



AX Portal ユーザーズガイド

翻訳版

©Seiko Epson Corporation 2026

Rev. 2.0

JAM265S8834F

目次

商標.....	7
ご注意	7
1.はじめに	8
1.1 プログラミングとロボットの制御	9
1.2 グラフィカルユーザーインターフェース (GUI)	9
1.3 ソフトウェアモジュール	9
1.3.1 TCP 接続モジュール	9
1.3.2 REST 接続モジュール	10
1.3.3 ROS 接続モジュール	10
1.3.4 Modbus 接続モジュール	10
1.3.5 パネルインターフェースモジュール	10
1.3.6 Cognex カメラモジュール	10
2.管理インターフェース (GUI)	11
2.1 概要.....	12
2.1.1 表示言語	13
2.1.2 ダッシュボード.....	13
2.2 ユーザー管理.....	13
2.2.1 ユーザーレベル	13
2.2.2 ログインページ	13
2.2.3 自分のアカウントを管理	14
2.2.4 ほかのアカウントを管理	15
2.3. システム設定	16
2.3.1. ネットワーク設定	16
2.3.2. 時刻設定	17
2.3.3 SSL 証明書.....	17
2.4 ソフトウェアアップデート	19
2.5 ヘルプとサポート	21
2.6 セキュア認証	22

2.6.1 SSL 証明書のエクスポート	22
2.6.2 ユーザーログイン	22
2.6.3 接続登録	23
3.USB エクスポートとリカバリーアップデート	24
3.1 概要.....	25
3.2 USB エクスポート.....	25
3.3 リカバリーアップデートを実施する.....	25
4.デスクトップインターフェース (GUI).....	26
4.1 概要.....	27
4.2 メインウィンドウ	28
4.2.1 メニューの概要	29
4.2.2 コックピット	30
4.2.3 アクティブ制御.....	31
4.2.4 プログラムの状態	32
4.2.5 動作モード	34
4.2.6 ジョグ制御.....	35
4.2.7 動作制御	36
4.3 ブロック/データベース	39
4.4 プロジェクトメニュー.....	40
4.5 構成メニュー	41
4.5.1 データベース (ダウンロード/アップロード)	41
4.5.2 I/O の設定	42
4.5.3 ツールの設定.....	45
4.5.4 ツールエディター	45
4.5.5 ロボットの起動.....	48
4.6 ティーチンメニュー	49
4.6.1 ポーズエディター	51
4.6.2 パスエディター	53
4.6.3 グリッドエディター	55
4.7 アプリケーションメニュー	58

4.7.1 アプリケーションエディター	59
4.7.2 アプリケーションコード	59
4.7.3 アプリケーション設定	61
4.7.4 アプリケーション変数	63
4.7.5 共有変数、グローバル変数	65
4.7.6 フォルダーエディター	65
4.7.7 ブロックプログラミング	66
4.8 I/O メニュー	72
4.9 シミュレーターメニュー	73
4.10 ハンドガイド	74
4.11. システムモニター	75
4.11.1 ロボットの状態	75
4.11.2 アプリケーションの出力	75
4.11.3 アプリケーションログ	75
4.11.4 メッセージ履歴	75
4.12 ヘルプメニュー	76
4.12.1 ロボット情報	76
4.12.2 キーボードショートカット	76
4.12.3 コードのドキュメント	76
4.12.4 オープンソースライセンス	76
4.12.5 お問い合わせ先	76
4.13 設定メニュー	77
4.13.1. GUI 設定	78
4.13.2 コックピットボタンのカスタマイズ	78
4.13.3 起動設定	79
4.13.4 動作中の挙動	79
4.14 安全設定メニュー	83
4.14.1 ユーザーインターフェース	83
4.14.2 安全設定	85
5. パネルインターフェース (GUI)	91

5.1 概要.....	92
5.2 パネルインターフェース用アプリケーションの設定	93
5.3 コックピットと設定.....	95
5.4 アプリケーションの制御.....	96
6.キネマティクス定義.....	97
6.1 座標.....	98
6.1.1 航法角	98
6.1.2 座標系.....	99
6.2 姿勢設定	100
6.3 ポーズ	100
6.4 ポイント.....	100
6.5 パス.....	100
6.6 セグメント	101
6.7 グリッド.....	101
7.接続モジュール	102
7.1 接続・認証・アクティブ制御.....	103
7.1.1 接続チャンネル.....	103
7.1.2 認証.....	103
7.1.3 アクティブ制御ステーション	104
7.2 TCP 接続モジュール	104
7.2.1 接続の確立.....	104
7.2.2 コマンドの送信.....	106
7.2.3 設定.....	107
7.2.4 メッセージプロトコル - コマンドの受信	108
7.2.5 メッセージプロトコル - コマンドの送信	112
7.3 REST 接続モジュール	124
7.3.1 接続の確立.....	124
7.3.2 コマンドの送信.....	125
7.3.3 設定.....	125
7.4 ROS ハンドラーモジュール	126

7.4.1 インストール	126
7.4.2 設定.....	127
7.4.3 Cyclone DDS によるネットワーク検出.....	127
7.4.4 接続の確立.....	130
7.4.5 ROS でロボットデータにアクセスする	131
7.4.6 スクリプトでの ROS 2 機能.....	131
7.4.7 アクションを呼び出す	131
7.4.8. 認証設定	132
7.5 Modbus 接続モジュール	133
7.5.1 概要.....	133
7.5.2 設定.....	133
7.5.3 レジスター書き込み (概要)	134
7.5.4 レジスターへの書き込み (詳細)	136
7.5.5 レジスターの読み取り	141
7.5.6 エラーレジスターマッピング	145
7.5.7 Python コード例	146
8.Cognex カメラモジュール	148
8.1 概要.....	149
8.2 カメラの設定	149
8.3 カメラアプリケーション	149

商標

Cognex® および In-Sight Explorer® は、米国およびその他の国における Cognex Corporation の商標または登録商標です。

Google Chrome™ および Chrome™ は、Google LLC の商標です。

Linux® は、米国およびその他の国における Linus Torvalds の登録商標です。

Apple®, macOS® および Safari® は、米国およびその他の国や地域における Apple Inc. の登録商標です。

Microsoft®, Windows®, Microsoft Edge®, Microsoft Visual C++® は、米国およびその他の国における Microsoft Corporation またはその関連会社の商標(または登録商標)です。

Mozilla および Firefox® は、米国およびその他の国における Mozilla Foundation の商標です。

補足事項: 本書で使用しているその他の製品名は、識別のためだけに記載しており、それぞれの所有者の商標である場合があります。エプソンは、それらの商標に関する一切の権利を放棄します。

すべてのスクリーンショットは、説明および教育目的のみに使用しています。商標権者との提携、または承認を意図または示唆するものではありません。

ご注意

本取扱説明書の一部、または全部を無断で複製や転載をすることはできません。

本書に記載の内容は、将来予告なく変更することがあります。

本書の内容について、誤りや、お気づきの点がありましたら、ご連絡くださいますようお願いいたします。

1.はじめに

1.1 プログラミングとロボットの制御

AX Portal は、エプソン人協働ロボット AX シリーズと直感的に通信できるソフトウェアです。AX Portal を使用すると、ロボットの操作や制御、**Python 3.13** を使用したプログラムアプリケーションを作成できます。Python の基本的なプログラミング機能は、オンラインで確認できます。基本的なコード例は、組み込みのコードドキュメントにも記載されています。

Python の豊富な基本機能に加え、AX Portal では AX ロボットの制御、メンテナンス、通信に特化した多くの機能を使用できます。AX Portal で使用可能なスクリプト関数の一覧は、組み込みのコードドキュメントを参照してください。

1.2 グラフィカルユーザーインターフェース (GUI)

AX Portal の GUI はウェブアプリケーションとして設計されています。一般的なウェブブラウザ上で実行でき、端末を選ばずに使用できます。以下のブラウザでの動作をサポートしています。

- Google Chrome (バージョン 147 以上)
- Mozilla Firefox (バージョン 150 以上)
- Apple Safari (バージョン 26 以上)
- Microsoft Edge (バージョン 147 以上)

ロボットと端末が同じネットワーク内にあるか、直接通信できる場合は、端末に専用ソフトウェアをダウンロードする必要はありません。AX Portal には、PC でのプログラミングに最適化されたデスクトップインターフェースに加え、事前にプログラム済みのロボットを操作するのに必要な機能のみを搭載したパネルインターフェースが用意されています。GUI とその機能についての詳細は、本マニュアルで後述します。

1.3 ソフトウェアモジュール

本書で説明する AX Portal の基本機能に加え、制御機能を拡張する組み込みソフトウェアモジュールが AX ロボットに搭載されています。以下で説明する接続モジュールを使うと、AX ロボットに遠隔から接続し、ロボットの制御や監視ができます。

1.3.1 TCP 接続モジュール

TCP 接続モジュールは、ロボットとの代替通信手段として使用できます。通常、ロボットはウェブブラウザの GUI を通じて制御します。ロボットを独自インターフェースや他の機器から制御する場合は、TCP コマンドを直接受信します。TCP 接続モジュールを使用すれば、ウェブブラウザで可能なすべての制御を TCP で実現できます。

1.3.2 REST 接続モジュール

REST 接続モジュールは、ロボットとの代替通信手段として使用できます。通常、ロボットはウェブブラウザの GUI を通じて制御します。ロボットを独自インターフェースや他の機器から制御する場合は、REST コマンドを直接受信します。REST 接続モジュールを使用すれば、ウェブブラウザで可能なすべての制御を REST で実現できます。

1.3.3 ROS 接続モジュール

ROS 接続モジュールは、ロボットを ROS2 環境に統合するための機能を提供します。ROS2 ネットワーク内の他のクライアントは、ロボットに制御や監視のアクションを送信できます。

1.3.4 Modbus 接続モジュール

Modbus 接続モジュールは、ロボットとの代替通信手段として使用できます。Modbus TCP プロトコルを使用し、標準 Modbus レジスターの読み取りと書き込みができます。PLC を使ってロボットを直接制御する場合は、このモジュールを使用してください。ロボットの電源オン、アプリケーションの実行、出力のトリガーなどの主要機能は、Modbus モジュールで実現できます。

1.3.5 パネルインターフェースモジュール

パネルインターフェースモジュールは、ロボットのオペレーター向けに機能を限定した GUI を提供します。このインターフェースは、アプリケーションの起動や監視ができ、プログラマーは利用シーンに合わせて起動や監視の内容を変更できます。パネルインターフェースはタッチスクリーンに対応し、最新のモバイル機器で使うアプリのような構成です。そのため、技術的な知識がなくても直感的に操作できます。

1.3.6 Cognex カメラモジュール

Cognex カメラモジュールは、Cognex カメラと AX Portal の通信インターフェースを提供します。通常、カメラはエンドエフェクターに取り付けるか、ロボットの動作領域を監視するために固定設置します。このモジュールは Cognex カメラの全機能をサポートし、視覚検査を柔軟にカスタマイズできます。

2.管理インターフェース (GUI)

2.1 概要



図 2.1.a: 管理インターフェースの画面例

AX Portal の管理インターフェースは、ユーザーアカウントの管理や、ロボットが動作する OS に関連する初期設定 (ネットワーク設定や時刻設定など) を変更できる GUI です。これらの設定は、プログラマーがロボットの制御アプリケーションを作成する前に、ロボットの設置時に一度のみ設定します。このインターフェースには、ウェブブラウザ (例: Google Chrome) でアクセスできます。ブラウザのアドレスバーにロボットの IP アドレスを入力すると、<https://<IP アドレス>/admin> に転送されます。正しい IP アドレスについては、"ロボットコントローラー RC-A101 マニュアル" を参照してください。



図 2.1.b: URL を入力して管理インターフェースに接続する

本章では、管理インターフェースのメニューや項目について説明します。

2.1.1 表示言語

表示言語を変更するには、画面左下の[言語]ドロップダウンリストをクリックし、表示したい言語を選択します。言語設定はブラウザーに記憶され、セッションやインターフェースが変わっても保持されます。

2.1.2 ダッシュボード

ログインすると、管理インターフェースのダッシュボードからサブメニューや AX Portal の他のインターフェースへ移動したり、ログアウトできます。ダッシュボードには、ユーザーレベル (管理者、安全管理者、プログラマー、オペレーター) に応じて、アクセスできる項目のみが表示されます。ほかのインターフェースに切り替えられないときは、ソフトウェアのアップデートが実行されているなど、AX Portal が正しく動作していない可能性があります。

2.2 ユーザー管理

2.2.1 ユーザーレベル

権限のないユーザーによるロボットの設定、プログラミング、操作を防ぐため、複数の認証レベルを導入しています。各ユーザーのユーザーレベルは、管理者のみ設定できます。認証レベルの高い順に、以下のユーザーレベルがあります。

- **管理者**は、ロボットのすべての制御インターフェースと機能にアクセスできます。また、ほかのユーザーの管理や、OS の重要なシステム設定にアクセスできます。
- **プログラマー**は、ロボットのすべての制御インターフェースと機能にアクセスできます。オペレーターが実行するプログラムアプリケーションを作成できます。管理インターフェースでは、プログラマーは自身のアカウント詳細へのアクセスとインターフェースの切り替えのみができます。
- **安全管理者**は、ロボットの安全設定を変更する権限を持ったプログラマー (詳細は上記参照) です。ほかのユーザーは、安全設定の確認のみできます。
- **オペレーター**は、基本のオペレーターインターフェースのみアクセスできます。ロボットを操作できますが、設定やプログラムは変更できません。管理インターフェースでは、オペレーターは自身のアカウント詳細のみアクセスできます。

2.2.2 ログインページ

管理者アカウントの有無によって、ログインページの表示内容が異なります。

- 管理者アカウントが作成されている場合、ユーザー名とパスワードを入力する通常のログインページが表示されます。入力中のパスワードは非表示ですが、パスワード欄の[パスワードを表示]ボタンをクリックすると表示されます。
- 管理者アカウントが作成されていない場合、作成を求められます。管理者アカウントを作成するまで、他の操作はできません。詳細は、"6 軸ロボット AX6 クイックスタート / セットアップマニュアル"を参照してください。ここでは必須項目のみ設定できます。アカウント

トのすべての設定を確認、変更するには、[自分のアカウントを管理]メニューを開きます。

ログアウトすると、ログインページに戻ります。

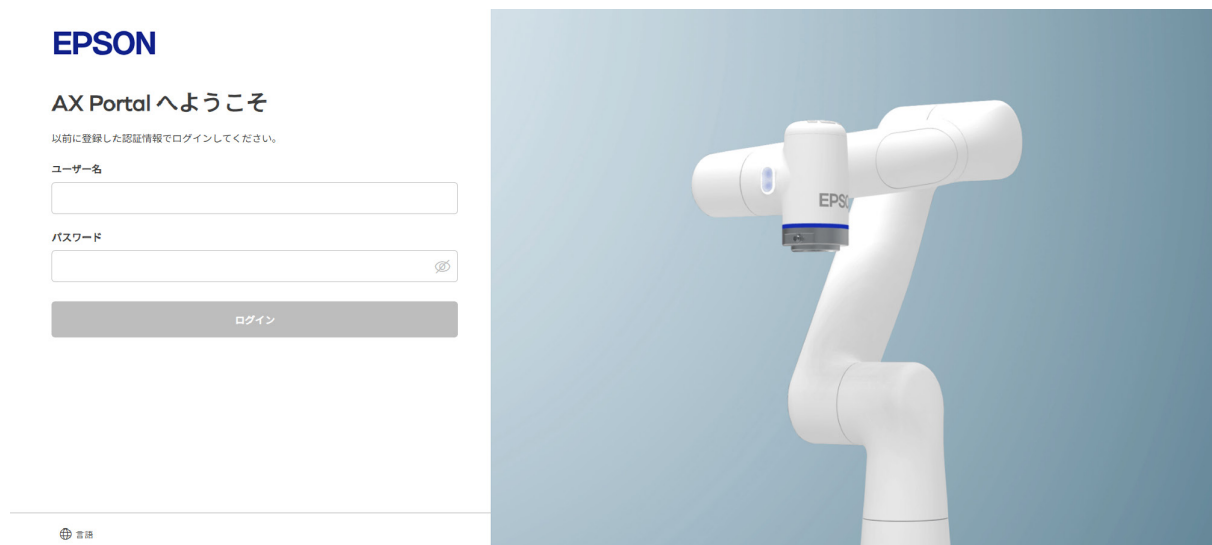


図 2.2.2: ログインページ

2.2.3 自分のアカウントを管理

[自分のアカウントを管理]メニューでは、ユーザー名、パスワード、プロフィール写真、起動時のインターフェースなどの個人設定を変更できます。

- [フルネーム]は、画面やメッセージ内での表示に使用されます。姓と名を入力する必須項目です。独自の命名ルールを使用することもできます。
- [ユーザー名]は、ユーザーアカウント作成時に管理者が設定し、アカウント作成後は変更できません。
- [起動時のインターフェース]は、ログイン後に表示するインターフェースを設定します。ユーザーレベルごとに、異なる起動時のインターフェースがあらかじめ設定されていますが、ここで変更できます。
- [パスワード変更]ボタンを押すと、アカウントのパスワードを変更できます。パスワード設定のエラーを防ぐため、現在のパスワードを再入力し、新しいパスワードを2回入力します。パスワードを忘れた場合、管理者がパスワードを変更できます。パスワードはプレーンテキストでは保存されないため、第三者がパスワードを取得することはできません。
- [プロフィール画像]の設定は任意です。プロフィール画像を設定しない場合、デフォルトのユーザーアイコンが表示されます。

EPSON
Admin Interface

自分のアカウントを管理

manual (user icon) ▼

フルネーム
manual

ユーザー名
manual

ユーザーレベル
管理者

起動時のインターフェース
管理インターフェース

パスワード変更

アカウント削除

キャンセル 保存

プロフィール画像の変更

図 2.2.3: 自分のアカウントを管理ページ

2.2.4 ほかのアカウントを管理

[ほかのアカウントを管理]メニューでは、現在登録されているすべてのユーザーの概要を確認できます。以下の操作ができます。

- あらかじめ設定されたユーザーレベル、初期パスワードで新規ユーザーを作成できます。初期パスワードは、初回ログイン後に必ず変更してください。
- ユーザー名以外のほとんどのプロパティを変更できます。ユーザー名は変更できません。
- データベースからユーザーを完全に削除できます。削除されたユーザーは、ロボットの制御機能を使用できなくなります。

EPSON
Admin Interface

ほかのアカウントを管理

manual (user icon) ▼

アカウント詳細

	フルネーム ↓	ユーザー名	ユーザーレベル		
(user icon)	User	user	管理者	✎	🗑
(user icon)	safety1	safety1	安全管理者	✎	🗑
(user icon)	user1	user1	安全管理者	✎	🗑
(user icon)	safe	safe	安全管理者	✎	🗑

新規アカウント追加

図 2.2.4: ほかのアカウントを管理ページ

2.3. システム設定

2.3.1. ネットワーク設定

[ネットワーク設定]メニューでは、ロボットのイーサネット接続に関する設定ができます。現在使用中の接続設定を変更する (たとえば、イーサネット経由で接続していて、イーサネットの IP アドレスを変更する) と、接続が切断されます。無効な設定を入力した場合、ロボットに接続できなくなる可能性があります。有効な接続が 1 系統しかない場合は、設定を変更する前に、2 系統の接続を確保することを推奨します。

イーサネット設定では、ロボットの IP アドレスを動的に取得するか、手動で設定するかを選択できます。IP アドレスを動的に取得する設定を選択した場合、取得した IP アドレスが画面に表示されます。手動での設定を選択した場合、IP アドレス、サブネットマスクを必ず入力します。必要に応じて、ゲートウェイアドレスと最大 2 つの DNS サーバーを入力できます。

The screenshot displays the 'EPSON Admin Interface' with a sidebar menu on the left containing options like 'デスクトップインターフェースに移動', 'パネルインターフェースに移動', '自分のアカウントを管理', 'ほかのアカウントを管理', 'ネットワーク設定' (selected), '時刻設定', 'SSL証明書', 'ソフトウェアアップデート', 'ヘルプとサポート', and '言語'. The main content area is titled 'ネットワーク設定' and includes a 'manual' link with a user icon. Under the 'イーサネットIPv4' section, there are radio buttons for '自動設定のIPアドレス' (selected) and '手動設定のIPアドレス'. Below these are input fields for 'IPアドレス' (192.168.91.22), 'サブネットマスク' (255.255.254.0), 'ゲートウェイ' (192.168.90.1), 'DNSサーバー1', and 'DNSサーバー2'. A 'ネットワーク情報' section contains a paragraph explaining that the second Ethernet port is used for AX Portal access, with a fixed IP of 10.0.0.203, and that manual IP entry is an alternative to connecting via <Robot-Serial> local. At the bottom are 'キャンセル' and '保存' buttons.

図 2.3.1: ネットワーク設定ページ

2.3.2. 時刻設定

[時間設定]メニューでは、ロボットの OS で使用する現在の日時とタイムゾーンが設定できます。正しい時刻と日付を入力してください。正しい時刻を設定すると、ログ解析やトラブルシューティングに使用でき、時間に関する機能が正しく動作します。時間設定を手動で変更する場合、[保存]ボタンを押したタイミングで設定した時間が適用されます。タイムゾーンを変更するときは、大陸名 (例: Europe)、国コード (例: GB)、都市名 (例: London)、またはそれらの組み合わせ (例: eur lon → Europe/London)を使用して、タイムゾーンのリストを絞り込みできます。

キーポイント: ロボットがネットワーク接続されている場合、NTP (Network Time Protocol)同期を有効にできます。

The screenshot displays the '時刻設定' (Time Setting) page within the EPSON Admin Interface. The left sidebar contains navigation links such as 'デスクトップインターフェースに移動' (Move to Desktop Interface), 'パネルインターフェースに移動' (Move to Panel Interface), '自分のアカウントを管理' (Manage My Account), 'ほかのアカウントを管理' (Manage Other Accounts), 'ネットワーク設定' (Network Settings), '時刻設定' (Time Setting), 'SSL 証明書' (SSL Certificate), 'ソフトウェアアップデート' (Software Update), 'ヘルプとサポート' (Help and Support), and '言語' (Language). The main content area is titled '時刻設定' and includes a 'manual' link with a user icon. Under 'システム時刻' (System Time), the current settings are: タイムゾーン (Time Zone) set to UTC, 日付 (Date) set to 2026-03-05, and 時刻 (Time) set to 00:56:10. The 'タイムゾーンの変更' (Change Time Zone) section shows a dropdown menu currently set to UTC. The 'システム時刻の変更' (Change System Time) section features a toggle for '自動設定 (インターネット接続が必要です)' (Automatic Setting (Internet connection required)) which is turned on, and a radio button for '手動設定' (Manual Setting) which is turned off. Below these are input fields for '日付設定' (Date Setting) showing 2026-03-05 and '時刻設定' (Time Setting) showing 00:56:10. At the bottom, there are 'キャンセル' (Cancel) and '保存' (Save) buttons.

図 2.3.2: 時刻設定ページ

2.3.3 SSL 証明書

[SSL 証明書]メニューでは、現在読み込まれている SSL 証明書の詳細を確認し、プライベート証明書やビジネス証明書に置き換えることができます。SSL (Secure Sockets Layer)証明書は、ウェブサイトの正当性を確認し、暗号化通信を可能にするデジタル証明書です。この証明書により、オンライン取引や顧客情報がウェブブラウザ内で安全に保護されます。初期設定では自己署名証明書がインストールされています。信頼できる認証局が署名した、環境に適した証明書に置き換えることを推奨します。

以下の場合、ブラウザーは証明書信頼できないものとして扱います。

- ロボットにアクセスする際のドメイン名または IP アドレスが、証明書に登録された情報と一致しない
- 証明書の有効期限が切れている
- 証明書が自己署名である

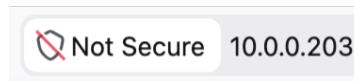


図 2.3.3.a: SSL 証明書が信頼できない、または有効期限が切れているというブラウザー通知例
SSL 証明書を置き換える場合は、秘密鍵と公開鍵のペアをアップロードしてください。

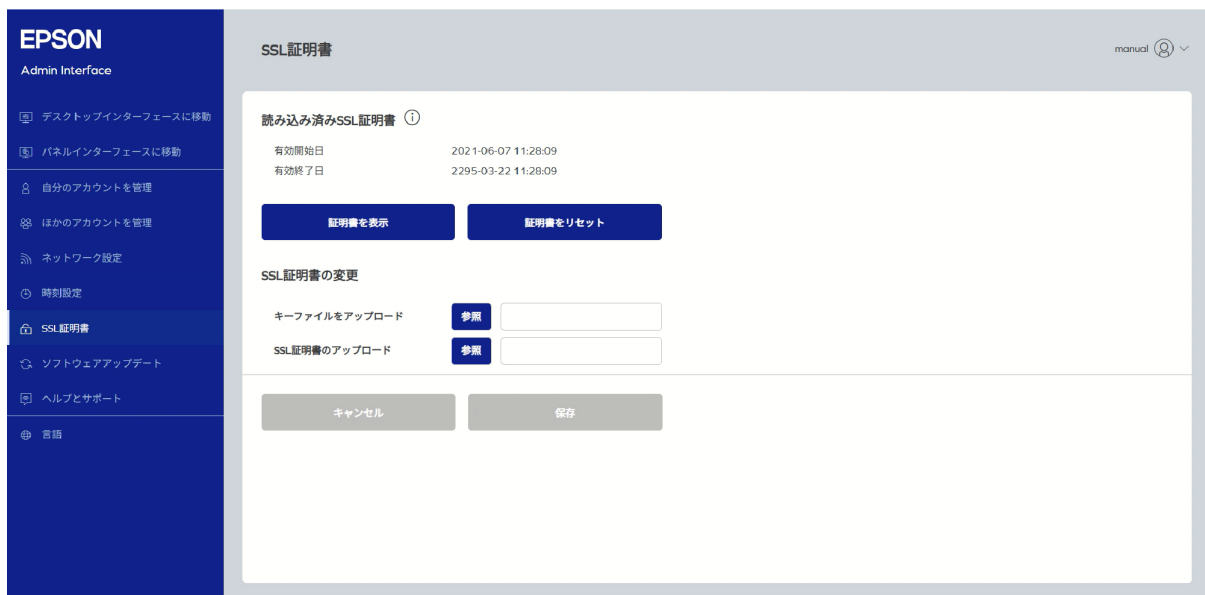


図 2.3.3.b: SSL 証明書設定ページ

2.4 ソフトウェアアップデート

[ソフトウェアアップデート]メニューでは、現在使用している AX Portal のバージョンを確認できます。新しいバージョンのソフトウェアをこの画面からアップロード、インストールできます。ソフトウェアアップデートは、特定の問題を解決したり、最新の機能や不具合修正を適用するために必要です。

ソフトウェアアップデートを実施する前に、必ずリリースノートを確認してください。特定のバージョンにアップデートする際のリスクに関する重要な情報が記載されています。アップデートにより、古い機能や機能が廃止される場合があります。アップデートを実施する前に、デスクトップインターフェースを使用して、ロボットのアプリケーションや設定を含むデータベースをバックアップしておくことを推奨します。



図 2.4.a: ソフトウェアアップデートページ

ソフトウェアアップデートをインストールするには、対象バージョンの AX Portal を含む BIN ファイルを入手する必要があります。アップデートのインストール方法は 3 種類あります。

- 1) [ファイルのアップロード]タブを選択します。端末からアップデートをアップロードします。[参照]ボタンを押して、BIN ファイルを選択し、アップデートをインストールします。
- 2) [USB ストレージデバイス上のファイル]タブを選択します。BIN ファイルを USB ストレージに保存し、ロボットのコントローラーにマウントします。USB ストレージのディレクトリーから BIN ファイルを選択し、アップデートをインストールします。
キーポイント: USB ストレージでサポートしているファイルシステムは exFAT と FAT32 です。
- 3) 管理インターフェースが正しく動作しないなどの理由で 1、2 の方法が使用できない場合、代わりにリカバリーアップデートを実行できます。

アップデート実行中は、アップデートの進行状況、完了したステップ、エラーが発生した場合はそのステップとエラーメッセージが表示されます。更新するパッケージによって、アップデートは最大 **15 分** かかります。ロボットの電源を切るなどして、アップデートを中断しないでください。最悪の場合、ロボットにアクセスできなくなることがあります。

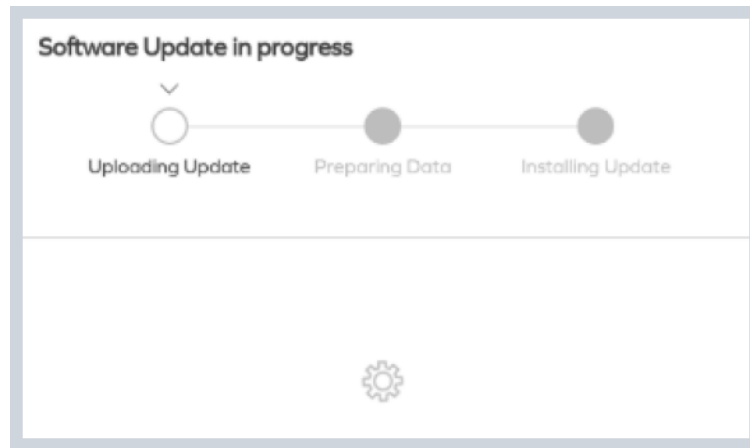


図 2.4.b: ソフトウェアアップデート中画面

アップデートが正常に完了した後は、アップデートを承認する必要があります。アップデートが承認されるまで、状態を示す赤いメッセージバーが表示されます。アップデートを承認せずにコントローラーを再起動すると、アップデートがリセットされます。アップデートを承認すると、以前のバージョンに戻すことはできなくなります。

2.5 ヘルプとサポート

[ヘルプとサポート]メニューでは、エプソンへの問い合わせ先を確認できます。E メールや電話でのお問い合わせの前に、マニュアル類をよくお読みください。



図2.5: ヘルプとサポートページ

2.6 セキュア認証

接続モジュールを使用して AX Portal にアクセスし、GUI を使用せずにロボットを制御する場合は、ロボットを制御する前にセキュアな 2 段階認証を実施する必要があります。

2.6.1 SSL 証明書のエクスポート

はじめに、内蔵の認証サーバーにログイン要求を送信します。

現在読み込まれている SSL 証明書は、ブラウザからエクスポートしてダウンロードできます。

以下は、Google Chrome での操作例です。他のブラウザでも同様の操作ができます。



図 2.6.1: SSL 証明書のエクスポート

1. サイト情報を確認
2. 証明書の詳細に移動
3. 証明書ビューアーの詳細に移動
4. 証明書をエクスポート

2.6.2 ユーザーログイン

はじめに、内蔵の認証サーバーにログイン要求を送信します。認証サーバーは **TCP ポート番号 5001** で接続できます。実施前に、ロボットの **IP アドレス**、ログイン認証情報 (ユーザー名とパスワード) を調べておき、ロボットと同じ **SSL 証明書** を使用してください。TCP ソケット経由でログイン要求を送信し、アクセストークンを返す Python の例は以下のとおりです。ほかのプログラミング言語でも実装できます。

キーポイント: 現時点では、TCP を使ったログイン方法のみ対応しています。ROS、REST、Modbus などの TCP 以外の接続モジュールを使用する場合も、TCP でアクセストークンを取得する必要があります。AX Portal の設定で、これらの接続モジュールに対する認証を無効化することもできます。注意: 認証機能を無効にすると、すべてのクライアントが事前認証なしで ROS や REST 経由でロボットを制御できます。

```
import json, socket, ssl

# define the robot ip address, login credentials, and path to SSL certificate
ip_address = "10.0.0.203"
username = "user_xyz"
password = "my_fancy_password"
session_id = "my_session_id"
ssl_cert = "/home/user/robot-ssl/cert.pem"

# create request to send
request = {"request": "authentication", "action": "login", "data":
           {"username": username, "password": password, "session_id": session_id}}

# prepare SSL context to use with the connection
ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
ssl_context.load_verify_locations(ssl_cert)
ssl_context.check_hostname = False
ssl_context.verify_mode = ssl.CERT_NONE

with socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0) as sock:
    with ssl_context.wrap_socket(sock, server_hostname=ip_address) as ssock:
        # connect to the authentication socket
        ssock.connect((ip_address, 5001))
        # send request (and end character)
        message = json.dumps(request) + "\x03"
        ssock.send(message.encode("UTF-8"))
        # receive answer (before end character)
        message = ssock.recv(1024).decode("UTF-8")
        answer = json.loads(message.split("\x03")[0])

token = answer["response"]["data"]["token"]
session_id = answer["response"]["data"]["session_id"]
```

2.6.3 接続登録

取得した有効なアクセストークンは、"token_login"機能で使⽤します。使⽤方法は、使⽤する接続モジュールについて説明している章を参照してください。

3.USB エクスポートとリカバリーアップデート

3.1 概要

コントローラーが動作しているにもかかわらず、管理インターフェースにアクセスできない場合、リカバリーアップデートでコントローラーを復旧できます。

3.2 USB エクスポート

USB ストレージを使用すると、Web インターフェースにアクセスせずにシステムログと AX Portal のデータベースをエクスポートできます。

1. USB ストレージは extFAT、FAT32、NTFS でフォーマットされたものを使用し、ファイルシステムラベルに LOG または RECOVERY が付いていないことを確認してください。
2. USB ストレージをコントローラーに接続し、数分間待ちます。
3. USB ストレージを取りはずします。USB ストレージのルートに、データベースと関連するシステムログを含む ZIP ファイルが作成されます。

3.3 リカバリーアップデートを実施する

次の手順で、リカバリーアップデートを実行します。

1. アップデートファイル (RBIN 形式)を取得します。アップデートファイルが手元にない場合は、販売元に連絡して入手してください。
2. アップデート用の USB ストレージを用意します。USB ストレージは extFAT、FAT32、NTFS でフォーマットされたものを使用し、最初のパーティションのファイルシステムラベルを "RECOVERY"に設定します。
3. USB ストレージのルートディレクトリーにアップデートファイルを配置します。
4. ロボットの電源を切り、再起動します。ロボットが正常に起動するまで、**1 分以上**待ちます。
5. コントローラーの USB 端子に用意した USB ストレージを接続します。
6. アップデートが完了するまで待ちます。**最大 15 分**かかることがあります。管理インターフェースにアクセスして、新しいバージョンがインストールされているかを確認します。
7. **重要:**ロボットから USB ストレージを取り外してください。取りはずしないと、ロボットが再起動したときにアップデートが意図せず再度実行されます。

4.デスクトップインターフェース (GUI)

4.1 概要

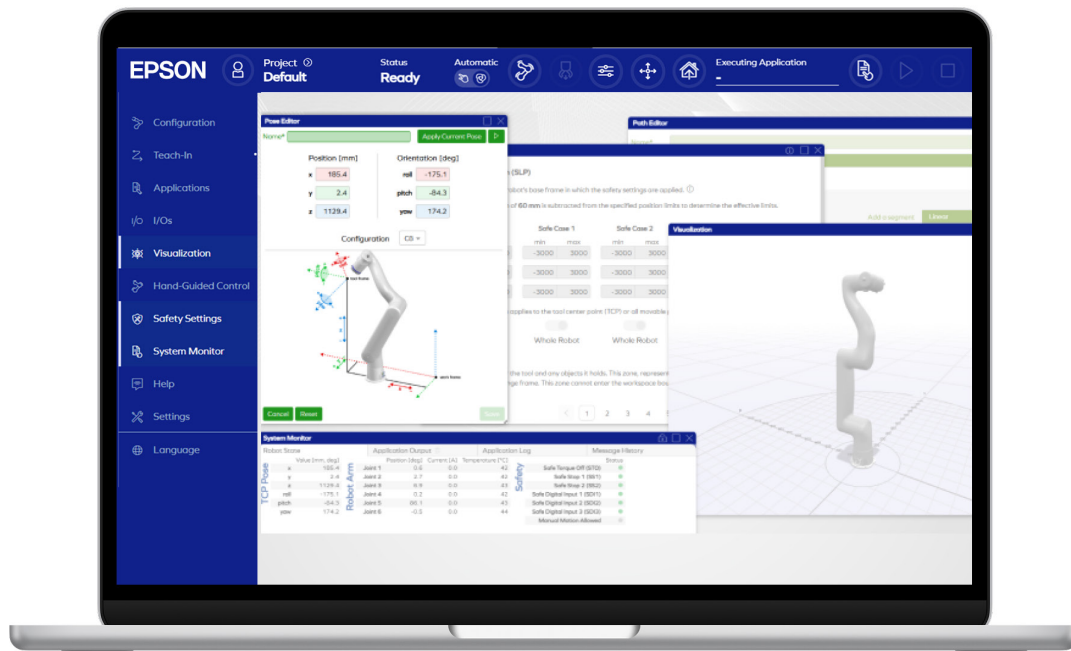


図 4.1.a: デスクトップインターフェースの画面例

AX Portal のデスクトップインターフェースは、ロボットの制御、アプリケーションのプログラミング、データベースの編集、ロボット状態の表示ができるメイン GUI です。この GUI は、ロボット操作の経験が少ないユーザーでも直感的に理解できるように設計されています。管理インターフェースにログイン後、[デスクトップインターフェースに移動]ボタンを選択すると、アクセスできます。



図 4.1.b: デスクトップインターフェースへのアクセス方法

1. ブラウザーのアドレスバーに IP アドレスを入力
2. ログイン用の認証情報を入力
3. デスクトップインターフェースに移動

デスクトップインターフェースには、ロボットの制御機能と状態を監視、変更するための追加機能が含まれています。アプリケーションや設定を変更せずに、事前にプログラミングされたアプリケーションを実行するのみであれば、パネルインターフェースの使用を推奨します。本章では、デスクトップインターフェースの画面、メニュー、項目について説明します。

一部のブラウザ (特に Windows) では、表示スケーリングやズームがデフォルトで高い値 (例: 150%) に設定されている場合があります。この設定により、インターフェースの一部 (コックピットやロゴなど) が切れて表示されることがあります。快適に使用するため、表示スケーリングやズームを 100% (またはそれ以下) に設定することを推奨します。

4.2 メインウィンドウ

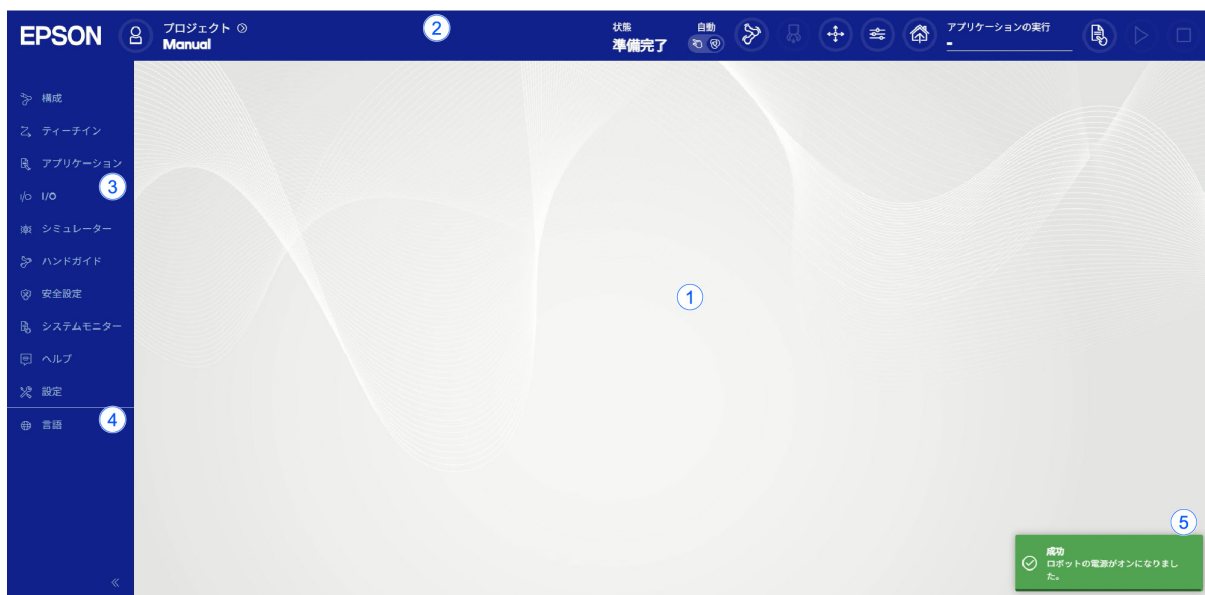


図 4.2: デスクトップインターフェースの要素

1. 作業エリア 2. コックピット 3. メニュー 4. 言語切り替え 5. ポップアップメッセージ

デスクトップインターフェースに接続すると、ウェブブラウザにメインウィンドウが表示されます。

メインウィンドウには以下の要素が含まれています。

- コックピット: 画面の最上段に表示され、現在のプログラムの状態、メッセージ、ロボット制御機能へのクイックアクセスが表示されます。
- メニュー: 画面の左側に表示され、利用可能なメニューに移動できます (詳細は、本章で後述します)。
- バージョン番号: 画面の右下に表示されます。
- メッセージ表示: 画面の右側にポップアップが表示されます。
- 作業エリア: ウィンドウエクスプローラー風にメニューを配置できます。

4.2.1 メニューの概要

メインウィンドウの左側のボタンから、AX Portal の各種メニューにアクセスできます。各メニューの詳細は、後述します。ここでは、使用可能なメニューの概要を説明します。



構成メニュー: ロボットの設定と電源オンができます。



ティーチインメニュー: ロボットのポーズ、パス、グリッドをティーチできます。



アプリケーションメニュー: アプリケーションの作成、修正、テストができます。



I/O メニュー: I/O 状態を監視し、出力信号を変更できます。



シミュレーターメニュー: 現在のロボット動作を表示します。



ハンドガイドメニュー: ロボットの関節を手で動かします。



安全設定メニュー: 安全管理者が安全設定を変更できます。



システムモニターメニュー: システムの出力を監視できます。



ヘルプメニュー: プログラムやシステムに関する情報を確認できます。



設定メニュー: 基本設定ができます。



言語ボタン: インターフェースの表示言語を変更できます。

4.2.2 コックピット

AX Portal のコックピットは GUI の上段に常に表示され、重要な情報や制御機能が配置されています。コックピットの項目を左から順に説明します。



図4.2.2: コックピット

1.ユーザーアカウント設定 2.プロジェクト選択 3.ロボット状態 4.ロボット制御 5.実行中/選択中アプリケーション 6.アプリケーション選択 7.メイン制御



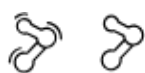
ユーザーボタン: アカウント設定の変更やログアウトができます。項目を選択すると、管理インターフェースに移動します。

プロジェクトボタン: プロジェクトメニューが表示され、プロジェクトの切り替えや概要確認ができます。ボタンの下には、選択中のプロジェクト名が表示されます。

操作を要求ボタン: インターフェースがロボットのアクティブ制御ステーションでない場合のみ表示されます。この安全機能について詳細は、「アクティブ制御」を参照してください。

状態メッセージバー: 現在のプログラム状態が表示されます。プログラム状態は、コマンドやボタン操作などのアクションが実行可能かを示します。

操作モードスライダー: ロボットの操作モード (手動/自動) を切り替えます。この安全機能について詳細は、「操作モード」を参照してください。



ハンドガイドを有効/無効にするボタン: ハンドガイド制御モードに切り替えます。有効にすると、状態が[ハンドガイド]に変わり、ロボットを手動で動かせます。状態が[準備完了]のときのみ、ハンドガイド制御モードを切り替えられます。ハンドガイドを無効にすると、状態は[準備完了]に戻ります。



ツールボタン: 取りつけたツールの[ピック]と[プレイス]機能を切り替えます。ツールエディターで、これらの機能をデジタル入力に関連づけることができます。動作が定義され、プログラム状態が[準備完了]または[ハンドガイド]のときのみ、ツールは動作します。



ジョグボタン: ジョグ制御画面を表示します。ロボットの座標や関節角度を個別に変更できます。



動作ボタン: 動作制御画面を表示します。ロボットの速度調整や、グローバル速度やグローバル加速度の制限変更、座標系の切り替えができます。



ホームボタン: ロボットがホームポジションに戻ります。この機能は、アプリケーション実行前に開始位置へ移動したり、電源オフ前にスリープ位置へ移動するためのショートカットです。ホームポジションは、設定メニューで変更できます。

選択項目: 制御ボタン (再生、一時停止、再開、停止)に関連づけられている項目 (アプリケーション、パス、ポーズ)が表示されます。表示される情報は、[再生]ボタンをクリックするか、アプリケーションを選択すると更新されます。



アプリケーションの選択メニュー: アプリケーションエディターで作成済みのアプリケーションを選択できます。

	再生ボタン: 選択したアプリケーションを実行します。状態が[準備完了]のときに、アプリケーションを実行できます。
	一時停止ボタン: 実行中のアプリケーションを一時停止します。ロボット動作中に一時停止すると、ロボットはスムーズに停止します。
	再開ボタン: 一時停止したアプリケーションを再開します。ロボット動作中に一時停止した場合、動作はスムーズに再開します。
	停止ボタン: 実行中または一時停止中のアプリケーションを停止します。ロボット動作中に停止すると、ロボットはただちに停止し、現在のポーズを保持します。このボタンを押すと、自動回復動作が実行され、状態が[準備完了]に戻ります。

4.2.3 アクティブ制御

ロボットが同時に複数のステーションから操作されないよう、インターフェースはアクティブ制御機能で保護されています。ロボットに最初にログインしたユーザーは、自動的に操作権限を取得します。この接続を「アクティブ制御ステーション」と呼びます。別の接続チャンネルや別のブラウザウィンドウからロボットにアクセスした場合、その接続は操作権限を持たず、ロボットの状態を監視する読み取り専用アクセスになります。この接続を「パッシブオブザーバー」と呼びます。

4.2.3.1 アクティブ制御ステーション

アクティブ制御ステーションでは、操作権限を解除できます。インターフェースのユーザーボタンをクリックし、**操作権限を解除**を選択します。複数のユーザーまたはインターフェースでロボットを操作する場合は、操作権限を解除し、接続チャンネルを閉じることを推奨します。

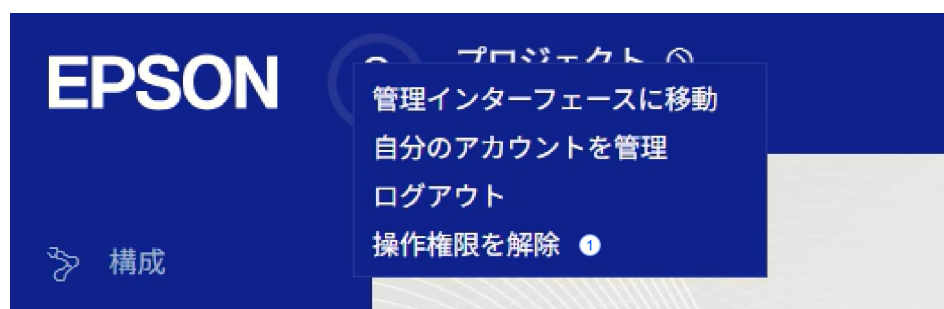


図4.2.3.1 操作権限を解除
1.操作権限を解除

4.2.3.2 パッシブオブザーバー

パッシブオブザーバーは、操作権限を要求できます。インターフェースのユーザーボタンをクリックし、**操作権限を要求**を選択します。または、コックピットに表示される**操作権限を要求**ボタンをクリックします。操作権限を要求すると、アクティブ制御ステーションとパッシブオブザーバーの両方に、アクセス確認のダイアログが表示されます。パッシブオブザーバーは操作権限が付与されるまで待機します。アクティブ制御ステーションは要求を承認または拒否できます。



図4.2.3.2 操作要求

1.操作権限を要求 2.操作を要求ダイアログ

4.2.4 プログラムの状態

プログラムやロボットの現在の状態はコックピットに表示されます。ステータスは AX Portal やロボットの現在の動作を示し、どのアクションやコマンドが実行可能かを左右します。各ステータスには固有の特性があり、一部の機能は特定のプログラム状態でのみ使用できます。各ステータスとその説明は以下のとおりです。

電源オフ	モーターの電源は入っておらず、デバイスは接続されていません。この状態は、電源を投入する前にロボットを設定するために使用します。電源オンスライダーを使用すると、ステータスが 電源オン中 に変わります。他の操作により、ステータスが 読み込み中 に変わる場合があります。特に、中断できない操作や完了まで時間がかかる操作ではこの状態になります。
読み込み中	ロボットがデータベースや設定の読み込みなどデータを処理しています。データの読み込みが完了すると、ステータスは 電源オフ に戻ります。
電源オン中	ロボットがモーターの電源を投入し、デバイスを接続しています。接続が完了すると、ロボットはモーターを制御し、状態は 準備完了 に変わります。
準備完了	コマンドの受信やアプリケーションの実行、ピック&プレイス操作などをプログラムが実行できる状態です。ロボットのアイドル状態です。
実行中	アプリケーションや動作が実行中です。アプリケーションや動作は 一時停止 ボタンで中断、 停止 ボタンで中止できます。アプリケーションや動作が最後まで終了すると、状態は 準備完了 に戻ります。
一時停止	アプリケーションや動作が一時停止中です。 再開 ボタンを押すと動作が再開し、状態は 実行中 に戻ります。 停止 ボタンを押すと動作を中止し、状態は 準備完了 に戻ります。

ハンドガイド	ロボットの1つ以上の関節でハンドガイド制御 (HGC)が有効です。この状態では、関節に力を加えてロボットを任意のポーズに手動で移動できます。ロボットは重力を補正し、自重を保持します。ロボットを動かすには、ハンドガイドボタンを押すか、イネーブルデバイスを有効にする必要があります。ハンドガイドを無効にすると、状態は準備完了に戻ります。
処理中	データの保存や読み込み、軌跡の計算など時間のかかる処理を実行中です。処理が完了すると、元の状態に戻ります。
停止中	通常、停止中状態はすぐに準備完了状態に自動的に復帰します。復帰しない場合、AX Portal の処理が正常に完了していない可能性があります。この場合、ロボットを再起動して復帰してください。
非常停止 通常停止 保護停止 衝突 安全停止 位置制限	これらの状態は、それぞれの割り込みに対応しています。割り込みが発生すると、対応する状態が表示され、割り込みウィンドウが表示されます。ここで割り込みを処理します。割り込みが解決すると、ステータスは通常続行に変わります。
続行	割り込みが解決すると、ステータスは続行に変わります。続行するには、ロボットの動作やアプリケーションが安全であることをユーザーが確認する必要があります。
Error	自動で復帰できない、またはユーザーによる対応が必要なエラーが発生しました。エラーメッセージを確認し、エラーを解決してください。
電源オフ中	ロボットは現在、モーターの電源をオフにし、デバイスを切断し、すべてのモジュールを終了しています。完了後、ステータスは電源オフに戻ります。

4.2.5 動作モード

AX ロボットには、手動/自動の 2 つの動作モードがあります。操作モードは、コックピットのスライダーで切り替えられます。

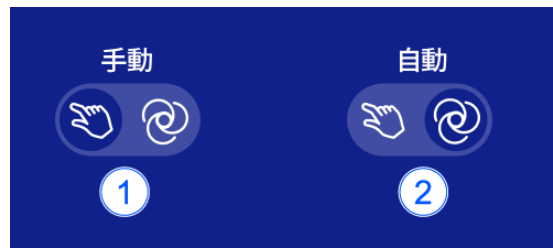


図 4.2.5 動作モード切り替えボタン

 警告	自動モードは、必要な安全機能を備え、テストを実施し、リスクアセスメントによって安全性が確認されたシステムでのみ使用してください。これ以外の状況で自動モードを使用することは、非常に危険です。詳細は、"AX6 / RC-A101 安全マニュアル"を参照してください。
---	---

4.2.5.1 自動モード

自動モードはアプリケーションが問題なく動作することが確認された後の、通常動作に使用します。このモードでは、ロボットはすべての速度範囲で動作できます。

4.2.5.2 手動モード

手動モードは、新しいポーズのティーチやアプリケーション変更のテストなど、ロボットのアプリケーションを変更するときに使用する安全機能です。手動モードでは、ロボットの最大速度は 250 mm/s に制限され、イネーブルデバイスを作動させている間のみ動作します。イネーブルデバイスは、3 ポジションの安全スイッチです。イネーブルデバイスが作動していないときは、ロボットの動作が許可されていないことを示す警告が表示されます。



図 4.2.5.2 イネーブルデバイス例

4.2.6 ジョグ制御

コックピットのジョグボタンを押すと、ジョグ制御画面が表示されます。ジョグを使用すると、ロボットを最大 0.1 [mm]/[deg]の精度で微調整できます。画面は 2 つのタブで構成されています。1 つ目のタブでは、ツールセンターポイント (TCP) のユークリッド座標を変更できます。これはワークフレームで表したツールフレームと一致します。2 つ目のタブでは、ロボットの関節角度を変更できます。

各タブには以下の機能があります。

入力フィールド	入力フィールドに数値を入力すると、座標や関節の値を変更できます。
スライダー	スライダーでは、制御ボタンを 1 回押したときにロボットが移動する距離を示す 増加量 と、ロボットの移動速度を示す 最大動作速度 を設定できます。
制御ボタン	制御ボタンを 1 回押す と、ボタンに表示された方向に指定した距離分移動します。制御ボタンを 長押し すると、ボタンに表示された方向に連続して移動します。
姿勢設定	ひとつの TCP に到達するには、最大 8 種類の関節角度の組み合わせがあります。この関節角度の組み合わせをポーズの姿勢設定と呼びます。ドロップダウンリストで 姿勢設定 を変更すると、現在のポーズに対する姿勢設定を選択できます。 次の姿勢設定 ボタンを押すと、現在のポーズで利用可能な姿勢設定を順に切り替えられます。姿勢設定を変更すると、ロボットのすべての関節が動作します。微調整には使用しないでください。
ポーズとして保存	ポーズとして保存ボタンを押すと、入力した名前でポーズが保存されます。利用可能なポーズは、ティーチインメニューで確認できます。

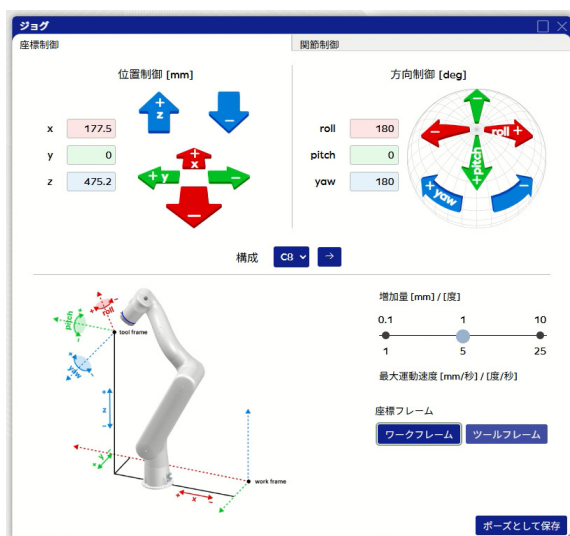


図 4.2.6.a: ジョグ - 座標制御

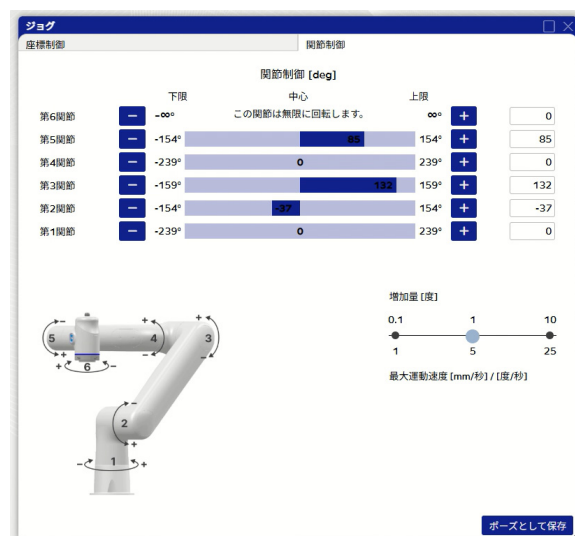


図 4.2.6.b: ジョグ - 関節制御

4.2.7 動作制御

コックピットの動作制御ボタンを押すと、動作制御画面が表示されます。画面は3つのタブで構成されています。

4.2.7.1 動作制限

動作制限タブでは、ロボットの最大速度を制限したり、すべての動作速度をスケーリング係数で調整できます。この設定は、ロボットアプリケーションのセットアップに利用できます。新しいアプリケーションの検証や、既存アプリケーションの動作変更時に、速度を制限できます。

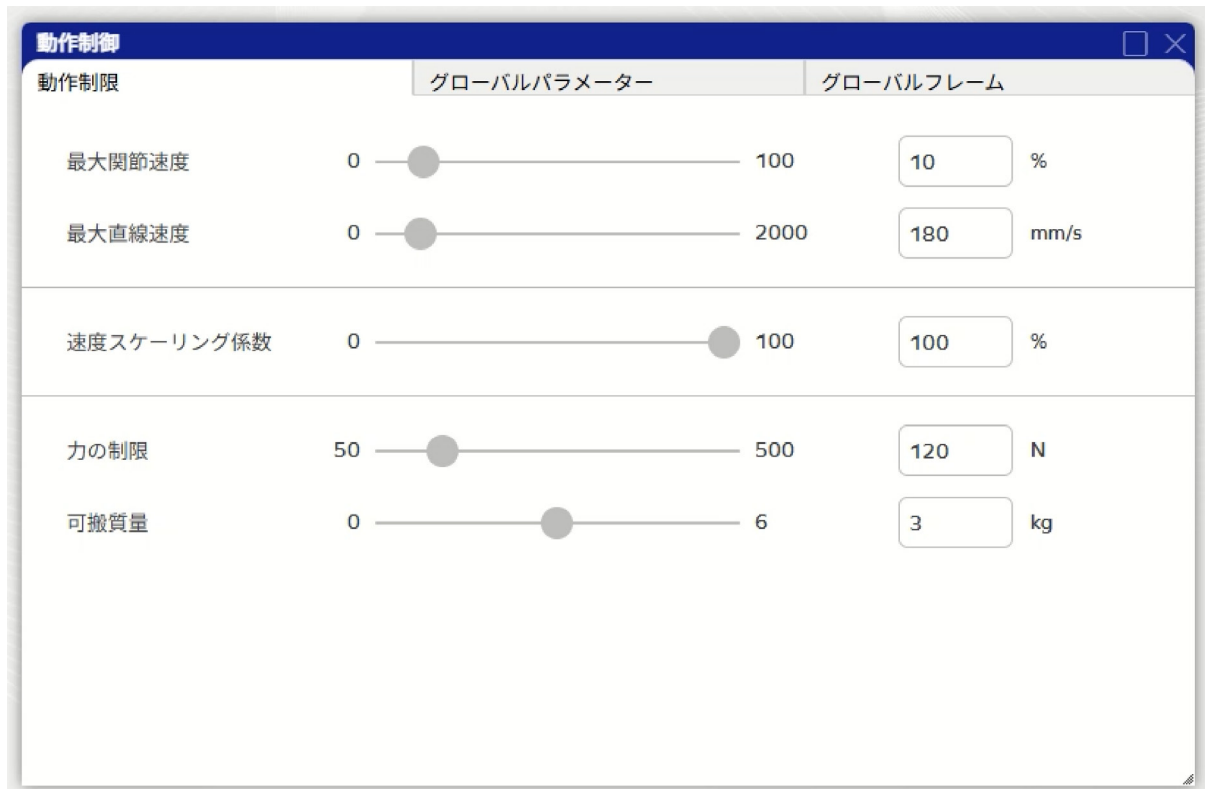


図 4.2.7.1: 動作制御 – 動作制限タブ

一方、力の制限と可搬質量は、ロボットの衝突感度を調整するために使用します。力の制限は、ツールに加わる外力の許容上限を定義します。可搬質量は、ツールを含むロボットが保持する総重量を定義します。

4.2.7.2 グローバルパラメーター

グローバルパラメータータブでは、動作コマンドに対するデフォルトの速度や加速度を設定できます。動作コマンドは3種類に分類されます。

- ロボットを最短時間で動作させるコマンドは、"joint_velocity"、"joint_acceleration"引数を使用します。これらのデフォルト値は、**グローバル関節速度**、**グローバル関節加速度**として定義されます。
- ロボットを直線、円弧など特定の軌跡で動作させるコマンドは、"linear_velocity"、"linear_acceleration"引数を使用します。これらのデフォルト値は、**グローバル直線速度**、**グローバル直線加速度**として定義されます。
- ロボットのツールセンターポイント (TCP)は変えずに姿勢のみを変更するコマンドは、"angular_velocity"、"angular_acceleration"引数を使用します。これらのデフォルト値は**グローバル角速度**、**グローバル角加速度**として定義されます。
- 複数の動作セグメントを含むパスに沿って移動するコマンドは、"blend_radius"引数を使用します。この引数のデフォルト値は、**グローバルブレンド半径**として定義されます。このパラメーターは、パスセグメント間の補間を滑らかにするために使用します。

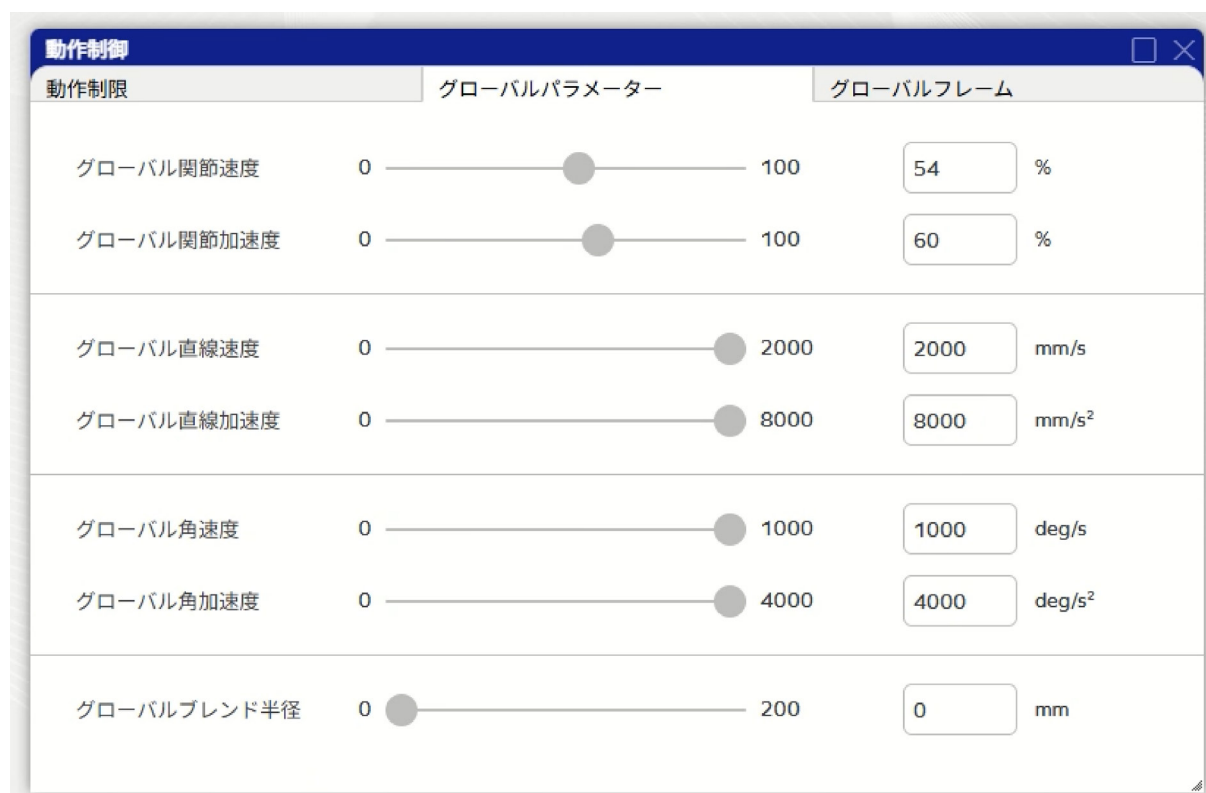


図4.2.7.2: 動作制御 - グローバルパラメータータブ

4.2.7.3 グローバルフレーム

グローバルフレームタブでは、ロボットのデフォルト座標フレームを定義できます。ベースフレーム、フランジフレームはロボットに固定されますが、ワークフレーム、ツールフレームは固定フレームに対して相対的に定義されます。

- ベースフレームはロボット下部の取り付け面中心に固定されており、Z 軸はアーム方向に、X 軸はケーブルの逆方向を向きます。
- フランジフレームはロボット上部の取り付けプレート中心に固定されています。すべての関節が 0° のとき、フランジフレームの向きはベースフレームと一致します。
- ワークフレームはアプリケーションでツールセンターポイント (TCP) の座標を指定するための基準フレームです。デフォルトでは、ワークフレームはロボットのベースフレームと一致しています。多くの動作コマンドで "work_frame" 引数を使用します。デフォルト値はグローバルワークフレームとして定義します。
- ツールフレームはフランジフレームに対してツールセンターポイント (TCP) の位置と向きを定義します。デフォルトでは、ツールフレームはロボットのフランジフレームと一致しています。多くの動作コマンドで "tool_frame" 引数を使用します。デフォルト値はグローバルツールフレームとして定義します。

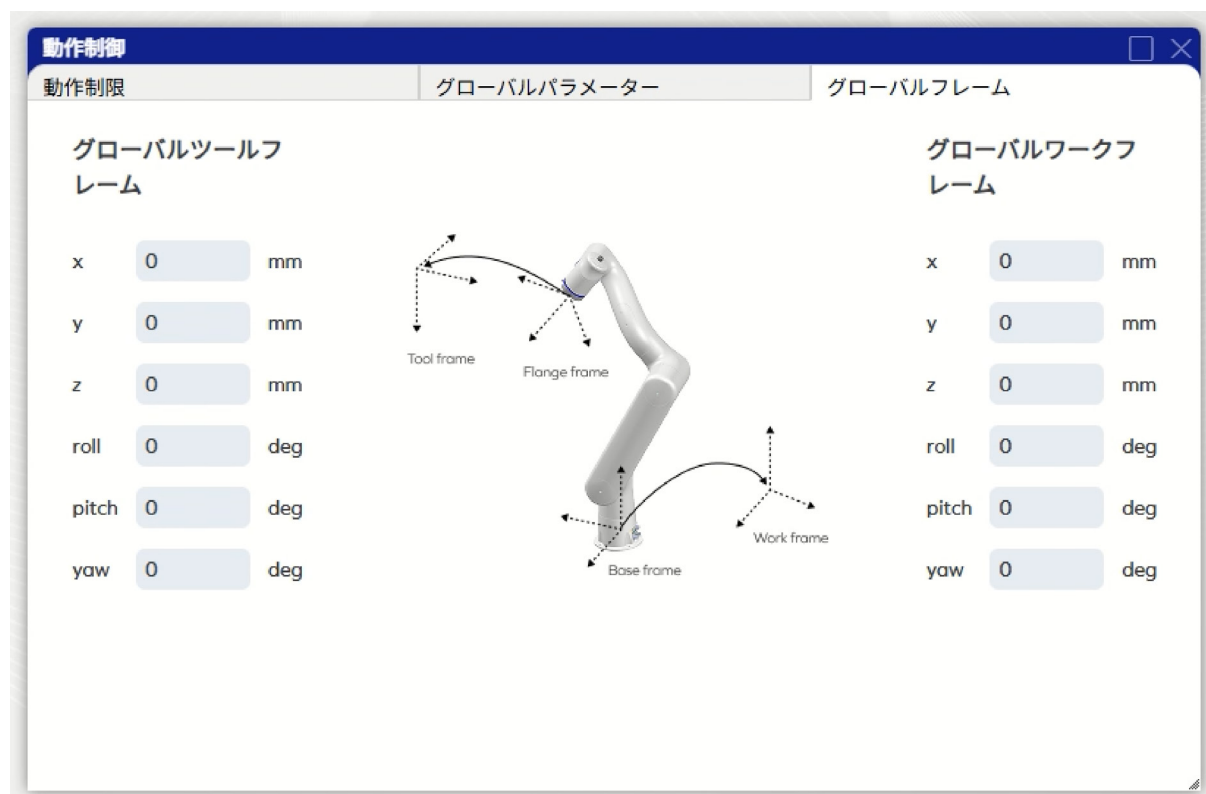


図 4.2.7.3: 動作制御 - グローバルフレームタブ

4.3 ブロック/データベース

AX Portal は、複数のブロックを使用するオブジェクト指向構造を採用しています。これにより、かんたんで直感的なアプリケーションのプログラミングができます。それぞれのブロック (以降、「エレメント」と呼びます) はデスクトップインターフェースから起動するエディターで個別に作成します。作成したエレメントを AX Portal でリンクさせることで、オブジェクト指向でアプリケーションが作成できます。数値や座標を多用せず、短く読みやすいコードになります。

エレメントの一覧は以下のとおりです。



アプリケーションには、プログラムコードやブロック、アプリケーション固有の設定が含まれます。プログラムコードは **Python 3.13** に対応しています。Python の基本機能に加えて、内部のロボット制御 API (Application Programming Interface) の機能を使用できます。アプリケーションエディターを使用して、作成、修正します。



ポーズは、ツールセンターポイント (TCP) の位置と姿勢です。関節角度の組み合わせにより一意に定義されます。ポーズエディターを使用して、作成、修正します。



パスは、TCP の滑らかな動きを表すポーズの連続です。ロボット全体の動作が定義されます。パスエディターを使用して、作成、修正します。



グリッドは、最大 3 次元までカスタマイズ可能なポイント配列です。事前に定義されたトレイの位置などを指定できます。グリッドで定義したポイントは、アプリケーション内でイテレーターとして使用します。グリッドエディターを使用して、作成、修正します。



I/O はデジタル信号とアナログ信号に分類されます。I/O 設定メニューで設定し、I/O メニューで値を確認できます。I/O は、外部デバイスとの接続や通信に使用します。

これらのエレメントは、AX Portal のデータベースに保存されます。データベースの内容は、構成メニューから設定のバックアップとしていつでもダウンロードできます。また、新しいデータベースをアップロードして、現在のデータベースとすべてのエレメントを上書きすることもできます。プロジェクトメニューでは、データベースに保存されているすべてのプロジェクトとエレメントの概要情報を確認できます。

4.4 プロジェクトメニュー

コックピットで現在選択しているプロジェクトをクリックすると、プロジェクトメニューが表示されます。このメニューでは、利用可能なエレメント (ポーズ、パス、グリッド、アプリケーション) の概要を確認できます。エレメントはプロジェクトごとにグループ化されます。同時に選択できるプロジェクトは 1 つのみです。AX Portal では、選択したプロジェクトに含まれるエレメントのみ表示、編集、使用できます。



図 4.4: プロジェクトメニュー



新規プロジェクト作成ボタンを押すと、空のプロジェクトが作成されます。



プロジェクト名の左に表示される丸印をクリックすると、プロジェクトが選択されます。



エレメントのコピーボタンを押すと、同じプロジェクト内にエレメントが複製されます。複製されたエレメントの名前は末尾に"(copy)"が追加されます。



プロジェクト名の右に表示される編集ボタンを押すと、プロジェクト名が変更できます。エレメントの編集ボタンを押すと、確認や修正するためのエディターが表示されます。



プロジェクト名の右に表示される削除ボタンを押すと、プロジェクトとプロジェクトに含まれるすべてのエレメントを削除します。エレメントの削除ボタンを押すと、プロジェクトからエレメントを削除します。

ポーズ、パスなどエレメントのアイコンを押すと、グループ化されたエレメントの表示/非表示を切り替えられます。

エレメントをマウスでドラッグし、別のプロジェクトにドロップすると、移動できます。ドラッグ中に **CTRL** キーを押すと、別のプロジェクトにエレメントをコピーできます。

4.5 構成メニュー

メインウィンドウの**構成**ボタンを押すと、構成メニューが表示されます。このメニューでは、データベースの設定、I/O やツールなどハードウェアの設定ができます。ロボットで特定のアプリケーションのみを実行する場合、構成は最初の 1 回のみ設定が必要です。次回以降はロボットの電源オンにこの画面を使用します。設定メニューで適切な設定をすると、電源オンやアプリケーションの実行を自動化できます。複数の用途でロボットを使用する場合、プロジェクトや目的に応じて異なるツールやデータベースを設定するため、このメニューを頻繁に使用します。



図 4.5: 構成メニュー

4.5.1 データベース (ダウンロード/アップロード)

データベースに関する項目では、現在のアプリケーションデータベースをダウンロードしたり、互換性のある別のデータベースをアップロードして、内容を上書きできます。**データベースのダウンロード**ボタンを押すと、データベースの内容を ZIP ファイルとしてローカルに保存できます。アプリケーションの数が多い場合や作成や調整に時間をかけている場合は、**定期的にデータベースをバックアップ**することを推奨します。**データベースをアップロード**ボタンを押すと、ZIP ファイルを選択する画面が表示されます。データベースファイルを選択すると、現在のデータベースがアップロードした内容で**上書き**されます。

ロボットからデータベースをダウンロードして、別のロボットにアップロードすることで、すべてのアプリケーションやエレメントをロボット間でコピーできます。データベースは定期的にバックアップすることを推奨します

データベースにはバージョンがあり、すべてのデータベースがすべての AX Portal バージョンと互換性があるわけではありません。古いデータベースは新しいバージョンのソフトウェアに読み込めますが、新しいデータベースは古いバージョンのソフトウェアに読み込みできません。つまり、データベースには後方互換性がありません。

4.5.2 I/O の設定

I/O の設定ボタンを押すと、I/O の設定画面が表示されます。この画面では、I/O を設定したり、定義済みの機能をデジタル入力に割り当てたり、I/O にカスタム名をつけたりできます。カスタム名をアプリケーションで使用すると、プログラムが読みやすくなります。この場合、整数の I/O 番号 (整数) を I/O 名 (文字列) に書き換えます。

ID	設定	カスタム名	機能	トリガー	状態/値
DI 01	プルダウン付き入力	未定義	未定義	N.A.	●
DI 02	プルダウン付き入力	未定義	未定義	N.A.	●
DI 03	プルダウン付き入力	未定義	オブジェクトをピック	立ち上がりエッジ	●
DI 04	プルダウン付き入力	未定義	未定義	N.A.	●
DO 01	出力	未定義	未定義	N.A.	■
DO 02	出力	未定義	未定義	N.A.	■
DO 03	出力	Pick	未定義	N.A.	■
DO 04	出力	Release	未定義	N.A.	■
AIO 01	入力 (0-10 V)	未定義	未定義	N.A.	0.0 V
AIO 02	入力 (0-10 V)	未定義	未定義	N.A.	0.0 V
AIO 03	入力 (0-10 V)	未定義	未定義	N.A.	0.0 V
AIO 04	入力 (0-10 V)	未定義	未定義	N.A.	0.0 V

アイコン	カスタム名	機能	トリガー	状態/値
①	未定義	未定義	N.A.	●
②	未定義	未定義	N.A.	●
🔍	未定義	オブジェクトをピック/プレース	立ち上がりエッジ	●
📄	未定義	ポーズを記録	立ち上がりエッジ	●
👤	未定義	ハンドガイド (HGC) を有効/無効にする	押している間アクティブ	●

図4.5.2.a: I/O の設定メニュー

AX シリーズには、手先にボタンインターフェースが搭載されています。このメニューでは、他のデジタル入力と同様に、2 つのカスタムボタンに機能を割り当てられます。

デジタル入力で機能をトリガーすると、現在の制御ステーションからアクティブ制御が解除されます。

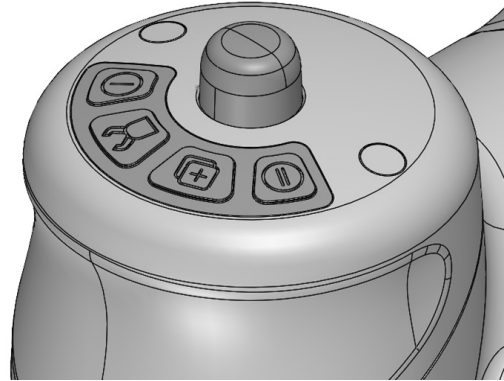


図 4.5.2.b: AX ボタンインターフェース

定義済みの以下の機能がデジタル入力に割り当てられます。トリガーは立ち上がりエッジ (デフォルト) または立ち下がりエッジから選択できます。

機能なし	何もトリガーされません。デフォルトで選択されています。
アプリケーションを一時停止	実行中のアプリケーションを一時停止します。 コックピットの[一時停止]ボタンを押したときと同様の動作です。
アプリケーションを停止	実行中または一時停止中のアプリケーションを停止します。 コックピットの[停止]ボタンを押したときと同様の動作です。
実行を開始	選択しているアプリケーションを実行します。 コックピットの[再生]ボタンを押したときと同様の動作です。 割り込みから復帰した場合、この機能はロボットを割り込み発生時のポーズに戻し、直前の動作を継続します。 割り込み発生後に画面に表示されるダイアログで、[続行]および[再開]ボタンを押したときと同様の動作です。
安全リセット	割り込み後の安全状態をリセットします。 割り込み発生後に画面に表示されるダイアログで、安全をリセットする操作と同様の動作です。
ハンドガイド (HGC) を有効にする	すべての関節に対して、ハンドガイドを有効にします (状態が[準備完了]のとき)。
ハンドガイド (HGC) を無効にする	すべての関節に対して、ハンドガイドを無効にします (状態が[ハンドガイド]のとき)。
ハンドガイド (HGC) を有効/無効にする	すべての関節に対して、ハンドガイドの有効、無効を切り替えます。トリガーされる機能はプログラムの状態により異なります。
オブジェクトをピック	定義済み、またはデジタル出力に割り当てられたツールのピック機能をトリガーします。コックピットの[ピック]ボタンを押したときと同様の動作です。
オブジェクトをプレース	定義済み、またはデジタル出力に割り当てられたツールのプレース機能をトリガーします。コックピットの[プレース]ボタンを押したときと同様の動作です。
オブジェクトのピック/プレース	ツールを使ったオブジェクトのピック、プレースを切り替えます。ツールの状態により、動作が異なります。
ポーズを記録	現在のロボットのポーズを記録し、"button_pose" (または同様の名前) として保存します。
手動モード	ロボットを手動モードに切り替えます。 コックピットでモードを切り替えたときと同様の動作です。
自動モード	ロボットを自動モードに切り替えます。 コックピットでモードを切り替えたときと同様の動作です。

4.5.3 ツールの設定

この画面では、ロボットのツールに関する設定ができます。サードパーティー製、または特定用途向けの独自ツールなどのカスタムツールを取りつける場合は、**カスタムツールの編集**ボタンをクリックします。利用可能なカスタムツールの一覧と設定項目が表示されます。

	新規ツール作成ボタンを押すと、ツールのキネマティクスプロパティ、動的プロパティ、ピック/プレースのトリガーを定義するツールエディターが表示されます。
	検索ボタンを押すと、カスタムツールを名前で絞り込みできます。検索フィルターを有効にするには、3 文字以上入力してください。
	編集ボタンを押すか、エレメントをダブルクリックすると、別ウィンドウでツールエディターが開き、確認、修正ができます。
	削除ボタンを押すと、データベースからツールを完全に削除します。

4.5.4 ツールエディター

ツールエディターは 4 つの部分から構成されます (以下の図を参照してください)。値を設定するときは、AX Portal に取りつけるツールの物理特性や電気特性を記載したデータシートで仕様を確認してください。

最初の部分では、**ツールセンターポイント (TCP)**を設定します。これはツールが動作する空間上のポーズです。オブジェクトをピックするツールの場合、ピックするときのツール位置が TCP として定義されます。TCP は上面のコネクタープレート中心にある座標フレームで定義されます。すべての関節が 0 度のとき、座標フレームはベースフレームと同じ向きになります。TCP が定義されていない場合、このフレームはデフォルトツールフレームです (x, y, z, ロール、ピッチ、ヨーがすべて 0)。ここからカスタムツールの動作位置にあわせて TCP を移動、回転させます。

次に、カスタムツールの**動的プロパティ**を定義します。重力補正や衝突検出などの機能は、ロボットの動的モデルの正確さによって品質が左右されます。ツールのプロパティを正確に測定し、定義してください。ツールの総質量と重心を定義します。重心は TCP と同様に、ロボットのデフォルトツールフレームで定義します。[回転慣性モーメント](#)はわかっている場合は手動で入力し、わからない場合は自動で推定します (AX Portal はツールの形状や質量分布を認識できません)。

ツールエディター:

名前*

1. グローバルツールフレームの定義

並進 [mm]

x

0

y

0

z

162

回転 [deg]

roll

0

pitch

0

yaw

0

2. 動的プロパティの定義

質量 [kg]

0

重心 [mm]

x

0

y

0

z

0

回転慣性モーメント [kg m²]

自動

手動

0

0

0

0

0

0

0

0

0

3. ピック動作の定義

モード

デュアル出力モード

ピック動作のデジタル出力を選択

tool DO 03 (Pick)

ブレース動作のデジタル出力を選択

tool DO 04 (Release)

ピックアンドブレース動作の出力状態

高

出力トリガー間の遅延 T_1 [ms]

15

Pick

Place

Pick Output

Place Output

T_1

T_1

キャンセル

リセット

保存 & 閉じる

図 4.5.4: ツールエディター

最後に、ツールにオブジェクトをピックする機構がある場合、コックピットやスクリプトからツールピック、ツールプレース機能を実行した際にトリガーされるデジタル出力を定義します。使用可能なピック動作は以下のとおりです。

4.5.4.1 シングル出力モード

最もシンプルなピック動作です。ピックとプレースを切り替えるデジタル出力を 1 つ定義し、HIGH または LOW 信号でピックがトリガーされるよう定義できます。



図 4.5.4.1: HIGH でピック、LOW でプレースするシングル出力モードの例

4.5.4.2 デュアル出力モード

ピックとプレースに異なるデジタル出力を割り当てるピック動作です。ツールの状態を切り替えると、最小遅延時間 T_1 (msec) で、両方の出力が連続してトリガーされます。また、アクティブ状態を HIGH 信号または LOW 信号のどちらにするかを定義できます。

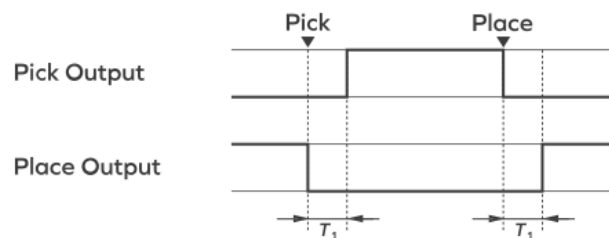


図 4.5.4.2: HIGH がアクティブ状態のデュアル出力モードの例

4.5.4.3 セパレートトリガーモード

セパレートトリガーモードでは、2 つのデジタル出力を定義します。ひとつはピックまたはプレースの動作を定義し、もうひとつは選択した動作のトリガーを定義します。動作の出力では、ピックに使用する信号を HIGH または LOW で指定します (もう一方はプレースです)。トリガーの出力では、トリガーに使用する信号を HIGH または LOW で指定します。さらに、動作選択からトリガーまでの最小遅延時間 T_1 (msec) と、トリガーがアクティブな状態を維持する最小時間 T_2 (msec) を定義します。

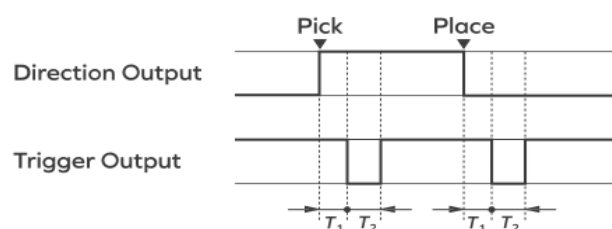


図 4.5.4.3: HIGH でピック動作、LOW でトリガーを定義したセパレートトリガーモードの例

4.5.5 ロボットの起動

データベースを読み込み、I/O やツールの設定が完了したら、ロボットを起動します。通常、ロボットの初期設定を変更せず、**ロボットの電源オン**スライダーを右端までドラッグします。初期化が始まります。

ハードウェアを起動せず (ロボットを動かしたり、I/O の読み書きをしない状態)アプリケーションやポーズなどのデータを編集する場合は、**シミュレーションモード**を有効にしてから**ロボットの電源オン**スライダーを右端までドラッグします。シミュレーションモードでは、すべての動作はリアルタイムでシミュレーションされます。

ロボットが電源オンの間、ロボットの初期構成は変更できません。データベースのダウンロードまたは**ロボットの電源オフ**のみ実行できます。ロボットがアイドルまたはスタンバイ状態のときは、ロボットの電源をオフにすることを推奨します。

安全管理者としてログインしている場合、**安全検証モード**を有効化できます。このモードを有効にすると、ソフトウェアは位置、速度、力の制限を無視します。これにより、制限違反時に安全装置が実際に作動するかを確認できます。このモードで動作制御を使用すると、制限は適用されません。

4.6 ティーチインメニュー

メインウィンドウのティーチインボタンを押すと、ティーチインメニューが表示されます。このメニューでは、現在選択しているプロジェクトに登録されているポーズ、パス、グリッドの概要を確認できます。

ポーズとパスは作成、編集、コピー、削除できます。ポーズ、パスをロボットで直接実行することもできます。グリッドは、繰り返しや等間隔の座標を整理するのに便利です。グリッドはロボットの動作に直接使用しませんが、座標値をセットでまとめて保存し、アプリケーションコードを構造化して座標入力の手間を省きます。グリッドからポイントリストを抽出できます。通常、グリッドのポイントはTCP (ツールセンターポイント)座標として登録されるため、移動機能の入力値に使用されます。



図4.6: ティーチインメニュー

	プラス/新規ボタンを押すと、ポーズ、パス、グリッドそれぞれのエディターが別ウィンドウで開きます。エディターの機能についての詳細は後述します。
	検索ボタンを押すと、ポーズ、パス、グリッドを名前で絞り込みできます。検索フィルターを有効にするには、3 文字以上入力してください。
	再生ボタンを押すと、選択したポーズにロボットを移動、または選択したパスを再生します。パスを再生するときに、ロボットがパスの開始ポーズにいない場合は、最初に開始ポーズに移動します。
	コピーボタンを押すと、選択したポーズ、パスを複製します。複製されたエレメントの名前は末尾に"(copy)"が追加されます。
	編集ボタンを押すか、ポーズ、パス、グリッドをダブルクリックすると、別ウィンドウでポーズ、パス、グリッドの確認、修正ができます。
	削除ボタンを押すと、データベースからポーズ、パス、グリッドを完全に削除します。
	フォルダーボタンを押すと、フォルダー内のポーズ、パス、グリッドの表示/非表示を切り替えます。名前に"/"を含むと、エレメントはフォルダー内に格納されます ("フォルダー/サブフォルダー/名前"のように命名します)。

4.6.1 ポーズエディター

ポーズエディターを使用すると、新しいポーズの作成や既存ポーズの編集ができます。ポーズの座標や姿勢設定を変更しても、ロボットが自動的に移動することはありません。移動させたいときは、ジョグ制御を使用してください。

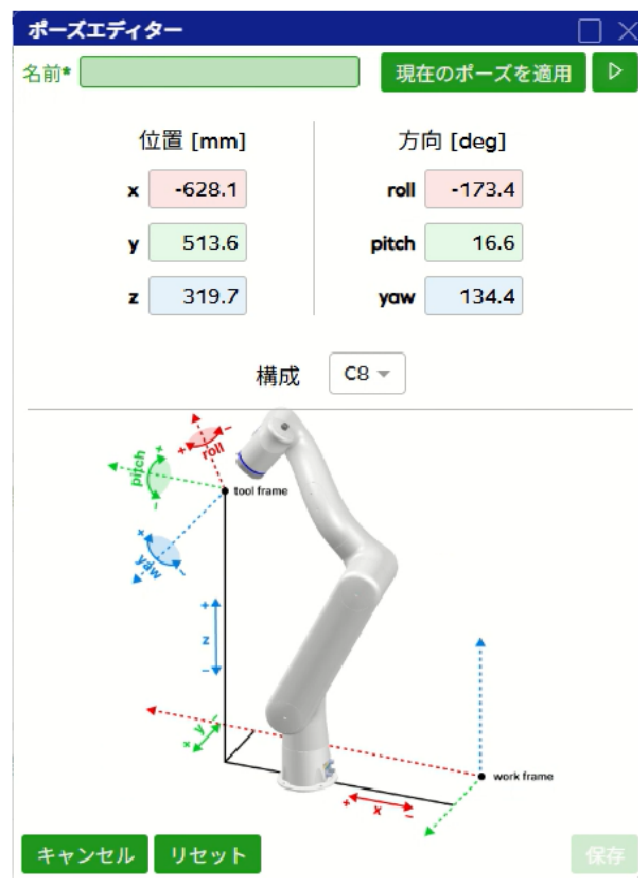


図 4.6.1: ポーズエディター

名前:

データベースに保存するポーズの名前を指定します。ポーズをフォルダーでグループ化するとき、名前に"/"を含めてください。

位置:

ツールセンターポイント (TCP)の位置を mm 単位で表示します。位置はグローバルワークフレームを基準に定義されます。ロボットアームが到達できない位置など、無効な位置情報も入力できます。ただし、再生ボタンを押すとエラーになるため、赤色で表示されます。異なる座標フレームやオフセットを使用すると有効になる場合があるため、入力できるようになっています。

方向:

ツールセンターポイント (TCP)の方向を deg 単位で表示します。方向はグローバルワークフレームを基準に定義されます。方向は航法角 (ロール、ピッチ、ヨー)で表現されます。位置の値と同様に、無効な方向値は赤色で表示されます。

姿勢設定:

ロボットアームの姿勢設定を指定します。姿勢設定 (C1-C8)をドロップダウンメニューから選択します。姿勢設定を変更しても、TCP 位置や姿勢は変わりません。指定したポーズを実現するための関節角度のみが変更されます。

ポーズエディターの操作項目

保存	保存ボタンをクリックすると、指定した名前 で現在のポーズを保存します。有効な名前を入力しないと、ポーズは保存できません。
キャンセル	キャンセルボタンをクリックすると、エディター内の保存していない変更をすべて破棄して、ウィンドウを閉じます。
リセット	リセットボタンをクリックすると、保存していない変更を取り消し、ポーズを最後に保存した状態に戻します。
 再生	ウィンドウ上部の再生ボタンをクリックすると、ロボットを定義したポーズに移動します。
現在のポーズを適用	現在のポーズを適用ボタンをクリックすると、ロボットの現在のポーズにあわせて、すべての項目が更新されます。

4.6.2 パスエディター

パスエディターを使用すると、新しいパスの作成や既存パスの編集ができます。パスエディターで編集できる部分は3つに分かれています。名前、初期ポーズ、パスのセグメントの3つです。

パスエディター: path_1

名前* path_1

初期ポーズ path_1/start

パスのセグメント

セグメント	目標ポーズ	速度	加速度
直線補間	path_1/middle	500 mm/s	1000 mm/s ²
円弧補間	path_1/start (中間ポーズ), path_1/end (目標ポーズ)	250 mm/s	500 mm/s ²

ブレンド半径 100 mm

セグメントを追加 円弧補間

キャンセル リセット 保存

図 4.6.2: パスエディター

名前:

データベースに保存するパスの名前を指定します。パスをフォルダーでグループ化するとき、名前に"/"を含めてください。

初期ポーズ:

パスの開始ポーズを指定します。定義したパスセグメントを順に実行する前に、ロボットはデフォルトの速度、加速度で初期ポーズに移動します。

パスのセグメント:

パスを構成する各セグメントを一覧表示し、定義します。使用可能なセグメントは以下のとおりです。

- 直線セグメント
- 円弧セグメント
- 座標セグメント
- ポーズセグメント
- 姿勢セグメント

各セグメントの詳細は、セグメントの説明を参照してください。

セグメントを追加するには、画面右下のドロップダウンメニューからセグメント種類を選択し、プラス (+) ボタンをクリックします。追加したセグメントの順番は、ドラッグ&ドロップで自由に並べ替えられます。

セグメントごとに目標ポーズを指定します。目標ポーズはセグメントの終了時のポーズです。目標ポーズはポーズエディターで事前に定義しておき、ドロップダウンメニューで選択します。円弧セグメントは、**中間ポーズ**を追加で指定します。中間ポーズも事前に定義したポーズから選択します。

セグメントごとに、**速度**と**加速度**を指定できます。指定しない場合は、グローバルデフォルト値が使用されます。

ブレンド半径は、セグメント間の遷移の滑らかさや動作効率を制御するために設定します。この値は、前のセグメントの終了ポーズから現在のセグメントの開始ポーズまでの最大許容偏差を mm 単位で指定します。

パスエディターの操作項目

パスエディターには、パスを効率的に管理、編集、整理するための機能があります。

保存	保存ボタンをクリックすると、指定した名前で現在のパスを保存します。有効な名前を入力しないと、パスは保存できません。
キャンセル	キャンセルボタンをクリックすると、エディター内の保存していない変更をすべて破棄して、ウィンドウを閉じます。
リセット	リセットボタンをクリックすると、保存していない変更を元に戻し、パスを最後に保存した状態に戻します。
 再生	ウィンドウ上部の 再生 ボタンをクリックすると、ロボットが定義したパスを実行します。ロボットがパスの開始ポーズにいない場合は、パスを実行する前に開始ポーズに移動します。
 削除	セグメントの横にある 削除 ボタンをクリックすると、パスからセグメントを削除します。

4.6.3 グリッドエディター

新しいグリッドを作成するときは、グリッドエディターを開きます。グリッドエディターでは、グリッド設定を順に案内します。ウィザードの次、戻るボタンを使って、グリッドポイントの順序を定義します。既存のグリッドを編集する場合は、ポイントの手動編集画面が表示されます。

はじめに、**グリッドの次元**を選択します。通常、グリッドは2次元配列として扱います。コンベアーからワークをピックアップする場合のように2次元を使わないポイントリストや、箱にワークを積み重ねる場合のような3次元構造も作成できます。グリッドは整列した状態で初期化されますが、ポイントはあとで変更、追加、削除できます。

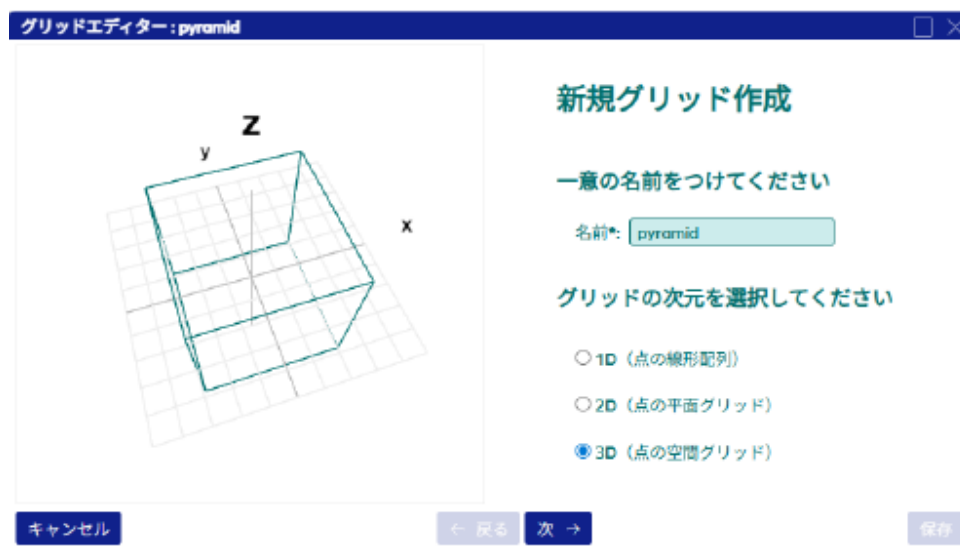


図 4.6.3.a: グリッドエディター (手順 1)

次に、選択したグリッドのコーナーポイントを選択します。グリッドの次元によってコーナーの数は異なります。2次元グリッドでは、コーナーはAからDまでのラベルがつけられます。ロボットのベースフレームでコーナー座標を入力するか、グリッドのコーナーにエンドエフェクターのTCPを移動させ、ティーチボタンを押して座標を保存します。

グリッドを定義するのに必要なコーナーをティーチしたら (2次元グリッドでは最低3ポイント)、**値を評価する**ボタンを押します。入力値が評価され、未入力の値が自動補完されます。2次元の場合は平行四辺形、3次元の場合は平行六面体になるように補完されます。**自動位置合わせ**を有効にすると、値の評価時に水平および垂直座標軸と平面を考慮し、できるだけ軸や平面に沿ってグリッドを配置します (離れすぎていない場合のみ)。

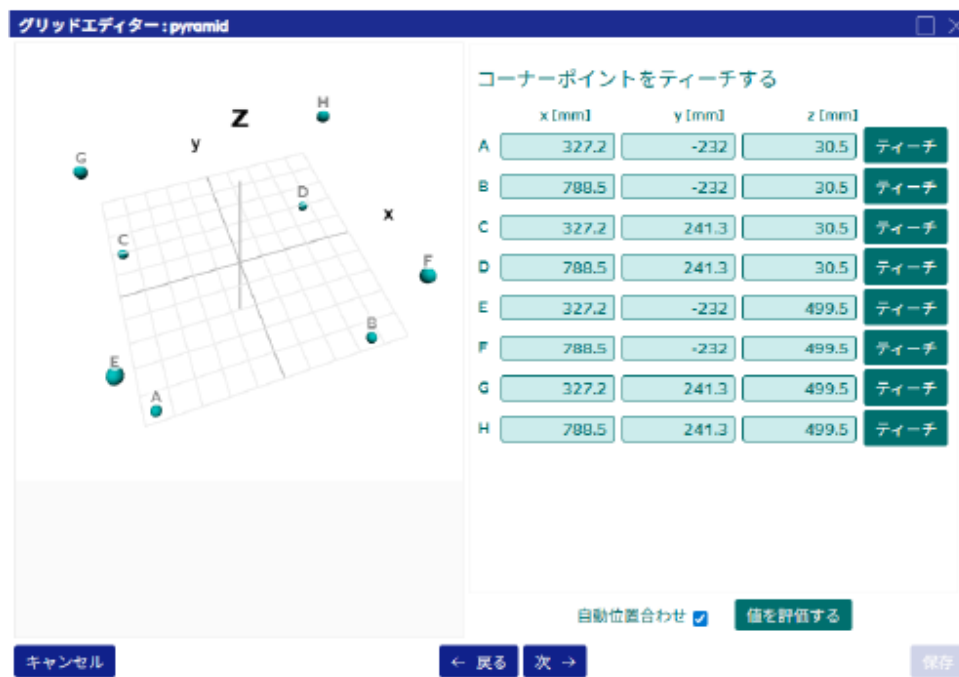


図 4.6.3.b: グリッドエディター (手順2)

次に、ポイント数、グリッドの拡張設定、反復方向を設定します。各設定の詳細は以下のとおりです。

- ポイント数は軸方向ごとに個別に設定します。行や列の軸方向が分かりにくい場合は、ティーチ済みのコーナーポイントのラベルを参照してください。
- 2次元グリッドの場合、グリッドを拡張を有効にすると、4つの隣接点の間に中間点を追加できます。3次元グリッドの場合、各グリッド平面や空間内の8つの隣接点に中間点を追加できます。
- 反復方向は、プログラムの繰り返し処理でポイントを返す方向です。開始点と各軸方向のベクトルで指定します。最後の方向は自動で選択され、確認用として表示されます。ドロップダウンメニューの文字は、グリッドのコーナーポイントのラベルと対応しています。



図4.6.3.c: グリッドエディター (手順3)

最後に、手動編集用のグリッドポイント一覧が表示されます。各ポイントはコピー、削除、移動できます。座標値は手動で変更できます。画面にはグリッドポイントの順序と相対座標が表示されます。

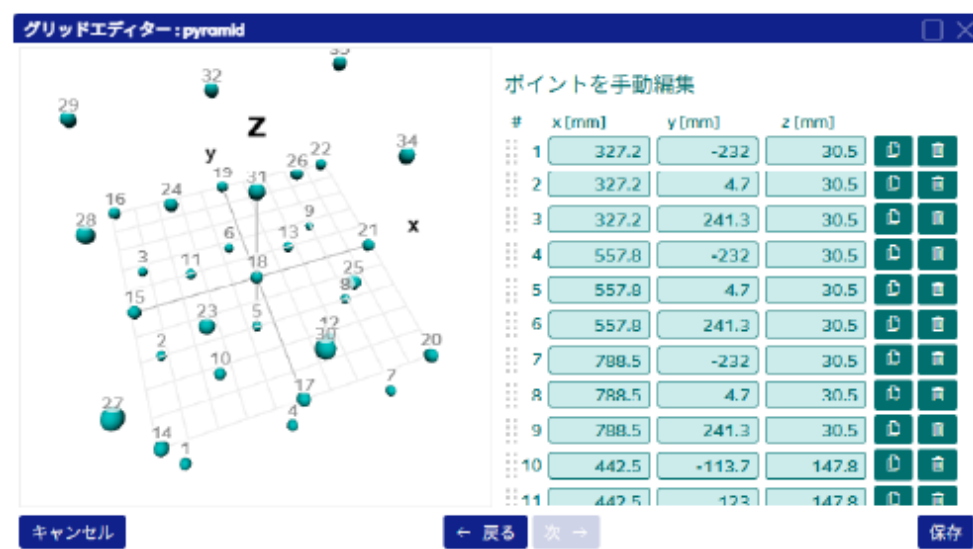


図4.6.3.d: グリッドエディター (手順4)

編集時はウィザードの前の画面に戻り、設定内容や値を確認できます。設定を変更することもできますが、変更すると以降の画面での設定がすべてリセットされます。

4.7 アプリケーションメニュー

メインウィンドウのアプリケーションボタンを押すと、アプリケーションメニューが表示されます。このメニューでは、保存済みアプリケーションの概要と、最後に実行したアプリケーションの出力を確認できます。アプリケーションエディターを開いて、新しいアプリケーションの作成、既存アプリケーションの編集ができます。

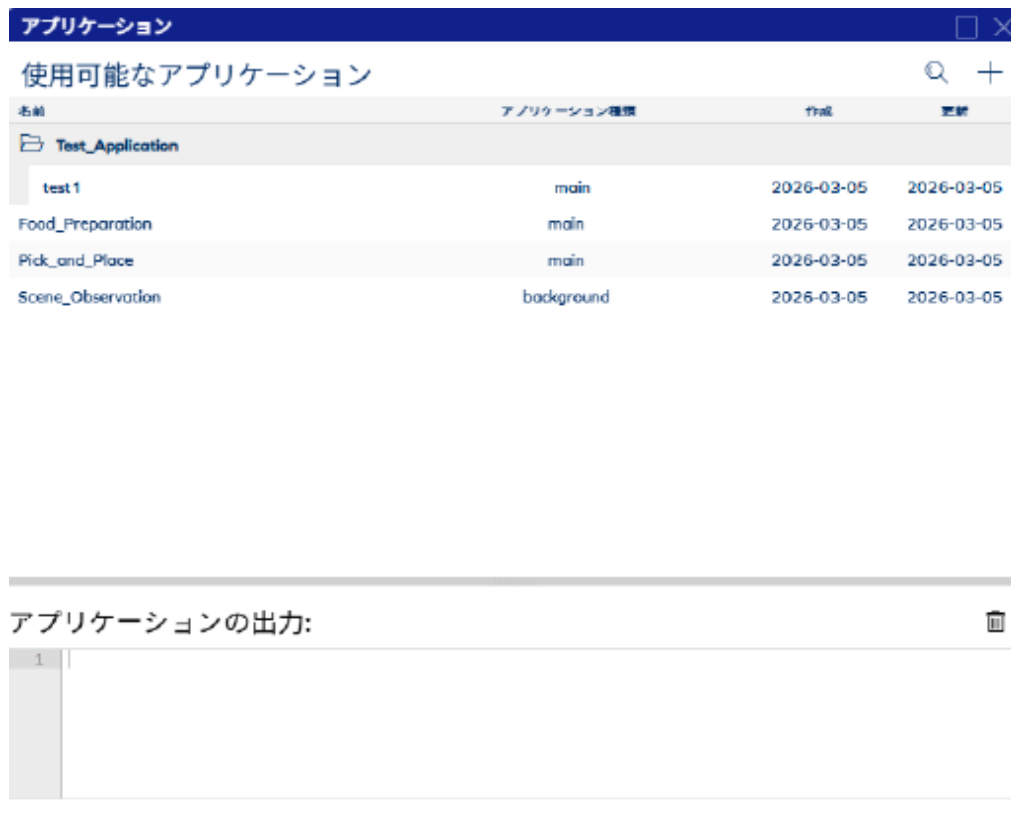


図 4.7: アプリケーションメニュー

- | | |
|---|---|
|  | プラス/新規ボタンを押すと、新しいアプリケーションを作成します。アプリケーションエディターが開きます。アプリケーションエディターについて詳細は、後述します。 |
|  | 検索ボタンを押すと、アプリケーションを名前で絞り込みできます。検索フィルターを有効にするには、3 文字以上入力してください。 |
|  | 再生ボタンを押すと、アプリケーションを実行します。この方法で実行できるのは "main" アプリケーションのみです。コックピットでも "main" アプリケーションのみ選択できます。 |
|  | コピーボタンを押すと、選択したアプリケーションを複製します。複製されたエレメントの名前は末尾に "(copy)" が追加されます。 |
|  | 編集ボタンを押すか、アプリケーションをダブルクリックすると、別ウィンドウでアプリケーションを確認、修正できます。 |
|  | 削除ボタンを押すと、データベースからアプリケーションを完全に削除します。プログラムコードとアプリケーション固有の設定も削除されます。 |



フォルダーを押すと、フォルダー内のアプリケーションの表示/非表示を切り替えます。名前に"/"を含むと、エレメントはフォルダー内に格納されます ("フォルダー/サブフォルダー/名前"のように命名します)。編集ボタンを押すと、フォルダーエディターが表示されます。

4.7.1 アプリケーションエディター

アプリケーションメニューの使用可能なアプリケーションには、保存したすべてのアプリケーションが一覧表示されます。編集ボタンをクリックすると、既存のアプリケーションを編集できます。新規アプリケーション作成をクリックすると、新しいアプリケーションを作成できます。どちらの場合も、アプリケーションエディター画面が表示されます。機能も同等です。アプリケーションエディターの主要なボタンは以下のとおりです。

保存	保存ボタンを押すと、入力した名前でアプリケーションを保存します。名前を入力しないと、アプリケーションは保存できません。アプリケーションをフォルダーでグループ化するときは、名前に"/"を含めてください。
キャンセル	キャンセルボタンを押すと、保存されていない変更を破棄して、エディターを閉じます。
▶ テスト	画面上部のアプリケーションのテストボタンを押すと、アプリケーションがテストモードで実行されます。テストモードでは、スクリプトが実行されますが、アプリケーションは編集を続けられます。テストを開始した後に追加した内容は実行されません。

4.7.2 アプリケーションコード

コードタブは、アプリケーションエディターの3つのサブウィンドウのうち、中心となるタブです。アプリケーションコードを表示し、AX Portal の関数ライブラリーやエレメントをコードに挿入できます。使用できるプログラミング言語は **Python 3.13** です。基本機能と一部のプレインストール済み Python パッケージを利用できます。また、AX Portal にはロボットアプリケーションの作成をかんたんにするための関数ライブラリーがあります。ロボットの移動、センサーデータの読み取り、ユーザーとの通信などを実行できます。これらの関数について詳細は、コードのドキュメントを参照してください。

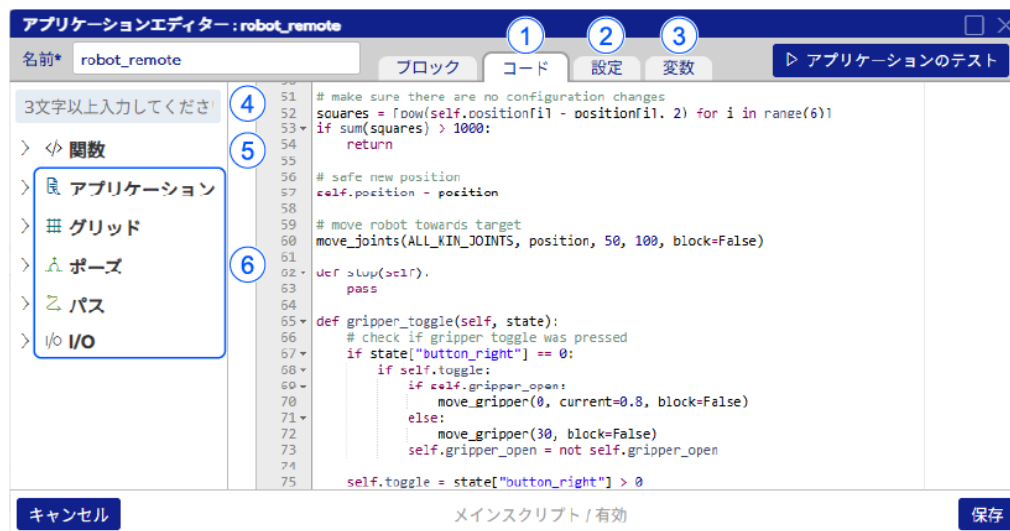


図 4.7.2: アプリケーションエディター (コード)

1. Python プログラムコードの記述場所 2. アプリケーション固有設定の変更 3. アプリケーション変数の定義と修正 4. 検索/絞り込み 5. 関数ライブラリー 6. 使用可能なエレメント

エディター画面の左側にある関数アイコンまたはエレメントアイコンを押すと、エレメントパレットが表示されます。使用可能なすべての関数が表示されます。エレメントをダブルクリックすると、選択した項目をコードに挿入できます。エレメントの名前を覚える必要はありません。

ヒント: 関数をダブルクリックしてコードに挿入すると、オプションの引数も挿入されます。Python コードを読みやすくするため、不要な引数は削除してください。以下に記述例を示します。

```
move_joints(joints, joint_position)
```

以下の記述は推奨しません。

```
move_joints(joints, joint_position, joint_velocity=None,  
joint_acceleration=None, relative=False, block=True)
```

4.7.3 アプリケーション設定

設定タブでは、以下のアプリケーション設定ができます。

- 実行可能フラグを無効に設定し、開発中のアプリケーションを実行しないようにします。
- パネルインターフェースで表示フラグを設定し、パネルインターフェースにアプリケーションを表示します。
- パネルインターフェースに表示するアプリケーションの画像をアップロードまたは置換します。
- ラベルを設定し、パネルインターフェースの選択言語に合わせてアプリケーション名を翻訳します。
- 最も重要なアプリケーション種類について詳細は、後述します。



図 4.7.3.a: アプリケーションエディター (設定)

アプリケーション種類は、アプリケーションを実行する状況を指定します。設定により、アプリケーションの実行方法や使用できる機能に違いがあります。

- **メインスクリプト**は、実行されるメインのアプリケーションです。メインスクリプトはすべてのスクリプト関数を使用でき、ロボットや外部機器を直接制御できるように設計されています。通常、単純なアプリケーションはひとつのスクリプトで記述します。アプリケーションが長く複雑になったり、可読性を高めたいときは、アプリケーションを複数のサブスクリプトに分割し、順に実行する方法が有効です。複数のメインスクリプトは並行して実行できません。並行処理をするときは「並列スクリプト」を使用します。"import"命令を使用して、メインスクリプト間で関数をインポートすることもできます。コックピットで選択できるのは、メインスクリプトのみです。
- **並列スクリプト**は個別のスレッドで実行され、メインスクリプトと並行して動作するように設計されています。並列処理は、センサーデータの継続的な取得や、ロボットが動作しているアイドル時間を利用した消費電力の計算などに便利です。並列スクリプトでは、ロボット制御など一部のスクリプト関数は使用できません。プログラムフローが複雑になったり、予期しない動作が呼び出されたりすることを防ぐためです。メインスクリプトが一時停止すると、並列スクリプトも一時停止します。メインスクリプトが停止すると、並列スクリプトも終了します。
- **バックグラウンドスクリプト**は、AX Portal のステータスに関わらずバックグラウンドで実行されるスクリプトです。データの監視、確認、ログ取得やプロジェクトの初期グローバル変数定義に使用します。各プロジェクトに含められるバックグラウンドスクリプトは1つだけです。バックグラウンドスクリプトが定義されている場合、プロジェクトの読み込み時(起動時やプロジェクト変更時)に自動実行されます。バックグラウンドスクリプトはステータスに関わらず動作します。ループを含む場合、ロボットの電源が切れるまで続きます。バックグラウンドスクリプトの開始時と終了時に通知が表示されます。コックピットからバックグラウンドスクリプトの開始、再開、停止ができます。



図4.7.3.b: バックグラウンドスクリプト制御

4.7.4 アプリケーション変数

アプリケーション変数タブでは、状態変数をドラッグ&ドロップでアプリケーションに追加できます。状態変数はメインスクリプトでのみ使用できます。状態変数はパネルインターフェースで使用し、アプリケーションの状態や進行状況の確認、入力値の指定に使用します。状態変数の書き込みには **set_state_variable(...)**関数、読み取りには **get_state_variable(...)**関数を使用します。

変数の種類とパネルインターフェースでの表示は以下のとおりです。

ブール値	ブール値は有効/無効を切り替えるスライダーで表示されます。Python の bool (ブール)型として読み書きされます。有効は True、無効は False です。
整数	編集モードで、整数は数値入力可能なテキストボックスで表示されます。+/- ボタンを使って値を増減できます。Python の int 型として読み書きされます。
フロート	編集モードで、フロートは数値入力可能なテキストボックスで表示されます。Python の float 型として読み書きされます。
スライダー	編集モードで、パーセントは最小値と最大値を指定したスライダーで表示されます。Python の float 型として読み書きされます。
テキスト	編集モードで、テキストはテキストボックスで表示されます。Python の str 型として読み書きされます。
ドロップダウン	編集モードで、ドロップダウンはドロップダウンリストで表示されます。Python の str 型として読み書きされます。



図 4.7.4: アプリケーションエディター (変数)

変数の設定項目は以下のとおりです。設定できる項目は変数の種類によって異なります。

名前	変数に名前を設定します。以降、変数名の文字列で変数にアクセスできます。
値	変数の値は、現在変数に設定されている値です。Python の対応する型で読み書きします。
デフォルト	変数のデフォルト値は、パネルインターフェースでボタンを使って、変数をリセットしたときに戻る値です。
スタイル	変数の編集可能条件や編集方法を定義します。
最小	数値について、最小値を設定します。最小値より小さい値を入力すると、最小値が適用されます。最小値/最大値はパーセント変数では必須です。整数、フロート変数では任意です。
最大	数値について、最大値を設定します。最大値より大きな値を入力すると、最大値が適用されます。最小値/最大値はパーセント変数では必須です。整数、フロート変数では任意です。
ユニット	数値について、任意で単位を追加できます。
オプション	ドロップダウンリストの選択肢をカンマ区切りで入力します。

変数のスタイルは以下のとおりです。スタイルは変数の使用方法を定義します。

- **編集可能:** アプリケーションの実行前に値を変更できます。実行中は変更できません。
- **保護:** 編集できますが、値を変更するには編集ボタンと保存ボタンを押す必要があります。
- **動的:** 編集可能で、アプリケーション実行中も動的に値を変更できます。
- **動的かつ保護:** 保護と動的を組み合わせたスタイルで、編集ボタンを押すと実行中に値を変更できます。
- **読み取り専用:** 編集できません。アプリケーションでのみ値を変更できます。アプリケーションの状態や進行状況の監視に使用します。
- **非表示:** パネルインターフェースに表示されませんが、ほかの状態変数と同様に読み書きできます。

4.7.5 共有変数、グローバル変数

前述のアプリケーション固有の状態変数に加え、並列実行中のスクリプト間や同一アプリケーションの複数セッション間でデータを共有するために、共有変数とグローバル変数を定義できます。

- 共有変数は、単一アプリケーションの実行中のみ存在します。並列または順次実行中のスクリプト間で値を共有するには、**set_shared_variable(...)**関数と**get_shared_variable(...)**関数を使用します。
- グローバル変数は、アプリケーション内のスクリプト間だけでなく、すべてのアプリケーション間で値を共有でき、ロボットを再起動しても保持されます。**set_global_variable(...)**関数と**get_global_variable(...)**関数を使用します。

4.7.6 フォルダーエディター

アプリケーション名に"/"を含む場合、アプリケーションは自動的にフォルダー構成に整理されます。フォルダーにマウスを合わせると表示される編集ボタンをクリックすると、フォルダーを編集できます。フォルダーエディターが開きます。フォルダーエディターでは、アプリケーションエディターと同様に、フォルダー名の翻訳ラベルを設定できます。フォルダーラベルとアプリケーションラベルはパネルインターフェースに表示されます。翻訳を設定することで、パネルインターフェースがかんたんに多言語対応できます。



図 4.7.6: フォルダーエディター

4.7.7 ブロックプログラミング

アプリケーションエディターのブロックタブでは、視覚的にプログラミングができるインターフェースが用意されています。プログラムフローエリアにプログラムブロックをドラッグ&ドロップするだけで、かんたんにプログラムフローを作成できます。

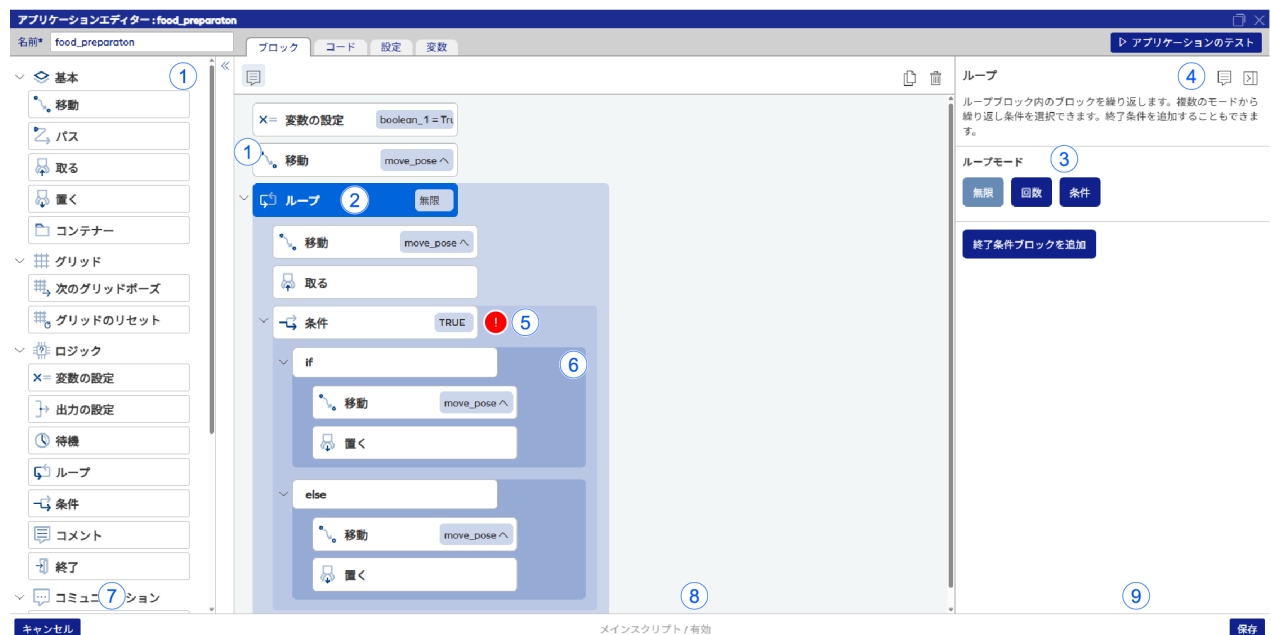


図4.7.7: アプリケーションエディター (ブロックプログラミング)

1. プログラムエリアにブロックをドラッグ
2. 編集するブロックを選択
3. ブロックの詳細を編集
4. コメント
5. エラーインジケーター
6. ブロックコンテナー
7. 使用可能なブロック
8. プログラムフロー
9. ブロックの詳細

プログラムフローのブロックを選択すると、ブロックの詳細が表示されます。プログラムを実行するために、入力が必要な項目があります (例: 移動のブロックの場合、ロボットの移動先を指定する必要があります)。ほかの項目は任意です (指定しない場合はデフォルト値を使用)。または、コメントなどプログラムの可読性を高めるために使用できます。ブロックが正しく定義されていない場合 (必須項目が定義されていない、無効な値が入力されたなど)、プログラムフローにエラーインジケーターが表示されます。ブロックを選択すると、エラー内容が詳細に表示されます。

ブロックを追加したときに、プログラムフローに色付きコンテナーが追加されることがあります。これは、ブロックがグループ化され、上から下に順に実行されることを示しています。代表例はループや条件分岐のブロックです。グループ化されたブロックを複数回実行したり、特定の条件下でグループ化されたブロックを実行します。

選択したブロックは、削除や複製ができます。ドラッグするとプログラム内の任意の位置に移動することもできます。

4.7.7.1 アプリケーション変数と式

ブロック詳細のテキストフィールドでは、アプリケーション変数や数式、論理式を使用できます。これらのフィールドでは、定義済みのアプリケーション変数の値に加え、デジタル入力やアナログ入力の値も挿入できます。

例を使って説明します。以下の変数を定義した場合を例にします。

状態変数

ブール値			
名前	flag	値	☐
スタイル	編集可能	デフォルト	☐

整数			
名前	counter	値	0
スタイル	編集可能	デフォルト	0
最小	0	最大	1000
		単位	

図 4.7.7.1.a: 変数例

有効な数値式

value (number)

42

value (number)

-7 * (8.2 - 3)

value (number)

{counter} + 1

value (number)

{Tool AIO 02}

有効な論理式

value (bool)

FALSE

value (bool)

NOT {flag}

value (bool)

{flag} AND ({counter} > 5)

value (bool)

{Main DIO 09}

図 4.7.7.1.b: 数式、論理式の例

変数や入力値は通常のテキストフィールドにも挿入できます。

value (text)

The counter is set to {counter}.

図 4.7.7.1.c: テキストフィールドの例

4.7.7.2 移動ブロック

移動ブロックは、ロボットを現在位置から目標位置まで移動します。移動時の軌跡が設定できます。

- **PTP**: 最短時間で、軌跡の制約なく目標位置まで移動します。
- **直線補間**: 直線軌跡に沿って目標位置まで移動します。向きも直線的に補間します。
- **円弧補間**: 円弧軌跡に沿って目標位置まで移動します。向きは直線的に補間します。円弧軌跡上に**中間位置**を指定する必要があります。

また、目標位置の指定方法を選択できます。

- **ポーズ名:** 事前に保存した目標ポーズを選択します。
- **グリッド名:** 事前に保存したグリッドのポイントに移動します。はじめに、グリッドの最初のポイントに移動し、次のグリッドポーズブロックを使って、グリッドポイントのカウンターを 1 つ増加させます。
- **関節角度:** 目標ポーズの関節角度を指定します。
- **座標:** 目標ポーズの座標を指定します。

以下の設定は任意で変更できます。設定しない場合、デフォルト値またはグローバル値が適用されます。

- **速度設定**
 - **関節速度/加速度:** PTP での移動時の速度を指定します。
 - **直線速度/加速度:** 直線補間または円弧補間での移動時の速度を指定します。
 - **角速度/加速度:** TCP の位置は変えずに姿勢のみ変更するときの速度を指定します。
- **座標フレームの設定**
 - **ツールオフセット:** フランジフレーム (ロボットアーム上部) に対して TCP をシフトします。
 - **ツールフレーム:** フランジフレーム (ロボットアーム上部) に対して TCP を変更します。
 - **ワークオフセット:** ベースフレーム (ロボットアーム下部) に対して基準フレームをシフトします。
 - **ワークフレーム:** ベースフレーム (ロボットアーム下部) に対して基準フレームを変更します。
- **その他の設定**
 - **相対座標:** 有効にすると、入力した座標値が現在の座標に加算されます。
 - **ブロッキング方式の動作:** 有効にすると、ロボットの動作完了後にのみプログラムフローが継続します。

4.7.7.3 パスブロック

パスブロックは、定義済みのパスを実行します。実行するパスを選択します。ロボットはまず開始ポーズに移動し、その後パスセグメントに沿って滑らかに動作します。

4.7.7.4 ピックブロック

ピックブロックは、接続されたツールにピック命令を送信します。角度グリッパーや平行グリッパーを使うときは、ワークを把持するためにグリッパーを閉じます。空気圧グリッパーを使うときは、ワークを把持するために吸着を有効にします。このブロックに設定できるパラメーターがありません。

4.7.7.5 プレースブロック

プレースブロックは、接続されたツールにプレース命令を送信します。角度グリッパーや平行グリッパーを使うときは、把持したワークを離すためにグリッパーを開きます。空気圧グリッパー

を使うときは、把持したワークを離すために吸着を無効にします。このブロックに設定できるパラメーターがありません。

4.7.7.6 コンテナブロック

コンテナブロックは、複数のブロックをグループ化します。コンテナ自体に特別な機能はありません。

4.7.7.7 次のグリッドポーズブロック

次のグリッドポーズブロックは、グリッド内のポイントカウンターを 1 つ増加させます。対象のグリッドを選択します。移動ブロックを使ってグリッドにロボットを移動させるときは、カウンターで指定されたポイントが目標ポーズになります。グリッド内のすべてのポイントに移動するには、以下の例のとおり、ループ構造の中で移動ブロックと次のグリッドポーズブロックを使用します。



図 4.7.7.7: グリッドブロックを使った例

4.7.7.8 グリッドのリセットブロック

グリッドのリセットブロックは、グリッド内で使用するポイントカウンターをゼロ (初期値)に戻します。対象のグリッドを選択します。

4.7.7.9 変数の設定ブロック

変数の設定ブロックは、アプリケーション変数の値を設定します。はじめに変数を選択し、その後値を入力します。数値式、論理式も使用できます。

4.7.7.10 出力の設定ブロック

出力の設定ブロックは、デジタル出力またはアナログ出力の値を設定します。設定する出力を選択し、デジタル出力にはブール値、アナログ出力には数値を設定します。

4.7.7.11 待機ブロック

待機ブロックは、特定のイベントが発生するまでプログラムフローを停止します。3 種類の待機モードを選択できます。

- **時間:** 指定時間が経過するまで待機します (数値式を使用できます)。
- **動作:** 実行中の動作命令が終了するまで待機します。
- **入力:** デジタル入力が指定値になるまで、またはアナログ入力がしきい値を超えるまで待機します。

4.7.7.12 ループブロック

ループブロックは、繰り返し実行するブロックのグループを作成します。3 種類のループモードを選択できます。

- **無限:** 無限に繰り返します。
- **回数:** 指定回数を繰り返します (数値式を使用できます)。
- **条件:** 指定条件を満たすまで繰り返します (論理式を使用できます)。

終了条件ブロックを追加して、繰り返しの終了条件を指定することもできます。

4.7.7.13 終了条件ブロック

終了条件ブロックはループブロックに関連づけられ、指定条件を満たすとループを終了します (論理式を使用できます)。

4.7.7.14 条件ブロック

条件ブロックは、"if"と"else"の 2 つのグループを持つコンテナを作成します。条件 (論理式を使用可能)に基づいて、プログラムフローを"if" (条件成立)グループまたは"else" (条件不成立)グループに分岐します。

4.7.7.15 コメントブロック

コメントブロックは、プログラムフローに情報コメントを追加します。どのブロックにもコメントを追加でき、プログラムエリアに表示されます。特定のブロックにコメントを関連づけず、独立したコメントも追加できます。

4.7.7.16 終了ブロック

終了ブロックは、実行中のプログラムを終了します。以降のブロックは実行されません。

4.7.7.17 ダイアログブロック

ダイアログブロックは、複数のグループを持つコンテナを作成します。実行すると、プログラムフローを中断し、ダイアログを表示します。プログラムフローを継続するには、ダイアログに応答する必要があります。グループ (デフォルトでは「OK」、「キャンセル」)は、ダイアログのボタンに対応します。押されたボタンに応じて、プログラムフローが分岐し、選択したグループの処理を継続します。ダイアログのメッセージ (変数の挿入可能)、ボタン名、ボタンの数はブロック詳細で設定できます。

4.7.7.18 出力するブロック

出力するブロックは、アプリケーションの出力にカスタムメッセージを出力します。

4.7.7.19 ログブロック

ログブロックは、カスタムログファイルにカスタムメッセージを記録します。ログファイルはデスクトップインターフェースからダウンロードできます。ショートカットキーは CTRL+SHIFT+L です。

4.7.7.20 通知するブロック

通知するブロックは、GUI にカスタム通知メッセージを表示します。通知が表示されても、プログラムフローは停止しません。

4.7.7.21 警告ブロック

警告ブロックは、GUI にカスタム警告メッセージを表示します。警告が表示されても、プログラムフローは停止しません。

4.7.7.22 エラーブロック

エラーブロックは、GUI にカスタムエラーメッセージを表示し、実行中のプログラムを中止します。

4.7.7.23 Python コードブロック

Python コードブロックは、独立した数行の Python コードを記述できます。ブロックプログラミングで使用できない関数を呼び出す場合に使用します。すべてのアプリケーション関数を使用できます。

4.8 I/O メニュー

メインウィンドウの **I/O** ボタンを押すと、I/O メニューが表示されます。I/O メニューでは、すべてのデジタル I/O、アナログ I/O について、I/O の設定画面で事前に定義した設定 (構成、割り当て機能、カスタム名) と、各 I/O の値を確認できます。入力を読み取り専用で、この画面では値を確認できます。出力はこの画面で設定できます。

デジタル I/O はブール値で表されます。**True**、**1**、**緑**はオン、**False**、**0**、**グレー**はオフを意味します。デジタル入力は、円形インジケータで値を表示し、デジタル出力は四角形インジケータを押してオン/オフを切り替えます。

アナログ I/O は 0.1 刻みの浮動小数点値で表示されます。アナログ I/O の単位は、I/O の設定で定義されます (電圧は V、電流は mA)。アナログ入力は数値で表示され、アナログ出力はテキストボックスで値を変更できます。

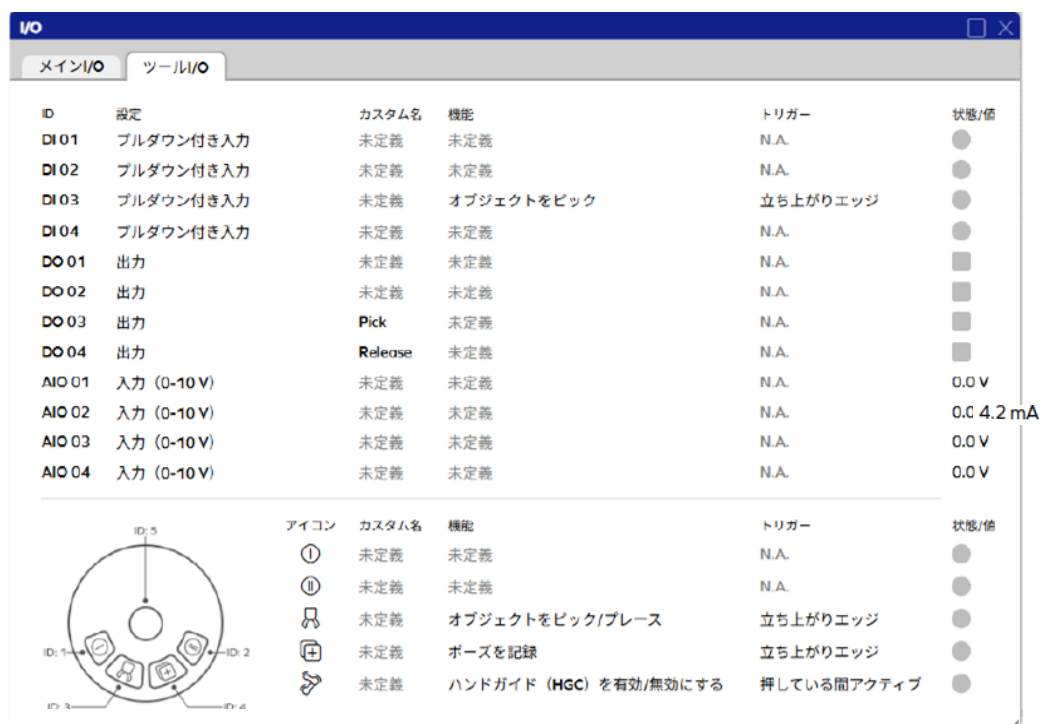


図4.8: I/O メニュー

4.9 シミュレーターメニュー

メインウィンドウのシミュレーターボタンを押すと、シミュレーターウィンドウが表示されます。このウィンドウでは、ロボットの現在のポーズを表示します。遠隔操作で直接視認できない場合に便利です。以前に保存したポーズを 3D モデルで確認することもできます。シミュレーターはインタラクティブに操作でき、次の操作が可能です。

- シミュレーターウィンドウ内でマウスを左クリックし、上下左右にドラッグすると画像が回転します。
- マウスホイールを上下に回すと拡大/縮小します。
- キーボードの矢印キーを押すか、マウスの右ボタンを押したまま左右にドラッグすると、視点を上下左右に移動します。

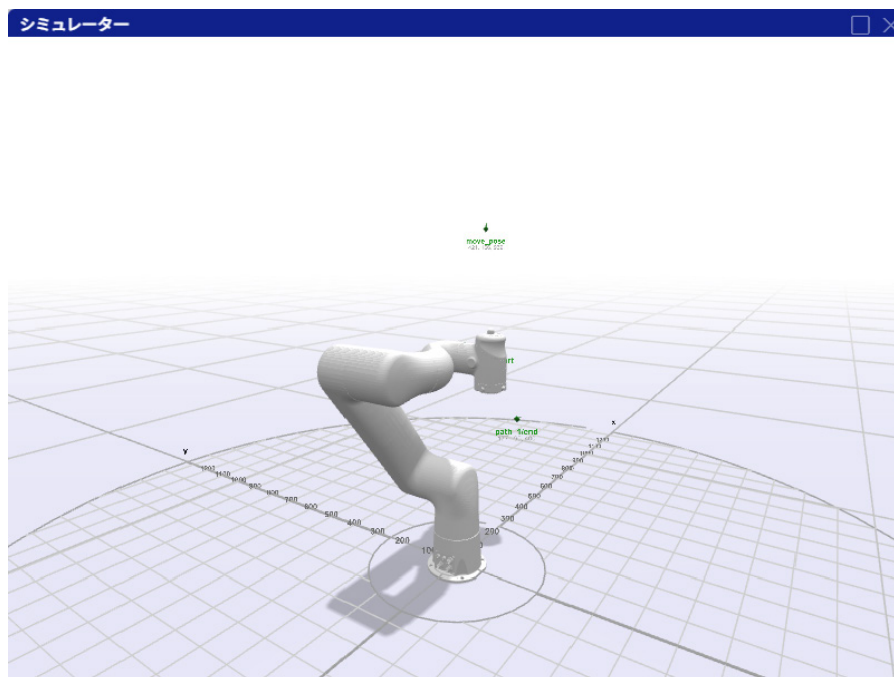


図 4.9: シミュレーターウィンドウ

4.10 ハンドガイド

メインウィンドウのハンドガイドボタンを押すと、ハンドガイドメニューが表示されます。このメニューでは、ハンドガイドモードで動かせるロボットの関節を指定できます。デフォルトでは、ロボットのハンドガイドボタン (およびコックピットの関連ボタン) を押すと、すべての関節を動かします。このメニューで各関節のインジケータをクリックすると、ハンドガイドの対象にする関節を有効化または無効化できます。

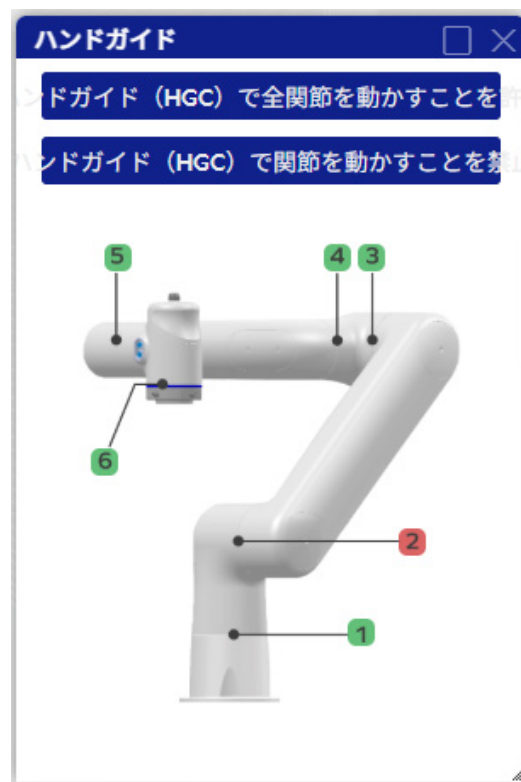


図4.10: ハンドガイドメニュー

4.11. システムモニター

メインウィンドウのシステムモニターボタンを押すと、システムモニターメニューが表示されます。このメニューでは、以下のシステム出力を確認できます。

4.11.1 ロボットの状態

ロボットの状態タブでは、ロボットの現在の状態を確認できます。ベースフレームに対するツール中心点 (TCP)座標、各モーターの位置、電流、温度が表示されます。安全入力の状態も確認できます。

4.11.2 アプリケーションの出力

Print(...)関数を使って、アプリケーションがテキストを返すと、テキストはアプリケーションの出力タブに表示されます。複数のアプリケーションが並列で実行されている場合、すべてのアプリケーションが出力タブにテキストを表示することがあります。

4.11.3 アプリケーションログ

log_message(...)関数を使って、アプリケーションがログデータを返すと、データはアプリケーションログタブに表示されます。ログデータは絞り込みや検索ができます。**CTRL + SHIFT + L**のキーボードショートカットを使用すると、カスタムログと組み込みログをまとめてダウンロードできます。

4.11.4 メッセージ履歴

インターフェース上にメッセージが表示されると、メッセージ履歴タブにも一覧で表示されます。表示状況に応じてメッセージは 4 種類に分類されます。



情報: 役立つ情報やヒントを表示します。



成功: コマンドやアクションが正常に実行されたことを通知します。



警告: 動作を続ける前に注意して読む必要がある重要な情報を通知します。警告は致命的ではなく、プログラムフローを中断しません。



エラー: 軽微なもの (アクションや関数引数の間違い)から重大なもの (ロボットの破損)までさまざまなものがあります。

4.12 ヘルプメニュー

ヘルプメニューでは、現在動作中のシステムやモジュール、使用できるキーボードショートカット、問題発生時の問い合わせ先などの情報を確認できます。

4.12.1 ロボット情報

ヘルプメニューのロボット情報タブでは、AX Portal の環境や接続しているハードウェアの詳細情報を確認できます。ロボット情報のモジュールには、インストール済みモジュールの一覧と、正しく読み込まれているかどうかが表示されます。ステータスが「OK」の場合のみ、モジュールの機能やメニューを使用できます。サポートに連絡するときは、AX Portal と接続ハードウェアの詳細情報を必ず伝えてください。[クリップボードにコピー]ボタンを押して、[等幅フォント](#)で貼りつけたものをサポートに送付してください。

4.12.2 キーボードショートカット

キーボードショートカットタブでは、使用できるショートカットの一覧を確認できます。ショートカットは、コックピットの制御ボタン操作などに使う一般ショートカットと、AX Portal アプリケーションのプログラミングに使うエディターショートカットに分類されています。macOS は Windows や Linux ブラウザーとキーボード配列が大きく異なるため、ショートカット一覧も異なります。

4.12.3 コードのドキュメント

コードのドキュメントタブでは、アプリケーションのプログラミングに使用できるすべての関数を説明したドキュメントが確認できます。ドキュメントは、常に使用できるコア関数と、追加モジュールに関連するアドオン関数に分類されています。タブのダウンロードボタンを押すと、コードのドキュメントを ZIP 形式でダウンロードできます。コードのドキュメントをオフラインで閲覧するには、ブラウザーで"index.html"を開きます。

4.12.4 オープンソースライセンス

オープンソースライセンスタブには、AX Portal で使用するオープンソースライブラリとモジュールの情報が記載されています。

4.12.5 お問い合わせ先

お問い合わせ先タブでは、サポートの連絡先を確認できます。サポートに連絡する前に、必ずマニュアルを確認してください。サポートに連絡するときは、ログファイルをダウンロードして、必ず一緒に送付してください。

4.13 設定メニュー

設定メニューでは、AX Portal の基本設定を変更できます。保存ボタンを押すと変更を適用し、キャンセルボタンを押すと変更を破棄します。リセットボタンを押すと、最後に保存した値に設定を戻します。設定できる項目は以下のとおりです。

設定

GUI設定

- ライブオートコンプリート ☒
- 図面のオブジェクト隠蔽表示 ☒
- フェードアウトするメッセージレベル
- メッセージのフェードアウト時間

コックピットボタンのカスタマイズ

- [ハンドメイド]ボタンを表示 ☒
- [ドリッパー]ボタンを表示 ☒
- [ジョグ]ボタンを表示 ☒
- [製作制御]ボタンを表示 ☒
- [ホーム]ボタンを表示 ☒
- 画面位置
 - 第1箇所
 - 第2箇所
 - 第3箇所
 - 第4箇所
 - 第5箇所
 - 第6箇所

起動設定

- スタートアッププロジェクト
- スタートアップアプリケーション
- イベントのトリガー

マーシオン動作

- 割り込み動作
- 衝突後の待機時間 (秒)
- 衝突後の待機タイムアウト
- 論理による速度制限 ☒

認証設定

- ROSの認証を無効にする ☒
- 特殊の認証を無効にする ☒
- 機群の認証を無効にする ☒
- Modbusの認証を無効にする ☒

デフォルトのTCP通信設定

- エンコード規格
- 読了文字
- ハートビートを送信 ☒

ROS機能設定

- 通信モード
- ROS 2ピアのIPアドレス
- ROS 2ポート

図4.13: 設定メニュー

4.13.1. GUI 設定

このカテゴリでは、ユーザーインターフェースに関する設定ができます。

ライブオートコンプリート	有効にすると、アプリケーションエディターでコード入力中にリアルタイムで自動補完のヒントを表示します。無効にしても、CTRL + SPACE のショートカットで入力中の単語を自動補完できます。
関数のオプション引数を表示	コードに関数スニペットを挿入したときに、関数のオプション引数を表示するかを選択します。アプリケーションのコードに不要な引数を含めないため、無効にすることを推奨します。組み込み関数に慣れていない場合は、この設定を有効にすることを推奨します。
フェードアウトするメッセージレベル	メッセージをフェードアウトするレベルを選択します。すべてのメッセージをフェードアウトするには、最も高いレベルを選択します。すべてのメッセージを固定表示するには、最も低いレベルを選択します。
メッセージのフェードアウト時間	メッセージがフェードアウトする時間を秒単位で指定します (1~60)。

4.13.2 コックピットボタンのカスタマイズ

このカテゴリでは、デスクトップインターフェースとパネルインターフェースのコックピットに表示されるボタンの有効/無効を設定できます。各ボタンの機能について詳細は、「コックピット」を参照してください。機能を使用しない場合や使用できない場合(例: ツールがインタラクティブでなくグリッパーボタンを使えない場合)、またはパネルインターフェースでオペレーターに操作させたくない場合は、ボタンを無効にしてください。

* ボタンを表示	有効にするとこのボタンを表示し、無効にすると非表示になります。
原点位置	原点位置の角度を指定します。この位置は、コックピットのホームボタンに対応します。値はカンマ区切りのテキストで、第 1 関節から第 6 関節の角度に対応します。

4.13.3 起動設定

このカテゴリでは、AX Portal の起動に関する設定ができます。設定を変更すると、起動時に同じ操作を繰り返す必要がなくなります。起動設定で最後に選択した項目 (デフォルト) を選択すると、最後のセッションを記憶し、読み込みます。

スタートアッププロジェクト	起動時に読み込むプロジェクトを選択します。定義済みのすべてのプロジェクトが一覧に表示されます。
スタートアップアプリケーション	プロジェクトが選択されている場合、起動時に読み込むアプリケーションを選択します。プロジェクト内の定義済みのすべてのアプリケーションが一覧に表示されます。
イベントのトリガー	ロボットの電源オンやアプリケーションの実行を起動時に自動実行したい場合は選択します。デフォルトでは、無効に設定されています。トリガーを選択するときは、意図しないアプリケーションの実行を防ぐため、前述の起動設定を事前に行っておく必要があります。

4.13.4 動作中の挙動

このカテゴリでは、ロボット動作中の挙動に関する設定ができます。

割り込み動作	割り込み時の動作を選択します。割り込み動作の詳細や割り込みの解決方法については、「割り込み時の動作」と「割り込みダイアログ」を参照してください。
衝突後の再開試行回数/衝突後の再開タイムアウト	衝突が発生した後に再開を試行する最大回数を設定します。設定したタイムアウト内で衝突が多発した場合、ロボットは再開を中止します。
温度による速度制限	駆動部が高温になった場合に、速度を自動で低減するかを選択します。この設定はデフォルトで有効です。70℃を超えると、過熱防止のため動作が自動で低速になります。90℃を超えると、過熱による損傷防止のため重大なエラーが発生します。この機能の詳細は、「人協働ロボット/6 軸ロボット AX6 マニュアル - 性能に関する制限」を参照してください。

4.13.4.1 割り込み動作

割り込みは、以下の理由で発生します。

- SS1 停止イベントが発生
- SS2 停止イベントが発生
- ロボットが人、物体、または周囲環境と衝突
- 安全停止割り込みが発生

ロボットの関節電流が動的モデルから、しきい値以上に外れると、衝突を検出します。衝突検出は追加の保護を提供する安全機能です。アプリケーションの一部として意図的に使用しないでください。繰り返し衝突すると、関節に負荷がかかり摩耗が早まります。

衝突検出を正しく機能させるため、プログラマーは物体をピックまたはプレースするたびに、ロボットのペイロードを正しい値に設定する必要があります。

物理的な衝突に加えて、ライトカーテンなどの外部安全装置をロボットに接続できます。これらのトリガーは物理的な接触の前に予防的に動作しますが、衝突イベントと同じ方法で処理されます。割り込み動作は、衝突や安全停止による割り込みの処理方法を定義します。

割り込み動作の推奨フローは次のとおりです。

一時停止して許可後再開	実行中のアプリケーションを一時停止し、アプリケーションを継続または停止がダイアログで選択されるまで待機します。
-------------	---

4.13.4.2 割り込みダイアログ

ロボット動作中に割り込み条件が発生すると、**割り込みダイアログ**が表示されます。割り込みの主な原因は、衝突検出、非常停止、安全停止のトリガー、その他の安全関連のイベントです。このダイアログは、条件を安全かつ制御された方法で解決するための手順を案内します。



図 4.13.4.2: 割り込み復旧

このダイアログは、2つのセクションで構成されています。

上部セクション – 割り込み情報と解決方法 このセクションでは、検出された割り込み条件を表示し、解決方法を示します。例として、次の操作があります。

- 衝突後にロボットの経路をクリアする。
- 非常停止やその他の安全装置を解除する。
- 必要に応じて安全条件をリセットする。

割り込みの種類に応じて、**承認**、**続行**、**リセット**などのアクションボタンが表示されます。ユーザーは指示に従って、表示されたボタンを押して操作を続行します。

下部セクション – 安全入力の状態ダイアログの下部では、すべての安全入力の状態をリアルタイムで表示します。ユーザーは、以下のような安全装置のうち、現在有効な装置をすばやく確認できます。

- 非常停止 (ティーチペンダントの非常停止を含む)
- 保護停止
- イネーブルデバイス
- ハンドガイドボタン
- 安全デジタル入力

緑のインジケータは正常状態、赤のインジケータは作動中またはブロック状態 (またはボタンが押されていない状態)を示します。

停止ボタン

停止ボタンは、前回のアプリケーションや動作を続行するか停止するかを選択する必要があるときに使用できます。このボタンを押すと、ロボットが停止し、[準備完了]状態に戻ります。この状態では新しい動作を実行できます。

電源オフボタン

電源オフボタンは、割り込みダイアログで常に使用できます。ロボットの電源をいつでもオフにできます。電源をオフにしても、割り込み状態は自動的に解除されません。原因 (例: 非常停止が押されている)は、電源を入れた後、通常動作を再開する前に解決する必要があります。

一般的なワークフロー

1. 割り込みが発生すると、ダイアログが自動的に表示されます。
2. 上部セクションで検出された条件の説明を確認します。
3. 下部セクションで安全入力の状態を確認し、原因を特定します。
4. 条件を物理的に解消します (例: 非常停止を解除、障害物を取り除く)。
5. 表示されたアクションボタン (承認、続行、リセット)を押して続行します。
6. 必要に応じて、ロボットが通常動作を再開するまで操作を繰り返します。

SLP 違反

割り込み復帰の特別なケースとして、SLP (安全位置監視)違反があります。これは、ロボットが安全設定で指定された位置制限の外へ移動しようとした場合に発生します。この状態になると、割り込みダイアログが表示され、手動モードへの切り替えが案内されます。手動モードでは、ジョグ操作で位置制限の外へ移動できます。ロボットが SLP 違反状態の間は、ジョグ操作以外の動作は許可されません。ロボットの電源がオフになっている場合、または SLP 違反が有効な状態で起動した場合は、次の手順を実行してください。

-
1. 手動モードに切り替えます。
 2. ロボットを起動します。
 3. ジョグメニューに移動します。
 4. ジョグ操作でロボットを位置制限の範囲内に戻します。

4.14 安全設定メニュー

安全設定メニューは、AX シリーズロボットの安全システムを管理する画面です。この画面は、**安全管理者**向けに設計されています。他のユーザーは、安全パラメーターの参照および取得のみ可能で、変更はできません。

4.14.1 ユーザーインターフェース

安全設定メニューを開くと、ロボットの安全システムに設定されているすべての安全パラメーターが表示されます。すべてのユーザーが参照できますが、変更できるのは**安全管理者**のみです。このメニューは複数のサブメニューに分かれています。ウィンドウ下部のページ番号や前後ボタンで項目を切り替えます。

エクスポート	エクスポートボタンを押すと、安全設定を*.yaml 形式でエクスポートします。安全設定のバックアップとして保存したり、ほかのロボットにインポートして同じ安全設定を複製できます。
編集	安全管理者は編集ボタンをクリックし、ユーザーパスワードを入力すると、編集モードに移行します。

編集モードでは、すべての設定項目を編集できます。ロボットの設置条件や使用目的に応じて、安全管理者が安全制限を変更できます。変更した安全設定は青色で表示され、識別しやすくなります。

インポート	インポートボタンを押すと、事前にエクスポートした安全設定ファイルを読み込みます。読み込むと、すべてのサブメニューで手動変更した値が上書きされます。
適用	適用ボタンを押し、ユーザーパスワードを入力すると、すべての安全設定が安全システムに送信され、フラグや制限が検証されます。検証に失敗した場合、無効な項目は赤色で表示されます。検証に成功すると、メニューは検証モードに切り替わります。この時点では設定値は保存されていますが、まだ検証されていません。ロボットを起動するには、検証が必要です。
キャンセル	キャンセルボタンを押すと、安全設定は適用せずに、すべての変更を破棄します。メニューはデフォルトの参照モードに切り替わります。

安全管理者が安全設定の編集を終えて、すべての値が適用されると、メニューは検証モードに移行します。適用済みで未検証の設定は黄色で表示されます。

検証	検証ボタンを押すと、安全管理者は現在表示されているサブメニューの設定を承認済みとしてマークし、次のサブメニューに切り替わります。最後のサブメニューで検証ボタンを押すと、設定を保存する準備が完了します。
----	--

新しい値を確認した後は、安全システムの動作を検証してください。これは、安全検証モードで電源をオンにすることで実施できます。安全検証モードでは、検証中にすべての安全制限を無効にし、安全システムが想定どおりに作動するかをテストできます。

安全設定

安全位置監視 (SLP)

ロボットのベースフレーム内で安全設定が適用される領域を設定します。①

有効な制限は、指定された位置制限から **60 mm** の安全マージンを差し引いて決定されます。

	デフォルト		セーフケース 1		セーフケース 2		セーフケース 3		
	min	max	min	max	min	max	min	max	
x	-3000	3000	-3000	3000	-3000	3000	-3000	3000	mm
y	-3000	3000	-3000	3000	-3000	3000	-3000	3000	mm
z	-3000	3000	-3000	3000	-3000	3000	-3000	3000	mm

この領域をツール中心点 (TCP) に適用するか、ロボットのすべての可動部に適用するかを設定します。

☐ ロボットのTCPのみを制限する
 ☒ ロボットのすべての部品を制限する

ツール安全ゾーン

ツールおよび把持物の安全ゾーンを定義します。ゾーンは指定半径の球体で、フランジフレーム中心からオフセットされます。作業空間の境界内には侵入できません。①

ツール安全オフセット mm
 ツール安全半径 mm

エクスポート

<

1

2

3

4

5

>

編集

図 4.14.1: 設定を編集集中の安全設定メニュー

4.14.2 安全設定

ほとんどの安全設定は、サブメニュー全体で 4 列に分かれています。各列はロボットのセーフケースを定義します。対応する安全デジタル入力 (SDI 1~3) が作動すると、セーフケースが有効になります。デフォルトパラメーターは、安全デジタル入力に関係なく常に有効です。

安全デジタル出力 (SDO) は設定できません。電気仕様の詳細は"ロボットコントローラー RC-A101 マニュアル"を参照してください。

一部の設定には、安全マージンが固定値で設定されています。この値は、安全システムより先にソフトウェアが作動するための許容範囲を指定し、安全違反を防止します。

4.14.2.1 安全位置監視 (SLP)

安全設定の 1 ページ目では、安全位置監視を設定します。ロボットのベースフレームに基づき、ロボットの位置を x, y, z 方向に制限できます。座標フレームがわからないときは、情報アイコンをクリックすると、座標図が表示されます。ロボット全体 (ベースやリンクを含む) の位置を制限するか、ツールセンターポイント (TCP) のみを制限できます。

安全設定

① □ ×

安全位置監視 (SLP)

ロボットのベースフレーム内で安全設定が適用される領域を設定します。①

有効な制限は、指定された位置制限から **60 mm** の安全マージンを差し引いて決定されます。

	デフォルト		セーフケース 1		セーフケース 2		セーフケース 3		
	min	max	min	max	min	max	min	max	
x	-3000	3000	-3000	3000	-3000	3000	-3000	3000	mm
y	-3000	3000	-3000	3000	-3000	3000	-3000	3000	mm
z	-3000	3000	-3000	3000	-3000	3000	-3000	3000	mm

この領域をツール中心点 (TCP) に適用するか、ロボットのすべての可動部に適用するかを設定します。

☐ ロボットの TCP のみを制限する
 ☒ ロボットのすべての部品を制限する

ツール安全ゾーン

ツールおよび把持物の安全ゾーンを定義します。ゾーンは指定半径の球体で、フランジフレーム中心からオフセットされます。作業空間の境界内には侵入できません。①

ツール安全オフセット mm
 ツール安全半径 mm

エクスポート

<

1

2

3

4

5

>

編集

図 4.14.2.1: 安全速度監視 (SLP) 設定例

4.14.2.2 関節角度の安全制限

安全設定の 2 ページ目では、関節角度の安全制限を設定します。デフォルト状態では、安全適合の関節角度監視は行われません。セーフケースを有効にすることで、関節角度の安全監視が有効になります。

ただし、セーフケースにおける関節角度の監視範囲は $\pm 180^\circ$ に制限されます。このため、ロボットの仕様上の可動範囲に対して、動作範囲が制限される場合があります。この制限の影響を回避するため、第 1, 4, 6 関節については、関節ごとに監視を無効にすることができます。

関節角度の監視は、下限値および上限値の両方が設定されている場合にのみ有効になります。下限値および上限値がともに未設定の場合、その関節の監視は無効になります。

安全設定

関節角度の安全制限

ロボットの各関節の位置制限を設定します。①

デフォルトの位置制限は関節の物理的制約により決定され、変更できません。

一部の関節は **180 deg** を超えて動くことがありますが、安全なケースではこれらの範囲はサポートされません。制限を無効にすると、関節の全可動域を使用できます。

有効な制限は、指定された位置制限から **5 deg** の安全マージンを差し引いて決定されます。

	デフォルト		セーフケース 1		セーフケース 2		セーフケース 3		deg
	min	max	min	max	min	max	min	max	
関節 6	無効		無効	無効	-179.9	179.9	-179.9	179.9	
関節 5	無効		-160	160	-160	160	-160	160	
関節 4	無効		無効	無効	-179.9	179.9	-179.9	179.9	
関節 3	無効		-165	165	-165	165	-165	165	
関節 2	無効		-160	160	-160	160	-160	160	
関節 1	無効		無効	無効	-179.9	179.9	-179.9	179.9	

エクスポート < 1 2 3 4 5 6 > 編集 検証を開始

図 4.14.2.2: 関節角度の安全制限設定例

4.14.2.3 安全速度監視 (SLS)

安全設定の 3 ページ目では、安全速度監視を設定します。ロボット関節の速度を個別に制限できます。ツール、ベース、リンクを含むロボット全体の動作速度を制限することもできます。

安全設定

安全速度監視 (SLS)

ロボットの各関節およびすべての可動部の速度制限を設定します。

有効な制限は、指定された速度制限から **7 deg/s** または **100 mm/s** の安全マージンを差し引いて決定されます。

	デフォルト	セーフケース 1	セーフケース 2	セーフケース 3	
関節 6 の速度	247	247	247	247	deg/s
関節 5 の速度	247	247	247	247	deg/s
関節 4 の速度	247	247	247	247	deg/s
関節 3 の速度	187	187	187	187	deg/s
関節 2 の速度	187	187	187	187	deg/s
関節 1 の速度	187	187	187	187	deg/s
速度	2000	2000	2000	2000	mm/s

エクスポート

< 1 2 3 4 5 >

編集

図 4.14.2.3: 安全速度監視 (SLS) 設定例

4.14.2.4 安全なパワーおよび力の制限設定 (PFL)

安全機能設定の 4 ページ目では、パワーおよび力に関する項目を設定します。

- セーフケースごとに、ロボットが人や物との衝突など外力に対してどの程度敏感になるかを定める力の制限値を設定できます。力の制限値はツールセンターポイント (TCP) に適用されます。
- 平均ペイロード (運転中の適切なペイロード値)を設定することが重要です。ペイロードはツールの重量と把持物の合計で定義されます。これにより、力の制限がアプリケーションに応じてより適切になります。
- ロボットの重力ベクトルは、設置方向にあわせる必要があります。ロボットが床面に設置されている場合、重力ベクトルは[0, 0, -9.81]です。ロボットが壁や斜面、天井に設置されている場合、重力ベクトルを事前に計算し、正しく入力する必要があります。

安全設定

力とパワーの安全制限 (PFL)

ツール中心点 (TCP) で許容される最大力を定義します。この制限を超えると衝突割り込みが発生します。制限を上げると衝突検出の感度が低下します。ペイロードの変動が大きい場合、計算誤差を考慮して力の制限を増やす必要があります。

	デフォルト	セーフケース 1	セーフケース 2	セーフケース 3	
力の制限	250	250	250	249	N

平均可搬質量

ロボットの平均可搬質量 (ツールと把持物の総重量) を設定します。運転中は物の把持や配置により質量が変化します。安全システムは平均値を用いて、加わる力が許容範囲内かを評価します。

平均可搬質量 kg

重力ベクトル

ロボットの取り付け方向に応じて、ベースフレーム内の重力ベクトルを設定します。

	x	y	z	
重力ベクトル	0	0	-9.81	m/s ²

エクスポート < 1 2 3 **4** 5 > **編集**

図 4.14.2.4: 安全なパワーおよび力の制限設定

4.14.2.5 安全 I/O およびオペレーターの制限

安全機能設定の 5 ページ目では、安全関連 I/O の動作およびオペレーター権限を設定できます。保護停止の動作、手動モードでの入力監視、リセット条件、モード切り替えの権限、およびモード変更時のパスワード可否を定義できます。

図4.14.2.5: 安全 I/O 設定

4.14.2.6 検証データ

安全機能設定の最終ページでは、安全機器に関する情報が表示されます。

- 安全機器から読み取ったチェックサム (トラブルシューティング用)
- 転送カウンター (安全パラメーターが書き込まれた回数)
- 安全パラメーターの最終更新日時
- 安全パラメーターの検証状態
- 安全機器のシリアル番号とファームウェアバージョン



安全設定

検証

このセクションでは、安全検証に使用するデータを表示します。チェックサムはすべての保存値から生成されます。タイムスタンプは最後の更新時刻を示します。検証ステータスは、最新の変更が安全管理者により確認・承認されたかを示します。

チェックサム AFB	4120550240
チェックサム LDV	3123018406
チェックサム PDV	2543847708
チェックサム PEV	2129800727
チェックサム TOC	1955707527
転送カウンター	180
最終更新	2026-04-18 08:40:14
検証済み	🟢

安全装置のバージョン管理

シリアル番号	231
ファームウェアバージョン	1.0.8.24.0

エクスポート < 1 2 3 4 5 6 > 編集 検証を開始

図4.14.2.6: 検証設定

4.14.2.7 安全出力機能

特定の安全機能の論理状態は、ロボットコントローラーの安全出力に関連づけられています。詳細は、「ロボットコントローラー RC-A101 マニュアル」を参照してください。

4.14.2.8 オプション機器の接続

タブレットホルダーやイネーブルデバイスなどのオプション機器の接続について詳細は、「ロボットコントローラー RC-A101 マニュアル」を参照してください。

5. パネルインターフェース (GUI)

5.1 概要

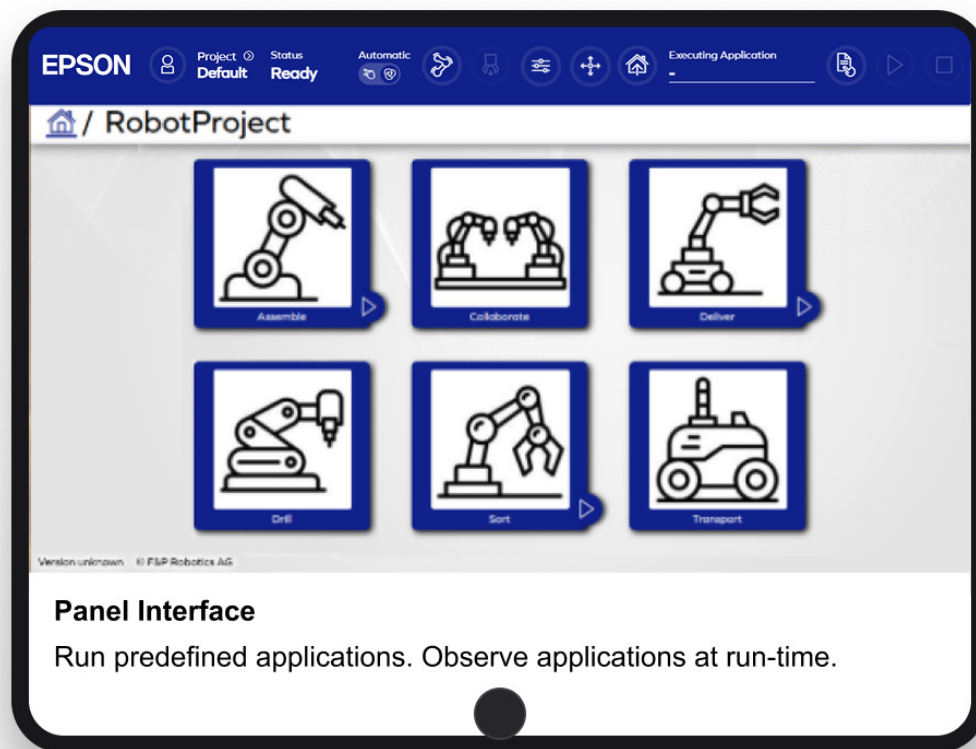


図 5.1.a: パネルインターフェースの画面例

パネルインターフェースは、メインアプリケーションにすばやくアクセスできるように設計された、デスクトップインターフェースと異なるインターフェースです。このインターフェースは、顧客のニーズに柔軟に対応できるよう設計されており、特にアプリケーションとその状態情報のみを表示します。パネルインターフェースはタッチスクリーンに対応し、最新のモバイル機器で使うアプリのような構成です。そのため、技術的な知識がなくても直感的に操作できます。また、パネルインターフェースを使用すれば、プログラミングやアプリケーションのコードを意識する必要はありません。該当するアプリをクリックするだけで、アプリケーションが起動します。管理インターフェースにログインし、[パネルインターフェースに移動]ボタンを選択するとアクセスできます。



図5.1.b: パネルインターフェースへのアクセス方法

一部のブラウザ (特に Windows) では、表示スケーリングやズームがデフォルトで高い値 (例: 150%) に設定されている場合があります。この設定により、インターフェースの一部 (コックピットやロゴなど) が切れて表示されることがあります。快適に使用するため、表示スケーリングやズームを 100% (またはそれ以下) に設定することを推奨します。

5.2 パネルインターフェース用アプリケーションの設定

初期状態では、パネルインターフェースにはアプリケーションが表示されません。オペレーターがすぐに実行できるように、表示するアプリケーションを指定します。パネルアプリケーションの設定手順は以下のとおりです。



図5.2.a: アプリケーションエディター (デスクトップインターフェース) でのパネルインターフェース関連設定

パネルインターフェースに関連する設定: 2., 3., 4.

1. デスクトップインターフェースでアプリケーションを編集し、アプリケーション設定を開きます。
2. パネルインターフェースで表示を有効にします。
3. 必要に応じて、パネルインターフェースのメインメニューでアプリケーション名とともに表示するアプリケーションの画像をアップロードします (推奨比率は 5:4)。
4. 必要に応じて、変数タブで状態変数を追加します。これにより、パネルインターフェースで変数の監視や変更が可能になります。
5. アプリケーションを保存します。

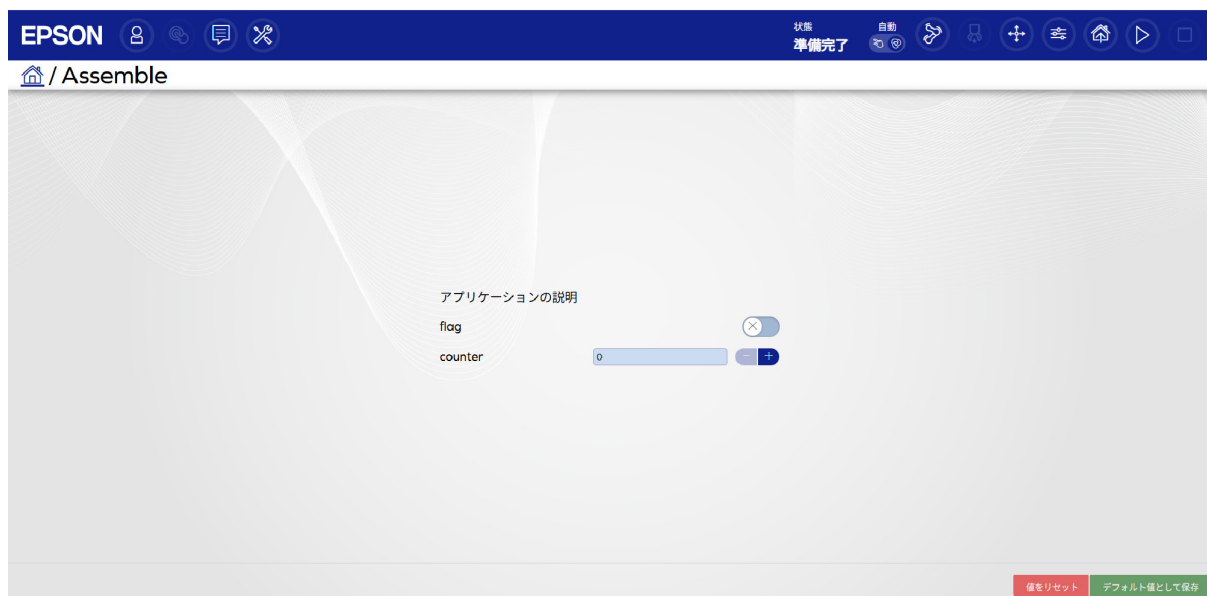


図 5.2.b: 状態変数を表示したパネルアプリケーション

5.3 コックピットと設定

パネルインターフェースのコックピットには、デスクトップインターフェースと同様のボタンが表示されます。各アイコンの詳細は「コックピット」を参照してください。これらの要素に加えて、左側にはパネルインターフェース専用のボタンが表示されます。



パネルボタンを押すと、パネルのメインページを表示し、アプリケーションを選択、実行できます。



メッセージボタンを押すと、メッセージ履歴が開き、警告やエラーを含むすべてのメッセージを時系列で確認できます。



設定ボタンを押すと、設定画面を表示します。この画面には、以下の重要な設定が含まれます。

- 接続済みツールの選択
- ロボットの電源オン/オフ
- ログファイルのダウンロード

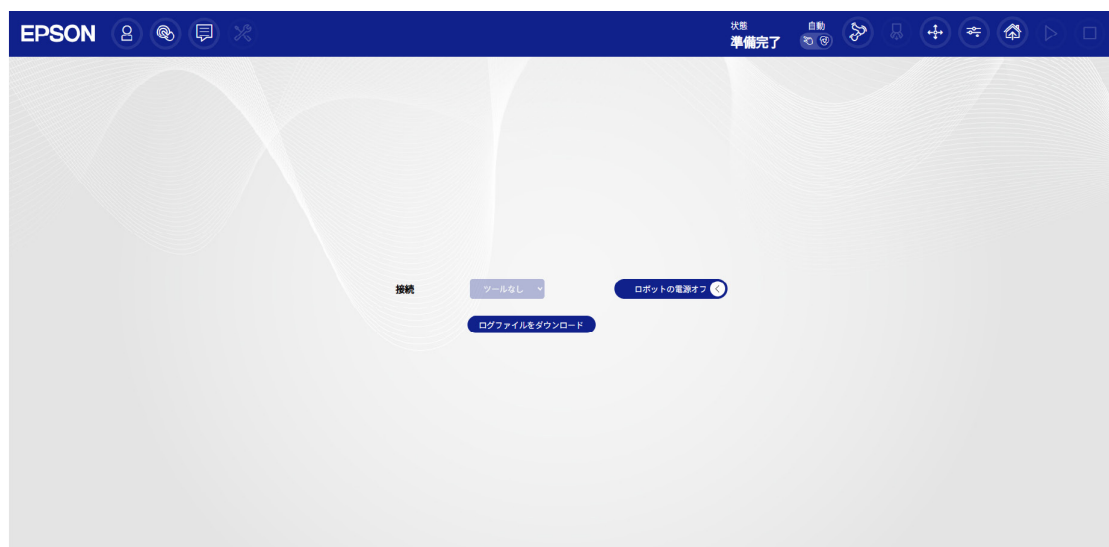


図 5.3: パネル設定画面

5.4 アプリケーションの制御

パネルインターフェースのメイン画面には、すべてのアプリケーションが名前と画像付きで一覧表示されます。パネルは 2 種類のアプリケーションに分類されます。

- **状態変数を持たないアプリケーション**には、再生ボタンが表示されます。アプリケーションボタンを押すと、アプリケーションが起動します。
- **状態変数を持つアプリケーション**には、再生ボタンが表示されません。アプリケーションボタンを押すと、アプリケーションの詳細画面が開き、すべての状態変数を表示します。この画面では、編集可能な状態変数を変更し、再生ボタンを押してアプリケーションを実行できます。戻るボタンを押すと、パネルのメイン画面に戻ります。

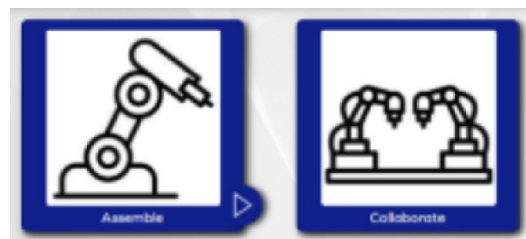


図 5.4: アプリケーションボタン

6.キネマティクス定義

ロボットキネマティクスは、ロボットの動作と、その動作の記述方法を定義します。キネマティクスを扱うには、利用できるキネマティクスパラメーターとオブジェクトの定義を理解する必要があります。

6.1 座標

座標はロボットの絶対位置と向きを定義します。位置は mm 単位の 3 要素ベクトル[x, y, z]で、向きは deg 単位の 3 要素ベクトル[ロール, ピッチ, ヨー]で表します。

初期化は以下のように実施します。

```
coord = Coordinates([600, 0, 300, 180, -90, 0])
# ... this is similar to ...
coord = Coordinates({"x": 600, "y": 0, "z": 300, "roll": 180, "pitch": -90, "yaw": 0})
```

各要素には以下のようにアクセスします。

```
coord = Coordinates([600, 0, 300, 180, -90, 0])

# accessing individual elements (x, y, z, roll, pitch, yaw)
x = coord.x
x = coord["x"]
x = coord[0]

# accessing position or orientation
position = coord.position
orientation = coord.orientation
```

6.1.1 航法角

航法角は座標の向き (ロール、ピッチ、ヨー) を定義する方法のひとつです。航法角は、航空機の向きを表すために一般的に使用されます。航法角では、3 次元の回転は以下のように定義されます。

- 1: -Z 軸周りの回転をヨー (ψ) 方向として定義
- 2: -Y 軸周りの回転をピッチ (θ) 方向として定義
- 3: +X 軸周りの回転をロール (ϕ) 方向として定義

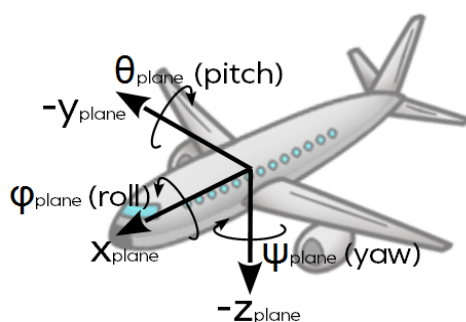


図 6.1.1.a: 航法角座標系

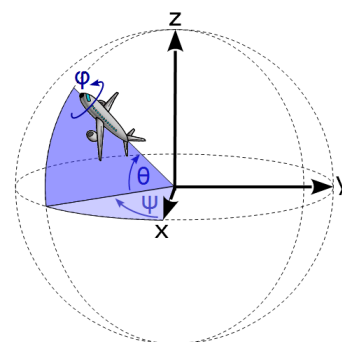


図 6.1.1.b: 航法角回転順序

ヒント: エンドエフェクターの 3 次元姿勢を扱うときは、座標系が一致するようにエンドエフェクターに飛行機が取り付けられていると想像すると理解しやすくなります。この方法で、向きを示す角度 (ロール、ピッチ、ヨー) を把握しやすくなります。

6.1.2 座標系

AX シリーズロボットを操作するには、複数の座標系を理解することが重要です。

- ロボットのベースフレームは、Z 軸が底面から垂直に離れる方向、X 軸が底面と平行でベース背面の端子から離れる方向に定義されます。Y 軸は右手の法則に従い定義されます。
- ロボットのワークフレームは、ベースフレームに対するユーザー定義のフレームです。指定しない場合はベースフレームと同一です。座標はこのフレームで定義されます。このフレームを使って、ロボットのカスタム作業領域を定義できます。
- ロボットのフランジフレームはロボットに取り付けられたツールによって異なります。
 - ツールが取り付けられていないときは、ロボット上部のフランジ (接続プレート) の中心がフレームの原点です。すべての関節角度が 0 のとき、フランジフレームの向きはベースフレームと一致します。
 - カスタムツールが取り付けられ、ツールエディターで定義されているときは、ツールで指定された位置と向きにフレームが移動します。
- ロボットのツールフレームは、フランジフレームに対するユーザー定義のフレームです。指定しない場合はフランジフレームと同一です。座標はこのフレームの原点に設定されます。このフレームを使ってツールセンターポイントを定義できます。

座標系の例として、ベース座標フレームとフランジ座標フレームを示した以下の図を参照してください。

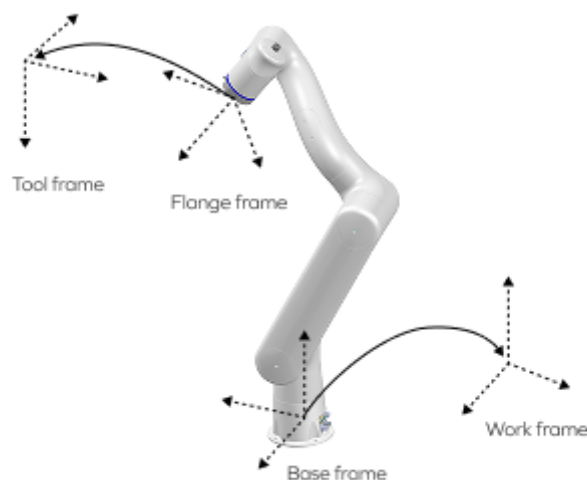


図 6.1.2: ロボット座標フレーム

6.2 姿勢設定

ロボットの位置と向きが与えられた場合、逆キネマティクス計算により、複数の異なる関節構成が得られます。これらの構成は、各関節の曲がり方や回転方向が異なります。姿勢設定は、これらの候補の中から特定の関節構成を1つ選択するために使用します。一般的な6軸ロボットでは、エンドエフェクターの位置と姿勢が同じであっても、肩、肘、手首の向きの組み合わせにより、最大8通りの姿勢構成が存在します。これらの構成を識別するため、姿勢設定は"c1"~"c8"と名前がついています。

姿勢設定は、2つのポーズを比較したり、パスがシームレスかどうかを確認する目的には適していません。姿勢設定を変更は、必ずしもパスを非シームレスにするわけではありません。また、非シームレスなパスが必ずしも姿勢設定の変更を含むとは限りません。

6.3 ポーズ

ポーズはロボットの座標と姿勢設定を定義します。データベース内の定義済みポーズをIDまたは名前で指定できます。座標と姿勢設定を指定してポーズの作成をするか、関節角度を指定して順キネマティクス関数でポーズを作成できます。

初期化は以下のように実施します。

```
# read predefined pose from database by index
pose = Pose(1)
# read predefined pose from database by name
pose = Pose("fancy_pose")
```

6.4 ポイント

ポイントは、パスを構成する離散要素です。ロボットのポーズに加えて、速度と前のポイントからの経過時間を指定します。PVT (Position, Velocity, Time)と呼ばれ、通常は関節角度ベクトル、関節速度ベクトル、時間間隔の浮動小数で保存されます。拡張版のポイントでは、ロボットの加速度を含める場合があります。

6.5 パス

パスは、ロボットが2つ以上のポーズ間をどのように移動するかを定義します。連続パス (CP)はツール空間でのみ定義され、複数の直線または円弧で構成されます。パスは離散化され、関節空間に変換されます。離散化したパスで、ロボットが姿勢設定変更なしで移動できる場合、パスはシームレスと呼ばれます。パスに沿って加速度が連続している場合、パスは連続的と呼ばれます。

パスは座標ではなく、指定したポーズから開始します。パスの姿勢設定は初期ポーズで決まるため、パスのセグメントは通常座標で定義されます。前の離散ポイントに最も近い逆キネマティクスの解を使用することで、可能な限り自動的にシームレスなパスが生成されます。ここでは、パスと軌跡を同じ意味で使用します。ただし、パスには技術的に速度情報は含まれません。

初期化は以下のように実施します。

```
# read predefined path from database by name
path = Path("fancy_path")
```

6.6 セグメント

セグメントはパスを構成する要素であり、TCP が目標に向かう動作を定義します。

使用可能なセグメントの種類は以下のとおりです。

- **直線補間:** TCP は、セグメントで定義された目標座標に向けて直線的に移動します。
- **円弧補間:** TCP は現在のポーズから定義された中間位置を経由し、セグメントで定義された目標座標に向けて円弧上を移動します。動作中の TCP の向きは、アルゴリズムによって自動的に決定されます。
- **座標:** TCP は最も効率的な経路で目標座標に移動します。これは関節空間での直線移動に相当します。
- **ポーズ:** TCP は最も効率的な経路で目標ポーズに移動します。これは関節空間での直線移動に相当します。
- **方向:** TCP は現在の位置を維持したまま、指定された目標方向に到達するために回転動作を行います。

セグメントごとに、速度と加速度を指定できます。

- 方向セグメントでは、角速度 [deg/s]および角加速度 [deg/s²]に相当します。
- ほかのセグメントでは、直線速度 [mm/s]および直線加速度 [mm/s²]に相当します。

ブレンド半径も指定できます。この値は、前のセグメントの終了ポーズから現在のセグメントの開始ポーズまでの最大許容偏差を mm 単位で定義します。ブレンド半径が 0 mm の場合、TCP は定義されたポーズを正確に通過します。値を大きくすると、セグメント間のパスが曲線となり、より滑らかで自然な遷移が可能になります。

6.7 グリッド

グリッドは最大 3 次元までカスタマイズ可能なポイント配列で、事前定義されたトレイの位置などを指定し、座標値のセットをまとめて保存できます。通常、グリッドは 2 次元配列として扱います。コンベアーからワークをピックアップする場合のように 2 次元を使わないポイントリストや、箱にワークを積み重ねる場合のような 3 次元構造も作成できます。

7.接続モジュール

AX Portal は、ロボットの制御や監視をするための直感的な GUI に加え、TCP、REST、ROS、Modbus などの通信方式を使用してロボットを制御、監視できる接続モジュールを備えています。これらの通信方式は、それぞれ異なる用途に適しています。

- **TCP** は、高速で継続的なデータ交換が必要なリアルタイム通信に最適です。直接制御アプリケーション、カスタムクライアントの実装、または永続的なソケット接続を必要とするシステムとの統合で一般的に使用されます。
- **REST** は、ウェブベースのサービスまたはアプリケーションとの統合に最適です。リアルタイム性能よりも、使いやすさ、拡張性、および標準的な HTTP ベースのシステムとの互換性が重視される場合に使用されます。
- **ROS** (Robot Operating System) は、ロボットの研究および開発向けに設計されています。ロボットを複雑なロボットシステムに統合する場合に特に有用で、センサーフュージョン、自律動作、他の ROS 対応デバイスとの連携などの機能を実現できます。
- **Modbus** は、産業用オートメーション環境で一般的に使用されます。信頼性と標準化が重要となる PLC (Programmable Logic Controllers) や、その他の産業機器との通信に適しています。

本章では、これらのモジュールの使用方法を説明します。

7.1 接続・認証・アクティブ制御

接続モジュールを使用する前に、モジュールとクライアントの接続方法、認証の仕組み、AX Portal におけるアクティブ制御の原理を理解することが重要です。

7.1.1 接続チャンネル

端末とロボットは、必ず特定の接続チャンネルを通じて接続されます。接続チャンネルの実装は、モジュールごとに異なります。

TCP モジュールは複数チャンネルをサポートしています。ロボットに接続する端末は、それぞれ独自の接続チャンネルを作成し、個別にログインします。いずれかのチャンネルを閉じると接続が終了し、アクティブ制御も解除されます (詳細は後述します)。

 警告	REST、ROS、Modbus モジュールは単一のチャンネルのみサポートしています。一度接続が確立し認証が完了すると、どの端末からでもロボットにコマンドを送信できます。このチャンネルは AX Portal の起動時に生成され、接続端末から閉じることできません。
---	---

7.1.2 認証

AX Portal では、安全な二段階認証を採用しています。はじめに、「セキュア認証」に記載されたリモートログイン手順に従って、有効なアクセストークンを取得します。この手順は必ず別の **TCP** 接続を通じて実行する必要があります。ログインしてアクセストークンを作成したら、選択した接続チャンネルで **token_login** コマンドを使用し、アクセストークンを登録してユーザーを認証します。

 注意	設定メニューで TCP 認証を無効にすると、認証手順を省略できます。 Modbus モジュールでは、 token_login コマンドを送信できません。使用するには、設定メニューで Modbus モジュールの認証を無効にする必要があります。
---	--

7.1.3 アクティブ制御ステーション

AX Portal には、アクティブ制御ステーションを保護する機能があります。他の接続チャンネルからはロボットを同時に制御できません。この保護機能がない場合、危険な状況になる可能性があります。

接続チャンネルがアクティブ制御ステーションになるには、**request_active_control** コマンドを使用します。このコマンドが正常に実行されると、チャンネルを閉じるかロボットの電源が切れるまで、その制御ステーションが制御権を保持します。ほかの制御ステーションが制御権を持っている場合、このコマンドは制御ステーションが制御要求を許可するまで待機します。

アクティブ制御を解除するには、アクティブ制御ステーションで **release_active_control** コマンドを使用します。チャンネルを使用しなくなったら、アクティブ制御を解除することを推奨します。解除しない場合、そのチャンネルがアクティブ制御を占有してしまいます。

7.2 TCP 接続モジュール

7.2.1 接続の確立

TCP 接続モジュールのサーバー側は [Python](#) ベースで構築され、標準の [socket ライブラリ](#) を使用して実装されています。デフォルトでは、**ポート番号 38152** を使用してサーバーに接続できます。メッセージは **UTF-8** でエンコードされます。メッセージは **End of Text** 文字 0x03 (Python では "\x03") で区切られます。ロボットに接続し、受信した応答をすべて表示するには、以下のサンプルスクリプトを使用します。受信メッセージの意味については、「受信コマンド」を参照してください。

```
import socket
import time

# define the robot ip address
ip_address = "10.0.0.203"

# create the tcp socket
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
    # connect to the server
    client_socket.connect((ip_address, 38152))

    # prepare message buffer
    message_buffer = ""
    while True:
        # read data and add it to message buffer
        try:
            message_buffer += client_socket.recv(1024).decode("UTF-8")
        except socket.timeout:
            time.sleep(0.1)
            continue

        # check for complete messages in buffer
        if "\x03" in message_buffer:
            # extract complete messages
            message_split = message_buffer.split("\x03")
            for message in message_split[:-1]:
                print(message)
            # incomplete message (if any) remains in buffer
            message_buffer = message_split[-1]
```

7.2.2 コマンドの送信

ロボットにコマンドを送信する前に、「接続・認証・アクティブ制御」で説明している原理を理解してください。以下の例では、初期の **token_login** コマンドと **request_active_control** コマンドを送信します。

```
import json
import socket
import time

# define the robot ip address, access token, and session identifier
ip_address = "10.0.0.203"
token = "MY_ACCESS_TOKEN"
session_id = "MY_SESSION_ID"

# create the tcp socket
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
    # connect to the server
    client_socket.connect((ip_address, 38152))

    # use the authentication token to log in (needed if authentication is not disabled)
    request = {"action": "token_login", "bridge": "core",
               "arguments": {"token": token, "session_id": session_id}}
    message = json.dumps(request) + "\x03"
    client_socket.sendall(message.encode("UTF-8"))

    time.sleep(0.5)

    # make sure to acquire active control, which makes the TCP connection the active
    # control station of the robot (only one control station can be active at any time)
    request = {"action": "request_active_control", "bridge": "core",
               "arguments": {}}
    message = json.dumps(request) + "\x03"
    client_socket.sendall(message.encode("UTF-8"))

    time.sleep(0.5)

# add more robot control commands here
```

7.2.3 設定

上記の例では、デフォルトの TCP 設定を使用しています。これらの設定は、デスクトップインターフェースの設定メニューで変更できます。TCP 認証を無効化したり、TCP メッセージのエンコードや終了文字を変更できます。接続が有効かどうかを定期的に確認するために、ハートビートを有効化できます。



認証設定

TCPの認証を無効にする ☐

デフォルトのTCP通信設定

エンコード規格

終了文字

ハートビートを送信 ☐

図 7.2.3: TCP モジュール設定

7.2.3.1 エンコード規格

TCP 接続で送信するメッセージのエンコードを指定します。一般的な [Python エンコード規格](#) が使用できます。一般的に使用される規格は、デフォルトの UTF-8 と ASCII です。

7.2.3.2 終了文字

TCP 接続で送信するメッセージの終了文字を変更できます。終了文字はメッセージの終わりを示します。メッセージを受信すると、終了文字は自動的に削除されます。

7.2.3.3 ハートビートを送信

TCP サーバーがハートビートを自動で送信するかを選択します。ハートビートは 50ms ごとに終了文字のみの空メッセージを送信します。

7.2.4 メッセージプロトコル - コマンドの受信

送受信するすべてのコマンドは [JSON](#) 形式の文字列です。メッセージは **UTF-8** でエンコードされ、**End of Text** 文字 0x03 (Python では"\x03")で区切られます。受信したコマンドは、キーと値を持つ JSON 形式の辞書型データです。キーはメッセージの種類を定義します。値はメッセージの内容が含まれます。ロボットからメッセージを受信するコードの例について詳細は、[接続の確立](#)を参照してください。

この一覧はすべてのコマンドを網羅していません。主要なコマンドのみを記載しています。

7.2.4.1 Status Info

ロボットの状態は変更されるたびに送信されます。

キー	値	タイプ
status_info	以下の一覧に基づく現在のロボットの状態。 ● 0: 電源オフ ● 1: 電源オン中 ● 2: 読み込み中 ● 3: 準備完了 ● 4: 処理中 ● 5: キャリブレーション中 ● 6: 実行中 ● 7: 自律モード ● 8: 一時停止 ● 9: 続行 ● 10: 保護停止 ● 11: 安全停止 ● 12: 位置制限 ● 13: ハンドガイド ● 14: 衝突 ● 15: 非常停止 ● 16: 通常停止 ● 17: エラー ● 18 電源オフ中 ● 19: 停止中	整数

```
{"status_info": 0}
```

7.2.4.2 Action Return Value

アクションを実行すると、アクションのコマンドを含む戻り値が送信されます。アクションに戻り値がない場合、null が送信されます。関数の実行中にエラーが発生しなかった場合のみ、success は true になります。シーケンス番号はオプションのパラメーターで、コマンド送信時にシーケンスを指定した場合のみ追加されます。

キー	値	タイプ
action_return_value	実行したアクション名、戻り値、アクションの成功/失敗。エラーが発生した場合、アクションは失敗します。	辞書

```
{"action_return_value": {"action": "release", "value": null,
                        "sequence": 100, "success": true}}
```

7.2.4.3 Error

エラーが発生すると、詳細情報を含むエラーメッセージが送信されます。

キー	値	タイプ
error	エラー情報。	辞書

```
{"error": {"text": "The argument 'variable_value' is missing for the
               'set_state_variable' action.",
          "title": "Minor Error",
          "color": 3,
          "code": "e0010035",
          "time": 1554907874.0436678}}
```

7.2.4.4 Message History

ユーザーインターフェースに送信された過去のメッセージ一覧です。

キー	値	タイプ
message_history	メッセージに含まれるテキスト、タイトル、コード、タイムスタンプ。	リスト

```
{"message_history": [{"text": "The specified bridge does not exist.",
                        "title": "Minor Error", "color": 3,
                        "code": "e0010019",
                        "time": 1554897027.5446029}]}
```

7.2.4.5 State Group Define

ロボットの位置データを定義します。ステートグループには、格納データの内容を示す ID が付与されます。この定義を使用して、ステートグループデータを識別できます。

キー	値	タイプ
state_group_define	ステートグループの定義。	辞書

```
{"state_group_define": {"id": "pose",  
  "label": "tcp_pose",  
  "members": [{"id": "x", "label": "x"},  
    {"id": "y", "label": "y"},  
    {"id": "z", "label": "z"},  
    {"id": "roll", "label": "roll"},  
    {"id": "pitch", "label": "pitch"},  
    {"id": "yaw", "label": "yaw"}],  
  "cols": [{"id": "value", "unit": "mm, deg"}]}}
```

7.2.4.6 State Group Data

ステートグループデータには、TCP データまたはモーターデータが含まれます。データを受信するには、"request_data"を使用します。

キー	値	タイプ
state_group_data	値に含まれるロボットの TCP (id = "pose")、モーターデータ (id = "robot")、安全フラグ (id = "safety")。	辞書

```
{"state_group_data": {"id": "pose",  
  "data": {"x": {"value": "0.0"},  
    "y": {"value": "0.0"},  
    "z": {"value": "1378.5"},  
    "roll": {"value": "0.0"},  
    "pitch": {"value": "90.0"},  
    "yaw": {"value": "180.0"}}}}}
```

7.2.4.7 Open Dialog

UI 上でダイアログが開くと、"open_dialog"コマンドを受信します。ダイアログには、"answer_dialog"アクションで応答できます。

キー	値	タイプ
open_dialog	ダイアログに関する以下の値。 ● id: ダイアログの一意的 ID (ダイアログへの応答に使用) ● kind ○ 0: 確認ダイアログ (確認/拒否ボタン) ○ 1: 入力ダイアログ (テキスト入力) ○ 2: ファイルダイアログ (ファイルアップロード) ○ 3: ドロップダウンダイアログ (選択肢リストが必要) ○ 4: 選択ダイアログ (選択肢リストが必要) ● default: 応答がない場合の戻り値 ● color ○ 0: 情報 ○ 1: 成功 ○ 2: 警告 ○ 3: エラー ● title: 表示するタイトル ● text: 表示するテキスト ● options: ドロップダウン/選択ダイアログ用の選択肢	辞書

```
{"open_dialog": {"id": "5cee85a8-9ab2-4de8-b1ea-52ded8cc68c0",  
  "kind": 0, "default": true, "color": 2,  
  "title": "Movement Warning",  
  "text": "In order to Calibrate, the robot will move.  
            Do you want to proceed?"}}
```

7.2.4.8 Script Send

スクリプトから送信されたメッセージを受信します。

キー	値	タイプ
script_send	スクリプトから送信された値。	多様なデータ型

```
{"script_send": 123}
```

7.2.5 メッセージプロトコル - コマンドの送信

送受信するすべてのコマンドは **JSON** 形式の文字列です。メッセージは **UTF-8** でエンコードされ、**End of Text** 文字 0x03 (Python では"\x03")で区切られます。送信するコマンドは、3 つの必須キーと 1 つの任意キーを含む JSON 形式の辞書型データである必要があります。ロボットにメッセージを送信するコードの例について詳細は、「コマンドの送信」を参照してください。

action	実行するコマンドを指定します。	str
bridge	コマンドを含むモジュールを指定します。通常は"core"です。	str
arguments	コマンドの引数を指定します。引数の詳細は後述します。	辞書
sequence	(任意)コマンドの識別子で、action_result_value の末尾に返されます。	int

この一覧はすべてのコマンドを網羅していません。主要なコマンドのみを記載しています。

7.2.5.1 Token Login

アクティブ接続を使用するには、最初にログインコマンドを送信する必要があります。「セキュア認証」に記載されたりリモートログイン手順に従って、有効なアクセストークンを取得します。

Action	token_login	
Bridge	core	
Argument キー	token	session_id
Argument 値	ログインコマンドで発行される個別のログイントークン	ログインコマンドで発行される一意のセッション ID
Argument タイプ	文字列	文字列

```
{"action": "token_login",  
 "bridge": "core",  
 "arguments": {"token": "MY_ACCESS_TOKEN",  
               "session_id": "MY_SESSION_ID"}}
```

7.2.5.2 Request Active Control

TCP 接続をロボットのアクティブ制御ステーションにするため、アクティブ制御を取得します。アクティブ制御ステーションにリクエストが送信され、許可/拒否が判断されます。

Action	request_active_control
Bridge	core

```
{"action": "request_active_control",  
  "bridge": "core",  
  "arguments": {}}
```

7.2.5.3 Release Active Control

TCP 接続をロボットのアクティブ制御ステーションから解除し、アクティブ制御を解除します。

Action	release_active_control
Bridge	core

```
{"action": "release_active_control",  
  "bridge": "core",  
  "arguments": {}}
```

7.2.5.4 Request Data

このコマンドはハートビートとして使用できます。さまざまな種類のデータを受信します。

Action	request_data
Bridge	core

```
{"action": "request_data",  
  "bridge": "core",  
  "arguments": {}}
```

7.2.5.5 Power Up

ロボットのモーターをオンにします。

Action	power_up
Bridge	core
Argument キー	tool_id
Argument 値	カスタムツールの ID、または 0 (ツールを自動検出)
Argument タイプ	整数または文字列

```
{"action": "power_up",  
  "bridge": "core",  
  "arguments": {"tool_id": 0}}
```

7.2.5.6 Power Down

接続を終了し、ロボットのモーターをオフにします。

Action	power_down
Bridge	core

```
{"action": "power_down",  
  "bridge": "core",  
  "arguments": {}}
```

7.2.5.7 Pause

実行中の動作関数やスクリプトを一時停止します。

Action	pause
Bridge	core

```
{"action": "pause",  
  "bridge": "core",  
  "arguments": {}}
```

7.2.5.8 Resume

一時停止中のロボット動作を再開します。

Action	resume
Bridge	core

```
{"action": "resume",  
  "bridge": "core",  
  "arguments": {}}
```

7.2.5.9 Release (ハンドガイド有効)

関節を重力補正モードにして、手動で動かせるようブレーキを解除します。ロボットが準備完了状態である必要があり、実行後はハンドガイド状態に移行します。

Action	release
Bridge	core
Argument キー	joint_ids
Argument 値	(任意)ハンドガイドを有効にする関節 ID のリスト。デフォルトでは全関節にハンドガイドを有効にします。
Argument タイプ	リスト

```
{"action": "release",  
  "bridge": "core",  
  "arguments": {"joint_ids": ["1", "2", "5", "6"]}}
```

7.2.5.10 Hold (ハンドガイド無効)

ロボットの関節を位置制御モードにし、手で動かさないようにブレーキを保持します。ロボットがハンドガイド状態である必要があり、実行後は準備完了状態に移行します。

Action	hold
Bridge	core
Argument キー	joint_ids
Argument 値	(任意)ブレーキを保持する関節 ID のリスト。デフォルトでは全関節のブレーキを保持します。
Argument タイプ	リスト

```
{"action": "hold",  
  "bridge": "core",  
  "arguments": {"joint_ids": ["1", "2", "5", "6"]}}
```

7.2.5.11 Stop

実行中の動作関数やスクリプトを停止します。

Action	stop
Bridge	core

```
{"action": "stop",  
  "bridge": "core",  
  "arguments": {}}
```

7.2.5.12 Change Project

現在のデータベースプロジェクトを、指定した ID のプロジェクトに変更します。ロボットが準備完了状態である必要があります。

Action	change_project
Bridge	core
Argument キー	project_id
Argument 値	切り替え先のプロジェクト ID。デフォルトプロジェクトの ID は 1 です。
Argument タイプ	整数

```
{"action": "change_project",  
  "bridge": "core",  
  "arguments": {"project_id": 1}}
```

7.2.5.13 Execute Script

指定した名前または ID でアプリケーションを実行します。ロボットが準備完了状態である必要があり、実行後は実行中状態に移行します。スクリプトを実行中は、ロボットを一時停止できます。スクリプトが完了すると、状態は準備完了に戻ります。

Action	execute_script	
Bridge	core	
Argument キー	script_name	script_id
Argument 値	実行するスクリプトの名前。	実行するスクリプトの ID。
Argument タイプ	文字列	整数

```
{"action": "execute_script",  
  "bridge": "core",  
  "arguments": {"script_id": 3}}  
  
{"action": "execute_script",  
  "bridge": "core",  
  "arguments": {"script_name": "my_application"}}
```

7.2.5.14 Test Script

アプリケーションの Python スクリプトを実行します。ロボットが準備完了状態である必要があり、実行後は実行中状態に移行します。スクリプトを実行中は、ロボットを一時停止できます。スクリプトが完了すると、状態は準備完了に戻ります。

Action	test_script		
Bridge	core		
Argument キー	script_code	script_type	script_id
Argument 値	テストするアプリケーションの Python コード。	スクリプトの種類。 “main” “parallel” “background”	テストするスクリプトの ID。データベースにスクリプトが存在しない場合、ID は 0 です。
Argument タイプ	文字列	文字列	整数

```
{"action": "test_script",  
  "bridge": "core",  
  "arguments": {"script_code": "print('Hello World')",  
                "script_type": "main",  
                "script_id": 0}}
```

7.2.5.15 Script Receive

接続チャネルからアプリケーションスクリプトにメッセージを送信するときに使用します。スクリプトの **receive_message** 関数でメッセージを受信できます。

Action	script_receive
Bridge	core
Argument キー	message
Argument 値	スクリプトから受信するメッセージ。
Argument タイプ	多様なデータ型

```
{"action": "script_receive",  
  "bridge": "core",  
  "arguments": {"message": 123}}
```

7.2.5.16 Move Pose

指定した ID または名前のポーズにロボットを移動します。ロボットが準備完了状態である必要があり、実行後は実行中状態に移行します。ロボットの動作中は、動作を一時停止できます。動作が完了すると、状態は準備完了に戻ります。

Action	move_pose
Bridge	core
Argument キー	pose_id
Argument 値	移動するポーズの ID または名前。
Argument タイプ	整数または文字列

```
{"action": "move_pose",  
  "bridge": "core",  
  "arguments": {"pose_id": 2}}
```

```
{"action": "move_pose",  
  "bridge": "core",  
  "arguments": {"pose_id": "start_pose"}}
```

7.2.5.17 Play Path

指定した ID または名前のパスを再生します。ロボットが準備完了状態である必要があり、実行後は実行中状態に移行します。ロボットの動作中は、動作を一時停止できます。動作が完了すると、状態は準備完了に戻ります。

Action	play_path
Bridge	core
Argument キー	path_id
Argument 値	再生するパスの ID または名前。
Argument タイプ	整数または文字列

```
{"action": "play_path",  
  "bridge": "core",  
  "arguments": {"path_id": 2}}
```

```
{"action": "play_path",  
  "bridge": "core",  
  "arguments": {"path_id": "p1"}}
```

7.2.5.18 Toggle Tool

ツールの機能を切り替えます。通常はピックとプレースの切り替えに使用します。ロボットが準備完了状態である必要があり、実行後は実行中状態に移行します。機能が完了すると、状態は準備完了に戻ります。

Action	toggle_tool
Bridge	core
Argument キー	pick
Argument 値	true: オブジェクトをピック、false: オブジェクトをプレース。「ピック動作」で定義されています。
Argument タイプ	ブール値

```
{"action": "toggle_tool",  
  "bridge": "core",  
  "arguments": {"pick": false}}
```

7.2.5.19 Write Digital & Analog Outputs

メインコントローラーやツールコントローラー（信号線はメインコントローラーからツールへ接続）に対して、デジタル/アナログの出力を 1 つ以上書き込みます。メイン/ツール、デジタル/アナログの組み合わせごとに、同じ引数を持つ異なる関数を使用します。すべての I/O の一覧は、「I/O の設定メニュー」、または「I/O メニュー」で確認できます。

Action	write_digital_main_output write_analog_main_output write_digital_tool_output write_analog_tool_io	
Bridge	core	
Argument キー	identifiers	values
Argument 値	単一の I/O を指定する方法は複数あり、以下のいずれかを使用します。 - 数値 (整数) - カスタム名 (文字列) 複数の I/O を同時に書き込む場合は、数値またはカスタム名をリストで指定できます。	選択した"identifiers"タイプにより、値は以下の形式になります。 - デジタル I/O: true/false (ブール値) - アナログ I/O: 数値 (フロート) 複数の I/O を同時に書き込む場合は、これらの値のリスト。
Argument タイプ	整数/文字列/リスト	ブール値/フロート/リスト

```
{
  "action": "write_digital_tool_io",
  "bridge": "core",
  "arguments": {
    "identifiers": 3,
    "values": true
  }
}
```

この例では、デジタルツール出力 No.3 に true を書き込みます。

```
{
  "action": "write_analog_tool_io",
  "bridge": "core",
  "arguments": {
    "identifiers": "my_analog_output",
    "values": 9.5
  }
}
```

この例では、カスタム名"my_analog_output"のアナログツール出力に 9.5V を書き込みます。

```
{
  "action": "write_digital_main_io",
  "bridge": "core",
  "arguments": {
    "identifiers": [1, 2],
    "values": [true, false]
  }
}
```

この例では、デジタルメイン出力 No.1 に true、No.2 に false を書き込みます。

7.2.5.20 Answer Dialog

開いているダイアログウィンドウに応答します。ダイアログの種類によって応答は異なります。

Action	answer_dialog		
Bridge	core		
Argument キー	dialog_id	dialog_answer	
Argument 値	応答を送信するダイアログを指定するための一意の ID。ID は受信した open_dialog 応答に含まれています。	ダイアログに送信する応答。	
Argument タイプ	文字列	ブール値/整数/文字列	

```
{ "action": "answer_dialog",  
  "bridge": "core",  
  "arguments": { "dialog_id": "5cee85a8-9ab2-4de8-b1ea-52ded8cc68c0",  
                 "dialog_answer": true }}
```

7.2.5.21 Set State Variable

状態変数"variable_name"の値を"variable_value"に設定します。変数は"script_id"のスクリプト内に事前に定義されている必要があります。スクリプトは、データベースに保存されている必要があります。スクリプト実行中にもこのコマンドは実行できます。

Action	set_state_variable		
Bridge	core		
Argument キー	script_id	variable_name	variable_value
Argument 値	変更する状態変数を含むスクリプトの ID。	変更する状態変数の名前。	新しい値。
Argument タイプ	整数	文字列	多様なデータ型

```
{ "action": "set_state_variable",  
  "bridge": "core",  
  "arguments": { "script_id": 2,  
                 "variable_name": "integer_1",  
                 "variable_value": 42 }}
```

7.2.5.22 Set Global Variable

グローバル変数"name"の値を"value"に設定します。グローバル変数は、スクリプトの実行終了後やロボットの再起動後も保持されます。

Action	set_global_variable	
Bridge	core	
Argument キー	name	value
Argument 値	グローバル変数の名前	グローバル変数の値
Argument タイプ	文字列	多様なデータ型

```
{"action": "set_global_variable",  
  "bridge": "core",  
  "arguments": {"name": "my_variable", "value": 3}}
```

7.2.5.23 Get Global Variable

グローバル変数"name"の値を取得します。グローバル変数は、スクリプトの実行終了後やロボットの再起動後も保持されます。

Action	get_global_variable	
Bridge	core	
Argument キー	name	
Argument 値	グローバル変数の名前	
Argument タイプ	文字列	

```
{"action": "get_global_variable",  
  "bridge": "core",  
  "arguments": {"name": "my_variable"}}
```

7.2.5.24 Set Shared Variable

共有変数"name"の値を"value"に設定します。共有変数はスクリプトの実行中のみ使用できます。共有変数は、スクリプトの実行終了後やロボットの停止後に削除されます。

Action	set_shared_variable	
Bridge	core	
Argument キー	name	value
Argument 値	共有変数の名前	共有変数の値
Argument タイプ	文字列	多様なデータ型

```
{"action": "set_shared_variable",  
  "bridge": "core",  
  "arguments": {"name": "my_variable", "value": 3}}
```

7.2.5.25 Get Shared Variable

共有変数"name"の値を取得します。共有変数はスクリプトの実行中のみ使用できます。共有変数は、スクリプトの実行終了後やロボットの停止後に削除されます。

Action	get_shared_variable	
Bridge	core	
Argument キー	name	
Argument 値	共有変数の名前	
Argument タイプ	文字列	

```
{"action": "get_shared_variable",  
  "bridge": "core",  
  "arguments": {"name": "my_variable"}}
```

7.3 REST 接続モジュール

7.3.1 接続の確立

REST 接続モジュールは、接続クライアントからのコマンドを受信するためのリソース

https://<IP アドレス>/rest/gateway を提供します。接続チャンネルは 1 つのみです。複数のユーザーが REST 経由で接続すると、同じチャンネルを共有します。

REST 接続を使用してロボットの制御を取得し、コマンドを送信するには、以下のスクリプトを使用します。コマンドを送信する前に、ログインしてアクティブ制御を取得してください。接続を使用しない場合は、制御を解除し、ログアウトすることを推奨します。

```
import json
import requests

IP_ADDRESS = "10.0.0.203"
TOKEN = "MY_ACCESS_TOKEN"
SESSION_ID = "MY_SESSION_ID"

REST_URL = f"https://{IP_ADDRESS}/rest/gateway"
HEADERS = {"Accept": "application/json", "Content-Type": "application/json"}

def request(method, url, headers, data, timeout=None):
    try:
        response = method(url=url, headers=headers, data=data, timeout=timeout,
                           verify=False)
        response_code = response.status_code

        # this is equal to 'response.status_code < 400' (errors have codes 400+)
        if response.ok:
            # try to parse json payload
            try:
                response_payload = response.json()
            except json.decoder.JSONDecodeError:
                # usually this happens because the response payload is empty
                # this happens for example when a request returns "", 200
                response_payload = dict()
                print("response payload json was empty")
        else:
            print("response.ok returned False")

    except requests.Timeout:
        response_payload = dict()
        response_code = 408 # timeout
    except Exception as e:
        # usually connection problems...
        print(f"rest client could not connect to specified endpoint: {e}")
        response_payload = dict()
        response_code = 404 # not found

    return response_code, response_payload
```

```
def post_request(data, timeout=None):
    return request(method=requests.post, url=REST_URL, headers=HEADERS, data=data,
timeout=timeout)

# login
data = {"bridge": "core", "action": "token_login", "arguments": {"token":
"<TOKEN>", "session_id": "<SESSION_ID>"}}
response = post_request(data, timeout=10.0)
# acquire the active control before start sending commands to the robot
data = {"bridge": "core", "action": "request_active_control", "arguments": {}}
response = post_request(data, timeout=1.0)

# add your commands here

# when you're done using this connection, give away the active control
data = {"bridge": "core", "action": "release_active_control", "arguments": {}}
response = post_request(data, timeout=1.0)
# when you're done working with this connection, you should logout
data = {"bridge": "core", "action": "logout", "arguments": {}}
response = post_request(data, timeout=1.0)
```

7.3.2 コマンドの送信

上記の例で紹介した以外にも、多くのコマンドが使用できます。詳細は、「TCP 接続モジュール」のコマンド一覧を参照してください。REST 接続でも同じコマンドを使用できます。

"bridge"、"action"、"arguments"、任意の"sequence"パラメーターに加えて、任意の"timeout"パラメーターを送信できます。timeout を送信すると、REST リクエストが正常に送信されても、デフォルトの応答(200, {})ではなく、コマンドが待機状態となり、(200, action_return_value)が返されます。Action Return Value には、呼び出された API 関数、関数の実行結果、戻り値が含まれます。

7.3.3 設定

デスクトップインターフェースの設定メニューで、REST 接続モジュールの認証を無効に設定できます。認証を無効にすると、ログインとログアウトが不要になります。

認証設定

RESTの認証を無効にする



図 7.3.3: REST モジュール設定

7.4 ROS ハンドラーモジュール

ROS ハンドラーモジュールを使用すると、ロボットを ROS 2 環境 (Robot Operating System) に統合し、ROS 2 コマンドで AX Portal と通信できます。ここで説明する手順は、[ROS ミドルウェア \(RMW\)](#)として ROS Jazzy および [CycloneDDS](#) に対応しています。

 警告	ROS ハンドラーは、一部の下位レベル機能にアクセスできます。たとえば、ロボットのモーターに直接制御コマンドを送信し、即時に実行できます。GUI では、一部の機能に追加の確認やダイアログによる承認が必要ですが、ここでは使用できない場合があります。そのため、ROS でロボットを直接制御する際は特に注意してください。
---	---

キーポイント:

- ROS 2 に対応した最初の AX Portal リリースであり、今後追加機能が提供される可能性があります。
- 現時点では、ROS ハンドラーは AX Portal シミュレーターでは利用できず、実機のみで利用できます。

7.4.1 インストール

AX Portal の ROS ハンドラーと通信するには、使用する PC に ROS をインストールする必要があります。[ROS 公式ドキュメント](#)で複数のセットアップ方法が提供されています (例: deb パッケージ、Docker イメージ、ソースインストールなど)。この例では、ROS 公式の Docker イメージを使用してロボットと通信します。以下の Docker Compose ファイルで、PC に簡単な ROS 環境を構築します。

```
services:
  ros2_development:
    image: osrf/ros:jazzy-desktop
    container_name: "ros2_development"
    command: /bin/bash -c
      "apt update &&
      apt install -y ros-jazzy-rmw-cyclonedds-cpp &&
      sleep infinity"
    environment:
      - RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
      - CYCLONEDDS_URI=/ros/cyclonedds_config.xml

  network_mode: "host"

volumes:
  - "./cyclonedds_config.xml:/ros/cyclonedds_config.xml"
```

起動時に `ros-jazzy-rmw-cyclonedds-cpp` をインストールし、ROS にロボットと同じ RMW である CycloneDDS を使用させるため、`RMW_IMPLEMENTATION` を `rmw_cyclonedds_cpp` に設定します。さらに、カスタムの CycloneDDS 設定を適用するため、`cyclonedds_config.xml` を共有し、そのパスを `CYCLONEDDS_URI` に設定します。

7.4.2 設定

ロボットと PC 間の ROS 通信には、両方の機器で設定が必要です。PC 側では、新しい環境変数を設定し、ネットワーク設定ファイルを用意する必要があります。ロボット側では、環境変数やネットワーク設定ファイルに直接アクセスできませんが、設定メニューの ROS 機能設定を使って編集できます。



ROS機能設定	
検出モード	Peer to Peer
ROS 2ピアのIPアドレス	127.0.0.1
ROS ドメインID	15

図 7.4.2: ROS ハンドラー設定

以降のセクションで、各設定の内容を説明します。PC 側で必要な調整についても説明します。

7.4.3 Cyclone DDS によるネットワーク検出

以前のセクションで Cyclone DDS について何度か触れました。AX Portal のユーザーとして Cyclone DDS について知っておくべきことは以下のとおりです。

Cyclone DDS を ROS ミドルウェアとして選択すると、ネットワーク上で ROS 関連のデータパッケージを配信する役割を担います。Cyclone DDS は、デバイス間のネットワーク接続を確立し、ROS 2 がその接続を通じてデータを送信できるようにします。

7.4.3.1 ネットワークインターフェース

まず、Cyclone DDS は ROS で使用するネットワークインターフェースを認識する必要があります。デフォルトはイーサネットです。

[ROS 機能設定]の[ネットワークインターフェース]設定で、使用するインターフェースを選択します。

キーポイント: AX Portal が ROS ハンドラーを起動すると、Cyclone DDS はネットワークインターフェースをスキャンし、接続先 PC の IP アドレスを検出します。このインターフェースにアクセスできない場合、ROS ハンドラーは読み込みに失敗します。そのため、ネットワークインターフェースをイーサネットに設定した場合は、*ロボットを起動する前に*ケーブルが接続されていることを確認してください。

7.4.3.2 検出方法

ネットワーク内の他のデバイスを検出するために、Cyclone DDS ではマルチキャストとピアツーピアの 2 つの方法が使用できます。マルチキャストでは、ネットワーク範囲内の到達可能なすべてのクライアントにネットワークパッケージを送信します。利点はシンプルさです。どのコンピューター間で通信できるかを定義する必要はありません。この方法では、ROS_DOMAIN_ID 環境変数 (下記参照) を使用して、2 台のロボットがメッセージを混同しないようにできます。

欠点はネットワークの負荷です。すべてのパッケージがそれぞれのネットワーククライアントで処理されます。異なる ROS_DOMAIN_ID を設定しても、防ぐことはできません。コンピューターに ROS がインストールされていなくても同じです。Cyclone DDS は UDP ベースのため、ネットワーク内のすべてのコンピューターが UDP パッケージを受信し、処理します。小規模なネットワークや少量のデータでは通常問題ありません。ただし、大規模ネットワークでロボットがすべてのクライアントにカメラストリームを送信すると、ネットワーク全体が大幅に遅くなる可能性があります。このような場合は、Cyclone DDS の設定でピアツーピア接続を許可する IP アドレスを明示的に定義する方が安全です。

以降の 2 つの項目で、マルチキャスト通信とピアツーピア通信のネットワーク設定方法を説明します。ヒント: ほとんどの Linux ディストリビューションでは、コマンド "ip addr show" で IP アドレスと対応するネットワークインターフェース名を確認できます。

7.4.3.2.1 ピアツーピア

前述のインストール手順で、Docker Compose ファイルに Cyclone DDS の設定ファイルをマウントしました。ピアツーピア通信を設定するための Cyclone DDS 設定ファイルの例は以下のとおりです。

```
<?xml version='1.0' encoding='utf-8'?>
<CycloneDDS xmlns="https://cdds.io/config"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://cdds.io/config
https://raw.githubusercontent.com/eclipse-
cyclonedds/cyclonedds/master/etc/cyclonedds.xsd">
  <Domain id="any">
    <General>
      <Interfaces>
        <NetworkInterface name="INTERFACE_NAME" multicast="false" />
      </Interfaces>
      <AllowMulticast>false</AllowMulticast>
      <EnableMulticastLoopback>false</EnableMulticastLoopback>
    </General>
    <Discovery>
      <MaxAutoParticipantIndex>119</MaxAutoParticipantIndex>
      <ParticipantIndex>auto</ParticipantIndex>
      <Peers>
        <Peer address="192.168.1.1" />
        <Peer address="192.168.1.2" />
      </Peers>
    </Discovery>
  </Domain>
</CycloneDDS>
```

まず、ROS 通信に使用する PC のネットワークインターフェース名を INTERFACE_NAME に置き換えます。例: enp1s0, wlp2s0

次に、[Peers]リストを設定し、ROS で相互接続するネットワーク内のデバイスの IP アドレスを反映します。

上記のファイルと同様のファイルがロボットにも保存されます。このファイルに直接アクセスできませんが、AX Portal の設定メニューにある[ROS 機能設定]を使って編集できます。[ROS 2 ピアの IP アドレス]設定では、IP アドレスを入力するテキストフィールドのリストが表示されます。

注意:

- 変更を有効にするには、ロボットの再起動が必要です。
- リスト内の IP アドレスが不正な場合、AX Portal はエラーを表示し、新しいリストは保存されません。
- リストにはロボット自身の IP アドレスを含める必要があります。現時点では、自動的に読み込まれません。

- IP アドレスをリストから削除して新しいリストを保存すると、その IP は Cyclone DDS の設定からも削除されます。
- ファイアウォールの設定により、すべての ROS 参加機器を同じネットワークサブレンジ内に配置する必要が生じることがあります。

7.4.3.2.2 マルチキャスト

マルチキャスト通信を設定するための Cyclone DDS 設定ファイルの例は以下のとおりです。

```
<?xml version="1.0" encoding="utf-8"?>
<CycloneDDS
  xmlns="https://cdds.io/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://cdds.io/config
https://raw.githubusercontent.com/eclipse-
cyclonedds/cyclonedds/master/etc/cyclonedds.xsd"
>
  <Domain Id="any">
    <General>
      <Interfaces>
        <NetworkInterface name="INTERFACE_NAME" multicast="true" />
      </Interfaces>
      <AllowMulticast>true</AllowMulticast>
      <EnableMulticastLoopback>true</EnableMulticastLoopback>
    </General>
  </Domain>
</CycloneDDS>
```

INTERFACE_NAME は、ROS 通信に使用する PC 上の実際のネットワークインターフェース名に置き換えてください。例: enp1s0, wlp2s0

ロボットをマルチキャストに切り替えるには、[マルチキャスト]設定ボタンを切り替えます。変更を有効にするには、ロボットの再起動が必要です。

7.4.4 接続の確立

PC で ROS 環境を起動するには、docker-compose ファイルのあるディレクトリに移動し、次のコマンドを実行します。

```
docker compose up -d
```

Docker コンテナが起動していることを確認します。

```
docker ps -a
```

以下のような出力が表示されます。

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
132bdd6fa4de	osrf/ros:jazzy-desktop	"/ros_entrypoint.sh ..."	2 seconds ago	Up 1 second		ros2_development

以下のコマンドでコンテナーに対話モードで入ります。

```
docker exec -it ros2_development bash
```

コンテナーに入るたびに ROS セットアップファイルを source してください。

```
source /opt/ros/jazzy/setup.bash
```

同じネットワークで複数のロボットを使用するため、各ロボットに異なる ROS_DOMAIN_ID を設定します。AX Portal の[ROS 機能設定]に移動し、ROS ドメイン ID を設定します。設定できる範囲は 6~41 です。設定後は、ロボットの再起動が必要です。ROS の Docker コンテナーを開いているターミナルに戻り、同じ値を ROS_DOMAIN_ID に設定します。

```
export ROS_DOMAIN_ID=SELECTED_NUMBER
```

AX Portal で"robot"という名前の ROS ノードが表示されます。

```
ros2 node list
```

7.4.5 ROS でロボットデータにアクセスする

ノードが表示されたら、次のコマンドで使用可能なトピック一覧を確認できます。

```
ros2 topic list
```

ROS 2 のトピックを使用して、ロボットからリアルタイムデータを取得できます。例: 関節位置や速度などを subscribe するには、次のコマンドを実行します。

```
ros2 topic echo /joint_states
```

7.4.6 スクリプトでの ROS 2 機能

スクリプトに rclpy を直接インポートできます。これにより、publisher や subscriber などの ROS 2 オブジェクトを作成できます。

ROS ハンドラーは独自の ROS ノードを作成します。ros_handler.ros_node()関数を使用してアクセスします。注意: このノードで rclpy.spin を実行しないでください。ROS ハンドラー内ですでに実行されています。

7.4.7 アクションを呼び出す

AX Portal の ROS ノードは、bridge 機能を実行するために"/robot/execute_function"というアクションサーバーを作成します。このアクションは generic_messages (https://github.com/Epson-Robots/generic_msgs)パッケージのカスタム定義された ExecuteFunction アクション定義が必要です。定義を取得するには、このパッケージをインストールします。パッケージをワークスペースにクローンし、colcon でビルドします。

```
mkdir -p /colcon_ws/src
cd /colcon_ws/src
```

```
git clone https://github.com/Epson-Robots/generic_msgs
cd ..
colcon build --packages-up-to generic_messages
source install/setup.bash
```

ROS 2 アクションをロボットに送信するには、クライアントがアクティブ制御を取得する必要があります。そのために、ターミナルで次のコマンドを実行し、AX Portal の GUI でダイアログを確認します。

```
ros2 action send_goal --feedback /execute_function
generic_msgs/action/ExecuteFunction "{action: request_active_control, bridge: core,
arguments: '{}'}"
```

次のコマンドで、ターミナルから bridge 機能を実行できます。

```
ros2 action send_goal --feedback /execute_function
generic_msgs/action/ExecuteFunction "{action: read_actuator_position, bridge: core,
arguments: '{\"actuator_ids\": 6}'}"
```

```
ros2 action send_goal /execute_function generic_msgs/action/ExecuteFunction
"{action: move_joints, bridge: core, arguments: '{\"joints\": 6,
\"joint_position\": 100.0}'}"
```

7.4.8. 認証設定

ROS ハンドラーは現在、認証なしでのみ動作します。デスクトップインターフェースの[設定]メニューで、この接続モジュールの認証を手動で無効にする必要があります。

認証設定

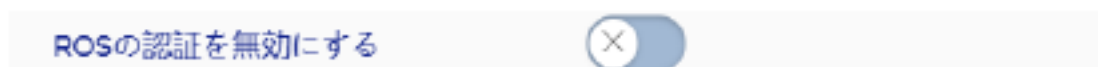


図 7.4.8: ROS 認証設定

7.5 Modbus 接続モジュール

7.5.1 概要

Modbus モジュールでは [TCP Modbus](#) プロトコルを使用します。サーバー側のモジュールは Python ベースで、[PyModbus ライブラリー](#) を使用して実装されています。詳細は、[PyModbus のドキュメント](#) を参照してください。モジュールは、読み取り用レジスター (アナログ入力レジスター: 30000~39999) と書き込み用レジスター (アナログ出力保持レジスター: 40000~49999) に分かれています。サーバーとの通信には、任意の TCP Modbus クライアントを使用できます。使用するポート番号は **5020** です。本章の末尾には、Python コードの例を掲載しています。その中には、Python スクリプトでサーバーへの接続を確立する方法も含まれています

7.5.2 設定

Modbus 接続は現在、認証なしでのみ動作します。デスクトップインターフェースの[設定]メニューで、この接続モジュールの認証を手動で無効にする必要があります。

認証設定

Modbusの認証を無効にする



図 7.5.2: Modbus モジュール設定

7.5.3 レジスター書き込み (概要)

Modbus アドオンは、コマンドレジスターとプロパティレジスターの情報変更に応じて動作します。書き込みレジスターのデータクリアは、ユーザーの責任にて実施してください。

7.5.3.1 コマンドレジスター

コマンドを実行するには、コマンドレジスター (30000) に 1 ビットを書き込みます。一部のコマンドでは、コマンドデータレジスター (30001) に引数が必要です。両方のレジスターを同時に書き込めない場合は、先にコマンドデータレジスターを書き込みます。

絶対アドレス: 30000 - 30001		
相対アドレス 0x000 - 0x001		
ビット	コマンドレジスター (30000)	データ (30001)
0	電源オン	ツール ID
1	電源オフ	
2	停止	
3	一時停止	
4	再開	
5	キャリブレーションする	スクリプト ID
6	ハンドガイド(HGC)を有効にする	
7	ハンドガイド(HGC)を無効にする	
8	プロジェクトを変更する	プロジェクト ID
9	スクリプトを開始する	スクリプト ID
10	ポーズに移動する	ポーズ ID
11	パスを再生する	パス ID
12	ツールのプレース動作	
13	ツールのピック動作	
14 ... 15	予約	

7.5.3.2 プロパティレジスター

コマンドを実行するには、プロパティレジスター (30002)に 1 ビットを書き込みます。一部のコマンドでは、プロパティデータレジスター (30003)に引数が必要です。両方のレジスターを同時に書き込めない場合は、先にプロパティデータレジスターを書き込みます。

絶対アドレス: 30002 - 30003		
相対アドレス 0x002 - 0x003		
ビット	プロパティレジスター (30002)	データ (30003)
0	デジタル出力を書き込み	IO 状態
1	スクリプトイベントを設定	イベント ID
2	エラーを確認	
3	位置を読み取り	
4	アクティブ制御を要求	
5	アクティブ制御を解除	
6	手動モードに切り替え	
7	自動モードに切り替え	
8	安全リセット	
9	動作の再開	
10 ... 15	予約	

7.5.3.3 ユーザー書き込みレジスター

絶対アドレス: 30016		...	絶対アドレス: 30255	
相対アドレス 0x010			相対アドレス 0x0FF	
0 ...15	カスタムデータ		0 ...15	カスタムデータ

ユーザー書き込みレジスターを使用して、スクリプトアプリケーションに情報を送信します。スクリプトアプリケーションでは、**modbus.write_user_register** 関数を使用してカスタムデータを読み取ることができます。

7.5.4 レジスターへの書き込み (詳細)

7.5.4.1 何もしない

書き込みレジスターのデータクリアは、ユーザーの責任にて実施してください。コマンドレジスター (30000)やプロパティーレジスター (30002)をクリアするには、整数値 0 を書き込みます。

0x000	0x001	0x002	0x003
0		0	

7.5.4.2 電源オン

ロボットを操作する前に、モーターの電源をオンにします。接続するカスタムツールの ID は、数値 n で指定します。ロボットの状態は、電源オフ (1, ビット 0)から準備完了 (8, ビット 3)に変わります。

0x000	0x001	0x002	0x003
1	n		

7.5.4.3 電源オフ

作業終了後、ロボットのモーター電源をオフにします。ロボットがどの状態であっても電源をオフにできます。電源がオフになると、ロボットの状態は、電源オフ (1, ビット 0)に変わります。

0x000	0x001	0x002	0x003
2			

7.5.4.4 停止

ロボットのすべての動作を停止します。状態は実行中 (64, ビット 6)から準備完了 (8, ビット 3)に変わります。

0x000	0x001	0x002	0x003
4			

7.5.4.5 一時停止

実行中 (64, ビット 6)にロボットの動作を一時停止します。状態は一時停止中 (128, ビット 7)に変わります。

0x000	0x001	0x002	0x003
8			

7.5.4.6 再開

一時停止中 (128, ビット 7)のロボットの動作を再開します。状態は実行中 (64, ビット 6)に変わります。

0x000	0x001	0x002	0x003
16			

7.5.4.7 キャリブレーション

現時点では、キャリブレーションは動作に影響しません。このビットは予約領域です。

0x000	0x001	0x002	0x003
32			

7.5.4.8 ハンドガイド(HGC)を有効にする

すべてのロボット関節でハンドガイド制御を有効にし、重力補正モードにして手で動かせるようにします。準備完了 (8, ビット 3)状態で実行でき、ハンドガイド (2048, ビット 11)状態に変わります。

0x000	0x001	0x002	0x003
64			

7.5.4.9 ハンドガイド(HGC)を無効にする

すべてのロボット関節を保持し、位置制御モードにして手で動かさない状態にします。ハンドガイド (2048, ビット 11)状態で実行でき、準備完了 (8, ビット 3)状態に変わります。

0x000	0x001	0x002	0x003
128			

7.5.4.10 プロジェクトを変更する

現在のデータベースプロジェクトを ID n のプロジェクトに変更します。準備完了 (8, ビット 3)状態で実行できます。

0x000	0x001	0x002	0x003
256	n		

7.5.4.11 スクリプトを開始する

ID n のスクリプトを実行します。準備完了 (8, ビット 3)状態で行行でき、実行中 (64, ビット 6)状態に変わります。スクリプトを実行中は、ロボットを一時停止できます。スクリプトが完了すると、状態は準備完了 (8, ビット 3)に戻ります。

0x000	0x001	0x002	0x003
512	n		

7.5.4.12 ポーズを移動する

ID n のポーズにロボットを移動します。準備完了 (8, ビット 3)状態で行行でき、実行中 (64, ビット 6)状態に変わります。ロボットの動作中は、動作を一時停止できます。動作が完了すると、状態は準備完了 (8, ビット 3)に戻ります。

0x000	0x001	0x002	0x003
1024	n		

7.5.4.13 パスを再生する

ID n のパスを再生します。準備完了 (8, ビット 3)状態で行行でき、実行中 (64, ビット 6)状態に変わります。ロボットの動作中は、動作を一時停止できます。動作が完了すると、状態は準備完了 (8, ビット 3)に戻ります。

0x000	0x001	0x002	0x003
2048	n		

7.5.4.14 ツールのプレース動作

ツールを使用して、オブジェクトをプレースします (例: グリッパーを開く)。準備完了 (8, ビット 3)状態で行行でき、実行中 (64, ビット 6)状態に変わります。グリッパーの動作中は、動作を一時停止できます。動作が完了すると、状態は準備完了 (8, ビット 3)に戻ります。

0x000	0x001	0x002	0x003
4096			

7.5.4.15 ツールのピック動作

ツールを使用して、オブジェクトをピックします (例: グリッパーを閉じる)。準備完了 (8, ビット 3)状態で行行でき、実行中 (64, ビット 6)状態に変わります。グリッパーの動作中は、動作を一時停止できます。動作が完了すると、状態は準備完了 (8, ビット 3)に戻ります。

0x000	0x001	0x002	0x003
8192			

7.5.4.16 デジタル出力の書き込み

デジタル出力の書き込み用の予約領域です。

0x000	0x001	0x002	0x003
		1	d

7.5.4.17 スクリプトイベントの設定

Modbus 端末から送信されるイベントを待機するユーザーアプリケーションでは、スクリプトイベントを使用できます。イベントは全部で 16 種類あります。データ d の値に応じて、イベントが設定またはクリアされます。**modbus.wait_for_event** スクリプト関数は、特定のイベントが設定されるまで待機します。d が 0 の場合、すべてのイベントがクリアされます。d が 65535 の場合、すべてのイベントが設定されます。d が 9 の場合、第 1 イベントと第 4 イベントが設定され、ほかのイベントはクリアされます。

0x000	0x001	0x002	0x003
		2	d

7.5.4.18 エラーの確認

エラーはクリアする前に、内容を確認する必要があります。ただし、エラーを読み取る場合のみ必要です。確認してもロボットの状態に影響はありません。重大エラーコードと軽微エラーコードの両方 (40004-40005) が 0 の場合、追加の確認は不要です。

0x000	0x001	0x002	0x003
		4	

7.5.4.19 位置情報の読み取り

ロボットの位置データを更新します。位置データが要求されると、ロボットの現在の関節角度や座標が対応する読み取りレジスター (40256-40267) に書き込まれます。

0x000	0x001	0x002	0x003
		8	

7.5.4.20 アクティブ制御の要求

ロボットにコマンドを送信するには、アクティブ制御を要求します。ユーザーデータレジスターと読み取りレジスターは、アクティブ制御なしでアクセスできます。現在アクティブ制御を保持しているクライアントには通知され、明示的に制御を解除する必要があります。制御が許可されると、対応する読み取りレジスター (40008) が設定されます。

0x000	0x001	0x002	0x003
		16	

7.5.4.21 アクティブ制御の解除

別のクライアントがロボットにコマンドを送信できるように、アクティブ制御を解除します。ユーザーデータレジスターと読み取りレジスターは、アクティブ制御なしでアクセスできます。対応する読み取りレジスター (40008)が 0 に設定されます。

0x000	0x001	0x002	0x003
		32	

7.5.4.22 手動モードに切り替え

手動モードに切り替えます。このモードでは、ユーザーがイネーブルボタンを押している間のみ、ロボットが動作します。モードを切り替えるには、安全リセットが必要です。

0x000	0x001	0x002	0x003
		64	

7.5.4.23 自動モードに切り替え

自動モードに切り替えます。モードを切り替えるには、安全リセットが必要です。

0x000	0x001	0x002	0x003
		128	

7.5.4.24 安全リセット

インターロックリセットを開始または再起動します。一部の安全機能では、動作を継続するために、ユーザーによる明確で意図的な操作が必要です。この操作が必要かどうかは、対応する読み取りレジスター (40009)で確認できます。

0x000	0x001	0x002	0x003
		256	

7.5.4.25 動作開始割り込み

安全割り込み後、ロボットは動作を再開、継続します。このプロパティを使用すると、ロボットを再び動作させることができます。それ以外の場合は、Stop コマンドを送信して、実行中のアプリケーションを停止できます。この操作が必要かどうかは、対応する読み取りレジスター (40009)で確認できます。

0x000	0x001	0x002	0x003
		512	

7.5.5 レジスターの読み取り

7.5.5.1 ステータスレジスター

ステータスレジスターを使用して、ロボットの現在の状態を取得します。状態はビット番号で表現されます。

7.5.5.1.1 第 1 ステータスレジスター

絶対アドレス: 40000	
相対アドレス 0x000	
0	電源オフ
1	電源オン中
2	読み込み中
3	準備完了
4	処理中
5	キャリブレーション中
6	実行中
7	自律モード
8	一時停止
9	続行
10	保護停止
11	安全停止
12	位置制限
13	ハンドガイド
14	衝突
15	非常停止

7.5.5.1.2 第 2 ステータスレジスター

絶対アドレス: 40001	
相対アドレス 0x001	
0	通常停止
1	Error
2	電源オフ中
3	停止中
4..15	予約

7.5.5.1.3 第 3 ステータスレジスター

絶対アドレス: 40002	
相対アドレス 0x002	
0 ... 15	予約

7.5.5.1.4 第 4 ステータスレジスター

絶対アドレス: 40003	
相対アドレス 0x003	
0 ... 15	予約

7.5.5.1.5 重大エラーレジスター

絶対アドレス: 40004	
相対アドレス 0x004	
0 ... 15	重大エラー情報

重大エラーレジスターは、現在のエラー識別子の先頭 2 バイトを読み取ります。次のエラーを読み取るには、プロパティレジスターを使用してエラーを確認します。
エラーレジスターの値について詳細は、"エラーレジスターマッピング"を参照してください。

7.5.5.1.6 軽微なエラーレジスター

絶対アドレス: 40005	
相対アドレス 0x005	
0 ... 15	軽微なエラー情報

軽微なエラーレジスターは、現在のエラー識別子の末尾 2 バイトを読み取ります。次のエラーを読み取るには、プロパティレジスターを使用してエラーを確認します。

エラーレジスターの値について詳細は、"エラーレジスターマッピング"を参照してください。

7.5.5.1.7 アプリケーションレジスター

絶対アドレス: 40006	
相対アドレス 0x006	
0 ...15	現在のスクリプト ID

アプリケーションレジスターは、現在実行中のスクリプトの ID を読み取ります。

7.5.5.1.8 キャリブレーションレジスター

絶対アドレス: 40007	
相対アドレス 0x007	
0	ロボットがキャリブレーション済みかどうか

キャリブレーションレジスターは、ロボットがキャリブレーション済みかどうかを読み取ります。キャリブレーションされていない場合は 0 です。キャリブレーションされている場合は 1 です。キャリブレーションについて正しい結果を得るには、ロボットをモーターに接続する必要があります。

7.5.5.1.9 アクティブ制御レジスター

絶対アドレス: 40008	
相対アドレス 0x008	
0	Modbus クライアントがアクティブ制御を保持しているかどうか

アクティブ制御レジスターは、Modbus クライアントがアクティブ制御を保持しているかどうかを読み取ります。Modbus がアクティブ制御を保持していない場合は 0 です。Modbus がアクティブ制御を保持している場合は 1 です。Modbus クライアントは、0x30002 0x04 を使用して、アクティブ制御を要求できます。

7.5.5.1.10 割り込みレジスター

割り込みレジスターは、有効な割り込みダイアログのオプションを示します。

絶対アドレス: 40009	
相対アドレス 0x009	
0	安全リセットが必要
1	動作の再開が必要
2	手動モードに切り替えが必要
3..15	予約

7.5.5.1.11 動作モードレジスター

動作モードレジスターは、ロボットが手動モード (1)か、自動モード (0)かを示します。

絶対アドレス: 40010	
相対アドレス 0x00A	
0	ロボットが手動モードかどうか

7.5.5.2 ユーザー読み取りレジスター

絶対アドレス: 40016 相対アドレス 0x010	...	絶対アドレス: 40255 相対アドレス 0x0FF
0 ...15 カスタムデータ		0 ...15 カスタムデータ

ユーザー読み取りレジスターを使用して、スクリプトアプリケーションから情報を受け取ります。スクリプトアプリケーションの **modbus.read_user_register** 関数がデータを読み取ります。

7.5.5.3 位置レジスター (第 1 関節～第 6 関節)

絶対アドレス: 40256 相対アドレス 0x100	...	絶対アドレス: 40261 相対アドレス 0x105
0 ...15 第 1 関節角度		0 ...15 第 6 関節角度

特定の関節角度値を 0.1 度単位で読み取ります。101 を読み取った場合、関節角度は 10.1°です。値は、プロパティーレジスターに読み取り位置フラグが書き込まれた場合のみ更新されます。32768 ($=2^{15}$) より大きな値を読み取った場合、マイナスの値に対応します。絶対値を求めるには 32768 を減算します。たとえば、32869 は、関節角度-10.1°に対応します。

7.5.5.4 座標レジスター (x, y, z, ロール, ピッチ, ヨー)

絶対アドレス: 40262 相対アドレス 0x106	...	絶対アドレス: 40267 相対アドレス 0x10B
0 ...15 TCP の x 座標		0 ...15 TCP のヨー座標

ツールセンターポイントの座標または向きは、x, y, z, ロール, ピッチ, ヨーの順で読み取ります。空間座標は、0.1 mm 単位で読み取ります。座標値が 314 の場合、値は 31.4mm です。回転座標は、0.1 度単位で読み取ります。座標値が 271 の場合、値は 27.1 (mm または°) です。値は、プロパティーレジスターに読み取り位置フラグが書き込まれた場合のみ更新されます。32768 ($=2^{15}$) より大きな値を読み取った場合、マイナスの値に対応します。絶対値を求めるには 32768 を減算します。たとえば、33039 は、関節角度-27.1 (mm または°) に対応します。

7.5.6 エラーレジスタマッピング

エラー情報は常に 2 つのレジスターに表示されます。マイナーエラーレジスターには、実際のエラー番号が格納されます。メジャーエラーレジスターには、次のいずれかの値が格納され、各値は異なるエラーカテゴリに対応しています。

101	ADDONS
102	CONFIG
103	DATABASE
104	CONTROLLER
105	EDITOR
106	BRIDGE
107	CONNECTION
108	DIALOG
109	DISRIBUTOR
110	EVENT
111	KINEMATICS
112	SOCKET
113	CORE
114	SAFETY
115	SETTINGS
116	SIGNAL
117	STATUS
118	SCRIPT
119	THREAD

201	BLOCK
202	COGNEX
203	CONTROL
204	ROS
205	MODBUS
206	REST
207	TCP

301	AX
302	CFG
303	CMK
304	CMD
305	CTL
306	ECAT
307	LIC
308	MATH
309	MCM
310	MISC
311	OS
312	PPM
313	PYAPI
314	SIG
315	SYS
316	TLC
317	UTIL

7.5.6.1 例

エラーレジスターの値が[113, 4]の場合、ロボットによって発生したエラーコード **CORE_004** に対応します。

7.5.7 Python コード例

これらの例は、ロボットにリモート接続する外部システムで使します。

7.5.7.1 例 1

```
from pymodbus.client.sync import ModbusTcpClient

def main():
    # connect to client
    client = ModbusTcpClient("192.168.80.203", port=5020)
    client.connect()

    # read all status
    address = 0x000
    count = 4
    status = client.read_input_registers(address, count)
    print("STATUS", status.registers[0], status.registers[1],
          status.registers[2], status.registers[3])

if __name__ == "__main__":
    main()
```

サーバーに接続します。ロボットの状態を読み取ります。

7.5.7.2 例 2

```
from pymodbus.client.sync import ModbusTcpClient

def main():
    # connect to client
    client = ModbusTcpClient("192.168.80.203", port=5020)
    client.connect()

    # command to connect to motors
    address = 0x000
    values = [1 << 0]
    client.write_registers(address, values, unit=1)

if __name__ == "__main__":
    main()
```

サーバーに接続します。ロボットをモーターに接続します。

ヒント: << コマンドは、ビットシフトに使します。

ビット	コマンド	整数
0	1 << 0	1
5	1 << 5	32

7.5.7.3 例 3

```
from pymodbus.client.sync import ModbusTcpClient

def main():
    # connect to client
    client = ModbusTcpClient("192.168.80.203", port=5020)
    client.connect()

    # command to run script with id 5
    address = 0x000
    values = [1 << 9, 5]
    client.write_registers(address, values, unit=1)

if __name__ == "__main__":
    main()
```

サーバーに接続します。ID5 のアプリケーションを実行します。

7.5.7.4 例 4

```
from pymodbus.client.sync import ModbusTcpClient

def main():
    # connect to client
    client = ModbusTcpClient("192.168.80.203", port=5020)
    client.connect()

    # read all error
    address = 0x004
    count = 2
    error = client.read_input_registers(address, count)
    print("ERROR", error.registers[0], error.registers[1])

    # acknowledge error
    address = 0x002
    values = [1 << 2]
    client.write_registers(address, values, unit=1)

if __name__ == "__main__":
    main()
```

サーバーに接続します。現在のエラーメッセージを読み取ります。エラーメッセージを確認します。

8.Cognex カメラモジュール

8.1 概要

Cognex カメラモジュールを使用すると、設定済みの Cognex カメラを AX Portal に接続できます。接続後、カメラに設定されたジョブを起動して、画像の撮影や処理ができます。処理されたデータは AX Portal に送信され、アプリケーションで読み取れます。

8.2 カメラの設定

AX Portal で Cognex カメラを使用するには、以下の手順で設定します。

1. Cognex 社の [In-Sight Explorer](#) ソフトウェアをダウンロードしてインストールします。
2. カメラの電源を入れ、ネットワークに接続します。ネットワーク設定が静的 IP アドレスの場合は、PC に直接接続できます。
3. In-Sight Explorer を起動し、アクセス許可を求められた場合は許可してください。必要なコンポーネント (Microsoft Visual C++, Microsoft .NET Framework など) をインストールします。
4. その後、Explorer の Easybuilder ビューが自動的に表示されます。
5. [System] → [Add Sensor/Device to Network] を選択します。カメラを探してネットワークに追加します。
6. カメラのネットワーク設定が AX シリーズロボットのネットワーク設定と一致していることを確認します。Telnet ポートが 3000 に設定されていることを確認します。
7. カメラを選択し、[Connect] ボタンを押して In-Sight Explorer と接続します。
8. これで、カメラのキャリブレーションやロボットがピックするオブジェクト認識のジョブを作成し、カメラに保存する準備が整いました。詳細は、Cognex のドキュメントを参照してください。

8.3 カメラアプリケーション

AX Portal アプリケーションで Cognex カメラを使用するには、以下の条件を満たす必要があります。

- ロボットとカメラが同一ネットワークにあること
- カメラの Telnet ポートが 3000 に設定されていること
- カメラがオンライン状態であること

以下は、設定済みのジョブを使用して円の座標を検出する例です。

```
# modify these settings to fit to your application
camera_ip = "10.0.0.210"
job_name = "CircleDetection"
event_id = 1
keyword = "coordinates"

# connect the camera to the application
cognex.connect_to_camera(camera_ip)
```

```
# select the job to trigger
cognex.change_job(camera_ip, job_name)
# trigger the event and wait for a response
response = cognex.get_camera_message(camera_ip, event_id, keyword)
print(response)
# close the connection
cognex.disconnect_camera(camera_ip)
```