



# Armv8-R AArch64 Reference Software Library

Version 1.0

**Non-Confidential**

Copyright © 2025 Arm Limited (or its affiliates).  
All rights reserved.

**Issue 01**

110457\_0100\_01\_en



# Armv8-R AArch64 Reference Software Library

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.

## Release information

### Document history

Issue	Date	Confidentiality	Change
0100-01	14 April 2025	Non-Confidential	Initial release

## Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm’s view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm

makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

## Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

## Product Status

The information in this document is Final, that is for a developed product.

## Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

## Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email [terms@arm.com](mailto:terms@arm.com).

# Contents

- 1. Armv8-R AArch64 reference software library overview..... 6
- 2. Linux..... 7
  - 2.1 Release 1 – Single Core..... 7
  - 2.2 Release 2 – SMP..... 10
  - 2.3 Release 3 – UEFI..... 13
- 3. Xen Hypervisor..... 20
  - 3.1 Release 4 – Xen..... 20
  - 3.2 Release 5 – InterVM communication with Linux containers..... 20
- 4. Zephyr..... 21

# 1. Armv8-R AArch64 reference software library overview

Welcome to the Armv8-R AArch64 reference software library.

This is a collection of example software compositions for the Armv8-R AEM FVP model, an implementation of the [Armv8-R AArch64 architecture](#), which is available [here](#).

The Open Source example software coupled with the Armv8-R AEM FVP enable the exploration of the general Armv8-R AArch64 Architecture as well as its specific implementation – Cortex-R82.

## 2. Linux

A software stack to boot to a Linux command prompt via a basic bootloader on the Armv8-R AEM FVP model.

The platform software is provided via a Yocto package which contains all the instructions necessary to fetch and build the source as well as where to download the model and launch the example.

Fixed Virtual Platforms (FVP) are complete simulations of an Arm system, including processor, memory and peripherals. These are set out in a “programmer’s view”, which gives you a comprehensive model on which to build and test your software.

The Armv8-R AEM FVP is a free of charge Armv8-R Fixed Virtual Platform. It supports the latest Armv8-R feature set.

This BSP implements a reference stack for the AArch64 support in the R-class, first announced with the [Cortex-R82 processor](#).

The Fast Models Fixed Virtual Platforms (FVP) Reference Guide can be found [here](#).

### 2.1 Release 1 – Single Core

BSP support:

- The fvp-baser-aemv8r64 Yocto MACHINE supports the following BSP components on the Yocto hardknott release branch, where a standard Linux kernel can be built and run (instructions below):
  - boot-wrapper-aarch64
  - Linux kernel: linux-yocto-5.10

The following instructions provide a quick start guide, explaining how to build and run the software:

#### Host environment setup

The following instructions have been tested on hosts running Ubuntu 18.04 and Ubuntu 20.04.

Install the required packages for the build host:

<https://docs.yoctoproject.org/3.3.1/singleindex.html#required-packages-for-the-build-host>

Install the kas setup tool for bitbake based projects:

```
$ pip3 install --user kas
```

For more details on kas, see <https://kas.readthedocs.io/>



The host machine should have at least 50 GBytes of free disk space for the next steps to work correctly.

## Fetch sources

Fetch the meta-arm repository into a build directory:

```
$ mkdir -p ~/fvp-baser-aemv8r64-build
$ cd ~/fvp-baser-aemv8r64-build
$ git clone --branch hardknott https://git.yoctoproject.org/git/meta-arm
```

## Build

Building with the standard Linux kernel:

```
$ cd ~/fvp-baser-aemv8r64-build
$ kas build meta-arm/kas/fvp-baser-aemv8r64-bsp.yml
```

## Networking

To enable networking on the FVP via a host network interface, you will need to install the following package(s):

- Ubuntu 18.04:

```
$ sudo apt-get install libvirt-bin
```

- Ubuntu 20.04:

```
$ sudo apt-get install libvirt-dev libvirt-daemon qemu-kvm libvirt-daemon-system
libvirt-clients bridge-utils
```

Once that is installed for your OS version, setup tap0 using the following commands:

```
$ sudo virsh net-start default
$ sudo ip tuntap add dev tap0 mode tap user $(whoami)
$ sudo ifconfig tap0 0.0.0.0 promisc up
$ sudo brctl addif virbr0 tap0
```

To clean up the tap0 interface on the host use the following commands:

```
$ sudo brctl delif virbr0 tap0
$ sudo ip link set virbr0 down
$ sudo brctl delbr virbr0
$ sudo virsh net-destroy default
$ sudo ip link delete tap0
```

## Run

To run the Fixed Virtual Platform simulation tool you must download “Armv8-R AEM FVP” from Arm developer (this might require the user to register) from this address:



<https://developer.arm.com/tools-and-software/simulation-models/fixed-virtual-platforms/arm-ecosystem-models>

and install it on your host PC.

To run an image after the build is done:

```
$ export YOCTO_DEPLOY_IMGS_DIR=~/.fvp-baser-aemv8r64-bsp/build/tmp/deploy/images/
fvp-baser-aemv8r64/"
$ cd <path-to-AEMv8R_base_pkg>/models/Linux64_GCC-6.4/
$ ./FVP_Baser_AEMv8R \
  -C bp.dram_metadata.init_value=0 \
  -C bp.dram_metadata.is_enabled=true \
  -C bp.dram_size=8 \
  -C bp.exclusive_monitor.monitor_access_level=1 \
  -C bp.pl011_uart0.unbuffered_output=1 \
  -C bp.pl011_uart0.untimed_fifos=true \
  -C bp.refcounter.non_arch_start_at_default=1 \
  -C bp.smc_91c111.enabled=0 \
  -C bp.vc_sysregs.mmbSiteDefault=0 \
  -C cache_state_modelled=true \
  -C cluster0.gicv3.cputif-mmap-access-level=2 \
  -C cluster0.gicv3.SRE-enable-action-on-mmap=2 \
  -C cluster0.gicv3.SRE-EL2-enable-RAO=1 \
  -C cluster0.gicv3.extended-interrupt-range-support=1 \
  -C cluster0.has_aarch64=1 \
  -C cluster0.NUM_CORES=4 \
  -C cluster0.stage12_tlb_size=512 \
  -C gic_distributor.GICD_CTLR-DS-1-means-secure-only=1 \
  -C gic_distributor.GITS_BASER0-type=1 \
  -C gic_distributor.ITS-count=1 \
  -C gic_distributor.ITS-hardware-collection-count=1 \
  -C gic_distributor.has-two-security-states=0 \
  -C pctl.startup=0.0.0.* \
  -C bp.virtio_net.enabled=1 \
  -C cache_state_modelled=0 \
  -C bp.vis.rate_limit-enable=0 \
  -C bp.virtio_net.hostbridge.interfaceName=tap0 \
  -a cluster0*=${YOCTO_DEPLOY_IMGS_DIR}/linux-system.axf \
  -C bp.virtio_blockdevice.image_path=${YOCTO_DEPLOY_IMGS_DIR}/core-image-minimal-
fvp-baser-aemv8r64.wic
```



The terminal console login is `root` without password.

## Devices supported in the kernel

Devices supported in the kernel:

- serial
- virtio disk
- virtio network
- watchdog
- rtc

Devices not supported or not functional:

- Only one CPU since SMP is not functional in boot-wrapper-aarch64 yet.

## 2.2 Release 2 – SMP

Change log:

- Enabled SMP support via boot-wrapper-aarch64 providing the PSCI CPU\_ON and CPU\_OFF functions
- Introduced Armv8-R64 compiler flags
- Added Linux PREEMPT\_RT support via linux-yocto-rt-5.10
- Added support for file sharing with the host machine using Virtio P9
- Added support for runfvp
- Added performance event support (PMU) in the Linux device tree

BSP support:

- The fvp-baser-aemv8r64 Yocto MACHINE supports the following BSP components, where either a standard or Real-Time Linux kernel (PREEMPT\_RT) can be built and run:
  - boot-wrapper-aarch64: provides PSCI support
  - Linux kernel: linux-yocto-5.10
  - Linux kernel with PREEMPT\_RT support: linux-yocto-rt-5.10



The Real-Time Linux kernel (PREEMPT\_RT) does not use the real-time architectural extensions of the Armv8-R feature set.

---

The following instructions provide a quick start guide, explaining how to build and run the software:

### Host environment setup

The following instructions have been tested on hosts running Ubuntu 18.04 and Ubuntu 20.04.

Install the required packages for the build host:

<https://docs.yoctoproject.org/singleindex.html#required-packages-for-the-build-host>

Install the kas setup tool for bitbake based projects:

```
$ pip3 install --user kas
```

For more details on kas, see <https://kas.readthedocs.io/>

To build the images for fvp-base machine, you also need to:

- Download the `FVP_Base_AEMv8R_11.15_14.tgz` image AEM V8-R FVP Installer (Linux) package from [Arm's website](#). You need to have an account and be logged in to be able to download it.
- Set absolute path to the `FVP_Base_AEMv8R_11.15_14.tgz` downloaded package in `FVP_BASE_R_AEM_TARBALL_URI`
- Accept EULA in `FVP_BASE_R_ARM_EULA_ACCEPT`

The variables should be set like so:

```
$ FVP_BASE_R_AEM_TARBALL_URI="file:///absolute/path/to/FVP_Base_AEMv8R_11.15_14.tgz"
$ FVP_BASE_R_ARM_EULA_ACCEPT="True"
```



The host machine should have at least 50 GBytes of free disk space for the next steps to work correctly.

## Fetch sources

Fetch the meta-arm repository into a build directory:

```
$ mkdir -p ~/fvp-baser-aemv8r64-build
$ cd ~/fvp-baser-aemv8r64-build
$ git clone https://git.yoctoproject.org/git/meta-arm
$ cd meta-arm
$ git checkout ba55d78916338d84965d77c586ea92628ac55831
```

## Build

Building with the standard Linux kernel:

```
$ cd ~/fvp-baser-aemv8r64-build
$ export FVP_BASE_R_AEM_TARBALL_URI="file:///absolute/path/to/
FVP_Base_AEMv8R_11.15_14.tgz"
$ export FVP_BASE_R_ARM_EULA_ACCEPT="True"
$ kas build meta-arm/kas/fvp-baser-aemv8r64-bsp.yml
```

Building with the Real-Time Linux kernel (PREEMPT\_RT):

```
$ cd ~/fvp-baser-aemv8r64-build
$ export FVP_BASE_R_AEM_TARBALL_URI="file:///absolute/path/to/
FVP_Base_AEMv8R_11.15_14.tgz"
$ export FVP_BASE_R_ARM_EULA_ACCEPT="True"
$ kas build meta-arm/kas/fvp-baser-aemv8r64-rt-bsp.yml
```

## Networking

To enable networking on the FVP via a host network interface, you will need to install the following package(s):

- Ubuntu 18.04:

```
$ sudo apt-get install libvirt-bin net-tools
```

- Ubuntu 20.04:

```
$ sudo apt-get install libvirt-dev libvirt-daemon qemu-kvm libvirt-daemon-system  
libvirt-clients bridge-utils net-tools
```

Once that is installed for your OS version, setup tap0 using the following commands:

```
$ sudo virsh net-start default  
$ sudo ip tuntap add dev tap0 mode tap user $(whoami)  
$ sudo ifconfig tap0 0.0.0.0 promisc up  
$ sudo brctl addif virbr0 tap0
```

To clean up the tap0 interface on the host use the following commands:

```
$ sudo brctl delif virbr0 tap0  
$ sudo ip link set virbr0 down  
$ sudo brctl delbr virbr0  
$ sudo virsh net-destroy default  
$ sudo ip link delete tap0
```

## Run

To run an image after the build is done with the standard Linux kernel:

```
$ kas shell --keep-config-unchanged \  
  meta-arm/kas/fvp-baser-aemv8r64-bsp.yml \  
    --command "../layers/meta-arm/scripts/runfvp \  
      --console \  
      -- \  
        --parameter 'bp.smc_91c111.enabled=1' \  
        --parameter 'bp.virtio_net.hostbridge.interfaceName=tap0'"
```

To run an image after the build is done with the Real-Time Linux kernel (PREEMPT\_RT):

```
$ kas shell --keep-config-unchanged \  
  meta-arm/kas/fvp-baser-aemv8r64-rt-bsp.yml \  
    --command "../layers/meta-arm/scripts/runfvp \  
      --console \  
      -- \  
        --parameter 'bp.smc_91c111.enabled=1' \  
        --parameter 'bp.virtio_net.hostbridge.interfaceName=tap0'"
```



Note

The terminal console login is `root` without password.

To finish the fvp emulation, you need to close the telnet session and stop the runfvp script:

- To close the telnet session:
  - Escape to telnet console with `ctrl+]`.

- Run `quit` to close the session.
- To stop the `runfvp`:
  - Type `ctrl+c` and wait for `kas` process to finish.

### File sharing between host and fvp

It is possible to share a directory between the host machine and the fvp using the virtio P9 device component included in the kernel. To do so, create a directory to be mounted from the host machine:

```
$ mkdir /path/to/host-mount-dir
```

Then, add the following parameter containing the path to the directory when launching the model:

```
$ --parameter 'bp.virtiop9device.root_path=/path/to/host-mount-dir'
```

Once you are logged into the fvp, the host directory can be mounted in a directory on the model using the following command:

```
$ mount -t 9p -o trans=virtio,version=9p2000.L FM /path/to/fvp-mount-dir
```

### Devices supported in the kernel

Devices supported in the kernel:

- serial
- virtual 9p
- virtio disk
- virtio network
- watchdog
- rtc

### Known Issues and Limitations

The following are known issues and limitations:

- Only PSCI CPU\_ON and CPU\_OFF functions are supported
- Linux kernel does not support booting from secure EL2 on Armv8-R AArch64
- Linux KVM does not support Armv8-R AArch64

## 2.3 Release 3 – UEFI

Changes since Release 2 - SMP:

- Added U-Boot v2021.07 for UEFI support

- Updated boot-wrapper-aarch64 revision and added support for booting U-Boot
- Included boot-wrapper-aarch64 PSCI services in /memreserve/ region
- Fixed the counter frequency initialization in boot-wrapper-aarch64
- Configured the FVP to use the default RAM size of 4 Gb
- Fixed PL011 and SP805 register sizes in the device tree
- Added virtio\_net User Networking mode by default and removed instructions about tap networking setup
- Updated Linux kernel version from 5.10 to 5.14 for both standard and Real-Time (PREEMPT\_RT) builds

#### BSP Support:

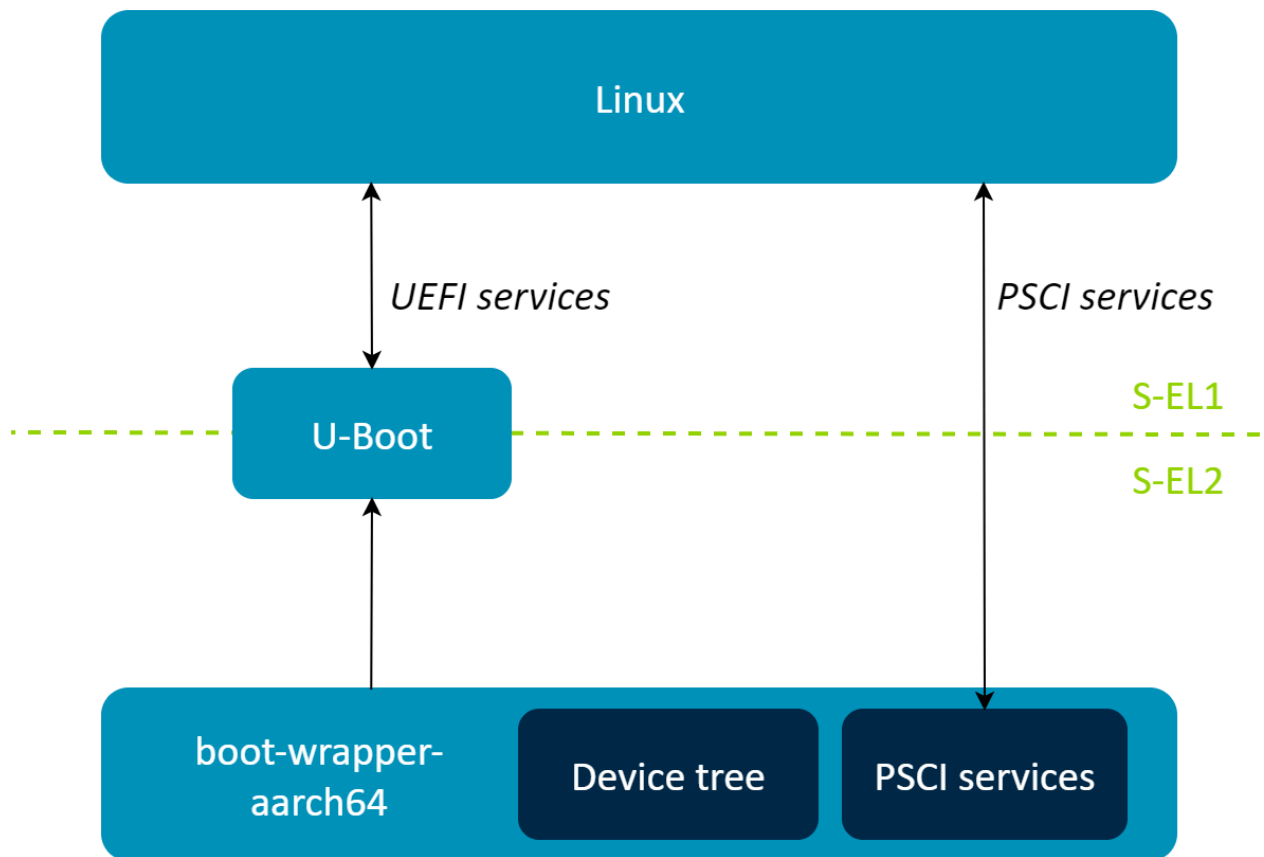
- The fvp-baser-aemv8r64 Yocto MACHINE supports the following BSP components, where either a standard or Real-Time Linux kernel (PREEMPT\_RT) can be built and run:
  - boot-wrapper-aarch64: provides PSCI support
  - U-Boot: v2021.07: provides UEFI services
  - Linux kernel: linux-yocto-5.14
  - Linux kernel with PREEMPT\_RT support: linux-yocto-rt-5.14

Note that the Real-Time Linux kernel (PREEMPT\_RT) does not use the real-time architectural extensions of the Armv8-R feature set

The following instructions provide a quick start guide, explaining how to build and run the software:

### High-Level architecture

The diagram below shows the current boot flow:

**Figure 2-1: Boot flow**

The firmware binary (generated as `linux-system.axf`) includes `boot-wrapper-aarch64`, the flattened device tree and U-Boot. U-Boot is configured to automatically detect a virtio block device and boot the UEFI payload at the path `/efi/boot/bootaa64.efi`. Using the standard build, the first partition contains a Grub image at this path, which boots the Linux kernel at `/Image` on the same partition. The second partition of the image contains the Linux root file system.

There is no EL3 or non-secure world in the Armv8-R AArch64 architecture, so the reset vector starts `boot-wrapper-aarch64` at S-EL2. `boot-wrapper-aarch64` is compiled with the `--enable-keep-el` flag, which causes it to boot U-Boot at S-EL2 too. U-Boot is compiled with the `CONFIG_ARMV8_SWITCH_TO_EL1` flag, which causes it to switch to S-EL1 before booting Linux.

The bundled device tree is passed to U-Boot via register `x0`. U-Boot passes the same device tree to Linux via the UEFI system table.

Power state management is provided by PSCI services in `boot-wrapper-aarch64`. Linux accesses the PSCI handler via HVC calls to S-EL2. U-Boot has been patched to prevent it from overriding the exception vector at S-EL2. The PSCI handler memory region is added to a `/memreserve/` node in the device tree.

Please note that the final firmware architecture for the fvp-baser-aemv8r64 is not yet stabilized. The patches in meta-arm-bsp are provided for development and evaluation purposes only, and should not be used in production firmware.

## Host environment setup

The following instructions have been tested on hosts running Ubuntu 18.04 and Ubuntu 20.04.

Install the required packages for the build host:

<https://docs.yoctoproject.org/singleindex.html#required-packages-for-the-build-host>

kas is a setup tool for bitbake based projects. The minimal supported version is 2.6, install it like so:

```
$ pip3 install --user --upgrade kas
```

For more details on kas, see <https://kas.readthedocs.io/>

To build the images for fvp-base machine, you also need to:

- Download the FVP\_Base\_AEMv8R\_11.17\_21.tgz image AEM V8-R FVP Installer (Linux) package from [Arm's website](#). You need to have an account and be logged in to be able to download it.
- Set absolute path to the FVP\_Base\_AEMv8R\_11.17\_21.tgz downloaded package in FVP\_BASE\_R\_AEM\_TARBALL\_URI
- Accept EULA in FVP\_BASE\_R\_ARM\_EULA\_ACCEPT

The variables should be set like so:

```
$ FVP_BASE_R_AEM_TARBALL_URI="file:///absolute/path/to/FVP_Base_AEMv8R_11.17_21.tgz"  
$ FVP_BASE_R_ARM_EULA_ACCEPT="True"
```



The host machine should have at least 50 GBytes of free disk space for the next steps to work correctly.

## Fetch sources

To fetch and build the ongoing development of the software stack follow the instructions on this document.

To fetch and build the version 1 (single core) find instructions at [Release 1 – Single Core](#)

To fetch and build the version 2 (linux smp) find instructions at [Release 2 – SMP](#)

Fetch the meta-arm repository into a build directory:

```
$ mkdir -p ~/fvp-baser-aemv8r64-build  
$ cd ~/fvp-baser-aemv8r64-build
```



```
$ git clone https://git.yoctoproject.org/git/meta-arm -b honister
```

## Build

Building with the standard Linux kernel:

```
$ cd ~/fvp-baser-aemv8r64-build
$ export FVP_BASE_R_AEM_TARBALL_URI="file:///absolute/path/to/
FVP_Base_AEMv8R_11.17_21.tgz"
$ export FVP_BASE_R_ARM_EULA_ACCEPT="True"
$ kas build meta-arm/kas/fvp-baser-aemv8r64-bsp.yml
```

Building with the Real-Time Linux kernel (PREEMPT\_RT):

```
$ cd ~/fvp-baser-aemv8r64-build
$ export FVP_BASE_R_AEM_TARBALL_URI="file:///absolute/path/to/
FVP_Base_AEMv8R_11.17_21.tgz"
$ export FVP_BASE_R_ARM_EULA_ACCEPT="True"
$ kas build meta-arm/kas/fvp-baser-aemv8r64-rt-bsp.yml
```

## Run

To run an image after the build is done with the standard Linux kernel:

```
$ kas shell --keep-config-unchanged \
  meta-arm/kas/fvp-baser-aemv8r64-bsp.yml \
  --command "../layers/meta-arm/scripts/runfvp \
  --console "
```

To run an image after the build is done with the Real-Time Linux kernel (PREEMPT\_RT):

```
$ kas shell --keep-config-unchanged \
  meta-arm/kas/fvp-baser-aemv8r64-rt-bsp.yml \
  --command "../layers/meta-arm/scripts/runfvp \
  --console "
```



**Note**

The terminal console login is `root` without password.

---

To finish the fvp emulation, you need to close the telnet session:

- Escape to telnet console with `ctrl+]`
- Run `quit` to close the session

## Networking

The FVP is configured by default to use “user-mode networking”, which simulates an IP router and DHCP server to avoid additional host dependencies and networking configuration. Outbound connections work automatically, e.g. by running:

```
$ wget www.arm.com
```

Inbound connections require an explicit port mapping from the host. By default, port 8022 on the host is mapped to port 22 on the FVP, so that the following command will connect to an ssh server running on the FVP:

```
$ ssh root@localhost -p 8022
```

Note that user-mode networking does not support ICMP, so `ping` will not work.

For more information about user-mode networking, please see <https://developer.arm.com/documentation/100964/1117/Introduction-to-Fast-Models/User-mode-networking?lang=en>

## File sharing between host and fvp

It is possible to share a directory between the host machine and the fvp using the virtio P9 device component included in the kernel. To do so, create a directory to be mounted from the host machine:

```
$ mkdir /path/to/host-mount-dir
```

Then, add the following parameter containing the path to the directory when launching the model:

```
$ --parameter 'bp.virtiop9device.root_path=/path/to/host-mount-dir'
```

For example, for the standard Linux kernel:

```
$ kas shell --keep-config-unchanged \  
  meta-arm/kas/fvp-baser-aemv8r64-bsp.yml \  
  --command "../layers/meta-arm/scripts/runfvp \  
  --console -- --parameter \  
  'bp.virtiop9device.root_path=/path/to/host-mount-dir'"
```

Once you are logged into the fvp, the host directory can be mounted in a directory on the model using the following command:

```
$ mount -t 9p -o trans=virtio,version=9p2000.L FM /path/to/fvp-mount-dir
```

## Devices supported in the kernel

Devices supported in the kernel:

- serial

- virtual 9p
- virtio disk
- virtio network
- watchdog
- rtc

## Known Issues and Limitations

The following are known issues and limitations:

- Only PSCI CPU\_ON and CPU\_OFF functions are supported
- Linux kernel does not support booting from secure EL2 on Armv8-R AArch64
- Linux KVM does not support Armv8-R AArch64
- Device DMA memory cache-coherence issue: the FVP `cache_state_modelled` parameter will affect the cache coherence behavior of peripherals' DMA. When users set `cache_state_modelled=1`, they also have to set `cci400.force_on_from_start=1` to force the FVP to enable snooping on upstream ports.

## 3. Xen Hypervisor

A software stack that introduces a type-1 hypervisor to enable hosting of OS images in separate Virtual Machines.

This example adds a new configuration to enable hosting of Linux and Zephyr OS instances as guests running on a Xen Hypervisor on the Armv8-R AArch64 FVP model. It is provided via a set of Yocto layers which contain all the instructions necessary to fetch and build the source as well as to download the model and launch the example on the Armv8-R AEM FVP model.

### 3.1 Release 4 – Xen

A software stack to boot a dual-OS system - Linux as the Rich OS and Zephyr as the RTOS - running in parallel using Xen as a type-1 hypervisor, is available [here](#).

### 3.2 Release 5 – InterVM communication with Linux containers

[An example software stack](#) to demonstrate a communication path between Xen guests and how data can be exported from a Zephyr RTOS to a Linux instance. Data hosted on the Linux guest can also be accessed via HTTP, from a container hosted instance of nginx.

## 4. Zephyr

A port of the Zephyr RTOS is available for the Armv8-R AEM FVP model which enables general OS features and basic peripherals.

If you are new to Zephyr follow the Getting Started Guide [here](#).

To run the Fixed Virtual Platform simulation tool and build basic Zephyr applications and kernel tests in this Arm FVP emulated environment, instructions can be found [here](#).