

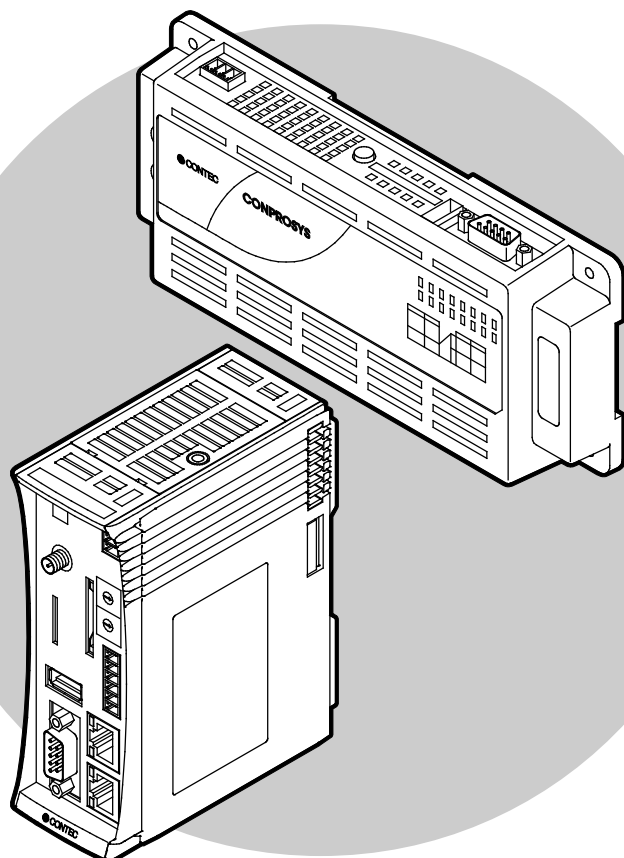
システムセットアップガイド

PAC シリーズ

コンパクトタイプ/スタックタイプ

目次

はじめに	6
安全にご使用いただくために	14
パソコンのネットワーク設定をする	18
CONPROSYS WEB Settingについて	24
CODESYSのインストール	44
プログラミングの基本	52
通信方法の設定をする	68
モニタリング編集をする	151
困った時は	166
仕様	169
付録	174
各種サービス・お問い合わせ	177
索引	180



目次

はじめに 6

- 1. 製品概要 7
 - 1. PACシリーズの機能 7
- 2. このマニュアルの構成 9
- 3. 製品使用までの流れ 10
- 4. 関連マニュアルのご案内 11
- 5. オンラインヘルプのご案内 12
- 6. ファームウェアのご確認 13

安全にご使用いただくために 14

- 1. 注意記号の説明 15
- 2. 取り扱い上の注意 16
- 3. セキュリティに関する注意 17
 - 1. セキュリティリスク 17
 - 2. セキュリティ対策事例 17

パソコンのネットワーク設定をする 18

- 1. パソコンとの接続 19
- 2. パソコンのネットワーク設定をする 20
 - 1. パソコンのネットワーク設定手順 (Windows 10の場合) 20
- 3. 通信しているか確認する 23

CONPROSYS WEB Setting について 24

- 1. CONPROSYS WEB Settingの概要 25
 - 1. 対応Webブラウザ 25
 - 2. CONPROSYS WEB Settingの起動 25
 - 3. CONPROSYS WEB Settingの基本操作 26
- 2. メニュー項目の機能一覧 27
- 3. 機能の詳細説明 28
 - 1. ネットワーク設定 28
 - 2. ユーザー/パスワード設定 30
 - 3. 時刻設定 31
 - 4. データ送信設定 32
 - 5. シリアル設定 33
 - 6. 設定バックアップ 34
 - 7. 設定リストア 35
 - 8. 設定保存 36
 - 9. 設定初期化 37
 - 10. システム情報 37
 - 11. 再起動/シャットダウン 40
 - 12. PLCプログラム保存 41
 - 13. ファームウェア更新 41

目次

14. Editor.....	42
15. Viewer	43
16. ページ保存.....	43

CODESYS のインストール 44

1. CODESYSをインストールする	45
2. CODESYS用パッケージをインストールする	47
3. パソコンとCONPROSYSを接続する	48
4. ファームウェアのバージョンアップ	50

プログラミングの基本 52

1. CODESYS各部の名称.....	53
1. デバイスウィンドウ、デバイスコンフィグレーションウィンドウ.....	53
2. STエディタウィンドウ.....	53
2. ST言語の書式と記述方法	54
3. 基本的なプログラミング手順	56
1. 新規プロジェクトの生成	56
2. CODESYSからコントローラに接続する	57
3. I/Oモジュールを追加する.....	58
4. I/O変数の定義	59
5. プログラム作成とビルド	60
6. プログラムのダウンロードと実行	61
7. PLCプログラムのROM保存	62
8. ROM保存したPLCプログラムの削除.....	62
4. 製品別インターフェイス定義	63
1. パラメーター	63
2. I/Oマッピング	64
3. シリアルポート	67
4. OPC UAシンボル設定	67

通信方法の設定をする 68

1. コントローラとPCでシリアル通信する	70
1. シリアル通信の準備	70
2. シリアル通信ライブラリの設定	70
3. 送信プログラムの作成と実行.....	71
4. 受信プログラムの作成と実行.....	73
2. EtherCAT Masterとして使用する	76
1. デバイスの準備	76
2. スレーブデバイスの追加	76
3. スレーブデバイスの設定	78
4. プログラムの作成とビルド.....	79
5. プログラムのダウンロードと実行	80
3. Modbus TCP Masterとして使用する	81
1. デバイスの準備	81
2. スレーブデバイスの追加	81

目次

3. Modbus TCP Masterの設定	84
4. プログラムの作成とビルド	86
5. プログラムのダウンロードと実行	87
4. Modbus TCP Slaveとして使用する	88
1. デバイスの準備	88
2. スレーブ設定をする	89
3. プログラムの作成とビルド	92
4. プログラムのダウンロードと実行	93
5. Modbus RTU Masterとして使用する	96
1. デバイスの準備	96
2. スレーブデバイスの追加	96
3. Modbus RTU Masterの設定	99
4. プログラムの作成とビルド	101
5. プログラムのダウンロードと実行	102
6. Modbus RTU Slaveとして使用する	103
1. デバイスの準備	103
2. スレーブ設定をする	104
3. プログラムの作成とビルド	108
4. プログラムのダウンロードと実行	109
7. OPC UA Serverを設定する	112
1. デバイスの準備	112
2. CODESYSの設定	112
3. OPC UA Clientの設定	114
8. カウンタ入力	116
1. 基本的な動作手順	116
2. イベント動作手順	117
3. カウント値設定動作手順	118
4. サンプルプログラム	119
9. ファイルにアクセスする	124
1. ファイルアクセスライブラリの設定	124
2. 関数一覧	124
3. データタイプ	125
4. 関数詳細	126
5. サンプルプログラム	132
10. クラウドサービスヘータを送信する	140
1. データ送信ライブラリの設定	140
2. 関数一覧	140
3. データタイプ	141
4. 関数詳細	143
5. サンプルプログラム	147

モニタリング編集をする 151

1. CONPROSYS HMIを使用する	152
1. CODESYSの設定	152
2. プログラムの作成とビルド	152
3. HMI Editorで画面を作成する	154
4. HMI Viewerの実行	157
5. IECデータタイプで利用できる部品	158

目次

2. CONPROSYS HMIの概要	159
1. CONPROSYS HMIでできること	159
2. 操作エリア	160
3. モニタリング画面の作成	160
4. モニタリング画面作成の基本操作	161
3. 表示部品一覧	165

困った時は..... 166

1. 故障かな？と思った時は	167
1. 全般	167

仕様 169

1. ハードウェア仕様	170
1. CPS-PC341EC-1-9201の仕様	170
2. CPS-PC341MB-ADSC1-9201の仕様	170
3. CPS-PCS341EC-DS1-1201の仕様	171
4. CPS-PCS341MB-DS1-1201の仕様	171
2. CONPROSYS HMI仕様	172

付録 174

1. 送信データフォーマット	175
----------------------	-----

各種サービス・お問い合わせ 177

1. 各種サービス	178
2. お問い合わせ	179

索引 180

はじめに

このリファレンスマニュアルは本製品のソフトウェア部分の設定方法を記載しているマニュアルです。

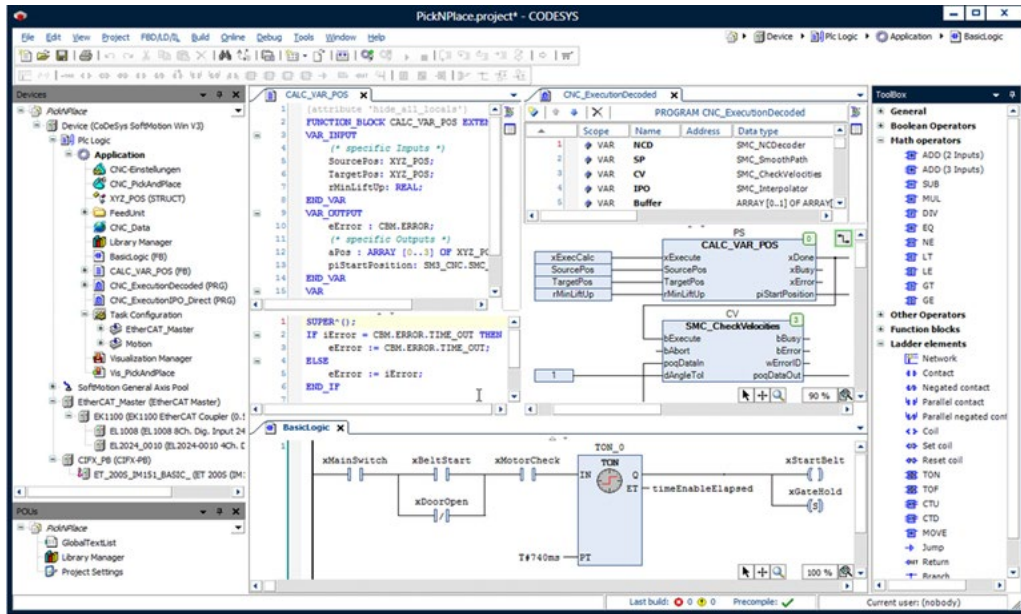
この章ではマニュアルの全体構成、本製品に関連する各種マニュアルなど、本製品をお使いの前に知っていただくべき情報を掲載しています。

1. 製品概要

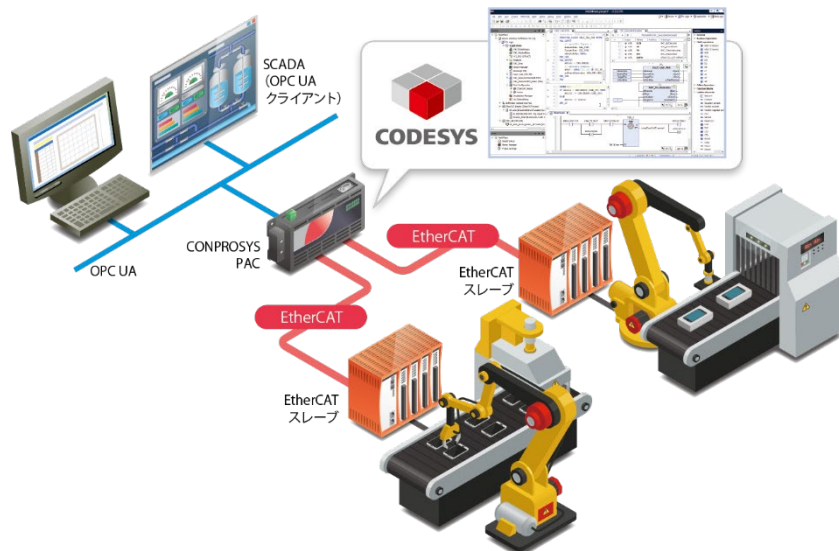
PACシリーズは、ソフトウェアPLC『CODESYS』を搭載したモデルです。

1. PACシリーズの機能

◆ 国際標準IEC61131-3準拠『CODESYS』プログラミング対応



- PLCプログラミング、フィールドバスの設定などの全てを統合した開発環境『CODESYS』が無償提供されます。
- 国際標準IEC61131-3準拠『CODESYS』によるオープンな通信の導入で、特定の通信規格に依存しないI/Oやドライブなどの周辺機器選択が可能となります。
- STをはじめ、LD、FBD、SFC、IL、CFCの計6言語に対応。IEC61131-3第三版定義されるオブジェクト指向プログラミングもサポートされます。



◆ フィールドバスのI/Oをダイレクトに変数使用

EtherCAT/Modbusフィールドバス対応

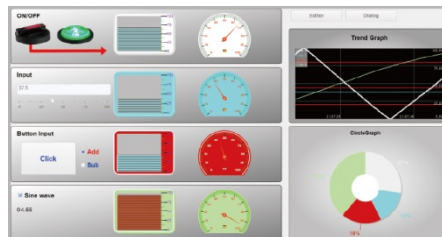
- オープンフィールドネットワークEtherCAT/Modbusのマスタ機能を内蔵。
内蔵I/OやスタックI/Oと同様、CODESYS統合開発環境上でフィールドバスのI/Oをダイレクトに変数へ割り当てることができます。



◆ Webブラウザで簡単に情報をモニタリング

Webモニタリング機能

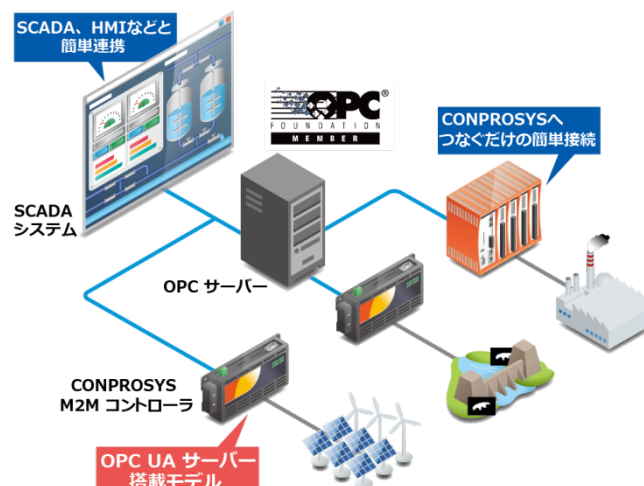
- コントローラ本体にCONPROSYS HMI内蔵。モニタ画面を自由に制作できます。
- 別途サーバーの用意がなくてもWebサーバー機能で設備の情報を手軽に表示できます。



◆ SCADA/MES・ERPシステムと安定したデータ交換

OPC UAサーバー機能(SCADA/MES/ERP連携対応)

コントローラ本体にOPC UAサーバー機能を内蔵。SCADAソフトウェアやMES・ERPシステムとの安全かつ安定したデータ交換が可能です。



2. このマニュアルの構成

このマニュアルは以下のように構成されています。

マニュアルの章	記載内容
はじめに	本製品に関するマニュアルをご紹介します。 必要に応じて関連マニュアルを参照ください。
安全にご使用いただくために	安全にご使用いただくための注意文を記載しています。
パソコンのネットワーク設定をする	パソコンの設定方法 本製品をご使用いただく前準備としてパソコンのネットワーク設定を行う必要があります。その方法を説明しています。
CONPROSYS WEB Settingについて	CONPROSYS WEB Settingの説明 本製品の各機能を設定する『CONPROSYS WEB Setting』で出来ることや、その方法を説明しています。
CODESYSのインストール	CODESYSの詳細な使用方法 『CODESYS』を使用した接続、機能設定などの詳細について説明しています。
プログラミングの基本	
通信方法の設定をする	
モニタリング編集をする	
困った時は	設定したのに動作しない場合など、困った時の解決策をご紹介します。
仕様	ハードウェア仕様およびCONPROSYS HMI仕様について説明しています。
付録	各データフォーマットの仕様などを記載しています。
各種サービス・お問い合わせ	当社のサービスやお問い合わせ先のご案内です。
索引	このマニュアルの索引です。

3. 製品使用までの流れ

本製品を使用するまでの、一般的な流れは以下のとおりです。



本製品を使用する環境やご使用のシステムの種類などによって、使用までの流れは異なる場合があります。

4. 関連マニュアルのご案内

本製品に関連するマニュアルは以下のように構成しています。

本書と併せてご活用ください。

◆ 必ずお読みください

名称	用途	内容	入手先
製品ガイド(または商品案内)	本製品開封後に必ずお読みください。	本製品をご使用になる前に同梱品を確認、注意いただくことについて説明しています。	製品に同梱(印刷物)
セットアップマニュアル	本製品をセットアップする時にお読みください。	セットアップに準備するものや接続、設置方法について説明しています。	 当社ホームページよりダウンロード(PDF)
リファレンスマニュアル (ハードウェア編)	本製品を運用する時にお読みください。	本製品の機能、設定などハードウェアに関する説明をしています。	 当社ホームページよりダウンロード(PDF)
リファレンスマニュアル (ソフトウェア編)またはシステムセットアップガイド	『CONPROSYS WEB Setting』を設定する時にお読みください。	『CONPROSYS WEB Setting』の各種設定方法について説明しています。	 当社ホームページよりダウンロード(PDF)

◆ 各種マニュアルのダウンロード

各種マニュアルは、以下のURLよりダウンロードしてご使用ください。

ダウンロード

<https://www.contec.com/jp/download/>

5. オンラインヘルプのご案内

演算や制御などの処理タスクを組み立てていく『CONPROSYS VTC』や、モニタリング画面の操作や編集をする『CONPROSYS HMI』の詳細情報はオンラインヘルプでも提供しています。

必要に応じてご活用ください。

◆ CONPROSYS VTC(Visual Task Control)

オンラインヘルプ

<http://data.conprosys.com/help/task/V1/jp/>



◆ CONPROSYS HMI(Human Machine Interface)

オンラインヘルプ

<http://data.conprosys.com/help/hmi/V1/jp/>



6. ファームウェアのご確認

ご使用を開始する前に、当社ホームページでファームウェアのバージョンをご確認いただき、常に最新バージョンのファームウェアをご使用ください。

最新のファームウェアにアップデートすることで、不具合が修正され、動作が安定します。

ダウンロード

<https://www.contec.com/jp/download/>

※ ファームウェアのアップデート方法は『**ファームウェア更新(P41)**』を参照ください。

安全にご使用いただくために

本製品を安全に使用するために、注意していただくことを説明しています。本製品をご使用になる前に、必ずお読みください。

1. 注意記号の説明

本書では、人身事故や機器の破壊をさけるため、次のシンボルで安全に関する情報を提供しています。
内容をよく理解し、安全に機器を操作してください。

 危険	この表示を無視して、誤った取り扱いをすると、人が死亡または重傷を負う危険が差し迫って生じることが想定される内容を示しています。
 警告	この表示を無視して、誤った取り扱いをすると、人が死亡または重傷を負う可能性が想定される内容を示しています。
 注意	この表示を無視して、誤った取り扱いをすると、人が損害を負う可能性が想定される内容および物的損害のみの発生が想定される内容を示しています。

2. 取り扱い上の注意

⚠ 注意

- 本製品または本書は機能追加、品質向上のため予告なく仕様を変更する場合があります。継続的にご利用いただく場合でも、必ず当社ホームページのマニュアルを読み、内容を確認してください。
- 本製品を改造しないでください。
改造をしたものに対しては、当社は一切の責任を負いません。
- 本製品の運用を理由とする損失、逸失利益などの請求につきましては、前項にかかわらず、いかなる責任も負いかねますのであらかじめご了承ください。

3. セキュリティに関する注意

ネットワークに接続する際は、存在するセキュリティリスクを考慮の上、セキュリティ対策事例を参考に本体および関連するネットワーク機器を適切に設定してください。

1. セキュリティリスク

- 外部ネットワークからの不正侵入に伴うシステムの停止、データの破損、情報の窃取、マルウェア※1 への感染。
- 侵入後にその機器を踏み台として、外部ネットワークへの攻撃。(被害者から加害者になる)
- 外部へのネットワーク接続に伴う意図しない情報漏洩。
- これら事故の二次被害として、風評被害、損害賠償負担、信用の失墜、機会損失等。

※1：マルウェア(Malicious Software)：悪意あるプログラム。ユーザーの望まない動作をするプログラム

2. セキュリティ対策事例

- 初期パスワードを変更する。(パスワード設定方法は、ご使用の製品の解説書/マニュアルを参照してください)
- パスワード強度の高いものを設定する。

半角英字小文字、大文字、数字等を含み、類推されにくいパスワードを使用する

- 定期的にパスワードを変更する。
- 不要なネットワークサービスや、不要な機能を停止(無効化)する。
- ネットワーク接続機器において、ネットワークでのアクセス元を制限する。※2
- ネットワーク接続機器において、ネットワークの解放ポートを制限する。※2
- 専用ネットワークやVPN※3 など閉域網を使ってネットワークを構築する。

※2：設定方法はネットワーク機器のメーカー各社へお問い合わせください。

※3：VPN(Virtual Private Network)：通信経路を認証や暗号化を用いて保護することにより、第三者が侵入することができない、安全なネットワークです。

不正アクセスの手段や抜け道(セキュリティホール)は、日夜新たに発見されており、それを防ぐ完璧な手段はありません。

インターネット接続には、常に危険が伴うことをご理解いただくとともに、常に新しい情報を入手し、セキュリティ対策を行うことを強くおすすめします。

パソコンのネットワーク 設定をする

本製品とパソコンの接続方法、ネットワーク設定、通信の
確認方法を説明しています。

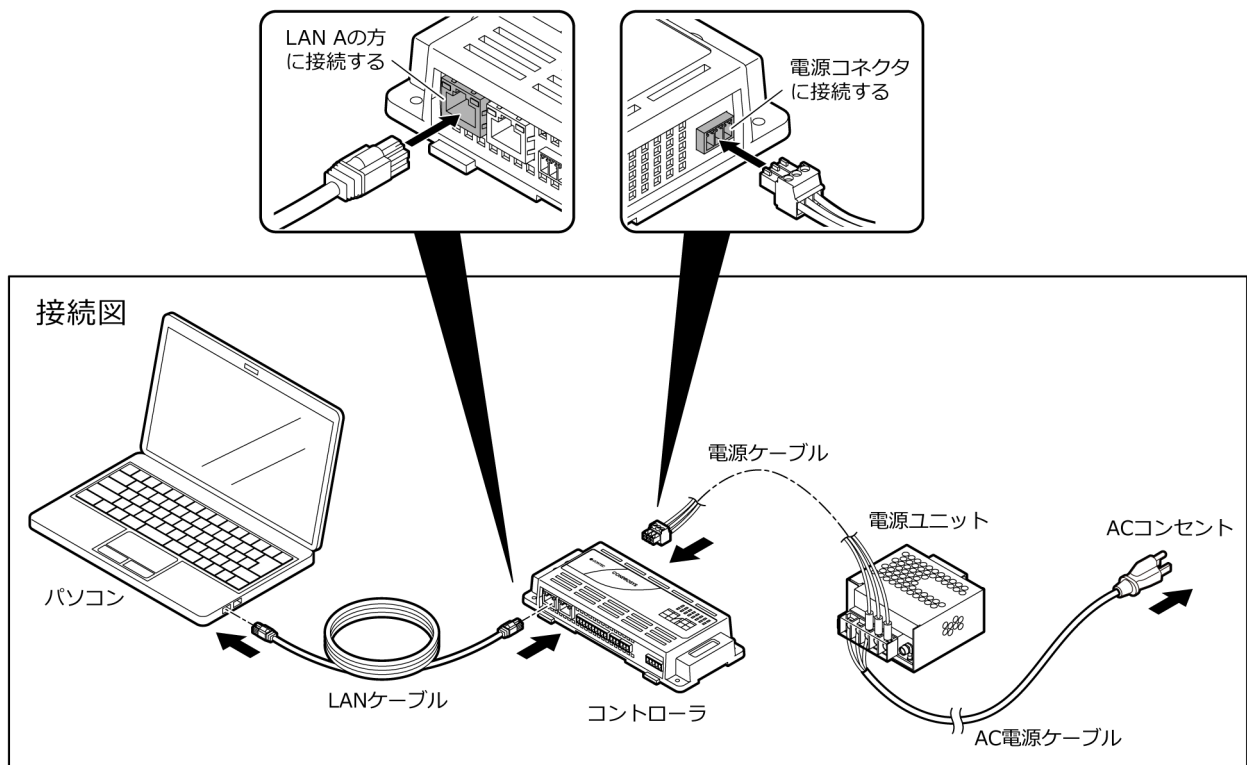
1. パソコンとの接続

本製品の各種設定を行うには、本製品と接続するパソコンの通信を確立するために、ネットワーク設定を行う必要があります。

はじめに、本製品とパソコンを接続します。

1 以下の接続図を参考にパソコン、コントローラ、電源ユニットを接続します。

※ 電源ケーブルの作り方は『リファレンスマニュアル(ハードウェア編)』を参照ください。



2 電源ユニットをACコンセントに接続し、パソコンのスイッチをONにします。

※ 電源ユニットをACコンセントに接続してからコントローラの起動が完了するまで、しばらく時間が掛かります。(1 - 2分が目安です)

2. パソコンのネットワーク設定をする

以下の設定手順を参考にして、次のようなIPアドレスになるようにパソコンのネットワーク設定を行います。

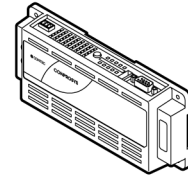
パソコン側



IPアドレス 10 . 1 . 1 . **200**

サブネットマスク 255 . 0 . 0 . 0

コントローラ側



IPアドレス 10 . 1 . 1 . **101** ※

サブネットマスク 255 . 0 . 0 . 0 ※

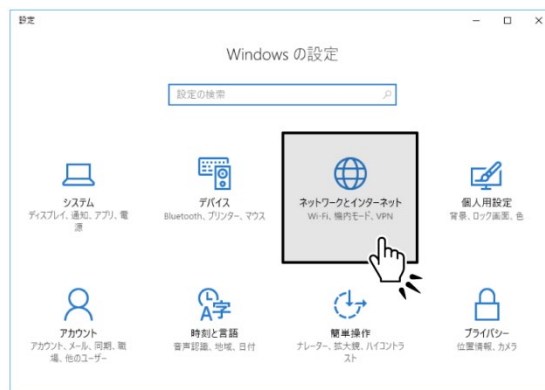
※出荷時の設定です

IPアドレスの太字部分(**200**と**101**)は重複しない番号を設定する必要があります。

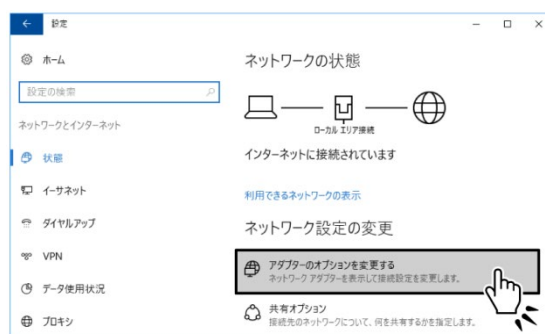
※使用するパソコンにプロキシが設定されている場合は、使用しないように設定する必要があります。

1. パソコンのネットワーク設定手順 (Windows 10の場合)

- 1 [Windowsの設定] 画面で [ネットワークとインターネット] をクリックします。



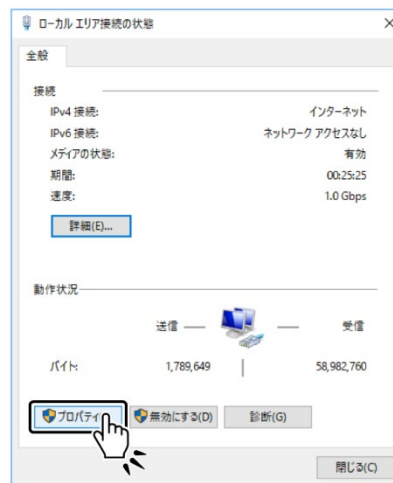
- 2 [ネットワーク設定の変更] 内の [アダプターのオプションを変更する] をクリックします。



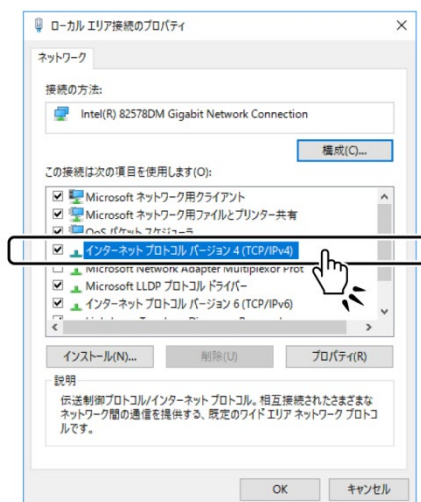
3 次に表示された画面の「ローカルエリア接続」をダブルクリックします。



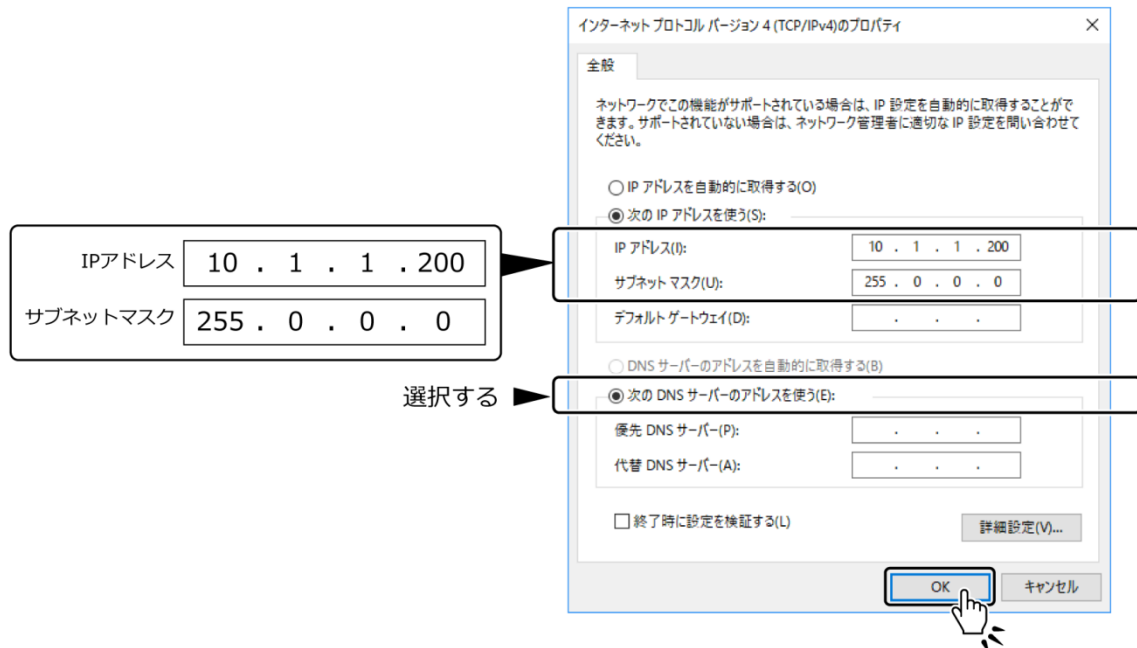
4 「ローカルエリア接続の状態」ダイアログボックスの「プロパティ」をクリックします。



5 「ローカルエリア接続のプロパティ」ダイアログボックスの中から「インターネットプロトコルバージョン4(TCP/IPv4)」をダブルクリックします。



- 6** [インターネットプロトコルバージョン4(TCP/IP)のプロパティ] で以下のようにIPアドレス、サブネットマスクを設定します。



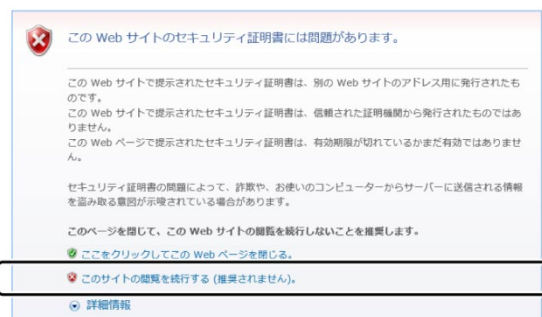
- 7** [OK] → [OK] → [閉じる] ボタンをクリックしてダイアログボックスを閉じ、ネットワーク設定を終了させます。

3. 通信しているか確認する

- 1** パソコンでWebブラウザを起動し、アドレスバーにコントローラのIPアドレス(10.1.1.101)を入力し、[Enter] キーを押します。
- ユーザー名とパスワードを入力する画面が表示されたら、ユーザー名とパスワードを入力して [OK] をクリックします。



- ※ 対応しているWebブラウザは『**対応Webブラウザ(P25)**』を参照ください。
- ※ 実際に運用する時は、CONPROSYS WEB Setting [メンテナンス] - [ユーザー/パスワード] でユーザー名とパスワードを変更してご使用ください。
- ※ IPアドレスを入力して [Enter] キーを押した後で、セキュリティ証明書の確認画面が表示された場合は [このサイトの閲覧を続行する] を選択します。



- 2** CONPROSYS WEB Settingの画面が表示されたら、パソコンとコントローラの通信が確立しています。

CONPROSYS WEB Settingについて

本製品のシステムや各種設定について説明をしています。

1. CONPROSYS WEB Settingの概要

本製品の各種設定は『CONPROSYS WEB Setting』で行います。
CONPROSYS WEB Settingは、Webブラウザ上で簡単に設定が行えます。

1. 対応Webブラウザ

CONPROSYS WEB Settingは以下のWebブラウザに対応しています。

対応Webブラウザ	対応バージョン
Microsoft Internet Explorer	Ver. 11以降
Google Chrome	Ver. 52以降
Mozilla Firefox	Ver. 55以降

※ 対応Webブラウザ以外では動作に不具合が生じる場合があります。必ず対応しているWebブラウザをご使用ください。

2. CONPROSYS WEB Settingの起動

本製品と接続したパソコンでWebブラウザを起動し、アドレスバーにコントローラのIPアドレス「10.1.1.101」を入力し、[Enter] キーを押します。

ユーザー名とパスワードを入力する画面が表示されたら、ユーザー名とパスワードを入力して [OK] をクリックします。

※ 本製品の接続方法については、『セットアップマニュアル』を参照ください。



※ IPアドレスを入力して [Enter] キーを押した後で、セキュリティ証明書の確認画面が表示された場合は
[このサイトの閲覧を続行する] を選択してください。

3. CONPROSYS WEB Settingの基本操作

◆ CONPROSYS WEB Setting画面の構成

CONPROSYS WEB Settingは画面左側のメニュー部分で設定項目をクリックすると、画面右側の表示部分にその設定項目内容が表示されます。



2. メニュー項目の機能一覧

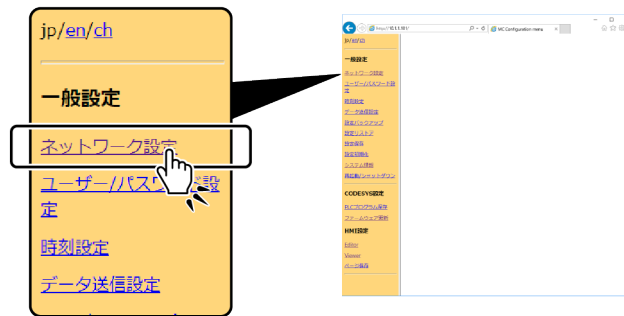
メニュー項目名	機能	詳細情報
一般設定		
ネットワーク設定	IPアドレスなど、ネットワークに関する設定を行います。	P28
ユーザー/パスワード設定	本製品にWebブラウザでログインするユーザー/パスワードを設定します。	P30
時刻設定	日付時刻を取得するNTPサーバーを設定します。	P31
データ送信設定	計測したデータの送信先サーバーを設定します。	P32
シリアル設定	シリアルポートの通信パラメーターを設定します。	P33
設定バックアップ	モニタリング画面、タスクプログラム、各種設定をバックアップします。	P34
設定リストア	バックアップしたファイルを元に、モニタリング画面、タスクプログラム、各種設定を復元します。	P35
設定保存	設定した各内容をROMに保存します。	P36
設定初期化	全ての設定を工場出荷の状態に戻します。	P37
システム情報	本製品の情報を表示します。	P37
再起動/シャットダウン	本製品の再起動/電源オフを行います。	P40
CODESYS設定		
PLCプログラム保存	PLCプログラムをROMに保存します。	P41
ファームウェア更新	ファームウェアを更新します。	P41
HMI設定		
Editor	モニタリング画面(CONPROSYS HMI)の編集画面を表示します。	P42
Viewer	モニタリング画面(CONPROSYS HMI)を表示します。	P43
ページ保存	Editorで作成した画面(CONPROSYS.HMI)の内容をROMに保存します。	P43

3. 機能の詳細説明

1. ネットワーク設定

本製品のIPアドレスの設定やネットワークの通信確認を行います。

※ LAN A、LAN Bそれぞれにネットワークの設定ができます。



ネットワーク設定

LAN A	eth0
Select	☒ 固定IPアドレス ☐ DHCP取得
IP address	10.1.1.101
Subnet mask	255.0.0.0
Default gateway	10.1.1.254
LAN B	eth1
Select	☒ 固定IPアドレス ☐ DHCP取得
IP address	192.168.1.101
Subnet mask	255.255.255.0
Default gateway	192.168.1.254
DNS server1	10.1.1.254
DNS server2	
set	

設定を有効にするには設定の保存と再起動が必要です

◆ Select

IPアドレスの設定方法を選択します。

【固定IPアドレス】：IPアドレスやサブネットマスクなどを設定します。

【DHCP取得】：DHCPクライアントを使用してIPアドレスを自動で取得します。

【デフォルト値】：固定IPアドレス

◆ IP address

IPアドレスを設定します。

【固定IPアドレス】を選択した場合に有効です。

【デフォルト値】：10.1.1.101

◆ Subnet mask

サブネットマスクを設定します。

【固定IPアドレス】を選択した場合に有効です。

【デフォルト値】：255.0.0.0

◆ Default gateway

デフォルトゲートウェイのIPアドレスを設定します。

【固定IPアドレス】を選択した場合に有効です。

設定しない場合は何も入力せず空欄にします。

【デフォルト値】：10.1.1.254

◆ DNS server1、DNS server2

DNSサーバーのIPアドレスを設定します。

【固定IPアドレス】を選択した場合に有効です。

設定しない場合は何も入力せず空欄にします。

【デフォルト値】：DNS server1：10.1.1.254

DNS server2：(入力なし)

2. ユーザー/パスワード設定

Webブラウザでログインするユーザー、パスワードの設定を行います。
出荷時のパスワードから変更してご使用ください。



ユーザー/パスワード設定

user name	
passwd	
	add

user name	
pc341	del

設定を有効にするには設定の保存と再起動が必要です

ユーザー/パスワード設定

WebブラウザでCONPROSYS WEB Setting画面にログインするユーザーの追加、削除ができます。

ユーザーおよびパスワードは、31文字以下の半角英数字で入力します。

設定の変更は、本製品の再起動後に有効になります。

セキュリティ強化のため、運用時にはデフォルトと異なるユーザーを追加し、デフォルトのユーザーを削除します。

3. 時刻設定

時刻設定では、日付時刻を取得するNTPサーバーの設定を行います。



時刻同期設定

現在の日時	2016-04-08 17:12:10	reload	write
同期サーバ	ntp.nict.jp		
同期を実施する時刻	02:20		
タイムゾーン	UTC+09(JST) ▼		

set
now

設定を有効にするには設定の保存と再起動が必要です

日付と時刻の設定

yyyy-mm-dd hh:mm:ss	2016	-	04	-	08	17	:	12	:	10
---------------------	------	---	----	---	----	----	---	----	---	----

set

◆ 時刻同期設定

現在の日時

現在の日時を表示します。

[reload] をクリックすると同期サーバから時刻を更新します。

[write] をクリックすると現在の時刻をROMに保存します。

同期サーバ

NTPを使用する場合、NTPサーバーのアドレスをセットします。

【設定範囲】：FQDNまたはIPアドレス

【デフォルト値】：ntp.nict.jp

同期を実施する時刻

本製品の起動時と毎日2:20に同期サーバから時刻を更新(同期)します。(設定項目ではありません)

タイムゾーン

世界標準時間と本製品を使用する地域の標準時間との時差を設定します。

【デフォルト値】：UTC+09(JST-9)

◆ 日付と時刻の設定

手動で時刻設定をする場合、フォームに日時、時間を入力して更新します。

【設定範囲】：1970年1月1日0時0分0秒 - 2038年1月19日3時14分7秒

【デフォルト値】：1970年1月1日0時0分0秒

4. データ送信設定

計測したデータの送信先サーバーを設定します。

送信データのフォーマットについては、『[送信データフォーマット\(P175\)](#)』を参照ください。



データ送信設定

データ送信先URL

set

設定を有効にするには設定の保存と再起動が必要です

◆ データ送信先URL

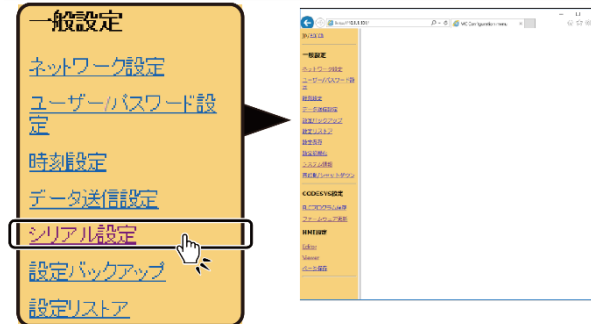
計測したデータを送信するサーバーのURLを入力します。

【デフォルト値】：設定なし

5. シリアル設定

シリアルポートの通信パラメーターを設定します。

シリアルポートで設定したいポート番号を選択してから、通信パラメーターを設定してください。



シリアル設定

シリアルポート	1 ▼
伝送モード	全二重 ▼

set

設定を有効にするには設定の保存と再起動が必要です

◆ シリアルポート

通信パラメーターの設定があるシリアルポート番号の一覧が表示されます。

ここでは設定したいシリアルポートの番号を選択します。

シリアルポート番号は、製品ごとに異なりますので、『シリアルポート(P67)』を参照してください。

◆ 伝送モード

データ伝送の通信方式を設定します。

ハードウェアの設定と合わせてください。

【設定範囲】：全二重、半二重

【デフォルト値】：全二重

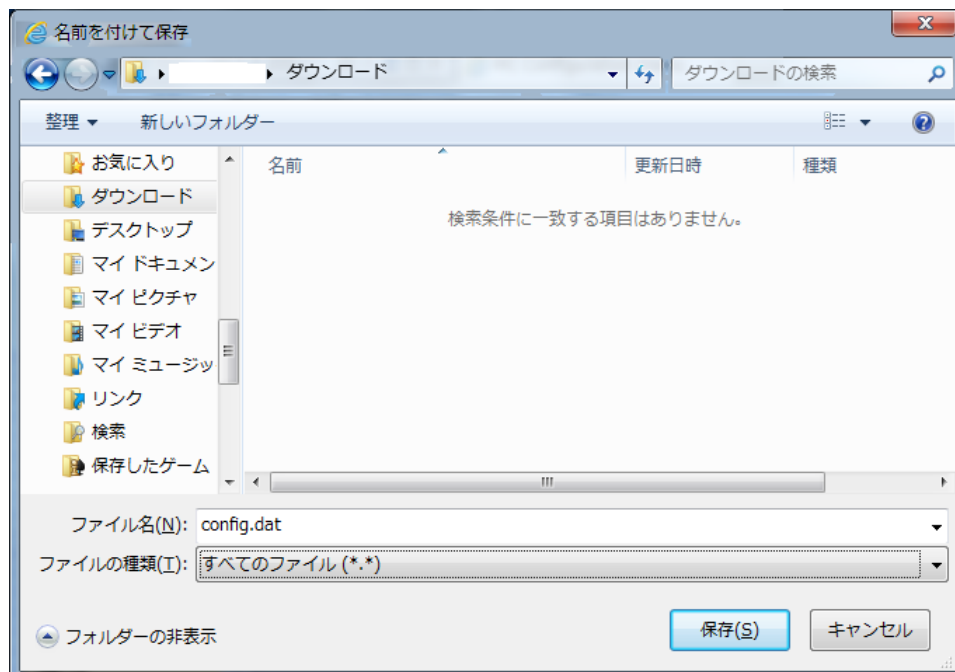
6. 設定バックアップ

モニタリング画面、タスクプログラム、各種設定をバックアップします。



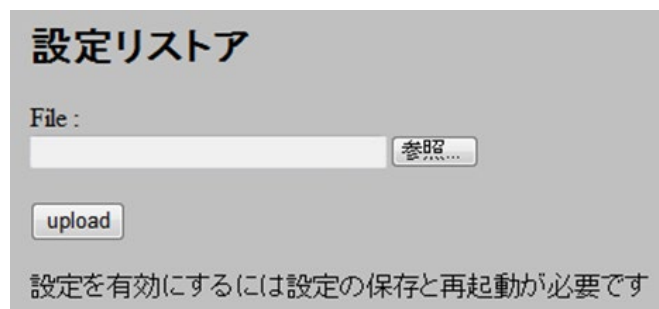
各種設定のバックアップは、[download] をクリックすると保存されます。

デフォルトのファイル名は、config.datになっていますので、任意のファイル名をつけて保存してください。



7. 設定リストア

バックアップしたファイルを元に、モニタリング画面、タスクプログラム、各種設定を復元します。



設定バックアップで保存したバックアップファイルを「参照」から選択して、「upload」をクリックしてください。

復元された各種設定は、それぞれの設定画面で確認できますが、設定を有効にするには「設定保存」と「再起動」を行ってください。

8. 設定保存

CONPROSYS WEB Settingで設定した内容をROMに保存します。

[save to ROM] をクリックするとPWRのLEDが点滅し、保存を開始します。



⚠ 注意

設定の保存中は、本体のPWR LEDの点滅が終了するまで電源を切らないでください。(所要時間：5秒程度)

設定保存を行わないで再起動や電源OFFをすると設定した内容が元に戻ります。

9. 設定初期化

本製品を工場出荷時のデフォルト設定に戻します。



設定初期化

initialize

設定を有効にするには設定の保存と再起動が必要です

[initialize] をクリックすると設定を初期化します。

初期化した設定を有効にするには、[設定保存] と [再起動] を行ってください。

10. システム情報

本製品の情報を表示します。



以下の画面が表示されます。

システム情報

バージョン	1.2.0	
シリアル番号	XXXXXXXXXXXX	
ID	XXXX - XXXX - XXXX - XXXX - XXXX - XXXX - XXXX - XXXX	
MACアドレス	LAN A	XX:XX:XX:XX:XX:XX
	LAN B	XX:XX:XX:XX:XX:XX
ランタイムバージョン	1.2.0.0	
ドライババージョン	1.1.0.0	
電池残量	あり	
サーバ接続ログ	表示	
詳細	表示	
license	表示	

◆ バージョン

ファームウェアのバージョンが表示されます。

◆ シリアル番号

本製品のシリアル番号が表示されます。

◆ ID

クラウドサーバーのCDSまたはCDS2に登録する際に必要なIDが表示されます。

◆ MACアドレス

有線LANのMACアドレスが表示されます。

◆ ランタイムバージョン

ランタイムのバージョンが表示されます。

◆ ドライババージョン

ドライバのバージョンが表示されます。

◆ 電池残量

電池残量の有無が表示されます。

◆ サーバー接続ログ

サーバーとの通信ログを表示します。

項目	説明
Webサーバー通信ログ	データ送信先URLに指定されたサーバーとの最新の通信ログを表示します。
NTPサーバー通信ログ	指定されたNTPサーバーとの最新の通信ログを表示します。

Webサーバ通信ログ

```
* Trying 10.1.1.1...
* % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total    Spent    Left   Speed

  0    0    0    0    0    0     0     0  --:--:-- --:--:-- --:--:--    0* connect to 10.1.1.1 port 80 failed: No route to host
* Failed to connect to 10.1.1.1 port 80: No route to host
* Closing connection 0
curl: (7) Failed to connect to 10.1.1.1 port 80: No route to host
```

NTPサーバ通信ログ

```
3 Jul 10:33:23 ntpdate[1376]: step time server 133.243.238.163 offset -2588356.802507 sec
```

メール送信ログ

ログが見つかりません

◆ 詳細

システム情報の詳細を表示します。

uptime

```
23:43:49 up 27 min,  0 users,  load average: 4.36, 4.27, 3.44
```

free

```
Mem:          total        used        free        shared        buffers
-/+ buffers:    513504      128316      385188           0          244
Swap:           0           0           0
```

df

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/root	31729	28133	1958	93%	/
none	256752	0	256752	0%	/var
none	256752	56	256696	0%	/tmp
none	256752	0	256752	0%	/dev
/dev/mtdblock5	18688	16496	2192	88%	/mnt/mtd
tmpfs	256752	37500	219252	15%	/home
tmpfs	32768	688	32080	2%	/home/CF
tmpfs	16384	12	16372	0%	/home/Ram

ps aux

PID	USER	TIME	COMMAND
1	root	0:02	init
2	root	0:00	[kthreadd]
3	root	0:00	[ksoftirqd/0]
5	root	0:14	[kworker/u:0]
6	root	0:00	[khelper]
7	root	0:00	[netns]
8	root	0:00	[sync_supers]
9	root	0:00	[bdi-default]
10	root	0:00	[kblockd]

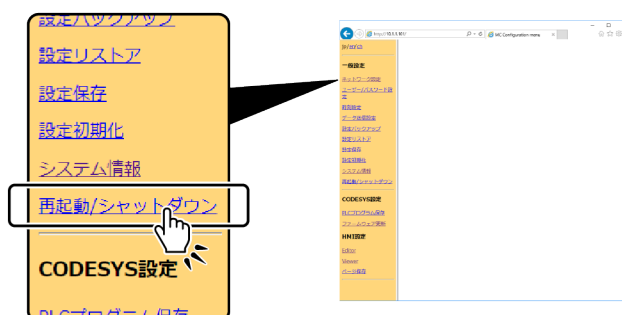
◆ license

「表示」をクリックすると、使用ソフトウェアのライセンス情報を表示します。

ソフトウェアの利用には承諾が必要です。

11. 再起動/シャットダウン

本製品の再起動または電源オフを行います。



再起動/シャットダウン

reboot ▼ start

reboot(再起動)または、shutdown(電源オフ)を選択して「start」をクリックするとLEDの点滅が始まり、再起動または電源オフを実行します。

12. PLCプログラム保存

PLCプログラムをROM領域に保存します。

〔書き込み〕ボタンをクリックするとST1とST2のLEDが点滅し、保存を開始します。
保存が完了すると点滅が停止します。



⚠ 注意

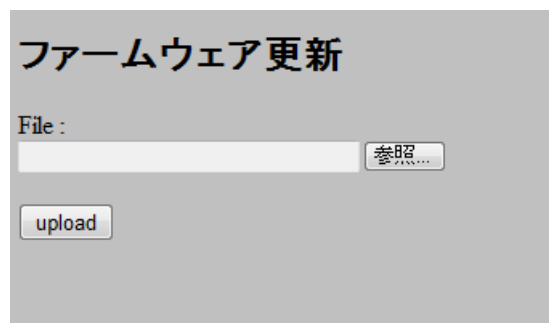
設定保存中は、本体LEDのST1とST2が点滅します。

その間、電源を切らないでください。(所要時間：5秒程度)

設定保存を行わないで再起動や電源OFFをすると設定した内容が元に戻ります。

13. ファームウェア更新

ファームウェアの更新を行います。



ファームウェアの更新ファイルは当社ホームページからダウンロードできます。

ダウンロード

<https://www.contec.com/jp/download/>

ダウンロードしたファイルを「参照」ボタンから選択し、「upload」ボタンをクリックしてください。

※ダウンロードしたファイルはzip形式で圧縮されているので、解凍し、拡張子binのファイルを選択してください。

⚠ 注意

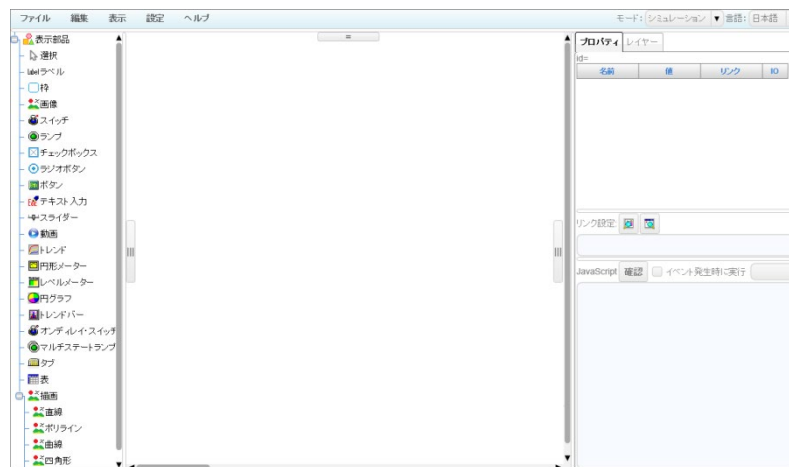
ファームウェアの更新中は、本体LEDのST1とST2が点滅します。

その間、絶対に電源を切らないでください。データが破損して起動できなくなります。

14. Editor

モニタリング画面を制作/変更を行う画面を表示します。

表示アイテムを並べて簡単に独自のモニタリング画面を作ることができます。

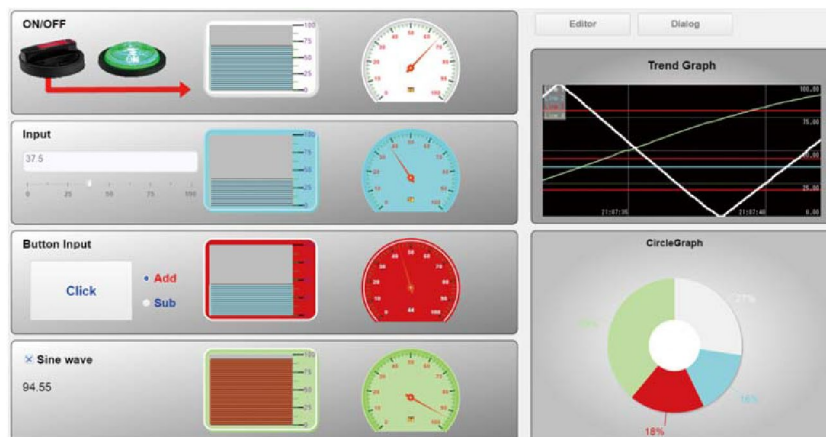


※モニタリング編集画面の詳細情報は『モニタリング編集をする(P151)』を参照ください。

15. Viewer

モニタリング画像を表示します。

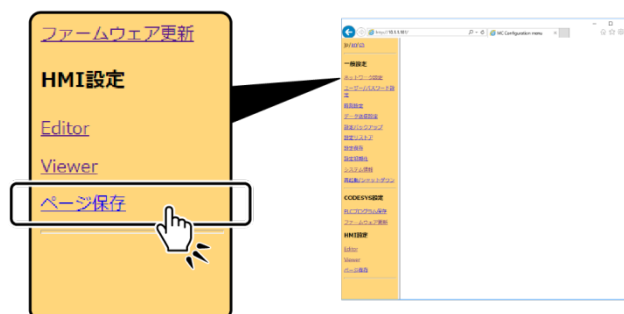
モニタリング画面は、入力されている信号の状態をモニタリングできます。



※ モニタリング編集画面の詳細情報は『モニタリング編集をする(P151)』を参照ください。

16. ページ保存

Editorで作成したページの内容をROM領域へ保存します。



CODESYSのインストール

CODESYSのインストール手順を説明しています。

1. CODESYSをインストールする

PACシリーズは、プラットフォームに依存せずPLCやHMIの開発が可能な国際標準IEC61131-3準拠の統合開発環境『CODESYS』が使用できます。

CODESYSを利用するには、CODESYSの開発環境および『CONPROSYS用CODESYSパッケージ』をパソコンにインストール必要があります。

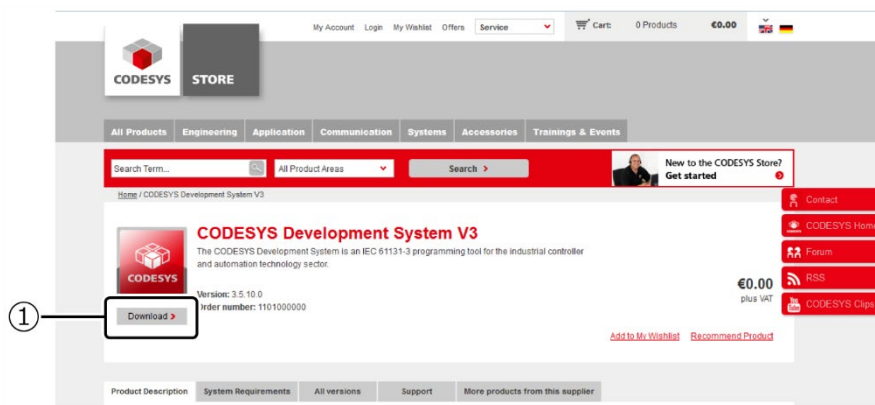
1 CODESYSの開発環境の入手方法

CODESYSの開発環境は、『CODESYS Store』からダウンロードしてください。

2 WebブラウザでCODESYS Storeにアクセスして、① [Download] ボタンをクリックします。

CODESYSは随時アップデートされますので最新のバージョンをダウンロードしてください。

ダウンロード <http://store.codesys.com/codesys.html>

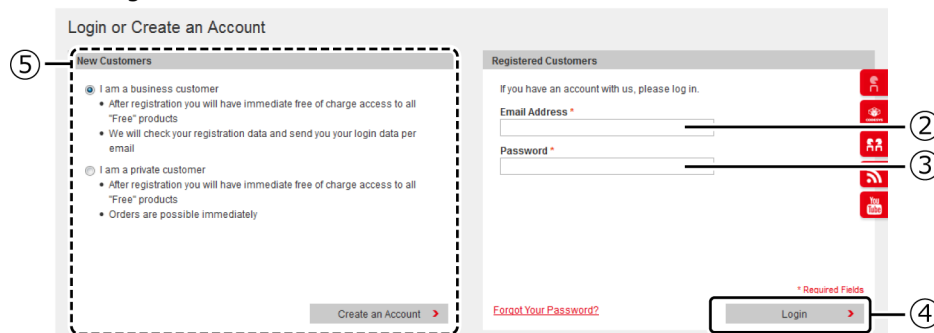


3 [Download] ボタンをクリックした後、『Login or Create an Account』画面が表示されます。

CODESYS Storeのアカウントが有る場合は『Registered Customers』で②『Email Address』と③『Password』を入力して④ [Login] ボタンをクリックします。

新規ユーザーの場合は、⑤『New Customers』でユーザー登録を行ってからLoginします。

▼『Login or Create an Account』画面



- 4 『Registered Customers』で『Email Address』と『Password』を入力してダウンロードセンターへログインします。ログイン後、[Download] をクリックすることで、ダウンロードが開始します。
- 5 ダウンロードしたファイルを解凍し、セットアップファイル(拡張子.exeのファイル)を実行して、CODESYSをインストールします。

2. CODESYS用パッケージをインストールする

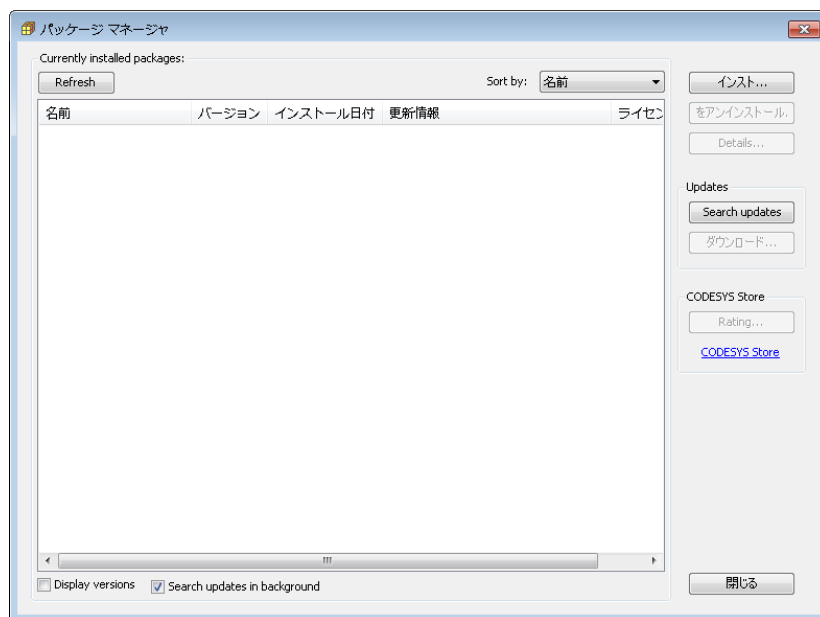
本製品をCODESYSで使用するためには、CONPROSYS用CODESYSパッケージのインストールが必要です。

- 1 当社ホームページのダウンロードメニューより [ドライバ・ソフトウェア] を選択し。製品名かカテゴリ(PACシステム)で絞り込んで [検索] をクリックし、本製品のドライバをクリックします。

ダウンロード <https://www.contec.com/jp/>

- 2 CODESYSパッケージをダウンロードします。
- 3 CODESYSを実行し、メニューの [ツール] から [パッケージマネージャ...] を選択します。
- 4 [インストール...] をクリックして、ダウンロードしたファイルを解凍し、CODESYSパッケージファイルを指定します。

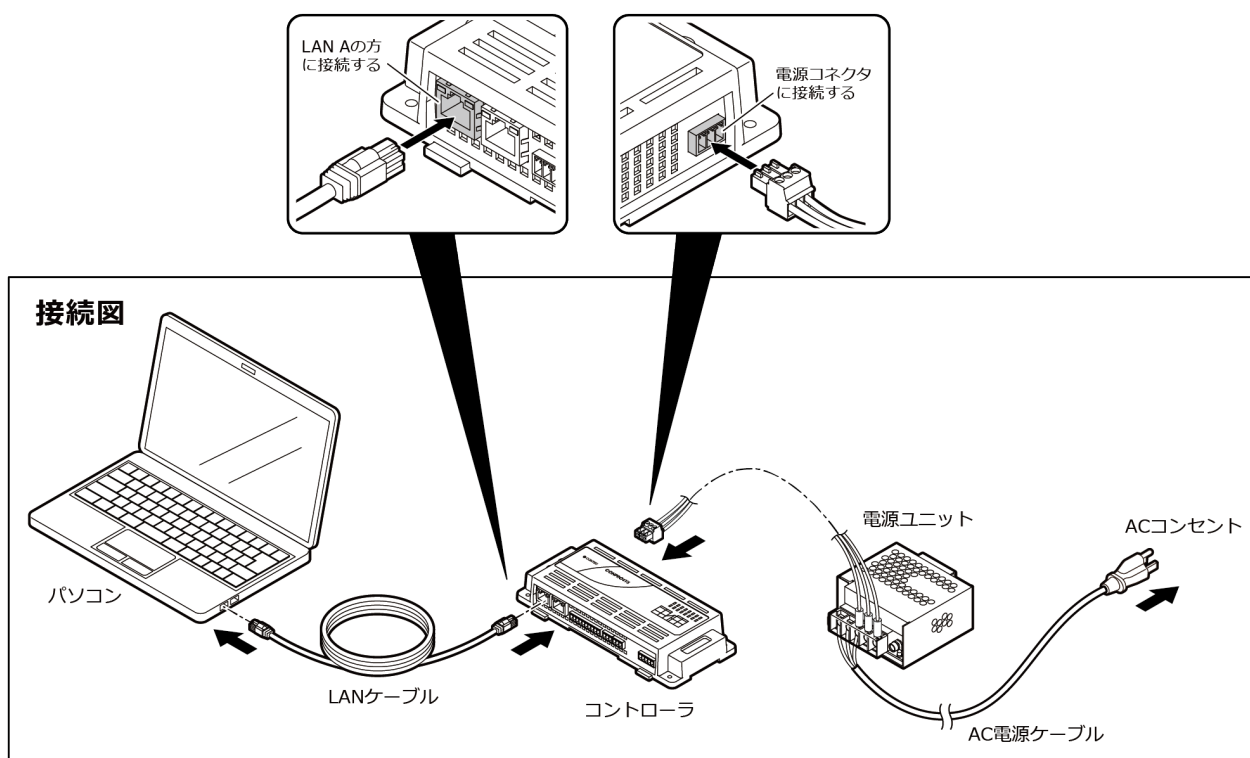
ファイル名 : CONTEC CONPROSYS PAC Series.package



3. パソコンとCONPROSYSを接続する

- 1 CODESYS開発環境パソコンのLANポートと本製品のLAN Aポートをイーサネットケーブルで接続します。

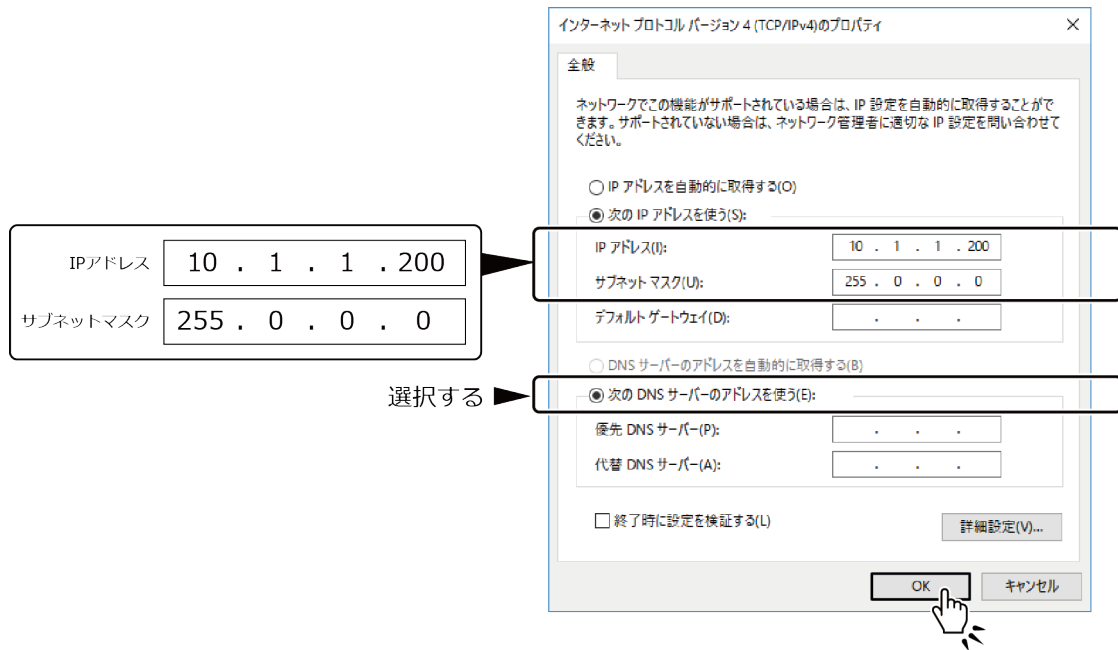
※ パソコンとの接続は、本製品のLAN Aポートを使用してください。LAN Bは、EtherCAT通信、Modbus TCP Slave接続用に使用します。



- 2 本製品の電源を投入します。
- 3 パソコンのネットワーク設定を変更します。

本製品のLAN Aポートは、工場出荷時設定ではIPアドレスが「10.1.1.101」、サブネットマスクは「255.0.0.0」となっています。

LAN AのIPアドレスが「10.1.1.101」の場合は、パソコンのIPアドレス設定を「10.1.1.200」などに設定してください。サブネットマスクは「255.0.0.0」に設定します。



※ PLCコントローラには、ネットワーク上で他のデバイスと重複しないIPアドレスを設定する必要があります。PLCコントローラのIPアドレスは、Webブラウザ経由で設定が可能です。

※ IPアドレス設定の詳細は『パソコンのネットワーク設定をする(P18)』を参照ください。

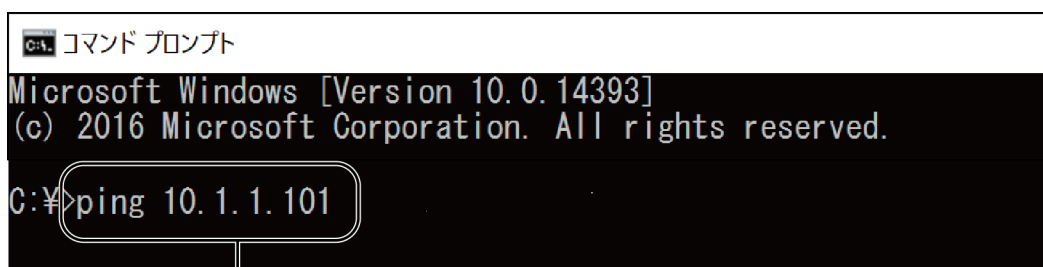
4 pingコマンドなどを使用して、パソコンからPLCコントローラへの接続が行えていることを確認してください。

コマンドプロンプトを開き、IPアドレス「10.1.1.101」に対してpingコマンドを実行します。

フォーマット

ping 機器のIPアドレス

応答があれば接続されていることが確認できます。



ping 10.1.1.101 と入力して [Enter] キーを押す

4. ファームウェアのバージョンアップ

CODESYS関連のファームウェアをバージョンアップします。

- 1 当社ホームページのダウンロードメニューより [ドライバ・ソフトウェア] を選択します。
製品名またはカテゴリ(PACシステム)で絞り込んで [検索] をクリックし、本製品のドライバをクリックします。

ダウンロード

<https://www.contec.com/jp/>

- 2 ファームウェアをダウンロードします。
(ダウンロードしたファイルはzip形式で圧縮されているので、解凍し、拡張子binのファイルを選択してください)
- 3 『パソコンとCONPROSYSを接続する(P48)』の手順で、パソコンと本製品を接続します。
- 4 Webブラウザを起動し、アドレス入力部へ「http://10.1.1.101/」(PLCコントローラのIPアドレス)を入力します。
※「https://10.1.1.101/」でもアクセス可能です。『証明書エラー』画面が表示される場合は、[このサイトの閲覧を続行する] を選択してください。
- 5 ユーザー名に「pc341」、パスワードに「pc341」と入力してログインします。
ファームウェアのバージョンは『一般設定』の『システム情報』画面で確認できます。



- 6 『CODESYS設定』の [ファームウェア更新] を選択して [ファイル選択] をクリックし、解凍したファームウェアを開きます。

- 7** [upload] ボタンをクリックするとファームウェアの更新が開始します。ファームウェア更新中は本体のST1とST2 LEDが点滅を繰り返します。

注意

ファームウェアの更新中は、本体LEDのST1とST2が点滅します。
その間、絶対に電源を切らないでください。データが破損して起動できなくなります。

- 8** 更新が完了すると、自動で再起動を行いますので、『システム情報』画面でバージョンが更新されていることを確認してください。

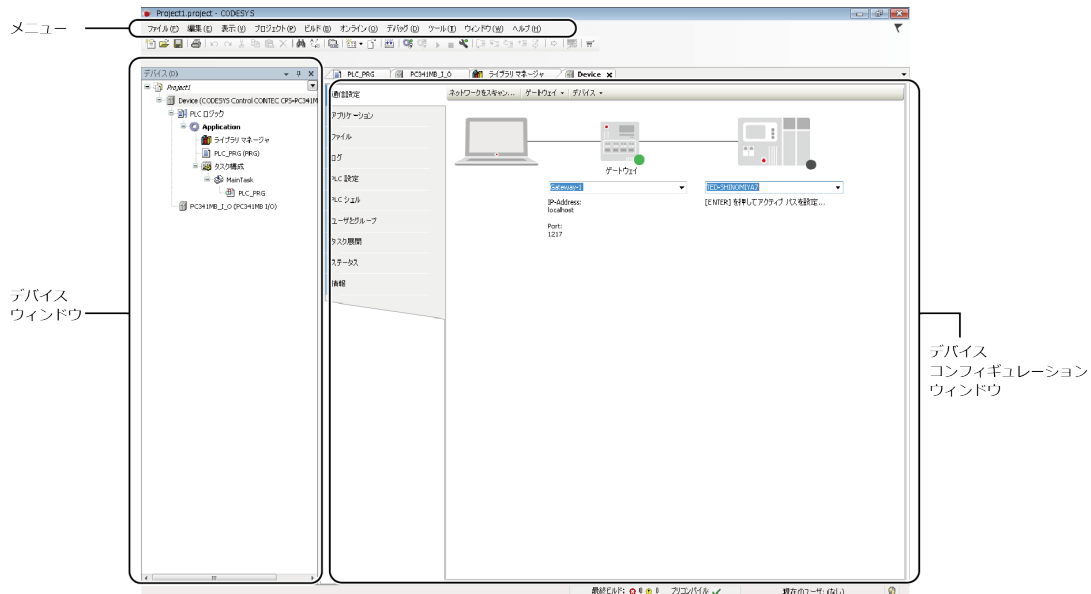
プログラミングの基本

CODESYSのプログラミングで、本製品を制御するための基本的な手順を記載しています。

1. CODESYS各部の名称

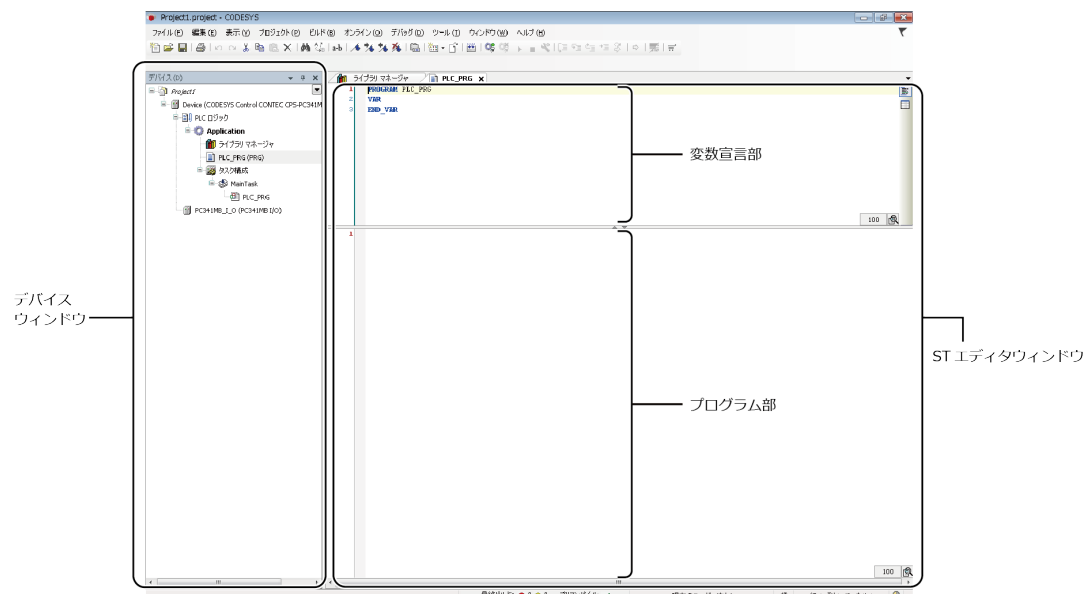
CODESYSの基本画面は、下図のようなウィンドウで表示します。

1. デバイスウィンドウ、デバイスコンフィギュレーションウィンドウ



※ デバイスコンフィギュレーションウィンドウは、デバイスウィンドウの『Device (CODESYS Control CONTEC CPS-PCXXXXXXXXXX)』アイコンをダブルクリックすると開きます。

2. STエディタウィンドウ



※ STエディタウィンドウは、デバイスウィンドウの『PLC_PRG(PRG)』アイコンをダブルクリックすると開きます。

2. ST言語の書式と記述方法

本書のプログラミング例は、ST言語を使用しています。

代入式や条件式、コメントアウトの方法など、ST言語の基本的な書式を示します。

◆ コメント

プログラムにコメントを追加します。

```
/* コメントの文章 */  
(* コメントの文章 *)  
// コメントの文章
```

- 複数行をコメントアウトする場合は、「/* */」または「(* *)」で囲みます。
- 1行のコメントアウトは、先頭に「//」を付けます。

◆ 代入式

変数に値を代入します。

```
D00 := 1;
```

- 「変数名」:=「値」; は、値を変数に代入します。
- 行末には「;」が必要です。

◆ 型宣言

変数のデータ型を宣言します。

```
StartFlag : BOOL := TRUE;  
Error : BOOL;  
TestString : STRING := 'Test String!';
```

- 1行目：変数「StartFlag」に代入する値をBoolean値(True/False)とし、Trueを代入する。
- 2行目：変数「Error」に代入する値をBoolean値とする。
- 3行目：変数「TestString」に代入する値を文字列とし、「TestString!」を代入する。

◆ 条件式(IF - THEN)

条件を判定してルーチンを実行/回避する。

```
IF StartFlag THEN  
  (ルーチン)  
END_IF
```

- 変数「StartFlag」の値が「True」であれば、END_IFで囲まれたルーチンを実行する。

◆ 条件式(CASE - OF)

条件を判定してルーチンを実行/回避する。

```
CASE State OF  
  0:  
    (ルーチン1)  
  1:  
    (ルーチン2)  
END_CASE
```

- 変数「State」の値が「0」であれば、ルーチン1を実行する。
- 変数「State」の値が「1」であれば、ルーチン2を実行する。

※ ST言語の情報については、PLCプログラミングの国際基準IEC61131-3を普及する第三者機関である『PLCopen』サイトを参照ください。

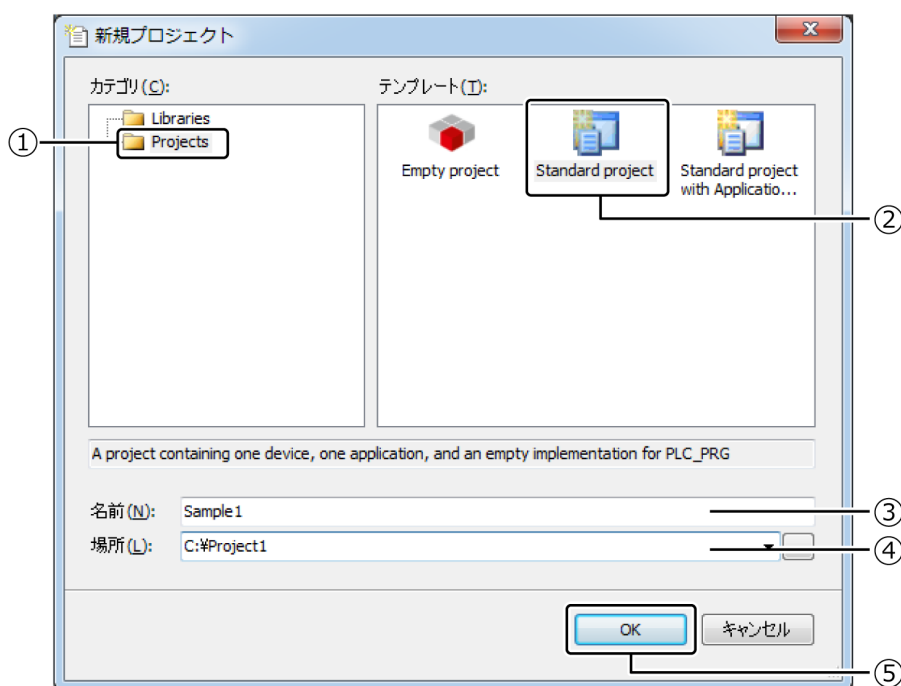
PLCopen

<https://www.plcopen-japan.jp/>

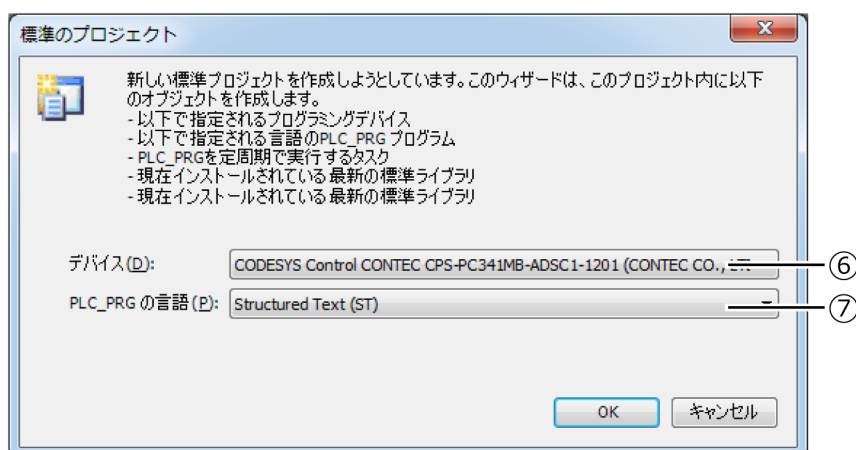
3. 基本的なプログラミング手順

1. 新規プロジェクトの生成

- 1 CODESYSを起動し、[ファイル] メニューから、[新規プロジェクト] を選択します。
- 2 『新規プロジェクト』のダイアログで、カテゴリから①「Projects」、テンプレートから②「Standard project」を選択します。
プロジェクトの③『名前』と④『場所』(保存先)を指定して⑤ [OK] をクリックします。

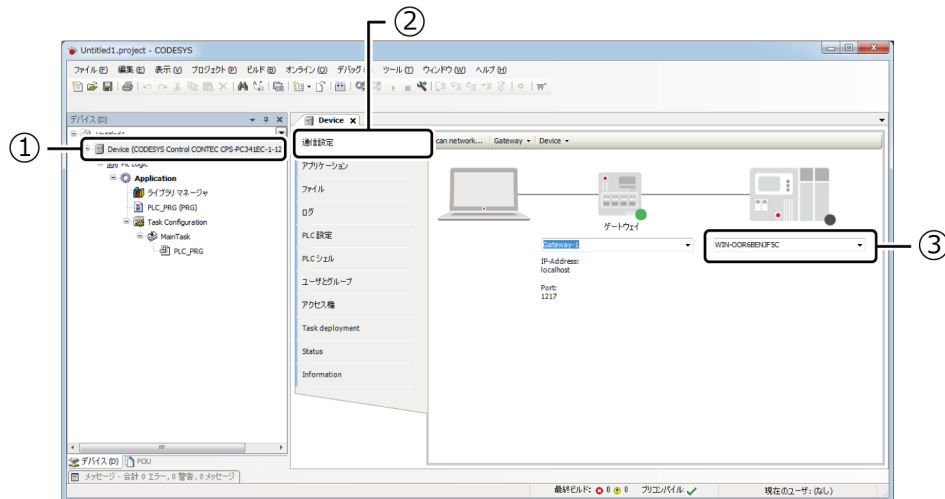


- 3 『標準のプロジェクト』のダイアログで⑥『デバイス』欄には、使用するコントローラを選択します。
デバイスは、「CODESYS Control CONTEC CPS-PCXXXXXXXXXX」のように表記されます。
⑦『PLC_PRGの言語』は、Structured Text (ST)を選択します。

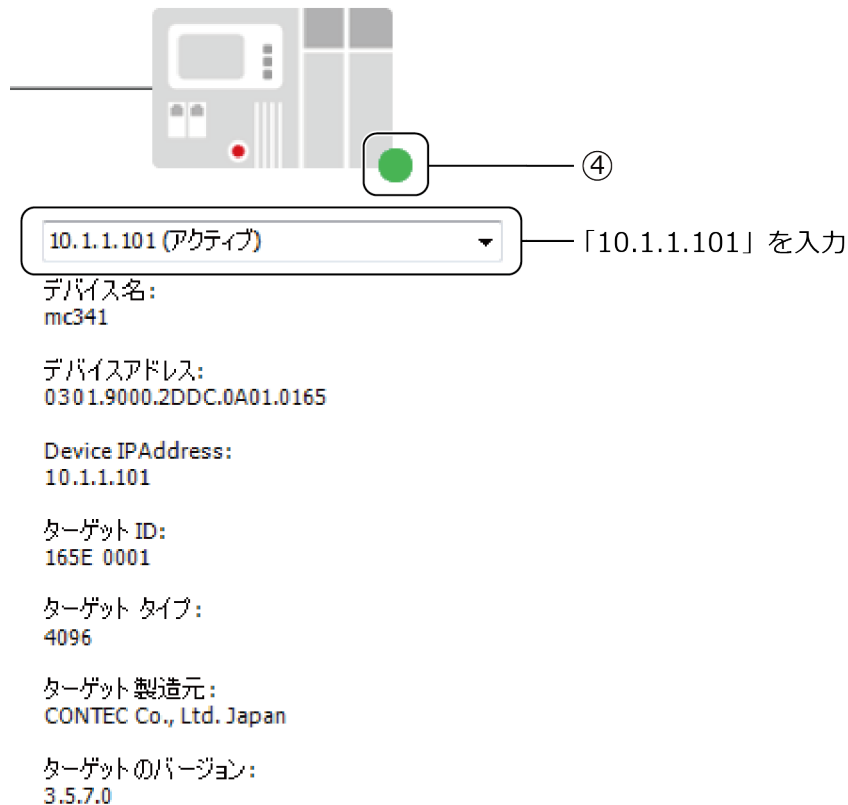


2. CODESYSからコントローラに接続する

- 1 デバイスウィンドウの①『Device(CODESYS Control CONTEC CPSPCXXXXXXXXXX)』アイコンをダブルクリックして『デバイスコンフィギュレーションウィンドウ』を開き、② [通信設定] タブをクリックします。
- 2 ③『コンボボックス』へ「10.1.1.101」を入力し、キーボードの [Enter] を押します。



- 3 コントローラへの接続が確立すると、コントローラ画像の④黒丸が緑色に変化します。

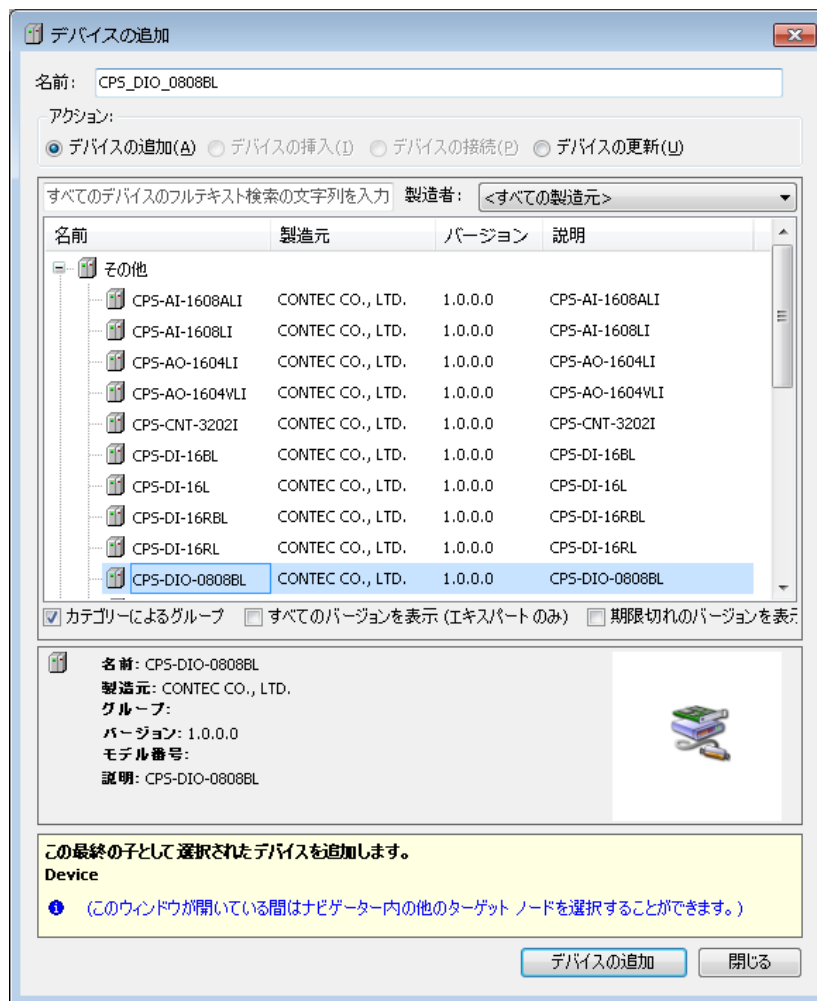


3. I/Oモジュールを追加する

スタック型コントローラには、自由な組み合わせでI/Oモジュールを追加することができます。

※ コントローラに搭載のI/O機能については、CODESYSデバイスウィンドウのコントローラのツリー階層内に登録されている(『PCS341XX I/O』のように表示される)ため以下の手順は不要です。

- 1 CODESYSデバイスウィンドウのコントローラを右クリックして「デバイスの追加」を選択します。
- 2 デバイスの追加ダイアログにて、追加するI/Oモジュールを選択して「デバイスの追加」ボタンをクリックします。I/Oモジュールは『その他』のツリー階層内に登録されています。



4. I/O変数の定義

CODESYSでI/O操作を行うためにI/Oモジュールの入出力チャンネルを変数定義するには、I/Oマッピングもしくはパラメーター内へ変数を入力します。

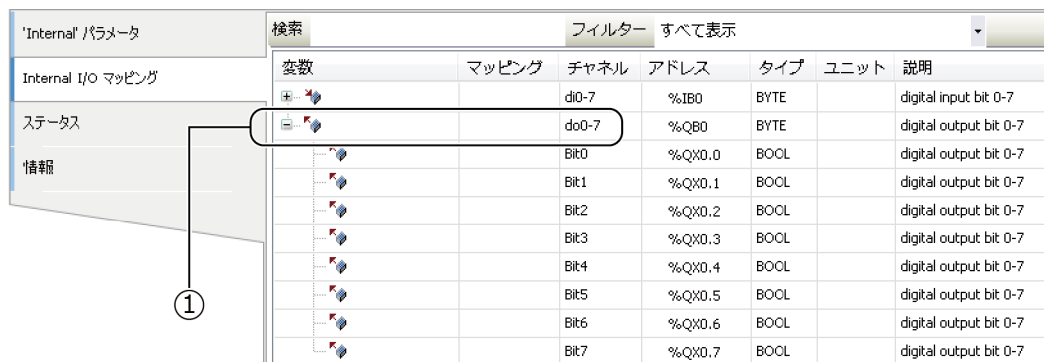
以下は、CPS-DIO-0808BLのDOポートのビット0に、変数「DO0」を割り当てるI/O変数の定義手順を紹介いたします。他のI/Oモジュールやコントローラに搭載のI/O定義もこの手順で行えます。

1 『I/Oモジュールを追加する(P58)』手順でCPS-DIO-0808BLを追加し、デバイスウィンドウのアイコンをダブルクリックします。

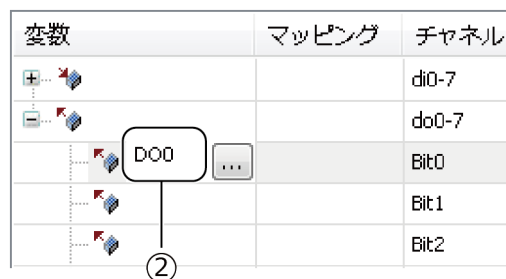
※コントローラに搭載のI/O機能は、CODESYSデバイスウィンドウのコントローラ(『PCS341XX I/O』のように表示される)のツリー階層内に登録されています。

2 コンフィギュレーションウィンドウが表示されますので、Internal I/O マッピングを選択します。

3 ① 『チャンネル』が『do0-7』の行のツリーを開きます。



4 ② 『チャンネル』が『Bit0』の行の『変数』列の空白部分をダブルクリックし、「DO0」と入力して変数を定義します。

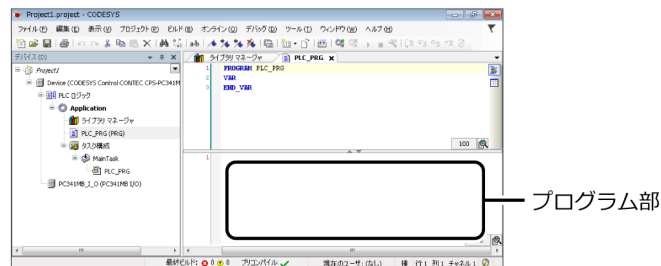


5. プログラム作成とビルド

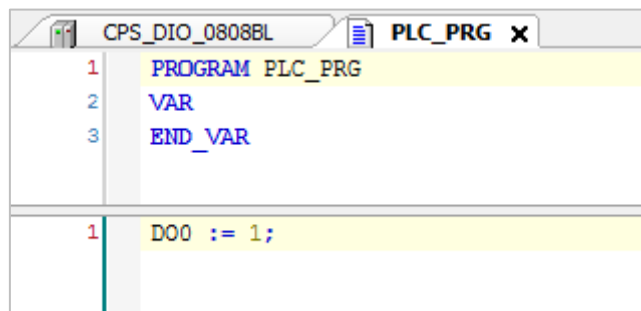
『**I/O変数の定義(P59)**』で定義されたI/O変数『DO0』を用いてPLCプログラムを作成し、デジタル出力ビット0からON出力を行います。プログラミングはST言語を使用します。

1 デバイスウィンドウ内の [PLC_PRG(PRG)] アイコンをダブルクリックすると、『STエディタウィンドウ』が表示されます。

2 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```
DO0 := 1;
```



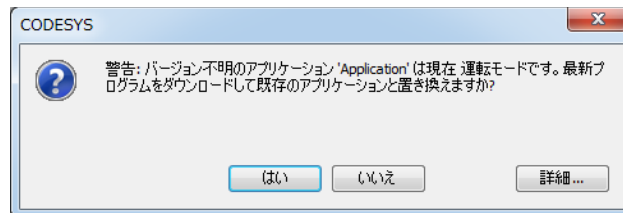
変数DO0はCPS-DIO-0808BLの出力ビット0に定義されているので、CPS-DIO-0808BLのデジタル出力ビット0からON信号を出力します。

3 CODESYSの [ビルド] メニューから [リビルド] を実行し、ビルドプロセスが正常に完了するかどうかを確認します。

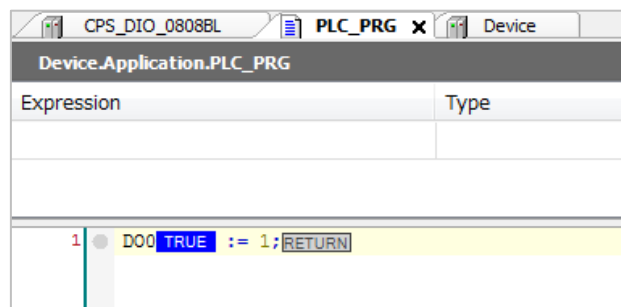
エラーになる場合、I/O定義が正しく設定できていないか、ファームウェアがバージョンアップされていない、またはライブラリマネージャに不明なライブラリが存在する可能性があります。

6. プログラムのダウンロードと実行

- 1 『プログラム作成とビルド(P60)』でビルドが完了したら、CODESYSの［オンライン］メニューから［ログイン］します。
- 2 ダウンロードの確認ダイアログが表示されますので［はい］をクリックします。



- 3 [デバッグ] メニューの [運転] を実行すると、プログラムの動作が開始します。
CPS-DIO-0808BLのフロントパネルあるDOビット0のLEDが点灯していることを確認できます。
また、STエディタウィンドウでは『DO0』変数の横に「TRUE」の現在値が表示されます。



7. PLCプログラムのROM保存

PLCプログラムはRAM領域にダウンロードするため、コントローラの電源をOFFにすると削除されます。これを回避するためには、プログラムをROM領域に保存します。

- 1 コントローラを接続したパソコンでWebブラウザを起動し、アドレス入力部に「http://10.1.1.101/」(PLCコントローラのIPアドレス)を入力します。
※ コントローラ接続の詳細は『**パソコンのネットワーク設定をする(P18)**』を参照ください。
※ 「https://10.1.1.101/」でもアクセス可能です。『**証明書エラー**』画面が表示される場合は、[このサイトの閲覧を続行する]を選択してください。
- 2 ユーザー名に「pc341」、パスワードに「pc341」と入力してログインします。
- 3 『CODESYS設定』の[PLCプログラム保存]を選択して[Save to ROM]をクリックします。書き込みが完了するまでの間、本体のST1とST2 LEDが点滅を繰り返します。

⚠ 注意

設定保存中は、本体LEDのST1とST2が点滅します。

その間、電源を切らないでください。(所要時間：5分程度)

設定保存を行わないで再起動や電源をOFFにすると設定した内容が元に戻ります。

8. ROM保存したPLCプログラムの削除

ROM領域に保存したPLCプログラムを削除するには、下記手順で操作を行ってください。

- 1 CODESYS開発環境でコントローラにログインします。
※ ログインについては『**CODESYSからコントローラに接続する(P57)**』を参照ください。
- 2 CODESYSメニューから[オンライン] - [リセット(PLC初期化)]を実行します。
- 3 『**PLCプログラムのROM保存(P62)**』と同様の手順で、PLCプログラムをROMへ保存します。
コントローラを再起動すると、PLCプログラムが削除されます。

4. 製品別インターフェイス定義

パラメーター、I/Oマッピング、シリアルポートの各種入出力インターフェイスについて、CODESYS上での定義方法について示します。

1. パラメーター

パラメーターは、コントローラの動作開始時に1度だけ設定される項目です。CODESYSでは [Internal Parameters] に表示されます。

製品型式	パラメーター名	CODESYSでの表記	意味
CPS-PC341EC-1-9201	なし	—	—
CPS-PC341MB-ADSC1-9201	デジタルフィルタ	di filter	デジタル入力フィルタの値(*1)を選択 デフォルト設定 : not used
CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	汎用入出力方向	GPIO0 - GPIO3	入出力方向を [Input] [Output] から 選択 デフォルト設定 : Input
CPS-DIO-0808L CPS-DIO-0808BL CPS-DIO-0808RL CPS-DI-16L CPS-DI-16RL	デジタルフィルタ	di filter	デジタル入力フィルタの値(*1)を選択 デフォルト設定 : not used
	内蔵電源	internal power	内蔵電源のON/OFFを選択 (CPS-DIO-0808BL) デフォルト設定 : ON
CPS-SSI-4P	結線方式	wire0-3	結線方式を [3-wire] [4-wire] から選 択 デフォルト設定 : 4-wire

*1 デジタル入力フィルタの値

入力フィルタ値	
not used	1.024 msec
0.25 µsec	2.048 msec
0.5 µsec	4.096 msec
1.0 µsec	8.192 msec
2.0 µsec	16.384 msec
4.0 µsec	32.768 msec
8.0 µsec	65.536 msec
16.0 µsec	131.072 msec
32.0 µsec	
64.0 µsec	
128.0 µsec	
256.0 µsec	
512.0 µsec	

2. I/Oマッピング

I/Oマッピングは、サイクルタイム毎に繰り返し実行される項目です。CODESYSでは[Internal I/O Mapping]に表示されます。

製品型式	I/Oマッピング名	CODESYSでの表記	意味
CPS-PC341EC-1-9201	バッテリーステータス	battery	バッテリーステータス 0=電池残量無し 1=電池残量有り
CPS-PC341MB-ADSC1-9201	デジタル出力	do0-1	デジタル出力ビット0 - 1
	デジタル入力	di0-3	デジタル入力ビット0 - 3
	カウンタ入力0	cnt0	カウンタ入力チャンネル0 範囲：0 - 16,777,215
	カウンタ入力1	cnt1	カウンタ入力チャンネル1 範囲：0 - 16,777,215
	アナログ入力0	ai0	アナログ入力チャンネル0 範囲：0 - 4,095
	アナログ入力1	ai1	アナログ入力チャンネル1 範囲：0 - 4,095
	カウンタクリア	cnt clear	カウンタ値クリアコマンド ビット0 = ONでカウンタ0のクリア ビット1 = ONでカウンタ1のクリア
	バッテリーステータス	battery	バッテリーステータス 0=電池残量無し 1=電池残量有り
CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	バッテリーステータス	battery	バッテリーステータス 0=電池残量無し 1=電池残量有り
	デジタル出力※	digital outputs (GPIO0..GPIO3)	デジタル出力ビット0 - 3
	デジタル入力※	digital inputs (GPIO0..GPIO3)	デジタル入力ビット0 - 3
CPS-DIO-0808L	デジタル出力	do0-7	デジタル出力ビット0 - 7
CPS-DIO-0808BL CPS-DIO-0808RL	デジタル入力	di0-7	デジタル入力ビット0 - 7
CPS-AI-1608LI CPS-AI-1608ALI	アナログ入力	ai0-7	アナログ入力チャンネル0 - 7 範囲：0 - 65,535
CPS-AO-1604LI CPS-AO-1604VLI	アナログ出力	ao0-3	アナログ出力チャンネル0 - 3 範囲：0 - 65,535
CPS-RRY-4PCC	デジタル出力	do0-3	デジタル出力ビット0 - 3
CPS-DI-16L CPS-DI-16RL	デジタル入力	di0 0-7 di1 0-7	デジタル入力0 ビット0 - 7 デジタル入力1 ビット0 - 7
CPS-DO-16L CPS-DO-16RL	デジタル出力	do0 0-7 do1 0-7	デジタル出力0 ビット0 - 7 デジタル出力1 ビット0 - 7
CPS-SSI-4P	温度入力	temp0-3	温度入力チャンネル0 - 3 範囲：-200 - 800、断線時-999
	SSIデータ入力(*1)	data0-3	SSIデータ入力チャンネル0 - 3

CPS-CNT-3202I	カウンタ入力	cnt val0-1	カウンタ入力チャンネル0 - 1
	ステータス入力	status0-1	ステータス入力チャンネル0 - 1
	制御フラグ出力	out ctrl0-1	制御フラグ出力チャンネル0 - 1 ビット0=OFFで停止、ONで開始 ビット1=OFFからONへの変化でカウンタ値セット
	制御フラグ入力	in ctrl0-1	制御フラグ入力チャンネル0 - 1
	イベント入力	event0-1	イベント入力チャンネル0 - 1 ビット0=ONでカウンタ比較値と一致
	イベントリセット出力	event reset0-1	イベントリセット出力チャンネル0 - 1 各ビットをOFFからONへ変化させることでイベントをリセット
	Z相モード出力	zmode0-1	Z相モード出力チャンネル0 - 1 0=未使用、1=次の1回、2=毎回
	Z相論理出力	zlogic0-1	Z相論理出力チャンネル0 - 1 0=正論理、1=負論理
	カウント方向出力	dir0-1	カウント方向出力チャンネル0 - 1 0=ダウncount、1=アップカウント
	相数出力(*2)	phase0-1	相数出力チャンネル0 - 1 0=単相、1=2相、2=ゲートコントロール
	逓倍出力(*2)	multi0-1	逓倍出力チャンネル0 - 1 0=1逓倍、1=2逓倍、2=4逓倍
	クリア出力(*2)	clr0-1	クリア出力チャンネル0 - 1 0=非同期クリア、1=同期クリア
	デジタルフィルタ出力(*3)	filter0-1	デジタルフィルタ出力チャンネル0 - 1
	ワンショットパルス幅(*4)出力	pulse0-1	ワンショットパルス幅出力チャンネル0 - 1
	カウントセット値出力	cnt set0-1	カウントセット値出力チャンネル0 - 1
	カウント比較値出力	cnt match0-1	カウント比較値出力チャンネル0 - 1

※ 汎用入出力(GPIO)では、入力/出力それぞれ4ビット分のI/Oマッピングが定義されていますが、実際に使用可能な入出力は、汎用入出力方向パラメーターによって決まります。

*1 SSIデータのフォーマット

	開始アドレス								開始アドレス+1								開始アドレス+2								開始アドレス+3												
ビット	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
	ステータス								符号	MSB LSB																											
	センサ異常	Reserved							有効データ	4096℃ ↓ 1℃ ↓ 1/1024℃ ↓																											
1℃	0	*	*	*	*	*	*	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0					
-1℃	0	*	*	*	*	*	*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0					
-1/1024℃	0	*	*	*	*	*	*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					
-999℃(断線時)	1	*	*	*	*	*	*	1	1	1	1	0	0	0	0	0	1	1	0	0	1	*	*	*	*	*	*	*	*	*	*	*					

*:規定値無し

*2 相数／逡倍／クリア

設定できるパルス信号の種類(動作モード)
2相入力、同期クリア、1 逡倍モード
2相入力、同期クリア、2 逡倍モード
2相入力、同期クリア、4 逡倍モード
2相入力、非同期クリア、1 逡倍モード
2相入力、非同期クリア、2 逡倍モード
2相入力、非同期クリア、4 逡倍モード
単相入力、非同期クリア、1 逡倍モード
ゲートコントロール付き単相入力、非同期クリア、1逡倍モード
ゲートコントロール付き単相入力、非同期クリア、2逡倍モード

*3 デジタルフィルタ

設定値	値
0h	0.1μsec
1h	6.52μsec
2h	25.7μsec
3h	32.1μsec
4h	204.9μsec
5h	211.3μsec
6h	230.5μsec
7h	236.9μsec
8h	819.3μsec
9h	825.7μsec
Ah	844.9μsec
Bh	851.3μsec
Ch	1024.1μsec
Dh	1030.5μsec
Eh	1049.7μsec
Fh	1056.1μsec

*4 ワンショットパルス幅

パルス幅設定データ×409.6=パルス幅(μsec)

3. シリアルポート

製品型式	製品表記	CODESYS上のポート番号
CPS-PC341MB-ADSC1-9201	COM A	ポート1
	COM B	ポート2
CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	COM A	ポート1
CPS-COM-1PC CPS-COM-1PD CPS-COM-2PC CPS-COM-2PD	1チャンネルタイプの場合COM A 2チャンネルタイプの場合COMA、COM B	シリアル通信モジュールのポート番号は、コントローラに近い方から順番にポート2、ポート3・・・と割り当てられます。 (ポート1はコントローラ本体COM用)

4. OPC UAシンボル設定

CODESYSのIECプログラム内で使われる変数をシンボル登録することで、OPC UA通信を行ったり、CONPROSYS HMIでのモニタリングを行ったりすることができます。

シンボルの設定はCODESYSから自由に行うことができますが、工場出荷時設定として下記のシンボルが設定されており、電源投入後に自動でOPC UAサーバーとして動作します。

製品型式	I/O名	OPC UAシンボル名	アクセス、データ型、範囲
CPS-PC341EC-1-9201	なし	—	—
CPS-PC341MB-ADSC1-9201	デジタル出力ビット0	DO0	Read/Write、BIT、0 or 1
	デジタル出力ビット1	DO1	
	デジタル入力ビット0	DI0	Read、BIT、0 or 1
	デジタル入力ビット1	DI1	
	デジタル入力ビット2	DI2	
	デジタル入力ビット3	DI3	
	アナログ入力0	AI0	Read、DWORD、0 - 4095
	アナログ入力1	AI1	
CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	カウンタ入力0	CNT0	Read、DWORD、0 - 16777215
	カウンタ入力1	CNT1	
	カウンタクリア0	CNTCLR0	Read/Write、BIT、0 or 1
	カウンタクリア1	CNTCLR1	
	デジタル入力ビット0	DI0	Read、BIT、0 or 1
	デジタル入力ビット1	DI1	
	デジタル入力ビット2	DI2	
	デジタル入力ビット3	DI3	

通信方法の設定をする

PACシリーズの通信方法および設定について説明しています。

下表は本章の各機能に対応・接続できる製品一覧です。

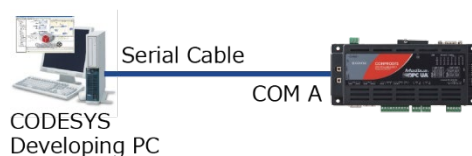
接続設定	対応する製品	Slaveとして接続できる製品
コントローラとPCでシリアル通信する	CPS-PC341MB-ADSC1-9201 CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	—
本製品をEtherCAT Masterとして使用する	CPS-PC341EC-1-9201 CPS-PCS341EC-DS1-1201	CPS-ECS341-1-011 他社メーカーのSlave対応デバイス
本製品をModbus TCP Masterとして使用する	CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	CPS-MC341-ADSC1-111などの 当社対応製品 他社メーカーのSlave対応デバイス
本製品をModbus TCP Slaveとして使用する	CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	—
本製品をModbus RTU Masterとして使用する	CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	CPSN-MCB271-S1-041などの当 社対応製品 他社メーカーのSlave対応デバイス
本製品をModbus RTU Slaveとして使用する	CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	—

1. コントローラとPCでシリアル通信する

対応する製品	Slaveとして接続できる製品
CPS-PC341MB-ADSC1-9201 CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	—

1. シリアル通信の準備

コントローラとPCをシリアルケーブルで接続し、コントローラからPCへのデータ送信、PCからコントローラへのデータ受信を行います。COMポートには、コントローラ本体のCOMポートAを使用します。

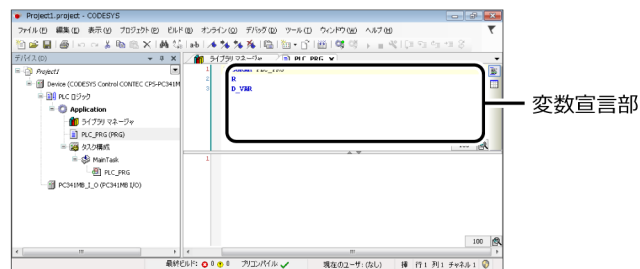


2. シリアル通信ライブラリの設定

- 1 新規CODESYSプロジェクトを作成します。
IECプログラム言語はSTを選択します。
※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。
- 2 デバイスウィンドウの[ライブラリマネージャ]をダブルクリックします。
- 3 [Add Library]をクリックし、『Use Cases』のツリー階層内にある[Serial Communication]を選択して[OK]をクリックします。

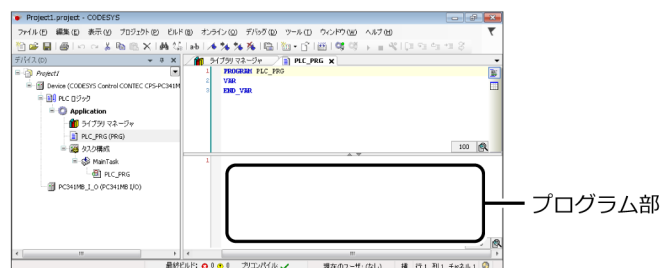
3. 送信プログラムの作成と実行

- 1 デバイスウィンドウ内の [PLC_PRG (PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に以下のソースコードを記述します。



```
StartFlag : BOOL := TRUE;  
State : INT := 0;  
Com1Params : ARRAY [1..7] OF COM.PARAMETER;  
Com1Open : COM.Open;  
Com1Close : COM.Close;  
Com1Write : COM.Write;  
Error : BOOL;  
TestString : STRING := 'Test String!';
```

- 3 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```
IF StartFlag THEN
CASE State OF
0:
  Com1Params[1].udiParameterId := COM.CAA_Parameter_Constants.udiPort;
  Com1Params[1].udiValue := 3;
  Com1Params[2].udiParameterId := COM.CAA_Parameter_Constants.udiBaudrate;
  Com1Params[2].udiValue := 9600;
  Com1Params[3].udiParameterId := COM.CAA_Parameter_Constants.udiParity;
  Com1Params[3].udiValue := INT_TO_UDINT(COM.PARITY.NONE);
  Com1Params[4].udiParameterId := COM.CAA_Parameter_Constants.udiStopBits;
  Com1Params[4].udiValue := INT_TO_UDINT(COM.STOPBIT.ONESTOPBIT);
  Com1Params[5].udiParameterId := COM.CAA_Parameter_Constants.udiTimeout;
  Com1Params[5].udiValue := 0;
  Com1Params[6].udiParameterId := COM.CAA_Parameter_Constants.udiByteSize;
  Com1Params[6].udiValue := 8;
  Com1Params[7].udiParameterId := COM.CAA_Parameter_Constants.udiBinary;
  Com1Params[7].udiValue := 0;
  Com1Open(xExecute:= TRUE, usiListLength:= SIZEOF(Com1Params) /
    SIZEOF(COM.PARAMETER), pParameterList:= ADR(Com1Params));
  IF Com1Open.xError THEN
    Error := TRUE;
    State := 1000;
  END_IF
  IF Com1Open.xDone THEN
    State := 1;
  END_IF
1:
  Com1Write(xExecute := TRUE, hCom:= Com1Open.hCom, pBuffer:=
    ADR(TestString), szSize:= SIZEOF(TestString));
  IF Com1Write.xError THEN
    Error := TRUE;
    State := 1000;
  END_IF
  IF Com1Write.xDone THEN
    State := 2;
  END_IF
2:
  Com1Close(xExecute := TRUE, hCom:= Com1Open.hCom);
  IF Com1Close.xError THEN
    Error := TRUE;
  END_IF
  IF Com1Close.xDone OR Com1Close.xError THEN
    State := 1000;
  END_IF
1000:
  StartFlag := FALSE;
END_CASE
END_IF
```

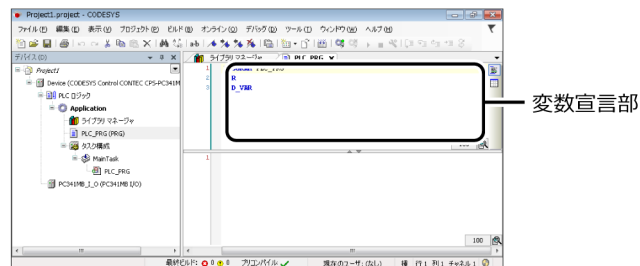
- 4** CODESYSの「ビルド」メニューから「リビルド」を実行し、ビルドプロセスが正常に完了するかどうかを確認します。

◆ 文字列送信の実行

- 1 CODESYSの [オンライン] メニューから [ログイン] を実行します。
- 2 ダウンロードの確認ダイアログが表示されますので [はい] をクリックします。
- 3 PC側でシリアルポートへのターミナル接続を行い、受信待ちの状態にしておきます。
- 4 CODESYSにて [デバッグ] メニューの [運転] を実行すると、プログラムの動作が開始します。PC側のターミナル画面に『Test String!』の文字列が表示されます。

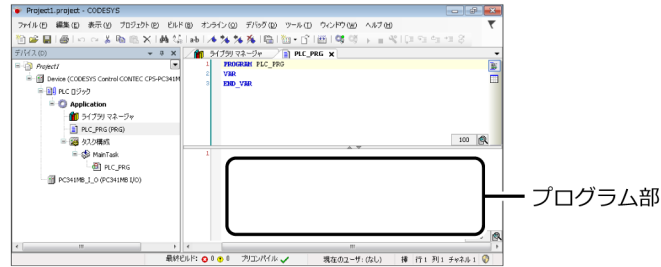
4. 受信プログラムの作成と実行

- 1 受信プログラム用に新規CODESYSプロジェクトを作成します。
IECプログラム言語はSTを選択します。
※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。
- 2 デバイスウィンドウ内の [PLC_PRG (PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 3 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に以下のソースコードを記述します。



```
StartFlag : BOOL := TRUE;  
State : INT := 0;  
Com1Params : ARRAY [1..7] OF COM.PARAMETER;  
Com1Open : COM.Open;  
Com1Close : COM.Close;  
Com1Read : COM.Read;  
Error : BOOL;  
TestString : STRING(25);
```

4 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```

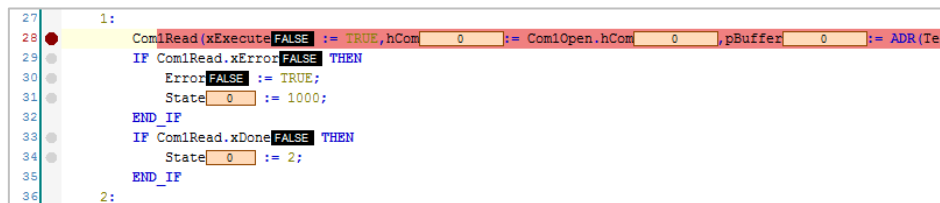
IF StartFlag THEN
CASE State OF
0:
    Com1Params[1].udiParameterId := COM.CAA_Parameter_Constants.udiPort;
    Com1Params[1].udiValue := 1;
    Com1Params[2].udiParameterId := COM.CAA_Parameter_Constants.udiBaudrate;
    Com1Params[2].udiValue := 9600;
    Com1Params[3].udiParameterId := COM.CAA_Parameter_Constants.udiParity;
    Com1Params[3].udiValue := INT_TO_UDINT(COM.PARITY.NONE);
    Com1Params[4].udiParameterId := COM.CAA_Parameter_Constants.udiStopBits;
    Com1Params[4].udiValue := INT_TO_UDINT(COM.STOPBIT.ONESTOPBIT);
    Com1Params[5].udiParameterId := COM.CAA_Parameter_Constants.udiTimeout;
    Com1Params[5].udiValue := 0;
    Com1Params[6].udiParameterId := COM.CAA_Parameter_Constants.udiByteSize;
    Com1Params[6].udiValue := 8;
    Com1Params[7].udiParameterId := COM.CAA_Parameter_Constants.udiBinary;
    Com1Params[7].udiValue := 0;
    Com1Open(xExecute := TRUE, usiListLength := SIZEOF(Com1Params) /
        SIZEOF(COM.PARAMETER), pParameterList := ADR(Com1Params));
    IF Com1Open.xError THEN
        Error := TRUE;
        State := 1000;
    END_IF
    IF Com1Open.xDone THEN
        State := 1;
    END_IF
1:
    Com1Read(xExecute := TRUE, hCom := Com1Open.hCom, pBuffer :=
        ADR(TestString), szBuffer := SIZEOF(TestString));
    IF Com1Read.xError THEN
        Error := TRUE;
        State := 1000;
    END_IF
    IF Com1Read.xDone THEN
        State := 2;
    END_IF
2:
    Com1Close(xExecute := TRUE, hCom := Com1Open.hCom);
    IF Com1Close.xError THEN
        Error := TRUE;
    END_IF
    IF Com1Close.xDone OR Com1Close.xError THEN
        State := 1000;
    END_IF

```

```
1000:  
    StartFlag := FALSE;  
END_CASE  
END_IF
```

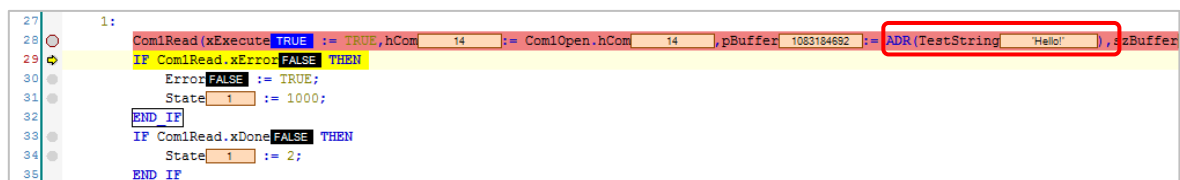
◆ 文字列送信の実行

- 1 CODESYSの［オンライン］メニューから［ログイン］を実行します。
- 2 ダウンロードの確認ダイアログが表示されますので［はい］をクリックします。
- 3 PC側でシリアルポートへのターミナル接続を行い、送信待ちの状態にしておきます。
- 4 プログラムソースコードの受信関数部分にブレークポイントを挿入します。Com1Read関数をコールする行で［デバッグ］ - ［新規ブレークポイント］を選択します。



```
27 1:  
28 Com1Read(xExecute:=FALSE, hCom:=0, pCom:=Com1Open.hCom, pBuffer:=0, i:=ADR(Te  
29 IF Com1Read.xError:=FALSE THEN  
30     Error:=FALSE; := TRUE;  
31     State:=0 := 1000;  
32 END_IF  
33 IF Com1Read.xDone:=FALSE THEN  
34     State:=0 := 2;  
35 END_IF  
36 2:
```

- 5 [デバッグ] メニューの [運転] を実行して、プログラムの動作を開始します。プログラムはブレークポイントの位置で一時停止します。
- 6 PCのターミナルから、送信文字列を入力します。
- 7 CODESYS側でF10キーを押して、プログラムを1ステップ実行します。TestString変数にPCのターミナルから送信した文字列が格納されていることが確認できます。



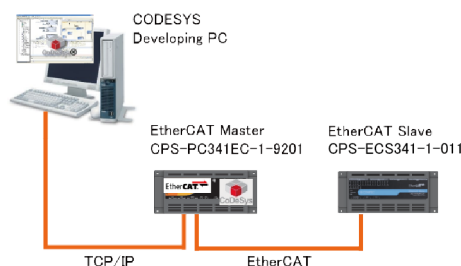
```
27 1:  
28 Com1Read(xExecute:=TRUE, hCom:=14, pCom:=Com1Open.hCom, pBuffer:=1083184692, i:=ADR(TestString), zBuffer  
29 IF Com1Read.xError:=FALSE THEN  
30     Error:=FALSE; := TRUE;  
31     State:=1 := 1000;  
32 END_IF  
33 IF Com1Read.xDone:=FALSE THEN  
34     State:=1 := 2;  
35 END_IF
```

2. EtherCAT Masterとして使用する

対応する製品	Slaveとして接続できる製品
CPS-PC341EC-1-9201 CPS-PCS341EC-DS1-1201	CPS-ECS341-1-011 他社メーカーのSlave対応デバイス

1. デバイスの準備

EtherCATスレーブデバイスとして当社製CPS-ECS341-1-011を使用した、CODESYSプログラミング例を解説します。



- 1 CPS-PC341EC-1-9201のLAN AとパソコンのLANポートをイーサネットケーブルで接続します。
- 2 CPS-PC341EC-1-9201のLAN BとCPS-ECS341-1-011のINポートをイーサネットケーブルで接続します。

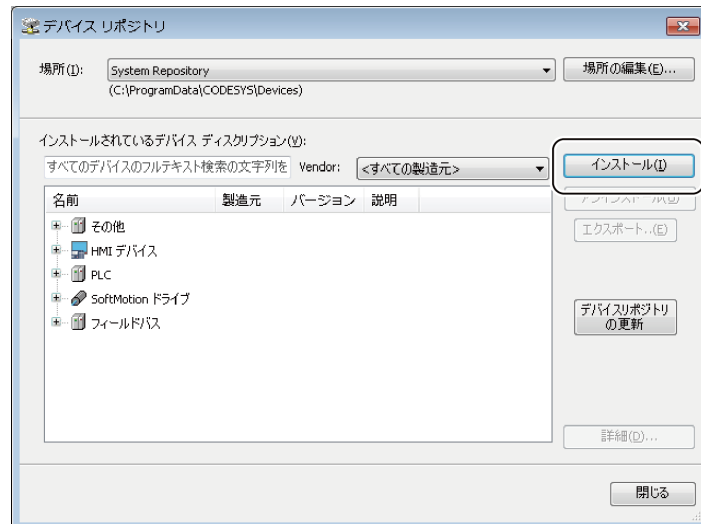
2. スレーブデバイスの追加

EtherCATスレーブコントローラを使用するには、ESI(EtherCAT Slave Information)ファイルの登録が必要です。

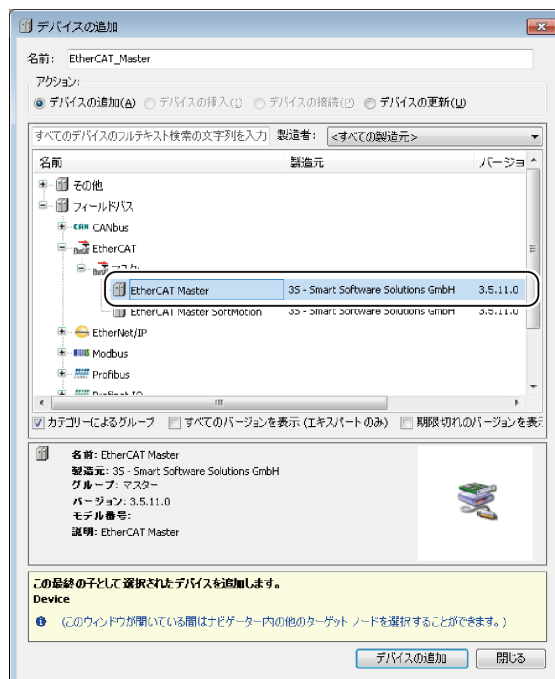
当社製スレーブコントローラを使用する場合は、ESIファイルはCODESYSパッケージに含まれているため、1～3の登録は不要です。他社製のスレーブデバイスを使用する場合は1～3の登録手順を行ってください。

- 1 新規CODESYSプロジェクトを作成します。
デバイスは接続しているEtherCATモデルの名称を、IECプログラム言語はSTを選択します。
※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。
- 2 ESIファイルを使用するデバイスメーカーからダウンロードします。
- 3 CODESYSの[ツール]メニューから[デバイスリポジトリ]を選択します。

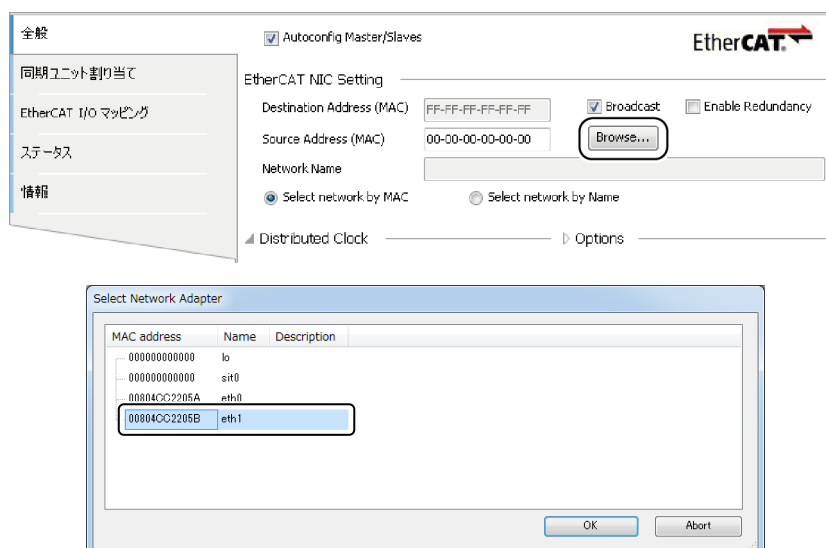
- 4 「インストール」をクリックして『CONTEC_CPS-ECS341-1-011_v100』のESIファイル(他社製のデバイスの場合は入手したESIファイル)を指定してください。



- 5 CODESYSデバイスウィンドウ内の [Device(CODESYS Control CONTEC CPS-PC341EC-1-9201)] アイコンを右クリックして、[デバイスの追加] を選択します。
- 6 デバイスの追加ダイアログにて、[EtherCAT Master] を選択し [デバイスの追加] ボタンをクリックします。[EtherCAT Master] は、[フィールドバス - EtherCAT - マスター] のツリー階層内に登録されています。



- 7 デバイスウィンドウに追加された [EtherCAT_Master] アイコンを右クリックして [デバイスの追加] を選択します。
- 8 デバイスの追加ダイアログで [CPS-ECS341-1-011] または使用するデバイス名を選択して [デバイスの追加] ボタンをクリックします。
『CPS-ECS341-1-011』は『フィールドバス - EtherCAT - スレーブ - CONTEC Co., Ltd. - CONPROSYS EtherCAT』のツリー階層内に登録されています。
- 9 デバイスウィンドウの [EtherCAT_Master (EtherCAT Master)] アイコンをダブルクリックして設定ウィンドウを開きます。
- 10 [全般] タブにて [Browse] ボタンをクリックし、[eth1] を選択します。



3. スレーブデバイスの設定

CPS-DIO-0808Lの出力ポート2ビット0を連続でON/OFFさせるための、EtherCATスレーブデバイスの設定について記述します。

- 1 デバイスウィンドウから [CPS-DIO-0808L] アイコンをダブルクリックします。
- 2 CPS-DIO-0808Lのコンフィギュレーションウィンドウが表示されたら、[EtherCAT I/O Mapping] を選択します。

3 ①『チャネル』列に「D00-7」と書かれた行のツリーを開きます。

General	Channels
Process Data	Variable Mapping Channel Address Type Unit Description
EtherCAT I/O Mapping	D00-7 %QB0 BYTE D00-7
Status	Bit0 %QX0.0 BOOL D00-7
Information	Bit1 %QX0.1 BOOL D00-7
	Bit2 %QX0.2 BOOL D00-7
	Bit3 %QX0.3 BOOL D00-7
	Bit4 %QX0.4 BOOL D00-7
	Bit5 %QX0.5 BOOL D00-7
	Bit6 %QX0.6 BOOL D00-7
	Bit7 %QX0.7 BOOL D00-7
	D08-15 %QB1 BYTE D08-15
	Set the digital filt... %QB2 BYTE Set the digital fi
	Set the DO Fault %QBR3 BYTE Set the DO Fault

4 『チャネル』列が「Bit0」となっている②『変数』列の空白部分をダブルクリックし、『DO0』変数を定義します。

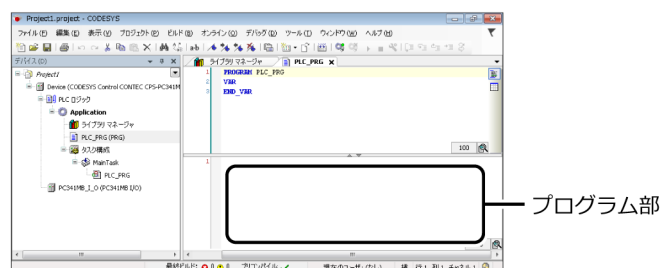
Variable	Mapping	Channel
		D00-7
DO0		Bit0
		Bit1
		Bit2

4. プログラムの作成とビルド

手順1、2は、CPS-DIO-0808Lの出力ポート2ビット0を連続でON/OFFさせるためのプログラム作成、手順3、4は、LEDの点滅を目視で確認するためのサイクルタイム設定です。

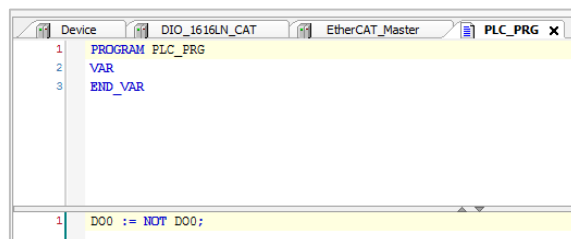
1 デバイスウィンドウ内の [PLC_PRG (PRG)] アイコンをダブルクリックします。 『STエディタウィンドウ』が表示されます。

2 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



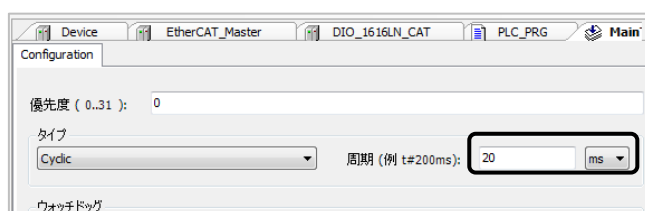
```
DO0 := NOT DO0;
```

上記のプログラムはサイクル毎に『DO0』変数へ「1」と「0」を繰り返しセットします。



3 デバイスウィンドウ内の [MainTask] アイコンをダブルクリックします。

4 『周期』を「4000μs」から「20ms」に変更します。



5 CODESYSの [ビルド] メニューから [リビルド] を実行し、ビルドプロセスが正常に完了するかどうかを確認します。

5. プログラムのダウンロードと実行

1 CODESYSの [オンライン] メニューから [ログイン] を実行します。

2 ダウンロードの確認ダイアログが表示されますので [はい] をクリックします。



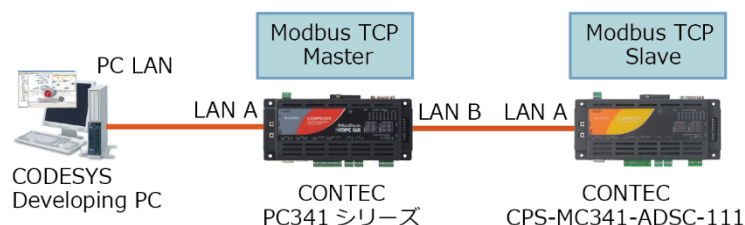
3 [デバッグ] メニューの [運転] を実行すると、EtherCAT動作が開始します。
CPS-DIO-0808LのBit0 LEDが点滅することが確認できます。

3. Modbus TCP Masterとして使用する

対応する製品	Slaveとして接続できる機器
CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	CPS-MC341-ADSC1-111などの当社 対応製品 他社メーカーのSlave対応デバイス

1. デバイスの準備

コントローラをModbus TCP Master機器として動作させ、Modbus TCP Slaveとのデータ入出力を行います。ここではModbus TCP MasterとしてCPS-PC341MB-ADSC1-9201、Modbus TCP SlaveとしてCONTEC CPS-MC341-ADSC1-111を使用します。

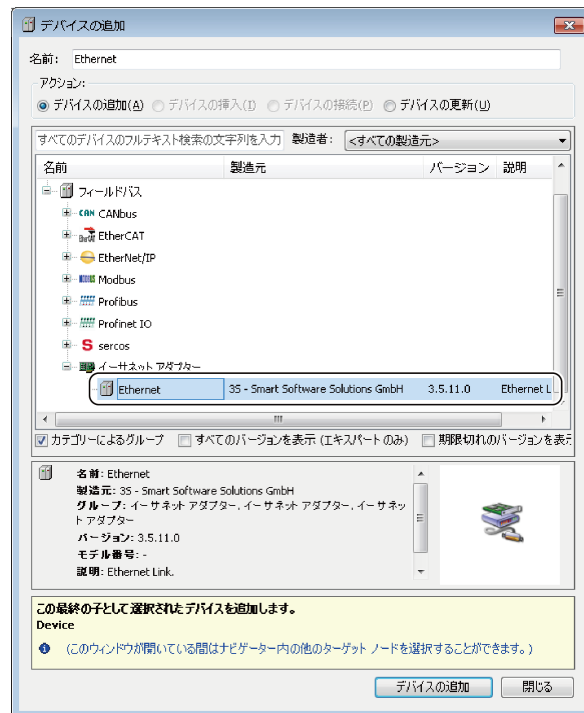


- 1 コントローラのLAN AをCODESYS開発環境のあるパソコンのLANポートと接続します。
- 2 コントローラのLAN BをCONTEC CPS-MC341-ADSC1-111へ接続します。

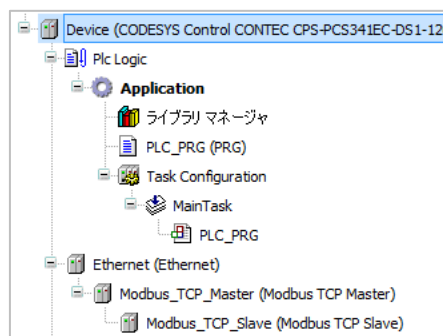
2. スレーブデバイスの追加

- 1 新規CODESYSプロジェクトを作成します。
デバイスは接続しているModbusモデルの名称を、IECプログラム言語はSTを選択します。
※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。
- 2 デバイスウィンドウのコントローラ [Device (CODESYS Control CONTEC CPS-PCXXXXXXXXXX)] を右クリックして [デバイスの追加] を選択します。

- 3** デバイスの追加ダイアログにて、[フィールドバス - イーサネット アダプタ - Ethernet] を選択して [デバイスの追加] ボタンをクリックします。

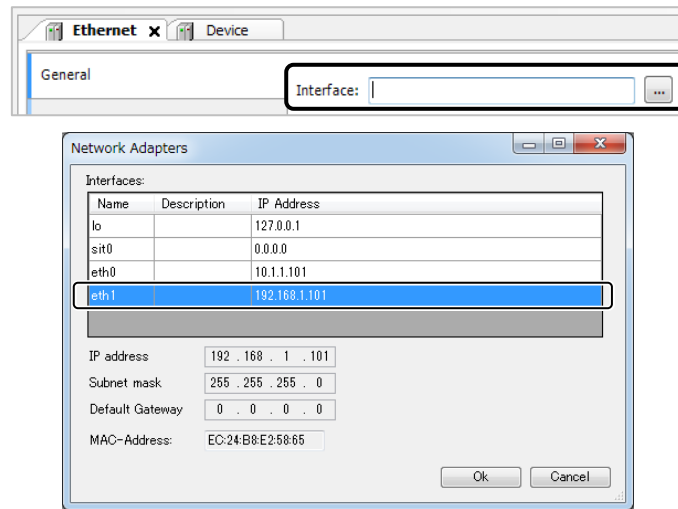


- 4** デバイスウィンドウの [Ethernet] を右クリックして [デバイスの追加] を選択します。
- 5** デバイスの追加ダイアログにて、[フィールドバス - Modbus - Modbus TCP マスター - Modbus TCP Master] を追加します。
- 6** 再度デバイスウィンドウの Modbus TCP Master を右クリックして [デバイスの追加] を選択し、[フィールドバス - Modbus - Modbus TCP スレーブ - Modbus TCP Slave] を追加します。

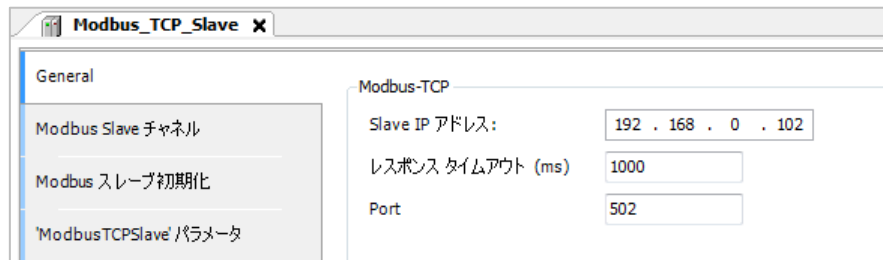


- 7** デバイスウィンドウから、[Ethernet] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。
- 8** [全般] タブを開きます。

- 9** [インターフェイ] テキストボックス横のボタンをクリックし、リストから [eth1] を選択します。
eth1はLAN Bの内部名になります。



- 10** デバイスウィンドウから、[Modbus_TCP_Slave] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。
- 11** [General] タブを開きます。
- 12** 『Slave IPアドレス』 ヘスレーブデバイスで設定されたIPアドレスを入力します。



3. Modbus TCP Masterの設定

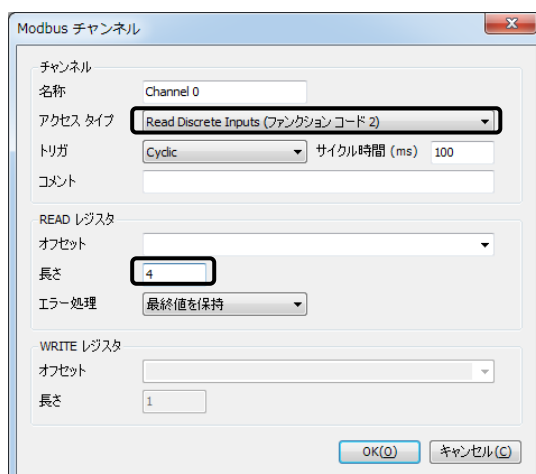
ここでは、2ビットのデジタル出力、4ビットのデジタル入力を行うサンプルを作成します。

使用するスレーブデバイスCPS-MC341-ADSC1-111のファンクションコードは下記のように定義されています。

名称/ファンクションコード	アドレス	データ
Read Coils(コード1)	0	デジタル出力ビット0
	1	デジタル出力ビット1
Read Discrete Inputs(コード2)	0	デジタル入力ビット0
	1	デジタル入力ビット1
	2	デジタル入力ビット2
	3	デジタル入力ビット3
Read Input Registers(コード4)	0	アナログ入力チャンネル0(16ビット)
	1	アナログ入力チャンネル1(16ビット)
	2	カウンタ入力チャンネル0上位16ビット
	3	カウンタ入力チャンネル0下位16ビット
	4	カウンタ入力チャンネル1上位16ビット
	5	カウンタ入力チャンネル1下位16ビット
Write Single Coil(コード5)	0	デジタル出力ビット0
	1	デジタル出力ビット1

※ サンプルとして作成する2ビットのデジタル出力は、Write Single Coil(コード5)、4ビットのデジタル入力はRead Discrete Inputs(コード2)を使用します。

- 1 デバイスウィンドウから、[Modbus_TCP_Slave] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。
- 2 [Modbus Slaveチャンネル] タブを開きます。
- 3 [チャンネルの追加] ボタンをクリックし、アクセスタイプから [Read Discrete Inputs(ファンクションコード2)] を選択、Readレジスタ長さには「4」と入力して [OK] をクリックします。



4 [チャンネルの追加] ボタンをクリックし、アクセスタイプから [Write Single Coil(ファンクションコード5)] を選択、Writeレジスタオフセットには「0」と入力して [OK] をクリックします。

5 [チャンネルの追加] ボタンをクリックし、アクセスタイプから再度 [Write Single Coil(ファンクションコード5)] を選択、Writeレジスタオフセットには「1」と入力して [OK] をクリックします。

名称	アクセス タイプ	トリガ	読み取りオフセ...	長さ	エラー処理	書き
0 Channel 0	Read Discrete Inputs (ファンクションコード 02)	サイクリック, t#100ms	16#0000	4	最終値を保持	
1 Channel 1	Write Single Coil (ファンクションコード 05)	サイクリック, t#100ms				16#
2 Channel 2	Write Single Coil (ファンクションコード 05)	サイクリック, t#100ms				16#

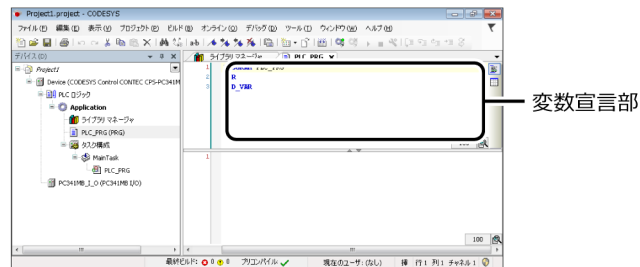
6 [ModbusTCPSlave I/O マッピング] をクリックします。

7 Channel0[0]、Channel1[0]、Channel2[0]にそれぞれ「DI0」、「DOBIT0」、「DOBIT1」を割り当てます。

変数	マッピング	チャンネル	アドレス	タイプ	ユニット	説明
DI0		Channel 0	%IB20	ARRAY [0..0] OF BYTE		Read Discrete Inputs
DOBIT0		Channel 0[0]	%IB20	BYTE		Read Discrete Inputs
DOBIT1		Channel 1	%QB20	ARRAY [0..0] OF BYTE		Write Single Coil
		Channel 1[0]	%QB20	BYTE		Write Single Coil
		Channel 2	%QB21	ARRAY [0..0] OF BYTE		Write Single Coil
		Channel 2[0]	%QB21	BYTE		Write Single Coil

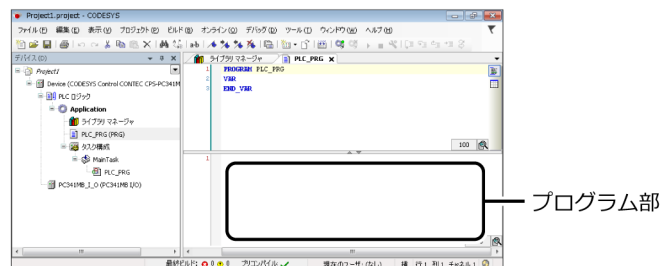
4. プログラムの作成とビルド

- 1 デバイスウィンドウ内の [PLC_PRG (PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に以下のソースコードを記述します。



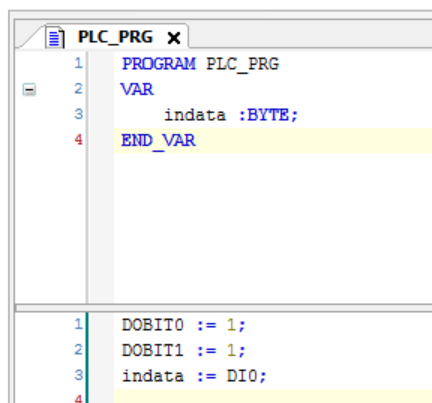
```
indata :BYTE;
```

- 3 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```
DOBIT0 := 1;  
DOBIT1 := 1;  
indata := DI0;
```

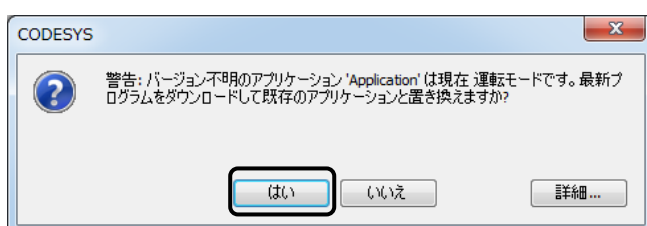
上記のプログラムは、CPS-PC341MB-ADSC1-9201のデジタル出力ビット0、1へON信号を出力し、デジタル入力ポートから信号を入力します。



- CODESYSの [ビルド] メニューから [リビルド] を実行し、ビルドプロセスが正常に完了するか確認します。

5. プログラムのダウンロードと実行

- CODESYSの [オンライン] メニューから [ログイン] を実行します。
- ダウンロードの確認ダイアログが表示されますので [はい] をクリックします。



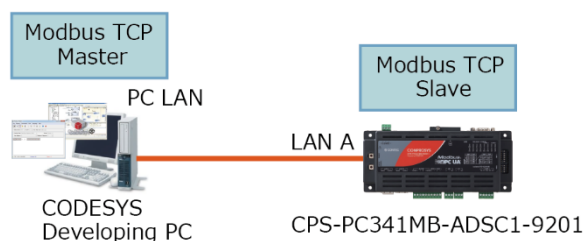
- [デバッグ] メニューの [運転] を実行すると、プログラムの動作が開始します。CPS-PC341MB-ADSC1-9201のフロントパネルにて、DOビット0、1のLEDが点灯することが確認できます。

4. Modbus TCP Slaveとして使用する

対応する製品	接続する機器
CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	—

1. デバイスの準備

コントローラをModbus TCP Slave機器として動作させ、Modbus TCP Masterからのデータ入出力に対応します。ここではModbus TCP MasterとしてCODESYS開発環境がインストールされているパソコンを、Modbus TCP Slaveとして、CPS-PC341MB-ADSC1-9201を使用した、CODESYSプログラミング例を解説します。Modbus TCP Masterソフトウェアとして、QModMasterを使用します。



- 1 Modbus TCP MasterソフトウェアQModMasterをパソコンにインストールします。
QModMasterはフリーソフトで、下記よりダウンロードできます。

ダウンロード <http://sourceforge.net/projects/qmodmaster/>

- 2 CPS-PC341MB-ADSC1-9201のLAN AとCODESYS開発環境のあるパソコンのLANポートをイーサネットケーブルで接続します。

2. スレーブ設定をする

1 新規CODESYSプロジェクトを作成します。

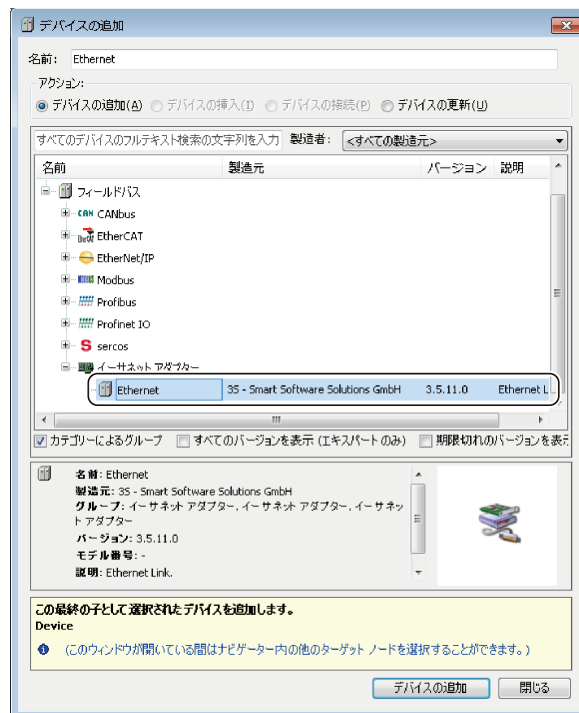
IECプログラム言語はSTを選択します。

※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。

2 デバイスウィンドウのコントローラ [Device (CODESYS Control CONTEC CPS-PCXXXXXXXXXX)] を右クリックして [デバイスの追加] を選択します。

3 デバイスの追加ダイアログにて、[フィールドバス - イーサネットアダプター - Ethernet] を選択して [デバイスの追加] ボタンをクリックします。

※ Ethernet Adapterが追加されます。



4 デバイスウィンドウの [Ethernet] を右クリックして [デバイスの追加] を選択します。

5 デバイスの追加ダイアログにて、[フィールドバス - Modbus - ModbusTCP スレーブデバイス - ModbusTCP Slave Device] を追加します。

※ Ethernet Adapterデバイスの下にModbus TCP Slaveデバイスが追加されます。

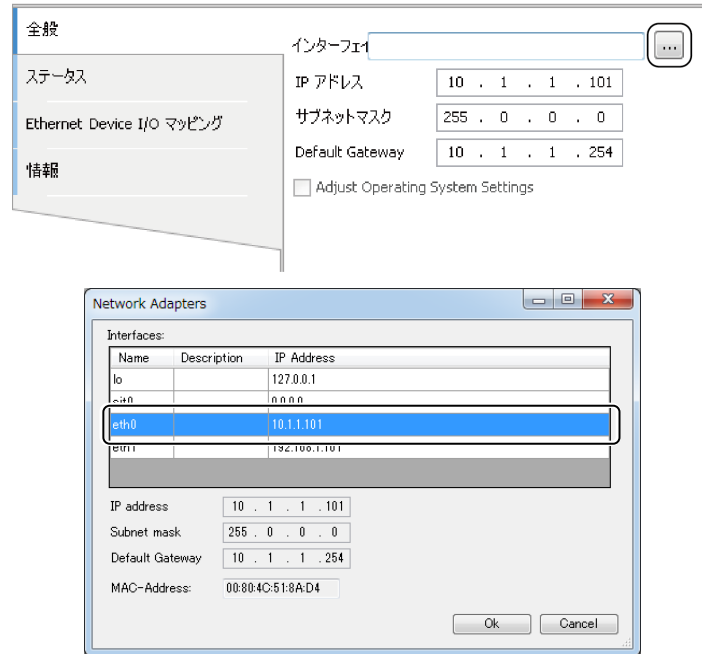
6 デバイスウィンドウから、[Ethernet] アイコンをクリックしてデバイスコンフィギュレーションウィンドウを開きます。

7 [全般] タブを開きます。

8 【インターフェイ】テキストボックス横のボタンをクリックし、リストから【eth0】を選択します。

※ Ethernet Adapterに対してIPアドレスが設定されます。

※ eth0はLAN Aの内部名です。



9 デバイスウィンドウから【PC341MB_I_O】アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。

10 【Internal I/O マッピング】をクリックします。

11 以下のとおりに変数を定義します。

変数(入力値)	チャネル	アドレス
	do0-1	
DO0	Bit0	%QX20.0
DO1	Bit1	%QX20.1
	Di0-3	
DI0	Bit0	%IX20.0
DI1	Bit1	%IX20.1
DI2	Bit2	%IX20.2
DI3	Bit3	%IX20.3
CNT0	cnt0	%ID6
CNT1	cnt1	%ID7
AI0	ai0	%ID8
AI1	ai1	%ID9

12 デバイスウィンドウから【ModbusTCP_Slave_Device】アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。

13 [Modbus TCP Slave Device I/O マッピング] をクリックします。

14 以下のとおりに変数を定義します。

変数(入力値)	チャネル	アドレス
	Input	%IW1
	Input[0]	%IW1
mDO0	Bit0	%IX2.0
mDO1	Bit1	%IX2.1
	:	
	Bit15	%IX3.7
	Input[1]	%IW2
	:	
	Input[9]	%IW10
	Output	%QW1
	Output[0]	%QW1
mDI0	Bit0	%QX0.0
mDI1	Bit1	%QX0.1
mDI2	Bit2	%QX0.2
mDI3	Bit3	%QX0.3
	:	
	Bit15	%QX1.7
mAI0	Output[1]	%QW2
MAI1	Output[2]	%QW3
mCNT0_L	Output[3]	%QW4
mCNT0_H	Output[4]	%QW5
mCNT1_L	Output[5]	%QW6
mCNT1_H	Output[6]	%QW7
	:	
	Output[9]	%QW10

※ I/Oマッピングの設定、Modbus Slaveに対してI/Oマッピングの定義がされました。Modbus SlaveのI/Oマッピングに変数を定義することで、Modbus Masterとのデータ通信が可能となります。

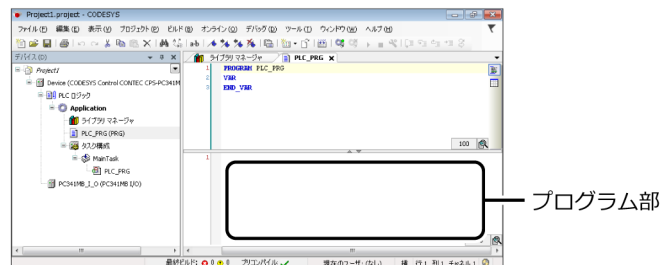
※ Inputsマッピング領域に出力用変数、Outputsマッピング領域に入力用の変数が定義されていますが、これはInputsマッピング領域がMasterからSlaveへの入力、Outputsマッピング領域がSlaveからMasterへの出力という意味となっているためです。

※ カウンタデータは24ビットでModbusのWORD領域に収まらないため、下位16ビットをmCNT0_Lへ、上位8ビットをmCNT0_Hへ格納します。

変数	マップ...	チャネル	アドレス	タイプ	ユニット	説明
		入力	%IW0	ARRAY [0..9] OF WORD		Modbus 保持レジスタ
mDO0		入力[0]	%IW0	WORD		
mDO1		BR0	%IX0.0	BOOL		
		BR1	%IX0.1	BOOL		
		BR2	%IX0.2	BOOL		
		BR3	%IX0.3	BOOL		
		BR4	%IX0.4	BOOL		
		BR5	%IX0.5	BOOL		
		BR14	%IX1.6	BOOL		
		BR15	%IX1.7	BOOL		
		出力[1]	%IW1	WORD		
		出力[2]	%IW2	WORD		
		出力[3]	%IW3	WORD		
		出力[9]	%IW9	WORD		
		出力[0]	%QW0	ARRAY [0..9] OF WORD		Modbus 入力レジスタ
mDI0		Bit0	%QX0.0	BOOL		
mDI1		Bit1	%QX0.1	BOOL		
mDI2		Bit2	%QX0.2	BOOL		
mDI3		Bit3	%QX0.3	BOOL		
		Bit4	%QX0.4	BOOL		
		Bit5	%QX0.5	BOOL		
		Bit14	%QX1.6	BOOL		
		Bit15	%QX1.7	BOOL		
mAI0		出力[1]	%QW1	WORD		
mAI1		出力[2]	%QW2	WORD		
mCNT0_L		出力[3]	%QW3	WORD		
mCNT0_H		出力[4]	%QW4	WORD		
mCNT1_L		出力[5]	%QW5	WORD		
mCNT1_H		出力[6]	%QW6	WORD		
		出力[7]	%QW7	WORD		
		出力[8]	%QW8	WORD		
		出力[9]	%QW9	WORD		

3. プログラムの作成とビルド

- 1 デバイスウィンドウ内の [PLC_PRG (PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```
mDI0 := DI0;  
mDI1 := DI1;  
mDI2 := DI2;  
mDI3 := DI3;  
mAI0 := DWORD_TO_WORD(AI0);  
mAI1 := DWORD_TO_WORD(AI1);  
mCNT0_L := DWORD_TO_WORD(CNT0 AND 16#0000FFFF);  
mCNT0_H := DWORD_TO_WORD(SHR(CNT0, 16) AND 16#0000FFFF);  
  
mCNT1_L := DWORD_TO_WORD(CNT1 AND 16#0000FFFF);  
mCNT1_H := DWORD_TO_WORD(SHR(CNT1, 16) AND 16#0000FFFF);  
DO0 := mDO0;  
DO1 := mDO1;
```

※ IECプログラム上で、デジタル入力、アナログ入力、カウンタ入力データをI/O変数からModbus変数へコピーします。デジタル出力データの場合は、逆にModbus変数からI/O変数へ内容をコピーします。

4. プログラムのダウンロードと実行

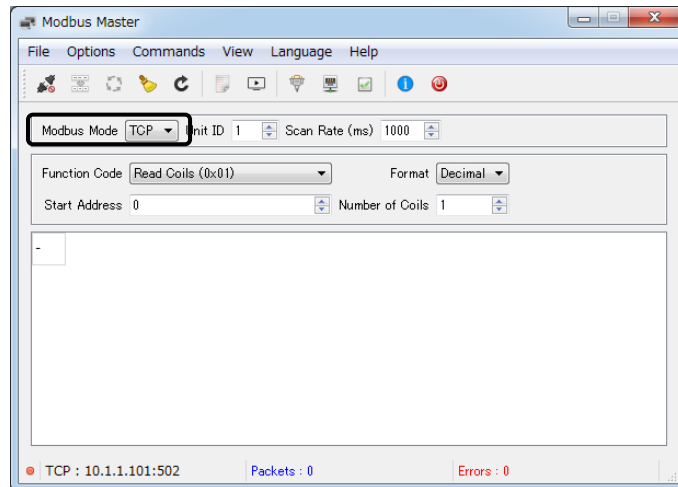
手順1 - 10ではデジタル出力ビット0に1を出力する例、手順11、12ではカウンタ入力0データを入力する例を解説しています。

- 1 CODESYSの [オンライン] メニューから [ログイン] を実行します。
- 2 ダウンロードの確認ダイアログが表示されますので [はい] をクリックします。

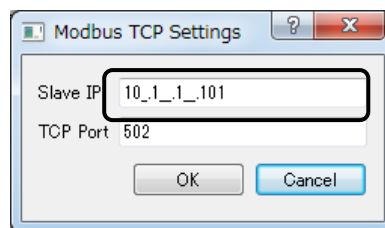


- 3 [デバッグ] メニューの [運転] を実行すると、プログラムの動作が開始します。
- 4 手順3の状態ではQModMasterを起動します。

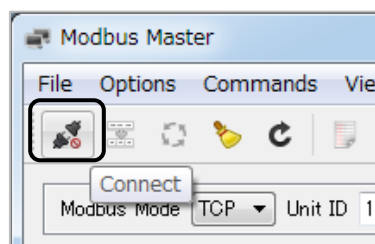
5 『Modbus Mode』 から [TCP] を選択します。



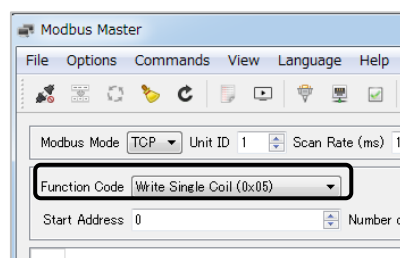
6 [Options] メニューから「Modbus TCP」を選択し、スレーブデバイスのIPアドレス「10.1.1.101」を設定します。



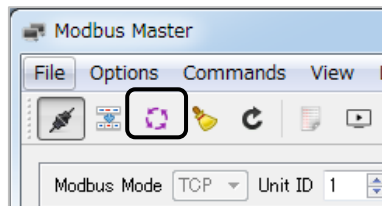
7 [Connect] アイコンをクリックします。



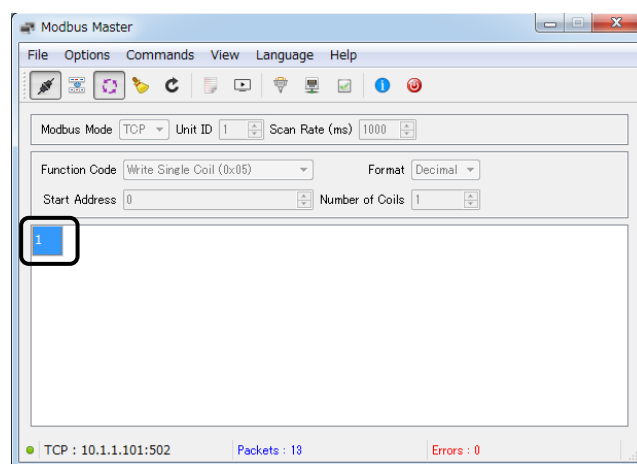
8 『Function Code』 から [Write Single Coil(0x05)] を選択します。



- 9 『Start Address』に「0」をセットして、[Scan] アイコンをクリックします。

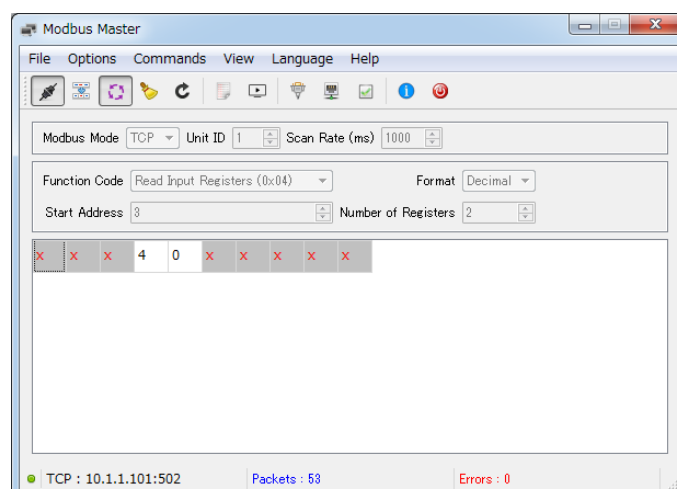


- 10 下の画面に「0」と表示されている部分をダブルクリックし、「1」に変更します。CONTEC CPS-PC341MB-ADSC1-9201のDO0 LEDが点灯します。



- 11 『Function Code』から [Read Input Registers(0x04)] を選択します。

- 12 『Start Address』に「3」、『Number of Registers』に「2」と入力し [Scan] アイコンをクリックします。左側にカウンタ0の下位16ビット、右側に上位8ビットが表示されます。

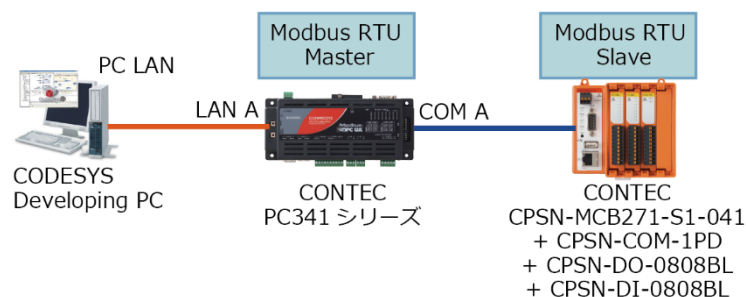


5. Modbus RTU Masterとして使用する

対応する製品	Slaveとして接続できる機器
CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	CPSN-MCB271-S1-041などの当社 対応製品 他社メーカーのSlave対応デバイス

1. デバイスの準備

コントローラをModbus RTU Master機器として動作させ、Modbus RTU Slaveとのデータ入出力を行います。ここではModbus RTU MasterとしてCPS-PC341MB-ADSC1-9201、Modbus RTU SlaveとしてCONTEC CPSN-MCB271-S1-041を使用します。



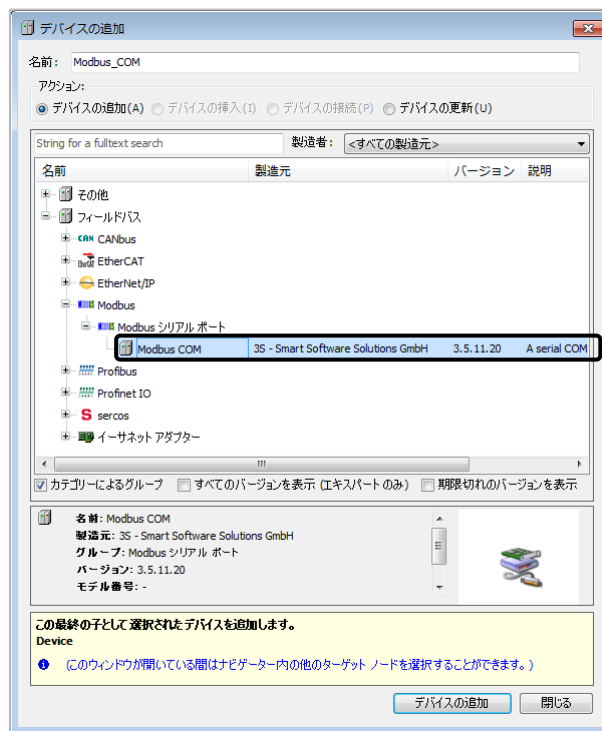
- 1 コントローラのLAN AをCODESYS開発環境のあるパソコンのLANポートと接続します。
- 2 コントローラのCOM AをCONTEC CPSN-MCB271-S1-041の拡張スロットに挿入したCPSN-COM-1PDのCOMポートへ接続します。

※ コントローラとCPSN-MCB271-S1-041のシリアル通信パラメーターを合わせてください。

2. スレーブデバイスの追加

- 1 新規CODESYSプロジェクトを作成します。
デバイスは接続しているModbusモデルの名称を、IECプログラム言語はSTを選択します。
※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。
- 2 デバイスウィンドウのコントローラ [Device (CODESYS Control CONTEC CPS-PCXXXXXXXXXX)] を右クリックして [デバイスの追加] を選択します。

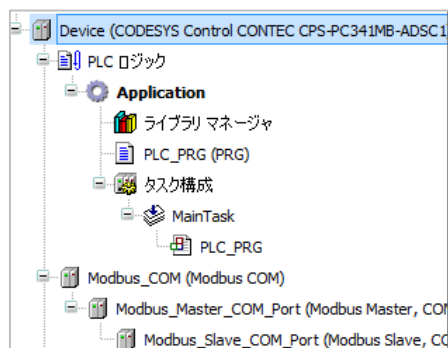
- 3** デバイスの追加ダイアログにて、[フィールドバス - Modbus - Modbusシリアルポート - Modbus COM] を選択して [デバイスの追加] ボタンをクリックします。



- 4** デバイスウィンドウの [Modbus_COM] を右クリックして [デバイスの追加] を選択します。

- 5** デバイスの追加ダイアログにて、[フィールドバス - Modbus - Modbusシリアルマスター - Modbus Master, COM Port] を追加します。

- 6** 再度デバイスウィンドウの [Modbus_Master_COM_Port] を右クリックして [デバイスの追加] を選択し、[フィールドバス - Modbus - Modbusシリアルスレーブ - Modbus Slave, COM Port] を追加します。



- 7** デバイスウィンドウから、[Modbus_COM] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。

- 8** [全般] タブを開きます。

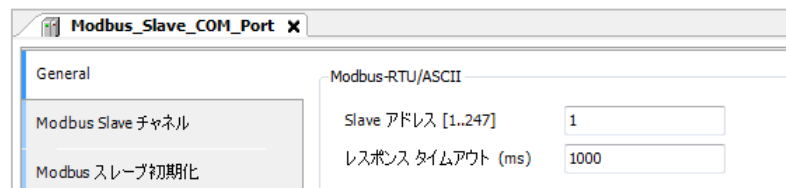
9 Modbus RTUで通信するために必要なパラメーターを設定します。



10 デバイスウィンドウから、[Modbus_Slave_COM_Port] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。

11 [General] タブを開きます。

12 『Slave アドレス』ヘスレーブデバイスのアドレスを入力します。



3. Modbus RTU Masterの設定

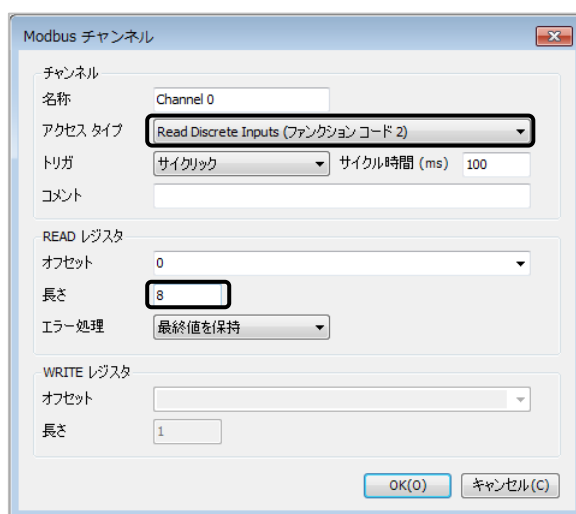
ここでは、2ビットのデジタル出力、8ビットのデジタル入力を行うサンプルを作成します。

本説明では、スレーブデバイスのModbusレジスタを下記のように登録します。

名称/ファンクションコード	アドレス	データ
Read Discrete Inputs(コード2)	0～7	CPSN-DI-0808BLのデジタル入力ビット0～7
Write Single Coil(コード5)	0	CPSN-DO-0808BLのデジタル出力ビット0
	1	CPSN-DO-0808BLのデジタル出力ビット1

※ サンプルとして作成する2ビットのデジタル出力は、Write Single Coil(コード5)、8ビットのデジタル入力はRead Discrete Inputs(コード2)を使用します。

- 1 デバイスウィンドウから、[Modbus_Slave_COM_Port] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。
- 2 [Modbus Slaveチャンネル] タブを開きます。
- 3 [チャンネルの追加] ボタンをクリックし、アクセスタイプから [Read Discrete Inputs(ファンクションコード2)] を選択、Readレジスタ長さには「8」と入力して [OK] をクリックします。

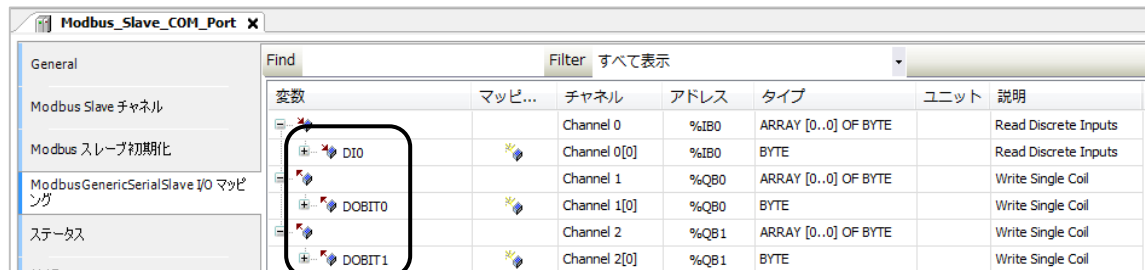


- 4 [チャンネルの追加] ボタンをクリックし、アクセスタイプから [Write Single Coil(ファンクションコード5)] を選択、Writeレジスタオフセットには「0」と入力して [OK] をクリックします。
- 5 [チャンネルの追加] ボタンをクリックし、アクセスタイプから再度 [Write Single Coil(ファンクションコード5)] を選択、Writeレジスタオフセットには「1」と入力して [OK] をクリックします。

Modbus_Slave_COM_Port							
General	名称	アクセス タイプ	トリガ	読み取りオフセ...	長さ	エラー処理	書き込み...
Modbus Slaveチャンネル	Channel 0	Read Discrete Inputs (ファンクションコード 02)	サイクリック, t#100ms	16#0000	8	最終値を保持	
	Channel 1	Write Single Coil (ファンクションコード 05)	サイクリック, t#100ms				16#0000
Modbus スレーブ初期化	Channel 2	Write Single Coil (ファンクションコード 05)	サイクリック, t#100ms				16#0001

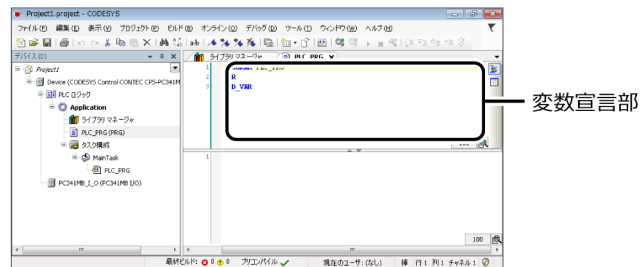
6 [ModbusGenericSerialSlave I/O マッピング] をクリックします。

7 Channel0[0]、Channel1[0]、Channel2[0]にそれぞれ「DI0」、「DOBIT0」、「DOBIT1」を割り当てます。



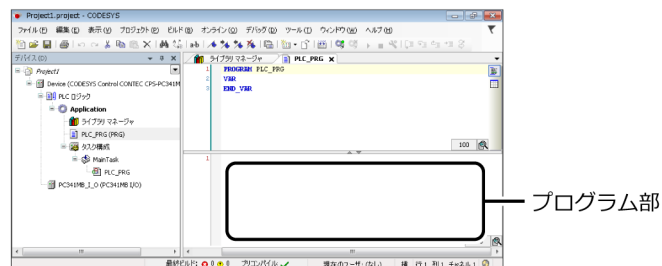
4. プログラムの作成とビルド

- 1 デバイスウィンドウ内の [PLC_PRG (PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に以下のソースコードを記述します。



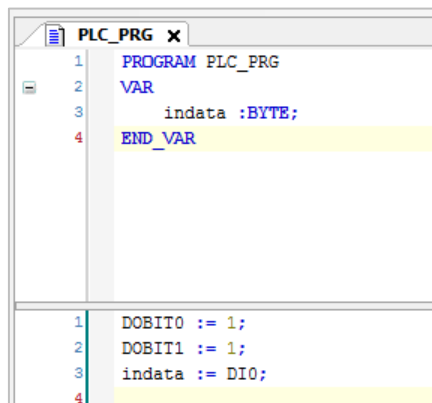
```
indata :BYTE;
```

- 3 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```
DOBIT0 := 1;  
DOBIT1 := 1;  
indata := DI0;
```

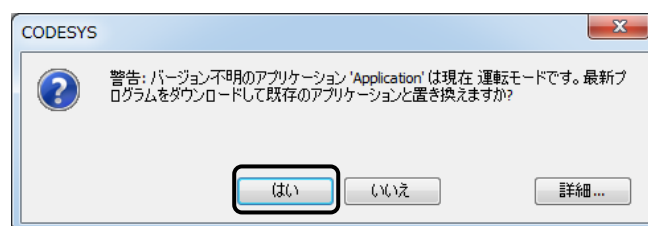
上記のプログラムは、CPSN-DO-0808BLのデジタル出力ビット0、1へON信号を出力し、CPSN-DI-0808BLのデジタル入力ポートから信号を入力します。



- CODESYSの [ビルド] メニューから [リビルド] を実行し、ビルドプロセスが正常に完了するか確認します。

5. プログラムのダウンロードと実行

- CODESYSの [オンライン] メニューから [ログイン] を実行します。
- ダウンロードの確認ダイアログが表示されますので [はい] をクリックします。



- [デバッグ] メニューの [運転] を実行すると、プログラムの動作が開始します。CPSN-DO-0808BLのフロントパネルにて、DOビット0、1のLEDが点灯することが確認できます。

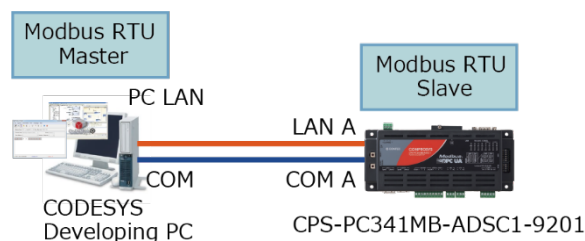
6. Modbus RTU Slaveとして使用する

対応する製品	接続する機器
CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	—

1. デバイスの準備

コントローラをModbus RTU Slave機器として動作させ、Modbus RTU Masterからのデータ入出力に対応します。ここではModbus RTU MasterとしてCODESYS開発環境がインストールされているパソコンを、Modbus RTU Slaveとして、CPS-PC341MB-ADSC1-9201を使用した、CODESYSプログラミング例を解説します。

Modbus RTU Masterソフトウェアとして、QModMasterを使用します。



1 Modbus RTU MasterソフトウェアQModMasterをパソコンにインストールします。

QModMasterはフリーソフトで、下記よりダウンロードできます。

ダウンロード <http://sourceforge.net/projects/qmodmaster/>

2 CPS-PC341MB-ADSC1-9201のCOM AとCODESYS開発環境のあるパソコンのCOMポートをシリアルケーブルで接続します。

2. スレーブ設定をする

1 新規CODESYSプロジェクトを作成します。

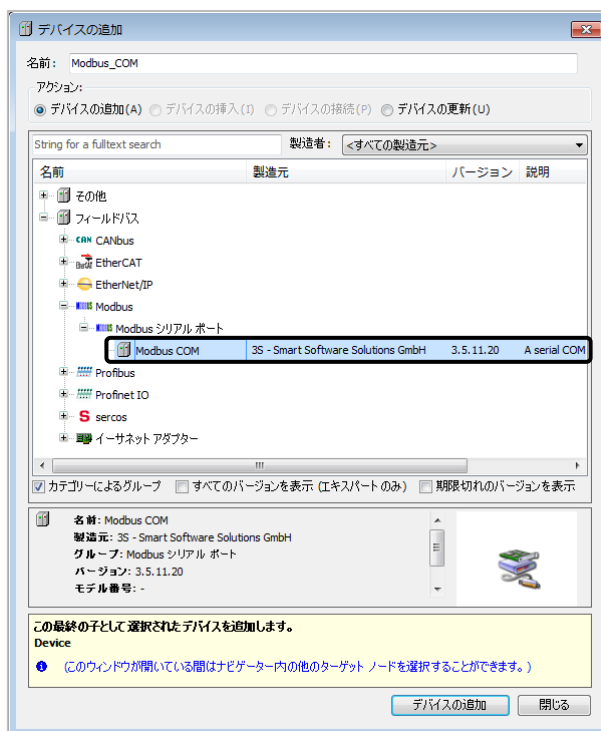
IECプログラム言語はSTを選択します。

※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。

2 デバイスウィンドウのコントローラ [Device (CODESYS Control CONTEC CPS-PCXXXXXXXXXX)] を右クリックして [デバイスの追加] を選択します。

3 デバイスの追加ダイアログにて、[フィールドバス - Modbus - Modbusシリアルポート - Modbus COM] を選択して [デバイスの追加] ボタンをクリックします。

※ Modbus_COM Adapterが追加されます。



4 デバイスウィンドウの [Modbus_COM] を右クリックして [デバイスの追加] を選択します。

5 デバイスの追加ダイアログにて、[フィールドバス - Modbus - Modbusシリアルデバイス - Modbus Serial Device] を追加します。

※ Modbus_COM Adapterデバイスの下にModbus Serial Deviceが追加されます。

6 デバイスウィンドウから、[Modbus_COM] アイコンをクリックしてデバイスコンフィギュレーションウィンドウを開きます。

7 [全般] タブを開きます。

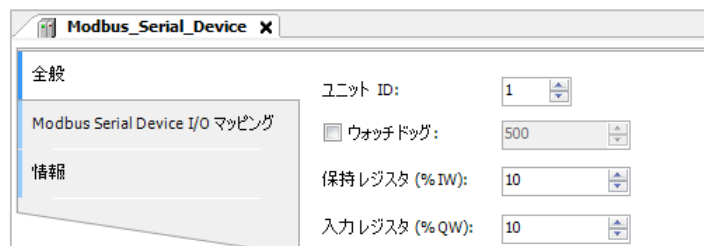
8 Modbus RTUで通信するために必要なパラメーターを設定します。



9 デバイスウィンドウから、[Modbus_Serial_Device] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。

10 [全般] タブを開きます。

11 『ユニット ID』ヘスレーブデバイスのアドレスを入力します。



12 デバイスウィンドウから [PC341MB_I_O] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。

13 [Internal I/O マッピング] をクリックします。

14 以下のとおりに変数を定義します。

変数(入力値)	チャネル	アドレス
	do0-1	
DO0	Bit0	%QX20.0
DO1	Bit1	%QX20.1
	Di0-3	
DI0	Bit0	%IX20.0
DI1	Bit1	%IX20.1
DI2	Bit2	%IX20.2
DI3	Bit3	%IX20.3
CNT0	cnt0	%ID6
CNT1	cnt1	%ID7
AI0	ai0	%ID8
AI1	ai1	%ID9

15 デバイスウィンドウから [Modbus_Serial_Device] アイコンをダブルクリックしてデバイスコンフィギュレーションウィンドウを開きます。

16 [Modbus Serial Device I/O マッピング] をクリックします。

17 以下のとおりに変数を定義します。

変数(入力値)	チャネル	アドレス
	入力	%IW1
	入力[0]	%IW1
mDO0	Bit0	%IX2.0
mDO1	Bit1	%IX2.1
	:	
	Bit15	%IX3.7
	入力[1]	%IW2
	:	
	入力[9]	%IW10
	出力	%QW1
	出力[0]	%QW1
mDI0	Bit0	%QX0.0
mDI1	Bit1	%QX0.1
mDI2	Bit2	%QX0.2
mDI3	Bit3	%QX0.3
	:	
	Bit15	%QX1.7
mAI0	出力[1]	%QW2
MAI1	出力[2]	%QW3
mCNT0_L	出力[3]	%QW4
mCNT0_H	出力[4]	%QW5
mCNT1_L	出力[5]	%QW6
mCNT1_H	出力[6]	%QW7
	:	
	出力[9]	%QW10

※ I/Oマッピングの設定、Modbus Slaveに対してI/Oマッピングの定義がされました。Modbus SlaveのI/Oマッピングに変数を定義することで、Modbus Masterとのデータ通信が可能となります。

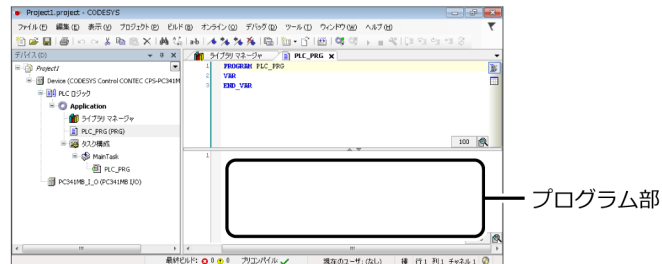
※ 入力マッピング領域に出力用変数、出力マッピング領域に入力用の変数が定義されていますが、これは入力マッピング領域がMasterからSlaveへの入力、出力マッピング領域がSlaveからMasterへの出力という意味となっているためです。

※ カウンタデータは24ビットでModbusのWORD領域に収まらないため、下位16ビットをmCNT0_Lへ、上位8ビットをmCNT0_Hへ格納します。

全般	検索	フィルター	すべて表示			
Modbus TCP Slave Device I/O マッピング	変数	マッピング...	チャネル	アドレス	タイプ	ユニット 説明
情報			入力	%IW0	ARRAY [0..9] OF WORD	Modbus 保持レジスタ
			入力[0]	%IW0	WORD	
	mDO0		Bit0	%IX0.0	BOOL	
	mDO1		Bit1	%IX0.1	BOOL	
			Bit2	%IX0.2	BOOL	
			Bit3	%IX0.3	BOOL	
			Bit4	%IX0.4	BOOL	
			Bit5	%IX0.5	BOOL	
			Bit6	%IX1.5	BOOL	
			Bit14	%IX1.6	BOOL	
			Bit15	%IX1.7	BOOL	
			入力[1]	%IW1	WORD	
			入力[2]	%IW2	WORD	
			入力[3]	%IW3	WORD	
			出力	%QW0	ARRAY [0..9] OF WORD	Modbus 入力レジスタ
			出力[0]	%QW0	WORD	
	mDI0		Bit0	%QX0.0	BOOL	
	mDI1		Bit1	%QX0.1	BOOL	
	mDI2		Bit2	%QX0.2	BOOL	
	mDI3		Bit3	%QX0.3	BOOL	
			Bit4	%QX0.4	BOOL	
			Bit5	%QX0.5	BOOL	
			Bit6	%QX1.5	BOOL	
			Bit14	%QX1.6	BOOL	
			Bit15	%QX1.7	BOOL	
	mAI0		出力[1]	%QW1	WORD	
	mAI1		出力[2]	%QW2	WORD	
	mCNT0_L		出力[3]	%QW3	WORD	
	mCNT0_H		出力[4]	%QW4	WORD	
	mCNT1_L		出力[5]	%QW5	WORD	
	mCNT1_H		出力[6]	%QW6	WORD	
			出力[7]	%QW7	WORD	
			出力[8]	%QW8	WORD	
			出力[9]	%QW9	WORD	

3. プログラムの作成とビルド

- 1 デバイスウィンドウ内の [PLC_PRG (PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```
mDI0 := DI0;
mDI1 := DI1;
mDI2 := DI2;
mDI3 := DI3;
mAI0 := DWORD_TO_WORD(AI0);
mAI1 := DWORD_TO_WORD(AI1);
mCNT0_L := DWORD_TO_WORD(CNT0 AND 16#0000FFFF);
mCNT0_H := DWORD_TO_WORD(SHR(CNT0, 16) AND 16#0000FFFF);
mCNT1_L := DWORD_TO_WORD(CNT1 AND 16#0000FFFF);
mCNT1_H := DWORD_TO_WORD(SHR(CNT1, 16) AND 16#0000FFFF);
DO0 := mDO0;
DO1 := mDO1;
```

※ IECプログラム上で、デジタル入力、アナログ入力、カウンタ入力データをI/O変数からModbus変数へコピーします。デジタル出力データの場合は、逆にModbus変数からI/O変数へ内容をコピーします。

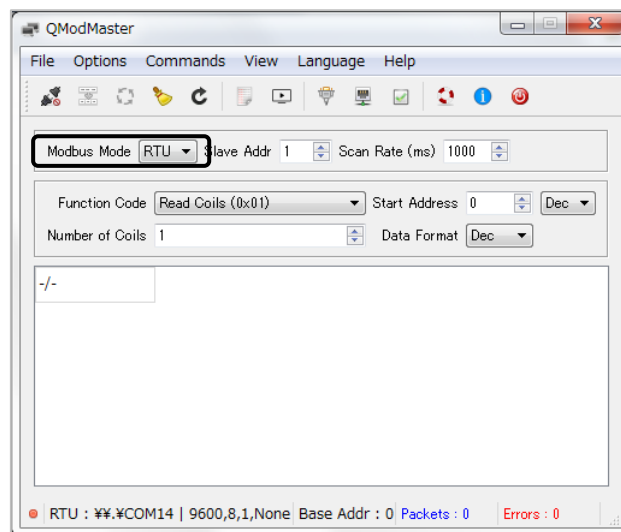
4. プログラムのダウンロードと実行

手順1～10ではデジタル出力ビット0に1を出力する例、手順11、12ではカウンタ入力0データを入力する例を解説しています。

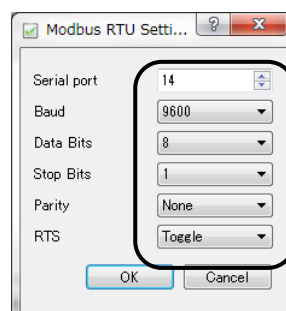
- 1 CODESYSの「オンライン」メニューから「ログイン」を実行します。
- 2 ダウンロードの確認ダイアログが表示されますので「はい」をクリックします。



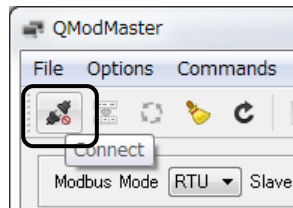
- 3 「デバッグ」メニューの「運転」を実行すると、プログラムの動作が開始します。
- 4 手順3の状態ではQModMasterを起動します。
- 5 『Modbus Mode』から「RTU」を選択します。



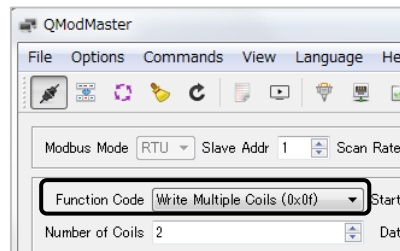
- 6 「Options」メニューから「Modbus RTU」を選択し、Modbus RTUで通信するために必要なパラメータを設定します。



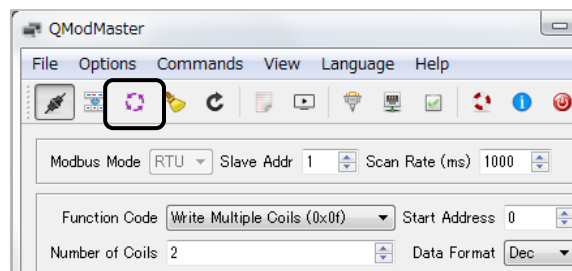
7 [Connect] アイコンをクリックします。



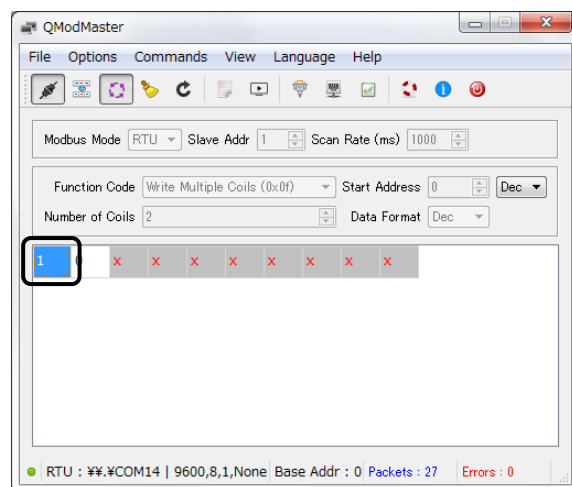
8 『Function Code』 から [Write Multiple Coils(0x0f)] を選択します。



9 『Start Address』 に「0」、『Number of Coils』 に「2」をセットして、[Scan] アイコンをクリックします。

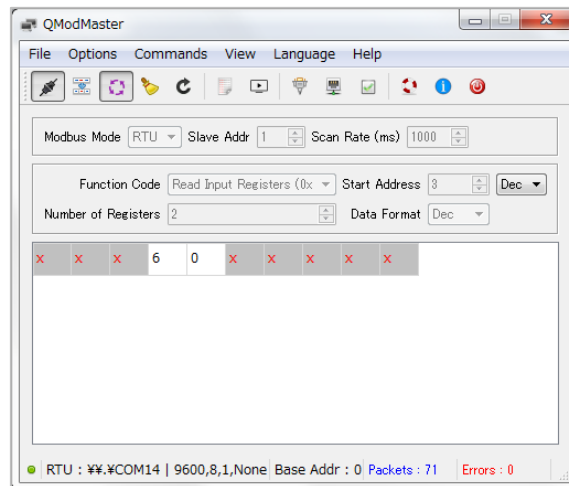


10 下の画面に「0」と表示されている部分をダブルクリックし、「1」に変更します。CONTEC CPS-PC341MB-ADSC1-9201のDO0 LEDが点灯します。



11 『Function Code』から [Read Input Registers(0x04)] を選択します。

12 『Start Address』に「3」、『Number of Registers』に「2」と入力し [Scan] アイコンをクリックします。左側にカウンタ0の下位16ビット、右側に上位8ビットが表示されます。



7. OPC UA Serverを設定する

CODESYSでOPC UA Server機能を付加します。

1. デバイスの準備

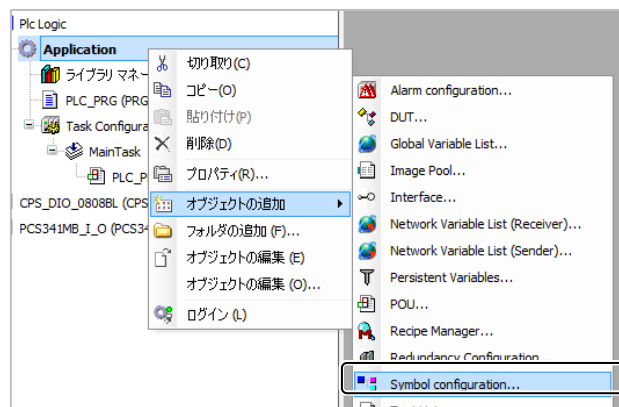
- 1 『基本的なプログラミング手順(P56)』で作成したプロジェクトをCODESYSで開きます。
- 2 OPC UAクライアントソフトとしてUnified Automation社から提供されるUaExpertを用意します。
UaExpertは下記URLより無償ダウンロード可能です。(ユーザー登録が必要となります)

ダウンロード

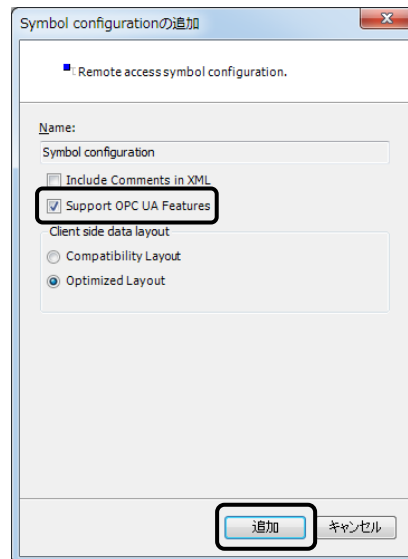
<https://www.unified-automation.com/downloads/opc-ua-clients.html>

2. CODESYSの設定

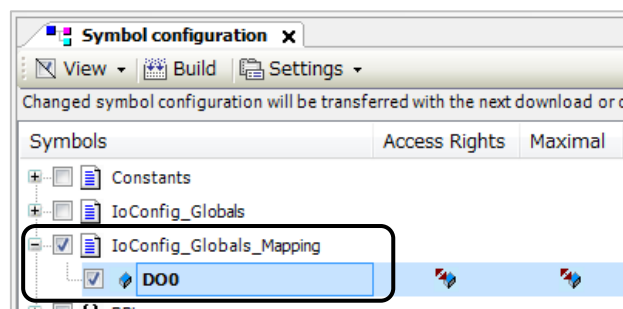
- 1 デバイスウィンドウに表示されている[Application]を右クリックして、[オブジェクトの追加] - [Symbol configuration]を選択します。



- 2 『Symbol configurationの追加』ダイアログにて [Support OPC UA Features] にチェックを入れ、[追加] ボタンをクリックします。



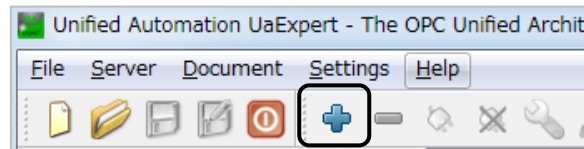
- 3 『Symbol configuration』ウィンドウが表示されるので、一覧からOPC UA Server用に使用する変数を選択します。ここでは、『IoConfig_Global_Mapping』以下の『DO0』変数にチェックを入れます。



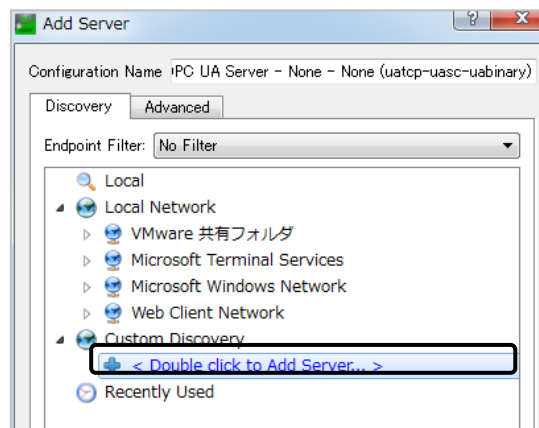
- 4 プログラムをビルドしてエラー等発生しないことを確認し、コントローラへログインして動作を開始させておきます。

3. OPC UA Clientの設定

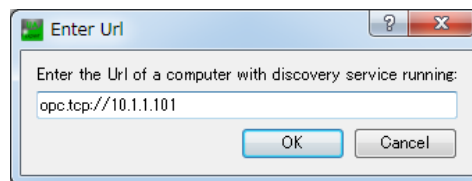
- 1 OPC UAクライアントソフトUaExpertを起動し [+] のアイコンをクリックしてサーバーを追加します。



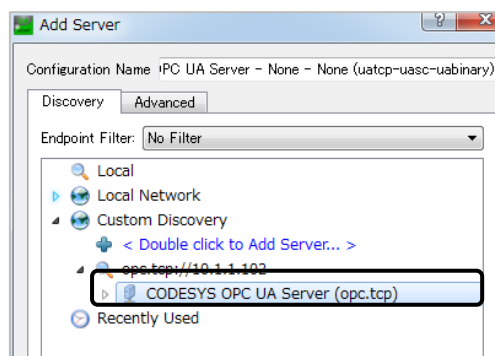
- 2 [Double click to Add Server...] をダブルクリックします。



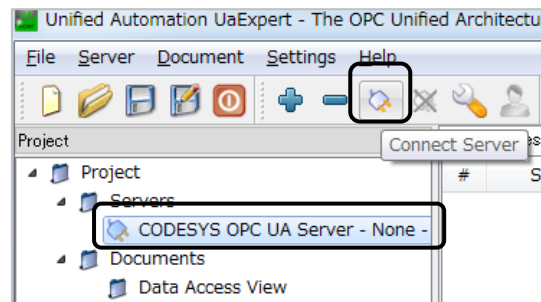
- 3 URLにコントローラのIPアドレス「opc.tcp://10.1.1.101」を入力します。



- 4 ツリーが作成されるので、[CODESYS OPC UA Server] を選択し [OK] をクリックします。

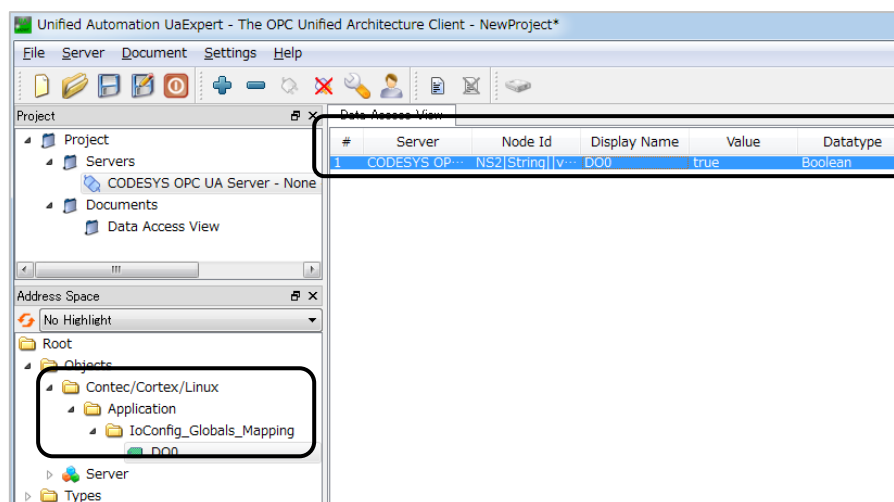


5 サーバーを選択して接続アイコンをクリックします。



6 『Address Space』の『Root¥Objects¥Contec/Cortex/Linux¥Application¥PLC_PRG¥』のツリー階層内に、CODESYS OPC Configuratorで登録した変数が表示されているので、モニタリングしたい変数(ここではD00)を『Data Access View』ウィンドウヘドラッグします。

変数の状態を確認することができます。

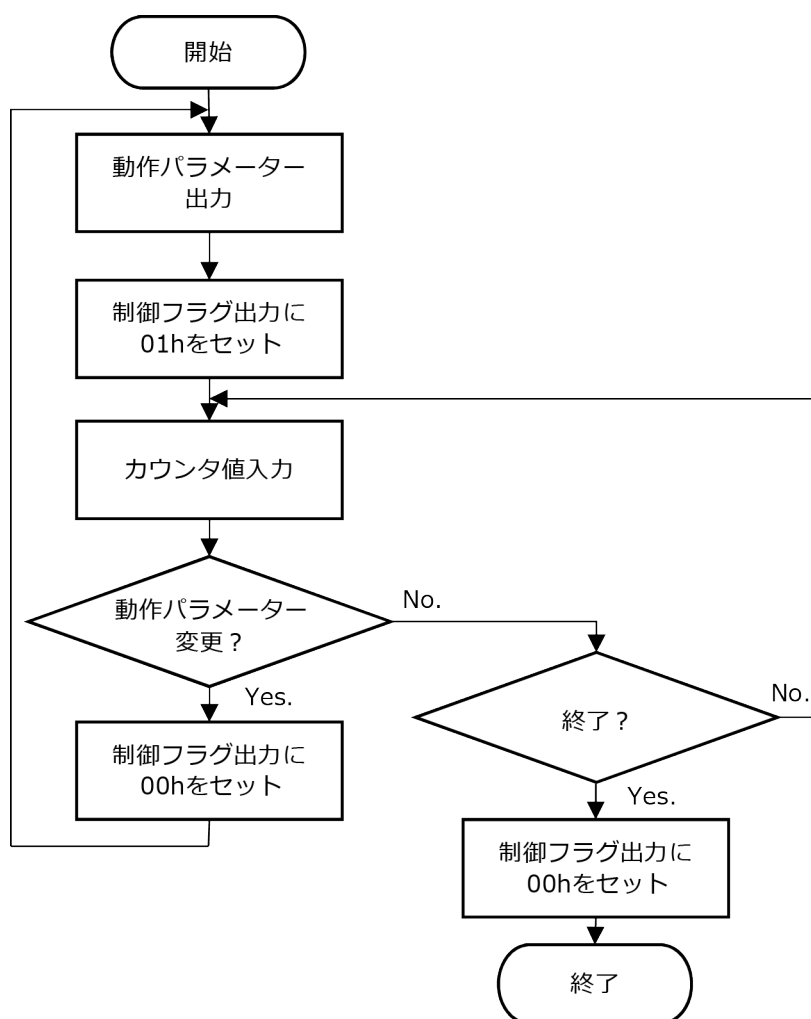


8. カウンタ入力

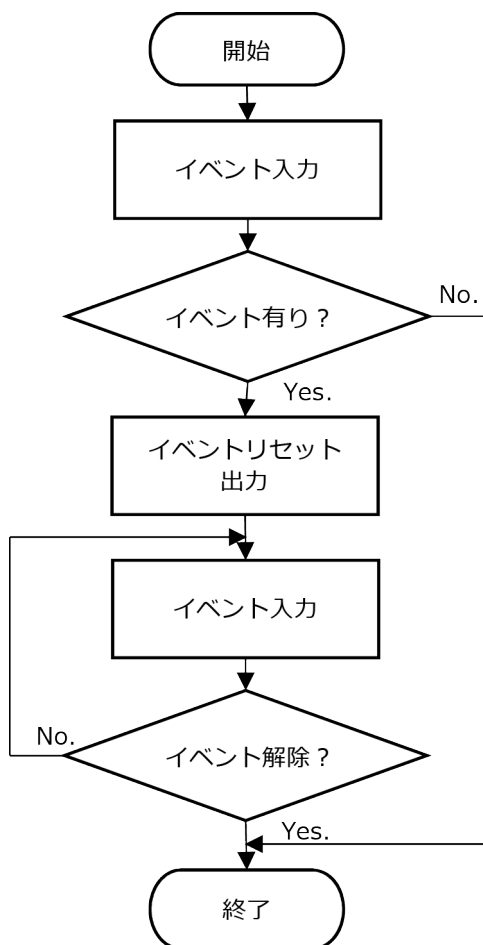
カウンタ入力スタックタイプのCPUコントローラにI/OモジュールCPS-CNT-3202Iを接続したときに使用出来ます。カウンタ入力機能を使用する場合は以下の手順が必要となります。

『**基本的なプログラミング手順(P56)**』によってプログラミングの準備ができましたら、以下の動作手順を参考にプログラミングを行ってください。

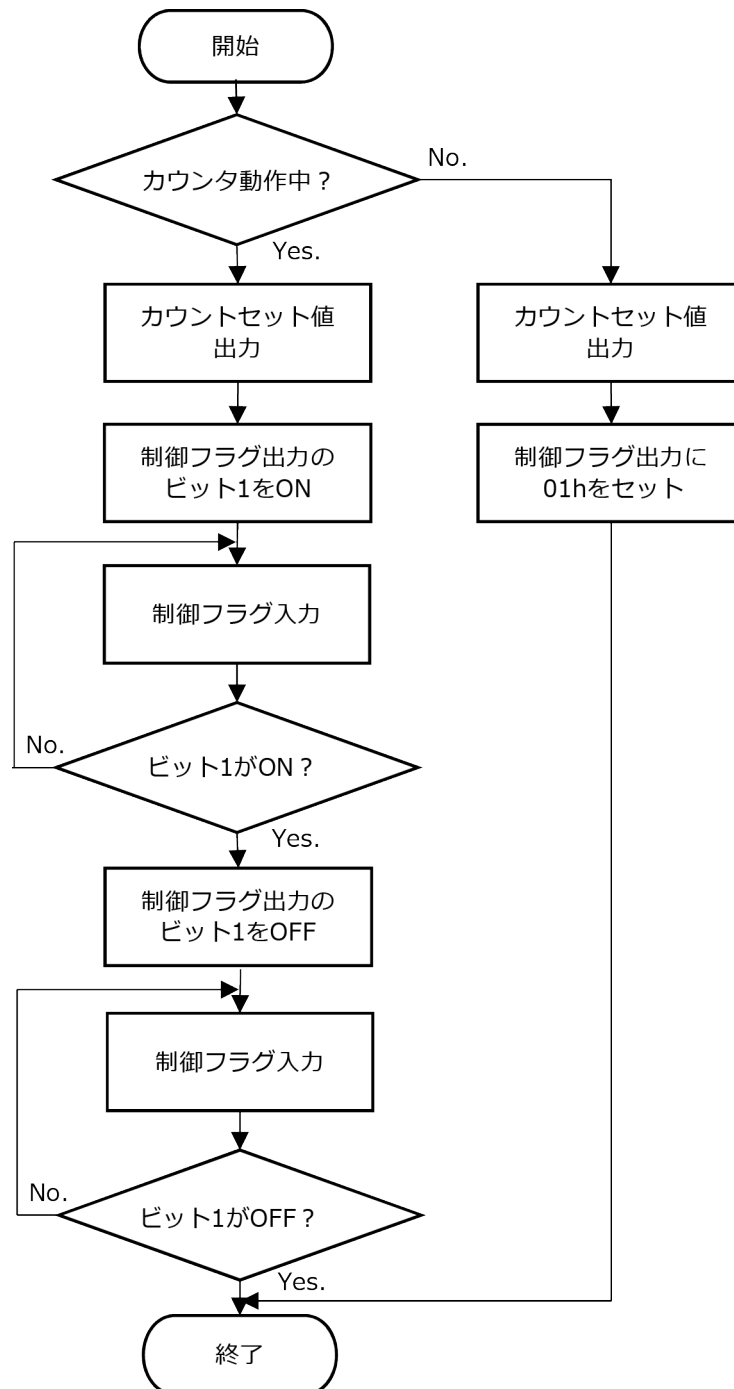
1. 基本的な動作手順



2. イベント動作手順



3. カウント値設定動作手順



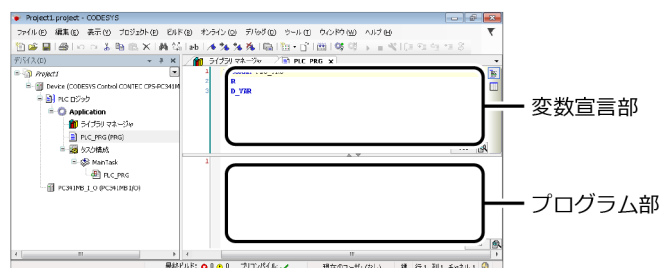
4. サンプルプログラム

前述で示した動作に関するサンプルプログラムを提供しています。

プログラムタイトル	機能
サンプル1	CPS-CNT-3202Iを設定してカウンタ入力を行う場合
サンプル2	比較値一致のイベントが発生した場合
サンプル3	動作中にカウント値を変更する場合

以下にサンプルプログラムの実行手順を示します。

- 1 デバイスウィンドウ内の [PLC_PRG(PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に変数宣言部のソースコードを記述します。
- 3 『STエディタウィンドウ』の『プログラム部』にプログラム部のソースコードを記述します。



◆ サンプル1

CPS-CNT-3202Iを設定してカウンタ入力を行う場合

変数宣言部

```
uiState: UINT := 0;  
dwCounter: DWORD;
```

プログラム部

```
CASE uiState OF  
  0:  
    CNT0_OutCtrl := 0;  
    CNT0_EventReset := 0;  
    uiState := 1;  
  1:  
    CNT0_ZMode := 0;  
    CNT0_ZLogic := 0;  
    CNT0_Dir := 0;  
    CNT0_Phase := 1;  
    CNT0_Multi := 0;  
    CNT0_Clear := 1;  
  
    CNT0_DiFilter := 0;  
    CNT0_Pulse := 0;  
    CNT0_CntSet := 0;  
    CNT0_CntMatch := 0;  
    uiState := 2;  
  2:  
    CNT0_OutCtrl := 16#01;  
    uiState := 3;  
  3:  
    dwCounter := CNT0_CntVal;  
    IF DWORD_TO_DINT(dwCounter) >= 100 THEN  
      uiState := 4;  
    END_IF  
  4:  
    CNT0_OutCtrl := 0;  
    uiState := 5;  
  5:  
END_CASE
```

◆ サンプル2

比較値一致のイベントが発生した場合

変数宣言部

```
uiState: UINT := 0;  
dwCounter: DWORD;  
byEvent: BYTE;
```

プログラム部

```
CASE uiState OF  
  0:  
    CNT0_OutCtrl := 0;  
    CNT0_EventReset := 0;  
    uiState := 1;  
  1:  
    CNT0_ZMode := 0;  
    CNT0_ZLogic := 0;  
    CNT0_Dir := 0;  
    CNT0_Phase := 1;  
    CNT0_Multi := 0;  
    CNT0_Clear := 1;  
    CNT0_DiFilter := 0;  
    CNT0_Pulse := 0;  
    CNT0_CntSet := 0;  
    CNT0_CntMatch := 100;  
    uiState := 2;  
  2:  
    CNT0_OutCtrl := 16#01;  
    uiState := 3;  
  3:  
    dwCounter := CNT0_CntVal;  
    byEvent := CNT0_Event;  
  
    IF (byEvent AND 16#01) = 16#01 THEN  
      uiState := 4;  
    END_IF  
  4:  
    CNT0_EventReset := 16#01;  
    uiState := 5;  
  5:  
    byEvent := CNT0_Event;  
    IF (byEvent AND 16#01) = 16#00 THEN  
      uiState := 6;  
    END_IF  
  6:  
    CNT0_OutCtrl := 0;  
    uiState := 7;  
  7:  
END_CASE
```

◆ サンプル3

動作中にカウント値を変更する場合

変数宣言部

```
uiState: UINT := 0;  
dwCounter: DWORD;  
byInCtrl: BYTE;
```

プログラム部

```
CASE uiState OF  
  0:  
    CNT0_OutCtrl := 0;  
    CNT0_EventReset := 0;  
    uiState := 1;  
  1:  
    CNT0_ZMode := 0;  
    CNT0_ZLogic := 0;  
    CNT0_Dir := 0;  
    CNT0_Phase := 1;  
    CNT0_Multi := 0;  
    CNT0_Clear := 1;  
    CNT0_DiFilter := 0;  
    CNT0_Pulse := 0;  
    CNT0_CntSet := 100;  
    CNT0_CntMatch := 0;  
    uiState := 2;  
  2:  
    CNT0_OutCtrl := 16#01;  
    uiState := 3;  
  3:  
    dwCounter := CNT0_CntVal;  
  
    IF DWORD_TO_DINT(dwCounter) >= 200 THEN  
      uiState := 4;  
    END_IF  
  4:  
    CNT0_CntSet := 100;  
    CNT0_OutCtrl := CNT0_OutCtrl OR 16#02;  
    uiState := 5;  
  5:  
    byInCtrl := CNT0_InCtrl;  
    dwCounter := CNT0_CntVal;  
  
    IF (byInCtrl AND 16#02) = 16#02 THEN  
      uiState := 6;  
    END_IF  
  6:  
    CNT0_OutCtrl := CNT0_OutCtrl AND 16#FD;  
    uiState := 7;
```

```
7:
  byInCtrl := CNT0_InCtrl;
  dwCounter := CNT0_CntVal;

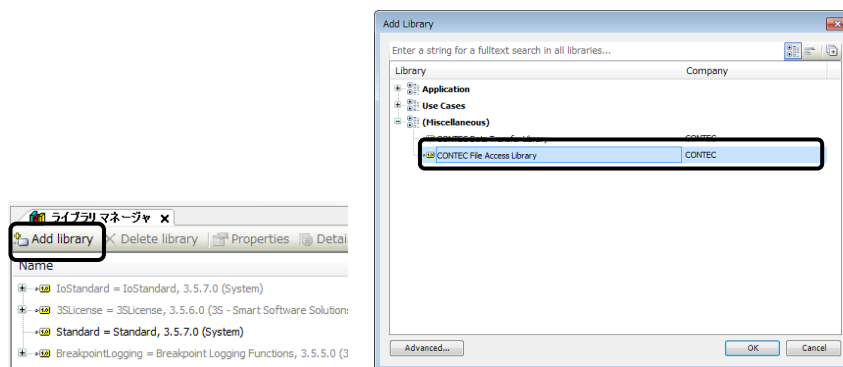
  IF (byInCtrl AND 16#02) = 16#00 THEN
    uiState := 8;
  END_IF
8:
  CNT0_OutCtrl := 0;
  uiState := 9;
9:
END_CASE
```

9. ファイルにアクセスする

本製品に接続された記憶デバイスのファイルに対してアクセス許可を割り当てます。

1. ファイルアクセスライブラリの設定

- 1 新規CODESYSプロジェクトを作成します。
IECプログラム言語はSTを選択します。
※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。
- 2 デバイスウィンドウの「ライブラリマネージャ」をダブルクリックします。
- 3 「Add Library」をクリックし、『(Miscellaneous)』のツリー階層内の「CONTEC File Access Library」を選択して「OK」をクリックします。



2. 関数一覧

関数名	機能
CFA_FileAccess	指定領域のアクセス制御
CFA_FileOpen	ファイルオープン
CFA_FileClose	ファイルクローズ
CFA_FileCloseAll	ファイル全クローズ
CFA_FileRead	ファイル読み込み
CFA_FileWrite	ファイル書き込み
CFA_FileSeek	ファイルポインター移動
CFA_FileGetLine	ファイルから1行分の文字列読み込み
CFA_FilePutLine	ファイルへの文字列書き込み
CFA_FileDelete	ファイル削除
CFA_StringSeparate	文字列分解

3. データタイプ

◆ ERROR

名前	値	意味
NO_ERROR	0	正常終了
INVALID_PARAM	1	パラメーターが不正です。
INTERNAL_ERROR	2	リソース不足です。
INVALID_HANDLE	3	ハンドル番号が不正です。
NOT_EXIST	4	ファイルが存在しません。
EXIST	5	ファイルが既に存在します。
READ_ONLY_FS	6	読み込み専用でマウントされたデバイス上でファイル書き込みを実行しました。
NOT_MOUNT	7	マウントされていません。
GENERAL_ERROR	8	一般エラーです。

◆ AREATYPE

名前	値	意味
ROM	0	ROM領域を指定します。
RAM	1	RAM領域を指定します。
SD	2	SDカードを指定します。
USB	3	USBデバイスを指定します。

◆ SEEKWHENCE

名前	値	意味
CUR	0	ファイルポインタの現在位置。
END	1	ファイルの終わり。
SET	2	ファイルの先頭。

4. 関数詳細

◆ CFA_FileAccess

機能

指定した領域のアクセス状態を制御します。

入力値

名前	型	意味
eArea	AREATYPE	アクセスする領域のタイプを指定します。 RAM領域は常に読み書き可能な状態であるので、この関数による制御は必要ありません。
bMount	BOOL	ファイルシステムへのマウント状態を指定します。 アクセスする領域がUSBデバイスの場合に必要です。 TRUE : ファイルシステムにマウントします。 FALSE : ファイルシステムをアンマウントします。
bWriteEnable	BOOL	書き込み許可状態を指定します。 TRUE : 書き込みを許可します。 FALSE : 書き込みを禁止します。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CFA_FileOpen

機能

指定した領域にあるファイルをオープンします。

入力値

名前	型	意味
eArea	AREATYPE	アクセスする領域のタイプを指定します。
sFileName	STRING(256)	ファイル名を指定します。
sFileMode	STRING(8)	ファイルのオープンモードを指定します。 "r" : 読み出し専用でオープンします。 "w" : 書き込み専用でオープンします。 "a" : 追加書き込み専用でオープンします。 "r+" : 読み書き専用でオープンします。 "w+" : 読み書き専用でオープンします。 "a+" : 読み出しと追加書き込み専用でオープンします。

出力値

名前	型	意味
eError	ERROR	エラーコード。
hFile	DWORD	ファイルのハンドル番号。

◆ CFA_FileClose

機能

ファイルをクローズします。

入力値

名前	型	意味
hFile	DWORD	ファイルのハンドル番号を指定します。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CFA_FileCloseAll

機能

オープンされている全てのファイルをクローズします。

入力値

なし

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CFA_FileRead

機能

ファイルからデータを読み込みます。

入力値

名前	型	意味
hFile	DWORD	ファイルのハンドル番号を指定します。
pBuffer	POINTOR TO BYTE	読み込んだデータを格納する領域のアドレスを指定します。
szBuffer	WORD	データを格納する領域のサイズを指定します。

出力値

名前	型	意味
eError	ERROR	エラーコード。
szSize	WORD	読み込んだデータ数。

◆ CFA_FileWrite

機能

ファイルにデータを書き込みます。

入力値

名前	型	意味
hFile	DWORD	ファイルのハンドル番号を指定します。
pBuffer	POINTOR TO BYTE	書き込むデータ領域のアドレスを指定します。
szSize	WORD	書き込むデータ数を指定します。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CFA_FileSeek

機能

ファイルポインターを指定位置に移動します。

入力値

名前	型	意味
hFile	DWORD	ファイルのハンドル番号を指定します。
diOffset	DINT	基準位置から移動するバイト数を指定します。
eWhence	SEEKWHENCE	基準位置を指定します。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CFA_FileGetLine

機能

ファイルから1行分の文字列を読み込みます。

読み込んだ改行文字は削除されます。

入力値

名前	型	意味
hFile	DWORD	ファイルのハンドル番号を指定します。
pszString	POINTOR TO STRING	読み込んだ文字列を格納する領域のアドレスを指定します。
szSring	WORD	文字列を格納する領域のサイズを指定します。 szString - 1が最大文字数です。 改行文字を含む十分なサイズを指定してください。

出力値

名前	型	意味
eError	ERROR	エラーコード。
szSize	WORD	読み込んだ文字列のサイズ。

◆ CFA_FilePutLine

機能

ファイルに文字列を書き込みます。

文字列の最後に改行文字が自動的に付加されます。

入力値

名前	値	意味
hFile	DWORD	ファイルのハンドル番号を指定します。
pszString	POINTOR TO STRING	書き込む文字列のアドレスを指定します。

出力値

名前	値	意味
eError	ERROR	エラーコード。

◆ CFA_FileDelete

機能

指定した領域にあるファイルを削除します。

入力値

名前	値	意味
eArea	AREATYPE	アクセスする領域のタイプを指定します。
sFileName	STRING(256)	ファイル名を指定します。

出力値

名前	値	意味
eError	ERROR	エラーコード。

◆ CFA_StringSeparate

機能

文字列を区切り文字で区切った文字列に分解します。

入力値

名前	値	意味
pszString	POINTER TO STRING	分解対象の文字列のアドレスを指定します。
pszSeparator	POINTER TO STRING	区切り文字群のアドレスを指定します。
pszLeft	POINTER TO STRING	区切り文字で区切った左辺側の文字列を格納する領域のアドレスを指定します。
pszRight	POINTER TO STRING	区切り文字で区切った右辺側の文字列を格納する領域のアドレスを指定します。

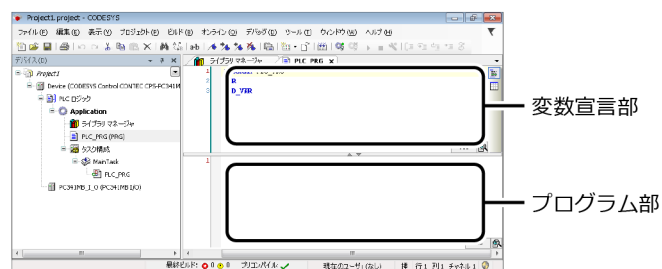
出力値

名前	値	意味
eError	ERROR	エラーコード。

5. サンプルプログラム

以下にサンプルプログラムの実行手順を示します。

- 1 デバイスウィンドウ内の [PLC_PRG(PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に変数宣言部のソースコードを記述します。
- 3 『STエディタウィンドウ』の『プログラム部』にプログラム部のソースコードを記述します。



◆ サンプル 1

USBデバイス上にファイルを書き込む場合

変数宣言部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
byBuffer: ARRAY[0..255] OF BYTE;
iIndex: INT;
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FileWrite: CONTEC_File_Access_Library.CFA_FileWrite;
```

プログラム部

```
CASE uiState OF
0:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := TRUE;
    CFA_FileAccess.bWriteEnable := TRUE;
    CFA_FileAccess(eError => eError);
    IF eError = 0 THEN
        uiState := 1;
    ELSE
        uiState := 5;
    END_IF
1:
    CFA_FileOpen.eArea := eAreaType;
    CFA_FileOpen.sFileName := sFileName;
    CFA_FileOpen.sFileMode := 'w';
    CFA_FileOpen(hFile => hFile, eError => eError);
    IF eError = 0 THEN
        uiState := 2;
    ELSE
        uiState := 4;
    END_IF
2:
    FOR iIndex := 0 TO 255 DO
        byBuffer[iIndex] := INT_TO_BYTE(iIndex);
    END_FOR
    CFA_FileWrite.pBuffer := ADR(byBuffer);
    CFA_FileWrite.szSize := 256;
    CFA_FileWrite(hFile := hFile, eError => eError);
    uiState := 3;
3:
    CFA_FileClose(hFile := hFile, eError => eError);
    uiState := 4;
4:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := FALSE;

    CFA_FileAccess.bWriteEnable := FALSE;
    CFA_FileAccess(eError => eError);
    uiState := 5;
5:
END_CASE
```

◆ サンプル2

USBデバイス上にあるファイルを読み込む場合

変数宣言部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
byBuffer: ARRAY[0..127] OF BYTE;
iIndex: INT;
szSize: WORD;
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FileRead: CONTEC_File_Access_Library.CFA_FileRead;
```

プログラム部

```
CASE uiState OF
  0:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := TRUE;
    CFA_FileAccess.bWriteEnable := FALSE;
    CFA_FileAccess(eError => eError);
    IF eError = 0 THEN
      uiState := 1;
    ELSE
      uiState := 5;
    END_IF

  1:
    CFA_FileOpen.eArea := eAreaType;
    CFA_FileOpen.sFileName := sFileName;
    CFA_FileOpen.sFileMode := 'r';
    CFA_FileOpen(hFile => hFile, eError => eError);

    IF eError = 0 THEN
      uiState := 2;
    ELSE
      uiState := 4;
    END_IF

  2:
    FOR iIndex := 0 TO 127 DO
      byBuffer[iIndex] := 0;
    END_FOR
    CFA_FileRead.pBuffer := ADR(byBuffer);
    CFA_FileRead.szBuffer := 128;
    CFA_FileRead(hFile := hFile, szSize => szSize, eError => eError);
```

```
IF eError = 0 THEN
  IF szSize = 0 THEN
    uiState := 3;
  END_IF
ELSE
  uiState := 3;
END_IF

3:
  CFA_FileClose(hFile := hFile, eError => eError);
  uiState := 4;

4:
  CFA_FileAccess.eArea := eAreaType;
  CFA_FileAccess.bMount := FALSE;
  CFA_FileAccess.bWriteEnable := FALSE;
  CFA_FileAccess(eError => eError);
  uiState := 5;

5:
END_CASE
```

◆ サンプル3

ファイルポインター移動関数を使用してファイルサイズを取得する場合

変数宣言部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
diFileSize: DINT := 0;
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FileSeek: CONTEC_File_Access_Library.CFA_FileSeek;
```

プログラム部

```
CASE uiState OF
0:
  CFA_FileAccess.eArea := eAreaType;
  CFA_FileAccess.bMount := TRUE;
  CFA_FileAccess.bWriteEnable := FALSE;
  CFA_FileAccess(eError => eError);
  IF eError = 0 THEN
    uiState := 1;
  ELSE
    uiState := 5;
  END_IF

1:
  CFA_FileOpen.eArea := eAreaType;
  CFA_FileOpen.sFileName := sFileName;
  CFA_FileOpen.sFileMode := 'r';
  CFA_FileOpen(hFile => hFile, eError => eError);
  IF eError = 0 THEN
    uiState := 2;
  ELSE
    uiState := 4;
  END_IF

2:
  CFA_FileSeek.diOffset := 0;
  CFA_FileSeek.eWhence := CONTEC_File_Access_Library.SEEKWHENCE.END;
  CFA_FileSeek(hFile := hFile, diPos => diFileSize, eError => eError);
  uiState := 3;

3:
  CFA_FileClose(hFile := hFile, eError => eError);
  uiState := 4;

4:
  CFA_FileAccess.eArea := eAreaType;
  CFA_FileAccess.bMount := FALSE;
  CFA_FileAccess.bWriteEnable := FALSE;
  CFA_FileAccess(eError => eError);
  uiState := 5;

5:
END_CASE
```

◆ サンプル4

USBデバイス上にあるファイルを削除する場合

変数宣言部

```
uiState: UINT := 0;  
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;  
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';  
eError: CONTEC_File_Access_Library.ERROR;  
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;  
CFA_FileDelete: CONTEC_File_Access_Library.CFA_FileDelete;
```

プログラム部

```
CASE uiState OF  
  0:  
    CFA_FileAccess.eArea := eAreaType;  
    CFA_FileAccess.bMount := TRUE;  
    CFA_FileAccess.bWriteEnable := TRUE;  
    CFA_FileAccess(eError => eError);  
    IF eError = 0 THEN  
      uiState := 1;  
    ELSE  
      uiState := 3;  
    END_IF  
  
  1:  
    CFA_FileDelete.eArea := eAreaType;  
    CFA_FileDelete.sFileName := sFileName;  
    CFA_FileDelete(eError => eError);  
    uiState := 2;  
  
  2:  
    CFA_FileAccess.eArea := eAreaType;  
    CFA_FileAccess.bMount := FALSE;  
    CFA_FileAccess.bWriteEnable := FALSE;  
    CFA_FileAccess(eError => eError);  
    uiState := 3;  
  
  3:  
END_CASE
```

◆ サンプル5

RAM上にあるファイルに対して1行単位で文字列を書いたり読み込んだりする場合

変数宣言部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.RAM;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample2.txt';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
sStringData1: STRING(256) := 'Entry1=1234567890';
sStringData2: STRING(256) := 'Entry2=abcdefghijklmnopqrstuvwxy';
sStringData3: STRING(256);
sSeparator: STRING(8) := '=';
sStringLeft: STRING(256);
sStringRight: STRING(256);
iIndex: INT;
szSize: WORD;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FilePutLine: CONTEC_File_Access_Library.CFA_FilePutLine;
CFA_FileGetLine: CONTEC_File_Access_Library.CFA_FileGetLine;
CFA_StringSeparate: CONTEC_File_Access_Library.CFA_StringSeparate
```

プログラム部

```
CASE uiState OF
  0:
    CFA_FileOpen.eArea := eAreaType;
    CFA_FileOpen.sFileName := sFileName;
    CFA_FileOpen.sFileMode := 'w';
    CFA_FileOpen(hFile => hFile, eError => eError);
    IF eError = 0 THEN
      uiState := 1;
    ELSE
      uiState := 7;
    END_IF

  1:
    CFA_FilePutLine.pszString := ADR(sStringData1);
    CFA_FilePutLine(hFile := hFile, eError => eError);
    IF eError = 0 THEN
      uiState := 2;
    ELSE
      uiState := 3;
    END_IF

  2:
    CFA_FilePutLine.pszString := ADR(sStringData2);
    CFA_FilePutLine(hFile := hFile, eError => eError);
    uiState := 3;
```

```
3:
  CFA_FileClose(hFile := hFile, eError => eError);
  uiState := 4;

4:
  CFA_FileOpen.eArea := eAreaType;
  CFA_FileOpen.sFileName := sFileName;
  CFA_FileOpen.sFileMode := 'r';
  CFA_FileOpen(hFile => hFile, eError => eError);
  IF eError = 0 THEN
    uiState := 5;
  ELSE
    uiState := 7;
  END_IF

5:
  FOR iIndex := 0 TO 256 DO
    sStringData3[iIndex] := 0;
    sStringLeft[iIndex] := 0;
    sStringRight[iIndex] := 0;
  END_FOR
  CFA_FileGetLine.pszString := ADR(sStringData3);
  CFA_FileGetLine.szString := 256;
  CFA_FileGetLine(hFile := hFile, szSize => szSize, eError => eError);

  IF eError = 0 THEN
    IF szSize <> 0 THEN
      CFA_StringSeparate.pszString := ADR(sStringData3);
      CFA_StringSeparate.pszSeparator := ADR(sSeparator);
      CFA_StringSeparate.pszLeft := ADR(sStringLeft);
      CFA_StringSeparate.pszRight := ADR(sStringRight);
      CFA_StringSeparate(eError => eError);
    ELSE
      uiState := 6;
    END_IF
  ELSE
    uiState := 6;
  END_IF

6:
  CFA_FileClose(hFile := hFile, eError => eError);
  uiState := 7;

7:
  END_CASE
```

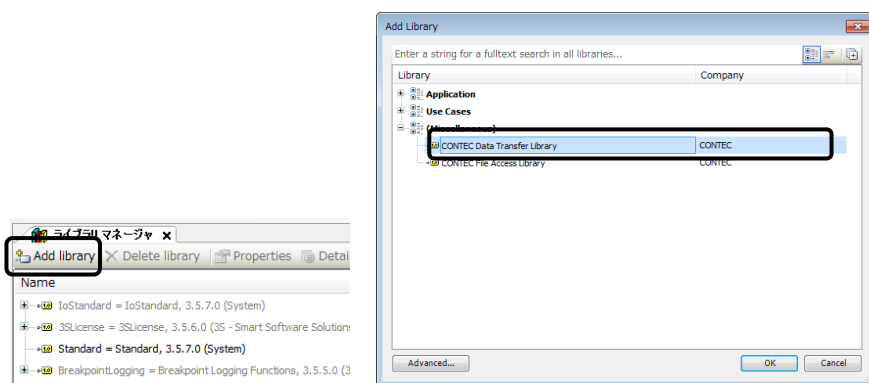
10. クラウドサービスヘータを送信する

本製品で収集したデータをクラウドサービスに送信するための設定を行います。

データはCSV形式ノカンマ区切りで保存されます。

1. データ送信ライブラリの設定

- 1 新規CODESYSプロジェクトを作成します。
IECプログラム言語はSTを選択します。
※プロジェクトの作成方法は『**新規プロジェクトの生成(P56)**』を参照ください。
- 2 デバイスウィンドウの [ライブラリマネージャ] をダブルクリックします。
- 3 [Add Library] をクリックし、『(Miscellaneous)』のツリー階層内の [CONTEC Data Transfer Library] を選択して [OK] をクリックします。



2. 関数一覧

関数名	機能
CDT_FileSetValue	データファイルへのデータセット
CDT_FileDelete	データファイルの削除
CDT_FileSetParameter	データファイル用のパラメーターセット
CDT_CloudSend	クラウド送信
CDT_CloudSetParameter	クラウド送信用のパラメーターセット
CDT_CloudGetInformation	クラウド送信用の情報取得
CDT_CloudControl	クラウド送信用の制御

3. データタイプ

◆ ERROR

名前	値	意味
NO_ERROR	0	正常終了
INVALID_PARAM	1	パラメーターが不正です。
INTERNAL_ERROR	2	リソース不足です。
TX_NODATA	3	データファイルが存在しない、もしくは、データファイルにデータが存在しません。
TX_PENDING	4	短い時間でクラウド送信関数を呼び出したため、データ送信が行われませんでした。
TX_ERROR	5	データ送信でエラーが返されたため、データファイルを削除しました。
TX_RESEND	6	データ送信に失敗したため、再送信の待ち行列にセットされました。

◆ DATATYPE

名前	値	意味
DATETIME	0	SYSTIME型の時刻データ。
CRLF	1	1行の最後を示す改行文字。
VAL_STRING	2	STRING型データ。
VAL_BOOL	3	BOOL型データ。メモリ領域のサイズは8ビットです。
VAL_BYTE	4	符号無し8ビットデータ。
VAL_WORD	5	符号無し16ビットデータ。
VAL_DWORD	6	符号無し32ビットデータ。
VAL_LWORD	7	符号無し64ビットデータ。
VAL_SINT	8	符号付き8ビットデータ。
VAL_INT	9	符号付き16ビットデータ。
VAL_DINT	10	符号付き32ビットデータ。
VAL_LINT	11	符号付き64ビットデータ。
VAL_REAL	12	単精度浮動小数点数データ。
VAL_LREAL	13	倍精度浮動小数点数データ。

◆ FILEPARAM

名前	値	意味
MAX_SIZE	0	1Kバイト単位のデータファイルの最大サイズ。
TIME_FORMAT	1	時刻データのフォーマットタイプ。 0 : YYYYMMDD,hhmmss,ミリ秒 1 : YYYYMMDDhhmmss 2 : YYYYMMDDhhmm 3 : YYYYMMDD 4 : hhmmss 5 : ミリ秒

◆ CLOUDPARAM

名前	値	意味
RETRY_TIME	0	再送信待ちが発生した後、データの再送信を行う時間(秒)。
RETRY_NUM	1	再送信待ちファイルを保持する個数

◆ CLOUDINFO

名前	値	意味
RETRY_NUM	0	再送信待ちファイル数。

◆ CLOUDCTRL

名前	値	意味
RETRY_CLR	0	再送信待ちファイルのクリア。

4. 関数詳細

◆ CDT_FileSetValue

機能

サーバーに送信するデータファイルを作成します。
データはCSV形式のカンマ区切りで保存されます。

入力値

名前	型	意味
iFileNo	INT	ファイル番号(0 - 9)を指定します。
eType	DATATYPE	データファイルにセットするデータのタイプを指定します。
pValue	POINTER TO BYTE	セットするデータのアドレス。 時刻データは、SysTimeRtcHighResGet()で取得した値をセットします。 0を指定した場合は、関数内部で時刻を取得してセットします。 改行文字の場合は、0を指定してください。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CDT_FileDelete

機能

作成中のデータファイルを削除します。

入力値

名前	型	意味
iFileNo	INT	ファイル番号(0 - 9)を指定します。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CDT_FileSetParameter

機能

データファイルに関するパラメーターをセットします。

入力値

名前	型	意味
iFileNo	INT	ファイル番号(0 - 9)を指定します。
eParamNo	FILEPARAM	パラメーター番号を指定します。 (MAX_SIZE) 1Kバイト単位のデータファイルの最大サイズを指定します。最大サイズをオーバーした場合は、古いデータが削除されます。 0を指定した場合は、サイズの制限はありません。 データ : WORD型 デフォルト設定 : 0 (TIME_FORMAT) 時刻データのフォーマットをセットします。 データ : INT型 デフォルト設定 : 1
pValue	POINTER TO BYTE	セットするデータのアドレス。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CDT_CloudSend

機能

作成したデータファイルをクラウドサービスに送信します。

送信したデータファイルは削除されます。

入力値

名前	型	意味
iFileNo	INT	ファイル番号(0 - 9)を指定します。
xExecute	BOOL	実行状態を指定します。FALSEからTRUEに変化した時に送信処理を開始します。

出力値

名前	型	意味
xDone	BOOL	正常終了。
xBusy	BOOL	実行中。
xError	BOOL	エラー終了。
eError	ERROR	エラーコード。

◆ CDT_CloudSetParameter

機能

クラウド送信に関するパラメーターをセットします。

入力値

名前	型	意味
eParamNo	CLOUDPARAM	パラメーター番号を指定します。 (RETRY_TIME) 再送信待ちが発生した後、データの再送信を行う時間(秒)をセットします。 0を指定した場合は、クラウド送信関数を実行した時に再送信待ちファイルを送信します。 データ : WORD型 デフォルト設定 : 0 (RETRY_NUM) 再送信待ちファイルを保持する個数をセットします。0を指定した場合、再送信待ちファイルは保持しません。 データ : DWORD型 デフォルト設定 : FFFFFFFh
pValue	POINTER TO BYTE	セットする値のアドレス。

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CDT_CloudGetInformation

機能

クラウド送信に関する情報を取得します。

入力値

名前	型	意味
eInfoNo	CLOUDINFO	情報番号を指定します。 (RETRY_NUM) 再送信待ちファイル数を取得します。 データ : DWORD型

出力値

名前	型	意味
eError	ERROR	エラーコード。

◆ CDT_CloudControl

機能

クラウド送信に関する制御を行います。

入力値

名前	型	意味
eCtrlNo	CLOUDCTRL	制御番号を指定します。 (RETRY_CLR) 再送信待ちファイルをクリアします。 データ : 無し
pValue	POINTER TO BYTE	制御情報を格納する領域のアドレス。

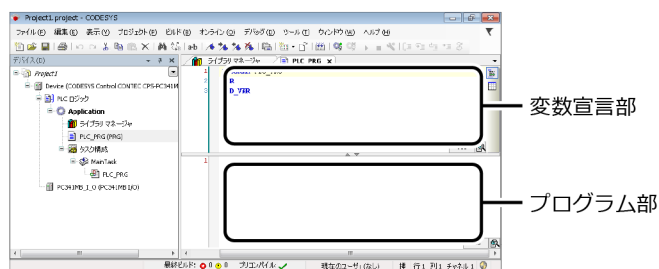
出力値

名前	型	意味
eError	ERROR	エラーコード。

5. サンプルプログラム

以下にサンプルプログラムの実行手順を示します。

- 1 デバイスウィンドウ内の [PLC_PRG(PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。
- 2 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に変数宣言部のソースコードを記述します。
- 3 『STエディタウィンドウ』の『プログラム部』にプログラム部のソースコードを記述します。



◆ サンプル1

CPS-DIO-0808RLのデジタル入力データとCPS-SSI-4Pの温度データをクラウドサービスに送信する場合
※ このサンプルプログラムは、スタックタイプのコントローラにCPS-DIO-0808RLとCPS-SSI-4Pを接続した状態で行います。

変数宣言部

```
DiData: ARRAY[0..7] OF BOOL;  
TempData: ARRAY[0..3] OF LREAL;  
dwIntervalTime: DWORD := 100;  
dwTimerCount1: DWORD;  
dwTimerCount2: DWORD;  
uiState: UINT := 0;  
uiCloudTxState: UINT;  
iFileNo: INT := 0;  
usFileSize: WORD := 8;  
iTimeFormat: INT := 1;  
dwCounter: DWORD;  
iIndex: INT;  
eError: CONTEC_Data_Transfer_Library.ERROR;  
eCloudTxError: CONTEC_Data_Transfer_Library.ERROR;  
CDT_FileSetValue: CONTEC_Data_Transfer_Library.CDT_FileSetValue;  
CDT_FileSetParameter: CONTEC_Data_Transfer_Library.CDT_FileSetParameter;  
CDT_CloudSend: CONTEC_Data_Transfer_Library.CDT_CloudSend;  
CDT_CloudSetParameter: CONTEC_Data_Transfer_Library.CDT_CloudSetParameter;  
CDT_CloudGetInformation: CONTEC_Data_Transfer_Library.CDT_CloudGetInformation;
```

プログラム部

```

CASE uiState OF
0:
  CDT_FileSetParameter.eParamNo := CONTEC_Data_Transfer_Library.FILEPARAM.MAX_SIZE;
  CDT_FileSetParameter.pValue := ADR(usFileSize);
  CDT_FileSetParameter(iFileNo := iFileNo, eError => eError);

  CDT_FileSetParameter.eParamNo :=
    CONTEC_Data_Transfer_Library.FILEPARAM.TIME_FORMAT;
  CDT_FileSetParameter.pValue := ADR(iTimeFormat);
  CDT_FileSetParameter(iFileNo := iFileNo, eError => eError);
  dwTimerCount1 := 0;
  dwTimerCount2 := 0;
  wCounter := 0;
  uiCloudTxState := 0;
  uiState := 1;

1:
  DO_BYTE := DO_BYTE + 1;
  DiData[0] := DI_BIT0;
  DiData[1] := DI_BIT1;
  DiData[2] := DI_BIT2;
  DiData[3] := DI_BIT3;
  DiData[4] := DI_BIT4;
  DiData[5] := DI_BIT5;
  DiData[6] := DI_BIT6;
  DiData[7] := DI_BIT7;
  TempData[0] := TEMP0;
  TempData[1] := TEMP1;
  TempData[2] := TEMP2;
  TempData[3] := TEMP3;

  dwTimerCount1 := dwTimerCount1 + dwIntervalTime;
  IF dwTimerCount1 >= 60000 THEN
    dwTimerCount1 := 0;

    CDT_FileSetValue.iFileNo := iFileNo;
    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.DATETIME;
    CDT_FileSetValue.pValue := 0;
    CDT_FileSetValue(eError => eError);

    dwCounter := dwCounter + 1;
    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.VAL_DWORD;
    CDT_FileSetValue.pValue := ADR(dwCounter);
    CDT_FileSetValue(eError => eError);

    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.VAL_BOOL;
    FOR iIndex := 0 TO 7 DO
      CDT_FileSetValue.pValue := ADR(DiData[iIndex]);
      CDT_FileSetValue(eError => eError);
    END_FOR

    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.VAL_LREAL;

    FOR iIndex := 0 TO 3 DO
      CDT_FileSetValue.pValue := ADR(TempData[iIndex]);
      CDT_FileSetValue(eError => eError);

```

```
END_FOR

CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.CRLF;
CDT_FileSetValue.pValue := 0;
CDT_FileSetValue(eError => eError);
END_IF

dwTimerCount2 := dwTimerCount2 + dwIntervalTime;

CASE uiCloudTxState OF
0:
    IF dwTimerCount2 >= 600000 THEN
        dwTimerCount2 := 0;
        CDT_CloudSend.iFileNo := iFileNo;
        CDT_CloudSend(xExecute := FALSE);
        CDT_CloudSend(xExecute := TRUE);
        uiCloudTxState := 1;
    END_IF
1:
    CDT_CloudSend();
    IF CDT_CloudSend.xBusy = FALSE THEN
        IF CDT_CloudSend.xDone OR CDT_CloudSend.xError THEN
            eCloudTxError := CDT_CloudSend.eError;
            uiCloudTxState := 0;
        END_IF
    END_IF
END_CASE
END_CASE
```

◆ サンプル 2

アプリケーションで取得した時刻データをセットする場合

SysTimeRtcHighResGet()は、[ライブラリマネージャ] の [System] - [SysLibs] にある [SysTime] を追加する必要があります。

変数宣言部

```
uiState: UINT := 0;  
iFileNo: INT := 0;  
TimeStamp: SYSTIME;  
eError: CONTEC_Data_Transfer_Library.ERROR;  
CDT_FileSetValue: CONTEC_Data_Transfer_Library.CDT_FileSetValue;
```

プログラム部

```
CASE uiState OF  
  0:  
    SysTimeRtcHighResGet(TimeStamp);  
  
    CDT_FileSetValue.iFileNo := iFileNo;  
    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.DATETIME;  
    CDT_FileSetValue.pValue := ADR(TimeStamp);  
    CDT_FileSetValue(eError => eError);  
  
    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.CRLF;  
    CDT_FileSetValue.pValue := 0;  
    CDT_FileSetValue(eError => eError);  
    uiState := 1;  
  
  1:  
END_CASE
```

モニタリング編集をする

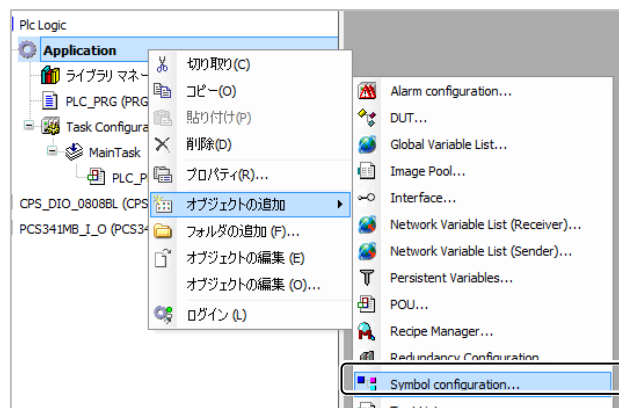
Webブラウザ上で簡単にモニタリング画面を作成し、稼働状況、故障の監視や運転、停止などをモニタリングできるCONPROSYS HMI(Human Machine Interface)の説明をしています。

1. CONPROSYS HMIを使用する

本製品でモニタリングを行う場合、付属の『CONPROSYS HMI』を使用して、モニタリング画面の制作ができます。

1. CODESYSの設定

- 1 『基本的なプログラミング手順(P56)』の『新規プロジェクトの生成(P56)』と『CODESYSからコントローラに接続する(P57)』手順でCODESYSのプロジェクトを作成します。
- 2 デバイスウィンドウに表示されている [Application] を右クリックして、[オブジェクトの追加] - [Symbol configuration] を選択します。



- 3 『Symbol configurationの追加』ダイアログで [追加] ボタンをクリックします。

2. プログラムの作成とビルド

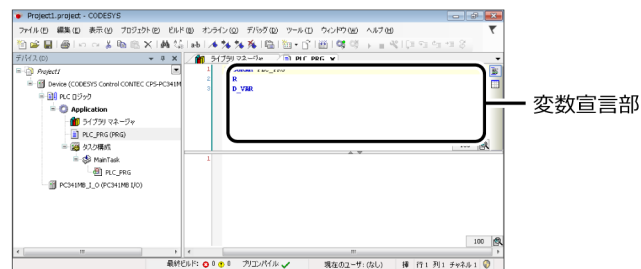
ここでは例として、ハードウェアのI/O機能を使用せずにソフトウェア変数のみのCODESYSプログラムを作成します。

Incrementerはサイクルタイム毎にカウントアップを行う変数です。

Switch変数で動作の開始/停止を制御し、Status変数は現在のステータスを文字列で保持します。

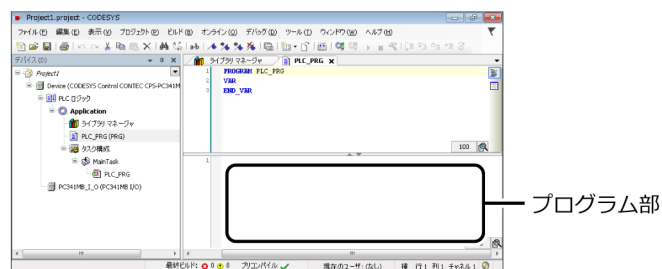
- 1 デバイスウィンドウ内の [PLC_PRG(PRG)] アイコンをダブルクリックします。
『STエディタウィンドウ』が表示されます。

- 2 『STエディタウィンドウ』の『変数宣言部』のVARとEND_VARの間に以下のソースコードを記述します。



```
Switch: BOOL;  
Incrementer: INT;  
Status: STRING[10];
```

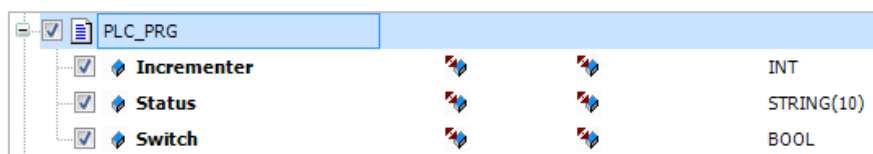
- 3 『STエディタウィンドウ』の『プログラム部』に以下のソースコードを記述します。



```
IF Switch = TRUE THEN  
  Incrementer := Incrementer + 1;  
  IF Incrementer = 100 THEN  
    Incrementer := 0;  
  END_IF  
  Status := 'BUSY';  
ELSE  
  Status := 'STOP';  
END_IF
```

- 4 CODESYSの[ビルド]メニューから[リビルド]を実行し、ビルドプロセスが正常に完了するかどうかを確認します。

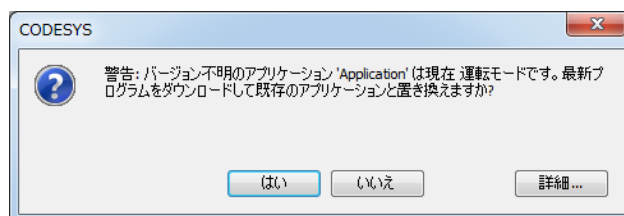
- 5** デバイスウィンドウ内の『Symbol configuration』をダブルクリックし、PLC_PRGのツリー階層内の『Incrementer』、『Status』、『Switch』変数にチェックを入れます。



※ Symbol configurationの『IoConfig_Globals_Mapping』または『PLC_PRG』のツリー階層内に登録される変数のみが、CONPROSYS HMIで使用できます。
その他のツリーに登録されている変数については、CONPROSYS HMIでは無視されます。

- 6** CODESYSの『オンライン』メニューから『ログイン』を実行します。

※ ダウンロードの確認ダイアログが表示されますので『はい』をクリックします。



3. HMI Editorで画面を作成する

- 1** コントローラを接続したパソコンでWebブラウザを起動し、アドレス入力部に「http://10.1.1.101/」(PLCコントローラのIPアドレス)を入力します。

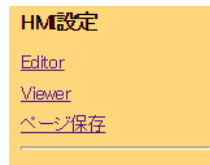
※ コントローラ接続の詳細は『パソコンのネットワーク設定をする(P18)』を参照ください。

※ 「https://10.1.1.101/」でもアクセス可能です。『証明書エラー』画面が表示される場合は、[このサイトの閲覧を続行する]を選択してください。

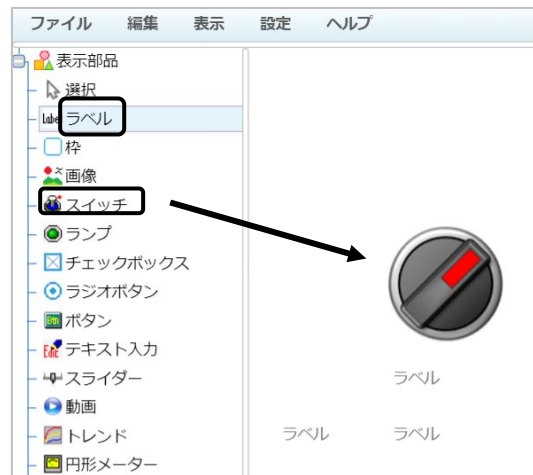
- 2** ユーザー名に「pc341」、パスワードに「pc341」と入力してログインします。



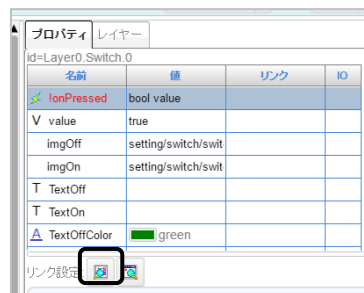
- 3 『HMI設定』の[Editor]を選択してCONPROSYS HMIを起動します。



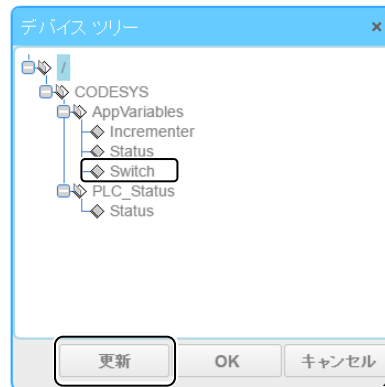
- 4 『表示コントロール選択エリア』から、1つの『スイッチ』と、3つの『ラベル』をドラッグ&ドロップでレイヤーエリアに配置します。



- 5 スイッチをクリックして選択状態にします。
次にエディタ画面右側のプロパティにある[リンク設定]アイコンをクリックします。



- 6** デバイスツリーで「更新」ボタンを押すと、CODESYSで使用されている変数一覧が更新されます。
「Switch」を選択して「OK」をクリックします。
スイッチ部品に対して変数Switchが割り当てられます。



- 7** 手順5、6と同様に2つのラベルに対してIncrementer変数とStatus変数を割り当てます。
もう1つのラベルには『Status』という文字を表示するために、プロパティのText部に「Status」と入力します。

4. HMI Viewerの実行

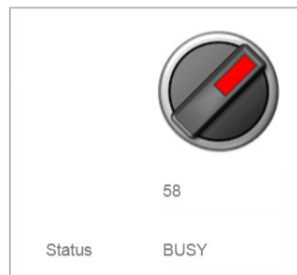
- 1 画面右上のモードを「実行」にします。



- 2 CODESYSの「デバッグ」メニューから「運転」を選択します。

- 3 キャンバスに配置したスイッチをONにすると、Incrementerの数値が0 - 100の範囲で更新されます。スイッチをOFFにすると、Incrementerの更新が停止します。

また、Status変数には1(PLC動作中)が表示されており、CODESYS側で動作を停止させると値が0に変更されます。



- 4 Editor内で動作させるのではなく、WebブラウザでViewer画面を実行します。

はじめに「ファイル」 - 「ページを保存」で作成したHMI画面を保存します。

保存場所はuserフォルダの下、保存名はPage1としてください。

次に、「ファイル」 - 「Viewerで実行」を選択すると、新しいタブが開きViewer画面が実行されます。

※ 保存した画面はコントローラを再起動すると失われます。ページのROM領域への保存は、以下の2種類の方法で行ってください。

- 「ファイル」 - 「設定をROMに保存」をクリックします。
- CONPROSYS WEB SettingのHMI設定で「ページ保存」をクリックします。

5. IECデータタイプで利用できる部品

CONPROSYS HMIでは以下のIECデータタイプが使用できます。

- BOOL
- DWORD
- INT
- STRING
- BYTE
- UDINT
- DINT
- USINT
- LWORD
- LINT
- WORD
- ULINT
- REAL
- UINT
- SINT
- LREAL

各部品で使用可能なIECデータタイプ

部品名	BOOL型	STRING型	それ以外
ラベル	○	○	○
枠	×	×	×
画像	×	×	×
スイッチ	○	×	×
ランプ	○	×	×
チェックボックス	○	×	×
ラジオボタン	×	×	×
ボタン	×	×	×
テキスト入力	○	○	○
スライダー	○	×	○
動画	×	×	×
トレンド	○	×	○
円形メーター	○	×	○
レベルメーター	○	×	○
円グラフ	○	×	○
トレンドバー	○	×	○
オンディレイ・スイッチ	○	×	×
マルチステートランプ	○	×	○

2. CONPROSYS HMIの概要

1. CONPROSYS HMIでできること

用意された表示部品を並べて、入力されている信号の状態をモニタリングできます。

Webブラウザだけで画面を作成することができ、プログラム言語の知識や特別な開発環境は必要ありません。

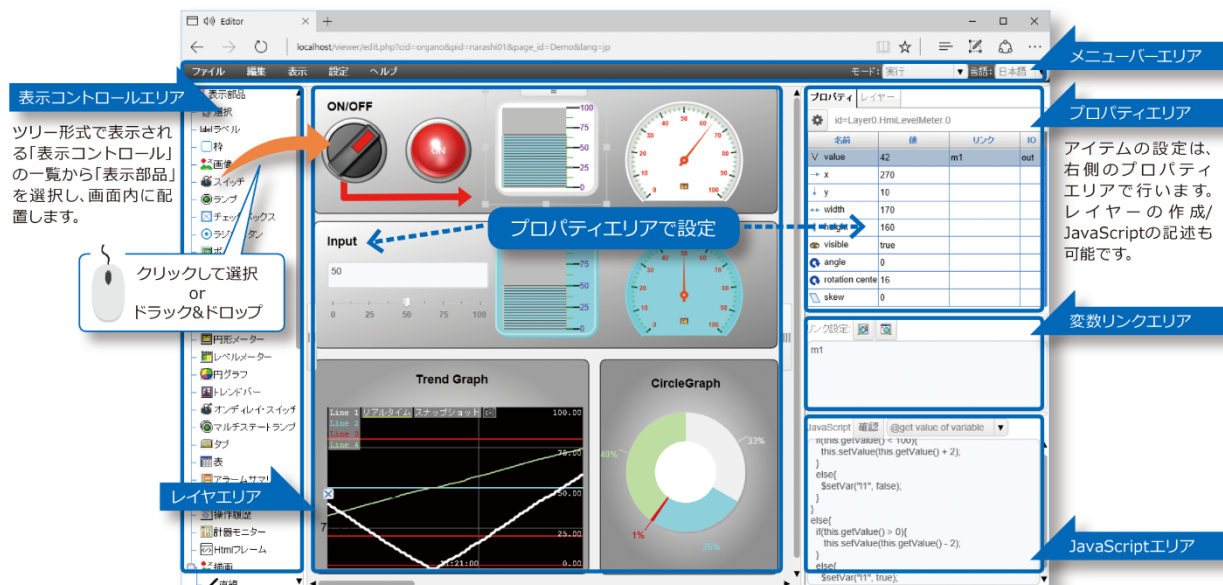
数多く用意されている表示部品をドラッグ&ドロップで作画していくことができます。

また、表示部品の設定やセンサーなどのデータのリンクは、プロパティ画面で設定できます。



設定 - 保存 - 実行がWebブラウザで完結

マルチプラットフォームで利用可能です。



上記画面はモニタリング画面を作成した際のイメージです。
出荷時の設定は、白色の背景が表示されます。

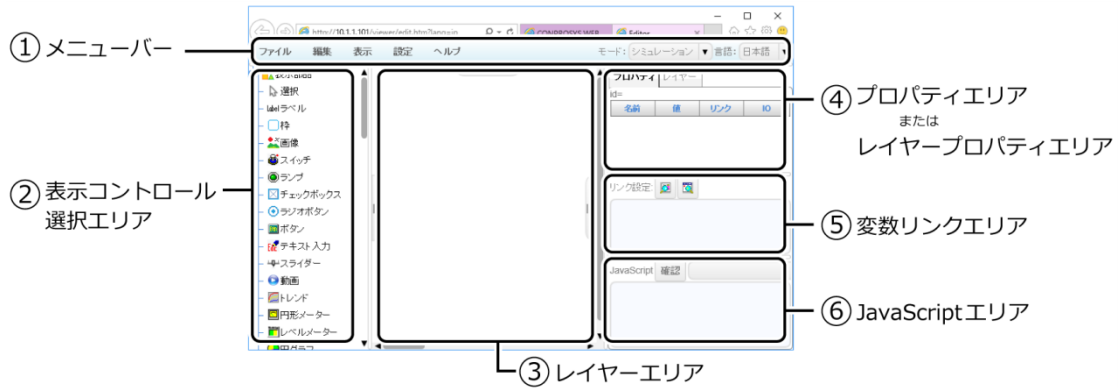
CONPROSYS VTCとのデータ連携

内部変数 (TAG) を用いて CONPROSYS VTC との連携が可能です。VTC で処理を行った結果を表示したり、HMI の操作を VTC のフロー制御へ反映させることができます。



2. 操作エリア

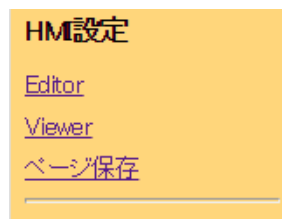
CONPROSYS HMIには以下の操作エリアがあります。



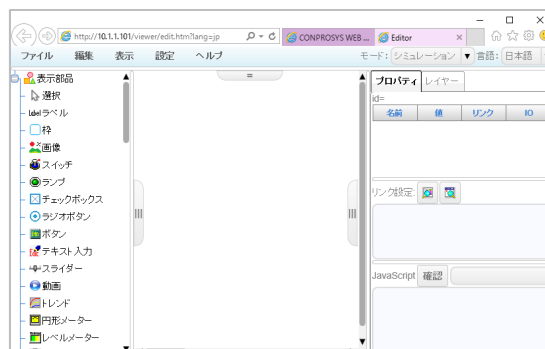
No.	名称	機能
①	メニューバー	メニューボタン形式でコマンドを実行します。
②	表示コントロール選択エリア	配置したい表示コントロールを選択できます。
③	レイヤーエリア	画面のデザインを行うエリアです。
④	プロパティエリア	表示コントロールのプロパティ変数を変更できます。
	レイヤープロパティエリア	レイヤーの追加、削除、またレイヤーのプロパティを変更できます。
⑤	変数リンクエリア	表示コントロールのプロパティと変数リンクを行います。
⑥	JavaScriptエリア	表示コントロールに対し、JavaScriptによる挙動を設定します。

3. モニタリング画面の作成

CONPROSYS WEB Settingの「HMI設定」より「Editor」をクリックします。



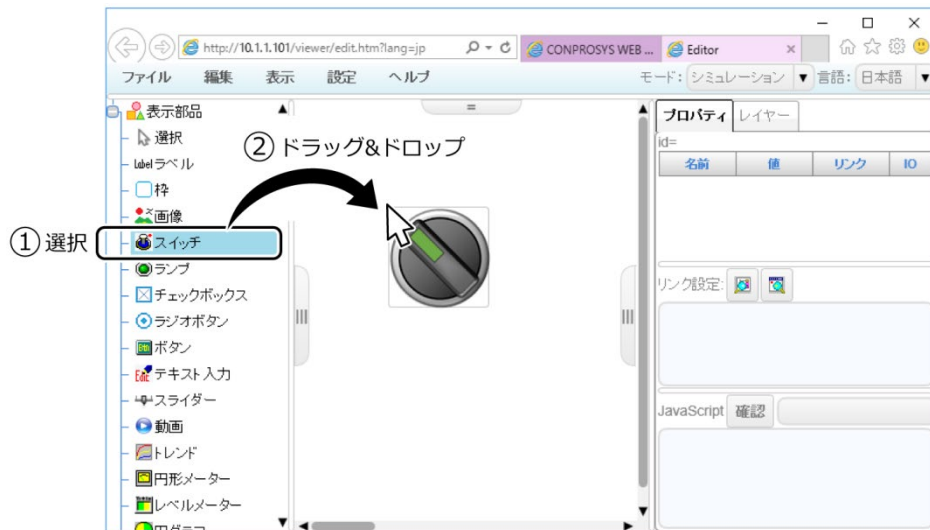
CONPROSYS HMIが起動し、モニタリング画面を作成していくことができます。



4. モニタリング画面作成の基本操作

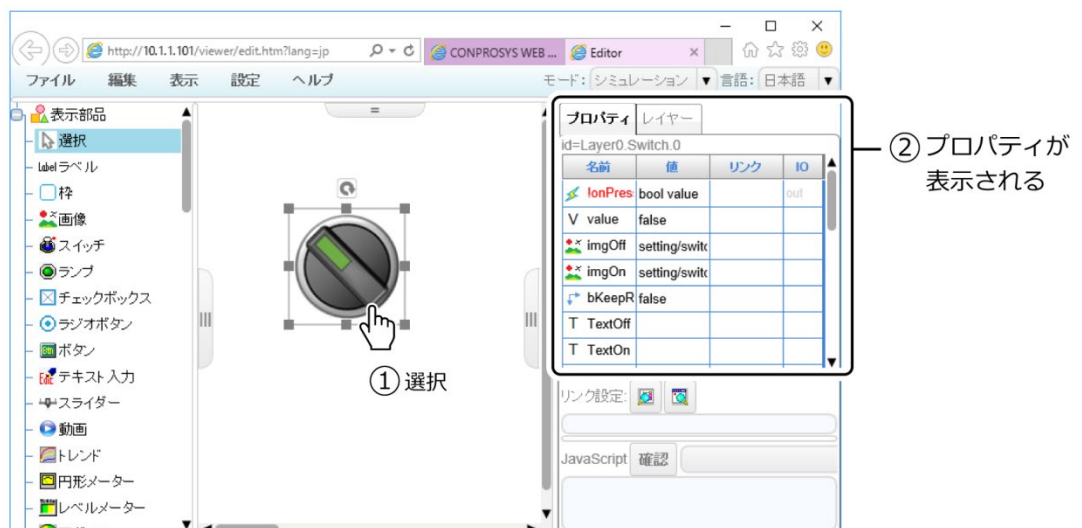
◆ 表示部品を配置する

①表示コントロール選択エリアのツリーから表示部品を選択し、②レイヤーエリアにドラッグ&ドロップで配置します。



◆ プロパティを設定する

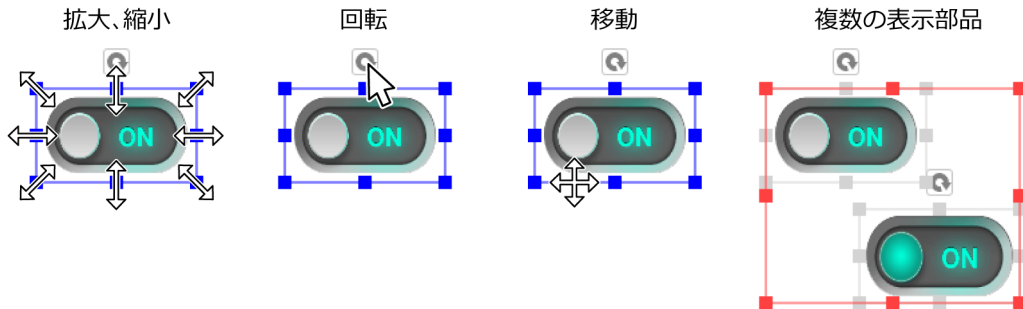
①配置した表示部品を選択すると、②プロパティエリアにその表示部品のプロパティが表示されます。
プロパティエリアでは、値を直接変更したり、入出力デバイスや他の部品とリンクするためのデータを設定したりすることができます。



◆ 表示部品の位置とサイズを調整する

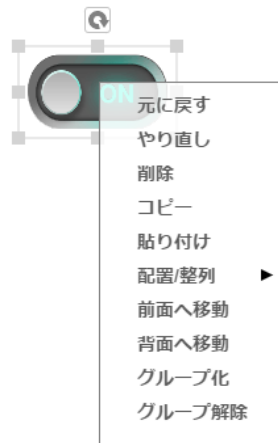
配置した表示部品を左クリックして選択した状態で、表示された枠をドラッグすると位置やサイズ、角度の調整が行えます。

同時に複数の部品を一度に選択し、位置、サイズを調整することもできます。



◆ 表示部品のコピーや削除

配置した表示部品を選択した状態で右クリックすると、編集メニューが表示され表示部品のコピーや削除等を行えます。

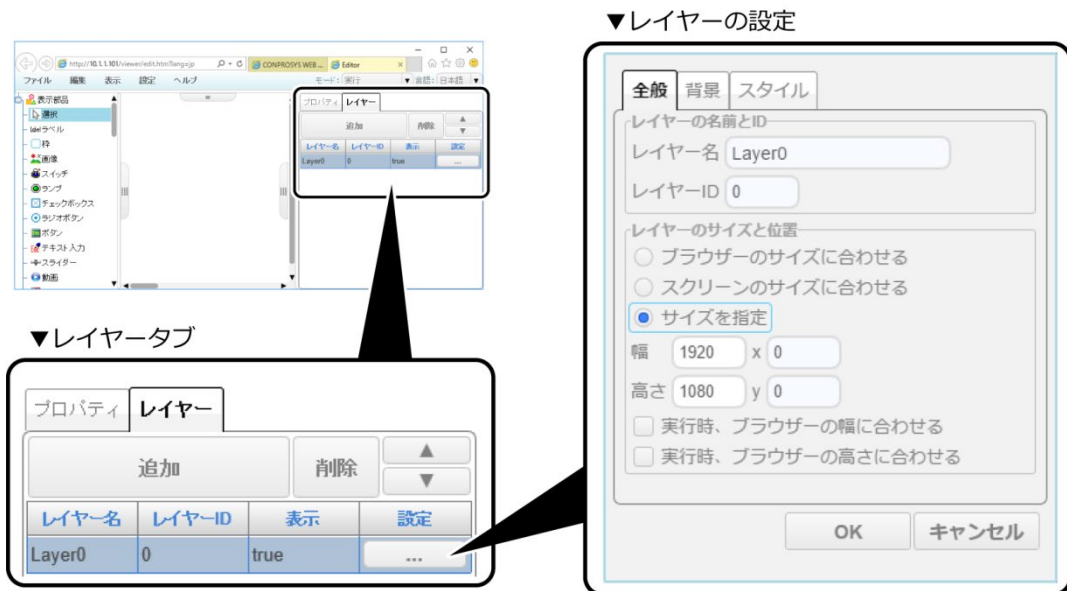


◆ レイヤーを設定する

画面右上から「レイヤー」タブを選択すると、レイヤープロパティエリアが表示されます。

レイヤープロパティエリア「設定」列の「...」ボタンをクリックすると、設定ダイアログが表示されます。

モニタリング画面のサイズや背景等を設定することができます。

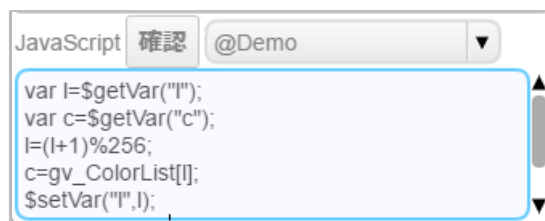


◆ Javascriptを記述する

必要に応じて、Javascriptを記述します。

独自の制御ロジックが必要な場合には、JavaScriptエリアにJavascriptを記述して動作させることもできます。

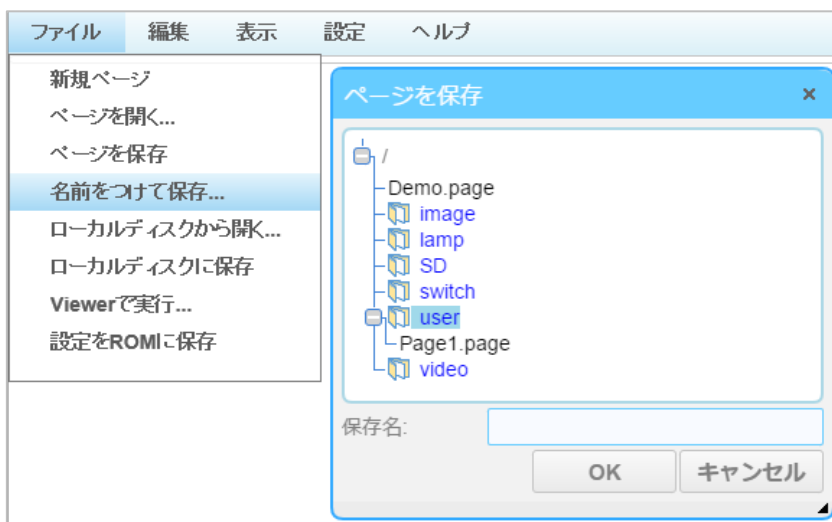
各表示部品で使用可能なJavascriptの関数については、『[オンラインヘルプ](#)』を参照ください。



◆ モニタリング画面を保存する

モニタリング画面の作成後、ファイルメニューから任意の名前を付けて画面を保存することができます。
保存後、電源OFFする前にメニューの［ファイル］－［設定をROMに保存］を行ってください。

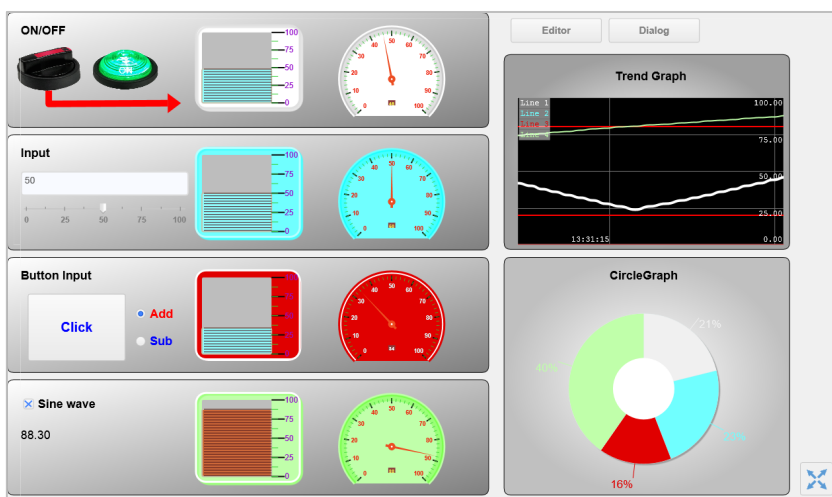
※［設定をROMに保存］を行わないと、再起動や電源をOFFにすることで作成したモニタリング画面が元の状態に戻ります。



◆ モニタリング画面を表示する

CONPROSYS WEB Settingの［ステータスメニュー］－［モニタリング表示］をクリックするとモニタリング画面が表示できます。

既定では「user/Page1.page」に保存した画面が表示されます。



※ 別の名前を付けた画面を表示する場合は、Webブラウザで以下のURLを指定してください。

**http://〈IPアドレス〉:〈ポート番号〉/viewer/view.htm?pagepath=〈ページファイルパス〉
&lang=〈表示言語〉**

〈ページファイルパス〉: ページのファイル名を指定します。例: /user/Page1.page

〈表示言語〉: 表示する言語名を指定します。例: 日本語はjpです。言語指定は省略することができます。

3. 表示部品一覧

CONPROSYS HMIのモニタリング編集で利用できる表示部品は以下のとおりです。

表示部品	名称	内容
	ラベル	文字列を表示できます。
	枠	タイトル表示付きの枠です。
	画像	画像を表示できます。
	スイッチ	ON/OFF状態を出力できるスイッチです。
	ランプ	ON/OFF状態を表示できるランプです。
	チェックボックス	ON/OFF状態を出力でき、文字列の表示ができるチェックボックスです。
	ラジオボタン	複数の条件から一つだけを選択する、ラジオボタンです。
	ボタン	文字列の表示、クリックができるボタンです。
	テキスト入力	文字列の入力、表示ができます。
	スライダー	スライダーによるデータ出力が行えます。
	動画	動画を再生できます。
	トレンド	時系列データをグラフ表示できます。
	円形メーター	データを円形メーター形式で表示できます。
	レベルメーター	データをレベルメーター形式で表示できます。
	円グラフ	データを円グラフ形式で表示できます。
	トレンドバー	データをトレンドまたはバー形式で表示できます。
	オンディレイ・スイッチ	押された一定の時間後にON/OFF状態を出力できるスイッチです。
	マルチステートランプ	複数の異なる状態値を表示することができるランプです。
	タブ	複数のタブを作成し、切り替えて表示することができます。
	表	データを表形式で表示、編集できます。
	Htmlフレーム	別のHtml文書をフレーム内に表示することができます。
	リスト	値をリスト形式で表示するコントロールです。
	直線	画面上に直線を描画できます。
	ポリライン	画面上にポリライン(連続直線)を描画できます。
	曲線	画面上に曲線を描画できます。
	四角形	画面上に四角形を描画できます。
	角丸四角形	画面上に角が丸い四角形を描画できます。
	多角形	画面上に多角形を描画できます。
	円	画面上に円を描画できます。
	パイプ	画面上にパイプ風の連続線を描画できます。

CONPROSYS HMIの操作や機能の詳細は『**オンラインヘルプ**』を参照ください。

オンラインヘルプ

<http://data.conprosys.com/help/hmi/V1/jp/>

困った時は

故障や各トラブルの原因と対処方法、確認方法の説明をしています。

1. 故障かな？と思った時は

ご使用に際して不具合が発生した場合は、以下の手順で確認してください。

1. 全般

◆ 前面のLEDを確認します。

- PWRのLEDが点灯していることを確認してください。
- ST1のLEDが点滅していることを確認してください。

◆ ネットワークポートのLEDを確認します。

前面のUTPコネクタのLEDを確認してください。

ネットワークケーブルがHUBに正しく接続されていれば、Link/Act LEDが点灯します。

点灯していない場合は、『リファレンスマニュアル(ハードウェア編)』を参照して確認を行ってください。

ネットワークポートで通信を行うとLink/Act LEDが点滅します。

◆ ホストコンピュータからPINGコマンドで応答があるか確認します。

本機器のIPアドレスに対してPINGを発行します。

機器が動作している状態であれば応答します。

例) 機器にIPアドレスを10.1.1.101に設定した場合

```
ping 10.1.1.101<Enter> :
```

```
Reply from 10.1.1.101: bytes=32 time<10ms TTL=255
```

```
Reply from 10.1.1.101: bytes=32 time<10ms TTL=255
```

```
Reply from 10.1.1.101: bytes=32 time<10ms TTL=255
```

応答が表示されます。

設定したIPアドレスがわからなくなった場合は、SW1-2のスイッチをON(左)に設定し電源を入れることで、工場出荷時の設定(IPアドレス : 10.1.1.101)で起動することができます。

⚠ 注意

スイッチを元に戻すと、ROMに保存された元の設定で起動します。

◆ ホストコンピュータのブラウザからの接続でユーザー名、パスワードの認証に失敗する場合

ユーザー名、パスワードは、大文字と小文字を別の文字として、判断しています。「Caps Lock」が入っていないか確認して、再度、入力してください。

設定したユーザー名、パスワードがわからなくなった場合は、SW1-2のスイッチをON(左)に設定し電源を入れることで、工場出荷時の設定で起動することができます。

(IPアドレスも工場出荷時の設定で起動します)

⚠ 注意

設定した他の項目も初期化されます。

◆ PINGコマンドでは、応答があるが、ブラウザから接続すると「ページを表示できません」と表示されてしまう場合

ブラウザの設定を下記のように設定してください。

プロキシサーバーの設定

「プロキシサーバーを使用しない」設定にしてください。

ダイヤルアップの設定

「ダイヤルしない」設定にしてください。

◆ 正常に起動しない場合

当社に交換修理依頼してください。

仕様

ハードウェア仕様およびCONPROSYS HMI仕様について説明しています。

1. ハードウェア仕様

1. CPS-PC341EC-1-9201の仕様

項目		CPS-PC341EC-1-9201
CODESYS対応機能	バージョン	V3.5 SP7 Patch2 以上
	言語	LD, SFC, FBD, ST, IL, CFC (IEC61131-3準拠)
	フィールドバス	EtherCAT Master、Modbus TCP Slave
	通信プロトコル	OPC-UA Server
プログラムサイズ	ROMサイズ	1MB
	最大ステップ数	250K ステップ
CPU基本性能	基本命令実行速度(LD)	1.6ns
	応用命令実行速度(ST)	5.8ns
	スキャン時間	74μs (20000 ステップ時)
EtherCAT性能	入力処理時間(LD)	144ns
	出力処理時間(ST)	138ns
	スキャン時間	166μs (64点入力、64出力時)

2. CPS-PC341MB-ADSC1-9201の仕様

項目		CPS-PC341MB-ADSC1-9201
CODESYS対応機能	バージョン	V3.5 SP7 Patch2 以上
	言語	LD, SFC, FBD, ST, IL, CFC (IEC61131-3準拠)
	フィールドバス	Modbus TCP Master / Slave, Modbus RTU Master / Slave
	通信プロトコル	OPC-UA Server
プログラムサイズ	ROMサイズ	1MB
	最大ステップ数	250K ステップ
CPU基本性能	基本命令実行速度(LD)	1.6ns
	応用命令実行速度(ST)	5.8ns
	スキャン時間	74μs (20000 ステップ時)

3. CPS-PCS341EC-DS1-1201の仕様

項目		CPS-PCS341EC-DS1-1201
CODESYS対応機能	バージョン	V3.5 SP7 Patch2 以上
	言語	LD, SFC, FBD, ST, IL, CFC (IEC61131-3準拠)
	フィールドバス	EtherCAT Master、Modbus TCP Slave
	通信プロトコル	OPC-UA Server
プログラムサイズ	ROMサイズ	1MB
	最大ステップ数	250K ステップ
CPU基本性能	基本命令実行速度(LD)	1.6ns
	応用命令実行速度(ST)	5.8ns
	スキャン時間	74μs (20000 ステップ時)
EtherCAT性能	入力処理時間(LD)	144ns
	出力処理時間(ST)	138ns
	スキャン時間	166μs (64点入力、64出力時)

4. CPS-PCS341MB-DS1-1201の仕様

項目		CPS-PCS341MB-DS1-1201
CODESYS対応機能	バージョン	V3.5 SP7 Patch2 以上
	言語	LD, SFC, FBD, ST, IL, CFC (IEC61131-3準拠)
	フィールドバス	Modbus TCP Master / Slave, Modbus RTU Master / Slave
	通信プロトコル	OPC-UA Server
プログラムサイズ	ROMサイズ	1MB
	最大ステップ数	250K ステップ
CPU基本性能	基本命令実行速度(LD)	1.6ns
	応用命令実行速度(ST)	5.8ns
	スキャン時間	74μs (20000 ステップ時)

2. CONPROSYS HMI仕様

項目		仕様
対応IECデータ型		BOOL BYTE USINT WORD UINT DWORD UDINT LWORD ULINT SINT INT DINT LINT REAL LREAL STRING
CONPROSYS HMI予約変数		PLCステータス アイテム名 : Status/PLCStatus アクセス : R データ型 : BYTE データ範囲 : 0(PLC停止中) or 1(PLC動作中)
使用可能な変数の最大数		128
工場出荷時設定	CPS-PC341EC-1-9201	なし
	CPS-PC341MB-ADSC1-9201	デジタル出力ビット0～1 アイテム名 : IoVariables/DO0～1 アクセス : R/W データ型 : BIT データ範囲 : 0 or 1 デジタル入力ビット0～3 アイテム名 : IoVariables/DI0～3 アクセス : R データ型 : BIT データ範囲 : 0 or 1 アナログ入力チャネル0～1 アイテム名 : IoVariables/AI0～1 アクセス : R データ型 : DWORD データ範囲 : 0～4095 カウンタ入力チャネル0～1 アイテム名 : IoVariables/CNT0～1 アクセス : R データ型 : DWORD データ範囲 : 0～16777215 カウンタ入力チャネル0～1クリア アイテム名 : IoVariables/CNTCLR0～1 アクセス : R/W データ型 : BIT データ範囲 : 0 or 1(1でクリア) PLCステータス アイテム名 : Status/PLCStatus アクセス : R データ型 : BYTE

項目		仕様
		データ範囲：0(PLC停止中) or 1(PLC動作中)
	CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	デジタル入力ビット0～3 アイテム名：IoVariables/DI0～3 アクセス：R データ型：BIT データ範囲：0 or 1 PLCステータス アイテム名：Status/PLCStatus アクセス：R データ型：BYTE データ範囲：0(PLC停止中) or 1(PLC動作中)

付録

本書で示している仕様や製品についての補足説明をしています。

1. 送信データフォーマット

本製品とサーバーとは、httpまたはhttpsを使用してデータの送信を行います。

設定されたURLへ下記のパラメーターでデータをPOSTします。

◆ 送信パラメーター一覧

送信内容	パラメーター1	パラメーター2
計測データファイル	file=data	filename=YYYYMMDDhhmmss_cdtxx.csv

◆ Webサーバーからの応答

応答	内容	本製品の動作
コード : 200 X-AggregateInfo-Result : OK	正常	送信したファイルを本体から削除します。
コード : 400	無効ID、認証コード異常、フォーマット異常など	送信に失敗したファイルを削除します。
上記以外	その他の異常	送信に失敗したファイルを再送用ファイルとして保持します。

◆ 電文例

http request
<pre>POST /XXXXX HTTP/1.1<CR_LF> User-Agent: XXXXX<CR_LF> Host: 192.168.1.110<CR_LF> Accept: */*<CR_LF> Content-Length: 40602<CR_LF> Expect: 100-continue<CR_LF> Content-Type: multipart/form-data; boundary=-----43ac9283b67c39f1<CR_LF> Content-Disposition: form-data; name="data"; filename="201401011000.csv"<CR_LF> Content-Type: text/plain; charset=UTF-8<CR_LF> <CR_LF> [データ] -----43ac9283b67c39f1<CR_LF> Content-Disposition: form-data; name="err"; filename="20161219100000_cdr00.csv "<CR_LF> Content-Type: text/plain; charset=UTF-8<CR_LF> <CR_LF></pre>
http response(正常)
<pre>HTTP/1.1 200 OK<CR_LF> Server: Apache-Coyote/1.1<CR_LF> Content-Type: text/plain; charset=UTF-8<CR_LF> Content-Length: XXXX<CR_LF> Date: Wed, 01 Jan 2014 10:00:01 GMT<CR_LF> X-AggregateInfo-Result: OK<CR_LF></pre>

接続タイムアウト……………20秒

Webサーバー応答タイムアウト……60秒

◆ 送信先Webサーバー

CONPROSYS WEB Settingの『データ送信設定』の『データ送信先URL』で指定したURLを使用します。

◆ データファイル

分類	項目	内容	備考
ヘッダー	端末ID	X(7)9(6)	製品のシリアル番号を記載
	予約		未使用
データ1 - データn			データ項目を「,」で区切ります
フッター	予約		未使用
	送信種別	9(1)	0：通常 1：再送

◆ データ

データのフォーマットは、日時以降は任意に指定できます。

カラム	内容	説明
1	日時	計測した日時(YYYYMMDDhhmm)
2	...	

各種サービス・お問い合わせ

当社の製品をより良く、より快適にご使用いただくために、
行っているサービス、サポートをご紹介します。

1. 各種サービス

株式会社コンテックでは当社製品をご使用いただく上で、技術資料のダウンロードをはじめ、様々な役に立つ情報を提供しています。

ダウンロード

<https://www.contec.com/jp/download/>

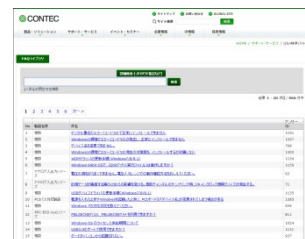
最新のドライバやファームウェア、解説書など技術資料がダウンロードいただけます。ご利用には会員登録(myCONTEC)が必要です。



FAQライブラリ

<https://contec.e-srv.com/>

よくあるご質問やトラブルシューティングをQ&A形式でご紹介しています。



コンテック ナレッジベース

<http://www.contec-kb.com/>

やりたいことが探せる、知識ベースの情報サイトです。
接続したい機器、やりたいことなど、目的から解決策を探せます。
お役立ち情報がいっぱいです。



インターネット通販

<https://www.contec-eshop.com/>

当社が運営する、最短翌日納品の大変便利なネット直販サービスです。



評価機無料貸出

<https://www.contec.com/jp/support/evaluation/>

当社製品を無料でお試しいただけるサービスです。
ご購入前の仕様確認、ご評価にぜひご活用ください。
ご利用には会員登録(myCONTEC)が必要です。



2. お問い合わせ

当社製品の技術的なことや質問など、またご購入に関するお問い合わせなど各種のお問い合わせを承ります。

技術的なお問い合わせ(テクニカルサポートセンター)

製品の使い方、初期不良、動作異常、環境対応など製品の技術的なお問い合わせに、専門技術スタッフが迅速かつ親切丁寧に対応します。

当社ホームページからお問い合わせください。

お問い合わせ

<https://www.contec.com/jp/support/technical-support/>

E-mail : tsc@jp.contec.com、TEL : 050-3736-7861 でも対応しております。

営業的なお問い合わせ

ご購入方法、販売代理店のご紹介、カスタム対応/OEM/ODMのご相談、システム受託開発のご依頼は当社支社(営業窓口)にお問い合わせください。

または、E-mail(sales@jp.contec.com)でもお問い合わせいただけます。

TEL、FAX番号については、当社ホームページまたはカタログの裏表紙に記載しています。

納期、価格、故障修理のご依頼、寿命部品交換のご依頼

当社製品取り扱いの販売代理店へお問い合わせください。

販売代理店

<https://www.contec.com/jp/support/distributors/>

索引

C

CODESYSの開発環境の入手方法	45
CODESYS各部の名称.....	53
CODESYS用パッケージをインストールする.....	47
CONPROSYS HMI	152
CONPROSYS HMIでできること	159
CONPROSYS WEB Settingの基本操作	26

E

EtherCAT Masterとして使用する	76
------------------------------	----

I

I/Oマッピング.....	64
---------------	----

M

Modbus RTU Masterとして使用する	96
Modbus RTU Slaveとして使用する	103
Modbus TCP Masterとして使用する.....	81
Modbus TCP Slaveとして使用する.....	88

O

OPC UA Serverを設定する	112
OPC UAクライアントソフト.....	112
OPC UAシンボル設定	67

P

PACシリーズの機能.....	7
-----------------	---

S

STエディタウィンドウ	53
ST言語の書式と記述方法	54

お

お問い合わせ	179
オンラインヘルプ.....	12

か

カウンタ入力	116
各種サービス	178
関連マニュアル	11

き

基本的なプログラミング手順	56
---------------------	----

く

クラウドサービスヘデータを送信する	140
-------------------------	-----

こ

コントローラとPCでシリアル通信する.....	70
-------------------------	----

し

シリアルポート	67
---------------	----

せ

製品使用までの流れ	10
セキュリティに関する注意.....	17

そ

送信データフォーマット.....	175
------------------	-----

た

対応Webブラウザ	25
-----------------	----

て

デバイスウィンドウ	53
デバイスコンフィグレーションウィンドウ.....	53

と

取り扱い上の注意	16
----------------	----

は

ハードウェア仕様.....	170
パラメーター	63

ふ

ファームウェアのバージョンアップ	50
------------------------	----

ファイルにアクセスする.....	124
------------------	-----

め

メニュー項目	27
--------------	----

も

モニタリング	152
--------------	-----

改訂履歴

改訂日	改訂内容
2015年12月	初版
2018年3月	レイアウト変更
2018年12月	シリアル設定、Modbus RTUの説明を追加

- 本書の内容について万全を期しておりますが、万が一不審な点や、記載もれなどお気づきのことがありましたら、お買い求めの販売店または総合インフォメーションへご連絡ください。
- CONPROSYSは、株式会社コンテックの登録商標です。CODESYS®は3S-Smart Software Solutions GmbHの登録商標です。その他、本書中に使用している会社名および製品名は、一般に各社の商標または登録商標です。

やりたいことから探せる

コンテック ナレッジベース

<http://www.contec-kb.com>

やりたいことが探せる、知識ベースの情報サイトコンテックナレッジベースをご利用ください。

お役立ち情報がいっぱい、目的から解決策を探せます。



株式会社コンテック

〒555-0025 大阪市西淀川区姫里3-9-31

<http://www.contec.com/>

本製品および本書は著作権法によって保護されていますので無断で複写、複製、転載、改変することは禁じられています。

PACシリーズ システムセットアップガイド

NA04614 (LYTK765) 12112018_rev5 [12152015]

2018年12月改訂