

# Using the Video Port of TMS320DM646x

Todd Hiers and Kedar Chitnis

## ABSTRACT

This application report has supplemental information about using the DM646x video port. The tips and tricks in this document are useful in video security applications as well as other applications that make use of the video port.

## Contents

1	Introduction .....	1
2	Video Port Interface (VPIF) .....	2
3	System .....	8
4	References .....	12

## List of Figures

1	External Video Decoder and DM646x Connection .....	2
2	Frame in Sync .....	2
3	Frame Out-of-Sync .....	2
4	External Video Decoder and DM646x Connection .....	4
5	Storage Format of Data in SDRAM (interlaced image) .....	5
6	External Video Decoder and DM646x Connection .....	6
7	Storage Format of Data in SDRAM (interlaced image) .....	7
8	Image of Horizontal Distance in Y/C Mode .....	7
9	YUV422 Planar .....	8
10	YUV420 Planar .....	9
11	YUV422 Semi-Planar .....	10
12	YUV420 Semi-Planar .....	10
13	YUV422 Interleaved.....	11
14	External Video Decoder and DM646x Connection.....	11
15	Data Flow Pipeline .....	12

## 1 Introduction

This document provides information on the following topics:

- Re-syncing to external video input without restarting the capture driver
- Understanding how video data is saved in DDR memory by the VPIF hardware
- Interfacing to external video decoder and finding the VPIF register settings
- Implementing a video capture data flow in DM646x using different hardware blocks like video port interface (VPIF), video data conversion engine (VDCE), etc.
- Understanding the different video data formats supported in DM646x

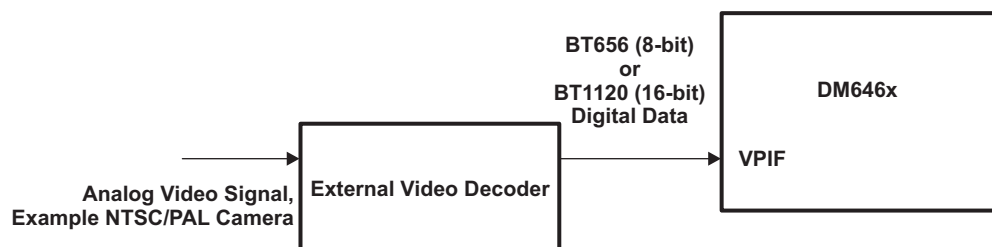
This document is not a replacement for the standard DM646x hardware data sheets and documentation available from TI. This document is meant to be more of a practical and easy-to-understand view of the DM646x from an applications point-of-view, and has information based on feedback received from customers and field application engineers.

All trademarks are the property of their respective owners.

## 2 Video Port Interface (VPIF)

### 2.1 Re-Syncing to External Video Input Without Restarting the Capture Driver

In DM646x, the VPIF is used to interface to the external video decoder like the TVP5158, Techwell TWxxxx. Here, the analog video signal is fed to the external video decoder and the digitized BT656/BT1120 data is received by the VPIF in the DM646x device as shown in [Figure 1](#).



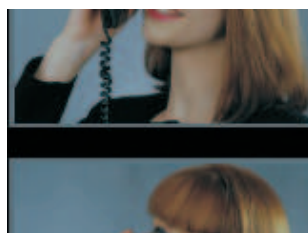
**Figure 1. External Video Decoder and DM646x Connection**

When the analog signal is active and VPIF is started, the VPIF syncs to the first frame it receives from the external video decoder and outputs captured data to the DDR memory.

If the analog signal is disconnected, the external video decoder may stop sending active video data and instead send blanking data to the VPIF; in this case, the exact behavior of the external video decoder depends on the type of the external video decoder. In any case, when analog signal is removed, the VPIF stops outputting data to DDR memory. When the analog signal is reconnected, the VPIF starts receiving frames from the external video decoder; however, the VPIF may receive frames that are *out-of-sync*. VPIF is not able to sync to the start of the frame, therefore, the data received in memory may show incorrect frames as shown in [Figure 3](#).



**Figure 2. Frame in Sync**



**Figure 3. Frame Out-of-Sync**

To make VPIF sync to the frame received from the external video decoder, stop and re-start it again.

By default, when using TI video for Linux 2 (V4L2) APIs means doing `STREAM_OFF` and then `STREAM_ON`.

However, there are issues present with this approach:

- You need to know when to re-start the VPIF
- The restart operation is not quick enough

The solution to solve these issues is to implement the new V4L2 ioctl in the capture driver as shown below:

```

/*
 call in a new ioctl in the davincihd_capture.c
 this sample code operates on VPIF CH0, modify appropriately when using any other capture channel
*/
int vpiif_sync_loss_detect_and_recover()
{
    int val;

    video_decoder_read_sync_status( &val);

    if( ! IS_VIDEO_HSYNC_VSYNC(val) ) {
        // video loss detected by application, stop VPIF
        val = regr(VPIF_CH0_CTRL) & 0xFFFFFFF0;
        regw(val, VPIF_CH0_CTRL);

        do {
            mdelay(60);
            video_decoder_read_sync_status( &val);
        } while(! IS_VIDEO_HSYNC_VSYNC(val) );

        // sync regained, reenable VPIF
        val = regr(VPIF_CH0_CTRL) | 0x00000001;
        regw(val, VPIF_CH0_CTRL);
    }
}

// function for TVP5147 are given below
// modify appropriately for other video decoders

#define IS_VIDEO_HSYNC_VSYNC(val) (((val) & 0x6)==0x6)

int video_decoder_read_sync_status( int *val)
{
    return tvp5147_i2c_read_reg (
        &tvp5147_channel_info[dec->channel_id].i2c_dev.client,
        0x3a, val
    );
}

```

The main steps in this sample code are:

1. Read the video decoder register (via I2C), which reports H sync, V sync *lock* status.

---

**NOTE:** The video decoder also reports the *signal present* and *video detected* status, but for this purpose you need to find out if the video decoder has detected a *locked* video signal; therefore, make sure to read the H sync, V sync *lock* status.

---

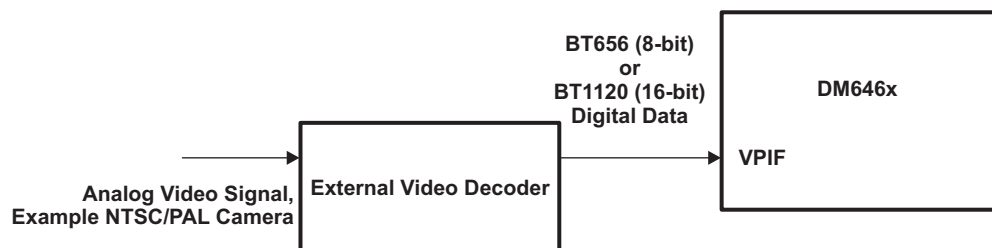
2. Disable VPIF if either H or V is not locked because this indicates that there has been a signal loss.
3. Keep checking the same status, after disabling VPIF, until the video decoder reports that it has detected a locked signal.
4. Re-enable VPIF once a locked signal is detected. Then, VPIF will sync to the first frame it receives and the following frames output to DDR memory will be in sync.

The final step is to call the ioctl corresponding to the above function `vpif_sync_loss_detect_and_recover()` in the application. The recommendation is to call the ioctl in a separate low priority thread in the application, and make the call once every 60 msecs, like the following example:

```
void *thread_main_vpif_sync_loss_detect_and_recover(void *)
{
    while(1) {
        usleep(60*1000); // 60msecs sleep
        /*
         * ioctl which results in vpif_sync_loss_detect_and_recover() getting called
         */
        ioctl(capture_fd, VPIF_RESYNC, 0);
    }
    return NULL;
}
```

## 2.2 Understanding How Video Data is Saved in DDR Memory by the VPIF Hardware

In DM646x, the VPIF is used to interface to the external video decoder like the TVP5158, Techwell TWxxxx. Here, the analog video signal is fed to the external video decoder and the digitized BT656/BT1120 data is received by the VPIF in the DM646x device as shown in [Figure 4](#).



**Figure 4. External Video Decoder and DM646x Connection**

When the analog signal is active and VPIF is started, the VPIF syncs to the first frame it receives from the external video decoder and outputs captured data to the DDR memory.

The data that is saved to external DDR memory depends on the register settings in the VPIF and also on the actual data that is sent from the external video decoder.

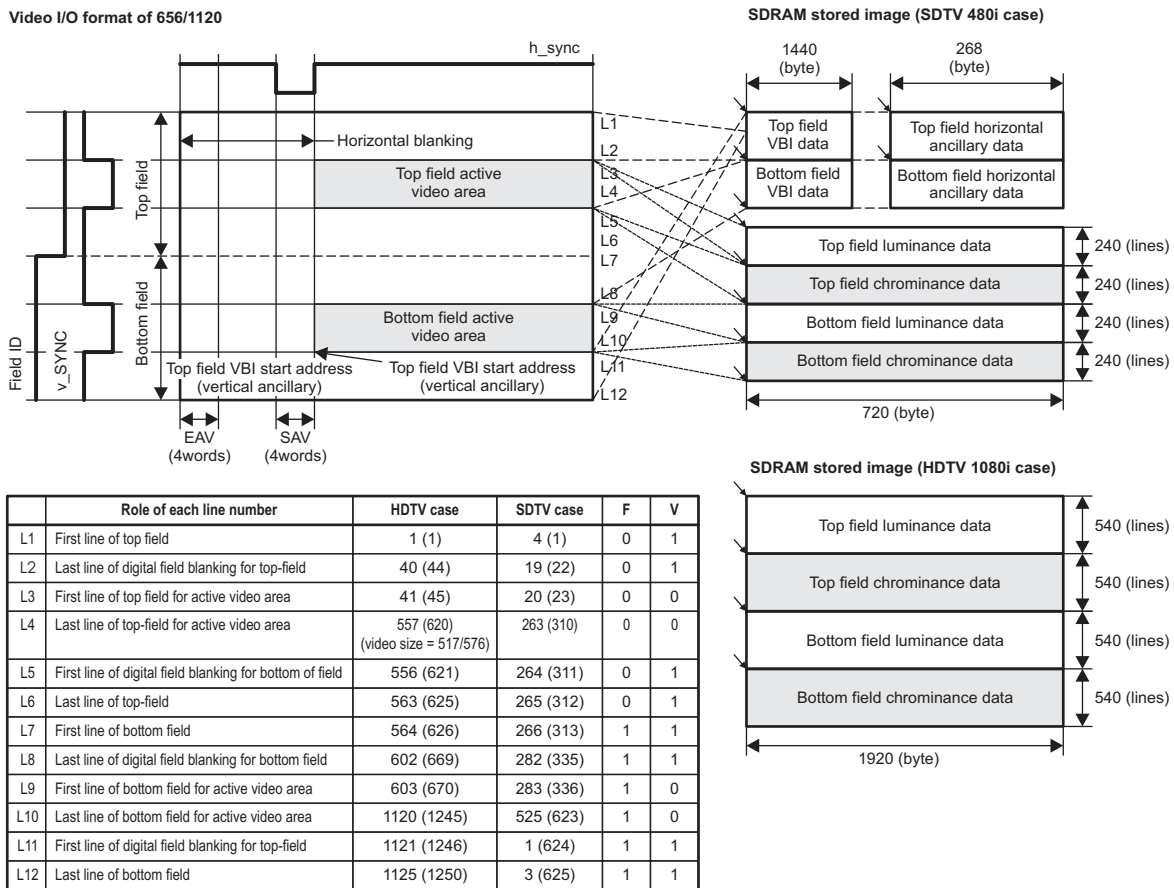


Figure 5. Storage Format of Data in SDRAM (interlaced image)

Figure 5 shows that the data that is saved to DDR is from line L3 to line L4 (for even field) and from line L9 to line L10 (for odd field). The data arrangement in DDR depends on the VPIF register setting. Figure 5 shows a separate field method of saving data to DDR. The VPIF also supports this interleaved method of saving data to memory. The VPIF keeps Y and C data in separate buffers in the DDR as shown in Figure 5. For more detailed information, see the *TMS320DM646x DMSoC Video Port Interface (VPIF) User's Guide (SPRUER9)*.

The VPIF allows you to set values for L1, L3, L5, L7, L9, L11, SAV2EAV, EAV2SAV, IMG\_ADD\_OFFSET. Out of these, values L3, L5, L9, L11, SAV2EAV and IMG\_ADD\_OFFSET are important when understanding how much data gets saved to DDR memory.

The size of data that gets saved to DDR memory when storage format is field interlaced (default storage format used in VPIF V4L2 capture driver) is:

$$((L5-L3) + (L11-L9)) \times 2 \text{ (for Y/C)} \times \text{CHn\_IMG\_ADD\_OFST}$$

CHn\_IMG\_ADD\_OFST is usually the number of pixels in a line unless you set it up differently.

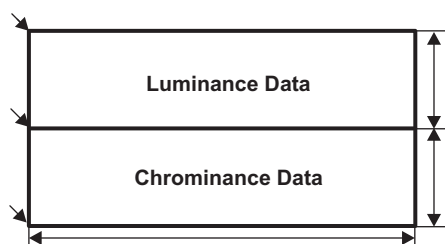
The number of pixels per line is the SAV2EAV register setting in the VPIF. Typically, CHn\_IMG\_ADD\_OFST = CEIL(SAV2EAV, 8), i.e., the next higher multiple of 8 pixels.

For settings shown in Figure 5, the size of the data saved in memory for the SDTV case is:

$$((264-20) + (525-283)) \times 2 \times 720 = 701280$$

Furthermore, the Y and C data buffer size is:  $701280/2 = 350640$

By default, the V4L2 capture driver will arrange the Y and C data buffer as shown below:

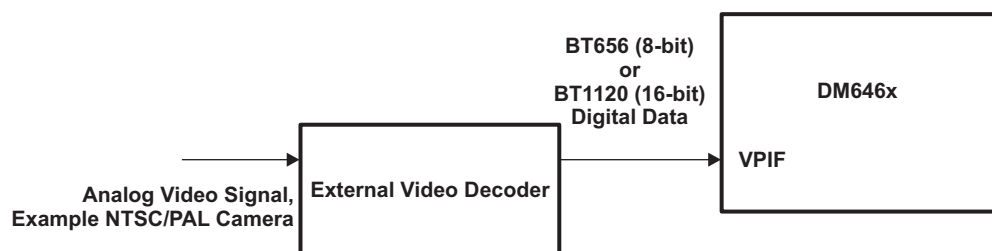


Therefore, if the allocated buffer size is less than the settings provided above, Y and C buffers could overlap each other, resulting in Y data corrupting C data and in the *green/pink* color artifact winding up at the top few lines of the buffer.

Note that the values used for L1, L3, L5, L7, L9, L11 and SAV2EAV depend on the video decoder and video standard being used for data capture.

### 2.3 Interfacing to External Video Decoder and Finding the VPIF Register Settings

In DM646x, the VPIF is used to interface to the external video decoder like the TVP5158, Techwell TWxxxx. Here, the analog video signal is fed to the external video decoder and the digitized BT656/BT1120 data is received by the VPIF in the DM646x device as shown in [Figure 6](#).



**Figure 6. External Video Decoder and DM646x Connection**

When the analog signal is active and VPIF is started, the VPIF syncs to the first frame it receives from the external video decoder and outputs captured data to the DDR memory.

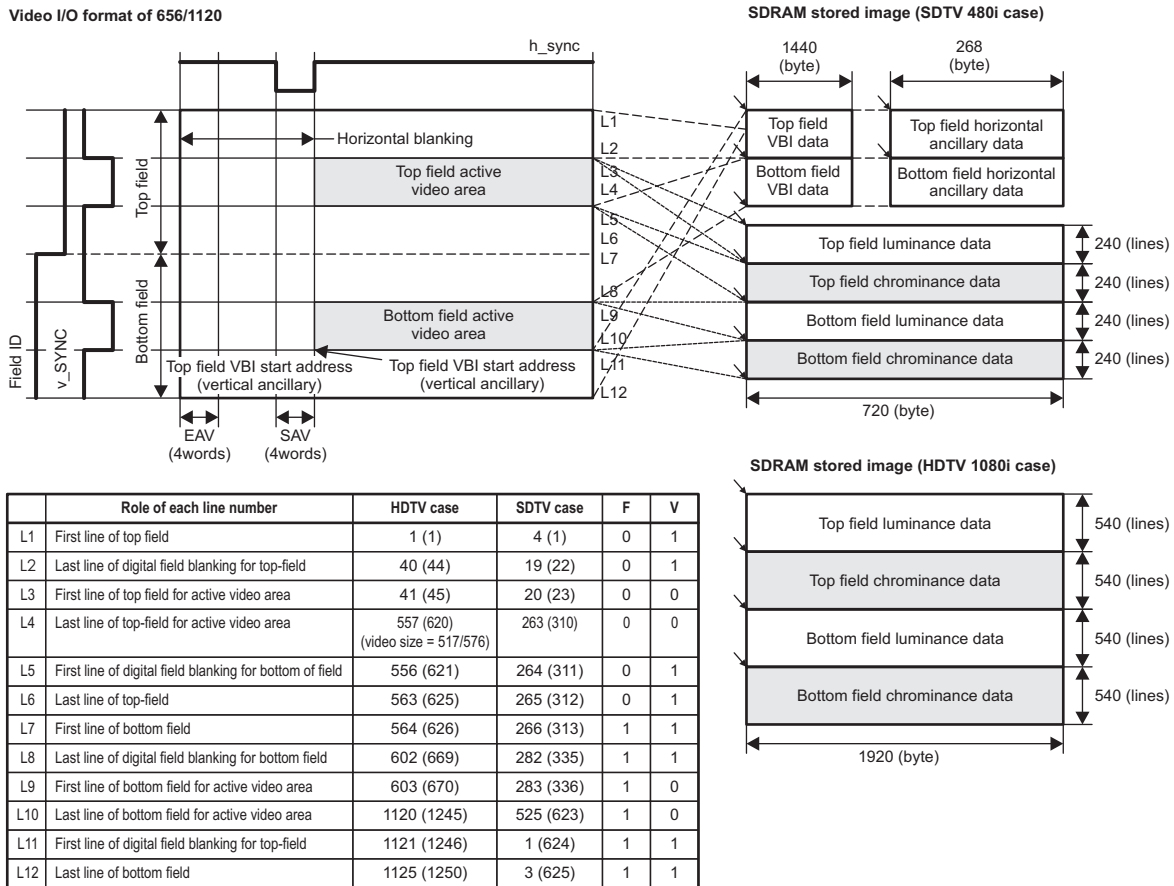


Figure 7. Storage Format of Data in SDRAM (interlaced image)

Figure 7 shows the timing of the data that is received from the external video decoder

The VPIF allows you to set values for L1, L3, L5, L7, L9, L11, SAV2EAV, EAV2SAV and IMG\_ADD\_OFFSET. For detailed register descriptions, see the *TMS320DM646x DMSoC Video Port Interface (VPIF) User's Guide (SPRUER9)*.

Do the following things when interfacing at the top level of the external video decoder:

- Initialize the external video in the correct mode of operation, e.g., NTSC or PAL
- Setup the external video decoder register for HBLANK, VBLANK and other frame timing settings
- Base the VPIF register settings on the timing of the data sent by the external video decoder, specifically L1, L3, L5, L7, L9, L11, SAV2EAV, EAV2SAV, VSZ
- L3, L5 and L9, L11 control which part of the data gets saved to DDR memory as described in Section 2.2
- EAV2SAV is set based on the HBLANK (or equivalent) register setting in the video decoder.

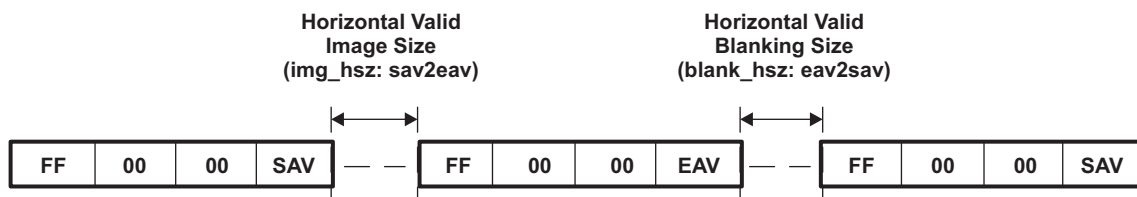


Figure 8. Image of Horizontal Distance in Y/C Mode

- As shown in Figure 8, EAV2SAV is specified in units of pixels, *excluding* the EAV and SAV BT.xxx codes.

- In some video decoders, HBLANK is specified in units of bytes and includes the BT.xxx codes. Make sure to appropriately interpret the HBLANK setting in the video decoder, then set the EAV2SAV in VPIF.
- If set incorrectly, EAV2SAV causes the VPIF not to capture any data and hang.
- Also, the video decoder needs to ensure that it does not vary the SAV2EAV and EAV2SAV from line-to-line and frame-to-frame, otherwise, the VPIF behavior is undefined.

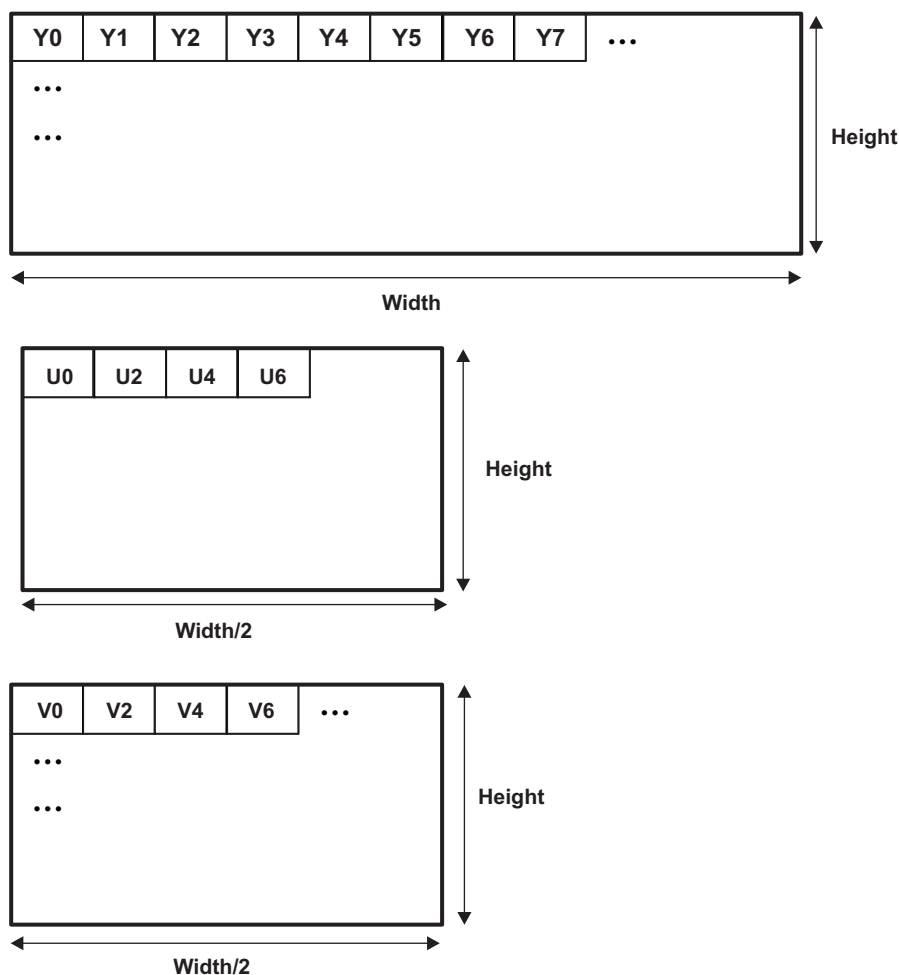
### 3 System

#### 3.1 Video Data Formats

This section describes the different video data formats encountered when working in DM646x.

##### 3.1.1 YUV422 Planar

Here the Y, U and V are stored in separate planes with U and V data sub-sampled by 2 in horizontal direction. This format is not used in DM646x and is described only for reference.



**Figure 9. YUV422 Planar**

##### 3.1.2 YUV420 Planar

This is similar to the YUV422 planar except that U and V data is sub-sampled by 2 in vertical as well as horizontal direction as shown in [Figure 10](#). This format is not used in DM646x and is described only for reference.



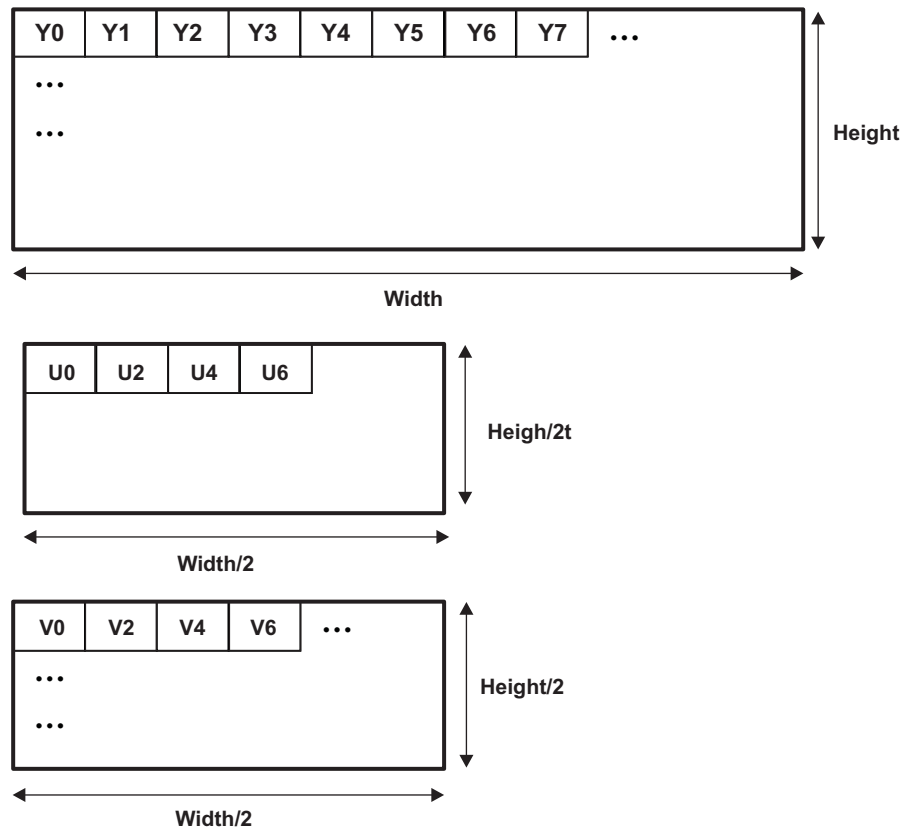
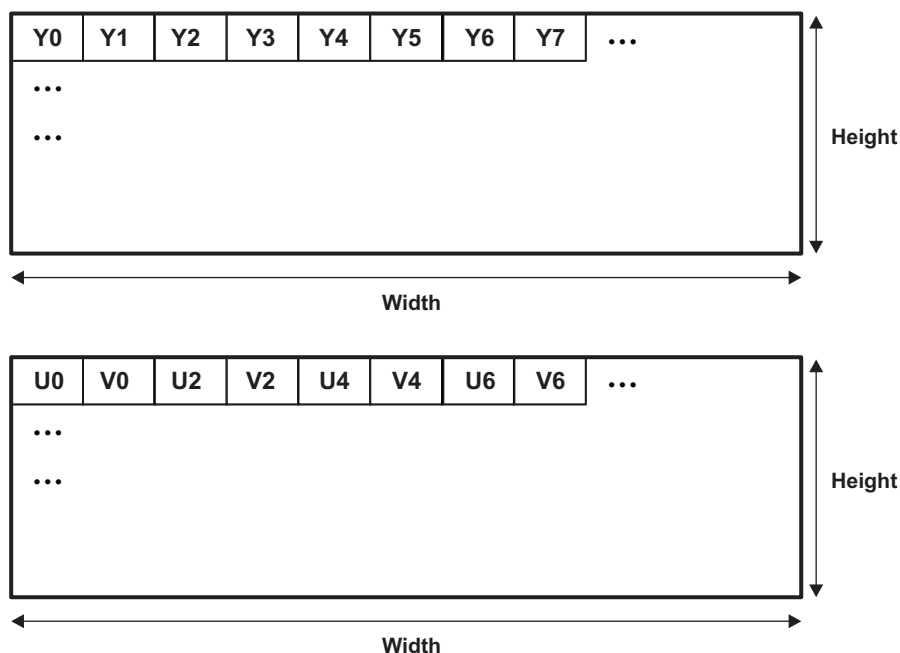


Figure 10. YUV420 Planar

### 3.1.3 YUV422 Semi-Planar

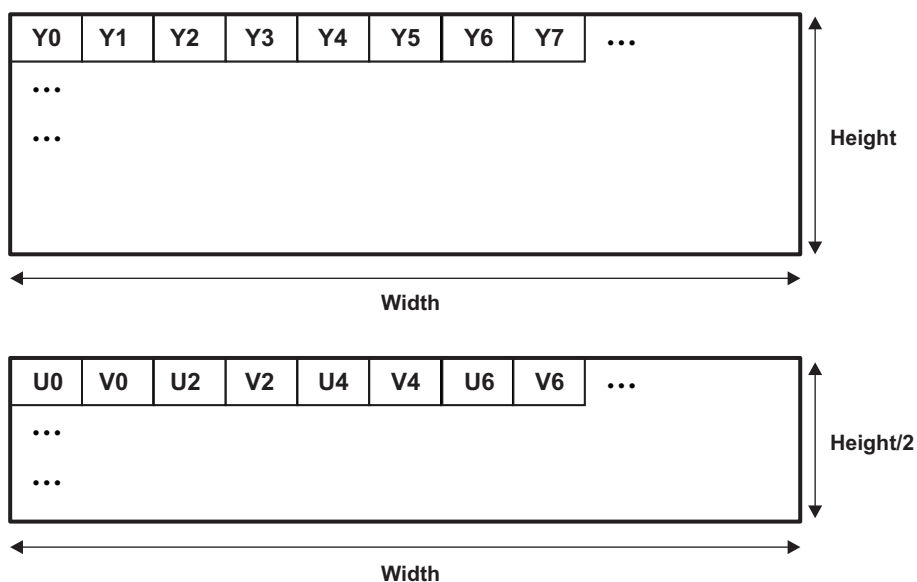
This is similar to the YUV422 planar except that U and V data is interleaved and stored in a single plane as shown in [Figure 11](#). This format is used in DM646x for VPIF capture and display.



**Figure 11. YUV422 Semi-Planar**

### 3.1.4 YUV420 Semi-Planar

This is similar to the YUV422 semi-planar, except that U and V data is interleaved and sub-sampled in the vertical direction by 2 and stored in a single plane as shown in [Figure 12](#). This format is used in DM646x for DSP-based codecs like H264 and MPEG4. The VDCE hardware engine in DM646x supports color conversion from the YUV422 semi-planar to the YUV420 semi-planar formats. This is useful when converting VPIF captured data and providing it as input to H264 encode.



**Figure 12. YUV420 Semi-Planar**

### 3.1.5 YUV422 Interleaved

In this format, Y, U and V are interleaved and saved in a single plane as shown in Figure 13. This format is not used in DM646x and is described only for reference. Other TI chips like DM6446, DM355 and DM365 use this format in certain modes.

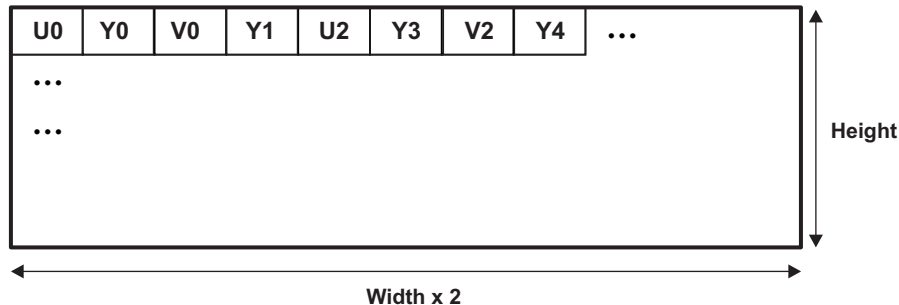


Figure 13. YUV422 Interleaved

### 3.2 Implementing a Video Capture Data Flow in DM646x Using Different Hardware Blocks Like VPIF and VDCE

In DM646x, the VPIF is used to interface to the external video decoder like the TVP5158, Techwell TWxxxx. Here, the analog video signal is fed to the external video decoder and the digitized BT656/BT1120 data is received by the VPIF in the DM646x device.

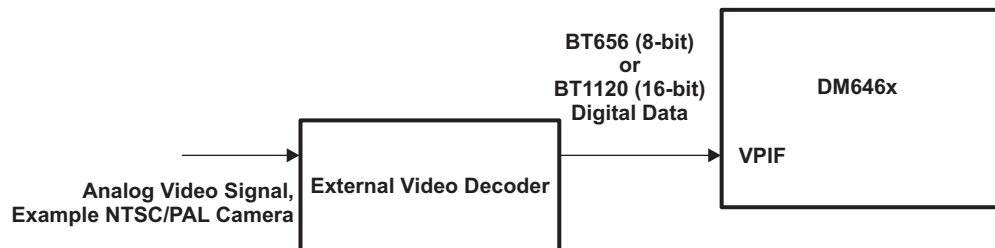


Figure 14. External Video Decoder and DM646x Connection

When the analog signal is active and VPIF is started, the VPIF syncs to the first frame it receives from the external video decoder and outputs captured data to the DDR memory.

The video data is processed via several different steps once it is received in the DDR memory, such as resizing, color conversion, compression, network, storage, etc.

This section shows an example data flow that could be used in a IPNC kind of application. Note that this is just an example and many different data flows are possible depending on the application requirements.

Some requirements for the IPNC data flow are described below:

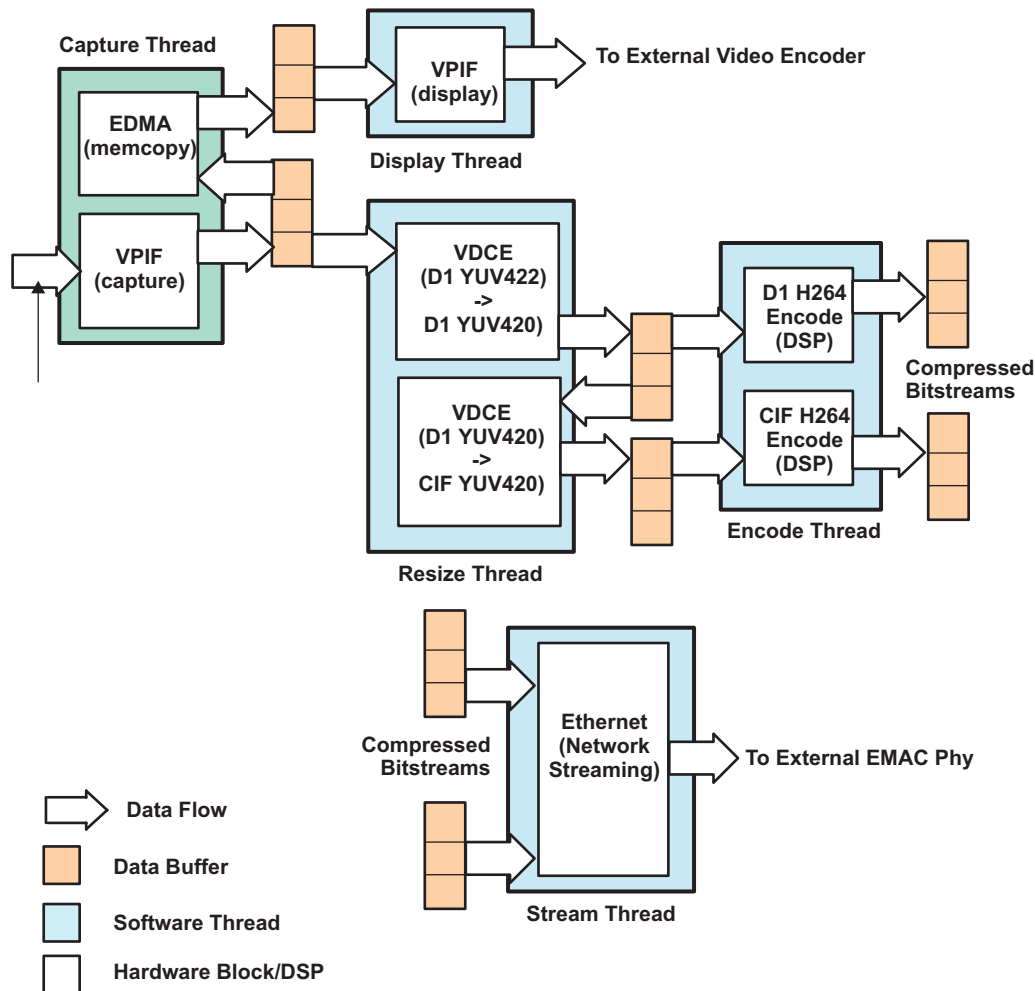
- D1 size video capture via external video decoder (YUV422 semi-planar)
- D1 size video display for local display via external D1 video encoder (YUV422 semi-planar)
- DSP side codec supports only YUV420 semi-planar input format
- D1 size H264 compression using DSP side codec (YUV420 semi-planar)
- Secondary stream of CIF size H264 compression using DSP side codec (YUV420 semi-planar)
- D1, CIF compressed bitstream are streamed over network via Ethernet interface

Data flow design decisions based on the above requirements:

- Since video capture and video display need the same video format, and are of the same size, no format conversion/resizing is required.
- D1 H264 compression needs D1 YUV420 semi-planar format, therefore, color convert D1 YUV422 data to D1 YUV420 using the VDCE engine.

- CIF H264 compression needs CIF YUV420 semi-planar format, therefore, resize D1 YUV420 data from the above step to CIF YUV420 data using VDCE engine.

Figure 15 shows the overall data flow pipeline based on the requirements and designs discussed in Section 3.2.



**Figure 15. Data Flow Pipeline**

- The data buffers connecting each step have multiple buffers so that different steps in the pipeline can run in parallel, providing maximum system performance.
- In general its recommended to have at least three buffers for capture and display, and at least two buffers for other steps in the processing pipeline
- The different software threads are also shown in Figure 15. By using multiple threads and data buffers each step like capture, display, resizing, encode and stream will run in parallel with each other; therefore, system throughput will be maximum.

#### 4 References

- *TMS320DM646x DMSoC Video Port Interface (VPIF) User's Guide* ([SPRUER9](#))

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<b>Products</b>		<b>Applications</b>	
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>	Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>	Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>	Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>	Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>	Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>	Energy	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>	Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>	Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>	Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>	Space, Avionics & Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
RF/IF and ZigBee® Solutions	<a href="http://www.ti.com/lprf">www.ti.com/lprf</a>	Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
		Wireless	<a href="http://www.ti.com/wireless-apps">www.ti.com/wireless-apps</a>