CI/CD Wizard in MPLAB X IDE User's Guide

Notice to Development Tools Customers

Important:
All documentation becomes dated, and Development Tools manuals are no exception. Our tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our website (www.microchip.com/) to obtain the latest version of the PDF document.

Documents are identified with a DS number located on the bottom of each page. The DS format is DS<DocumentNumber><Version>, where <DocumentNumber> is an 8-digit number and <Version> is an uppercase letter.

For the most up-to-date information, find help for your tool at onlinedocs.microchip.com/.

Table of Contents

1. CI/CD Wizard

Note: This content is also in the "MPLAB X IDE User’s Guide" (DS-50002027).

The *MPLAB X IDE CI/CD Wizard* helps you set up an automated build and test pipeline tailored to your MPLAB X IDE project.

CI/CD Requirements

The CI/CD Wizard is a feature of MPLAB X IDE, so the requirements found in “Release Notes for MPLAB X IDE” apply. In addition:

- An MPLAB XC **Workstation or Network Server license** is needed to enable the CI/CD Wizard for script generation.
- An MPLAB XC **Network Sever license** will be required to build your projects with PRO optimization levels in a generated Docker container.
- If you are using a **virtual machine network license**, ensure that this license and any Microchip **Network Server licenses** are placed in the same folder.
- If using the Jenkins Pipeline:
  - The MISRA Check (Static Code Analysis) and MPLAB Code Coverage (Hardware Testing) tools will require MPLAB X IDE v6.00 (or later) and an MPLAB Analysis Tool Suite **Network Server license** for tool execution in a generated Docker container.
  - Hardware Testing will require a tool with networking capabilities which includes:
    - MPLAB ICE 4 in-circuit emulator, supported in MPLAB X IDE v6.00 or later.
    - MPLAB ICD 5 in-circuit debugger, supported in MPLAB X IDE v6.10 or later.
- To use Jenkins and/or Docker, you will need to install these applications. For addition requirements when using Jenkins, see 1.3.1. Prerequisites.
  Note: MPLAB X IDE CI/CD Wizard output execution relies on the third-party applications Jenkins and Docker. For issues with these tools, please see the corresponding website for support.

CI/CD Standard Use

The standard operating system used with Jenkins-Docker or Docker is Linux OS, and for the CI/CD Wizard output is specifically Debian Linux. To use other operating systems, please refer to the documentation for Jenkins and Docker. For example:

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

1.1 About CI/CD

CI/CD (or CICD) is a process for the continuous integration and continuous delivery or deployment of software. Instead of developing software modules fully and then trying to integrate them and deliver a product at the end (as has been done in the past), software is continually integrated and continuously delivered or deployed in small increments, ensuring that a product can be produced with minimal issues at any point.

Continuous integration moves the integration and testing activity early into the development cycle. This helps you reduce the project risks and increasing the likelihood of a successful product.

To support the CI/CD pipeline, containers can be used to build and test code using a consistent environment that the team can share. Developing the instruction files to create and run a container (e.g., Dockerfile) and run container contents (e.g., Jenkinsfile) is non-trivial. This is where the CI/CD Wizard can help with development.

1.1.1 Why Continuous Integration with build and test automation?

Continuous Integration (CI) is a developer practice of frequently committing your code changes to a source code repository, e.g., git. Each commit triggers an automated build and test cycle. If the build breaks or a test fails, you are notified. Doing Continuous Integration helps build quality into the product from the start, coordinates code changes with your team, and reduces the risks associated with late integration and late validation feedback.

The Continuous Integration developer practice works together with a **build automation** system. This system automatically starts jobs/processes like compile, test, package, and deploy. This helps you keep the code quality high by uncovering issues early. There are many different build automation tools available. Jenkins is a free and open-source software. It is a popular tool that we will use as an example. If you are using a different system, look at the Jenkinsfiles generated by the CI/CD Wizard for inspiration.

1.1.2 Docker and Containers

A key concept in build automation is **containerization**. When executing a build job, it should produce repeatable and consistent results independent of where and when the build was executed. **Docker containers** can help achieve this. Use of Docker containers enables repeatability, flexibility, and scalability. Every time a job is executed it has the same environment. Docker provides OS-level virtualization which enables you to separate your applications from your infrastructure.

For more information about how to install Docker and use Dockerfiles, see www.docker.com

1.2 CI/CD Wizard Feature Overview

1. Create tailor-made **Dockerfiles**. These files are recipes for generating Docker images. Docker images are used to create Docker containers. Docker containers contain all your MPLAB X IDE project settings needed to build, i.e., the version of MPLAB X IDE, MPLAB XC toolchain, XC license handling, Device File Pack (DFP) and tool pack.
2. Create **Jenkinsfiles**. These files are used to set up a Jenkins pipeline that describes how to build and test your MPLAB X IDE project on a Jenkins Build server using Docker containers.
3. Handle MPLAB XC PRO compiler licenses in a build server setting.
4. Create and launch a Docker container using a **Jenkins server**. This enables you to quickly test out build pipelines or serves as a base for a locally-hosted Jenkins server.
5. Set up the automated build pipeline to include steps like:
   - MISRA checking
   - Executing tests with the simulator
   - Executing tests on hardware
   - Collecting code coverage data
   - Generating documentation with Doxygen

1.3 Getting Started with Jenkins

Jenkins is an open source automation server which enables developers to automate build, test, and deploy their software. For more information go to Jenkins official site.

This section describes the steps needed to build an MPLAB X IDE project on a Jenkins server.

1.3.1 Prerequisites

1.3.1.1 Jenkins Server

- A Jenkins server needs to be available. If you do not have access to one you can set one up yourself by following the Installing Jenkins guide, or you can use the `cicdw jenkins-server` command to create a local test server (see 1.4. Getting Started with Jenkins Server).

- You need to have a user account that allows you to configure a new build job on the Jenkins server.
- The Jenkins server needs to have build nodes that supports Docker. If not see Jenkins and Docker.
- The Jenkins server must be setup with the following plugins:
  - Pipeline Aggregator View
  - Pipeline Utility Steps
  - Docker Pipeline
  - Workspace Cleanup
  - Warnings Next Generation

  Note: Additional plugins may be required for optional build steps, such as MISRA and Simulator.

1.3.1.2 Source Control System

- Your project source code needs to be hosted on a source control system (example: Subversion, CVS, or Git).
- The Jenkins server needs to support your source control system and have access to your project repository.

1.3.1.3 License Server

- An MPLAB XC license server needs to be configured and available on a given address and port. For more information see MPLAB XC License Server Manual.
- Other license servers may be necessary to support other licensed options, such as the MPLAB Analysis Tool Suite (MISRA Check and MPLAB Code Coverage).

1.3.1.4 Testing Tools

In order to use either software (e.g., Simulator) or hardware (e.g., MPLAB ICE 4 in-circuit emulator) tools for testing within a container, you must consider your setup.

- Use Unity to perform unit tests and output results with `printf` or `putchar` using serial communications. For hardware tools, additional code is required at the target. To learn more about Unity, see 1.5. How to Write and Run Unit Tests with Unity. For information about the virtual COM port on hardware tool, see the tool user's guide section "USB CDC Virtual COM Port."
- Use the Microchip Debugger (MDB) command-line interface to support basic debug commands. For hardware tools, use MDB to program the target. To learn more about MDB, see Microchip Debugger (MDB) User's Guide.

1.3.2 Step 1 - Generate Jenkinsfiles from MPLAB X IDE

1. Open your project in MPLAB X IDE and navigate to *Tools>CI/CD Wizard*.
2. Select **Jenkins Pipeline files** as output type.
3. Step through the wizard and select the tools you would like to run in the pipeline.
4. After generation, verify that a *Jenkinsfile* and *Dockerfile* have been added to your project base folder. Other files are may also be added depending on the selected options.

1.3.2.1 CI/CD Wizard Dialog 1

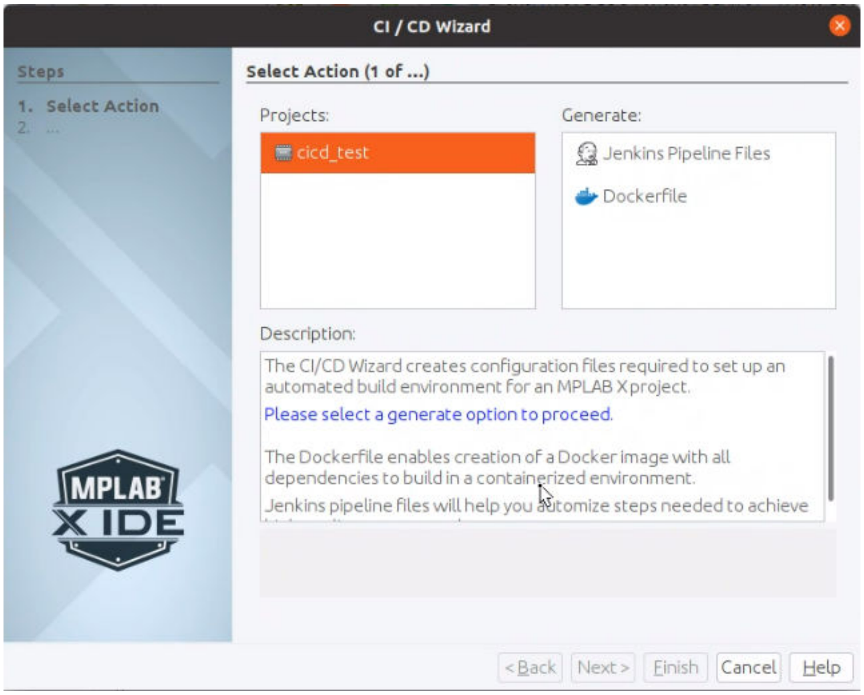
The first dialog of the wizard displays the following note and only the **Cancel** button is active if you do not have an MPLAB XC C compiler license.

To generate files for use in a CICD pipeline, you need a licensed compiler.

See 1. CI/CD Wizard for supported licenses. You can purchase a license for your compiler under MySoftware. Click the **Configure Licenses** button to go to the Change Licensing Type dialog to set up your license.

If you have a compiler license, you will be able to select what file(s) you want to generate. Select Jenkins to create a Jenkinsfile and Dockerfile. Select Docker to create only a Dockerfile.

Figure 1-1. No Generate Selection



The CI/CD Wizard creates configuration files required to set up an automated build environment for an MPLAB X project.

Please select a generate option to proceed.

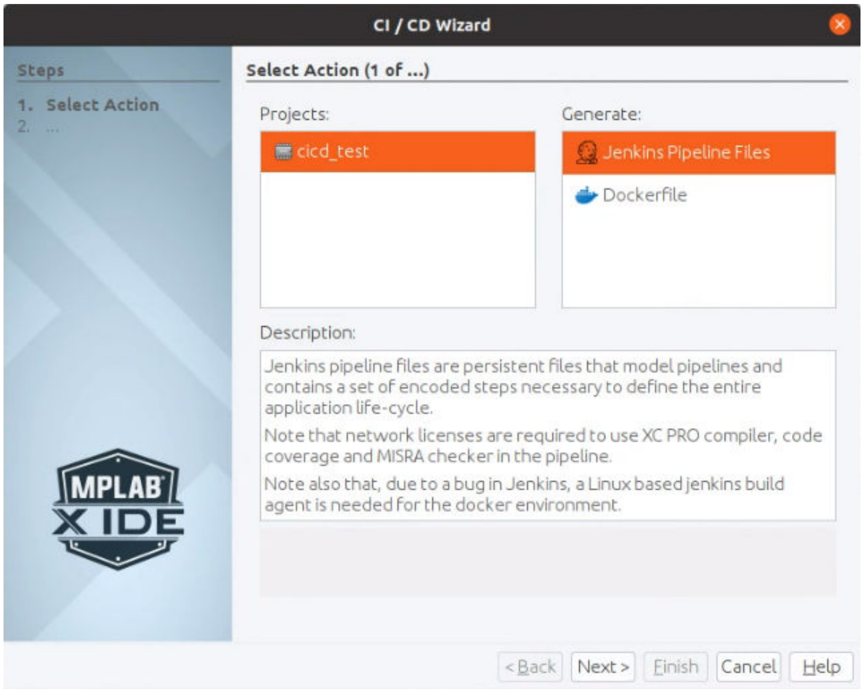
The Dockerfile enables creation of a Docker image with all dependencies to build in a containerized environment.

Jenkins pipeline files will help you automize steps needed to achieve

For this example, select “Jenkins Pipeline Files.” Read the text about this selection that now appears under “Description.”

Click **Next** to proceed through the dialogs.

Figure 1-2. Generate Selection



Jenkins pipeline files are persistent files that model pipelines and contains a set of encoded steps necessary to define the entire application life-cycle.

Note that network licenses are required to use XC PRO compiler, code coverage and MISRA checker in the pipeline.

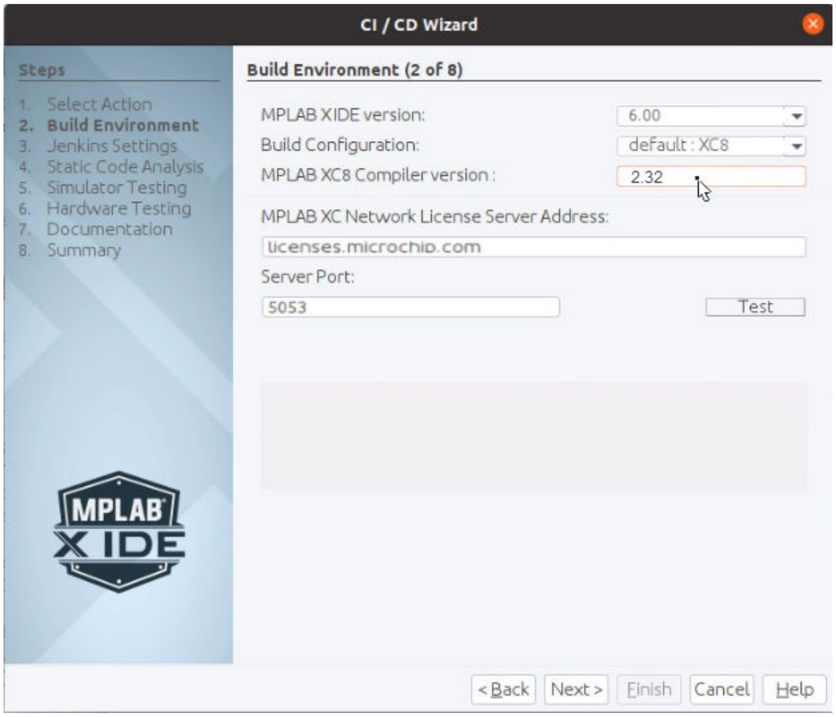
Note also that, due to a bug in Jenkins, a Linux based jenkins build agent is needed for the docker environment.

1.3.2.2 CI/CD Wizard Dialog 2

Set up the build environment to be used in a container. If you have a Network Server license, you can provide information on how to access it for the build. You can even test the connection (see figure below).

Note: If you have already set up the Network Server license using MPLAB X IDE *Tools>Licenses*, that information will auto-populate in this dialog.

Figure 1-3. Network Server License



Build Environment (2 of 8)

MPLAB XIDE version: 6.00

Build Configuration: default : XC8

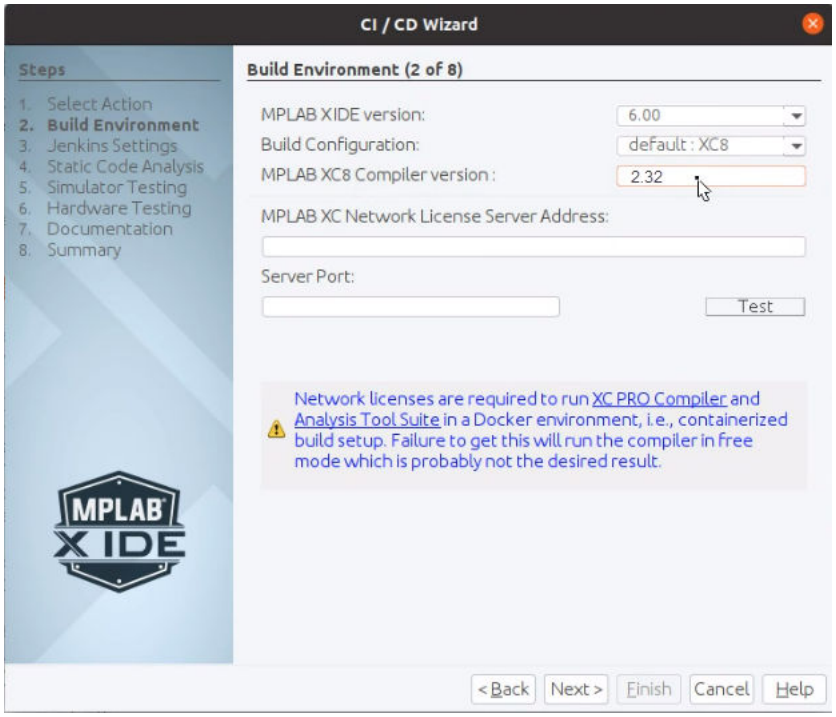
MPLAB XC8 Compiler version : 2.32

MPLAB XC Network License Server Address: licenses.microchip.com

Server Port: 5053

Otherwise leave the text boxes blank (though a note will appear).

Figure 1-4. Workstation License



Build Environment (2 of 8)

MPLAB X IDE version: 6.00

Build Configuration: default : XC8

MPLAB XC8 Compiler version : 2.32

MPLAB XC Network License Server Address:

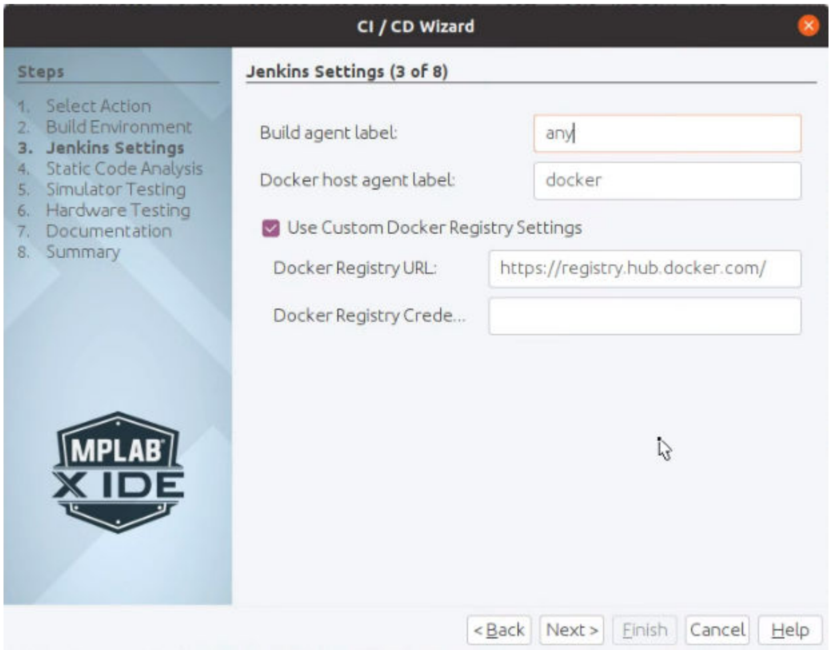
Server Port:

⚠ Network licenses are required to run XC PRO Compiler and Analysis Tool Suite in a Docker environment, i.e., containerized build setup. Failure to get this will run the compiler in free mode which is probably not the desired result.

Option	Description
MPLAB IDE version	Select supported versions from the drop-down list. Default is the latest version.
Build Configuration	Select a project configuration for the build.
MPLAB XC Compiler version	Select a compiler version for the build.
MPLAB XC Network License Server Address	Enter the address of the server. For example: licenses.microchip.com.
Server Port	Enter the port number of the server. For example: 5053
Informational Note (conditional)	If you have a Workstation license, information on buying a Network Server license will be shown.

1.3.2.3 CI/CD Wizard Dialog 3

Determine the location from which Docker will download the image. Default Docker information is entered at the top. To use a custom docker registry, enter information below.



CI / CD Wizard

Steps

1. Select Action
2. Build Environment
3. **Jenkins Settings**
4. Static Code Analysis
5. Simulator Testing
6. Hardware Testing
7. Documentation
8. Summary

Jenkins Settings (3 of 8)

Build agent label: any

Docker host agent label: docker

☑ Use Custom Docker Registry Settings

Docker Registry URL: https://registry.hub.docker.com/

Docker Registry Crede...

Option	Description
Build agent label	The default is set to “any,” but you may add a specific label. Please ensure your Jenkins agents are labeled accordingly in the Configure System settings on the Jenkins Server.
Docker host agent label	The Docker host agent label is the label that is placed on a Jenkins agent that can support the building and running of Docker containers. Please keep in mind there are standard naming syntax that may need to be followed. Consult the Jenkins documentation for these naming conventions.
Use Custom Docker Registry Settings	Use your own docker registry for the base image to download. For more on this, see How to use your own Registry. The base image will be downloaded from Docker Hub. You can specify a path for a customized base image if needed.
Docker Registry URL	Your registry location.
Docker Registry Credentials ID	Your credentials to push and pull. For details see Using Jenkins - Credentials.

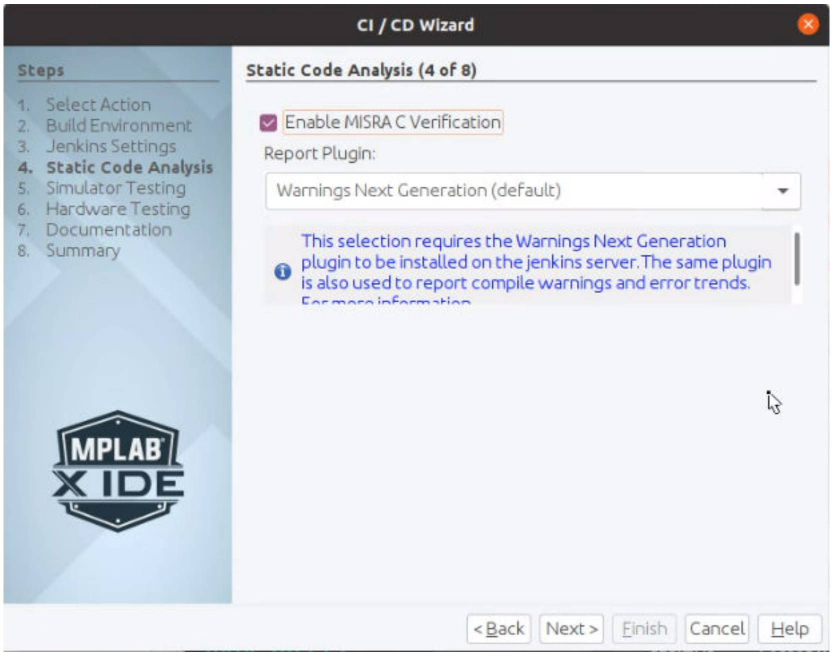
1.3.2.4 CI/CD Wizard Dialog 4

Check to enable MISRA C verification. For more on MISRA, go to www.misra.org.uk/.

Requirements for using this features are:

- A Jenkins plugin is required for either report selection. An informational note will provide a link.
- MISRA C verification is only available for MPLAB X IDE v6.00 and later. An informational note will be displayed if the version does not support this feature.
- An MPLAB Analysis Tool Suite Network Server license is required for tool execution in a generated Docker container.

Figure 1-5. Warnings Next Generation



Static Code Analysis (4 of 8)

☑ Enable MISRA C Verification

Report Plugin:

Warnings Next Generation (default)

This selection requires the Warnings Next Generation plugin to be installed on the jenkins server. The same plugin is also used to report compile warnings and error trends. For more information

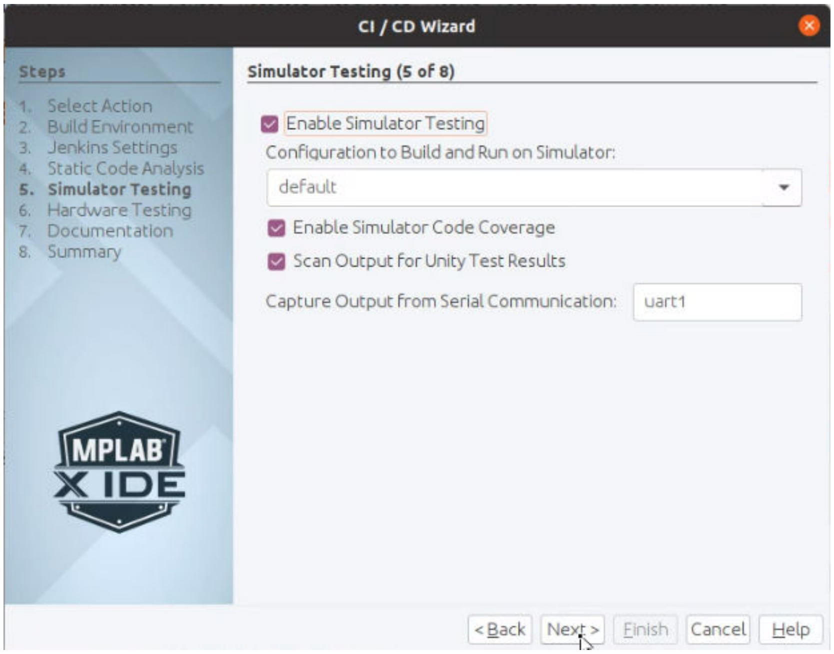
Option	Description
Enable MISRA-C Verification	Check to enable MISRA verification.
Report Plugin	Select from: • Warnings Next Generation • Cppcheck

1.3.2.5 CI/CD Wizard Dialog 5

Check to enable the simulator for use as a test image and specify other options. This can be used to run unit tests.

A Microchip Debugger (MDB) script *mdb-simulator-script.txt* will be added to your project and this is executed during the simulator build step.

The default script will run the simulator for 10 seconds and capture output and coverage based on your settings. To change this, modify the script manually or regenerate using other options. Customize the input file as needed to fit your testing needs. Refer to the Microchip Debugger (MDB) User's Guide.



Simulator Testing (5 of 8)

☑ Enable Simulator Testing

Configuration to Build and Run on Simulator: default

☑ Enable Simulator Code Coverage

☑ Scan Output For Unity Test Results

Capture Output From Serial Communication: uart1

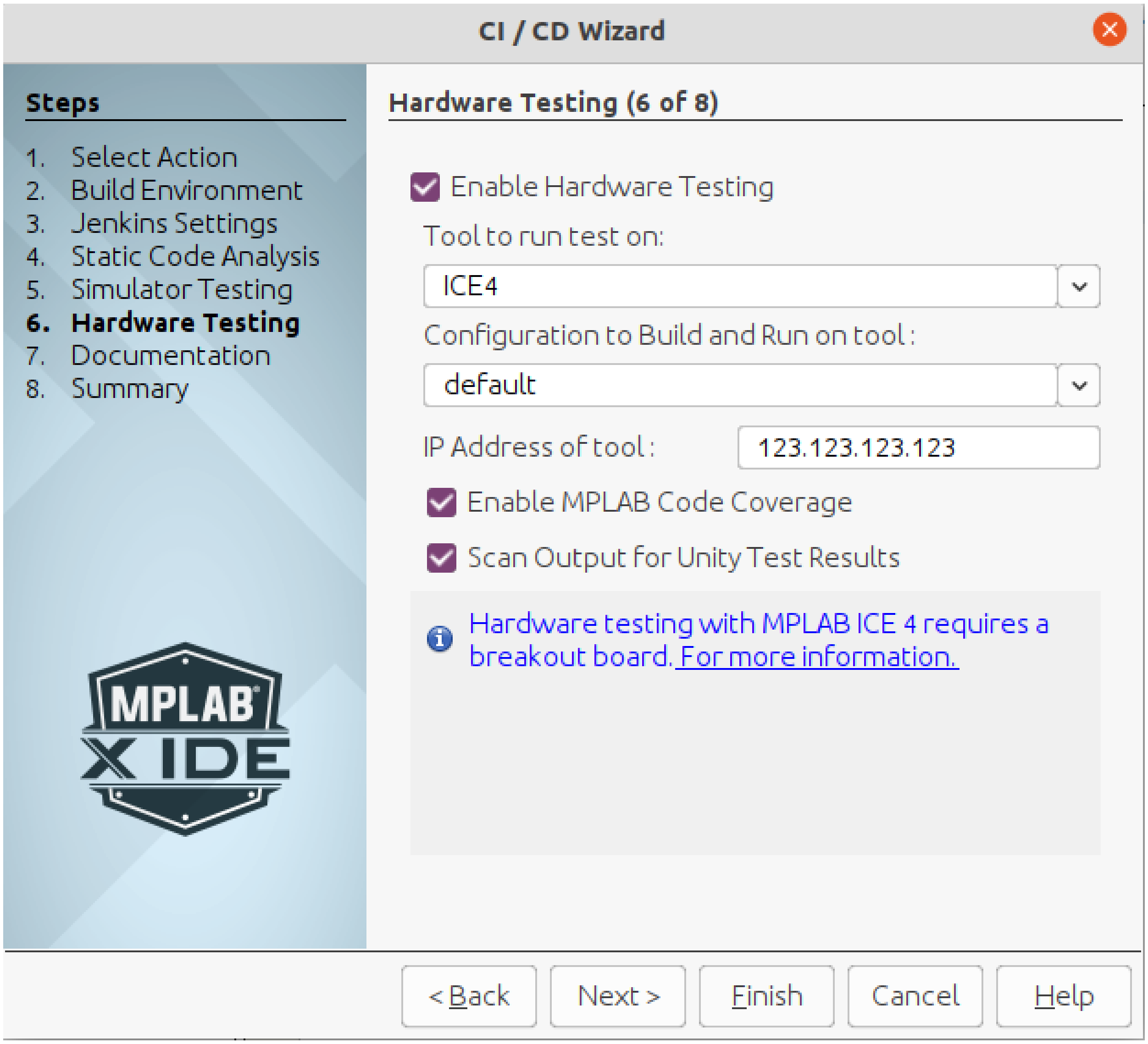
Option	Description
Enable Simulator	Enable the simulator as the debug tool.
Configuration to Build and Run on Simulator	Select a project configuration that uses the simulator; either default or one dedicated to simulator use.
Enable Simulator Code Coverage	Enable Code Coverage feature built into the Simulator. **Note:** The Code Coverage API plugin must be available on your Jenkins server if coverage reporting is enabled.
Scan Output for Unity Test Results	Enable if the configuration builds Unity test runners, and the build job should create a report based on the resulting output. For more info on how to write Unity tests see 1.5. How to Write and Run Unit Tests with Unity.
Capture Output from Serial Communication	This option should be used together with the *Scan output for Unity test results* option. The simulator needs to know from which serial communication module instance to capture the output from. The default value is *uart1*, but this is dependent on which device you are using and how you have configured your test setup.

1.3.2.6 CI/CD Wizard Dialog 6

Check to enable a hardware tool for use as a test image and specify other options. This can be used to run unit tests.

A Microchip Debugger (MDB) script *mdb-hardware-script.txt* will be added to your project and this is executed during the hardware build step.

The default script will run for 10 seconds and capture output and coverage based on your settings. To change this, modify the script manually or regenerate using other options. Customize the input file as needed to fit your testing needs. Refer to the Microchip Debugger (MDB) User's Guide.



CI / CD Wizard

Steps

1. Select Action
2. Build Environment
3. Jenkins Settings
4. Static Code Analysis
5. Simulator Testing
6. **Hardware Testing**
7. Documentation
8. Summary

Hardware Testing (6 of 8)

☑ Enable Hardware Testing

Tool to run test on: ICE4

Configuration to Build and Run on tool : default

IP Address of tool : 123.123.123.123

☑ Enable MPLAB Code Coverage

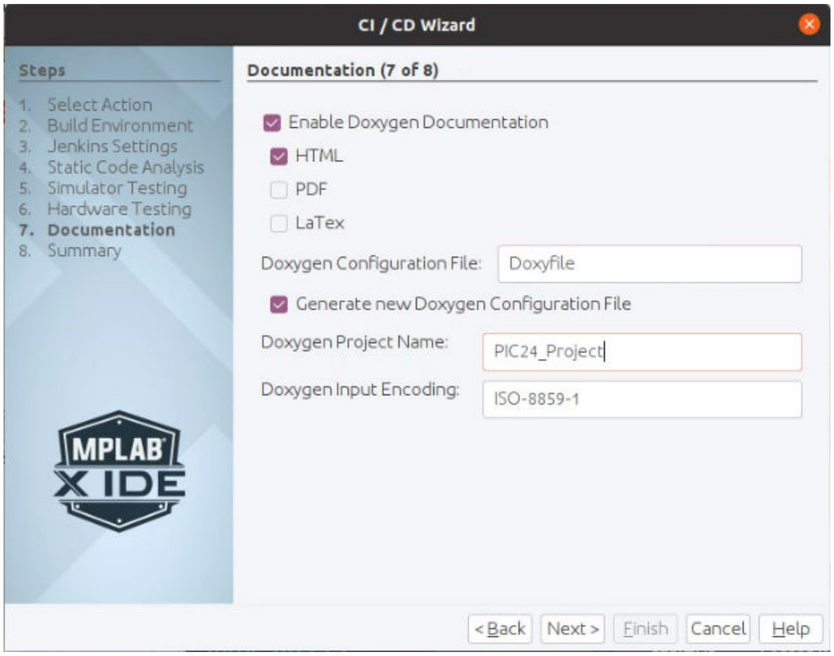
☑ Scan Output For Unity Test Results

Hardware testing with MPLAB ICE 4 requires a breakout board. For more information.

Option	Description
Enable Hardware Testing	Enable hardware selection for testing
Tool to run test on	Enable a debugger/emulator as a test tool.
Configuration to Build and Run on tool	Select a project configuration that uses the test tool; either default or one dedicated to hardware use.
IP Address of tool	Enter the IP address of the tool you wish to use. If you do not know the IP address of the tool, you can open *Tools>Manage Network Tools* and scan for network tools. This will list the tool with its IP address.
Enable MPLAB Code Coverage	Enable the MPLAB Code Coverage feature. An MPLAB Analysis Tool Suite Network Server license is required for tool execution in a generated Docker container. **Note:** The Code Coverage API plugin must be available on your Jenkins server if coverage reporting is enabled.
Scan Output for Unity Test Results	Enable if the configuration builds Unity test runners, and the build job should create a report based on the resulting output. For more info on how to write Unity tests see 1.5. How to Write and Run Unit Tests with Unity.

1.3.2.7 CI/CD Wizard Dialog 7

Check to enable Doxygen as a documentation tool and specify the output. For more info on Doxygen see Doxygen on GitHub.



Documentation (7 of 8)

☑ Enable Doxygen Documentation

☑ HTML

☐ PDF

☐ LaTex

Doxygen Configuration File: Doxyfile

☑ Generate new Doxygen Configuration File

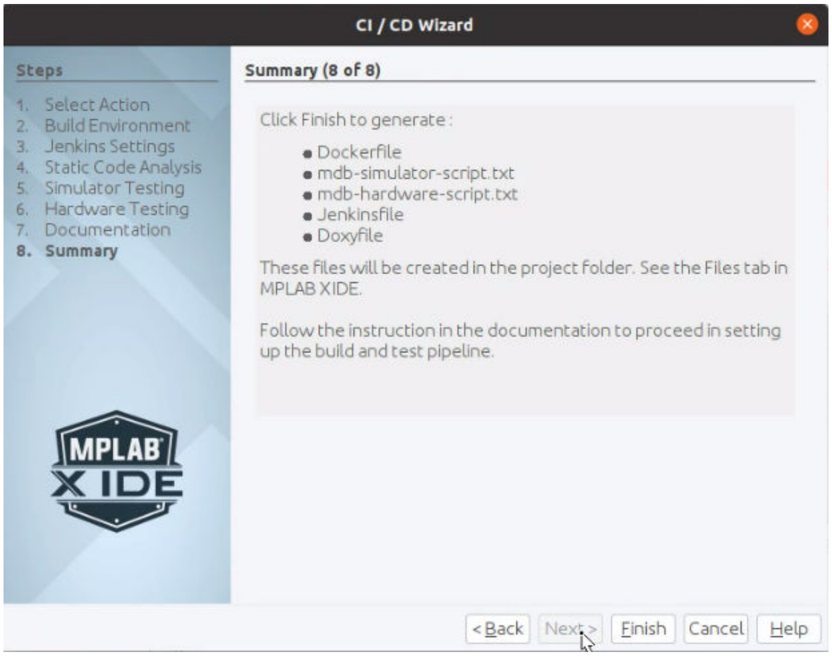
Doxygen Project Name: PIC24_Project

Doxygen Input Encoding: ISO-8859-1

Option	Description
Enable Doxygen Documentation	Enable documentation tool.
Documentation Format	Select one or several output formats.
Doxygen Configuration File	Specify configuration file.
Generate new Doxygen Configuration File	Check to create or overwrite an existing file. Uncheck to keep the existing file.
Doxygen Project Name	Specify project name.
Doxygen Input Encoding	Specify document encoding.

1.3.2.8 CICD Wizard Dialog 8

Review the expected output in the Summary screen. To make changes click **Back**.



Summary (8 of 8)

Click Finish to generate :

- Dockerfile
- mdb-simulator-script.txt
- mdb-hardware-script.txt
- Jenkinsfile
- Doxyfile

These files will be created in the project folder. See the Files tab in MPLAB X IDE.

Follow the instruction in the documentation to proceed in setting up the build and test pipeline.

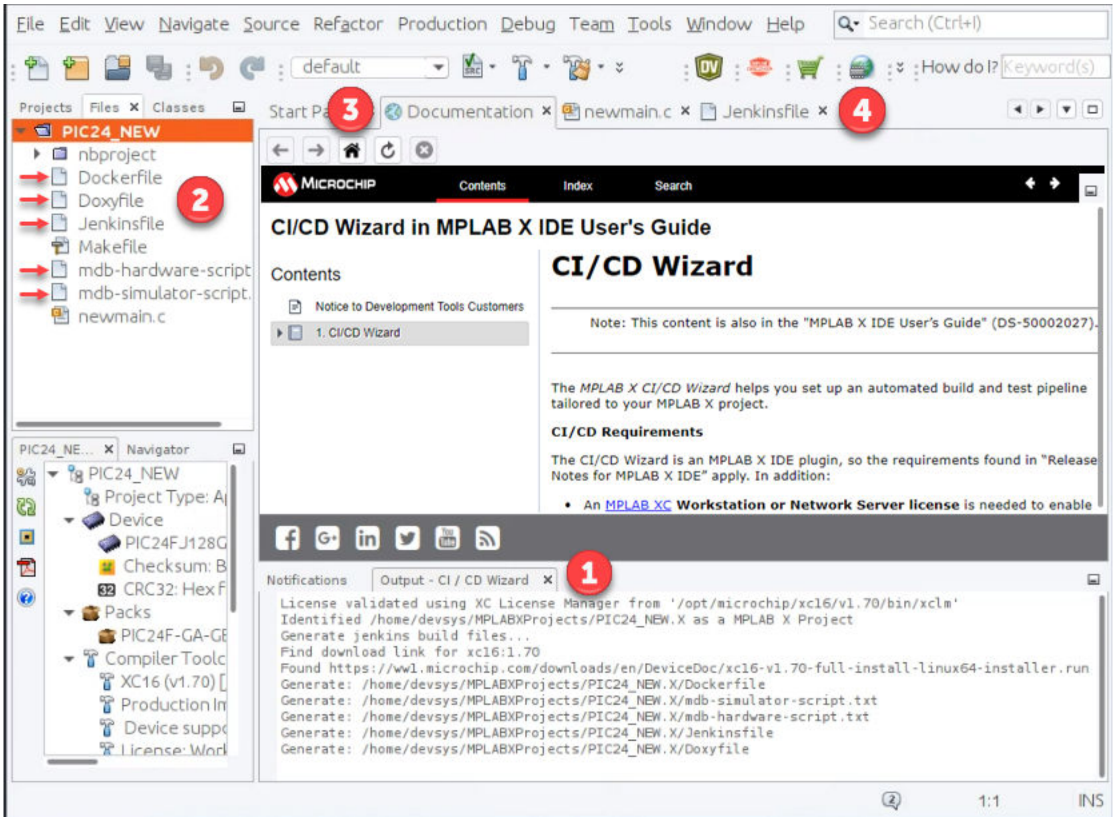
Click **Finish** to complete the setup. CI/CD Wizard will then generate files based on the setup information.

1.3.2.9 CI/CD Wizard Output

In MPLAB X IDE, you can see:

1. Files that are being generated in the Output window.
2. Resulting files added to the project folder.
3. WebHelp in its own window.
4. Jenkinsfile in its own window.

Figure 1-6. File Generation in MPLAB X IDE



CI/CD Wizard in MPLAB X IDE User's Guide

Contents

- Notice to Development Tools Customers
- 1. CI/CD Wizard

CI/CD Wizard

Note: This content is also in the "MPLAB X IDE User's Guide" (DS-50002027).

The *MPLAB X CI/CD Wizard* helps you set up an automated build and test pipeline tailored to your MPLAB X project.

CI/CD Requirements

The CI/CD Wizard is an MPLAB X IDE plugin, so the requirements found in "Release Notes for MPLAB X IDE" apply. In addition:

- An MPLAB XC **Workstation or Network Server license** is needed to enable

```
License validated using XC License Manager from '/opt/microchip/xc16/v1.70/bin/xclm'
Identified /home/devsys/MPLABXProjects/PIC24_NEW.X as a MPLAB X Project
Generate jenkins build files...
Find download link for xc16:1.70
Found https://ww1.microchip.com/downloads/en/DeviceDoc/xc16-v1.70-full-install-linux64-installer.run
Generate: /home/devsys/MPLABXProjects/PIC24_NEW.X/Dockerfile
Generate: /home/devsys/MPLABXProjects/PIC24_NEW.X/mdb-simulator-script.txt
Generate: /home/devsys/MPLABXProjects/PIC24_NEW.X/mdb-hardware-script.txt
Generate: /home/devsys/MPLABXProjects/PIC24_NEW.X/Jenkinsfile
Generate: /home/devsys/MPLABXProjects/PIC24_NEW.X/Doxyfile
```

Details of files generated:

Jenkinsfile

A text file that contains the definition of a Jenkins Pipeline. For more information, see Using a Jenkinsfile.

Dockerfile

A customized text file used to build the container. For more information, see Dockerfile reference.

mdb-simulator-script.txt

This file will be produced only if the simulator was selected in Step 5 of the assistant. It is a file to setup simulator usage in the Microchip Debugger (MDB), the GUI-less form of MPLAB X IDE. You can customize this file to your testing needs. For more information see the Microchip Debugger (MDB) User's Guide.

mdb-hardware-script.txt

This file will be produced only if a hardware tool was selected in Step 6 of the assistant. It is a file to setup hardware tool usage in the Microchip Debugger (MDB), the GUI-less form of MPLAB X IDE. You can customize this file to your testing needs. For more information see the Microchip Debugger (MDB) User's Guide.

Doxygen

This file will be produced only if documentation was selected in Step 7 of the assistant.

Doxygen is a standard tool for generating documentation from annotated source code. For details on the configuration file format, see Configuration.

1.3.3 Step 2 - Add Jenkinsfiles to Your Repository

Use your source control system and add the generated files to your project repository. Jenkins will add the project in this repository to the created container.

1.3.4 Step 3 - Configure Build Job on the Jenkins Server

This section gives a short summary of how to configure a new build job on a Jenkins server using the generated files. For more details see Using a Jenkinsfile.

Short summary:

1. Log in to the Jenkins server web interface and select **New Item**.
2. Enter a build job name and select **Pipeline** as project type.
3. In the new build job configuration, go to the **Pipeline** section and configure:
    a. Select **Pipeline script from SCM** and configure the Source Control Management (SCM) info required for the Jenkins server to check out your project. This may include *repository url*, *credentials* and also *branch specifier*.
    b. Specify *Jenkinsfile* as the **Script Path**. (This needs to match the name of the generated Jenkinsfile).
    c. Save the new build job configuration and start a new build job (**Build Now**).

The result should be a working build pipeline that first sets up a docker build environment before building and analyzing the MPLAB X IDE project.

1.3.5 Suggested Next Steps

The *Jenkinsfile* defines build steps that are run inside a docker image. The docker image is defined by the generated *Dockerfile* and provides a stable Linux environment where the required MPLAB XC toolchain, device support and MPLAB X IDE binaries are available.

You can modify the generated Jenkinsfile by following the Jenkinsfile documentation.

1.3.5.1 Gating

Once you have a stable build job you can implement a gated commit process where each commit is automatically tested before it is rejected or merged to your main branch. Such a process makes sure that the main branch remains stable even when there are many developers working on the same project.

Setting up a gated build process can be done in different ways depending on which source code management infrastructure you are using. Here is an example on how to use the Atlassian Bitbucket integration.

1.4 Getting Started with Jenkins Server

Jenkins is an open source automation server which enables developers to automate build, test, and deploy their software. For more information go to Jenkins official site.

The CI/CD Wizard offers a solution to run an automatically configured Jenkins server instance in a Docker container. The Jenkins server can be configured to build an existing MPLAB X IDE project from a local folder or from git. The server can be a starting point for further configuration and customization, or as a local test bed for the build pipelines generated by the wizard.

1.4.1 Prerequisites

Docker

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files.

- Read more on docker.com

- Get started with Docker

1.4.2 Create and Start a Jenkins Server

To create and start a Jenkins server, `cicdw`, the command line interface for the CI/CD Wizard, will be used. For more information on this tool, see 1.7. CICDW Command Line Tool.

1.4.2.1 Create a “Plain” Jenkins Server

To create a plain Jenkins server without any preconfigured build jobs:

```
cicdw jenkins-server create
```

1.4.2.2 Create a Jenkins Server with a Git-Based Build Job

To create a Jenkins server preconfigured with a build job to build against your git repository:

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

This will extract the git server information associated with the git remote named "origin" for your local git repository. For this to work, the project path (default is the current folder) must point to a valid MPLAB X IDE folder that contains a Jenkinsfile and is checked into a remote git repository.

Note: If you don't already have a Jenkinsfile associated with your project, use the CI/CD Wizard to generate the files for you. See 1.3. Getting Started with Jenkins.

1.4.2.3 Create a Jenkins Server with a Local Build Job

To create a Jenkins server preconfigured with a build job to build against your local folder:

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

When triggered, the build job will look for a Jenkinsfile on a local folder on your computer. Use the [`path`] setting when starting the server to provide the local folder path (this is not a production server mode).

Note: If you don't already have a Jenkinsfile associated with your project, use the CI/CD Wizard to generate the files for you. See 1.3. Getting Started with Jenkins.

1.4.2.4 Start the Jenkins Server

Once a Jenkins server is created, it can be started by entering the following command:

```
cicdw jenkins-server start
```

Note that a server configured with a local build job will use the project path (default is the current folder) given to the start command as its local folder.

Once started, the Jenkins server can be accessed through a browser.

The default location is **http://localhost:8080/** - this can be changed using the `--server-name` and the `--server-port` parameters.

A default user is created with the credentials **mplabcicd/mplabcicd** - this can be changed using the `--admin-user` and the `--admin-password` parameters.

1.4.3 Credentials

When interacting with external resources like a git repository, various credentials will often be required to authenticate and authorize the build system. Jenkins credentials are handled by the credentials manager, and can be provided when a new server is created.

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

There are two types of credentials, `usr_pwd` or `sshkey`.

usr_pwd is the default, so the parameter --credentials my_id::my_username::my_password will be interpreted as TYPE=usr_pwd, ID=my_id, USERNAME=my_username, PASSWORD=my_password.

For the type sshkey, the fourth element will be interpreted as the path to a keyfile if only four elements are provided. With five elements, the fourth will be interpreted as a password for the keyfile, and the fifth as the path to the keyfile.

One or more credential parameters can be listed after the --credentials flag.

1.4.3.1 Examples

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

or

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

will create a username/password credential with the username theuser and the password secret123 as the credential id httpuser.

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

will create a ssh credential with the username theuser and the keyfile key from the folder path/to/.ssh/ as the credential id sshuser.

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

will be the same as the previous example, but will now use the passphrase keyfilepwd to unlock the keyfile.

1.4.3.2 Use Credentials with a Build Job

A git based build job will often require credentials. To use the credentials created by the --credentials parameter, add the --git-job-credentials-id parameter. --gjci for short.

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

Note that username/password authentication will not work for any ssh based remote url.

- 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 Edit the Jenkins Server Build Files

By default, the cicdw jenkins-server create command will write its configuration files to a temporary folder. By including the --build-path parameter, this can be changed to a more permanent location. Add the --only-generate flag to generate build files without creating a docker image.

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

This will generate three files inside that folder.

- Dockerfile - An instruction on how to build the docker image for the Jenkins server
- plugins.txt - A list of plugins to preinstall in the Jenkins server
- casc.yaml - Configuration instructions for the Jenkins server

Refer to the official JCasC documentation for information on how to edit these files.

To use the edited files, add the `--from-build-files` flag to the `cicdw jenkins-server create` command. Note that you will have to remove any previously created server before you can create a new one.

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

1.5 How to Write and Run Unit Tests with Unity

Unity is an MIT Licensed framework that makes it easy to write unit tests. The official getting started guide is available here: Unity - Getting Started.

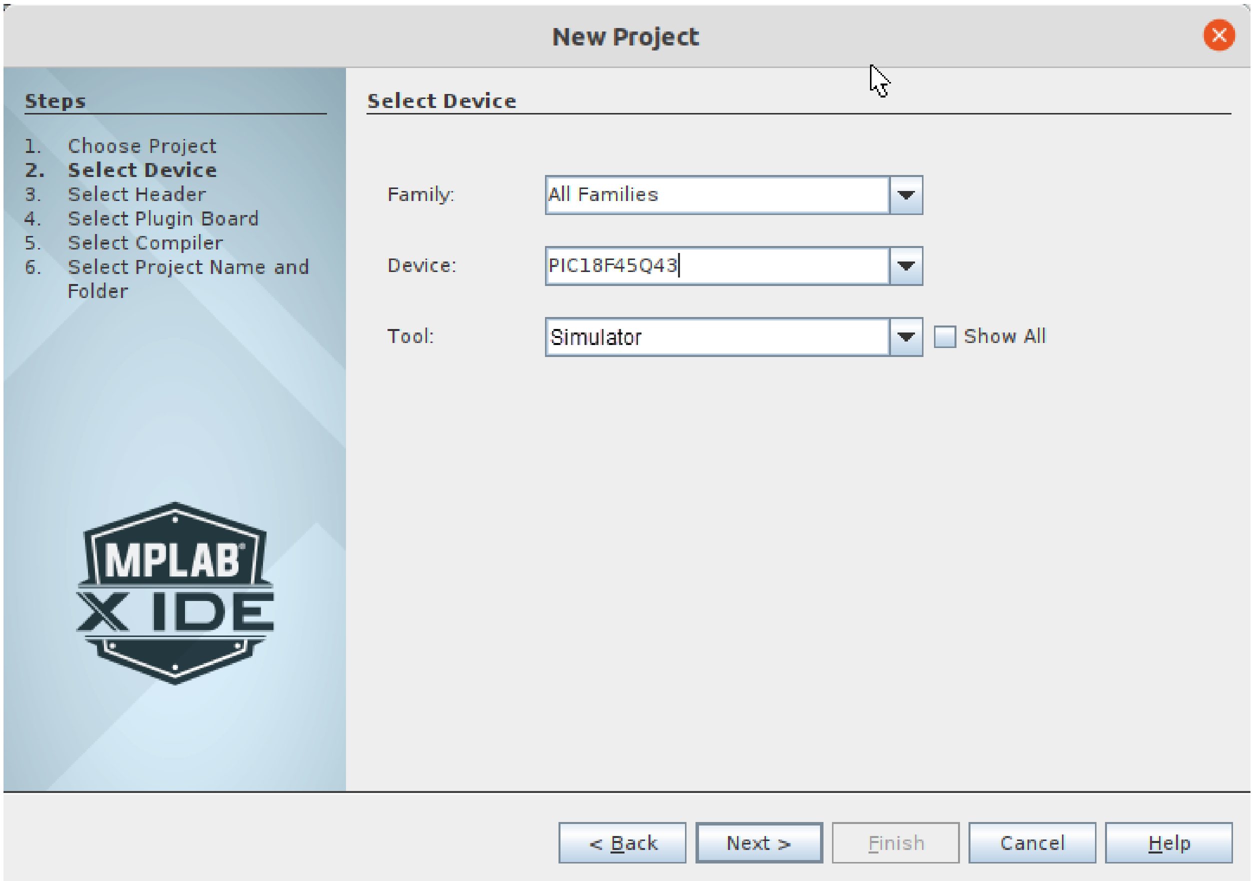
Examples of how to use this product in the MPLAB X IDE context are shown in the following sections.

1.5.1 Running Unit Tests on the Simulator

Follow the steps below for an example of how to use Unity with the MPLAB X Simulator.

1.5.1.1 Step 1. Create New Project

Create a new Standalone Project in MPLAB X IDE, which includes selecting the device and toolchain.



New Project

Steps

1. Choose Project
2. **Select Device**
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder

Select Device

Family: All Families

Device: PIC18F45Q43

Tool: Simulator

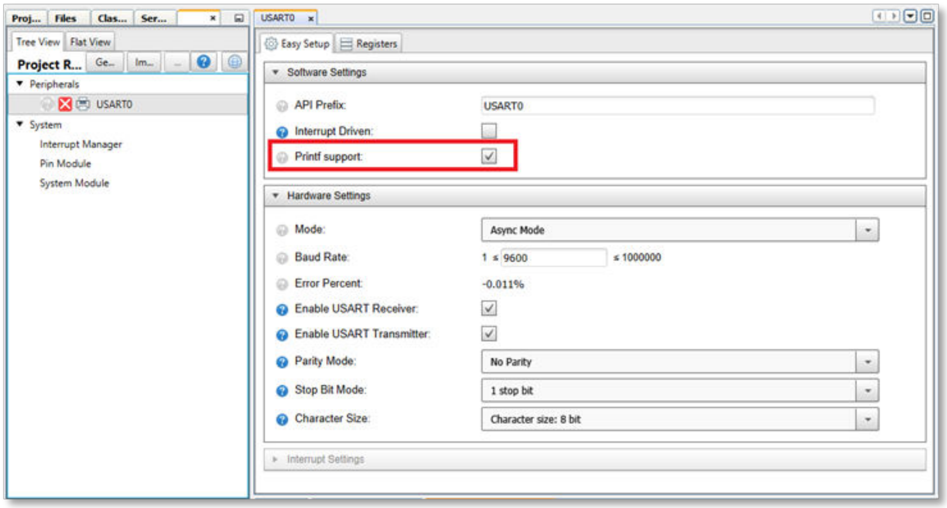
□ Show All

1.5.1.2 Step 2. Configure Serial Output

Unity requires support for *printf* to output test results. This can be done in several ways.

One approach is to install and use the MPLAB X IDE plugin *MPLAB Code Configurator* (MCC). Once installed, open MCC and add an USART (or UART) from the *Device Resources* view. Make sure to enable *Printf* support as part of the configuration.

The following example shows a configured USART for the *ATtiny817* device with *Printf* in MCC:



Software Settings

API Prefix: USART0

Interrupt Driven: ☐

Printf support: ☑

Hardware Settings

Mode: Async Mode

Baud Rate: 1 ≤ 9600 ≤ 1000000

Error Percent: -0.011%

Enable USART Receiver: ☑

Enable USART Transmitter: ☑

Parity Mode: No Parity

Stop Bit Mode: 1 stop bit

Character Size: Character size: 8 bit

Interrupt Settings

When you have configured the USART (or UART) in MCC remember to click “Generate” and verify that the configured files are added to your project.

1.5.1.3 Step 3. Verify Serial Output with MDB Script and the Simulator

You should verify that your serial output and `printf` works as expected before continuing with Unity.

Step 3.A – Test printf from main.c

The setup will differ from selected device and how `printf` was setup.

The following `main.c` example assumes that the device is ATtiny817 and that a configured USART is generated by MCC in the `mcc_generated_files` sub folder:

```
#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 – Create Simulator MDB Script

Create an MDB script that runs your program locally on the simulator and logs your serial communication to a local file.

An example of `mdb.script` using ATTiny817 is:

Note: You will need to modify the path to your elf file.

```
# 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 – Run MDB Script and Verify Output

Make sure that your project is built, then open a command line and navigate to your project. Run `mdb.sh` or `mdb.bat` with your mdb script as argument.

Here is an example:

```
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
```

If you now open `sercom_output.txt` you should get the output:

```
Hello world!
```

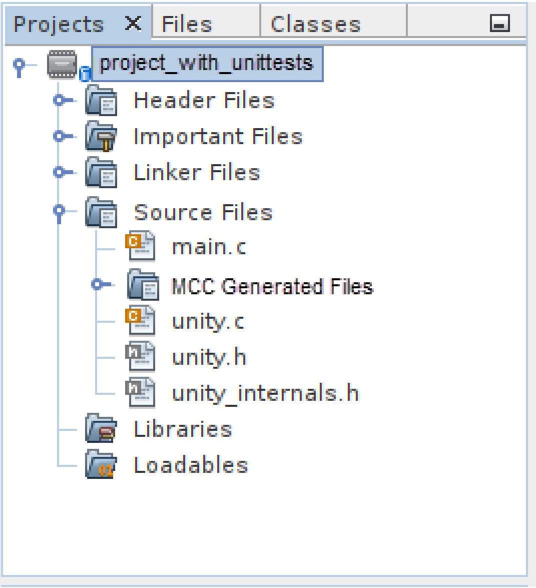
1.5.1.4 Step 4. Add Unity Test Framework

Now that you have a working serial output, you can add unit tests.

You need to manually download and add the Unity test framework to your project. The files you need are available from https://github.com/ThrowTheSwitch/Unity/tree/master/src.

Download them and add them to your project.

Your project should now look something like this:



1.5.1.5 Step 5. Write a Testrunner

Rewrite your `main.c` and add tests to it. For information on how to write more advanced tests see: https://github.com/ThrowTheSwitch/Unity/blob/master/docs/UnityGettingStartedGuide.md.

For example:

```
#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. Run Tests on the Simulator

Rerun the `mdb` script that you have created (as described in Step 3.C).

This time the output should be:

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

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

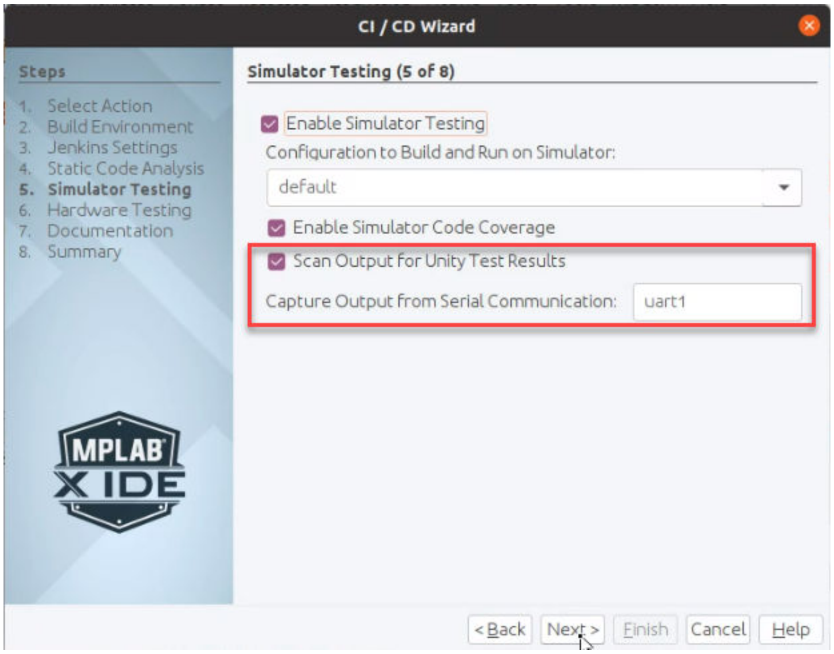
The suggested `mdb` script only waits for 10 seconds before exiting. If your tests require more time, you can increase this in the `wait` command in `mdb.script`.

1.5.1.7 Step 7. Use the Simulator with CI/CD Wizard

When you have a stable setup that works locally, run the CI/CD Wizard and integrate your tests.

Enable the simulator test step in the CI/CD Wizard when generating Jenkinsfiles.

Enable “Scan Output for Unity Test Results” and fill in the name of the serial communication instance to capture output from:



CI / CD Wizard

Simulator Testing (5 of 8)

☑ Enable Simulator Testing

Configuration to Build and Run on Simulator: default

☑ Enable Simulator Code Coverage

☑ Scan Output for Unity Test Results

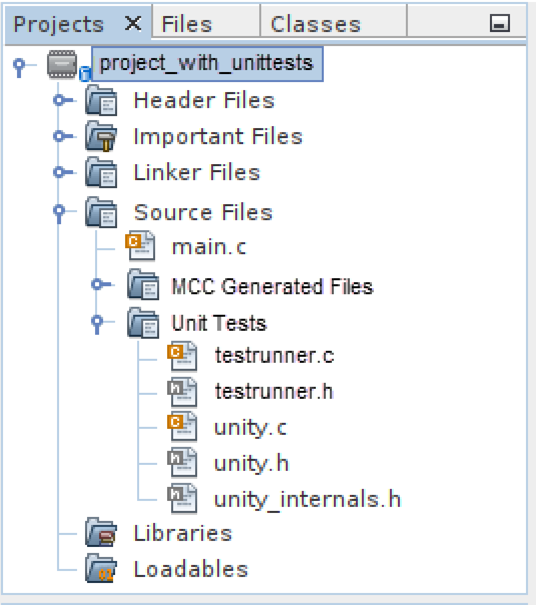
Capture Output From Serial Communication: uart1

The wizard will generate its own `mdb` script and use this during Jeknins builds. Test results will be parsed from the output will be presented by Jenkins after builds.

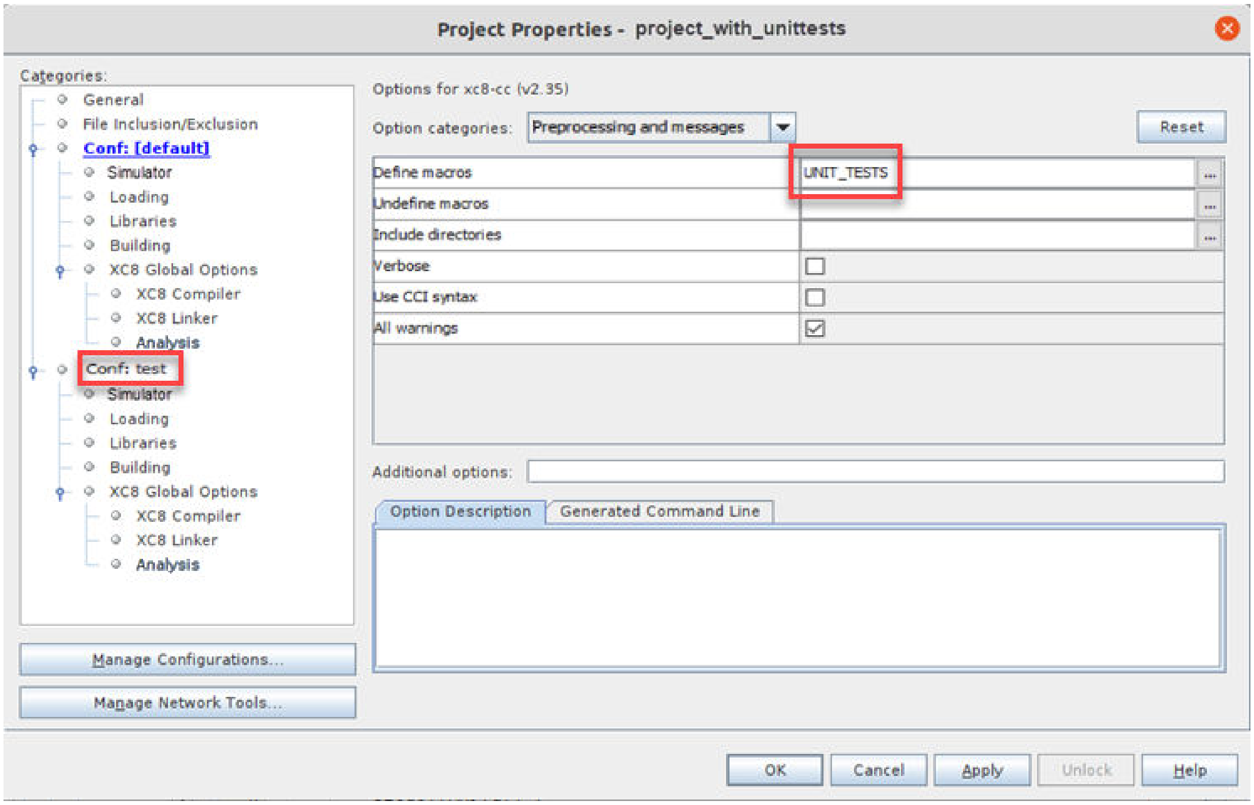
1.5.1.8 Optional Steps

Separate Test and Production Code

To better organize your project, you should move unit support files and testrunners into a separate subfolder:



You can also add a separate test configuration to your project:



Options for xc8-cc (v2.35)

Option categories: Preprocessing and messages

Define macros	UNIT_TESTS
Undefine macros	
Include directories	
Verbose	☐
Use CCI syntax	☐
All warnings	☑

Additional options:

Because you have a test configuration that defines UNIT_TESTS (see picture above) and you moved test running to a separate subfolder, you can modify your `main.c` to something like this:

```
#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;
}
```

This way you can have one common project but one configuration that optimizes the build for production, and one test configuration that runs the test.

Add Code Coverage

If you want to enable code coverage, you can add the following lines to your `mdb.script` file before the `program` command:

```
# 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
```

This will write coverage data to `simulator_coverage.txt` while running the `mdb` script.

1.5.2 Running Unit Tests on MPLAB ICE 4

Follow the steps below for an example of how to use Unity with the hardware tool MPLAB ICE 4 In-Circuit Emulator.

1.5.2.1 Step 1. Setup a Project with Tests and Verify on Simulator

It is advised to make your tests work on the simulator before continuing with MPLAB ICE 4 in-circuit emulator. Therefore, begin by following the steps described in 1.5.1. Running Unit Tests on the Simulator. Make sure you have a project with test runner and a serial driver with `printf` support.

Once your tests are working on the simulator you can continue with the MPLAB ICE 4. For the serial output to be captured by the emulator, the communication settings in the table below need to be used.

Serial driver setting	Required values
Baud rate	115200
Transmission bits	8
Reception bits	8
Data parity	No parity

1.5.2.2 Step 2. Connect MPLAB ICE 4 to a Target

Connect the MPLAB ICE 4 In-Circuit Emulator (DV244140) to your target hardware using one of the following methods:

1. Connect MPLAB ICE 4 40-pin ribbon cable to the MPLAB ICE 4 Breakout Board (AC244141). Then connect the Breakout Board pins to the corresponding target ICSP modular connector pins as specified in the table below.

2. Connect MPLAB ICE 4 40-pin ribbon cable directly to a 40-pin connector on the target. Ensure that the 40-pin connector pins map to the target device ICSP pins as specified in the table below.

The ICSP pins are required to debug and program the device, and the DGI_VCP_TXD and DGI_VCP_RXD pins are required to read the serial output. The DGI_VCP_TXD and DGI_VCP_RXD need to be connected to the pins which you have configured for your serial driver to use.

Table 1-1. Connection Summary

ICE 4 Function Names[1,2]	ICE 4 Breakout Board Pins[1]	ICE 4 Connector Pins at Target[2]	Target ICSP Names[3]	Target ICSP Pins[3]
TVPP_IO	J4/5 19	37	$\overline{\text{MCLR}}$/VPP	1
TVDD_PWR	J2/3/4/5 20	39,40	VDD	2
GND	J2/3 6,8,10,12,14,16	12,16,20,24,28,32	VSS	3
TPGD_IO	J2/3 19	38	TGO/PGD	4
TPGC_IO	J4/5 18	35	TCK/PGC	5
TAUX_IO	J2/3 18	36	TAUX	6
DGI_VCP_TXD	J4/5 13	25	TMS[4]	7 - TX pin used by configured serial driver
DGI_VCP_RXD	J4/5 12	23	TDI[4]	8 - RX pin used by configured serial driver

1. Refer to the *MPLAB ICE 4 Breakout Board User's Guide* (DS-50003479)
2. Refer to the *MPLAB ICE 4 In-Circuit Emulator User's Guide* (DS-50003242), 3.3.1 "Target Connection Pinout" table.
3. Refer to the *MPLAB ICE 4 In-Circuit Emulator User's Guide*, 3.3.5.1 "ICSP Target Connection" figures.
4. The MPLAB ICE 4 ICSP Adapter Board connects TMS and TDI to ICE 4 Connector Pins 34 and 33 respectively. Therefore this board cannot be used for CI/CD.

Figure 1-7. Hardware Connections using MPLAB ICE 4 Breakout Board

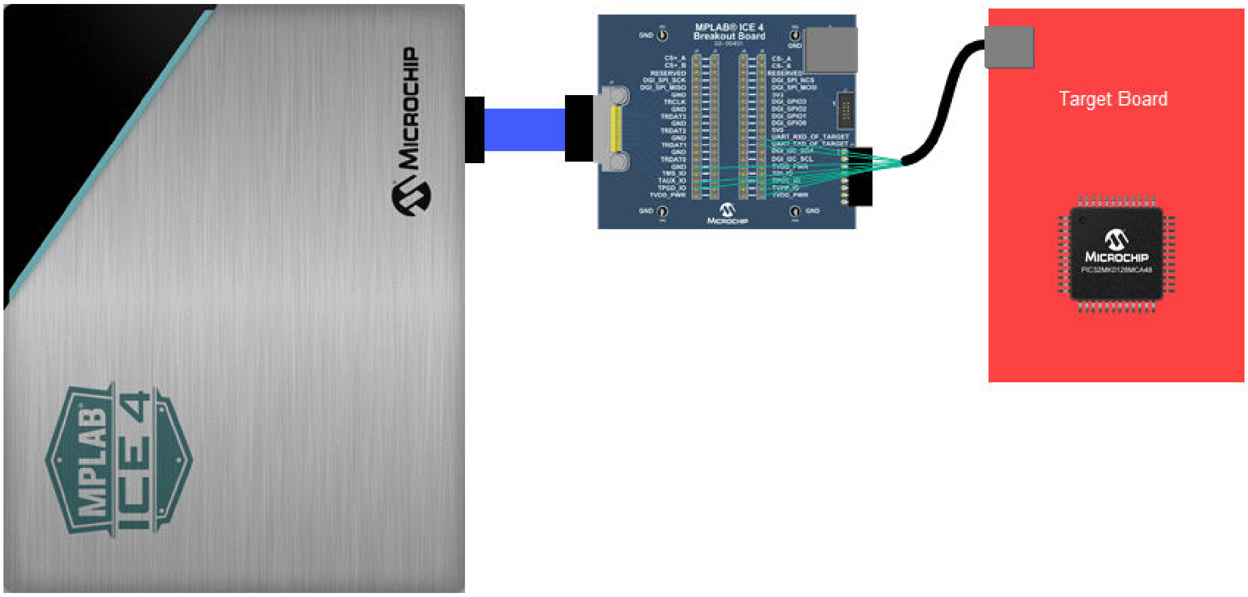
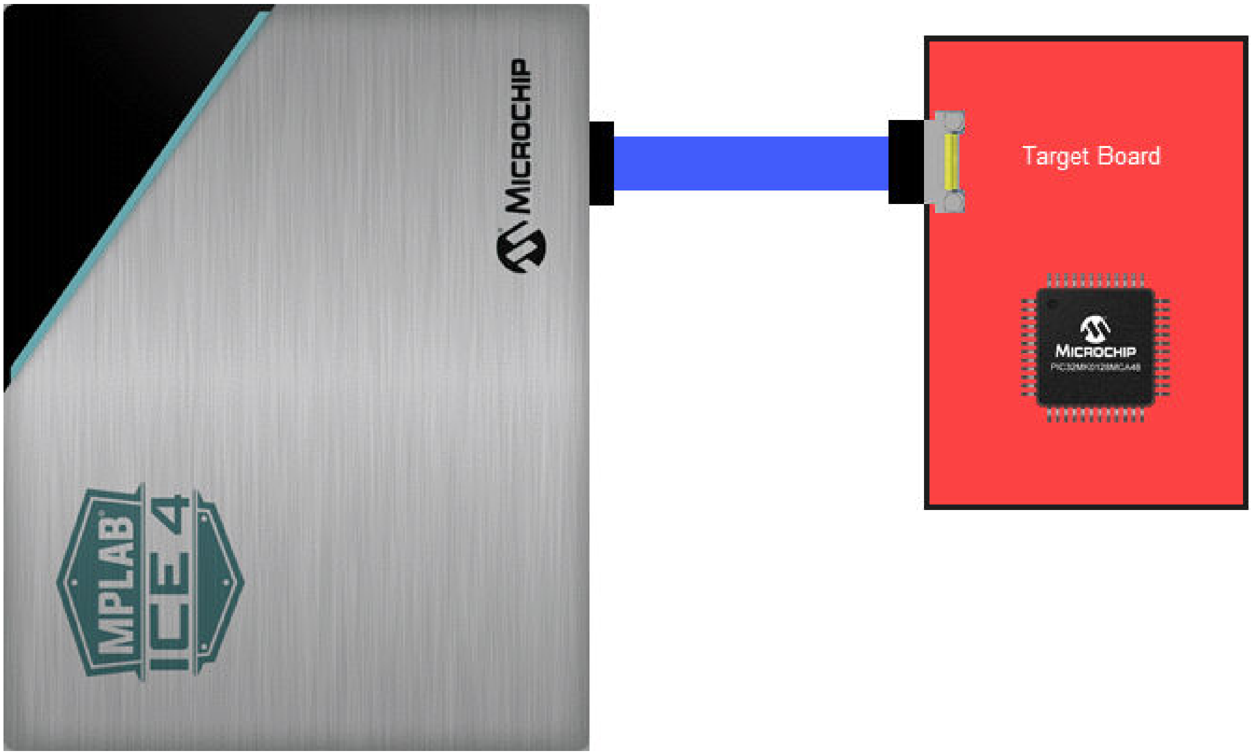


Figure 1-8. Hardware Connections directly from the ICE 4 40-pin Cable



1.5.2.3 Step 3. Verify Connection Using Local ICE 4

Connect the MPLAB ICE 4 to your computer using USB. Open and use MPLAB X IDE to verify that you can program and run your test application on the target using the emulator and the ICE 4 breakout board.

If this fails, you need to review your breakout board connections.

Note: You will not be able to get the serial output yet.

1.5.2.4 Step 4. Configure MPLAB ICE 4 as Network Tool

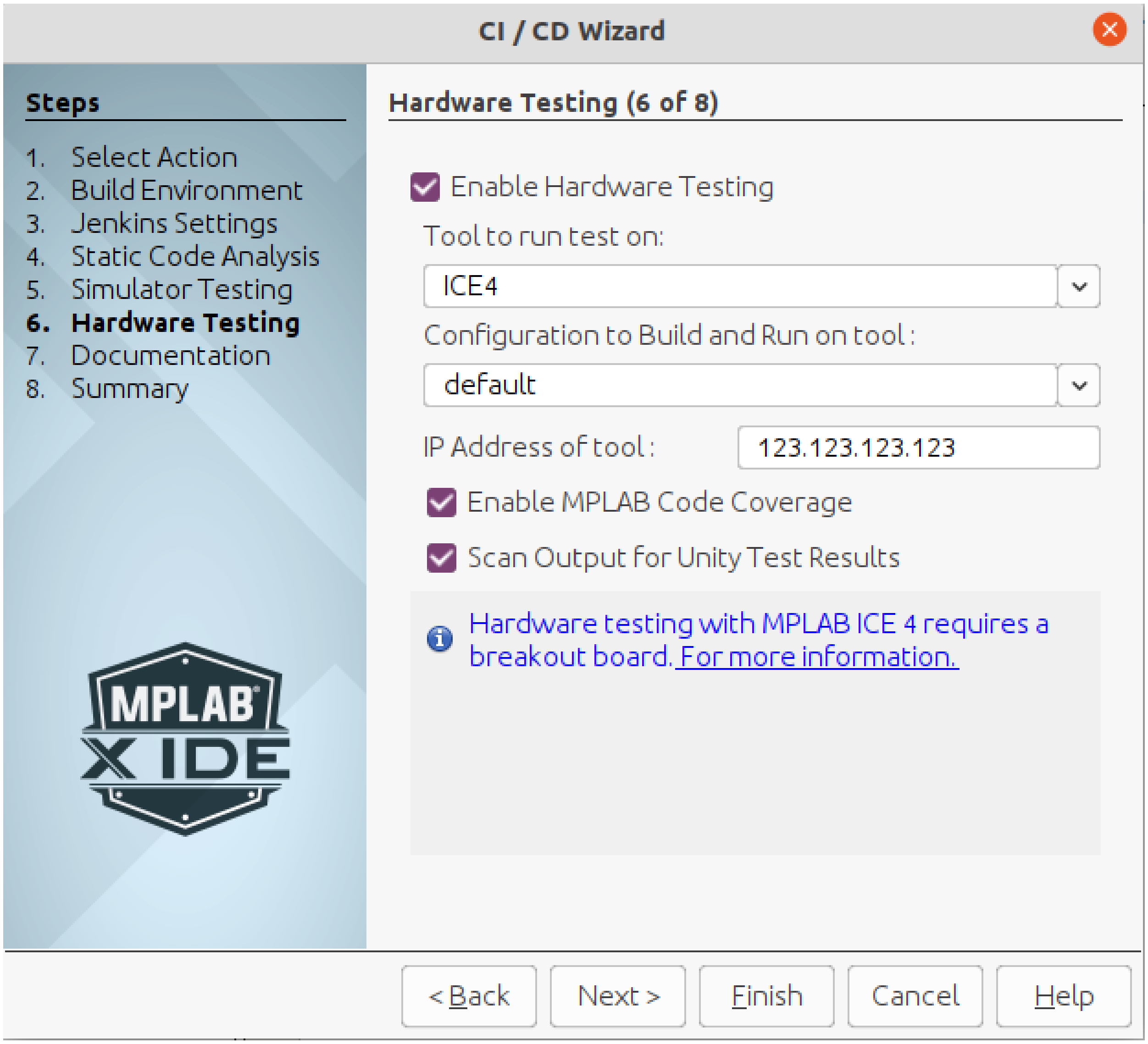
In MPLAB X IDE open *Tools>Manage Network Tools* and configure your connected MPLAB ICE 4 to use Ethernet instead of USB. The MPLAB ICE 4 needs to be connected to an Ethernet cable and you need to detach the USB cable and toggle the MPLAB ICE 4 power afterward. See the "*MPLAB ICE 4 In-Circuit Emulator User's Guide*" for details.

Once configured, the MPLAB ICE 4 should be discovered from the Manage Network Tools dialog when scanning for network tools and you should take a note of the IP address. To verify the Ethernet setup, you can now use the MPLAB ICE 4 as a remote tool in MPLAB X IDE to program and run your test application.

Note: If your network does not provide static IP addresses, you will need to repeat this step periodically.

1.5.2.5 Step 5. Add MPLAB ICE 4 in CI/CD Pipeline

From the CI/CD Wizard you should now be ready to enable the MPLAB ICE 4 hardware testing as part of the Jenkins build pipeline generation.



CI / CD Wizard

Steps

1. Select Action
2. Build Environment
3. Jenkins Settings
4. Static Code Analysis
5. Simulator Testing
6. **Hardware Testing**
7. Documentation
8. Summary

Hardware Testing (6 of 8)

☑ Enable Hardware Testing

Tool to run test on: ICE4

Configuration to Build and Run on tool : default

IP Address of tool : 123.123.123.123

☑ Enable MPLAB Code Coverage

☑ Scan Output For Unity Test Results

Hardware testing with MPLAB ICE 4 requires a breakout board. For more information.

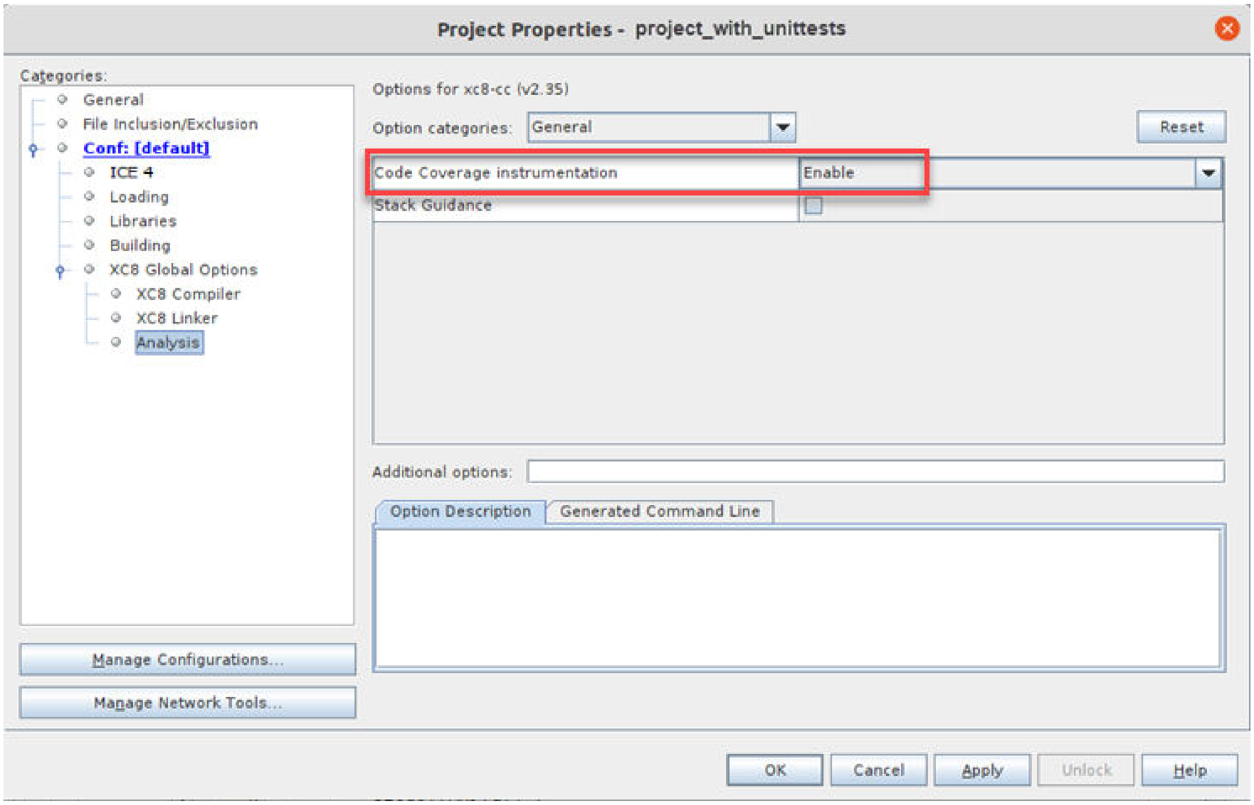
Select the build configuration you want to test.

Fill in the IP address that you noted when configuring the MPLAB ICE 4 on Ethernet.

Enable *MPLAB Code Coverage* if you want coverage data. But note that this requires that your project is set up to instrument your build with code coverage. To do this you need to open project properties and configure the Code Coverage instrumentation in the Analysis section (see figure below).

Enable *Scan Output for Unity Test Results* which will parse the output from your serial port and generate a test report in the Jenkins build job.

Figure 1-9. Enable Code Coverage



Options for xc8-cc (v2.35)

Option categories: General

Code Coverage instrumentation: Enable

Stack Guidance: □

When everything is configured, the generated Jenkins build job will do the following when run:

- Build your project using the selected build configuration
- Communicate with the MPLAB ICE 4 and capture serial output
- Run your test application using a generated MDB script
- When done, parse the serial output and display test results
- Publish coverage data

1.5.2.6 Step 6. Update MPLAB ICE 4 IP address (optional)

If your MPLAB ICE 4 is later moved or your network uses dynamic IP addresses, you might need to update your build scripts with the new IP address.

In MPLAB X IDE select *Tools>Manage Network Tools* and scan for tools. Find the MPLAB ICE 4 and take a note of the new IP address.

In the *Jenkinsfile* update the *HARDWARE_TEST_TOOL_IP* field:

```
HARDWARE_TEST_TOOL_IP = ‘123.123.12.34’
```

In *mdb-hardware-script.txt* update the *addip* and *hwtool* commands:

```
# 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: The mdb scripts use a zero padded format, so use *123.123.012.034* instead of *123.123.12.34*

1.5.3 Running Unit Tests on MPLAB ICD 5

How to use Unity with the hardware tool MPLAB ICD 5 In-Circuit Debugger is very similar to using Unity with the MPLAB ICE 4 In-Circuit Emulator. Therefore follow the steps under 1.5.2. Running Unit Tests on MPLAB ICE 4 except for those listed below.

1.5.3.1 Step 2. Connect MPLAB ICD 5 to Target

Connect the MPLAB ICD 5 In-Circuit Debugger (DV164055) to your target hardware using one of the following methods:

1. Connect MPLAB ICD 5 via an 8-pin RJ-45 cable to the Debugger Adapter Board (AC102015). Then connect the Adapter Board pins to the corresponding target ICSP modular connector pins as specified in the table below.
2. Connect MPLAB ICD 5 via an 8-pin RJ-45 cable directly to an 8-pin connector on the target. Ensure that the connector pins map to the target device ICSP pins as specified in the table below.

The ICSP pins are required to debug and program the device, and TX and RX pins are required to read the serial output. The TX and RX need to be connected to the pins which you have configured for your serial driver to use.

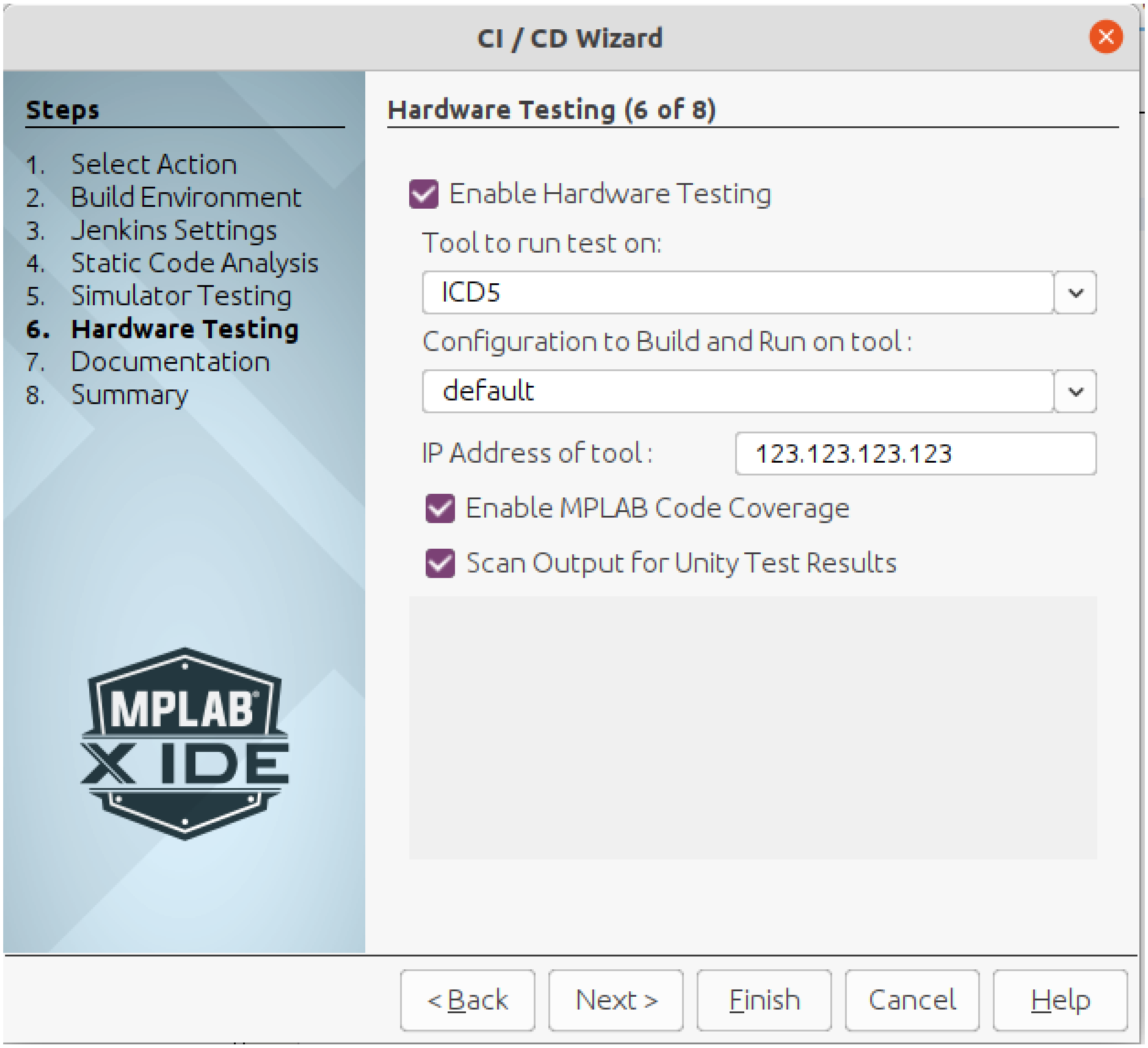
Table 1-2. Connection Summary

ICD 5 Function Names[1]	ICD 5 Connector Pins at Target[1]	Target ICSP Names[1]	Target ICSP Pins[1]
TX	8	TX [2]	1
TVPP	7	$\overline{\text{MCLR}}$/VPP	2
TVDD	6	VDD	3
GND	5	GND	4
PGD	4	DAT	5
PGC	3	CLK	6
RX[3]	2	RX[2,3]	7
RX	1	RX[2]	8

1. Refer to the *MPLAB ICD 5 In-Circuit Debugger User's Guide*, 3.3.1 "Target Connection Pinout" tables.
2. Pin used by configured serial driver.
3. For SAM MCU devices, RX is on a different pin.

1.5.3.2 Step 5. Add MPLAB ICD 5 in CI/CD Pipeline

From the CI/CD Wizard you should now be ready to enable the MPLAB ICD 5 hardware testing as part of the Jenkins build pipeline generation.



CI / CD Wizard

Steps

1. Select Action
2. Build Environment
3. Jenkins Settings
4. Static Code Analysis
5. Simulator Testing
6. **Hardware Testing**
7. Documentation
8. Summary

Hardware Testing (6 of 8)

☑ Enable Hardware Testing

Tool to run test on: ICD5

Configuration to Build and Run on tool : default

IP Address of tool : 123.123.123.123

☑ Enable MPLAB Code Coverage

☑ Scan Output For Unity Test Results

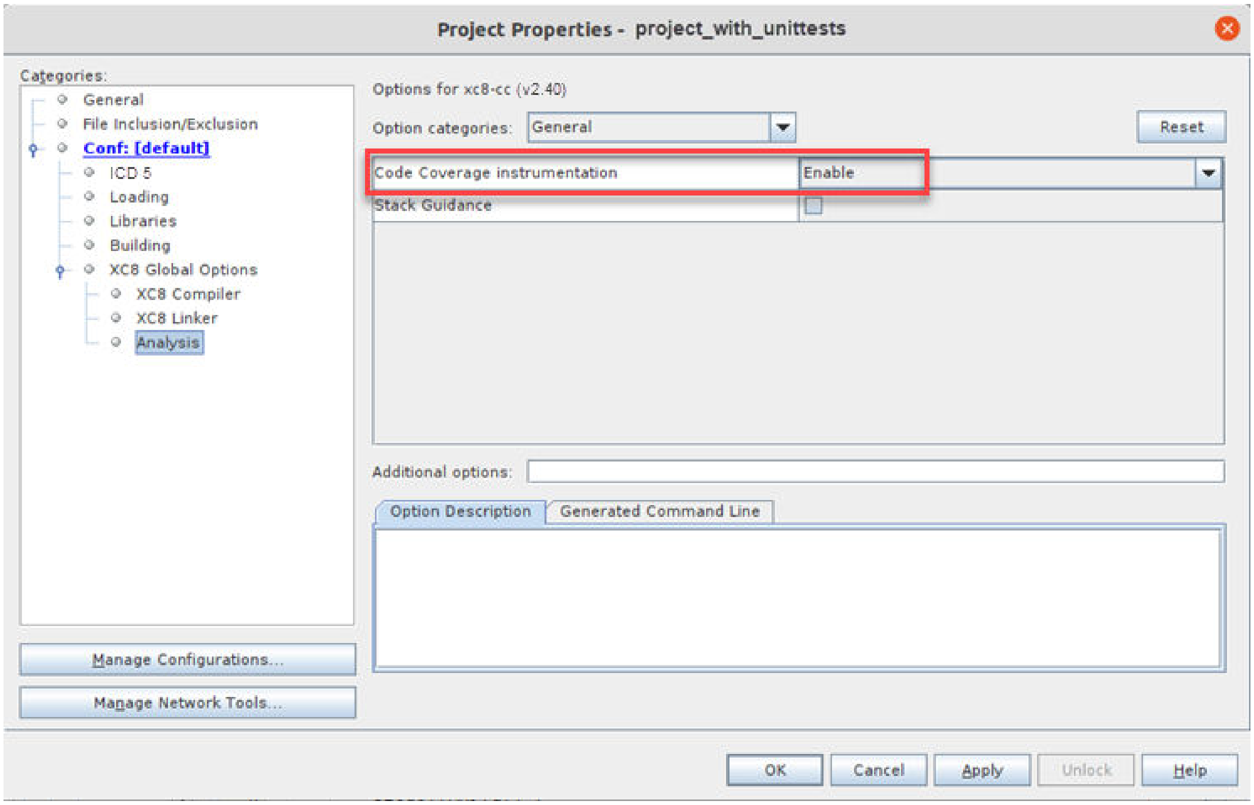
Select the build configuration you want to test.

Fill in the IP address that you noted when configuring the MPLAB ICD 5 on Ethernet.

Enable *MPLAB Code Coverage* if you want coverage data. But note that this requires that your project is set up to instrument your build with code coverage. To do this you need to open project properties and configure the Code Coverage instrumentation in the Analysis section (see figure below).

Enable *Scan Output for Unity Test Results* which will parse the output from your serial port and generate a test report in the Jenkins build job.

Figure 1-10. Enable Code Coverage



Options for xc8-cc (v2.40)

Option categories: General

Code Coverage instrumentation	Enable
Stack Guidance	□

When everything is configured, the generated Jenkins build job will do the following when run:

- Build your project using the selected build configuration
- Communicate with the MPLAB ICD 5 and capture serial output
- Run your test application using a generated MDB script
- When done, parse the serial output and display test results
- Publish coverage data

1.6 Getting Started with Dockerfile

Docker is an open platform for developing, shipping, and running applications. Docker provides OS-level virtualization which enables you to separate your applications from your infrastructure.

Docker will read the instructions from the Dockerfile and build a Docker image. A Docker image is like a template. The Docker image is used to build and run a Docker container. To do all this you must have Docker installed.

For more information about how to install Docker and use Dockerfiles, see www.docker.com.

1.6.1 Prerequisites

1.6.1.1 License Server

- To use the MPLAB XC PRO compilers in a setup with Docker containers, an MPLAB XC network license server must be accessible from the build node. MPLAB XC License Server Manual (DS50002334) describes how to set up this server.
- Other features may need a license server to verify their licenses, such as the MPLAB Analysis Tool Suite (for MISRA check and MPLAB Code Coverage).

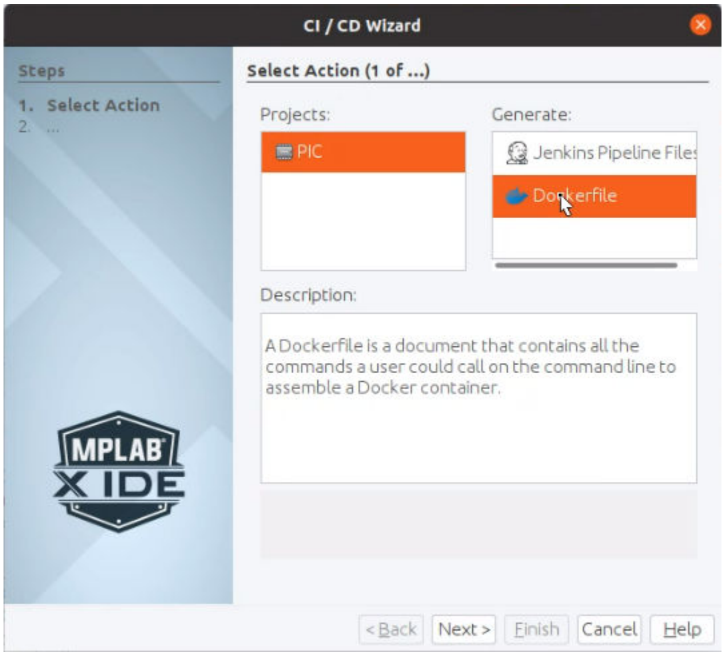
1.6.2 Generate Dockerfile from MPLAB X IDE

1. Open your project in MPLAB X IDE and open *Tools>CI/CD Wizard*.
2. Select **Docker File** as output type.
3. Fill in the remaining information as required in the wizard. See the following sections.
4. After generation, verify that a *Dockerfile* has been added to your project base folder.

1.6.2.1 CI/CD Wizard Dialog 1 - Docker Only

For this example, select “Docker File.” Click **Next** to proceed.

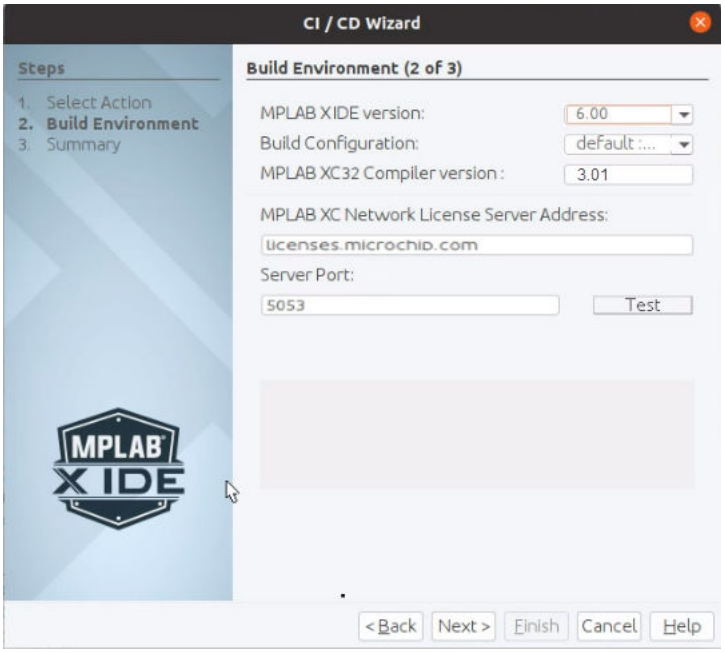
Note: You will need an MPLAB XC compiler license to create the Docker File.



A Dockerfile is a document that contains all the commands a user could call on the command line to assemble a Docker container.

1.6.2.2 CI/CD Wizard Dialog 2 - Docker Only

Set up the build environment to be used in a container. If you have a Network Server license, you can provide information on how to access it for the build. Otherwise leave the text boxes blank (though a note will appear).



CI / CD Wizard

Build Environment (2 of 3)

MPLAB X IDE version: 6.00

Build Configuration: default :...

MPLAB XC32 Compiler version : 3.01

MPLAB XC Network License Server Address:

licenses.microchip.com

Server Port:

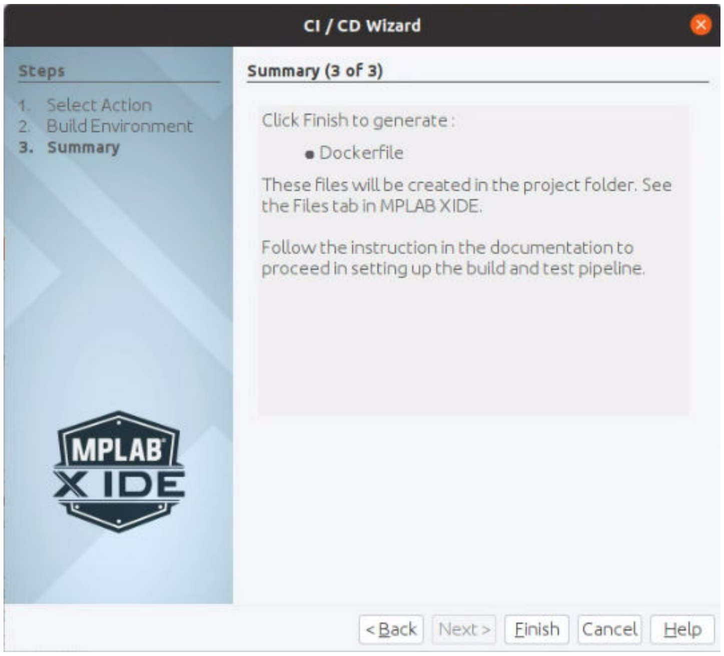
5053

Option	Description
MPLAB IDE version	Select supported versions from the drop-down list. Default is the latest version.
Build Configuration	Select a project configuration for the build
MPLAB XC Compiler version	Select a compiler version for the build
MPLAB XC Network License Server Address	Enter the address of the server. For example: licenses.microchip.com
Server Port	Enter the port number of the server. For example: 5053
Informational Note (conditional)	If you have a Workstation license, information on buying a Network Server license will be shown.

1.6.2.3 CI/CD Wizard Dialog 3 - Docker Only

View the summary to see that only the Dockerfile will be generated.

Click **Back** to change setup items. Click **Finish** to complete the setup. CI/CD Wizard will then generate Dockerfile based on the setup information.



Summary (3 of 3)

Click Finish to generate :

- Dockerfile

These files will be created in the project folder. See the Files tab in MPLAB XIDE.

Follow the instruction in the documentation to proceed in setting up the build and test pipeline.

1.6.2.4 CI/CD Wizard Output - Docker Only

In MPLAB X IDE, you can see:

1. Files that are being generated in the Output window.
2. Resulting files added to the project folder.
3. WebHelp in its own window.
4. Dockerfile in its own window.

This output is a subset of the Jenkins output.

1.6.3 Content of Generated Dockerfile

The generated Dockerfile is tailor-made for the specific MPLAB X IDE project.

The setup includes:

- Debian GNU/Linux 10
- Basic tools such as *curl*, *make* and *unzip*
- MPLAB X IDE installation including the binaries:
  - *mdb* - Microchip Debugger
  - *prjMakefilesGenerator* - CLI for generating Makefiles for MPLAB X IDE projects
  - *xclm* - XC Toolchain license manager
  - *IPE* - MPLAB Integrated Programming Environment
- MPLAB XC Toolchain installation. The installation uses the same version as specified in the MPLAB X IDE project
- The same Device Family Packs (DFPs) as specified in the MPLAB X IDE project

1.6.4 Suggested Next Steps

The generated Dockerfile can be used to set up a build environment for your MPLAB X IDE project. For more information, see Dockerfile reference.

1.6.4.1 Test Image Locally

The dockerfile makes it easy to create and test a docker image locally. If you do not have Docker installed locally, you can get it from www.docker.com.

Commands to build and run your image:

```
# 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
```

Once you have a started the container, you can run the following commands to build your mapped MPLAB X IDE project:

```
# 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 Use Dockerfile in a CI/CD Framework

The CI/CD Wizard has built-in support for **Jenkins**; you can generate support files for this framework. For other frameworks you need to refer to their documentation on how to use Docker.

1.6.4.3 Extending the Dockerfile with Additional Tools/Linux Packages

Additional Linux packages can be installed by modifying your dockerfile. To get started you can first start the image in interactive mode and then manually install the packages from command line:

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

When you have verified that the packages are available and works, you can add them your Dockerfile:

```
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: The last clean up line is used to keep the Docker image as small as possible.

1.7 CICDW Command Line Tool

If you would prefer to run the command-line version of the CI/CD Wizard, it can be found under `<MPLAB X IDE version install directory>mplab_platform/bin/cicdw.exe`.

Use `cicdw.exe -h` to get help on options (see below). For command specific help, use the command and `–h`; for example: `cicdw.exe generate-jenkins -h`.

Note: All command options must be on the same line, as in:

```
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 Example

Bolded text represents content specified by the developer. All content below must be on one line.

```
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 Example

Bolded text represents content specified by the developer. All content below must be on one line.

```
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
```

2. Revision History

The following is a list of changes by version to this document.

Note: Some revision letters are not used - the letters `I` and `O` - as they can be confused for numbers in some fonts.

Revision B (April 2023)

- **1. CI/CD Wizard**: MPLAB ICD 5 added as hardware tool with network support.
- **1.3.1.4 Testing Tools** and **1.3.2.9 CI/CD Wizard Output**: Sections made more generic with reference to supported hardware tools.
- **1.3.2.6 CI/CD Wizard Dialog 6**: Dialog updated to allow selection of MPLAB ICE 4 or MPLAB ICD 5.
- **1.5.2.2 Step 2. Connect MPLAB ICE 4 to Target**: Title updated to remove “Using the Breakout Board” and section updated to add information on using the high-speed cable to connect to the target as well. Updated Table 1-1. “Connection Summary.”
- **1.5.3 Running Unit Tests on MPLAB ICD 5**: Section added to describe how to use MPLAB ICD 5 to run Unity tests.

Revision A (December 2021)

Initial release of this document.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	[X][(1)]	-	X	/XX	XXX
Device	Tape and Reel Option		Temperature Range	Package	Pattern

Device:	PIC16F18313, PIC16LF18313, PIC16F18323, PIC16LF18323	
Tape and Reel Option:	Blank	= Standard packaging (tube or tray)
	T	= Tape and Reel[(1)]
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
Package:[(2)]	JQ	= UQFN
	P	= PDIP
	ST	= TSSOP
	SL	= SOIC-14
	SN	= SOIC-8
	RF	= UDFN
Pattern:	QTP, SQTP, Code or Special Requirements (blank otherwise)	

Examples:

- PIC16LF18313- I/P Industrial temperature, PDIP package
- PIC16F18313- E/SS Extended temperature, SSOP package

Notes:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet- Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, 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, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, NVM Express, NVMe, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time are registered trademarks of Microchip Technology Inc. in other countries.

ISBN: 978-1-6683-2377-9

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820