Zephyr Installation Lab

Rev1.1, August 18, 2022

Estimated Time: 30 mins – 1 hr

Objectives

In this lab, you will learn

- → How to install tool dependencies for Zephyr such as Cmake, Python, and Devicetree compiler
- → How to install Zephyr using West and updating to most recent release
- → Link the gnuarmemb toolchain to run from MCUXpresso IDE
- → Set Windows Environment Variables
- → Using a specific Zephyr release

Software Pre-Requisites

This installation lab is written for Windows10, but Zephyr and these tools can also be easily installed on Linux. If using these other tools, it is recommended to install them first:

- Required:
  - Terminal Program, like PuTTY or Tera Term
- Recommended:
  - NXP recommends using Segger Jlink debug probes with Zephyr and NXP MCUs, although other tools may be an option like pyOCD or OpenOCD. Install J-link Software and Documentation pack (v7.22+ required)
- Optional:
  - MCUXpresso IDE, can be used instead of GCC build tools or Zephyr SDK

Installing Zephyr

The steps to install Zephyr are detailed in Zephyr's documentation. But before those steps, please review the following sections for NXP's recommendations:

Python Virtual Environment

The tools for Zephyr rely on Python. It is easy to run into package incompatibilities when installing dependencies at a system or user level. This situation can happen, for example, if working on multiple Zephyr versions at the same time. For this reason it is suggested to use Python virtual environments. When following the steps in Zephyr Getting Started document, create a Python virtual environment after installing Python, and before installing West.

Installing a Zephyr Release:

Zephyr's Getting Started document provides the steps to install Zephyr from the latest commit in the main branch of the Zephyr repository. But NXP recommends starting with a Zephyr release instead. Releases go through a Stabilization Period and the Quality Process, and are typically a better starting point. Installing the latest Zephyr release is typical, and at the time of this document is **v3.0.0**. To confirm the latest release, check the Zephyr Release Notes page. The Zephyr version can also be easily changed after installing, and details to do this are later in this document.

When using a specific release, you can save some steps and time by running the west init command with the option "--mr v3.0.0". This option tells the West tool to initialize the workspace using the v3.0.0 release of Zephyr. When the instructions at the web page below say to run the west init command, use the command like this instead:

```
west init --mr v3.0.0 zephyrproject
```

Using MCUXpresso IDE instead of installing another toolchain

Building Zephyr applications requires having a toolchain installed. The Zephyr Getting Started steps install the Zephyr SDK, which is a package of several toolchains bundled together. If using NXP's MCUs with ARM Cortex-M cores, the build tool required is GNU ARM Embedded. This build tool is already included in NXP's MCUXpresso IDE. If MCUXpresso IDE is already installed, or will be used for debugging Zephyr applications, there is no need to install the Zephyr SDK. You can review the tools included in this SDK in the Zephyr SDK release notes. These lab steps assume MCUXpresso IDE is already installed, and will setup the environment variables to use the toolchain integrated in MCUXpresso IDE.

Therefore, in Zephyr's Getting Started document, the section "**Install a toolchain**" can be skipped, and Zephyr SDK does not need to be installed.

Follow Zephyr's Getting Started Guide

With these above sections in mind, follow Zephyr's Getting Started Guide for the instructions to install the software and tools, download and prepare the Zephyr repository. Stop and return to this guide before building the Blinky application. Those build steps are detailed in a subsequent NXP lab guide.

Set Windows Environment Variables

It is useful to reboot at this point, after installing all the packages in previous steps.

Here are the steps NXP recommends for setting the environment variables. Create a command file that sets these variables, and run that script in the Command Window before building a Zephyr application. First, create a zephyrrc.cmd file in your %userprofile% directory. Note the paths below to MCUXpresso IDE and Jlink should match the paths in your host system. Use a text editor to add the following environmental variables:

```
set ZEPHYR_TOOLCHAIN_VARIANT=gnuarmemb
set GNUARMEMB_TOOLCHAIN_PATH=C:\nxp\MCUXpressoIDE_11.6.0_8187\ide\tools
set PATH=%PATH%; C:\Program Files\SEGGER\JLink
```

With that file written, now the command file zephyr-env.cmd can be run any time to set these environment variables. Run this command file in a command window once after opening that window, before building or flashing any Zephyr application. This step will also be repeated in subsequent NXP lab guides. Execute the file using this command:

```
%userprofile%\zephyrproject\zephyr\zephyr-env.cmd
```

Zephyr Installed

Congratulations! Zephyr should now be installed and ready to use. Refer to the next NXP lab guide to build and flash your first application.

Changing Zephyr Version

Our training labs are using Zephyr's latest release v3.0.0, and we should confirm the Zephyr repository is setup for this release. If you ran the west init command with the "--mr v3.0.0" option, the repo should be ready. Otherwise, it is easy to change to that release. To confirm the current Zephyr commit, run this Git command to check the repo state:

```
cd %ZEPHYR_BASE%
git log -1
```

If using v3.0.0 release, the log command will output below, confirming this commit has the v3.0.0 tag. In that case, you have v3.0.0 release ready, and can skip the rest of this section.

```
commit 4f8d78ceeb436e82f528511998515f6fc137c6cd (HEAD, tag: zephyr-v3.0.0, tag: v3.0.0)
Author: Dan Kalowsky <dank@deadmime.org>
Date:   Fri Feb 18 15:13:37 2022 -0800

    release: Zephyr 3.0.0

    Update version for release v3.0.0

    Signed-off-by: Dan Kalowsky <dank@deadmime.org>
```

When running the west init command with no options, it will default to checking out the Main branch, instead of a specific Zephyr release. The git log command will print something like below, showing it is using the main branch, and not print anything about the v3.0.0 release tag.

```
commit c9aa260f0cfc54453aef70f876e9ba9385428a30 (HEAD, origin/main)
```

To checkout the release, use this Git command

```
git checkout v3.0.0
```

If you are using a previous version of zephyr and the tag is not available, have git fetch the tag from the remote first:

```
git fetch origin
```

Whenever you check out a different revision in your manifest repository (in this case, the zephyr repo), you should run "west update" to make sure your workspace synchronizes the other project repositories with your

current Zephyr version. Run the command:

```
west update
```

Now your Zephyr repository and other project repos are configured for v3.0.0 Release.

Revision History

Rev	Date	Details
1.0	5/29/2022	Initial Version
1.1	8/18/2022	Updated version of MCUXpresso IDE path