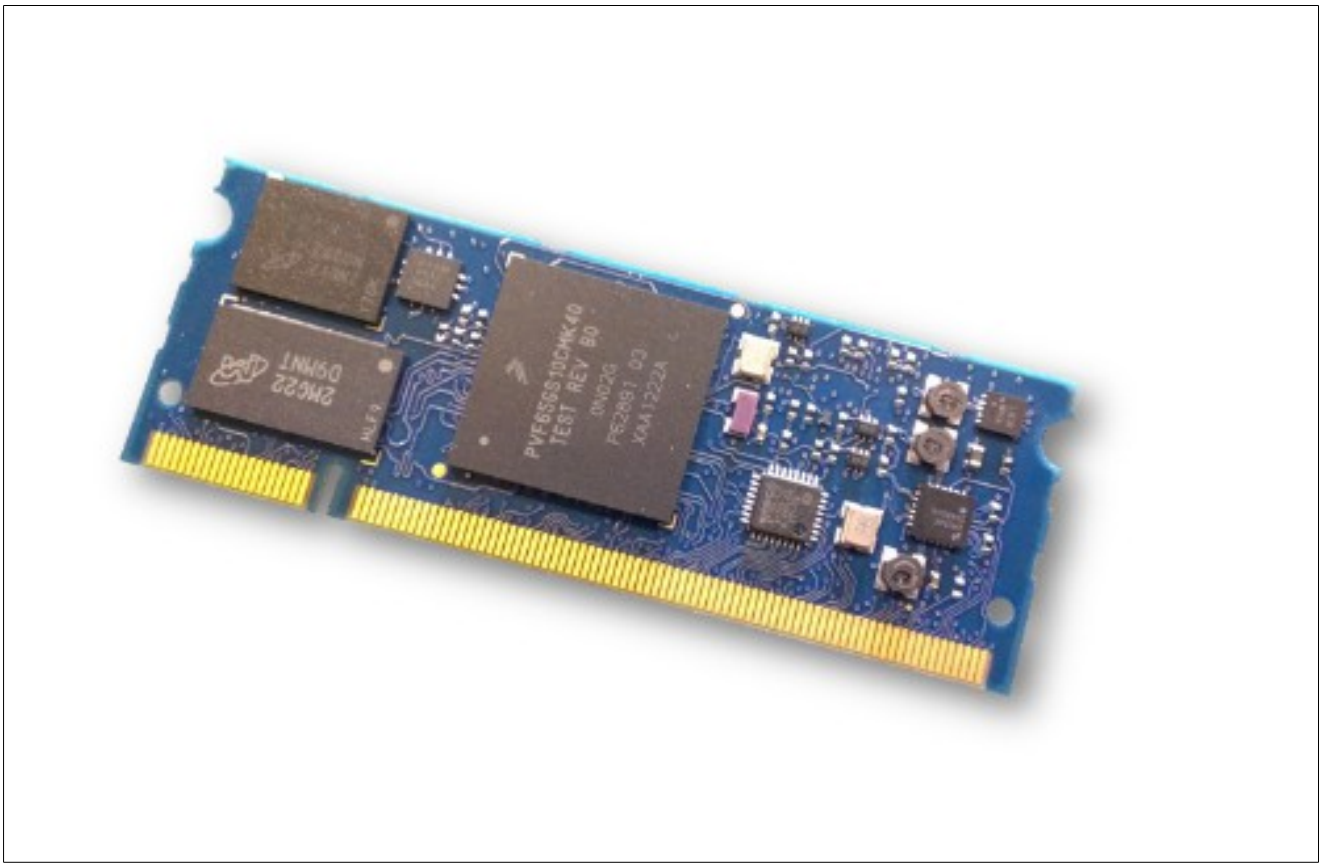
CHIMERA

Software getting started manual



***** REV 1.0.0 *****

DATE	REVISION	CHANGE DESCRIPTION
19/05/14	1.0.0	Release

Summary

1. Overview

The CHIMERA module is a SOM based on Freescale Vybrid VF61 processor.

CHIMERA supports Linux for Cortex A5 core and MQX for Cortex M4.
An inter-core communication API called MCC is provided to exchange data between Linux running on Cortex A5 and MQX applications running on the Cortex M4.

Linux SDK for Cortex A5 includes the U-Boot, the Linux Kernel e the filesystem for Cortex A5 and Freescale MQX for Cortex M4.

Chimera module can be booted using SD card or NAND memory. Please refer to Chimera hardware manual for details.

The U-Boot is the system's boot loader. It starts the machine and contains the basic informations to start the operative system. By the U-Boot it's possible to program the module and run the basic configurations on the machine (e.g. indicate the locations in which to find the Kernel and file system). The type of media to use to boot: NAND memory, MMC card or NFS and related parameters such as the screen resolution, the console settings, and much more.

The Kernel is the actual operating system of the module. It contains all the drivers of the machine as well as a real Linux Kernel. It's possible to change the Kernel by adding and removing modules or drivers according to customer's needs. Please for further information refer to the Chapter 4

The filesystem is the "hard drive" of the module. It contains all programs and applications required to operate the machine. These applications and programs specifically compiled for the platform ARM of the module can be managed using the configuration tool of the filesystem, LTIB. The filesystem also contains its own applications compiled as well as all the necessary files to your applications.

When stored on NAND these three components have a defined memory location that is used by default on all ENGICAM's modules, however, can be changed depending on the needed by the end user. These are the default locations

U-Boot	0x000000	0x400000
Kernel	0x400000	0x400000
filesystem	0x400000	size NAND

The best way to connect to the Linux console of the CHIMERA module is to plug the serial cable on the J28 connector of the CHIMERA starter kit using the following parameters: 115kbps, 8 bits, 1 stop and no parity. You can connect also the SSH, if it's pre-configured and supported.

1.1 The Virtual Machine

In order to develop your application and configure the U-Boot, the Kernel and the filesystem ENGICAM provides to the customer a virtual machine preconfigured and ready to work. The virtual machine is compatible with both VMware Player and VirtualBox. To use VMware player a commercial license is required, but a free evaluation version is also available by Vmware.
VirtualBox instead is usable with GPL licence.
ENGICAM recommends to use the last version available of Vmware player.
Please refer to Appendix at the end of the manual for VM login details.

1.2 BSP overview

The following software materials are available in the BSP:

- A virtual machine copy
- Documents and Manuals
- The default image to the recovery of the modules
- Any additional development tools

1.3 Licenses

Before install and run the virtual machine, it is recommended a careful reading of the license and the conditions of use of the various software available in the BSP.

1.4 Virtual Machine directories

Inside the virtual machine are located the LTIB configuration tools packages specifically tailored to the modules. So it is to be noted that the configuration and building of U-Boot and Kernel will be done outside of the LTIB and their compilation should not ever be included in the LTIB configuration. Please refer to the Chapter 5.1.

Uboot	/data/CHIMERA/uboot-chimera
Kernel	/data/CHIMERA/kernel-chimera
ltib directory	/data/CHIMERA/ltib
Target filesystem	/data/CHIMERA/ltib/rootfs
Cortex A5 toolchain	/opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/
MQX	/data/CHIMERA/MQX
Cortex M4 toolchain	/opt/arm-2012/ (to be installed by user, please refers to Cortex-M4 Toolchain installation chapter)
Default images	/home/user/Desktop/default_image

2. CHIMERA default image programming

The modules are usually provided by ENGICAM with only U-Boot loaded, the kernel and filesystem shall be programmed by the user. The BSP anyway contains the default image of Kernel and filesystem. Please consider that images by default does not contain all of the drivers and packages available, but only the basic ones.

The default images can be found on the VM Desktop in the /home/user/Desktop/default_image directory. The directory include the following files:

uboot.imx	u-boot image for SD card boot
uboot.nand	u-boot image for NAND memory boot
uImage	Linux kernel image
rootfs.tgz	compressed filesystem

2.1 SD card and NAND memory partitions

A SD card running on a Chimera Linux include the following sections/partitions:

- uBoot	A raw copy of uboot bootloader
- uImage	A raw copy of Linux kernel image
- rootfs	An EXT3 partition containing the filesystem.

The CHIMERA NAND memory is partitioned in the folllowing way:

/dev/mtd0	uBoot
/dev/mtd1	uImage
/dev/mtd2	It formats into ubifs and extracts the file rootfs.tgz

Each mtd partition which corresponds to memory areas is defined in the Kernel cmdline. To change these locations a change settings of the bootargs on uboot is needed.

2.2 Programming the SD card

In this chapter is described how to flash a Linux system on a SD card with a PC equipped with a SD card Reader and running the VM.

IMPORTANT: always remember to safety remove the SD card from your PC and never interrupt suddenly data transfer.

2.2.1 Flashing U-Boot

Please insert SD card in the PC card Reader and connect it on VM using Removable Device menu. A new device must be seen by VM (usually a new disk /dev/sdb). Please verify it before proceed using the following command:

```
user@ubuntu1004desktop:~/Desktop/default_image$ cat /proc/partitions
major minor  #blocks  name
8     0      41943040 sda
8     1      40450048 sda1
8     2      1        sda2
8     5      1489920  sda5
8     16     7561216  sdb
8     17     7540736  sdb1
```

When the SD card is properly seen please use the following commands to load u-boot:

```
$ cd  /home/user/Desktop/default_image
$ sudo dd if=u-boot.imx of=/dev/sdb bs=512 seek=2 && sudo sync
```

2.2.2 Flashing Linux Kernel

With the SD card Reader, we will flash the second part. Keep in mind that 1MB=1048576 Bytes -> Kernel Offset.
Take the Kernel files result as specified in the chapter "How to compile the Kernel"

```
$ cd  /home/user/Desktop/default_image
$ sudo dd if=uImage of=/dev/sdb bs=1M seek=1
```

2.2.3 Create ext3 partition

With the SD card Reader, an ext3 partition shall be created. The gparted tool can be used to partition the SD card:

```
$ sudo gparted
```

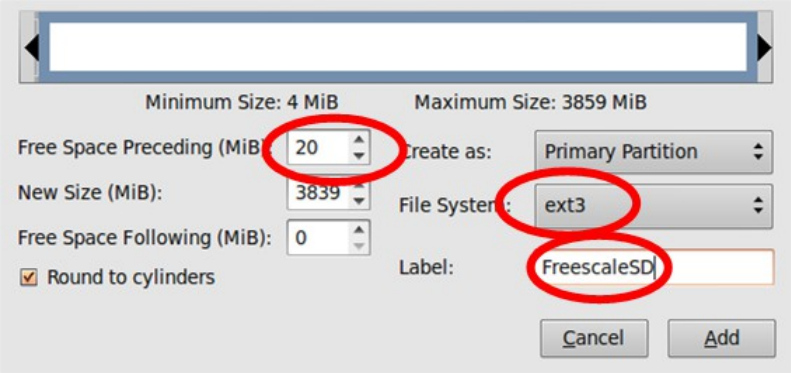


fig1

ADD and then Create a new ext3 partition, with 20MB of offset labelled FreescaleSD using Apply button

2.2.4 Copying filesystem and default image on SD

The following commands allow to copy the filesystem on SD:

```
$ cd  /home/user/Desktop/default_image
$ sudo tar xzvf rootfs.tgz -C /media/FreescaleSD/
$ sudo cp uImage /media/FreescaleSD/
$ sudo cp rootfs.tgz /media/FreescaleSD/
$ umount /media/FreescaleSD
```

The uImage and rootfs.tgz are also copied in order to be used for NAND programming.

2.3 Boot from SD card

Now it's possible to start the module running the U-Boot directly from the SD card by closing the jumper JM3 and powering the EVABOARD. Alternatively, you can still start the U-Boot already pre-installed on NAND flash and perform the Kernel and the filesystem present on SD.

In both cases, you must stop the machine in U-Boot (press a key on startup). The U-Boot is preconfigured to boot the machine from NAND but running the command

```
set bootcmd 'mmc read ${loadaddr} 0x800 0x2000;bootm'
set bootargs mem=128M console=ttymxc1 root=/dev/mmcblk0p1 rootwait rw mtdparts=NAND:4m(u-
boot),4m(kernel),246m(rootfs),-(aux)
```

The U-Boot will boot the module from SD card. After a few seconds the module will start to run. You will be prompted the Linux login, enter as **root** user.

2.4 How to program kernel and filesystem on NAND

Please boot the system from SD card, wait Linux boot and perform the root login.
The following script are already present in default rootfs:

```
sd_prog_all.sh      // Re-program  kernel and fs using file u-boot.bin, uImage and rootfs.tgz
sd_prog_fs.sh       // Re-program  fs using file rootfs.tgz
sd_prog_kernel.sh   // Re-program  kernel using file uImage
```

When the login is preformed the following steps shall be performed:

```
root@chimera ~$ cd /
root@chimera /$ /sd_prog_all.sh
```

Once the script is run the NAND memory is programmed and the system must be restarted. In order to run the NAND kernel and filesystem the u-boot variables must be changed as shown below:

```
Vybrid U-Boot > setenv bootcmd 'nand read 0x80010000 0x400000 0x400000; bootm 0x80010000'
Vybrid U-Boot > set bootargs mem=128M console=ttymxc1 ubi.mtd=2 root=ubi0:rootfs rootfstype=ubifs rootwait rw
mtdparts=NAND:4m(u-boot),4m(kernel),246m(rootfs),-(aux)
```

3. U-Boot

As already mentioned, you can change and configure U-Boot according to your own needs. It not recommended to make changes and even delete the first sectors of flash U-Boot where it resides, this to avoid "Brik " the module.
The official boot application is the "Das-U-Boot"

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

For further information about U-Boot please go to the official manual page:

http://www.denx.de/wiki/DULG/Manual

3.1 U-Boot compiling

In the following folder:

/data/CHIMERA/uboot-chimera

you can find the configuration file you can use to make the U-Boot for your modules.
There is a single file to configure and compile the U-Boot for CHIMERA modules:

1. **./config_chimera.sh**

the exit file of the compiling is one of the following files:

1. **u-boot.imx** **the padded file for SD card boot**
2. **u-boot.nand** **the non-padded file to be programmed on NAND**

To program the U-Boot refer to the chapter on programming images by default.

3.2 U-Boot Environment Variables

Via U-Boot it's possible to manage the environment variables. Using the "**printenv**" command it's possible to have a list of the environment variables. Using the "**setenv**" command it's possible to setting variable values,

set serverip 192.168.1.2

To delete an environment variable using the following command

set serverip

To reprogram the U-Boot, the environment variables are reset to the default value that should already be pre-set to boot from NAND module. In the case of loss of environment variables these are the default values:

baudrate=115200
bootargs=mem=128M console=ttymxc1 root=/dev/mmcblk0p1 rootwait rw mtdparts=NAND:4m(u-boot),4m(kernel),246m(rootfs),-(aux)
bootcmd=nand read 0x80010000 0x400000 0x400000; bootm 0x80010000
bootdelay=3
eth1addr=00:e0:0c:bc:e5:61
ethact=FEC0
ethaddr=00:e0:0c:bc:e5:60
gatewayip=192.168.2.1
ipaddr=192.168.2.107
loadaddr=0x80010000
mem=129024k

```
netmask=255.255.255.0
serverip=192.168.2.57
stderr=serial
stdin=serial
stdout=seriall
```

3.2.1 Parameters to boot from SD card

Use the following command to set the parameter to boot from SD:

```
Vybrid U-Boot > set bootcmd 'mmc read ${loadaddr} 0x800 0x2000;bootm'
Vybrid U-Boot > set bootargs mem=128M console=ttymxc1 root=/dev/mmcblk0p1 rootwait rw mtdparts=NAND:4m(u-
boot),4m(kernel),246m(rootfs),-(aux)
```

3.2.2 Parameters to boot from NAND

Use the following command to set the parameter to boot from NAND FLASH:

```
Vybrid U-Boot > setenv bootcmd 'nand read 0x80010000 0x400000 0x400000; bootm 0x80010000'
Vybrid U-Boot > set bootargs mem=128M console=ttymxc1 ubi.mtd=2 root=ubi0:rootfs rootfstype=ubifs rootwait rw
mtdparts=NAND:4m(u-boot),4m(kernel),246m(rootfs),-(aux)
```

3.2.3 Parameters to boot from network

Use the following command to set the parameter to boot from NAND FLASH:

```
Vybrid U-Boot > set bootcmd 'tftp uImage;bootm'
Vybrid U-Boot > set bootargs mem=128M console=ttymxc1 ip=<chimera ip> root=/dev/nfs nfsroot=<VM ip
address>:/nfs_chimera rootwait rw mtdparts=NAND:4m(u-boot),4m(kernel),246m(rootfs),-(aux)
```

where **<chimera ip>** is the IP address of the CHIMERA board and **<VM ip address>** is the IP address of VM machine that runs the NFS server.

3.3 How to program Linux kernel on NAND from U-Boot

With the U-Boot is also possible to write on the NAND flash. It is possible to download data and files with tftp tools (via LAN). Following an example of the Kernel programming command:

```
Vybrid U-Boot > tftp 0x80010000 uImage
Vybrid U-Boot > nand erase 400000 400000
Vybrid U-Boot > nand write 0x80010000 400000 400000
```

3.4 U-Boot Self-Upgrade on NAND

To reprogram u-boot directly by u-boot command line, the following procedure shall be used:

```
Vybrid U-Boot > tftp u-boot.nand
Using FEC0 device
TFTP from server 192.168.2.58; our IP address is 192.168.2.107
Filename 'u-boot.nand'.
Load address: 0x80010000
Loading: ###############
done
Bytes transferred = 204924 (3207c hex)

Vybrid U-Boot > nand erase 0x0 0x400000
NAND erase: device 0 offset 0x0, size 0x400000
```

Erasing at 0x3e0000 -- 100% complete.
OK

Vybrid U-Boot > nb_update 0x80010000 0x322000 0x80000
Flashing image @ 0x80010000; size 3284992 to 0x80000

3.5 Running Scripts From Flash

In addition to set the values manually, to download applications and to run command by command it's possible to group them in a script file which will be loaded via network in ram (the implementation of the scripts will be described later in the Chapter 10.2). All scripts and all file will be shared via tftp server.

The virtual machine contains an internal tftp server already preconfigured in which the folder labelled tftpboot is already set for file transfer. See Chapter 10.10 in the appendix for configurations and settings

Similarly you can use a TFTP server for window, tool tftp32.exe. This server present on the DVD's virtual machine makes available a folder in which are running the tftp services.

Depending on the server used, the proper IP address will be set on the U-Boot: the IP of the physical machine (for tftp32.exe) or the IP of the virtual machine to atftp.
These are the steps required to load and run a script:

- Enter in U-Boot
- Set **serverip** variable with the VM IP using the command "***set serverip <192.XXX.XXX.XXX>***"
- The module must have the same subnet of your pc you can check writing the "***print ipaddr***"
- If you have not the same subnet, the IP should be changed with the command: "***set ipaddr 192.XXX.XXX.XXX***"
- Try to ping your own pc "***ping 192.xxx.xxx.xxx***"
- If all is OK run the command: "***tftp <name>.scr***"
- Then run: "***so***"
- All the file will be download
- Once the prompt is available again restart the module

All operations present in <name>. scr will be executed.

4. Kernel

The Kernel is common, referring to the various "sizes" of the CHIMERA modules. The only different that remains is for modules type RQS or standard. This information must be included in the configuration setup. Once compiled, follow the instructions of the programming of the default images for how to do the Kernel loading.

4.1 How to compile the Kernel

To configure and compile the Kernel go to the following directory:

/data/CHIMERA/kernel-chimera

and run the kernel configuration and build batches:

./config_chimera.sh

./build_linux.sh

At the end of build you'll find the file uImage in the following directory:

***/data/CHIMERA/kernel-chimera*/arch/arm/boot**

Internal Kernel modules can be built using the following script:

./build_modules.sh

The following procedure shall be used to build external Kernel modules:

export ARCH=arm
export CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/arm-none-linux-gnueabi-
export PATH=$PATH:/opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/
make modules

To install external modules use the command below:

INSTALL_MOD_PATH=/data/CHIMERA/ltib/rootfs/lib/modules make ARCH=arm CROSS_COMPILE=arm-linux-modules_install

5. Filesystem

The file system contains all the packages and applications needed to run Linux on the module. To generate the filesystem it's possible to use a LTIB Linux target configuration customized for the module and for Freescale processors. LTIB is a configuration tool that allows the user to select the individual packages, to compile them along with their dependencies and it places them correctly in the filesystem, inside the rootfs folder.

By adding or removing the various packages will make possible the creation of the different executables and files in the rootfs folder.
For more information on LTIB:

http://ltib.org/

5.1 How to generate a filesystem

To start the configuration use the following command:

./ltib -c

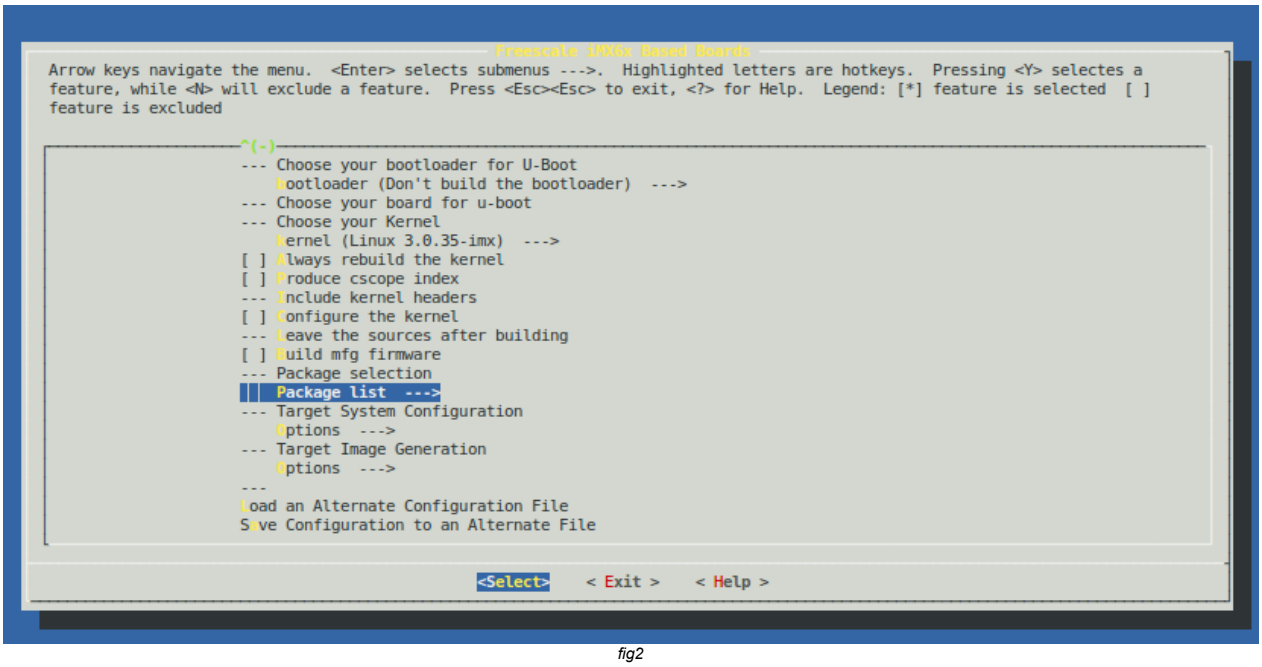


fig2

Select the package list option, as in the previous figure and then select the packages to install on the target. (See figure below)

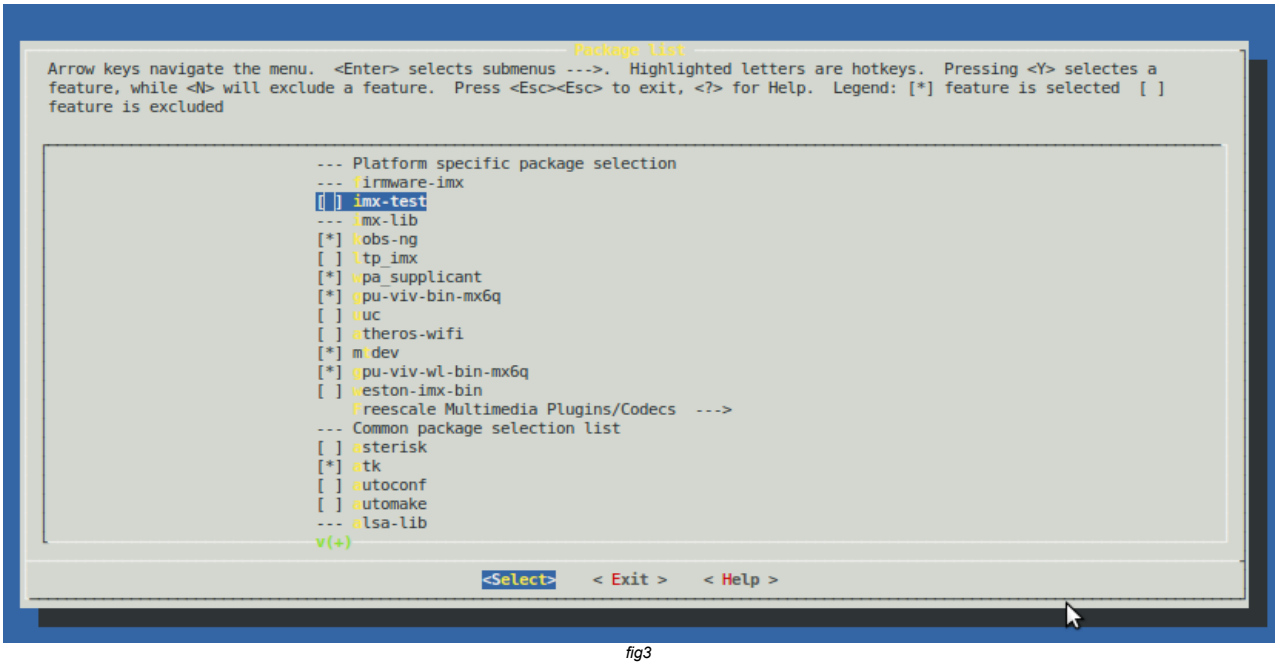


fig3

After the build command it's possible to find the files that make filesystem inside the folder:

/data/CHIMERA/ltib/rootfs

Now it's possible to customize the installation with an own file application, or with an external lib.

5.2 Configuration of LTIB

Since the Kernel and the U-Boot are independent from LTIB, it must be use only to generate the filesystem; for this reason the following options have not to be selected.

Leave them unselected:

[ ] Always rebuild the kernel
[ ] Produce cscope index
[ ] Include kernel headers
[ ] Configure the kernel

Even about the Qt is to be noted that the default image, (prepared and pre-installed Qt4.8 version) has been inserted externally to LTIB.

5.3 Adding a Package to LTIB

It's possible to manually add the new software packages in the list of configuration tools LTIB.
Adding the source package, LTIB is in charge of running the configure running the make command and the install procedure into the rootfs. To add or upgrade a package with a new version, please refer to the official documentation on the website or inside the virtual machine:

/home/user/Desktop/Manuals/LTIB_generic_v1.4_-_version_6.4.1.pdf

WARNING:
The add packages operation in LTIB is suggested only for advanced users.

It's possible to add software packages also simply cross-compiling for the target and then insert them manually in the rootfs folder.

5.4 Flashing filesystem on SD card

Now it's possible to copy the folder /data/CHIMERA/ltib/rootfs inside the ext3 partition of an SD card formatted and prepared as described in the chapter 2.

A ./make_tgz script can be found in ltib directory to compress the filesystem in a rootfs.tgz file.

5.5 Flashing filesystem on NAND flash

To save the data on the NAND memory is necessary to realize an rootfs.tgz archive, which will then be copied to an SD card with a Linux support system containing the programming scripts (please refer to chapter 2 for details).

6. How to build a C application

Built a "Hello World" application in C is easy and fast. For example try to compile the following code:

Hellomake.c

```
#include "hellomake.h"
int main()
{
        // call a function in an other file
        myPrintHelloMake();
        return(0);
}
```

Hellomake.h

```
//example include file
void myPrintHelloMake(void);
```

Hellofunc.c

```
#include "hellomake.h"
void myPrintHelloMake(void)
{
        printf("Hello makefiles!\n");
        return;
}
```

Makefile

```
CC=/opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/arm-linux-gcc
CFLAGS=-I.
all:  hellomake

hellomake: hellomake.o hellofunc.o
        $(CC) -o hellomake hellomake.o hellofunc.o -I.
clean:
        $(RM) *.o *.lo hellomake
```

Note that the only difference between a normal compiling is the calling in the makefile to the ARM compiler version of gcc.

To modify the makefile run the following command from prompt:

make **CC=*arm-linux-gcc***

Now it's possible to copy and run the application on the end target.

7. How to build a QT application

This section will show all the processes of creation, compilation, exportation and execution of a QT application of the virtual machine to the final target.

Warning
Remember to never remove the rootfs folder which contains the Qt4.8 compiled for CHIMERA. This operation could generate errors during the compile of your Qt application

7.1 How to create or import a project in Qt Creator

The Qt-creator is also installed into the VM with the target setted up for ARM and for x86.

Go to → Application → Programming → Qt-creator

It's possible to open an existing project by the ***file --> open file or project*** like the example in the folder

/home/user/Desktop/Examples/hello_world

Or start a new project by selecting from the file menu, new file or new project and selecting **Qt Gui Application**

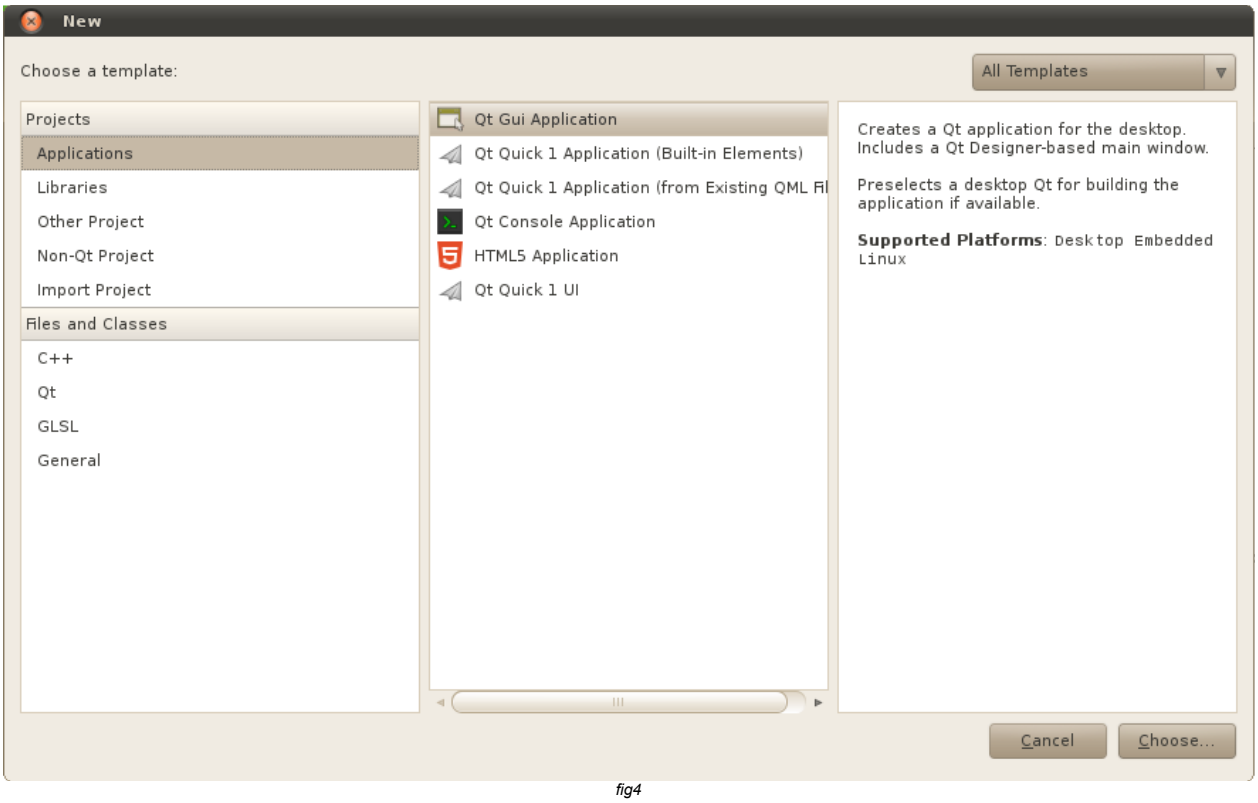


fig4

Insert the path of the project and press "**Next**", add the ARM on kit selection and press "**Next**".

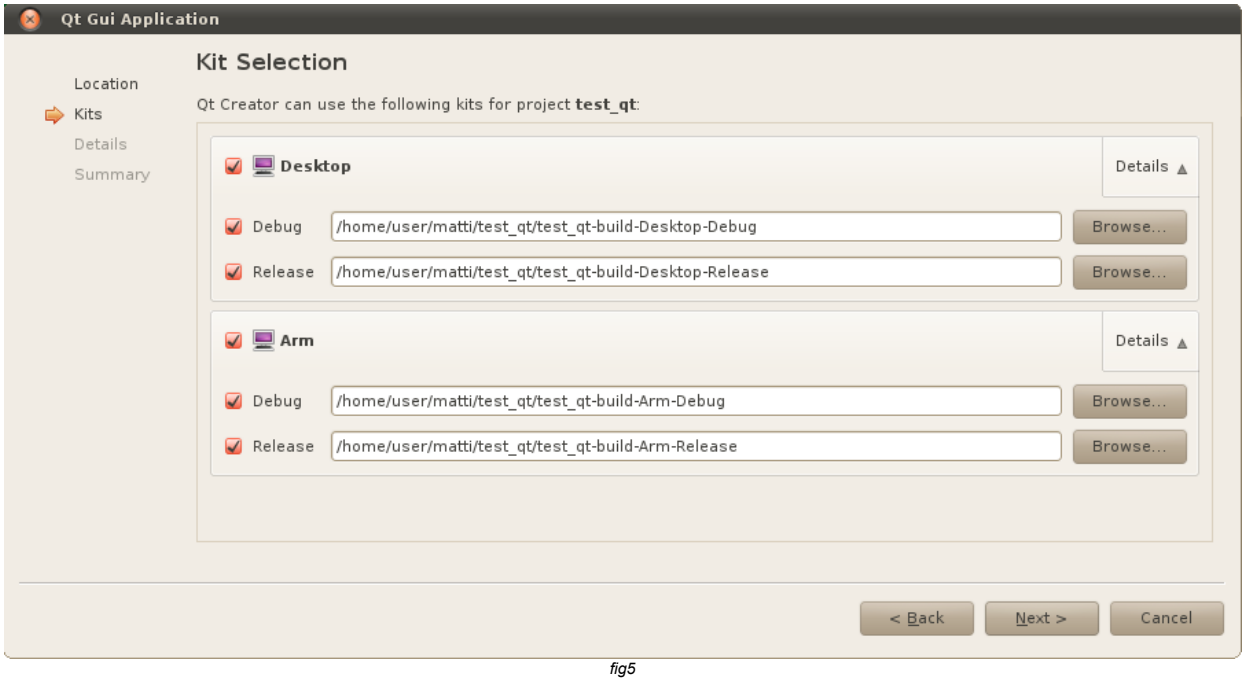


fig5

Press **Next** button on class Information and then press **Finish**.

Now it's possible to work editing and modifying the project.

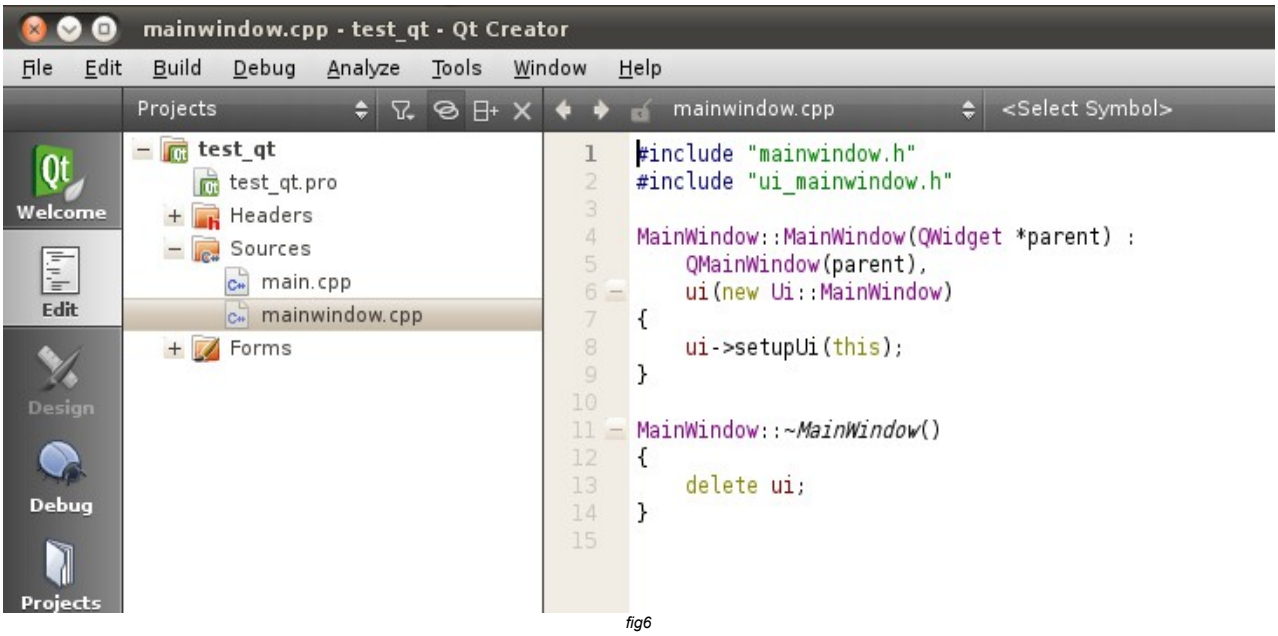


fig6

4 different type of build were done in the project: 2 for the module and the other 2 for x86. In this way it's possible to compile and to debug from PC and when required move the compiling on the module.
It's possible to move from the 2 configurations by following the commands shown in the next image:

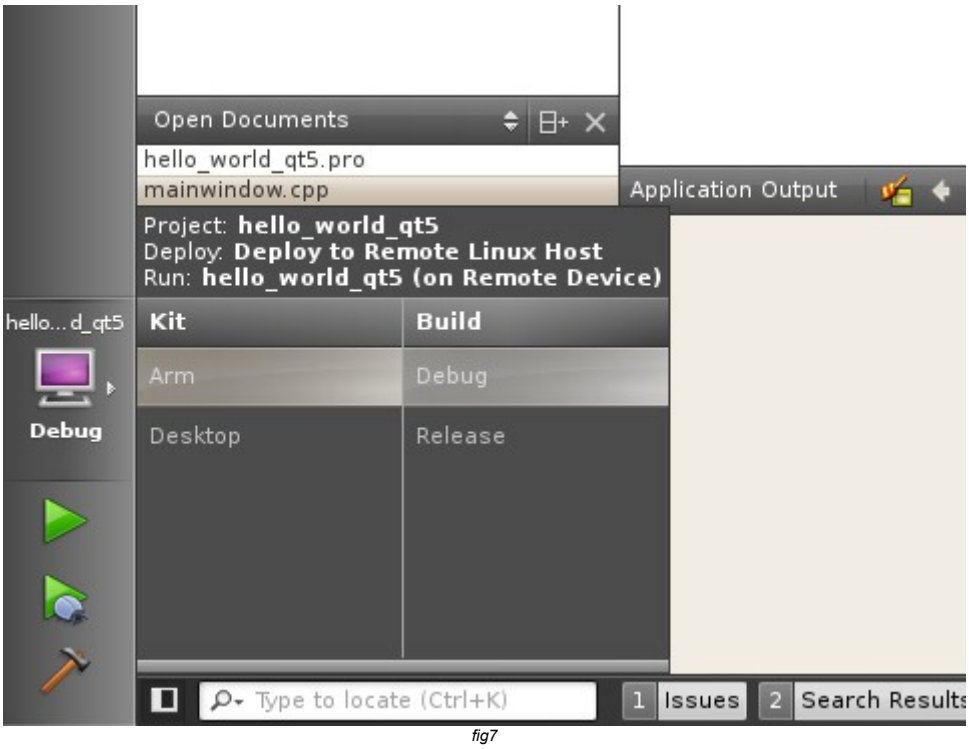


fig7

It's possible to run or to debug the Desktop QT by running the test program included in Ubuntu.
By pressing the hammer icon it's possible to build only the project for the selected configuration. At the end of building the compiled binaries will be present inside the solution folder.

7.2 Move the application on the target

First of all the compiled binaries and the required files for its execution, have to to be transferred on the final target before being run them. Now there are three different ways of operating. The best way and certainly quicker, is to create an own projects inside the directory **/nfs_chimera**, mount it using NFS server and work via LAN. Once compiled the binary file with the PC it's possible to find it in the right folder.

7.2.1 NFS shared folder

Inside the virtual machine exists a NFS folder **/nfs_mount** already preconfigured to share files on the network. To further informations see the Chapter 10.6, in the Appendix, that describes the installation and configuration of a NFS server. To mount a directory available on a NFS server on a target's local directory on Linux use the following command

mount -t nfs -o nfsvers=3,nolock <server IP>:/nfs_chimera <target directory>

Remember that the network must be properly functioning between the target and the virtual machine.

7.2.2 Example of mounting MMC card

Use the following command to mount the SD Card

```
mount -t vfat /dev/mmcblk1 /mnt
```

7.2.3 Example of mounting USB stick

A USB stick can be mounted using the following command:

```
mount -t vfat /dev/sda1 /mnt
```

7.2.4 Client FTP

The default package openssh is able to manage sftp server. Connecting to CHIMERA using a FTP client like Filezilla can be done using the following data:

HOST: sftp://<CHIMERA ip addr>
User: root
Password: <the password inserted by passwd command on CHIMERA>

Please remember that a password for root user must be set before proceed, by the command **passwd.**

7.3 Setting of the environment variables

In order to successfully run a QT application it is necessary that it is correctly attached to the own input device which can be a USB mouse, a touch screen resistive or capacitive touch screen. See the next section for how to properly attach the own application at the correct input device.
If application that requires the TSLIB is executed (e.g. the resistive touch screen of our evaluation board), it is necessary to set the following environment variables in the session (shell) from which running the application:
These command are used to attach the Tslib to the correct device's touch screen present on the board. Remember to run the touch screen calibration at the first time the module is powered after the environment variables setting.

```
export TSLIB_CALIBFILE="/etc/pointercal"
export TSLIB_CONFFILE="/etc/ts.conf"
export TSLIB_TSDEVICE="/dev/input/event0"
export QWS_MOUSE_PROTO=tslib:/dev/input/event0
```

Remember that the **/dev/input/event#(0 .. 1 .. 2)** represent the various input devices. Depending on the touch screen used, the device on kernel may be different.

The environmental variables may vary depending on the session. It's possible to check the environment variables using the command that will return the list of variables present.

```
export
```

Always, before performing an application that requires the touch screen, check the presence of the variables described above.
E.g. for the applications that are run automatically when the module starts, these environment variables are set in the script that run the application:

```
/etc/rc.d/rc.local
```

Inside the script file it's possible to see the setting of these variables before launching the executable.

To permanently set the variables in the system shell, and then have them available at every time the module boot, it's possible to put them in an appendix in the configuration file:

/etc/profile

7.4 Execution of a Qt application

To run the application use the command:

/mnt/<application_name> -qws

8. Automatic execution of an application

To automatically run a Qt application is necessary to link the startup script (rc.local) to an external script, or to the application itself inside the directory. It's possible to edit the file from the BSP before generating the image for the target, or do it directly on the target, using the text editor "**vi"**, already pre-installed on the target.

Using the following command to modified the script

```
vi /etc/rc.d/rc.local
```

Remember to add the export commands, which are discussed in the chapter "Setting of the environment variables" remembering that the setting of these will have life only inside the script and not on the entire system.

Remember also that it's possible to launch the application with the suffix &, this will put the execution in the background and the user will still have the access to the system shell.

```
export TSLIB_TSDEVICE=/dev/input/event0
export TSLIB_CALIBFILE=/etc/ts.calib
export TSLIB_CONFFILE=/usr/etc/ts.conf

<App> &
<Qt4App> -qws &
```

8.1 Compiling of the Qt4.8

It's possible to find all the updated informations to re-compile the Qt at the following link:

https://community.freescale.com/docs/DOC-94066

As alternative these are the steps to do for running the recompile. This operation is suggest only for advanced users.

1. Download the git repository for Qt4.8:

```
$ git clone http://git.gitorious.org/qt/qt.git qt
$ cd qt
```

Consider this as <*QTDir*>

2. If not exist, create /*tftpboot* and point your target fileystem. As like

```
$ mkdir -p /tftpboot
$ cd /tftpboot
$ ln -s $(ROOTFFS) rootfs
```

TBD: Need to work on this to use sysroot option

3. Create a build directory to install for the Qt4 packages. This directory can be in any location. For example,

```
$ mkdir /opt/qt4
$ sudo chown -R <username> /opt/qt4
```

Consider the <installdir> as */opt/qt4*

4. Extract the attached mkspecs(*linux-imx6-g++.tar.gz*) to *<QTDir>/mkspecs/qws/*
5. Apply the attached cd *0001-add-i.MX6-EGL-support.patc*h attached to enable egl support for i.MX

```
$ cd <QTDir>
$ patch -p1<0001-add-i.MX6-EGL-support.patch
```

6. Export *CROSS-COMPILE* location path to *PATH*

```
$ export PATH=$PATH:/opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/
```

7. Enter to the <*QTDir*>. Do configure. Select the desired options. Here is an example:

```
$ cd <QTDir>
```

```
$ ./configure -qpa -arch arm -xplatform qws/linux-imx6-g++ -no-largefile -no-accessibility \
-opensource -verbose -system-libpng -system-libjpeg -system-freetype -fast -opengl es2 -egl -confirm-license \
-qt-zlib -qt-libpng -no-webkit -no-multimedia \
-make examples -make demos \
-release -make libs -exceptions -no-qt3support -prefix <installdir>
```

8. When the configure summary is shown make sure the Qt has *OpenGL ES 2.0* support. Do build

```
$ make
$ make install
```

9. Now need to build *eglfs* plugin

```
$ cd <QTDir>/src/plugins/platforms/eglfs
$ make
$ make install
```

Now the eglfs will be installed to the QT Install directory.

10. By now all required QT files are in <install directory>
11. Copy the install directory to target filesystem

```
$ cp -rf /opt/qt4 /tftpboot/rootfs/opt/.
```

12. Running Qt apps on target:

    - Boot the target either with NFS or SD Image
    - Ensure that folder <installdir> is copied on target file system at "/usr/local"
    - Launch application using

```
$ cd /opt/qt4/examples/opengl/hellogl_es2
$ ./hellogl_es2 -platform eglfs
```

9. MQX for Cortex M4 core

9.1 Cortex M4 toolchain

The Cortex M4 toolchain can be downloaded from the following URL:

https://sourcery.mentor.com/GNUToolchain/package10387/public/arm-none-eabi/arm-2012.03-56-arm-none-eabi.bin

The toolchain must be installed with the superuser privileges as follows:

```
$ sudo /bin/sh ./arm-2012.03-56-arm-none-eabi.bin -i console
```

During the interactive toolchain installation some options are required. It is strongly recommended to install the toolchain in the **/opt/arm-2012** directory without creating links.

A toolchain is already installed in VM anyway is strongly recommended to install again it and read carefully the licence agreement.

9.2 MQX installation directory

MQX 4.0.1 source is already installed in the VM in the following directory:

/data/CHIMERA/MQX/

The MQX target twrvf65gs10_m4 is already patched for CHIMERA board. The following changes are applied:

- enabled TTYD in user_config.h in /data/CHIMERA/MQX/config/twrvf65gs10_m4 directory
- enabled UART3 IOMUX in /data/CHIMERA/MQX/mqx/source/bsp/twrvf65gs10_m4/init_gpio.c
- modified /data/CHIMERA/MQX/lib/twrvf65gs10_m4.gcc_cs/release/bsp/ram.ld in order to place DATA after TEXT segment

9.3 MQX build

In order to build MQX please enter the /data/CHIMERA/MQX/build/twrvf65gs10_m4/make directory and run the following command:

```
make TOOLCHAIN_ROOTDIR=/opt/arm-2012/ BOARD=twrvf65gs10_m4 TOOL=gcc_cs CONFIG=release CC=/opt/arm-2012/bin/arm-none-eabi-gcc build
```

9.4 MQX example build

Please enter the /data/CHIMERA/MQX/mqx/examples/hello/ directory and open the hello.c file. This is a very simple program to print "Hello world" on stdout every 500 milliseconds.

This file can be compiled entering in the following directory:

/data/CHIMERA/MQX/mqx/examples/hello/make

and run the following command:

```
make TOOLCHAIN_ROOTDIR=/opt/arm-2012/ BOARD=twrvf65gs10_m4 TOOL=gcc_cs CONFIG=release CC=/opt/arm-2012/bin/arm-none-eabi-gcc LOAD=ram build
```

Then the .elf file can be converted using:

```
/opt/arm-2012/bin/arm-none-eabi-objcopy -O binary twrvf65gs10_m4.gcc_cs/release_ram/hello.elf
twrvf65gs10_m4.gcc_cs/release_ram/hello.bin
```

The program entry point can be found by the following command:

```
/opt/arm-2012/bin/arm-none-eabi-readelf twrvf65gs10_m4.gcc_cs/release_ram/hello.elf -h
```

The hello.bin file can be copied to CHIMERA target and loaded in RAM using the mqxboot command:

```
# modprobe mcc.ko
# mqxboot hello.bin   0x3f000000 0x3f002d85
Loading hello.bin to 0x3f000000 ...
....
Loaded 34536 bytes. Booting at 0x3f002d85...
done
```

Now the "Hello world" output can be verified on J30 connector of the CHIMERA starter kit. The J30 serial port is configured 11500,8,N,1.

10. Appendix

This section of the manual contains some additional content on the module and how to use some software packages useful for development.

10.1 Use of TFTP

On the module is possible to reprogram the kernel via TFTP using the script ker.scr already pre-compiled. This option can be very useful working on the kernel as it speeds up the programming process than using SD card.
To properly configure the module to work via ftp refer to the chapter (Running Scripts From Flash Chapter 3.5) about the network settings. Once the system is been setup, load the pre-configured ker.scr script:

```
tftp ker.scr
```

then run it from the U-Boot prompt:

```
so
```

10.2 How to generate a file.scr

To run the **mk** script install the uImage packet with command:

```
sudo apt-get install uboot-mkimage
```

The script files have ran from the U-Boot that run operations in sequential mode as indicated into the script file.
The script can be used to do several operations including, download files, fill a memory area, set up of the environmental variables and much more.

The file script must be formatted in a standard mode to be read from the boot.
Please refer to the script available inside the default module's images, the file .scr are formatted for U-Boot while the file .txt are used to the changes.

To make a script, a file.txt must be created with the operations specified inside. It's possible to edit the file.txt already available in the default images' folder. After that copy the file .txt in the following directory:

```
/data/CHIMERA/custom_script/make_progscr
```

then run the command:

```
./mkscript <file_name> without extension
```

In this way it's possible to obtain an own script <file_name>.scr that can be executed in U-Boot (see dedicate chapter in U-Boot section).

10.3 MAC address

Each Chimera module has a serial number uniquely associated with an own MAC Address and regularly re-served. The MAC address is written in U-Boot environment variable:

```
printenv ethaddr
ethaddr=11:22:33:44:55:66
```

This value is automatically passed to Linux kernel using the Linux cmdline fec_mac argument.

10.4 TSLIB Calibration of touchscreen

For the calibration of touch screen, the ts_calibration utility included in Tslib packet is used.
Before launching the calibration of the touch, remember to export the environment variables of the touchscreen (described in Setting environment variables), after that run the command and execute the instructions on the device's display:

```
./ts_calibrate
```

If you miss the ts.conf file this is an example of how to edit a new one.

```
# Uncomment if you wish to use the linux input layer event interface
module_raw input grab_events=1

# Uncomment if you're using a Sharp Zaurus SL-5500/SL-5000d
# module_raw collie

# Uncomment if you're using a Sharp Zaurus SL-C700/C750/C760/C860
# module_raw corgi

# Uncomment if you're using a device with a UCB1200/1300/1400 TS interface
# module_raw ucb1x00

# Uncomment if you're using an HP iPaq h3600 or similar
# module_raw h3600

# Uncomment if you're using a Hitachi Webpad
# module_raw mk712

# Uncomment if you're using an IBM Arctic II
# module_raw arctic2

module pthres pmin=1
module variance delta=30
module dejitter delta=100
module linear
```

10.5 Passwords and users

Default user and password for console:

```
user:       root
password:
```

default user and password for virtual machine:

```
user:       user
password:   user
```

root password (for the administrator privileges): ***user***

10.6 Install NFS Server in Ubuntu

Inside the ENGICAM virtual machine, the folder "\nfs_icore" is already configured by default to be used as NFS. Following is shown the steps for installing and configuring a NFS server on Ubuntu.

Install the NFS server with the command line:

```
apt-get install nfs-kernel-server
```

Edit the server configuration:

```
sudo gedit /etc/exports
```

Add the line (example with the nfs_icore folder):

```
/nfs_gea  *(rw,sync,no_subtree_check)
```

Restart the server:

```
sudo /etc/init.d/nfs-kernel-server restart
```

10.7 How to use a GPIO

Following is shown how to move the GPIO pin using the Kernel or the user space. In both cases, the pins of the GPIO must be initialized and the IOMUX and pad must be configured.

10.7.1 IOMUX configuration

Each pin of the processor can be configured and connected to various peripherals. The IOMUX takes care of set each pin on a device.
To find what function may have a particular pin and what values must be set on the IOMUX is necessary to identify through our hardware manual the name given to the PIN from Freescale. Once located it, open the Freescale reference manual and in the section IOMUX find the selected pin. This section shows all the functions available for each MUX value.

To set this value the following file must be edited:

/data/CHIMERA/kernel-chimera/arch/arm/mach-mvf/board-twr-vf700.c

In the file there will be two initialization tables for each pin where the GPIO to move must be added to set it up with the right value of MUXING.

10.7.2 Use GPIO from user space

After configuring the IOMUX and loaded the modified Kernel, the support for sys/class/gpio must be added. Enter in the configuration tool of the Kernel and set:

```
Device Drivers --->
   [*] /sys/class/gpio... (sysfs interface)
```

Now calculate the number of GPIO that to move using the following formula:

GPIO6_3, which corresponds to GPIO number [(6-1)*32 + 3] = 163.

Once this is done set the GPIO as output and then move it

```
echo 163    > /sys/class/gpio/export                 Export GPIO163 to user space
echo out    > /sys/class/gpio/gpio163/direction      Set the GPIO as output
echo 1      > /sys/class/gpio/gpio163/value          Set the GPIO to 1
echo 0      > /sys/class/gpio/gpio163/value          Set the GPIO to 0
```

Or set the GPIO as input and read its value

```
echo 163    > /sys/class/gpio/export                 Export GPIO163 to user space
echo in     > /sys/class/gpio/gpio163/direction      Set the GPIO ad input
Cat         > /sys/class/gpio/gpio163/value          reading value
```

10.7.3 Use GPIO from Kernel space

To move a GPIO from Kernel space is always necessary to configure the IOMUX in the tables as written in the previous chapter. At the same time it's possible to run the release of the resource in the same function.

10.8 How to use the CAN BUS

To use the CAN Bus on the Chimera modules, it is necessary the **iproute** and the **cantest** package on the **ltib**. Following the instructions to enable the can and some example to how to use:

1) Configure the bit rate on target:

```
/ # ip link set can0 type can bitrate 125000
```

2) Enable the interface on target:

```
/ # ifconfig can0 up
```

3) To send a frame:

```
/ # cantest can0 5A1#11.2233.44556677.88
```

4) To receive a frame:

```
/ # cantest can0
```

10.9 How to modify the Kernel splashscreen

This procedure is used to modify the graphic image of the kernel which appears when the machine is started. The image must be converted in .ppm format using GIMP, which is free downloadable from the internet (the use of GIMP windows version is recommended). After an image of the exact size of the screen is been created, follow these steps to convert it into a readable format. The images larger than the screen resolution will not be displayed.

- **Create an image of the desired size (e.g. 800x480 for 7" display)**
- **Open the image using GIMP**
- **Right-click on the image**
  Image->Mode->indexed (if already indexed, move it on RGB)
  Then, put it again on Indexed to open the options window.
- **Select: Generate optimum palette, maximum number of colors 220**
- **Save the picture in ppm**
- **Select as ASCII**
- **Rename the image as "logo_linux_clut224.ppm"**

- **Enter into the folder $ LINUX_SRC_DIR / drivers / video / logo**
- **Replace the old file with this one**
- **Delete the file logo_linux_clut224.c logo_linux_clut224.o**

10.10 Installation and configuration of the atftp server

Even this atftp server should already be configured and pre-installed inside the virtual machine, however the steps for installation and configuration are listed:

10.10.1 Install atftp Server in Ubuntu

Install the atftp server typing:

sudo aptitude install atftpd

This will complete the installation.

By default atftpd server starts using INETD so we need to tell atftpd to run as a server directly, not through inetd.Edit /etc/default/atftpd file using the following command:

sudo gedit /etc/default/atftpd

Change the following line:

USE_INETD=true

with

USE_INETD=false

save and exit the file.

Now run the following command:

sudo invoke-rc.d atftpd start

10.10.2 Configuring atftpd

First it's necessary to create a directory where place the files

```
sudo mkdir /tftpboot
sudo chmod -R 777 /tftpboot
sudo chown -R nobody /tftpboot
sudo /etc/init.d/atftpd restart
```

Some level of security can be gained using atftp libwrap support. Adding proper entry to /etc/hosts.allow and /etc/hosts.deny will restrict access to trusted hosts. Daemon name to use in these files is in.tftpd.

/etc/hosts.allow /etc/hosts.deny

in.tftpd: FQD or IP

10.10.3 Atftp client installation

Advance Trivial file transfer protocol client, atftp is the user interface to the Internet ATFTP (Advanced Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote host may be specified on the command line, in which case atftp uses host as the default host for future transfers.

```
sudo aptitude install atftp
```

That's all now it's possible to transfer the files using tftp clients

10.10.4 Ttftp client

An alternative to atftp, for windows system, is the TFTP. The Trivial File Transfer Protocol allows users to transfer files from a remote machine simply run the file .exe, tftpd32.exe. For the use of Tftp refer to the specific chapter.

It's strongly recommended to disable the firewalls before running the program.

10.10.5 Testing tftp server

Transferring file hda.txt from 192.168.1.100 (Client using tftp) to 192.168.1.2 (Server 192.168.1.100). Get an example file to transfer (eg. hda.txt)

```
touch /tftpboot/hda.txt
chmod 777 /tftpboot/hda.txt
ls -l /tftpboot/
total 0
-rwxrwxrwx 1 ruchi ruchi 223 hda.txt
atftp 192.168.1.2
atftp> put hda.txt
Sent 722 bytes in 0.0 seconds
atftp> quit
ls -l /tftpboot/
```

11. Vmware Player sample user guide

During the first installation of the virtual machine some problems with the VMPlayer could be appear. In this chapter is given some recommendations to facilitate this work.
To work properly with the virtual machine the user should be able to

- Access the network
- Read and write a SD card
- Copy and move files

11.1 Use of the last VMware Player version

From the player version 5.0 have been were introduced substantial changes and improvements in the detection of devices between the physical machine and the virtual machine. ENGICAM suggests to use always the last version or a version after the 5.

11.2 Installing the latest version of VMware Tools

Re-install the VMware Tools to the latest version, follow the instructions and once the installation is finished restart the virtual machine

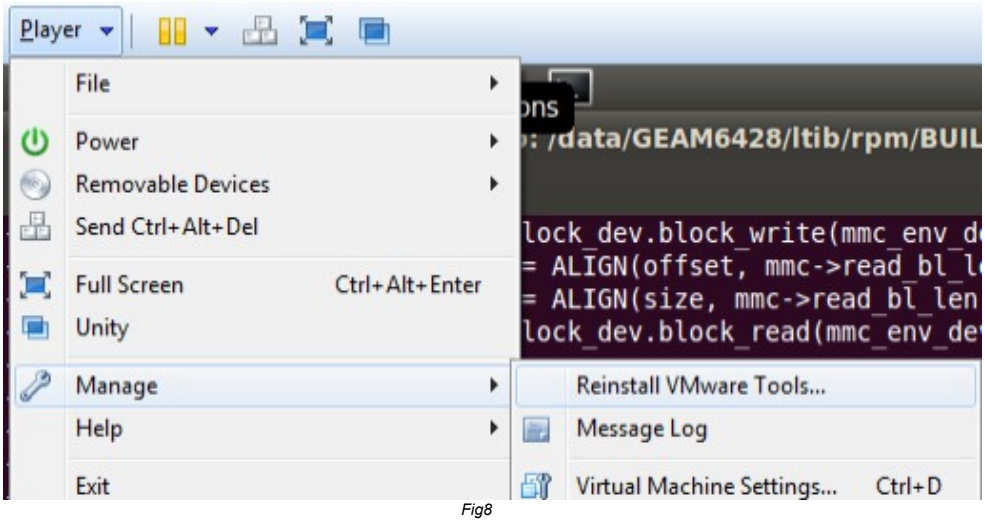


Fig8

11.3 Network troubleshooting

The virtual machine can access to the network and communicate with the module, therefore via virtual machine it's possible to:

- ping from the virtual machine to the PC and vice versa
- ping from the virtual machine to another address on the Net
- ping from the virtual machine to the module and vice versa

For problems of communications it's possible to have the following solutions:

- Disable any WiFi device before turning on the virtual machine
- Do not connect the PC to a point to point network. Each time the connection is shut down, the PC must be restarted, since the virtual network between the physical machine and VMplayer will lose the connection.
- Disable any anti-virus and firewall

For any problems on network, always reboot the PC. The default network configurations is the following:

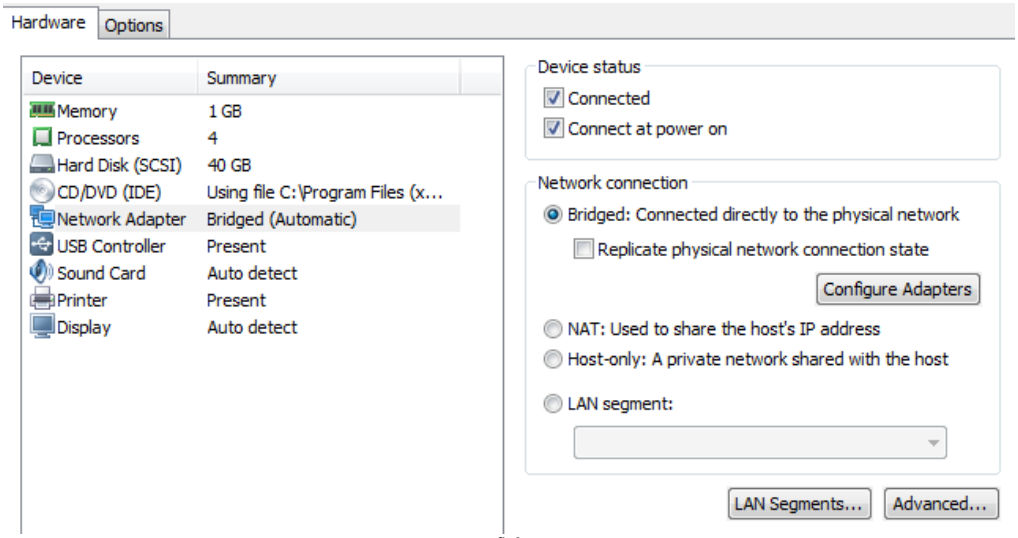


fig9

Set the IP Address manually or via DHCP. Verify the network connection using ifconfig command and check the icon in the top right position as shown in figure below.

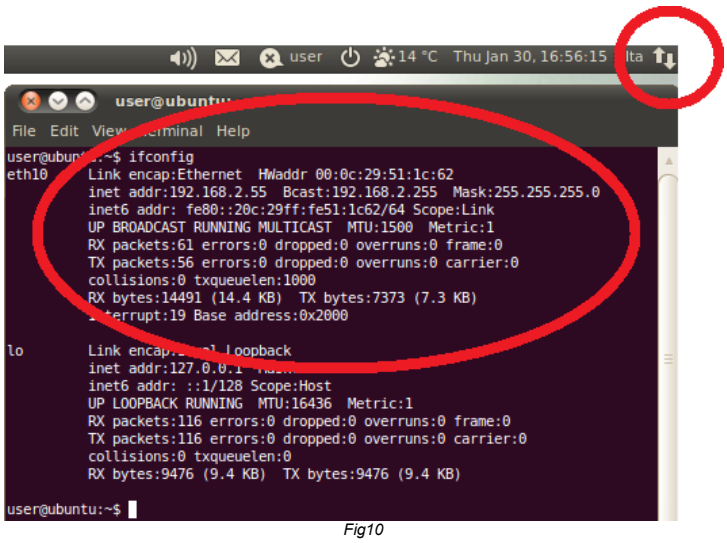


Fig10

11.4 Troubleshooting SD

The write operation of an SD card can create some problems for the correct writing of the files. Sometimes there may be problems to recognize the micro SD to the virtual machine that contains the linux BSP. In case of trouble, try the following steps.
Some problems associated with the use of SD card readers may occur on the physical bus of the machine. In case of problems it is recommended to use USB SD readers who are generally more compatible with VMPlayer than those on the bus.

11.4.1 Verify that the physical machine is able to see the driver

Sometimes there may be also physical problems with the SD reader. If you connect the drive to the Windows machine having the virtual machine closed and the card is formatted in ext3 you should see the following message:

fig11

If the message does not appear, there may be hardware problems on the SD reader, or driver problems with Windows. If the SD card is formatted as FAT32 or with another type of Windows capable filesystem, it 'll be open a normal Windows' folder.

11.4.2 VMPlayer status icons

If the USB driver is not recognized correctly appear on the player VMPlayer the following icons:

- External disk Icon
- Driver connect: if connected will be flagged otherwise press connect

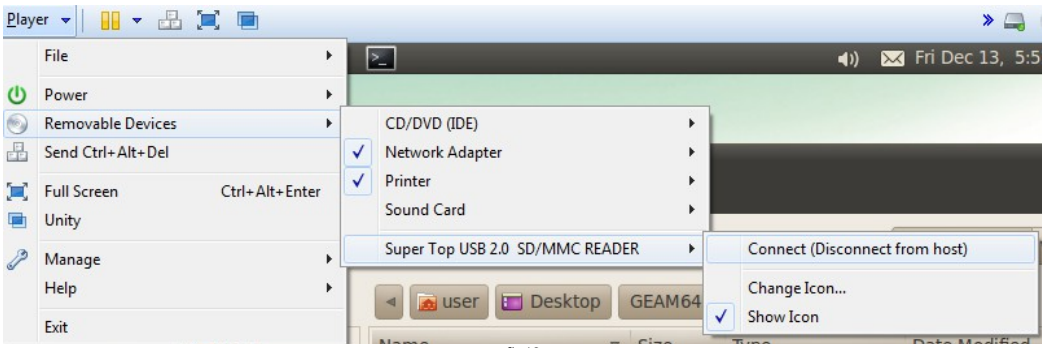


fig12

11.4.3 Verify the device presence on the virtual machine

If the device is properly connected you will see the resource available and access to its content by navigating within it through the shell by entering the path /media/rootfs

fig13

In the event that there is status icon of the VMPlayer, but there is no access to the disk of the virtual machine. Restart the virtual machine and the physical machine and try again. It is recommended to connect the SD card when the focus on the virtual machine is active.

11.4.4 Safe disconnection of the SD card

Once the use of the SD card is ended is recommended to ALWAYS give the sync command to synchronize the files and then safely remove the card using the **umount** command form shell or clicking the button **exject** which is also seen in the previous figure.

11.4.5 General note

Sometimes it may also happen that, with the continued use, the wrong removals, especially when used on the target, the SD card may be damaged and no more readable. In these cases, it's only the partition that was damaged and may need to reformat the partition, following the procedure described in this manual.

11.5 Copy files between the virtual machine and the physical machine

You should be able to copy and move files from the virtual machine using copy and paste or using drag-and-drop mode from the virtual desktop to the real one. If it is not possible to perform all these operations, it depends on the correct installation of VMware Tools.

12. Technical support

Engicam provides to its customers the latest versions of its BSP of development. In the time between one version and the other a few changes or improvements can be published and shared via patch or textual documents.
Every customer that bought a BSP has at its disposal a virtual account in which find all this informations.
In case you do not have a FTP account, contact your own dealer to obtain the access.

Engicam recommends to always refer to the latest files of your FTP account. Every new release of a BSP will contain all the patches and improvements described and released during the time between the release of a BSP and the other. For this reason, older versions will gradually be removed from the account. We recommend to keep aligned the own material with respect to the official releases of a BSP.

12.1 Ftp account structure

- **Virtual Machine**
  It can be in the form of archive or Windows installer

- **PCN Folder**
  Containing all the hardware and software PCN (Product Change Notification) of the Module

- **Doc Folder**
  Containing the documents of both hardware and software up to date with all the manuals of the module.

- **Patch Folder**
  Divided between boot and kernel patch contains all the patches for some bug fixes or improvements introduced. The patch.description file contains the description of each patch.

12.2 Upgrading the BSP using patch

Once started the development and customization of the kernel, it's possible to keep it up to date through the use of patches that are on your account. This chapter gives some information on how to manage and update the BSP. Once achieved a stable version of the system it is advisable to only apply patches that may affect the own application.

12.2.1 Structure of the patch folder

This folder contains any patches of the downloadable version. The patches have the following order:

PATCH_*KERNEL/BOOT*_**BSP**_*X.X*

In the name it's specified "KERNEL or BOOT", where the patch must be applied and the BSP revision "X.X"

Inside the folder it's possible to find the main patches that have the following method:

*00X*_**main_patch**_*YYMMDD*.**patch**

main patch at the date YYMMDD that aligns the whole kernel with our mainline.

It's possible to find the singular patch with the following method:

*00X.0Y_argument_name***.patch**

where "00X" specified the membership to the main patch, "0Y" is an incremental number, the "argument_name" is the name of the fixed problem by the patch.
These Patches solves the individual problem and are usually used by the customer who has already customized the kernel so that is not necessary having to apply the main patch.

12.2.2 Patch structure

A patch consists of several sections of code to add or remove, these sections are localized reporting the previous and subsequent lines of code where to edit the changes.

```
1 diff --git a/arch/arm/mach-mx6/Kconfig b/arch/arm/mach-mx6/Kconfig
2 index 0c6e89e..db1a58b 100644
3 --- a/arch/arm/mach-mx6/Kconfig
4 +++ b/arch/arm/mach-mx6/Kconfig
5 @@ -222,6 +222,14 @@ config MACH_MX6Q_MINIMUM_FREQ400
6           This features set the minimum CPU clock frequency to 400 Mhz instead of 200 Mhz.
7           Recommended option for the use of video codecs.
8
9 +config MACH_MX6Q_ICORE_OPENFRAME_RESISTIVE
0 +       bool "Support i.CoreM6 resistive OpenFrame"
1 +       depends on MACH_MX6Q_ICORE
2 +       help
3 +         Include the support for Engicam openframe. This features enabled
4 +         the correct power up and power down of the on-board LVDS controller.
5 +          This features is mandatory.
6 +
7  config MACH_MX6Q_ICORE_STARTERKIT_CAP_EDT
8         bool "Support i.Core capacitive starterkit"
9         depends on MACH_MX6Q_ICORE
```

fig14

Diff —- indicates the file and the location you want to edit.

@ indicates the code lines where edit the modifies.
Following are shown the lines of code that should not be changed.
With **+** or **–** are shown the lines to add or remove to edit the file. The remaining code is used to identify where to apply the patch

12.2.3 How to apply the patch

A patch is applied if the command patch can correctly identify where to insert the code parts. Otherwise, it generates an error file and the patch must be changed manually.
In case that the user have customized the code, he can follow the structure of the patch and apply it manually to avoid errors in editing. Refer to the previous chapter for a description of the structure of the patch. To apply a patch follow the below procedures. Enter the folder to edit (U-Boot or kernel). Copy the patch inside the folder:

```
cd linux
```

with the dry-run command try to apply a patch and see if it returns an error to evaluate whether it is possible to apply the patch.

```
patch -p1 --dry-run < 002.02_iCoreM6_openframe_lvds.patch
```

Once tested the application, if there are not too many mistakes, it's possible to apply the patch using the command:

```
patch -p1 < 002.02_iCoreM6_openframe_lvds.patch
```

If the application is successful, rebuild the kernel and update it on the device. In the case of errors an output like this will appear:

```
patching file arch/arm/mach-mx6/Kconfig
Hunk #1 FAILED at 222.
1 out of 1 hunk FAILED -- saving rejects to file arch/arm/mach-mx6/Kconfig.rej
patching file arch/arm/mach-mx6/board-mx6q_icore.c
Hunk #1 FAILED at 107.
Hunk #2 succeeded at 264 (offset -6 lines).
Hunk #3 succeeded at 417 (offset -8 lines).
Hunk #4 succeeded at 1427 with fuzz 2 (offset -19 lines).
Hunk #5 succeeded at 1629 (offset -33 lines).
1 out of 5 hunks FAILED -- saving rejects to file arch/arm/mach-mx6/board-mx6q_icore.c.rej
```

In the above example the patch failed the application of the Kconfig file and board-mx6q_icore.c.
The failed parts are available in the file .rej that are generated by the patch command. Then open the file and manually enter the parts not included.

13. Technical support

For help, write an email to:

support@engicam.com

14. Useful links

http://www.imxdev.org/

http://www.imxcommunity.org/

http://www.freescale.com/

http://www.engicam.com/