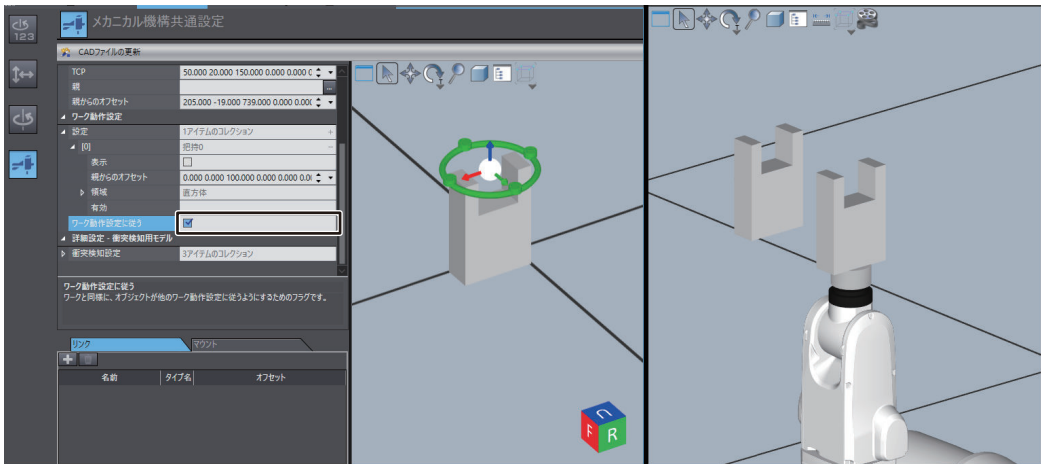
A-4-3　ロボットハンドのツールチェンジャーを実現する方法

ロボットハンドのツールチェンジャーをワーク動作設定で実現することが可能です。実現する手順は以下のとおりです。

1　ツールの装着を行うマスタープレートに「把持」ワーク動作設定を追加します。



2 **装着するツールの編集画面を開き、「ワーク動作設定に従う」にチェックを入れます。**



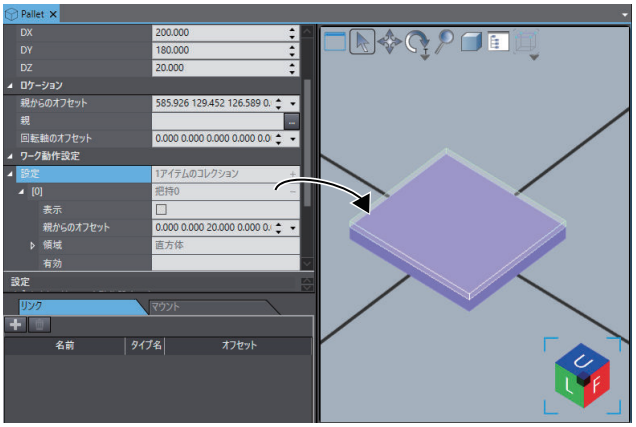
ツールがマスタープレートの「把持」ワーク動作設定の領域に入り、かつ「把持」ワーク動作設定が有効になった際、ツールがマスタープレートに追従して動作します。

A-4-4　パレットを実現する方法

ワーク動作設定を用いて、パレットを実現することができます。実現する手順は以下のとおりです。

1 **マルチビューエクスプローラにパレットとなる CAD データを追加します。**

2 **手順 1 で追加した CAD データに「把持」ワーク動作設定を追加します。**



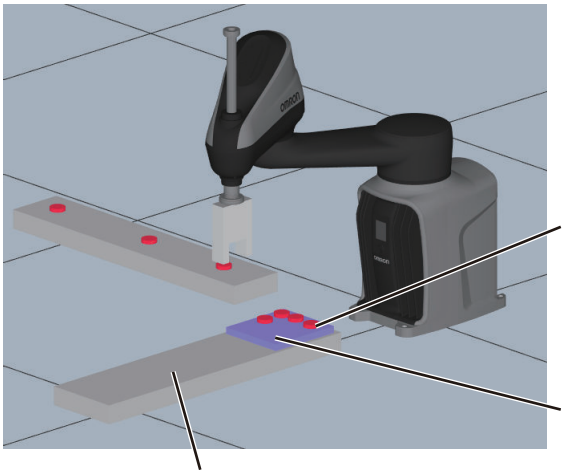
3 **手順 1 とは別の 3D 形状データに「生成」ワーク動作設定を追加し、1)の CAD データを生成するように設定します。**

4 **ワークを生成する「生成」ワーク動作設定を追加します。**

5 **ワークの把持やパレットの搬送など、その他の設定を行います。**

6 **ワーク動作設定によるシミュレーションを実行します。**

「生成」ワーク動作設定により生成された CAD データは、「把持」ワーク動作設定を持ちます。そのため、ワークがパレットの「把持」ワーク動作設定の領域に入り、かつ「把持」ワーク動作設定が有効の間、パレットがワークを「把持」することができます。
また、パレットが「生成」ワーク動作設定により生成されているため、パレットを「搬送」ワーク動作設定や「把持」ワーク動作設定により、動作させることができます。この場合、パレットが「把持」しているワークもパレットに追従して動作します。



パレットに接したワークはパレットに「把持」され、パレットの移動に追従します。

「生成」ワーク動作設定で生成されたパレットは、「把持」動作設定を継承しワークを把持します。

「搬送」ワーク動作設定によりパレットを搬送します。

I

索引

索引

A

AddToGetVariableList....A-54
AddToSetVariableList....A-56

C

ClampPallet....A-24
ClampPalletBySignal....A-26
ClampPart....A-23
ClampPartBySignal....A-25
ClampPartOnPallet....A-24
ClampPartOnPalletBySignal....A-27
CreateGetVariableList....A-53
CreateMultipleObjects....A-8
CreateMultipleObjectsForCameraLatch....A-8
CreateMultipleObjectsForNonCameraLatch....A-11
CreateObject....A-7
CreateSetVariableList....A-56

D

DetectMechanicalComponentsCollision....A-35
DetectPartCollision....A-34

G

Get**ArrayVariable....A-51
Get**Variable....A-39
GetBoolArrayVariable....A-42
GetBoolVariable....A-38
GetByteArrayVariable....A-41
GetCollidingSourceNames....A-37
GetCurrentControllerTime....A-58
GetDwordArrayVariable....A-46
GetIntegerArrayVariable....A-43
GetIntegerVariable....A-39
GetLrealArrayVariable....A-49
GetLwordArrayVariable....A-47
GetRealArrayVariable....A-48
GetStringArrayVariable....A-50
GetValueFromVariableValueList....A-55
GetVariableValues....A-54
GetWordArrayVariable....A-45

I

InitializeMechanicalConveyor....A-22

L

LoadPallet....A-14
LoadPart....A-12
LoadPartOnPallet....A-15

M

MoveObjectOnBelt....A-21
MoveObjectOnMechanicalConveyor....A-22
MovePalletOnPackManagerSensorLatchBelt....A-59
MovePartOnPackManagerSensorLatchBelt....A-58

P

PushPallet....A-19
PushPart....A-19

R

ReleasePallet....A-29
ReleasePalletBySignal....A-32
ReleasePart....A-28
ReleasePartBySignal....A-31
ReleasePartOnPallet....A-30
ReleasePartOnPalletBySignal....A-33

S

Set**ArrayVariable....A-52
Set**Variable....A-40
SetBoolArrayVariable....A-43
SetBoolVariable....A-38
SetByteArrayVariable....A-42
SetClampStatus....A-36
SetDwordArrayVariable....A-46
SetIntegerArrayVariable....A-44
SetIntegerVariable....A-39
SetLrealArrayVariable....A-49
SetLwordArrayVariable....A-47
SetNextStep....A-34
SetPackManagerPartPosition....A-60
SetRealArrayVariable....A-48
SetStringArrayVariable....A-50
SetVariableValues....A-57
SetWordArrayVariable....A-45
StartStopwatch....A-62
StopStopwatch....A-62
Sysmac Studio オプション....2-3
Sysmac Studio オプションライセンス....1-3

U

UnloadCollidingPallet....A-18
UnloadCollidingPart....A-17
UnloadPallet....A-17
UnloadPart....A-16
UpdatePartDataFromPackManager....A-61
UpdatePartDataOnPalletFromPackManager....A-61

オムロン株式会社 インダストリアルオートメーションビジネスカンパニー

製品に関するお問い合わせ先

お客様相談室

フリー通話 **0120-919-066**（クイック　オムロン）

携帯電話の場合、
055-982-5015（有料）をご利用ください。

受付時間：9:00～17:00（土・日・12/31～1/3を除く）

オムロンFAクイックチャット
www.fa.omron.co.jp/contact/tech/chat/
技術相談員にチャットでお問い合わせいただけます。（I-Webメンバーズ限定）

受付時間：平日9:00～12:00 / 13:00～17:00（土日祝日・年末年始・当社休業日を除く）
※受付時間、営業日は変更の可能性がございます。最新情報はリンク先をご確認ください。

その他のお問い合わせ：納期・価格・サンプル・仕様書は貴社のお取引先、または貴社担当オムロン販売員にご相談ください。オムロン制御機器販売店やオムロン販売拠点は、Webページでご案内しています。

オムロン制御機器の最新情報をご覧いただけます。緊急時のご購入にもご利用ください。 **www.fa.omron.co.jp**

●本誌に記載のない条件や環境での使用、および原子力制御・鉄道・航空・車両・燃焼装置・医療機器・娯楽機械・安全機器、その他人命や財産に大きな影響が予測されるなど、特に安全性が要求される用途に使用される際には、当社の意図した特別な商品用途の場合や特別の合意がある場合を除き、当社は当社商品に対して一切保証をいたしません。
●本製品の内、外国為替及び外国貿易法に定める輸出許可、承認対象貨物（又は技術）に該当するものを輸出（又は非居住者に提供）する場合は同法に基づく輸出許可、承認（又は役務取引許可）が必要です。

オムロン商品のご用命は

Man. No. SBCA-482L　2025年4月現在

©OMRON Corporation 2020-2025 All Rights Reserved.
お断りなく仕様などを変更することがありますのでご了承ください