TMS320DM642 EVM OSD FPGA User's Guide

Literature Number: SPRU295
June 2003

TEXAS INSTRUMENTS

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Preface

Read This First

About This Manual

The document gives explains how to use the TMS320DM642 evaluation module with the on-screen display of the field−programmable gate array.

Notational Conventions

This document uses the following conventions.

❑ Program listings, program examples, and interactive displays are shown in a `special typeface` similar to a typewriter's. Examples use a **`bold version`** of the special typeface for emphasis; interactive displays use a **`bold version`** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

  Here is a sample program listing:

```
0011  0005  0001         .field    1, 2
0012  0005  0003         .field    3, 4
0013  0005  0006         .field    6, 3
0014  0006               .even
```

  Here is an example of a system prompt and a command that you might enter:

```
C:  csr −a /user/ti/simuboard/utilities
```

❑ In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

  .asect **"***section name***",** *address*

  .asect is the directive. This directive has two parameters, indicated by *section name* and *address*. When you use .asect, the first parameter must be an actual section name, enclosed in double quotes; the second parameter must be an address.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you do not enter the brackets themselves. Here is an example of an instruction that has an optional parameter:

  LALK *16–bit constant [, shift]*

  The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

  Square brackets are also used as part of the pathname specification for VMS pathnames; in this case, the brackets are actually part of the pathname (they are not optional).

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

  ```
  { * | *+ | *− }
  ```

  This provides three choices: `*`, `*+`, or `*−`.

  Unless the list is enclosed in square brackets, you must choose one item from the list.

- Some directives can have a varying number of parameters. For example, the .byte directive can have up to 100 parameters. The syntax for this directive is:

  .byte *value$_1$ [, ... , value$_n$]*

  This syntax shows that .byte must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

Related Documentation From Texas Instruments

The following books describe the TMS320VC5509 and related support tools. To obtain a copy of any of these TI documents, go to the TI website: www.ti.com.

TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor Data Manual (SPRS200) provides timing and electrical specifications for the TMS320DM642.

TMS320C6201/C6701 Peripherals Reference Guide (literature number SPRU190) describes common peripherals available on the TMS320C6201 and TMS320C6701 digital signal processors. This book includes information on the internal data and program memories, the external memory interface (EMIF), the host port interface (HPI), multichannel buffered serial ports (McBSPs), direct memory access (DMA), enhanced DMA (EDMA), expansion bus, clocking and phase-locked loop (PLL), and the power-down modes.

Related Documentation From Spectrum Digital

DM642 EVM Technical Reference (Spectrum Digital) describes the evaluation module for the Texas Instruments TMS320DM642.

Trademarks

Trademarks are the property of their respective owners.

This page is intentionally left blank.

Contents

Figures

Tables

TMS320DM642 OSD FPGA EVM

This document describes the operation of the on-screen display (OSD) field programmable gate array (FPGA) used on the TMS320DM642 evaluation module (EVM). The FPGA is a Xilinx XC2S300E−6PQ208C.

1 Features

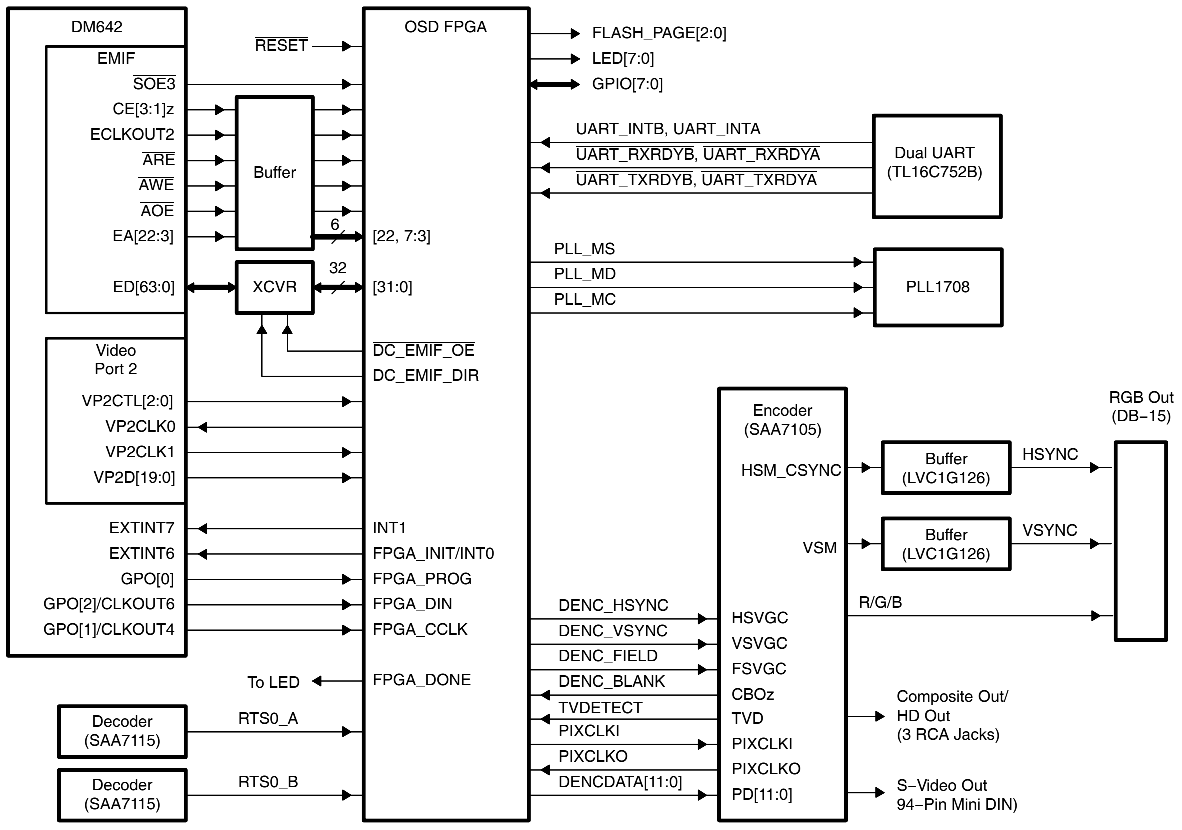
The OSD FPGA has the following features:

- ❑ Controls eight light-emitting diodes (LEDs) accessible by the DM642 external memory interface (EMIF) through a register
- ❑ Controls eight general-purpose input/output (GPIO) bits through registers accessible by the DM642 EMIF
- ❑ Generate control signals (DIR and OEz) for EMIF buffers
- ❑ Provides serial control interface to PLL1708.
- ❑ Generates three page bits for the FLASH
- ❑ Implements an interrupt control logic that monitors interrupt signals from the dual UART and video decoder and generates an edge-triggered interrupt to the DM642
- ❑ Provides an interface between the DM642 video port and the SAA7105 video encoder. This interface can operate in the following modes:
  - ■ Transparent mode where the 8-bit video output from the DM642 video port is passed directly to the SAA7105 video encoder.
  - ■ Performs 2:1 multiplex operation to support SVGA and HD. The DM642 video port outputs 16-bit video data on the rising edge of the clock to the OSD FPGA. The OSD FPGA converts this to dual clock edge 8-bit video data where the upper 8 bits are clocked out on the rising edge of the clock and the lower 8 bits on the falling edge.
  - ■ One-bit alpha blending with 7-bit color look-up table (CLUT) for 8-bit video output from the DM642. The alpha blended video data is then passed on to the SAA7105 video encoder.
  - ■ One-bit Alpha blending with 7-bit CLUT for 16-bit video output from the DM642 to support HD mode of operation. The alpha blended video

data is converted to dual clock edge 8-bit data and passed on to the SAA7105 video encoder.

1.1 OSD FPGA System Block Diagram

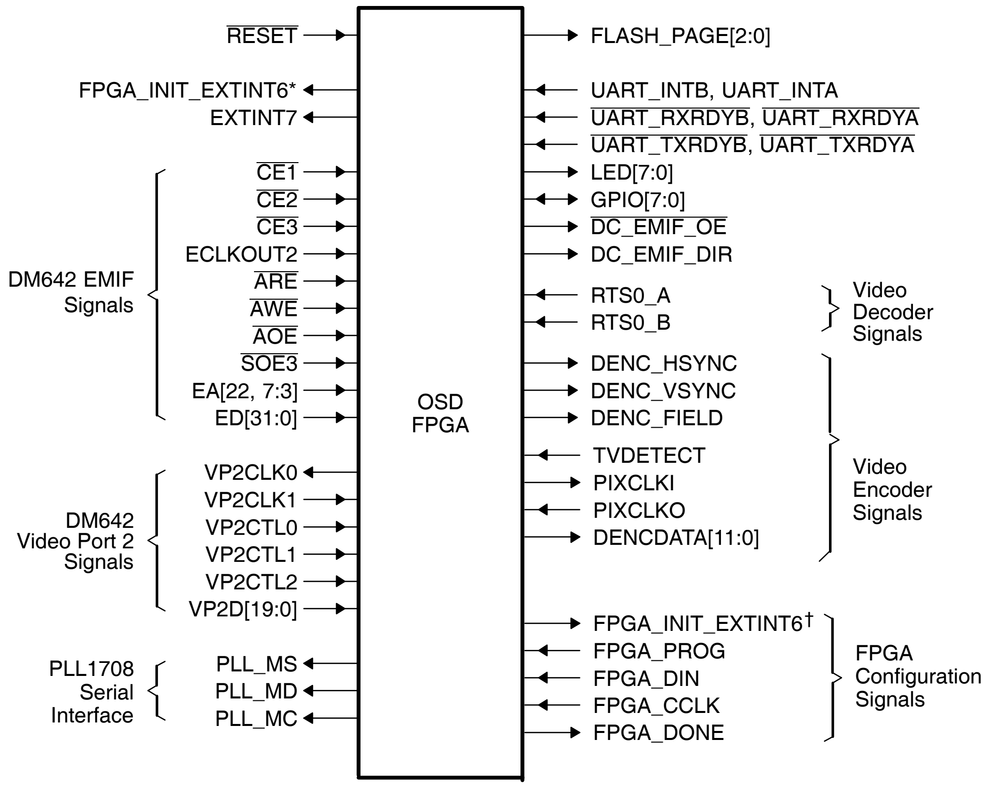
Figure 1. OSD FPGA System Block Diagram



1.2 OSD FPGA Signals

Figure 2 shows all the signals going to the OSD FPGA and Table 1 describes the signals.

Figure 2. OSD FPGA Signals



† FPGA_INIT_EXTINT6 is used as the FPGA INIT signal for configuration when FPGA_PROG is pulled low. When FPGA_PROG is high, this pin is used as EXTINT6 to the DSP.

Table 1. Signal Definitions

Signal	Type	Description
		Miscellaneous Signals
RESET	I	Asynchronous system reset
FLASH_PAGE[2:0]	O	Page bits for flash on EVM
LED[7:0]	O	LED outputs which drive LEDs on EVM

Table 1. Signal Definitions (Continued)

Signal	Type	Description
GPIO[7:0]	I/O	General purpose inputs/outputs which go to the daughtercard connectors on EVM
$\overline{\text{DC\_EMIF\_OE}}$	O	Output enable for transceiver on EMIF data bus
DC_EMIF_DIR	O	Direction control for transceiver on EMIF data bus
		DM642 Interrupts
EXTINT6	O	Edge-driven external interrupt input to the DM642.
EXTINT7	O	Edge-driven external interrupt input to the DM642.
		DM642 EMIF Signals
$\overline{\text{CE1}}$	I	EMIF memory space 1 enable. This memory space is used for all asynchronous interfaces on the EVM. This includes the flash, Dual UART, and the OSD FPGA registers. This bit is also used to generate the $\overline{\text{DC\_EMIF\_OE}}$ signal. The CE1 memory space is configured for 8-bit wide asynchronous memory interface.
$\overline{\text{CE2}}$	I	EMIF memory space 2 enable. This bit is also used to generate the $\overline{\text{DC\_EMIF\_OE}}$ signal. The CE2 memory space is reserved for the daughtercard interface and can be configured anyway the daughtercard needs.
$\overline{\text{CE3}}$	I	EMIF memory space 3 enable. This memory space is used for synchronous assesses to the OSD FPGA and daughtercard interface. This bit is also used to generate the $\overline{\text{DC\_EMIF\_OE}}$ signal. The CE3 memory space is configured for 32-bit wide synchronous memory interface.
ECLKOUT2	I	EMIF output clock 2. Used for synchronous interface on memory space 3.
$\overline{\text{ARE}}$	I	EMIF asynchronous memory read enable/programmable synchronous interface address strobe or read enable
		DM642 EMIF Signals
$\overline{\text{AWE}}$	I	EMIF asynchronous memory write enable/programmable synchronous interface write enable
$\overline{\text{AOE}}$	I	EMIF asynchronous memory output enable/programmable synchronous interface output enable
$\overline{\text{SOE3}}$	I	EMIF synchronous memory output enable for memory space $\overline{\text{CE3}}$
EA[22, 7:3]	I	EMIF address bus

Table 1. Signal Definitions (Continued)

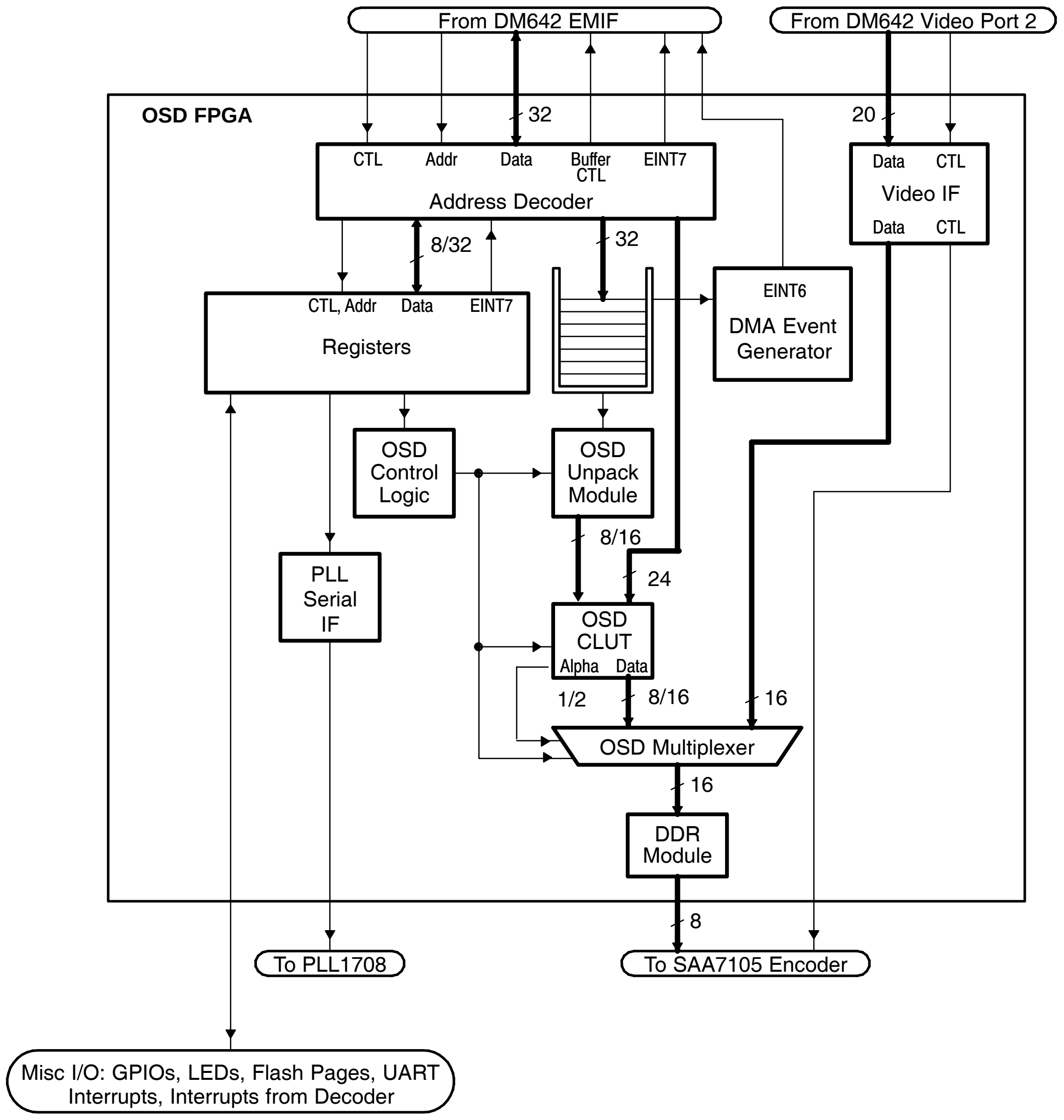
Signal	Type	Description
ED[31:0]	I/O	EMIF data bus
		DM642 Video Port 2 Signals
VP2CLK0	O	Video port 2 clock 0. Input to DM642 video port.
VP2CLK1	I	Video port 2 clock 1. Output from DM642 video port.
VP2CTL0	I	Video port 2 control 0. Is used to provide HSYNC to OSD FPGA.
VP2CTL1	I	Video port 2 control 1. Is used to provide VSYNC to OSD FPGA.
VP2CTL2	I	Video port 2 control 2. Is used to provide FIELD to OSD FPGA.
VP2D[19:0]	I	Video port 2 data bus. Output from DM642 to OSD FPGA.
		Video Encoder Signals
TVDETECT	I	Interrupt if TV is detected at DAC output of SAA7105
PIXCLKI	O	Pixel clock output from OSD FPGA to SAA7105
PIXCLKO	I	Pixel clock input to OSD FPGA from SAA7105
DENCDATA[11:0]	O	Video data
		Video Decoder Signals
RTS0_A	I	Real time status or sync information from SAA7115
RTS0_B	I	Real time status or sync information from SAA7115
		PLL1708 Serial Interface
PLL_MS	O	Chip select for serial control
PLL_MD	O	Data for serial control
PLL_MC	O	Bit clock for serial control
		Dual UART Interface
UART_INTB, UART_INTA	I	Interrupts from dual UART chip
$\overline{\text{UART\_RXRDYB}}$, $\overline{\text{UART\_RXRDYA}}$	I	Receive ready from dual UART chip
$\overline{\text{UART\_TXRDYB}}$, $\overline{\text{UART\_TXRDYA}}$	I	Transmit ready from dual UART chip

Table 1. Signal Definitions (Continued)

Signal	Type	Description
		FPGA Configuration Signals
FPGA_INIT	O	Delay configuration, indicate configuration clearing or error
FPGA_PROG	I	Asynchronous reset to configuration logic
FPGA_DIN	I	Serial configuration data input
FPGA_CCLK	I	Configuration clock
FPGA_DONE	O	Configuration status and start-up control

2 Architecture

Figure 3. OSD FPGA Architecture



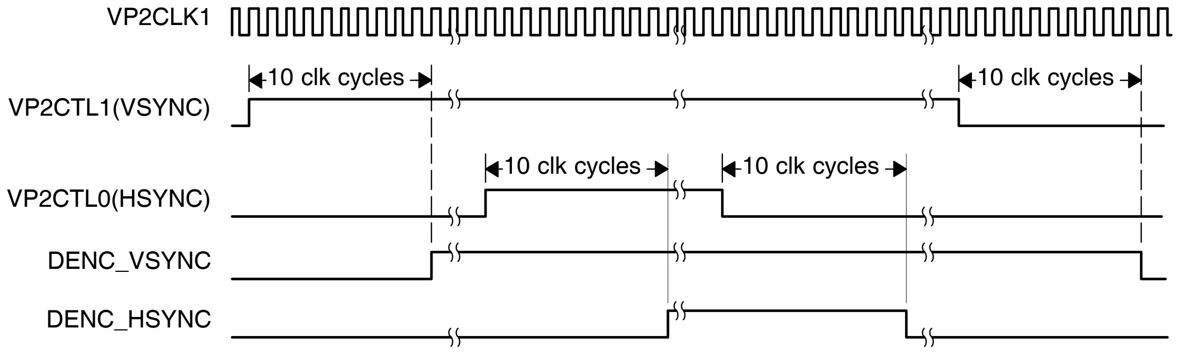
The OSD FPGA interfaces to the DM642 EMIF and video port 2. The FPGA also interfaces to the video encoder, clock PLL, dual UART, GPIOs, and LEDs. The different modules in the OSD FPGA and their functionality are described below.

2.1 Video Interface (IF)

The Video IF interfaces to the DM642 video port 2. The video port can be configured as either an 8-bit or 16-bit display interface. The Video IF module registers all the data and control signals received from the video port. The Video IF module provides data to the OSD MUX module and the control signals to the OSD Control Logic as well as to the SAA7105 encoder external to the FPGA.

The design in the FPGA adds a 10-clock latency to the video data. The control signals are also delayed by 10 clock cycles to match the data delay. See Figure 4.

Figure 4. Video Syncs



2.2 Address Decoder

The address decoder interfaces to the DM642 EMIF interface. The address decoder module registers all the incoming data and address signals and does first level of address decoding. It divides the CE1 address space into space for the Flash, space for dual UART, and space for asynchronous registers inside the OSD FPGA. It also divides the CE3 address space into space for external synchronous logic, space for FIFOs inside the FPGA, and space for synchronous registers inside the OSD FPGA.

2.3 Registers

The registers section actually includes two register modules, one for synchronous registers and the other for asynchronous registers.

The synchronous registers can be 32 bits wide. Though the synchronous registers module has logic for both reads and writes of these registers, the read back of these registers is reliable only up to ECLKOUT2 clock speed of 70 MHz. The synchronous registers include a test register and a clock PLL data register.

The asynchronous registers are all 8 bits wide. The asynchronous registers module also includes interrupt control logic and logic to control both LEDs and GPIOs.

2.4 On-Screen Display (OSD) Data First-In, First-Out (FIFO)

The OSD data FIFO is 256 words deep and 32 bits wide. The address decoder module controls the write enable and writes data to the OSD data FIFO. The unpack module controls the read enable and reads the FIFO data output. The OSD data FIFO can be written to by writing to the address 0xB000 0040 in the CE3 address space.

2.5 Direct Memory Access (DMA) Event Generator

The DMA event generator module monitors the OSD data FIFO usage and the number of events generated per field. It generates a DMA event whenever the FIFO has space for a DMA equal to the DMA threshold size, the number of events generated for the current field has not exceeded the number of events specified in Events Per Field Register, and event generation has been enabled by the OSD control logic. Once a DMA event is generated, the DMA event generator stops event generation until the DMA requested is completed. The DMA event generator counts the writes made to the OSD data FIFO and when the counter reaches the DMA threshold size, event generation is reenabled. See Section 4 for further explanation.

2.6 OSD Control Logic

The OSD control logic controls the behavior of the OSD functionality in the OSD FPGA. It controls the OSD unpack module, the DMA event generator, and the OSD MUX. The OSD control logic monitors the video control signals VSYNC and AVID to control the different OSD modules. Provided the OSD data FIFO is not empty, the OSD data is multiplexed within the OSD window specified by the OSD XSTART, OSD YSTART, OSD XSTOP and OSD YSTOP registers. Within the OSD window, the OSD control logic enables reading of OSD data FIFO, unpacking of OSD data, and multiplexing of video and OSD data.

The OSD control logic provides a control signal to control the OSD MUX, unpack signal to the OSD unpack module, and event enable signal to the DMA event generator.

Figure 5. OSD State Machine Diagram

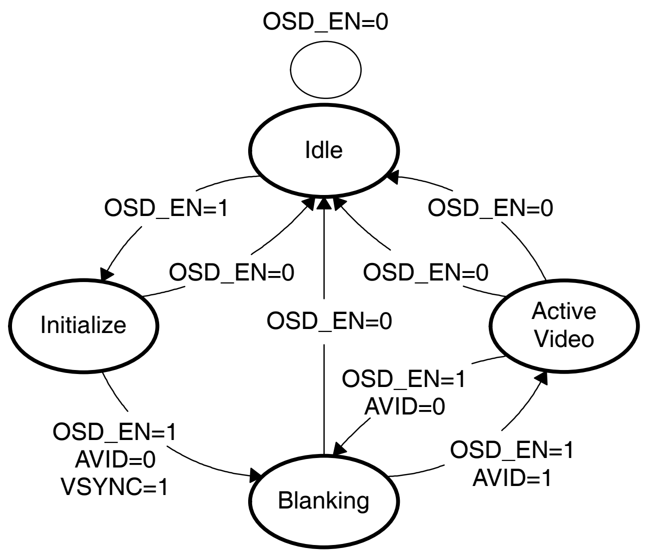


Table 2. OSD State Machine State Deviations

State	Description
Idle	The state machine idles in this state till OSD_EN goes active. The video data is passed through the FPGA when the state machine is in the Idle state.
Initialize	Whenever OSD_EN goes active, this state is reached. In this state, the state machine enables DMA events. The state machine waits in this state till a blanking period is detected.
Blanking	The state machine maintains the blanking period as long as AVID is zero. During this state, the FPGA passes the video through without any modifications. The first time this state is reached field detection is enabled. Once Field detection completes, the OSD functionality is enabled on Field 1 to ensure synchronization of OSD data
Active Video	The state machine maintains the active video period as long as AVID is one. If OSD functionality has been enabled, the state machine signals the OSD MUX module to multiplex the OSD data with video data based on the alpha information received in the OSD Data FIFO within the OSD window specified by the OSD XSTART, OSD YSTART, OSD XSTOP and OSD YSTOP provided the FIFO is not empty. If the FIFO is not empty, unpack signal to the OSD Unpack module is asserted.

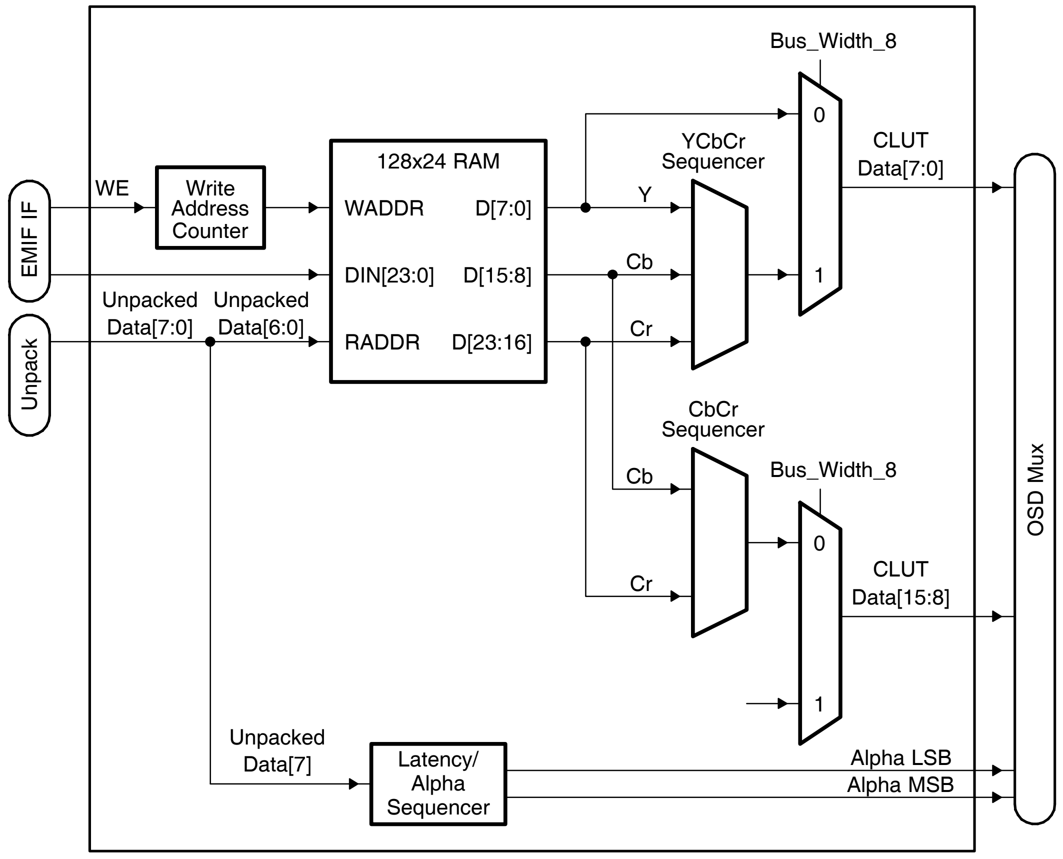
2.7 OSD Unpack Module

The OSD Unpack Module reads data from the OSD Data FIFO and unpacks it into 8-bit data. Whenever the video port bus width is 8 bits wide, the OSD Unpack Module unpacks the data into 8-bit data every other clock cycle.

Whenever the video port bus width is 16 bits wide, the OSD Unpack Module unpacks the FIFO data into 8-bit data every clock cycle. The most significant bit (MSB) of 8-bit data is used as alpha-bit for OSD functionality. The rest of the data bits are passed on to the OSD CLUT as address bits.

2.8 OSD Color Look-Up Table(CLUT)

Figure 6. Logic Diagram for OSD CLUT Module



The OSD CLUT memory is 128 words deep and 24 bits wide. Although the OSD CLUT memory is a RAM memory, it acts like a FIFO when interfacing to the EMIF. The address decoder module controls the write enable and writes data to the OSD CLUT. Whenever the CLEAR_CLUT bit in the OSD control register is set, an internal write pointer in the OSD CLUT is reset to zero. For every write to the OSD CLUT, the write pointer increments, and points to the next memory location in the OSD CLUT. It is left to the software to ensure that the correct number of writes are performed to the OSD CLUT.

The 24-bit CLUT holds Y, Cb, and Cr data for 128 different colors. Bits 7−0 are Y data, 15−8 are Cb data, and 23−16 are Cr data.

The OSD unpack module output data drives the read address of the OSD CLUT. The OSD CLUT output data is read by OSD MUX module. The OSD CLUT looks up the Y Cb Cr conversion for each pixel of unpacked data and controls the output sequence of the OSD CLUT. Output sequence for 8 bit Video is Cb, Y, Cr, Y. The sequence is Cb0, Y0, Cr0, Y1, Cb2, Y2, Cr2, Y3... etc. For 16 bit Video the luma is output from this module on the lower 8 bits and the chroma on the upper 8 bits. The sequence on Y bits is Y0, Y1, Y2..etc and on chroma bits, Cb0, Cr0, Cb2, Cr2, Cb4, Cr4.. etc.

OSD CLUT module outputs alpha captured from the MSB bit of pixel data. The alpha is MSB bit of pixel P0 for Cb0, Y0, Cr0, pixel P1 for Y1 and pixel P2 for Cb2, Y2, Cr2 P2 etc from the OSD unpack module.

The OSD CLUT can be written to by writing to address 0xB000 0044 in the CE3 space.

2.9 On-Screen Display (OSD) Multiplexer (MUX)

The OSD MUX module multiplexes the video data with OSD data. The OSD MUX module outputs OSD data if the multiplex control from the OSD control logic indicates OSD data and alpha signal coming from the OSD Unpack module is active. Otherwise, the OSD MUX module outputs video data.

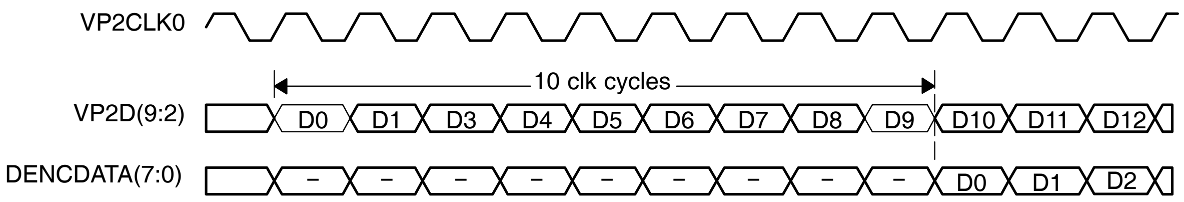
The OSD MUX module accepts either 8-bit or 16-bit wide video data. It multiplexes the appropriate OSD CLUT data with the video data.

2.10 Double Data Rate (DDR) Module

The DDR module outputs data on both the rising and falling clock edge of the clock.

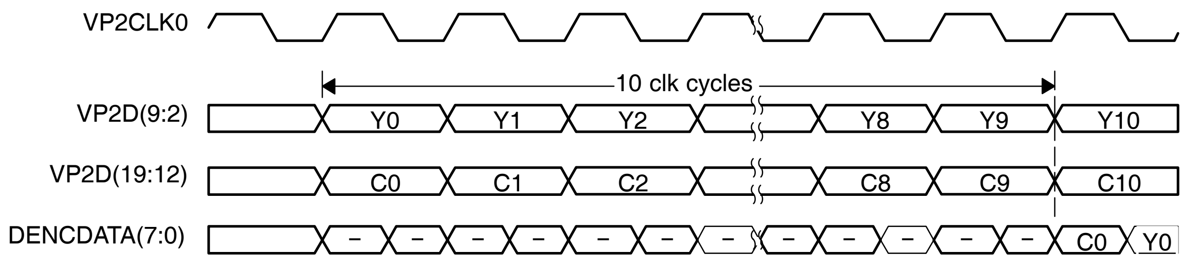
When the video data received from video port is 8-bit wide, the DDR module outputs the same data on both the rising and the falling edges of the clock. See Figure 7.

Figure 7. Video 8-bit, Single-edge Output



When the video data received from video port is 16-bit wide, the DDR module outputs the MSB bits on falling edge of the clock, and least significant bit (LSB) bits on the rising edge of the clock. For BT.656 mode, this translates to chroma on falling edge of the clock and luma on the rising edge of the clock.

Figure 8. Video 16-bit, Double-clock-edge Output



2.11 Phase-Locked Loop (PLL) Serial Interface (IF)

This module drives the serial control interface to the PLL1708.

3 EMIF Configuration

The FPGA uses two of the DM64 EMIF CE spaces. Memory space CE1 is used for asynchronous register read/writes and CE3 is used for DMA transfers and synchronous register read/writes.

3.1 Registers Read/Write

You can read/write asynchronous registers in the OSD FPGA by using the CE space CE1. This CE space should be configured as 8-bit asynchronous memory interface. The CE space CE3 is used to read and write synchronous registers in the OSD FPGA. This CE space should be configured as 32-bit programmable synchronous memory interface.

The DM642 EMIF CE Space Control Register (CE1CTL0 must have the following minimal values:

- Read Setup = 6 clock cycles
- Write Setup = 6 clock cycles
- Read strobe = 2 clock cycles
- Write strobe = 2 clock cycles
- Read Hold = 3 clock cycles
- Write Hold = 3 clock cycles
- Turn around = 3 clock cycles

See the *TMS320DM642 Video/Imaging Fixed–Point Digital Signal Processor Data Manual* (literature number SPRS200) for further information on the EMIF registers. Figure 9 and Figure 10 show the asynchronous read and write cycles.

Figure 9. DSP Asynchronous Write

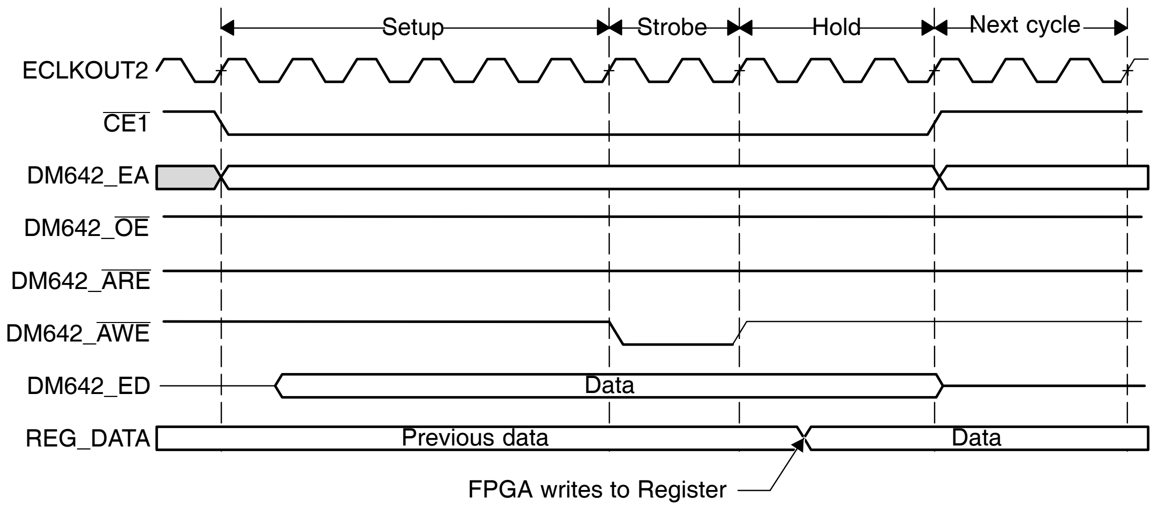
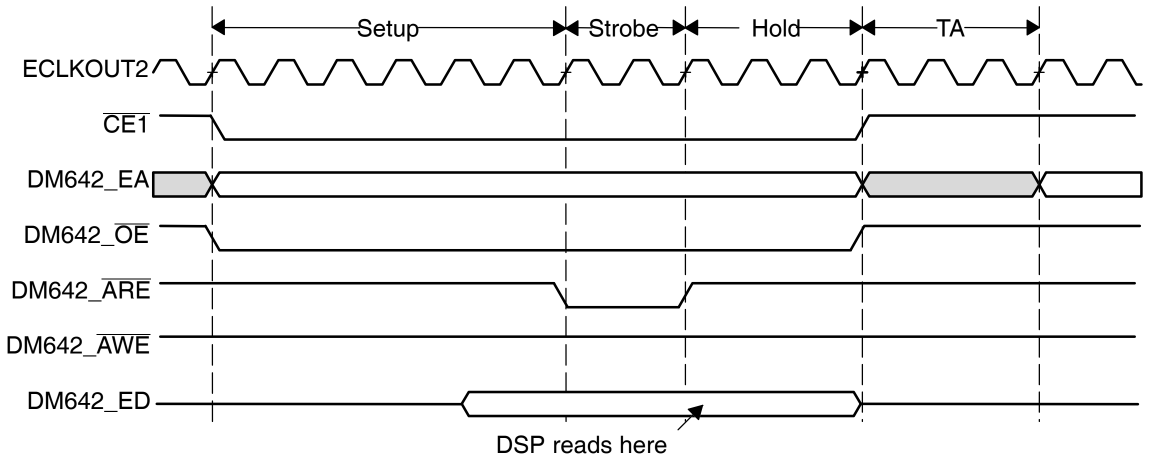


Figure 10. DSP Asynchronous Read



For CE3 space setting recommendations please see section 3.2. Figure 11 and Figure 12 show the synchronous read and write cycles.

Figure 11. DSP Synchronous Read

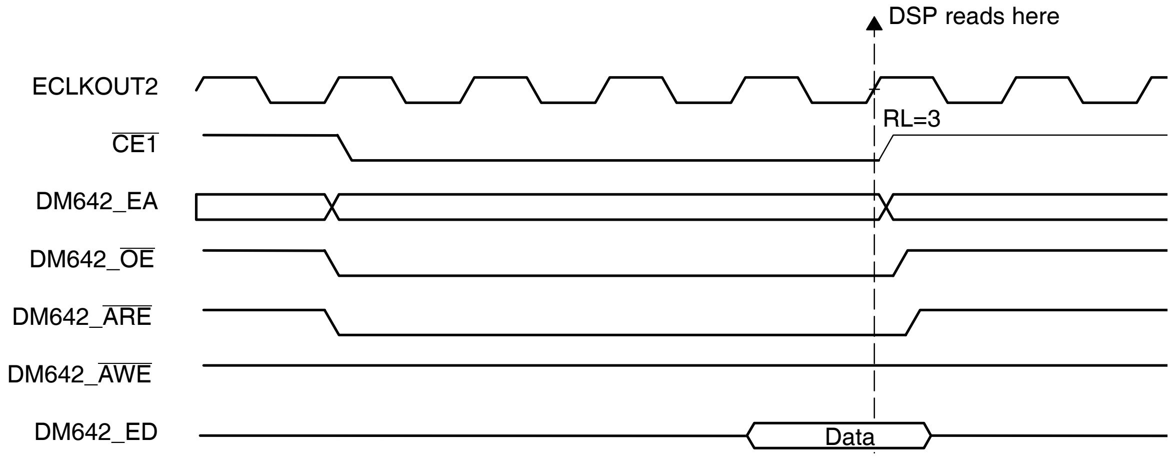
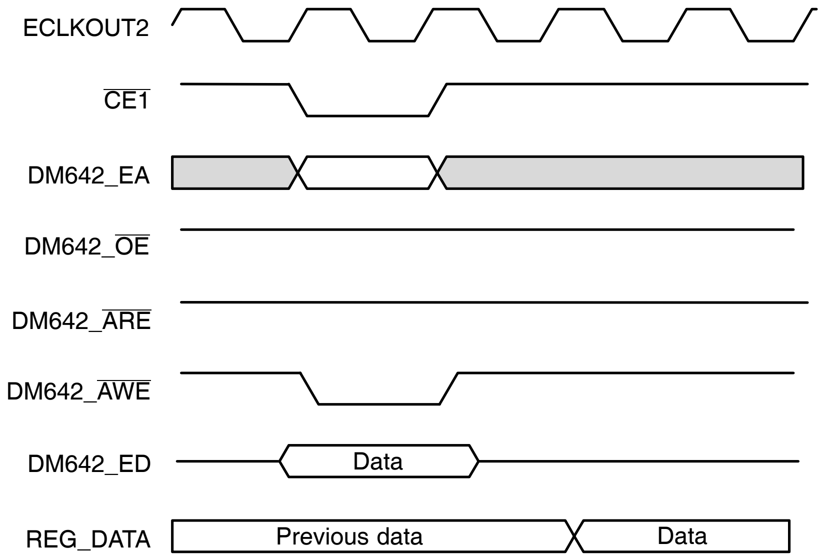


Figure 12. DSP Synchronous Write



3.2 DMA Transfers

The CE space CE3 is also used for DMA transfers to the FIFOs in the OSD FPGA and for synchronous registers in the OSD FPGA. You should configure this CE space as 32-bit programmable synchronous memory interface.

The DM642 EMIF CE Space Secondary Control Register (CESEC3, DM642 Hex address 0x0180 0054) must have the following values set. See the TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor data manual (literature number SPRS200) for further information on the EMIF registers.

- ❑ CEEXT = 1 (On read cycles, the CE signal goes active when SOEz goes active and stays active until SOEz goes inactive.)
- ❑ SYNCRL = 11 (3 cycle read latency)
- ❑ SYNCWL = 00 (0 cycle write latency)
- ❑ RENEN = 1 (Read enable mode)
- ❑ SNCCLK = 1 (Control/data for this CE space are synchronized to ECLKOUT2)

4 Display Event Generation and Processing

The OSD FPGA has one OSD Data FIFO 256 words deep. This FIFO is used for storage of OSD data. The FIFO has an address associated with it, which is write-only and is used by DMAs to fill the FIFO with OSD data. The OSD FPGA multiplexes the data from the FIFO with video data to generate the output video data stream.

The Events Per Field Register specifies the number of DMA events needed for each field. The DMA threshold registers specify the size of each DMA. A DMA event is generated whenever the number of events generated in the field are less than those specified in the Events Per Field register and the FIFO has room to receive another DMA.

The FPGA keeps track of number of words written to the FIFO. When the DMA is completed (the FIFO receives data equal to the DMA size), DMA events are re-enabled.

4.1 Endian Support and FIFO Unpacking

OSD data is always packed into the FIFOs in 32-bit words and must be unpacked before being sent to the OSD pipeline. The OSD FPGA works only in little-endian mode and data is unpacked from right to left.

4.2 Address Map

Table 3 shows the memory map of the OSD FPGA for CE1 memory space. Base address for CE1 memory space is 9000 0000. Addresses specified are in Hex.

Table 3. Memory Map of OSD FPGA for CE1 Memory Space

Addr	Register	Type	Bits
00000–7FFFF	Flash Address Space	R/W	8
80000–80007	Dual UART Registers for UART 1 (TI TL16C752B)	R/W	8
80008–8000F	Dual UART Registers for UART 2 (TI TL16C752B)	R/W	8
80010	OSD Control Register	R/W	6
80011	DMA Threshold LSB Register	R/W	8
80012	DMA Threshold MSB Register	R/W	8
80013	Interrupt Status Register	R	7
80014	Interrupt Enable Register	R/W	5
80015	GPIO Direction Register	R/W	8
80016	GPIO Status Register	R/W	8
80017	LED Register	R/W	8
80018	Flash Page Register	R/W	3
80019–8001E	Reserved	R	8
8001F	FPGA version	R	8
80020	Not available	–	–

Base address for $\overline{\text{CE3}}$ memory space is B000 0000. Addresses specified are in Hex.

Table 4. Memory Map of OSD FPGA for $\overline{\text{CE3}}$ Memory Space

Addr	Register	Type	Bits
00	Synchronous Test Register	R/W†	32
04	PLL Data Register	R/W†	16
01C-3C	Reserved	R	32
0C	OSD YSTART Register	R/W	12
08	OSD XSTART Register	R/W	12
10	OSD XSTOP Register	R/W	12
14	OSD YSTOP Register	RW	12

† The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

Table 4. Memory Map of OSD FPGA for $\overline{CE3}$ Memory Space (Continued)

Addr	Register	Type	Bits
18	Events Per Field Register	R/W	16
40	OSD Data FIFO	W	32
44	OSD CLUT	W	32
48-7C	Reserved	R	32
80–1FFFFC	Not available	–	–
200000 –	Available for External Use	R/W	32

† The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

5 Asynchronous Register Definitions

5.1 OSD Control Register

The DM642 can write to this register to control the working of the OSD FPGA and the OSD functionality of the FPGA.

Figure 13. OSD Control Register

7	6	5	4	3	2	1	0
Reserved		DLL RST	Clear CLUT	Bus Width	Int EN	OSD EN	RST
R–000			R/W–0	R/W–0	R/W–0	R/W–0	R/W–0

Note: R/W-0 – R – Read, W – Write,

Bit	Name	Description
7:6	Reserved	
5	DLL RST	Reset for video clock DLL
		0 Brings DLL out of reset
		1 Places video clock DLL in reset. The DLL should be reset any time the video clock frequency is changed.
4	CLEAR CLUT	Clear Color Look Up Table
		0 Does not clear the Color Look Up Table (CLUT). Writing is allowed
		1 Clears the CLUT and writing to the CLUT is prohibited

Figure 13. OSD Control Register (Continued)

Bit	Name	Description	
3	BUS WIDTH	Video data bus width	
		0	Sets the video data bus width to 16 bits
		1	Sets the video data bus width to 8 bits
2	INT EN	Interrupt enable	
		0	Interrupts from the OSD FPGA to the DSP are not enabled.
		1	Enables interrupts from the OSD FPGA to the DSP
1	OSD EN	OSD enable. The OSD FPGA multiplexes video data and OSD data.	
		0	Disables OSD functionality of the FPGA and the Video data is passed through
		1	Enables the OSD functionality of the OSD FPGA
0	RST	Software reset for the OSD FPGA	
		0	Brings the FPGA out of reset
		1	Places the FPGA in reset. All the registers in the FPGA take their default values, and writing to any of the register except the control register, bit 0, is prohibited.

5.2 DMA Threshold LSB Register

The DMA Threshold LSB and MSB Registers set the OSD Data FIFO DMA threshold to determine when to load more OSD data. Every time the number of **32-bit words** in the OSD Data FIFO drops below DMA threshold and the number of events is less than the events specified in Events-Per-Field register, a DMA event is generated.

Figure 14. DMA Threshold LSB Register

7 … 0
DMA THLD LSB
R/W-0

Note: R/W-0 – R – Read, W – Write,

Bit	Name	Description
7–0	DMA THLD LSB	These bits select the LSB portion of the DMA threshold that triggers DMA requests.

5.3 DMA Threshold MSB Register

The DMA Threshold LSB and MSB Registers set the OSD Data FIFO DMA threshold to determine when to load more OSD data. Every time the number of **32-bit words** in the OSD Data FIFO drops below DMA threshold and the number of events is less than the events specified in Events-Per-Field register, a DMA event is generated.

Figure 15. DMA Threshold MSB Register

7 … 0
DMA THLD MSB
R/W-0

Note: R/W-0 – R – Read, W – Write,

Bit	Name	Description
7–0	DMA THLD MSB	These bits set the MSB portion of the DMA threshold that triggers DMA requests.

5.4 Interrupt Status Register

This register reflects the status of all the interrupt sources and status signals inside and outside the OSD FPGA. It is a read-only register.

Figure 16. Interrupt Status Register

7	6	5	4	3	2	1	0
Res	DLL LOCK	PLL TX END	UART INTB	UART INTA	RTS1B	RTS1A	FIFO URUN
R-00	R-0	RWC-0	R-0	R-0	R-0	R-0	R-0

Note: R – Read, RWC – Read and clear on write

Bit	Name	Description
7	Reserved	

Figure 16. Interrupt Status Register (Continued)

Bit	Name	Description
6	DLL LOCK	Video DLL lock status. This bit does not generate an interrupt. 0 Unlocked 1 Video DLL is locked. If this bit does not set in spite of a long wait, reset the video DLL by writing a 1, followed by a 0 to the DLL RST bit in the OSD Control Register.
5	PLL TX END	PLL Transmit end. When set to 1, indicates that the PLL Serial Interface has completed transmitting the data to the Clock PLL. This bit is cleared by writing a 1 to this bit in this register. This bit does not generate a interrupt.
4	UART INTB	UART B interrupt status. This bit reflects the status of the interrupt signal coming from the UART B. If interrupts are enabled and the corresponding enable bit is set in the Interrupt Enable Register, the DSP is interrupted.
3	UART INTA	UART A interrupt status. This bit reflects the status of the interrupt signal coming from the UART A. If interrupts are enabled and the corresponding enable bit is set in the Interrupt Enable Register, the DSP is interrupted.
2	RTS1B	RTS1_B signal status. This bit reflects the status of the RTS1_B signal coming from the decoder. If interrupts are enabled and the corresponding enable bit is set in the Interrupt Enable Register, the DSP is interrupted..
1	RTS1A	RTS1_A signal status. This bit reflects the status of the RTS1_A signal coming from the decoder. If interrupts are enabled and the corresponding enable bit is set in the Interrupt Enable Register, the DSP is interrupted.
0	FIFO URUN	FIFO underrun. When set, indicates that the display FIFO ran out of data. If interrupts are enabled and the corresponding enable bit is set in the Interrupt Enable Register, the DSP is interrupted. This bit is cleared when the OSD data FIFO is no longer empty.

5.5 Interrupt Enable Register

The Interrupt Enable Register controls the interrupts from the OSD FPGA.

Figure 17. Interrupt Enable Register

7–5	4	3	2	1	0
Reserved	UART INTB EN	UART INTA EN	RTS1B EN	RTS1A EN	URUN EN
R-000	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R/W-0 – R – Read, W – Write,

Bit	Name	Description
4	UART INTB EN	When set, enables UART INTB interrupt provided the INTEN bit is set in OSD CTL register.
3	UART INTA EN	When set, enables UART INTA interrupt provided the INTEN bit is set in OSD CTL register.
2	RTS1B EN	When set, enables RTS1B interrupt provided the INTEN bit is set in OSD CTL register.
1	RTS1A EN	When set, enables RTS1A interrupt provided the INTEN bit is set in OSD CTL register.
0	URUN EN	When set, enables Underrun interrupt provided the INTEN bit is set in OSD CTL register.

5.6 GPIO Direction Register

The GPIO Direction Register controls the direction of the GPIOs. The GPIOs all default to inputs and the DM642 EVM has external pullups on these signals to prevent them from floating.

Figure 18. GPIO Direction Register

7	6	5	4	3	2	1	0
GPIO7 DIR	GPIO6 DIR	GPIO5 DIR	GPIO4 DIR	GPIO3 DIR	GPIO2 DIR	GPIO1 DIR	GPIO0 DIR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R/W-0 – R – Read, W – Write,

Bit	Name	Description	
7	GPIO7–GPIO0 DIR	Sets the direction for GPIOn, where n = the GPIO number between 7 and 0	
		0	Sets the GPIOn as input
		1	Sets the GPIOn as output

5.7 GPIO Register

The GPIO Register either controls or reflects the status of the GPIO signals.

Figure 19. GPIO Register

7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R/W-0 − R − Read, W − Write,

Bit	Name	Description
7−0	GPIO7−GPIO0	When GPIOn (where n = a number between 0 and 7) is set up as input this bit is read only and reflects the status of the GPIOn input.
		0 Holds the GPIOn output high when GPIOn is set up as output
		1 Holds the GPIOn output high, when GPIOn is set up as output

5.8 LED Register

The DM642 can write to this register to individually control 8 LEDs on the EVM.

Figure 20. LED Register

7	6	5	4	3	2	1	0
LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R/W-0 − R − Read, W − Write,

Bit	Name	Description
7	LED7−0	LED switch. LEDn, where n = number 7−0, turns the LED on or off.
		0 Turns the LED on
		1 Turns the LED off

5.9 Flash Page Register

The DM642 can write to this register to set the 3 flash page bits.

Figure 21. Flash Page Register

7 ... 3	2	1	0
Reserved	BIT2	BIT1	BIT0
R-00000	R/W-0	R/W-0	R/W-0

Note: R/W-0 – R – Read, W – Write,

Bit	Name	Description
7–3	Reserved	These bits are reserved and always read 0.
2	BIT2	Flash page bit 2
1	BIT1	Flash page bit 1
0	BIT0	Flash page bit 0

5.10 FPGA Version

Figure 22. FPGA Version Register

7 ... 0
FPGA Version
R-0

Note: R = Read, -n = reset value

Bit	Name	Description
7–0	FPGA Version	These bits define the FPGA version number.

6 Synchronous Register Definitions

6.1 Test Register

The DM642 can read and write to this register to verify proper connections for all the 32 bits of the data bus.

Figure 23. Test Register

31	24
TEST	
R/W-0000 0000 0000 0000	

23	16
TEST	
R/W-0000 0000 0000 0000	

15	8
TEST	
R/W-0000 0000 0000 0000	

7	0
TEST	
R/W-0000 0000 0000 0000	

† The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

Note: R/W-0 – R – Read, W – Write,

6.2 PLL Data Register

Whenever the DM642 writes to this register, it triggers a write to the Clock PLL (PLL1708). The data written in the 16 LSB bits of this register are written to the Clock PLL. The PLL TX END bit is set in the Interrupt Status Register on completing the transmission.

Figure 24. PLL Data Register

31	24
TEST	
R-0000 0000 0000 0000	

23	16
TEST	
R-0000 0000 0000 0000	

15	8
PLL TX Data	
R/W-0000 0000 0000 0000	

7	0
PLL TX Data	
R/W-0000 0000 0000 0000	

† The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

Note: R/W-0 – R – Read, W – Write,

See the *PLL1707, PLL1708 3.3-V Dual PLL Multi-Clock Generator Data Sheet* (literature number SLES065) for device register definitions.

6.3 OSD XSTART Register

This register defines the X start position for the OSD window.

Figure 25. OSD XSTART Register

31	16
Reserved	
R-0000 0000 0000 0000	

15–12	11–0
	OSD XSTART
R-0000	R/W – 0000 0000 0000

The OSD XSTART is defined in terms of active pixels. The design maintains an FP_COUNT, which counts active pixels within each line, and an FL_COUNT, which counts active lines within each field. Both the FP_COUNT and FL_COUNT counters start counting from 0. When the FP_CONT = OSD

XSTART and FL_COUNT = OSD YSTART the OSD multiplexer multiplexes OSD data in output video stream provided the alpha for the corresponding pixel is 1.

Note: The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

6.4 OSD YSTART Register

This register defines the Y start position for the OSD window.

Figure 26. OSD YSTART Register

31	16
Reserved	
R-0000 0000 0000 0000	

15	12	11	0
		OSD YSTART	
R-0000		R/W – 0000 0000 0000	

The OSD YSTART is defined in terms of active lines. The design maintains an FL_COUNT, which counts active lines within each field. The FL_COUNT counter starts counting from 0. When the FP_COUNT = OSD XSTART and FL_COUNT = OSD YSTART, the OSD multiplexer multiplexes OSD data in output video stream provided the alpha for the corresponding pixel is 1.

If the video stream is interlaced stream, the OSD_YSTART must be set to a minimum value of 1 for OSD to work properly.

Note: The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

6.5 OSD XSTOP Register

This register defines the X stop position for the OSD window.

Figure 27. OSD XSTOP Register

31	16
Reserved	
R-0000 0000 0000 0000	

15	12	11	0
		OSD XSTOP	
R-0000		R/W – 0000 0000 0000	

The OSD XSTOP is defined in terms of active pixels. When the FP_COUNT = OSD XSTOP and FL_COUNT = OSD YSTOP, the OSD multiplexer stops multiplexing OSD data in output video stream.

Note: The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

6.6 OSD YSTOP Register

This register defines the Y stop position for the OSD window.

Figure 28. OSD YSTOP Register

31–16
Reserved
R-0000 0000 0000 0000

15–12	11–0
	OSD YSTOP
R-0000	R/W – 0000 0000 0000

The OSD YSTOP is defined in terms of active lines. When the FP_COUNT = OSD XSTOP and FL_COUNT = OSD YSTOP the OSD multiplexer stops multiplexing OSD data in output video stream.

Note: The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

6.7 Events-Per-Field Register

This register defines the number of DMA events needed per field.

Figure 29. Events-Per-Field Register

31–16
Reserved
R-0000 0000 0000 0000

15–0
EVTS PER FIELD
R/W-0000 0000 0000 0000

The design maintains EVT_COUNT counter that counts the number of events generated within each field. The EVTS PER FIELD is the maximum number of events that can be requested by the design in each field. When EVT_COUNT = EVTS PER FIELD, the DMA event generator stops generating events till the start of the next field.

Note: The synchronous registers can be read reliably at a maximum ECLKOUT2 of 70 MHz.

7 Different Video/OSD Modes of Operation

The OSD FPGA acts as a bridge between the Video Port of DM642 and SAA7105 Encoder. The Video Port can be programmed as 8/10/16/20-bit port.

However, the SAA7105 accepts only 8-bit inputs. For modes such as Y/C where 16-bit operation is needed, the SAA7105 expects data on both rising and falling edge of the clock.

When the Video Port of DM642 is operated in 8-bit mode, the OSD FPGA passes data through when OSD functionality is disabled. When OSD is enabled, the 8-bit video data is multiplexed with OSD data and sent to SAA7105 Encoder.

When the Video Port is operated in 16-bit mode, the OSD FPGA multiplexes the MSB and LSB bits of the video data on rising and falling edge of clock and sends this data to SAA7105 Encoder. If OSD is enabled, OSD data is multiplexed with video data and then the 16-bit data is multiplexed on rising edge and falling edge of clock to get 8-bit data to SAA7105 encoder.

The following sections describe programming steps for different modes of operation of OSD FPGA.

8-Bit Video, OSD Disabled

Step 1: Set the OSD EN bit to 0 and BUS WIDTH bit to 1 by writing to the OSD Control Register. This disables the OSD functionality and passes the video data received from the Video Port straight to the SAA7105 encoder.

16-Bit Video, OSD Disabled

Step 1: Set the OSD EN bit to 0 and BUS WIDTH bit to 0 by writing to the OSD Control Register. This disables the OSD functionality and converts the 16-bit video data to 8-bit dual clock edge data before passing it to the SAA7105 encoder.

8-Bit Video, OSD Enabled

Step 1: Set the BUS WIDTH bit to 1 by writing to the OSD Control Register.

Step 2: Clear the CLUT in the OSD FPGA by writing a 1 to the Clear CLUT bit in the OSD Control Register.

Step 3: Set the Clear CLUT bit to 0 by again by writing to the OSD Control Register.

Step 4: Program the 7-bit CLUT memory by writing to the OSD CLUT address in the CE3 space. The CLUT is 128 deep. You must be sure that the correct number of writes are made to the OSD CLUT.

Step 5: Program the FIFO threshold level to trigger the DMAs by writing to the DMA Threshold LSB and MSB Registers.

Step 6: Enable FIFO Underrun interrupt if desired by writing a 1 to URUN EN bit in the Interrupt Enable Register and a 1 to INT EN bit in the OSD Control Register.

Step 7: Write to the OSD XSTART, OSD YSTART, OSD XSTOP, and OSD YSTOP registers to set the OSD window size.

Step 8: Write to the Events-Per-Field register to set the number of events desired per field.

Step 9: Enable the OSD functionality by writing a 1 to the OSD EN bit in the OSD Control Register.

Step 10: Wait for DMA request from the OSD FPGA. Whenever requested, transfer data of size programmed in the DMA Threshold LSB and MSB Registers.

Step 11: Continue repeating step 8.

16-Bit Video, OSD Enabled

Step 1: Set the BUS WIDTH bit to 0 by writing to the OSD Control Register.

Step 2: Clear the CLUT in the OSD FPGA by writing a 1 to the Clear CLUT bit in the OSD Control Register.

Step 3: Set the Clear CLUT bit to 0 by again writing to the OSD Control Register.

Step 4: Program the 7-bit CLUT memory by writing to the OSD CLUT address in the CE3 space. The CLUT is 128 deep. The user should ensure that the correct number of writes are made to the OSD CLUT.

Step 5: Program the FIFO threshold level to trigger the DMAs by writing to the DMA Threshold LSB and MSB Registers.

Step 6: Enable FIFO Underrun interrupt if desired by writing a 1 to URUN EN bit in the Interrupt Enable Register and a 1 to INT EN bit in the OSD Control Register.

Step 7: Write to the OSD XSTART, OSD YSTART, OSD XSTOP, and OSD YSTOP registers to set the OSD window size.

Step 8: Write to the Events-Per-Field register to set the number of events desired per field.

Step 9: Enable the OSD functionality by writing a 1 to the OSD EN bit in the OSD Control Register.

Step 10: Wait for DMA request from the OSD FPGA. Whenever requested, transfer data of size programmed in the DMA Threshold LSB and MSB Registers. If desired, you can change the OSD window size and events per field dynamically. These changes are latched by the FPGA at the next VSYNC.

Step 11: Continue repeating step 8.

7.1 Timing Constraints

The OSD FPGA constraints are as follows:

EMIF Interface
Input timing requirements for EMIF control, address, and data
Setup time: 1 .9 ns
Hold time: 0 ns

Output timing requirements for EMIF data
Output access time: 8ns
Output hold time: Cannot be specified.

Video Port Interface
Input timing requirements for control (VP2CTL[2:0]) and data with respect to VP2CLK0 (clock from FPGA to DM642).
Setup time: 1.7 ns
Hold time: – 0.4 ns

Encoder Interface
Output timing requirements for control and data with respect to PIXCLKI (clock from FPGA to SAA7105)
Output access time: 3 ns
Output hold time: 3 ns (This output hold time is achieved by phase shifting PIXCLKI from the clock driving the encoder data)

7.2 FPGA Configuration File Generation

Tools Used:

- ❑ Synplify 7.2 for synthesis
- ❑ Xilinx 5.1i for place and route
- ❑ iMPACT from Xilinx for hex file generation

Following steps are followed to get a hex file to configure the OSD FPGA

Step 1: Compile and Synthesize

The Synplify tool from Synplicity is used to compile the VHDL source code and generate an .edf file. The inputs given to the Synplify tools include all the VHDL source code files except for the files generated with Core Generator. The Core Generator files are the FIFO and RAM files. Also a constraints file is given as an input to the Synplify tool indicating timing constraints that the design has to meet.

It is best to use the project file generated for the OSD FPGA as that has all the settings needed to compile and synthesize the design correctly.

Step 2: Translate, Map and Place and Route the Design

The .edf output from the Synplify tool is brought into the Xilinx ISE tool. Xilinx ISE tool should be set up for EDF flow. The .edf file is added as a source file in the Xilinx Project. A constraints file (.ucf) is also added as a source file. All the Core Generator files are placed in the project directory.

The Xilinx Implement Design utility is run to translate, map, and place and route the design. Again, it is best to use the project file generated for the OSD FPGA as that has all the setting needed to fit the design and meet the timing constraints. The project file includes all the properties used for the various stages of fitting and place and route that help with meeting the timings.

Step 3: Generate .bit Programming File

Run the generate programming file utility in the Xilinx ISE to generate a .bit programming file.

Step 4: Convert the .bit file to a .hex file

The iMPACT utility is run to convert the .bit file to .hex file. This utility can be run within the Xilinx ISE GUI or can be run individually outside the GUI.

When iMPACT comes up, choose “Prepare Configuration Files”. Choose “PROM File” for type of file. Choose serial EEPROM as a target, and .hex for PROM file format. Make sure that the “swap bits” dialogue box is not checked. Provide a name for the output .hex file and a location. Select “Auto PROM Select”. Give the name of the input .bin file when asked, and select “yes” when asked about generating a PROM file.

Then, the iMPACT utility generates a .hex file.