emYoctoLayers

Documentation

Rev001 / 22.06.2015

emtrion GmbH

Revision: **001 / 22.06.2015**

Rev	Date/Signature	Changes
001	22.06.15/Ml	First revision
002	04.09.15/Ml	Make some states a little more clearly

2 Contents

3 Terms and Definitions

Term	Definition
Dev-Kit/Target	Emtrion Developer Kit
Host	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	Directory that has to be shared for updating and booting via NFS
U-Boot	Bootloader, starts the operating system
YP	Yocto Project
INST_DIR	Directory where Yocto is installed
MACHINE	Defines the machine of the layer

4 Introduction

This documentation provides information about the basic handling around the emtrion's Yocto/OpenEmbedded layers we use to build the images for some of our Dev-Kits.

Please understand we can not go into more details about YP inside this documentation, because Yocto/OpenEmbedded is a powerful but also complex build system. So we can not support problems about handling the build system for free and ask you to refer to the official documentation. If the official documentation does not help you, we also offer training sessions about Yocto/OpenEmbedded.

Following are some helpful links where you can get detailed information.

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

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

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

5 emtrion's Dev-Kit layers

Currently, emtrion provides layers based on YP release version 1.7.2(dizzy) for the following Dev-Kits.

Dev-Kit	<MACHINE>	Name of layer	Remarks
dimm-am335x	dimm-am335x	meta-emtrion-<MACHINE>	
	dimm-am335x-rt		with RT-Preempt
sbc-sama55d36	sbc-sama5d36		

Additional to the layers above there is a common used layer **meta-emtrion**. That layer contains some basic recipes.

5.1 Directory structure

The common layer is delivered within the directory **meta-emtrion**, while each of the Dev-Kit layers has its own subdirectory inside the directory **meta-emtrion-bsp**.

5.1.1 Misc files

Within each Dev-Kit layer a directory **default-config** and **scripts** exist.

5.1.1.1 conf-Files

In default-config there are two conf-files, **bblayers.conf** and **local.conf**. Both files are modified by the setup process before copying to the build directory. Bblayers.conf includes all the required layers and local.conf includes some definition of variables.

5.1.1.2 setup file

In scripts there is located the machine dependent setup script. This script is responsible for setting up of the machine dependent build directory.

The name of the script is machine dependent and defined to

yocto_setup_<MACHINE>.

6 Installation

The installation of the emtrion layers has to be performed in order of the following steps.

1. **meta-emtrion** and **meta-emtrion-bsp** have to be included either in an already existing or not yet existing but intended INST_DIR of Yocto.
2. Setting up of the machine dependent build directory by souring of the corresponding setup script.

6.1 Setting up the machine specific build directory

Setting up a machine, Yocto sets up a build directory inside its directory structure as default. However it is recommended to leave this directory clean and create a new one, especially if you are working on several machines.

The setup script considers this issue and automates the setup process. First it creates a directory structure with the top level directory **YoctoBuildDirectory** inside of INST_DIR, including

- a machine dependent build directory
- a common downloads directory
- a common sstate-cache directory

Second, the required layers will be updated or installed, dependent if they exist or not. Then the defined YP release is checked out.

In a further step the conf-files will be modified and copied to the build directory.

At last the build environment of the created build directory is set.

All what you have to do setting up the build directory is sourcing the script inside of INST_DIR by replacing the correct <MACHINE>.

source ./meta- emtrion-bsp/meta-emtrion-<MACHINE>/scripts/yocto_setup_<MACHINE>

After sourcing the script the prompt automatically changed to the build directory.

6.2 The misc directories

Normally, one downloads and sstate-cache directory exists for each <MACHINE>. Saving space and time, the setup script creates a common used downloads and sstate-cache directory.

Setting up a machine with the machine specific setup script supports sharing of these directories, and allows executing of more than one build processes at the same time.

7 Creating the images

After setting up the machine you can start building recipes and images for the corresponding Dev-Kit.

Creating of an image is a process of two steps.

1. Creating of the ramdisk. This is required for update purpose.
   bitbake core-image-purs
2. Creating of the final image
   bitbake emtrion-devkit-image

7.1 Output files

While creating an image various output files will be produced. The files are saved inside

tmp/deploy/images/<MACHINE> in the build directory.

The produced files contain the corresponding <MACHINE> as part of the name.

The following table lists the relevant output files.

Output file	naming scheme	type	remarks
ramdisk	ramdisk-<MACHINE>.igz		
kernel-image	<type>-<MACHINE> <type>-<MACHINE>	uImage zImage	dependent on the Dev-Kit
RootFS-image	rfs_image-<MACHINE>.rootfs.<type>	tar.bz2 ubifs	dependent on the Dev-Kit
devicetree	none devicetree-<type><MACHINE>.dtb devicetree-<type>-<MACHINE>.dtb	"" zImage	dependent on the Dev-Kit

8 Set up a nfs share for booting and updating

Updating of the kernel or RootFS is used by NFS. Avoiding the update process for test purpose after each modification while developing, it is recommended to use NFS for booting, too.

For that a NFS-Server must be available on the Host and a <NFS_SHARE> has to be exported. However, setting up a NFS-Server and exporting a NFS-share can to be different from the linux distribution. Be sure to make this work correct, please inform yourself how this work has to be done at your distribution.

8.1 Structure of the NFS_SHARE

In general the <NFS_SHARE> has pointed to the unpacked RootFS. While booting or updating, the directory **boot** of the RootFS takes a central position. It includes all the needed components used by the different processes.

Due to of the central position of the directory "boot", the need of an unpacked RootFS is not necessary while updating. In this case the <NFS_SHARE> must only contain a subdirectory **boot** which includes the required files. These are in general the files

- uboot_script
- emPURS_plat
- kernel image
- ramdisk
- devicetree(as needed)

If you want to update the RootFS using its archive, you also have to locate the archive there.

The basic structure of the <NFS_SHARE> looks like as following.

<NFS_SHARE>/boot

9 Perform updating and booting

Booting the operating system and updating of the kernel and RootFS is supported by specific scripts of the U-Boot's environment and needs to set some environment variables. The most important U-Boot commands to manipulate variables require some basic U-boot operations. The most important U-Boot commands are listed in the following table.

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.

To work with U-Boot, first use a terminal program 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

9.1 Preparation of the U-Boot environment

In the case of updating the kernel or RootFS or booting via network, the U-Boot environment has to setup as following:

The variable serverip has to be set to the IP-address of the Host. You can get the IP-address by entering **ifconfig** in the terminal of the Host.

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]
```

The variable nfsroot has to be set to the exported <NFS_SHARE> at the Host. Assign the <NFS_SHARE> to the variable nfsroot in the U-Boot console.

```
UBoot # setenv nfsroot <NFS_SHARE>
```

Further settings have to be done dependent on using a dhcp or not. If using a dhcp the variable ip-method has assigned to dhcp, otherwise to static.

```
UBoot # setenv ip-method [dhcp or static]
```

In the case of static, the variables ipaddr and netmask need correct values.

```
U-Boot # setenv ipaddr [ip-address for device]
U-Boot # setenv netmask [netmask for device]
```

To make the changes persistent, enter saveenv in the U-Boot console.

```
UBoot # saveenv
```

9.2 Updating

9.2.1 Updating the RootFS

Update of the RootFS either by its archive or unpacked version is performed by the following command in the U-Boot console.

```
UBoot # run update_rootfs
```

9.2.2 Updating the kernel

Update of the kernel is performed by the following command in the U-Boot console.

```
UBoot # run update_kernel
```

9.3 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.

9.3.1 Booting from on-board flash

This is the default boot option configured when you receive the Dev-Kit from emtrion. To start it manually, simply use this command:

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

9.3.2 Booting via network using a NFS share

Booting via network enter following command in the U-Boot console

```
UBoot # run net_boot
```

10 SDK

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

One of the methods is creating a SDK by using the build directory. Use the following command for performing.

bitbake emtrion-devkit-image –c populate_sdk

The result is a toolchain installer containing the toolchain and the sysroot that includes and matches the target root file system. You will find the installer in

tmp/deploy/sdk in the build directory

10.1 Installing the SDK

Executing the toolchain installer, first you will asked for the installation directory. Then the installation process is performed.

10.1.1 Setting up the SDK environment

Before you can start with developing you have setup the environment.

For that open a terminal and source the setup script of the SDK. The setup script is stored in the installation directory and is visible by the string "environment-setup" as part of the script's filename.

If you want to develop a qt application you also have to set the qt environment. This is done by sourcing of the script qtopia.sh. The script is stored inside the subdirectory "environment-setup.d" in the installation directory.

10.1.2 Developing

After setting up the environment you can start with developing.

If you are interested in more information about developing applications using YP, you can find it here: http://www.yoctoproject.org/docs/1.7.2/adt-manual/adt-manual.html

10.2 Creating a RootFS archive

After developing and testing of the modified root file system, sometimes an archive of it is required. Due to the type of the archive is dependent on the Dev-Kit, following table informs how you can create an archive for a specific Dev-Kit.

Dev-Kit	command
dimm-am335x	tar –jcvf rfs_image-<MACHINE>.rootfs.tar.bz2 RFS_PATH
sbc-sama5d36	mkfs.ubifs –r RFS_PATH -o ./rfs_image-<MACHINE>.rootfs.ubifs -m 2048 -e 124KiB -c 3988

RFS_PATH: path of the root file system

11 Further Information

11.1 Online resources

Further information can be found on the emtrion support pages.

www.support.emtrion.de

11.2 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.