# Hands-On Workshop: Build Your First Zephyr Application on i.MX RT

Maureen Helm, NXP
Thea Aldrich, Linux Foundation

June 2019 | Session #AMF-SOL-T3639

SECURE CONNECTIONS FOR A SMARTER WORLD

Company Public – NXP, the NXP logo, and NXP secure connections for a smarter world are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2019 NXP B.V.

## Agenda

- Introduce the Zephyr Project
- Review High-level Software Features and Hardware Support
- Set up a Development Environment
- Hands-on: Build, Flash, and Debug an Application

# Zephyr Project Introduction

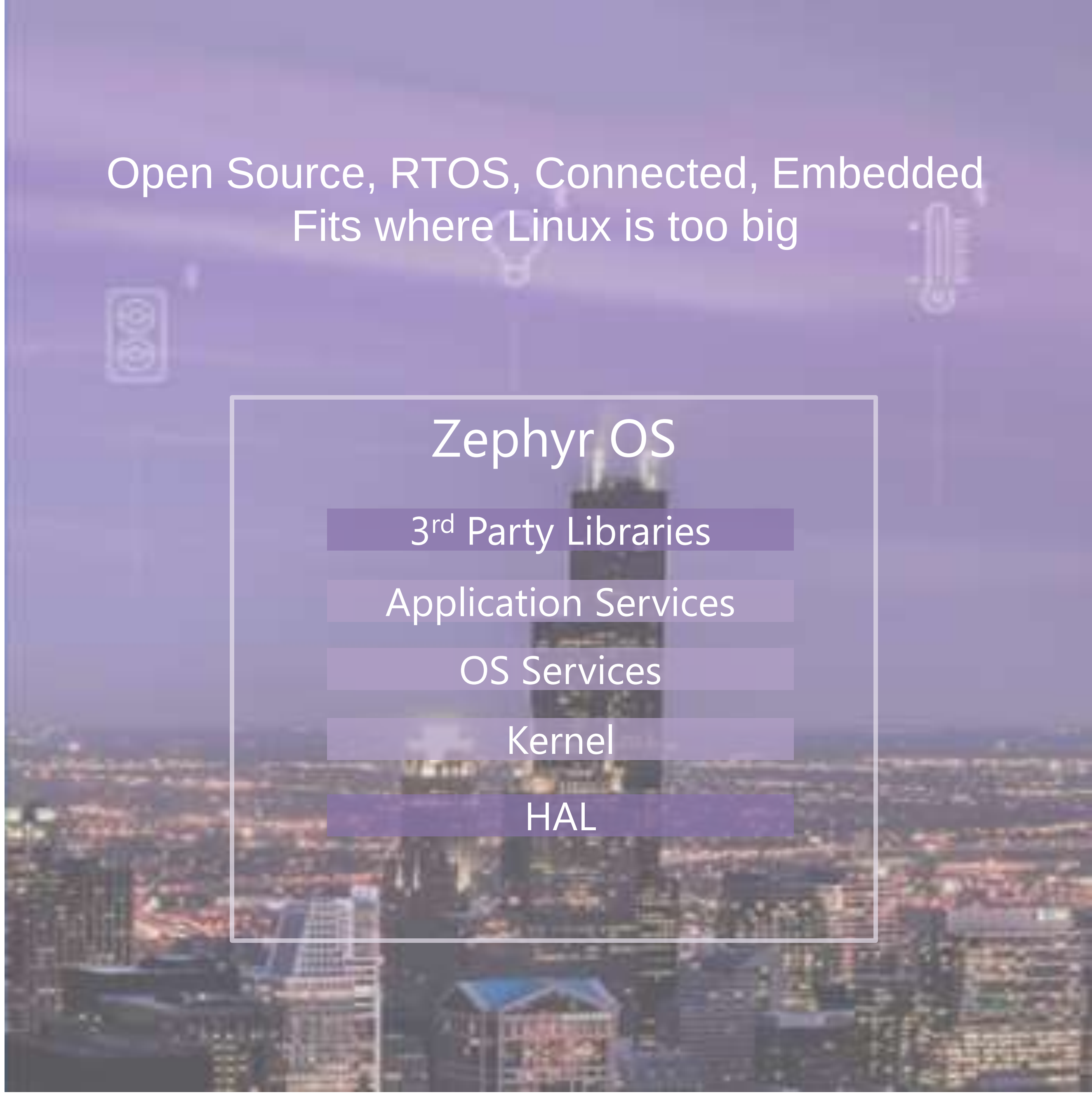
What is the Zephyr Project? Why should I use it?

## Zephyr Project

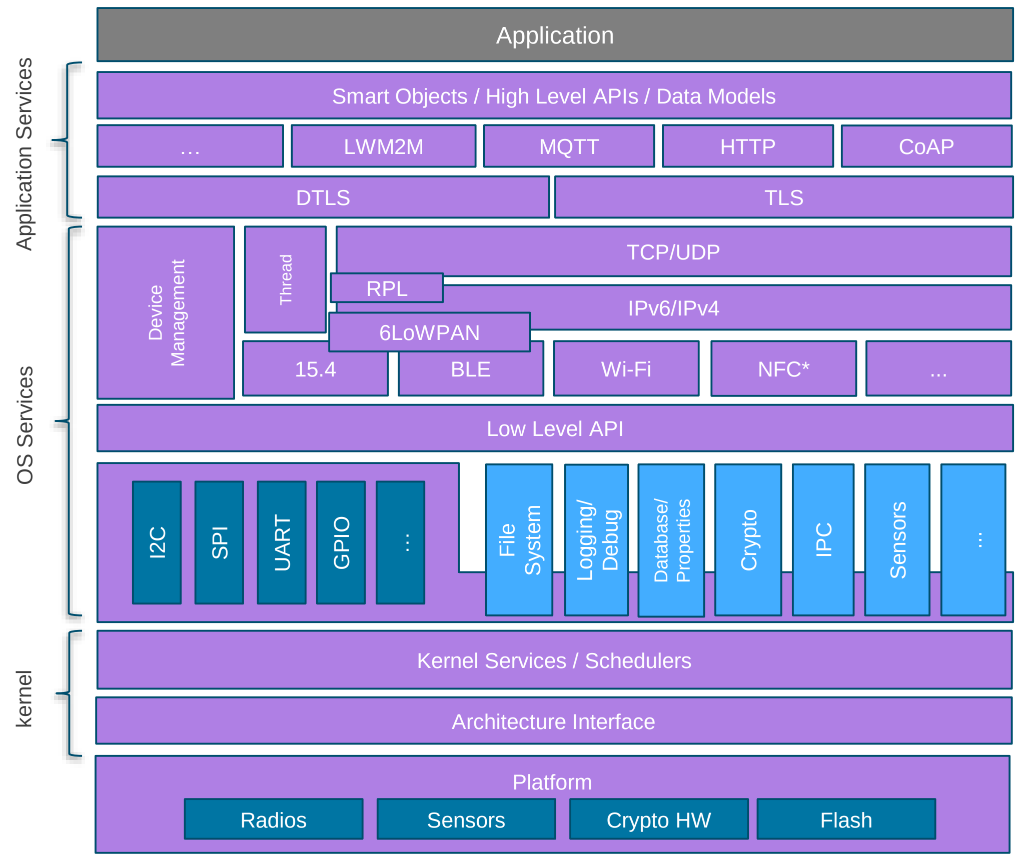
- Open source real time operating system
- Vibrant Community participation
- Built with safety and security in mind
- Cross-architecture with growing developer tool support
- Vendor Neutral governance
- Permissively licensed - Apache 2.0
- Complete, fully integrated, highly configurable, modular for flexibility, better than roll-your-own
- Product development ready with LTS
- Certification ready with Auditable

THE LINUX FOUNDATION PROJECTS

Open Source, RTOS, Connected, Embedded
Fits where Linux is too big



# Architecture



- Highly Configurable, Highly Modular
- Cooperative and Pre-emptive Threading
- Memory and Resources are typically statically allocated
- Integrated device driver interface
- Memory Protection: Stack overflow protection, Kernel object and device driver permission tracking, Thread isolation
- Bluetooth® Low Energy (BLE 4.2, 5.0) with both controller and host, BLE Mesh
- Native, fully featured and optimized networking stack

**Fully featured OS allows developers to focus on the application**

# NXP Board Support

- i.MX RT Series (Cortex M7)
  - RT1015 EVK
  - RT1020 EVK
  - RT1050 EVK
  - RT1060 EVK
  - RT1064 EVK

- Kinetis Series (Cortex M4, M0+)
  - FRDM-K64F
  - FRDM-KW41Z
  - FRDM-KL25Z
  - TWR-KE18F
  - Hexiwear

- i.MX 6/7 Series (Cortex M4 subsystem)
  - UDOO Neo Full
  - Colibri iMX7
  - WaRP7

- LPC Series (Cortex M4, M0+, M33)
  - LPCXpresso54114
  - LPCXpresso55S69 (coming soon)

https://docs.zephyrproject.org/latest/boards/index.html

## NXP Board Support

- Upstream
  - Contributed and maintained by NXP and the community
  - NXP active in upstream working groups

- Built upon MCUXpresso SDK
  - SDK bare metal drivers and CMSIS device headers contributed upstream
  - Shim drivers adapt SDK interfaces to Zephyr interfaces
  - Maximizes code reuse

- Tested on hardware in NXP board farm

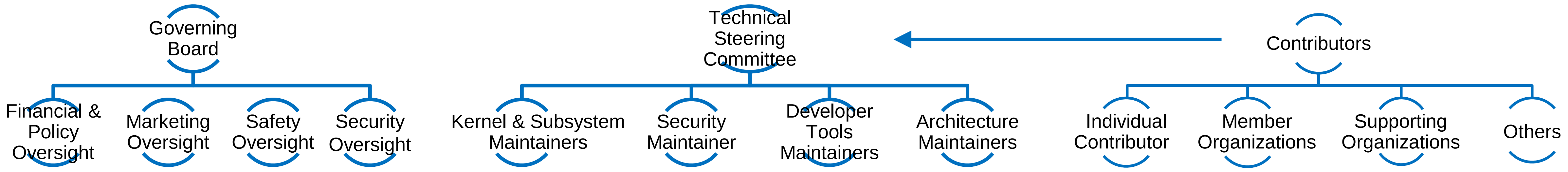
## Long Term Support (LTS) Release

- Product-focused release will receive bug fixes and maintain stable APIs for two years
- Extended stabilization period enabled more testing and bug fixing prior to release
- Baseline for auditable version of Zephyr

- Released in Apr 2019 (Zephyr v1.14.0)
- Supports over 160 board configurations across 8 architectures
- Contributions from 250 developers

- Hands-on exercises in this workshop use the LTS release

# Zephyr Project Governance



## Goal: Separate business decisions from meritocracy, technical decisions

### Governing Board

- Decides project goals
- Sets business , marketing and legal decisions
- Prioritizes investments and oversees budget
- Oversees marketing such as PR/AR, branding, others
- Identifies member requirements

### Technical Steering Committee

- Serves as the highest technical decision body consisting of project maintainers and voting members
- Sets technical direction for the project
- Coordinates X-community collaboration
  - Sets up new projects
  - Coordinates releases
  - Enforces development processes
  - Moderates working groups
- Oversees relationships with other relevant projects

### Community

- Code base open to all contributors, need not be a member to contribute.
- Path to committer and maintainer status through peer assessed merit of contributions and code reviews
- Ecosystem enablement

## Zephyr Project Membership

February 2016

May 2019

and others….

## Zephyr Development Environment

What tools do I need? How do I install them on my PC?

## Development Environment Introduction

- Zephyr applications can be developed on Windows, Linux, or macOS host operating systems
- CMake and Python enable portability across host operating systems
- Detailed instructions are documented in the Getting Started Guide

- Major components:
  - Python 3: Script interpreter and packages
  - CMake/Ninja/Make: Build system
  - Device Tree Compiler: Compiles device tree hardware descriptions
  - Toolchain: gcc for Arm, RISC-V, x86, etc.
  - Debug/Flash Tools: J-Link, pyOCD, OpenOCD, etc.
  - West: Custom tool for repository management, build/flash/debug assistance, and image signing
  - Zephyr Git repositories: The source code!

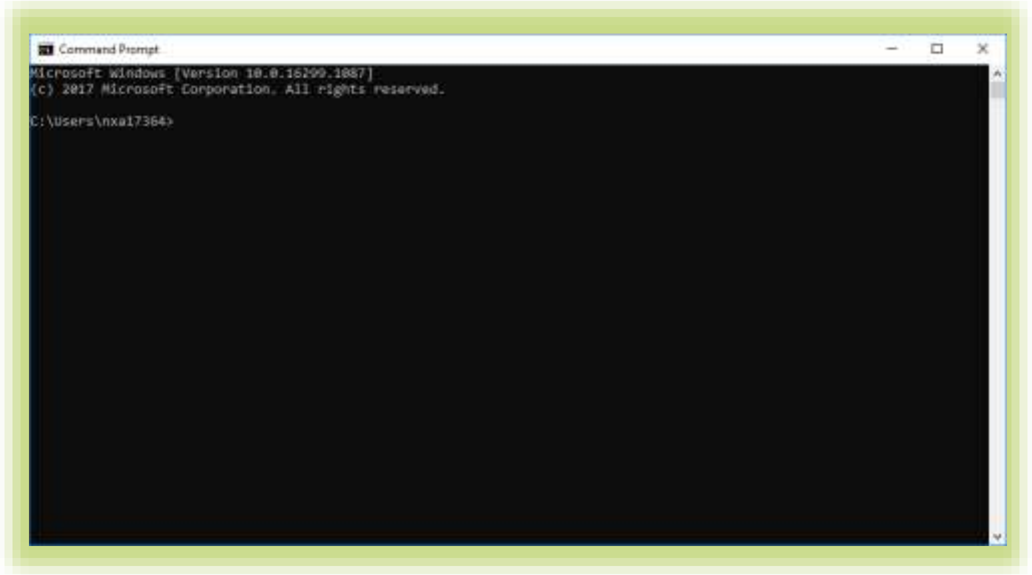
- Zephyr SDK provides toolchains and some debug/flash tools for Linux only

## Windows: Command Prompt, WSL, or VM?

- Windows Command Prompt: Requires manual toolchain installation, but can use debug/flash tools like J-Link and pyOCD. Recommended for new developers

- Windows Subsystem for Linux (WSL): Can use Zephyr SDK toolchains and sanitycheck, but does not support debug/flash tools like J-Link and pyOCD. Not recommended

- Linux Virtual Machine (VM): Can use Zephyr SDK toolchains, sanitycheck, and debug/flash tools like J-Link and pyOCD; but requires installing a virtual machine. Recommended for experienced developers

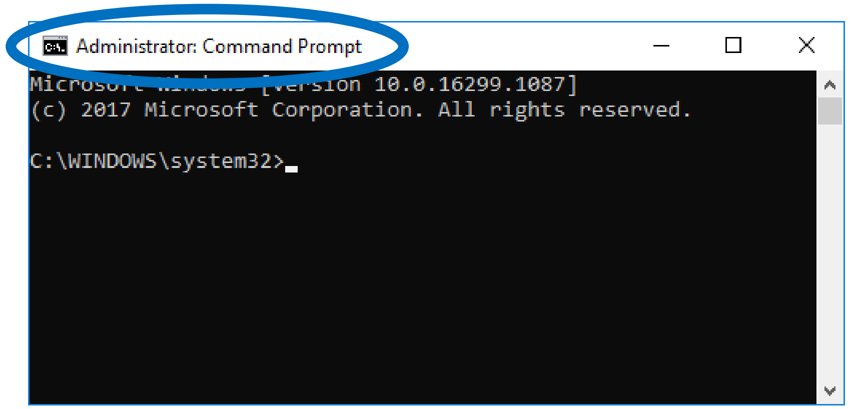


```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\nxa17364>
```

```
maureen@NXL08208:~$
```

# Windows: Install Chocolatey and Packages

- Open an administrator command prompt


Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>
```

- Install Chocolatey package manager
  - Similar to apt on Ubuntu

- Disable global confirmation

```
> choco feature enable -n allowGlobalConfirmation
```

- Use Chocolatey to install CMake

```
> choco install cmake --installargs 'ADD_CMAKE_TO_PATH=System'
```

- Use Chocolatey to install dependencies

```
> choco install git python ninja dtc-msys2 gperf
```

# Windows: Bootstrap West and Clone Zephyr Repos

- Open a normal command prompt

- Bootstrap west

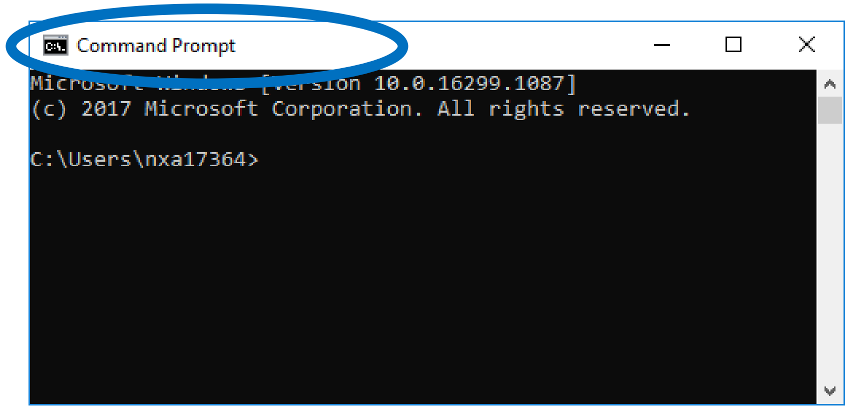
```
> pip3 install west
```

- Clone the Zephyr git repositories

```
> cd %userprofile%
> west init --mr v1.14.0 zephyrproject
> cd zephyrproject
> west update
```

- Install python dependencies

```
> pip3 install -r zephyr/scripts/requirements.txt
```



```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\nxa17364>
```

## Windows: Install Toolchain and Flash/Debug Tools

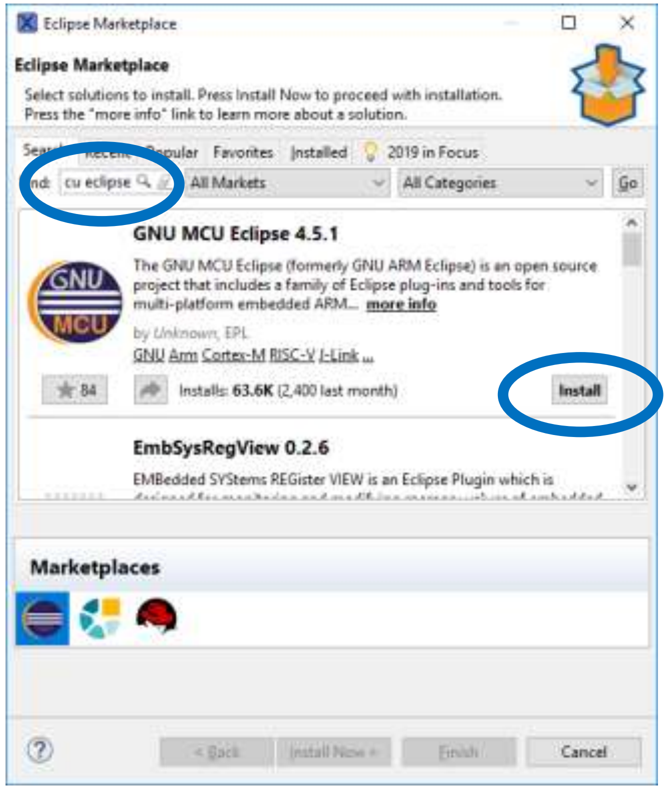
- Install GNU Arm Embedded toolchain
  - Use Windows ZIP instead of Windows Installer. This will allow you to define an installation path without spaces
  - Skip this step if you already have MCUXpresso IDE installed
- Install J-Link flash/debug tools with Windows installer
  - Required for i.MX RT and LPC boards, optional for Kinetis boards
  - Skip this step if you already have MCUXpresso IDE installed
- Create zephyrrc.cmd file in %userprofile% directory

```
set ZEPHYR_TOOLCHAIN_VARIANT=gnuarmemb
set GNUARMEMB_TOOLCHAIN_PATH=C:\nxp\MCUXpressoIDE_10.3.1_2233\ide\tools
set PATH=%PATH%;C:\Program Files (x86)\SEGGER\JLink_V642b
```

## Install Eclipse IDE Plugins

- Install Eclipse IDE for C/C++ Developers
  - Skip this step if you already have MCUXpresso IDE installed

- Install GNU MCU Eclipse plug-ins
  - From the Help menu, select Eclipse Marketplace
  - Search for "*gnu mcu eclipse*" and click Install



Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

GNU MCU Eclipse 4.5.1

The GNU MCU Eclipse (formerly GNU ARM Eclipse) is an open source project that includes a family of Eclipse plug-ins and tools for multi-platform embedded ARM... more info

by Unknown, EPL

GNU Arm Cortex-M RISC-V J-Link ...

Installs: 63.6K (2,400 last month)

EmbSysRegView 0.2.6

EMBedded SYStems REGister VIEW is an Eclipse Plugin which is

## Hands-On Exercises

The Fun Part!

## Hands-On Overview

- Exercise #1: Blinky
  - Build and flash a simple application
  - Examine application source code and build artifacts
- Exercise #2: Eclipse IDE Debugging
  - Generate and import an Eclipse IDE project
  - Create and launch a debug configuration
- Exercise #3: Display and Graphics with LittlevGL Integration
  - Build and flash an LCD application
- Exercise #4: Configuration and Memory Footprint
  - Examine flash/ram footprint with rom_report and ram_report
  - Change the configuration and rebuild

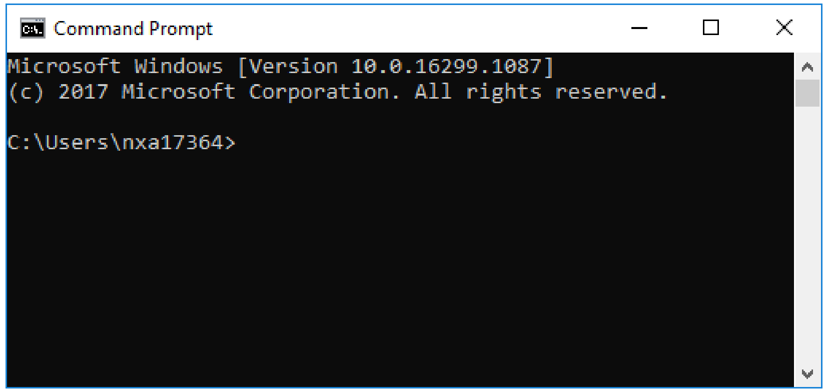
## Exercise #1: Blinky

Build and flash a simple application

Examine application source code and build artifacts

# Build and Flash Blinky

- Open a normal command prompt



Command Prompt

```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\nxa17364>
```

- Set up the build environment

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

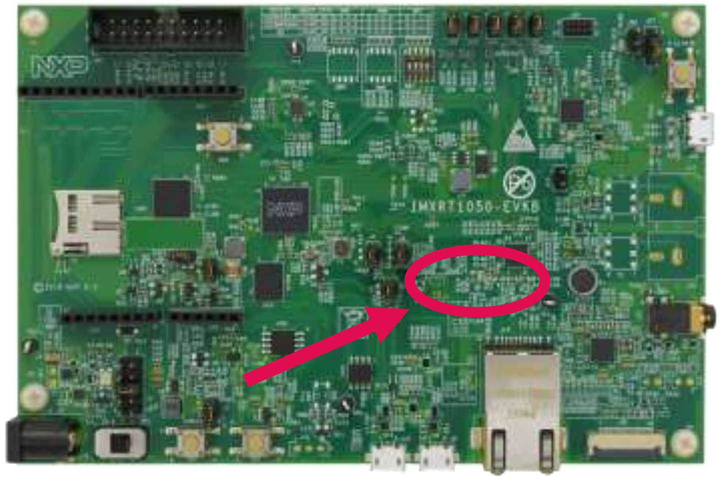
- Build the blinky sample application

```
> west build -b mimxrt1050_evk -d build\blinky samples\basic\blinky
```

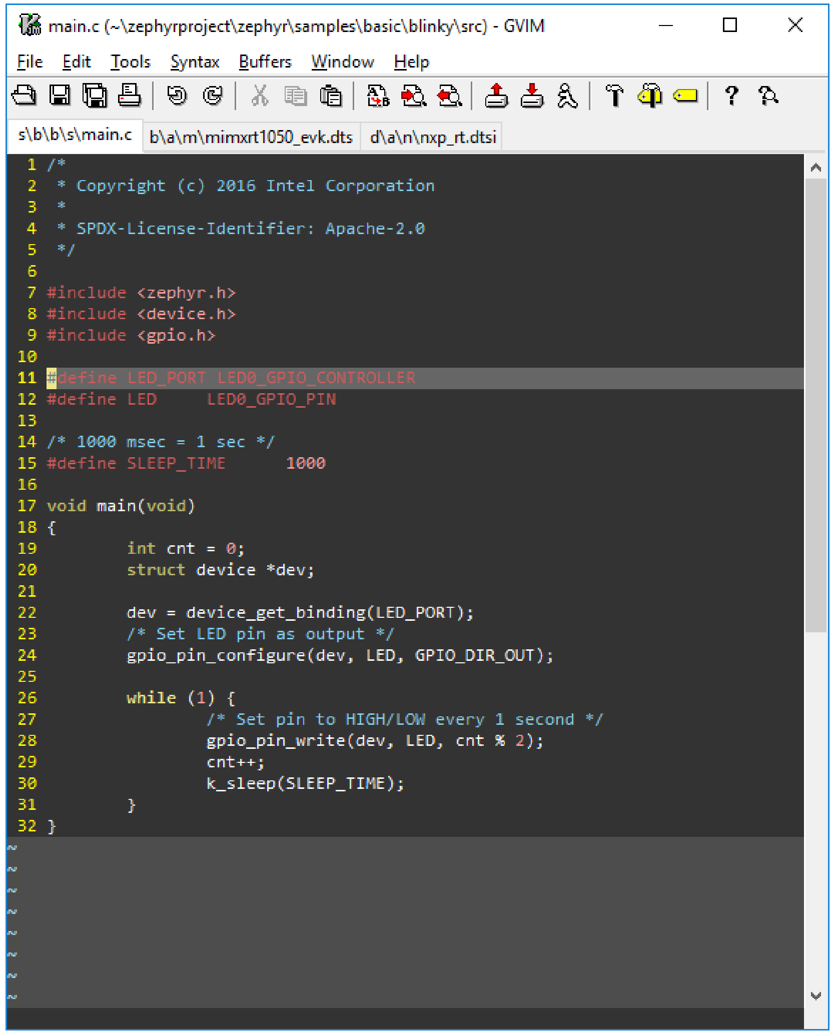
- Flash it to the board

```
> west flash -d build\blinky
```

- See the LED blinking



# Blinky Application Source Code

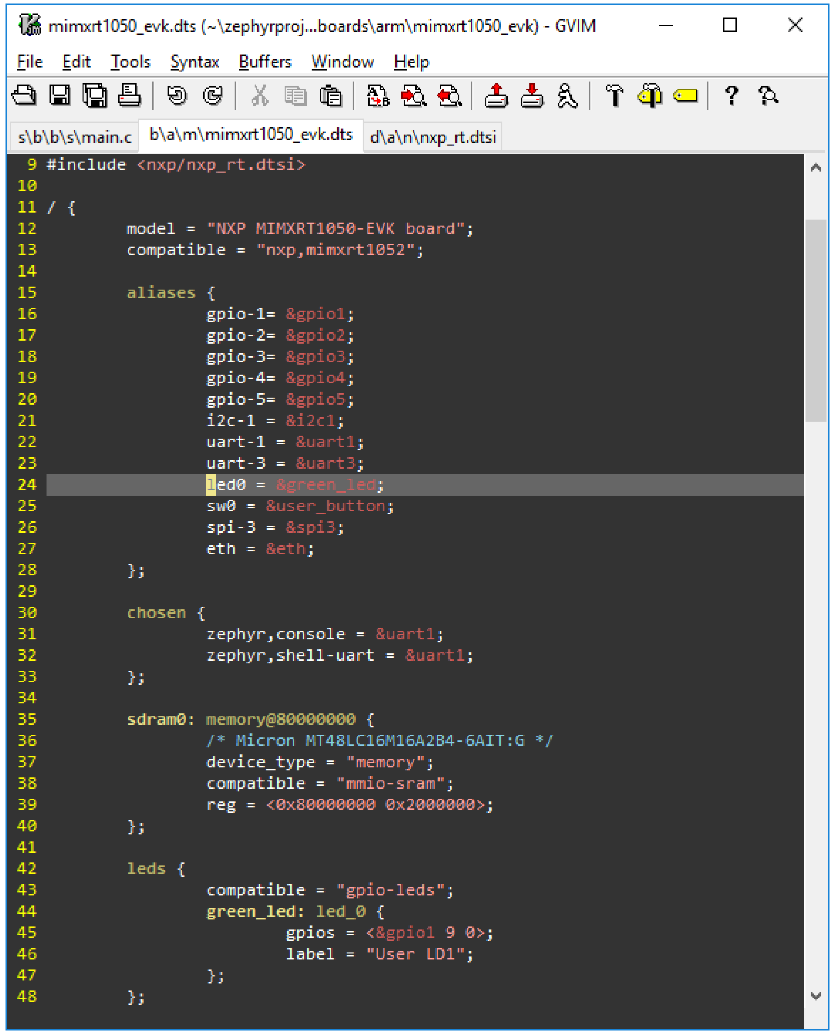


main.c (~\zephyrproject\zephyr\samples\basic\blinky\src) - GVIM
File Edit Tools Syntax Buffers Window Help
s\b\b\s\main.c
b\a\m\mimxrt1050_evk.dts
d\a\n\nxp_rt.dtsi

```c
/*
 * Copyright (c) 2016 Intel Corporation
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <zephyr.h>
#include <device.h>
#include <gpio.h>

#define LED_PORT LED0_GPIO_CONTROLLER
#define LED      LED0_GPIO_PIN

/* 1000 msec = 1 sec */
#define SLEEP_TIME      1000

void main(void)
{
        int cnt = 0;
        struct device *dev;

        dev = device_get_binding(LED_PORT);
        /* Set LED pin as output */
        gpio_pin_configure(dev, LED, GPIO_DIR_OUT);

        while (1) {
                /* Set pin to HIGH/LOW every 1 second */
                gpio_pin_write(dev, LED, cnt % 2);
                cnt++;
                k_sleep(SLEEP_TIME);
        }
}
```

- samples\basic\blinky\src\main.c
- Same application source code works on many different boards, not just i.MX RT1050-EVKB
- Standard GPIO interface APIs
  - `gpio_pin_configure()` and `gpio_pin_write()`
- Standard LED macros generated from device tree
  - `LED0_GPIO_CONTROLLER` and `LED0_GPIO_PIN`

## i.MX RT1050-EVK Board Device Tree


mimxrt1050_evk.dts (~\zephyrproj...boards\arm\mimxrt1050_evk) - GVIM
File Edit Tools Syntax Buffers Window Help
s\b\b\s\main.c
b\a\m\mimxrt1050_evk.dts
d\a\n\nxp_rt.dtsi

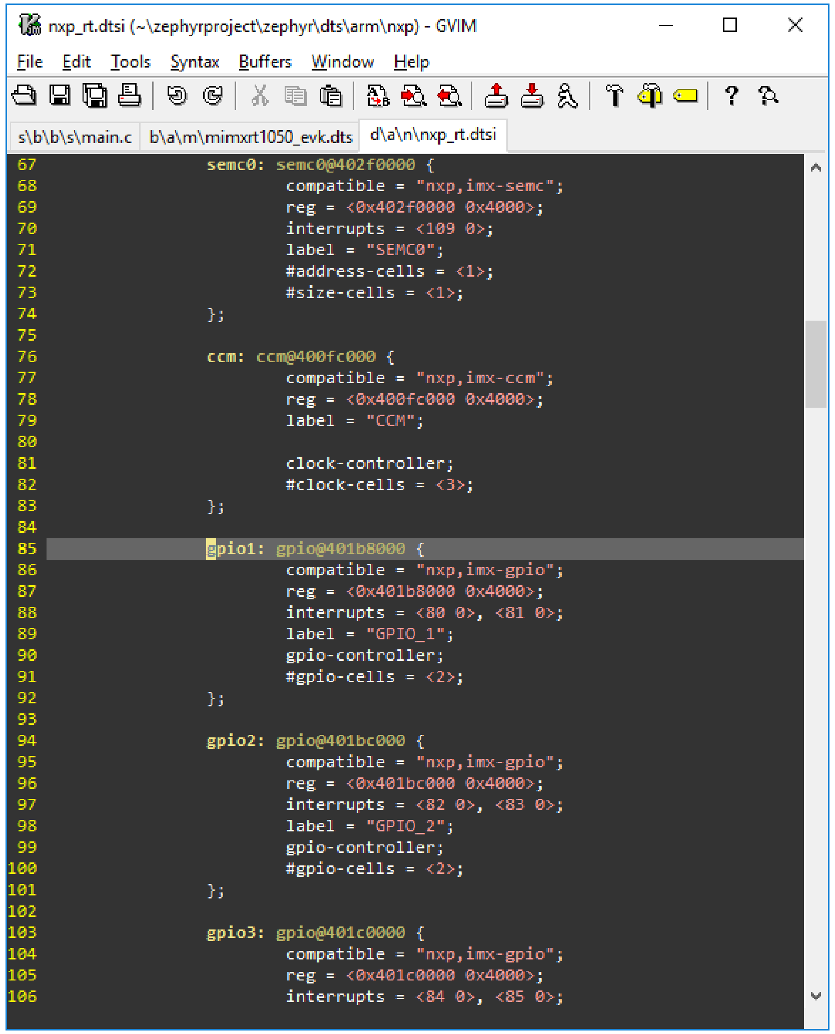
```
#include <nxp/nxp_rt.dtsi>

/ {
	model = "NXP MIMXRT1050-EVK board";
	compatible = "nxp,mimxrt1052";

	aliases {
		gpio-1= &gpio1;
		gpio-2= &gpio2;
		gpio-3= &gpio3;
		gpio-4= &gpio4;
		gpio-5= &gpio5;
		i2c-1 = &i2c1;
		uart-1 = &uart1;
		uart-3 = &uart3;
		led0 = &green_led;
		sw0 = &user_button;
		spi-3 = &spi3;
		eth = &eth;
	};

	chosen {
		zephyr,console = &uart1;
		zephyr,shell-uart = &uart1;
	};

	sdram0: memory@80000000 {
		/* Micron MT48LC16M16A2B4-6AIT:G */
		device_type = "memory";
		compatible = "mmio-sram";
		reg = <0x80000000 0x2000000>;
	};

	leds {
		compatible = "gpio-leds";
		green_led: led_0 {
			gpios = <&gpio1 9 0>;
			label = "User LD1";
		};
	};
```

- boards\arm\mimxrt1050_evk\mimxrt1050_evk.dts

- Defines board hardware components such as LEDs, sensors, and external memories
  - LED node defines GPIO instance and pin
  - Memory nodes define SDRAM and Hyperflash sizes
  - Chosen node selects UART instance for console

- Includes SoC device tree

# i.MX RT1050 SoC Device Tree


nxp_rt.dtsi (~\zephyrproject\zephyr\dts\arm\nxp) - GVIM
File Edit Tools Syntax Buffers Window Help
s\b\b\s\main.c
b\a\m\mimxrt1050_evk.dts
d\a\n\nxp_rt.dtsi

```
                semc0: semc0@402f0000 {
                        compatible = "nxp,imx-semc";
                        reg = <0x402f0000 0x4000>;
                        interrupts = <109 0>;
                        label = "SEMC0";
                        #address-cells = <1>;
                        #size-cells = <1>;
                };

                ccm: ccm@400fc000 {
                        compatible = "nxp,imx-ccm";
                        reg = <0x400fc000 0x4000>;
                        label = "CCM";

                        clock-controller;
                        #clock-cells = <3>;
                };

                gpio1: gpio@401b8000 {
                        compatible = "nxp,imx-gpio";
                        reg = <0x401b8000 0x4000>;
                        interrupts = <80 0>, <81 0>;
                        label = "GPIO_1";
                        gpio-controller;
                        #gpio-cells = <2>;
                };

                gpio2: gpio@401bc000 {
                        compatible = "nxp,imx-gpio";
                        reg = <0x401bc000 0x4000>;
                        interrupts = <82 0>, <83 0>;
                        label = "GPIO_2";
                        gpio-controller;
                        #gpio-cells = <2>;
                };

                gpio3: gpio@401c0000 {
                        compatible = "nxp,imx-gpio";
                        reg = <0x401c0000 0x4000>;
                        interrupts = <84 0>, <85 0>;
```

- dts\arm\nxp\nxp_rt.dtsi
- Defines SoC peripheral addresses, interrupts, and device driver labels
- Clocks properties used by peripheral drivers to configure UART, I2C baud rates

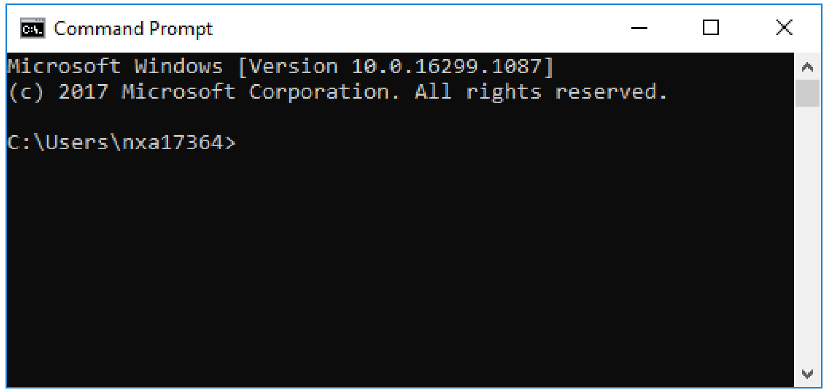
# Exercise #2: Eclipse IDE Debugging

Generate and import an Eclipse IDE project

Create and launch a debug configuration

## Generate an Eclipse IDE Project

- Open a normal command prompt



```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\nxa17364>
```

- Set up the build environment

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

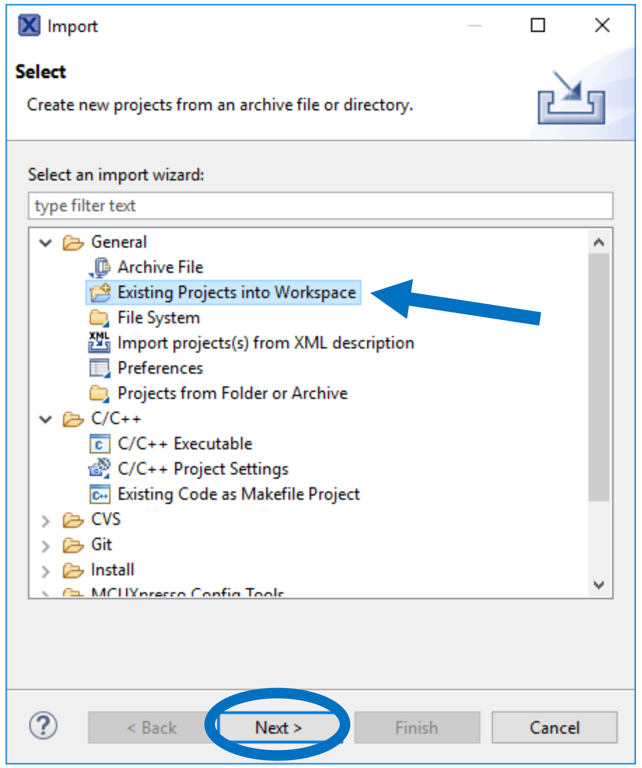
- Move to a directory outside the Zephyr tree. This is required only when generating Eclipse projects

```
> cd %userprofile%
```

- Generate and build an Eclipse project for the hello_world application

```
> west build -b mimxrt1050_evk %ZEPHYR_BASE%\samples\hello_world -
- -G"Eclipse CDT4 - Ninja"
```

# Import the Eclipse IDE Project

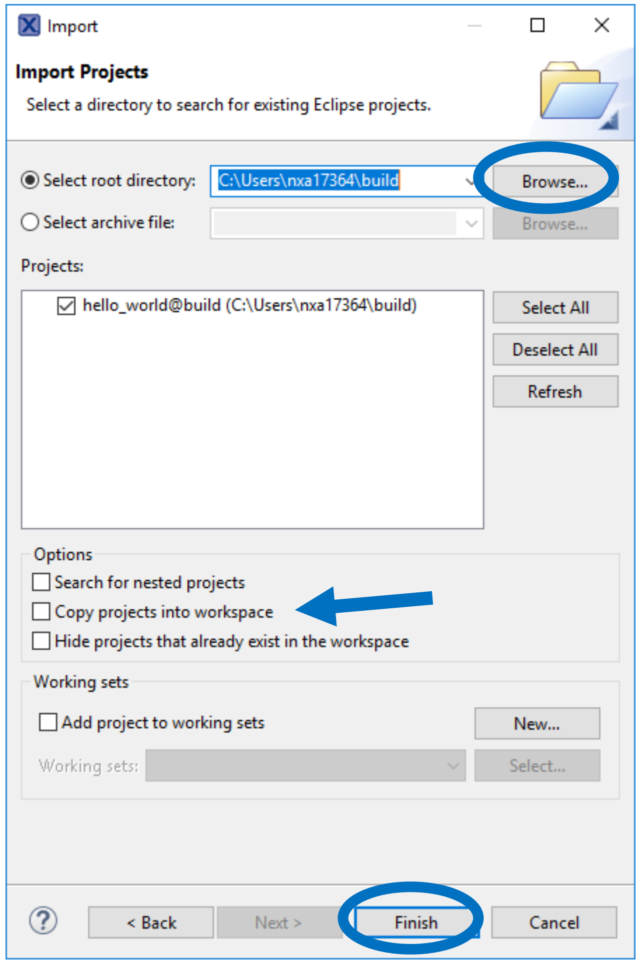


Import

**Select**

Create new projects from an archive file or directory.

Select an import wizard:



Import

**Import Projects**

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\Users\nxa17364\build

( ) Select archive file:

Projects:

☑ hello_world@build (C:\Users\nxa17364\build)

Options

☐ Search for nested projects

☐ Copy projects into workspace

☐ Hide projects that already exist in the workspace

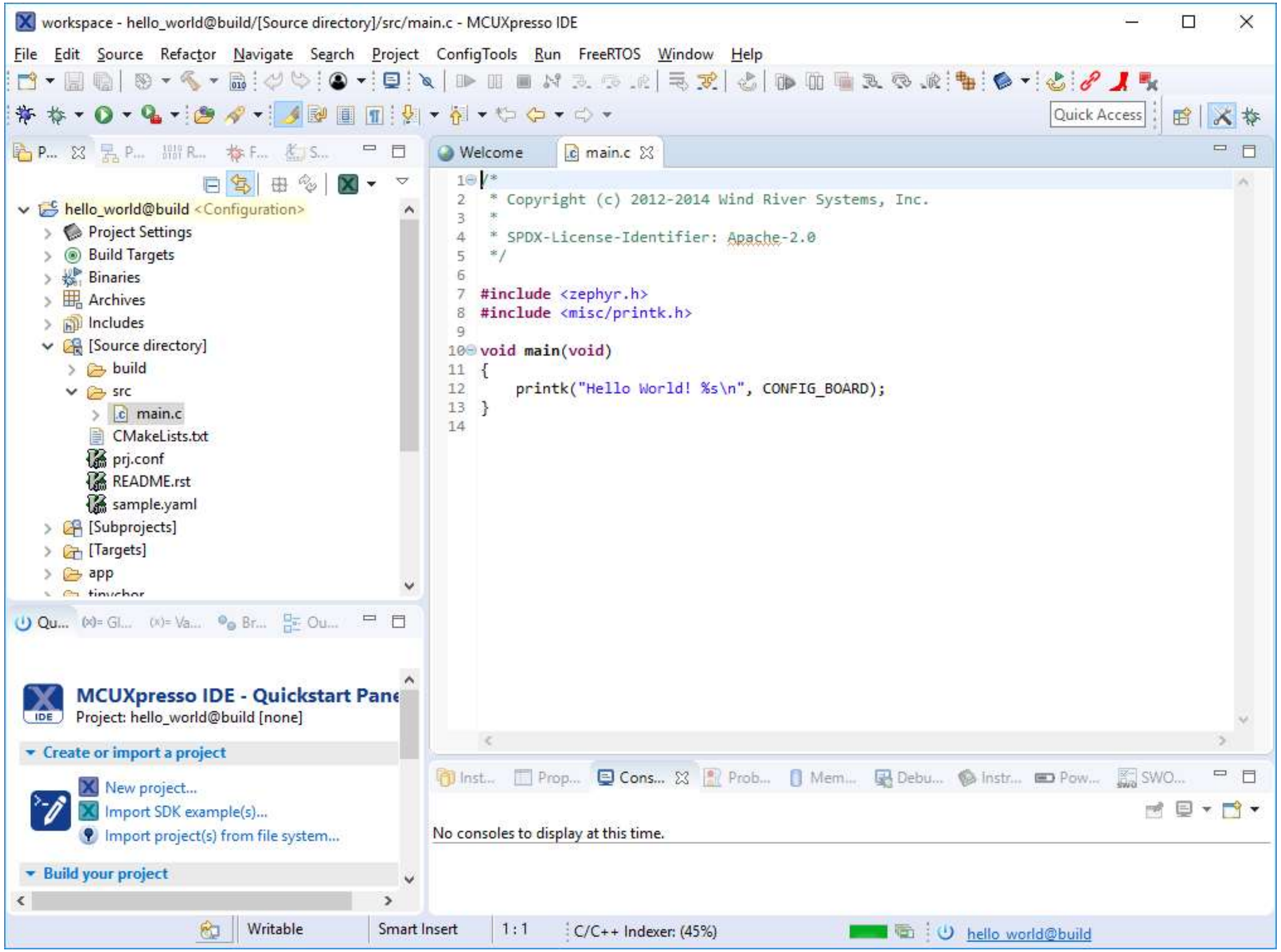
Working sets

☐ Add project to working sets

Working sets:

- Open MCUXpresso IDE
- From the File menu, select Import…
- Select Existing Projects into Workspace
- Select Next
- Select Browse and navigate to your build directory
- Select Finish

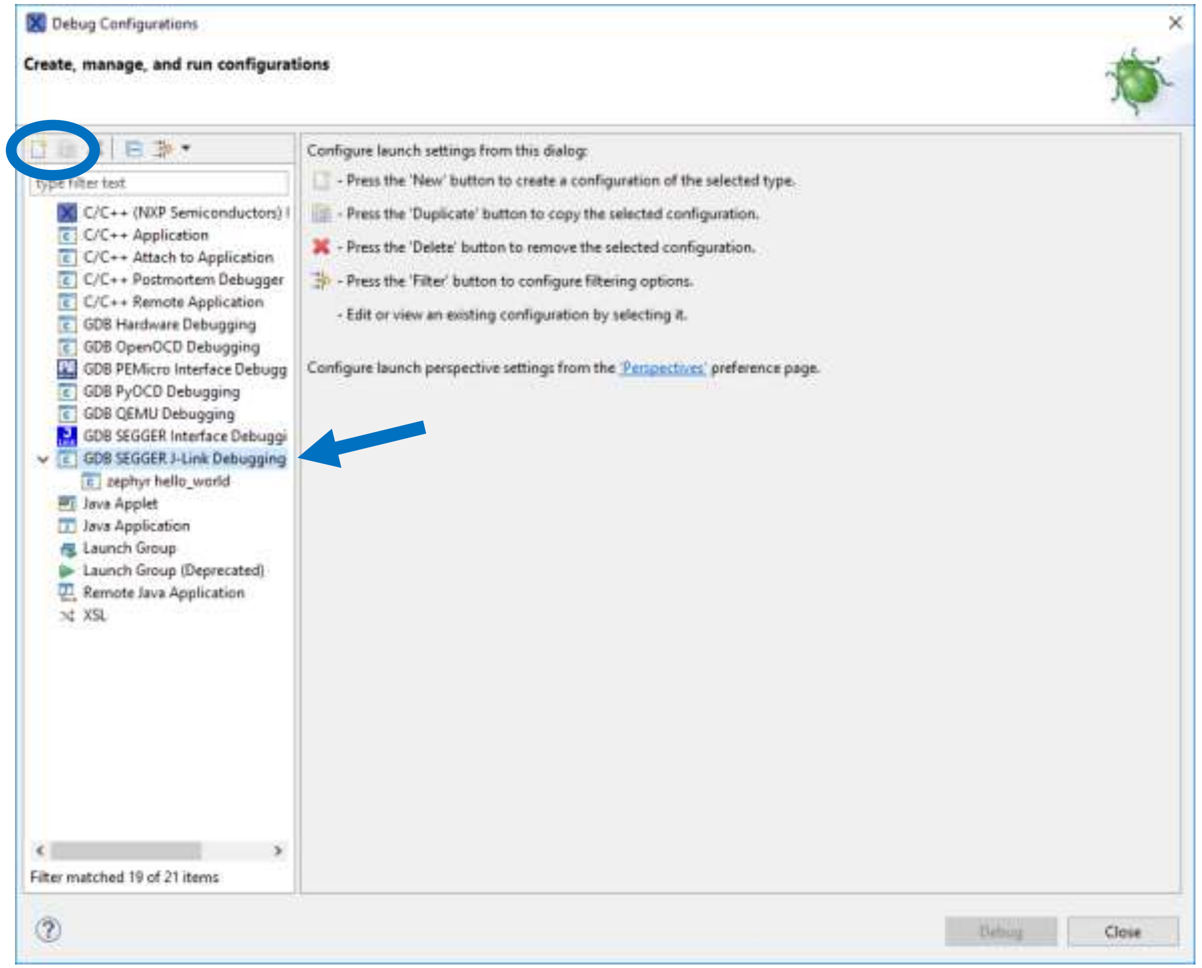
Warning: Do not check Copy projects into Workspace


workspace - hello_world@build/[Source directory]/src/main.c - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run FreeRTOS Window Help
Quick Access
hello_world@build <Configuration>
Project Settings
Build Targets
Binaries
Archives
Includes
[Source directory]
build
src
main.c
CMakeLists.txt
prj.conf
README.rst
sample.yaml
[Subprojects]
[Targets]
app
Welcome
main.c
MCUXpresso IDE - Quickstart Pane
Project: hello_world@build [none]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Writable
Smart Insert
1 : 1
C/C++ Indexer: (45%)
hello_world@build

```c
/*
 * Copyright (c) 2012-2014 Wind River Systems, Inc.
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <zephyr.h>
#include <misc/printk.h>

void main(void)
{
	printk("Hello World! %s\n", CONFIG_BOARD);
}
```

```
No consoles to display at this time.
```

# Create a New Debug Configuration

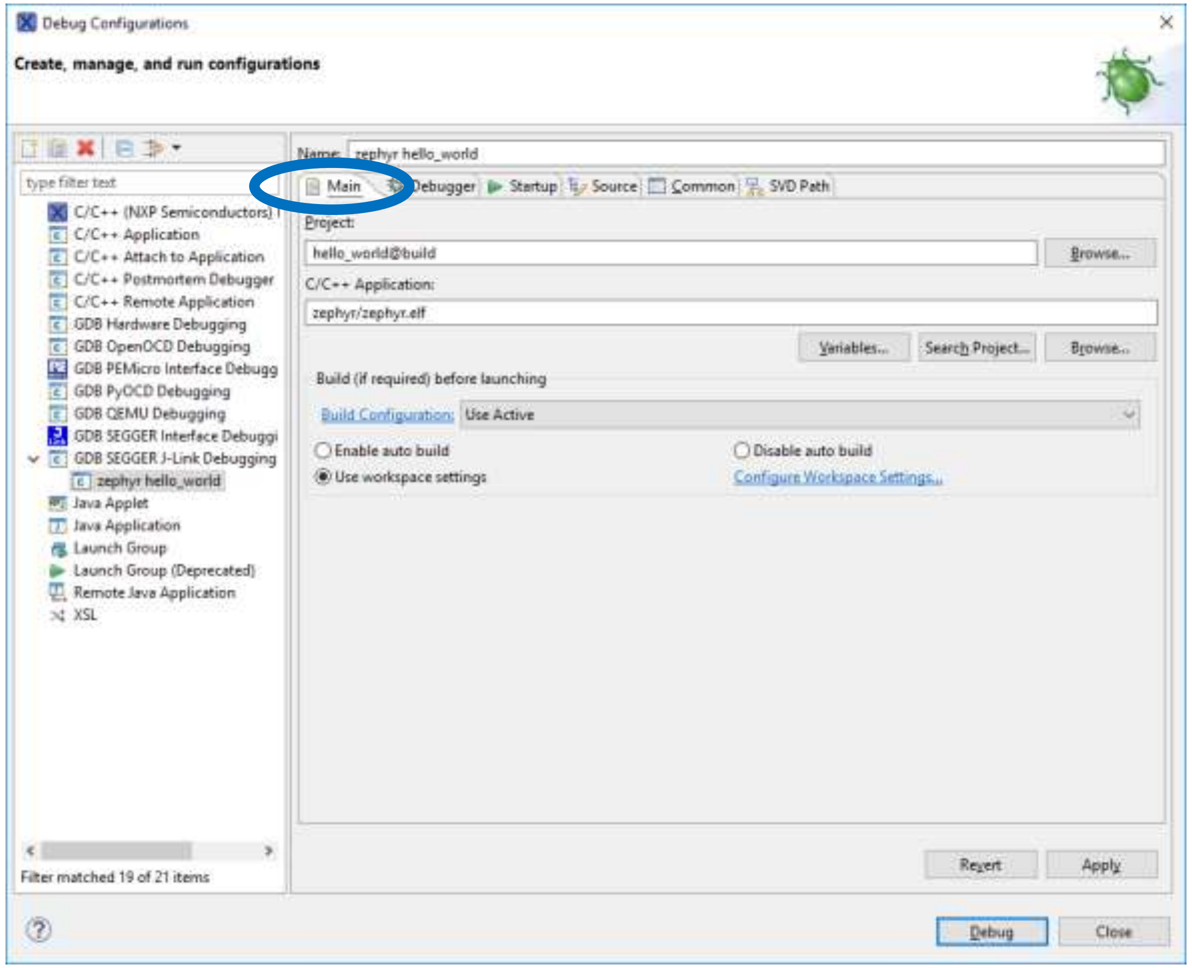

Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ (NXP Semiconductors)
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
GDB OpenOCD Debugging
GDB PEMicro Interface Debugg
GDB PyOCD Debugging
GDB QEMU Debugging
GDB SEGGER Interface Debuggi
GDB SEGGER J-Link Debugging
zephyr hello_world
Java Applet
Java Application
Launch Group
Launch Group (Deprecated)
Remote Java Application
XSL
Filter matched 19 of 21 items
Debug
Close

Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

- From the Run menu, select Debug Configurations…
- Select GDB SEGGER J-Link-Debugging, and click the New button
- Warning: Do not select GDB SEGGER Interface Debugging

# J-Link Debug Configuration: Main



Debug Configurations

**Create, manage, and run configurations**

type filter text

- C/C++ (NXP Semiconductors)
- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB PEMicro Interface Debugg
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER Interface Debuggi
- GDB SEGGER J-Link Debugging
  - zephyr hello_world
- Java Applet
- Java Application
- Launch Group
- Launch Group (Deprecated)
- Remote Java Application
- XSL

Filter matched 19 of 21 items

Name: zephyr hello_world

Main | Debugger | Startup | Source | Common | SVD Path

Project:
hello_world@build

C/C++ Application:
zephyr/zephyr.elf

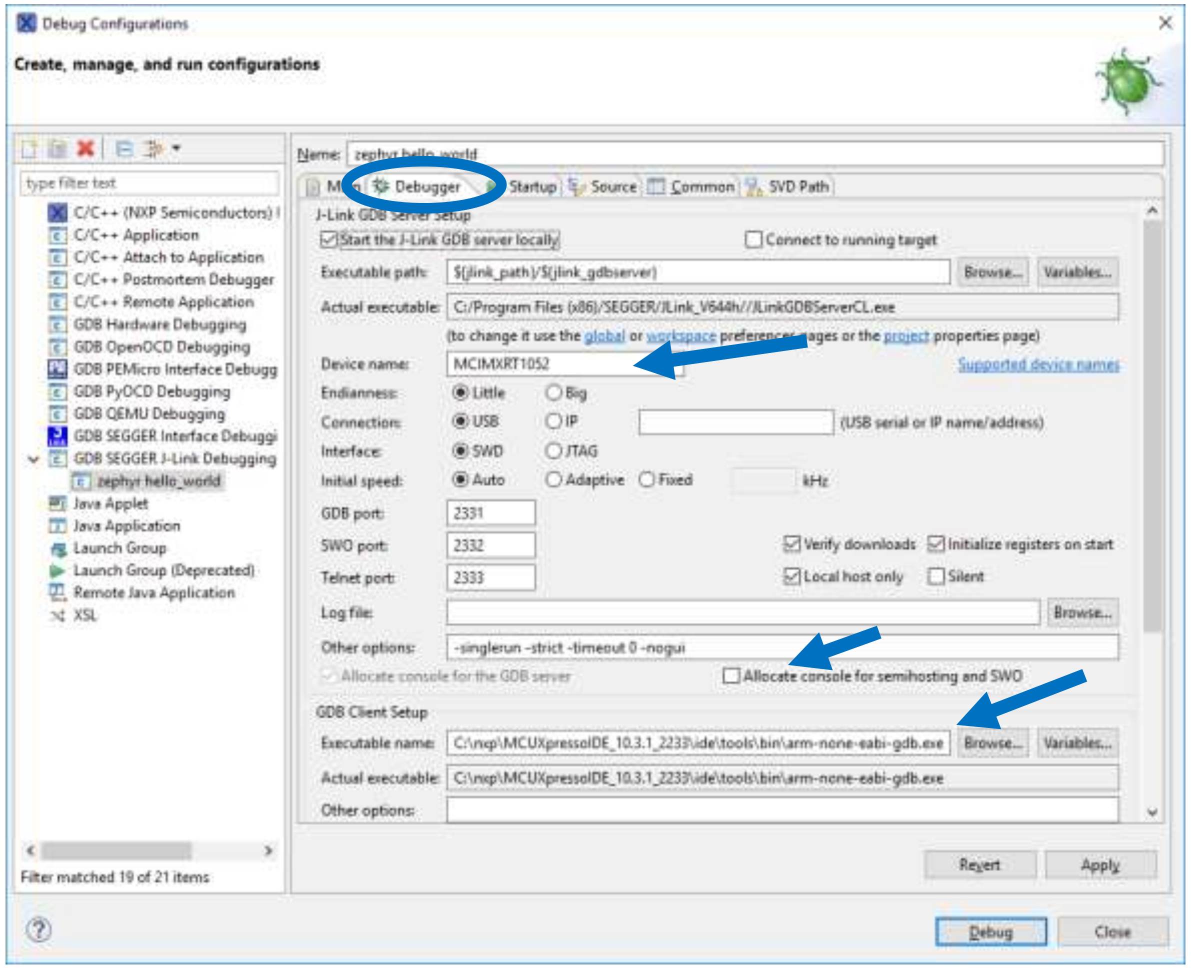
Build (if required) before launching

Build Configuration: Use Active

( ) Enable auto build
( ) Disable auto build
(●) Use workspace settings
Configure Workspace Settings...

- Select the Main tab and configure the following settings:
- Project: hello_world@build
- C/C++ Application: zephyr/zephyr.elf

# J-Link Debug Configuration: Debugger



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ (NXP Semiconductors) I
- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB PEMicro Interface Debugg
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER Interface Debuggi
- GDB SEGGER J-Link Debugging
  - zephyr hello_world
- Java Applet
- Java Application
- Launch Group
- Launch Group (Deprecated)
- Remote Java Application
- XSL

Filter matched 19 of 21 items

Name: zephyr hello_world

Main | Debugger | Startup | Source | Common | SVD Path

J-Link GDB Server Setup

[✓] Start the J-Link GDB server locally  [ ] Connect to running target

Executable path: ${jlink_path}/${jlink_gdbserver}

Actual executable: C:/Program Files (x86)/SEGGER/JLink_V644h//JLinkGDBServerCL.exe

(to change it use the global or workspace preferences pages or the project properties page)

Device name: MCIMXRT1052  Supported device names

Endianness: [● Little] ○ Big

Connection: [● USB] ○ IP  (USB serial or IP name/address)

Interface: [● SWD] ○ JTAG

Initial speed: [● Auto] ○ Adaptive ○ Fixed  kHz

GDB port: 2331

SWO port: 2332  [✓] Verify downloads  [✓] Initialize registers on start

Telnet port: 2333  [✓] Local host only  [ ] Silent

Log file:

Other options: -singlerun -strict -timeout 0 -nogui

Allocate console for the GDB server  [ ] Allocate console for semihosting and SWO

GDB Client Setup

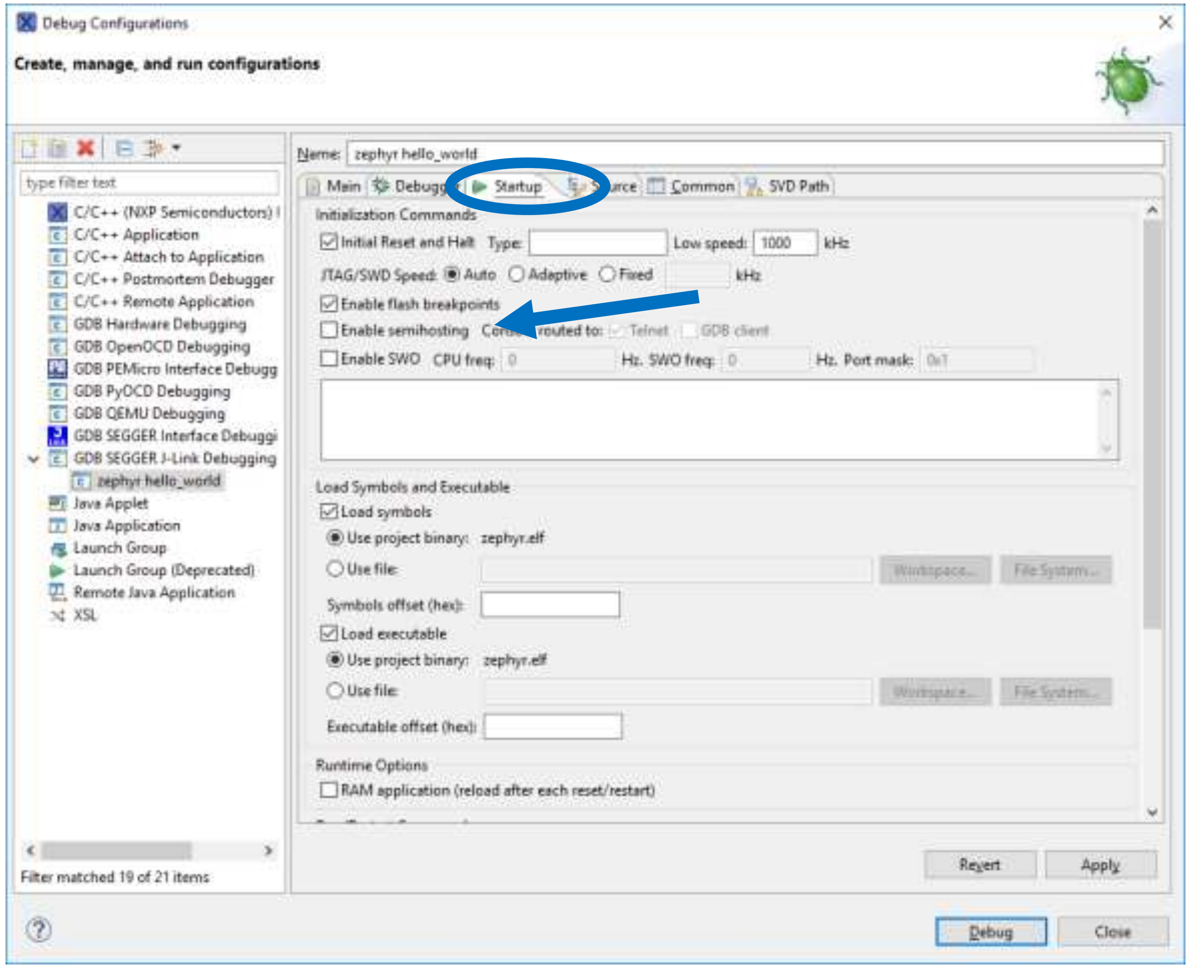
Executable name: C:\nxp\MCUXpressoIDE_10.3.1_2233\ide\tools\bin\arm-none-eabi-gdb.exe

Actual executable: C:\nxp\MCUXpressoIDE_10.3.1_2233\ide\tools\bin\arm-none-eabi-gdb.exe

Other options:

- Select the Debugger tab and configure the following settings:
- Device name: MCIMXRT1052
- GDB Client Executable name: C:\nxp\MCUXpressoIDE_10.3.1_2233\ide\tools\bin\arm-none-eabi-gdb.exe
- Uncheck Allocate console for semihosting and SWO

# J-Link Debug Configuration: Startup



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ (NXP Semiconductors) I
- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB PEMicro Interface Debugg
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER Interface Debuggi
- GDB SEGGER J-Link Debugging
  - zephyr hello_world
- Java Applet
- Java Application
- Launch Group
- Launch Group (Deprecated)
- Remote Java Application
- XSL

Filter matched 19 of 21 items

Name: zephyr hello_world

Main | Debugger | Startup | Source | Common | SVD Path

Initialization Commands

- [✓] Initial Reset and Halt Type: Low speed: 1000 kHz
- JTAG/SWD Speed: [✓ Auto] Adaptive Fixed kHz
- [✓] Enable flash breakpoints
- [ ] Enable semihosting Console routed to: [✓] Telnet [ ] GDB client
- [ ] Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask: 0x1

Load Symbols and Executable

- [✓] Load symbols
- [✓ Use project binary: zephyr.elf]
- Use file: Workspace... File System...
- Symbols offset (hex):
- [✓] Load executable
- [✓ Use project binary: zephyr.elf]
- Use file: Workspace... File System...
- Executable offset (hex):

Runtime Options

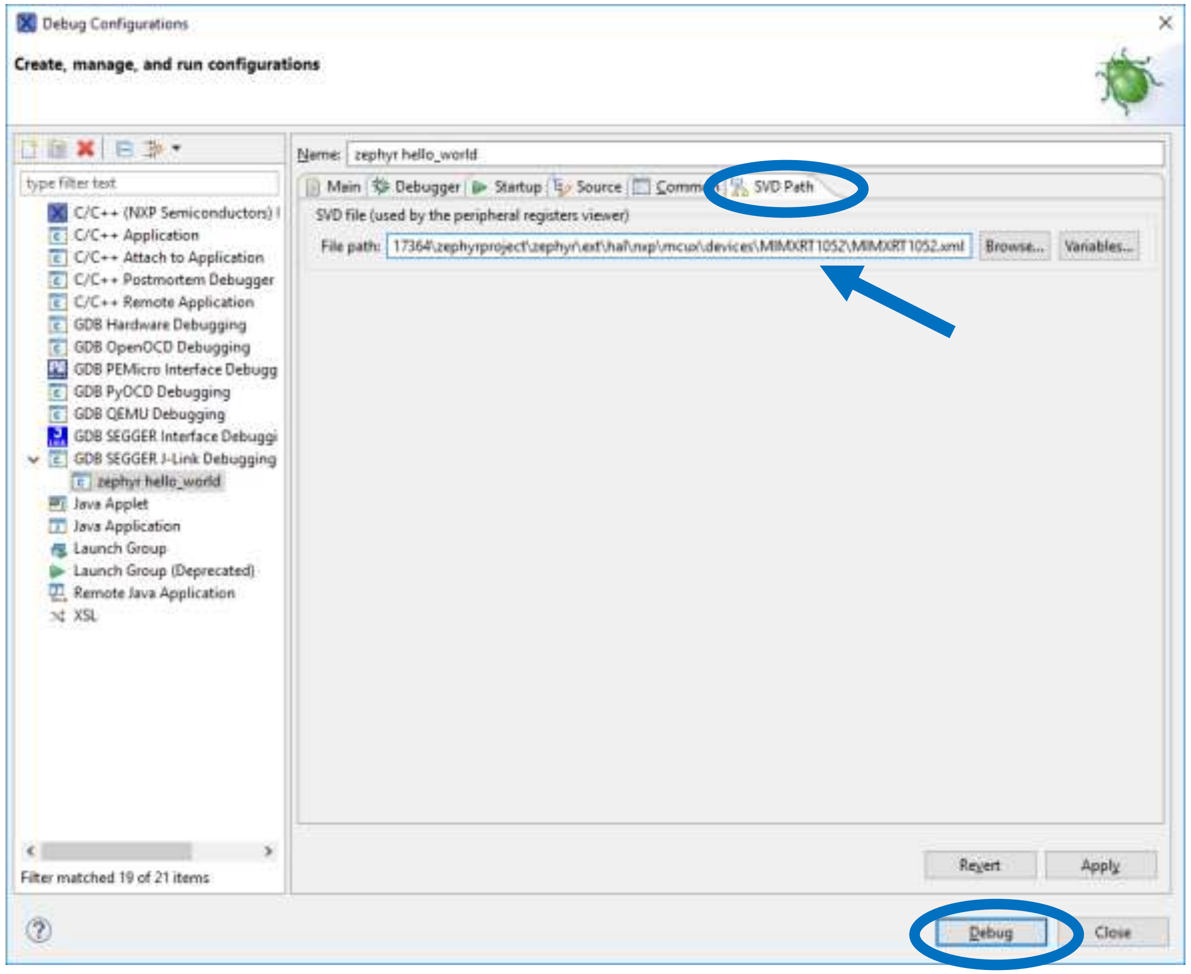
- [ ] RAM application (reload after each reset/restart)

Revert Apply

Debug Close

- Select the Startup tab
- Uncheck Enable semihosting
- Uncheck Enable SWO

## J-Link Debug Configuration: SVD Path

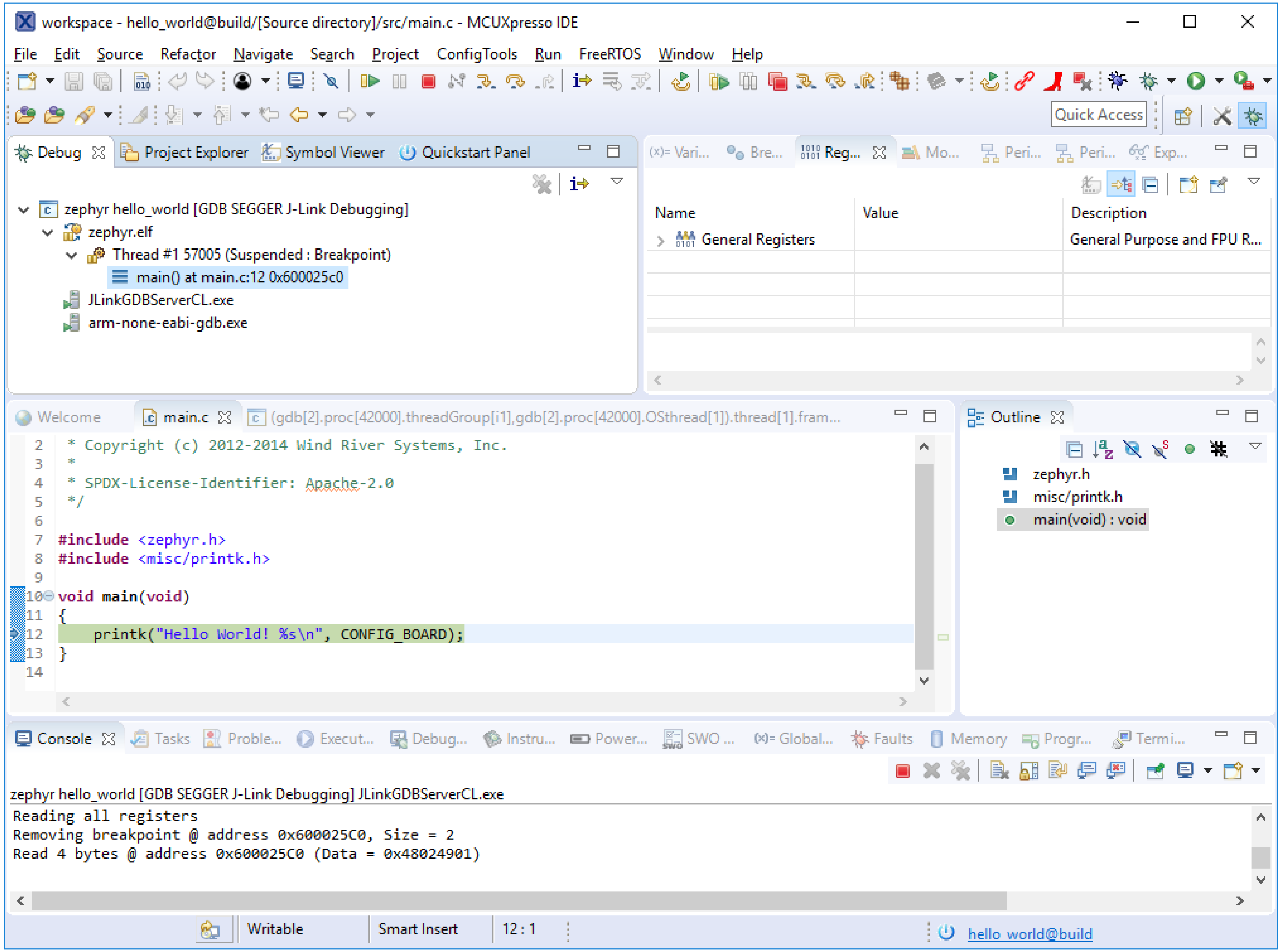


Name: zephyr hello_world

SVD file (used by the peripheral registers viewer)

File path: 17364\zephyrproject\zephyr\ext\hal\nxp\mcux\devices\MIMXRT1052\MIMXRT1052.xml

- Select the SVD Path tab and configure the following settings:
- SVD file path: C:\Users\NXPTraining\zephyrproject\zephyr\ext\hal\nxp\mcux\devices\MIMXRT1052\MIMXRT1052.xml
- Select Debug to start the debugger!



```
zephyr hello_world [GDB SEGGER J-Link Debugging]
  zephyr.elf
    Thread #1 57005 (Suspended : Breakpoint)
      main() at main.c:12 0x600025c0
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb.exe
```

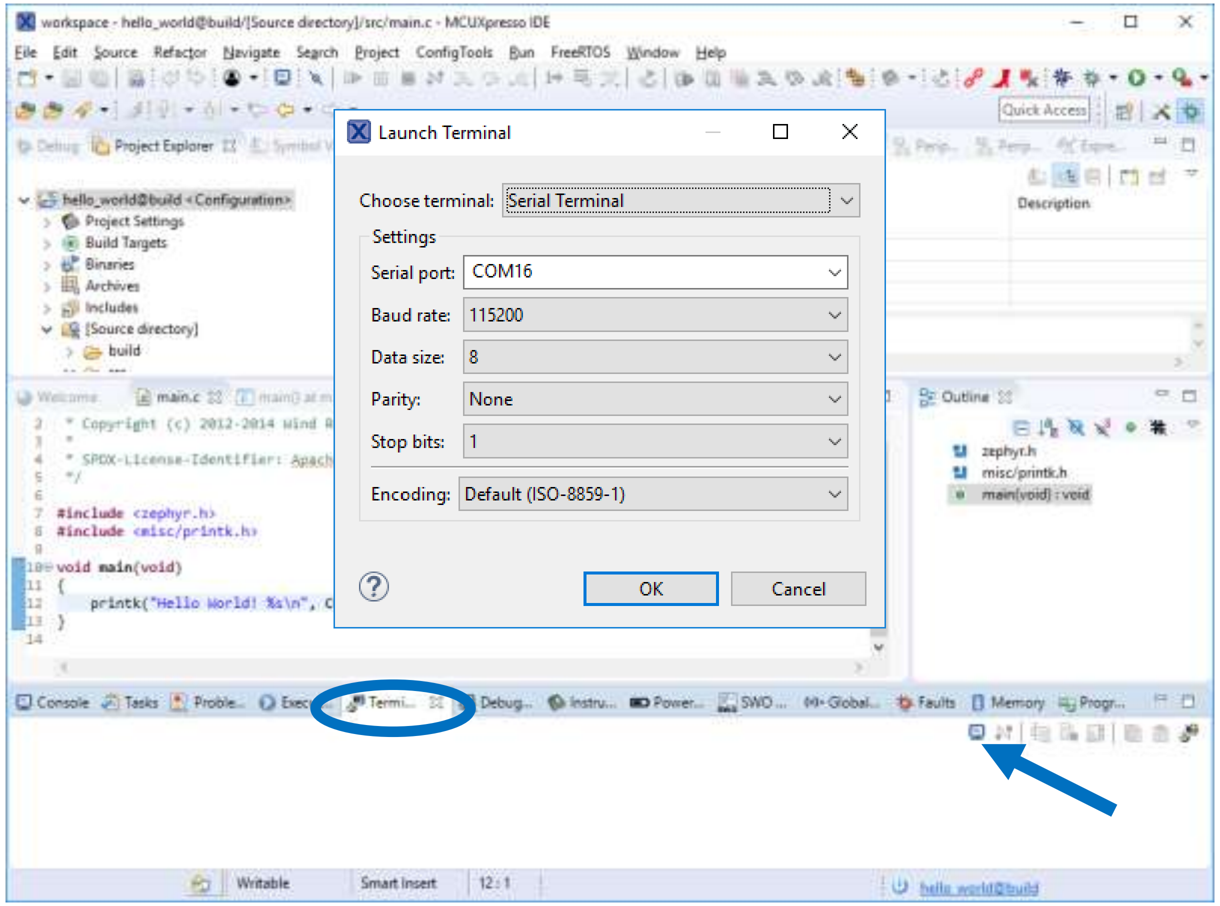
| Name | Value | Description |
|---|---|---|
| General Registers | | General Purpose and FPU R... |

```
 * Copyright (c) 2012-2014 Wind River Systems, Inc.
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <zephyr.h>
#include <misc/printk.h>

void main(void)
{
    printk("Hello World! %s\n", CONFIG_BOARD);
}
```

zephyr hello_world [GDB SEGGER J-Link Debugging] JLinkGDBServerCL.exe

```
Reading all registers
Removing breakpoint @ address 0x600025C0, Size = 2
Read 4 bytes @ address 0x600025C0 (Data = 0x48024901)
```

# Open a Serial Terminal



workspace - hello_world@build/[Source directory]/src/main.c - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run FreeRTOS Window Help
Quick Access
Project Explorer
hello_world@build <Configuration>
Project Settings
Build Targets
Binaries
Archives
Includes
[Source directory]
build
main.c
Outline
zephyr.h
misc/printk.h
main(void) : void
Description
Console Tasks Proble... Exec... Termi... Debug... Instru... Power... SWO... Global... Faults Memory Progr...
Writable
Smart Insert
12 : 1
hello_world@build

```
 * Copyright (c) 2012-2014 Wind R
 *
 * SPDX-License-Identifier: Apach
 */

#include <zephyr.h>
#include <misc/printk.h>

void main(void)
{
	printk("Hello World! %s\n", C
}
```

Launch Terminal

Choose terminal: Serial Terminal

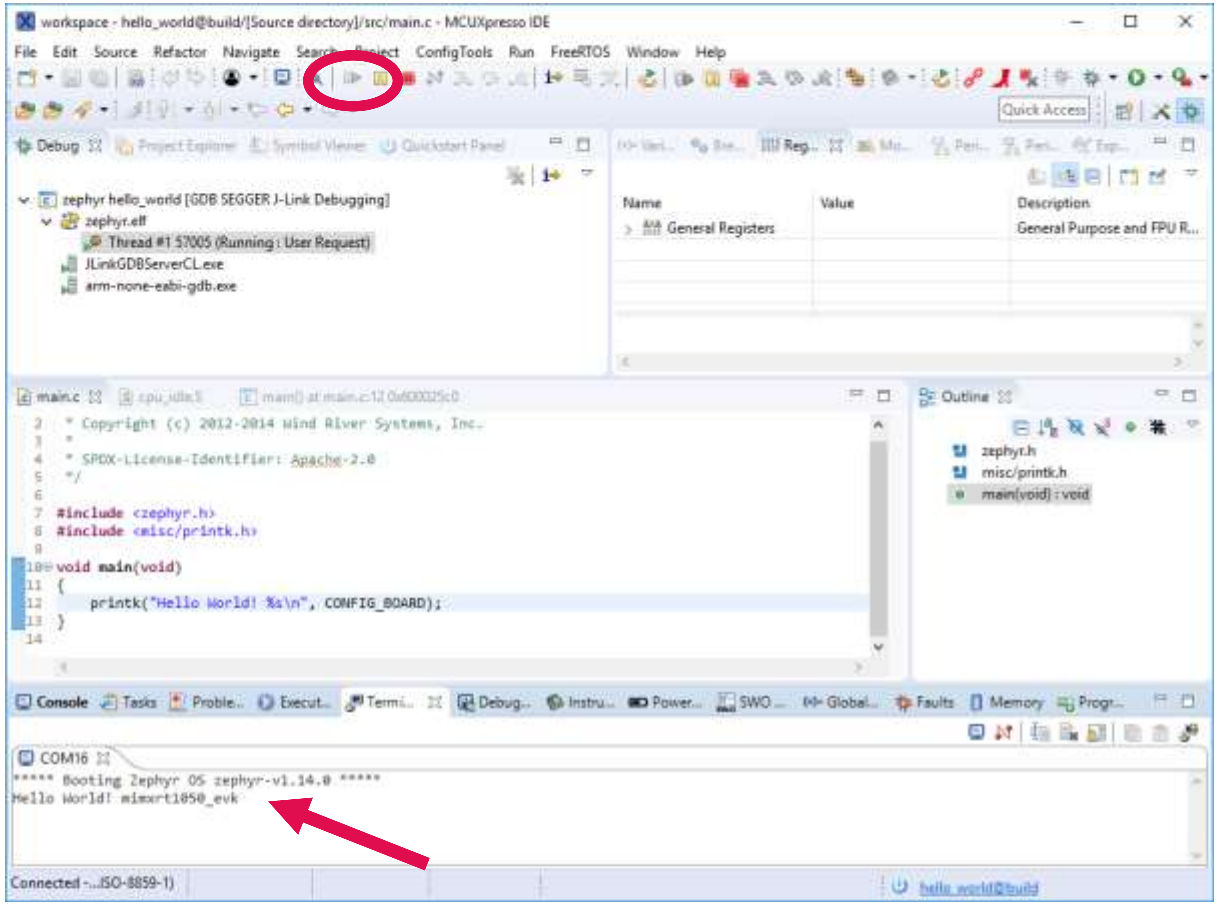
Settings

| | |
|---|---|
| Serial port: | COM16 |
| Baud rate: | 115200 |
| Data size: | 8 |
| Parity: | None |
| Stop bits: | 1 |
| Encoding: | Default (ISO-8859-1) |

OK Cancel

- From the Window menu, select Show View->Terminal
- Select the Terminal tab in the bottom third of the window
- Select Open a Terminal
- Enter serial port settings as shown (COM number may be different)

## Run the Application


workspace - hello_world@build/[Source directory]/src/main.c - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run FreeRTOS Window Help
Quick Access
Debug
Project Explorer
Symbol Viewer
Quickstart Panel
Name
Value
Description
General Registers
General Purpose and FPU R...
main.c
cpu_idle.S
main() at main.c:12 0x600025c0
Outline
zephyr.h
misc/printk.h
main(void) : void
Console Tasks Proble... Execut... Termi... Debug... Instru... Power... SWO ... Global... Faults Memory Progr...
COM16
Connected - ...ISO-8859-1)
hello_world@build

```
zephyr hello_world [GDB SEGGER J-Link Debugging]
  zephyr.elf
    Thread #1 57005 (Running : User Request)
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb.exe
```

```
 * Copyright (c) 2012-2014 Wind River Systems, Inc.
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <zephyr.h>
#include <misc/printk.h>

void main(void)
{
	printk("Hello World! %s\n", CONFIG_BOARD);
}
```

```
***** Booting Zephyr OS zephyr-v1.14.0 *****
Hello World! mimxrt1050_evk
```

- Select Resume to run the application

- See in the terminal:

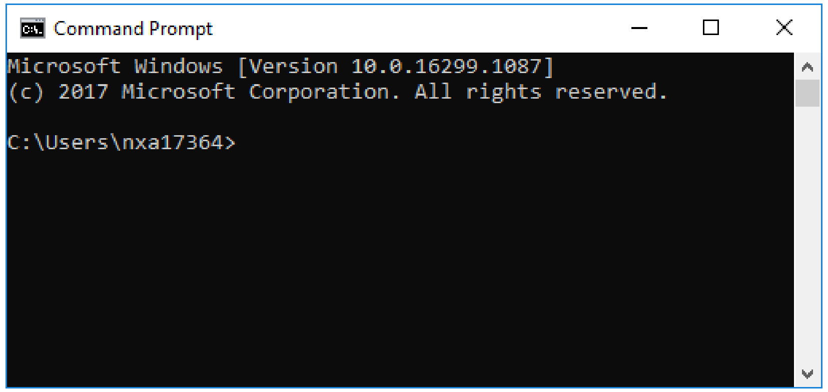
```
Hello World! mimxrt1050_evk
```

## Exercise #3: Display and Graphics with LittlevGL Integration

Build and flash an LCD application

# Build and Flash LittlevGL

- Open a normal command prompt



```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\nxa17364>
```

- Set up the build environment

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

- Build the LittlevGL sample application

```
> west build -b mimxrt1050_evk -d build\lvgl samples\gui\lvgl
```

- Flash it to the board

```
> west flash -d build\lvgl
```

- See “Hello world!” on the LCD

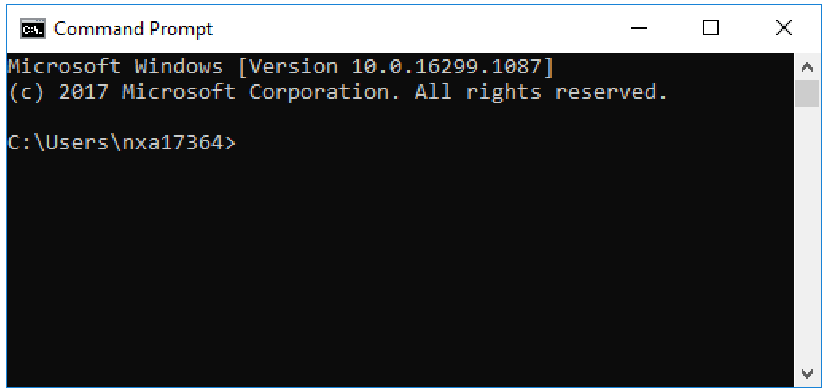
# Exercise #4: Configuration and Memory Footprint

Examine flash/ram footprint with rom_report and ram_report

Change the configuration and rebuild

# Examine Memory Footprint

- Open a normal command prompt



```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\nxa17364>
```

- Set up the build environment

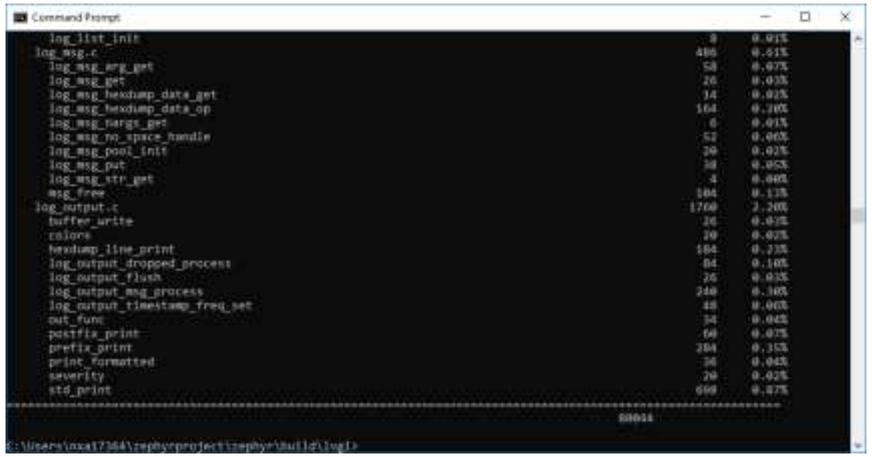
```
> cd %userprofile%\zephyrproject\zephyr
> zephyr-env.cmd
```

- Move to the LittlevGL sample application build directory

```
> cd build\lvgl
```

- Run reports to see flash and ram memory footprints

```
> ninja rom_report
> ninja ram_report
```



```
    log_list_init                                                        8     0.01%
  log_msg.c                                                            486     0.61%
    log_msg_arg_get                                                     58     0.07%
    log_msg_get                                                         26     0.03%
    log_msg_hexdump_data_get                                            14     0.02%
    log_msg_hexdump_data_op                                            164     0.20%
    log_msg_nargs_get                                                    6     0.01%
    log_msg_no_space_handle                                             52     0.06%
    log_msg_pool_init                                                   20     0.02%
    log_msg_put                                                         38     0.05%
    log_msg_str_get                                                      4     0.00%
    msg_free                                                           104     0.13%
  log_output.c                                                        1760     2.20%
    buffer_write                                                        26     0.03%
    colors                                                              20     0.02%
    hexdump_line_print                                                 184     0.23%
    log_output_dropped_process                                          84     0.10%
    log_output_flush                                                    26     0.03%
    log_output_msg_process                                             240     0.30%
    log_output_timestamp_freq_set                                       48     0.06%
    out_func                                                            34     0.04%
    postfix_print                                                       60     0.07%
    prefix_print                                                       284     0.35%
    print_formatted                                                     36     0.04%
    severity                                                            20     0.02%
    std_print                                                          698     0.87%
========================================================================================
                                                                 80044

C:\Users\nxa17364\zephyrproject\zephyr\build\lvgl>
```

# Change Configuration and Rebuild

- Open samples\gui\lvgl\prj.conf in a text editor and disable logging

```
CONFIG_LOG=n
```

- Open a normal command prompt

- Set up the build environment

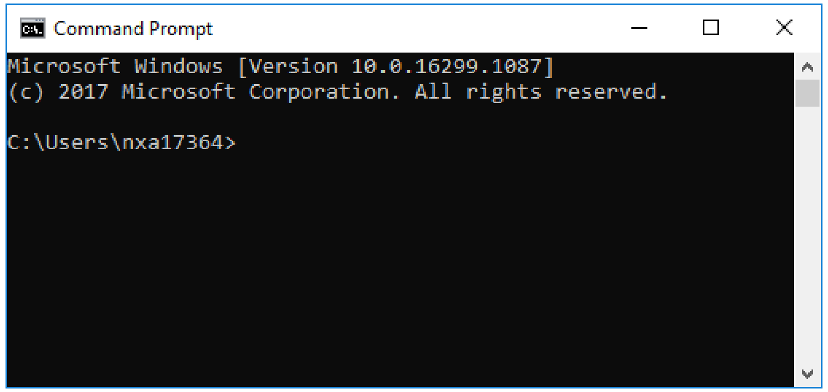
```
> cd %userprofile%\zephyrproject\zephyr
> zephyr-env.cmd
```

- Rebuild the LittlevGL sample application with the new configuration

```
> west build -d build\lvgl -c
```

- Move to the LittlevGL sample application build directory and rerun reports

```
> cd build\lvgl
> ninja rom_report
> ninja ram_report
```



```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\nxa17364>
```

| CONFIG_LOG | ROM (B) | RAM (B) |
|---|---:|---:|
| Y | 80044 | 590628 |
| N | 72376 | 588425 |
| **Delta** | **7668** | **2203** |

# Backup

## References

- https://docs.zephyrproject.org/latest/boards/index.html
- https://github.com/zephyrproject-rtos/zephyr/releases/tag/zephyr-v1.14.0
- https://docs.zephyrproject.org/1.14.0/getting_started/index.html#build-and-run-an-application
- https://docs.zephyrproject.org/1.14.0/application/index.html#eclipse-debugging

SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2019 NXP B.V.