# Lab – IPC Lab

## Projects and source code

The original files for this Lab are part of the MCSDK release. The student will copy MCSDK release into his private directory (studentN/MCSDK\_X\_XX) before changing any file.

## Purpose

The purpose of this lab is to demonstrate messages transfer between the ARM and the DSP cores. The source code may be a starting point for customer who needs sending messages and data between cores.

# Task 1: Run the demo from a Web Server

## Step 1

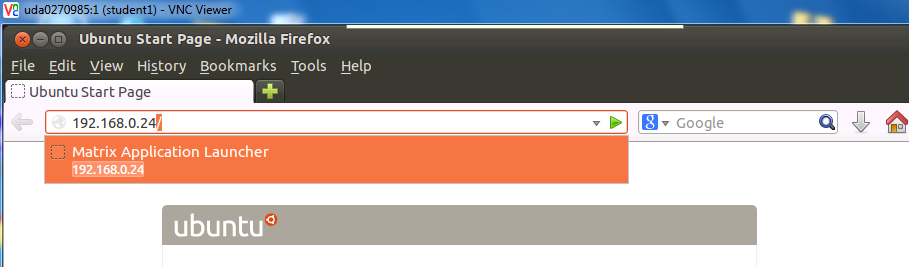
The file system that is loaded into the EVM should be tisdk based file system. Boot the EVM using NFS (mount) boot and wait until the display on the EVM gives the IP address of the board. Note that the display flips between several messages. IP address is one of the messages.

## Step 2

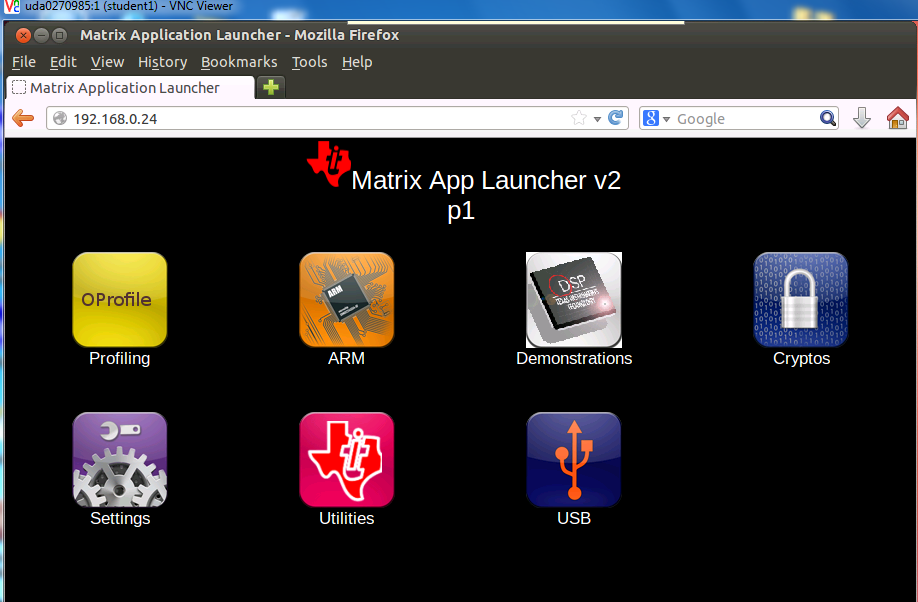
Open a terminal into the EVM and log in as root

## Step 3

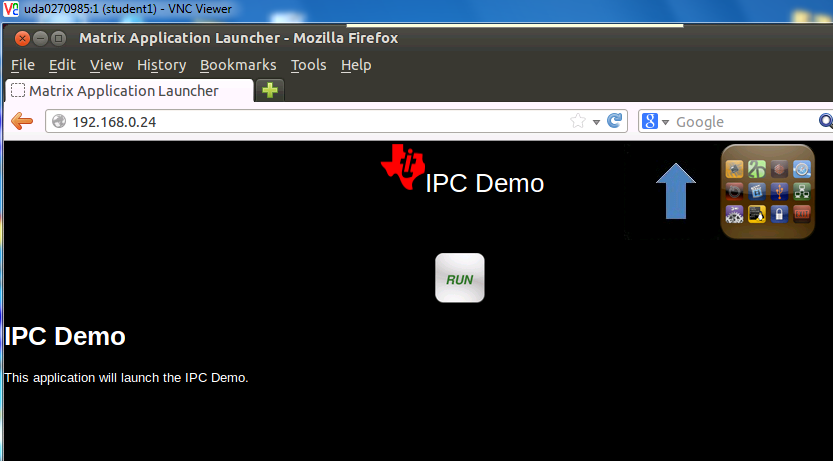
From a computer that is on the same sub-net as the EVM, or from vnc into a computer that is on the same subnet as the EVM start Firefox (or any other browser) and put the ip address of the EVM as shown in the following screen shot. The EVM in the screen shot has ip address of 192.168.0.24.



If the Firefox is connected via vnc, Ubuntu may ask you if you have a display device. Answer OK. The EVM respond with a set of out-of-the-box applications as seen in the next screen:

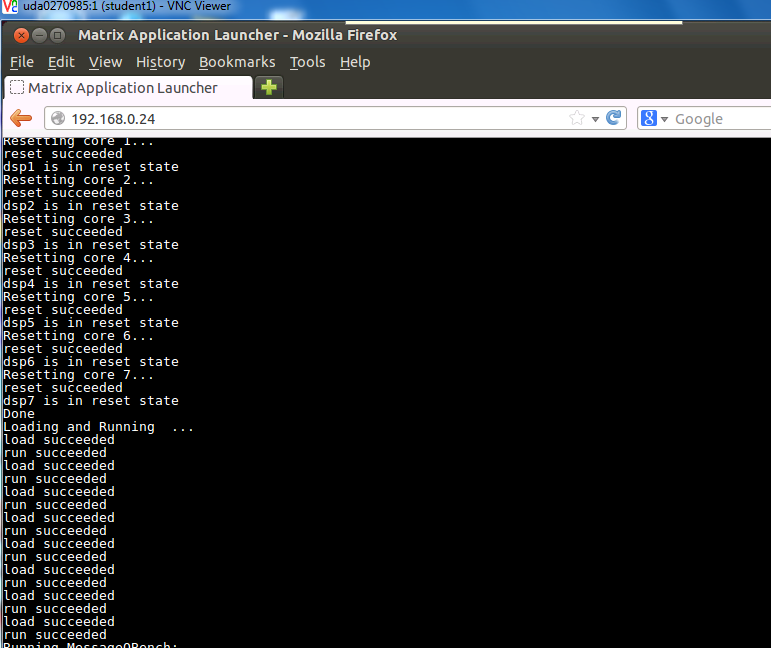


It is highly recommend going through each one of the applications, however, in this Lab we only use the IPC demo. Click on the Demonstrations tab and then the IPC Demo. The next screen shot will be displayed:

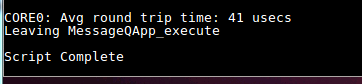


Click on RUN and follow the progress on the browser and on the terminal that is connected to the EVM.

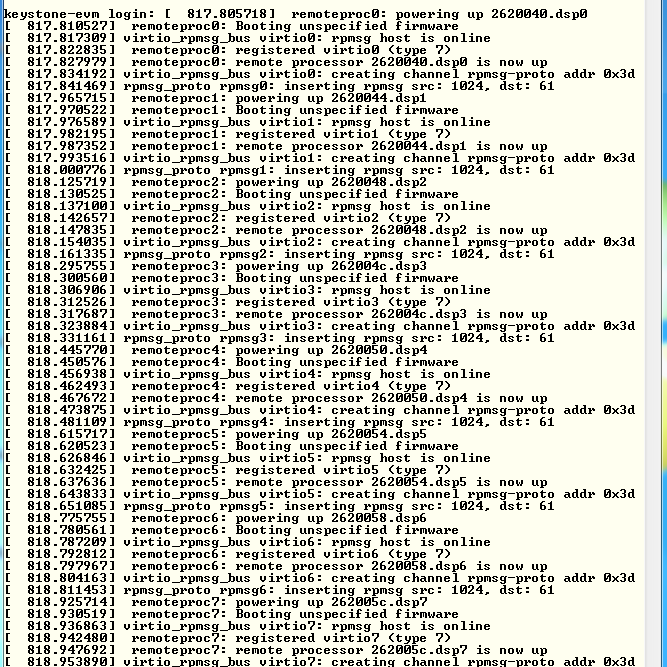
The browser display



You may have to use the arrow to see the complete execution of the demo. The last lines are the following:



The terminal that is connected to the EVM displays something like:



Follow the messages on the terminal and see what software modules are used (remoteproc, virtio, rpmsg)

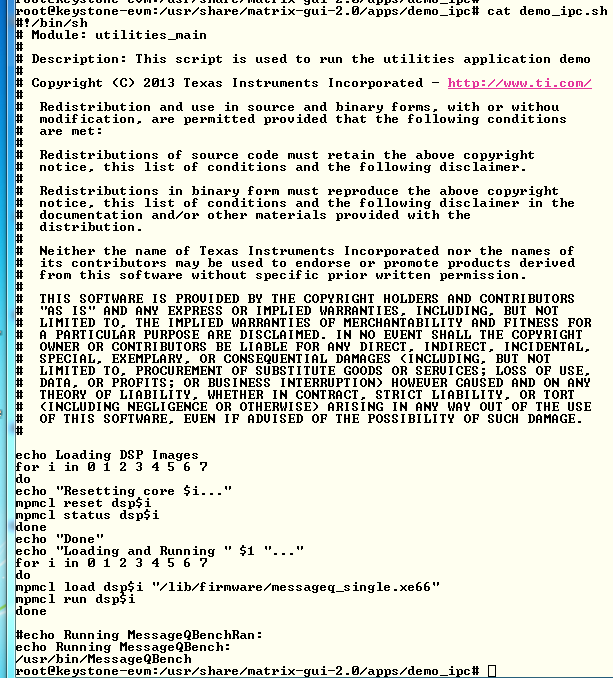
# Task 2: Run the demo from the terminal

## Purpose

The purpose of this task is to familiar the user with the directory structure and the main files of the demo

## Step 1 – Understand the file demo\_ipc.sh

In the directory /usr/share/matrix-gui-2.0/apps/demo\_ipc there are three files demo\_ipc.desktop, demo\_ipc.sh, and desc\_demo\_ipc.html. Look at the file demo\_ipc.sh using either vi, more, less, cat or any other utility. The screen shot was taken using cat:



Notice that the DSP code is loaded from directory /lib/firmware and the name of the execution is message\_single.xe66. The Linux code is loads from directory /usr/bin and the executable name is MessageQBench. In the next task you will build these two executable files in your local directory and move them to your private filesystem

## Step 2 – Run the file demo\_ipc.sh

In the terminal, move to directory /usr/share/matrix-gui-2.0/apps/demo\_ipc (cd /usr/share/matrix-gui-2.0/apps/demo\_ipc ) and run the sh file ./demo\_ipc.sh as in the following screen shots:



After the run the terminal looks like the following:



# Task 3: rebuild the executable

## Purpose

In this task the student will re-build the DSP and ARM executable to prepare for modifying the code for a new project

## Step 1 – Get a private copy of IPC from the release

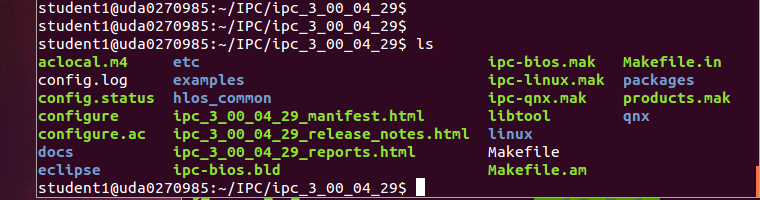
In your home directory /home/studentN create a new directory named IPC and move there

* sudo mkdir IPC
* cd IPC

Copy the ipc directory from the latest release into the new created private directory. Currently the latest release is release 3\_15

* sudo cp -R /opt/ti/MCSDK\_3\_15/ipc\_3\_00\_04\_29 / .
* cd ipc\_3\_00\_04\_28
* ls

The directory should look like the following:



## Step 2 – Build the ARM executable

Detailed instructions how to install and build the Linux version of IPC are in the file IPC\_Install\_Guide\_Linux.pdf that is part of the release in directory \MCSDK\_3\_15\ipc\_3\_00\_04\_29\docs. The install part of the IPC is already in the release, so we will start with the build procedure.

First a set of environment variables should be set in the file products.mak in the directory ipc\_3\_00\_04\_29 (or any later version of the IPC). The following is a list of environment variables that need to be updated:

TOOLCHAIN\_INSTALL\_DIR

TOOLCHAIN\_LONGNAME

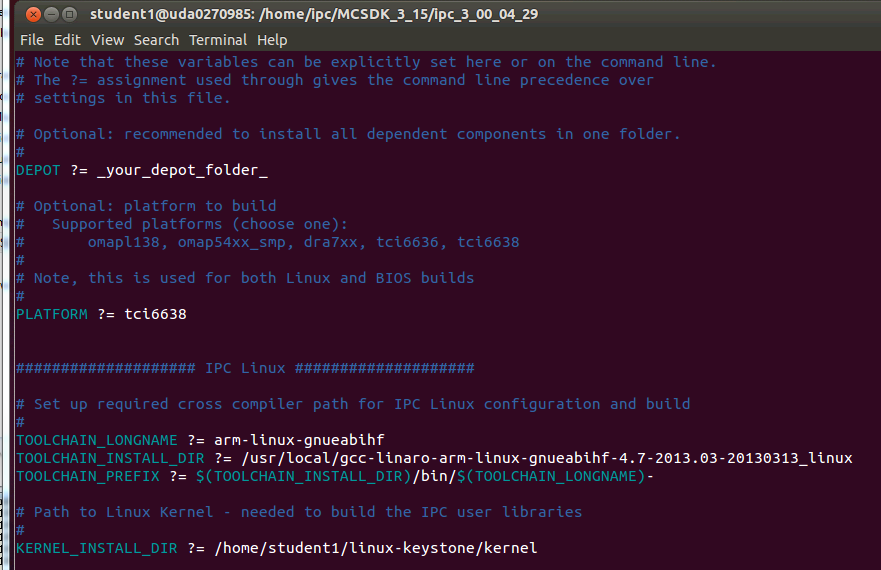
PLATFORM

XDC\_INSTALL\_DIR

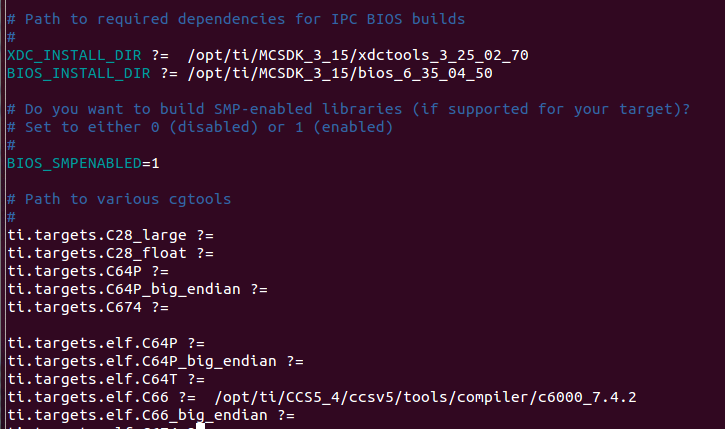
BIOS\_INSTALL\_DIR

ti.targets

The user guide from above explains what needs to be set in the variables that are mentioned above. The following is an example of the definition. You may have to modify the variables based on structure of your system



The platform is tci6638 (Hawing). The TOOLCHAIN\_INSTALL\_DIR is the location where the Linaro tools were installed on the server. The KERNEL\_INSTALL\_DIR is where the kernel sources were installed using the git repository. Instructions how to install the kernel sources using the git repository are given in Lab 2 task 2 of the KeyStone II Multicore Workshop Arm-Based Lab manual.



XDC)INSTALL\_DIR is part of the MCSDK release and the location points to where MCSDK was installed. The same is true for the BIOS\_INSTALL\_DIR. The ti.targets.elf.c66 is the location of the code generating tools for this platform. The tools location is part of the CCS release. The CCS was installed in /opt/ti /CCS5\_4 directory.

In order to manipulate files in the ipc directory you have to change the permission. Since this is a private copy on a local network, you can give full permission

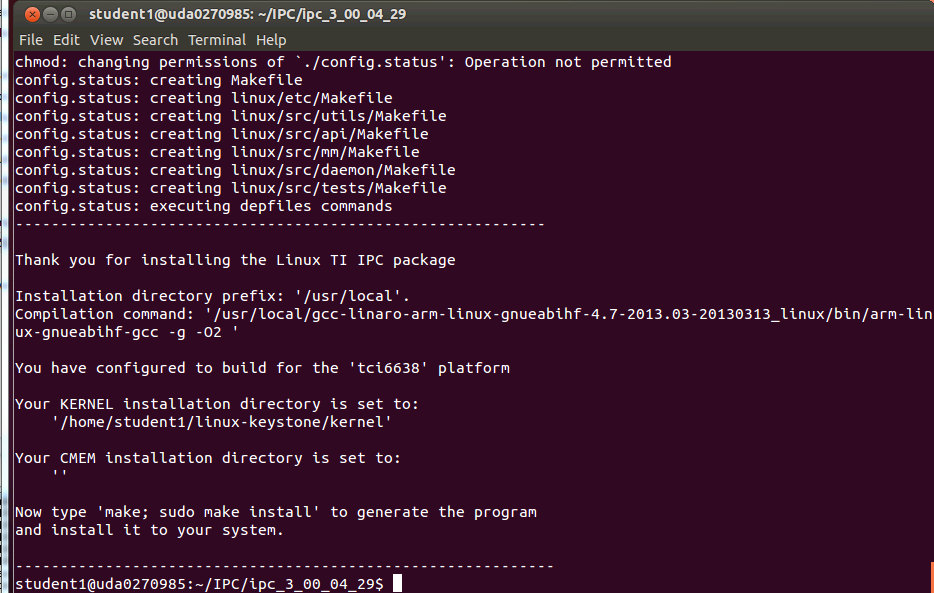
* cd ..
* sudo chmod 777 –R ipc\_3\_00\_04\_29
* cd ipc\_3\_00\_04\_29

The first step is cleaning the older build:

* sudo make clean

Next step is to run the make utility with the linux makefile:

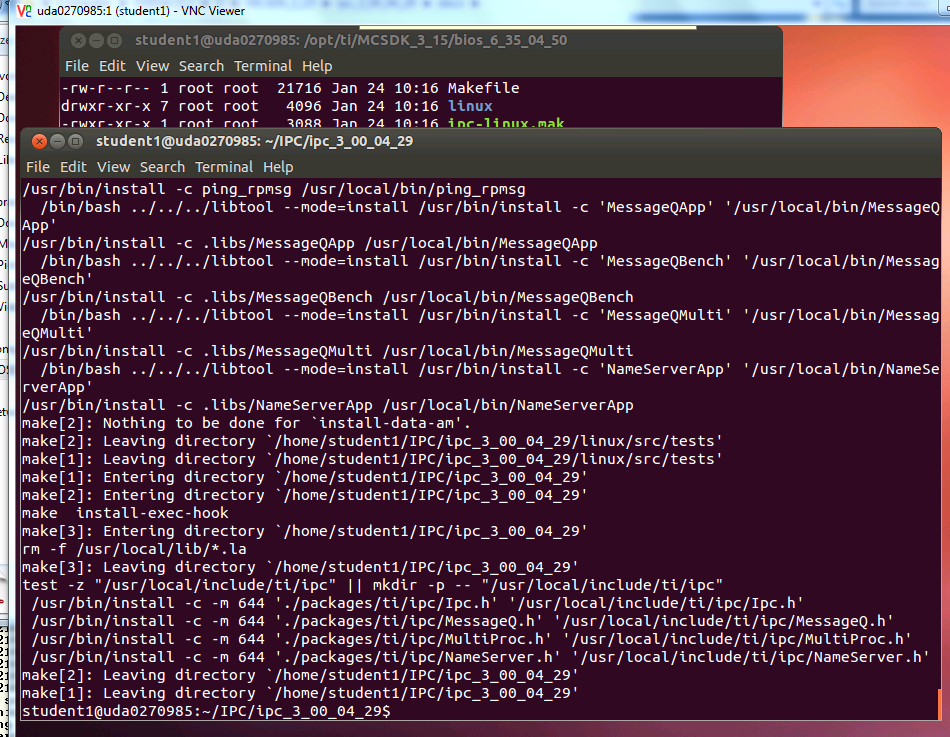
* make –f ipc-linux.mak config



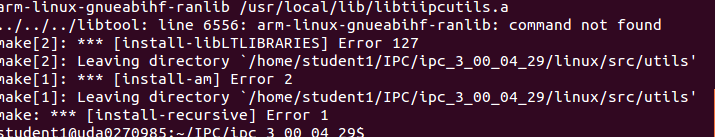
The screen prompts you to run the next make

* make ; sudo make install

Follow the printing in the window. It may ask you for a password for the sudo part. The following screen shot shows the end of the build. In case of error, see the note after the screen shot.



Note – some of the libraries may not be rebuilt, but the output files are built. The screen shot may be like the following:



The build process builds several files. The file MessageQBench in directory linux/src/tests/ is a temporary wrapper script file that shows how the build is done. The file MessageQBench (yes, the same name) in directory linux/src/tests/.libs is the executable. After the build move this file to the file system location

* sudo cp linux/tests/.libs/MessageQBench /opt/filesys/studentN/mcsdk\_x\_xx/usr/bin/.

Verify that the executable in the /opt/filesys/studentN/mcsdk\_x\_XX/lib/usr/bin directory is the one that you built by doing ls –ltr MessageQBench and verify the data and time

## Step 3 – Build the DSP executable

Detailed instructions how to install and build the DSP-BIOS version of IPC are in the file IPC\_Install\_Guide\_BIOS.pdf that is part of the release in directory \MCSDK\_3\_15\ipc\_3\_00\_04\_29\docs. The install part of the IPC is already in the release, so we will start with the build procedure.

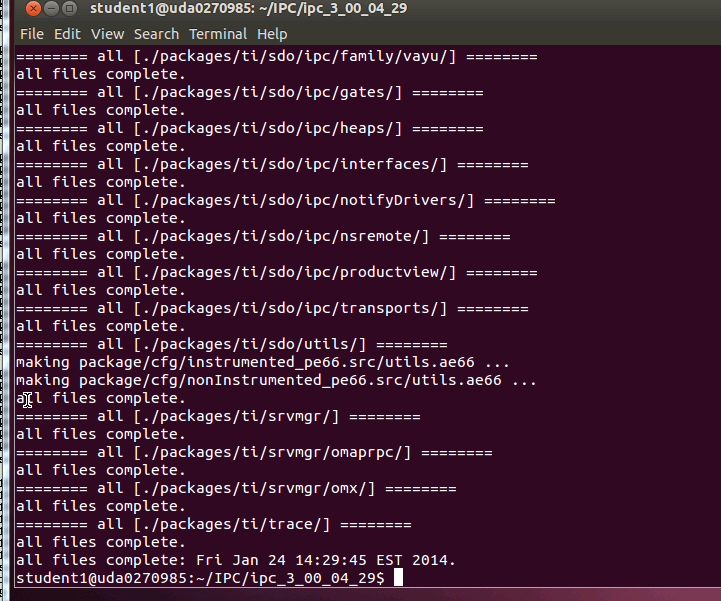
The environment variables in the file products.mak were already set in the previous step.

Return back to the ipc directory

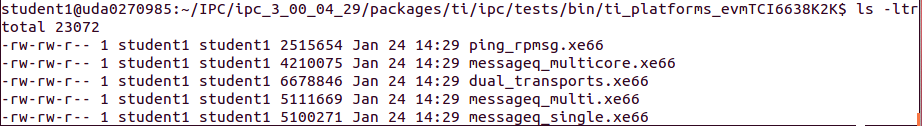
Next step is to run the make utility with the linux makefile:

* cd ~/IPC/ipc\_3\_00\_04\_29 (or a newer version of IPC)
* make –f ipc-bios.mak all

The build will take several minutes. The following is a screen shot when the build is done:



The executables are built in the directory IPC/ipc\_3\_00\_04\_29/packages/ti/ipc/tests/bin/ti\_platforms\_evmTCI6638K2K. There are 5 executables as follows:



The file messageQ\_signal.xe66 should be copied to the file server at location /opt/filesys/studentN/mcsdk\_x\_XX/lib/firmware

* sudo cp packages/ti/ipc/tests/bin/ti\_platforms\_evmTCI6638K2K/messageq\_single.xe66 /opt/filesys/student1/mcsdk\_3\_14/lib/firmware/.

Verify that the executable in the /opt/filesys/studentN/mcsdk\_x\_XX/lib/firmware directory is the one that you built by doing ls –ltr messageQ\_single.xe66 and verify the data and time

# Task 4 – Optional: Modify the source code and rebuild the executable

## Purpose

In this task the student will change the ARM code and the DSP code, build and run the executable

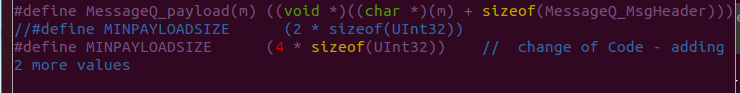
## Step 1 – Modify the ARM code

The demo source file is MessageQBench.c in directory /IPC/ipc\_3\_00\_04\_29/linux/src/tests. Following the source file, it looks like the ARM allocates a message, sends it to the DSP, and then gets the message back and make sure that the data was not corrupted during the transfer. The changes that are suggested for the ARM side are the following:

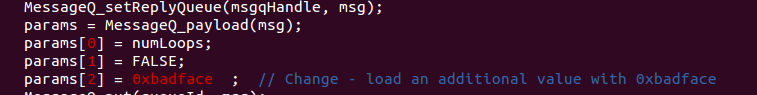
1. Modify a printf statement to show that the code has been modified
2. Adding an additional value (or values) to the message, load the additional value with a known data, and print the additional information from the DSP side

Here are the suggested changes:

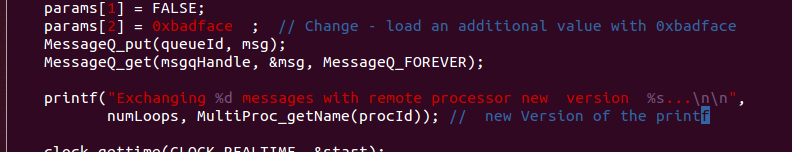
1. Add two more values to the message:



1. Load the additional value with whatever you wish, in the example it is 0xbadface



1. Modify the print statement



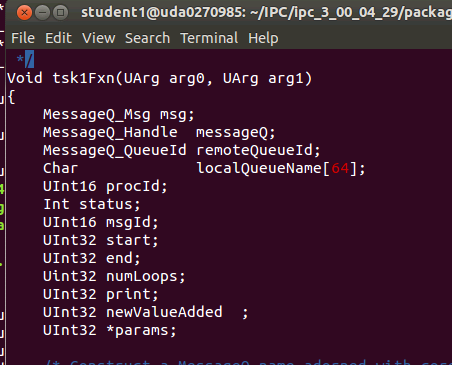
## Step 2 – Build the ARM code

Repeat the steps from the previous task and re-build the ARM executable MessageQBench

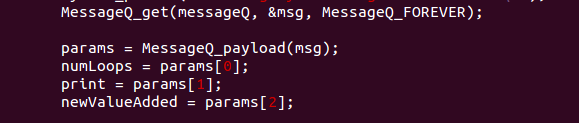
## Step 3 – Modify the DSP code

The DSP source function messageq\_single.c is in directory IPC/ipc\_3\_00\_04\_29/packages/ti/ipc/tests .

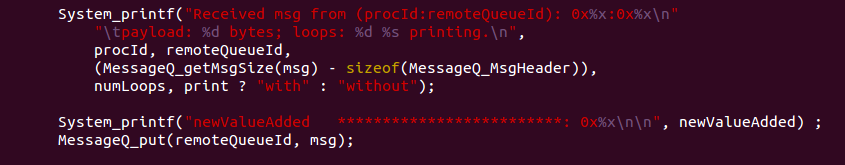
1. Add a new variable newValueAdded to the tsk1Fxn



1. Load the newValueAdded with the last value in the receive message from the ARM



1. Print the value of the newValueAdded



## Step 4 – Build the DSP code

Follow the instructions from the previous task

## Step 5 – Run the file demo\_ipc.sh

Follow the instructions of task 2

1. Observe the printing on the terminal to see the changes that you did in the printf of the ARM code
2. Look at the ARM trace buffer of DSP core 0 to see if the added value is printed. Look at

* Cat /debug/remoteproc/remoteproc0/trace0

The value in newValueAdded should be printed in the trace (in addition to other printings)