MITSUBISHI ELECTRIC
Changes for the Better

e-F@ctory

三菱電機汎用シーケンサ
C言語コントローラユニット クイックスタートガイド

はじめよう、
C言語コントローラ!
Q12DCCPU-V

C Controller

Smart & Easy

より高性能に、より手軽に、組込みシステムプラットフォームを手に入れよう

ガイドの見方

本クイックスタートガイドで使用する記号とその内容について説明します。

記号	内容	例
	知っておく必要のある内容を記載しています。	C 言語コントローラユニットのプログラムの演算は，スイッチの状態が RUN/STOP にかかわらず実行されます。
	参照マニュアルや詳細を記載しているページを紹介しています。	下記マニュアルを参照してください。 ☞ C 言語コントローラユニットユーザーズマニュアル (ハードウェア設計・機能解説編) ：SH-080764
	用語の説明を記載しています。	バッファメモリ：インテリジェント機能ユニット内部の記憶領域で，C 言語コントローラユニットと受け渡しするデータ (設定値，モニタ値など) が格納されます。
	作業を行う上で必ず注意することを記載しています。	ユニットを取り付けるときは，必ず電源を遮断してください。
[ ]	メニューバーのメニュー名 ([ ] → [ ] はドロップダウンメニューを示します。)	メニュー [Project] → [Properties]
	画面のボタン	ボタン
	キーボードのキー	 キー

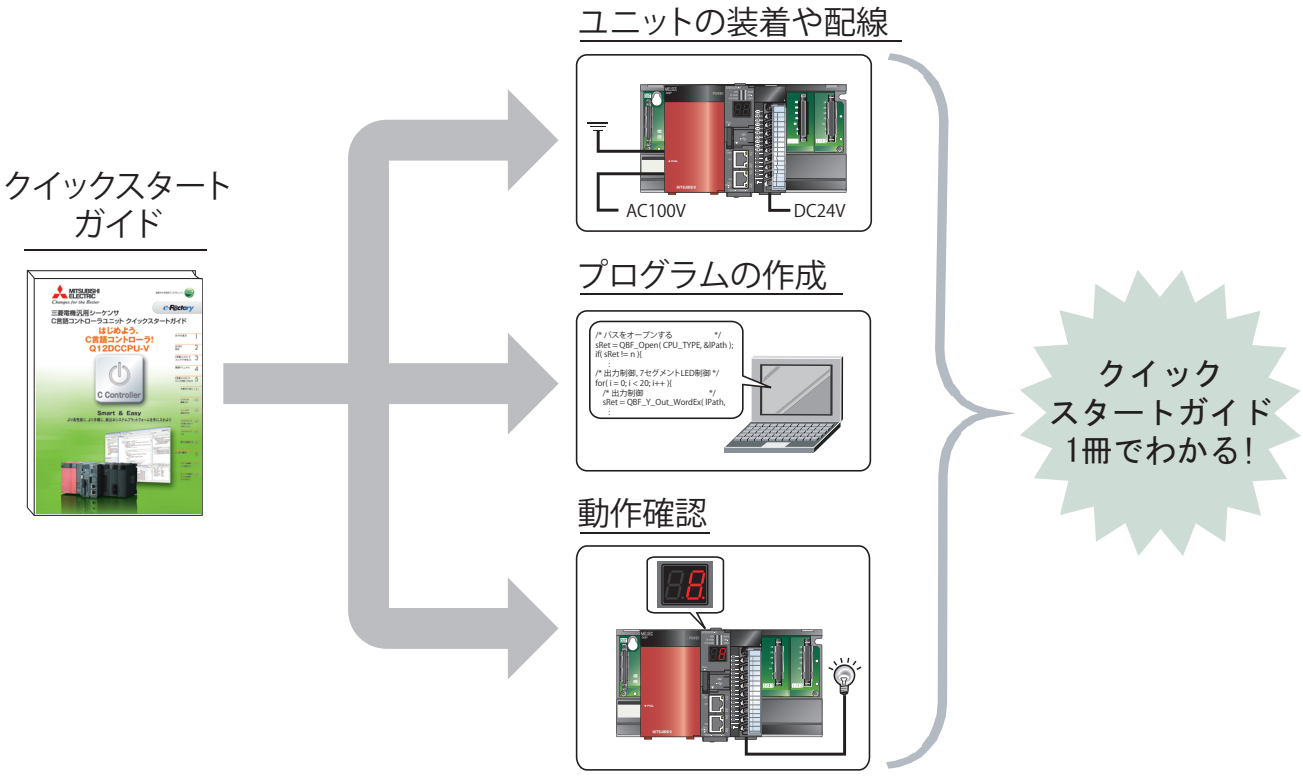
はじめに

本クイックスタートガイドは，三菱電機シーケンサ MELSEC-Q シリーズ C 言語コントローラユニット (以下 C 言語コントローラユニット) を初めて使用する場合の基本的な導入手順を，わかりやすく説明しています。

MELSEC-Q シリーズを初めて使用する方で，下記のような方が，C 言語コントローラユニットの使い方を簡単に理解することができるようになっています。

・C 言語または C++ 言語プログラムのプログラミング経験がある

・マイコンボードやパソコン環境から C 言語コントローラユニットを使ったシステムへの置換えを考えている



参考

●使用上の注意事項

C 言語コントローラユニットを安全に使用するために,C 言語コントローラユニットユーザーズマニュアルの「安全上のご注意」をよく読んだ上で，使用してください。

注意

本クイックスタートガイドは，「〈2〉システムを構築する」(P.13) に示すシステム構成での操作を前提としています。

実際にシステムを設計／運用する場合には，下記ページで紹介しているマニュアルを必ずお読みください。

「関連マニュアル」(P.10)

目次

MEMO

C 言語コントローラユニットでできること

■ 複雑かつ高速な処理や上位サーバとの通信処理ができます

C 言語コントローラユニットは，C 言語または C++ 言語プログラムを使って MELSEC-Q シリーズのユニットの管理や入出力機器の制御を行う CPU ユニットで，下記のようなことが可能です。

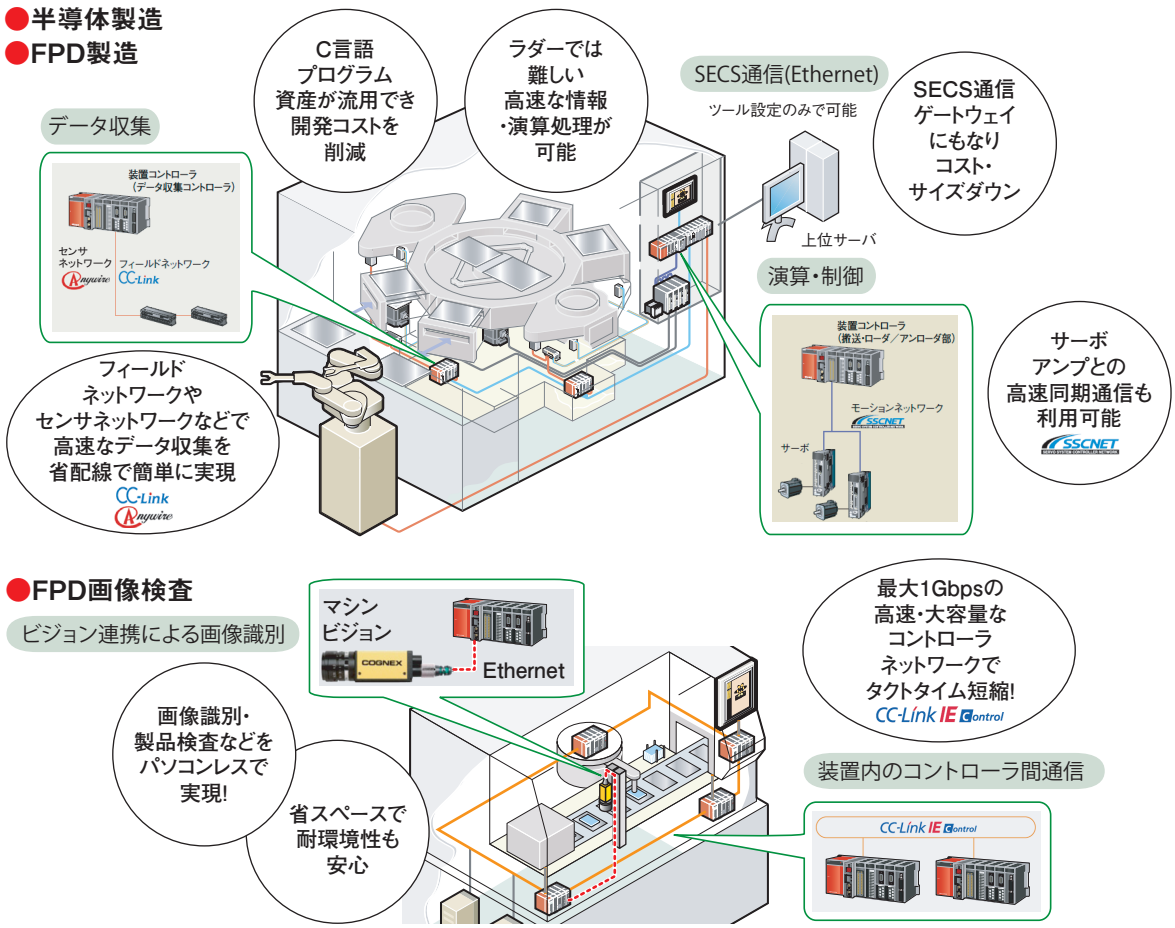
・マイコンボードやパソコン環境で開発した C 言語または C++ 言語プログラムの流用

・ラダープログラムでは困難な，複雑かつ高速な情報・演算処理が必要な分野での活用 (半導体製造，FPD 製造，太陽電池製造，公共インフラ (電気，ガス，水道など) 遠隔監視など)

C 言語コントローラユニットは，ユーザプログラムによって，さまざまな機能を簡単に実現できます。また，パートナ製品を活用することで，例えば下記のような機能も簡単に実行できます。

・SECS 通信ソフトウェアパッケージにより，半導体製造分野などでよく使われる SECS 通信にプログラムレスで対応し，上位サーバとの通信もゲートウェイパソコンなしで直接実行

・ビジョンシステムと連携し，画像識別や製品検査などをパソコンレスで実行



■ 豊富な機能を実装し，リアルタイムな制御を実現します

C 言語コントローラユニットは，実績と信頼性のあるリアルタイム OS，VxWorks(米国ウインドリバー・システムズ社製) を標準搭載しています。(ランタイムライセンス費用は発生しません。)

VxWorks は，プリエンティブ方式 *1 によって，リアルタイムな演算処理が可能です。

そのため，パソコン環境とは異なり，割込みや定時性が求められる処理などにも確実に対応できます。

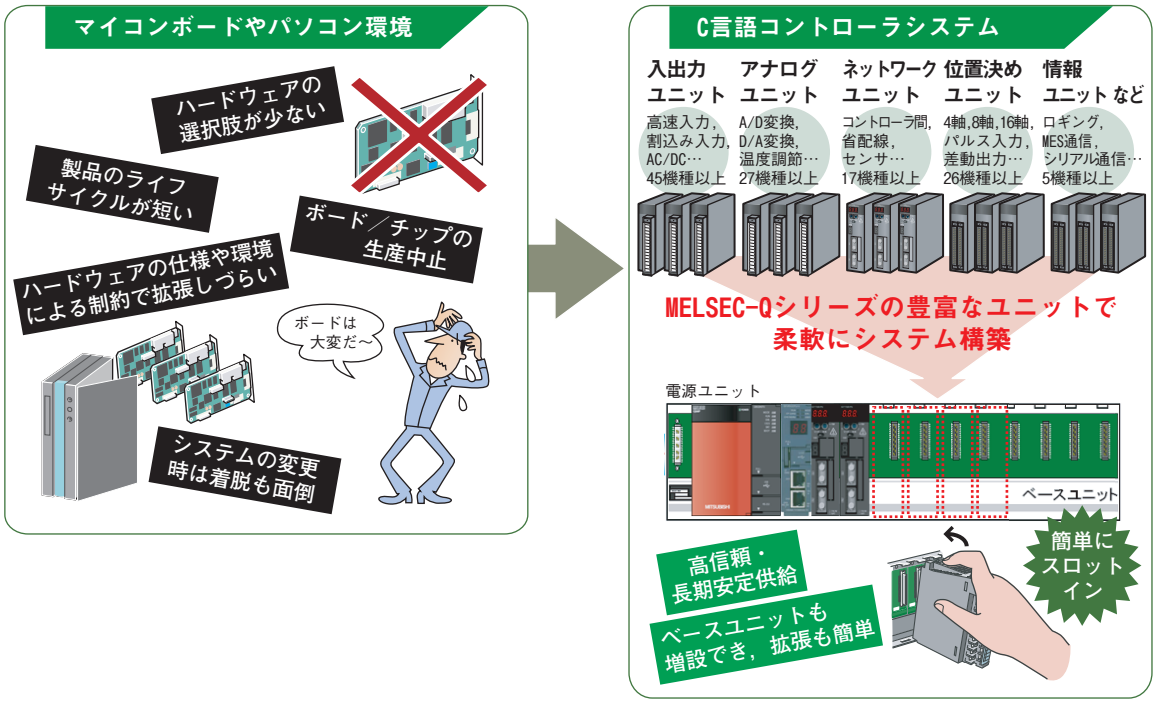
また，VxWorks は，ファイルアクセスやネットワーク機能のドライバ，入出力や通信のライブラリなど，豊富な機能を標準で実装しているため，さまざまな用途に即応できます。

＊ 1　複数のプログラムを実行する上で，特定のプログラムだけがプロセッサ (CPU) を専有することなく，公平に実行時間を割り振ることができる処理方式。

■ C 言語コントローラユニットの特長

1. MELSEC-Q シリーズの豊富なユニットで柔軟にシステム構築！

C 言語コントローラシステムでは，プログラム資産を流用できるだけではなく，MELSEC-Q シリーズの豊富なユニットを活用でき，システム構成の変更も簡単です。



2. OS，ドライバ，ライブラリを実装済みで，ユーザプログラム開発に専念！

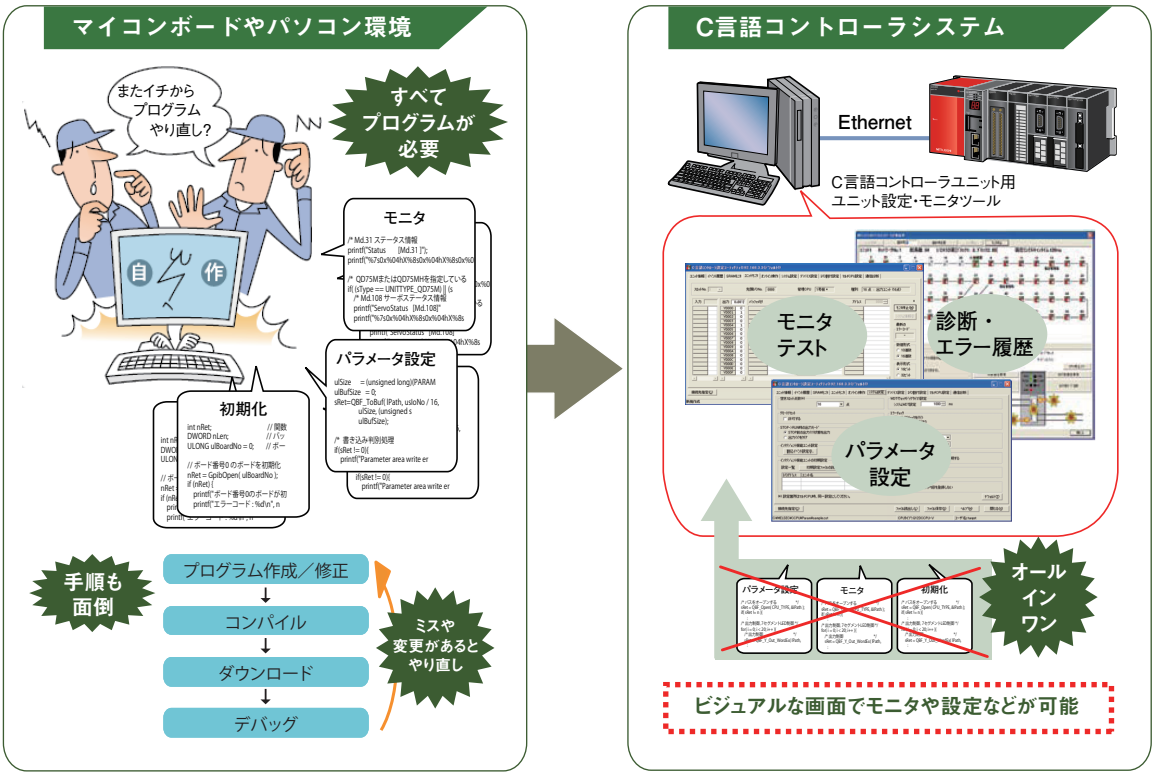
C 言語コントローラユニットには，OS と通信ドライバが組み込まれています。マイコンボードやパソコン環境での面倒な作業 (OS 移植，ドライバ開発，ROM への OS 書込みなど) が不要なため，ユーザプログラムの開発に専念することができます。

また，MELSEC-Q シリーズの各ユニットには，C 言語コントローラユニットの専用ライブラリ関数 (バスインタフェース関数，MELSEC 通信関数) により，簡単にアクセスできます。

3. 初期化やパラメータ設定，モニタ・テストが，プログラムレスで可能！

C 言語コントローラユニットの初期化，システム設定，ネットワークユニットのパラメータ設定などのために，複雑なプログラムを組む必要はありません。C 言語コントローラユニット用ユニット設定・モニタツールのビジュアルな画面を操作するだけで，簡単に作業が完了します。

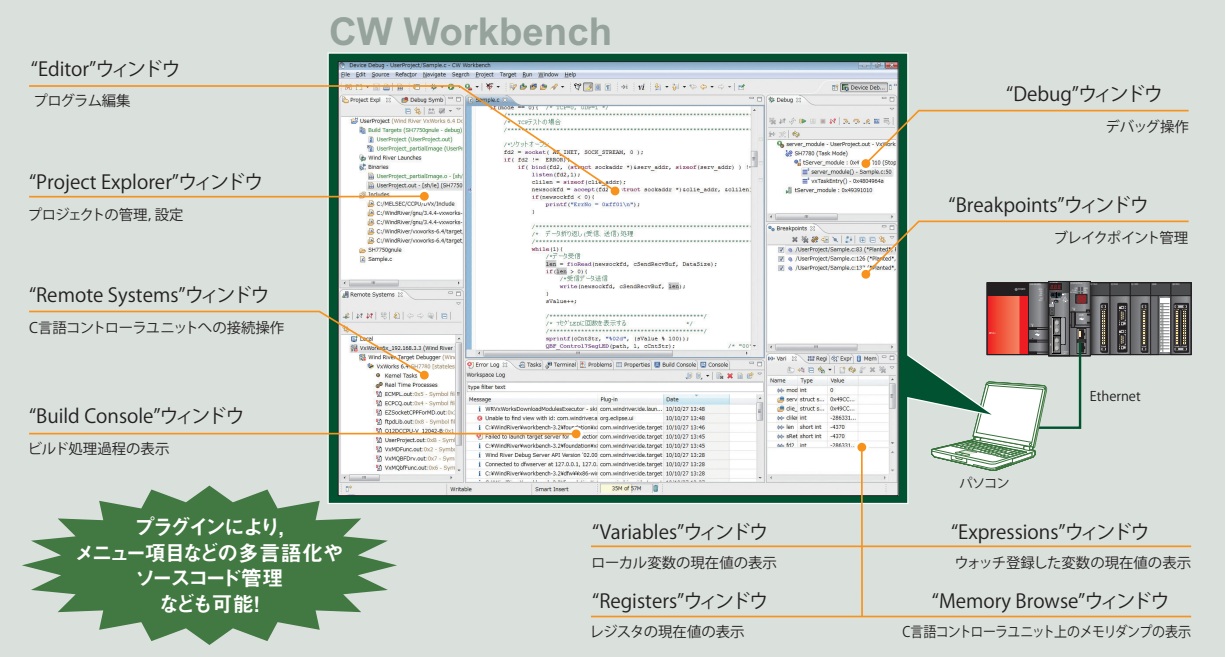
また，各ユニットの状態，C 言語コントローラユニットやユーザプログラムで発生したエラーなどの確認，ケーブル断線や通信状態などの確認も，プログラムレスで可能です。



4. 手軽な統合開発環境「CW Workbench」でクイックスタート！

プログラムの編集から実行モジュールの生成，デバッグまでの基本機能を備えた C 言語コントローラ用エンジニアリングツール「CW Workbench」により，手軽に C 言語コントローラユニットのユーザプログラムを開発できます。

また，Eclipse ベースの CW Workbench は，サードパーティ製のプラグインソフトにより機能拡張が可能です。

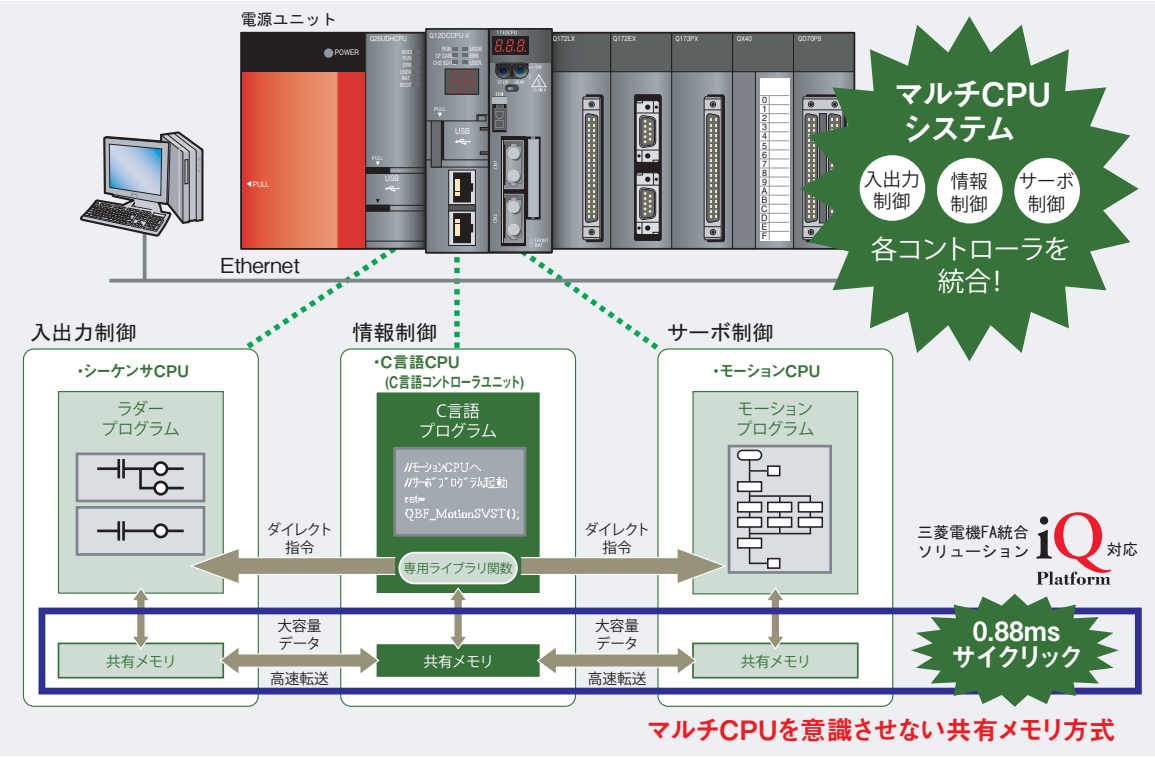


5. 高速バス通信で連携するマルチ CPU システムにより，高速・高精度な制御を実現！

CPU 間高速バス通信により，モーション CPU の演算周期 0.88ms に同期したリアルタイムな逐次制御や時々刻々と目標値が変化する追従制御などに対応できます。

プログラムレスで最大 14K ワードの大容量データを 0.88ms 周期で高速転送することができ，CPU 間同士でデータを共有することができます。

また，マルチ CPU システムによって工場内の制御の中枢となる CPU を統合すれば，システム全体を効率的に制御することができ，負荷が分散されます。



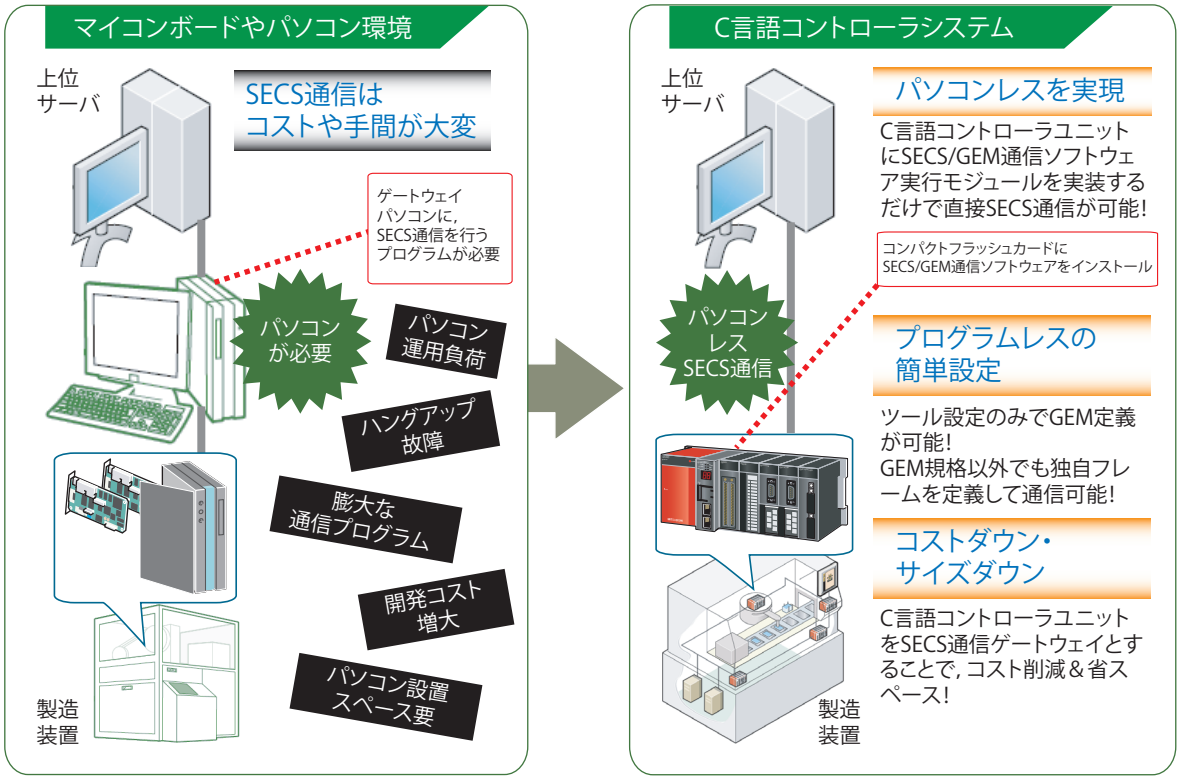
6. パートナ製品の活用により用途拡大！

C 言語コントローラユニットに下記などのパートナ製品を組み合わせれば，さらに高機能で使いやすい情報連携を行うことができます。

(1) SECS 通信ソフトウェアパッケージとの情報連携

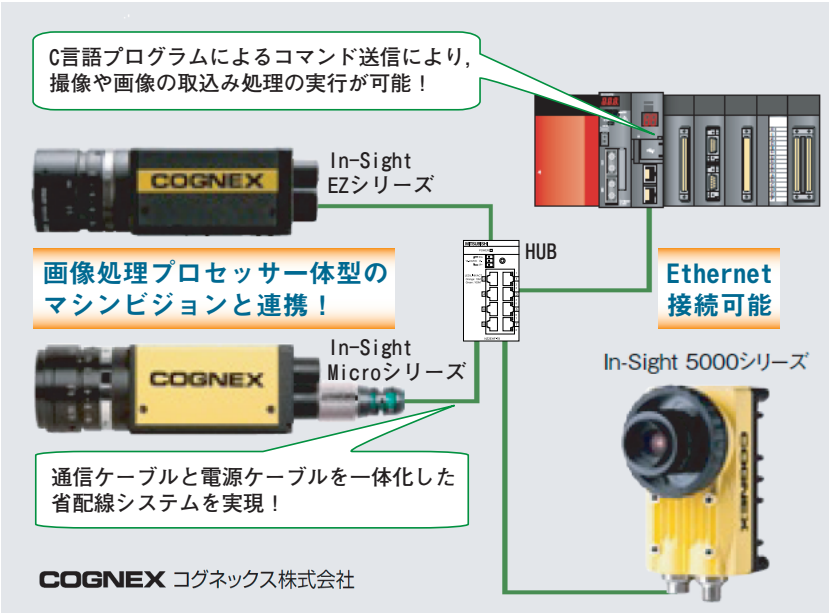
SECS/GEM 通信ソフトウェアを導入すれば，パソコンレス，プログラムレスで上位サーバとの SECS 通信 (GEM[*1] ／非 GEM) を実現でき，製造装置の状態管理や情報収集が可能になります。

＊1　半導体の製造ラインで使用される，業界標準通信プロトコルの 1 つ。



(2) ビジョンシステム (COGNEX In-Sight EZ, In-Sight Micro, In-Sight5000 シリーズ) との連携

COGNEX マシンビジョンと C 言語コントローラユニットの連携により，製品の測定，検査，識別など製造工程の自動化を簡単に実現できます。



関連マニュアル

本クイックスタートガイドでは，C 言語コントローラユニットの基本的な導入手順を紹介しています。
C 言語コントローラユニットを十分に活用するために，目的に応じて，下記のマニュアルをお読みください。

■ C 言語コントローラユニットについて詳しく知りたいとき

●MELSEC-Q C 言語コントローラユニットユーザーズマニュアル
..... SH-081077
C 言語コントローラユニットのシステム構成，仕様，機能，取扱い，配線，トラブルシューティング，および関数とプログラミングについて説明しています。

●C 言語コントローラ設定・モニタツール Version 4 オペレーティングマニュアル
.....SH-081076
C 言語コントローラ設定・モニタツールのシステム構成，操作方法について説明しています。

■ CW Workbench について詳しく知りたいとき

●CW Workbench オペレーティングマニュアル SH-080981
CW Workbench のシステム構成，インストール／アンインストール，仕様，機能，トラブルシューティングについて記載しています。

マニュアルの PDF を，下記からダウンロードできます。
三菱電機 FA サイトホームページ
(www.MitsubishiElectric.co.jp/fa)

C 言語コントローラユニットを使ってみよう

C 言語コントローラユニットは次のような手順で導入します。

〈1〉 作業を行う前に (P.12)
必要な機材を準備します。

〈2〉 システムを構築する (P.13)
準備した機材を設置して配線し，電源を入れます。

1) システム構成例 (P.13)
本クイックスタートガイドの操作説明で使用するシステム構成例です。
2) ユニットを装着する (P.14)
システムを構成するユニットを，ベースユニットに装着します。
3) ユニットの配線を行う (P.15)
電源ユニットおよび出力ユニットへの配線をします。
4) 電源が正常か確認する (P.17)
システムの電源を入れて，ユニットの状態を確認します。

〈3〉 ユニットの設定をする (P.19)
C 言語コントローラ設定 ・ モニタツールを使って， C 言語コントローラユニットを動作させるための設定を行います。

1) C 言語コントローラユニットを初期化する (P.19)
設定を行う前に，標準 RAM を使用可能な状態にします。
2) パラメータを設定する (P.21)
C 言語コントローラユニットに，パラメータを設定します。

〈4〉 プログラミングする前に知っておきたいこと (P.24)
バスインタフェース関数についての説明です。

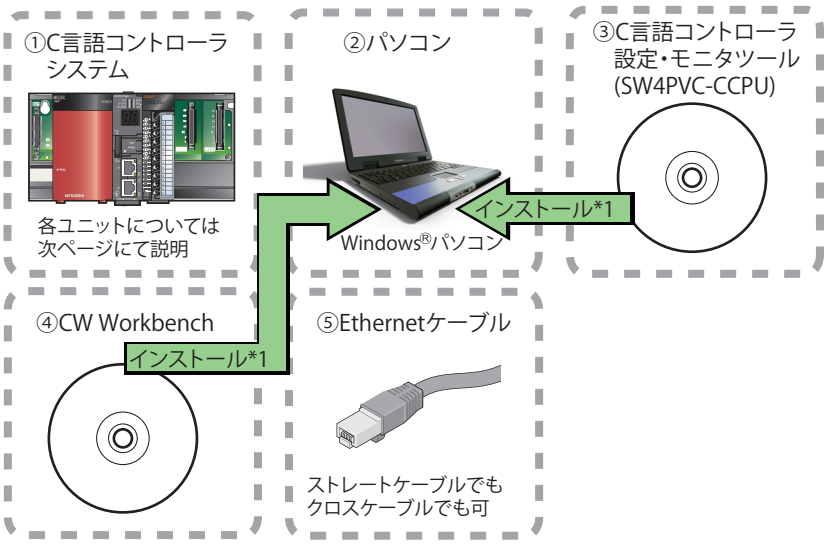
〈5〉 プログラミングする (P.27)
CW Workbench を使って，プログラムを作成します。

1) プロジェクトを作成する (P.30)
CW Workbench を起動し，プロジェクトの作成と設定を行います。
2) ユーザプログラムを作成する (P.34)
C 言語コントローラシステムの制御を行うユーザプログラムを作成します。
3) ユーザプログラムから実行モジュールを生成する (P.36)
作成したプログラムを，実行可能なモジュールに変換 (ビルド) します。
4) C 言語コントローラユニットと CW Workbench を接続する (P.37)
デバッグを行うために，C 言語コントローラユニットと CW Workbench を接続します。
5) ユーザプログラムをデバッグする (P.39)
作成したプログラムが正しく動作するか確認します。
6) 実行モジュールを登録する (P.44)
プログラムを稼動用にビルドし，C 言語コントローラユニットに格納します。

〈6〉 動作を確認する (P.46)
プログラムを実行し，動作を確認します。

〈1〉作業を行う前に

必要な機材を準備します。



＊1 あらかじめ，同じパソコンに C 言語コントローラ設定・モニタツール（SW4PVC-CCPU），CW Workbench をインストールしておいてください。

参考

C言語コントローラ設定・モニタツールのインストールについては, 下記を参照してください。

☞ SW4PVC-CCPU インストール手順書：BCN-P5924

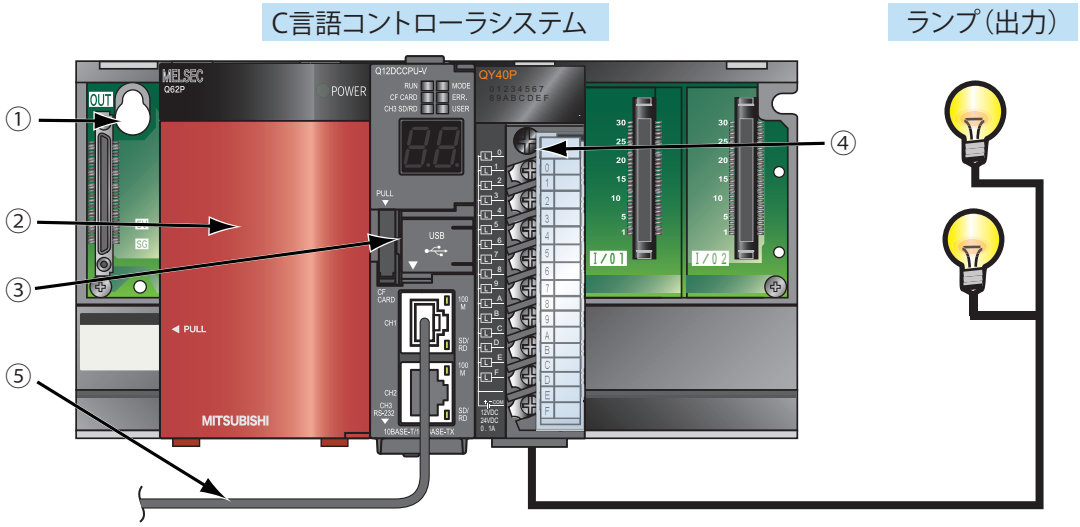
CW Workbench のインストールについては，下記を参照してください。

☞ CW Workbench オペレーティングマニュアル：SH-080981

〈2〉システムを構築する

1) システム構成例

本クイックスタートガイドでは，下記のシステム構成を例に挙げて説明します。



＊電源ユニットへの配線は省略しています。

No.	名称	形名	説明
①	ベースユニット	Q33B	電源ユニット，C 言語コントローラユニット，入出力ユニットなどを装着するユニットです。
②	電源ユニット	Q62P	C 言語コントローラユニット，入出力ユニットなど各ユニットに電気を供給するユニットです。
③	C 言語コントローラユニット	Q12DCCPU-V	C 言語コントローラシステムの制御を統括するユニットです。
④	出力ユニット	QY40P	−
⑤	接続ケーブル (Ethernet ケーブル)	10BASE-T/100BASE-TX の規格を満たす Ethernet ケーブル	SW4PVC-CCPU，CW Workbench をインストールしたパソコンと C 言語コントローラユニットを接続します。

2) ユニットを装着する

準備したユニットをベースユニットに取り付けます。

C 言語コントローラユニットを初めて使用する場合は，バッテリコネクタの装着が必要です。

⚠ 注意

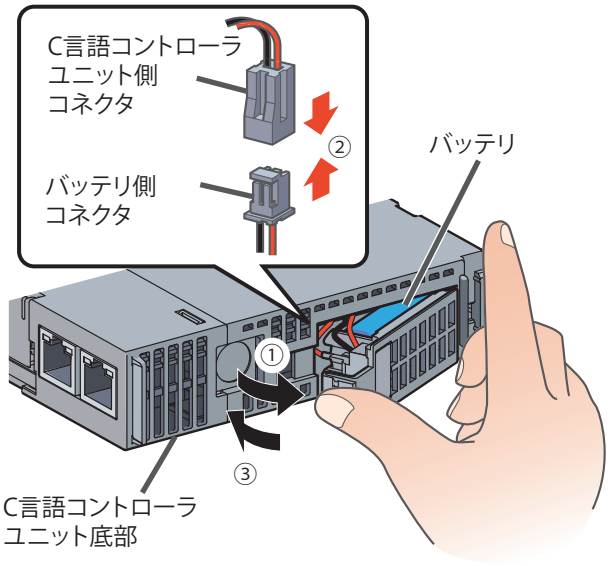
●バッテリは，必ず装着して運転を行ってください。

●ユニットを取り付けるときは，必ず電源を遮断してください。

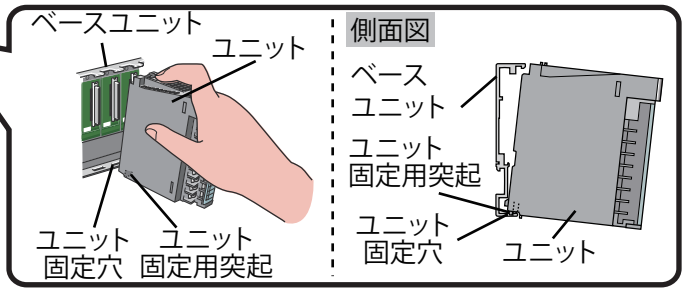
Point

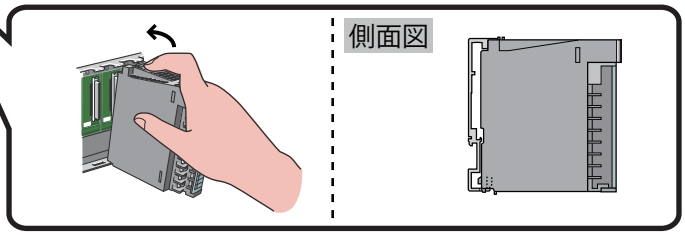
●C 言語コントローラユニットへのバッテリの装着方法

①C言語コントローラユニット
底部のカバーを開ける
↓
②バッテリ側コネクタを
C言語コントローラ
ユニット側コネクタに，
方向を確認して挿し込む
↓
③C言語コントローラユニット
底部のカバーを閉じる
↓
完了



①ユニット固定用突起を
ベースユニットのユニット
固定穴に挿し込む
↓
②ユニット固定穴を支点に，
矢印方向に押して，カチッ
と音がするまで挿し込む
↓
完了





参考

ユニットの取り外し方については，下記マニュアルを参照してください。

☞ QCPU ユーザーズマニュアル (ハードウェア設計・保守点検編) : SH-080472

3) ユニットの配線を行う

電源ユニットの配線を行います。

⚠ 注意

ユニットの配線を行うときは，必ず電源を遮断してください。

参考

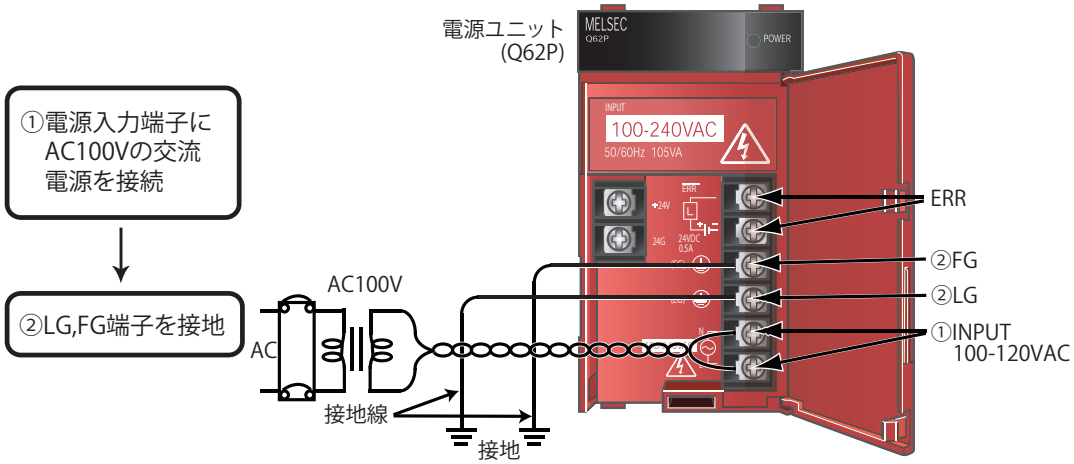
配線についての注意事項の詳細については，下記マニュアルを参照してください。

☞ QCPU ユーザーズマニュアル (ハードウェア設計・保守点検編) : SH-080472

1. 電源ユニットへの配線をする

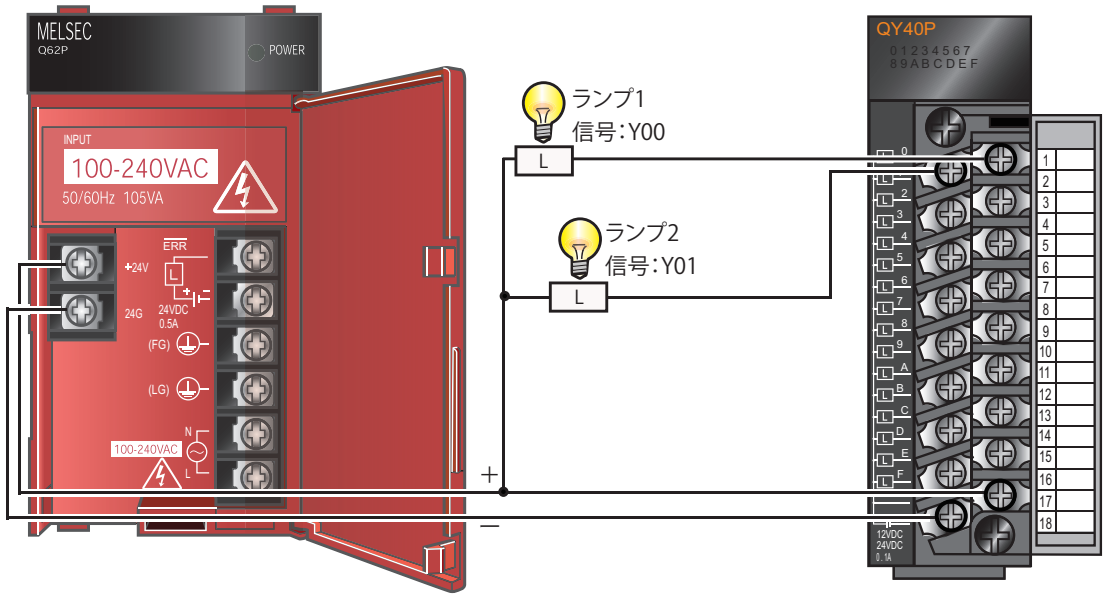
ベースユニットへの電源線，接地線の配線例を下記に示します。

接地は，感電，誤動作を防止するために行う配線です。



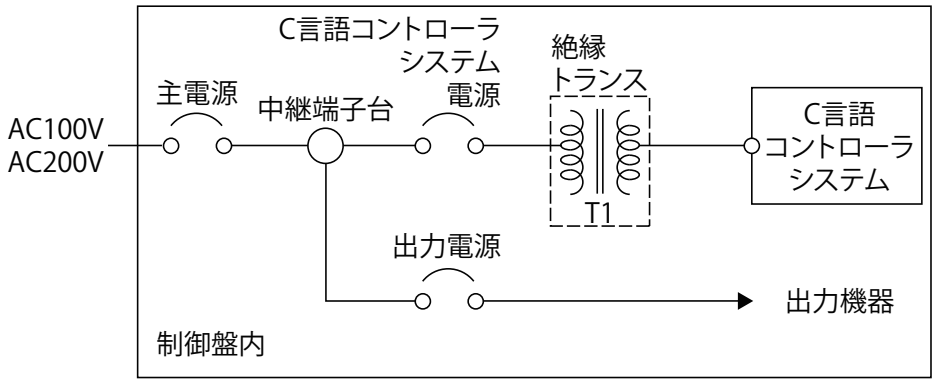
2. 出力ユニットへの配線をする

出力ユニット (QY40P) への配線例を下記に示します。



Point

出力電源と C 言語コントローラシステム電源は，下図のように，系統を分離して配線を行ってください。



4) 電源が正常か確認する

システムの設置，ユニットの取付け，配線をした後に，電源が正常に入ることを確認します。

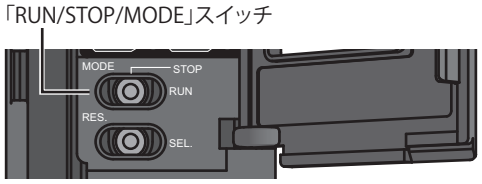
操作手順

1. 電源を入れる前の確認

・電源の配線
・電源電圧

2. C 言語コントローラユニットの状態を STOP にする

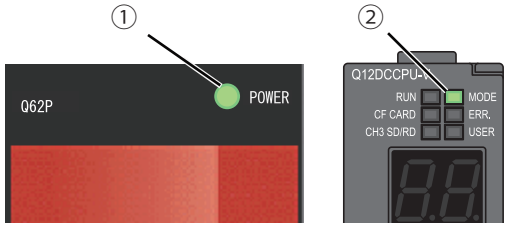
C 言語コントローラユニット前面のカバーを開き，「RUN/STOP/MODE」スイッチを「STOP」の位置にします。



3. 電源を投入

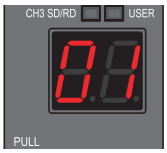
4. 電源が正常であるかを確認

各ユニットの前面 LED を確認します。
正常な状態のときの LED 表示を下記に示します。
① 電源ユニット：「POWER」LED が緑色点灯
② C 言語コントローラユニット
：「MODE」LED が緑色点灯



C 言語コントローラユニットが出荷時の状態の場合 (標準 RAM が初期化されていない状態), 7 セグメント LED に "01" が点滅表示されますが, この段階では問題ありません。
ユニットの初期化を行うと消灯します。

☞「〈3〉 ユニットの設定をする」(P.19)



以上で，システムの構築が完了しました。

Point

電源を投入しても電源ユニットの「POWER」LED が消灯している場合は，電源の配線，装着が正しく行われているか確認してください。

参考

「ERR.」LED が点灯／点滅した場合は，下記マニュアルを参照してトラブルシューティングしてください。

☞ C 言語コントローラユニットユーザーズマニュアル (ハードウェア設計・機能解説編)
：SH-080764

〈3〉ユニットの設定をする

C 言語コントローラユニットを動作させるための設定を行います。

1) C 言語コントローラユニットを初期化する

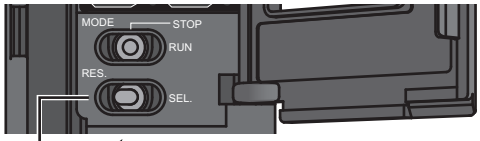
設定を行う前に，C 言語コントローラユニットの標準 RAM を使用可能な状態にします。

⚠ 注意

ユニットを初期化すると，標準 RAM 内の全ファイルが消去されます。

操作手順

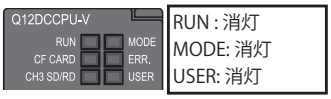
① ユニット前面のカバーを開け，「RESET/SELECT」スイッチを「RESET」側に倒します。



「RESET」側に倒します。

「RESET/SELECT」スイッチ

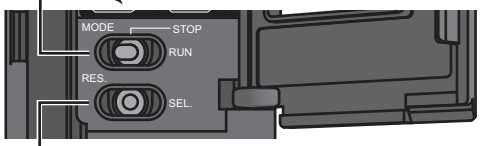
② 「MODE」LED が消灯していることを確認してください。



③ 「RUN/STOP/MODE」スイッチを「MODE」側に保持し，「RESET/SELECT」スイッチを中央に戻します。

「RUN/STOP/MODE」スイッチ

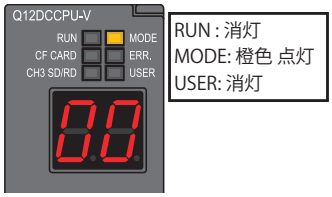
「MODE」側に保持します。



中央に戻します。

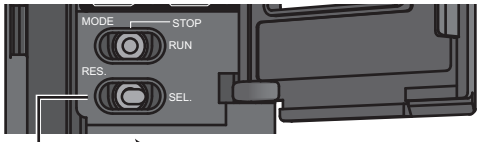
「RESET/SELECT」スイッチ

④ 「MODE」LED が " 橙色 " に点灯し，7 セグメント LED が "00" を表示していることを確認してください。



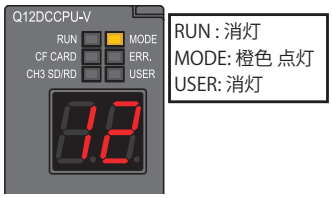
⑤ 「RUN/STOP/MODE」スイッチから手を離します。スイッチが「STOP」位置に戻ります。

⑥ 7 セグメント LED に "12"(ユニット初期化設定（機能拡張モード　））が表示されるまで，「RESET/SELECT」スイッチを「SELECT」側に繰り返し倒します。

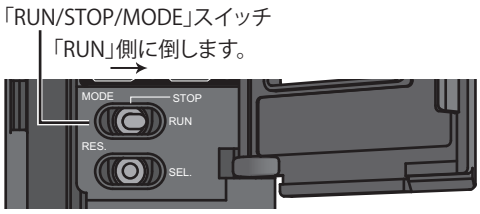


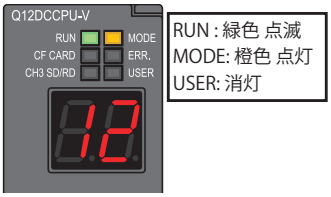
「SELECT」側に繰り返し倒します。

「RESET/SELECT」スイッチ

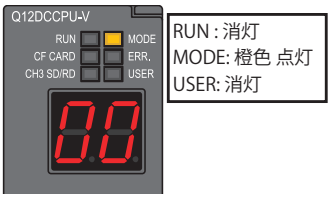


⑦「RUN/STOP/MODE」スイッチを「RUN」側に倒し，ユニット初期化を実行します。実行中は「RUN」LED が点滅します。





⑧「RUN」LED の消灯および，7 セグメント LED の値が "00" になったことを確認後，C 言語コントローラユニットをリセットしてください。



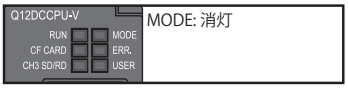
Point

リセットの操作手順

① C言語コントローラユニット前面の「RESET/SELECT」スイッチを，「RESET」側に倒します。

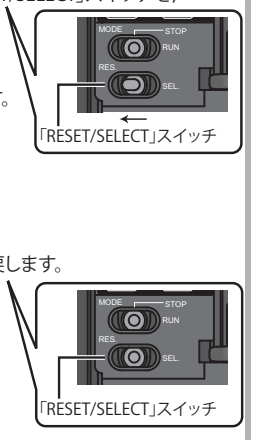
↓

②「MODE」LEDが消灯したことを確認します。
[リセット完了の状態]

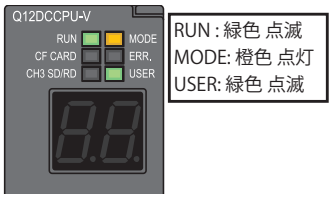


↓

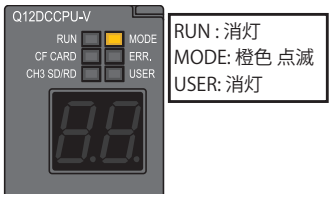
③「RESET/SELECT」スイッチを中央位置に戻します。



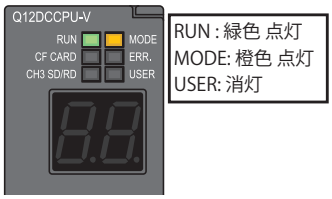
⑨ C 言語コントローラユニットをリセットすると，標準 RAM のフォーマットが実行されます。
「RUN」LED と「USER」LED が緑色点滅します。



⑩ 標準 RAM のフォーマットが完了すると，「RUN」LED，「USER」LED の点滅が終了し，「MODE」LED が橙色点滅します。



⑪ C 言語コントローラユニットをリセットしてください。
標準 RAM のフォーマットが正常完了した場合は，「RUN」LED が緑色点灯，「MODE」LED が橙色点灯します。



⚠ 注意

スイッチを操作する際に，ドライバなど先のとがった道具を使用しないでください。
破損する恐れがあります。

2) パラメータを設定する

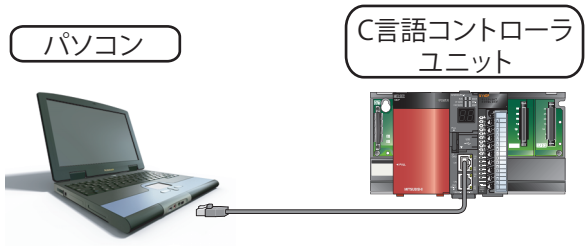
C 言語コントローラユニットに，パラメータを設定します。

用語

パラメータ ：C 言語コントローラシステムを動作させるために必要な設定情報のことです。
C 言語コントローラ設定・モニタツールを使って，C 言語コントローラシステム内の各ユニットおよびネットワークの設定を行います。

1. C 言語コントローラユニットとパソコンを接続する

C 言語コントローラユニットの CH1 とパソコンを，Ethernet ケーブルで接続します。



⚠ 注意

接続の際には，C 言語コントローラユニットとパソコンの IP アドレスを，同一セグメントに設定する必要があります。本クイックスタートガイドでは，C 言語コントローラユニットの IP アドレスは初期値 (192.168.3.3) を使用するため，パソコン側の IP アドレスを "192.168.3.＊ (＊：0, 3, 255 以外)" に設定してください。
また，パソコンのサブネットマスクを "255.255.255.0" に設定してください。

参考

IP アドレスの変更については，下記を参照してください。

☞ C 言語コントローラユニットユーザーズマニュアル (ハードウェア設計・機能解説編) : SH-080764

2. パソコンでパラメータを作成する

操作手順

① C 言語コントローラ設定・モニタツールを起動

[スタート] → [すべてのプログラム] → [MELSEC] → [C 言語コントローラ Ver.4] → [C 言語コントローラ設定・モニタルーツ] を選択します。

② プロジェクトの新規作成

[プロジェクト] → [プロジェクトの新規作成] を選択します。

CPU タイプは「Q12DC-V」を選択します。

③ CCPU パラメータの設定

プロジェクトビュー→ " パラメータ " → "CCPU パラメータ " →「I/O 割付設定」タブを選択します。

参考

各設定画面および設定項目の内容は，下記を参照してください。

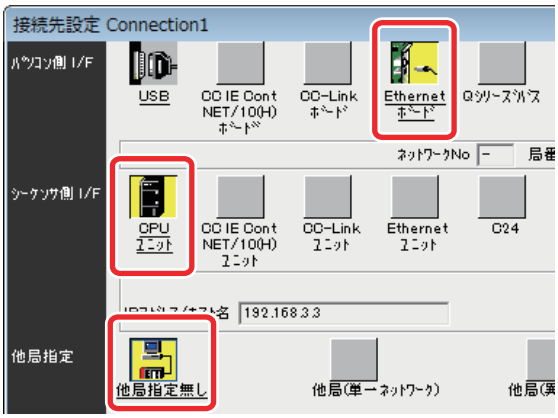
☞ C 言語コントローラ設定・モニタツール Version 4 オペレーティングマニュアル：SH-081076

3. C 言語コントローラにパラメータを書き込む

操作手順

① 接続先の設定

ナビゲーションウィンドウ→接続先ビュー→ " 現在の接続先"→"Connection1"をダブルクリックします。



下記のとおり設定します。

パソコン側 I/F：Ethernet ボード

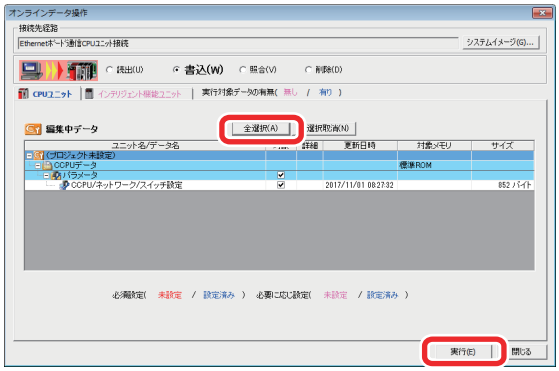
シーケンサ側 I/F：CPU ユニット

他局指定：他局指定無し

② パラメータの書込み

[オンライン] → [CCPU 書込] を選択します。

③ パラメータを全選択し，実行(E) ボタンをクリック

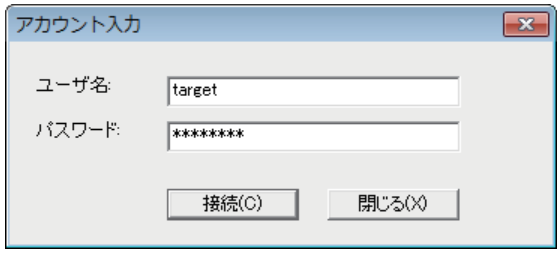


Point

アカウント入力画面が表示された場合は，下記を入力してください。

ユーザ名：target

パスワード：password



4. C 言語コントローラシステムをリセットする

書き込んだパラメータが反映されます。

〈4〉プログラミングする前に知っておきたいこと

1. バスインタフェース関数とは

バスインタフェース関数は，C 言語コントローラユニットの専用ライブラリ関数の一つです。バスインタフェース関数をユーザプログラムに使用すれば，MELSEC-Q シリーズの各ユニットを簡単に制御することができます。

(1) バスのオープン／クローズ

バスインタフェース関数を使用する場合は，プログラムの開始時にバスをオープンし，プログラムの終了時にバスをクローズします。

バスのオープン／クローズ関数

関数名	機能
QBF_Open	バスオープン
QBF_Close	バスクローズ

Point

バスのオープン／クローズ (QBF_Open/QBF_Close) 処理は，プログラムの最初と最後の 1 回のみ行ってください。
プログラムの最初と最後の 1 回にすることで，通信性能を向上させることができます。

(2) 入出力アクセス

入出力アクセスには，1 点単位でのアクセスとワード単位でのアクセスがあります。

① 1 点アクセス：1 点分の情報 (スイッチの ON/OFF，ランプの点灯／消灯など) を扱う関数

1 点アクセス関数の具体例

関数名	機能
QBF_X_In_BitEx	入力信号 (X) を 1 点読み出す。
QBF_Y_Out_BitEx	出力信号 (Y) を 1 点出力する。
QBF_Y_In_Bit_Ex	出力信号 (Y) を 1 点読み出す。

② ワード単位アクセス：ワード (16 ビット) 単位分の情報 (数値，文字列など) を扱う関数

ワード単位アクセス関数の具体例

関数名	機能
QBF_X_In_WordEx	入力信号 (X) をワード単位で読み出す。
QBF_Y_Out_WordEx	出力信号 (Y) をワード単位で出力する。
QBF_Y_In_WordEx	出力信号 (Y) をワード単位で読み出す。

(3) ユーザ LED 制御

ユーザ LED 制御には，C 言語コントローラユニット前面の USER LED 制御と 7 セグメント LED 制御があります。

ユーザ LED 制御関数の具体例

関数名	機能
QBF_ControlLED	C 言語コントローラユニットの USER LED を制御する。
QBF_Control7SegLED	C 言語コントローラユニットの 7 セグメント LED を制御する。

参考

ここでは，最も基本的なバスインタフェース関数を紹介しています。
上記のほかにも，各ユニットの制御を行うための便利なバスインタフェース関数や MELSEC 通信関数があります。

☞ SW □ PVC-CCPU のバスインタフェース関数 HELP，MELSEC 通信関数 HELP

バスインタフェース関数 HELP，MELSEC 通信関数 HELP は，下記からダウンロードできます。
三菱電機 FA サイトホームページ
(www.MitsubishiElectric.co.jp/fa)

☞ C 言語コントローラユニットユーザーズマニュアル (ユーティリティ操作・プログラミング編)：SH-080765

2. 今回使用するバスインタフェース関数

今回作成するプログラムには，出力アクセスと 7 セグメント LED 制御の基本的なバスインタフェース関数を使います。

・バスオープン／クローズ：QBF_Open/QBF_Close 関数

□書式

```
ret = QBF_Open(unit, path);
```

型	名称	内容	IN/OUT
short	ret;	戻り値	OUT
short	unit;	ユニット識別 (2固定)	IN
long	*path;	オープンされたユニットのパスのポインタ	OUT

□書式

```
ret = QBF_Close (path);
```

型	名称	内容	IN/OUT
short	ret;	戻り値	OUT
long	path;	オープンされたパスのパス	IN

・出力アクセス：QBF_Y_Out_WordEx 関数

□書式

```
ret = QBF_Y_Out_WordEx(path, sFlg, usYno, usSize, pusDataBuf, usBufSize);
```

型	名称	内容	IN/OUT
short	ret;	戻り値	OUT
long	path;	オープンされたパスのパス	IN
short	sFlg;	アクセスフラグ(0:通常アクセス、1:高速アクセス、0,1以外:リザーブ(通常アクセス))	IN
unsigned short	usYno;	先頭出力番号(Y)	IN
unsigned short	usSize;	書き込みワード数	IN
unsigned short	*pusDataBuf;	書き込みデータ	IN
unsigned short	usBufSize;	ダミー(0固定)	IN

・7 セグメント LED 制御：QBF_Control7SegLED 関数

□書式

```
ret = QBF_Control7SegLED(path, mode, data);
```

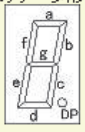
型	名称	内容	IN/OUT
short	ret;	戻り値	OUT
long	path;	オープンされたパスのパス	IN
long	mode;	モード (0: Manualモード、1: Autoモード、その他: 0と同じ)	IN
char	*data;	LEDデータ	IN

□説明

・ modeに指定された方法に従って、LEDデータに指定された値を7セグメントLEDに表示します。
【モード 0:Manualモードの時】
data[0] 7セグメントLED：一位のデータ
data[1] 7セグメントLED：十位のデータ
として、下記の形式にて指定されたデータを表示します。

各位のデータ形式：

ビット	b7	b6	b5	b4	b3	b2	b1	b0
セグメント	DP	g	f	e	d	c	b	a



※セグメント：左図参照。
※ビット値
0：LED OFF(消灯)
1：LED ON (点灯)

参考

C 言語コントローラユニットで使用する C 言語および C++ 言語プログラムのデータ型には，主に下記のようなものがあります。

データ型	ビット幅	呼称
byte	8	符号なし整数型
char	8	文字型
unsigned char	8	符号なし文字型
short	16	符号付き短長整数型
unsigned short	16	符号なし短長整数型
int	32	符号付き (倍長) 整数型
long	32	
unsigned long	32	符号なし (倍長) 整数型
float	32	単精度実数型
double	64	倍精度実数型
void	-	-

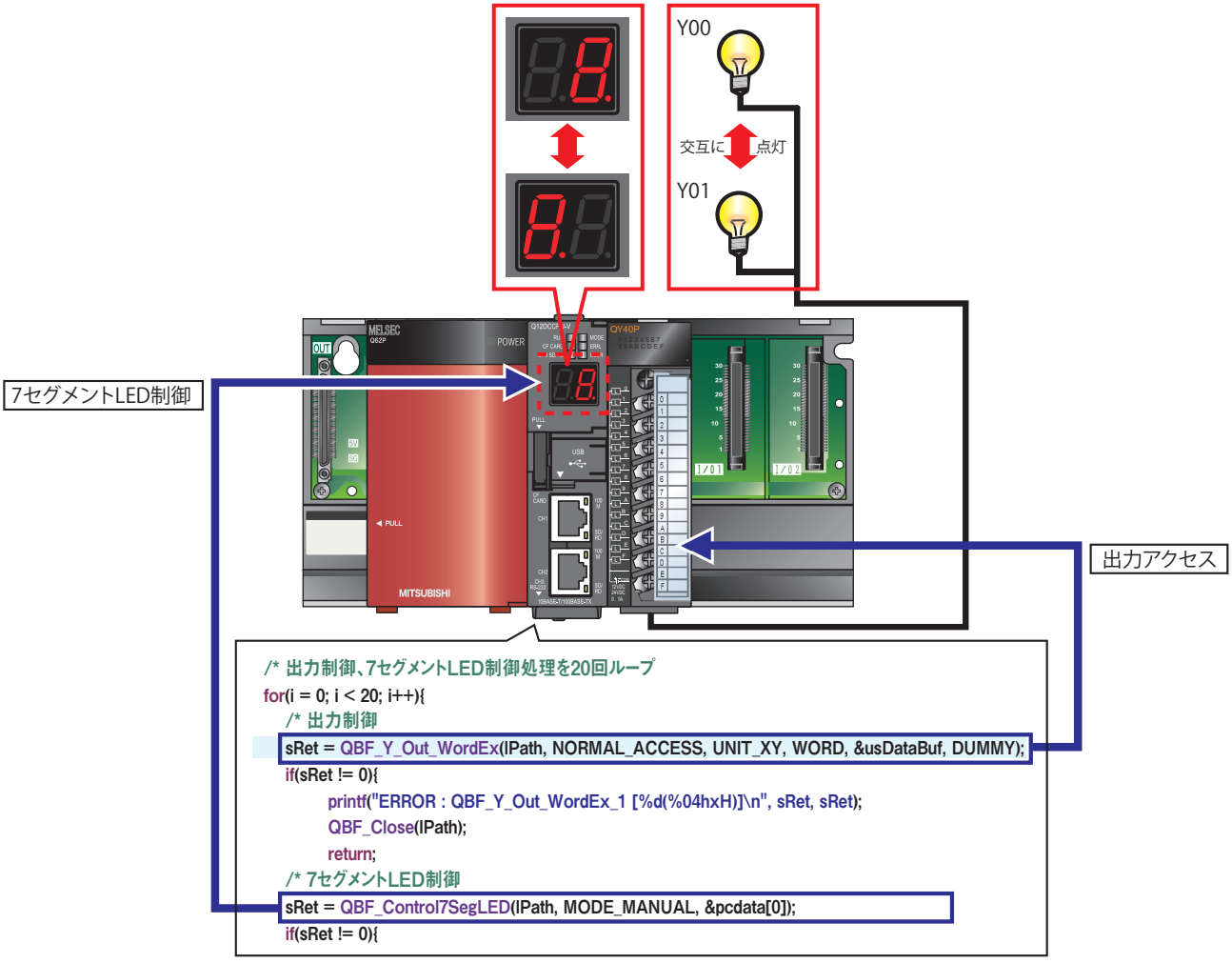
〈5〉プログラミングする

出力ユニットに接続したランプと C 言語コントローラユニット前面の 7 セグメント LED を点滅させるプログラムを作成してみましょう。

1. 今回作成するプログラムと制御内容

今回作成するプログラムでは，下記の制御を行います。

C 言語コントローラユニットを RUN すると，出力ランプ Y00，Y01 が交互に点灯を繰り返します。
また，出力ランプの点灯に同期して，C 言語コントローラユニット前面の 7 セグメント LED の一位と十位が交互に全点灯を繰り返します。



2. ソースコード

ソースコードを下記に示します。

```
1:  /*****************************************************************************************************/
2:  /* 関数ヘッダ                                                                                        */
3:  /*****************************************************************************************************/
4:  #include <vxworks.h>          /* VxWorks関数ヘッダ                                                    */
5:  #include <taskLib.h>          /* VxWorks関数ヘッダ                                                    */
6:  #include <stdio.h>            /* 標準関数ヘッダ                                                       */
7:  #include "QbfFunc.h"          /* バスインタフェース関数ヘッダ                                         */

8:  /*****************************************************************************************************/
9:  /* 定義                                                                                              */
10: /*****************************************************************************************************/
11: /* デバッグ用                                                                                        */
12: #define UNIT_XY              0x0000          /* ユニットの先頭入出力番号                            */
13: #define QY_LED               0x5555          /* Y信号の初期出力値（偶数ビット点灯）                 */
14: #define SEG_LED1             0xFF            /* 7セグメントLED出力初期値（一位）                    */
15: #define SEG_LED2             0x00            /* 7セグメントLED出力初期値（十位）                    */

16: /*****************************************************************************************************/
17: /* QBF関数用                                                                                         */
18: #define CPU_TYPE             2               /* CPU識別フラグ（CCPU：2）                            */
19: #define WORD                 1               /* ワード単位指定                                      */
20: #define NORMAL_ACCESS        0               /* 通常アクセス指定                                    */
21: #define DUMMY                0               /* ダミー                                              */
22: #define MODE_MANUAL          0               /* 7セグメントLED制御モード                            */

23: /*****************************************************************************************************/
24: /* Y出力、7セグメントLED制御処理                                                                     */
25: /*****************************************************************************************************/
26: void Q12_SampleTask()
27: {
28:     /* ローカル変数の宣言                                                                            */
29:     short           sRet;           /* QBF関数の戻り値                                              */
30:     long            lPath;          /* バスのパス                                                   */
31:     unsigned short  usDataBuf;      /* Y信号（ワード単位）                                          */
32:     unsigned short  usEmptyDataBuf; /* Y信号リセット用                                              */
33:     char            pcdata[2];      /* 7セグメントLED点灯値                                         */
34:     short           i;              /* ループ用                                                     */

35:     /* バスをオープンする                                                                            */
36:     sRet = QBF_Open(CPU_TYPE, &lPath);
37:     if(sRet != 0){
38:         printf("ERROR : QBF_Open [%d(%04hxH)]\n", sRet, sRet);
39:         return;
40:     }

41:     /* Y信号の出力値をセット（偶数ビットを点灯）                                                     */
42:     usDataBuf = QY_LED;

43:     /* 7セグメントLEDの出力値をセット（一位のみ全灯）                                                */
44:     pcdata[0] = SEG_LED1;
45:     pcdata[1] = SEG_LED2;

46:     /* 出力制御、7セグメントLED制御処理を20回ループ                                                  */
47:     for(i = 0; i < 20; i++){
48:         /* 出力制御                                                                                  */
49:         sRet = QBF_Y_Out_WordEx(lPath, NORMAL_ACCESS, UNIT_XY, WORD, &usDataBuf, DUMMY);
50:         if(sRet != 0){
51:             printf("ERROR : QBF_Y_Out_WordEx_1 [%d(%04hxH)]\n", sRet, sRet);
52:             QBF_Close(lPath);
53:             return;
54:         }

55:         /* 7セグメントLED制御                                                                        */
56:         sRet = QBF_Control7SegLED(lPath, MODE_MANUAL, &pcdata[0]);
57:         if(sRet != 0){
58:             printf("ERROR : QBF_Control7SegLED_1 [%d(%04hxH)]\n", sRet, sRet);
59:             QBF_Close(lPath);
60:             return;
61:         }

62:         /* Y信号の出力値を反転する（奇数ビット→偶数ビット→...を点灯）                                  */
63:         usDataBuf = ~usDataBuf;
```

ライブラリ関数などが使えるように，関数一覧を定義したファイルを宣言します。

今回制御するための，各値を定義します。

プログラムの最初で，バスインタフェース関数を使える状態にします。

バスインタフェース関数を使い，出力ユニットの制御を行います。

バスインタフェース関数を使い，C言語コントローラユニット前面の7セグメントLEDの制御を行います。

```
64:         /* 7セグメントLEDの出力値を反転する（一位全灯→十位全灯→...）                    */
65:         pcdata[0] = ~pcdata[0];
66:         pcdata[1] = ~pcdata[1];

67:         /* ウェイト                                                                  */
68:         taskDelay(40);
69:     }
70:     /* Y信号をリセットする                                                           */
71:     usEmptyDataBuf = 0x00;
72:     sRet = QBF_Y_Out_WordEx(lPath, NORMAL_ACCESS, UNIT_XY, WORD,
73:         &usEmptyDataBuf, DUMMY);
74:     if(sRet != 0){
75:         printf("ERROR : QBF_Y_Out_WordEx_2 [%d(%04hxH)]\n", sRet, sRet);
76:         QBF_Close(lPath);
77:         return;
78:     }

79:     /* 7セグメントLEDをリセットする                                                  */
80:     pcdata[0] = 0x00;
81:     pcdata[1] = 0x00;
82:     sRet = QBF_Control7SegLED(lPath, MODE_MANUAL, &pcdata[0]);
83:     if(sRet != 0){
84:         printf("ERROR : QBF_Control7SegLED_2 [%d(%04hxH)]\n", sRet, sRet);
85:         QBF_Close(lPath);
86:         return;
87:     }

88:     /* バスをクローズする                                                            */
89:     QBF_Close(lPath);
90:     return;
91: }
```

出力ユニットの出力およびC言語コントローラユニット前面の7セグメントLEDの表示を，すべてOFFします。

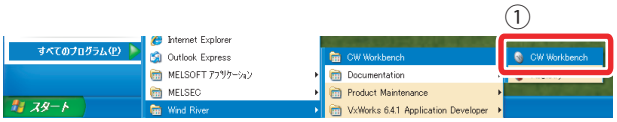
プログラムの最後で，バスインタフェース関数を使えない状態にします。

1) プロジェクトを作成する

1. CW Workbench を起動する

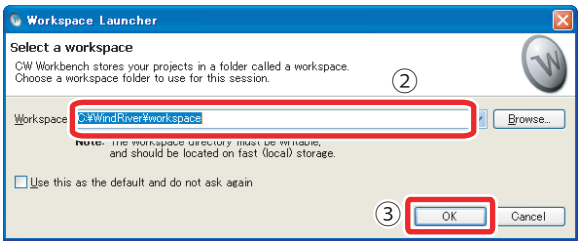
操作手順

① [スタート]→[すべてのプログラム]→[Wind River]
→ [CW Workbench] → [CW Workbench] を選択

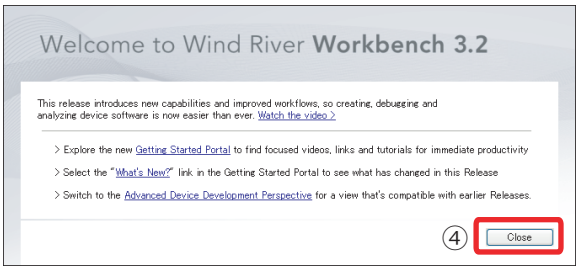


② 起動後，ワークスペースの保存先フォルダを入力します。
ここでは，"C:￥WindRiver￥workspace" とします。

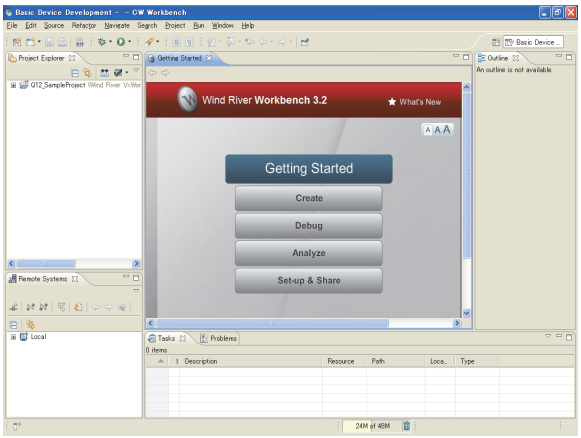
③ OK ボタンをクリック



④ Close ボタンをクリック



CW Workbench のメイン画面が表示されます。



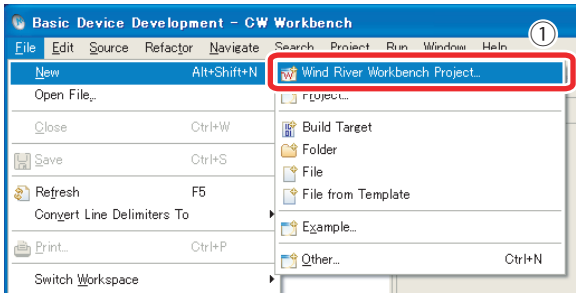
参考

- CW Workbench の初期状態の各ウィンドウの大きさやアイコンなどの配置は，お使いのパソコンによって変わります。本クイックスタートガイドに記載している画面と違う場合は，各ウィンドウの大きさを調整してください。
- 拡大したり消したりした各ウィンドウを初期状態に戻すには，メニュー [Window] → [New Window] を選択してください。

2. 新規プロジェクトを作成する

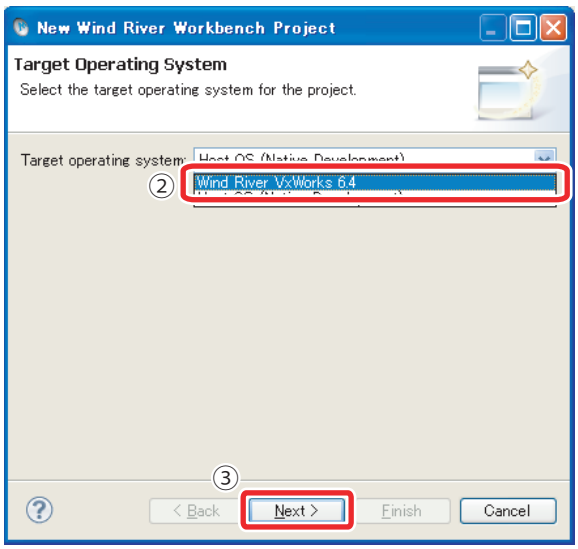
操作手順

① メニュー[File] → [New] → [Wind River Workbench Project...] を選択



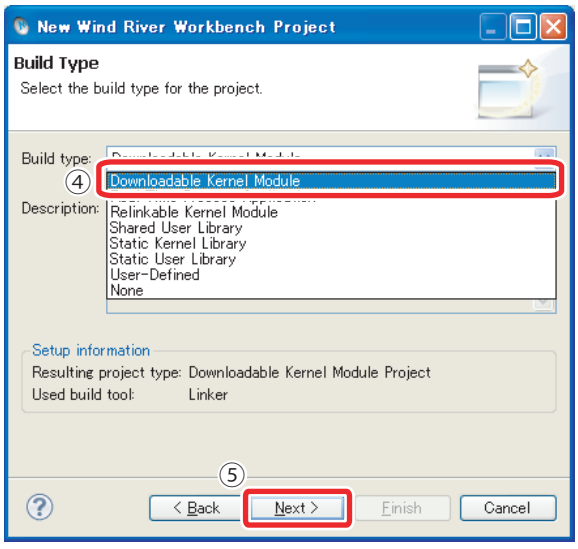
②「Wind River VxWorks6.4」を選択

③ Next > ボタンをクリック



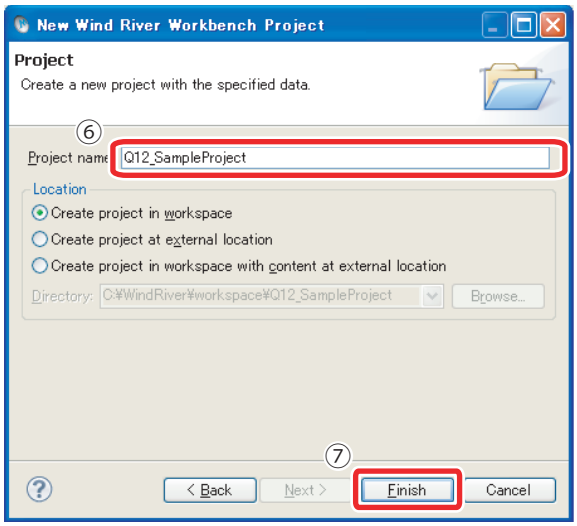
④「Downloadable Kernel Module」を選択

⑤ Next > ボタンをクリック



⑥ プロジェクト名を入力
ここでは，"Q12_SampleProject" とします。

⑦ Finish ボタンをクリック



これで，プロジェクト作成が完了しました。

3. プロジェクトのプロパティを設定する

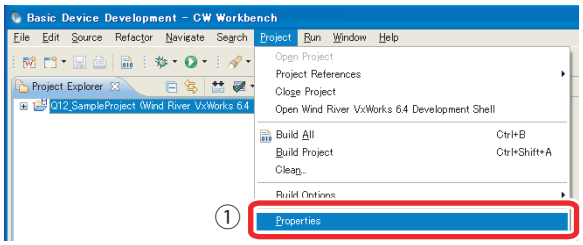
作成したプロジェクトを，C 言語コントローラユニットで実行可能なモジュールに変換 (ビルド) するための設定を行います。

用語

ビルド：プロセッサに応じたソースコードのコンパイル，インクルードファイルとのリンクを行います。

(1) 使用するプロセッサを設定

① Project Explorer ウィンドウから作成したプロジェクトを選択し，メニュー[Project] → [Properties] をクリック

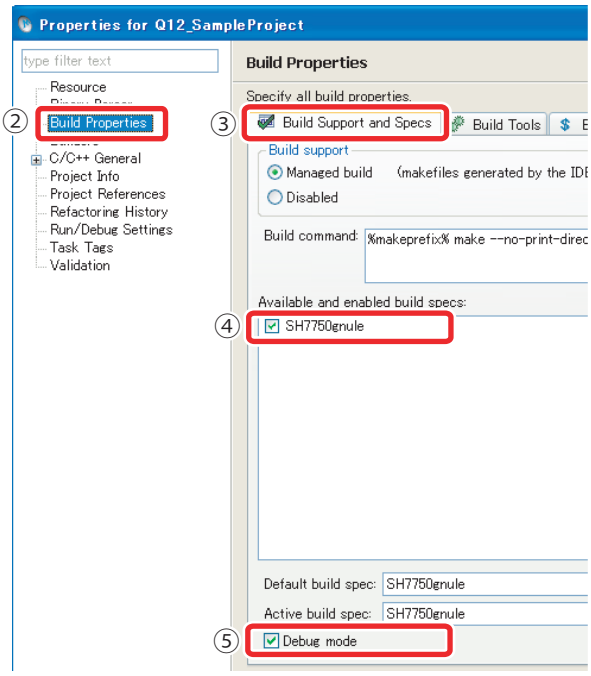


② 画面左側のツリーから「Build Properties」を選択

③「Build Support and Specs」タブをクリック

④「Available and enabled build specs」で「SH7750gnule」のみをチェック

⑤「Debug mode」をチェック



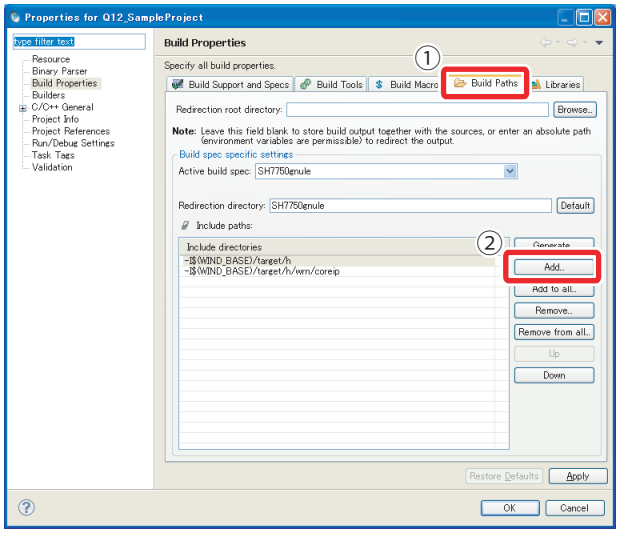
Point

正式なシステム運用，稼動時は，「Debug mode」のチェックをはずしてください。

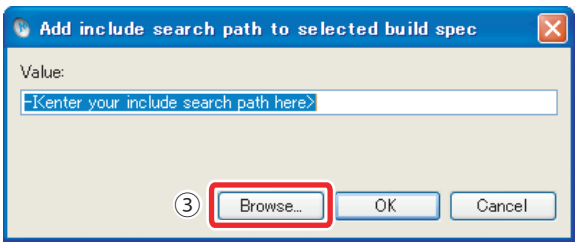
(2) インクルードファイルを設定

①「Build Paths」タブをクリック

② Add... ボタンをクリック

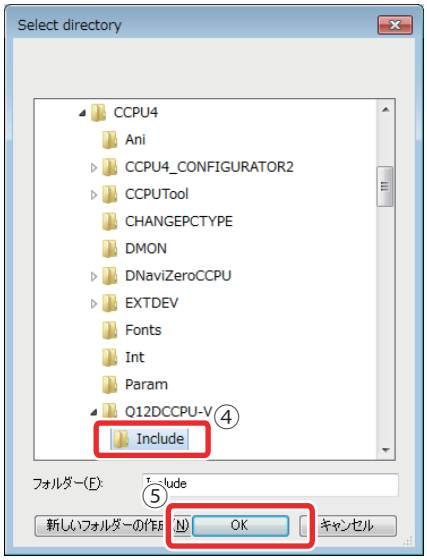


③ Browse... ボタンをクリック



④「Select directory」画面から C 言語コントローラユニット専用のインクルードフォルダを選択
ここでは C 言語コントローラ設定・モニタツールを"C:¥MELSEC"へインストールした場合のフォルダになります。

⑤ OK ボタンをクリック

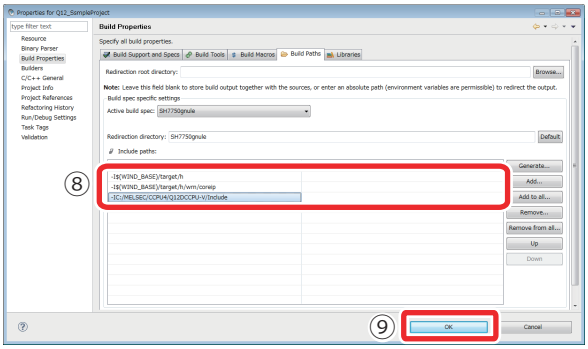


⑥「Select directory」画面で選択したフォルダが指定されていることを確認

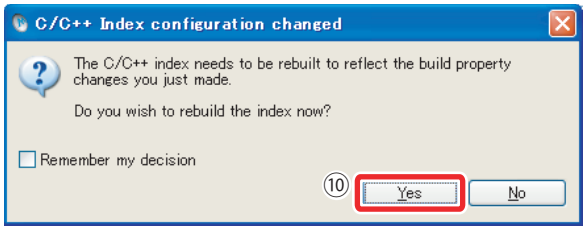
⑦ OK ボタンをクリック

⑧「Include paths」で追加したインクルードパスが表示されていることを確認

⑨ OK ボタンをクリック



⑩ OK ボタンをクリック後，下記のメッセージが表示された場合は，Yes ボタンをクリック



これで，プロジェクトのプロパティ設定が完了しました。

2) ユーザプログラムを作成する

C 言語コントローラシステムの制御を行うユーザプログラムを作成しましょう。

下記 1. で示すユーザプログラムの作成は，本クイックスタートガイド専用のサンプルプログラムを使用することで省略できます。サンプルプログラムを使用する場合は，2. から始めてください。

参考

本クイックスタートガイド専用のサンプルプログラムは，下記を参照してください。

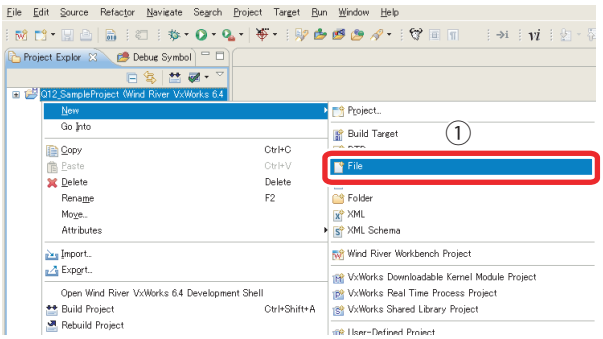
☞ 三菱電機 FA サイトホームページ
(www.MitsubishiElectric.co.jp/fa)

・C 言語コントローラユニット クイックスタートガイド

1. ユーザプログラムを作成する

操作手順

① Project Explorer ウィンドウから作成したプロジェクトを選択して右クリックし，メニュー [New] → [File] をクリック

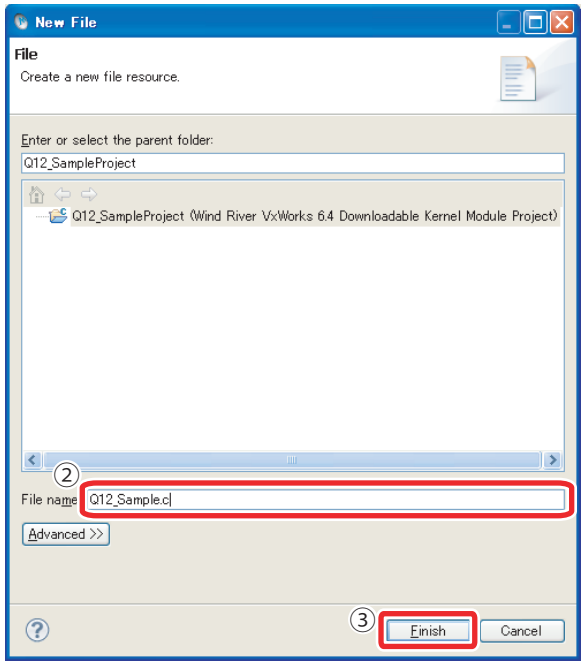


② 「File Name」に新規作成するソースファイル名を入力
ここでは "Q12_Sample.c" と入力します。

Point

ファイル名は拡張子まで入力してください。
またファイル名に全角文字は使用できません。使用すると，コンパイル時にコンパイルエラーが発生します。

③ Finish ボタンをクリック



④ Editor ウィンドウで，出力ユニットへのアクセスおよび 7 セグメント LED 制御するための「ソースコード」(P.28) を記述します。

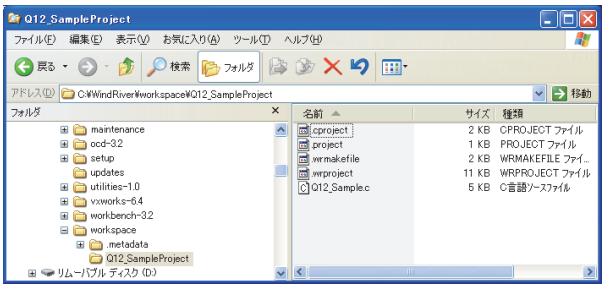
ソースコードの記述が終了したら「3) ユーザプログラムから実行モジュールを生成する」(P.36) へ移ってください。

2. サンプルプログラムを追加する

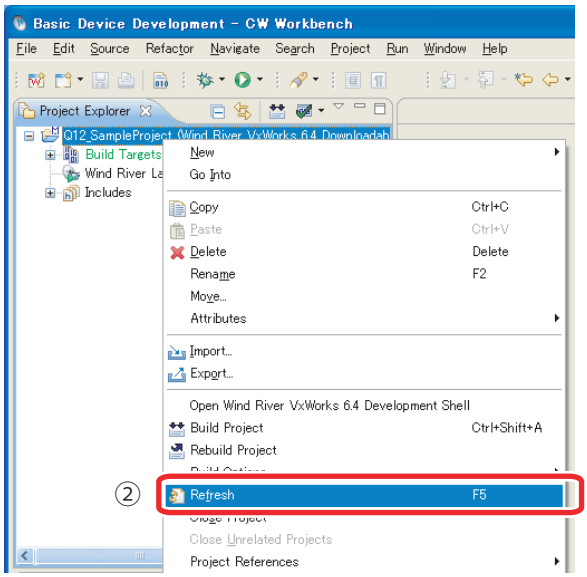
操作手順

① 今回作成したプロジェクトフォルダ直下に，本クイックスタートガイド専用のサンプルプログラムを格納します。

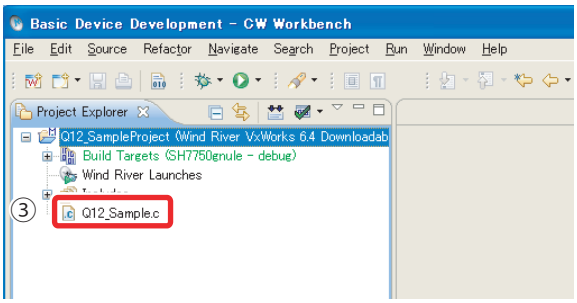
"C:￥WindRiver￥workspace￥Q12_SampleProject"



② Project Explorer ウィンドウから作成したプロジェクトを選択して右クリックし，メニュー [Refresh] をクリック



③ 手順①で格納したサンプルプログラムがプロジェクトに追加されます。



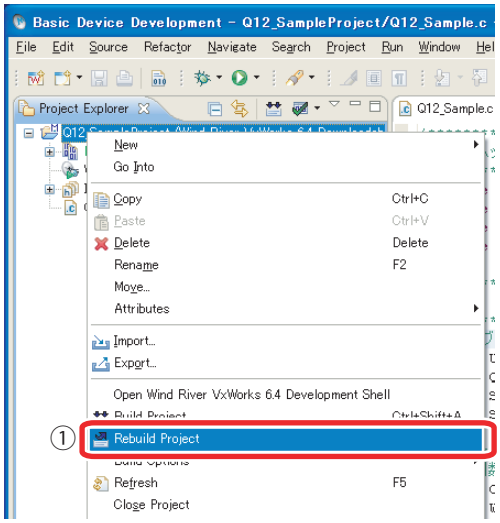
これで，サンプルプログラムの追加が完了しました。

3) ユーザプログラムから実行モジュールを生成する

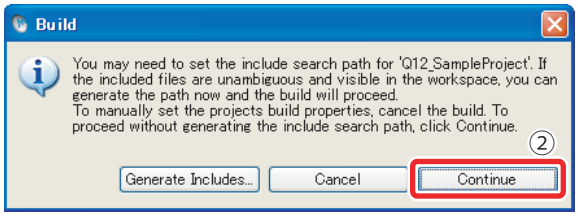
作成したプログラムを，C 言語コントローラユニットで実行可能なモジュールに変換 (ビルド) します。

操作手順

① Project Explorer ウィンドウから作成したプロジェクトを選択して右クリックし，メニュー [Rebuild Project] をクリック

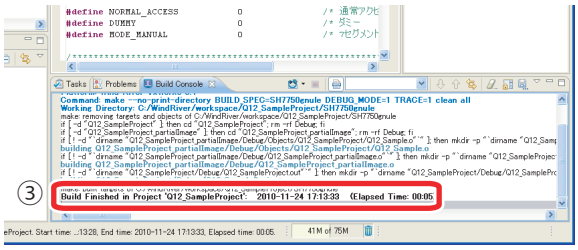


② メニュー [Rebuild Project] をクリック後，下記のメッセージが表示された場合は，Continue ボタンをクリック



プロジェクトのビルドが開始され，Build Console ウィンドウにビルドの処理過程が表示されます。

③ Build Console ウィンドウで，"Build Finished..." が表示されることを確認してください。



"Build Finished..." が表示されると，ユーザプログラムの作成およびビルドは完了です。

Point

"Build Finished..."が表示されずエラーが発生した場合は，エラー内容を確認してプログラムを修正してください。
プログラムを修正後，再度「3) ユーザプログラムから実行モジュールを生成する」(P.36) から実施してください。

4) C 言語コントローラユニットと CW Workbench を接続する

CW Workbench でデバッグを行うために，C 言語コントローラユニットと CW Workbench を接続します。

本手順の前に，あらかじめ 47 ページの手順 1 を実行して，ユーザプログラムからの出力（Y）を許可にしておいてください。

操作手順

① C 言語コントローラユニットから VxWorks イメージファイルを取得するために，エクスプローラを起動し，アドレス欄に下記の形式で入力します。

ftp://192.168.3.3/SYSTEMROM/OS_IMAGEFILE/

「ログイン」画面が表示されます。

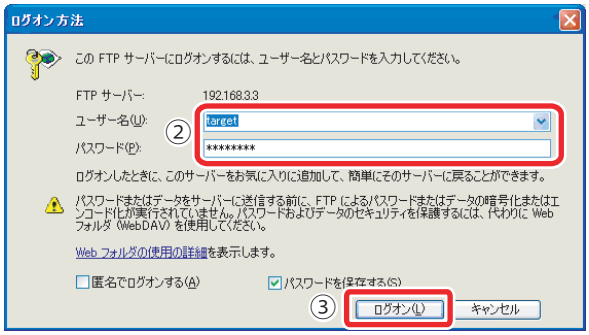
Point

C言語コントローラユニットとパソコンで通信する場合は，双方で同じ VxWorks イメージファイルを指定する必要があります。

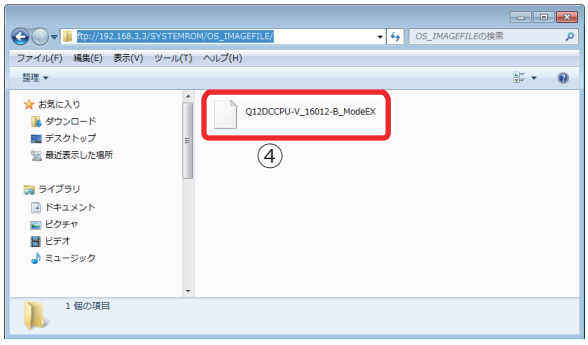
②「ログイン」画面で下記のユーザ名とパスワードを入力

- ユーザ名　　：target
- パスワード　：password

③ ログオン(L) ボタンをクリック



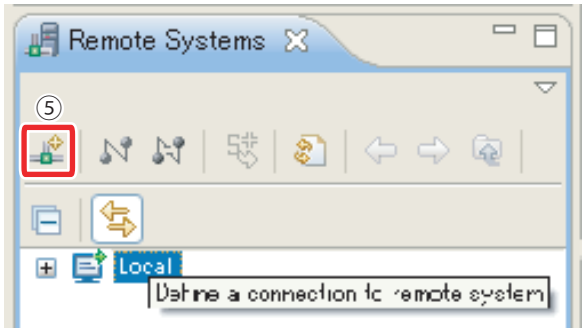
④ C 言語コントローラユニットに格納されている VxWorks イメージファイルを，「C:￥MELSEC￥CCPU4￥CCPUTool」へコピーします。



Point

「C:￥MELSEC￥CCPU4￥CCPUTool」は，C 言語コントローラ設定・モニタツールを「C:￥MELSEC」にインストールした場合に作成されるフォルダです。

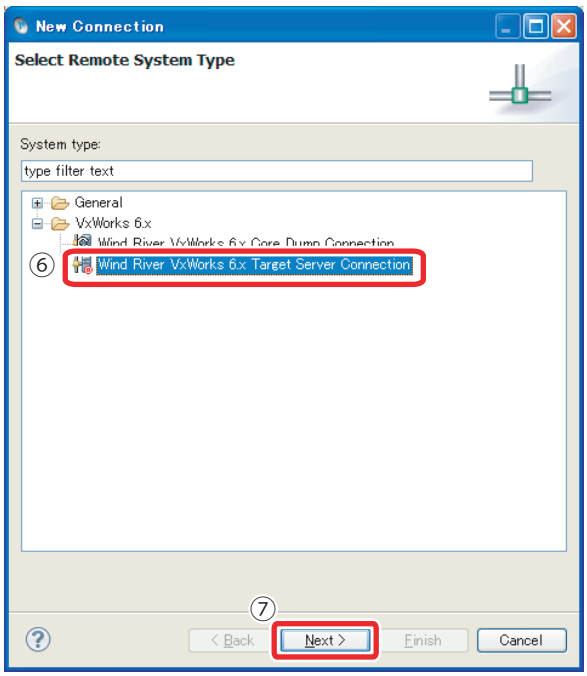
⑤ Remote Systems ウィンドウ内の をクリック



「New Connection」画面が表示されます。

⑥「New Connection」画面にて「Wind River VxWorks 6.x Target Server Connection」を選択

⑦ Next > ボタンをクリック

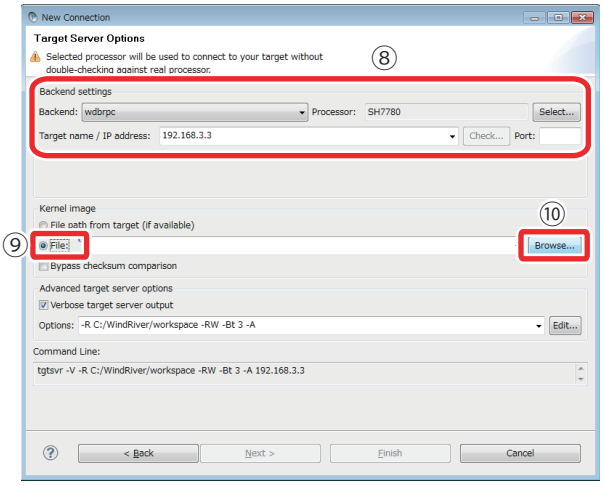


⑧「Backend settings」の設定項目で，下記の内容を設定

- Backend　：wdbrpc
- Processor　：SH7780(Select... ボタンをクリックし，ツリーから選択)
- IP Address　：192.168.3.3(デフォルト)
- Port　：空欄

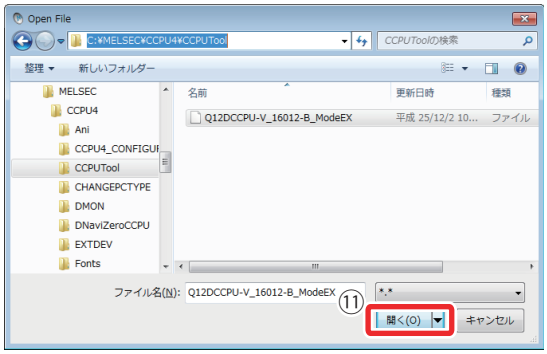
⑨「Kernel image」で「File」を選択

⑩ Browse... ボタンをクリック

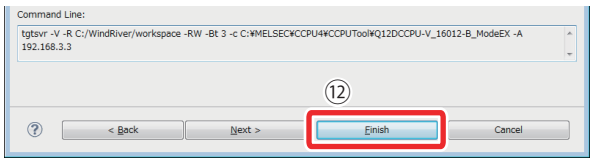


「Open File」画面が表示されます。

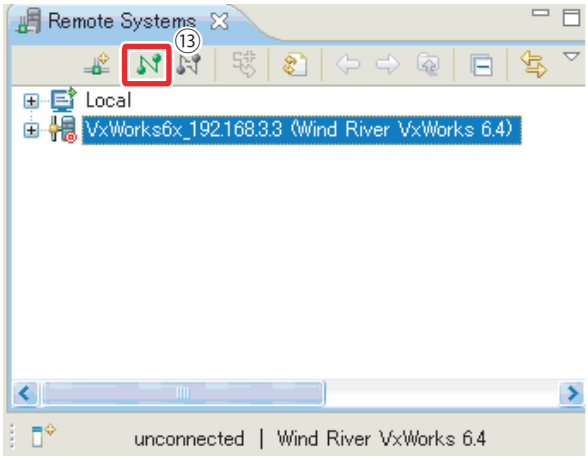
⑪ 手順④でコピーした VxWorks イメージファイルをツリーから選択 (C:￥MELSEC￥CCPU4￥CCPUTool) し，開く(O) ボタンをクリック



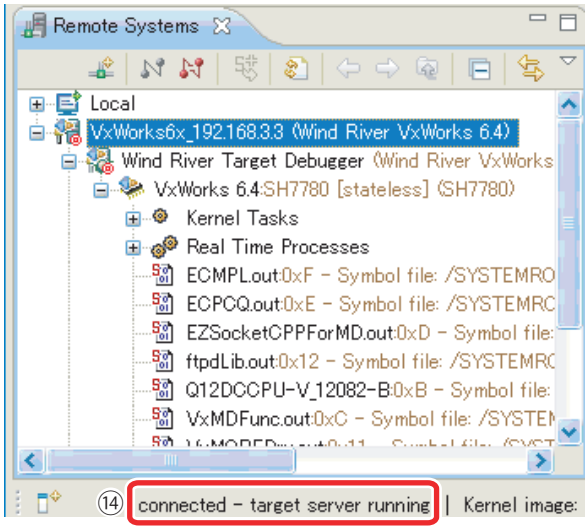
⑫ Finish ボタンをクリック



⑬ Remote Systemsウィンドウで追加したターゲットサーバを選択し，をクリック



⑭ をクリック後，Remote Systems ウィンドウの下部に "connected - target server running" が表示されると接続完了です。



Point

"connected - target server running" が表示されない場合は，C 言語コントローラユニットの電源が正常に投入されていることを確認し，再度「「4) C 言語コントローラユニットと CW Workbench を接続する」(P.37) から実施してください。

5) ユーザプログラムをデバッグする

作成したプログラムが正しく動作するか確認します。

1. ユーザプログラムを C 言語コントローラユニットにダウンロードする

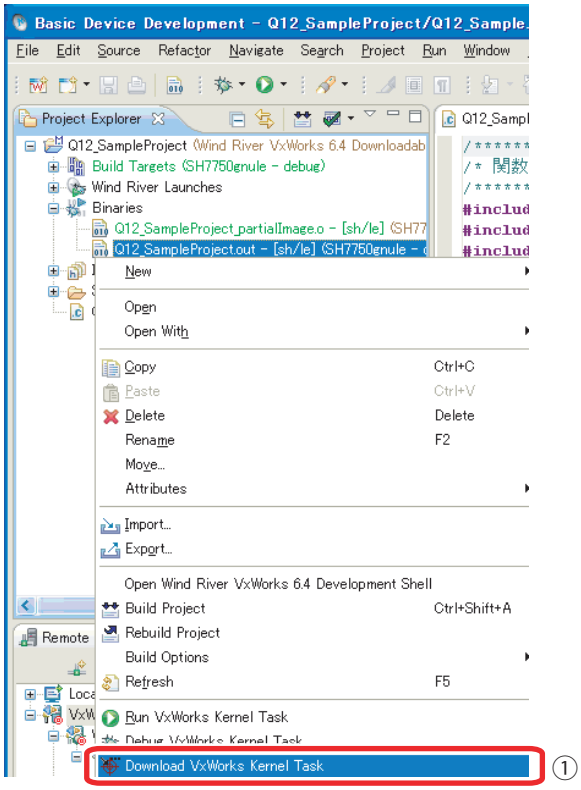
デバッグを行うために，実行モジュールを C 言語コントローラユニットのメモリにダウンロードします。

ダウンロードを行うと，スクリプトファイルがなくてもプログラムが実行できます。

用語

スクリプトファイル：C 言語コントローラユニットを立上げ時に起動するユーザプログラムのロード先，起動順序などを記述するファイルです。

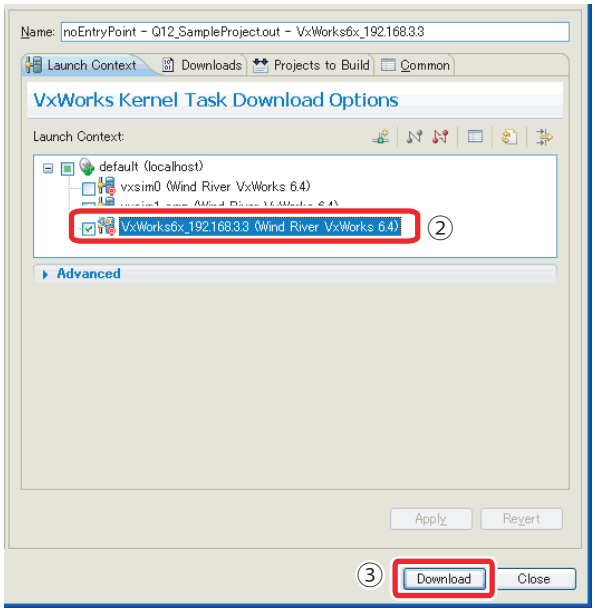
① Project Explorer ウィンドウから作成したモジュールファイル"Q12_SampleProject.out"を選択して右クリックし，メニュー [Download VxWorks Kernel Task] をクリック



「Download Configurations」画面が表示されます。

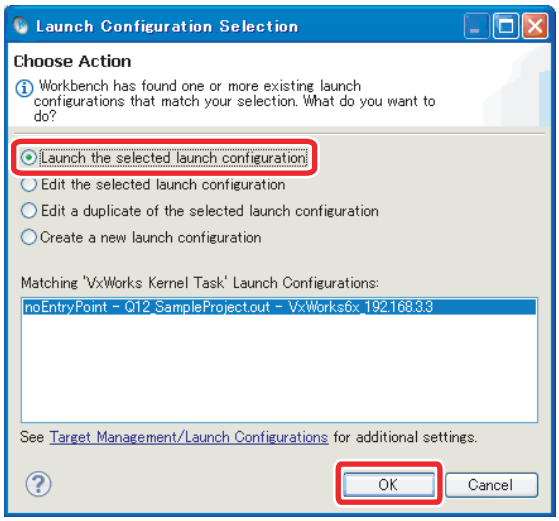
② 「Launch Context」にて「VxWorks6x_192.168.3.3 (Wind River VxWorks 6.4)」のみをチェック

③ Download ボタンをクリック



Point

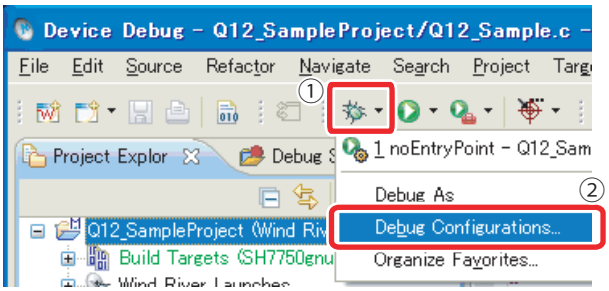
2 回目以降の②の操作では，
「Launch Configuration Selection」画面が表示されます。「 Launch the selected launch configuration 」を選択し，OK ボタンをクリックしてください。



2. ユーザプログラムをデバッグする

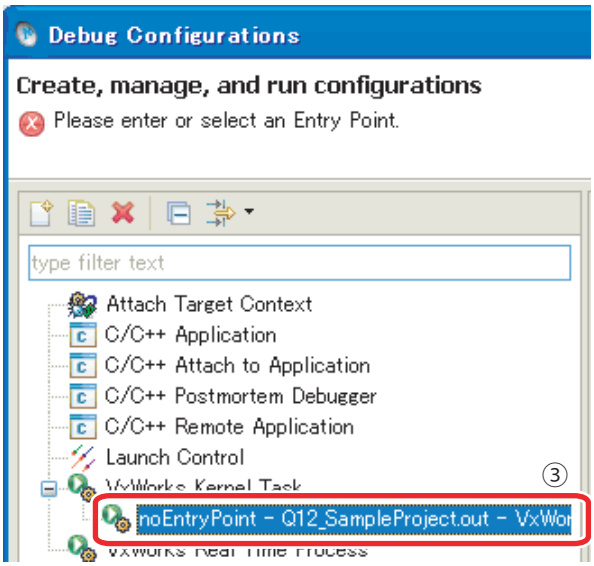
① Project Explorer ウィンドウから，作成したプロジェクトを選択し，ツールバーの 右脇の▼をクリック

② メニュー [Debug Configurations...] をクリック



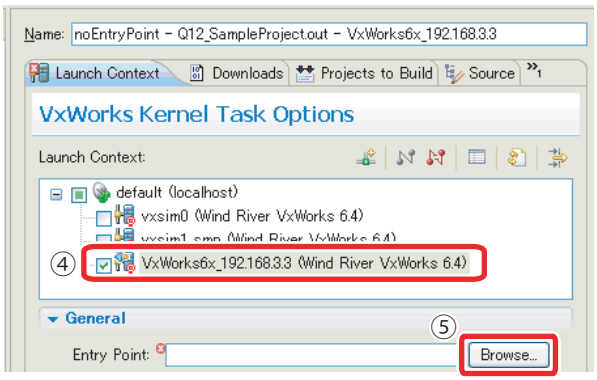
「Debug Configurations」画面が表示されます。

③「VxWorks Kernel Task」からダウンロードしたモジュール "Q12_SampleProject.out" をクリック



④ C 言語コントローラユニットとの接続を示すターゲットサーバをチェック

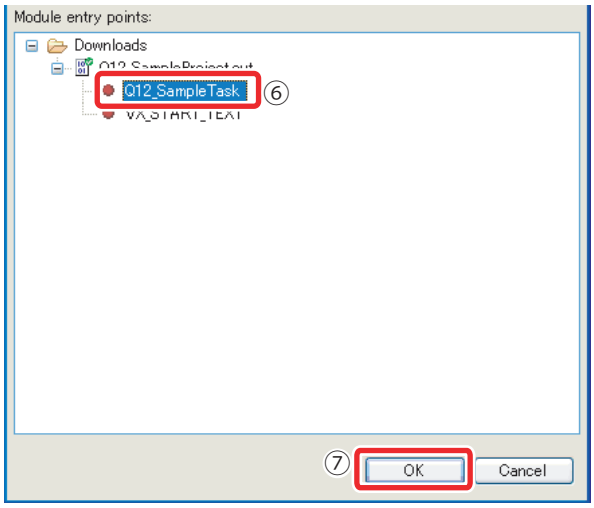
⑤ Browse... ボタンをクリック



「Entry Points」画面が表示されます。

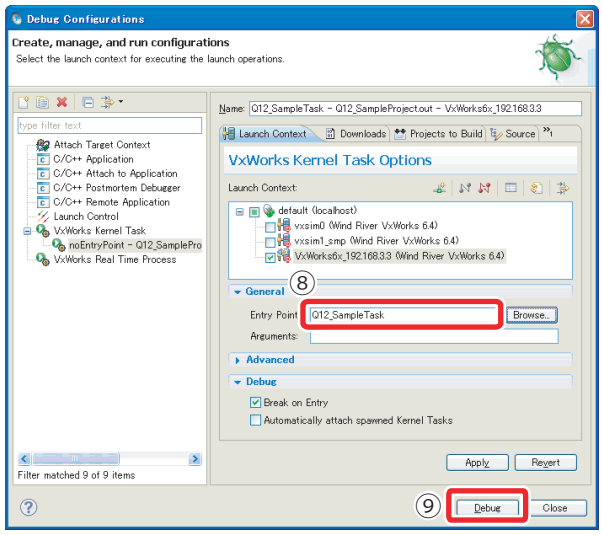
⑥ デバッグを開始する関数(Q12_SampleTask)を選択

⑦ OK ボタンをクリック

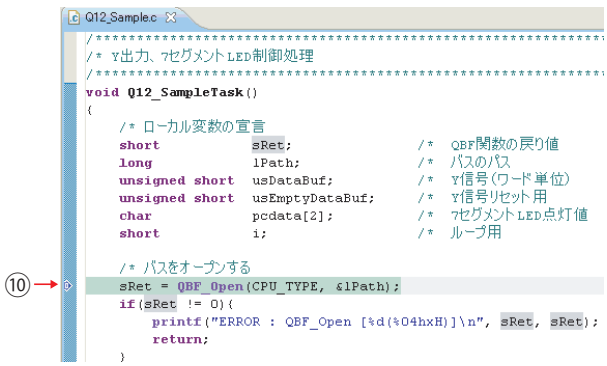


⑧「Entry Point」に手順⑥で選択した関数名が設定されていることを確認

⑨ Debug ボタンをクリック

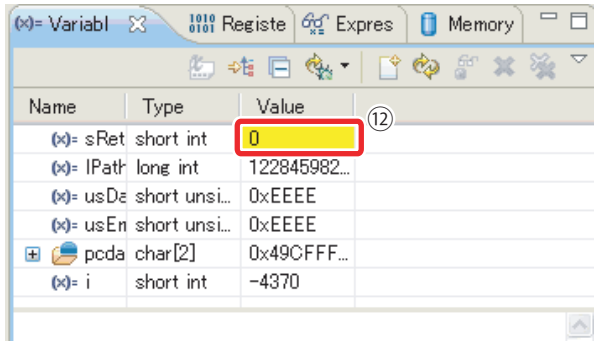


⑩ デバッグが開始され，「Entry Point」で指定した関数の先頭でプログラムの実行が止まります。

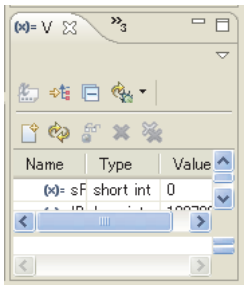


⑪ Debug ウィンドウの で，1 ステップごとにデバッグを行います。

⑫ 画面右下の「Variables」ウィンドウ *1 のタブをクリックし，各変数の値の確認や変更を行うことができます。
ここでは "QBF_Open" 関数の戻り値である "sRet" が 0(正常値) であることを確認します。



＊1　お使いのパソコンによっては，「Variables」ウィンドウが下記のように表示される場合がありますので，ウィンドウの大きさを調節してください。



手順⑪⑫で，作成したプログラム全体をデバッグします。

参考

バスインタフェース関数の戻り値が0以外の場合は，下記を参照してトラブルシューティングしてください。

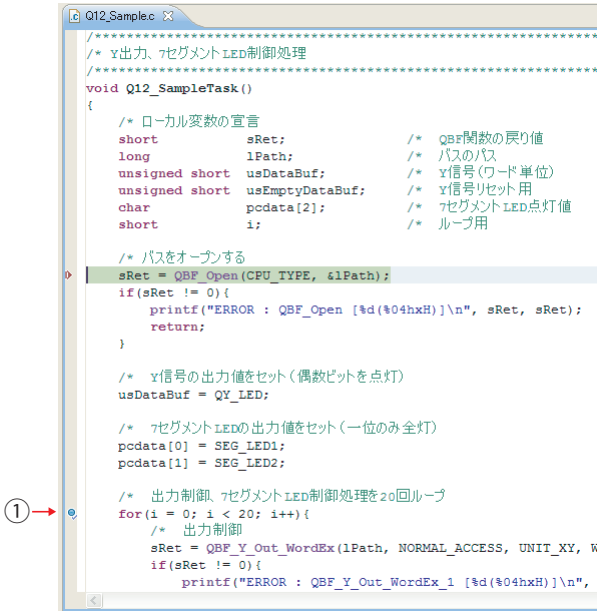
☞ SW □ PVC-CCPU のバスインタフェース関数 HELP

☞ C 言語コントローラユニットユーザーズマニュアル (ハードウェア設計・機能解説編)：SH-080764

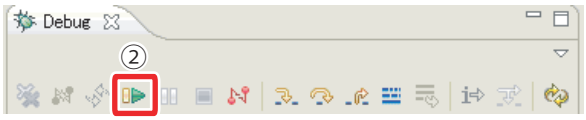
＜ Breakpoint を使用したデバッグ方法＞

左記⑪の 1 ステップごとのデバッグではなく，プログラム内の任意の位置に Breakpoint を指定してデバッグを進めることができます。

① ソースファイルの左端をダブルクリックし，Breakpoint 挿入します。



② をクリック



指定した Breakpoint の位置までプログラムが実行されます。

参考

各アイコンの内容は下記のとおりです。

: 1 ステップ実行
1 ステップ単位で実行し，関数の場合は当該関数の中に入りステップ実行を続けます。

: 1 関数単位実行
1 ステップ単位で実行し，関数の場合は当該関数へは入らず関数単位でステップの実行を続けます。

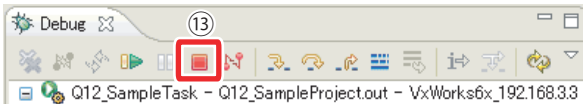
: 関数の最後 (Return) まで実行します。

: プログラム実行

: プログラム停止

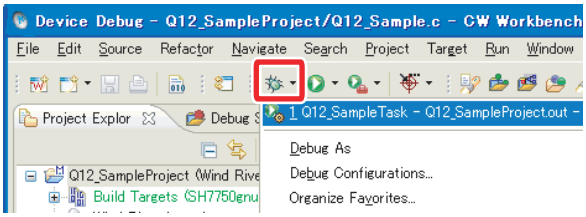
: デバッグ終了

⑬ Debug ウィンドウの をクリックし，デバッグを終了します。



Point

再度デバッグを開始する場合は，ツールバーの右脇の▼をクリックし，表示されたポップアップメニュー上部にある生成済みデバッグ構成を選択することでデバッグを開始できます。
上記の手順①～⑩を省くことができます。



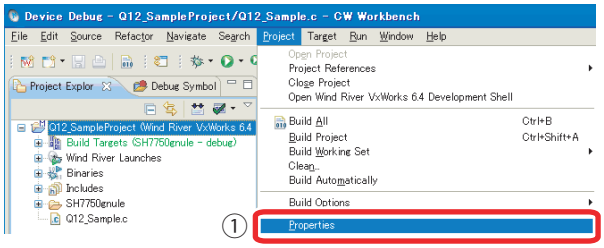
6) 実行モジュールを登録する

作成したプログラムを稼動用にビルドし，C 言語コントローラユニットに格納します。

操作手順

1. ユーザプログラムをビルドする

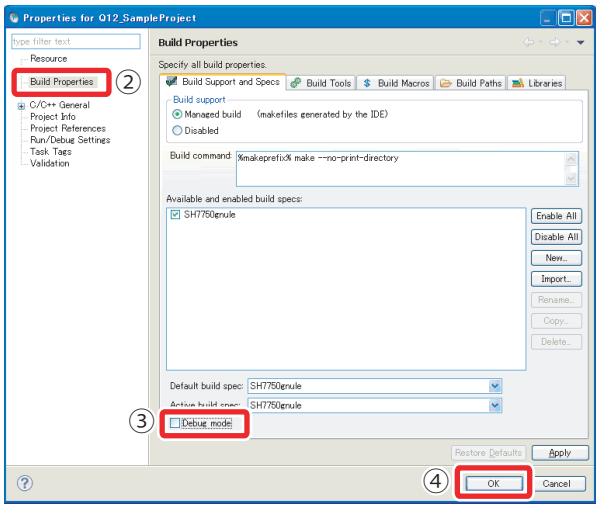
① Project Explorer ウィンドウから作成したプロジェクトを選択し，メニュー[Project] → [Properties] をクリック



② 画面左側のツリーから「Build Properties」を選択

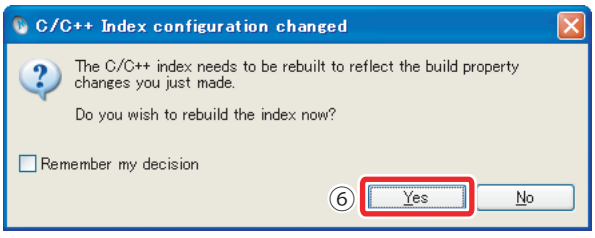
③「Debug mode」のチェックをはずす。

④ OK ボタンをクリック



⑤「3)ユーザプログラムから実行モジュールを生成する」(P.36) に従ってビルドします。

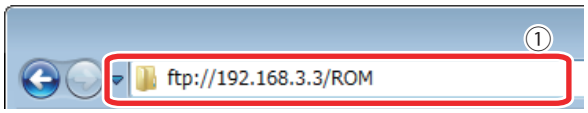
⑥ 下記のメッセージが表示された場合は，Yes ボタンをクリックしてください。



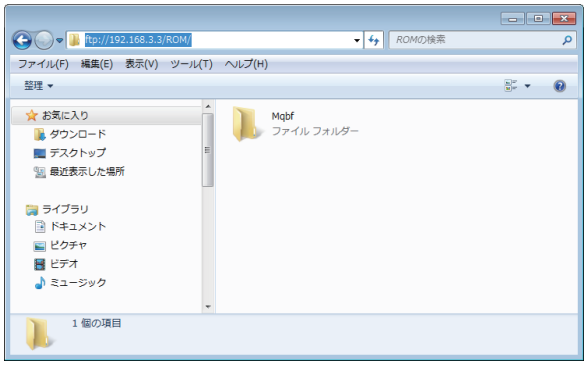
2. ユーザプログラムを格納する

① エクスプローラを起動し，C 言語コントローラユニットのアドレス欄を，下記の形式で入力します。

ftp://192.168.3.3/ROM



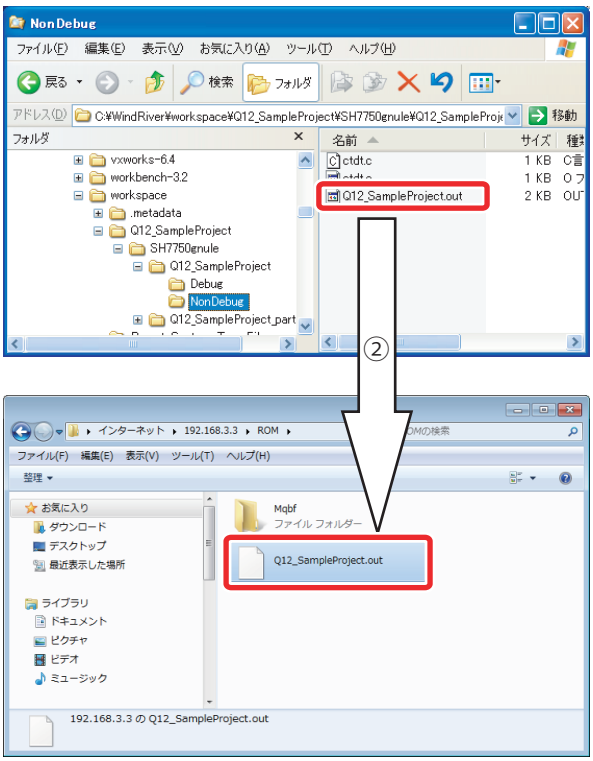
C 言語コントローラユニットへログインすると，下記のように表示されます。



② 作成したユーザプログラム "Q12_SampleProject.out" をドラッグ＆ドロップで C 言語コントローラユニットの標準 ROM へコピーします。

本クイックスタートガイドで作成したユーザプログラムは，下記に格納されます。

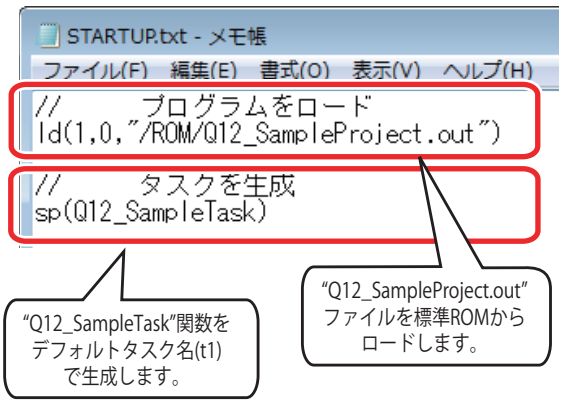
C: ￥WindRiver ￥workspace ￥Q12_SampleProject ￥SH7750gnule￥Q12_SampleProject￥NonDebug



3. スクリプトファイルを作成・格納する

C 言語コントローラユニットを起動時に，実行モジュールを自動的にロードするためのスクリプトファイルを作成します。

① テキストファイルを開き，下記のようにユーザプログラムのロード，タスク生成を行うスクリプトファイルを記述します。



② ファイル名を "STARTUP.CMD" として保存します。

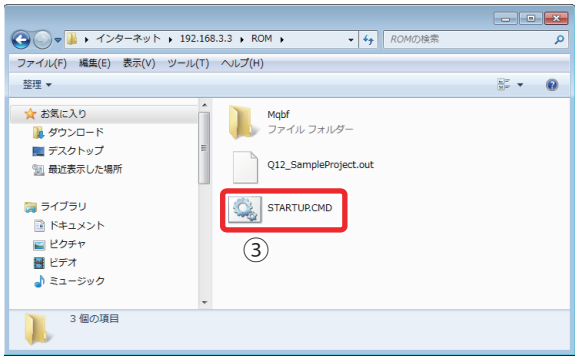
参考

今回のスクリプトファイルは，下記からダウンロードできます。

☞ 三菱電機 FA サイトホームページ
(www.MitsubishiElectric.co.jp/fa)

・C 言語コントローラユニット クイックスタートガイド

③ 作成したスクリプトファイルを，C 言語コントローラユニットの標準 ROM へコピーします。
ftp://192.168.3.3/ROM



これでスクリプトファイルの作成・格納は完了です。

Point

ユーザプログラム，スクリプトファイルは標準 ROM 以外に，コンパクトフラッシュカードにも格納することができます。
スクリプトファイルを両方に格納した場合は，コンパクトフラッシュカード内のスクリプトファイルが優先的に起動します。

〈6〉動作を確認する

C 言語コントローラユニットへ登録したプログラムを実行し，動作を確認します。
操作には C 言語コントローラユニット前面の「RUN/STOP/MODE」スイッチと「RESET/SELECT」スイッチを使用します。
［RUN/STOP/MODE スイッチの用途］
・RUN　：ユーザプログラムからの出力 (Y)，バッファメモリ書込みを許可
・STOP　：ユーザプログラムからの出力 (Y)，バッファメモリ書込みを禁止
・MODE　：ハードウェア自己診断機能などで使用

［RESET/SELECT スイッチの用途］
・RESET　：ハードウェアリセット，プログラムリセット
・SELECT　：ハードウェア自己診断機能などで使用

Point

C 言語コントローラユニットのプログラムの演算は，スイッチの状態が RUN/STOP にかかわらず実行されます。

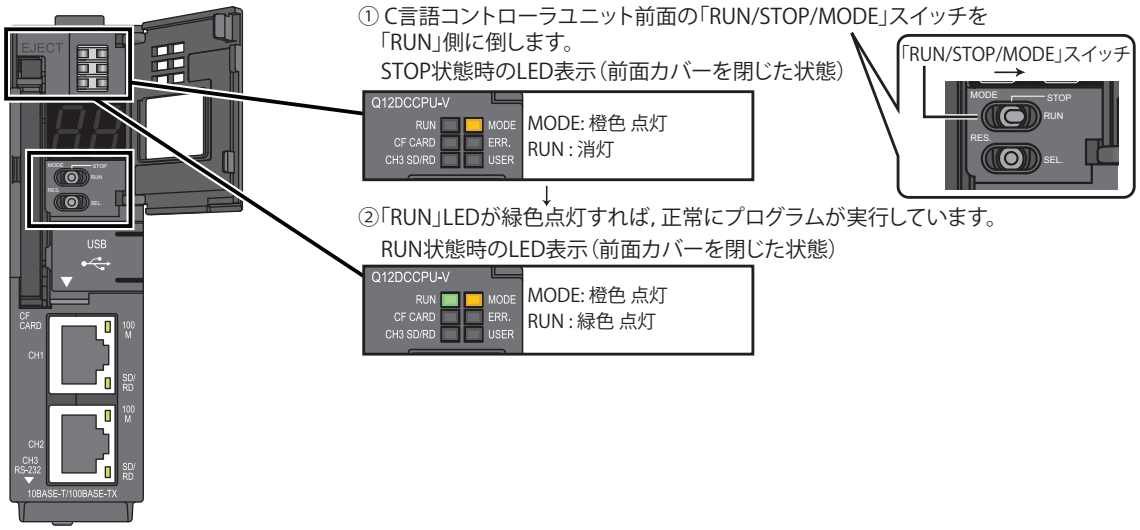
参考

「RUN/STOP/MODE」スイッチおよび「RESET/SELECT」スイッチの詳細については，下記マニュアルを参照してください。

☞ C 言語コントローラユニットユーザーズマニュアル (ハードウェア設計・機能解説編)
：SH-080764

操作手順

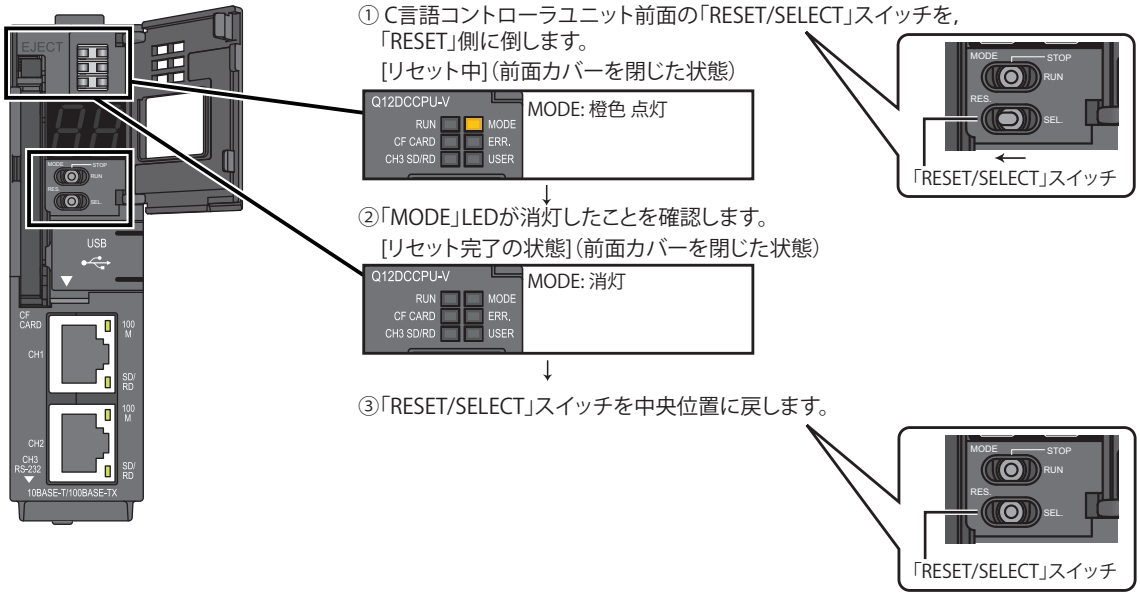
1. ユーザプログラムからの出力 (Y) を許可



Point

「RUN/STOP/MODE」スイッチを「STOP」の位置にすると，ユーザプログラムからの出力 (Y) は禁止されます。

2. C 言語コントローラユニットのリセットを実行



参考

「ERR.」LED が点灯／点滅した場合は，下記マニュアルを参照してトラブルシューティングしてください。

☞ C 言語コントローラユニットユーザーズマニュアル (ハードウェア設計・機能解説編) ：SH-080764

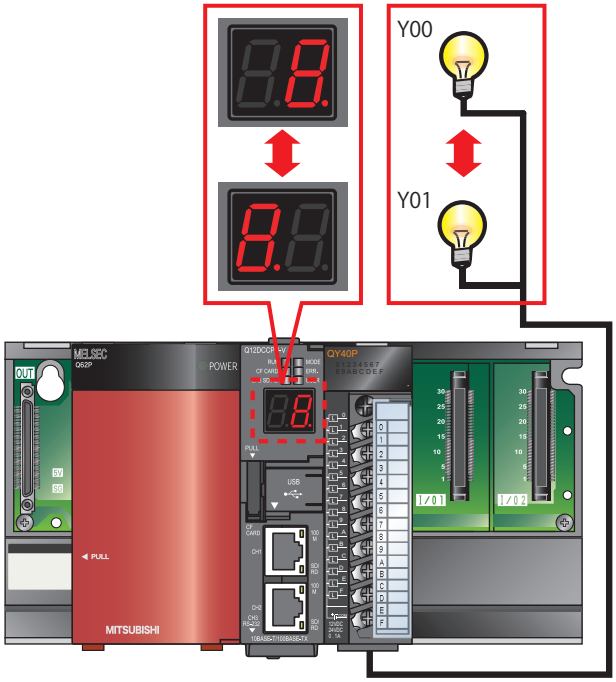
〈6〉

3. 7 セグメント LED，ランプを使って，動作を確認する

C 言語コントローラユニット前面の 7 セグメント LED と出力ランプが，下記のように点灯します。

① C 言語コントローラユニット前面の 7 セグメント LED が，交互に 20 回，全点灯を繰り返します。

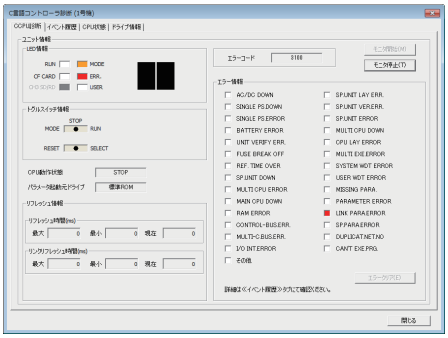
② ①に同期し，出力ランプ Y00 と Y01 が交互に点灯を繰り返します。

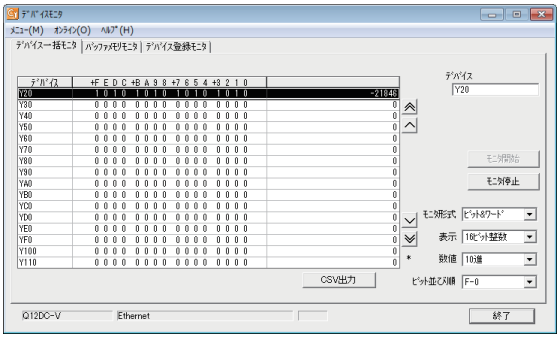


③ 再度，確認したい場合は，C 言語コントローラユニットをリセットします。

参考

7 セグメント LED および出力ランプの状態は，C 言語コントローラ設定・モニタツールでも確認できます。(P.49)





よく使う機能

C 言語コントローラユニットで，システム立上げ時や運用・稼動後の保守などによく使う機能を紹介します。

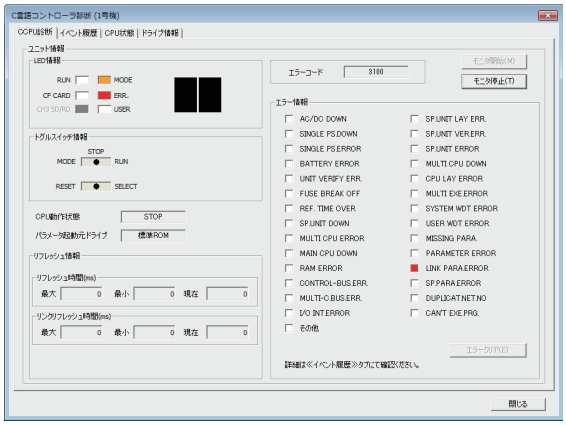
〈1〉エラーを確認して対処する

C 言語コントローラ設定・モニタツールを使って，発生したエラーの内容を確認し，対処することができます。

1) エラーが発生した場合の確認と対処方法

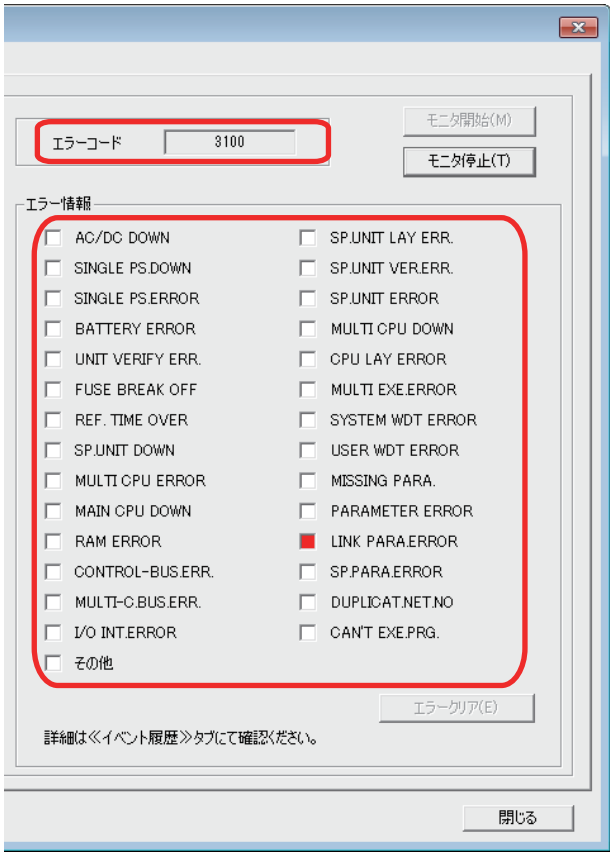
操作手順

① [診断] → [CCPU 診断] を選択します。



② 画面にエラーコードが表示されます。

③ 発生したエラーに該当するエラー項目が■(赤)になります。
モニタ中は常に最新のエラーコードに更新されます。



2) 今までに発生したエラー履歴の確認方法

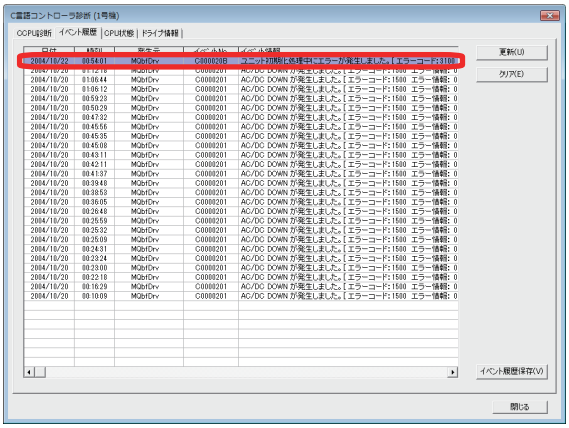
今までに発生したエラーの履歴と詳細情報を確認することができます。
いつ，どのようなエラーが発生したかを知ることができ，トラブルの解析に役立ちます。

操作手順

① [診断] → [イベント履歴] を選択します。

② 今までに発生したエラーの履歴と詳細情報が表示されます。

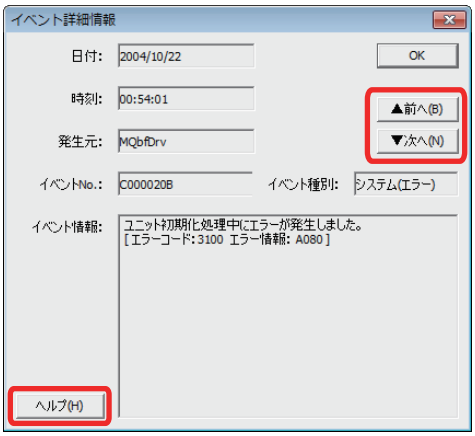
③ 詳細を知りたいエラーの行をダブルクリックします。



「イベント詳細情報」画面が表示されます。

④ ▲前へ(B) または ▼次へ(N) ボタンをクリックすると，直前または直後のエラーの詳細情報に切り替わります。

⑤ ヘルプ(H) ボタンをクリックすると，選択しているエラーのヘルプが表示されます。



〈2〉ユニット状態のモニタと動作テストを行う

C 言語コントローラ設定・モニタツールを使って，ユニットの入出力やバッファメモリの状態が確認できます。また，立上げ時や保守時などに，一時的な入出力の確認や動作テストができます。

1) ユニットの入出力とバッファメモリの状態の確認方法

ユニットの入力 (X)，出力 (Y)，バッファメモリの状態をモニタできます。

用語

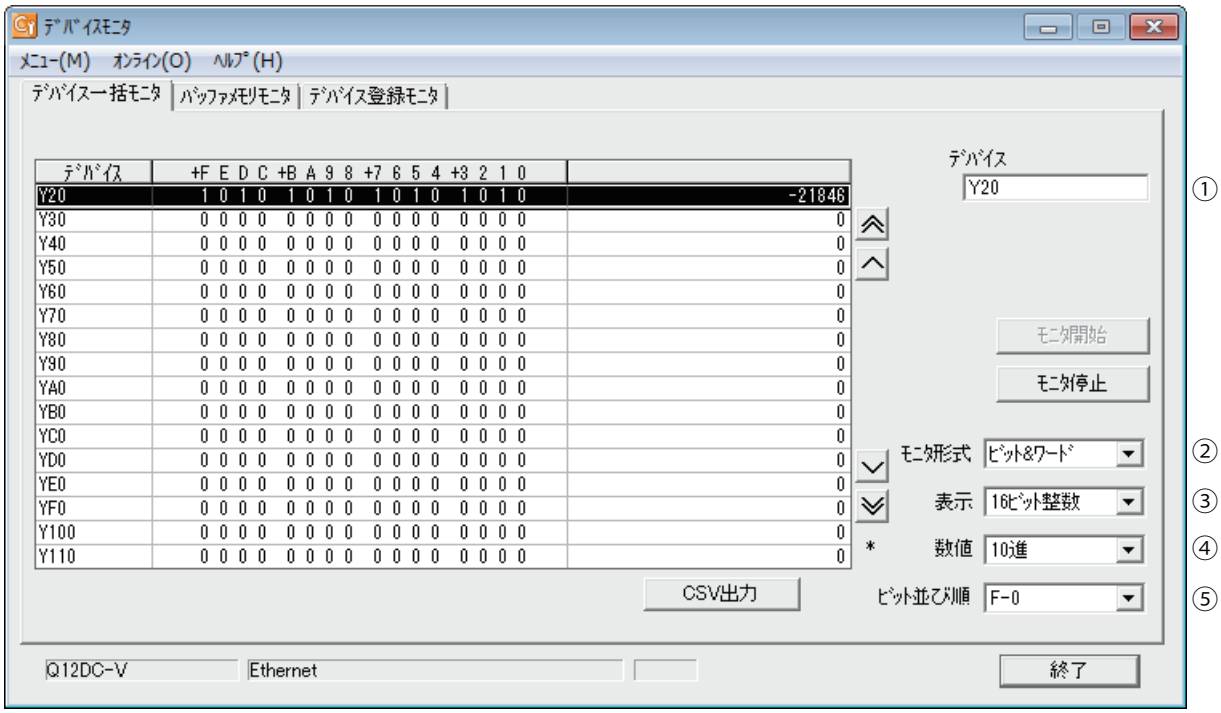
バッファメモリ　：インテリジェント機能ユニット (A/D，D/A 変換ユニットなど，入出力以外の機能を持つユニット) 内部の記憶領域で，C 言語コントローラユニットと受け渡しするデータ (設定値，モニタ値など) が格納されます。

操作手順

1. [オンライン] → [デバイスモニタ] を選択

「デバイスモニタ」画面が表示されます。

2. 「デバイスモニタ」画面の確認

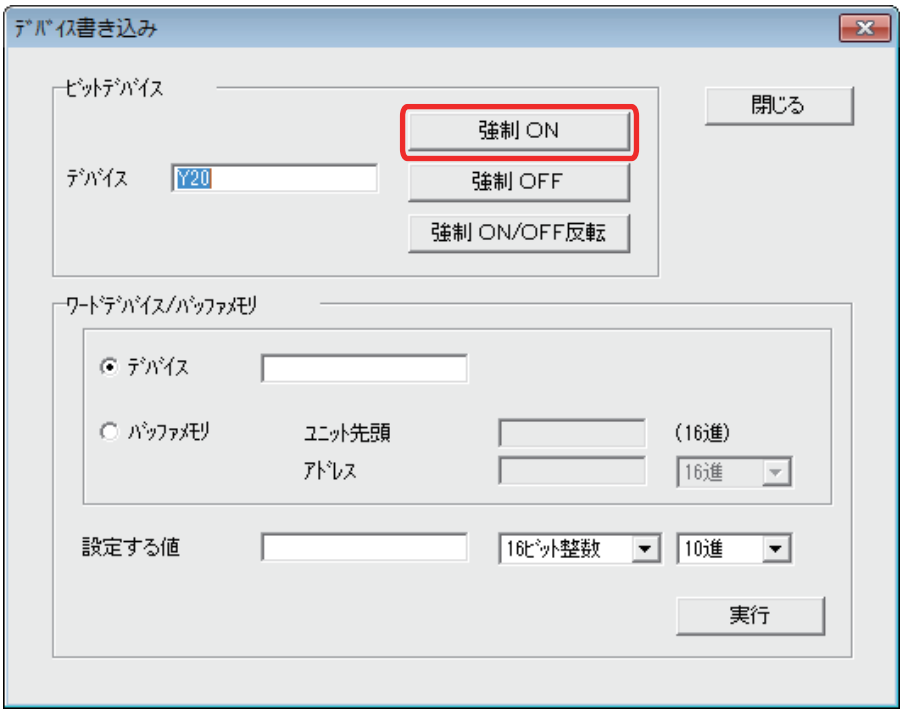


No.	名称	内容
①	デバイス	デバイス名を入力します。
②	モニタ形式	モニタ形式を指定します。 ビット & ワード：モニタ画面をビットおよびワード表示に変更します。 ビット多点：モニタ画面をビット表示のみに変更します。 ワード多点：モニタ画面をワード表示のみに変更します。
③	表示	モニタ形式が，" ビット & ワード " または " ワード多点 " のときに，表示するデバイスの表示形式を指定します。
④	数値	表示が "16 ビット整数 " または "32 ビット整数 " のときの基数 (10 進 /16 進) を指定します。
⑤	ビット並び順	モニタ中のビットデバイスの並び順を指定します。 F-0：左から F，E，…，1，0 の順に並びます。 0-F：左から 0，1，…，E，F の順に並びます。

2) 強制出力による動作テストの実施方法

出力 (Y) の強制出力を行うことで，ユニットの動作テストを実施できます。ここでは，出力 (Y) の強制出力の手順を示します。

1. 「デバイス書き込み」画面で 強制 ON ボタンをクリック



2. 出力（Y）の強制出力が実行されます。

Point

バッファメモリへの強制書込みによる動作テストも，同様の手順で行うことができます。

MEMO

MEMO

ご採用に際してのご注意

このガイドは、Qシリーズシーケンサの代表的な特長機能を説明した資料です。使用上の制約事項、ユニットの組合わせによる制約事項などがすべて記載されているわけではありません。ご使用にあたりましては、必ず製品のユーザーズマニュアルをお読みいただきますようお願い申し上げます。
当社の責に帰すことができない事由から生じた損害、当社製品の故障に起因するお客様での機会損失、逸失利益、当社の予見の有無を問わず特別の事情から生じた損害、二次損害、事故補償、当社製品以外への損傷およびその他の業務に対する保証については、当社は責任を負いかねます。

⚠安全にお使いいただくために

- このガイドに記載された製品を正しくお使いいただくために、ご使用の前に必ず「マニュアル」をお読み下さい。
- この製品は一般工業等を対象とした汎用品として製作されたもので、人命にかかわるような状況下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。
- この製品を原子力用、電力用、航空宇宙用、医療用、乗用移動体用の機器あるいはシステムなど特殊用途への適用をご検討の際には、当社の営業担当窓口までご照会ください。
- この製品は厳重な品質管理体制の下に製造しておりますが、この製品の故障により重大な事故または損失の発生が予測される設備への適用に際しては、バックアップやフェールセーフ機能をシステム的に設置してください。

三菱電機 汎用 シーケンサ
C言語コントローラユニット クイックスタートガイド

三菱電機株式会社 〒100-8310 東京都千代田区丸の内2-7-3 (東京ビル)

お問い合わせは下記へどうぞ

本社機器営業部	〒110-0016	東京都台東区台東1-30-7 (秋葉原アイマークビル)	(03) 5812-1450
北海道支社	〒060-8693	札幌市中央区北二条西4-1 (北海道ビル)	(011) 212-3794
東北支社	〒980-0013	仙台市青葉区花京院1-1-20 (花京院スクエア)	(022) 216-4546
関越支社	〒330-6034	さいたま市中央区新都心11-2 (明治安田生命さいたま新都心ビル)	(048) 600-5835
新潟支店	〒950-8504	新潟市中央区東大通1-4-1 (マルタケビル)	(025) 241-7227
神奈川支社	〒220-8118	横浜市西区みなとみらい2-2-1 (横浜ランドマークタワー)	(045) 224-2624
北陸支社	〒920-0031	金沢市広岡3-1-1 (金沢パークビル)	(076) 233-5502
中部支社	〒450-6423	名古屋市中村区名駅3-28-12 (大名古屋ビルヂング)	(052) 565-3314
豊田支店	〒471-0034	豊田市小坂本町1-5-10 (矢作豊田ビル)	(0565) 34-4112
関西支社	〒530-8206	大阪市北区大深町4-20 (グランフロント大阪タワーA)	(06) 6486-4122
中国支社	〒730-8657	広島市中区中町7-32 (ニッセイ広島ビル)	(082) 248-5348
四国支社	〒760-8654	高松市寿町1-1-8 (日本生命高松駅前ビル)	(087) 825-0055
九州支社	〒810-8686	福岡市中央区天神2-12-1 (天神ビル)	(092) 721-2247

三菱電機 FA 検索

www.MitsubishiElectric.co.jp/fa

メンバー登録無料!

インターネットによる情報サービス「三菱電機FAサイト」

三菱電機FAサイトでは、製品や事例などの技術情報に加え、トレーニングスクール情報や各種お問い合わせ窓口をご提供しています。また、メンバー登録いただくとマニュアルやCADデータ等のダウンロード、eラーニングなどの各種サービスをご利用いただけます。

三菱電機FA機器電話

●電話技術相談窓口　受付時間※1　月曜～金曜 9:00～19:00、土曜・日曜・祝日 9:00～17:00

対象機種			電話番号	自動窓口案内選択番号※7
自動窓口案内			052-712-2444	-
エッジコンピューティング製品		産業用PC MELIPC	052-712-2370※2	8
		Edgecross対応ソフトウェア(NC Machine Tool OptimizerなどのNC関連製品を除く)		
シーケンサ	MELSEC iQ-R/Q/Lシーケンサ (CPU内蔵Ethernet機能などネットワークを除く)		052-711-5111	2→2
	MELSEC iQ-F/FXシーケンサ全般		052-725-2271※3	2→1
	ネットワークユニット(CC-Linkファミリー/MELSECNET/Ethernet/シリアル通信)		052-712-2578	2→3
	MELSOFTシーケンサエンジニアリングソフトウェア	MELSOFT GXシリーズ (MELSEC iQ-R/Q/L/QnAS/AnS)	052-711-0037	2→2
	MELSOFT統合エンジニアリング環境	MELSOFT Navigator/ MELSOFT Update Manager	052-799-3591※2	2→6
	iQ Sensor Solution			
	MELSOFT通信支援ソフトウェアツール	MELSOFT MXシリーズ	052-712-2370※2	2→4
	MELSECパソコンボード	Q80BDシリーズなど		
	C言語コントローラ/C言語インテリジェント機能ユニット			
	MESインタフェースユニット/高速データロガーユニット/高速データコミュニケーションユニット/OPC UAサーバユニット		052-799-3592※2	2→5
	システムレコーダ			
	MELSEC計装/iQ-R/Q二重化	プロセスCPU/二重化機能 SIL2プロセスCPU (MELSEC iQ-Rシリーズ)	052-712-2830※2※3	2→7
		プロセスCPU/二重化CPU (MELSEC-Qシリーズ)		
		MELSOFT PXシリーズ		
	MELSEC Safety	安全シーケンサ (MELSEC iQ-R/QSシリーズ)	052-712-3079※2※3	2→8
		安全コントローラ (MELSEC-WSシリーズ)		
	電力計測ユニット/絶縁監視ユニット	QEシリーズ/REシリーズ	052-719-4557※2※3	2→9
FAセンサ MELSENSOR		レーザ変位センサ	052-799-9495※2	6
		ビジョンセンサ		
		コードリーダ		

対象機種		電話番号	自動窓口案内選択番号※7
表示器 GOT	GOT2000/1000シリーズ	052-712-2417	4→1
	MELSOFT GTシリーズ		4→2
SCADA GENESIS64™		052-712-2962※2※6	-
サーボ/位置決めユニット/モーションユニット/シンプルモーションユニット/モーションコントローラ/センシングユニット/組込み型サーボシステムコントローラ	MELSERVOシリーズ	052-712-6607	1→2
	位置決めユニット (MELSEC iQ-R/Q/Lシリーズ)		1→2
	モーションユニット (MELSEC iQ-R/iQ-Fシリーズ)		1→1
	モーションソフトウェア		1→1
	シンプルモーションユニット (MELSEC iQ-R/iQ-F/Q/Lシリーズ)		1→2
	モーションCPU (MELSEC iQ-R/Qシリーズ)		1→1
	センシングユニット (MR-MTシリーズ)		1→2
	シンプルモーションボード/ポジションボード		1→2
	MELSOFT MTシリーズ/MRシリーズ/EMシリーズ		1→2
センサレスサーボ	FR-E700EX/MM-GKR	052-722-2182	3
インバータ	FREQROLシリーズ	052-722-2182	
三相モータ	三相モータ225フレーム以下	0536-25-0900※2※4	-
産業用ロボット	MELFAシリーズ	052-721-0100	5
電磁クラッチ・ブレーキ/テンションコントローラ		052-712-5430※5	-
データ収集アナライザ	MELQIC IU1/IU2シリーズ	052-712-5440※5	-
低圧開閉器	MS-Tシリーズ/MS-Nシリーズ	052-719-4170	7→2
	US-Nシリーズ		
低圧遮断器	ノーヒューズ遮断器/漏電遮断器/MDUブレーカ/気中遮断器 (ACB) など	052-719-4559	7→1
電力管理用計器	電力量計/計器用変成器/指示電気計器/管理用計器/タイムスイッチ	052-719-4556	7→3
省エネ支援機器	EcoServer/E-Energy/検針システム/エネルギー計測ユニット/ B/NETなど	052-719-4557※2※3	7→4
小容量UPS (5kVA以下)	FW-Sシリーズ/FW-Vシリーズ/FW-Aシリーズ/FW-Fシリーズ	052-799-9489※2※6	7→5

お問い合わせの際には、今一度電話番号をお確かめの上、お掛け間違いのないようお願いいたします。
※1：春季・夏季・年末年始の休日を除く　※2：土曜・日曜・祝日を除く　※3：金曜は17:00まで　※4：月曜～木曜の9:00～17:00と金曜の9:00～16:30
※5：受付時間9:00～17:00 (土曜・日曜・祝日・当社休日を除く)　※6：月曜～金曜の9:00～17:00
※7：選択番号の入力は、自動窓口案内冒頭のお客様相談内容に関する代理店、商社への提供可否確認の回答後にお願いいたします。

L(名)08206-C　1712(KWIX)

2017年12月作成