Integration Examples

Link-chain Framework

This framework is developed (as an example) so that it becomes easier for the individual modules to integrate themselves with others, create a chain of different modules and test them together for a particular use case. It is based on few assumptions and may not be suitable for all possible use cases.

A link is a stand-alone independent task which is created on top of a specific module (like capture, scalar, noise filter etc) and is meant to perform a predefined operation. For e.g. the display link will only take care of the display related functionalities, a noise-filter link will only look towards noise filtering etc.

A chain is a set of links joined together to perform a complete sequence of operations desired for the specific use case. For e.g. capture and display links can be joined to form a chain which will capture frames from the input source and display them on the connected output.

During initialization, each link registers itself with this framework and then creates its own individual task. This task listens to the incoming messages and takes the appropriate actions.

Each link accepts messages from the previous link(s) (the only exception is capture link) and sends output to the connected link(s) (an exception is display). The link is informed by its predecessor link once the data is available for its use. The link can then start processing the data and once it is done, it informs the next link. The next connected link in the chain will then start processing the data and the process continues.

Link-Chains Examples

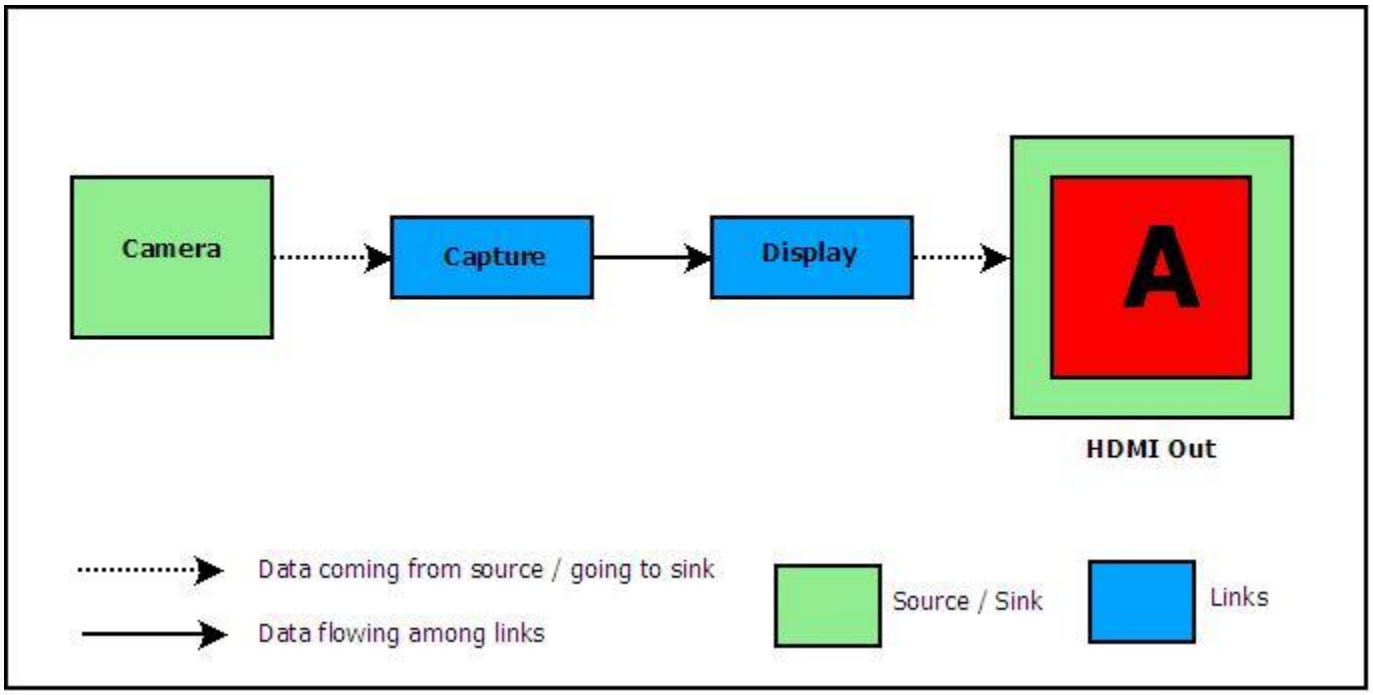
Below are a few examples of chains having different links with detailed diagrams and performing various tasks.

The chains and links implementation can be found in **\packages\ti\psp\examples\common\iss\chains** folder.

Main source file for all chains is: **src\chains_main.c**

NOTE: The examples shown in this section is not a exhaustive list of link-chains that are actually implemented in the ISS chains examples. See next section for list of all links-chains that are implemented

Single-channel Capture + Display



This is the simplest chain which uses capture and display links. Capture link takes single input from the source (camera) and informs the display link once the frames are captured. Display link then configures the attached display (for e.g. HDMI) and starts displaying captured frames from a single channel on the same. Both capture and display links operate in single channel mode here.

Source file: *src\chains_CaptureDisplay.c*

Compiling HDVPSS Drivers

HDVPSS drivers and examples can be built using the *Makefile* present in the HDVPSS installation directory. Before doing so, user may have to modify the*Rules.make* file (present in the installation directory), depending upon his build environment.

Open the *Rules.make* file and make sure that the paths for the following tool-chains are correct (default installation paths for all the required tool-chains can be found in this file):

- **CODEGEN_PATH_M3** := Points to the Codegen toolchain for M3.
- **hdvpss_PATH** := Points to the HDVPSS installation directory.
- **bios_PATH** := Points to the BIOS installation directory.
- **xdc_PATH** := Points to the XDC installation directory.

- **ipc_PATH** := Points to the IPC installation directory.

Important

Make sure that the above mentioned paths don't have white spaces in them. In case the installation directory has any white space, use the short names generated by issuing `dir /X` on the DOS command prompt, which replaces the white spaces by ~. Also, use forward slash '/' instead of back slash '\' in all the above paths.

To build on Linux, an additional step is required to set the `OS` environment variable to `Linux`. Alternatively, it can be passed to the gmake command as indicated in the below steps:

After editing the *Rules.make* file, open a command prompt and `cd` (change directory) to the HDVPSS installation directory.

```
>cd $(HDVPSS_INSTALL_DIR)
```

Provide the command `gmake -s all`. This will clean and recursively build all the libraries and examples for the default platform (ti816x-evm) and default profile (whole_program_debug).

```
>make -s all
```

To build on Linux, if the `OS` environment variable is not set, the gmake command can be invoked as:

```
>gmake -s all OS=Linux
```

If the command prompt can't locate *gmake* command, then add the directory where *gmake* is present in the PATH environmental variable. Typically, *gmake* comes along with CCS/XDC installation and can be found at *$(CCS_INSTALL_DIR)/xdctools_XX_YY_ZZ_WW*.

Note

Default platform and profile can be changed by modifying `PLATFORM` and `PROFILE_$(CORE)` in the *Rules.make* file respectively.

During development, the below *gmake* targets can also be used be used for convenience:

- **gmake -s hdvpss** - incrementally builds only HDVPSS drivers
- **gmake -s examples** - incrementally builds HDVPSS drivers and all examples
- **gmake -s examples_netcam** - incrementally builds HDVPSS drivers and all examples for the netcam/vcam usecase

- **gmake -s clean** - clean all drivers and examples
- **gmake -s examplesclean** - clean all examples ONLY
- **gmake -s *example_name*** - incrementally builds HDVPSS drivers and the specific example ONLY. Values for *example_name* can be - i2c, captureVip, chains, display etc.

Compiling ISS Drivers

ISS drivers and examples can be built using the *Makefile* present in the ISS installation directory. Before doing so, user may have to modify the *Rules.make* file (present in the installation directory), depending upon his build environment.

Open the *Rules.make* file and make sure that the paths for the following tool-chains are correct (default installation paths for all the required tool-chains can be found in this file):

- **CODEGEN_PATH_M3 :=** Points to the Codegen toolchain for M3.
- **iss_PATH :=** Points to ISS installation directory.
- **edma3lld_PATH :=** Points to EDMA installation directory.
- **fc_PATH :=** Points to FrameWork Components installation directory.
- **bios_PATH :=** Points to BIOS installation directory.
- **xdc_PATH :=** Points to XDC installation directory.
- **ipc_PATH :=** Points to IPC installation directory.
- **syslink_PATH :=** Points to SYSLINK installation directory.
- **hdvpss_PATH :=** Points to HDVPSS installation directory.
- **xdais_PATH :=** Points to XDAIS installation directory.

Important

Make sure that the above mentioned paths don't have white spaces in them. In case the installation directory has any white space, use the short names generated by issuing `dir /X` on the DOS command prompt, which replaces the white spaces by ~. Also, use forward slash '/' instead of back slash '\' in all the above paths.

To build on Linux, an additional step is required to set the `OS` environment variable to `Linux`. Alternatively, it can be passed to the gmake command as indicated in the below steps:

After editing the *Rules.make* file, open a command prompt and `cd` (change directory) to the ISS installation directory.

```
>cd $(ISS_INSTALL_DIR)
```

Provide the command *gmake -s all*. This will clean and recursively build all the libraries and examples for the default platform (ti814x-evm) and default profile (whole_program_debug).

```
>gmake -s iss chains
```

To build on Linux, if the *OS* environment variable is not set, the gmake command can be invoked as:

```
>gmake -s all OS=Linux
```

If the command prompt can't locate *gmake* command, then add the directory where *gmake* is present in the PATH environmental variable. Typically, *gmake* comes along with CCS/XDC installation and can be found at *$(CCS_INSTALL_DIR)/xdctools_XX_YY_ZZ_WW*.

Note

Default platform and profile can be changed by modifying `PLATFORM` and `PROFILE_$(CORE)` in the *Rules.make* file respectively.

During development, the below *gmake* targets can also be used be used for convenience:

- **gmake -s iss** - incrementally builds only ISS drivers
- **gmake -s chains**- incrementally builds ISS drivers and chains example
- **gmake -s clean** - clean all drivers and examples
- **gmake -s chains_clean**- clean chains example ONLY

Running ISS Application TI814x

This section describes the out of the box experience for the ISS drivers. It shows how to run the ISS application involving ISS drivers.

Demo application on VCAM board

Following are the application details which is running on the VCAM application board:

- Capture Display for MT9J003 Sensor.

Steps to run the Demo on VCAM Board

Steps for examples

- Switch on the board.(No Boot Mode(Sw4 all to 0) or at u-boot TI_MIN)
- Open the CCS
- Load gel file ***IPNC_A8_DDR3.gel*** under directory *$ISS_INSTALL_DIR/utils/ti814x* for A8 processor.
- connect to TI814x (CortexA8) using the CCS and debugger. This will run the script and enable the DDR, Ducati, HDMI, ISS and DSS.
- Load gel file ***IPNC_Ducati.gel*** file in ISS_M3 processor under directory *$ISS_INSTALL_DIR/utils/TI814x*.
- Connect to CortexM3_ISS this will run the script and enables the cache on Ducati.
- Connect the HDMI output of the VCAM board to the HDMI input of the TV.
- Load and Run $ISS_INSTALL_DIR/ /build/iss_examples_chains/bin/ti814x-evm/iss_examples_chains_m3vpss_debug.xem3
- Once you are at prompt Switch to CortexA8and load and run TI814x_A8_HDMI_Sample.out found under *$HDVPSS_INSTALL_DIR/docs/ti814x* . You will get below output on console

```
Do you want color bars? [Y/N]
Y

 1 for 720p60
 2 for 1080p30
 3 for 1080i60
 4 for 1080p60
Please choose the resolution
4
 Configured for 1080 P 60
HDMI Clk enable in progress
HDMI Clocks enabled successfully
Started HDMI
To stop ->1, To Restart ->2, To Read EDID ->3, Cloe Re Open ->4,
To Exit -> 90
```

- Go to Scripts and run ***IPNC_A8_DDR3.gel > Centaurus2 INDIVIDUAL PLL Config > HDMI_PLL_Config_1_485_GHz***
- Now switch to CortexM3_ISS and Select option 1 at " Enter Choice:" option.
- You can see 1080P60 output on Display.