i.MX61 IPU Application Notes

by Yanfei Sun   
Multimedia Applications Division  
Freescale Semiconductor, Inc.  
Shanghai, China

Contents

[1 System Overview 2](#_Toc302314625)

[2 IPU Task Management 4](#_Toc302314626)

[2.1 Image Rendering 5](#_Toc302314627)

[2.1.1 IDMAC 5](#_Toc302314628)

[2.1.2 DMFC 5](#_Toc302314629)

[2.1.3 DP 6](#_Toc302314630)

[2.1.4 DC 7](#_Toc302314631)

[2.1.5 DI 7](#_Toc302314632)

[2.2 Image Processing 7](#_Toc302314633)

[2.3 CSI Preview 9](#_Toc302314634)

[2.4 CSI Capture 12](#_Toc302314635)

[3 Use Cases 13](#_Toc302314636)

[3.1 Image Rendering Instance 13](#_Toc302314637)

[3.1.1 IOMUX Settings 13](#_Toc302314638)

[3.1.2 Clock Settings 14](#_Toc302314639)

[3.1.3 IPU Hardware Configuration 15](#_Toc302314640)

[Appendix A References 20](#_Toc302314641)

# System Overview

The IPU is a part of the video and graphics subsystem in an application processor. The goal of the IPU is to provide comprehensive support for the flow of data from an image sensor and/or storage to a display device.

This support covers all aspects of these activities:

* Connectivity to relevant devices - displays, graphics accelerators, TV encoders
* Related image processing and manipulation: sensor image signal processing, display processing, image conversions, etc.
* Synchronization and control capabilities (to avoid tearing artifacts).

This integrative approach leads to several significant advantages:

* Automation: The involvement of the ARM platform in image management is minimized. In particular, display refresh/update can be performed completely autonomously. The resulting benefits are reducing the overhead due to SW-HW synchronization, freeing the ARM platform to perform other tasks and reduced power consumption (when the ARM core is idle and can be powered down).
* Optimal data path: Access to system memory is minimized. In particular, significant processing can be performed on-the-fly while sending data to a display. System memory is used essentially only when a change in pixel order or frame rate is needed. The resulting benefits are reduced load on the system bus and further reduction of power consumption.
* Resource sharing: Maximal HW reuse for different applications, resulting with the support of a wide range of requirements with minimal HW.

The HW reuse mentioned above is enabled by a sophisticated configurability of each HW block. This configurability also allows the support of a wide range of external devices, data formats and operation modes. The resulting flexibility is important also because the support requirements are evolving significantly, so expected future changes need to be anticipated and accounted for.

There are 2 equivalent instances of IPU in i.MX61. They are located in memory map of i.MX61 at addresses:

IPU1 base address – 0x02400000

IPU2 base address – 0x02800000

A simple block diagram of IPU is given in figure 1-1.

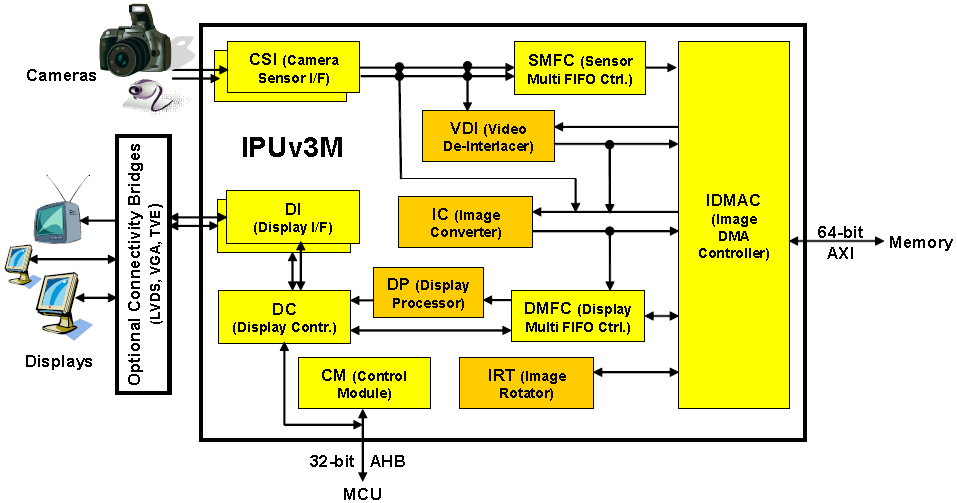


Figure 1-1 IPU block diagram

The role of each block is decribed in table 1-1

Table 1-1 IPU block description

|  |  |
| --- | --- |
| Block | Description |
| CSI - Camera Sensor Interface | Controls a camera port; provides interface to an image sensor or a related device. Each IPU includes 2 such blocks |
| DI - Display Interface | Provides interface to displays, display controllers and related devices. Each IPU includes 2 such blocks |
| DC - Display Controller | Controls the display ports. |
| IC - Image Converter | Performs resizing, color conversion/correction, combining with graphics, and horizontal inversion |
| DP - Display Processor | Performs the processing required for data sent to display. |
| IRT - Image Rotator | Performs rotation (90 or 180 degrees) and inversion (vertical/horizontal). |
| IDMAC - Image DMA Controller | Controls the memory port; transfers data to/from system memory. |
| DMFC - Display Multi FIFO Controller | Controls FIFOs for IDMAC channels related to the display system. |
| SMFC - Sensor Multi FIFO Controller | Controls FIFO’s for output from the CSI’s to system memory |
| VDIC – Video De-Interlaced and Combiner | Convert interlaced image into progressive and layers combination |
| CM - Control Module | Provides control and synchronization. |

# IPU Task Management

the detailed IPU diagram is shown as figure 2-1.

Figure 2-1 detailed IPU block diagram

The IPU tasks can be categorized into five kinds:

1. Image rendering. The data flow is from memory to display.
2. Image processing. The data flow is from memory to memory.
3. CSI Preview. The data flow is from CSI to display. A direct path without memory involved.
4. CSI Capture. The data flow is from CSI to memory.
5. Mixed mode. It can be combinations of two or several of above.

In the tasks 1&2, the image is provided by external devices, such as sensor, DVD player, etc. CSI is the interface between them and IPU.

In this document, we will focus on the image rendering flow. Others will be included later on.

## Image Rendering

Image rendering means put the image data stored in the memory to display device. The display could be parallel dumb panel, smart panel, or other further processed sinks, such as HDMI/DVI monitor(imx61 provide the HDMI/DVI transmitter&PHY convert the data into serialized differentiated data lanes.), LVDS panels(imx61 provide LDB as a bridge to LVDS display).

A simple display flow could be memory->IDMAC->DMFC(->DP)->DC->DI->display.

### IDMAC

IDMAC is the DMA bridge between external memory and IPU blocks. There are total 64 DMA channel inside IPUv3, and each channel is dedicated as read/write channel to/from memory. The detailed channel description can be found in the IPU spec.

The configuration parameters for each IDMAC channel are held in the CPMEM. For each channel there are two mega-words to describe the properties of the it. Each mega-word is 160 bits wide. It includes the information such as data format, frame width and height, burst size, stride line, bit per pixel setting, etc.

IDMAC can support interleaved mode and non-interleaved mode data transfer. No matter what the data format stored in the memory is, IDMAC will pack(in write direction) or unpack(in read direction) it. That means all the data flow through IDMAC to other blocks of IPU would be YUVA4444 or RGBA8888 mode.

There are several IDMAC events/interrupts for system control and debug purpose. The most import of them is EOF(end of frame) and NF(new frame). These two events are usually used to indicate the frame status and drive the whole flow.

### DMFC

The Display Multi Fifo Control manages Multi channels FIFOs. The DMFC serves the following clients

* IDMAC - both read and write
* DP - read only
* DC - both read and write
* IC - write only
* AHB - both read and write

The DP and the DC read channels are physically attached to an IDMAC or an IC channel. As the IC has only one output channel connected to the DMFC. When the input is coming from the IC it replaces a channel that was physically attached to the IDMAC. The DMFC uses a single physical memory that serves the DP and DC read channels. The AHB accesses to the DC and the DC’s write channel (read from display) use a separate physical memory. This is used to write an external device directly through AHB bus, or to configure a smart panel.

In image rendering, DMFC is served as FIFO between IDMAC (fetching data from external memory) or IPU sub blocks (such as IC, DP, DC). The physical memory of DMFC is partitioned to 8 segments. For each channel the user has to define the start address at a segment’s boundary using the DMFC\_ST\_ADDR parameter and the size of the FIFO allocated to a channel by the DMFC\_FIFO\_SIZE parameter. The user must allocate the FIFO and avoid overlapping between FIFOs. Besides, The FIFO’s burst length is also configurable, and it should match with the IDMAC burst length in order to get the best performance.

There is watermark setting to dynamically tune the channel’s priority on the IDMAC’s arbitration. DMFC\_WM\_SET and DMFC\_WM\_CLR are used to trigger the watermark signals.

### DP

Each IPU can support two synchronous display flows concurrently. One is through DP BG/FG, the other is through DC block.

The DP processes the image prior to sending it to the display. The main task performed by the DP is combining between 2 planes. The DP has 2 input FIFOs holding the data of the full plane and the partial plane. The two planes can be blended as per local or global alpha setting, the mode is chosen by DP\_GWAM\_SYNC. For global alpha, the alpha value is configured in DP register DP\_GWAV\_SYNC. In addition the DP performs some image enhancement functions like gamma correction, Color space conversion including Gamut mapping.

In combining, the background is a full plane, and the foreground is a partial plane. Left and top offsets of the foreground can be set in register DP\_FG\_POS\_SYNC, size of the foreground is determined by corresponding IDMAC descriptor.

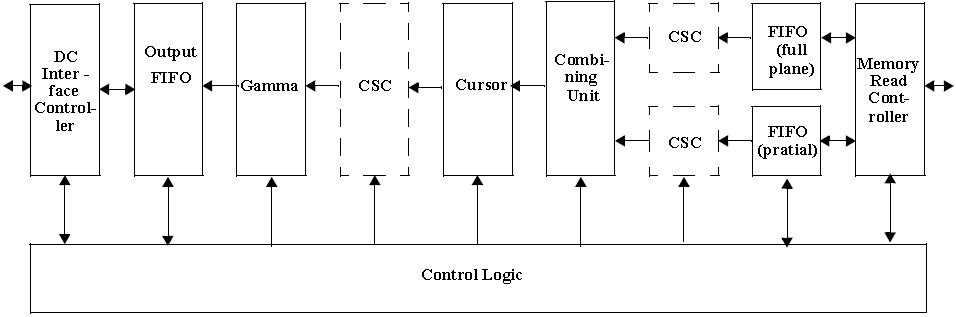


Figure 2-2 Data Processor architecture diagram

The diagram of DP is given is Figure 2-2.

The combination task can also be done in IC block, but there is limitation that the size of the two planes must be the same, and the performance of combining is higher in DP so it’s the first choice for two-layer blending.

Note that the register of DP cannot be accessed directly. For example, if you want to configure the register of DP\_COM\_CONF\_SYNC, you have to access its shadow register SRM\_DP\_COM\_CONF\_SYNC. DP\_S\_SRM\_MODE setting will indicate how the changes in shadow registers would be updated in actual registers.

### DC

The DC controls the flows coming to and from the DI port. The DC manages the flows, decides which flows are currently active and when each flow is activated. The DC arbitrates between the active flows, gets the data from the predefined source and distribute it to the correct DI.

The DC’s core is the microcode. The microcode contains a set of routine. A routine is built of a set of commands stored in the template’s (microcode) memory. For each event (like new frame, end of frame etc.) a specific routine is executed. The users write the routines to describe the rules of processing, and then map them to specific events. The routine contains instructions to the DC about the way of handling the data/address/commands associated with the display. The routine may contain information about the data’s mapping, about waveform’s characteristics, and more.

In the DC block, the data coming from IDMAC is linked to a display interface. It will also set the interface format(parallel or serial, interlaced or progressive, etc.), to which display the flow is attached, based on which waveform of the DI that the data will be processed, how to map the data to the sink device. Through DC the rendered image data is finally sent to the DI.

### DI

The DI provides arbitrates access to up to three displays with time multiplexing. It converts a data from the DC or the MCU (low level access for serial interface only) to a format suitable for the specific display interface. The DI generates display clocks and other display control signals such as HSYNC, VSYNC and DRDY with programmable timings. The DI outputs data to or inputs from parallel and/or serial interfaces.

This module generates all the control signals sent to the display. The DC sends to the DI; the data for the display and a set of control signals. The controls coming from the DC are used in order to generate the control signals sent to the display. One exception is serial low level access (LLA) where the DC is bypassed and the data is directly coming from the MCU.

DI also set the attributes of the interfaces to the display. According the different types of display, the timing and polarity of signals would be set in the DI block.

## Image Processing

Image processing will perform resizing, rotation, color space conversion, multi-layer combination with alpha blending, de-interlaced, gamma correction, gamut mapping, etc.

The main image processing block is IC and VDIC.

The IC contains three processing sections: downsizing, main processing and rotation. The module is controlled via the peripheral bus registers. Some processing parameters should be written by the MCU to the Task Parameter Memory. Writing to the memory is performed via the AHB bus.

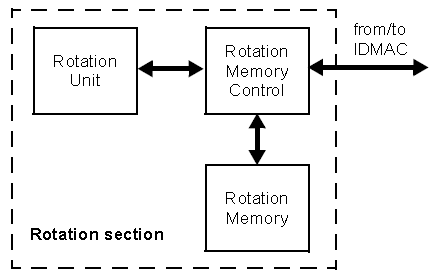
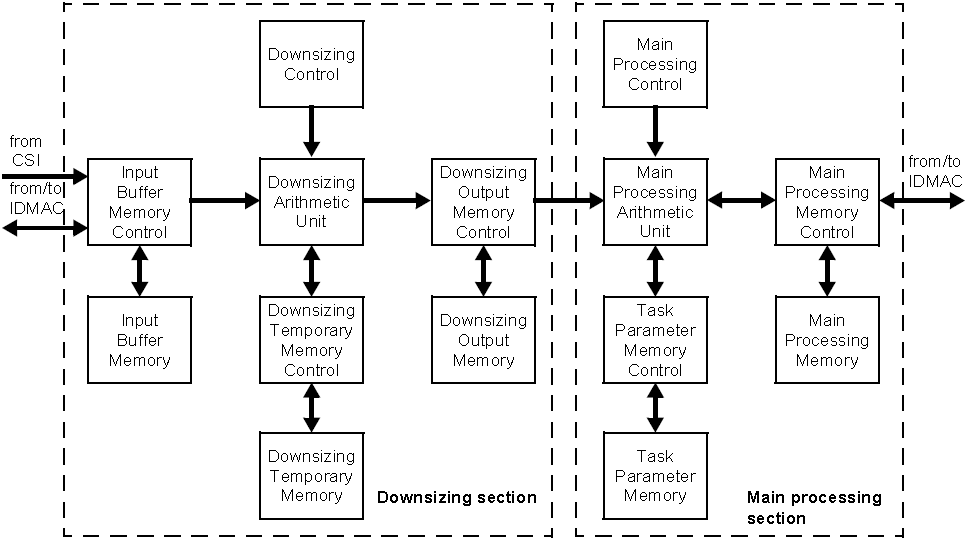


Figure 2-3 diagram of image converter

IC has three processing sections perform up to three processing tasks with time sharing mode that means three sets of configuration can be set at the same time but they share the unique set of hardware accelerators. For post processing task, it has dedicated input and output channel. But for preprocessing tasks, encoder and viewfinder share the same input but have their own separate output channels.

* Downsizing.

In this block, IC performs 1x, 2x, 4x downsizing operations on the input image. The downsizing ratio can be set in DS\_R\_H for horizontal or DS\_R\_V for vertical.

* Main processing

The main processing block reads the data from Downsizing output and is able to perform the following operations in each task:

* Horizontal/Vertical Flip by HF/VF settings. The corresponding DMA channel descriptor should be changed accordingly.
* Horizontal/Vertical Resizing by bilinear interpolation. There is a formula to calculate the resizing ration. Resizing\_ratio = floor(2^13\*(SI-1)/(SO-1)), where SI means the input size and SO means the output size. In Resizing block, the output should be no more than 1024 in horizontal due to the FIFO width limitation.
* Color Space Conversion. The conversion matrix is user configurable, and it can support SAT\_MODE and NON\_ SAT\_MODE. In SAT\_MODE, the range of Y is [0, 235], range of U/V is [16, 240]. In NON\_SAT\_MODE, the range of Y/U/V are all [0, 255].
* Combination. IC can support local alpha blending, global alpha blending, and use of key color. The size of the two layers for combining must be the same.
* Rotation

Rotation is done by IC and IDMAC together. The image for rotation is dived into 8\*8 blocks, the IDMAC must work in block mode and perform data rearrangement within the blocks. IC will provide proper rotation of the whole frame in block unit.

The Video De-interlace and Combination block (VDIC) can de-interlace standard interlaced video to progressive video that is used for upsizing to HD formats or for display on progressive displays. For VDI operation three sequential fields are necessary F(n-1),F(n),F(n+1). There is some per-designed de-interlace algorithm stored in the VDIC block as firmware. By setting the motion level(high-motion or low-motion) the de-interlace will be done and output a progressive whole frame.

The VDIC can also perform on-the-fly combination and color keying. The position and size of the foreground layer are configurable.

## CSI Preview

Image preview is a direct path from CSI to display. The CSI gets data from the sensor, synchronizes the data and the control signals to the IPU clock (HSP\_CLK), and transfer it according to configuration of DATA\_DEST register to one or more of the following: IC or SMFC. When data is transferred to the IC module then routed to display module, it is called image preview.

CSI supports two types of interfaces. The interface is determined via the DATA\_SOURCE register.

* parallel interface

In this mode, a single value arrives in each clock, except when working in BT.1120 mode, in which two values arrive in each cycle. Each value can be 8-16 bit wide according to configuration of DATA\_WIDTH. If DATA\_WIDTH is configured to N, then 20-N LSB bits are ignored.

CSI can work with several data formats according to SENS\_DATA\_FORMAT configuration. In case the data format is YUV, the output of the CSI is always YUV444 (even if the data arrives in YUV422 format).

The polarity of the inputs can be configured using the registers SENS\_PIX\_CLK\_POL, DATA\_POL, HSYNC\_POL and VSYNC\_POL.

* High-speed serial interface - MIPI (Mobile Industry Processor Interface).

In MIPI interface two values arrive in each cycle. Each value is 8 bit wide, which means 16 MSB bits of the data bus input are treated, while 4 LSB bits are ignored.

When working in this mode, the CSI can handle up to 4 streams of data. Each stream is identified with DI (data identifier) that includes the virtual channel and the data type of this stream. Each stream that is handled is defined in registers MIPI\_DI0-3. Only the main stream (MIPI\_DI0) can be sent to all destination units while the other streams are sent only to the SMFC as generic data.

In this mode SENS\_DATA\_FORMAT and DATA\_WIDTH registers are ignored, since this information is coming to the CSI via the MCT\_DI bus.

CSI can work in several timing/data mode protocols according to SENS\_PRTCL configuration.

* Gated mode

In this mode, VSYNC is used to indicate beginning of a frame, and HSYNC is used to indicate beginning of a raw. Sensor clock is ticking all the time.

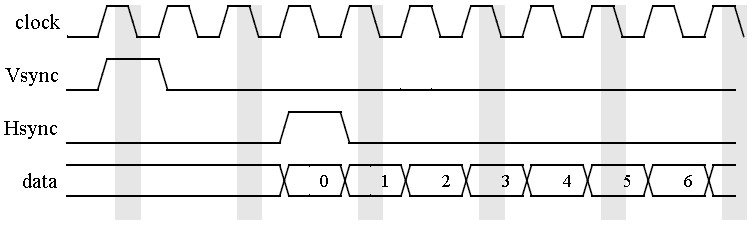


Figure 2-4 CSI gated mode

* Non-gated mode

In this mode, In this mode VSYNC is used to indicate beginning of a frame. Sensor clock is ticking only when data is valid. HSYNC is not used.

When working with MIPI, the non-gated mode should be configured.

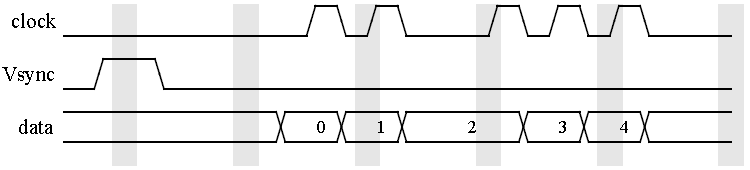


Figure 2-5 CSI non-gated mode

* BT656 mode

BT656 describes a simple digital video protocol for streaming uncompressed PAL or NTSC Standard Definition TV (525 or 625 lines) signals. The protocol builds upon the 4:2:2 digital video encoding parameters which provides interlaced video data, streaming each field separately, and uses the YCbCr color space and a 13.5 MHz sampling frequency for pixels.

The timing reference signals (frame start, frame end, line start, line end) are embedded in the data bus input. Each timing reference signal consists of a four word sequence. The first three words are fixed and configured in the CCIR\_PRECOM register. The fourth word contains information defining field, the state of field blanking and the state of line blanking. These states are configured in registers CCIR\_CODE\_1 (for field 0) and CCIR\_CODE\_2 (for field 1).

For example, for PAL mode, the CCIR\_CODE could be configured as below:

CCIR\_CODE\_1: 0xd07df

CCIR\_CODE\_2: 0x40596

CCIR\_CODE\_3: 0xff0000

In BT656 mode in each cycle one value of data arrives.

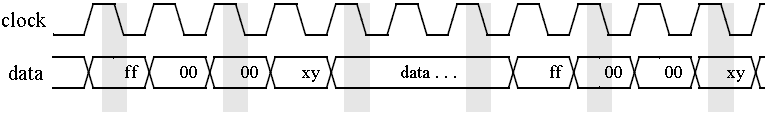


Figure 2-6 BT656 mode

* BT1120 mode

In this mode, CSI can work in SDR or DDR mode.

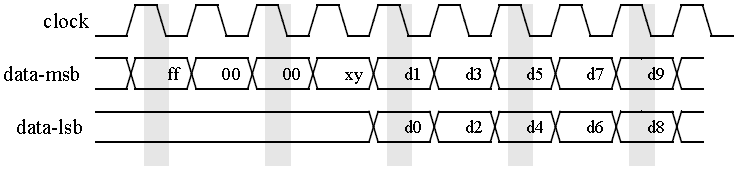


Figure 2-7 BT1120 SDR mode

In DDR mode, data will arrive on both rising and falling edge of a clock, that mean in each clock period, two values of data arrive.

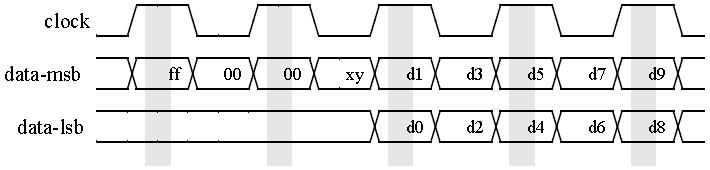


Figure 2 8 BT1120 DDR mode

For direct path from CSI to IC, CSI\_MEM\_WR\_EN and RWS\_EN located in IPU\_IC\_CONF will be used to choose the data flow. CSI\_SEL in IPU\_CONF will determine which CSI is selected as the direct input to IC block. There is limitation in this task, the refresh rate of the display device must be the same with the CSI input frame rate, or else the screen may be not functional due to the frame rate dismatch.

Beside, In the CSI block, image crop can be done by setting the actual window size. Pay attention that below rules must be followed.

SENS\_FRM\_HEIGHT >= VSC + ACT\_FRM\_HEIGHT

SENS\_FRM\_WIDTH >= HSC + ACT\_FRM\_WIDTH

## CSI Capture

In CSI capture task, data will be got from sensor and output to memory through SMFC and IDMAC.

The SMFC(Sensor Multi FIFO Controller) is used as buffer between CSI and IDMAC. The two masters (CSIs) can be connected to SMFC. Both masters can be active simultaneously.

There are four channels can be used as CSI output channel, they are channel 0~3 of the IPU DMA channels. The frame from CSI can be mapped to one of four IDMAC channels via SMFC mapping registers. Each DMA channel has dedicated FIFO. And the burst length of FIFO must match with the DMA settings. The FIFO size attached to each DMA channel is flexible according to the number of channel required. All the four channels share the whole FIFO, and if only one of them is enabled, the entire FIFO can be allocated to the channel.

## Mixed Task

The mixed task could be a collection of above. For example, a CSI captured image can be stored in memory and then resize to full screen for display.

For a complex task, in order to support automatically controlled without CPU involved, CM is used for the flow management. After the different blocks are connected together by CM, the flow will be auto-driven by internal events, such as NF, EOF, etc.

There are five registers for CM to configure.

* IPU\_FS\_PROC\_FLOW1
* IPU\_FS\_PROC\_FLOW2
* IPU\_FS\_PROC\_FLOW3
* IPU\_FS\_DISP\_FLOW1
* IPU\_FS\_DISP\_FLOW2

The first three are used to set the processing tasks, and the last two are used for setting the display flows. For each task, the source and destination must be configured to form a round linkage between blocks.

# Clocks

IPU work clock can be generated by external PLL or internal clock divider. On iMx61, the default main clock of IPU is 264MHz. For example, to drive a display of XGA resolution, we need a 65MHz pixel clock. There are two ways to get the clock.

* Divided from internal clock. We can get the 65MHz pclk by dividing the 264MHz IPU HSP clock by 4. IPU can also support fractional division, but usually we don’t need that precise clock for image rendering. The clock source of DI must set to be internal by configuring DI\_CLK\_EXT to be 0.

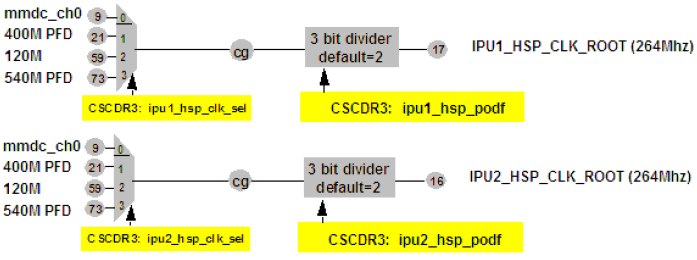


Figure 3-1 clock tree for IPU HSP clock

* Get the clock from external PLLs. Below is the chart for IPU DI clock generating from PLL/CCM.

The clock tree only works when the DI\_CLK\_EXT is set to 1, which means the clock is generated externally.

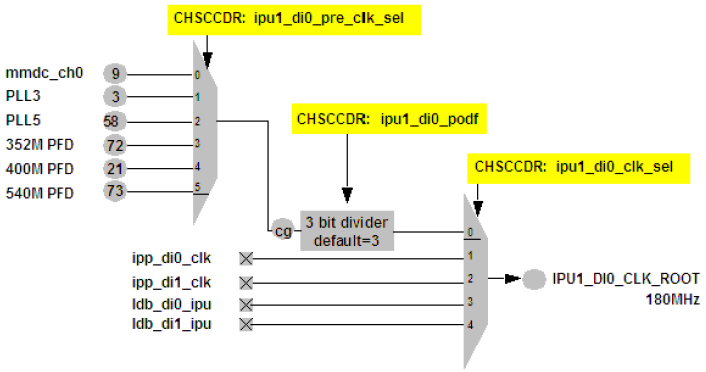


Figure 3-2 clock tree for IPU DI0 clock

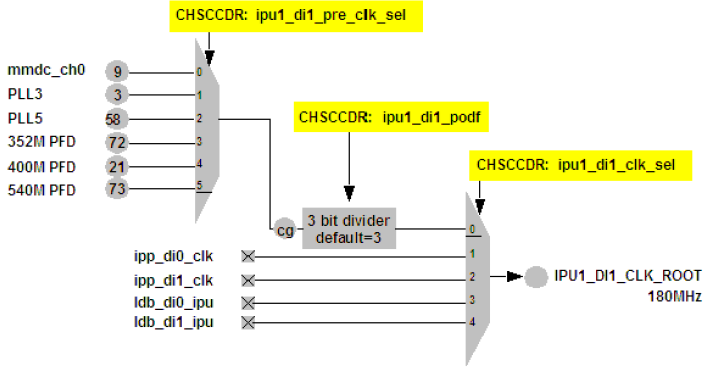


Figure 3-3 clock tree for IPU DI1 clock

# IOMUX Setting

IPU has two sets of display interface to the outside world. For a parallel display, IPU provide data lanes, vsync, hsync, date ready and pixel clock to drive the panel. For other further-processed display, such as HDMI or LVDS, the IPU output signals are internally muxed to the relative modules.

For example, to connect the LVDS with IPU, we can configure LVDSx\_MUX\_CTL as below:

* 00 – IPU1 DI0, connect LVDSx to IPU1 DI0. x means 0 or 1, there are two sets of LVDS display interfaces.
* 01 – IPU1 DI1
* 10 – IPU2 DI0
* 11 – IPU2 DI1

To connect HDMI with IPU, we can configure HDMI\_MUXCTRL as:

* 00 – IPU1 DI0, connect HDMI to IPU1 DI0.
* 01 – IPU1 DI1
* 10 – IPU2 DI0
* 11 – IPU2 DI1

For parallel displays, the output signals of IPU must be set according to the schematic together with SoC datasheet. Sync signals are provided by general IPU display waveform pins, and they must match with the waveform settings in DI block.

Table 4-1 shows a typical IOMUX mapping for IPU parallel panel through DI0.

Table 4-1 Typical IOMUX mapping for IPU parallel panel through DI0 (depends on board)

|  |  |  |  |
| --- | --- | --- | --- |
| **Signals** | **Driver** | | |
| **PAD** | **MUX** | **SION** |
| DI0 display clock | DI0\_DISP\_CLK | ALT0 | 0 |
| DRDY | DI0\_PIN15 | ALT0 | 0 |
| HSYNC | DI0\_PIN2 | ALT0 | 0 |
| VSYNC | DI0\_PIN3 | ALT0 | 0 |
| DI0 data0~23 | DISP0\_DATx | ALT0 | 0 |

# Use Cases

This section describes how to program I2C controller registers I2CR, I2SR and I2DR in transferring data on I2C bus. Pseudo code is provided wherever necessary.

## Single Image Rendering Instance

Image rendering(single image display) is the basic use case done in the IPU. The example shown below will give a general introduction of how the IPU is configured to show an RGB image on the screen.

### IPU Process Flow

DMFC

DP

DC

DI

Memory

IDMAC

Figure 5-1 IPU Process for Single Image Rendering

As described in section 2.1, the display flow consists of several blocks to cooperate together. Figure 5-1 shows the IPU process for displaying a single image to screen from memory. All the information of input and output should be known before we start setting the hardware registers.

### Hardware Configuration

The configuration includes the following steps:

1. Configure the IPU DMA channel.

In this step, API ipu\_disp\_bg\_idmac\_config () is called. Since the DP BG path is chosen for display, we have no other choice but the channel 23 as the DMA channel to fetch data from memory.

The input data format is interleaved RGB565, so the relative bit fields must be set as the following:

* Bpp = 0x3, which means bit per pixel is 16.
* Pfs, which indicates the data format to be interleaved RGB mode.
* Wid0=5-1, off0=0;

Wid1=6-1, off1=5;

Wid2=5-1, off2=11;

Wid3=0, off3=16.

15

12 11

0

5 4

R

G

B

Wid is the actual width of the component subtracting 1. Off means the start address of the component within the pixel.

* FW, which is the actual frame width - 1
* FH , which is the actual frame width – 1
* Stride line, which means the offset of the next line in bytes.

For IPU DMA channel, it can support single buffer mode or double buffer mode by setting the MOD\_SEL bit of each channel. In double buffer mode, the channel will fetch data from EBA0 and EBA1 alternately.

The DMFC is configured for channel 23. In order to support multiple displays, the FIFO is equally split by the DP and DC synchronous display channel.

1. Configure the DP block

DP is the data processor for image combination, color space conversion, gamma correction and gamut mapping. In this use case, we have only one layer and the input/output are all in RGB mode, so the data flow through the DP is bypassed with no any processing.

1. Configure the DC block

In this block, API ipu\_dc\_config() is called. Three microcodes are created including New Data, New Line and End of Line. These three events are synchronized with the DI waveform which generates the active data by setting the sync field of the microcode.

Mapping unit in DC block is used to pack the data output from DC to DI to the data format that the display device can support. For example, if the display can accept RGB666 mode, we need to pack the RGBA8888 data flow into RGB666 format.(As we described in IDMAC chapter, all data flow through the sub blocks of IPU is YUVA4444 or RGBA8888 unpacked by IDMAC block). This operation is done in ipu\_dc\_map().

There are 3 sets of data mapping unit. Which set is chosen is determined by the mapping bit field of the microcode. Finally the microcode is written into an space in template memory by ipu\_dc\_microcode\_config(), and then attached to the event by ipu\_dc\_microcode\_event(). The event priority can be set individually.

The DC block also provide some connection information between DI and DC. ipu\_dc\_display\_config() and ipu\_dc\_write\_channel\_config() are used to determine that to which DI will the DC be connected, which format is the display interface, what the data width is, and to which port the display is selected.

1. Configure the DI block

DI block is the interface of IPU to the display panels or other display processing modules. In this block, the timing to display is generated by the general waveform sets inside DI block.

For a parallel panel, IPU needs to provide pixel clock, HSYNC, VSYNC, DRDY and data lines. As described, pixel clock can be generated internally or externally. And in external mode, the pixel clock is always equal to the di\_clk\_root shown in Figure 3-2.

For a waveform generator, it has several parameters to generate a proper signal.

* syncWaveformGen.runValue , it means the number of periods based on the reference clock.
* syncWaveformGen.runResolution, it indicates the reference clock for the waveform generator, it will trigger the counter to decrease.
* syncWaveformGen.offsetValue, predefined offset in the unit of offsetResolution.
* syncWaveformGen.offsetResolution , offset reference clock.
* syncWaveformGen.cntAutoReload. in auto-reload mode, the counter will reload the predefined value(runValue) when the counter decrease to zero.
* syncWaveformGen.stepRepeat , this parameter is valid only in non auto-reload mode. When the counter is decreased to zero it will reload the value defined in stepRepeat.
* syncWaveformGen.cntClrSel, it is sourc to clear the non auto-reload waveform counter.
* syncWaveformGen.cntPolarityGenEn = 0;
* syncWaveformGen.cntPolarityTrigSel = 0;
* syncWaveformGen.cntPolarityClrSel = 0;

the above three parameters are used to set/clear the polarity of the waveform

* syncWaveformGen.cntUp = 0;
* syncWaveformGen.cntDown = 2;

cntUp and cntDown are used to indicate the rising and falling edge of the waveform.

Some key parameters are used to specify the timing of the display. Below is an example for how to set them.



Figure 5-3 vertical time of display

In vertical, there is blanking time VBL between two vsync active periods. VBL can be divided into three parts:

* VFP, vertical front porch
* VBP, vertical back porch
* VSYNC, sync width in vertical

In the code, vSyncStartWidth indicates the start width of blanking in a whole vsync period, and vSyncEndWidth means the end width of blanking in a whole vsync period.



Figure 5-4 horizontal timing of display

Figure 5-4 shows the horizontal timing of the dislay.

In horizontal, there is blanking time HBL between two hsync active periods. HBL can also be divided into three parts:

* HFP, horizontal front porch
* HBP, horizontal back porch
* HSYNC, sync width in horizontal

In the code, hSyncStartWidth indicates the start width of blanking in a whole hsync period, and hSyncEndWidth means the end width of blanking in a whole hsync period.

DE(or named DRDY) has the same frequency with HSYNC, but the active period of DE indicates data lines are active at that period.

Based on the timing diagram, the parameters are configured as below:

hSyncStartWidth = HSYNC + HBP;

hSyncWidth = HSYNC;

hSyncEndWidth = HFP;

delayH2V = tHV;

vSyncStartWidth = VSYNC + VBP;

vSyncWidth = VSYNC;

vSyncEndWidth = VFP;

hDisp = HDISP;

vDisp = VDISP;

hDisp and vDisp means the frame width and height of the screen.

All the waveforms in the DI block are for general usage. Some of them are used for internal logic, and some of them are used as output signals. The output pins are determined by the schematic design, and DI must bind the pins to output by setting VSYNC\_SEL and DISP\_Y\_SEL in the ipu\_di\_interface\_set() function.

Besides, the polarity of each output signals can also be configured in the ipu\_di\_interface\_set().

1. Enable the blocks involved in the display flow.

This is the last step of hardware settings. In the display flow, we need IDMAC, DMFC, DP, DC, and DI. All these sub blocks can be selected in IPU\_CONF register.

## Image Combining Instance

The image combining use case performs combining between the full and partial planes. Each one of the planes may be graphics or video plane. Fig5-5 shows the planes displayed on a display. The partial plane’s position is defined relatively to the upper left corner of the full plane. The size of the partial and full planes is defined on the corresponding IDMAC’s channels’ FW and FH parameters. The cursor position and parameters are set in the DP\_CUR\_POS register.

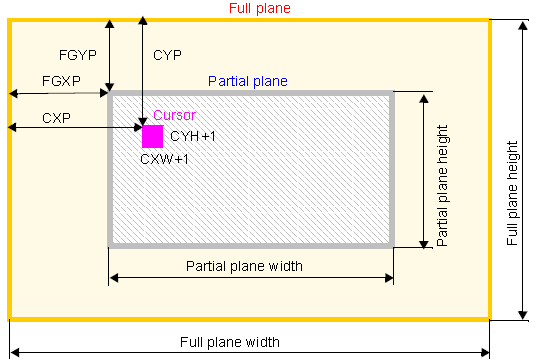


Fig 5-5 Display planes

### IPU Process Flow

Figure 5-6 shows the IPU process for displaying two combined images to screen from two separated memories. The background image is sent to its DMFC through IDMAC main plane channel. The foreground image is send to its DMFC through IDMC auxiliary plane. The combining options are set in DP module. Compared with the single image rendering (Fig 3-1), IPU hardware configuration is different in IDMAC, DMFC and DP modules.

DMFC

DP

DC

DI

Foreground

Memory

Background

Memory

IDMAC

main plane

IDMAC

auxiliary plane

DMFC

Figure 5-6 IPU Process for Image Combining

### IPU Configuration

1. Configure IPU DMA channel

Table 5-1 lists the memory to display main plane and auxiliary plane channels. This use case calls API ipu\_disp\_bg\_idmac\_config() to configure channel #23 for main plane and ipu\_disp\_fg\_idmac\_config() to configure channel #27 for auxiliary plane. This use case uses global alpha. If using local alpha, channel #31 should also be configured.

Table 5-1 Channel for combiner setting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Channel#** | **Source** | **Destination** | **Purpose** | **Data type** |
| 23 | Fmem | DP | DP primary flow – main plane | Pixel |
| 27 | Fmem | DP | DP primary flow – auxiliary plane | Pixel |
| 31 | Fmem | DP | Transparency (alpha for channel 27) | Generic |
| 24 | Fmem | DP | DP secondary flow – main plane | Pixel |
| 29 | Fmem | DP | DP secondary flow – auxiliary plane | Pixel |
| 33 | Fmem | DP | Transparency (alpha for channel 29) | Generic |

Please refer to 5.1.2 part 1 for the relative bit fields’ setting for each channel.

1. Allocate DMFC

Allocate DMFC for both main plane (background) and auxiliary plane (foreground) IDMA channels.

1. Configure DP module

DP module could set some combining options:

* local alpha blending
* global alpha blending
* use of key color
* order of the planes (full is presented over the partial plane and vice versa)

The relative bit fields for combining show as following:

* DP\_FGXP\_SYNC / DP\_FGYP\_SYNC set the left upper corner position for foreground on display on screen.
* DP\_FG\_EN\_SYNC must be set 1 to enable the partial plane channel.
* DP\_GWAM\_SYNC selects the use of alpha to be global or local.
  + 1 Global Alpha.
  + 0 Local Alpha.
* DP\_GWAV\_SYNC defines the global alpha value of background (main plane).

1. Other modules

The settings are the same as those for corresponding modules stated in 5.1.2.

## Image Rotate Instance

### IPU Process Flow

Figure 5-7 shows the IPU process for rotating an image and displaying it on a screen. Rotation is performed by the IDMAC and the Rotation unit inside the IC. The frame is partitioned into 8X8 pixels blocks. The IC reorders the pixels within a block. The IDMAC reorders the block.

Image Rending Process

Memory

Memory

IDMAC

DMFC

DP

DC

DI

IDMAC

IC

Rotation

Section

Figure 5-7 IPU Process for Image Rotation

### IPU Configuraiton

1. Configure IDMAC channels for IC task

Table 5-2 lists the IDMAC channels of IC rotate tasks. This use case takes input channel #47 and output channel #50 for PP (postprocessing) task. API calls ipu\_rotate\_idmac\_config() to set the IDMAC for IC rotation tasks.

Table 5-2 Channels for Rotation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Channel#** | **IC’s Channel Name** | **R/W** | **Source** | **Destination** | **Purpose** |
| 45 | CB10 | Read | Memory | ENC ROT | Preprocessing data for rotation (encoding task) |
| 48 | CB8 | Write | ENC ROT | Memory | Preprocessing data after rotation (encoding task) |
| 46 | CB11 | Read | Memory | VF ROT | Preprocessing data for rotation (viewfinder task) |
| 49 | CB9 | Write | VF ROT | Memory | Preprocessing data after rotation (viewfinder task) |
| 47 | CB13 | Read | Memory | PP ROT | Postprocessing data for rotation |
| 50 | CB12 | Write | PP ROT | Memory | Postprocessing data after rotation |

The rotation related bit fields’ setting of input channel #47 is as follows:

* NPB (Number of pixels per burst access) must be set as 7, which means 8 pixels per burst.
* ROT (Rotation) is enabled, which means 90 degree rotation clockwise.
* BM (Block Mode) is set as 0x01, which means 8X8 pixels blocks.

The rotation related bit fields’ setting of output channel #50 is as follows:

* NPB (Number of pixels per burst access) must be set as 7, which means 8 pixels per burst.
* ROT (Rotation) is disabled. The rotation is performed in the input channel.
* HF (Horizontal Flip) is enabled depends on the use case.
* VF (Vertical Flip) is enabled depends on the use case.
* BM (Block Mode) is set as 0x01, which means 8X8 pixels blocks.

Please refer to 5.1.2 part 1 for the other bit fields settings.

1. Configure IC tasks.

* T3\_ROT is enabled, which means rotation for PP task.
* T3\_ FLIP\_LR is enabled depends on the use case, which means the LEFT/RIGHT flip for PP task.
* T3\_FLIP\_UD is enabled depends on the use case, which means the UP/DOWN flip for PP task.

**Note:** These three fields should be the same as those in IDMAC.

* PP\_EN is enabled, which enables the PP task.
* PP\_ROT\_EN is enabled, which enables PP rotation task.

1. Set IDMAC buffer ready

Set IDMAC buffer ready after configuring and enable the IC task. Set the output IDMAC channel buffer ready first and then the input IDMAC channel buffer.

1. Please refer to 5.1 for the settings of image rending process.

## Image Resizing Instance

### IPU Process Flow

The image resizing is performed in IC module. Figure 5-8 shows the IPU process for resizing an image and displaying it on a screen. The Main Processing Unit reads pairs of pixels from the Downsizing Output Memory background part.

Image Rending Process

Memory

Memory

IDMAC

DMFC

DP

DC

DI

IDMAC

IC

Downsize

Section

Main Processing

Section

Figure 5-8 IPU Process for Image Rotation

### IPU Configuration

1. Configure IDMAC channels for IC task

Table 5-3 lists the IDMAC channels of IC tasks. This use case takes input channel #11 and output channel #22 for PP (postprocessing) task. API calls ipu\_resize\_idmac\_config() to set the IDMAC for IC resizing tasks.

Table 5-3 Channels for Resizing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Channel#** | **IC’s Channel Name** | **R/W** | **Source** | **Destination** | **Purpose** |
| 11 | CB5 | Read | Memory | IC PP | Postprocessing data from memory |
| 22 | CB2 | Write | IC PP | Memory | Postprocessing data from IC to memory |
| 12 | CB6 | Read | Memory | IC VF | Preprocessing data from sensor stored in memory (for example Bayer) |
| 21 | CB1 | Write | IC VF | Memory/DMFC | Preprocessing data from IC (viewfinder task) to memory; This channel can be configured to send the data directly to the DMFC. This is done by programming the ic\_dmfc\_sel bit. |
| 20 | CB0 | Write | IC ENC | Memory | Preprocessing data from IC (encoding task) to memory |

The resizing related bit fields’ setting of input channel #11 is as follows:

* NPB (Number of pixels per burst access) is determined by frame width. The frame width must be multiple of burst size - 8 or 16 pixels as defined.

If frame width is a multiple of 16, set NPB as 16 or 8. Otherwise, NPB must be set as 8.

Note: The input’s frame width to the IC must be a multiplication of 8 pixels

The resizing related bit fields’ setting of output channel #22 is as follows:

* NPB (Number of pixels per burst access) is determined by frame width. The frame width must be multiple of burst size - 8 or 16 pixels as defined.

If frame width is a multiple of 16, set NPB as 16 or 8. Otherwise, NPB must be set as 8.

**Note:** The input’s frame width to the IC must be a multiplication of 8 pixels

Please refer to 5.1.2 part 1 for the other bit fields settings.

1. Configure IC tasks.

* CB2\_BURST\_16 defines the number of active cycles within a burst (burst size) coming from the IDMAC for IC’s CB2 (Channel #22). For pixel data the number of pixels should match the NPB[6:2] value on the IDMAC’s CPMEM.

CB5\_BURST\_16 defines the number of active cycles within a burst (burst size) coming from the IDMAC for IC’s CB5 (Channel #11). For pixel data the number of pixels should match the NPB[6:2] value on the IDMAC’s CPMEM.

* T3\_FR\_HEIGHT sets Frame Height for Post Processing (PP) task. The value of this field must be identical to the corresponding FH channel’s parameters in the IDMAC’s CPMEM. This parameter refers to the output’s size - 1.

T3\_FR\_WIDTH sets Frame Width for Post Processing (PP) task. The value of this field must be identical to the corresponding FW channel’s parameters in the IDMAC’s CPMEM. This parameter refers to the output’s size - 1.

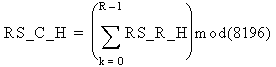
* PP\_DS\_R\_H sets Post-processing task downsizing horizontal ratio.

PP\_RS\_R\_H sets Post-processing task resizing horizontal ratio.

Horizontal resizing by bilinear interpolation between two adjacent pixels received from the Downsizing Output Memory according to the equation:

Resize_equal.bmp

where, RS\_C\_H - the current horizontal resizing coefficient. The calculation result is rounded to 8 bits. The resizing coefficient is calculated as



* PP\_DS\_R\_V sets Post-processing task downsizing vertical ratio.

PP\_RS\_R\_V sets Post-processing task resizing vertical ratio.

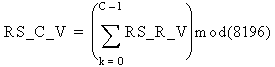
Vertical resizing by bilinear interpolation between the current and previous results of horizontal

resizing. Both current and previous results of horizontal resizing is stored in the Task Parameter

Memory. Resizing is accomplished according to the equation:

Resize_equal_v.bmp

where, RS\_C\_V - the current vertical resizing coefficient. The calculation result is rounded to 8 bits. The resizing coefficient is calculated as



* PP\_EN is enabled, which enables the PP task.

1. Set IDMAC buffer ready

Set IDMAC buffer ready after configuring and enable the IC resizing task. Set the output IDMAC channel buffer ready first and then the input IDMAC channel buffer.

1. Please refer to 5.1 for the settings of image rending process.

## Color Space Conversion Instance

### IPU Process Flow

There are two hardware modules to implement CSC (Color Space Conversion) in IPU. Figure 5-9 shows the IPU process for CSC in IC module.

Memory

Memory

IDMAC

DMFC

IDMAC

IDMAC

CH21

IC

Main Processing

Section

Downsize

Section

Figure 5-9 IPU Process for CSC (IC task)

Figure 5-10 shows the IPU process for CSC in DP module. DP module connects to DI, so this CSC is used for display when CSC is needed.

DP

DMFC

CSC

DC

DI

Memory

IDMAC

Figure 3-9 IPU Process for CSC (DP module)

### IPU Configurations for IC Task

The CSC is performed in the main processing unit inside IC.

1. Configure IDMAC channels for IC task

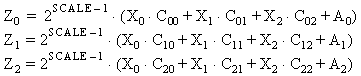
Please refer to 5.4.2 part 1. This use case takes input channel #11 and output channel #22 for PP (postprocessing) task.

1. Configure IC tasks.

First color space conversion YUV to RGB or RGB to YUV with the conversion matrix CSC1. The

conversion matrix coefficients are programmable. They are stored in the Task Parameter Memory.

The conversion equations are:

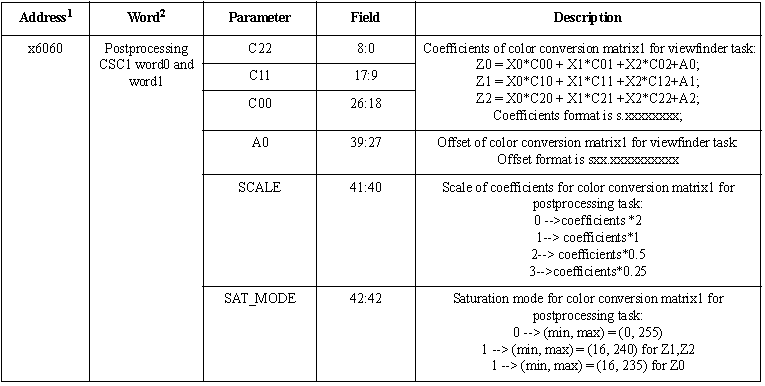


where for YUV to RGB: X0=Y, X1=U, X2=V, Z0=R, Z1=G, Z2=B,

for RGB to YUV: X0=R, X1=G, X2=B, Z0=Y, Z1=U, Z2=V

The resizing related bit fields’ setting of PP CSC1 task is as follows:

Table 5-4 PP task IC Parameters for CSC



**Note:** The Main Processing Unit reads pairs of pixels from the Downsize Output Memory background part. So the Downsize Unit should be also configured and enabled. Please refer to 5.4.2 part 2 for help.

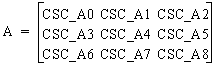
### IPU Configurations for DP Task

API calls ipu\_dp\_csc\_config() to do CSC inside DP. The conversion formula is:

CSC_DP_Fomula.bmp

Where

A is a 3x3-dimensional matrix of weights, each a 10-bit signed number with 8 fractional digits



B is a 3-dimensional vector of offsets, each a 14-bit signed number with 2 fractional digits

CSC_DP_B.bmp

E is an exponent, assuming one of the following values: -1,0,1,2 (allowing weights up to 8).

CSC_DP_E.bmp

The CSC related bit fields’ setting of DP module is as follows:

* DP\_CSC\_DEF\_SYNC is set as 1 to enable CSC in DP.
* DP\_CSC\_A\_SYNC\_ set **A** parameter of CSC.
* DP\_CSC\_B0\_SYNC/ DP\_CSC\_B1\_SYNC/ DP\_CSC\_B2\_SYNC set **B** parameter of CSC.
* DP\_CSC\_S0\_SYNC/ DP\_CSC\_S1\_SYNC/ DP\_CSC\_S2\_SYNC set **E** parameter of CSC.

1. References
2. i.Mx61 User Guide
3. Interface for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601
4. Platlib SDK code

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

http://www.freescale.com/support

USA/Europe or Locations Not Listed:

Freescale Semiconductor

Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
+1-800-521-6274 or +1-480-768-2130  
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH

Technical Information Center

Schatzbogen 7

81829 Muenchen, Germany

+44 1296 380 456 (English)

+46 8 52200080 (English)

+49 89 92103 559 (German)

+33 1 69 35 48 48 (French)

www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.

Headquarters

ARCO Tower 15F

1-8-1, Shimo-Meguro, Meguro-ku,

Tokyo 153-0064, Japan

0120 191014 or +81 3 5437 9125

support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.  
Exchange Building 23F  
No. 118 Jianguo Road  
Chaoyang District  
Beijing 100022  
China  
+86 010 5879 8000  
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center

P.O. Box 5405

Denver, Colorado 80217

1-800-441-2447 or 303-675-2140

Fax: 303-675-2150

LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale, the Freescale logo, CodeWarrior, ColdFire, PowerQUICC, QorIQ, StarCore, and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. CoreNet, QorIQ Qonverge, QUICC Engine, and VortiQa are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. ARM is the registered trademark of ARM Limited. ARMnnn is the trademark of ARM Limited.

© Freescale Semiconductor, Inc. 2011.