Mendel University in Brno
Faculty of Business and Economy

MicroPython Utilizing Zephyr Port and NXP FRDM-MCXN947

Bachelor thesis

Supervisor:
Ing. Jan Kolomazník, Ph.D.

Dmitrii Titarenko

Brno 2025

Acknowledgment

I would like to express my gratitude to everyone who contributed to completion of this thesis.
I would like to thank my thesis supervisor, Ing. Jan Kolomazník, Ph.D. , for his guidance and support. Furthermore, I express my gratitude to Ph.D. Zbynek Fedra for guiding me during the creation of this thesis.
I am also very grateful to all the professors in the Department of Informatics at Mendel University for their teaching, openness, and constant support throughout my study.
A big thank you to family members and friends for their support and encouragement.
Finally, I express gratitude to the people who directly or indirectly contributed to the creation of this work. To those who came before me and on whose shoulders I stand.

Declaration

Brno 03.08.2025 ....

signature

Abstract

Titarenko, D. MicroPython Utilizing Zephyr Port and NXP FRDM-MCXN947. Bachelor thesis. Brno, 2025.

This thesis explores MicroPython support on Zephyr RTOS using the NXP FRDM-MCXN947 development board. Zephyr RTOS itself provides extended support and easy-to-use APIs to many embedded devices and their peripherals, but the support of MicroPython remains limited and not consistent with MicroPython ports developed for target devices. This work analyzes the current limitations of MicroPython on Zephyr RTOS and selects specific functionality for implementation. The implementation involves initializing Zephyr and MicroPython development environments adjusted to FRDM-MCXN947, conducting functional testing of the development board peripherals with Zephyr and MicroPython environment, and comparing the comparability of native and Zephyr ports of MicroPython.

Key words: MicroPython, thesis, Zephyr RTOS, FRDM-MCXN947

Contents

1 Introduction

The world of microcontrollers and embedded devices continues to grow, and such devices become more common with every day. Even though they are seldom noticeable we more often might find ourselves surrounded by them. From home appliances, to cars, to factory machines to city-wide networks embedded devices reach wide and deep in our lives.

But with the growing count of embedded devices grows complexity of functions they implement. Hence arises a need for Operating Systems(OSs) to manage sets of complex programs and provide a layer of abstraction to ease development in such constrain but demanding environments.

To cover demand of an Operating System in embedded devices the Zephyr Real Time Operating System (RTOS) was created. Zephyr RTOS is an open-source operating system with build-in security and optimization for resource limited devices. Zephyr kernel supports ARM, Intel x86, ARC, RISC-V, Nios II, Tensilica Xtensa and large number of development boards, among others NXPs FRDM-MCXN947. ALso Zephyr has rich API that allows developers to write high-level code for embedded devices.

Writing software for embedded devices is still a complex task regardless of what underlying technologies are used. Writing it in a language such as C adding an additional complexity due to need of managing program memory allocation and deallocation by hand. Leaving unhandled memory sector could lead to memory leaks or worse opens an opportunity for an attacker to execute malicious code on an embedded device. The consequences of such problems become much grater when occurring in the embedded world. MicroPython is an optimized subset of Python 3 programming language for embedded devices. It aims to ease writing software by managing memory using its garbage collector system, using easy to read Python-like syntax and providing various modules to enable work with different peripherals. Additionally, MicroPython allows for code portability, meaning that code written for FRDM-MCXN947 could be ported and ran on ESP32 with minimal updates to code.

But MicroPython does not support every single device straightaway – ported versions of MicroPython are submitted to the MicroPython repository, by a manufacturer or an enthusiasts. Later submitted port will be reviewed and tested by MicroPython maintainers, which is a lengthy process, for example MicroPython port for Zephyr was under review for 2 years.

By combining Zephyr RTOS and MicroPython in one technological stack the best of both technologies could be utilized. Potential exists for developers to write highly readable, easy-to-understand and efficient code with MicroPython that make use of various hardware support introduced by Zephyr RTOS. Yet the state of this development environment is not yet firm and have plenty of rough edges and unapparent problems that could arise during the process of software development.

The aim of this thesis is to construct a method for configuring the development environment for MicroPython, Zephyr RTOS and FRDM-MCXN947 development board, provide insight of compatibility challenges of both platforms and propose potential solutions for extending MicroPython port to Zephyr RTOS. The finding of this thesis will contribute to understanding of the state of both platforms and their integration, informing future developers and increasing usability of MicroPython and Zephyr.

2 Background

This chapter introduces the reader to an information about Zephyr RTOS, MicroPython and FRDM- MCXN947 board that is needed for understanding this thesis.

2.1 Zephyr RTOS

Zephyr is an Operation System designed for resource-constrained and embedded system from simple sensors to smart industrial embedded solutions. It supports a broad list of embedded devices, development boards and peripherals. Zephyr offers extensive number of features and services including multi-threading, inter-thread data passing, inter-thread synchronization, dynamic memory allocation, interrupt service, power management, networking. Zephyr project is open-source, distributed under Apache 2.0 license and was created under Linux Foundation organization.(Zephyr Project – Intorduction, 2024)

2.1.1 West

West is a part of Zephyr's tool-chain used for building and configuring. West can initiate Zephyr workspace from official upstream repository, update or change version of a local Zephyr workspace to any version in official repository, build Zephyr application from source, flash built application to a board.(Zephyr Project – West (Zephyr's meta-tool), 2024)

2.1.2 Kconfig

Kconfig is Zephyr's kernel, drivers and subsystems configuration system that allows to configure Zephyr at a build time. Kconfig goal is to enable configuration without introducing changes to source code.

The initial board configuration can be found in **<board>_defconfig** files. For example configuration file for FRDM-MCXN947 is located at **boards/nxp/frdm_mcxn947/ frdm_mcxn947_mcxn947_cpu0_defconfig**. The board configuration for NXPs' FRDM-MCXN947 is as follows:

```
CONFIG_CONSOLE=y
CONFIG_UART_CONSOLE=y
CONFIG_SERIAL=y
CONFIG_UART_INTERRUPT_DRIVEN=y
CONFIG_GPIO=y
CONFIG_PINCTRL=y
CONFIG_ARM_MPU=y
CONFIG_HW_STACK_PROTECTION=y
CONFIG_TRUSTED_EXECUTION_SECURE=y
```

Kconfig values can be set to a **<board>_defconfig** files, temporarily with terminal graphical interfaces or with a **prj.conf** file at application level which overrides the initial configuration during application build.(Zephyr Project – Configuration System (Kconfig), 2022)

2.1.3 Devicetree

Devicetree is a data structure for describing hardware. It's a community driven standard that is heavily used in Zephyr project. In Zephyr devicetrees are usually build inherently meaning that for example FRDM-MCXN947 has a devicetree configuration **board/nxp/frdm_mcxn947/frdm_mcxn947_mcxn947_cpu0.dts** which mainly enables peripheral devices, but includes FRDM-MCXN947 specific configuration from **frdm_mcxn947.dtsi** (include file), which in turn includes **frdm_mcxn947-pinctrl.dtsi** file that mostly defines pinmux groups. Additionally the **frdm_mcxn947_mcxn947_cpu0.dts** includes **nxp_mcxn94x.dtsi** file that defines memory ranges for SRAM, FLEXSPI and peripherals and includes **nxp_mcxn94x_common.dtsi** include file where most of devices including CPU, GPIO, CTIMER and others are defined and assigned memory ranges.(devicetree.org, Devicetree Specification Release v0.4, 2023)

Same as Kconfig Devicetrees can be overwritten or have some specific devices configured differently with **overlay** files, which as well needs to be placed in build directory, from there **west tool** will use it to edit the Devicetree configuration.

3 References

Zephyr Project – Intorduction [onlne], 2024 Available from: https://docs.zephyrproject.org/latest/introduction/index.html.

Zephyr Project – West (Zephyr's meta-tool) [online], 2024 Alailable from: https://docs.zephyrproject.org/latest/develop/west/index.html.

Zephyr Project – Configuration System (Kconfig) [onlne], 2022 Available from: https://docs.zephyrproject.org/latest/build/kconfig/index.html.

NXP, MCX Nx4x Reference Manual, 2025 MCXNX4XRM.

devicetree.org, Devicetree Specification Release v0.4, 2023.