freescale semiconductor	MSG Software	
TECHNICAL SOLUTION AREA	Revision 0.3 S32V234, DCU, FRAME, DISPLAY, LINUX AMCU_SW_s32v234	Page i of 13

Frame Output library SW User Guide for s32v234 DCU

ABSTRACT:
This is the Software User Guide Document applicable for Frame Output library modules using s32v234 DCU.
KEYWORDS:
SRAM, allocator, Linux, driver
APPROVED:

AUTHOR	SIGN-OFF SIGNATURE #1	SIGN-OFF SIGNATURE #2
Tomas Babinec		

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
0.1	18-March-16	Tomas Babinec	Initial version.
0.2	4-May-16	Tomas Babinec	Updated for VSDK 0.9.3 release.
0.3	10-August-16	Tomas Babinec	Updated base od review.

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to define the API of the `FrameOutputDCU` and `FrameOutputV234Fb` objects from frame_io library. For exact definitions and implementation details please refer to [1].

1.2 Scope and Objective

This document should give a highlevel understanding of the mechanisms available to configure and use the ISP preprocessing pipeline from the user applications including the access to preprocessed frames in DDR.

1.3 Audience Description

This document is intended for internal use by s32v234 Vision SDK developers.

1.4 References

Id	*Title*	*Location*
[1]	*Frame Output source code*	*VSDK release*
[2]	*SDI SW ADD*	*Vision sdk git, folder:* s32v234_sdk\docs\internal\arch\
[3]	*Frame_output_v234fb*	*Vision sdk git, folder:* s32v234_sdk\docs\internal\arch\
[4]	*Frame_output_dcu*	*Vision sdk git, folder:* s32v234_sdk\docs\internal\arch\

Table 1 References Table

1.5 Definitions, Acronyms, and Abbreviations

Term/Acronym	*Description*
ADD	*Architecture Design Document*
SADS	*Software Architectural Design Specification*

SW	*Software*
HW	*Hardware*
IP	*Intellectual Property*
API	*Application Programming Interface*
SRAM	*Static Random Access Memory*
DRAM	*Dynamic Random Access Memory*
DDR	*Double Data Rate DRAM*
fDMA	*fast Direct Memory Access HW block*
SDI	*Sensor Data Interface library*
SoC	*System on Chip*
ISP	*Image Signal Processing subsystem*
DCU	*Display Controller Unit*
FB	*Linux Frame Buffer concept*
2D ACE	*Two Dimensional Animation and Compositing Engine (DCU in S32V234 SoC)*

Table 2 Acronyms Table

1.6 Document Location

This document is available at the following location: vision SDK Git repository (ssh://git@sw-stash.freescale.net/vswat/vsdk.git) in folder s32v234_sdk\docs\internal\arch\.

2 General Description

The `FrameOutputDCU` and `FrameOutputV234Fb` objects are part of the frame_io library from VSDK. They have been developed to create a simple mechanism for graphics data display output using the DCU (Display Controller Unit) available in s32v234 SoC - the 2D ACE.

The `FrameOutputDCU` and `FrameOutputV234Fb` objects have to be understood as a convenience layer for fast demo creation and are not intended to provide full support for the features of the underlying HW.

3 Functional Description

The `FrameOutputDCU` and `FrameOutputV234Fb` objects act as a convenience layer between the DCU HW drivers' API and user application (see Figure 1). The `FrameOutputDCU` is used in case of a baremetal (standalone build) version of the application and internally invokes the DCU driver API directly. The `FrameOutputV234Fb` is intended for Linux environment. It is build above the Linux framebuffer device file and it's API, which is responsible for the DCU driver configuration.

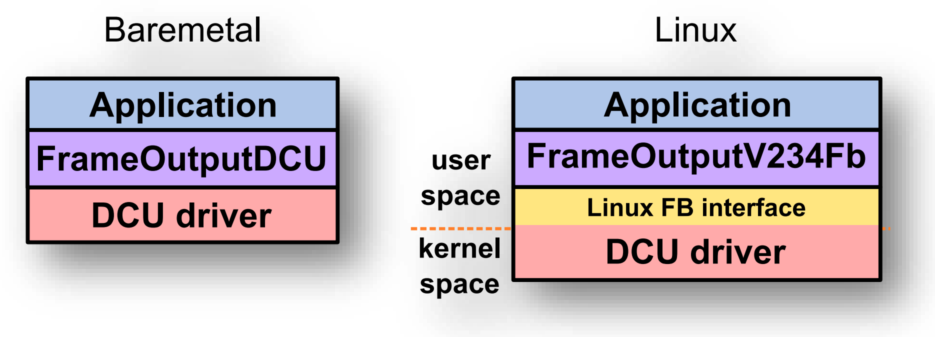

Figure 1: FrameOutput position

3.1 Main tasks

- Configuration of the DCU HW = 2D ACE.
- Output of graphical data to display:

3.2 FrameOutputDCU

3.2.1API

Method	Description
`FrameOutputDCU`	Constructor. Calls Init(). Parameters specify resolution and pixel format of the data to be displayed.
`Init`	Reinitialize the DCU HW. Can be used also to reset the previous parameters and setup a new DCU display mode. Parameters specify resolution and pixel format of the data to be displayed.
`Reset`	SW reset of the DCU block.
`Disable`	Disables DCU output.

`PutFrame` (overrides the parent object's pure virtual method)	Displays the data buffer provided in a form of a pointer in the first parameter. There is no internal copy of the data involved. If the second parameter is set to true cache flush will be done before displaying the data.
`GetStatus`	Returns current status of the object.
`BaseInit`	(private) Initialization of the object members to default values.

Table 3: FrameOutputDCU API

3.2.2Workflow

To use the `FrameOutputDCU` object the user application has to include the `frame_output_dcu.h` header.

Now the `FrameOutputDCU` object can be created by the following constructor call:

```
#define CHNL_CNT io::IO_DATA_CH3

io::FrameOutputDCU lDcuOutput(  WIDTH,
                                HEIGHT,
                                io::IO_DATA_DEPTH_08,
                                CHNL_CNT);
```

The constructor parameters define the required display geometry (`WIDTH` and `HEIGHT`), pixel format (`io::IO_DATA_DEPTH_08` = 8 bits per pixel color component, CHNL_CNT = number of pixel components, e.g. RGB => 3).

By default the DCU configuration expects the 1920x1080 LVDS display to be connected. To use external display with different resolution different DCU timing might be required.

When the `FrameOutputDCU` object was created successfully the application can start to display the image data by calling `FrameOutputDCU.PutFrame()` method.

```
lDcuOutput.PutFrame(lpFrame, false);
```

First parameter is the physical address of the buffer to be displayed. If the second parameter is not used cache flush will be invoked before the address is provided to the DCU HW.

The provided buffer is being displayed directly. No data copy is done internally and therefore the data should not be updated until a different buffer is displayed. Otherwise distortion artefacts might be observed on the screen.

3.2.3Supported resolutions and pixel formats

Currently also the following ones are supported, 1920x1080@60Hz and 1280x1024@60Hz. To switch between the DCU timing settings please use corresponding defines in s32v234_sdk/platform/s32_v234/config.h.

Currently the bits per pixel and number of components per pixel are ignored internally and by default the data are expected to come in RGB888 format. To modify the default pixel format additional parameter has to be added to the `FrameOutputDCU` constructor. The tested values of the pixel format parameter are:

- **DCU_BPP_8**

  Grayscale image, 8bits per pixel,
- **DCU_BPP_YCbCr422**

  YUV, 8bits U, 8bits Y, 8bits V, 8bits Y, …
- **DCU_BPP_24**

  RGB, 8bits R, 8bits G, 8bits B.

The `FrameOutputDCU` object uses directly the user provided address as a source of the data for the DCU (no data copy involved). If the data at the provided address are being modified while set to the `FrameOutputDCU` as current frame buffer source, tearing artefacts can be observed on the screen. Therefore the user application is responsible to ensure proper scheme of the frame-buffer address update.

3.3 FrameOutputV234Fb

3.3.1API

Method	Description
`FrameOutputV234Fb`	Constructor. Calls Init(). Parameters specify resolution and pixel format of the data to be displayed.
`Init`	Initializes the use of the Linux FB and sets up the required configuration. Can be used also to reset the previous parameters and setup a new display mode.
`Disable`	Un-maps FB buffers, closes file descriptors.
`PutFrame` (overrides the parent object's pure virtual method)	Displays the data buffer provided in a form of a pointer in the first parameter. Internally the data is copied (memcpy) to the memory block mapped from the framebuffer. If the second parameter is set to false no cache flush will be done before displaying the data.
`PutFrameAlpha`	Displays the data buffer provided in a form of a pointer in the first parameter. Internally the data is copied (for cycle) to the memory block mapped from the framebuffer and to each pixel the 0xff alpha byte is added. If the second parameter is set to false no cache flush will be done before displaying the data.
`GetStatus`	Returns current status of the object.

Table 4: FrameOutputV234Fb API

3.3.2Workflow

To use the `FrameOutputV234` object the user application has to include the `frame_output_v234fb.h` header.

Now the `FrameOutputV234` object can be created by the following constructor call:

```
#define CHNL_CNT io::IO_DATA_CH3

io::FrameOutputV234Fb lDcuOutput(  WIDTH,
                                HEIGHT,
                                io::IO_DATA_DEPTH_08,
                                CHNL_CNT);
```

The constructor parameters define the required display geometry (`WIDTH` and `HEIGHT`), pixel format (`io::IO_DATA_DEPTH_08` = 8 bits per pixel color component, CHNL_CNT = number of pixel components, e.g. RGB => 3). The bits per pixel parameter is ignored by the object at the current state.

By default the framebuffer configuration is set based on the DCU node in Linux devicetree. If any modification is required with respect to the DCU output timing is required it should be done there.

The Linux framebuffer driver exports 8 DCU layers as special devicefiles /dev/fb0…7. The `FrameOutputV234` object interfaces only with the /dev/fb0.

When the `FrameOutputV234` object was created successfully the application can start to display the image data by calling `FrameOutputV234.PutFrame()` method.

```
lDcuOutput.PutFrame(lpFrame, false);
```

First parameter has to be the virtual address of the buffer to be displayed. If the second parameter is not used cache is invalidated before the data are displayed.

The provided buffer is not being displayed directly. Data copy is done internally.

3.3.3Supported resolutions and pixel formats

The `FrameOutputV234` object always configures the FB in a double buffer scheme. It was tested to work well up to 1280x720, 24bpp. Attempts to use higher resolution (e.g. 1920x1080, 24bpp) might fail because of problems with allocation of contiguous buffers above certain size in Linux kernel.

Currently the `FrameOutputV234` object supports the following pixel formats:

- **RGB888**

  8bits Red, 8bits Green, 8bits Blue,

  In case of 3 channels per pixel requested.

- **RGBA8888**

  8bits Red, 8bits Green, 8bits Blue, 8bits Alpha,

  In case of 4 channels per pixel requested.

- **BGR565**

  BGR, 5bits Red, 6bits Green, 5bits Blue, In case of 2 channels per pixel requested.

- **DCU_BPP_YCbCr422**

  YUV, 8bits U, 8bits Y, 8bits V, 8bits Y, …

4 High Level Design

4.1 System Decomposition

The `FrameOutputV234` and `FrameOutputDCU` objects are part of the `frame_io` library and create a utility layer to access the DCU configuration and display output. They are derived from the `FrameOutputBase` object which defines the least supported API.

4.2 File Structure

The `FrameOutputV234` and `FrameOutputDCU` objects' code is located in VSDK under s3234_sdk/libs/io/frame_io folder. Internally it has the following structure:

- build-* – build folders for supported platforms (standalone and Linux)
  - Makefile
- include
  - frame_output_dcu.h – declaration of the object and its API,
  - frame_output_v234fb.h – declaration of the object and its API,
- src
  - frame_output_dcu.cpp – definition of the object and its API,
  - frame_output_v234fb.cpp – definition of the object and its API,
- BUILD.mk – defines build details

4.3 Module Usage

< This section contains module usage restrictions.>