Yocto DevKit for SBC-RZN1D

Documentation

Rev003 / 23.04.2019

emtrion GmbH

Revision: **003 / 23.04.2019**

Rev	Date/Signature	Changes
001	Wi/04.06.2018	First release
002	Wi/06.08.2018	Minor changes in the bitbake process
003	Wi/23.04.2019	Rebranding changes

emtrion
embedded systems

Inhaltsverzeichnis

1 Introduction

This developer kit is based on the module SBC-RZN1D. The BSP contains a Linux kernel version 4.9.0, provided by Renesas and optimized and adapted by emtrion. The RootFS has been created with Yocto OpenEmbedded, Version 2.2.2 (Codename Morty). The base recipes are coming from Renesas, created for a special Renesas board, but they have been modified and extended by emtrion to perfectly fit the SBC-RZN1D.

This manual describes the contents of the developer kit DVD, how to set it up and gives a short overview on how to debug applications with Eclipse on the target.

It is assumed that users of emtrion Linux developer kits are already familiar with U-boot, Linux, Yocto and creating and debugging applications with Eclipse. 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 SBC-RZN1D
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
U-Boot	Bootloader, hardware initialization, updating images, starting OS
YP	Yocto Project
INST_DIR	Directory where Yocto and the meta-layers are installed
MACHINE	Specifies the target device for which the image is built. The variable is set to emconrzg1e (or emconrzg1m resp. emconrzg1h – depending on the used module) here
BUILD_DIR	Machine dependent build directory
BSP	Board Support Package
SDK	Software Development Kit

3 Preconfigured Linux virtual machine

emtrion delivers a DVD accompanying the Yocto Linux developer kits. This DVD contains a VMware virtual machine running Debian 8 (Jessie). VMware player or VMware Workstation is used to start the virtual machine. For general information about VMware Player please read following guide:

Getting started with VMware Player 12:

https://docs.vmware.com/en/VMware-Workstation-Player/12.0/workstation-player-12-windows-user-guide.pdf

https://docs.vmware.com/en/VMware-Workstation-Player/12.0/workstation-player-12-linux-user-guide.pdf

The virtual machine on the DVD is a compressed ZIP archive. Uncompress it on your system where it suits you. Please consider that the size of the virtual disk file will increase while you are working with it.

After uncompressing, basically two things have to be setup correctly to be able to use the virtual machine with our developer Kit:

- **Network Connection**: this should work directly, but depending on your PC you have to make adjustments. If so please read the guide, mentioned above (Chapter "Configuring Network Connections")

- **Serial Connection**: See p.100 in the guide above on how to use the serial port of your PC.

Some useful facts about the guest system inside the virtual machine:

- For login use **username: hico, password: hico** – it is recommended to change the password
- A TFTP Server is preinstalled and running by default. Its root directory is **/srv/tftp**
- Serial ports added to the virtual machine appear at **/dev/ttySn**, USB serial converters at **/dev/ttyUSBn**, baudrates can be configured using the stty tool, picocom can be used as a terminal
- An NFS server is running and exporting the directories in **/home/hico/yocto/nfs** and **/home/hico/yocto/sdk**
- The serial terminal program **picocom** for connecting to the target is installed in the VM
- Additional packages required by the build system Yocto
- The Yocto Layer **meta-emtrion-rzn1** can be found in /home/hico/yocto/build
- The folder **rzn1** in /home/hico/yocto/build contains the **dfu-util** which can optionally be used to update the U-boot bootloader on the target
- The version control software **git** is also installed

Due to the settings of the VM and getting a system with good performance, the host has to be equipped with sufficient RAM (8 GBytes), free disk space (150 GBytes) and at least 4 cores for compiling Yocto.

3.1 Device Start Up

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

sudo picocom -b 115200 /dev/ttySx

ttySx has to be replaced with the device assigned to the connected serial port e.g. ttyS1 . In the case of using an USB serial adapter replace it by the corresponding **ttyUSBn**.

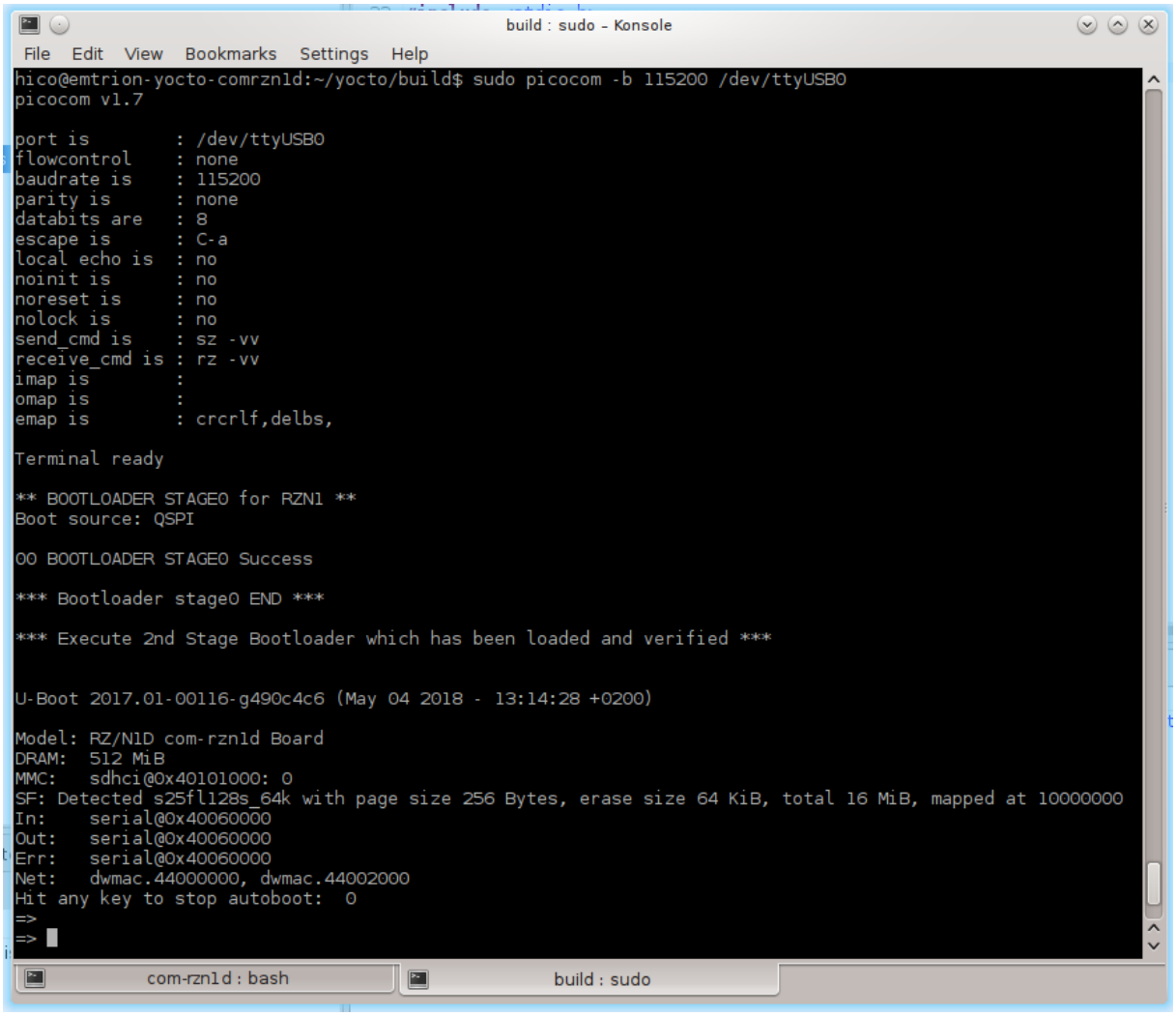

Figure 1: Serial terminal showing U-Boot prompt

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

After the developer kit booted you are prompted for login:

- **user: root**
- **password: no password set**

3.2 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” or “ip addr show eth0”.

```
root@com-rzn1d:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:0a:02:57:00
          inet addr:172.26.1.12  Bcast:172.26.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10027 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5131 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11589770 (11.0 MiB)  TX bytes:890910 (870.0 KiB)
          Interrupt:46 Base address:0xa000

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:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:140 (140.0 B)  TX bytes:140 (140.0 B)

usb0      Link encap:Ethernet  HWaddr ea:3e:d7:8e:e6:f5
          inet addr:192.168.7.2  Bcast:192.168.7.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@com-rzn1d:~#
```

Figure 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”.

Write down the IP address of the device. You need it later to setup the connection in Eclipse.

3.3 Preconfigured Directories

Some directories have been created and filled with precompiled images. Below are their descriptions.

Placeholder	Path	Remark
NFS_ROOTFS	/home/hico/yocto/nfs/rootfs	Root filesystem for mounting via NFS
NFS_RESTORE	/home/hico/yocto/nfs/restore	Location of the files for updating the onboard flashes
INST_DIR	/home/hico/yocto/build	Directory where Yocto and all the meta-layers will be stored to
BUILD_DIR	/home/hico/yocto/build/builddir	Build directory of the build system.
DOWNLOAD_DIR	/home/hico/Downloads	SDK as a self extracting bash script
SDK_DIR	/home/hico/yocto/sdk	SDK with tools, sources and libraries
WORKSPACE_DIR	/home/hico/workspace	Default Eclipse workspace
TFTP_DIR	/srv/tftp	Used by the TFTP server for exporting files

These variables are used as a place holder in this manual. They are set as environment variables.

3.4 Pre-built images and installations

In the directory **$NFS_RESTORE** are two images of the root file system located, one with and one without SDK extensions. In addition you can find the SDK as a self extracting bash script in the directory **$DOWNLOAD_DIR**.

To keep the virtual machine as small as possible for transport, the directory **$BUILD_DIR**, which usually contains about 25 GiB of Yocto installation files has been deleted. In order to create new root filesystems yourself, please follow the steps described in chapter 4. Of course emtrion has prepared everything you need to start developing immediately.

3.4.1 Target Root Filesystem

The target root filesystem for the SBC-RZN1D is available here:

$NFS_RESTORE/images/emtrion-image-sbc-rzn1d.tar.bz2

The extracted image can be found in the directory **$NFS_ROOTFS**.

3.4.2 SDK Root Filesystem

The SDK root filesystem containing the SDK extensions is available here:

$NFS_RESTORE/images/emtrion-image-sbc-rzn1d-sdk.tar.bz2

The extracted SDK image can be found in the directory

$SDK_DIR/sysroots/armv7vehf-vfpv4d16-poky-linux-gnueabi

The image with SDK extensions is for development purpose and is not suitable for normal use. You need this root filesystem during the development of your applications with Eclipse. How you can boot the installed root filesystem using NFS is described in the paragraph *Boot via network using a NFS share* of chapter 5.2.7.

3.4.3 Unpacking the Root Filesystems

You can unpack the provided root filesystems using

sudo tar xfvj $NFS_RESTORE /<rootfs-tar> -C <target-directory>

4 The meta-layer for SBC-RZN1D

NOTE: If you do not want to create new root filesystems, you can skip this chapter.

The meta-layer for SBC-RZN1D is the meta-layer provided by Renesas for RZ/N1 SoCs with some additions to adapt this meta-layer to the emtrion boards. You find this meta-layer in the **$INST_DIR** directory.

4.1 Preparing the environment

Before creating an image, you will have to prepare the environment accordingly. First go to the **$INST_DIR/meta-emtrion-rzn1** directory and execute the setup script:

source setup_environment

This will download all needed files and prepare the environment. When finished you will find your shell prompt at **$INST_DIR/builddir/emtrion/machines/sbc-rzn1d**.

4.2 Creating the images

Now it's time to start building images for the emtrion board.

The build process is done in three steps. The first step creates an initial ramdisk which contains emPURS (emtrion Production, Update and Recovery System):

bitbake core-image-purs

The second one creates the target root filesystem image:

bitbake emtrion-image-sbc-rzn1d

The third creates the SDK root filesystem image:

bitbake emtrion-image-sbc-rzn1d-sdk

After you have created the SDK root filesystem image, you can now generate the SDK installer:

bitbake emtrion-image-sbc-rzn1d-sdk -c populate_sdk

In order to compile the U-Boot image, use the following:

bitbake u-boot-rzn1

In order to generate the bootloader images, use the following:

bitbake u-boot-rzn1 -c install

You will find the U-Boot images in the following directory:

$INST_DIR/builddir/emtrion/machines/sbc-rzn1d/tmp/work/sbc_rzn1d-poky-linux-gnueabi/u-boot-rzn1/git-r0/build/

If you only want to rebuild the Linux kernel image, do the following:

bitbake linux-rzn1

You will then find the zImage in the following directory:

$INST_DIR/builddir/emtrion/machines/sbc-rzn1d/tmp/deploy/images/sbc-rzn1d/

4.3 Output files

During the build process lots of objects and images are created. However, the most relevant images are installed into

$BUILD_DIR/emtrion/machines/sbc-rzn1d/tmp/deploy/images/sbc-rzn1d

and

$BUILD_DIR/emtrion/machines/sbc-rzn1d/tmp/deploy/sdk

The bootloader images can be found in

$BUILD_DIR/emtrion/machines/sbc-rzn1d/tmp/work/sbc_rzn1d-poky-linux-gnueabi/u-boot-rzn1/git-r0/build

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

Images	description
u-boot.bin[1]	U-Boot
zImage-sbc-rzn1d.bin[1]	Linux Kernel
zImage-sbc-rzn1d.dtb[1]	Device tree
emtrion-image-sbc-rzn1d.tar.bz2	RootFS
emtrion-image-sbc-rzn1d-sdk.tar.bz2	RootFS for SDK
poky-glibc-x86_64-emtrion-image-sbc-rzn1d-sdk-armv7vehf-vfpv4d16-toolchain-2.2.2.sh	SDK installer

[1] symbolic link

4.3.1 Root Filesystem

As shown in the list above, the output of the root file system is a tar.bz2 archive. You can decompress it using the tar command. For test purpose we recommend to decompress the archive to the **$NFS_ROOTFS**. Navigate to the directory **$BUILD_DIR** and call

sudo tar xfvj emtrion/machines/sbc-rzn1d/tmp/deploy/images/sbc-rzn1d/<rootfs-name>.tar.bz2 -C $NFS_ROOTFS

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

4.3.2 boot directory

The directory structure of the root file system includes a directory /boot. In addition to the kernel image, device tree, restore script and restore root file system and U-Boot binary a file **uboot_script** is located there.

This text file implements some U-Boot command sequences. You can use it for the purpose of updating and booting via network, like kernel, U-Boot and RootFS.

However, the environment of the U-Boot has to be set up before. This is discussed in detail in chapter 5.2.1.

4.4 Further readings on Yocto

Yocto reference manual: https://www.yoctoproject.org/docs/2.2/ref-manual/ref-manual.html

Yocto SDK manual: https://www.yoctoproject.org/docs/2.2/sdk-manual/sdk-manual.html

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

Yocto repositories: https://git.yoctoproject.org/

5 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 RootFS and of U-Boot itself. Furthermore you can configure the medium the operating system should be booted from, for example eMMC, NFS or a serial terminal.

5.1 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.

5.2 Using U-Boot to change boot device or update parts of 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 entering **ip addr** 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]
```

5.2.1 Boot setup and updating the root file system

Updating the kernel or the root file system is done by using the NFS export of the VM.

Updates of the boot loader are performed using a TFTP server, which has to run in the VM. It is already set up to run, when you boot it for the first time. The root folder of the TFTP server is **/srv/tftp**. Every file which is needed in the update process must be copied into this folder.

File names:

The files needed in the update process need to have specific file names. These file names are specified in the following environment variables of U-Boot:

Variable name	Default settings	Description
image.linux	zImage	Filename of the Linux kernel image
image.uboot	u-boot.img	Filename of the U-Boot image

5.2.2 Updating Linux kernel and root filesystem (using NFS)

To update both the kernel and the root filesystem you must copy the image file (.tar.bz2 file) to **$NFS_RESTORE** in your VM. This file has already been placed there by emtrion and can be created again during the bitbaking process. The name of this image is **emtrion-image-sbc-rzn1d.tar.bz2** or **emtrion-image-sbc-rzn1d-sdk.tar.bz2**. Now you can use the following commands in the U-Boot prompt:

```
U-Boot # setenv serverip        [ip-address of VM]
U-Boot # setenv nfsroot /home/hico/yocto/nfs/restore
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 filesystem image via network and decompressing it to the flash storage can take a few minutes.

5.2.3 Updating of U-Boot bootloader (using TFTP)

Attention: If the board is turned off while updating the bootloader or another error occurs, the board will be rendered unusable to you. Please only update the bootloader if you are explicitly instructed to do so by emtrion.

If you have generated a new bootloader image **u-boot.img** as described in 4.2, copy it to the TFTP folder **/srv/tftp** in the VM. Otherwise you can use the image emtrion has already copied to **/srv/tftp**. Then you can start the update process with the following command in U-Boot prompt:

```
U-Boot # setenv serverip        [ip-address of VM]
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 # run update_uboot
```

5.2.4 Updating of U-Boot SPL (using TFTP)

Attention: If the board is turned off while updating the bootloader or another error occurs, the board will be rendered unusable to you. Please only update the bootloader if you are explicitly instructed to do so by emtrion.

It might be necessary to update the U-Boot SPL. The SPL binary **spl-rel.spkg** has to be in the TFTP folder **/srv/tftp** in the VM. emtrion has already copied the current SPL binary to **/srv/tftp**. Then you can start the update process with the following command in U-Boot prompt:

```
U-Boot # setenv serverip         [ip-address of VM]
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 # run update_spl
```

5.2.5 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.

5.2.6 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
```

5.2.7 Boot via network using a NFS share

In order to boot via network you have to perform the following commands in U-Boot:

```
U-Boot # setenv serverip         [ip-address of VM]
U-Boot # setenv nfsroot          [path to the RootFS in VM]
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 # saveenv
U-Boot # run net_boot
```

Now the board should boot via network using the NFS share in VM.

6 SDK

In order to develop applications outside the Yocto build system (= outside the directory **$BUILD_DIR**) you need to set up your host development system. For this purpose the YP offers several installation methods.

The bitbaking process described in chapter 4.2 creates an SDK installer containing the toolchain and the sysroot, which includes and matches the target root file system. The installer is stored in

$BUILD_DIR/emtrion/machines/sbc-rzn1d/tmp/deploy/sdk

You don't have to do the steps of bitbaking in chapter 4.2 to create the SDK installer yourself as emtrion has done that for you. You can find the installer in

$DOWNLOAD_DIR

6.1 Installing the SDK

NOTE: emtrion has already done this step for its customers. You find the installed SDK in the directory **$SDK_DIR**.

In order to install the SDK, go to

$BUILD_DIR/emtrion/machines/sbc-rzn1d/tmp/deploy/sdk

or

$DOWNLOAD_DIR

and execute the installer

./poky-glibc-x86_64-emtrion-image-sbc-rzn1d-sdk-armv7vehf-vfpv4d16-toolchain-2.2.2.sh

You are asked for the installation directory. You can either accept the suggested one or create one yourself.

6.1.1 Setting up the SDK environment

Before you can start developing apps 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-armv7vehf-vfpv4d16-poky-linux-gnueabi

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

6.1.2 SDK Root Filesystem

Note, that in the directory **$SDK_DIR/sysroots/armv7vehf-vfpv4d16-poky-linux-gnueabi** you will find the unpacked SDK Root Filesystem.

In your development phase it's recommended to use this SDK root filesystem on the target by setting the U-Boot environment variable *nfsroot* accordingly. Boot the target and stop the boot process by hitting a key. Then set the *nfsroot* variable

setenv nfsroot /home/hico/yocto/sdk/sysroots/armv7vehf-vfpv4d16-poky-linux-gnueabi

and the *serverip* variable to your VM's IP address

setenv serverip [ip address]

Then if desired save the environment

saveenv

Now you can boot the target

run net_boot

and start developing and debugging on the target with the Eclipse IDE as described in the next chapter.

6.2 Developing and Debugging with the Eclipse IDE

Using the popular IDE Eclipse and the Yocto Eclipse Plugin you can develop and debug your application directly on the SBC-RZN1D target. emtrion has already setup Eclipse and the Yocto plugin, so that you only have to change the IP address of your target.

In order to change the IP address in the Eclipse configuration, navigate to the directory **$WORKSPACE_DIR**.

NOTE: Make sure, Eclipse is currently NOT running.

Now replace the default IP address with your target's IP address. Open the file

nano .metadata/.plugins/org.eclipse.core.runtime/.settings/org.eclipse.remote.core.prefs

and change the IP address at the end of the first line. Then open the second file

nano .metadata/.plugins/org.eclipse.rse.core/profiles/PRF.emtrion-yocto-sbcrzn1d_0/H.sbc-rzn1d_0/node.properties

and change the IP address at the end of the fifth line.

Then start Eclipse. To do so click on the blue icon in the bottom left corner and afterwards click on the Eclipse IDE icon.

After Eclipse has started you will find an example project called *example* with a file called *example.c* which you can use as a blueprint for your projects.

6.2.1 Run application on target

You can run the example project by clicking on the arrow to the right of the run icon and choose *example_gdb_arm-poky-linux-gnueabi* as shown in figure 3.

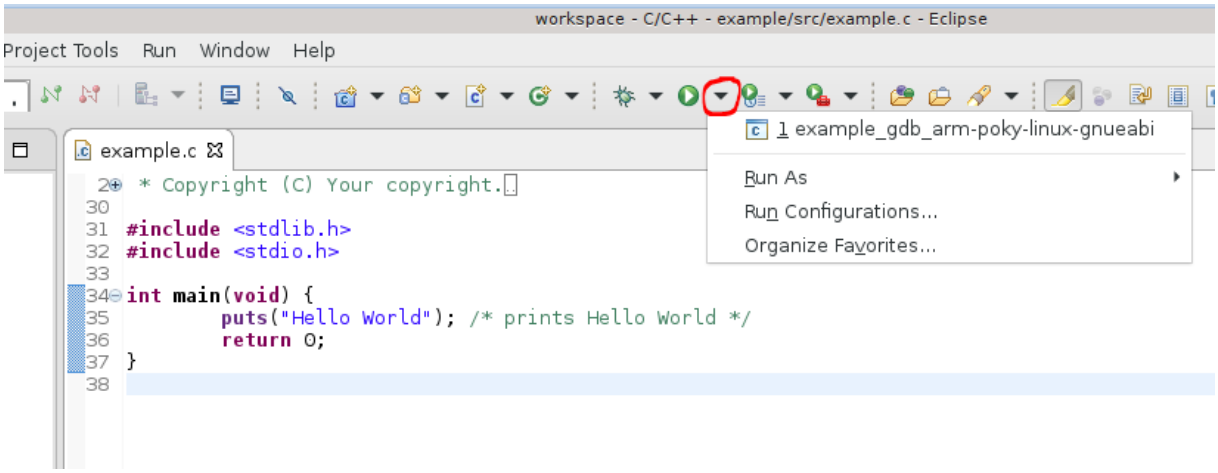

Figure 3: Run application on target

You might be asked to accept the new fingerprint of the connection, if so click "Yes".

6.2.2 Debug application on target

You can debug the example project by clicking on the arrow to the right of the debug icon and choose *example_gdb_arm-poky-linux-gnueabi* as shown in figure 4.

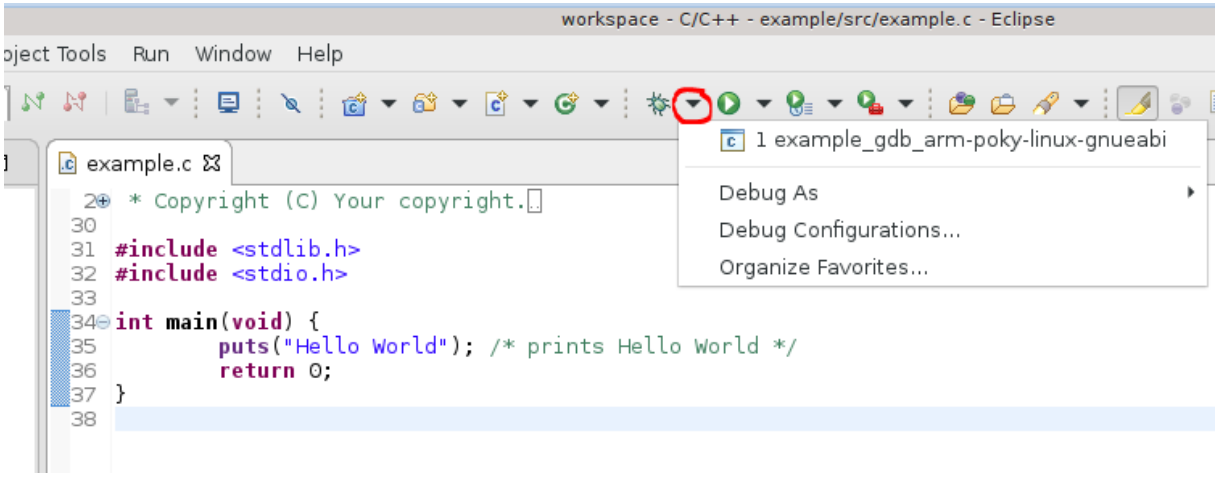

Figure 4: Debug application on target

6.2.3 Create project

A detailed description on how to create a new project can be found in the Yocto SDK manual:

https://www.yoctoproject.org/docs/2.2/sdk-manual/sdk-manual.html#neon-creating-the-project

6.2.4 Build project

A detailed description on how to build the new project can be found in the Yocto SDK manual:

https://www.yoctoproject.org/docs/2.2/sdk-manual/sdk-manual.html#neon-building-the-project

6.2.5 Adjust run configuration on new project

If you create a new project you have to adjust the run configuration. You can either compare the settings in the *example* project or follow below instructions.

Note: Build your project as described in chapter 6.2.4 first.

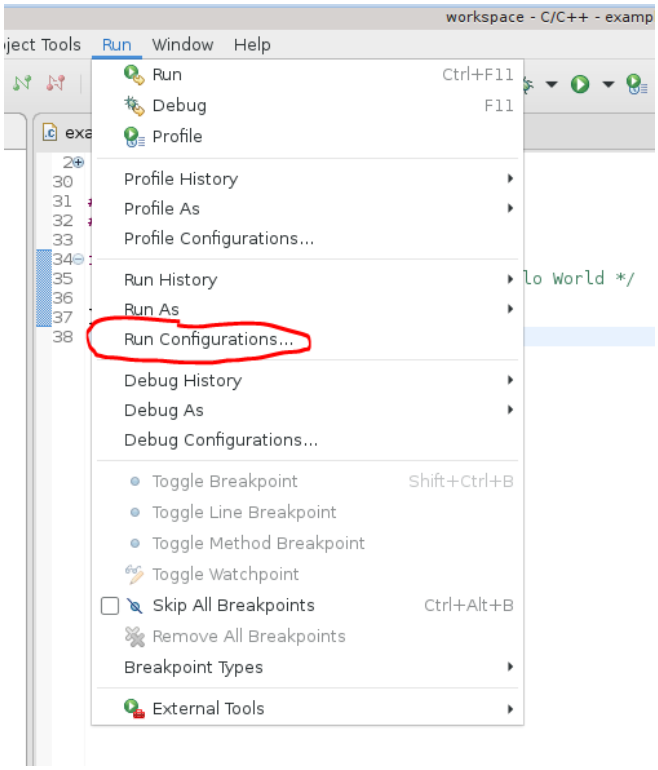

Figure 5: Select "Run configurations..."

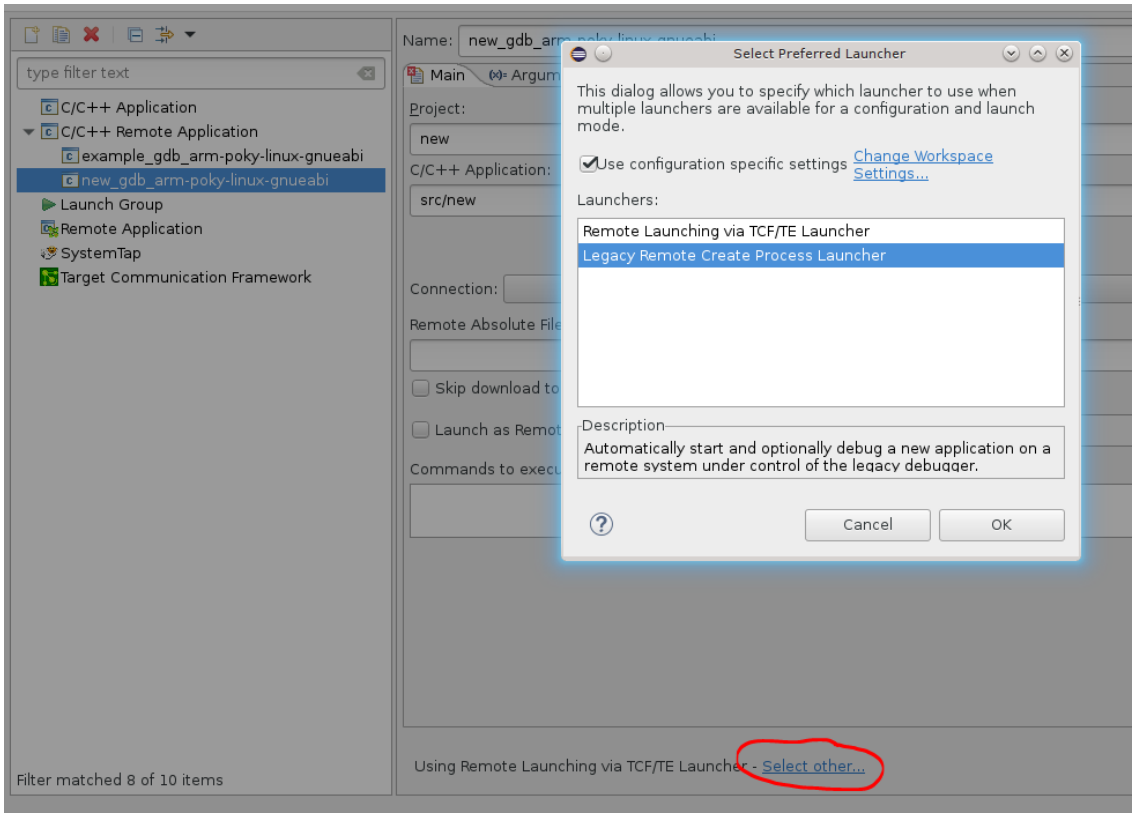

Figure 6: Select other... -> Legacy Remote Create Process Launcher

Figure 7: Choose "Remote Host"

In the line “Remote Absolute File Path for C/C++ Application” as seen in figure 8 insert:

/home/root/<name_of_your_project>

In the line “Commands to execute before application” as seen in figure 8 insert:

chmod 777 /home/root/<name_of_your_project>

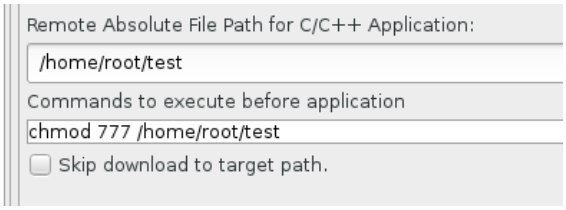

Figure 8: Adjust file settings

Now you can click on “Run” in the bottom right corner and the application should be executed on the target.

6.2.6 Adjust debug configuration on new project

If you create a new project you have to adjust the debug configuration. You can either compare the settings in the *example* project or follow below instructions.

Note: Build your project as described in chapter 6.2.4 first.

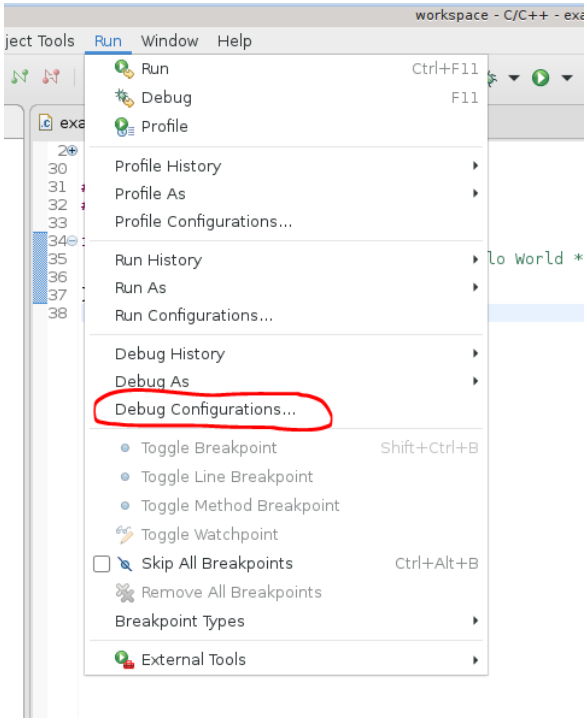

Figure 9: Choose "Debug Configurations..."

If you've already configured the run configuration you only have to change the GDB path as shown in figure 10, otherwise it's supposed to be configured as shown in figure 11.

In the line GDB debugger add the following:

/home/hico/yocto/sdk/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb

And in the line GDB command file add the following:

/home/hico/workspace/<name_of_your_project>/.gdbinit

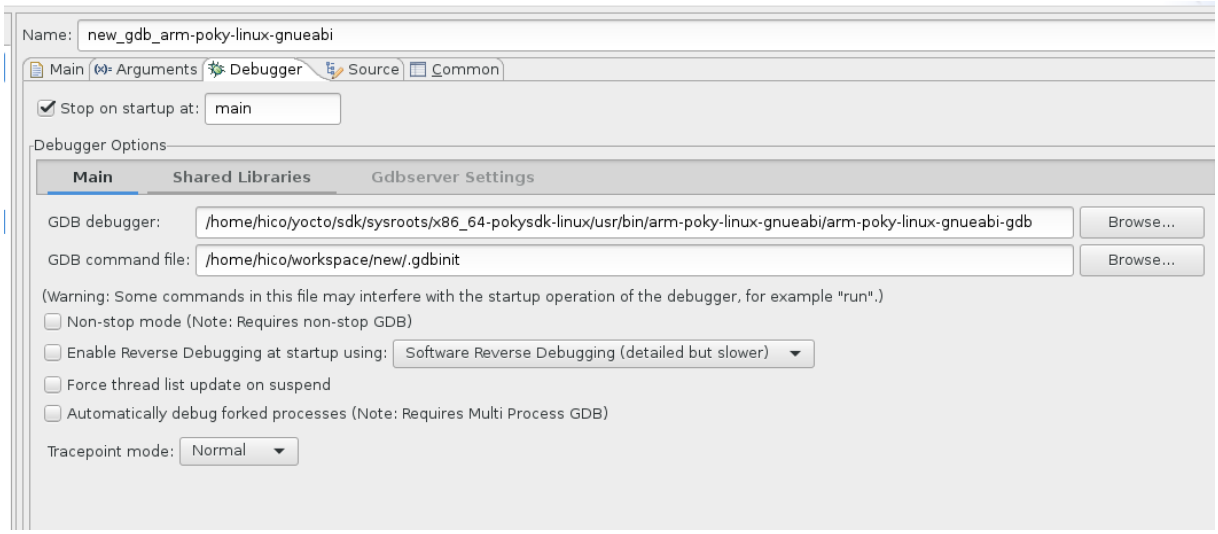

Figure 10: Choose correct GDB debugger and GDB command file

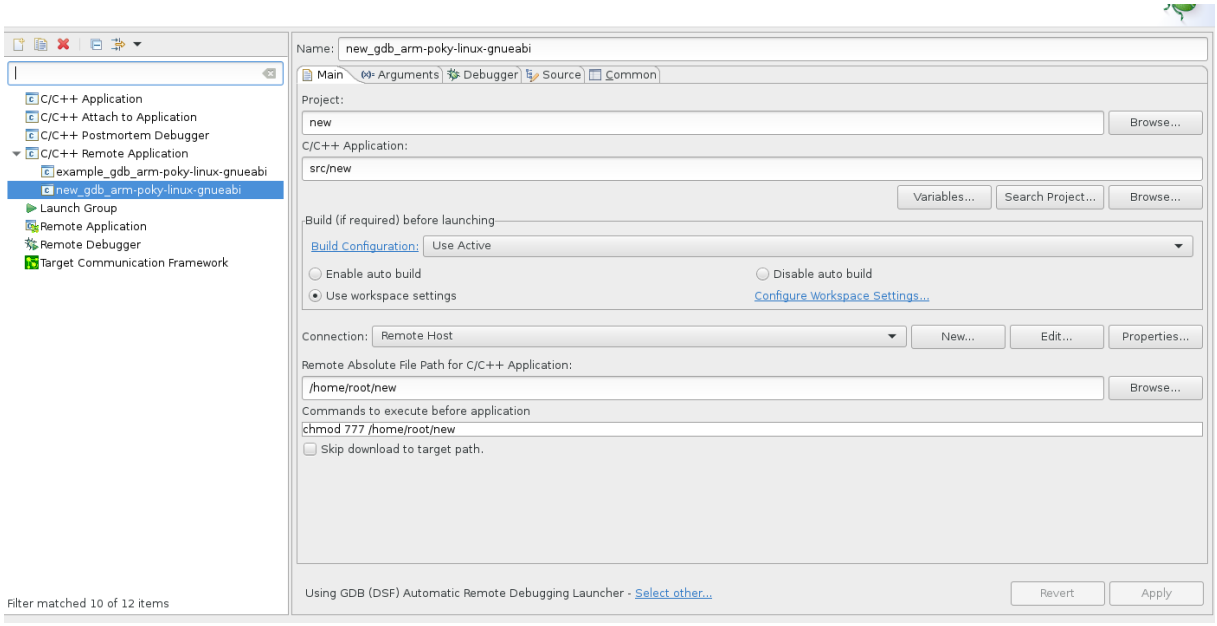

Figure 11: Debug configuration settings

Afterwards you can click on "Debug" in the bottom right corner and you should see the Debug view opening.

7 Using the Developer Kit without the VM

If you want to, you can use the developer kit contents on a native Linux machine. You have to consider the system requirements of Yocto 2.2 (Morty). You will find these requirements here: https://www.yoctoproject.org/docs/2.2/ref-manual/ref-manual.html#intro-requirements

Important are the supported Linux Distributions and the required packages which must be installed before you can start. Also you should install and configure an NFS server and a TFTP server. The NFS server should export the directory **$NFS_ROOTFS** and **$NFS_RESTORE**. The TFTP server must use **$TFTP_DIR** as its base directory (see 3.3 for the path).

You also have to copy the complete contents of the */home/hico/yocto* folder of the VM. In the native machine, you have to place the copied folder within the home directory of your user. After you have done this you should declare the placeholder listed in table in chapter 3.3 as environment variables. If you have done this you can use the developer kit as it is described in the corresponding chapters above.

8 Further Information

8.1 Online resources

Further information can be found on the emtrion support pages.

https://support.emtrion.de

8.2 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.