MMRFIC BSP

User Manual

0.5, 08 Dec 2022

Table of Contents

Chapter 1. Tracking Information

1.1. Document Details

Project ID	MMRFIC-BSP
Client	MMRFIC
Document No.	F/QP16/03
Owner	Vijay Kumar

1.2. Revision History

Rev	Date	Comment
0.1	19 Nov 2022	• Initial revision.
0.2	25 Nov 2022	• Added sections for Python and C++ OpenCV examples
0.3	28 Nov 2022	• Added section for Big data example
0.4	05 Dec 2022	• Added section for GPIO example
0.5	08 Dec 2022	• Removed dependency on workdrive

Chapter 2. Conventions

In the following examples, command to be run on the host are indicated using `$` prompt as shown below.

```
$ echo hello
```

Commands to be run on the target are indicated using `target#` prompt as shown below.

```
target# echo hello
```

Chapter 3. Tasks

3.1. Run Pre-built IPC Example

In this task, we will run the pre-built IPC example `ex02_messageq` available in the rootfs, that demonstrate the interaction between the Linux running on A15 core and the DSP and IPU cores. The target board used here is the Phytec evaluation board.

The prebuilt binaries of host and slave processors for IPC demonstration is available on root filesystem of the target at `/usr/bin/ipc/examples/ex02_messageq/debug/`

DSP1	`/usr/bin/ipc/examples/ex02_messageq/debug/server_dsp1.xe66`
DSP2	`/usr/bin/ipc/examples/ex02_messageq/debug/server_dsp2.xe66`
IPU1	`/usr/bin/ipc/examples/ex02_messageq/debug/server_ipu1.xem4`
IPU2	`/usr/bin/ipc/examples/ex02_messageq/debug/server_ipu2.xem4`
HOST	`/usr/bin/ipc/examples/ex02_messageq/debug/app_host`

Note

The prebuilt binaries seems to be properly working only on Phytec SDK version `BSP-Yocto-TISDK-AM57xx-PD18.2.0`.

- Step 1: Download the boot images and root filesystem from http://artifactory.phytec.com/artifactory/am57xx-images-released-public/BSP-Yocto-TISDK-AM57xx-PD18.2.0/
- Step 2: Create the SD-Card image using instructions from https://develop.phytec.com/phycore-am57x/latest/software-development/how-to-guides/create-a-bootable-sd-card Boot the system using SD-Card with the root filesystem, from the SDK.
- Step 3: Copy or link the pre-built example firmware files to the standard firmware load paths.

```
target# ln -sf /usr/bin/ipc/examples/ex02_messageq/debug/server_dsp1.xe66 \
        /lib/firmware/dra7-dsp1-fw.xe66
target# ln -sf /usr/bin/ipc/examples/ex02_messageq/debug/server_dsp2.xe66 \
        /lib/firmware/dra7-dsp2-fw.xe66
target# ln -sf /usr/bin/ipc/examples/ex02_messageq/debug/server_ipu1.xem4 \
        /lib/firmware/dra7-ipu1-fw.xem4
target# ln -sf /usr/bin/ipc/examples/ex02_messageq/debug/server_ipu2.xem4 \
        /lib/firmware/dra7-ipu2-fw.xem4
```

- Step 4: Unbind and bind the remoteproc interface, using the following commands. This will cause the firmware to be loaded and executed in the corresponding cores.

```
target# echo 40800000.dsp > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 40800000.dsp > /sys/bus/platform/drivers/omap-rproc/bind

target# echo 41000000.dsp > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 41000000.dsp > /sys/bus/platform/drivers/omap-rproc/bind

target# echo 58820000.ipu > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 58820000.ipu > /sys/bus/platform/drivers/omap-rproc/bind
```

```
target# echo 55020000.ipu > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 55020000.ipu > /sys/bus/platform/drivers/omap-rproc/bind
```

- Step 5: Run the host application on Linux, using the following commands.

```
target# cd /usr/bin/ipc/examples/ex02_messageq/debug
target# ./app_host DSP1
target# ./app_host DSP2
target# ./app_host IPU1
target# ./app_host IPU2
```

3.2. Build and Run IPC Example

In this task, we will build the IPC example `ex02_messageq` from source, and then test them on the Phytec evaluation board, using the Phytec SDK 18.2.0 kernel and root filesystem.

3.2.1. Step 1: SDK Installation

Install the Linux SDK and RTOS SDK using the instructions available at Section 4.1, “SDK Installation”.

3.2.2. Step 2: Build

The source of the IPC examples are available in the RTOS SDK, below is the path for the same,

`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/`

```
$ cd <linux-sdk-path>
$ export TI_RTOS_PATH=<rtos-sdk-path>
$ make ti-ipc-linux-examples
```

The built binaries will be available in the following path

DSP1	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/dsp1/server_dsp1.xe66`
DSP2	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/dsp2/server_dsp2.xe66`
IPU1	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/ipu1/server_ipu1.xem4`
IPU2	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/ipu2/server_ipu2.xem4`
HOST	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/host/bin/debug/app_host`

3.2.3. Step 3: Run

- Use the root filesystem from Phytec SDK, as specified in Section 3.1, “Run Pre-built IPC Example”.
- Copy the built firmware files to the SD-Card root filesystem, and run the examples using the instructions given in Section 3.1, “Run Pre-built IPC Example”.

3.3. Build and Run IPC Example on Phytec SDK 20.1.2 Root FS

In this task, we will build the IPC example `ex02_messageq` from source, and then test them on the Phytec evaluation board, using the Phytec SDK 20.1.2 kernel and root filesystem.

3.3.1. Step 1: SDK Installation

Install the Linux SDK and RTOS SDK using the instructions available at Section 4.1, “SDK Installation”.

3.3.2. Step 2: Fixup VRING Address

There is a difference between the reserved memory assigned for the DSP on the Phytec SDK 20.1.2 Root FS, and the memory address specified during the firmware build.

Ensure the below macros for VRING addresses for DSP are updated in the following file:

```
<rtos-sdk-path>/processor_sdk_rtos_am57xx_08_01_00_09
               /ipc_3_50_04_08/packages
               /ti/ipc/remoteproc/rsc_table_vayu_dsp.h
```

with the values specified below.

```
#if defined (VAYU_DSP_1)
#define PHYS_MEM_IPC_VRING      0x97800000
#elif defined (VAYU_DSP_2)
#define PHYS_MEM_IPC_VRING      0x95000000
#endif
```

Ensure the below macros for VRING addresses for IPU are updated in the following file:

```
<rtos-sdk-path>/processor_sdk_rtos_am57xx_08_01_00_09
               /ipc_3_50_04_08/packages
               /ti/ipc/remoteproc/rsc_table_vayu_ipu.h
```

with the values specified below.

```
#if defined (VAYU_IPU_1)
#define PHYS_MEM_IPC_VRING      0x95800000
#elif defined (VAYU_IPU_2)
#define PHYS_MEM_IPC_VRING      0x9B800000
#endif
```

3.3.3. Step 3: Build

The source of the IPC examples are available in the RTOS SDK, below is the path for the same,

`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/`

```
host$ cd <linux-sdk-path>
host$ export TI_RTOS_PATH=<rtos-sdk-path>
host$ make ti-ipc-linux-examples
```

The built binaries will be available in the following path

DSP1	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/dsp1/server_dsp1.xe66`
DSP2	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/dsp2/server_dsp2.xe66`
IPU1	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/ipu1/server_ipu1.xem4`

IPU2	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/ipu2/server_ipu2.xem4`
HOST	`<rtos-sdk-path>/ipc_3_50_04_08/examples/DRA7XX_linux_elf/ex02_messageq/host/bin/debug/app_host`

3.3.4. Step 4: Run

- Use the root filesystem from Phytec SDK, as specified in Section 3.1, “Run Pre-built IPC Example”.
- Copy the built firmware files to the SD-Card root filesystem, and run the examples using the instructions given in Section 3.1, “Run Pre-built IPC Example”.

3.4. Build and Run IPC Example using CCS

In this task, we will provide an example project that can be used to build and run a DSP firmware on the Phytec evaluation board, using the Phytec SDK 20.1.2 kernel and root filesystem.

3.4.1. Step 1: CCS Installation

Install CCS using the instructions available at Section 4.2, “Code Composer Studio Installation”

3.4.2. Step 2: Build

1. Download `IPC_Example_CCS.tar.xz` from the release folder.
2. Open CCS, Go to `File -> Import -> CCS Projects`
3. Click on `Select Archive File`, browse to downloaded project path, select the `IPC_Example_CCS.tar.xz` file and click on finish.
4. Click on `Project -> Build` to compile the example. Upon completion of the build, the binary will be available at `<workspace>/<project>/Debug/rtos_template_app_am572x_c67_with_ipc.xe66`

3.4.3. Step 3: Run

- Use the root filesystem from Phytec SDK.
- Copy the built firmware files to the SD-Card root filesystem, and run the examples using the instructions given in Section 3.1, “Run Pre-built IPC Example”.

3.5. Record Camera using GStreamer CLI

The following gstreamer pipeline can be used to record camera data to a file.

```
target# gst-launch-1.0 v4l2src device=/dev/video1 name=cam_src ! videoconvert ! \
               queue ! videoconvert ! avimux ! filesink location=output2.avi
```

3.6. Record Camera using OpenCV in Python

This example Python script saves the camera feed into a file.

3.6.1. Step 1: Save the file

Save the below Python code on the target’s filesystem as **opencv_save.py**.

```
import cv2
```

```
def save_feed_to_file(path, fps, codec):
    # open camera
    cam = cv2.VideoCapture(1)
    width = cam.get(cv2.CAP_PROP_FRAME_WIDTH)  # float
    # Get current height of frame
    height = cam.get(cv2.CAP_PROP_FRAME_HEIGHT)  # float

    # setup video writer object
    codec = cv2.VideoWriter_fourcc(*codec)
    out = cv2.VideoWriter(path, codec, fps, (int(width), int(height)))

    while (cam.isOpened()):
        # read a frame
        ret, frame = cam.read()

        # write frame using video writer
        out.write(frame)

def main():
    save_feed_to_file("test.avi", 60, "MJPG")

if __name__ == "__main__":
    main()
```

3.6.2. Step 2: Run the script

Use the below command to run the script.

```
target# python3 <path-to-file>/opencv_save.py
```

Note

To stop saving the feed press **ctrl+c**.

3.7. Record Camera using OpenCV in C++

This cpp example code saves the camera feed as a file.

3.7.1. Step 1: Save the file

Save the below file on target’s filesystem as **opencv_save.cpp**.

```
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

using namespace std;
using namespace cv;
```

```
int main(int argc, char *argv[])
{
        Mat frame;
        VideoCapture cap(1);

        int width = cap.get(cv::CAP_PROP_FRAME_WIDTH);
        int length = cap.get(cv::CAP_PROP_FRAME_HEIGHT);
        int fps = cap.get(cv::CAP_PROP_FPS);
        Size sizeFrame(width, length);

        if (!cap.isOpened()) {
                cerr << "Error in opening feed\n";
                return -1;
        }

        //Video Codec, create video writer object
        int codec = VideoWriter::fourcc('M', 'J', 'P', 'G');
        VideoWriter writer("./feed_save.avi", codec, fps, sizeFrame);

        if (!writer.isOpened()) {
                cerr << "Error in opening writer\n";
                return -1;
        }

        while (1) {

                // read frame
                if (!cap.read(frame)) {
                        cerr << "Error, blank frame\n";
                        return -1;
                }

                // write frame
                writer.write(frame);
        }
        cap.release();
        writer.release();
        return 0;
}
```

3.7.2. Step 2: Compile the source

Use the below command to compile.

```
target# arm-linux-gnueabihf-g++ <path-to-file>/opencv_save.cpp \
        -I/usr/include/opencv4 -L/usr/local/libc -lopencv_videoio \
        -lopencv_imgproc -lopencv_highgui -lopencv_core -o opencv_save
```

3.7.3. Step 3: Run the executable

Run the exacutable as shown below.

```
target# ./opencv_save
```

Note

To stop saving the feed press **ctrl+c**.

3.8. Build and run big data IPC on DSP1

3.8.1. Step 1: SDK Installation

Install the Linux SDK and RTOS SDK using the instructions available at Section 4.1, "SDK Installation".

3.8.2. Step 2: Fixup VRING Address

There is a difference between the reserved memory assigned for the DSP on the Phytec SDK 20.1.2 Root FS, and the memory address specified during the firmware build.

Ensure the below macros for VRING addresses are updated in the following file:

```
<linux-sdk-path>/example-applications/big-data-ipc-demo-linux-01.03.00.00/
                host_linux/simple_buffer_example/shared/DRA7XX/rsc_table_dsp.h
```

with the values specified below.

```
#if defined (VAYU_DSP_1)
#define PHYS_MEM_IPC_VRING      0x97800000
```

3.8.3. Step 4: Change example to send 32MB data

In file `<linux-sdk-path>/example-applications/big-data-ipc-demo-linux-01.03.00.00/host_linux/simple_buffer_example/host/App.c`

change `BIG_DATA_POOL_SIZE` to 0x2000000

```
#define BIG_DATA_POOL_SIZE 0x2000000
```

3.8.4. Step 5: Build big data IPC demo

Big data IPC demo is available at `<linux-sdk-path>/example-applications/ big-data-ipc-demo-linux-01.03.00.00/host_linux/simple_buffer_example`

```
host$ cd <linux-sdk-path>/example-applications/big-data-ipc-demo-linux-01.03.00.00/ \
         host_linux/simple_buffer_example
```

Edit the products.mk file in this directory with the following values:

```
DEPOT = <RTOS_SDK_INSTALL_DIR>
TOOLCHAIN_LONGNAME = arm-none-linux-gnueabihf
TOOLCHAIN_INSTALL_DIR = <linux-sdk-path>/linux-devkit/sysroots/x86_64-arago-linux/usr
TOOLCHAIN_PREFIX = $(TOOLCHAIN_INSTALL_DIR)/bin/$(TOOLCHAIN_LONGNAME)-

BIOS_INSTALL_DIR = $(DEPOT)/bios_6_76_03_0
IPC_INSTALL_DIR        = $(DEPOT)/ipc_3_50_04_08
XDC_INSTALL_DIR        = $(DEPOT)/xdctools_3_55_02_22_core

ti.targets.arm.elf.M4  = $(DEPOT)/ti-cgt-arm_18.12.5.LTS
ti.targets.elf.C66     = $(DEPOT)/ti-cgt-c6000_8.3.2
```

Comment the following line in the makefile present in this directory:

```
PROC_IPU_NAME = "IPU1"
```

export the PLATFORM environment variable

```
host$ export PLATFROM=DRA7XX
```

Build the demo

```
host$ cd <linux-sdk-path>/example-applications/big-data-ipc-demo-linux-01.03.00.00/
host$ make host_linux
```

The linux binary `app_host` is available at `<linux-sdk-path>/example-applications/ big-data-ipc-demo-linux-01.03.00.00/host_linux/simple_buffer_example/ host/bin/ DRA7XX/release`

The DSP binary is `server_dsp.xe66` available at `<linux-sdk-path>/ example-applications/ big-data-ipc-demo-linux-01.03.00.00/host_linux/ simple_buffer_example/dsp/bin/ DRA7XX/release`

Copy the built firmware files to the home directory of root filesystem.

3.8.5. Step 6: Run big data IPC demo on target

- Stop OpenCL application as it interferes with DSP

```
target# systemctl stop ti-mct-daemon.service
```

- Load the DSP1 firmware

```
target# ln -sf /home/root/server_dsp.xe66 /lib/firmware/dra7-dsp1-fw.xe66
```

- Reload DSP1 with the new firmware

```
target# echo 40800000.dsp > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 40800000.dsp > /sys/bus/platform/drivers/omap-rproc/bind
```

- Run the host program on DSP1

```
target# ./app_host DSP1
```

3.9. Build and run big data IPC on DSP2

3.9.1. Step 1: SDK Installation

Install the Linux SDK and RTOS SDK using the instructions available at Section 4.1, “SDK Installation”.

3.9.2. Step 2: Fixup VRING Address

There is a difference between the reserved memory assigned for the DSP on the Phytec SDK 20.1.2 Root FS, and the memory address specified during the firmware build.

Ensure the below macros for VRING addresses are updated in the following file:

```
<linux-sdk-path>/example-applications/big-data-ipc-demo-linux-01.03.00.00/
                host_linux/simple_buffer_example/shared/DRA7XX/rsc_table_dsp.h
```

with the values specified below.

```
#elif defined (VAYU_DSP_2)
#define PHYS_MEM_IPC_VRING      0x95000000
#endif
```

Change definition of `VAYU_DSP_1` in the following `<linux-sdk-path>/example-applications/big-data-ipc-demo-linux-01.03.00.00/ host_linux/simple_buffer_example/shared/DRA7XX/rsc_table_dsp.h` file to:

```
#define VAYU_DSP_2
```

3.9.3. Step 3: Build big data IPC demo

Modify Makefile to build for DSP2

Modify makefile present in `<linux-sdk-path>/example-applications/ big-data-ipc-demo-linux-01.03.00.00/host_linux/simple_buffer_example/`

```
PROC_DSP_NAME = "DSP2"
```

Follow the steps at Section 3.8.4, “Step 5: Build big data IPC demo” to build the binary for DSP2.

3.9.4. Step 4: Run big data ipc demo on target

- Stop OpenCL application as it interferes with DSP

```
systemctl stop ti-mct-daemon.service
```

- Load the DSP2 firmware

```
target# ln -sf /home/root/server_dsp.xe66 /lib/firmware/dra7-dsp2-fw.xe66
```

- Reload DSP2 with the new firmware

```
target# echo 41000000.dsp > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 41000000.dsp > /sys/bus/platform/drivers/omap-rproc/bind
```

- Run the host program on DSP2

```
target# ./app_host DSP2
```

3.10. Build and run GPIO example on DSP1 using CCS

In this task, we will provide an example project that can be used to build and run a GPIO firmware on the Phytec evaluation board, using the Phytec SDK 20.1.2 kernel and root filesystem.

3.10.1. Step 1: CCS Installation

Install CCS using the instructions available at Section 4.2, “Code Composer Studio Installation”

3.10.2. Step 2: Build the example

1. Download `GPIO_Example.tar.xz` from the release folder.

2. Open CCS, Go to `File -> Import -> CCS Projects`
3. Click on `Select Archive File` and browse to select the downloaded project path, select the `GPIO_Example.tar.xz` file and click on finish.
4. Click on `Project -> Build` to compile the example. Upon completion of the build, the binary will be available at `<workspace>/<project>/Debug/GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.out`
5. Make a copy of the binary and rename it as `GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.xe66`
6. Copy the `GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.xe66` binary to home directory of root filesystem.

3.10.3. Step 3: Run the example on DSP1

1. Load the firmware using the following command

```
target# ln -sf /home/root/GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.xe66 \
        /lib/firmware/dra7-dsp2-fw.xe66
```

1. Reload DSP1 with the new firmware

```
target# echo 40800000.dsp > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 40800000.dsp > /sys/bus/platform/drivers/omap-rproc/bind
```

1. Observe the `LED, D8` on the carrier board blinking.

3.11. Build and run GPIO example on DSP2 using CCS

In this task, we will provide an example project that can be used to build and run a GPIO firmware on the Phytec evaluation board, using the Phytec SDK 20.1.2 kernel and root filesystem.

3.11.1. Step 1: CCS Installation

Install CCS using the instructions available at Section 4.2, “Code Composer Studio Installation”

3.11.2. Step 2: Import the example on CCS

1. Download `GPIO_Example.tar.xz` from the release folder.
2. Open CCS, Go to `File -> Import -> CCS Projects`
3. Click on `Select Archive File` and browse to select the downloaded project path, select the `GPIO_Example.tar.xz` file and click on finish.

3.11.3. Step 2: Modify the code to build for DSP2

1. In the home directory of project folder open the file `rsc_table_vayu_dsp.c` on CCS
2. Modify the following line `#define VAYU_DSP_1` to

```
#define VAYU_DSP_2
```

3.11.4. Step 3: Build the project

1. Click on `Project -> Build` to compile the example. Upon completion of the build, the binary will be available at `<workspace>/<project>/Debug/GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.out`

2. Make a copy of the binary and rename it as `GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.xe66`
3. Copy the `GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.xe66` binary to home directory of root filesystem.

3.11.5. Step 3: Run the example on DSP2

1. Load the firmware using the following command

```
target# ln -sf /home/root/GPIO_LedBlink_evmAM572x_c66xExampleProject_with_ipc.xe66 \
        /lib/firmware/dra7-dsp2-fw.xe66
```

1. Reload DSP2 with the new firmware

```
target# echo 41000000.dsp > /sys/bus/platform/drivers/omap-rproc/unbind
target# echo 41000000.dsp > /sys/bus/platform/drivers/omap-rproc/bind
```

1. Observe the `LED, D8` on the carrier board blinking.

Chapter 4. Common Procedures

4.1. SDK Installation

Download the linux SDK and RTOS SDK from the below links,

Linux SDK [https://dr-download.ti.com/software-development/software-development-kit-sdk/MD-L1u0FxxpZf/08.02.01.00/ti-processor-sdk-linux-am57xx-evm-08_02_01_00-Linux-x86-Install.bin]

RTOS SDK [https://dr-download.ti.com/software-development/software-development-kit-sdk/MD-f80FNuZJIh/08.01.00.09/processor_sdk_rtos_am57xx_08_01_00_09-linux-x64-installer.tar.gz]

Follow the below installation steps.

4.1.1. RTOS SDK Installation

Use the below commands,

```
$ export SDK_INSTALL_PATH=<RTOS_SDK_INSTALL_DIR>
$ export TOOLS_INSTALL_PATH=<RTOS_SDK_INSTALL_DIR>
$ cd <RTOS_SDK_INSTALL_DIR>/processor_sdk_rtos_am57xx_08_01_00_09
$ source ./setupenv.sh
```

4.1.2. Linux SDK Installation

Use the below commands,

```
$ chmod u+x <downloaded-linux-SDK-installer-path>
$ ./<downloaded-linux-SDK-installer>
```

Proceed with the installation process mentioned in the pop-up window.

4.2. Code Composer Studio Installation

Download and install Code Composer Studio (CCS) from the following link.

Code Composer Studio [https://dr-download.ti.com/software-development/ide-configuration-compiler-or-debugger/MD-J1VdearkvK/12.1.0/CCS12.1.0.00007_linux-x64.tar.gz]

Start the CCS application once installed, it can be started as below

```
$ <ccs-installation-path>/ccs1210/ccs/eclipse/ccstudio
```

Click on, `Window -> Preferences -> Code Composer Studio -> Products`, click on `Add` and select the path where the RTOS SDK is installed, then click on `Refresh`, select all the products from the pop-up window which shows the list of products discovered and click install.

Chapter 5. eMMC Flashing instructions

5.1. Boot from SD card

Download `tisdk-rootfs-image-am572x-bel-mmrfic.tar` , `u-boot.img` and `MLO` available in the release folder on workdrive.

Create a bootable SD card and boot the board.

Copy the images downloaded from the release folder to the SD card's rootfs

5.2. Create partitions on eMMC

Run the following command:

```
target# fdisk /dev/mmcblk1
```

The above command displays the fdisk prompt.

Type the below commands in fdisk prompt to create the partition. the explanation is mentioned in the comments following the command:

```
command:: o             # Create a DOS partition
command:: n             # Add a new partition for boot
command:: p             # Make it as primary partition
command:: 1             # Create partition number 1
command:: 2048          # Allocate size of first sector
command:: +2M           # Allocate size of last sector
command:: t             # Change parition type
Hex code (type L to list all codes): c # Select partition type as W95 FAT32 (LBA)
command:: a             # Turns on bootable flag on parition 1
command:: n             # Add a new partition for rootfs
command:: 2             # Create partition number 2
command:: <enter>       # Allocate default size for first sector
command:: <enter>       # Allocate default size for last sector
command:: w             # Write table to disk and exit
```

5.3. Create file system type using mkfs

Create VFAT file system type for Boot partition

```
target# mkfs.vfat /dev/mmcblk1p1
```

Create EXT4 file system type for rootfs

```
target# mkfs.ext4 -b 4096 /dev/mmcblk1p2
```

5.4. Mount the eMMC partitions

```
target# mkdir -p  /mnt/uboot/
```

```
target# mount /dev/mmcblk1p1 /mnt/uboot
target# mkdir -p /mnt/rootfs
target# mount /dev/mmcblk1p2 /mnt/rootfs
```

5.5. Copy Rootfs and U-Boot to mount points

```
target# cp </path/to/MLO> </path/to/u-boot.img> /mnt/uboot
target# tar -xvf /path/to/tisdk-rootfs-image-am572x-bel-mmrfic.tar -C /mnt/rootfs
```

5.6. Create a test file

The file eMMC is created in the root directory of rootfs. This is to test whether the board has booted from eMMC and the flashing procedure was successful.

```
target# touch /mnt/rootfs/eMMC
```

5.7. Boot from eMMC

Follow the below step to boot from eMMC:

1. Power OFF the board
2. Unmount the SD card from board
3. Power ON the board
4. Check for the eMMC file in root directory of file system