MCSDK Image Processing Demonstration Guide

Multicore Software Development Kit
Image Processing Demonstration
User's Guide

Last updated: //

Overview

The Image Processing Demonstration illustrates the integration of key components in the Multicore Software Development Kit (MCSDK) on Texas Instruments (TI) multicore DSPs and System-on-Chips. The purpose of the demonstration is to provide a multicore software development framework on an evaluation module (EVM).

This document covers various aspects of the demonstration application, including a discussion on the requirements, software design, instructions to build and run the application, and troubleshooting steps. Currently, only SYS/BIOS is supported as the embedded OS.

This application shows implementation of an image processing system using a simple multicore framework. This application will run TI image processing kernels (a.k.a, *imagelib*) on multiple cores to do image processing (eg: edge detection, etc) on an input image.

There are three different versions of this demonstration that are included in the MCSDK. However, not all three versions are available for all platforms.

- *Serial Code*: This version uses file i/o to read and write image file. It can run on the simulator or an EVM target platform. The primery objective of this version of the demo is to run Prism and other software tools on the code to analyze the basic image processing algorithm.
- *IPC Based*: The IPC based demo uses SYS/BIOS IPC component to communicate between cores to perform an image processing task parallel. See below for details.
- *OpenMP Based*: (Not available for C6657) This version of the demo uses OpenMP to run the image processing algorithm on multiple cores.

Note: The current implementation of this demonstration is not optimized. It should be viewed as the initial implementation of the BIOS MCSDK software eco-system for creating an image processing functionality. Further analysis and optimization of the demonstration are under progress.

Note: There are three versions of the demo provided in the release. The IPC based version runs on multiple cores and shows explicit IPC programming framework. The serial version of the demo runs on the simulator. The OpenMP version uses OpenMP to communicate between cores to process the input images. Unless explicitly specified, the IPC based version is assumed in this document.

Requirements

The following materials are required to run this demonstration:

- TMS320C6x low cost EVMs [Check Image Processing release notes for supported platforms]
- Power cable
- Ethernet cable
- Windows PC with CCSv5

Software Design

The following block diagram shows the framework used to implement the image processing application:

The following diagram shows the software pipeline for this application: .

More about processing algorithms

The application will use imagelib APIs for its core image processing needs.

Following steps are performed for edge detection

- Split input image into multiple overlapping slices
- If it is a RGB image, separate out the Luma component (Y) for processing (See YCbCr [1] for further details)
- Run Sobel operator [2] (IMG_sobel_3x3_8) to get the gradient image of each slices
- Run the thresholding operation (IMG_thr_le2min_8) on the slices to get the edges
- Combine the slices to get the final output

Framework for multicore

The current framework for multicore is either IPC Message Queue based framework or OpenMP. Following are the overall steps (the master and threads will be run on 1 or more cores)

- The master thread will preprocess the input image (described in User interface section) to make a gray scale or luma image
- The master thread signal each slave thread to start processing and wait for processing complete signal from all slave threads
- The slave threads run edge detection function (described above) to generate output edge image of the slice
- Then the slave threads signal master thread indicating the processing completed
- Once master thread receives completion signal from all threads it proceeds with further user interface processing (described in User interface section)

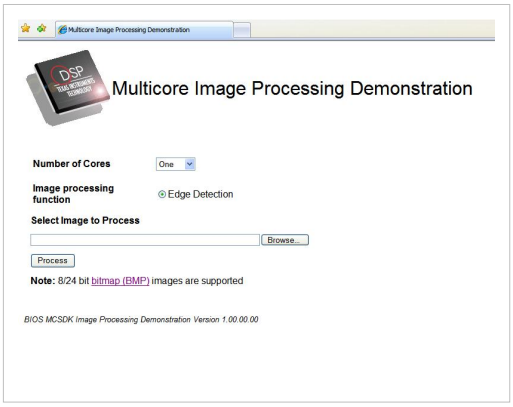
Profiling of the algorithm

- The profiling information live processing time will be presented at the end of the processing cycle
- Core image processing algorithms is instrumented using UIA for analysis and visualization using MCSA (Multicore System Analyzer)

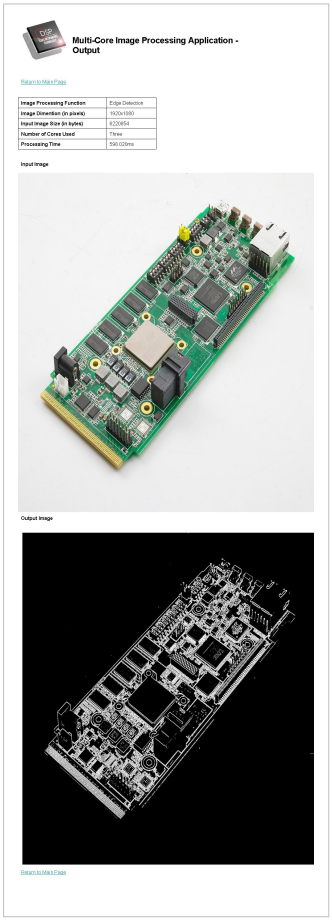
User interface

The user input image will be a BMP image. The image will be transferred to external memory using NDK (http). Following are the stapes describing application user interface and their interaction

- At the time of bootup the board will bring configure IP stack with static/dynamic IP address and start a HTTP server
- The board will print the IP address in CCS console
- The user will use the IP address to open the index/input page (see link Sample Input Page)
- The application will support BMP image format
- The master thread will extract the RGB values from BMP image
- Then the master thread will initiate the image processing (as discussed above) and wait for its completion
- Once the processing completes, it will create output BMP image
- The master thread will put input/output images in the output page (see link Sample Output Page)



Software outline of the OpenMP demo



- The main task is called by OpenMP in core 0, spawns a task to initialize NDK, then gets/sets IP address and starts a web service to transfer user inputs and images. The main task then creates a mailbox and waits on a message post to the mailbox.
- The NDK calls a callback function to the application to retrieve the image data from user. The function reads the image and posts a message with the image information to the main task. Then it waits on a mailbox for a message post from main task.
- After receiving the message, the main task extracts RGB, splits the image into slices and processes each slices in different cores. A code snippet of this processing is provided below.

```
pragma omp parallel for shared(p_slice, number_of_slices, ret_val) private(i) for (i = 0; i < number_of_slices; i++ ) {
```

```
 DEBUG_PRINT(printf("Processing slice # %d\n", i);)
   /* Process a slice */
   process_rgb (&p_slice[i]);
 if (p_slice[i].flag != 0) {
   printf("mc_process_bmp: Error in processing slice %d\n", i);
```

```
pragma omp atomic
```

```
   ret_val = -1;
 }
 DEBUG_PRINT(printf("Processed slice # %d\n", i);)
```

```
}
```

if (ret_val == -1) {

```
 goto close_n_exit;
```

}

- After processing is complete, the main task creates the output image and sends the image information to the callback task using a message post. Then it waits on the mailbox again.
- The NDK callback task wakes up with the message post and sends the result image to the user.

Different Versions of Demo

Software Directory Structure Overview

The Image Processing Demonstration is present at <MCSDK INSTALL DIR>\demos\image_processing

- <MCSDK INSTALL DIR>\demos\image_processing\ipc\common directory has common slave thread functions which runs on all cores for the IPC based demo; The image processing function runs in this slave thread context
- <MCSDK INSTALL DIR>\demos\image_processing\ipc\master directory has main thread, which uses NDK to transfer images and IPC to communicate to other cores to process the images
- <MCSDK INSTALL DIR>\demos\image_processing\ipc\slave directory has the initialization function for all slave cores
- <MCSDK INSTALL DIR>\demos\image_processing\openmp\src directory has the main thread, which uses NDK to transfer images and OpenMP to communicate between cores to process the image
- <MCSDK INSTALL DIR>\demos\image_processing\ipc\evmc66##l\[master|slave] directories have the master and slave CCS project files for the IPC based demo
- <MCSDK INSTALL DIR>\demos\image_processing\openmp\evm66##l directory has the CCS project files for the OpenMP based demo
- <MCSDK INSTALL DIR>\demos\image_processing\######\evmc66##l\platform directory has the target configuration for the project
- <MCSDK INSTALL DIR>\demos\image_processing\serial directory has the serial version of the implementation
- <MCSDK INSTALL DIR>\demos\image_processing\utils directory has utilities used on the demo, like MAD config files
- <MCSDK INSTALL DIR>\demos\image_processing\images directory has sample BMP images

Serial Code

Run Instructions for Serial based demo application

The pre-compiled libraries are provided as a part of MCSDK release.

Please follow the procedures below to load images using CCS and run the demo.

Please refer the hardware setup guide for further the setup details.

- Connect the board to a Ethernet hub or PC using Ethernet cable.
- The demo runs in Static IP mode if User Switch 1 (SW9, position 2) is OFF else if it is ON then it runs DHCP mode. See the Hardware Setup section for the location of User Switch 1.
- If it is configured in static IP mode, the board will come up with IP address 192.168.2.100, GW IP address 192.168.2.101 and subnet mask 255.255.254.0
- If it is configured in DHCP mode, it would send out DHCP request to get the IP address from a DHCP server in the network.
- There is one image to be loaded on core 0. The image name is <MCSDK INSTALL DIR>\demos\image_processing\serial\Debug\image_processing_serial_simc6678.out.
- Connect the debugger and power on the board.
- It should open debug perspective and open debug window.
- Connect to only core 0, if the board is in no-boot mode make sure gel file is run to initialize ddr.
- Load image_processing_seria_simc6678.out to core 0.
- Run the corre 0, in the CIO window, the board should pint IP address information (eg: Network Added: If-1:192.168.2.100)
- Open a web browser in the PC connected to the HUB or the board.

- Enter the IP address of the board, it should open up the image processing demo web page.
- Please follow the instructions in the web page to run the demo.
- Note that sample BMP images are provided in <MCSDK INSTALL DIR>\demos\image_processing\images

Build Instructions for Serial based demo application

Please follow the steps below to re-compile the Serial based demo image (These steps assume you have installed the MCSDK and all dependent packages).

- Open CCS->Import Existing... tab and import project from <MCSDK INSTALL DIR>\demos\image_processing\serial.
- It should import image_processing_serial_simc6678 project.
- The project should build fine for Release and Debug profile.

IPC-Based

Run Instructions for IPC based demo application

The pre-compiled libraries are provided as a part of MCSDK release.

Please follow the procedures below to load images using CCS and run the demo.

Please refer the hardware setup guide for further the setup details.

- Connect the board to a Ethernet hub or PC using Ethernet cable.
- The demo runs in Static IP mode if User Switch 1 (SW9, position 2) is OFF else if it is ON then it runs in DHCP mode. See the Hardware Setup section for the location of User Switch 1.
- If it is configured in static IP mode, the board will come up with IP address 192.168.2.100, GW IP address 192.168.2.101 and subnet mask 255.255.254.0
- If it is configures in DHCP mode, it would send out DHCP request to get the IP address from a DHCP server in the network.
- There are two images to be loaded to master (core 0) and other cores. The core 0 to be loaded with <MCSDK INSTALL DIR>\demos\image_processing\ipc\evmc66##l\master\Debug\image_processing_evmc66##l_master.out image and other cores (referred as slave cores) to be loaded with <MCSDK INSTALL DIR>\demos\image_processing\ipc\evmc66##l\slave\Debug\image_processing_evmc66##l_slave.out image.
- Connect the debugger and power on the board.
- In CCS window, launch the target configuration file for the board.
- It should open debug perspective and open debug window with all the cores.
- Connect to all the cores and load image_processing_evmc66##l_master.out to core 0 and image_processing_evmc66##l_slave.out to all other cores.
- Run all the cores, in the CIO console window, the board should print IP address information (for eg: Network Added: If-1:192.168.2.100)
- Open a web browser in the PC connected to the HUB or the board.
- Enter the IP address of the board, it should open up the image processing demo web page.
- Please follow the instructions in the web page to run the demo.
- Note that, sample BMP images are provided in <MCSDK INSTALL DIR>\demos\image_processing\images

Note: If you want to run the demo in static IP address mode, make sure the host PC is in same subnet or can reach the gateway. A sample setup configuration is shown below.

In Windows environment

Set up TCP/IP configuration of 'Wired Network Connection' as shown in Wired Network Connection in Windows.

In Linux environment

Run following command to set the static IP address for the current login session on a typical Linux setup.

```
sudo ifconfig eth0 192.168.2.101 netmask 255.255.254.0
```

Build Instructions for IPC based demo application

Please follow the steps below to re-compile the IPC based demo image (These steps assume you have installed the MCSDK and all the dependent packages).

- Open CCS->Import Existing... tab and import project from <MCSDK INSTALL DIR>\demos\image_processing\ipc.
- It should import two projects image_processing_evmc66##l_master and image_processing_evmc66##l_slave.
- Right click on each project->Properties to open up the properties window.
- Goto CCS Build->RTSC and check if in other repository have link to <MCSDK INSTALL DIR> (the actual directory).
- If IMGLIB C66x is unchecked, please select 3.0.1.0 to check it.
- The RTSC platform should have demos.image_processing.evmc66##l.platform.
- The project should build fine.

OpenMP-Based

Run Instructions for OpenMP based demo application

The pre-compiled libraries are provided as a part of MCSDK release.

Please follow the procedures below to load images using CCS and run the demo.

Please refer the hardware setup guide for further the setup details.

- Connect the board to a Ethernet hub or PC using Ethernet cable.
- The demo runs in Static IP mode if User Switch 1 is OFF else if it is ON then it runs in DHCP mode. See the Hardware Setup section for the location of User Switch 1.
- If it is configured in static IP mode, the board will come up with IP address 192.168.2.100, GW IP address 192.168.2.101 and subnet mask 255.255.254.0
- If it is configures in DHCP mode, it would send out DHCP request to get the IP address from a DHCP server in the network.
- There ONE image to be loaded to core 0. The image name is <MCSDK INSTALL DIR>\demos\image_processing\openmp\evmc66##l\Release\image_processing_openmp_evmc66##l.out.
- Connect the debugger and power on the board.
- In CCS window, launch the target configuration file for the board.
- It should open debug perspective and open debug window.
- Connect to only core 0, if the board is in no-boot mode make sure gel file is run to initialize ddr.
- Load image_processing_openmp_evmc66##l.out to core 0.
- Run the core 0, in the CIO console window, the board should print IP address information (for eg: Network Added: If-1:192.168.2.100)
- Open a web browser in the PC connected to the HUB or the board.
- Enter the IP address of the board, it should open up the image processing demo web page.
- Please follow the instructions in the web page to run the demo.
- Note that, sample BMP images are provided in <MCSDK INSTALL DIR>\demos\image_processing\images

Build Instructions for OpenMP based demo application

Please follow the steps below to re-compile the OpenMP based demo image (These steps assume you have installed the MCSDK and all the dependent packages).

- Open CCS->Import Existing... tab and import project from <MCSDK INSTALL DIR>\demos\image_processing\openmp\evmc66##l.
- It should import image_processing_openmp_evmc66##l project.
- The project should build fine for Release and Debug profile.

Multicore System Analyzer integration and usage

The System Analyzer provides correlated realtime analysis and visiblity into application running on single or multicore. Analysis and visibility includes Execution Graph, Duration Analysis, Context Aware Profile, Load Analysis and Statistics Analysis. Basic instrumentation using Unified Instrumentation Architecture (UIA) collects data in realtime and transport via Ethernet or JTAG to host where it it decoded, correlated, analyzed and visualized.

System Analyzer is automatically added to CCS5.0 by the MCSDK istaller. CCS5.1 is shipped with the System Analyzer included.

The Image Processing Demo has been instrumented for duration/benchmark and CPU load analysis. Detailed information on running the demo with System Analyzer is provided in System Analyzer and the MCSDK Demo [3] page.

Image Processing Demo Analysis with Prism

This section we will use Prism software [4] to analyze the serial version of image processing demo. The steps below would assume Prism with C66x support is installed in the system and user completed the *Prism Getting Started - Tutorials* provided in the help menu.

Bring up the demo with Prism software

- Bring up the CCS as specified in the Prism documentation
- Edit *macros.ini* file from *<MCSDK INSTALL DIR>\demos\image_processing\serial* directory and change *../../../../imglib_c66x_#_#_#_#* to static path to IMGLIB package
- Open *Import Existing CCS Eclipse Project* and select search directory *<MCSDK INSTALL DIR>\demos\image_processing\serial*
- Check *Copy projects into workspace* and click *Finish*. This will copy the project to the workspace for Prism.
- Clean and re-compile the project
- Open the C6678 simulator, load the image to core0
- Open *Tools->GEL files*, select *Load GEL* and load/open *tisim_traces.gel* from *<CCSv5 INSTALL DIR>\ccs_base_####\simulation_csp_ny\env\ccs\import* directory
- Then select *CPU Register Trace->StartRegTrace* to start the trace, then run the program, wait till it finishes, then select *CPU Register Trace->StopRegTrace* to stop the trace
- Open *Prism->Show Prism Perspective*. It will start Prism Perspective
- Right click on the project and select *Create New PAD File*, it would open the New Prism Analysis Definition window, hit next
- It will open *Architecture Selection* window, select C6671 (single core) template. Then select Finish to open PAD file
- Select *Run->Debug Configurations->prismtrace*, this will convert the simulator generated traces to the traces required by Prism

- The PAD window should have filled in with default trace (PGSI, PGT) file names generated in above step
- Select *Read Trace* to read the trace
- After it read the trace, then select the complete trace from overview window and hit *Load Slice*
- The *Functions* tab will show the functions and their cycle information during the execution
- Observe the *Core 0* scheduling in the schedule window, you can place a marker for this run

What If analysis

The Prism tool allows user to analyze *What If* scenarios for the code

- *What If* the code is run on multiple cores
  - In the function window, right click on *process_rgb* function, select *Force Task* and hit *Apply*. This would make *process_rgb* function simulated as a separate task
  - In the *Architecture* tab, select *C6678 (8 Core)* template, hit *Apply*
  - Observe in *Schedule* tab, the change in execution when selected the *process_rgb* is simulated to run on 8 cores
  - A marker can be placed to compare the improvement
- *What If* the dependencies are removed
  - The *Dependencies* window helps to see and analyze the dependencies (which are preventing the task to be executed in multiple cores simultaneously)
  - Un-check *Serialized* check-boxes against dependency rows and hit *Apply*
  - Add the comparison marker in the *Schedule* tab and check the improvement

The Prism supports more functionality then described in this section. Please see Prism documentation for more information.

Multicore booting using MAD utilities

The detailed information on the Multicore Application Deployment a.k.a MAD utility is provided in MAD user guide [5] page.

This section will provide you the detail instructions on how the tool and boot the demo from flash/ethernet.

Linking and creating bootable application image using MAD utilities

The BIOS MCSDK installation provides MAD tool in *<MCSDK INSTALL DIR>\tools\boot_loader\mad-utils*. This package contains necessary tools to link the application to a single bootable image.

The image processing demo has following updates to create MAD image:

- The master and slave images are linked with *--dynamic* and *--relocatable* options.
- The MAD config files used to link the master and slave programs are provided in *<MCSDK INSTALL DIR>\demos\image_processing\utils\mad\evmc66##l\config-files*. Following are few items to note on the config file.
  - *maptoolCfg_evmc#####.json* has the directory and file name information for the tools
  - *deployment_template_evmc#####.json* has the deployment configuration (it has device name, partition and application information). Following are some more notes on the configuration file.
    - For C66x devices, the physical address is 36 bits and virtual address is 32 bits for external devices, this includes MSMC SRAM and DDR3 memory subsystem.
    - The *secNamePat* element string is a regular expression string.
    - The sections *bss*, *neardata*, *rodata* must be placed in one partition and in the order it is shown here

- The build script *<MCSDK INSTALL DIR>\demos\image_processing\utils\mad\evmc66##l\build_mad_image.bat* can be used to re-create the image

Note: The compilation will split out lots of warning like *Incompatible permissions for partition ...*, it can be ignored for now. This is due to mis-match in partition permissions wrt. the sections placed in the partition

- The bootable image is placed in *<MCSDK INSTALL DIR>\demos\image_processing\utils\mad\evmc66##l\images*

Pre-link bypass MAD image

Please see MAD user guide for more information on pre-link bypassed MAD image. The build script *build_mad_image_prelink_bypass.bat* can be used to build images with this mode.

Booting the application image using IBL

This image can be booted using IBL bootloader.

Following things to be noted on booting the image

- The image type/format is *ibl_BOOT_FORMAT_BBLOB*, so the IBL needs to be configured to boot this format
- The branch address (Branch address after loading) of the image [it is set to *0x9e001040* (or *0x80001040* if you are using BIOS MCSDK v 2.0.4 or prior) in MAL application], is different from default IBL boot address, so the IBL configuration needs to be updated to jump to this address

The following sections will outline the steps to boot the image from Ethernet and NOR using IBL. Please see IBL documentation on the detail information on booting.

Booting from Ethernet (TFTP boot)

- Change IBL configuration: The IBL configuration parameters are provided in a GEL file *<MCSDK INSTALL DIR>\tools\boot_loader\ibl\src\make\bin\i2cConfig.gel*. All the changes needs to be done in the function *setConfig_c66##_main()* of the gel file.
  - The IBL configuration file sets PC IP address 192.168.2.101, mask 255.255.255.0 and board IP address as 192.168.2.100 by default. If these address needs to be changed, open the GEL file, change *ethBoot.ethInfo* parameters in function *setConfig_c66##_main()*
  - Make sure the *ethBoot.bootFormat* is set to *ibl_BOOT_FORMAT_BBLOB*
  - Set the *ethBoot.blob.branchAddress* to *0x9e001040* (or *0x80001040* if you are using BIOS MCSDK v 2.0.4 or prior).
  - Note that the application name defaults to *app.out*

```
menuitem "EVM c66## IBL";
```

hotmenu setConfig_c66##_main() {

```
ibl.iblMagic = ibl_MAGIC_VALUE;
ibl.iblEvmType = ibl_EVM_C66##L;


...
```

```
ibl.bootModes[2].u.ethBoot.doBootp = FALSE;
ibl.bootModes[2].u.ethBoot.useBootpServerIp = TRUE;
ibl.bootModes[2].u.ethBoot.useBootpFileName = TRUE;
ibl.bootModes[2].u.ethBoot.bootFormat = ibl_BOOT_FORMAT_BBLOB;
```

```
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.ipAddr, 192,168,2,100);
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.serverIp, 192,168,2,101);
```

```
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.gatewayIp, 192,168,2,1);
SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.netmask, 255,255,255,0);
```

```
...
```

```
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0] = 'a';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[1] = 'p';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[2] = 'p';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[3] = '.';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[4] = 'o';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[5] = 'u';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[6] = 't';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[7] = '\0';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[8] = '\0';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[9] = '\0';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[10] = '\0';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[11] = '\0';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[12] = '\0';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[13] = '\0';
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[14] = '\0';
```

```
ibl.bootModes[2].u.ethBoot.blob.startAddress = 0x9e000000 /*0x80000000 for BIOS MCSDK v2.0.4 or prior*/; /* Load start address */

ibl.bootModes[2].u.ethBoot.blob.sizeBytes = 0x20000000;

ibl.bootModes[2].u.ethBoot.blob.branchAddress = 0x9e001040 /*0x80001040 for BIOS MCSDK v2.0.4 or prior*/; /* Branch address after loading */
```

```
ibl.chkSum = 0;
```

}

- Write IBL configuration:
  - Connect the board using JTAG, power on the board, open CCS, load the target and connect to core 0. Select *Tools->GEL Files* and in the GEL Files window right click and load GEL. Then select and load *<MCSDK INSTALL DIR>\tools\boot_loader\ibl\src\make\bin\i2cConfig.gel*.
  - Load I2C writer *<MCSDK INSTALL DIR>\tools\boot_loader\ibl\src\make\bin\i2cparam_0x51_c66##_le_0x500.out* to Core 0 and run. It will ask to run the GEL in console window. Run the GEL script from *Scripts->EVM c66##->setConfig_c66##_main*.
  - Open the CCS console window and hit enter to complete the I2C write.
- Booting the image:
  - Disconnect the CCS from board, power off the board.
  - Connect ethernet from board to switch/hub/PC and UART cables from board to PC.
  - Make sure your PC have the IP address specified above.
  - Set the board dip switches to boot from ethernet (TFTP boot) as specified in the hardware setup table (TMDXEVM6678L [6] TMDXEVM6670L [7])
  - Copy the demo image *<MCSDK INSTALL DIR>\demos\image_processing\utils\mad\evmc66##\images\mcip-c66##-le.bin* to tftp directory and change its name to *app.out*
  - Start a tftp server and point it to the tftp directory
  - Power on the board. The image will be downloaded using TFTP to the board and the serial port console should print messages from the demo. This will also print the configured IP address of the board

- Use the IP address to open the demo page in a browser and run the demo

Booting from NOR

- Change IBL configuration: The IBL configuration parameters are provided in a GEL file *<MCSDK INSTALL DIR>\tools\boot_loader\ibl\src\make\bin\i2cConfig.gel*. All the changes needs to be done in the function *setConfig_c66##_main()* of the gel file.
  - Make sure the *norBoot.bootFormat* is set to *ibl_BOOT_FORMAT_BBLOB*
  - Set the *norBoot.blob[0][0].branchAddress* to *0x9e001040* (or *0x80001040* if you are using BIOS MCSDK v 2.0.4 or prior)

menuitem "EVM c66## IBL";

hotmenu setConfig_c66##_main() {

```
ibl.iblMagic = ibl_MAGIC_VALUE;
ibl.iblEvmType = ibl_EVM_C66##L;
```

```
...
```

```
ibl.bootModes[0].bootMode = ibl_BOOT_MODE_NOR;
ibl.bootModes[0].priority = ibl_HIGHEST_PRIORITY;
ibl.bootModes[0].port = 0;
```

```
ibl.bootModes[0].u.norBoot.bootFormat = ibl_BOOT_FORMAT_BBLOB;
ibl.bootModes[0].u.norBoot.bootAddress[0][0] = 0; /* Image 0 NOR offset
 byte address in LE mode */
ibl.bootModes[0].u.norBoot.bootAddress[0][1] = 0xA00000; /* Image 1 NOR
 offset byte address in LE mode */
ibl.bootModes[0].u.norBoot.bootAddress[1][0] = 0; /* Image 0 NOR offset
 byte address in BE mode */
ibl.bootModes[0].u.norBoot.bootAddress[1][1] = 0xA00000; /* Image 1 NOR
 offset byte address in BE mode */
ibl.bootModes[0].u.norBoot.interface = ibl_PMEM_IF_SPI;
ibl.bootModes[0].u.norBoot.blob[0][0].startAddress = 0x9e000000
/*0x80000000 for BIOS MCSDK v2.0.4 or prior*/; /* Image 0 load start
address in LE mode */
ibl.bootModes[0].u.norBoot.blob[0][0].sizeBytes = 0xA00000; /* Image 0
size (10 MB) in LE mode */
ibl.bootModes[0].u.norBoot.blob[0][0].branchAddress = 0x9e001040
/*0x80001040 for BIOS MCSDK v2.0.4 or prior*/; /* Image 0 branch
address after loading in LE mode */


...
```

```
ibl.chkSum = 0;
```

}

- Write IBL configuration:
  - Connect the board using JTAG, power on the board, open CCS, load the target and connect to core 0. Select *Tools->GEL Files* and in the GEL Files window right click and load GEL. Then select and load *<MCSDK INSTALL DIR>\tools\boot_loader\ibl\src\make\bin\i2cConfig.gel*.

  - Load I2C writer *<MCSDK INSTALL DIR>\tools\boot_loader\ibl\src\make\bin\i2cparam_0x51_c66##_le_0x500.out* to Core 0 and run. It will ask to run the GEL in console window. Run the GEL script from *Scripts->EVM c66##->setConfig_c66##_main*
  - Open the CCS console window and hit enter to complete the I2C write
- Write NOR image:
  - Copy application image (*<MCSDK INSTALL DIR>\demos\image_processing\utils\mad\evmc66##l\images\mcip-c66##-le.bin*) to *<MCSDK INSTALL DIR>\tools\writer\nor\evmc66##l\bin\app.bin*
  - Connect the board using JTAG, power on the board, open CCS, load the target and connect to core 0. Make sure the PLL and DDR registers are initialized from the platform GEL (if it is not done automatically, run Global_Default_Setup function from the GEL file). Load image *<MCSDK INSTALL DIR>\tools\writer\nor\evmc66##l\bin\norwriter_evm66##l.out*
  - Open memory window and load the application image (*<MCSDK INSTALL DIR>\demos\image_processing\utils\mad\evmc66##l\images\mcip-c66##-le.bin*) to address *0x80000000*
  - Be sure of your **Type-size** choice 32 bits
  - Hit run for NOR writer to write the image
  - The CCS console will show the write complete message
- Boot from NOR:
  - Disconnect CCS and power off the board
  - Set the board dip switchs to boot from NOR (NOR boot on image 0) as specified in the hardware setup table (TMDXEVM6678L [6] TMDXEVM6670L [7])
  - Connect ethernet cable from board to switch/hub
  - Connect serial cable from board to PC and open a serial port console to view the output
  - Power on the board and the image should be booted from NOR and the console should show bootup messages
  - The demo application will print the IP address in the console
  - Use the IP address to open the demo page in a browser and run the demo

Performance numbers of the demo

The following table compares the performance between OpenMP and explicit IPC based image processing demo applications.

Note: The numbers are based on a **non-DMA** based implementation with **non-optimized RGB to Y** kernel. The L2 cache is set to 256KB.

Note: The results shown were taken during the BIOS-MCSDK 2.1.0 beta, results taken with current component versions may vary.

	Processing time for a ~16MB BMP image (in msec)	
Number of Cores	**OpenMP based demo**	**Explicit IPC based demo**
1	290.906	290.378
2	150.278	149.586
3	101.697	100.75
4	77.147	77.485
5	63.154	63.318
6	54.709	54.663
7	49.144	47.659
8	42.692	42.461

The performance numbers seem to be similar in both cases. This might be due to the nature of the demo application. It spends most of its processing time on actual image processing and sends, at most, 16 IPC messages between cores. So the contribution of the communication delays (IPC vs. OMP/IPC) are very minimal compared to any significant difference in processing times.

References

[1] http://en.wikipedia.org/wiki/Ycbcr
[2] http://en.wikipedia.org/wiki/Sobel_operator
[3] http://processors.wiki.ti.com/index.php/MCSA_and_the_MCSDK_Demo
[4] http://www.criticalblue.com/prism/
[5] http://processors.wiki.ti.com/index.php/MAD_Utils_User_Guide
[6] http://processors.wiki.ti.com/index.php/TMDXEVM6678L_EVM_Hardware_Setup#Boot_Mode_Dip_Switch_Settings
[7] http://processors.wiki.ti.com/index.php/TMDXEVM6670L_EVM_Hardware_Setup#Boot_Mode_Dip_Switch_Settings

Article Sources and Contributors

MCSDK Image Processing Demonstration Guide *Source*: http://processors.wiki.ti.com/index.php?oldid=121431 *Contributors*: A0792105, ChrisRing, Csmith, DanRinkes, Jbtheou, Mdamato, RajSivarajan, Sajeshsaran

Image Sources, Licenses and Contributors

Image:Inputpage.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Inputpage.jpg *License*: unknown *Contributors*: Sajeshsaran

Image:Outputpage.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Outputpage.jpg *License*: unknown *Contributors*: Sajeshsaran

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.
BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

License

1. Definitions

a. "**Adaptation**" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.
b. "**Collection**" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.
c. "**Creative Commons Compatible License**" means a license that is listed at http://creativecommons.org/compatiblelicenses that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.
d. "**Distribute**" means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.
e. "**License Elements**" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.
f. "**Licensor**" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.
g. "**Original Author**" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.
h. "**Work**" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.
i. "**You**" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
j. "**Publicly Perform**" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.
k. "**Reproduce**" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights

Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
b. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";
c. to Distribute and Publicly Perform the Work including as incorporated in Collections; and,
d. to Distribute and Publicly Perform Adaptations.
e. For the avoidance of doubt:

i. **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;
ii. **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,
iii. **Voluntary License Schemes.** The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.
b. You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.
c. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.
d. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.