

Emtrion Developer Kit for dimm-am335x

Yocto based BSP manual

Rev001 / 23.06.2017

© Copyright 2017 **emtrion GmbH**

All rights reserved. This documentation may not be photocopied or recorded on any electronic media without written approval. The information contained in this documentation is subject to change without prior notice. We assume no liability for erroneous information or its consequences. Trademarks used from other companies refer exclusively to the products of those companies.

Revision: **001 / 23.06.2017**

Rev	Date/Signature	Changes
001	23.06.2017/MI	First release
002	26.07.2017/MI	Supporting the base Idren

1	INTRODUCTION	5
2	TERMS AND DEFINITIONS	5
3	THE LINUX VIRTUAL MACHINE VM	6
3.1	CONTENT	6
3.2	STARTING THE VM	7
3.2.1	<i>Login account</i>	7
3.3	PRECONFIGURED VARIABLES.....	7
4	DEVICE START UP	8
4.1	DEMO.....	8
4.2	DEVICE NETWORK SETUP.....	9
5	PRE-BUILT IMAGES AND INSTALLATIONS.....	9
5.1	INSTALLATION.....	9
5.1.1	<i>Installation manually</i>	10
5.1.1.1	rfs-<MACHINE>.tar.bz2	10
5.1.1.2	rfs-<MACHINE>-sdk.tar.bz2.....	10
5.1.1.3	poky-glibc-x86_64-rfs-armv7ahf-neon-toolchain-2.2.2.sh	10
5.1.1.4	update-<MACHINE>.tar.gz.....	10
5.1.1.5	meta-emtrion-dimm-am335x.tar.gz	10
5.1.2	<i>Installation automatically</i>	11
6	THE LAYER META-EMTRION-DIMM-AM335X.....	11
6.1	INSTALLATION.....	11
6.2	OVERVIEW OF THE RECIPE STRUCTURE	11
6.3	PROVIDED IMAGES.....	13
6.4	CONFIGURATION.....	13
6.5	SETTING UP THE BUILD SYSTEM.....	14
6.5.1	<i>Behaviour of the setup script “setup-environment”</i>	14
6.5.2	<i>Performing the setup script</i>	14
6.6	CREATING AN IMAGE	16
6.7	OUTPUT FILES	16
6.7.1	<i>Root File System</i>	17
6.7.2	<i>boot directory</i>	17
6.8	FURTHER READINGS ON YOCTO.....	17
7	U-BOOT BOOTLOADER.....	17
7.1	BASIC U-BOOT OPERATION.....	17
7.2	USING U-BOOT TO CHANGE BOOT DEVICE OR UPDATE THE SYSTEM.....	18
7.2.1	<i>Boot setup and updating of the system</i>	18
7.2.1.1	Updating of the system(root file system and kernel)	18
7.2.1.2	Booting	19
8	SDK	20
8.1	INSTALLING THE SDK.....	20
8.1.1	<i>Setting up the SDK environment</i>	20
9	FURTHER INFORMATION	20
9.1	ONLINE RESOURCES.....	20

9.2 WE SUPPORT YOU 20

1 Introduction

Emtrion produces and offers various base boards and modules. This software supports the developer kit with the combination of the module DIMM-AM335x and the bases below.

Base	Module	variant	Machine	Device tree
Cadun	DIMM-AM335x	-	dimmm-am335x	am335x-dimm
Idren	DIMM-AM335x	-	dimmm-am335x-idren	am335x-dimm-idren

The BSP is a ti-linux-4.4.y kernel based on mainline 4.4.67. Supporting emtrion’s developer kit is managed by the specific kernel configuration file and the device tree file. Graphic support is OpenGL ES 2.

The RootFS has been created with Yocto Openembedded, **morty** 2.2.2.

The recipes used by the emtrion’s meta-layers are mostly based on recipes of several other meta-layers. These recipes have been modified or extended by emtrion.

This manual describes the scope of the developer kit, how to set it up.

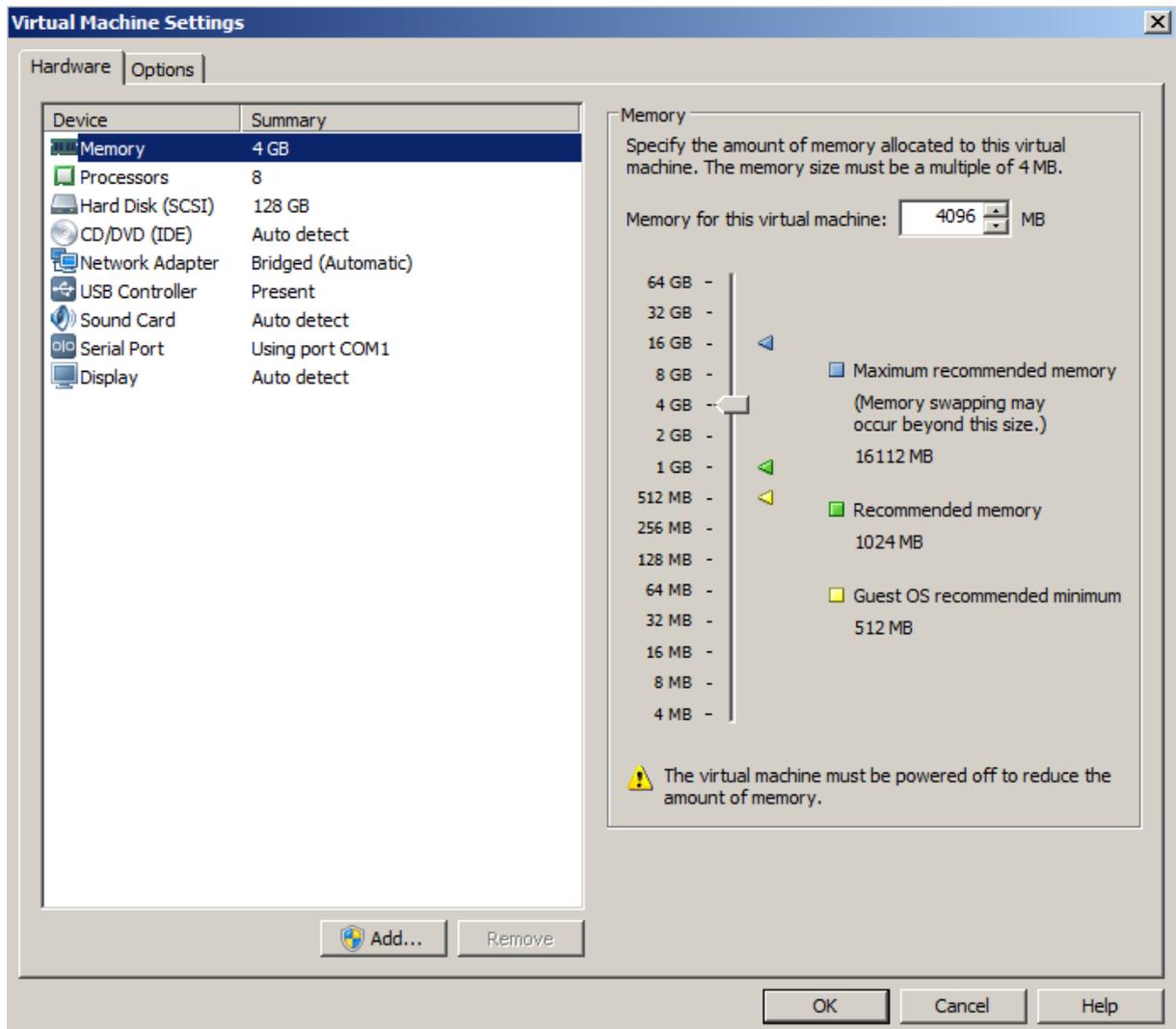
It is assumed that users of emtrion Linux developer kits are already familiar with U-boot, Linux, Yocto and creating and debugging applications. General Linux and programming knowledge are out of the scope of this document. emtrion is happy to assist you in acquiring this knowledge. If you are interested in training courses or getting support, please contact the emtrion sales department.

2 Terms and Definitions

Term	Definition
Target	Module dimmm-am335x with the respective baseboard
Host	Workstation, Developer PC
Toolchain	Compiler, Linker, etc.
RootFS	Root file system, contains the basic operating system
Console	Text terminal interface for Linux
NFS	Network File System, can share directories over network
NFS_SHARE	Location that is exported by the NFS for the purpose of updating and booting by using NFS
U-Boot	Bootloader, hardware initialization, updating images, starting OS
YP	Yocto Project
INST_DIR	Location where Yocto and the meta-layers are installed
MACHINE	Specifies the target device for which the image is built. Dependent on the base the machine name is dimmm-am335x or dimmm-am335x-idren .
BUILD_DIR	Machine dependent build directory
BSP	Board Support Package
SDK	Software Development Kit

3 The Linux virtual machine VM

To support the development with emtrion’s Yocto-Layer a VMware virtual machine was configured. To obtain a good performance the VM is configured with following properties.



However, the settings are strongly dependent on the PC and have to be adjusted later on your PC.

- Used memory
- Number of processors
- Network Adapter
- Serial Port

3.1 Content

As Linux Distribution **Debian 9** (stretch) is used. The Distribution was setting up by general Yocto Project system requirements described in the chapter “1.3. System Requirements” of the Reference Manual Yocto Project 2.2.2 Release.

<http://www.yoctoproject.org/docs/2.2/ref-manual/ref-manual.html>

Further components are included

- ❖ Serial ports added to the virtual machine appear at /dev/ttySn, USB serial converters at /dev/ttyUSBn
- ❖ A NFS server exporting the nfs share **/home/hico/nfs**
- ❖ The serial terminal program **picocom** for connecting to the target
- ❖ The Yocto-Layers **meta-emtrion** and **meta-emtrion-bsp**
- ❖ SDK and pre-built images of the Sub-Layer **meta-emtrion-dimm-am335x** in **~/Downloads**

3.2 Starting the VM

The VM is a compressed ZIP archive. Changing settings and starting the VM is used by the VMware Player or VMware Workstation. Here the link for downloading the VMware Player

<https://www.vmware.com/go/downloadplayer>

The corresponding manual is behind this link

[VMware® Player™ Manual](#)

After decompressing of the VM and importing it by the VMware Player, first check if the settings above are fit to your PC. If not, adjust the settings by reading the corresponding chapters of the specified manual.

Please note, the size of the VM will increase up to 128 GB while you are working with it.

3.2.1 Login account

The login data are specified as follow

username: **hico**

password: **hico**

3.3 Preconfigured variables

Within the VM there are used some predefined locations. In the scope of this document the locations are assigned to specified placeholders. They are listed in the table below.

Placeholder	Assignment	remark
MACHINE	dimm-am335x dimm-am335x-idren	machine name, dependent on the base is used
HOME_DIR	/home/hico	home directory of user hico
NFS_SHARE	< HOME_DIR >/nfs	Location exported by the NFS
NFS_ROOTFS	< NFS_SHARE >/< MACHINE >/root/rootfs	nfs share for booting the root file system by using NFS
NFS_UPDATE	< NFS_SHARE >/< MACHINE >/update	nfs share for updating the root file system by using NFS
INST_DIR	< HOME_DIR >/openembedded	Location where Yocto and all the meta-layers will be stored to
BUILD_DIR	< INST_DIR >/builddir/emtrion/machines/ < MACHINE >	Location of the build system.
BUILD_DWNL	< INST_DIR >/builddir/downloads	Location of the fetched

		downloads while the build process
BUILD_SSTATE	<INST_DIR>/builddir/sstate-cache	Location of the sstate-cache while the build process
HOME_DWNL	<HOME_DIR>/Downloads	Location of the pre-built images, SDK
SDK_DIR	/opt/poky/2.2.2	SDK with tools, sources and libraries

4 Device Start Up

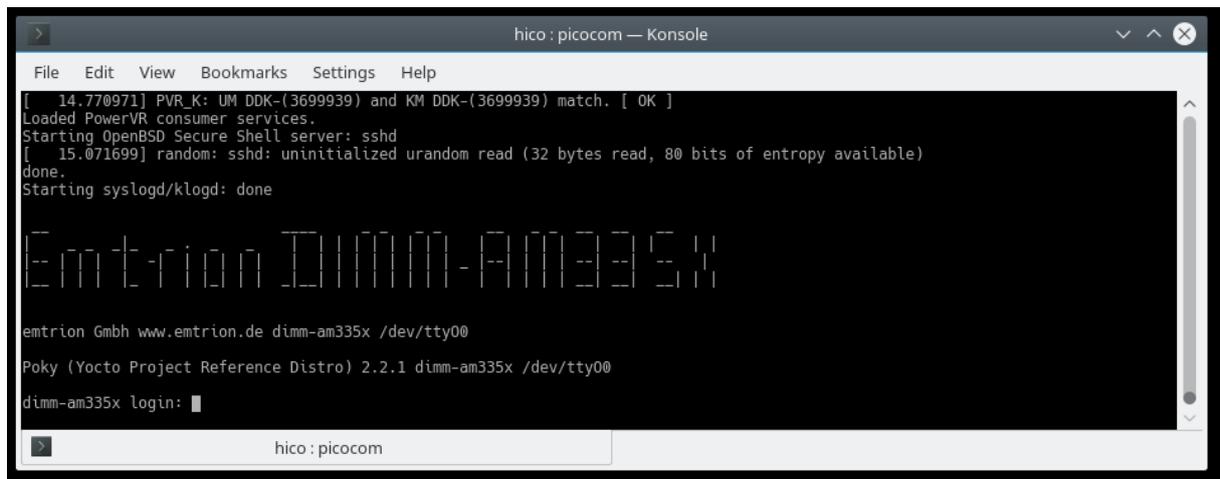
Connect the developer kit to the serial port attached to the virtual machine and your network. Open a console in the VM and open a serial terminal by enter

picocom -b115200 /dev/ttySx.

ttySx has to be replaced with the device assigned to the connected serial port.

In the case of using an USB serial adapter replace it by the corresponding **ttyUSB**.

You may now power on the developer kit. You should see booting U-Boot and Linux.



1: Serial terminal showing login prompt

After the developer kit is booted you are prompted for login:

- dimm-am335x login: **root** (Cadun)
- dimm-am335x-idren login: **root** (ldren)

4.1 Demo

The Root File System includes a demo where you can see if OpenGL is supported. You can start it by enter in the terminal

root@emtrion-mx6:~# kmscube &

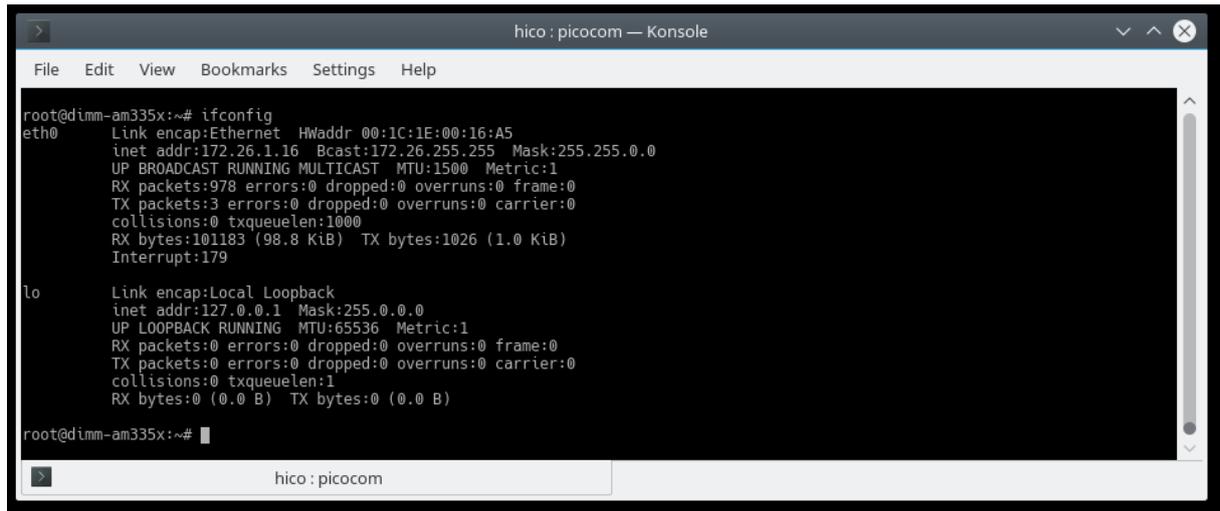
To terminate the demo you can enter **killall kmscube** in the terminal

```
root@emtrion-mx6:~# killall kmscube
```

4.2 Device Network Setup

Per default the developer kit is setup to use a dhcp server. This is configurable by a bootloader environment variable "ip-method". This variable can have the values "dhcp" or "static".

You can check if there is a valid ip address with the command **ifconfig**.



```
hico: picocom — Konsole
File Edit View Bookmarks Settings Help
root@dimm-am335x:~# ifconfig
eth0    Link encap:Ethernet  HWaddr 00:1C:1E:00:16:A5
        inet addr:172.26.1.16  Bcast:172.26.255.255  Mask:255.255.0.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:978 errors:0 dropped:0 overruns:0 frame:0
        TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:101183 (98.8 KiB)  TX bytes:1026 (1.0 KiB)
        Interrupt:179

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@dimm-am335x:~#
```

2: ifconfig output

If the setup is not correct you have to do it manually. Please check the description of the bootloader configuration on how to set up the variable "ip-method".

5 Pre-built images and installations

To reduce the size of the VM for delivering, the VM is shipped only with the pre-built images and not including the outputs of the build process.

The pre-built images and others are located in **<HOME_DOWNL>**

- rfs-**<MACHINE>**.tar.bz2
- rfs-**<MACHINE>**-sdk.tar.bz2
- poky-glibc-x86_64-rfs-armv7ahf-neon-toolchain-2.2.2.sh
- update-**<MACHINE>**.tar.gz
- meta-emtrion-dimm-am335x.tar.gz

The images have been tested and can be used for designing applications.

5.1 Installation

Dependent on the focus of the user, the appropriate archives have to be installed before. You can do the installation either manually or automatically.

5.1.1 Installation manually

5.1.1.1 *rfs-<MACHINE>.tar.bz2*

This archive is the image without SDK extensions. It contains the root file system as well as the kernel.

Installing the image, you can boot the system by using NFS. From the home directory enter

```
sudo tar xf <HOME_DOWNLOAD>/rfs-<MACHINE>.tar.bz2 -C <NFS_ROOTFS>
```

How you can boot it by using NFS is described in the chapter 7 of the Bootloader.

5.1.1.2 *rfs-<MACHINE>-sdk.tar.bz2*

This archive is similar to the previous image with SDK extensions.

You can install it as the previous image. However, this is not suitable for normal use.

5.1.1.3 *poky-glibc-x86_64-rfs-armv7ahf-neon-toolchain-2.2.2.sh*

This file presents the SDK as a self extracted script. The SDK is required for developing applications outside the <BUILD_DIR>.

Install the SDK from the home directory by prompting

```
<HOME_DOWNLOAD>/poky-glibc-x86_64-rfs-armv7ahf-neon-toolchain-2.2.2.sh
```

for the MACHINE **dim-am335x**

and

```
<HOME_DOWNLOAD>/poky-glibc-x86_64-rfs-armv7ahf-neon-toolchain-2.2.2-dimm-am335x-idren.sh
```

for the MACHINE **dim-am335x-idren**

While performing the script, you will be asked for the installation directory. Let the default **/opt/poky/...** and confirm it.

5.1.1.4 *update-<MACHINE>.tar.gz*

This archive includes files which support the update process of the root file system and kernel of the developer kit.

Install the files of the archive to the nfs share <NFS_UPDATE>/boot by prompting

```
tar xf <HOME_DOWNLOAD>/update-<MACHINE>.tar.gz -C <NFS_UPDATE>/boot
```

How you can update the system on the developer kit is described in the chapter 7 of the Bootloader.

5.1.1.5 *meta-emtrion-dimm-am335x.tar.gz*

This archive contains the meta-layers for emtrion's dimm-am335x. You have to install it if you want to create your own images.

Install the layer by prompting

```
tar xf <HOME_DOWNL>/meta-emtrion-dimm-am335x.tar.gz -C <INST_DIR>
```

5.1.2 Installation automatically

The described installation steps above are implemented in a shell script. The shell script is located in <HOME_DOWNL> and can be used for the purpose of installation. Perform the script from within <HOME_DOWNL> with the corresponding machine name as parameter.

```
./install_prebuild_binaries.sh <MACHINE>
```

6 The layer meta-emtrion-dimm-am335x

The layer meta-emtrion-dimm-am335x consists of two meta-layers, a basic meta-layer and a specific meta-layer for the dim-am335x.

6.1 Installation

The installation of the layer described in the previous chapter, will locate the layer in the location <INST_DIR>.

6.2 Overview of the recipe structure

The recipe structure provided by the layers is shown below. The various recipes implement and define what packages are included in the image provided by the layer.

```

openembedded : bash — Konsole
File Edit View Bookmarks Settings Help
meta-emtrion
├── conf
├── recipes
│   ├── empurs-scripts
│   │   └── files
│   ├── emtrion-config
│   │   └── files
│   ├── gpio-utils
│   │   └── files
│   ├── images
│   ├── meta
│   └── packagegroups
├── recipes-core
│   └── initscripts
│       └── files
├── recipes-kernel
│   └── gator
│       ├── gator-daemon
│       └── gator-driver
├── recipes-support
│   ├── devmem2
│   └── devmem2
└── meta-emtrion-bsp
    └── meta-emtrion-dimm-am335x
        ├── conf
        ├── machine
        ├── recipes
        │   ├── empurs-scripts
        │   ├── emtrion-config
        │   └── images
        ├── recipes-bsp
        │   └── powervr-drivers
        ├── recipes-core
        │   ├── base-files
        │   └── packagegroups
        ├── recipes-graphics
        │   ├── libgles
        │   ├── mesa
        │   └── packagegroups
        ├── recipes-kernel
        └── linux
41 directories
hico@emtrion-devKit-VM:~/openembedded$
openembedded : bash

```

Location	Remarks
meta-emtrion/	emtrion's basic meta-layer
├── conf	configuration file of this meta-layer
├── recipes	
│ ├── empurs-scripts	emtrion's basic update support
│ │ └── files	
│ ├── emtrion-config	basic configuration files
│ │ └── files	
│ ├── gpio-utils	example using gpio
│ │ └── files	
│ ├── images	provides the basic image recipes
│ ├── meta	not used
│ └── packagegroups	not needed in our case
├── recipes-core	
│ └── initscripts	additional init scripts
│ └── files	
├── recipes-kernel	
│ └── gator	gator not yet supported

	gator-daemon	
	gator-driver	
	recipes-support	
	devmem2	
	devmem2	not yet included
meta-emtrion-bsp/		
	meta-emtrion-dimm-am335x	specific meta-layer of dimm-am335x
	conf	configuration file of this meta-layer
	machine	machine configuration files
	recipes	
	empurs-scripts	emtrion's update support for dimm-am335x
	emtrion-config	Implementation of emtrion's production process, managing uboot_script, some basic configuration
	images	provides the final image recipes
	recipes-bsp	
	powervr-drivers	append recipe support of sgx for kernel version 4.4
	recipes-core	
	base-files	emtrion's login screen
	packagegroups	sgx graphic support
	recipes-graphics	
	libgles	for future use
	mesa	append recipe for mesa
	packagegroups	empty
	recipes-kernel	kernel
	linux	kernel recipe of ti kernel based on 4.4.67 mainline

6.3 Provided images

The layer provides three images.

- **core-image-purs**
initramfs used for emtrion's update mechanism
- **emtrion-devkit-image**
Image based on the packagegroup-core-boot provided by the layer meta of poky, however with extensive functionality
- **emtrion-devkit-image-sdk**
The SDK variant of emtrion-devkit-image

6.4 Configuration

Before starting of any bitbaking process, the Yocto build system has to know about some details about the machine and all of the layers required for. This information is stored in several configuration files.

- layer.conf
→ Specifies the layer's priority and makes it recipes visible while the build process.
- <MACHINE>.conf

→ Defines the target specific properties like what kernel will be used, type of kernel image, type of the RFS image, features will be added to the RFS image and more. Dependent on the developer kit the variable MACHINE has assigned to **dimmm-am335x** in the case of base **Cadun** is used or **dimmm-am335x-idren** in the case of the base **Idren**.

- **bblayers.conf**
→ Lists all the meta-layers required by the build process. The various meta-layers provide the packages used by the build process and included in the final image.
- **local.conf**
→ Include settings of the build environment. The global variable MACHINE is assigned, the max number of processors are available, number of running threads at the same time, location of directories **<BUILD_DWNL>** and **<BUILD_SSTATE>** and more. The file is automatically created by the setup script of the layer.

6.5 Setting up the build system

Setting up the build system, a user friendly setup script "**setup-environment**" exists. This script is located in the root of the layer **meta-emtrion-bsp**. Performing the script, all the required meta-layers will be installed for the release **morty** and the build environment is completed. In addition, to save disk space, the central locations **<BUILD_DWNL>** and **<BUILD_SSTATE>** are created.

6.5.1 Behaviour of the setup script "setup-environment"

The setup script can not only used for setting up a clean build system. It is also useful in the case you have closed the console of a completed build environment and you want to reopen a new one.

However, restarting the script to reopen a new build session, the behavior of the setup script is different if it is performed the first time.

The differences are listed below.

behaviour items	at first time	at restart	remarks
meta-layers	fetched	updated	
<BUILD_DIR>	created	obtained or created*	
configuration file	created	updated or created*	local.conf
configuration file	copied	copied	bbayers.conf
directory downloads	created	obtained	
directory sstate-cache	created	obtained	
asking for deleting build system	no	yes	delete process can take several minutes

(*) if deleting of the build system is performed

6.5.2 Performing the setup script

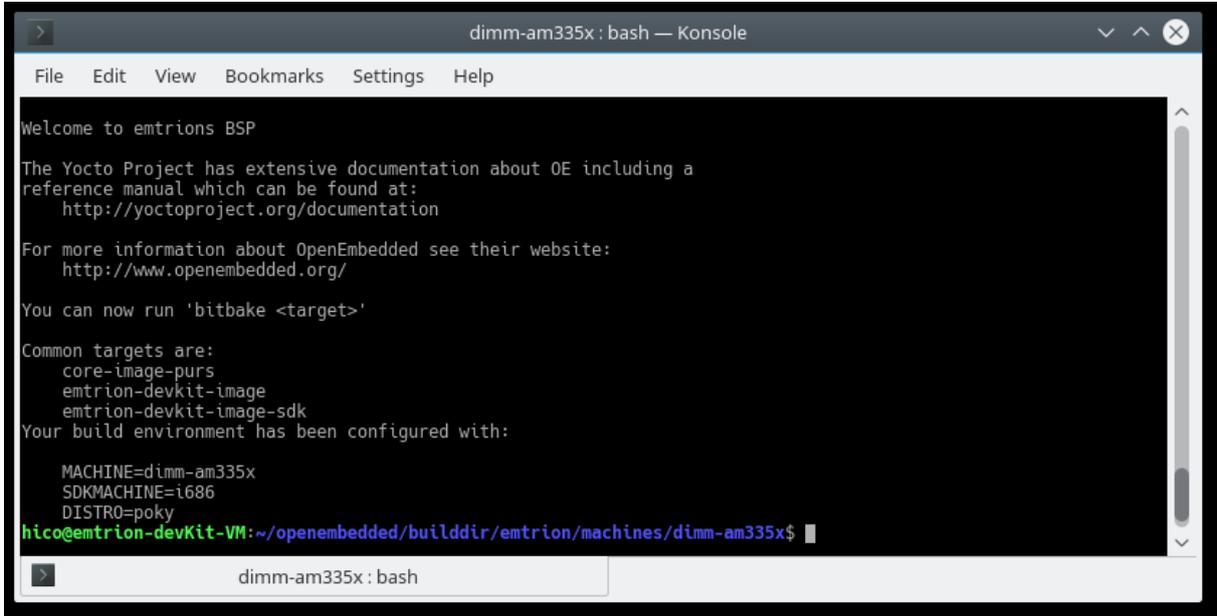
Starting the setup process, the script has to be sourced. Below the steps are described for setting up the build environment of the machine **dimmm-am335x**. Doing it for another machine, replace **dimmm-am335x** respectively.

Enter from the layer's location **<INST_DIR>/meta-emtrion-bsp**

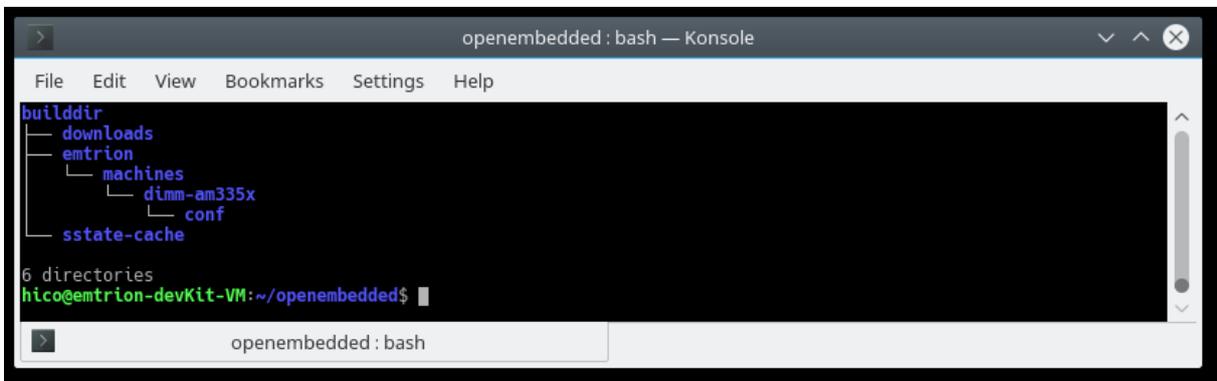
MACHINE=**dimmm-am335x** source ./setup-environment

The setup completes and finishes with prompting the build environment as shown below. At the output you can see the provided images.

From now you are able to build images for your developer kit with dimm-am335x.



The created directory structure while performing the setup script is shown below.



Location	Remarks
builddir/	
├── downloads	locates fetched data while the build process
├── emtrion	
│ ├── machines	locates build directories of various machines
│ │ ├── dimmm-am335x	build directory of dimmm-am335x
│ │ └── conf	locates local.conf, bblayers.conf
└── sstate-cache	locates the build states while the build process

6.6 Creating an image

After setting up the build system you can start building recipes and images for the dimm-am335x.

As mentioned before, the layer meta-emtrion-mx6 provides three images. You can start building an image by prompting bitbake following the name of the image recipe. Enter in the terminal of the build environment

```
bitbake <name_of_image_recipe>
```

- bitbake **core-image-purs**
Builds the initramfs that is used for emtrion's update mechanism. Due to the image is included by the images emtrion-devkit-image and emtrion-devkit-image-sdk, the image is automatically build by bitbaking of these images, but only if the image was still not yet built. For this reason the image has to build explicitly, if any changes were made before building one of the other images.
- bitbake **emtrion-devkit-image**
Builds the image for dimm-am335x. It creates a root file system with OpenGL support and adds a OpenGL demo. Additionally it includes the initramfs, the kernel and device tree.
- bitbake **emtrion-devkit-image-sdk**
Builds the SDK variant of the emtrion-devkit-image with source information
- bitbake **emtrion-devkit-image-sdk** -c populate_sdk
Builds the SDK installer containing the toolchain for the purpose of application development.

6.7 Output files

During the build process a lot of objects and images are created. However, the most relevant images are installed in

```
<BUILD_DIR>/tmp/deploy/images/<MACHINE>
```

respectively

```
<BUILD_DIR> /tmp/deploy/sdk.
```

The exact names of the images are listed below. Note: Some of them are symbolic links.

Images	Description
zImage-<MACHINE>	Kernel
zImage-am335x-dimm.dtb (*)	Device tree for Cadun
zImage-am335x-dimm-idren.dtb(*)	Device tree for Idren
rfs-<MACHINE>.tar.bz2	RootFS
rfs-<MACHINE>-sdk.tar.bz2	RootFS for SDK
ramdisk-<MACHINE>.cpio.gz	Ramdisk for update mechanism
poky-glibc-x86_64-rfs-armv7ahf-neon-toolchain-2.2.2.sh	SDK installer

(*) means a symbolic link

6.7.1 Root File System

As shown in the list above, the output of the root file system is a bz2 archive. You can decompress it by the tar command. For testing we recommend to decompress the archive to the `<NFS_SHARE>`. Navigate to the directory `<BUILD_DIR>` and call

```
sudo tar xf tmp/deploy/images/<MACHINE> /name_of_rootfs_archive -C <NFS_ROOTFS>
```

Don't forget "sudo" otherwise the kernel won't be able to modify the files during starting of the system.

6.7.2 boot directory

The directory structure of the root file system includes a location boot. In addition to the kernel image, device tree and ramdisk(restore root file system) a file **uboot_script** is located there.

This file implements some U-Boot command sequences. You can use it for the purpose of updating and booting the RootFS by using NFS.

However, the environment of the U-Boot has to be set up before. This is discussed in detail in the chapter 7 of the Bootloader.

6.8 Further readings on Yocto

YP documentations: <https://www.yoctoproject.org/documentation/archived>

OpenEmbedded: <http://www.openembedded.org/wiki/OpenEmbedded-Core>

YP-Repositories: <https://git.yoctoproject.org/>

7 U-Boot Bootloader

The basic task of U-Boot is to load the operating system from bulk memory into RAM and then start the kernel. You can also use it to initiate an update of the kernel, the RootFS and of U-Boot itself. Furthermore you can configure directly from the medium the operating system is to be booted from, for example eMMC or NFS.

7.1 Basic U-Boot operation

To work with U-Boot, first use a terminal program like picocom to connect to the serial line of the board. As soon as the U-Boot prompt appears in the terminal, U-Boot is ready to receive commands. The general U-Boot documentation can be found here: <http://www.denx.de/wiki/U-Boot/Documentation>

U-Boot has a set of environment variables which are used to store information needed for booting the operating system. Variables can contain information such as IP addresses, but they can also contain a whole script of actions to perform sequentially. The following commands explain the basic handling of environment variables:

U-Boot command	Explanation
printenv [variable]	This shows the value of the specified variable. If no variable is specified, the whole environment is shown.

setenv [variable] [value]	Set a variable to a specific value. If no value is specified, the variable gets deleted.
saveenv	Make your changes permanent, so they remain after power off or reboot.

7.2 Using U-Boot to change boot device or update the system

This chapter describes how U-Boot has to be setup for updating and booting.

The variable serverip has to be set to the IP-address of the VM. You can get the IP-address by prompting

sudo ip a

in the terminal of your VM.

Take the IP-address of the corresponding network adapter and assign it to the variable serverip in the U-Boot console. The format of [IP-address] is dot decimal notation.

```
U-Boot # setenv serverip <IP-address>
```

7.2.1 Boot setup and updating of the system

7.2.1.1 Updating of the system(root file system and kernel)

Due to an image archive contains the root file system as well as the kernel, updating of the system affects always both.

Before performing the update process, following steps have to be done. Please note, remove a possible plugged SDcard while performing the update process.

On VM

- Install the update archive **update-*<MACHINE>*.tar.gz** as described in the chapter 5, if not yet done.
- Copy the image to the nfs share ***<NFS_UPDATE>/boot***.

For the image created by the Yocto build process enter the command

```
cp <BUILD_DIR>/tmp/deploy/images/<MACHINE>/rfs-<MACHINE>.tar.bz2
<NFS_UPDATE>/boot/
```

On U-Boot

- Setting up the environment as follow.

```
U-Boot # setenv serverip          <ip-address of the VM>
U-Boot # setenv nfsroot           <NFS_UPDATE>
U-Boot # setenv ip-method        <dhcp or static>
U-Boor # setenv ipaddr <ip-address for device, only needed for static ip>
U-Boot # setenv netmask <netmask for device, only needed for static ip>
```

```
U-Boot # run restore_sys
```

This starts the update process. Please be patient as the process of fetching the root file system image via network and decompressing it to the flash storage can take a few minutes.

7.2.1.2 Booting

The default boot device in U-Boot is determined by the variable "bootcmd". If you want to set up one of the following boot options as a default you have to set "bootcmd" to the command mentioned below.

Boot from on-board flash

This is the default boot option configured when you receive the developer kit from emtrion. To start it manually simply use this command:

```
U-Boot # run flash_boot
```

Boot via network using NFS

Before perform booting, following steps have to be done.

On VM

- Clean a possibly installed image in the nfs share
`sudo rm -r <NFS_ROOTFS>/*`
- Install of either the pre-build image or the image created by the Yocto build process to the nfs share <NFS_ROOTFS>.

In case of the created image enter the command

```
sudo tar xf <BUILD_DIR>/tmp/deploy/images/<MACHINE>/\  
rfs-<MACHINE>.tar.bz2 -C <NFS_ROOTFS>
```

On U-Boot

- Setting up the environment as follow.

```
U-Boot # setenv serverip <ip-address of the VM>  
U-Boot # setenv nfsroot <NFS_ROOTFS>  
U-Boot # setenv ip-method <dhcp or static>  
U-Boot # setenv ipaddr <ip-address for device, only needed for static ip>  
U-Boot # setenv netmask <netmask for device, only needed for static ip>  
U-Boot # saveenv  
U-Boot # run net_boot
```

Now the board should boot via network by using NFS.

8 SDK

In order to develop applications outside of the Yocto build system you need to set up your host development system. For this purpose the YP offers several installation methods.

One of the methods to create a SDK is using the build system as described in the chapter 6.6.

The result is a SDK installer containing the toolchain and the sysroot which includes and matches the target root file system. The installer is stored in

```
<BUILD_DIR>/tmp/deploy/sdk/
```

8.1 Installing the SDK

Performing the SDK installer, you are asked for the installation directory. The default location is **/opt/poky/...** Let the default and confirm it. From inside the location **<BUILD_DIR>** start the installer as follows.

```
./tmp/deploy/sdk/ poky-glibc-x86_64-rfs-armv7ahf-neon-toolchain-2.2.2.sh
```

8.1.1 Setting up the SDK environment

Before you can start developing applications you have to setup the environment. For that purpose a script is installed during the installation process of the SDK. The script is stored in the SDK's directory of **<SDK_DIR>**.

Performing the setup procedure, the script has to be sourced as follows.

```
source <SDK_DIR>/environment-setup-armv7ahf-neon-poky-linux-gnueabi
```

The environment is only valid in the context of the terminal where this script has been called.

9 Further Information

9.1 Online resources

Further information can be found on the emtrion support pages.

www.support.emtrion.de

9.2 We support you

emtrion offers different kinds of services, among them Support, Training and Engineering. Contact us at sales@emtrion.com if you need information or technical support.