Freescale Semiconductor
Application Note

Document Number: AN4514
Rev. 1, 05/2012

MPC5604E Video Encoder Wrapper

by: Pavel Bohacik
AISG Application Engineer

Contents

1 Introduction

The MPC5604E microcontroller is the newer member of the 32-bit microcontroller family built on Power Architecture® technology. This device is targeted at the chassis and safety market segment visual-based driver assistance, especially the CMOS Vision Sensor Gateway, Radar Sensor Gateway, and Infotainment Network Gateway. This application note describes how to connect video device to the MPC5604E and how to configure MPC5604E Video Encoder Wrapper correctly to get MJPEG video stream from your YUV422 video data.

2 Video Encoder features

- Video data is YUV422 or ITU656 like compliant stream
- Input pixel clock up to 96 MHz.
- Baseline/extended sequential ISO/IEC 10918-1 JPEG encoder (JPEG 8-bit and 12-bit encoder)
- 10-bit mode using 12-bit encoder
- Programmable Huffman tables (2AC, 2DC) and quantization tables (4) (JPEG standard)
- Rate control via quantization table update
- Receives ‘embedded line’ information from sensor, containing all register settings, as ‘subchannel information’ into dedicated SRAM.
- Encoded stream is stored in SRAM output buffer memory, accessible over slave AHB bus.

3 Video Encoder features

Video Encoder Wrappper consist of the following:

Configuration interface controls Image size, operation control and operation status.

JPEGIn interface controls In JPEG syntax, upload Huffman tables and Quantization Tables (for rate control) MJPEG encoder which performs 8-bit/12-bit data encoding.

JPEGOut Interface controls bitstream and moving data into 2Kx32bit output buffer - 8KB which is approximately 5 Ethernet packages.

The video encoder receives parallel video YUV422 data at the pixel clock from camera interface. This data can have embedded vertical and horizontal synchronization signals (VSYNC and HREF) in case of ITU-BT656 format is used or these signals are distributed on the two separate lines in case external synchronization mode is used. Input logic decodes different polarity of clock, data and synchronization signals. This data is synchronized with peripheral video clock named ipg_video_clk (see MPC5604E Reference manual - Figure 8. MPC5604E Clock Selection), with the help of asynchronous FIFO and then this data is made to pass through decoder block for sync extraction in case of ITU-BT656 format (FF-00-00). The data is fed directly to the reordering logic, which downsamples the data from YUV422 to YUV420 data. This data is fed to the MJPEG encoder in 8×8 Blocks (8×8 block = Minimum Coding Unit = MCU), 4 Luma blocks then 2 Chroma blocks, which compresses the JPEG Image and stores the data in a circular output buffer, from which the data can be read via Amba Slave Interface.

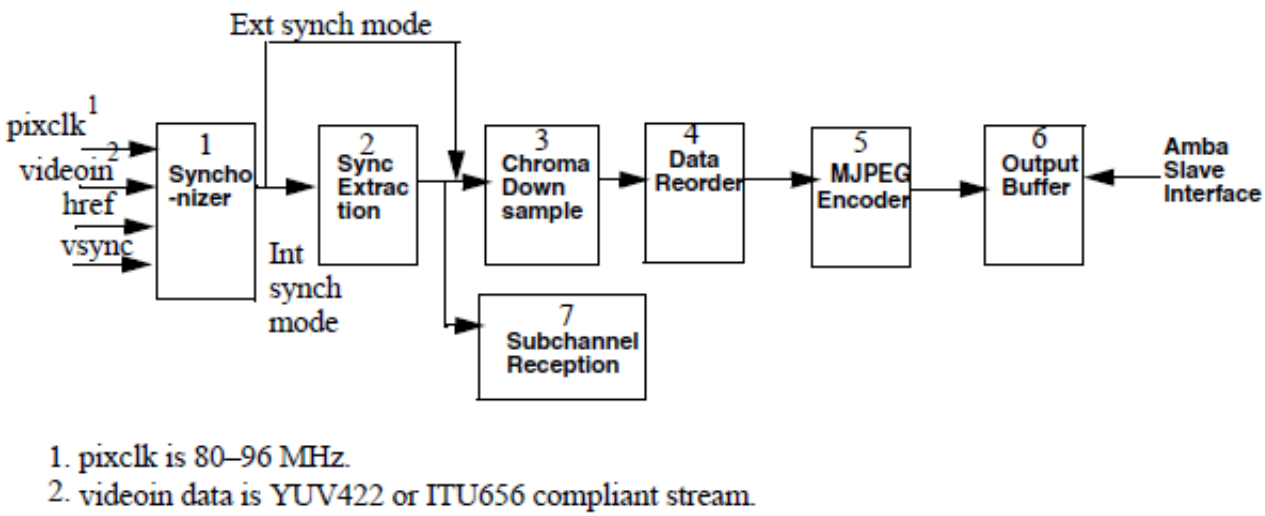


Figure 1. Video Encoder block diagram

1. Synchronizer – data synchronization with peripheral system clock (ipg_video_clk)
2. Sync Extraction – extraction of vsync and href in case of ITU-BT656. If the video stream is not ITU-BT656 compliant, video data is directly fed to the third block.
3. Chroma Down Sample - converts the 4:2:2 stream to a 4:2:0 stream by providing interpolation on the chroma components of the stream.
4. Data Reorder - performs the reordering of the stream from scan line order to MCU block order.
5. MJPEG Encoder - video stream is MJPEG encoded.
6. Output Buffer - the encoded stream is written to a circular buffer SRAM.
7. Subchannel reception – read 256 bytes of YUV422 video data which can be used for reading subchannel information or debugging purposes.

MPC5604E RAM memory is designed for processing. At the input buffer side the RAM can hold up to 16 lines which needs to be buffered for pixel reordering. At the output buffer side the RAM is designed to hold compressed data for 16 lines @ ~100Mbits/sec. Storing a complete image at high bitrates requires ~200KB and more. Processing of the raw image data, for example, image scaling, is not supported by the MPC5604E. It is also not possible to view uncompressed video through the MPC5604E. Only a snapshot of 8-bit data with 256 bytes can be done using the snapshot feature. Video Encoder module has an 8KB output buffer which stores the output data. Fast Ethernet controller (FEC) can be directly read from this buffer which means the main SRAM memory is bypassed.

CAUTION

It is NOT possible to store complete pictures in the internal memory because it is a functional safety requirement, since use case is a driver assistance system here and in worst case, it resends an old outdated image.

4 Camera input interface

The input interface accepts ITU-BT656 mode data with external HREF/VSYNC sync inputs. In case mode is set to ITU-BT656, syncing is embedded per ITU-BT656 specification in the 8 most significant data lines. In case HREF/VSYNC syncing is enabled, HREF and VSYNC signal are input. HREF is active when line pixel data is valid, VSYNC becomes active during the vertical blanking. For more information about External Sync Interface or ITU-BT656 see MPC5604E Reference manual.

4.1 Input interface pixel data format

Input interface support YUV422 pixel data. There is no support for RAW, YUV420 or gray level only video data. For the sequence YUYV the MPC5604E requires the camera to send the data in 4 cycle:

- 1st cycle: Y
- 2nd cycle: U
- 3rd cycle: Y
- 4th cycle : V

It is possible to change the luma-chroma order to UYUV. YUYV sequence which is done within two cycles and is NOT suported by MPC5604E camera interface:

- 1st cycle: YU (NOT SUPPORTED)
- 2nd Cycle: YV (NOT SUPPORTED)

MPC5604E Video Encoder is capable for 1Mp x 30fps@100MHz.

4.2 Pixel clock calculation

Let us have camera device with following parameters:

1280x800 = 1Mpixel

YUV422 = 2 cycles / pixel (1 cycle / pixel is not supported by MPC5604E)

h-blanking = 1.5x (1920 pixels)

v-blanking = 40 lines

frame rate = 30fps

pixel clock = h-resolution x (v-resolution + v-blanking) x number of cycles per pixel x frame rate x h-blanking =1280 (800+40) x 2 x 30 x 1.5 = ~96.000.000 Hz

4.3 Bit width and camera clock generation

The video interface can interface with 8-bit, 10-bit or 12-bit input. Input format is 8/10/12 bit, programmable. While our Video encoder can connect to sensors with 8,10 or 12-bit, there is no 10-bit JPEG standard. The baseline JPEG with 8-bit and the extended JPEG with 12-bit is supported. To support 10-bit camera sensors, the extended JPEG compression needs to be used. The Video encoder will pad the 2 LSBs. Camera sensor usually requires an input clock that can be provided by MPC5604E. There are following possibilities:

- Using eTimer
- Use output clock (clk_out- see MPC5604E Reference manual, Figure 8. MPC5604E Clock Selection)) feature of the clock generation module (MC_CGM).

There are also following possibilities for camera exposure synchronization:

- Using AVB real time clock (asynchronous to Camera Clock)
- Using eTimer

4.4 Signal Polarity

Input logic of Video Encoder can be configured via Status_Config register to accept various polarities of Clock, VSYNC and HREF signals. Polarity configuration of VSYNC and HREF signals also drives Sub-channel start point, from where Sub-channel data is going to be recorded. As mentioned before, the luma-chroma order can be selected for YUYV or UYVY.

5 Circular output buffer

The MJPEG Encoder store encoded data into circular output buffer of size 2048x32bits. This buffer is accessible via crossbar switch to other modules connected into it. The circular buffer is directly visible and is memory mapped from 0x5000_0000 to 0x5000_3FFF.

NOTE

Output buffer RAM should not be used to extend general purposed SRAM memory.

Circular output buffer must be configured before MJPEG encoding process starts. It’s required to configure dma_vstart_address, dma_alarm_address and enable dma_package_data_ready interrupt. For details about initialization sequence see chapter configuration sequence.

6 Huffman and Quantization tables

The MJPEG Encoder needs to be pre-configured to define the Huffman & Quantization tables, frame format, length of restart interval and others. This information is provided to the MJPEG IP using its configuration Interface. To implement this, the video encoder wrapper has a JPEGIn Buffer (256x32 bits). This buffer RAM needs to be configured with the configuration stream prior to starting the MJPEG encoder – see chapter initialization sequence. For every resolution, some dummy ppm image needs to be generated as input with any graphics tool. Important are IMAGE, QUALITY FACTOR, RESTART (number of macro blocks) and the threshold settings for the rate control. Configuration stream is then represented as exact JPEG syntax except that between SOS and EOS there are no bits.

7 Bandwidth control

In case of video data stream going into the Video encoder at the 1280×800× 30 fps×10-bit, the 307.2Mbit/s bandwidth is required to send uncompressed data through Ethernet interface. Thus, compression (JPEG) is required to shrink the data volume to below 100Mbit/s.

There are two ways to control bandwidth of the output stream on VE, one through quality factor (does not guarantee the output bandwidth) the other is bandwidth limiter (Guarantees not overrun the circular output buffer).

Data are passed through MJPEG encoder which has a configurable quality factor between 1 and 100, which should represent the human perception of picture quality. User software should implement bandwidth control algorithm, where each measures the size of each generated picture and compares that to an expected value. It then adjusts the quality factor to that value. Typical value of bandwidth after MJPEG encoder is 20 and 70Mbit/s. At 20Mbit/s the quality actually looks good already. In addition to quality factor control there is BW limiter in place which is image independent. Once it sent the maximum level, say 70Mbps, the output will not exceed irrespective of the input image.

It is recommended to implement a subset of AVB for Automotive and that standard recommends that it does not use more than 75% of a link for streaming purposes. That means application needs to limit the image streaming to 75Mbit/s maximum, many light conditions, even a quality factor of 100 will not produce 75 Mbit/s JPEG output. The bandwidth generated by the JPEG encoder is extremely dependent on the dynamics of the image which is processed.

Bandwidth limiter is controlled via Luminance and Chrominance component thresholds which are calculated following way:

JPEG standard (for YUV420) defines Macroblock with size of 16x16 pixels.

$$Bits\ per\ frame = \frac{bit\ rate}{Frame\ Per\ Second}$$

$$number\ of\ bits\ per\ macroblock = \frac{bits\ per\ frame \text{x} 16 \text{x} 16}{picture\ width \text{x} picture\ height} = 4 \text{x} luma\ threshold + 2 \text{x} chroma\ threshold$$

where 16 x 16 is a block size in pixels

$$luma\ threshold = ratio \text{x} chroma\ threshold$$

$$chroma\ threshold = \frac{number\ of\ bits\ per\ macroblock}{4 \text{x} ratio + 2}$$

Ratio represents percent of luminance and chrominance in the output image.

8 Sub-channel mode

Sub-channel mode allows access 256 bytes of uncompressed YUYV video data. This feature allows transmitting imager register values or histogram data during the vertical blanking. It also allows dumping any other usable data for debugging purpose with maximum size of 256 bytes.

Following sequence is recommended for sub-channel mode:

1. Configuration of Sub-channel Start point and Sub-channel Buffer Start Sub-channel data are recorded from different positions based on sub-channel start point bit and sub-channel buffer start configuration.
   Sub-channel Start point determines the following:

- pixel clock counting starts from the starting edge of VSYNC (Sub-channel Start Point = 0) (FIGURE2)
- pixel clock counting starts from the first valid pixel of the frame (Sub-channel Start Point = 1). First valid pixel of the frame means first HREF's starting edge after VSYNC in case of external sync mode (FIGURE3) and first HSYNC's negative edge after VSYNC in case of embedded sync mode (FIGURE4).

Sub-channel buffer start address specifies the number of pixel clocks after HREG/VSYNC from where recording data to sub-channel SRAM is expected.

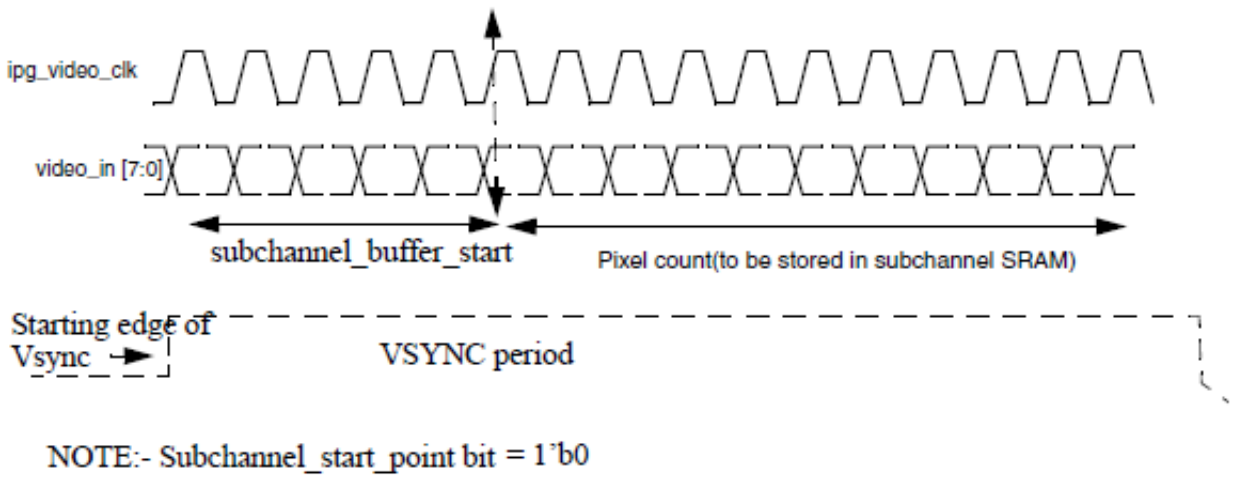


Figure 2. Sub-channel data for external/embedded sync, Sub-channel Start point = 0

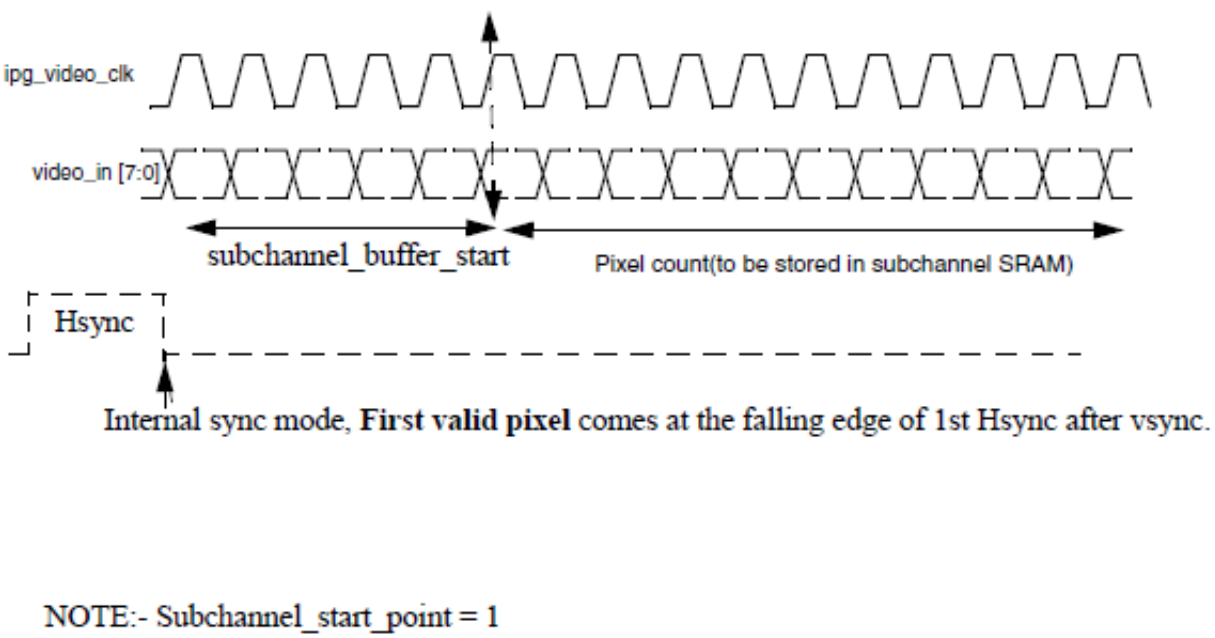
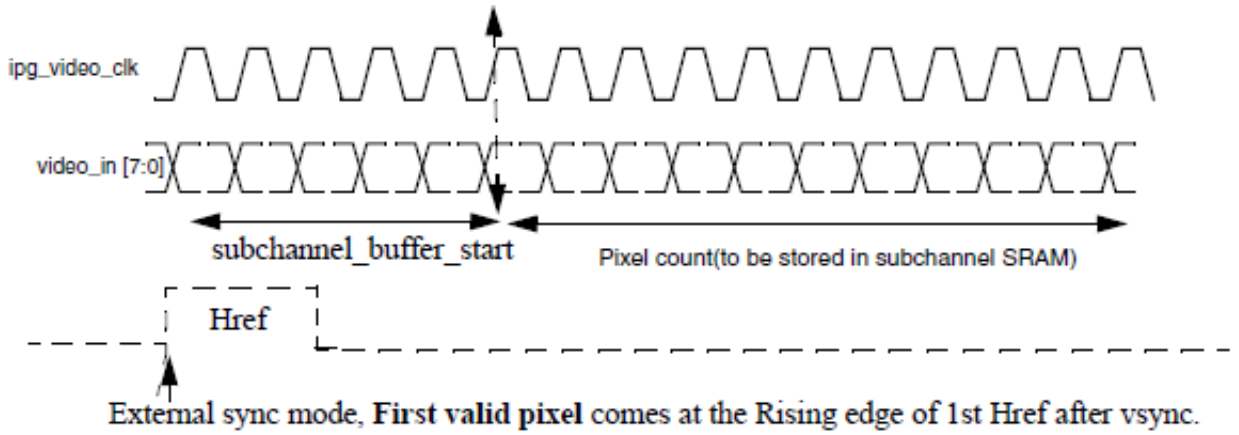


Figure 3. Embedded Sync mode for Sub-channel start point = 1



NOTE:- Subchannel_start_point = 1

Figure 4. External Sync mode for Sub-channel start point = 1

```
VE.STATUS_CONFIG.B.SUBCHANNEL_START_POINT=0; // pixel counting starts from the starting
edge of VSYNC
VE.SUBCHANNEL_BUFFER_START.B.SUBCHANNEL_BUFFER_START=80; //80 pixel clocks delay after
starting edge of VSYNC. After 80 pixel clocks data will be recorded into Sub-channel
SRAM.
```

2. Configure pixel count.

   Defines number of pixels to be stored in sub-channel SRAM.

   ```
   VE.PIXEL_COUNT.B.PIXEL_COUNT=256; // 256 bytes will be stored in the Sub-channel SRAM
   ```

3. Enable sub-channel interrupt request.
   Sub-channel Interrupt Request allows decrease Core loading by interrupt which signalize that sub-channel data are received and ready for processor read sub-channel data are successfully stored on the sub-channel memory and core can access it.

   ```
   VE.STATUS_CONFIG.B.SUBCHANNEL_IRQ_EN=1; //enable Sub-channel Interrupt Request
   ```

   INTC.PSR[181].R = 1; // Increase priority of sub-channel interrupt source #181 above priority of the current process in the Interrupt controller.

4. Enable sub-channel data request.
   This flag allows sub-channel mode know that user requests sub-channel data from the following frame. Based on the sub-channel Start point, Buffer start address and pixel count configuration sub-channel data are stored to 64x32 SRAM. User software will get information from sub-channel interrupt that sub-channel data are successfully stored and core can access it.

   ```
   VE.STATUS_CONFIG.B.SUBCHANNEL_DATA_REQUEST=1; //Request sub-channel data
   ```

9 Video Encoder memory map description

Video Encoder memory map (see MPC5604E Reference manual) can be divided into the following three sections:

Video Encoder registers: Registers in this section contains configuration registers for input logic, information about image resolution, interrupt and DMA configuration. It also contains Subchannel buffer, JPEGIn buffer and Circular output buffer spaces.

Control registers: Registers in this section control start or stop or bandwidth of MJPEG compression.

Status registers: Registers in this section shows status of MJPEG module.

Following text describes usecase of several registers in the Video Encoder.

Dma_vstart_address – user may initialize vstart of next frame by DMA_VSTART address register. For example 1st frame will have DMA_VSTART address configured to 0 (0x0 + offset of the circular output buffer). This address can be reconfigured for 2nd frame if needed.

Dma_vend_address – status register which correspond to the address where current frame ends in the circular output buffer.

Dma_alarm_address – this register contains watermark of the circular output buffer which may generate interrupt, if enabled in status_config register. Within this interrupt SW should start emptying output buffer. Subchannel_buffer start address – specifies the number of pixel clocks after HREG/VSYNC from where recording data to subchannel SRAM is expected.

JPEG in offset address – contains address of JPEG configuration stream in the JPEGIn buffer space. Change of JPEG in offset address can be used in the way when change of quality factor is required, it is possible to use different JPEG configuration streams located on different addresses and point MJPEG encoder to correct or change configuration with use of JPEG in offset address.

10 Video Encoder interrupt sources

Video Encoder has 6 interrupt sources available. These interrupt sources can handle more than one interrupt event from Video Encoder. Following table shows relationship between interrupt sources of Interrupt controller (INTC) and interrupt events of Video Encoder.

Interrupt source number in INTC	Resource	Description
179	Vertical Image Start	Interrupt signaling the start of encoding of current frame – Start of Image (SOI) marker of the MJPEG Encoder i.e. the MJPEG is configured for encoding. (VE.STATUS_CONFIG.VSTART_IRQ)
180	Vertical Image End	Interrupt signaling the completion of encoding of current frame. It is asserted at the detection of End of Image (EOI). (VE.STATUS_CONFIG.VEND_IRQ)
181	Sub-channel Ready	Subchannel data received and ready for processor read. Subchannel data are successfully stored on the Subchannel memory and core can access it. (VE.STATUS_CONFIG.SUBCHAN_IRQ)
182	Package Data Ready	DMA TXFER - DMA transfer request - given every time when dma_address >= dma_alarm_address
183	INT	ALL interrupts
184	ERR	all errors (length error, count error)

11 Initialization Sequence

User should follow initialization sequence for correct initialization of Video Encoder module. Following configuration sequence will configure Video Encoder for video stream with resolution 1280x800, 8bit.

1. Input and output pin configuration

   It is required to configure System Integration Unit Light (SIUL) and required pins and their direction.

```
SIUL.PCR0.R  = 0x0107;   // Input Buffer enabled, Slew Rate Control set for fastest
configuration. VID_D11 – Input
SIUL.PCR1.R  = 0x0107;  // VID_D10 - Input
SIUL.PCR2.R  = 0x0107;  // VID_D9 - Input
SIUL.PCR3.R  = 0x0107;  // VID_D8 - Input
SIUL.PCR4.R  = 0x0107;  // VID_D7 - Input
SIUL.PCR5.R  = 0x0107;  // VID_CLK - Input
SIUL.PCR6.R  = 0x0107;  // VSYNC, IBE enabled, Slew Rate Control set for fastest
configuration.
SIUL.PCR7.R  = 0x0107; // HREF, IBE enabled, Slew Rate Control set for fastest
configuration.
//PSMI settings for Video
SIUL.PSMI[24].B.PADSEL = 1; // VID_D0 Input connected to PCR34, PCR6 used for VSYNC
SIUL.PSMI[25].B.PADSEL = 1;  // VID_D1 Input connected to PCR35, PRC7 used for HREF
SIUL.PCR8.R  = 0x0107; // Input Buffer enabled, Slew Rate Control set for fastest
configuration.  VID_D6 - Input
SIUL.PCR9.R  = 0x0107;  // VID_D5 - Input
SIUL.PCR10.R= 0x0107;  // VID_D4 - Input
SIUL.PCR11.R= 0x0107;  // VID_D3 - Input
SIUL.PCR12.R= 0x0107;  // VID_D2 - Input
SIUL.PCR34.R= 0x0107;  // VID_D0 - Input
SIUL.PCR35.R= 0x0107;  // VID_D1 - Input
```

2. Clock source for Camera input clock – in our example Clock Generation Module (MC_CGM) is used to generate required clock source for camera interface via clk_out pad. The maximum frequency supported by clk_out pad is 32 MHz. Camera interface should increase this frequency using its own PLL to required pixel clock frequency.
3. Configuration of input circuit of Video Camera module.

```
VE.PICTURE_SIZE.B.PICTURE_HSIZE= 0x28; // Number of pixels in line. Value written to
the register should be multiplication of 32. 40 × 32 = 1280 pixels
VE.PICTURE_SIZE.B.PICTURE_VSIZE= 0x32; // Number of lines in field. Value written to
the register should be multiplication of 16. 50 × 16 = 800 lines
VE.STATUS_CONFIG.B.SYNC_MODE=0; // Defines whether ITU-BT656 like embedded sync
(SYNC_MODE=1)or Hsync/Vsync external signals are used for syncing (SYNC_MODE=0)
VE.STATUS_CONFIG.B.BIT_WIDTH_IN=0; // 8 bits WidthIn is used for data transfer.
VE.STATUS_CONFIG.B.PIXEL_ORDER=0; // Defines whether first chroma pixel or luma pixel
will come in the camera input stream. If PIXEL_ORDER = 0, luma pixel first in YCbCr
stream.
VE.STATUS_CONFIG.B.VSYNC_POL=1; // Defines the active polarity of VSYNC signal. If
VSYNC_POL = 1, VSYNC is active high.
VE.STATUS_CONFIG.B.HREF_POL=1; // Defines the active polarity of HREF signal. If
HREF_POL = 1, HREF is active high.
VE.STATUS_CONFIG.B.PIXEL_CLOCK_POLARITY=1; // Defines whether data is sampled on
positive or negative edge of pixel clock. If PIXEL_CLOCK_POLARITY = 1, data is sampled
on positive edge of pixel clock.
```

4. This JPEG configuration stream is compliant to the JPEG Syntax. Configuration sections are marked in the JPEG syntax by 0xFF in the bit stream. Given below are some typical configurations that are required to be adjusted to the application are given. The start codes for these fields are marked with gray background.

   Frame Size: This is part of the “Start Of Frame” (SoF) header indicated by 0xFFC0 for 8 bit mode and 0xFFC1 for 12 bit mode. The header is followed by 0x0011 and 0x08/0x0C (8-bit/12 bit mode) followed by 16 bit values for the image geometry (gray marked), first Y-size (0x0320 = ‘d800) followed by X-size (0x0500 = ‘d1280), i.e., 1280x800 pixel resolution.

Quantization tables: These DQT sections are indicated by 0xFFDB header. The header is followed by the table length and precision (0x00430 for 8 bit quantization values used for 8 or 12 bit mode or 0x00831 for 16 bit quantization values used for the 12-bit mode), the quantization table identifier (0x0 typically used for the Y-channel or 0x1 typically used for the U/V-channel), and 64 quantization values (each 8bit or 16bit) in zigzag order. The first of the 64 quantization values is also marked gray.

Restart Interval: The DRI section is indicated by the 0xFFDD marker. The marker is followed by 0x0004 and the Restart Interval in Minimum Coding Units (16x16 pixel blocks for YUV420). In the given example this value is 0x0050 = ‘d80 MCUs corresponding to one 16 lines row in the image. The Restart Interval is optional and can be set to any value independently of the X-size of the image. Using the Restart interval allows to decoder to recover during a frame in case a bit error happened.

The following gives an example for a JPEG configuration stream.

```
uint32_t JPEG_Conf_Stream[] = { 0xffd8ffe0, 0x00104a46, 0x49460001, 0x01000001,
0x00010000, 0xffffffff, 0xdb004300, 0x100b0c0e, 0x0c0a100e, 0x0d0e1211, 0x10131828,
0x1a181616, 0x18312325, 0x1d283a33, 0x3d3c3933, 0x38374048, 0x5c4e4044, 0x57453738,
0x506d5157, 0x5f626768, 0x673e4d71, 0x79706478, 0x5c656763, 0xffffffff, 0xdb004301,
0x11121218, 0x15182f1a, 0x1a2f6342, 0x38426363, 0x63636363, 0x63636363, 0x63636363,
0x63636363, 0x63636363, 0x63636363, 0x63636363, 0x63636363, 0x63636363, 0x63636363,
0x63636363, 0x63636363, 0xffc00011, 0x08032005, 0x00030122, 0x00021101, 0x031101ff,
0xc4001f00, 0x00010501, 0x01010101, 0x01000000, 0x00000000, 0x00010203, 0x04050607,
0x08090a0b, 0xffc400b5, 0x10000201, 0x03030204, 0x03050504, 0x04000001, 0x7d010203,
0x00041105, 0x12213141, 0x06135161, 0x07227114, 0x328191a1, 0x082342b1, 0xc11552d1,
0xf0243362, 0x7282090a, 0x16171819, 0x1a252627, 0x28292a34, 0x35363738, 0x393a4344,
0x45464748, 0x494a5354, 0x55565758, 0x595a6364, 0x65666768, 0x696a7374, 0x75767778,
0x797a8384, 0x85868788, 0x898a9293, 0x94959697, 0x98999aa2, 0xa3a4a5a6, 0xa7a8a9aa,
0xb2b3b4b5, 0xb6b7b8b9, 0xbac2c3c4, 0xc5c6c7c8, 0xc9cad2d3, 0xd4d5d6d7, 0xd8d9dae1,
0xe2e3e4e5, 0xe6e7e8e9, 0xeaf1f2f3, 0xf4f5f6f7, 0xf8f9faff, 0xc4001f01, 0x00030101,
0x01010101, 0x01010100, 0x00000000, 0x00010203, 0x04050607, 0x08090a0b, 0xffc400b5,
0x11000201, 0x02040403, 0x04070504, 0x04000102, 0x77000102, 0x03110405, 0x21310612,
0x41510761, 0x71132232, 0x81081442, 0x91a1b1c1, 0x09233352, 0xf0156272, 0xd10a1624,
0x34e125f1, 0x1718191a, 0x26272829, 0x2a353637, 0x38393a43, 0x44454647, 0x48494a53,
0x54555657, 0x58595a63, 0x64656667, 0x68696a73, 0x74757677, 0x78797a82, 0x83848586,
0x8788898a, 0x92939495, 0x96979899, 0x9aa2a3a4, 0xa5a6a7a8, 0xa9aab2b3, 0xb4b5b6b7,
0xb8b9bac2, 0xc3c4c5c6, 0xc7c8c9ca, 0xd2d3d4d5, 0xd6d7d8d9, 0xdae2e3e4, 0xe5e6e7e8,
0xe9eaf2f3, 0xf4f5f6f7, 0xf8f9faff, 0xdd000400, 0x50ffda00, 0x0c030100, 0x02110311,
0x003f00ff, 0xd9000000};
```

size_JPEG_Conf = sizeof(JPEG_Conf_Stream); // get size of JPEG configuration stream which vary for different resolution

for (i = 0; i < (size_JPEG_Conf/4); i++)

(*((volatile unsigned*) (JPEG_IN_BUFFER_START_ADDR+4*i))) = JPEG_Conf_Stream[i]; //Load JPEG configuration stream to JPEGIn buffer space. Based on Video Encoder memory map JPEG_IN_BUFFER_START_ADDR= 0xFFFF9000.

VE. JPEG_IN_OFFSET.B. JPEGIN_DATA_OFFSET = 0 // VE_JPEG_DATA_IN_OFFSET- offset to buffer RAM where JPEG in stream will be sourced next time encoder request

5. Enable write of Output Buffer and enable Video Encoder.

```
VE.STATUS_CONFIG.B.BUFFER_WRITE_ON=1; //
Enable write of Output Buffer through VE. This also enables read access to output
buffer, once Video Encoder is on.
VE.STATUS_CONFIG.B.VIDEO_ENCODER_ON=1; // Turn video encoder on
```

6. Program parameters and required interrupt for data handling with use Output buffer.

```
VE.DMA_ALARM_ADDRESS.R = 0x400; // Programming DMA Alarm Address
VE.STATUS_CONFIG.B. DMA_TXFER_IRQ_EN = 1; //Enable DMA transfer interrupt at Video
Encoder
INTC.PSR[182].R = 0x02; // Increase priority of DMA_TXFER_IRQ above priority of the
current process.
VE.CFG_MODE.R = 0xFF; // Configure bitmapped register that controls JPEGE-X output
stream generation mode.
See MPC5604E Reference manual for details.
VE.MODE.R = 0xE0; // Configure bitmapped register that controls JPEGE-X operation mode.
Write 0xE0 value to the Video Encoder mode register cause, that the MJPEG core enters
configuration mode, afterwards, the core exits idle mode and baseline sequential mode
of operation is selected for 8-bit width in. For more information see MPC5604E
Reference manual.
```

As soon as Video Encoder receive expected Video data and Synchronization signals, Packet Data Ready interrupt #182 occurs when DMA thresh hold crossed. It is upto the user application to handle data from the output buffer.

12 Conclusion

This application note describes basic configuration sequence for Video Encoder Wrapper module. Freescale delivers Freescale Software for Surround Camera System using MPC5604E based on Autosar 3.0 RTOS for MPC5604E. This professional software solution for Ethernet camera system using MPC5604E offers the following main features: Stream Builder Interface module, with support for UDP and IEEE1722 (AVB), IP and Precision Time Protocol (PTP) Stack, Video Encoder and I2C drivers, Imager configuration and control drivers and others.

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 10 5879 8000
support.asia@freescale.com

Document Number: AN4514
Rev. 1, 05/2012