An OS to build, deploy and securely manage billions of devices

Latest News:

Apache Mynewt 1.10.0, Apache NimBLE 1.5.0 (/download) released (May 6, 2022)

Porting Mynewt OS

This chapter describes how to adapt the Mynewt OS to different platforms.

- Description
- Board Support Package (BSP) Dependency
- MCU Dependency

- MCU HAL
- CPU Core Dependency

Description

The Mynewt OS is a complete multi-tasking environment with scheduler, time control, buffer management, and synchronization objects. it also includes libraries and services like console, command shell, image manager, bootloader, and file systems etc.

Thee majority of this software is platform independent and requires no intervention to run on your platform, but some of the components require support from the underlying platform.

The platform dependency of these components can fall into several categories:

- **CPU Core Dependencies** – Specific code or configuration to operate the CPU core within your target platform
- **MCU Dependencies** – Specific code or configuration to operate the MCU or SoC within your target platform
- **BSP Dependencies** – Specific code or configuration to accommodate the specific layout and functionality of your target platform

Board Support Package (BSP) Dependency

With all of the functionality provided by the core, MCU, and MCU HAL (Hardware Abstraction Layer), there are still some things that must be specified for your particular system. This is provided in Mynewt to allow you the flexibility to design for the exact functionality, peripherals and features that you require in your product.

In Mynewt, these settings/components are included in a Board Support Package (BSP). The BSP contains the information specific to running Mynewt on a target platform or hardware board. Mynewt supports some common open source hardware as well as the development boards for some common MCUs. These development systems might be enough for you to get your prototype up and running, but when building a product you are likely going to have your own board which is slightly different from those already supported by Mynewt.

For example, you might decide on your system that 16 Kilobytes of flash space in one flash device is reserved for a flash file system. Or on your system you may decide that GPIO pin 5 of the MCU is connected to the system LED. Or you may decide that the OS Tick (the underlying time source for the OS) should run slower than the defaults to conserve battery power. These types of behaviors are specified in the BSP.

The information provided in the BSP (what you need to specify to get a complete executable) can vary depending on the MCU and its underlying core architecture. For example, some MCUs have dedicated pins for UART, SPI etc, so there is no configuration required in the BSP when using these peripherals. However some MCUs have a pin multiplexor that allows the UART to be mapped to several different pins. For these MCUs, the BSP must specify if and where the UART pins should appear to match the hardware layout of your system.

- If your BSP is already supported by Mynewt, there is no additional BSP work involved in porting to your platform. You need only to set the `bsp` attribute in your Mynewt target using the newt command tool (../../../newt/index.html).
- If your BSP is not yet supported by Mynewt, you can add support following the instructions on BSP Porting (port_bsp.html).

MCU Dependency

Some OS code depends on the MCU or SoC that the system contains. For example, the MCU may specify the potential memory map of the system - where code and data can reside.

- If your MCU is already supported by Mynewt, there is no additional MCU work involved in porting to your platform. You need only to set the `arch` attribute in your Mynewt target using the newt command tool (../../../newt/index.html).
- If your MCU is not yet supported by Mynewt, you can add support following the instructions in Porting Mynewt to a new MCU (port_mcu.html).

MCU HAL

Mynewt's architecture supports a hardware abstraction layer (HAL) for common on or off-chip MCU peripherals such as GPIO, UARTs, flash memory etc. Even if your MCU is supported for the core OS, you may find that you need to implement the HAL functionality for a new peripheral. For a description of the HAL abstraction and implementation information, see the HAL API (../../modules/hal/hal.html)

CPU Core Dependency

Some OS code depends on the CPU core that your system is using. For example, a given CPU core has a specific assembly language instruction set, and may require special cross compiler or compiler settings to use the appropriate instruction set.

- If your CPU architecture is already supported by Mynewt, there is no CPU core work involved in porting to your platform. You need only to set the `arch` and `compiler` attributes in your Mynewt target using the newt command tool (../../../newt/index.html).
- If your CPU architecture is not supported by Mynewt, you can add support following the instructions on Porting Mynewt to a new CPU Architecture (port_cpu.html).

Previous: Split Images (../../modules/split/split.html) Next: BSP Porting (port_bsp.html)