Linux

For

BCT TM1 / HB5

User Guide

Document Reference: BCTTM1HB5 Linux User Guide

Document Issue: 1.8.1

Associated SDK release: 2.03

Associated TM1 Update Utility release: 1.27

Contents

1. Introduction

The content of this document provides information required to start building Linux operating systems for the BCT TM1 / HB5 platform. It covers:

- The tools and components required for building a Linux operating system
- How to install the build components
- How to compile the U-Boot boot loaders stand alone
- How to compile the Linux Kernel 3.14 stand alone
- How to compile the Linux Kernel 4.9 stand alone
- How to setup a root filesystem using Ubuntu 14.04 LXDE
- How to setup a root filesystem using Ubuntu 18.04 LXDE
- How to build a root filesystem including QT5 using build root
- How to boot Linux on the TM1/HB5 platform
- How to setup and deploy a simple QT5 application to TM1/HB5

2. Environment Setup

2.1 Embedded Linux Components

The components involved in a typical Embedded Linux system targeting the ARM architecture are:

1. Bootloader (Typically uboot)
2. Linux Kernel
3. Root filesystem.

U-boot 2014.04 was ported to provide the bootloader functionality for the TM1/HB5.

Linux kernel 3.14.28 and 4.9.88 have been ported to be compatible with the BCT TM1/HB5 platform.

Pre-built Ubuntu root filesystems are provided for demonstration purposes. As an alternative to Ubuntu, a Buildroot environment is provided to allow bespoke root filesystems to be generated for the TM1/HB5 platform. Section 3 describes the procedure for building an image with the Ubuntu filesystem; section 4 describes the procedure for building an image with Buildroot.

The TM1 software components above have all been tested to compile using an Ubuntu 18.04 LTS development machine.

2.2 Installation of the Embedded Linux build components

Create the top level build BSP directory and grant it universal read/write/execute access as follows:

```
cd /
sudo mkdir embedded
sudo chmod 777 embedded
```

Copy the latest TM1/HB5 Linux components to the "/embedded" directory. Sources can be distributed in different ways, but usually they can be downloaded from our web site.
https://www.bluechiptechnology.com/product/tm1/

Download the Linux source code for TM1/HB5 using the command:

```
wget http://dl.bluechiptechnology.com/dl/tm1/software/tm1linuxv203.tar.bz2
wget
http://dl.bluechiptechnology.com/dl/tm1/software/tm1linuxv203.tar.bz2.md5
```

Check that the integrity of the download is ok by issuing the following command:

```
md5sum -c tm1linuxv203.tar.bz2.md5
cd /
```

Extract the tar ball by issuing the command:

```
tar xvpjf tm1linuxv203.tar.bz2
```

Setup git to pull latest code from Bluechip Technology

Kernel 3.14 (GCC4.9):

```
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga/linux-tm1/
git remote rm origin
git remote add origin
http://dl.bluechiptechnology.com/dl/tm1/software/linux/L3.14.28_1.0.1_ga/li
nux-tm1.git
git remote update
git branch --set-upstream-to=origin/master master
git pull
```

Kernel 4.9 (GCC7.3):

```
cd /embedded/projects/tm1/L4.9.88_2.0.0/linux-tm1/
git remote rm origin
git remote add origin
http://dl.bluechiptechnology.com/dl/tm1/software/linux/L4.9.88_2.0.0/linux-
tm1.git
git remote update
git branch --set-upstream-to=origin/imx_4.9.88_2.0.0_ga imx_4.9.88_2.0.0_ga
git pull
git show
```

Once extracted the build components will be laid out in the following structure on the development machine. The BSP is capable of building images containing either kernel 3.14 or kernel 4.9. The first directory ("embedded") is the folder created in the root of the filesystem.

Directory	Directory	**Description**
/embedded/toolchains	gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabihf	Prebuilt cross compiling tool chain based on GCC 4.9, for building ARMhf software.
	gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabihf	Prebuilt cross compiling tool chain based on GCC 7.3, for building ARMhf software.
/embedded/projects/tm1/	u-boot-tm1	Source code for the U-boot boot loader including configuration for BCT TM1/HB5
/embedded/projects/tm1/L3.14.28_1.0.1_ga	linux-tm1	3.14.28 kernel source code with configuration for BCT TM1/HB5
	rootfs	Staging directory used for holding the Ubuntu root filesystem of the target device during development
	rootfsimages	Directory containing prebuilt root filesystems, including: Demo Ubuntu 14.04 image distributed with TM1/HB5. Quickboot demonstration image that will play videos installed in the root of a USB flash drive. Built with buildroot using tm1_mplayerquickbootdemo_defconfig. QT5 demonstration image including sample applications. Built with Buildroot using tm1_qt5sample_defconfig
	buildroot-2016.02	Buildroot 2016.02 release with configurations for building rootfs images, linux kernel, and linux device tree blobs for TM1.
	wilink8-build-utilites	Source code to the software that supports the Wi-Fi and BT module implemented on TM1. Cloned and configured from https://git.ti.com/wilink8-wlan/build-utilites
/embedded/projects/tm1/L4.9.88_2.0.0	linux-tm1	4.9.88 kernel source code with configuration for BCT TM1/HB5
	rootfs	Staging directory used for holding the Ubuntu root filesystem of the target device during development
	rootfsimages	Directory containing prebuilt root filesystems, including: Demo Ubuntu 18.04 image distributed with TM1/HB5. Quickboot demonstration image that will play videos installed in the root of a USB flash drive. Built with buildroot using tm1_mplayerquickbootdemo_defconfig.

		QT5 demonstration image including sample applications. Built with Buildroot using
	buildroot-2018.02.8	Buildroot 2018.02 release with configurations for building rootfs images, linux kernel, and linux device tree blobs for TM1.

2.3 Development Machine Setup

Where possible build scripts have been provided for the various components included with the Linux SDK for BCT TM1/HB5. These scripts presume the following has been setup on the Ubuntu 18.04 LTS development machine.

- A TFTP server serving files from a tftpboot directory in the root of the filesystem. (/tftpboot)
- An NFS server serving files from /nfs.

The following links provide information on setting up an NFS and TFTP server.

http://www.debianhelp.co.uk/tftp.htm

Setup Required

```
sudo apt-get install tftpd
sudo mkdir /tdtpboot
sudo chmod 777 /tftpboot
```

http://www.debianhelp.co.uk/nfs.htm

Setup Required

```
sudo apt-get install nfs-kernel-server nfs-common portmap
cd /
sudo mkdir nfs
cd nfs
sudo mkdir rootfs
cd /
```

In the file /etc/exports , add the line

*/nfs/rootfs *(rw,sync,no_root_squash)*

The /nfs directory should be symbolically linked to the root staging directory (see table above). From the /nfs directory issue the following command.

```
ln -s /embedded/projects/tm1/L3.14.28_1.0.1_ga/rootfs 3.14.rootfs
ln -s /embedded/projects/tm1/L4.9.88_2.0.0/rootfs 4.9.rootfs
```

The following packages are known to be required on an Ubuntu 14.04 development machine to successfully build the components. It is recommended that these components are installed to later versions of Ubuntu. v18.04 has also been trialled.

```
sudo apt-get install build-essential
sudo apt-get install u-boot-tools
sudo apt-get install flex

sudo apt-get install lzop
sudo apt-get install ncurses-dev
```

Before compiling various stand alone Linux components we must set some environment variables. This is to ensure the configuration tools build for the correct architecture and can find the cross compiling tool chain. To make this task simpler a script file is provided to configure the environment for a BCT TM1/HB5 build. Issue the following commands to run the script:

Kernel 3.14 (GCC4.9):

```
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga
source ./setenv-hfp.sh
```

Kernel 4.9 (GCC7.3):

```
cd /embedded/projects/tm1/L4.9.88_2.0.0
source ./setenv-hfp.sh
```

3. Building required components for Ubuntu Core

3.1 Installing the Ubuntu Core root filesystem

There are many Linux distributions available that are compatible with BCT TM1/HB5. Ubuntu was chosen as the main distribution for TM1, as it has a large pre-compiled package database, and easy to use configuration tools. Ubuntu is not the ideal choice for all Linux projects, but it will allow a balsic operating system to be constructed quickly to allow evaluation of the BCT TM1/HB5. For smaller embedded OS requirement consider using Buildroot to generate a root filesystem from scratch. See section 4 for details.

The process of creating an Ubuntu root filesystem for BCT TM1/HB5 consists of the following steps.

- Extract an Ubuntu image into the staging directory
- Add kernel modules and other specific support to the root filesystem.
- Boot the generated Ubuntu root filesystem on a BCT TM1/HB5.
- Configure the Ubuntu root filesystem using, apt-get, synaptic package manager, or another package manager.

To support the BCT TM1/HB5 hardware and kernel, various files need to be copied to the root filesystem. To simplify this process all build components that need to modify the root filesystem are configured to do so at the nfs staging location, "/nfs/rootfs". We must extract a root filesystem as a starting point to this location.

For convenience a pre-built root filesystem is included in the TM1/HB5 download. It can be extracted using the following commands.

Kernel 3.14 (GCC4.9):

```
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga
./extractubunturootfsimage1.26.sh
```

Kernel 4.9 (GCC7.3):

```
cd /embedded/projects/tm1/L4.9.88_2.0.0
./extractubunturootfsimage1.26.sh
```

At this point Linux Kernel modules, and any other specific support must be added to the root filesystem. The following sections describe this process.

3.2 Compiling the Linux Kernel

To compile the kernel we mxjust enter the root of the kernel source tree, make some configuration changes and use `make` to start the compile. Issue the following commands. The process for compiling kernel 3.14 and kernel 4.9 is identical apart from the root directory where the build commands are issued:

Kernel 3.14:

```
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga
source ./setenv-hfp.sh
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga/linux-tm1/
```

Kernel 4.9:

```
cd /embedded/projects/tm1/L4.9.88_2.0.0
source ./setenv-hfp.sh
cd /embedded/projects/tm1/L4.9.88_2.0.0/linux-tm1/
```

```
./build.sh
```

When the compilation process has completed it will leave a Linux kernel (zImage) at, "./arch/arm/boot/zImage", and, "/tftpboot/zImage".

The TM1/HB5 kernel implements the device tree model for configuring a hardware platform. The TM1/HB5 kernel includes four configurations for the various versions of the HB5 host board.

Device tree definition	Description
tm1-hb5-43-c.dtb	HB5 with 4.3 Inch LCD and capacitive touch
tm1-hb5-43-r.dtb	HB5 with 4.3 Inch LCD and resistive touch
tm1-hb5-7-c.dtb	HB5 with 7 Inch LCD and capacitive touch
tm1-hb5-7-r.dtb	HB5 with 7 Inch LCD and resistive touch
tm1-hb5-9-c.dtb	HB5 with 9 Inch LCD and capacitive touch
tm1-hb5-cb3-43-c.dtb	HB5 + CB3 with 4.3 Inch LCD and capacitive touch
tm1-hb5-cb3-43-r.dtb	HB5 + CB3 with 4.3 Inch LCD and resistive touch
tm1-hb5-cb3-7-c.dtb	HB5 + CB3 with 7 Inch LCD and capacitive touch
tm1-hb5-cb3-7-r.dtb	HB5 + CB3 with 7 Inch LCD and resistive touch
tm1-hb5-cb3-9-c.dtb	HB5 + CB3 with 9 Inch LCD and capacitive touch

build.sh is an example of a script file that simplifies the process of building BCT TM1 /HB5 Linux components. Please study these files for an understanding of their purpose.

If changes are required to the kernel configuration the command "`make menuconfig`" can be used to present a menu based configuration utility for the Linux kernel. If any changes are made using the menuconfig tool, the "`./rebuild.sh`" command must be issued.

If you change the configuration your new configuration can be saved with

```
make savedefconfig
cp defconfig /arch/arm/configs/tm1_defconfig
rm defconfig
```

Once the kernel has been compiled, the kernel modules must be copied to the root filesystem. Issuing the following command performs this task.

```
sudo ./installmodulesrootfs.sh
```

The modules are installed to the rootfs directory where the root filesystem that was previous extracted to in section 3.1.

Note:

zimage can be found in sdcard partition1 /dev/mmcblk0p1

The modules generated by the build process can be copied directly to the SDcard or emmc in the /lib/modules directory to avoid a complete reprogram using the TMx-Reprogramming utility

Note: zimage can be found in sdcard partition1 /dev/mmcblk0p1

3.2.1 Compiling the Linux Kernel 4.9 with modularised WiFi drivers

Linux kernel 4.9 for TM1 on-board WiFi can be compiled either with embedded WiFi driver or modular WiFi driver. The Linux kernel build config file used in the previous section (**tm1_defconfig**) builds the WiFi driver as embedded. To build the Linux kernel with modularised WiFi drivers use **tm1wl_defconfig** configuration file. The pros and cons for these two build options are as follows:

Embedded WiFi driver:

- A module file (.ko) is not required to be present on the root file system to make the WiFi functional. This may reduce root-fs size for size critical systems.
- WiFi Firmware blobs are embedded in the kernel binary. This reduces the size and maintenance of the root-fs, but the firmware blobs can't be updated without recompiling the whole kernel.
- TI's 'wlconf' tool can't be used to dynamically change and test WiFi options that affect speed and connection quality.

Modularised WiFi driver:

- Matching module files (.ko) are required to be present on the root file system to make the WiFi functional. This adds size and maintenance overhead.
- WiFi firmware blobs are required to be present on the root file system:
    - /lib/firmware/ti-connectivity/TIInit_11.8.32.bts
    - /lib/firmware/ti-connectivity/wl18xx-conf.bin

    - /lib/firmware/ti-connectivity/wl18xx-fw-4.bin
    - /lib/firmware/TIInit_11.8.32.bts
    - /lib/firmware/regulatory.db
    - /lib/firmware/regulatory.db.p7s
- TI's 'wlconf' tool can be used to dynamically change and test WiFi options that affect speed and connection quality.

Note: ensure the Linux kernel git repository is updated to be able to build the modularised WiFi driver (see section 2.2 for more information).

3.3 U-Boot Bootloader – Ported (http://www.denx.de/wiki/U-Boot)

U-Boot 2014.04 has been ported to work with TM1/HB5. Its purpose is to initialise the hardware, and boot a Linux operating system.

To build U-Boot for BCT TM1/HB5 issue the following commands.

Kernel 3.14 & Kernel 4.9:

```
cd /embedded/projects/tm1/uboot-tm1mak
./buildlinuxtm1hb5.sh
```

The compiled boot loader is "u-boot.imx". The script file also copies the boot loader the /tftpboot directory.

3.4 WiLink8 Wi-FI/BT support

This section only applies to images using kernel 3.14.

To support the Wi-Fi/BT module implemented on TM1, various software components must be installed in the root filesystem. These components include, Wi-Fi driver kernel modules, Wi-Fi firmware, BT firmware, and BT UIM utility. To install the components in the Ubuntu core root filesystem issue the following commands.

```
cd /embedded/projects/tm1/L3.14.28_1.0.0_ga/wilink8-build-utilites
./buildtm1linux.sh
```

The script file uses the setup-envlinux environment configuration, which tells the tool where the cross compiling toolchain, Linux kernel, and root filesystem are located.

After running the buildtm1linux.sh script the various software components will be present in the target root filesystem.

3.5 Ubuntu Core system components summary

So far this document has described how to set up a build environment and how to build the various components of a Linux Ubuntu Core operating system for BCT TM1/HB5. The components we have built are as follows:

Kernel 3.14

Component	Location
Ubuntu Root filesystem	/embedded/projects/tm1/L3.14.28_1.0.1_ga/rootfs
Linux Kernel	/embedded/projects/tm1/L3.14.28_1.0.1_ga/linux-tm1/arch/arm/boot/zImage
Device Tree Configurations	/embedded/projects/tm1/L3.14.28_1.0.1_ga linux-tm1/arch/arm/boot/dts/*.dtb
U-Boot	/embedded/projects/tm1/uboot-tm1/u-boot.imx

Kernel 4.9

Component	Location
Ubuntu Root filesystem	/embedded/projects/tm1/L4.9.88_2.0.0/rootfs
Linux Kernel	/embedded/projects/tm1/L4.9.88_2.0.0/linux-tm1/arch/arm/boot/zImage
Device Tree Configurations	/embedded/projects/tm1/L4.9.88_2.0.0/linux-tm1/arch/arm/boot/dts/*.dtb
U-Boot	/embedded/projects/tm1/uboot-tm1/u-boot.imx

The install program replaces hb5.dtb on mmcblk0p1 with the contents of the appropriate dtb for the selected system.

3.6 Kernel 4.9 Persistant logo boot

A persistant logo boot option has been added to the 4.9 kernel, this hides the normal frame buffer until a command is sent to display it.

This feature can be enabled from the kernel command line or by setting it active with DTB option

Enabling using u-boot

Power on TM1 with a serial terminal connected to the debug com port

Press SPACE during the detection period to enter uboot shell

(very fast by default – press SPACE repeatedly as you power on)

Add persistantlogo=1 to mmcargs (or to netargs if booting by PXE)

Eg. change

mmcargs=setenv bootargs console=${console},${baudrate} root=${mmcroot} loglevel=0 consoleblank=0

To

mmcargs=setenv bootargs console=${console},${baudrate} root=${mmcroot} loglevel=0 consoleblank=0 persistantlogo=1

With the command

setenv mmcargs setenv bootargs console=${console},${baudrate} root=${mmcroot} loglevel=0 consoleblank=0 persistantlogo=1

Then save the environment with

saveenv

Reboot the TM1

Enabling using DTB

In the file

/embedded/projects/tm1/L4.9.88_2.0.0/linux-tm1/arch/arm/boot/dts/tm1-hb5.dts

Change persistantlogo = <0x0>; to persistantlogo = <0x1>;

To regenerate the dtb files, Issue the commands

cd /embedded/projects/tm1/L4.9.88_2.0.0/

Source ./setenv-hfp.sh

cd linux-tm1

./rebuild.sh

Then copy the dtb files generated

/embedded/projects/tm1/L4.9.88_2.0.0/linux-tm1/arch/arm/boot/dts/*.dtb

To the TMx installer directory

Once boot has completed setting a 0 into /sys/class/graphics/fb0/device/show_fb2 will display the normal screen and setting a 1 will hide it again.

From the debug terminal linux command line

```
sudo chmod 777 /sys/class/graphics/fb0/device/show_fb2
```

`echo 0 > /sys/class/graphics/fb0/device/show_fb2` To display normal framebuffer, or

`echo 1 > /sys/class/graphics/fb0/device/show_fb2` To blank framebuffer

4.0 Building embedded Linux with Buildroot

4.1.1 Buildroot introduction

Buildroot is a build system that aids the process of building the various components of an Embedded Linux system in a single environment. We think Buildroot is easy to get to grips with, and provides a reasonable amount of package support.

Buildroot 2016.02 (for kernel 3.14) and buildroot 2018.02 (for kernel 4.9) are provided in the Linux download for TM1/HB5. Each contain two sample configurations which build the Linux kernel, and a root filesystem.

4.1.2 Buildroot git repository

Buildroot source code for kernel 4.9 can be integrated with Blue Chip's git repository (if not done already).

To check whether the Buildroot source code is backed by a git repository use the following commands:

```
cd /embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8
git remote -v
```

If the response reads:

```
fatal: Not a git repository (or any of the parent directories): .git
```

then the git repository is not set-up. You can set-up the git repository as follows:

```
cd /embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8
git init
git remote add origin http://dl.bluechiptechnology.com/dl/tm1/software
      /linux/L4.9.88_2.0.0/buildroot-2018.02.8.git
```

Note that the above 2 lines are a single command starting with 'git remote ...'.

```
git fetch origin
git reset --hard origin/master
```

If the git repo is already initialised, you can update the contents by issuing the following command:

```
git pull origin master
```

4.2.1 Quickboot demo, with MPlayer support

This configuration is designed to be small and demonstrate a quick ~3 second boot time. MPlayer is included in the configuration, and is configured to automatically play videos found in the root of a USB flash drive during boot. AVI, and MP4 video formats are supported, and video resolutions must match the target LCD screen resolution.

To reduce the final image size this configuration uses the Buildroot uclibc cross compiling toolchain.

To build the quickboot configuration issue the following commands.

Kernel 3.14:

```
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga/buildroot-2016.02
make distclean
make tm1_mplayerquickbootdemo_defconfig
make -j2
```

Kernel 4.9:

```
cd /embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8
make distclean
make tm1_mplayerquickbootdemo_defconfig
make -j2
```

4.2.2 QT5, and BlueZ 5

This configuration will build a root filesystem containing QT5 libraries, QT5 sample applications, and BlueZ libraries. To aid in remote QT5 application deployment, the image is configured with an SSH server, and will print the local IP address to the LCD screen at boot time. The root user is configured with a password of, “password”.

To build this configuration issue the following commands.

Kernel 3.14:

```
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga/buildroot-2016.02
make distclean
make tm1_qt5sample_defconfig
make -j2
```

Kernel 4.9:

```
cd /embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8
Make distclean
make tm1_qt5_defconfig
make -j2
```

4.3.1 Adding WiFi/BT components to the Buildroot staging area

This section is only required for images using kernel 3.14.

The WiFi/BT software components are not integrated into the Buildroot environment, and must be manually added to the Buildroot staging area.

After successful completion of either section 4.2.1 or 4.2.2 issue the following commands:

```
cd /embedded/projects/tm1/L3.14.28_1.0.1_ga/wilink8-build-utilites/
./buildtm1buildroot.sh
```

After completion of the above command the WiFi/BT components will be present in the Buildroot staging area. To repackage the Buildroot root filesystem, and include the WiFi/BT components issue the following commands:

```
cd /embedded/projects/tm1/buildroot-2016.02
make -j2
```

4.3.2 Building Buildroot with modularised WiFi/BT drivers

This option is only available for images using kernel 4.9.

The **Buildroot git repository and Linux kernel 4.9 git repository must be updated** to make the following commands to work (see section 4.1.2 for more information how to update Buildroot git repository and its contents).

To build QT5 Buildroot demo image with modularised WiFi kernel drivers (see section 3.2.1 for more information) issue the following commands:

```
cd /embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8
Make distclean
make tm1_qt5_wl_defconfig
make -j2
```

4.4 Buildroot outputs

After the build completion of either section 4.2.1 or 4.2.2, or 4.3, the built components of the embedded Linux system as follows:

Buildroot 2016

Component	Location
Root filesystem	/embedded/projects/tm1/buildroot-2016.02/output/images/rootfs.tar.bz2
Linux Kernel	/embedded/projects/tm1/buildroot-2016.02/output/images/zImage
Device Tree Configurations	/embedded/projects/tm1/buildroot-2016.02/output/images /*.dtb

Buildroot 2018

Component	Location
Root filesystem	/embedded/projects/tm1/buildroot-2018.02.8/output/images/rootfs.tar.bz2
Linux Kernel	/embedded/projects/tm1/buildroot-2018.02.8/output/images/zImage
Device Tree Configurations	/embedded/projects/tm1/buildroot-2018.02.8/output/images /*.dtb

5. Updating the firmware / software on TM1

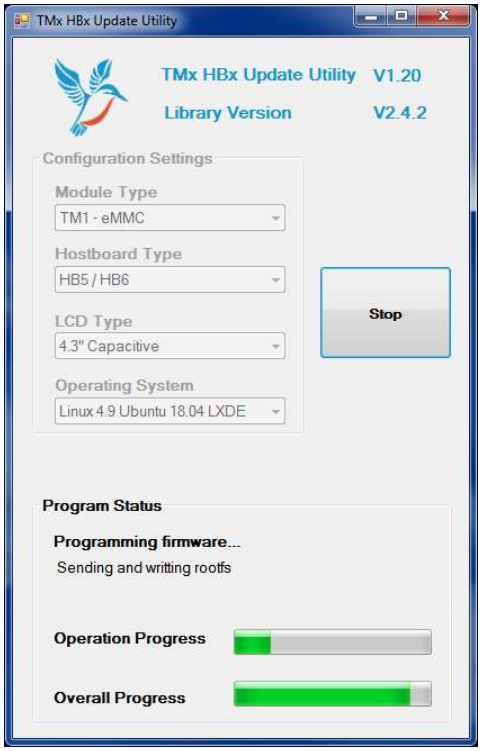
5.1 TMx update utility operation

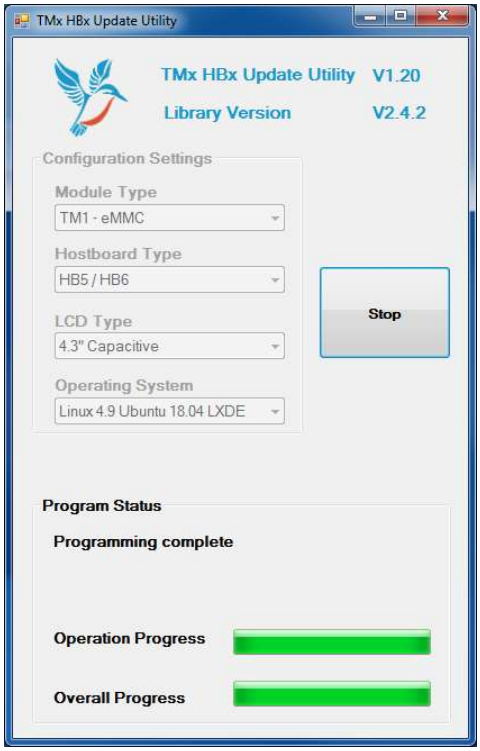
The TMx update utility is a Windows based tool that provisions for programming operating system firmware into the onboard storage of TM1. The utility can be downloaded and installed from the Blue Chip Technology website. See the following link for the latest version of the utility. At time of writing V1.27 is the latest version.

https://www.bluechiptechnology.com/product/tm1/

To update a TM1 module firmware using this utility, follow the below steps:

1. Launch the utility using either the desktop or start menu shortcut
2. Select module type. Note: TM1 emmc / uSD options refer solely to the storage media populated on the TM1 module.
3. Select host board type
4. Select LCD and touch screen type
5. Select desired operating system.
6. Press the start button
7. Attach a USB A -> mini B cable between the PC and TM1 / HB5.
8. Power on the hardware with the BOOT_MODE# pin shorted to ground.
9. Follow the onscreen messages and wait for completion
10. Reboot the unit to try out the new operating system.





5.2 TMx update utility firmware locations

Depending on the selected options in the update utility, specific firmware will be written to the TM1. For Linux operating systems the associated firmware files are located in the following location

<Install Path>\firmware\tm1\linuxfiles

If the default installation path was chosen for the install the firmware files will be located in the following locations:

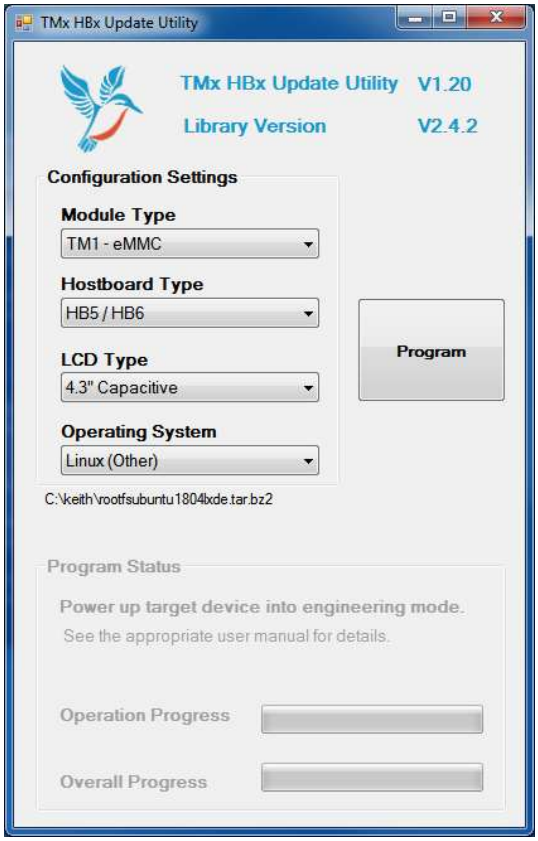
C:\Program Files (x86)\Blue Chip Technology\TM1 Update Utility Version 1.27\firmware\tm1\linuxfiles\4.9

The files in this directory are for kernel 4.9 and have the following purpose

Filename	Description
tm1-hb5-43-c.dtb	HB5 with 4.3 Inch LCD and capacitive touch
tm1-hb5-43-r.dtb	HB5 with 4.3 Inch LCD and resistive touch
tm1-hb5-7-c.dtb	HB5 with 7 Inch LCD and capacitive touch
tm1-hb5-7-r.dtb	HB5 with 7 Inch LCD and resistive touch
tm1-hb5-9-c.dtb	HB5 with 9 Inch LCD and capacitive touch
tm1-hb5-cb3-43-c.dtb	HB5 + CB3 with 4.3 Inch LCD and capacitive touch
tm1-hb5-cb3-43-r.dtb	HB5 + CB3 with 4.3 Inch LCD and resistive touch
tm1-hb5-cb3-7-c.dtb	HB5 + CB3 with 7 Inch LCD and capacitive touch
tm1-hb5-cb3-7-r.dtb	HB5 + CB3 with 7 Inch LCD and resistive touch
tm1-hb5-cb3-9-c.dtb	HB5 + CB3 with 9 Inch LCD and capacitive touch
zImageLinuxTM1	Linux kernel 4.9 built with tm1wl_defconfig used for Ubuntu 22.04 rootfs
zImageLinuxTM1brmp	Linux kernel 4.9 built with tm1wl_defconfig used for buildroot mplayer demo rootfs

zImageLinuxTM1brqt5	Linux kernel 4.9 built with tm1wl_defconfig used for buildroot QT5 demo rootfs
rootfsubuntu2204lxde.tar.bz2	Ubuntu 22.04 root filesystem. See section 3
qt5rootfs.tar.bz2	QT5.9 root filesystem. See section 4.2.2
mplayerquickbootrootfs.tar.bz2	MPlayer quick boot root filesystem. See section 4.2.1

By over-writing these files with the files generated in sections 3 and 4 it is possible to deploy compiled firmware to the TM1 module. Alternatively, if the only required update is to the root filesystem, the TMx update utility has a, “Linux (Other)” option for manually selecting a custom root filesystem from the local machine. The root filesystem must be in tar.bz2 format.



Note: the TMx Update Utility version 1.27 no longer offers installation options based on Linux kernel 3.14. If you need to install Operating systems based on Linux kernel 3.14 please use previous version of TMx Update Utility 1.26 available upon request. In such case, make sure the Linux kernel 3.9 was rebuilt with the latest fixes available from Git repository to avoid potential compatibility issues.

6. BCT TM1/HB5 Hardware Setup in Linux

6.1 Debug Serial Console

Linux and U-boot for BCT TM1/HB5 heavily relies on access to a serial console. By default U-boot and Linux are configured to use the RS232 port available on P2 of the HB5. By default the board is set to communicate at 115200, 8, n, 1. Before turning on the TM1/HB5 for the first time it is recommended that this port be connected to a PC with terminal emulator software running. E.g. HyperTerminal.

6.2 BCT TM1/HB5 Serial Ports

The UARTs on the HB5 are mapped as follows:

HB5 Header	Linux Device Name
P4(RS232 RX + RS232 TX)	/dev/ttymxc1
P4(CRX1N CRX1P CTX1N CTX1P)	/dev/ttymxc2
P2	/dev/ttymxc0 (Linux console port)

6.2.1 RS-485 Manual Transmit Control

/dev/ttymxc2 is an RS485 / RS422 compatible port which has a transmit enable signal. This signal can be controlled using GPIO 67. From the Linux console this signal can be manipulated using the commands:

```
echo 67 >> /sys/class/gpio/export
echo out >> /sys/class/gpio/gpio67/direction
echo 1 >> /sys/class/gpio/gpio67/value
echo 0 >> /sys/class/gpio/gpio67/value
```

6.2.2 RS-485 Automatic Transmit Control

To improve software efficiency when communicating over an RS485 interface it is possible to configure the TM1 Linux kernel to automatically control the transmit enable. The linux API for configuring the UART in RS-485 mode can be viewed using the following link.

https://www.kernel.org/doc/Documentation/serial/serial-rs485.txt

The BCT application note RS485_BETA_APP_NOTE also provides useful information on implementing RS-485 with the TM1 platform.

6.2.3 UART DMA and FIFO Threshold

The TM1 UART driver in the Linux kernel is designed to be efficient at high throughputs and baud rates. One technology that the driver uses is DMA (direct memory access) to provide efficient transfer of data. A second technique that the driver uses is setting a high FIFO threshold to limit the amount of interrupts requiring software servicing. While these driver optimisations give good performance and efficiency at high throughputs, this is not always the case for low baud rates and small amounts of data which tends to be the case with protocols using RS-485.

To allow the DMA function to be disabled a file called dmaenabled has been added to the sysfs for UARTS.

To disable UART DMA for the RS485 UART on TM1 the following command can be issued at a console or through application software.

```
echo 0 > /sys/class/tty/ttymxc2/device/dmaenabled
```

Note: DMA must be disabled before application software opens the UART.

To allow the UART FIFO threshold to be configured a file called rxfifothreshold has been added to the sysfs for UARTS.

To modify the UART FIFO threshold to 1 for the RS485 UART on TM1 the following command can be issued at a console or through application software.

```
echo 1 > /sys/class/tty/ttymxc2/device/rxfifothreshold
```

The rxfifothreshold can be set to any value between 1 and 32.

Note: The FIFO threshold must be set before application software opens the UART.

6.3 BCT HB5 GPIO

The recommended way to access the GPIO is using the SYSFS interface. This can be done using the command line (or scripts), or can be done from inside an application.

The Linux GPIO documentation can be found here:
https://www.kernel.org/doc/Documentation/gpio/sysfs.txt

This following page also has some useful examples:
http://falsinsoft.blogspot.co.uk/2012/11/access-gpio-from-linux-user-space.html

By default, the GPIOs on TM1/HB5 are setup with pull-ups enabled. They are defined in the pinctrl_hog_hostboard structure of the hb5.dts file (/embedded/projects/tm1/linux-tm1/arch/arm/boot/dts/hb5.dts)

The logical GPIOs on the P15 header of HB5 map to the physical GPIO pins on the SOC as follows:

GPIO1 - 148
GPIO2 - 147
GPIO3 - 149
GPIO4 - 146
GPIO5 - 128
GPIO6 - 125
GPIO7 - 127
GPIO8 - 13
GPIO9 - 31
GPIO10 - 52
GPIO11 - 108
GPIO12 - 144

To setup and control GPIO 0 the following commands would be used:

```
echo 148 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio148/direction
echo 1 > /sys/class/gpio/gpio148/value
```

..

6.4 TM1 Wi-Fi Operation

Presuming that the Wi-Fi kernel modules have been compiled into the root filesystem, the Wi-Fi kernel modules can be loaded with the following commands.

```
modprobe wl18xx
modprobe wlcore_sdio
ls
```

If the modules were successfully loaded the wlan0 network device should be present. This can be checked by issuing the following command.

```
ifconfig –a
```

The following commands can be used to enable the wlan0 interface, and scan for networks.

```
ifconfig wlan0 up
iw wlan0 scan | grep SSID
```

6.5 TM1 BT 4.0 Operation

Presuming that the BT kernel modules have been compiled into the root filesystem, the BT kernel modules can be loaded with the following commands.

```
modprobe btwilink
```

If the module was successfully loaded the hci0 BT device should be present. This can be checked by issuing the following command.

```
hciconfig
```

The following commands can be used to enable the BT interface, and scan for devices.

```
hciconfig hci0 up
hcitool scan
```

6.6 TM1 Audio

The audio CODEC featured on TM1 implements the standard Linux ALSA API framework. Standard commands like alsamixer, aplay, arecord, speaker-test will work.

6.7 HB5 uSD Card

The uSD card connector featured on HB5 is mapped to mmc3 in the Linux kernel

6.8 TM1 Watchdog

The i.MX6 implemented on the TM1 module includes a watchdog that can reset the system. It is implemented using the standard Linux Watchdog API.

http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/watchdog/watchdog-api.txt

6.9 TM1 Power management

TM1 implements power and thermal management under software control. This can be configured using the DVFS framework in the Linux kernel.

https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

TM1 supports suspend to RAM, which allows the system to enter a low power mode (~<70mw) while retaining the contents of RAM. This allows the system to resume to an operational state in a very short period of time. To enter suspend to RAM mode the following command can be issued.

```
echo mem > /sys/power/state
```

The SLEEP_RQ# signal on HB5 is configured to wake the system up when in suspend to RAM mode.

6.10 HB5 Class-D amplifier

The class D amplifier implemented on HB5 can be controlled using GPIO 66.

6.11 CB3 CAN Bus

The CB3 dual CAN bus are exposed by network devices can0 and can1.

Can-Utils can be used to test the CAN interfaces, which is preinstalled in the QT5 demo image.

https://github.com/linux-can/can-utils

The following commands can be issued in Linux to configure can0 to display all messages received on the bus.

ip link set can0 type can bitrate 1000000 triple-sampling on

ifconfig can0 up

candump can0

6.12 LCD Backlight

The LCD backlight can be controlled using the standard Linux sysfs backlight class.

https://www.kernel.org/doc/Documentation/ABI/stable/sysfs-class-backlight

Kernel 3.14:

The following commands would set the backlight to 0%:

```
echo 0 > /sys/class/backlight/pwm-backlight.23/brightness
```

The following commands would set the backlight to 50%:

```
echo 50 > /sys/class/backlight/pwm-backlight.23/brightness
```

The following commands would set the backlight to 100%:

```
echo 100 > /sys/class/backlight/pwm-backlight.23/brightness
```

Kernel 4.9:

The following commands would set the backlight to 0%:

```
echo 0 > /sys/class/backlight/pwm-backlight/brightness
```

The following commands would set the backlight to 50%:

```
echo 50 > /sys/class/backlight/pwm-backlight/brightness
```

The following commands would set the backlight to 100%:

```
echo 100 > /sys/class/backlight/pwm-backlight/brightness
```

7. UBOOT operation

The u-boot version ported to the TM1 platform is V2014.04. At a high level its primary purpose is to copy the Linux kernel, device tree configuration, and bootargs into memory before passing execution over to the Linux kernel.

7.1 Configuring uboot

Configuration of uboot is performed by issuing commands over the debug serial console available on P2 of HB5. The UART is configured to communicate with 115200,8,n,1 parameters. Connecting a null modem cable between the HB5 and a development PC makes it possible to configure uboot using a terminal emulator. E.g. Putty or HyperTerminal.

To enter configuration mode, uboot must receive a character over the serial port during power on. The bootdelay parameter is set to 0 by default to give a fast boot time, which means that the time window pressing the key is short.

The four most common commands used in uboot for TM1 are.

1. setenv – used to set an environment variable to a value.
2. printenv – used to display the current value of an environment variable.
3. editenv – used to edit an environment variable.
4. saveenv – save the environment

The remainder of this section will focus on the TM1 specific environment variables and how they should be edited. Other commands are available, which can be viewed by issuing the command "`help`". The official uboot website is also a good source of information on uboot.

http://www.denx.de/wiki/U-Boot/

7.2 Uboot environment variables

The following table defines the Uboot variable related to the TM1 platform.

Variable	Description
bootdelay	The time window in seconds that Uboot will wait for a key press to enter configuration mode. Default value is 0
mmcargs	The bootargs passed to the Linux kernel when booting from uSD or eMMC storage.
netargs	The bootargs passed to the Linux kernel when booting from NFS storage.
mmcroot	The uSD/emmc partition to mount as the root filesystem.
fdt_file	The device tree blob file to load
serverip	The IP address of tftpserver, and NFS server. Used when booting over a network.
nfsroot	The nfs root directory to mount on the host PC when booting over NFS.

7.3 Uboot configuration examples

7.3.1 Changing the Uboot boot delay

```
setenv bootdelay 3
saveenv
```

7.3.2 Booting the Linux kernel over tftp and mounting a rootfs over NFS

For 3.14.28 kernel

```
setenv serverip <IP address of development machine>
setenv nfsroot /nfs/3.14.rootfs
setenv bootcmd run netboot
saveenv
```

For 4.9.88 kernel

```
setenv serverip <IP address of development machine>
setenv nfsroot /nfs/4.9.rootfs
setenv bootcmd run netboot
saveenv
```

7.3.3 Enable capacitive multitouch in the Linux kernel

```
editenv mmcargs
```

append, "`multitouch`" and press return.

```
saveenv
```

7.3.4 Boot a root filesystem from the HB5 uSD

```
set mmcroot /dev/mmcblk3p1 rootwait rw
saveenv
```

8. QT5 Application development introduction

The following section will detail how QT creator can be installed and configured to deploy a simple "Hello World" app to the TM1 / HB5 platform over Ethernet. QT 5.5 and QT5.9 will be covered.

If you are using the BSP VM2.03 or later it is recommended that you increase the available screen area displayable

Click

Program start tab, settings, display, resolution, keep this Configuration

8.1 QT5.5

QT5.5 has been tested with linux kernel L3.14 for the Linux 4.9 kernel use QT5.9 (see section 8.2)

The L3.14 kernel is not prebuilt in the board support package or on the BSP VM and this must be built prior to configuring QT5,5 See section 4.2.2

8.1.1 Download and install QT Creator to the development machine

On the same development machine that was used build the QT5 Buildroot root filesystem issue the following commands to download QT creator, mark the download as executable, and run the installer.

```
wget https://download.qt.io/new_archive/qt/5.5/5.5.0/qt-opensource-linux-x64-5.5.0-2.run

chmod a+x ./qt-opensource-linux-x64-5.5.1.run
./qt-opensource-linux-x64-5.5.1.run
```

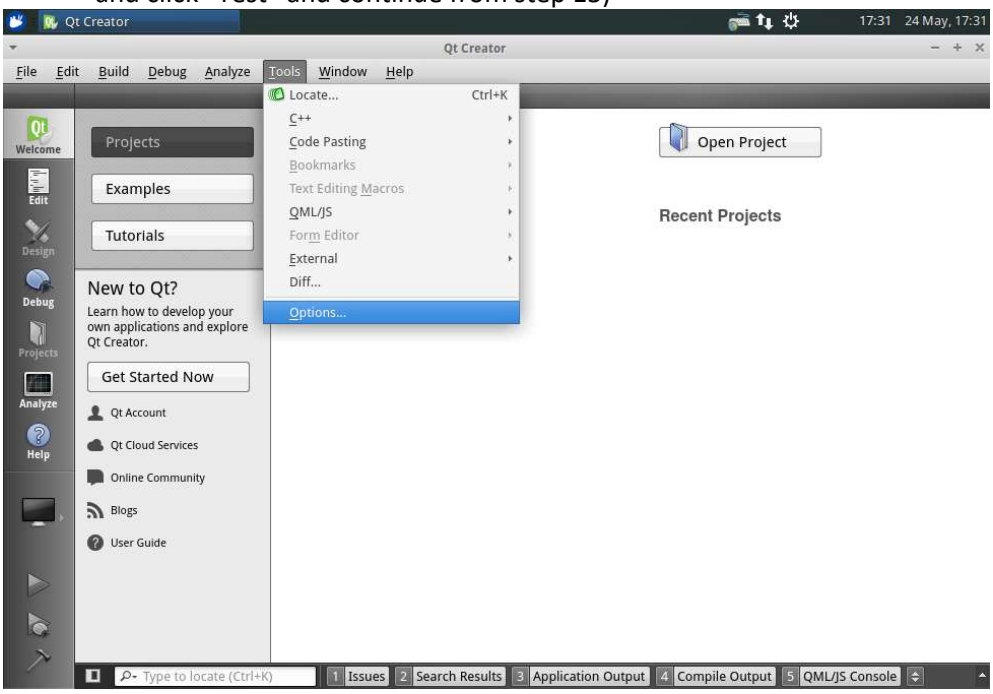
Follow the installer prompts, and use the default configuration suggestions.

8.1.2 Setup the TM1 / HB5 environment in QT Creator

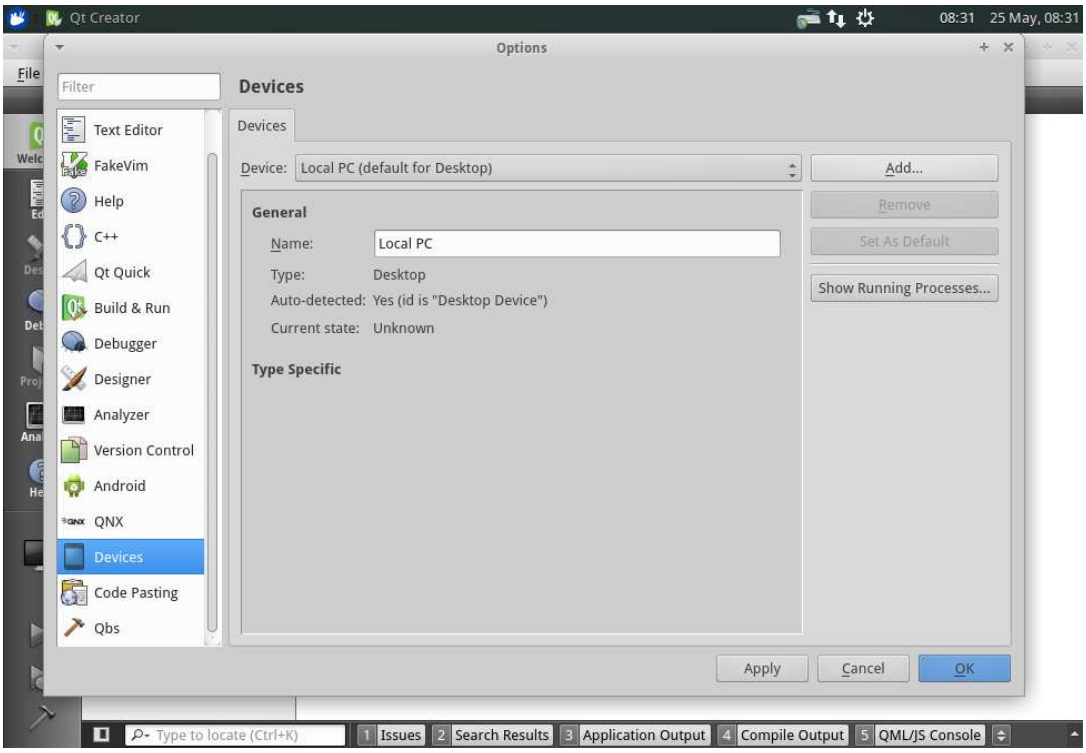
QT Creator uses the notion of "Kits" which refer to development environment configurations, targeting specific architectures, and devices. By default only a single kit is installed in QT creator that targets applications running in the host environment. This section will focus on the setup of a kit targeting TM1 running the Buildroot generated QT5 root filesystem created in section 4.2.2.

1. Ensure section 4.2.2 has been followed to create a QT5 based root filesystem for TM1/HB5
2. Use the TMx update utility to apply the QT5 image to TM1.
3. Boot the unit with an Ethernet cable attached. The LCD will display the IP address obtained via DHCP. Make a note of this IP Address.
4. Launch QT creator

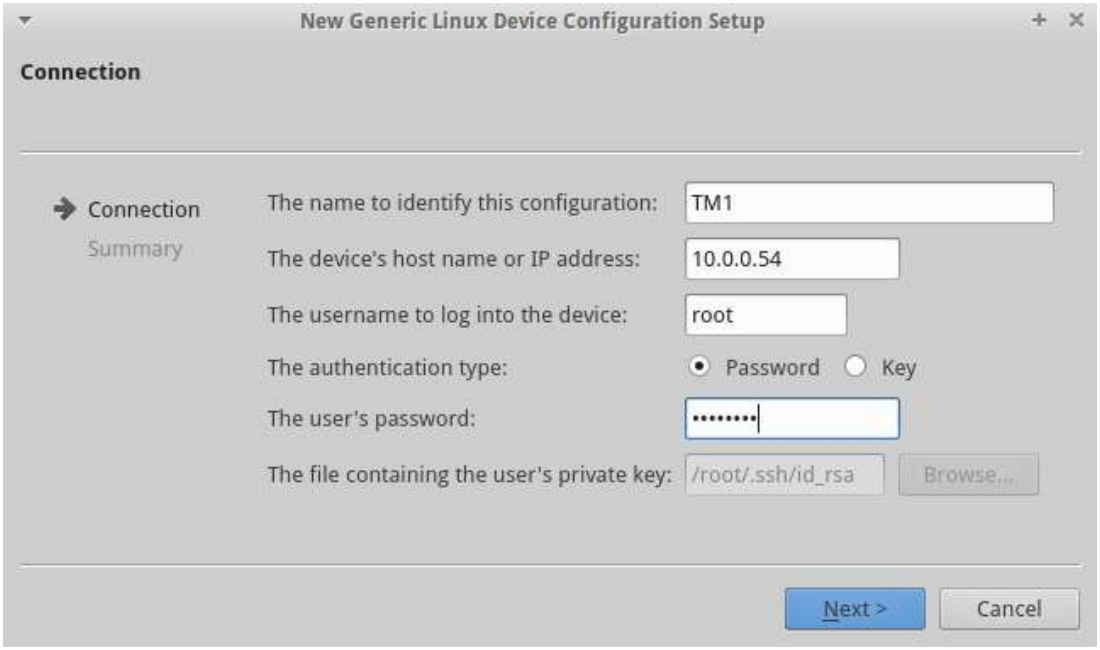
5. Navigate to Tools -> Options (if you are using BSP VM 2.03 enter the IP address from TM1 and click “Test” and continue from step 15)



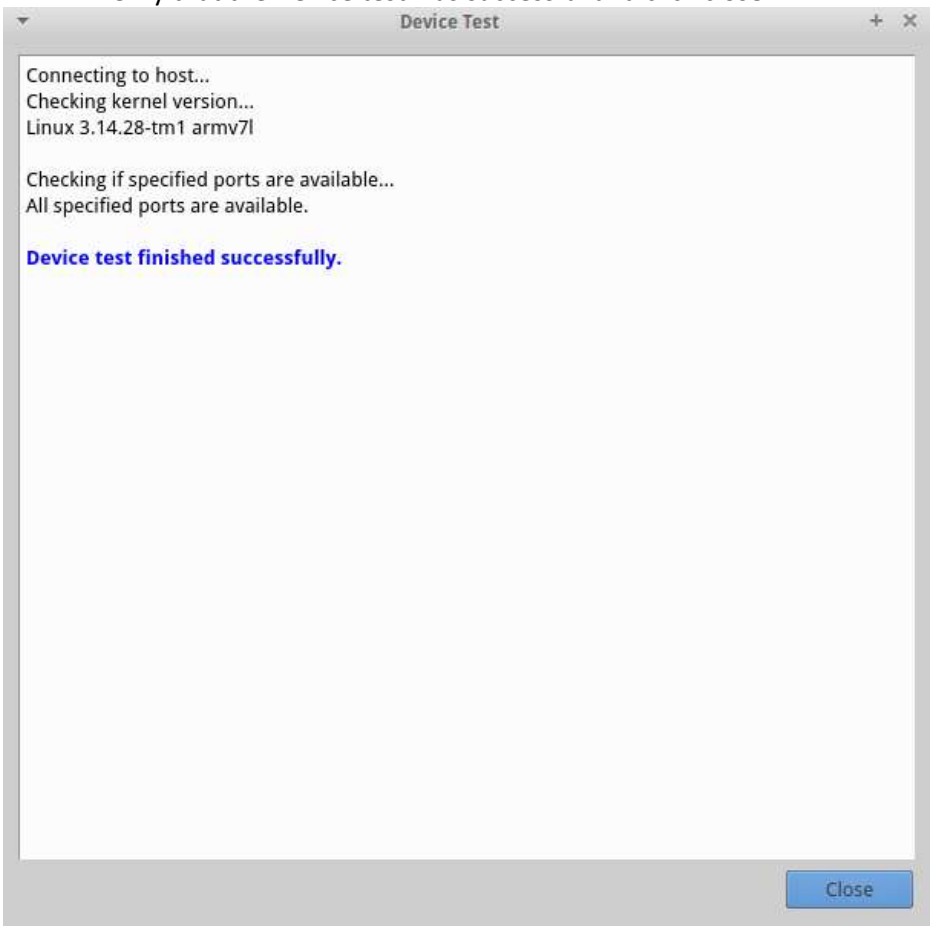
6. In the left hand pane select “Devices”, and then lick “add”.



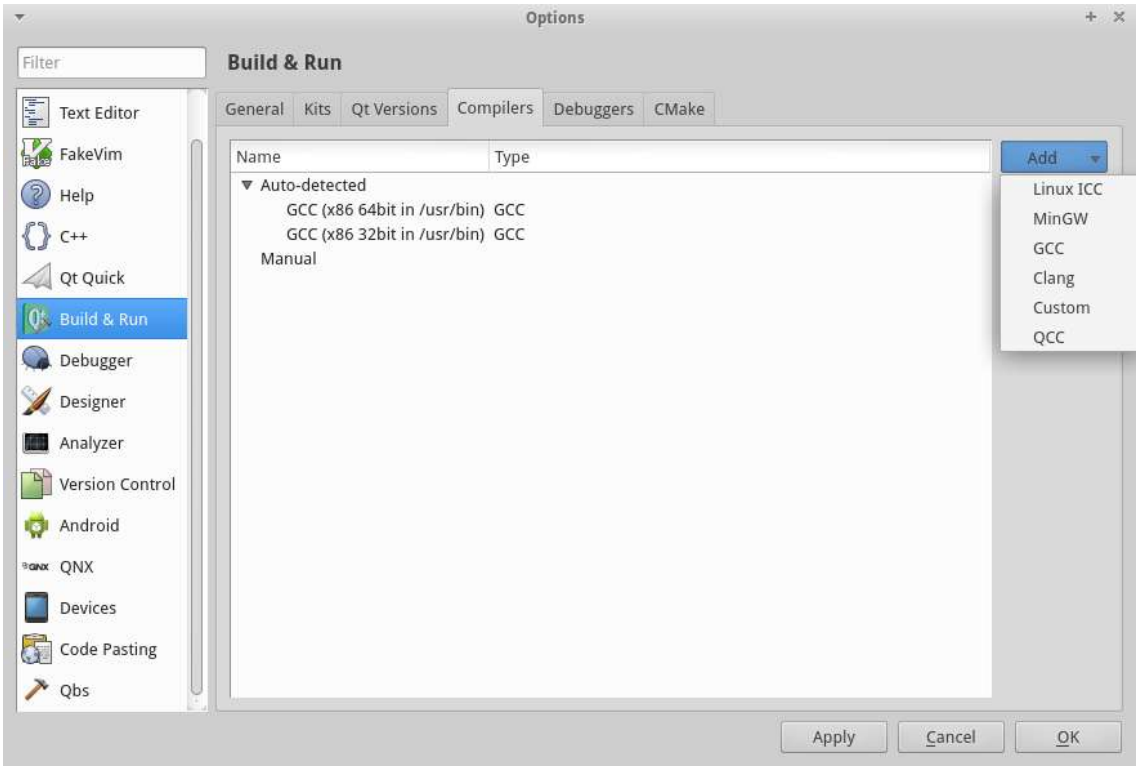
7. Select, “Generic Linux Device” and click, “Start Wizard”.
8. Set the device name to, “TM1”
9. Set the host name or IP address to the IP address noted down in step 3.
10. Set the username to, “root”
11. Set the authentication type to password.
12. Set the users password to, “password”
13. Click next, and then click finish.



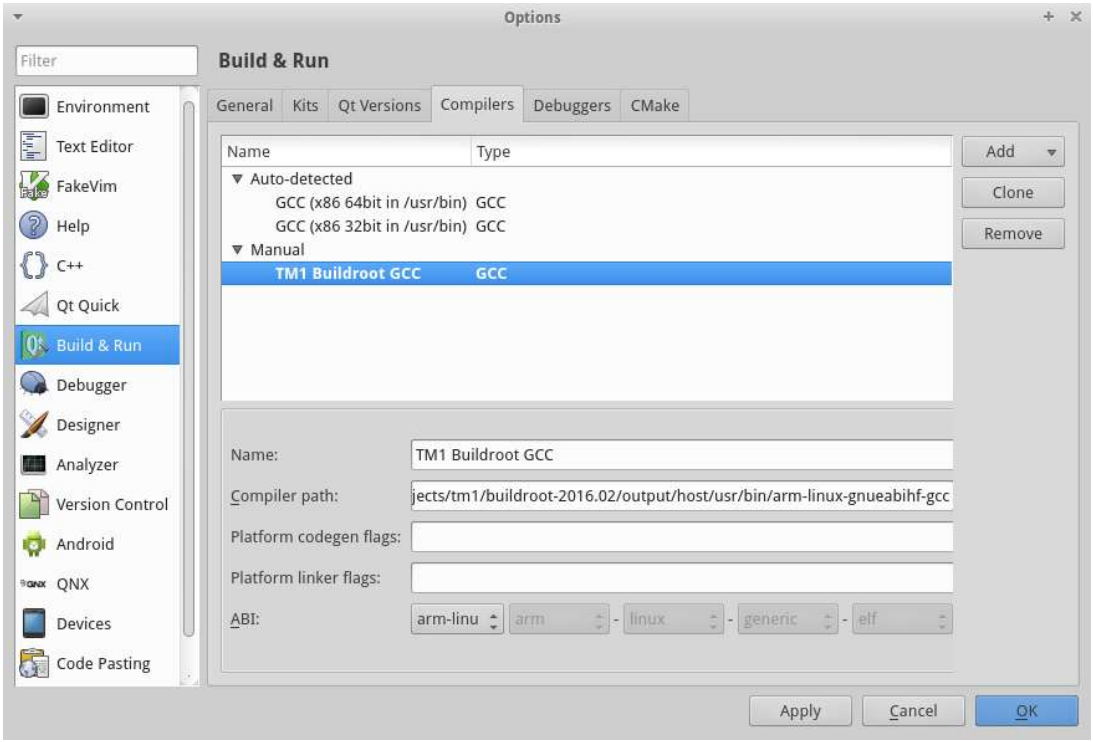
14. Verify that the Device test was successful and click close.



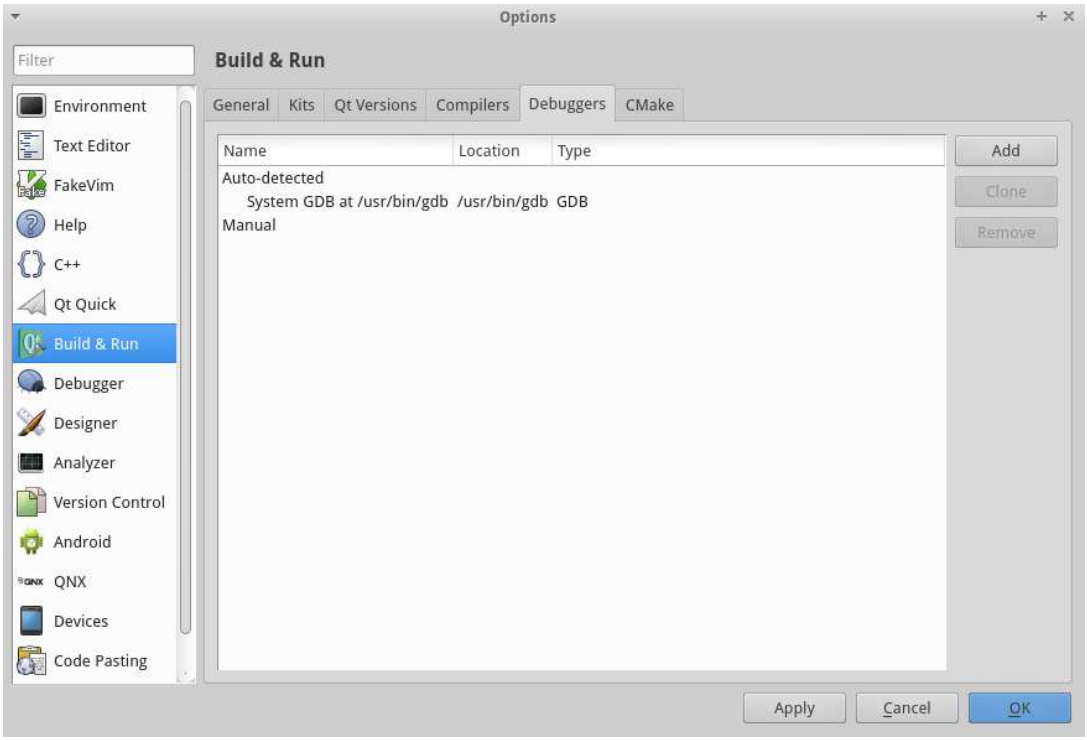
15. Press Apply in the Options Window.
16. In the left hand pane select “Build and Run”, and then select the “Compilers” tab. Click “add” -> “GCC”.



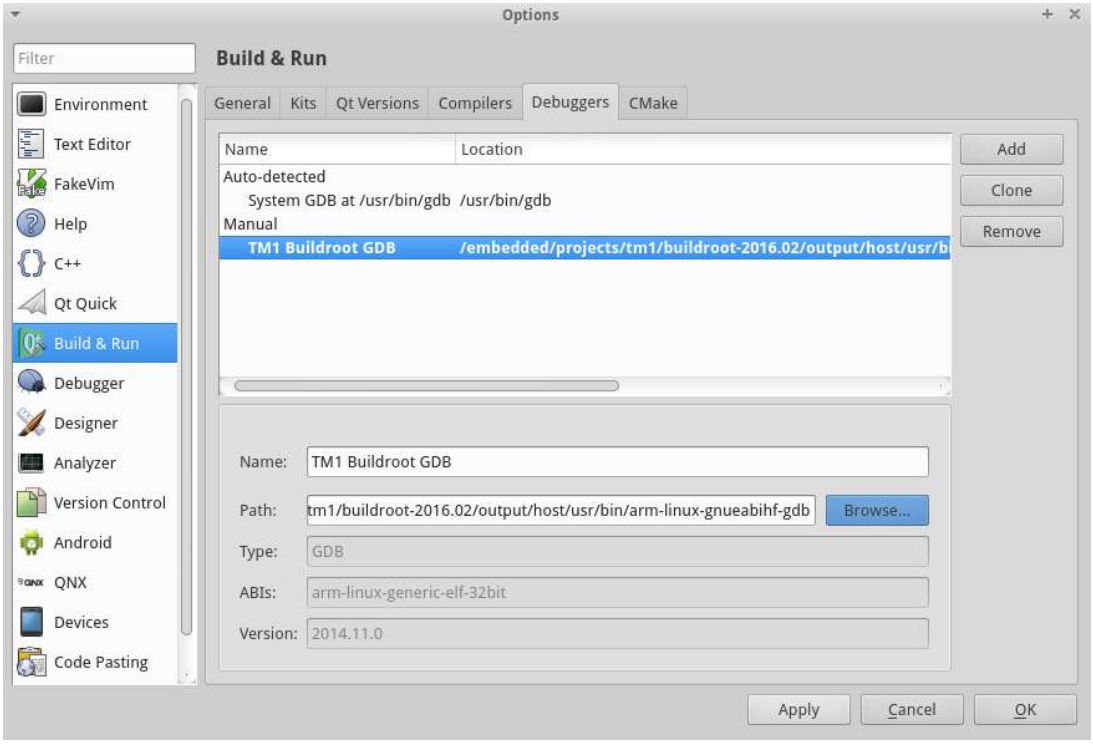
17. Set Name to, “GCC QT5.5”.
18. Set Compiler path to, “/embedded/projects/tm1/L3.14.28_1.0.1_ga/buildroot-2016.02/output/host/usr/bin/arm-linux-gnueabihf-gcc”
19. Click Apply



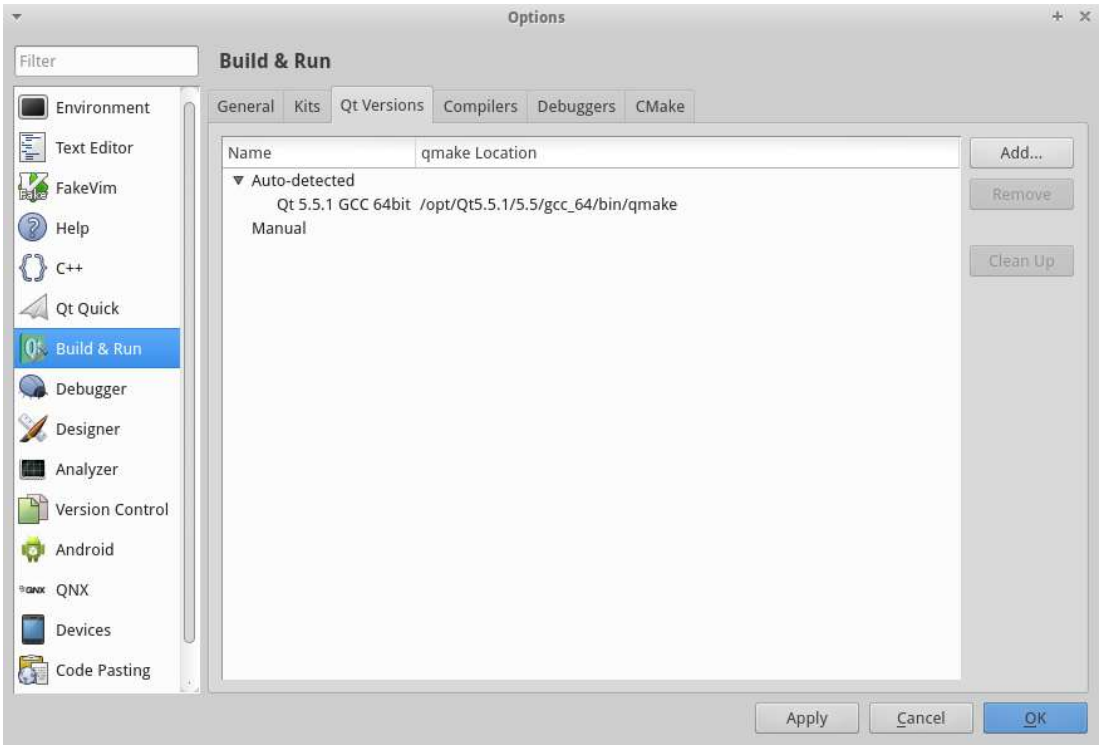
20. In the left hand pane select, “Build and Run” and then select the “Debuggers” tab. Click “add”.



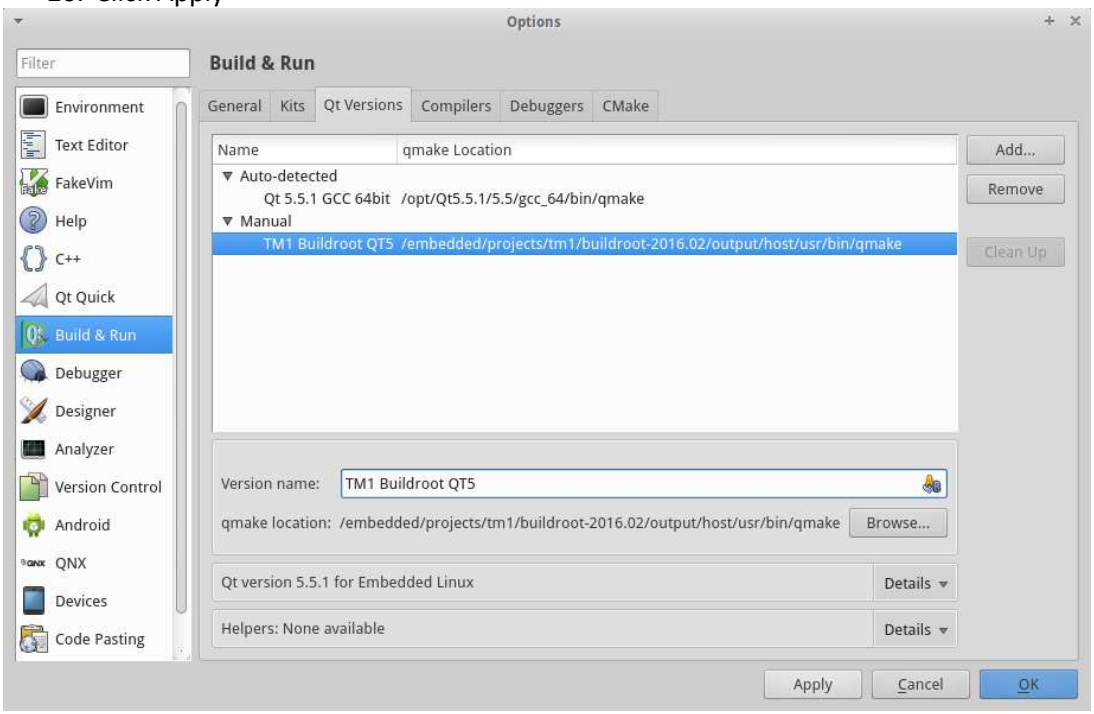
21. Set the name to, “TM1 QT 5.5 Debugger”
22. Set the path to, “/embedded/projects/tm1/L3.14.28_1.0.1_ga/buildroot-2016.02/output/host/usr/bin/arm-linux-gnueabihf-gdb”
23. Click Apply



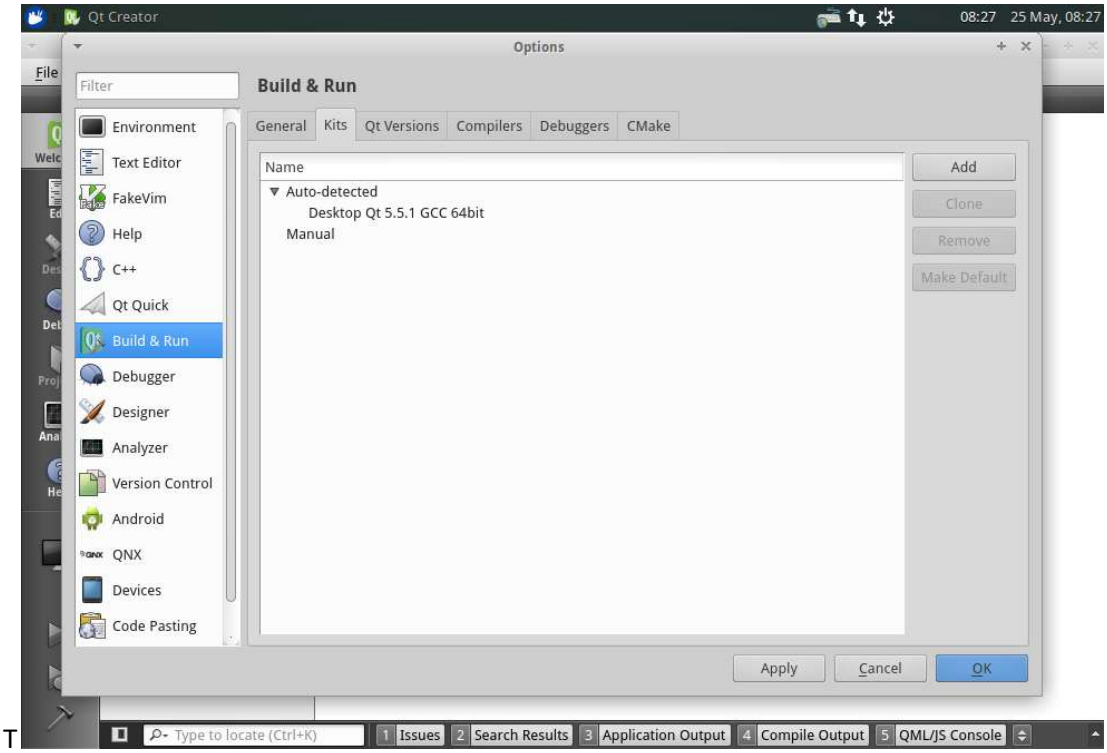
24. In the left hand pane select, “Build and Run” and then select the “Qt Versions” tab. Click “Add”.



25. Click add and select the qmake executable, “/embedded/projects/tm1/L3.14.28_1.0.1_ga/buildroot-2016.02/output/host/usr/bin/qmake”
26. Click Apply



27. In the left hand pane select, “Build and Run” and then select the “Kits” tab. Click “Add”. If Using BSP VM v2.03 or later select TM1 QT5.5 and continue from step 33



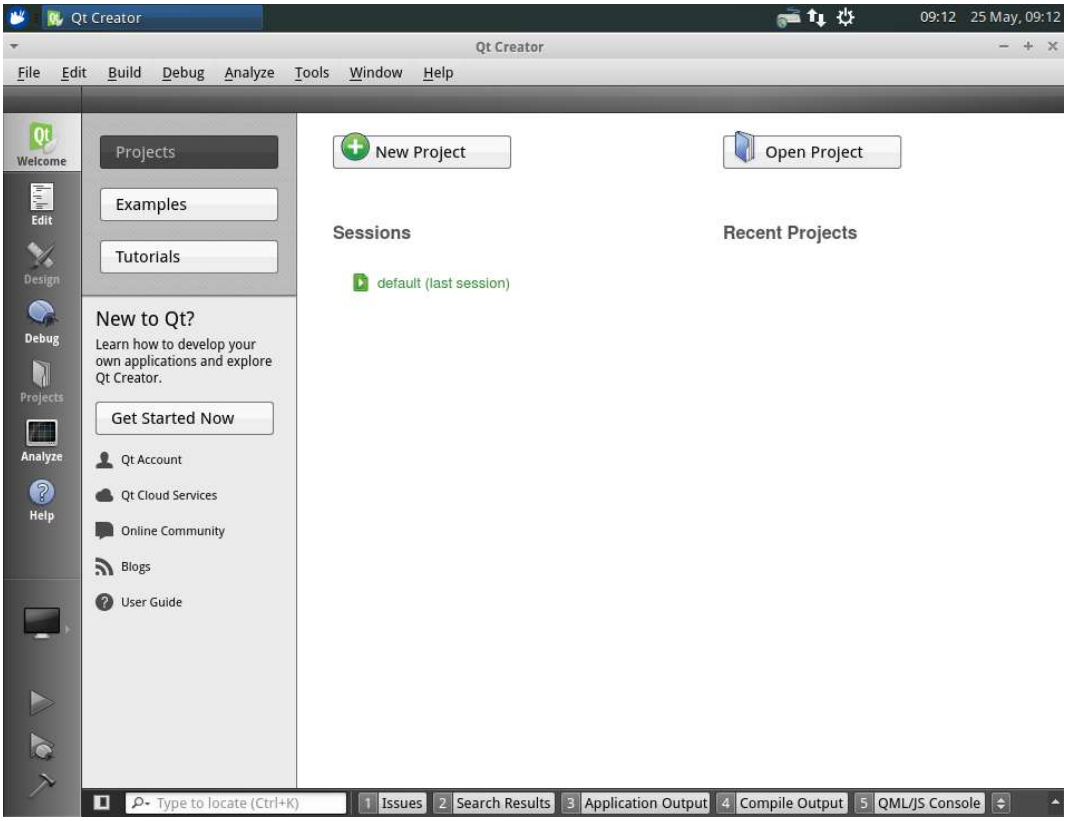
T

28. Set Name to, “TM1 QT5.5”
29. Set File system name to, “TM1”
30. Select Device Type to, “Generic Linux Device”
31. Select Device to, “TM1 (default for Generic Linux)”
32. Set Sysroot to, “/embedded/projects/tm1/L3.14.28_1.0.1_ga/buildroot-2016.02/output/target”
33. Set Compiler to, “GCC QT5.5”
34. Set debugger to, “TM1QT5.5 Debugger”
35. Set Qt Version to, “Qt 5.5.1 (System)”
36. Click “manage” for Cmake tool
37. Click “Add”
38. Rename name to “CMake QT5.5”
39. Set path to “/embedded/projects/tm1/L3.14.28_1.0.1_ga/buildroot-2016.02/output/host/usr/bin/cmake”
40. Press “Make Default”
41. Press OK.
42. In the left hand pane select, “Build and Run” and then select the “Kits” tab.
43. Select “TM1 QT5.5”
44. Set CMake Tool to “Cmake QT5.5”
45. Press “Make Default”
46. Press OK.

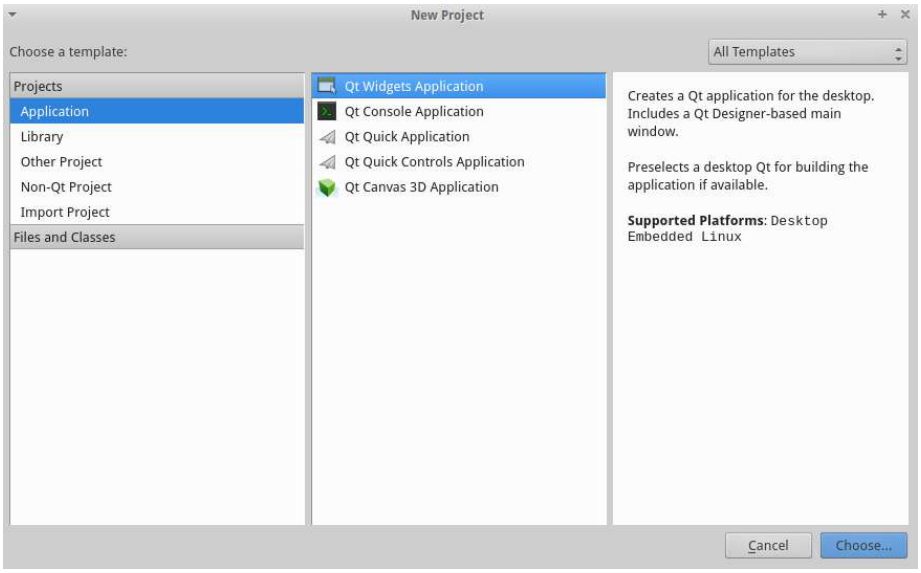
8.1.3 Setup a simple QT5 “Hello World” application

The following section will describe how to setup and deploy a simple “Hello world” application to TM1.

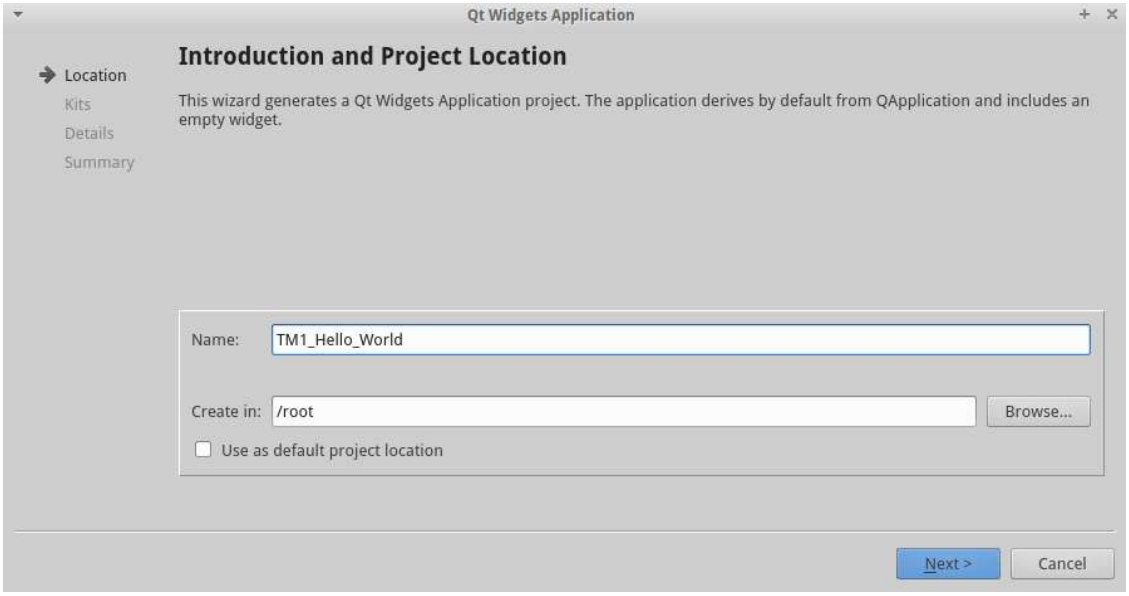
1. Launch QT Creator
2. Select, “New Project”



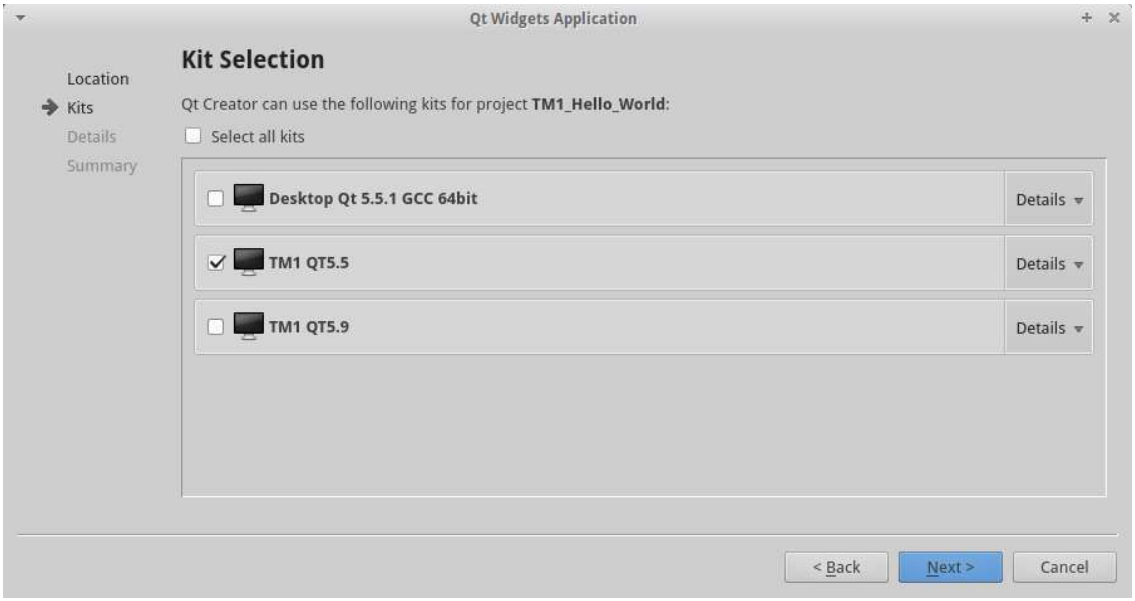
3. Select, “Qt Widgets Application” and click, “Choose”.



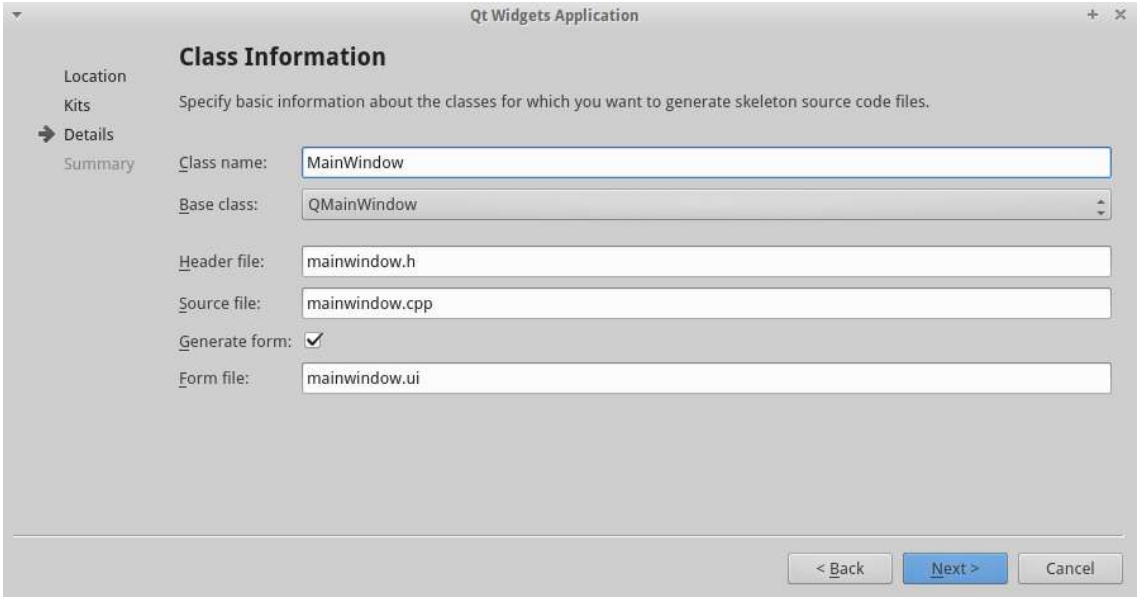
4. Set the name to, “TM1_Hello_World” and click next



5. Select just the, “TM1 QT5.5” kit, and click next.

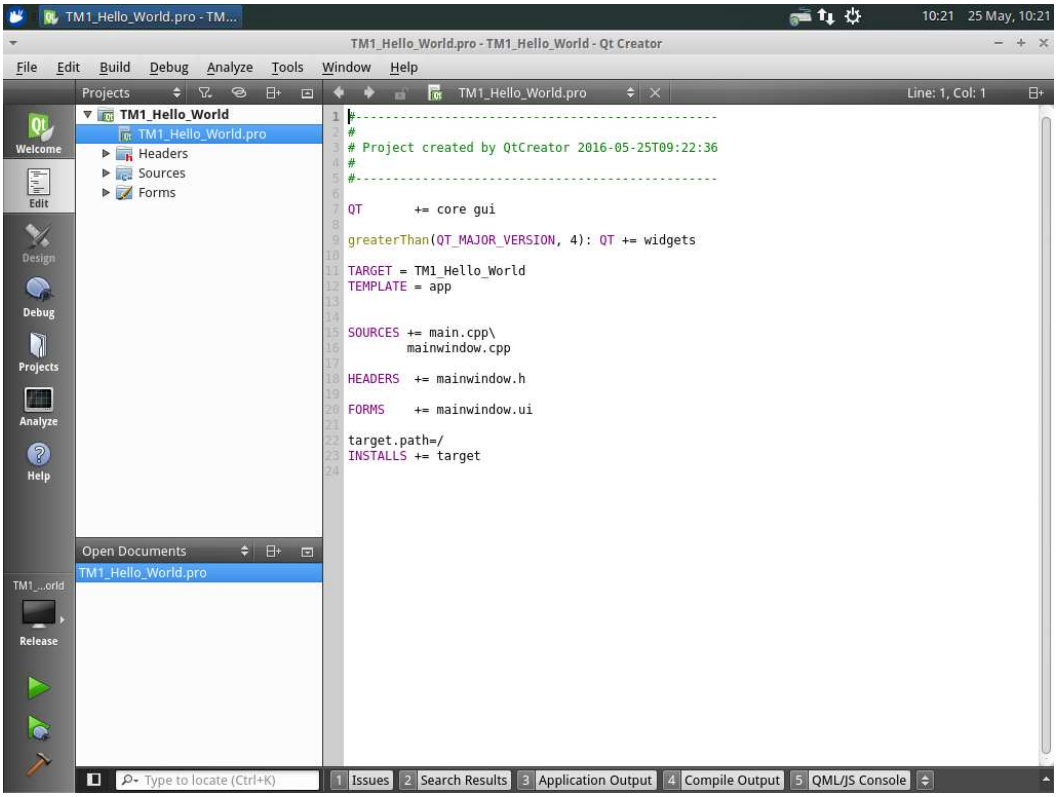


6. Use the default Class Information and click Next

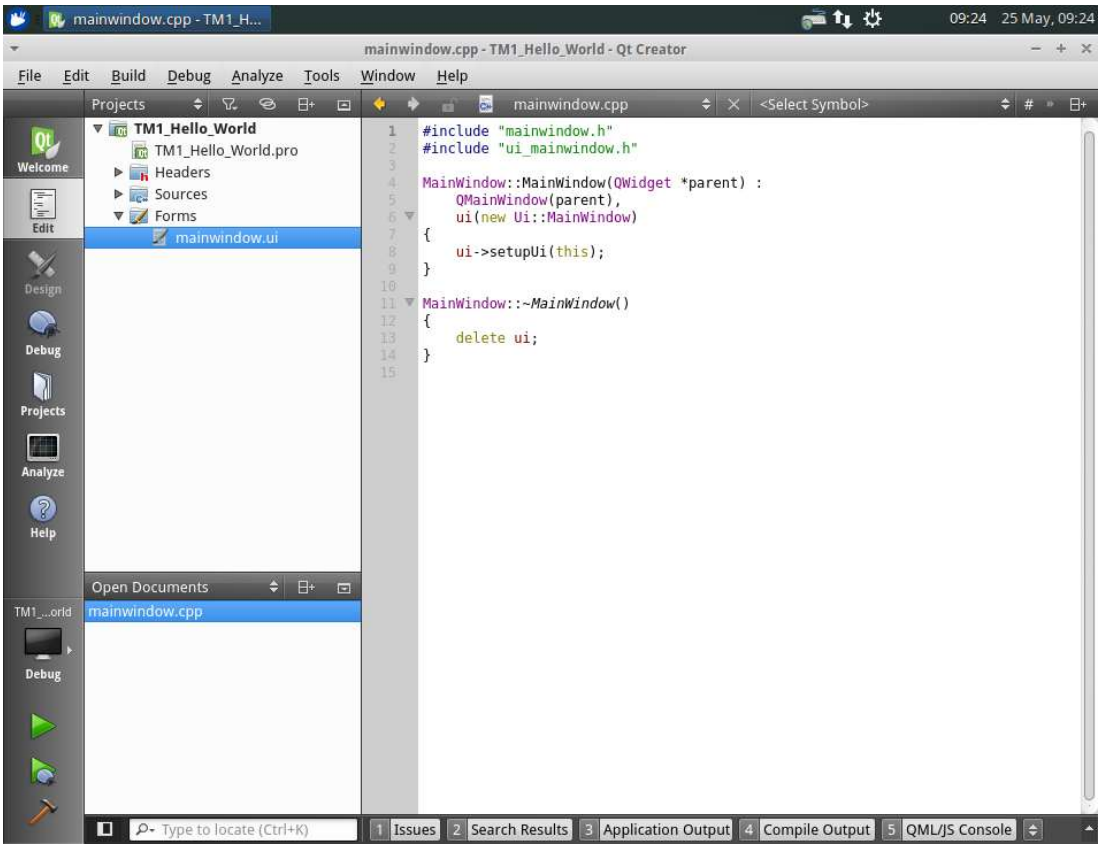


7. Click Finish
8. In the Projects view, double click, “TM1_Hello_World.pro” file, to open the project editor.
9. Append the following to the bottom of the project configuration. This will tell the deployment tool where on the target root filesystem to put the executable.

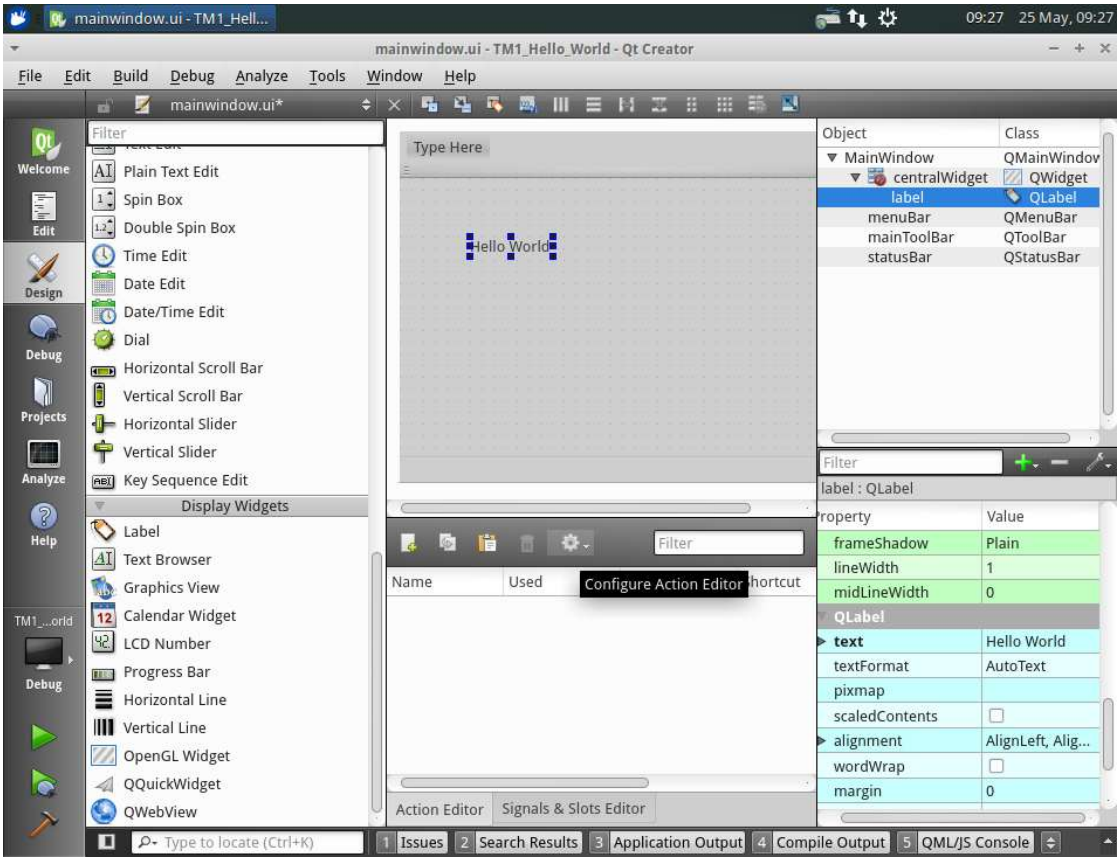
```
target.path=/
INSTALLS += target
```



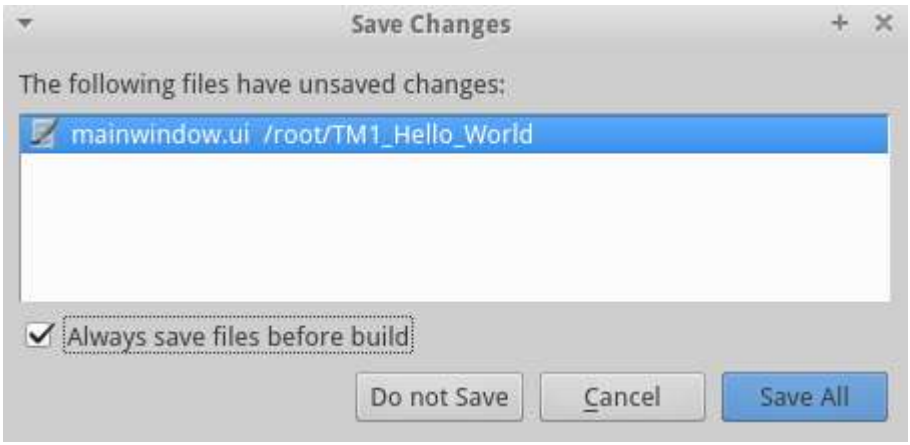
10. In the Projects view, double click, “mainwindow.ui”, to open the forms designer.



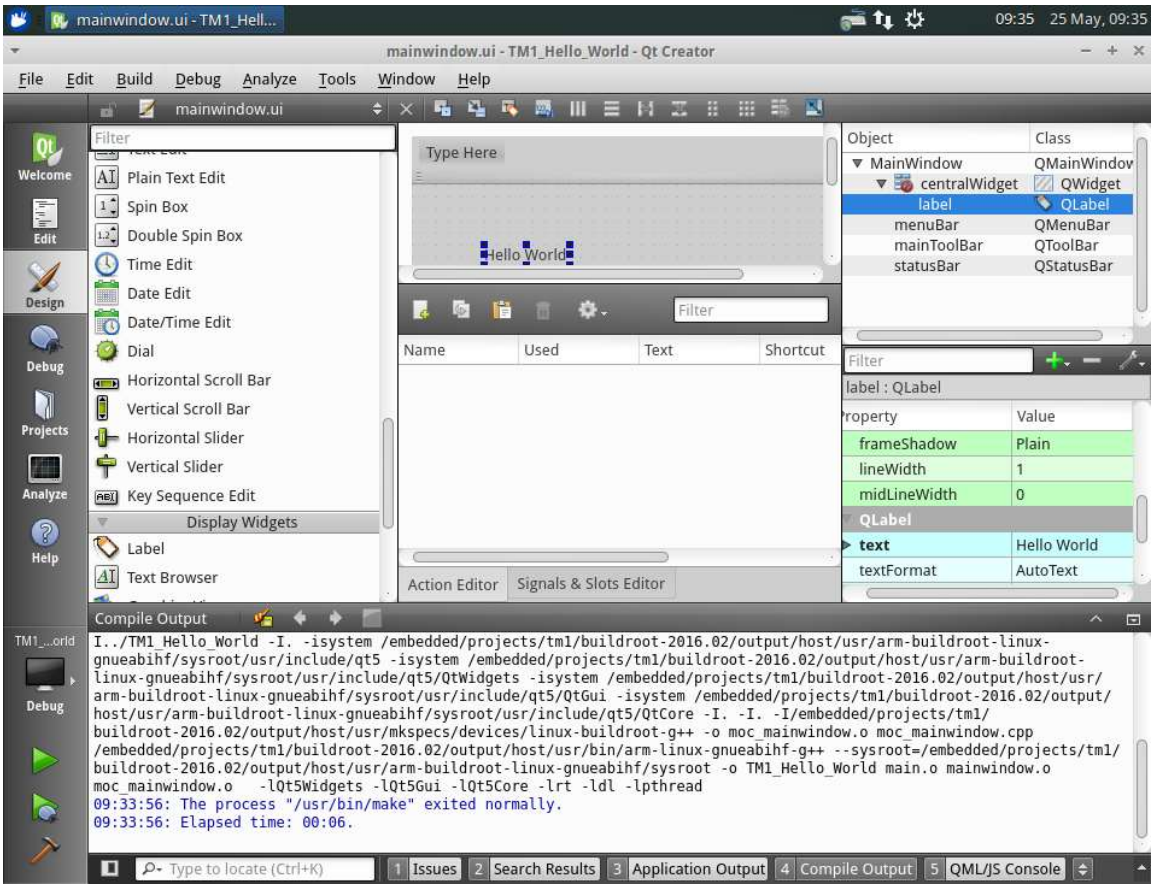
11. Scroll down to Display Widgets, and drag a label widget onto the form.
12. Use the property editor to change the label text to, “Hello World”



13. Select Build -> Build All (ctrl + shift +B), and click, “Save All” changes if prompted.



14. Monitor the “4 Compile Output” window for build completion without errors.



15. Select Build -> Run (ctrl + R) to deploy and run the application on the TM1 hardware.

8.2 QT5.9

8.2.1 Download and install QT Creator to the development machine

On the same development machine that was used build the QT5 Buildroot root filesystem issue the following commands to download QT creator, mark the download as executable, and run the installer.

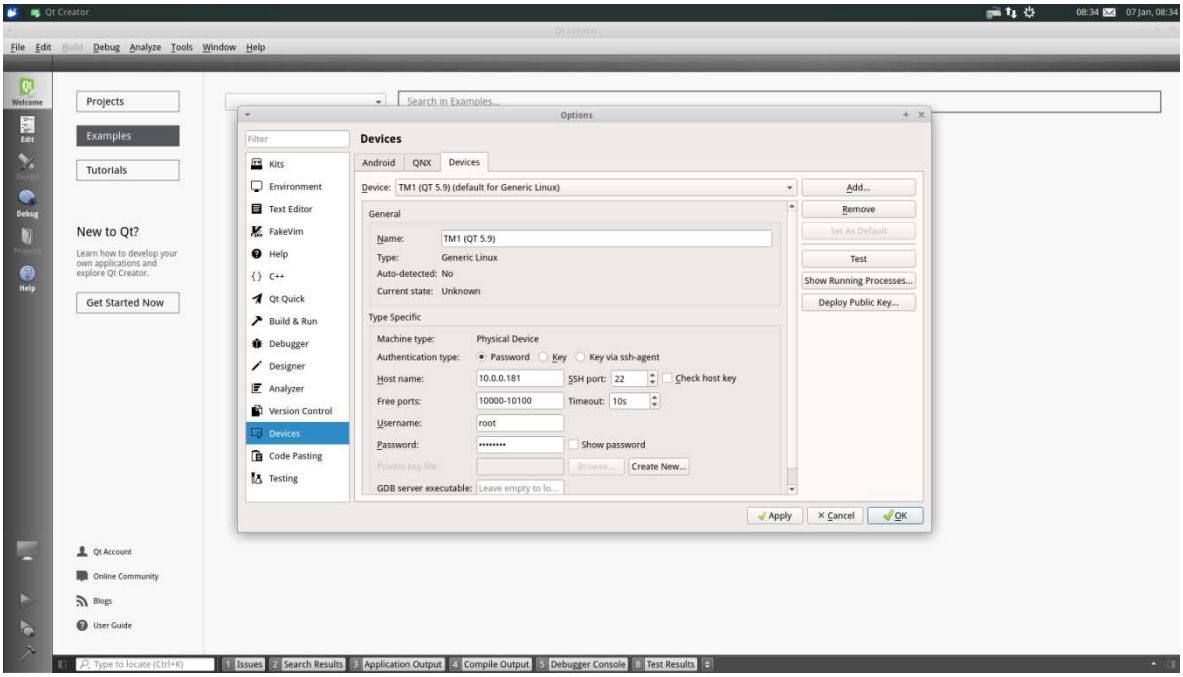
```
wget https://download.qt.io/archivechmod/qt/5.9/5.9.0/qt-opensource-linux-x64-5.9.0.run
chmod a+x ./qt-opensource-linux-x64-5.9.0.run
./qt-opensource-linux-x64-5.9.0.run
```

Follow the installer prompts, and use the default configuration suggestions.

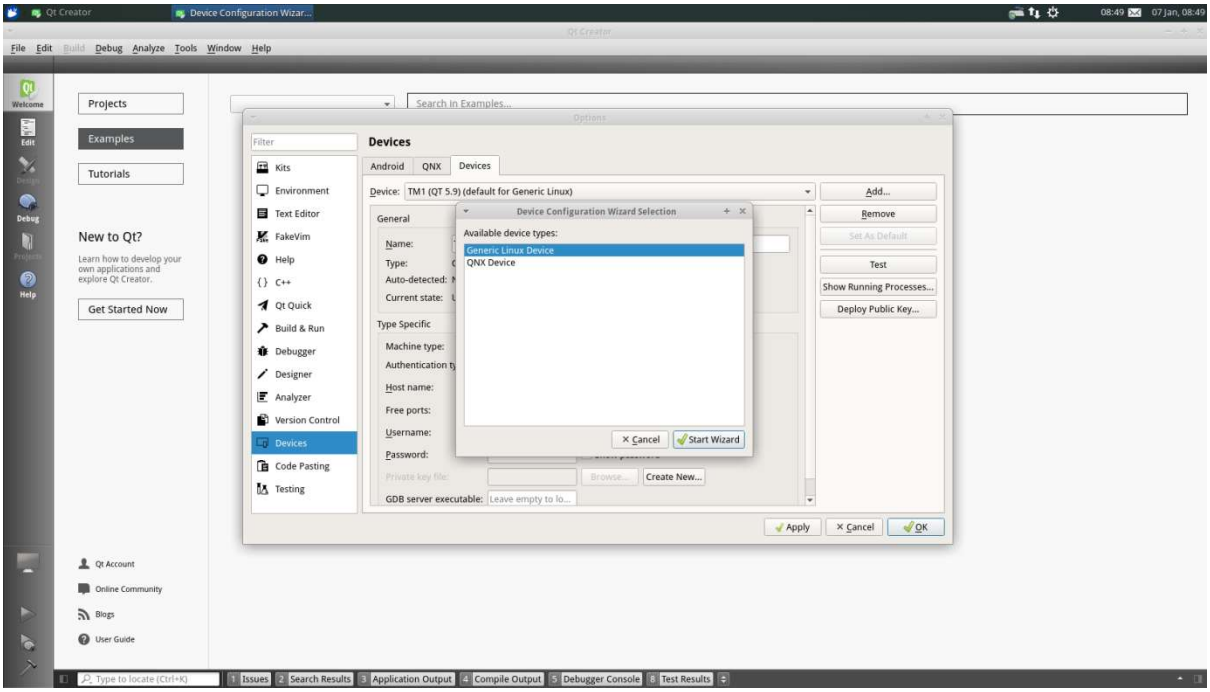
8.2.2 Setup the TM1 / HB5 environment in QT Creator

QT Creator uses the notion of “Kits” which refer to development environment configurations, targeting specific architectures, and devices. By default only a single kit is installed in QT creator that targets applications running in the host environment. This section will focus on the setup of a kit targeting TM1 running the Buildroot generated QT5 root filesystem created in section 4.2.2.

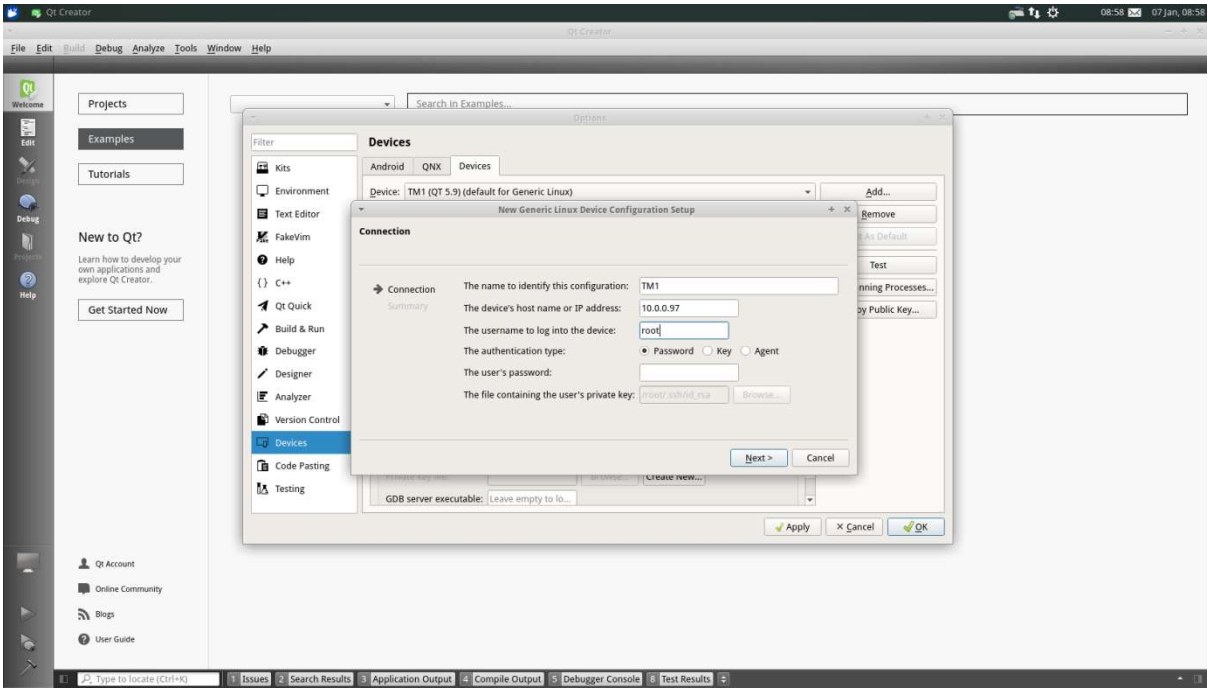
47. Ensure section 4.2.2 has been followed to create a QT5 based root filesystem for TM1/HB5
48. Use the TMx update utility to apply the QT5 image to TM1.
49. Boot the unit with an Ethernet cable attached. The LCD will display the IP address obtained via DHCP. Make a note of this IP Address.
50. Launch QT creator
51. Navigate to Tools -> Options
52. In the left hand pane select “Devices” and then select the Devices tab. (if you are using BSP VM 2.03 enter the IP address from TM1 and click “Test” and continue from step 60)



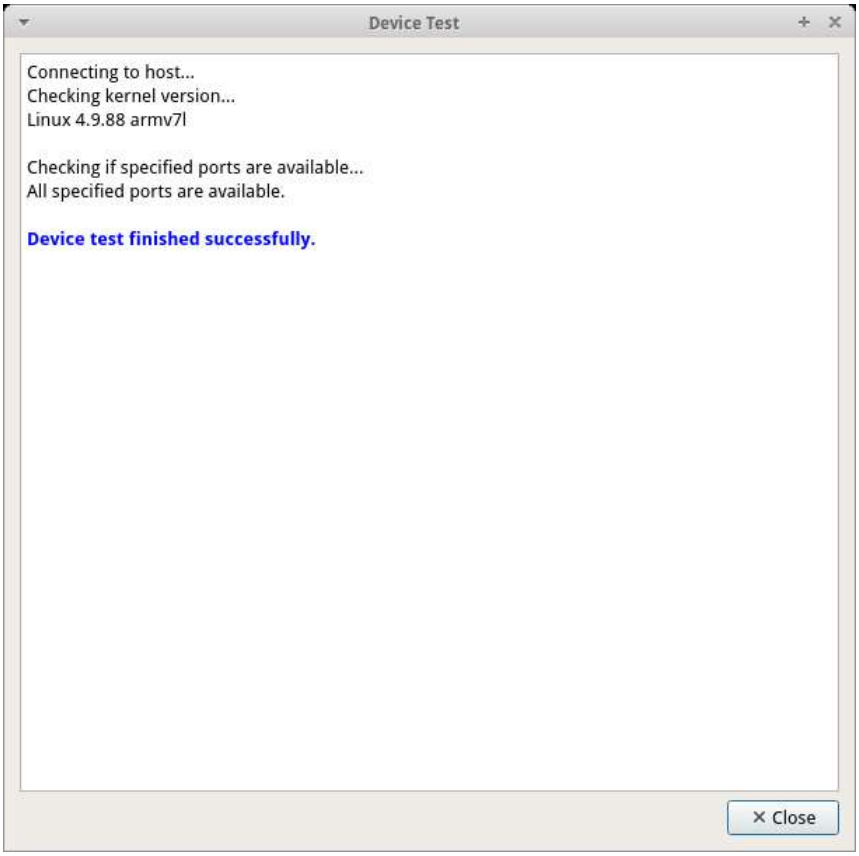
53. Click Add, select, “Generic Linux Device” and click, “Start Wizard”.



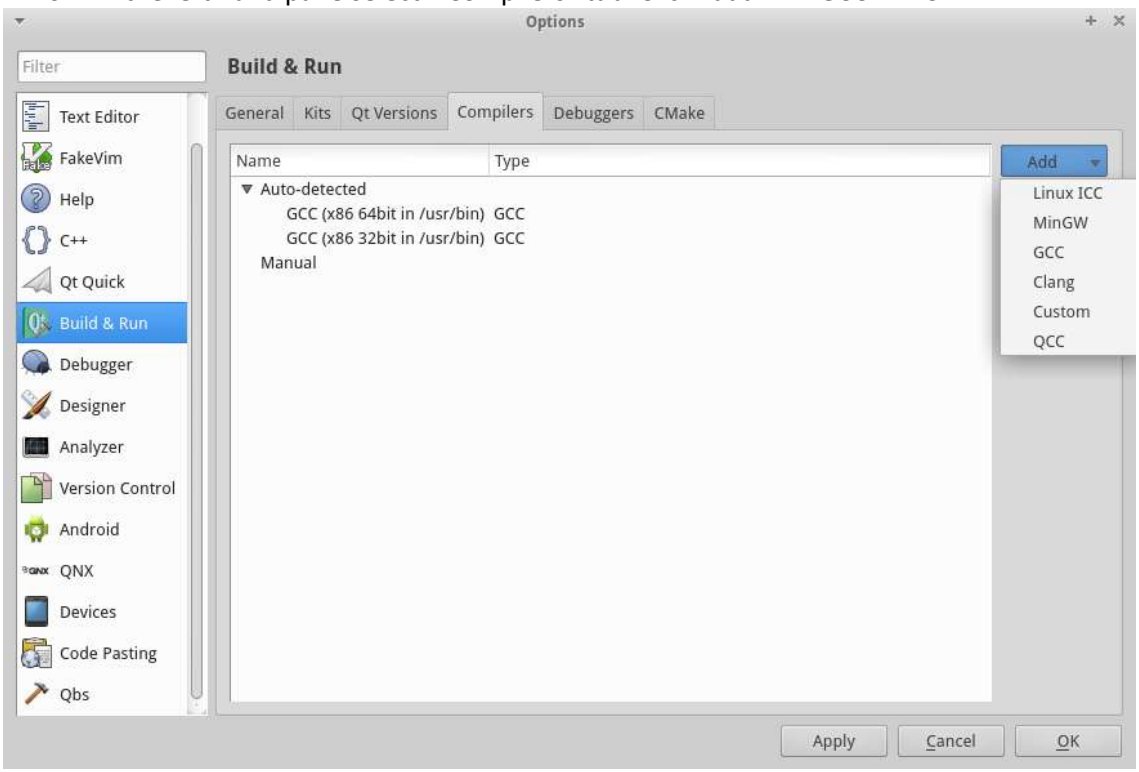
54. Set the device name to, “TM1 QT5.9”
55. Set the host name or IP address to the IP address noted down in step 3.
56. Set the username to, “root”
57. Set the authentication type to password.
58. Set the users password to, “password”



59. Click Next, click Finish and verify that the Device test was successful and click close.

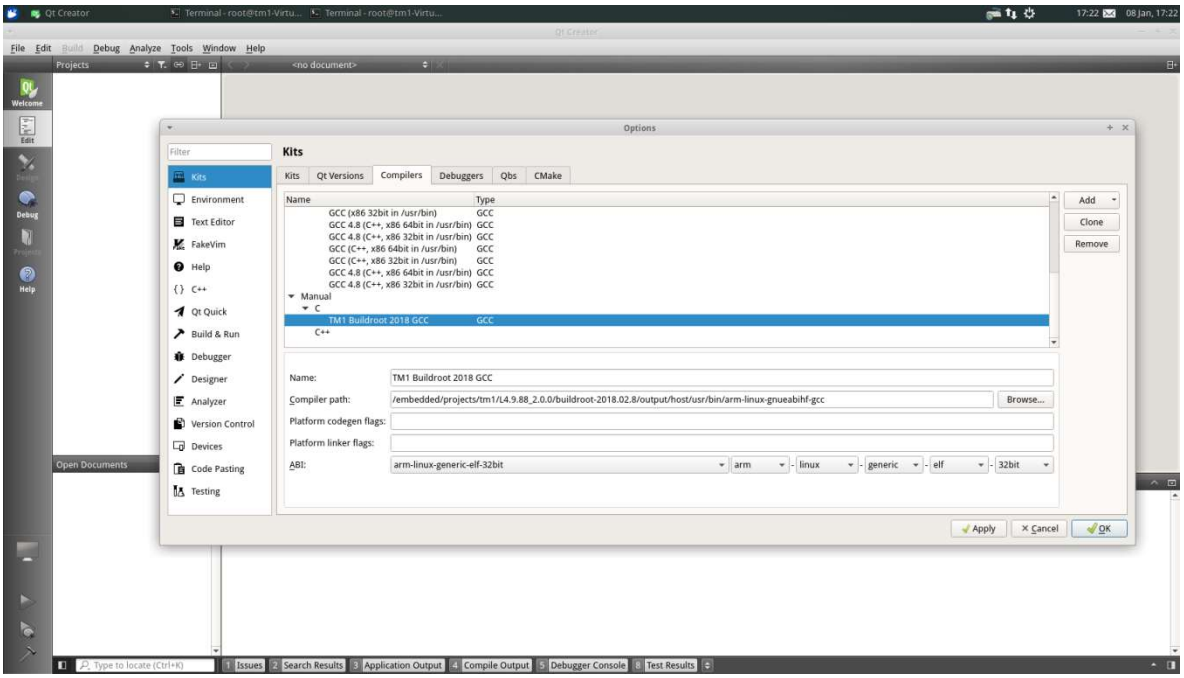


60. Press Apply in the Options Window.
61. In the left hand pane select “Compilers” tab. Click “add” -> “GCC ”->”C”



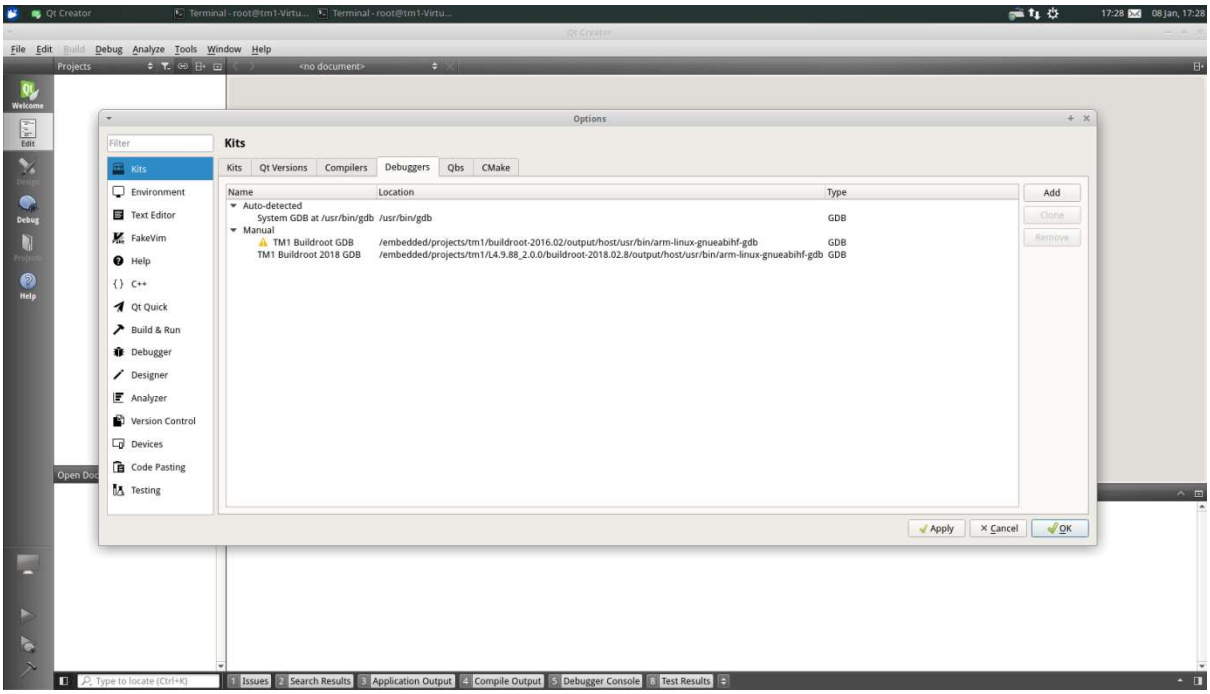
62. Set Name to, “TM1 QT5.9”.

63. Set Compiler path to, "/embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8/output/host/usr/bin/arm-linux-gnueabihf-g++”
64. Click Apply
65. Click “add” -> “GCC ”->”C++”
66. Set Name to, “TM1 QT5.9”.
67. Set Compiler path to, "/embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8/output/host/usr/bin/arm-linux-gnueabihf-gcc”

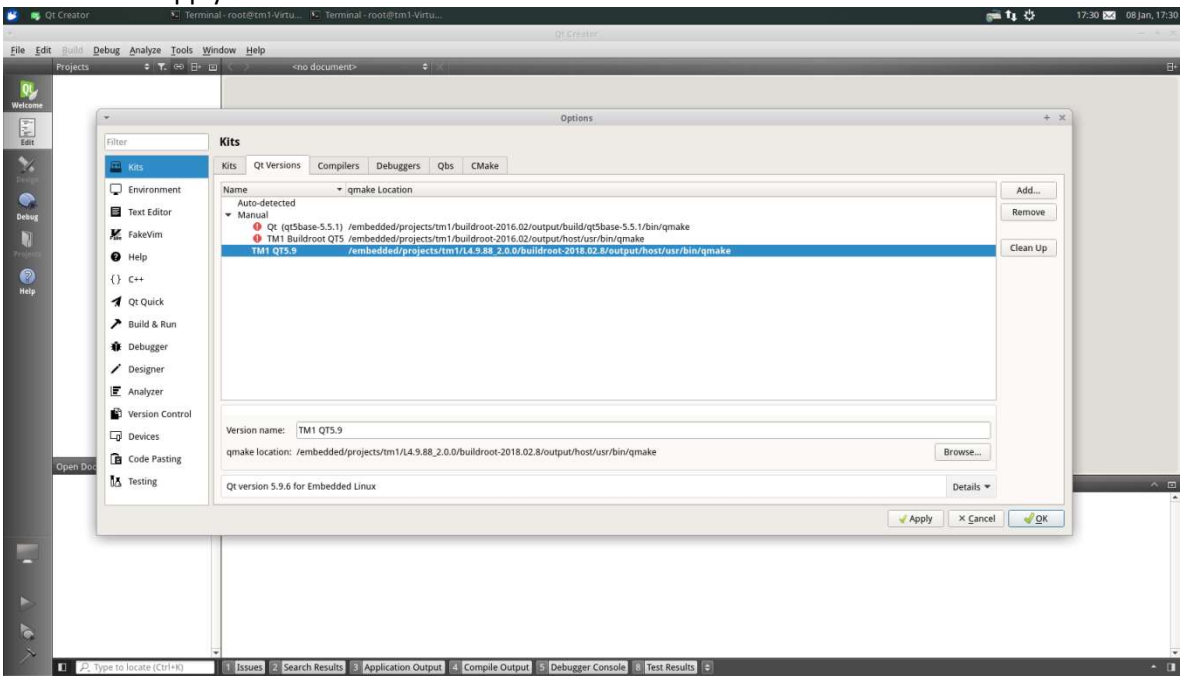


68. In the left hand pane select, the “Debuggers” tab. Click “add”.

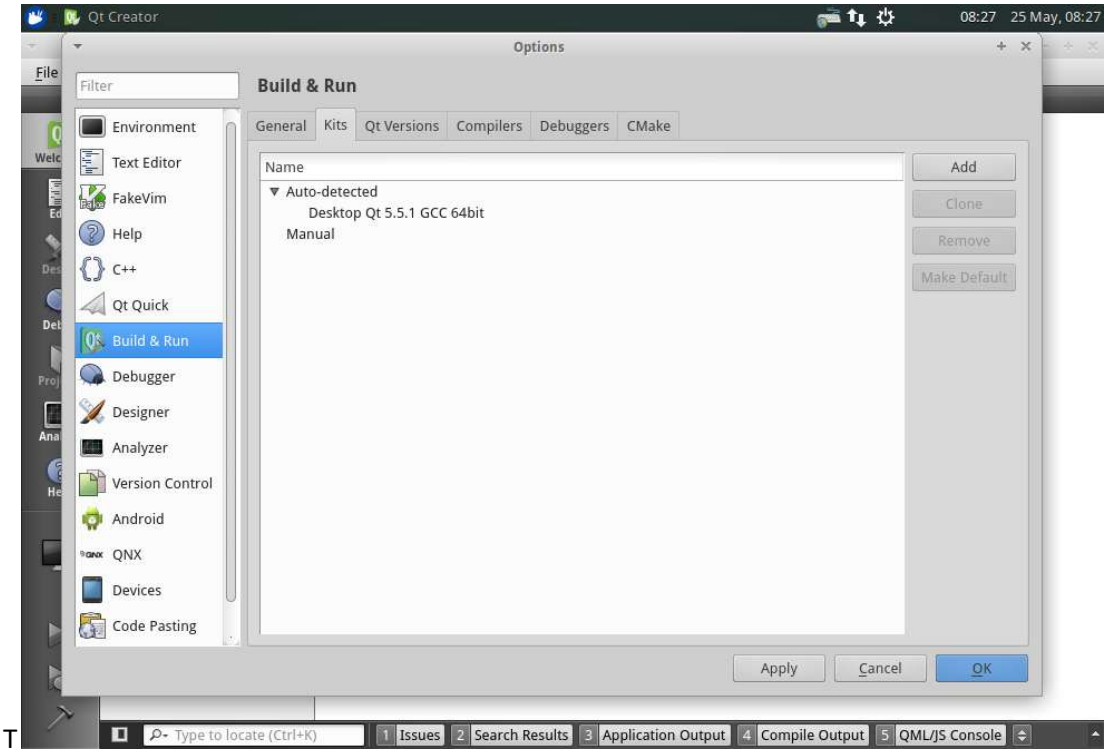
69. Set the name to, “TM1 QT 5.9 Debugger”
70. Set the path to, "/embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8/output/host/usr/bin/arm-linux-gnueabihf-gdb”
71. Click Apply



72. In the left hand pane select “Qt Versions” tab. Click “Add”.
73. Click add and select the qmake executable, /embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02.8/output/host/usr/bin/qmake”
74. Click Apply



75. In the left hand pane select, “Build and Run” and then select the “Kits” tab. Click “Add”.



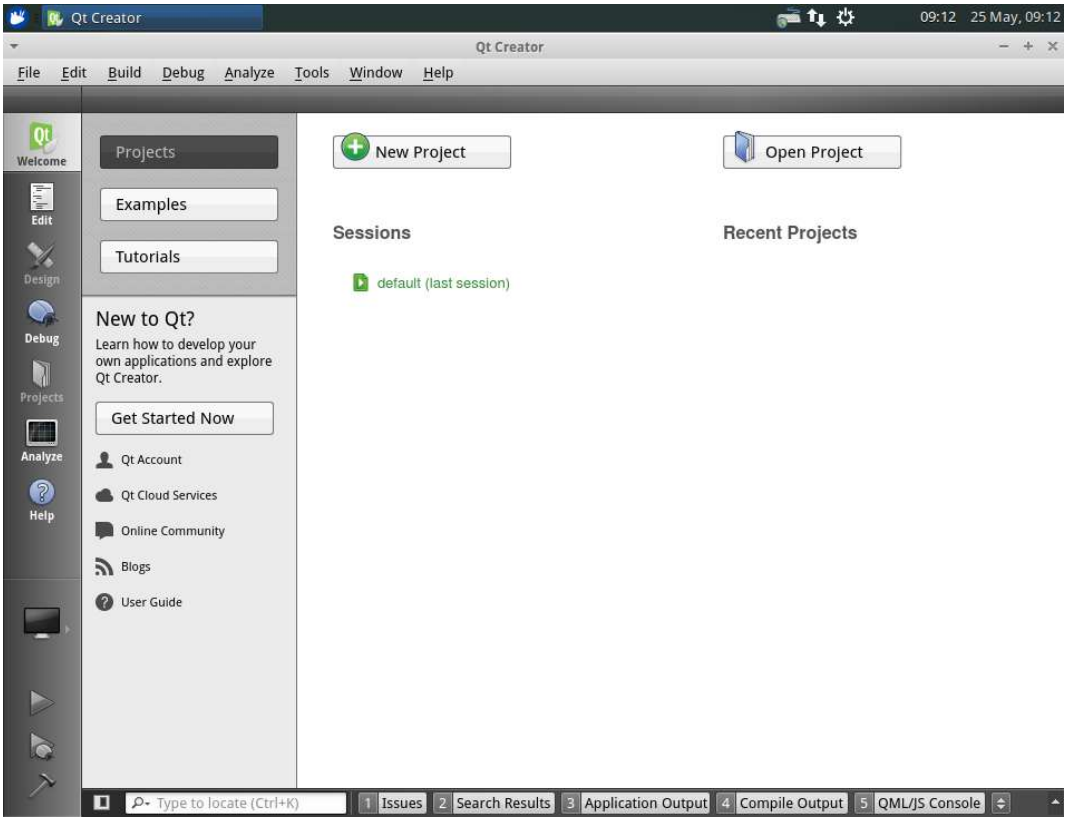
T

76. Set Name to, “TM1 (5.9)”
77. Set File system name to, “TM1”
78. Select Device Type to, “Generic Linux Device”
79. Select Device to, “TM1 (QT 5.9) (default for Generic Linux)”
80. Set Sysroot to, “/embedded/projects/tm1/L4.9.88_2.0.0/buildroot-2018.02/output/target”
81. Set Compiler C to, “TM1 QT5.9”
82. Set Compiler C++ to, “TM1 QT5.9”
83. Set debugger to, “TM1 QT 5.9 Debugger”
84. Set Qt Version to, “Qt 5.9.6 (System)”
85. Press “Make Default”
86. Press OK.

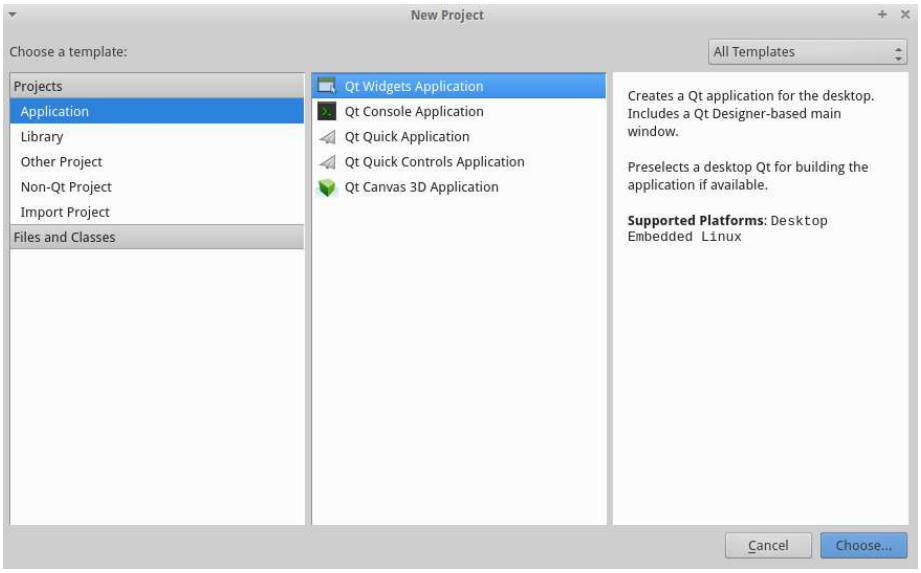
8.2.3 Setup a simple QT5 “Hello World” application

The following section will describe how to setup and deploy a simple “Hello world” application to TM1.

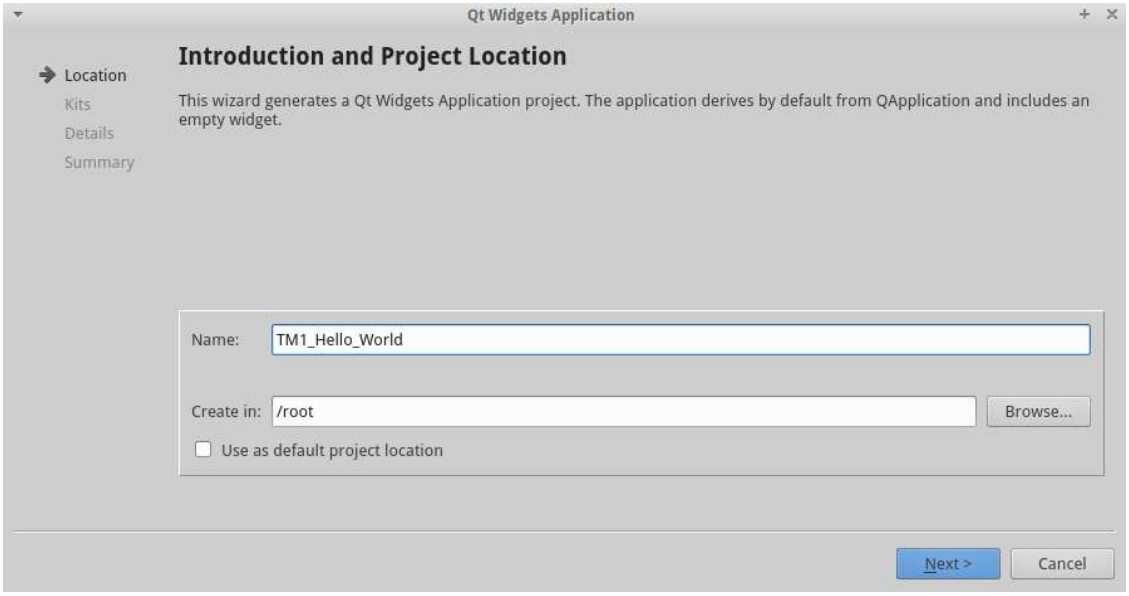
16. Launch QT Creator
17. Select, “New Project”



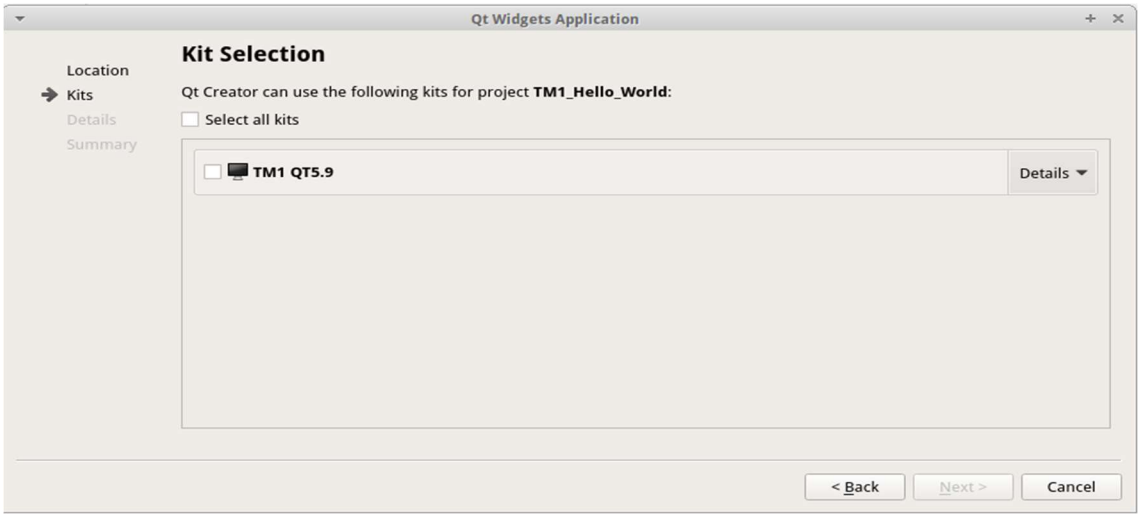
18. Select, “Qt Widgets Application” and click, “Choose”.



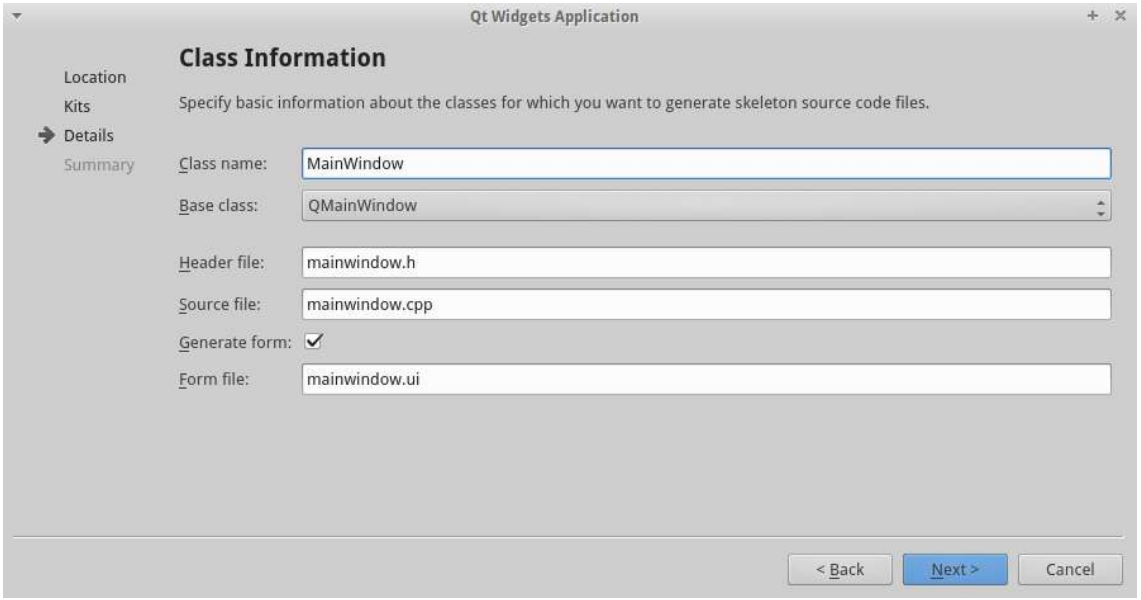
19. Set the name to, “TM1_Hello_World” and click next



20. Select the, “TM1 5.9” kit, and click next.



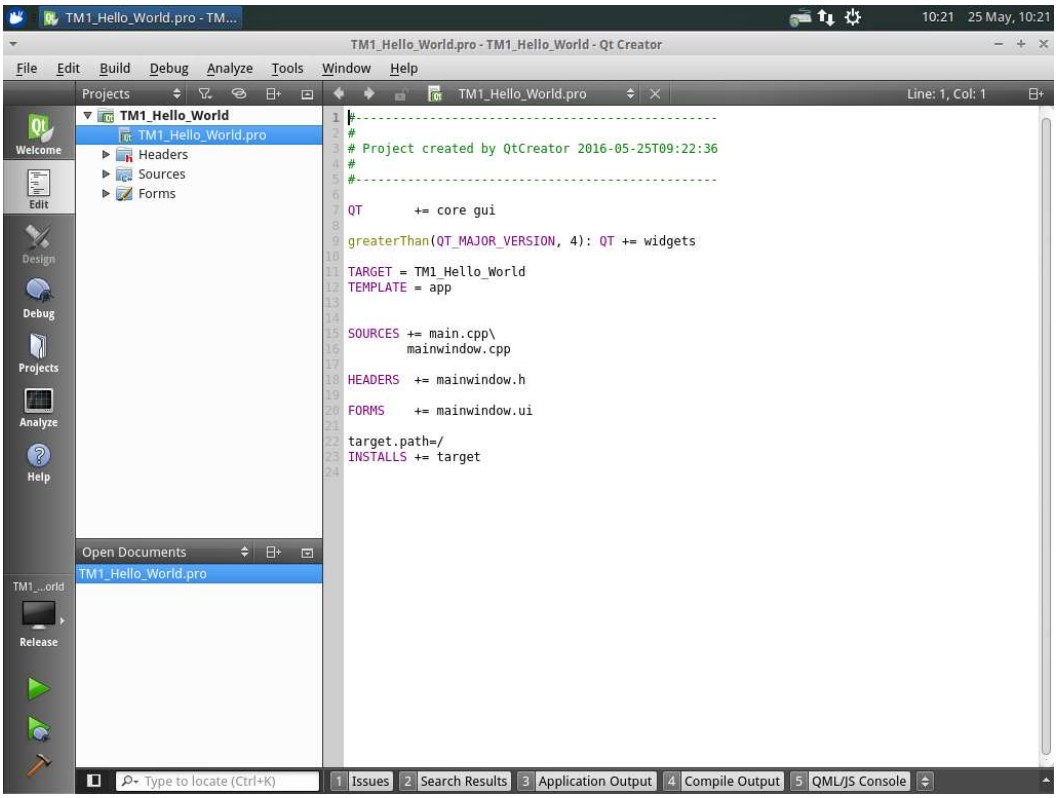
21. Use the default Class Information and click Next



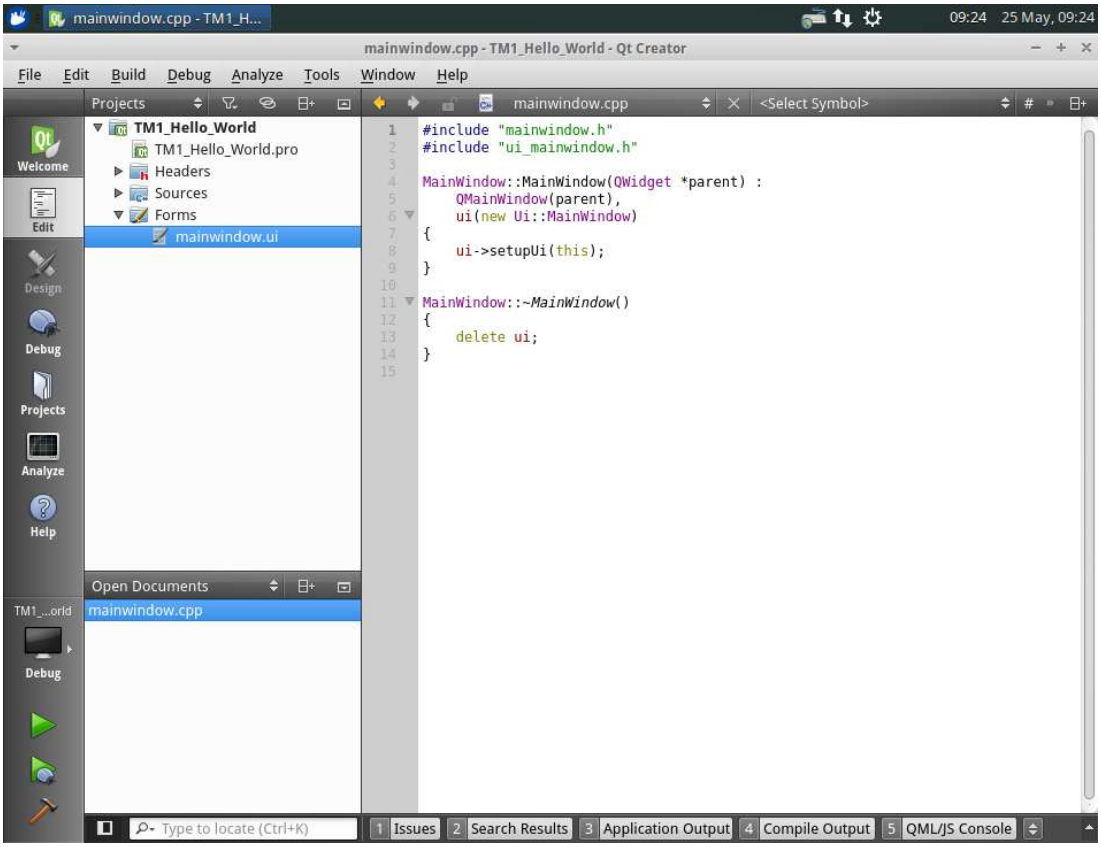
22. Click Finish
23. In the Projects view, double click, “TM1_Hello_World.pro” file, to open the project editor.

24. Append the following to the bottom of the project configuration. This will tell the deployment tool where on the target root filesystem to put the executable.

```
target.path=/
INSTALLS += target
```

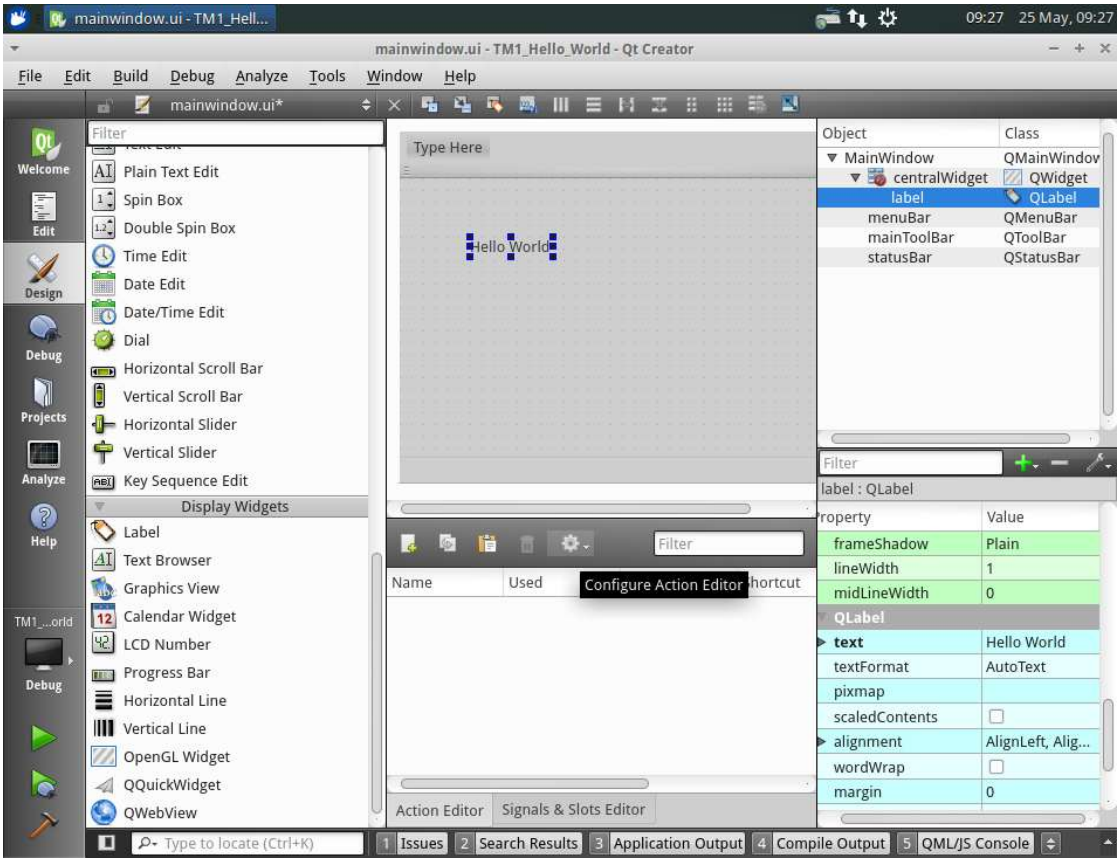


25. In the Projects view, double click, “mainwindow.ui”, to open the forms designer.

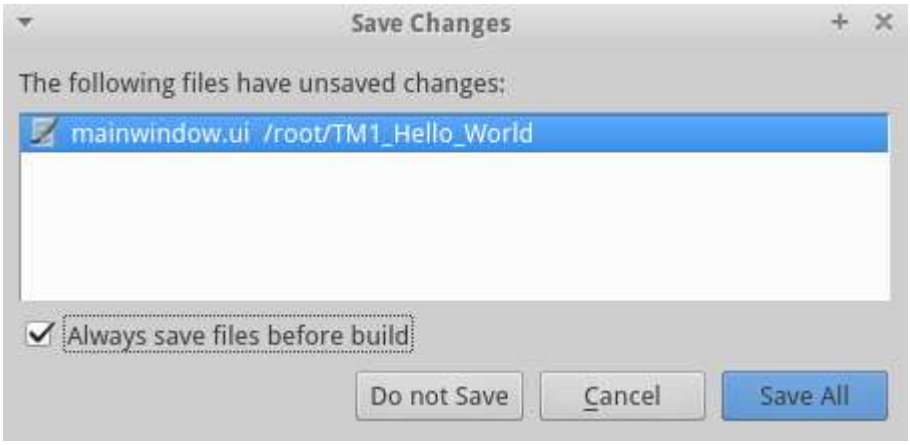


26. Scroll down to Display Widgets, and drag a label widget onto the form.

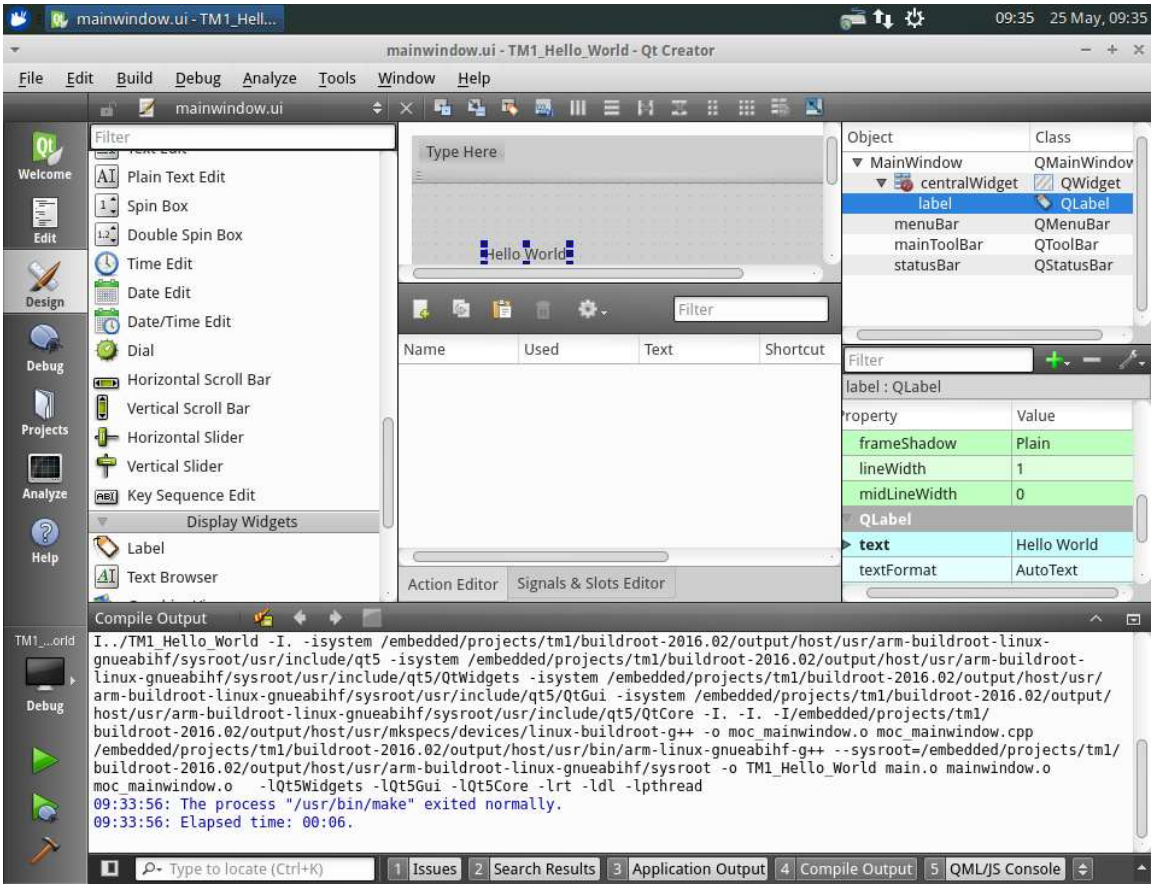
27. Use the property editor to change the label text to, “Hello World”



28. Select Build -> Build All (ctrl + shift +B), and click, “Save All” changes when prompted.



29. Monitor the “4 Compile Output” window for build completion without errors.



Select Build -> Run (ctrl + R) to deploy and run the application on the TM1 hardware.

Appendix A - Known Problems

With Releases prior to TMx-Update Utility 1.26 in the 4.9.xx linux kernel a mac address was set by default overriding the mac address programmed during manufacture. If you see a mac address of 00:C0:46:00:00:01 you may resolve the issue by updating to the latest source code tm1linuxv203.tar.bz2, git commit or later or by removing the reference to ethaddr in the uboot environment.

Power on TM1, Press space (repeatedly) in a connected serial terminal (115200, 8,1,n)

Once you have entered the uboot prompt issue

setenv ethaddr

saveenv

Appendix B - Change Log

Issue	Date	Author	Changes
1.4	13/03/2019	K Robinson	First formal version including L4.9.88 release
1.5	30/05/2019	D Robinson	Moved u-boot directory Added RS-485 UART section Added UART DMA and FIFO threshold control section
1.6	03/11/2020	D Robinson	Linux 4.9 updated to resolve suspend resume, and fix RTC
1.7	20/12/2021	D Burnard	Added section 3.6 Kernel 4.9 Persistent logo boot Documented fix & workaround for fixed mac address Updated to match BSP 2.03 Updated to match BSP 2.03 Virtual Machine Updated to match TMx Update Utility 1.26 Updated to support QT5.5 anf QT5.9 Build Process Updated to correctly reflect QT5.9 Build Process Updated GPIO pin definitions to match Hardware Docunmentation Kernels updated to resolve a compatibility issue with recent batches of uSD Cards
1.8	18/06/2023	M Olejnik	Added section 3.2.1 Compiling the Linux Kernel 4.9 with modularised WiFi drivers Added section 4.1.2 Buildroot git repository Added section 4.3.2 Building Buildroot with modularised WiFi/BT drivers
1.8.1	19/06/2023	M Olejnik	Updated section 5.2. Removed references to kernel 3.14 and added legacy kernel installation note.