注意: この日本語版文書は参考資料としてご利用ください。
最新情報は必ずオリジナルの英語版をご参照願います。

MPLAB® X IDEユーザガイドのCI/CDウィザード

開発ツールユーザへの注意

Important:
どのような文書でも内容は時間が経つにつれ古くなります。本書も例外ではありません。Microchip社のツールとマニュアルは、お客様のニーズを満たすために常に改良を重ねており、実際のダイアログやツールの内容が本書の説明とは異なる場合があります。最新PDF文書はMicrochip社のウェブサイト(www.microchip.com/)をご覧ください。

文書は各ページ下部に表記している「DS」番号で識別します。DS番号のフォーマットはDS<文書番号><リビジョン>です。<文書番号>は8桁の番号、<リビジョン>はアルファベットの大文字です。

最新情報はツールのヘルプ(onlinedocs.microchip.com/)をご覧ください。

目次

1. CI/CDウィザード

Note: 本書の内容は『MPLAB X IDEユーザガイド』(DS-50002027)にも含まれています。

MPLAB X IDE CI/CDウィザードはMPLAB X IDEプロジェクトに合わせた自動ビルドおよびテスト パイプラインの設定を支援します。

CI/CDの要件

CI/CDウィザードはMPLAB X IDEの機能なので、『MPLAB X IDEのリリースノート』に記載されている要件が適用されます。さらに以下の要件があります。

- CI/CDウィザードによるスクリプト生成を有効にするには、MPLAB XC **WorkstationまたはNetwork Server ライセンス**が必要です。
- 生成されたDockerコンテナにおいてPRO最適化レベルでプロジェクトをビルドするには、MPLAB XC **Network Severライセンス**が必要です。
- **バーチャルマシン ネットワーク ライセンス**を使っている場合、このライセンスとMicrochip **Network Server ライセンス**を必ず同じフォルダに配置してください。
- Jenkins Pipelineを使っている場合、以下の要件があります。
  - MISRA Check(静的コード解析)およびMPLAB Code Coverage(ハードウェア テスト)ツールは、生成されたDockerコンテナでのツール実行にMPLAB X IDE v6.00以降とMPLAB解析ツールスイート **Network Serverライセンス**が必要です。
  - ハードウェア テストにはネットワーク機能を備えたツールが必要です。現在、こうしたツールはMPLAB ICE 4 インサーキット エミュレータのみです。このツールはMPLAB X IDE v6.00以降でサポートされています。
- JenkinsとDockerを使うには、これらのアプリケーションをインストールする必要があります。Jenkinsを使う際の追加要件は、1.3.1.「要件」を参照してください。
  Note: MPLAB X IDE CI/CDウィザード出力の実行には、サードパーティ アプリケーションであるJenkinsとDockerを使います。これらのツールの問題に対するサポートは、該当するウェブページを参照してください。

CI/CDの標準的な使い方

Jenkins-DockerまたはDockerで使う標準的なオペレーティング システムはLinux OSで、CI/CDウィザードの出力には特にDebian Linuxを使います。それ以外のオペレーティング システムは、JenkinsとDockerの文書を参照してください。以下に例を示します。

- JENKINS-60473: Docker pipeline: 'absolute path' error when running linux container under windows host

1.1 CI/CDについて

CI/CD(またはCICD)はソフトウェアの継続的インテグレーションと継続的デリバリまたはデプロイメントのためのプロセスです。従来のようにソフトウェア モジュールを完全に開発し、それらを統合して最後に製品を供給するのではなく、ソフトウェアを継続的に統合して少しずつ供給または展開するため、どの時点でも問題を最低限に抑えて製品を生産できます。

継続的インテグレーションは統合とテストを開発サイクルの早い段階に移行します。これにより、プロジェクトのリスクを軽減して製品を成功に導く可能性を高める事ができます。

CI/CDパイプラインをサポートするために、コンテナを使い、チームが共有できる一貫した環境を使ってコードを構築してテストできます。コンテナを作成および実行するための命令ファイル(Dockerfile等)とコンテナの中身を実行するための命令ファイル(Jenkinsfile等)を開発する事は簡単ではありません。ここでCI/CDウィザードが開発の助けになります。

1.1.1 ビルドとテストの自動化に継続的インテグレーションを使う理由

CI(**継続的インテグレーション**)は、git等のソースコード リポジトリに開発者がコードの変更を頻繁にコミットする作業の事です。コミットのたびに自動ビルドおよびテストサイクルがトリガされます。ビルドが中断するかテストに失敗すると、通知が送信されます。CIを行う事で最初から品質を製品に組み込み、コードの変更をチームと調整し、統合の遅れと検証のフィードバックの遅れに伴うリスクを軽減できます。

開発者が行うCIは**ビルド オートメーション** システムと連携します。このシステムはコンパイル、テスト、パッケージ化、デプロイ等のジョブ/プロセスを自動的に開始します。これにより、問題を早期に発見する事でコードの品質を高く維持できます。ビルド オートメーション ツールには様々なものがあります。Jenkinsは無償のオープ

ンソース ソフトウェアです。広く普及しているツールを、ここでは例として使います。別のシステムを使っている場合、CI/CDウィザードで生成されたJenkinsfileを参考にしてください。

1.1.2 Dockerとコンテナ

ビルド オートメーションの鍵となる考え方は**コンテナ化**です。ビルドジョブを実行する場合、ビルドの実行時期と場所に関係なく再現性と一貫性のある結果を生成する必要があります。**Dockerコンテナ**はこの作業を支援できます。Dockerコンテナを使って再現性、柔軟性、拡張性を実現します。ジョブを何度実行しても、環境は同じです。DockerはOSレベルの仮想化を実現し、アプリケーションとインフラストラクチャの分離を可能にします。

Dockerのインストール方法とDockerfileの使い方の詳細はwww.docker.comを参照してください。

1.2 CI/CDウィザードの主な特長

1. 専用の**Dockerfile**を作成します。これらのファイルはDockerイメージを生成するためのレシピです。Dockerイメージを使ってDockerコンテナを作成します。Dockerコンテナにはビルドに必要な全てのMPLAB X IDEプロジェクト設定、つまりMPLAB X IDEのバージョン、MPLAB XCツールチェーン、XCライセンス処理、DFP(デバイスファイル パック）、ツールパックが含まれています。
2. **Jenkinsfile**を作成します。Dockerコンテナを使ってJenkins BuildサーバでMPLAB X IDEプロジェクトをビルドおよびテストする方法を記述するJenkins Pipelineを設定するためにこれらのファイルを使います。
3. ビルドサーバ設定でMPLAB XC PROコンパイラ ライセンスを処理します。
4. **Jenkinsサーバ**を使ってDockerコンテナを作成して起動します。これにより、ビルド パイプラインを迅速にテストする事も、ローカルでホストされたJenkinsサーバのベースとして使う事もできます。
5. 自動ビルド パイプラインを設定して以下のような手順を含めます。
   - MISRAチェック
   - シミュレータによるテストの実行
   - ハードウェアに対するテストの実行
   - コードカバレッジ データの収集
   - Doxygenによる文書の生成

1.3 Jenkinsの使い方

Jenkinsは開発者がソフトウェアのビルド、テスト、デプロイを自動化するためのオープンソースのオートメーション サーバです。詳細はJenkins公式ウェブサイトを参照してください。

以下では、JenkinsサーバでMPLAB X IDEプロジェクトをビルドするために必要な手順について説明します。

1.3.1 要件

1.3.1.1 Jenkinsサーバ

- Jenkinsサーバを利用できる必要があります。Jenkinsへのアクセス権がない場合は、『Installing Jenkins』ガイドに従って自分で設定するか、`cicdw jenkins-server`コマンドを使ってローカルテスト サーバを作成します(1.4.「Jenkinsサーバの使い方」参照)。
- Jenkinsサーバで新しいビルドジョブを設定するためのユーザ アカウントが必要です。
- JenkinsサーバにはDockerをサポートするビルド ノードが必要です。ビルドノードがない場合、「Jenkinsと Docker」を参照してください。
- Jenkinsサーバには以下のプラグインを設定する必要があります。
  - Pipeline Aggregator View
  - Pipeline Utility Steps
  - Docker Pipeline
  - Workspace Cleanup
  - Warnings Next Generation

  Note: オプションのビルドステップには、MISRAやシミュレータ等の追加のプラグインが必要な場合があります。

1.3.1.2 ソース制御システム

- プロジェクトのソースコードはソース制御システムにホストされている必要があります(例: Subversion、CVS、Git)。
- Jenkinsサーバはソース制御システムをサポートし、プロジェクト リポジトリへのアクセス権を持っている必要があります。

1.3.1.3 ライセンスサーバ

- MPLAB XCライセンスサーバが設定され、指定したアドレスとポートで利用できる必要があります。詳細は『MPLAB XC License Server Manual』を参照してください。
- MPLAB解析ツールスイート (MISRA CheckとMPLAB Code Coverage)等の他のライセンス オプションをサポートするために他のライセンス サーバが必要な場合があります。

1.3.1.4 テストツール

コンテナ内のテストにソフトウェア (シミュレータ)またはハードウェア(MPLAB ICE 4インサーキット エミュレータ)のいずれかのツールを使うには、設定を考慮する必要があります。

- Unityを使ってユニットテストを行い、シリアル通信を使って`printf`や`putchar`で結果を出力します。ハードウェア ツールの場合、ターゲットに追加コードが必要です。Unityの詳細は1.5.「Unityによるユニットテストの記述および実行方法」を参照してください。インサーキット エミュレータMPLAB ICE 4の仮想COMポートは、ユーザガイドの「USB CDC Virtual COM Port」を参照してください。
- 基本的なデバッグコマンドをサポートするためにMDB (Microchip Debugger)コマンドライン インターフェイスを使います。ハードウェア ツールの場合、MDBを使ってターゲットをプログラミングします。MDBの詳細は、『Microchip Debugger (MDB) User's Guide』(DS52102)を参照してください。

1.3.2 Step 1 - MPLAB X IDEからのJenkinsfileの生成

1. MPLAB X IDEでプロジェクトを開き、[Tools] > [CI/CD Wizard]に移動します。
2. 出力タイプとして**[Jenkins Pipeline files]**を選択します。
3. ウィザードのステップを実行し、パイプラインで実行するツールを選択します。
4. 生成後、プロジェクトのベースフォルダにJenkinsfileとDockerfileが追加されている事を確認します。選択したオプションによっては他のファイルも追加される場合があります。

1.3.2.1 [CI/CD Wizard]ダイアログ1

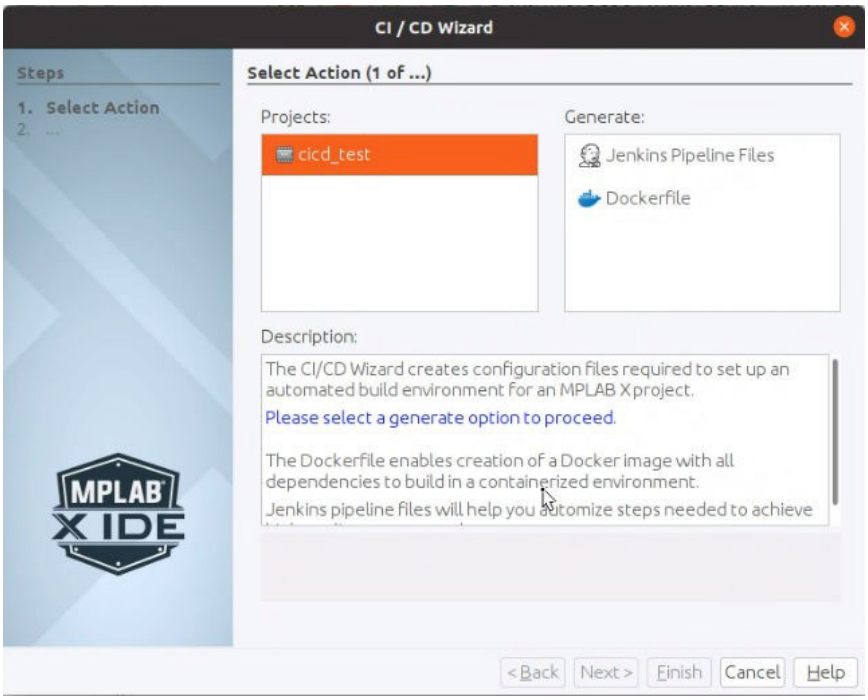
ウィザードの最初のダイアログにはこの注記が表示され、MPLAB XC Cコンパイラのライセンスを持っていない場合は**[Cancel]**ボタンのみが有効になります。

CICDパイプラインで使用するファイルを生成するには、ライセンス取得済みコンパイラが必要です。

サポートされているライセンスは1.「CI/CDウィザード」を参照してください。MySoftwareでコンパイラのライセンスを購入できます。**[Configure Licenses]**ボタンをクリックして[Change Licensing Type]ダイアログでライセンスを設定します。

コンパイラのライセンスがある場合、生成するファイルを選択できます。JenkinsfileとDockerfileを作成する場合、[Jenkins]を選択します。Dockerfileのみを作成する場合、[Docker]を選択します。

図1-1. 生成するものを選択していない状態



この例では[Jenkins Pipeline Files]を選択します。[Description]の下に表示されるこの選択項目に関するテキストを参照してください。

[Next]をクリックしてダイアログを移動します。

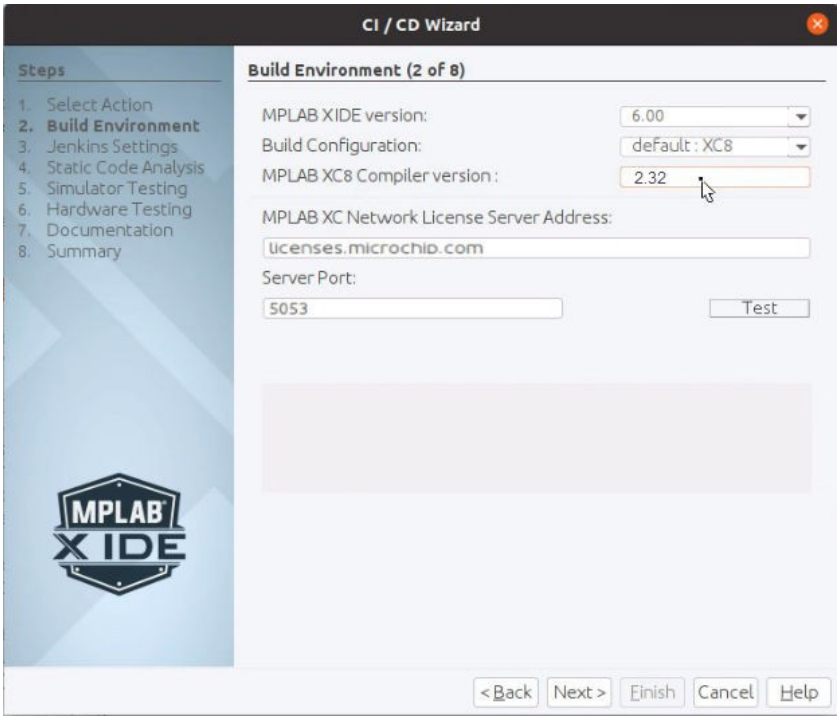
図1-2. 生成するものを選択した状態

1.3.2.2 [CI/CD Wizard]ダイアログ2

コンテナで使用するためのビルド環境を設定します。ネットワーク サーバのライセンスがある場合、ライセンスへのアクセス方法に関する情報をビルド用に提供できます。接続をテストする事もできます(下図参照)。

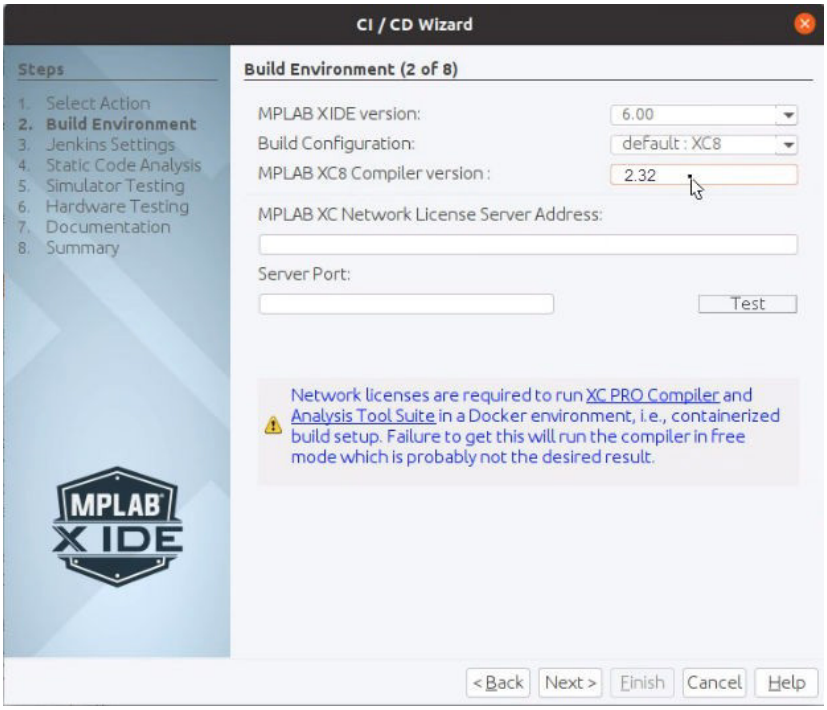
Note: MPLAB X IDEの[Tools] > [Licenses]を使ってネットワーク サーバのライセンスを設定済みの場合、その情報がこのダイアログに自動的に入力されます。

図1-3. ネットワーク サーバライセンス



ライセンスを設定していない場合、テキストボックスを空白のままにします(ただし、注意が表示されます)。

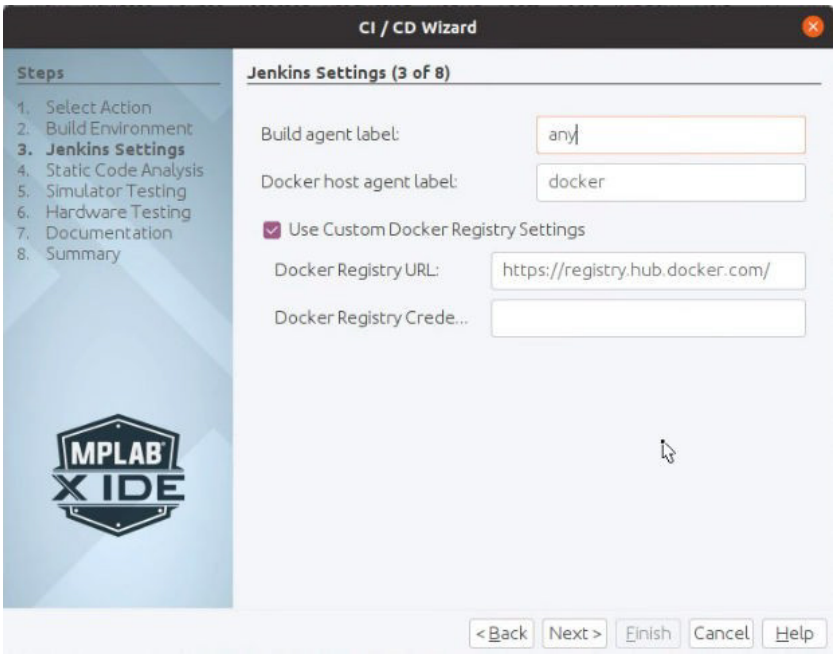
図1-4. ワークステーション ライセンス



オプション	説明
MPLAB IDE version	サポートされているバージョンをドロップダウン リストから選択します。 既定値では最新バージョンです。
Build Configuration	このビルド用のプロジェクト設定を選択します。
MPLAB XC Compiler version	このビルド用のコンパイラ バージョンを選択します。
MPLAB XC Network License Server Address	サーバのアドレスを入力します。例: licenses.microchip.com
Server Port	サーバのポート番号を入力します。例: 5053
情報注記(条件による)	ワークステーション ライセンスがある場合、Network Serverライセンスの購入に関する情報が表示されます。

1.3.2.3 [CI/CD Wizard]ダイアログ3

Dockerがイメージをダウンロードする場所を決定します。既定値のDocker情報が入力されます。カスタムDockerレジストリを使う場合、以下の情報を入力します。



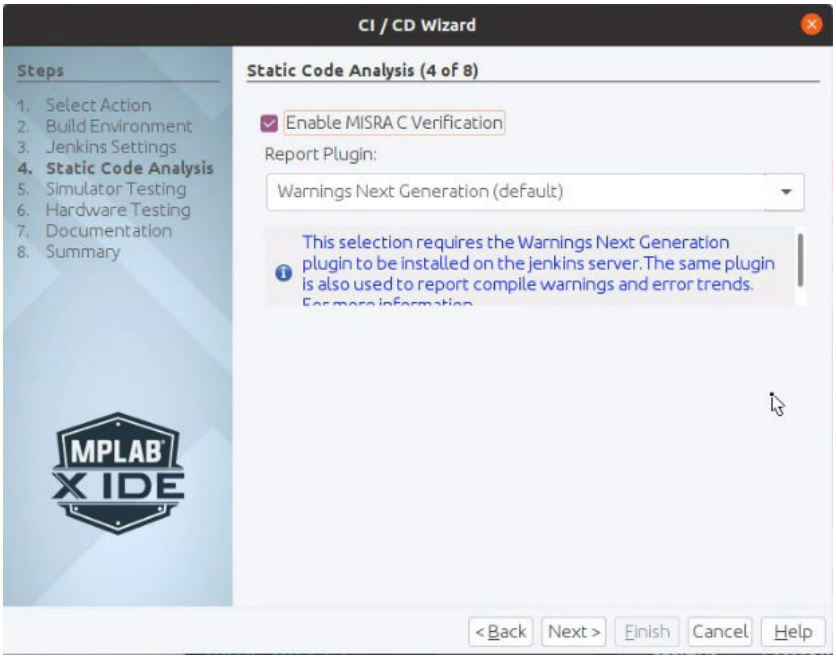
オプション	説明
Build agent label	既定値は[any]に設定されていますが、特定のラベルを追加できます。Jenkinsサーバの[Configure System]設定で、Jenkinsエージェントがそれに従ってラベル付けされている事を確認する必要があります。
Docker host agent label	Dockerホスト エージェント ラベルは、Dockerコンテナのビルドと実行をサポートできるJenkinsエージェントに付けられるラベルです。標準的な命名規則があり、それに従わなければならない場合があるので注意する必要があります。これらの命名規則はJenkinsの文書を参照してください。
Use Custom Docker Registry Settings	ダウンロードする基本イメージに自分のDockerレジストリを使います。詳細は「How to use your own Registry」を参照してください。 基本イメージがDocker Hubからダウンロードされます。必要に応じて、カスタマイズした基本イメージのパスを指定できます。
Docker Registry URL	自分のレジストリの場所です。
Docker Registry Credentials ID	プッシュおよびプルする認証情報です。詳細は「Using Jenkins - Credentials」を参照してください。

1.3.2.4 [CI/CD Wizard]ダイアログ4

MISRA C検証を有効にする場合にチェックを入れます。MISRAの詳細はwww.misra.org.uk/を参照してください。
この機能を使う上での要件を以下に示します。

- どちらのレポートを選択する場合でもJenkinsプラグインが必要です。情報注記にリンクが表示されます。
- MISRA C検証はMPLAB X IDE v6.00以降でのみ利用可能です。バージョンがこの機能をサポートしていない場合、情報注記が表示されます。
- 生成されたDockerコンテナでツールを実行するにはMPLAB解析ツールスイート Network Serverライセンスが必要です

図1-5. Warnings Next Generation



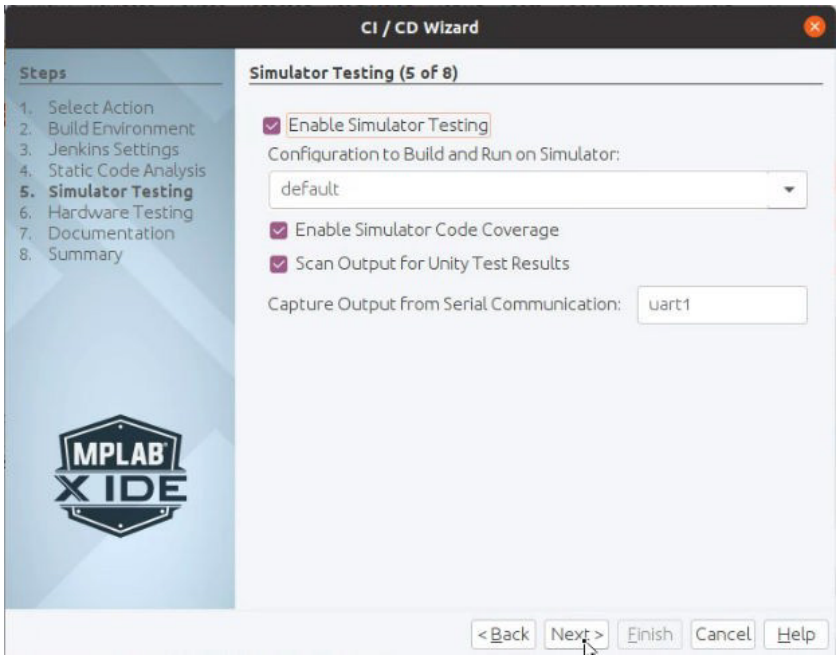
オプション	説明
Enable MISRA-C Verification	MISRA C検証を有効にする場合にチェックを入れます。
Report Plugin	以下のいずれかを選択します。 • Warnings Next Generation • Cppcheck

1.3.2.5 [CI/CD Wizard]ダイアログ5

テストイメージとして使うシミュレータを有効にする場合にチェックを入れ、その他のオプションを指定します。これはユニットテストを実行する場合に使用できます。

MDB (Microchip Debugger)スクリプトmdb-simulator-script.txtがプロジェクトに追加され、シミュレータのビルドステップでこのスクリプトが実行されます。

この既定値のスクリプトはシミュレータを10秒間実行し、設定に基づいて出力とカバレッジをキャプチャします。これを変更するにはスクリプトを手動で変更するか、別のオプションを使って再生成します。テストの要件に応じて入力ファイルをカスタマイズします。『Microchip Debugger (MDB)ユーザガイド』を参照してください。



オプション	説明
Enable Simulator	シミュレータをデバッグツールとして有効にします。
Configuration to Build and Run on Simulator	シミュレータを使用するプロジェクト設定([default]または[one dedicated to simulator use])を選択します。
Enable Simulator Code Coverage	シミュレータに実装されているコードカバレッジ機能を有効にします。 **Note:** カバレッジレポートを有効にする場合、JenkinsサーバでCode Coverage APIプラグインを利用可能にする必要があります。
Scan Output for Unity Test Results	設定がUnity Test Runnerをビルドし、ビルドジョブが結果の出力に基づいてレポートを作成する場合に有効にします。Unityテストの記述方法は1.5.「Unityによるユニットテストの記述および実行方法」を参照してください。
Capture Output from Serial Communication	このオプションは、[Scan output for Unity test results]オプションと一緒に使用する必要があります。シミュレータは、どのシリアル通信モジュール インスタンスから出力をキャプチャするのかを知る必要があります。既定値は[uart1]ですが、これはどのデバイスを使用するか、テスト設定をどのように行ったかによって決まります。

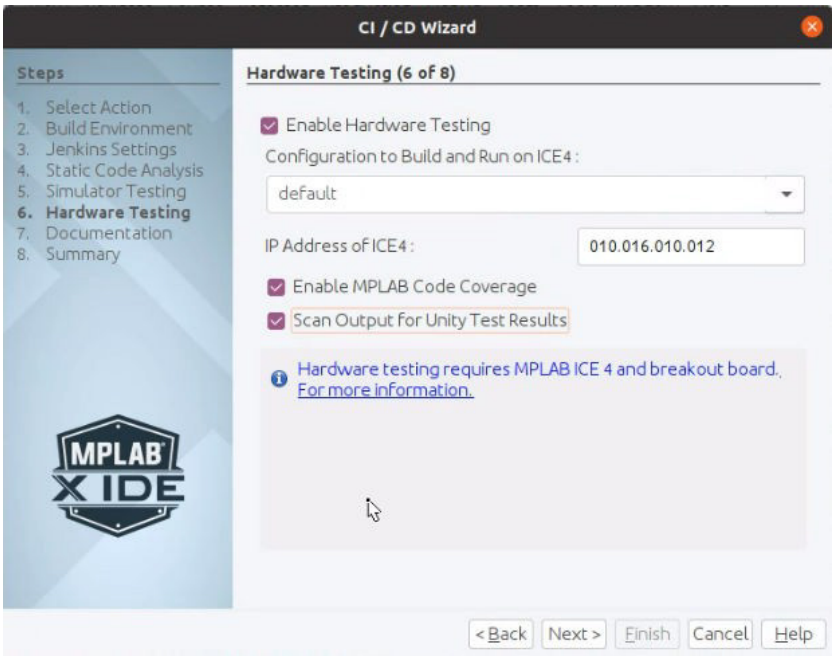
1.3.2.6 [CI/CD Wizard]ダイアログ6

テスト イメージとして使うハードウェア ツールを有効にする場合にチェックを入れ、その他のオプションを指定します。これはユニットテストを実行する場合に使用できます。

Note: 現在、ネットワーク機能を備えているハードウェア ツールはMPLAB ICE 4インサーキット エミュレータのみです。MPLAB ICE 4はMPLAB X IDE v6.00以降でのみサポートされます。

MDB (Microchip Debugger)スクリプトmdb-hardware-script.txtがプロジェクトに追加され、ハードウェアのビルドステップでこのスクリプトが実行されます。

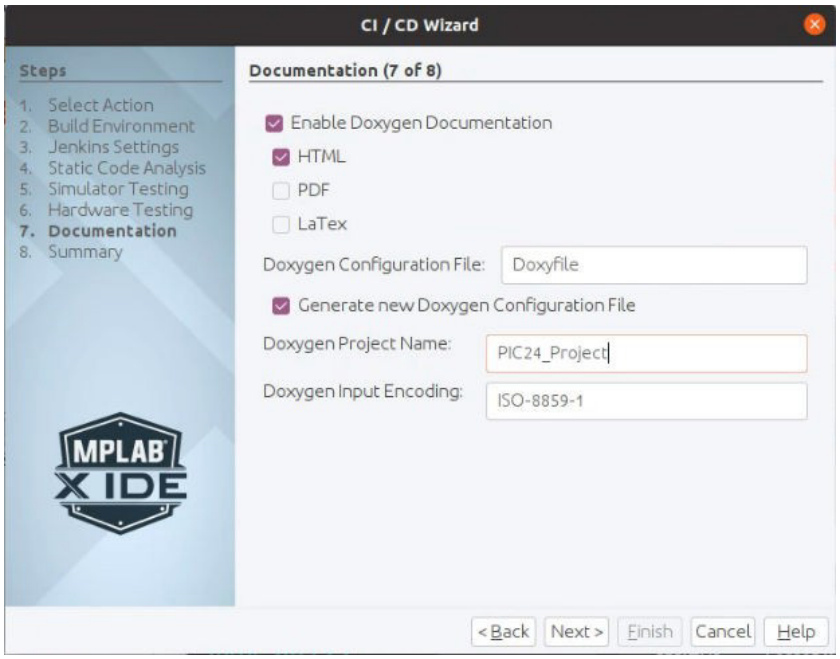
この既定値のスクリプトは10秒間実行され、設定に基づいて出力とカバレッジをキャプチャします。これを変更するにはスクリプトを手動で変更するか、別のオプションを使って再生成します。テストの要件に応じて入力ファイルをカスタマイズします。『Microchip Debugger (MDB)ユーザガイド』を参照してください。



オプション	説明
Enable Hardware Testing	エミュレータをテストツールとして有効にします。
Configuration to Build and Run on ICE4	エミュレータを使用するプロジェクト設定([default]または[one dedicated to hardware use])を選択します。
IP Address of ICE4	使用するエミュレータのIPアドレスを入力します。エミュレータのIPアドレスが分からない場合、[Tools] > [Manage Network Tools]を開いてネットワーク ツールをスキャンできます。これにより、エミュレータのリストがIPアドレスと共に表示されます。
Enable MPLAB Code Coverage	MPLABコードカバレッジ機能を有効にします。生成されたDockerコンテナでツールを実行するにはMPLAB解析ツールスイート Network Serverライセンスが必要です **Note:** カバレッジレポートを有効にする場合、JenkinsサーバでCode Coverage APIプラグインを利用可能にする必要があります。
Scan Output for Unity Test Results	設定がUnity Test Runnerをビルドし、ビルドジョブが結果の出力に基づいてレポートを作成する場合に有効にします。Unityテストの記述方法は 1.5.「Unityによるユニットテストの記述および実行方法」 を参照してください。

1.3.2.7 [CI/CD Wizard]ダイアログ7

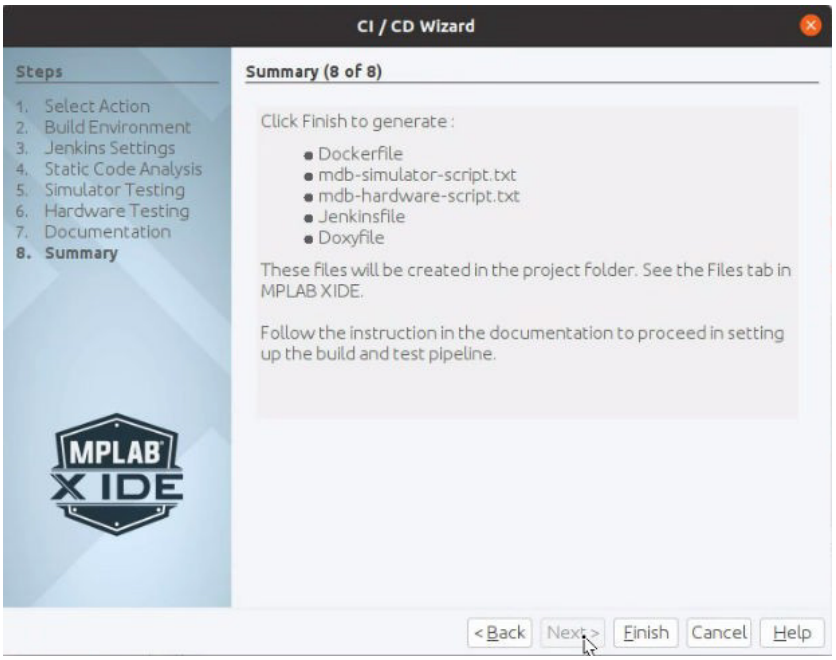
文書ツールとしてDoxygenを有効にする場合にチェックを入れ、出力を指定します。Doxygenの詳細はDoxygenに関するページ (GitHub)を参照してください。



オプション	説明
Enable Doxygen Documentation	文書ツールを有効にします。
Documentation Format	1つまたは複数の出力フォーマットを選択します。
Doxygen Configuration File	コンフィグレーション ファイルを指定します。
Generate new Doxygen Configuration File	ファイルを新規作成するか既存のファイルを上書きする場合にチェックを入れます。既存のファイルを保持する場合はチェックを外します。
Doxygen Project Name	プロジェクト名を指定します。
Doxygen Input Encoding	文書のエンコーディングを指定します。

1.3.2.8 [CI/CD Wizard]ダイアログ8

予測される出力を[Summary]画面で確認します。変更するには**[Back]**をクリックします。



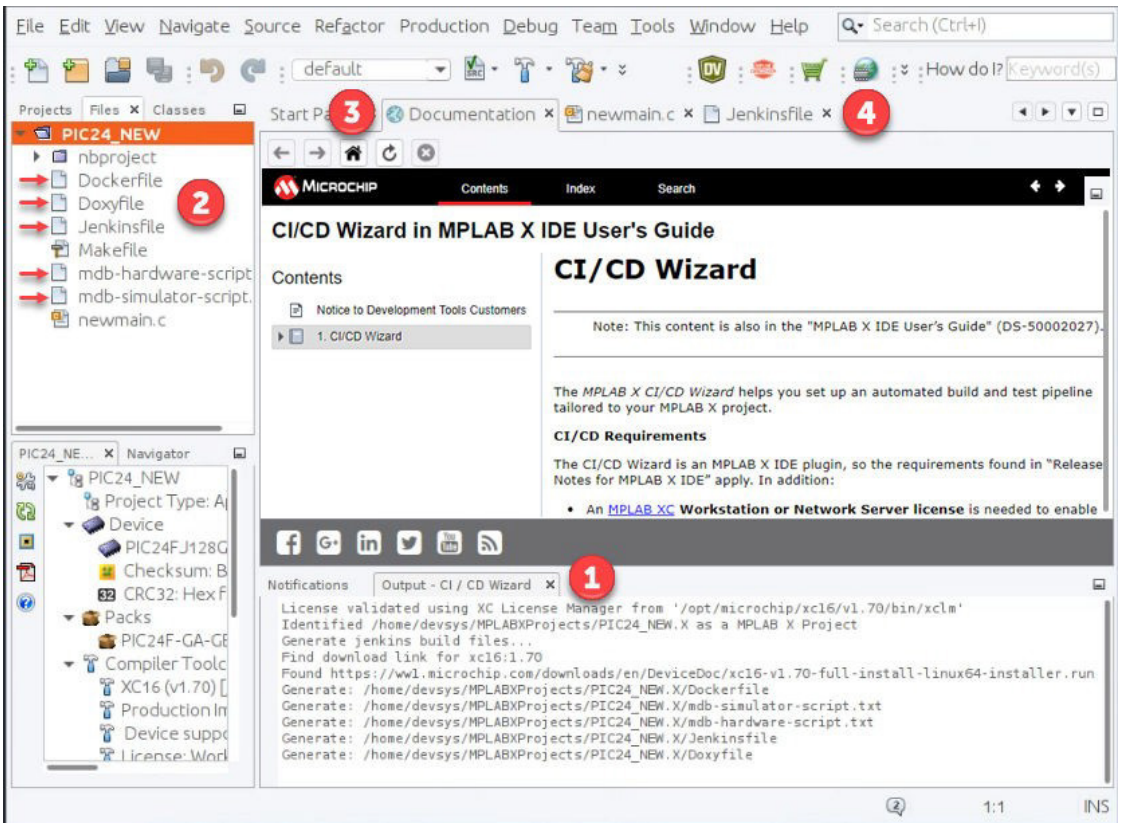
設定を完了するには**[Finish]**をクリックします。設定が完了すると、CI/CDウィザードが設定情報を基にファイルを生成します。

1.3.2.9 CI/CDウィザードの出力

MPLAB X IDEに以下が出力されます。

1. 生成したファイル([Output]ウィンドウ)
2. プロジェクト フォルダに追加された結果ファイル
3. オンラインヘルプ(専用のウィンドウ)
4. Jenkinsfile(専用のウィンドウ)

図1-6. MPLAB X IDEでのファイル生成



生成されたファイルの詳細:

Jenkinsfile

Jenkins Pipelineの定義を含むテキストファイル。詳細は「Using a Jenkinsfile」を参照してください。

Dockerfile

コンテナのビルドに使うカスタマイズされたテキスト ファイル。詳細はDockerfileリファレンスを参照してください。

mdb-simulator-script.txt

このファイルは、アシスタントのステップ5でシミュレータを選択した場合にのみ生成されます。MPLAB X IDEのGUIがないバージョンであるMDB (Microchip Debugger)でシミュレータの使用方法を設定するためのファイルです。このファイルはテストのニーズに合わせてカスタマイズできます。詳細は『Microchip Debugger (MDB)ユーザガイド』を参照してください。

mdb-hardware-script.txt

このファイルは、アシスタントのステップ6でMPLAB ICE 4インサーキット エミュレータを選択した場合にのみ作成されます。MPLAB X IDEのGUIがないバージョンであるMDB (Microchip Debugger)でエミュレータの使用方法を設定するためのファイルです。このファイルはテストのニーズに合わせてカスタマイズできます。詳細は『Microchip Debugger (MDB)ユーザガイド』を参照してください。

Doxygen

このファイルは、アシスタントのステップ7で文書を選択した場合にのみ作成されます。

Doxygenは、注釈付きソースコードから文書を生成するための標準ツールです。コンフィグレーション ファイル フォーマットの詳細は「Configuration」を参照してください。

1.3.3 Step 2 - リポジトリへのJenkinsfileの追加

ソース制御システムを使い、生成されたファイルをプロジェクトのリポジトリに追加します。Jenkinsは、作成したコンテナにこのリポジトリにあるプロジェクトを追加します。

1.3.4 Step 3 - Jenkinsサーバでのビルドジョブの設定

本セクションでは、生成されたファイルを使ってJenkinsサーバで新しいビルドジョブを設定する方法について簡単に説明します。詳細は「Using a Jenkinsfile on Jenkins.io」を参照してください。

簡単な手順:

1. Jenkinsサーバウェブ インターフェイスにログインして**[New Item]**を選択する。
2. ビルドジョブ名を入力してプロジェクトタイプとして**[Pipeline]**を選択する。
3. 新しいビルドジョブの設定で、**[Pipeline]**セクションに移動して設定する。
   a. **[Pipeline script from SCM]**を選択し、Jenkinsサーバがプロジェクトをチェックアウトするために必要なSCM (Source Control Management)情報を設定する。これにはリポジトリURL、認証情報、さらにブランチ指定子を含める事が可能。
   b. **[Script Path]**としてJenkinsfileを指定する(これは生成されたJenkinsfileの名前と一致させる必要がある)。
   c. 新しいビルドジョブの設定を保存して新しいビルドジョブを開始する(**[Build Now]**)。

これにより、MPLAB X IDEプロジェクトをビルドして解析する前にDockerビルド環境を最初に設定するビルド パイプラインが動作します。

1.3.5 次の推奨手順

JenkinsfileはDockerイメージの内部で実行されるビルド手順を定義します。Dockerイメージは、生成したDockerfileによって定義され、必要なMPLAB XCツールチェーン、デバイスサポート、MPLAB X IDEバイナリを利用できる安定したLinux環境を提供します。

生成したJenkinsfileを変更するにはJenkinsfile文書の内容に従います。

1.3.5.1 ゲーティング

安定したビルドジョブができたら、ゲーテッド コミットプロセスを実装できます。このプロセスでは、各コミットが拒否されるかメインブランチにマージされる前に自動的にテストできます。こうしたプロセスにより、多くの開発者が同じプロジェクトで作業していてもメインブランチは安定した状態を保つ事ができます。

ゲーテッドビルド プロセスの設定は、使用しているソースコード管理インフラストラクチャに応じて複数の異なる方法で実行できます。ここでは、Atlassian Bitbucket統合の使用方法の例について説明します。

1.4 Jenkinsサーバの使い方

Jenkinsは開発者がソフトウェアのビルド、テスト、デプロイを自動化するためのオープンソースのオートメーション サーバです。詳細はJenkins公式ウェブサイトを参照してください。

CI/CDウィザードは、自動的に設定されたJenkinsサーバ インスタンスをDockerコンテナで実行するソリューションを提供します。Jenkinsサーバは、ローカルフォルダまたはgitから既存のMPLAB X IDEプロジェクトをビルドするように設定できます。このサーバは、追加の設定とカスタマイズの出発点として利用する事も、ウィザードで生成したビルド パイプラインのローカルテスト ベッドとして利用する事もできます。

1.4.1 要件

Docker
Dockerは、OSレベルの仮想化を使ってコンテナと呼ばれるパッケージでソフトウェアを提供するPaaS (Platform as a Service)製品のセットです。コンテナは互いに分離され、独自のソフトウェア、ライブラリ、コンフィグレーション ファイルを備えています。

- docker.comで詳細を確認
- Dockerの使い方

1.4.2 Jenkinsサーバの作成と起動

Jenkinsサーバの作成と起動には、CI/CDウィザードのコマンドライン インターフェイスである`cicdw`を使います。このツール詳細は1.7.「CICDWコマンドライン ツール」を参照してください。

1.4.2.1 「プレーン」Jenkinsサーバの作成

ビルドジョブを事前設定していないプレーンなJenkinsサーバを作成するには以下のコマンドを入力します。

```
cicdw jenkins-server create
```

1.4.2.2 GitベースのビルドジョブによるJenkinsサーバの作成

gitリポジトリに対してビルドするためのビルドジョブが事前設定されたJenkinsサーバを作成するには以下のコマンドを入力します。

```
cicdw jenkins-server create --with-git-job origin
```

これにより、ローカルgitリポジトリに対して「origin」という名前のgitリモートに関連するgitサーバ情報が抽出されます。これを動作させるには、プロジェクトパス(既定値は現在のフォルダ)は、Jenkinsfileを含んでリモートgitリポジトリにチェックインされている有効なMPLAB X IDEフォルダを指し示す必要があります。

Note: プロジェクトに関連するJenkinsfileがまだない場合は、CI/CDウィザードを使ってファイルを生成してください。1.3.「Jenkinsの使い方」を参照してください。

1.4.2.3 ローカルビルド ジョブによるJenkinsサーバの作成

ローカルフォルダに対してビルドするためのビルドジョブが事前設定されたJenkinsサーバを作成するには以下のコマンドを入力します。

```
cicdw jenkins-server create --with-local-job
```

トリガされると、ビルドジョブはコンピュータのローカルフォルダにあるJenkinsfileを探します。サーバ起動時に[path]設定を使ってローカルフォルダのパスを提供します(これは量産用サーバモードではありません)。

Note: プロジェクトに関連するJenkinsfileがまだない場合は、CI/CDウィザードを使ってファイルを生成してください。1.3.「Jenkinsの使い方」を参照してください。

1.4.2.4 Jenkinsサーバの起動

Jenkinsサーバを作成したら、以下のコマンドを入力すると起動できます。

```
cicdw jenkins-server start
```

ローカルビルド ジョブで設定されたサーバは、startコマンドで指定されたプロジェクトのパス(既定値は現在のフォルダ)をローカルフォルダとして使います。

Jenkinsサーバを起動すると、ブラウザからアクセスできます。

既定値の場所はhttp://localhost:8080/です。これは`--server-name`および`--server- port`パラメータを使って変更できます。

既定値のユーザは**mplabcicd/mplabcicd**という認証情報で作成されます。
これは`--admin-user`および`--admin-password`パラメータを使って変更できます。

1.4.3 認証情報

gitリポジトリのような外部リソースとやり取りする場合、ビルドシステムを認証して認可するために各種認証情報が必要になる事がよくあります。Jenkins認証情報は認証情報マネージャによって処理され、新しいサーバを作成する時に提供できます。

```
cicdw jenkins-server create --credentials [TYPE::]ID::USERNAME[::PASSWORD][::KEYFILE] ...
```

認証情報には`usr_pwd`または`sshkey`の2種類があります。

`usr_pwd`が既定値のため、パラメータ`--credentials my_id::my_username::my_password`は`TYPE=usr_pwd`, `ID=my_id`, `USERNAME=my_username`, `PASSWORD=my_password`として解釈されます。

`sshkey`タイプの場合、4つの要素のみが指定されると4番目の要素はキーファイルへのパスとして解釈されます。5つの要素がある場合、4番目はキーファイルのパスワード、5番目はキーファイルへのパスとして解釈されます。

1つまたは複数の認証情報パラメータを`--credentials`フラグの後に列挙できます。

1.4.3.1 例

```
cicdw jenkins-server create --credentials httpuser::theuser::secret123
```

または

```
cicdw jenkins-server create --credentials usr_pwd::httpuser::theuser::secret123
```

は、ユーザ名theuser、パスワードsecret123を認証情報ID httpuserとしてユーザ名/パスワードの認証情報を作成します。

```
cicdw jenkins-server create --credentials sshkey::sshuser::theuser::path/to/.ssh/key
```

は、ユーザ名theuserとフォルダpath/to/.ssh/にあるキーファイルkeyを認証情報ID sshuserとしてssh認証情報を作成します。

```
cicdw jenkins-server create --credentials
sshkey::sshuser::theuser::keyfilepwd::path/to/.ssh/key
```

は前の例と同じですが、ここではパスフレーズkeyfilepwdを使ってキーファイルのロックを解除します。

1.4.3.2 ビルドジョブでの認証情報の使用

gitベースのビルドジョブは多くの場合、認証情報を必要とします。--credentialsパラメータで作成した認証情報を使用するには、--git-job-credentials-idパラメータ(短縮形は--gjci)を追加します。

```
cicdw jenkins-server create --credentials cred_id::user::pwd --git-job-credentials-id cred_id
```

ユーザ名/パスワード認証はsshベースのリモートURLでは機能しません。

- ssh://git@some-git-server.com/projects/project.git
- ssh://user@some-git-server.com/~/projects/project.git
- git://some-git-server.com/projects/project.git
- git@some-git-server.com:projects/project.git

1.4.4 Jenkinsサーバ ビルドファイルの編集

既定値では、cicdw jenkins-server createコマンドはコンフィグレーション ファイルを一時フォルダに書き込みます。--build-pathパラメータを指定する事で、より永続的な場所に変更できます。Dockerイメージを作成せずにビルドファイルを生成するには、--only- generateフラグを追加します。

```
cicdw jenkins-server create --only-generate --build-path my/build/files/location
```

これにより、フォルダ内に3つのファイルが生成されます。

- Dockerfile - Jenkinsサーバ用のDockerイメージのビルド方法に関する命令
- plugins.txt - Jenkinsサーバにプリインストールするためのプラグインのリスト
- casc.yaml - Jenkinsサーバ用の設定の命令

これらのファイルの編集方法は公式のJCasC文書を参照してください。

編集したファイルを使うには、cicdw jenkins-server createコマンドに--from-build-filesフラグを追加します。新しいサーバを作成する前に、以前に作成したサーバを削除する必要があります。

```
cicdw jenkins-server remove
cicdw jenkins-server create --build-path my/build/files/location --from-build-files
```

1.5 Unityによるユニットテストの記述および実行方法

UnityはMITライセンスのフレームワークで、ユニットテストを簡単に記述できます。公式の入門ガイドは『Unity - Getting Started』で参照できます。

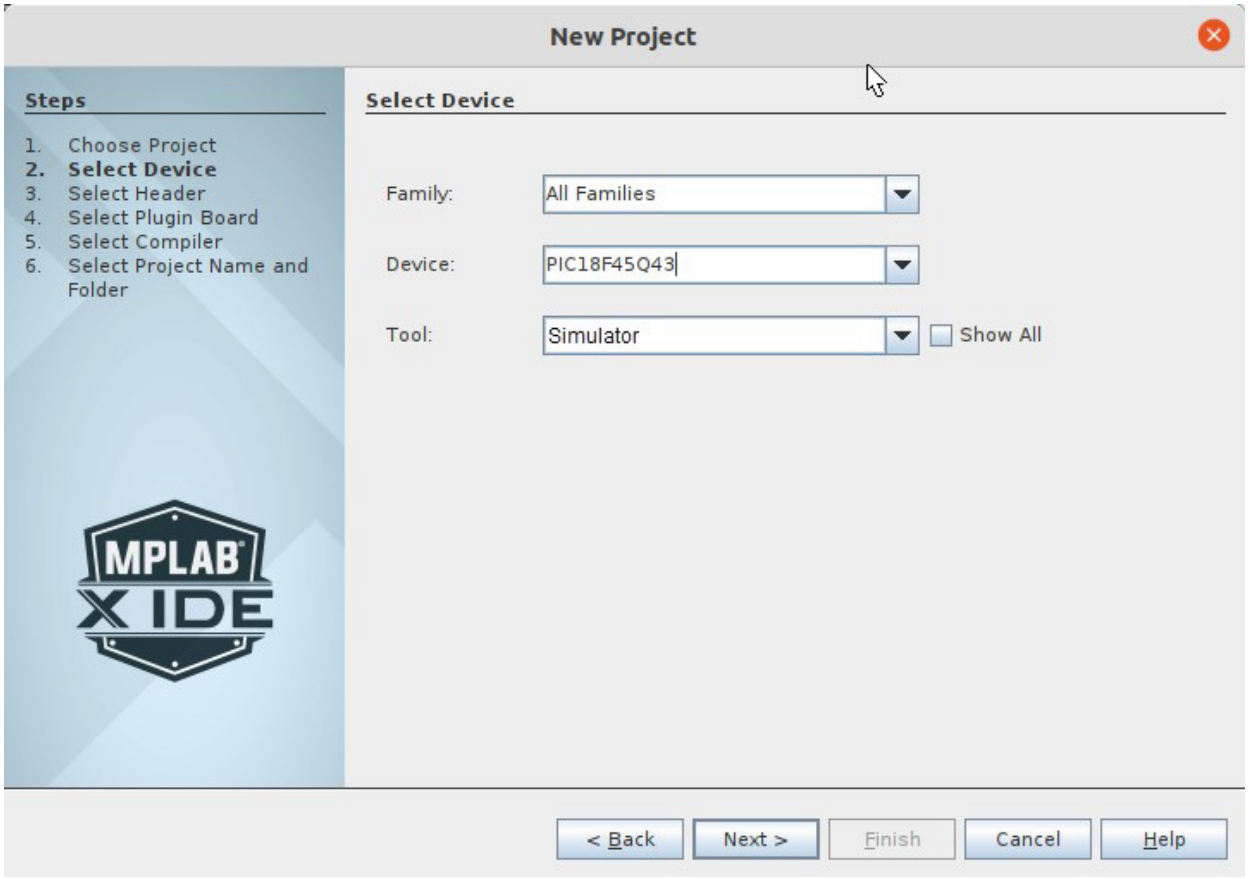
MPLAB X IDEのコンテキストで本製品を使用する例を以下のセクションに示します。

1.5.1 シミュレータによるユニットテストの実行

MPLAB Xシミュレータ でUnityを使用する例については以下の手順を実行します。

1.5.1.1 Step 1. プロジェクトの新規作成

MPLAB X IDE  でスタンドアロン プロジェクトを新規作成します。この手順にはデバイスとツールチェーンの選択が含まれています。

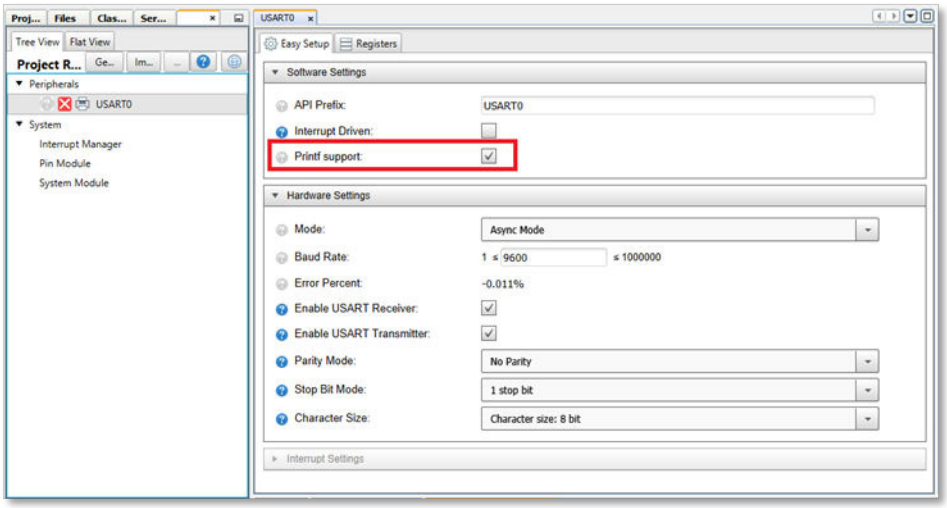


1.5.1.2 Step 2. シリアル出力の設定

Unityは、テスト結果を出力するためにprintfのサポートを必要とします。printfをサポートするには複数の方法があります。

1つの方法としてMPLAB X IDEのプラグインMCC (MPLAB Code Configurator)をインストールして使用するやり方があります。インストールしたら、MCCを開いて[Device Resources]ビューからUSART(またはUART)を追加します。Printfのサポートを設定の一部として必ず有効にしてください。

以下の例は、MCCでPrintfを使ってATtiny817デバイス用に設定したUSARTです。



MCCでUSART(またはUART)を設定したら必ず[Generate]をクリックし、設定したファイルがプロジェクトに追加された事を確認します。

1.5.1.3 Step 3. MDBスクリプトとシミュレータによるシリアル出力の検証

Unityの操作を続ける前にシリアル出力と`printf`が期待通りに動作する事を確認する必要があります。

Step 3. A - main.cからのprintfのテスト

選択したデバイスと`printf`の設定方法によって設定は異なります。

以下のサンプル`main.c`は、デバイスがATtiny817で、設定したUSARTがMCCによって`mcc_generated_files`サブフォルダ内に生成されている事を想定しています。

```
#define F_CPU 1000000UL

#include "mcc_generated_files/mcc.h"
#include <util/delay.h>

int main(void)
{
    // Initialize drivers from MCC
    SYSTEM_Initialize();
    _delay_ms(1000);

    printf("Hello world!\n");
}
```

Step 3. B - シミュレータMDBスクリプトの作成

シミュレータでプログラムをローカルに実行してシリアル通信のログをローカルファイルに記録するMDBスクリプトを作成します。

ATTiny817を使った`mdb.script`の例を以下に示します。

Note: パスをelfファイルに変更する必要があります。

```
# Set tool and device
device ATtiny817
hwtool SIM

# Write serial communication from usart0 to file.
set usart0io.uartioenabled true
set usart0io.output file
set usart0io.outputfile ./sercom_output.txt

# Program and run your binary file
program ./dist/[INSERT_YOUR_BINARY_FILE_PATH_HERE].elf

run
```

```
# Wait 10 seconds before exiting
wait 10000
halt
quit
```

Step 3. C - MDBスクリプトの実行と出力の検証

プロジェクトがビルドされた事を確認したら、コマンドラインを開いてプロジェクトに移動します。mdb.shまたはmdb.batにmdbスクリプトを引数として指定して実行します。以下に例を示します。

```
me@mypc:/project_with_unittests.X$ "/opt/microchip/mplabx/6.00/mplab_platform/bin/mdb.bat"
mdb.sh

device ATtiny817

hwtool SIM

Resetting SFRs
Resetting peripherals

set usart0io.uartioenabled true
set usart0io.output file
set usart0io.outputfile ./sercom_output.txt

program /project_with_unittests.X/dist/default/production/
project_with_unittests.X.production.elf
Programming target...
Resetting SFRs
Resetting peripherals
Program succeeded.

run
Running

wait 10000
halt
Simulator halted
quit
```

sercom_output.txtを開くと、以下のように出力されているはずです。

```
Hello world!
```

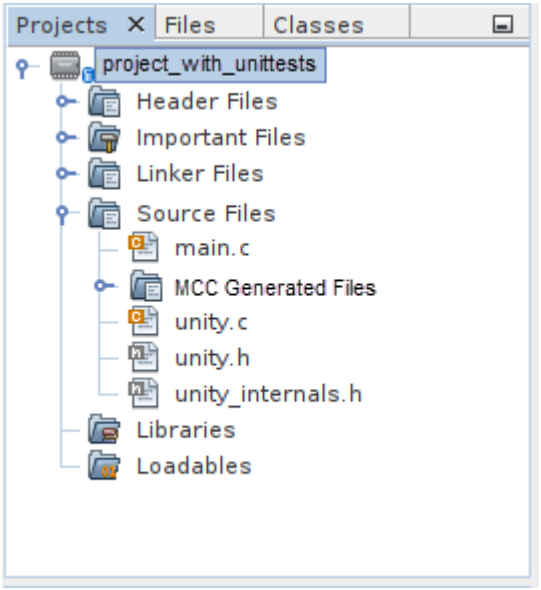
1.5.1.4 Step 4. Unityテスト フレームワークの追加

機能するシリアル出力ができたため、ユニットテストを追加できます。

Unityテスト フレームワークを手動でダウンロードし、プロジェクトに追加する必要があります。必要なファイルはhttps://github.com/ThrowTheSwitch/Unity/tree/master/srcから入手できます。

ファイルをダウンロードしてプロジェクトに追加します。

これでプロジェクトは以下のようになります。



1.5.1.5 Step 5. テストランナーの記述

`main.c`を再記述してテストを追加します。より詳細なテストの記述方法は https://github.com/ThrowTheSwitch/Unity/blob/master/docs/UnityGettingStartedGuide.mdを参照してください。.

以下に例を示します。

```
#define F_CPU 1000000UL

#include "mcc_generated_files/mcc.h"
#include <util/delay.h>

#include "unity.h" void

test_function1() {
    // Simple demo of working test
    TEST_ASSERT_TRUE(1);
}

void test_function2() {
    // Simple demo of failing test
    TEST_ASSERT_FALSE(1);
}

int run_unit_tests(void)
{
    UnityBegin("main.c");
    RUN_TEST(test_function1);
    RUN_TEST(test_function2);
    UnityEnd();
    return 0;
}

int main(void)
{
    // Initialize drivers from MCC
    SYSTEM_Initialize();
    _delay_ms(1000);

    run_unit_tests();
}

void setUp (void) {}
void tearDown (void) {}
```

1.5.1.6 Step 6. シミュレータでのテストの実行

作成済みの`mdb`スクリプトを再実行します(Step 3.C参照)。

出力は以下の通りです。

```
main.c:21:test_function1: PASS
main.c:15:test_function2: FAIL: Expected FALSE Was TRUE

-------------------------------
2 Tests 1 Failures 0 Ignored
 FAIL
```

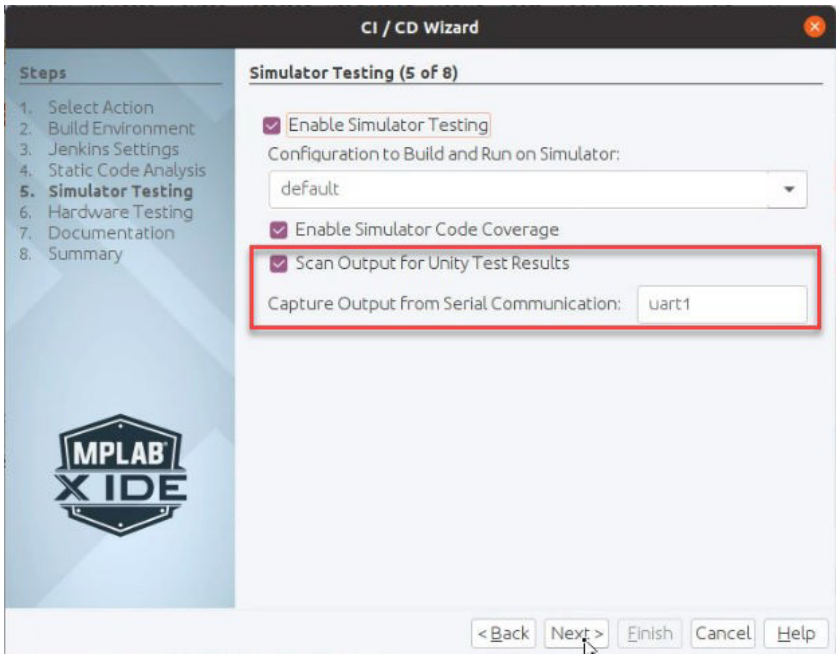
提示されたmdbスクリプトは終了する前に10秒間だけ待機します。テストでもっと時間が必要な場合、mdb.scriptのwaitコマンドでこの値を増やす事ができます。

1.5.1.7 Step 7. CI/CDウィザードでのシミュレータの使用

ローカルで動作する安定した設定ができたら、CI/CDウィザードを実行してテストを組み込みます。

CI/CDウィザードでJenkinsfileを生成する時にシミュレータ テストステップを有効にします。

[Scan Output for Unity Test Results]を有効にし、出力をキャプチャするシリアル通信インスタンス名を入力します。

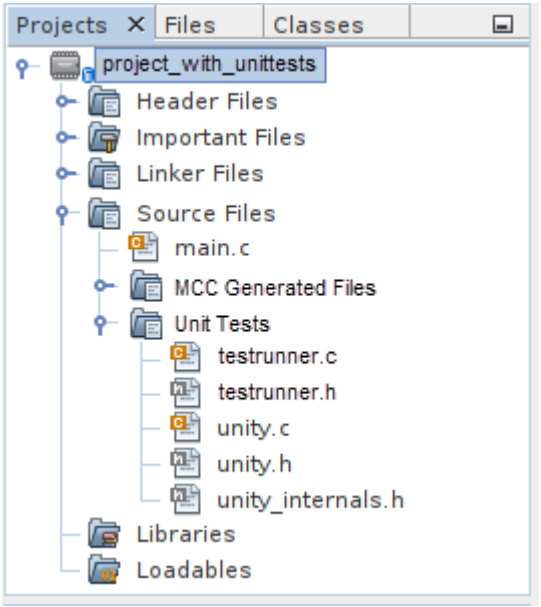


ウィザードは独自のmdbスクリプトを生成し、Jekninsのビルド時にこれを使います。ビルド後にJenkinsが表示する出力からテスト結果が解析されます。

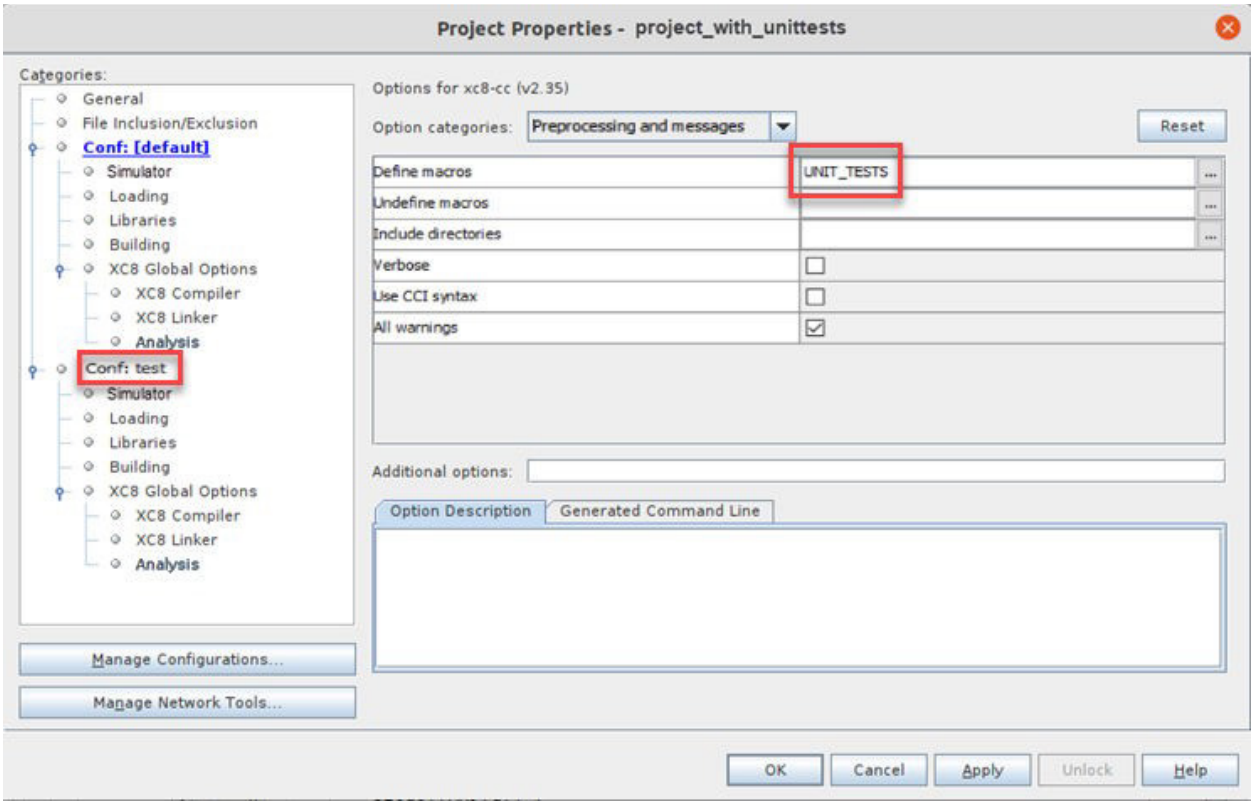
1.5.1.8 オプション手順

テストと量産コードの分離

プロジェクトをよりきれいに整理するために、ユニットサポート ファイルとテストランナーを別のサブフォルダに移動する必要があります。



別のテスト設定をプロジェクトに追加する事もできます。



UNIT_TESTSを定義したテスト設定があり(上図参照)、テスト実行を別のサブフォルダに移動したため、`main.c`を以下のように修正できます。

```
#include "mcc_generated_files/mcc.h"
#include <util/delay.h>

#ifdef UNIT_TESTS
#include "unit_tests/testrunner.h"
#else
int do_normal_stuff();
```

```
#endif

int main(void)
{
    // Initialize drivers from MCC
    SYSTEM_Initialize();
    _delay_ms(1000);

#ifdef UNIT_TESTS
    printf("Starting unit tests\n");
    run_unit_tests();
#else
    printf("Starting normal execution\n");
    return do_normal_stuff();
#endif
}

int do_normal_stuff()
    { return 0;
}
```

これにより、1つの共通プロジェクトでありながら、量産用にビルドを最適化する設定とテストを実行する設定を格納できます。

コードカバレッジの追加

コードカバレッジを有効にする場合、`mdb.script`ファイルで`program`コマンドの前に以下の行を追加します。

```
# Generate simulator coverage data and write it to a file
set codecoverage.enabled Enabled_Reset_on_run
set codecoverage.enableoutputtofile true
set codecoverage.outputtofile simulator_coverage.txt
```

これにより、`mdb`スクリプトを実行しながら`simulator_coverage.txt`にカバレッジデータが書き込まれます。

1.5.2 MPLAB ICE 4でのユニットテストの実行

ハードウェアツールMPLAB ICE 4インサーキット エミュレータでUnityを使用する例については以下の手順を実行します。

1.5.2.1 Step 1. テスト付きプロジェクトの設定とシミュレータでの検証

シミュレータでテストを機能させてからMPLAB ICE 4インサーキット エミュレータで操作を続ける事を推奨します。そのため、1.5.1.「シミュレータによるユニットテストの実行」で説明した手順をまず実行します。テストランナー付きのプロジェクトと`printf`をサポートしたシリアルドライバがある事を確認してください。

シミュレータ上でテストが機能していたら、MPLAB ICE 4で操作を続ける事ができます。シリアル出力をエミュレータでキャプチャするには、下表の通信設定を使う必要があります。

シリアルドライバ設定	必要な値
baudレート	115200
送信ビット	8
受信ビット	8
データパリティ	パリティなし

1.5.2.2 Step 2. ブレークアウト ボードを使ったMPLAB ICE 4とターゲットとの接続

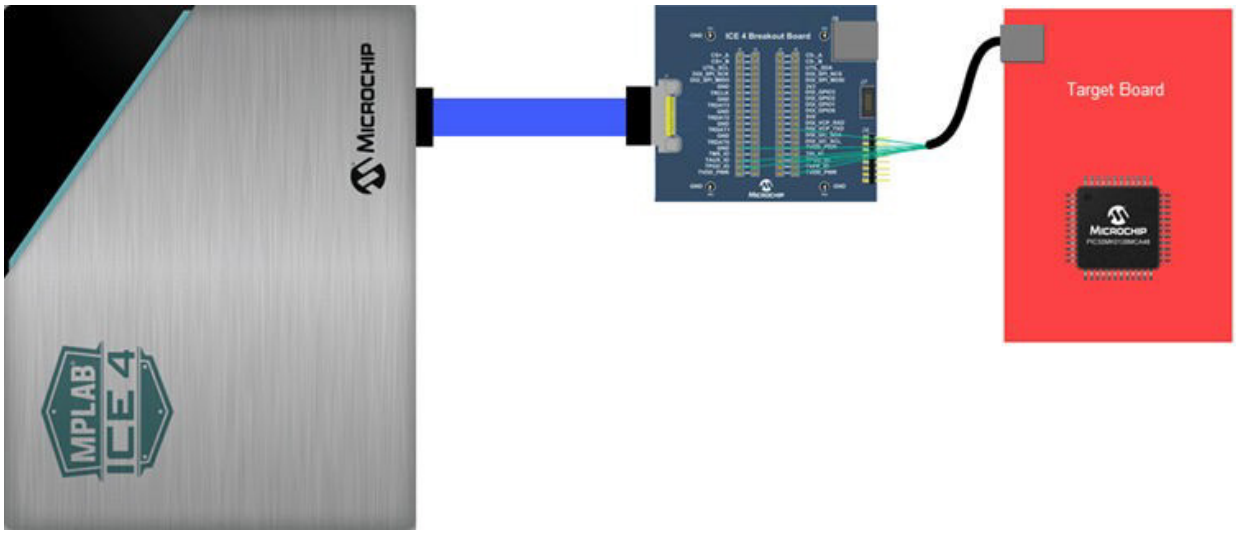
ICE 4ブレークアウト ボード(AC244141)を使ってMPLAB ICE 4をハードウェアに接続します。ICSPピンはデバイスのデバッグとプログラミングに、DGI_VCP_TXDピンはシリアル出力の読み出しに必要です。DGI_VCP_TXDはシリアルドライバ用に設定したピンに接続する必要があります。

表1-1. 接続の概要

ICE 4ブレークアウト ボードピン名	ターゲット
TVPP_IO	ICSPヘッダ1

.....続き	
ICE 4ブレークアウト ボードピン名	**ターゲット**
TVDD_PWR	ICSPヘッダ2
GND	ICSPヘッダ3
TPGD_IO	ICSPヘッダ4
TPGC_IO	ICSPヘッダ5
TAUX_IO	ICSPヘッダ6
DGI_VCP_TXD	設定したシリアルドライバで使用するTXピン

図1-7. ハードウェアの接続



1.5.2.3 Step 3. ローカルICE 4を使った接続の検証

MPLAB ICE 4をコンピュータにUSBで接続します。MPLAB X IDEを開いて使い、エミュレータとICE 4ブレークアウト ボードを使ってターゲットでテスト アプリケーションをプログラミングおよび実行できる事を検証します。

検証に失敗した場合、ブレークアウト ボードの接続を見直す必要があります。

Note: まだシリアル出力はできません。

1.5.2.4 Step 4. ネットワーク ツールとしてのMPLAB ICE 4の設定

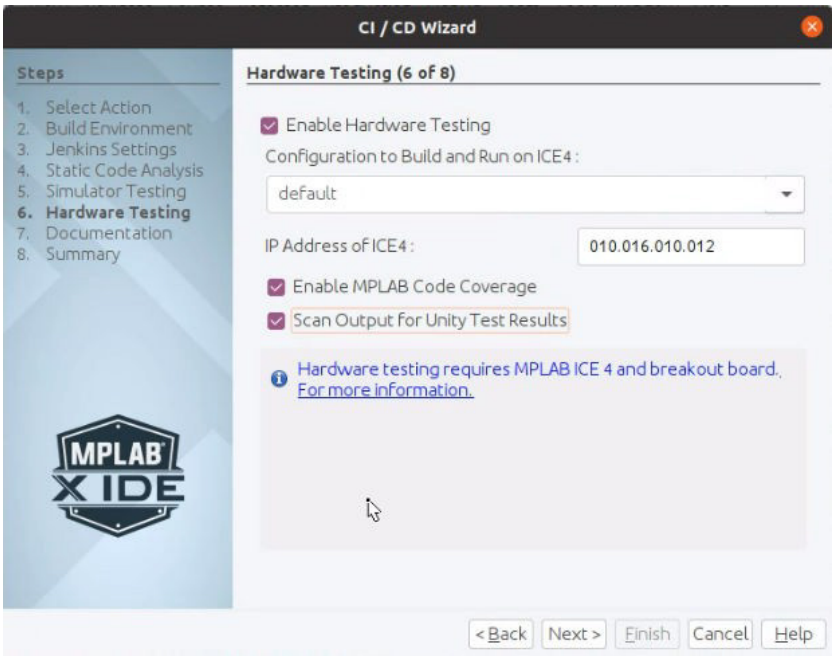
MPLAB X IDEで[Tools] > [Manage Network Tools]を開き、接続しているMPLAB ICE 4がUSBの代わりにEthernetを使うように設定します。MPLAB ICE 4をEthernetケーブルに接続し、USBケーブルを取り外してMPLAB ICE 4の電源を後で切り換える必要があります。詳細は『MPLAB ICE 4インサーキット エミュレータ ユーザガイド』を参照してください。

一度設定すると、ネットワーク ツールのスキャン時に[Manage Network Tools]ダイアログにMPLAB ICE 4が検出され、IPアドレスを記録する必要があります。Ethernetの設定を検証するために、MPLAB X IDEでMPLAB ICE 4をリモートツールとして使ってテスト アプリケーションをプログラミングおよび実行できるようになりました。

Note: ネットワークが静的IPアドレスを提供しない場合、この手順を定期的に繰り返す必要があります。

1.5.2.5 Step 5. CI/CDパイプラインへのMPLAB ICE 4の追加

CI/CDウィザードにより、Jenkinsビルド パイプライン生成時のMPLAB ICE 4ハードウェア テストを有効にする準備が整いました。



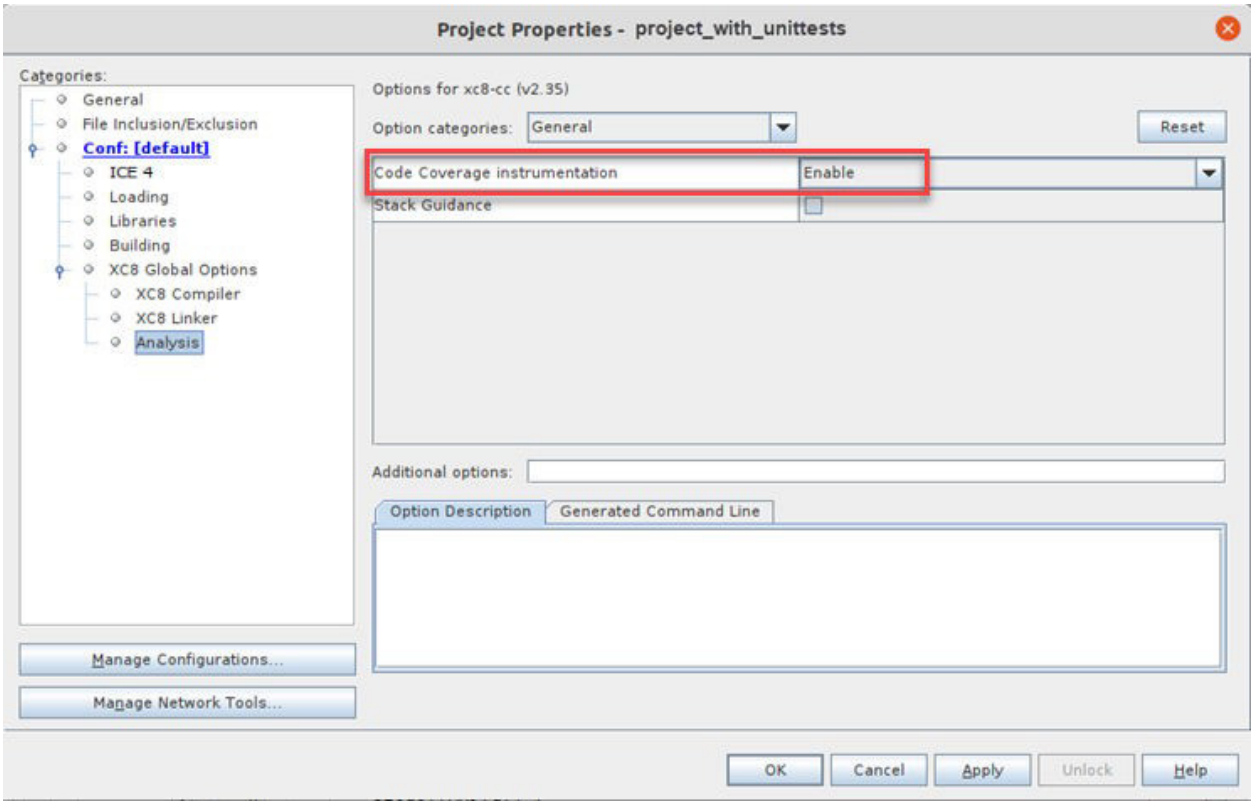
テストするビルド設定を選択します。

MPLAB ICE 4をEthernetで設定する時に記録したIPアドレスを入力します。

カバレッジデータが必要な場合、[Enable MPLAB Code Coverage]を有効にします。ただしこの場合、ビルドにコードカバレッジを実装するようにプロジェクトを設定する必要があります。それには、プロジェクトのプロパティを開き、[Analysis]セクションで[Code Coverage instrumentation]を設定する必要があります(下図参照)。

[Scan Output for Unity Test Results]を有効にします。これにより、シリアルポートからの出力が解析され、Jenkinsビルドジョブでテストレポートが生成されます。

図1-8. コードカバレッジの有効化



設定が全て完了したら、生成したJenkinsビルドジョブが実行時に以下を行います。

- 選択したビルド設定を使ってプロジェクトをビルドする
- MPLAB ICE 4と通信してシリアル出力をキャプチャする
- 生成したMDBスクリプトを使ってテスト アプリケーションを実行する
- 完了したら、シリアル出力を解析してテスト結果を表示する
- カバレッジデータを公開する

1.5.2.6 Step 6. MPLAB ICE4 IPアドレスの更新(任意)

MPLAB ICE 4を後で移動した場合、またはネットワークが動的IPアドレスを使っている場合、新しいIPアドレスでビルドスクリプトを更新しなければならない事があります。

MPLAB X IDEで[Tools] > [Manage Network Tools]を選択し、ツールをスキャンします。MPLAB ICE 4を見つけて新しいIPアドレスを記録します。

JenkinsfileでHARDWARE_TEST_TOOL_IPフィールドを更新します。

```
HARDWARE_TEST_TOOL_IP =  '123.123.12.34'
```

mdb-hardware-script.txtでaddipおよびhwtoolコマンドを更新します。

```
# Add IP of tool so it can be detected
addip 123.123.012.034
# (Optional) List tools to verify that tool was detected
hwtool
# Connect to network configured tool
hwtool ICE4 <ipa>123.123.012.034
```

Note: mdbスクリプトはゼロでパディングするフォーマットを使うため、123.123.12.34ではなく123.123.012.034と表記します。

1.6 Dockerfileの使い方

Dockerはアプリケーションを開発、配布、実行するためのオープン プラットフォームです。DockerはOSレベルの仮想化を実現し、アプリケーションとインフラストラクチャの分離を可能にします。

Dockerfileから命令を読み出してDockerイメージをビルドします。Dockerイメージはテンプレートのようなものです。Dockerイメージを使ってDockerコンテナをビルドおよび実行します。これらの操作にはDockerをインストールしておく必要があります。

DockerのインストールとDockerfileの使用方法の詳細はwww.docker.comを参照してください。

1.6.1 要件

1.6.1.1 ライセンスサーバ

- Dockerコンテナを備えた設定でMPLAB XC PROコンパイラを使うには、ビルドノードからMPLAB XCネットワーク ライセンスサーバにアクセスできる必要があります。『MPLAB XC License Server Manual』(DS50002334)でこのサーバの設定方法を説明しています。
- MPLAB解析ツールスイート (MISRAチェックおよびMPLAB Code Coverage用)等、ライセンスを検証するためにライセンス サーバが必要な機能もあります。

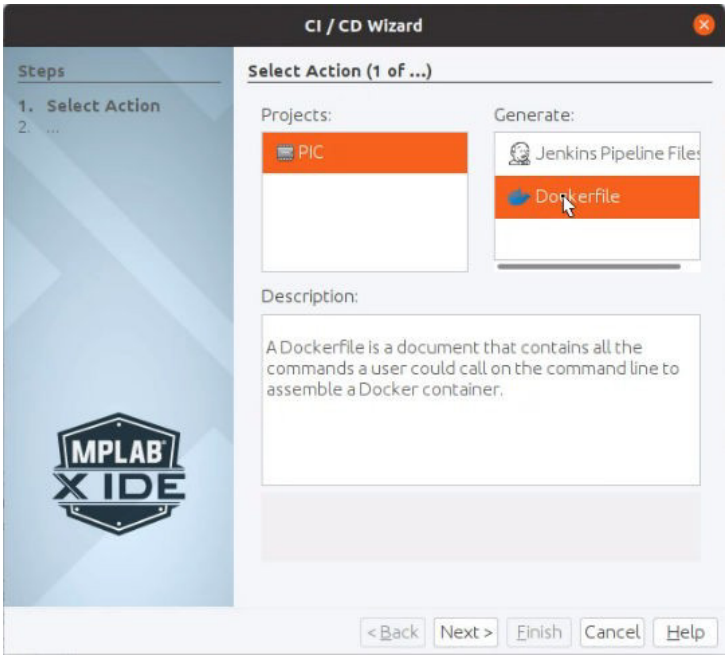
1.6.2 MPLAB X IDEからのDockerfileの生成

1. MPLAB X IDEでプロジェクトを開き、[Tools] > [CI/CD Wizard]を選択します。
2. 出力タイプとして**[Docker File]**を選択します。
3. 残りの情報を必要に応じてウィザードに入力します。以下のセクションを参照してください。
4. 生成後、プロジェクトのベースフォルダにDockerfileが追加されている事を確認します。

1.6.2.1 [CI/CD Wizard]ダイアログ1 - Dockerのみ

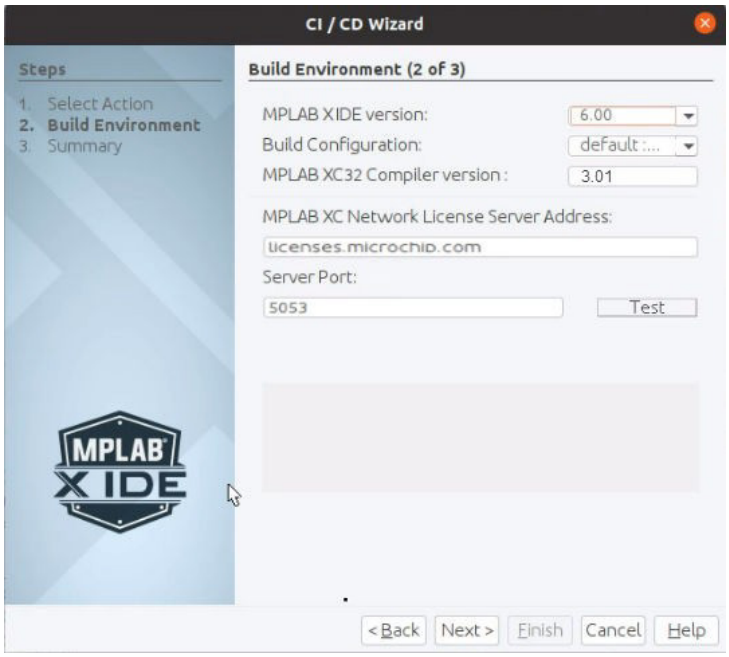
この例では[Docker File]を選択します。**[Next]**をクリックして次へ進みます。

Note: Dockerファイルを作成するにはMPLAB XCコンパイラのライセンスが必要です。



1.6.2.2 [CI/CD Wizard]ダイアログ2 - Dockerのみ

コンテナで使用するためのビルド環境を設定します。ネットワーク サーバのライセンスがある場合、ライセンスへのアクセス方法に関する情報をビルド用に提供できます。ライセンスを設定していない場合、テキストボックスを空白のままにします(ただし、注意が表示されます)。



オプション	説明
MPLAB IDE version	サポートされているバージョンをドロップダウン リストから選択します。既定値では最新バージョンです。
Build Configuration	このビルド用のプロジェクト設定を選択します。
MPLAB XC Compiler version	このビルド用のコンパイラ バージョンを選択します。
MPLAB XC Network License Server Address	サーバのアドレスを入力します。例: licenses.microchip.com
Server Port	サーバのポート番号を入力します。例: 5053
情報注記(条件による)	ワークステーション ライセンスがある場合、Network Serverライセンスの購入に関する情報が表示されます。

1.6.2.3 [CI/CD Wizard]ダイアログ3 - Dockerのみ

[Summary]で、Dockerfileのみが生成される事を確認します。

設定項目を変更するには**[Back]**をクリックします。設定を完了するには**[Finish]**をクリックします。CI/CDウィザードが設定情報に基づいてDockerfileを生成します。

1.6.2.4 CI/CDウィザードの出力 - Dockerのみ

MPLAB X IDEに以下が出力されます。

1. 生成したファイル([Output]ウィンドウ)
2. プロジェクト フォルダに追加された結果ファイル
3. オンラインヘルプ(専用のウィンドウ)
4. Dockerfile(専用のウィンドウ)

この出力はJenkins出力のサブセットです。

1.6.3 生成したDockerfileの内容

生成したDockerfileは特定のMPLAB X IDEプロジェクト専用です。

設定は以下を含みます。

- Debian GNU/Linux 10
- curl、make、unzip等の基本的なツール
- バイナリを含むMPLAB X IDEのインストール:
  - mdb - Microchip Debugger
  - prjMakefilesGenerator - MPLAB X IDEプロジェクト用にMakefileを生成するためのCLI
  - xclm - XCツールチェーン ライセンス マネージャ
  - IPE - MPLAB統合プログラミング環境
- MPLAB XCツールチェーンのインストール。このインストールはMPLAB X IDEプロジェクトで指定したものと同じバージョンを使う
- MPLAB X IDEプロジェクトで指定したのと同じDFP(デバイスファミリ パック)

1.6.4 次の推奨手順

生成したDockerfileはMPLAB X IDEプロジェクトのビルド環境の設定に使用できます。詳細はDockerfileリファレンスを参照してください。

1.6.4.1 ローカルでのイメージのテスト

Dockerfileを使うと、Dockerイメージをローカルで簡単に作成およびテストできます。Dockerをローカルにインストールしていない場合、www.docker.comから入手できます。

以下に、イメージをビルドして実行するコマンドを示します。

```
# Build an image based on the generated Dockerfile
docker build --tag mplabx-env --file Dockerfile .

# Check that the 'mplabx-env' image was created
docker images

# Start a docker container in interactive mode and map MPLAB X project source
docker run -it -v PATH_TO_MY_PROJECT_HERE:/usr/src/project mplabx-env bash
```

コンテナを起動したら、割り当てられたMPLAB X IDEプロジェクトをビルドするために以下のコマンドを実行できます。

```
# Navigate to the mapped MPLAB X project
cd /usr/src/project

# Generate Makefiles
prjMakefilesGenerator .

# Build project
make

# List files generated in the dist folder
find dist
```

1.6.4.2 CI/CDフレームワークでのDockerfileの使用

CI/CDウィザードは**Jenkins**のサポートを備えています。このフレームワーク用のサポートファイルを生成できます。他のフレームワークについては、そのフレームワークの文書でDockerの使用方法を参照する必要があります。

1.6.4.3 追加ツール/Linuxパッケージによる Dockerfileの拡張

Dockerfileを変更する事で追加のLinuxパッケージをインストールできます。最初にインタラクティブ モードでイメージを起動すると、コマンドラインから手動でパッケージをインストールできます。

```
apt-get update
apt-get install [packageName]
```

パッケージが入手可能で機能している事を検証していたら、Dockerfileに追加できます。

```
RUN apt-get update -yq \
 && apt-get install -yq --no-install-recommends
 \ [packageName] \
 && apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

Note: 最後のクリーンアップ行は、Dockerイメージのサイズを可能な限り最低限に抑えるために使います。

1.7 CICDWコマンドライン ツール

コマンドライン版のCI/CDウィザードを実行する場合、<MPLAB X IDEバージョンのインストール ディレクトリ>mplab_platform/bin/cicdw.exeの下にあります。

cicdw.exe -hを使うと、オプションに関するヘルプが表示されます(以下参照)。コマンド固有のヘルプを表示するには、コマンドと-hを使います(例: cicdw.exe generate-jenkins -h)。

Note: 以下のように全てのコマンド オプションを同じ行に入力する必要があります。

```
cicdw.exe generate-jenkins --misra --simulator ...
```

cicdw Help

```
usage: cicdw [-h] [-v] [-q] [--version]
             {generate-jenkins,generate-docker,generate-doxyfile,container,jenkins-server}
             ...

CI/CD Wizard for MPLAB X projects
```

```
optional arguments:
  -h, --help            show this help message and exit
  -v                    Enable verbose logging (Use -vv for extra verbose
                        logging)
  -q                    Show less output. (Use -qq to only show errors and
                        warnings and -qqq to only show errors)
  --version             Print version information

Commands:
  {generate-jenkins,generate-docker,generate-doxyfile,container,jenkins-server}
    generate-jenkins    Generate jenkins build files (Jenkinsfile and
                        Dockerfile)
    generate-docker     Generate dockerfile
    generate-doxyfile   Generate Doxyfile for doxygen documentation
    container           Manage local build container
    jenkins-server      Manage jenkins server

Type {command} -h for command specific help
```

1.7.1 CICDW Jenkinsの例

太字は開発者が指定した内容を表します。以下の内容は全て1行にする必要があります。

```
MPLABPath\mplab-nbplatform\nbbuild\netbeans\bin\cicdw.exe> generate-jenkins
--xclm ToolchainPath \xc8\v2.32\bin\xclm.exe
--configuration default
--mplabx-version 6.00
--toolchain-version 1.61
--license-server-name licenses.microchip.com
--license-server-port 5053
--docker-host-agent-label docker
--misra --use-cppcheck-trend
--simulate --simulate-with-coverage --simulate-with-unity --simulate-sercom-output uart1
 --simulate-build-configuration default
--doxygen --doxygen-configuration-file Doxyfile --doxygen-generate-doxyfile
--doxygen-project-name test --doxygen-input-encoding ISO-8859-1
--doc-artifact-html doc-html.zip --doc-artifact-latex doc-latex.zip --doc-artifact-pdf doc-
pdf.zip
--hardware-test --hardware-test-build-configuration default --hardware-test-tool-ip
192.168.16.3
--hardware-test-with-coverage --hardware-test-with-unity
ProjectPath
```

1.7.2 CICDW Dockerの例

太字は開発者が指定した内容を表します。以下の内容は全て1行にする必要があります。

```
MPLABPath\mplab-nbplatform\nbbuild\netbeans\bin\cicdw.exe> generate-docker
--xclm ToolchainPath\xc8\v2.32\bin\xclm.exe
--configuration default
--mplabx-version 6.00
--toolchain-version 2.32
--license-server-name licenses.microchip.com
--license-server-port 5053
ProjectPath
```

Microchip社ウェブサイト

Microchip社はウェブサイト(www.microchip.com)を通してオンライン サポートを提供しています。当ウェブサイトでは、お客様に役立つ情報やファイルを簡単に見つけ出せます。以下を含む各種の情報をご覧になれます。

- **製品サポート** - データシートとエラッタ、アプリケーション ノートとサンプル プログラム、設計リソース、ユーザガイドとハードウェア サポート文書、最新のソフトウェアと過去のソフトウェア
- **技術サポート** - FAQ(よく寄せられる質問)、技術サポートのご依頼、オンライン ディスカッション グループ、Microchip社のデザイン パートナー プログラムおよびメンバーリスト
- **ご注文とお問い合わせ** - 製品セレクタと注文ガイド、最新プレスリリース、セミナー/イベントの一覧、お問い合わせ先(営業所/正規代理店)の一覧

製品変更通知サービス

Microchip社の製品変更通知サービスは、お客様にMicrochip社製品の最新情報をお届けする配信サービスです。ご興味のある製品ファミリまたは開発ツールに関する変更、更新、リビジョン、エラッタ情報をいち早くメールにてお知らせします。

http://www.microchip.com/pcnにアクセスし、登録手続きをしてください。

お客様サポート

Microchip社製品をお使いのお客様は、以下のチャンネルからサポートをご利用頂けます。

- 正規代理店
- 技術サポート

サポートは正規代理店にお問い合わせください。各地の営業所もご利用になれます。本書の最後のページに各国の営業所の一覧を記載しています。

技術サポートは以下のウェブページからもご利用頂けます。www.microchip.com/support

Microchip社のデバイスコード保護機能

Microchip 社製品のコード保護機能について以下の点にご注意ください。

- Microchip社製品は、該当するMicrochip 社データシートに記載の仕様を満たしています。
- Microchip社では、通常の条件ならびに動作仕様書の仕様に従って使った場合、Microchip 社製品のセキュリティ レベルは、現在市場に流通している同種製品の中でも最も高度であると考えています。
- Microchip社はその知的財産権を重視し、積極的に保護しています。Microchip 社製品のコード保護機能の侵害は固く禁じられており、デジタル ミレニアム著作権法に違反します。
- Microchip社を含む全ての半導体メーカーで、自社のコードのセキュリティを完全に保証できる企業はありません。コード保護機能とは、Microchip 社が製品を「解読不能」として保証するものではありません。コード保護機能は常に進化しています。Microchip 社では、常に製品のコード保護機能の改善に取り組んでいます。

法律上の注意点

本書および本書に記載されている情報は、Microchip 社製品を設計、テスト、お客様のアプリケーションと統合する目的を含め、Microchip 社製品に対してのみ使う事ができます。それ以外の方法でこの情報を使う事はこれらの条項に違反します。デバイス アプリケーションの情報は、ユーザの便宜のためにのみ提供されるものであり、更新によって変更となる事があります。お客様のアプリケーションが仕様を満たす事を保証する責任は、お客様にあります。その他のサポートはMicrochip 社正規代理店にお問い合わせ頂くか、https://www.microchip.com/en-us/support/design-help/client-support-servicesをご覧ください。

Microchip 社は本書の情報を「現状のまま」で提供しています。Microchip 社は明示的、暗黙的、書面、口頭、法定のいずれであるかを問わず、本書に記載されている情報に関して、非侵害性、商品性、特定目的への適合性の暗黙的保証、または状態、品質、性能に関する保証をはじめとするいかなる類の表明も保証も行いません。

いかなる場合もMicrochip 社は、本情報またはその使用に関連する間接的、特殊的、懲罰的、偶発的または必然的損失、損害、費用、経費のいかんにかかわらず、またMicrochip 社がそのような損害が生じる可能性について報告を受けていた場合あるいは損害が予測可能であった場合でも、一切の責任を負いません。法律で認められる最大限の範囲を適用しようとも、本情報またはその使用に関連する一切の申し立てに対するMicrochip 社の責任限度額は、使用者が当該情報に関連してMicrochip 社に直接支払った額を超えません。

Microchip 社の明示的な書面による承認なしに、生命維持装置あるいは生命安全用途にMicrochip社の製品を使う事は全て購入者のリスクとし、また購入者はこれによって発生したあらゆる損害、クレーム、訴訟、費用に関して、Microchip 社は擁護され、免責され、損害をうけない事に同意するものとします。特に明記しない場合、暗黙的あるいは明示的を問わず、Microchip社が知的財産権を保有しているライセンスは一切譲渡されません。

商標

Microchip 社の名称とロゴ、Microchip ロゴ、Adaptec、AVR、AVRロゴ、AVR Freaks、BesTime、BitCloud、CryptoMemory、CryptoRF、dsPIC、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi ロゴ、MOST、MOST ロゴ、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32 ロゴ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST ロゴ、SuperFlash、Symmetricom、SyncServer、Tachyon、TimeSource、tinyAVR、UNI/O、Vectron、XMEGA は米国とその他の国におけるMicrochip TechnologyIncorporated の登録商標です。

AgileSwitch、APT、ClockWorks、The Embedded Control SolutionsCompany、EtherSynch、Flashtec、Hyper Speed Control、HyperLightLoad、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus ロゴ、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、TrueTime、ZL は米国におけるMicrochip Technology Incorporated の登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、Augmented Switching、BlueSky、BodyCom、Clockstudio、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、Espresso T1S、EtherGREEN、GridTime、IdealBridge、In-Circuit Serial Programming、ICSP、INICnet、Intelligent Paralleling、IntelliMOS、Inter-Chip Connectivity、JitterBlocker、Knob-on-Display、KoD、maxCrypto、maxView、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified ロゴ、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、RippleBlocker、RTAX、RTG4、SAM-ICE、Serial Quad I/O、simpleMAP、SimpliPHY、SmartBuffer、SmartHLS、SMART-I.S.、storClad、SQI、SuperSwitcher、SuperSwitcher II、Switchtec、SynchroPHY、TotalEndurance、Trusted Time、TSHARC、USBCheck、VariSense、VectorBlox、VeriPHY、ViewSpan、WiperLock、XpressConnect、ZENAは米国とその他の国におけるMicrochip Technology Incorporated の商標です。

SQTP は米国におけるMicrochip Technology Incorporated のサービスマークです。

Adaptec ロゴ、Frequency on Demand、Silicon Storage Technology、Symmcom はその他の国におけるMicrochip Technology Incorporatedの登録商標です。

GestIC は、その他の国におけるMicrochip Technology Germany II GmbH & Co. KG (Microchip Technology Incorporated の子会社) の登録商標です。

その他の商標は各社に帰属します。

ISBN: 978-1-6683-0236-1

品質管理システム

Microchip社の品質管理システムについてはwww.microchip.com/qualityをご覧ください。

各国の営業所とサービス

南北アメリカ	アジア/太平洋	アジア/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術サポート: http://www.microchip.com/support URL: www.microchip.com **アトランタ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **オースティン、TX** Tel: 512-257-3370 **ボストン** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **シカゴ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ダラス** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **デトロイト** Novi, MI Tel: 248-848-4000 **ヒューストン、TX** Tel: 281-894-5983 **インディアナポリス** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ロサンゼルス** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ローリー、NC** Tel: 919-844-7510 **ニューヨーク、NY** Tel: 631-435-6000 **サンノゼ、CA** Tel: 408-735-9110 Tel: 408-436-4270 **カナダ - トロント** Tel: 905-695-1980 Fax: 905-695-2078	**オーストラリア - シドニー** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10 -8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 **中国 - 香港SAR** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 厦門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**インド - バンガロール** Tel: 91-80-3090-4444 **インド - ニューデリー** Tel: 91-11-4160-8631 **インド - プネ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ソウル** Tel: 82-2-554-7200 **マレーシア - クアラルンプール** Tel: 60-3-7651-7906 **マレーシア - ペナン** Tel: 60-4-227-8870 **フィリピン - マニラ** Tel: 63-2-634-9065 **シンガポール** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **タイ - バンコク** Tel: 66-2-694-1351 **ベトナム - ホーチミン** Tel: 84-28-5448-2100	**オーストリア - ヴェルス** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **デンマーク - コペンハーゲン** Tel: 45-4485-5910 Fax: 45-4485-2829 **フィンランド - エスポー** Tel: 358-9-4520-820 **フランス - パリ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ドイツ - ガーヒンク** Tel: 49-8931-9700 **ドイツ - ハーン** Tel: 49-2129-3766400 **ドイツ - ハイルブロン** Tel: 49-7131-72400 **ドイツ - カールスルーエ** Tel: 49-721-625370 **ドイツ - ミュンヘン** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ドイツ - ローゼンハイム** Tel: 49-8031-354-560 **イスラエル - ラーナナ** Tel: 972-9-744-7705 **イタリア - ミラノ** Tel: 39-0331-742611 Fax: 39-0331-466781 **イタリア - パドヴァ** Tel: 39-049-7625286 **オランダ - ドリューネン** Tel: 31-416-690399 Fax: 31-416-690340 **ノルウェー - トロンハイム** Tel: 47-7288-4388 **ポーランド - ワルシャワ** Tel: 48-22-3325737 **ルーマニア - ブカレスト** Tel: 40-21-407-87-50 **スペイン - マドリッド** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **スウェーデン - ヨーテボリ** Tel: 46-31-704-60-40 **スウェーデン - ストックホルム** Tel: 46-8-5090-4654 **イギリス - ウォーキンガム** Tel: 44-118-921-5800 Fax: 44-118-921-5820