iMARV™

Full-HD Video Display Processor with embedded SDRAM

MDIN-3xx Series

Datasheet

Version 0.7

Preliminary

2012-04-01

Macro Image Technology, Inc.

Ordering Information

Requests for copies of this document should be directed to Macro Image Technology, Inc.
East Bldg., 6th Floor, IT Venture Tower,
78 Garakbon-Dong, Songpa-Gu, Seoul, Korea 138-803
Tel : +82-2-2142-4000
Fax : +82-2-2142-4099
Email : sales@mitinc.co.kr
http://www.mitinc.co.kr

Restriction on Use

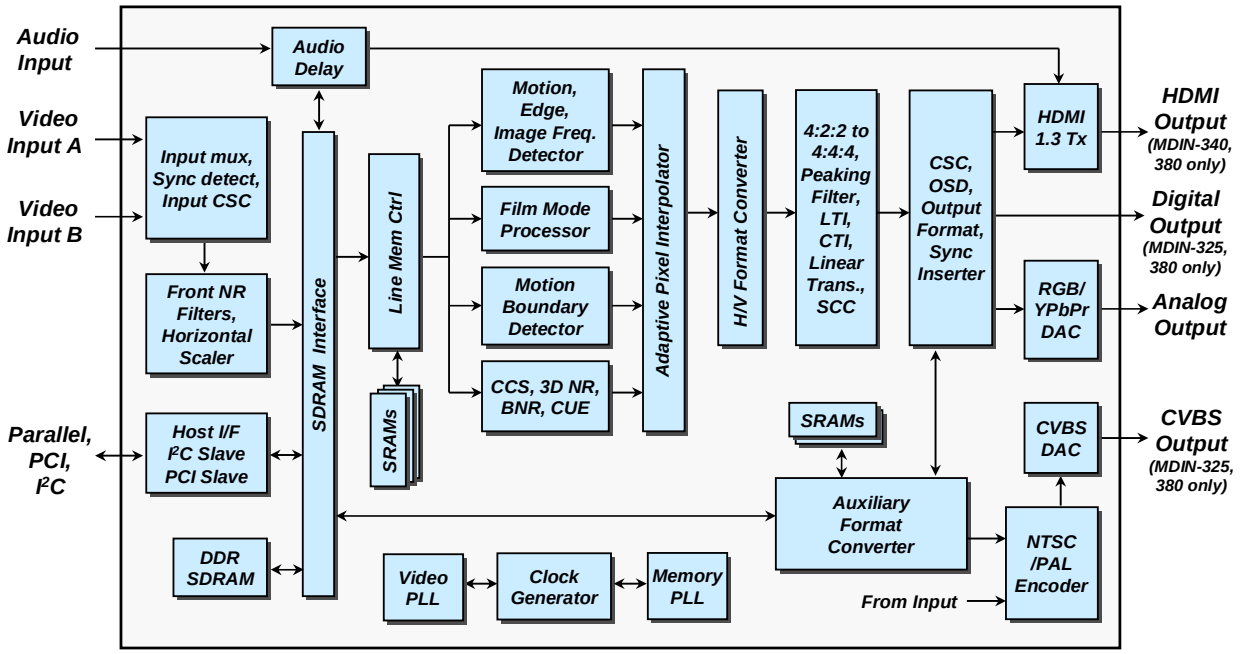
** The contents of this document can be changed without notice.*

MDIN-3xx series are highly integrated single chip implementations of deinterlacing and format conversion, which includes an HDMI transmitter, an SDRAM and tripple 10-bit video DACs. They provide configurable digital video input ports for interlaced or progressive scan type video with 10-bit precision per color component. They receive any format of interlaced scan type video and performs deinterlacing and format conversion to produce any desired format of progressive scan video with excellent signal quality preservation. All the MDIN-3xx chips have an analog component output port for interlaced or progressive scan type video with 10-bit precision per color component. MDIN-340 and MDIN-380 provide an HDMI output port supporting HDMI 1.3 specification. MDIN-325 and MDIN-380 provide a flexible digital output port with various data formats and an analog CVBS output port.

MDIN-3xx chips provide high quality edge preserving deinterlacing with the 5th generation motion adaptation 3-D deinterlacing algorithm and performs proper processing for fast motion and film video sources. It also supports high performance 3-D noise reduction and cross color suppression. In addition they scale up or down the input video with an arbitrary scale ratio and they also provide frame rate conversion capability. The Macro Image Technology's proprietary iMARV™ technology makes MDIN-3xx chips the highest quality upconverter.

MDIN-3xx series' high quality deinterlacing and video processing capabilities are suitable for high quality display format conversion application such as DVD players, set-top boxs, network cameras and DVRs. Their low power comsumption feature is suitable for battery-powered devices such as camcorders and PMPs.

Block Diagram



Common Features

- Configurable two digital video input ports for interlaced or progressive video up to 1920x1080p
- Generates the interlaced or progressive-scanned video output up to 1920x1080p
- Main and auxiliary video paths for PIP or dual video output
- Programmable color space conversion on input and output
- Performs high quality 3-D deinterlacing for interlaced video input up to 1920x1080i
- High quality edge preserving deinterlacing thru multiple directional edge detection and interpolation
- Deinterlacing with fast motion detection and motion boundary preservation
- Deinterlacing with excellent film mode detection and processing for film source
- 3-D noise reduction in both temporal and spatial domain
- Cross color suppression for the video without being processed by 3-D comb filter
- MPEG noise reduction (mosquito noise, color upsampling error correction)
- Independent horizontal and vertical scaling with anti-aliasing interpolation filter
- Frame rate conversion with arbitrary conversion ratio
- Programmable front noise reduction filter for input video
- One dimensional LTI and CTI for edge improvement
- Programmable high order peaking filter for horizontal sharpness control
- Piecewise Linear Transformation for luminance and chrominance
- Adaptive contrast enhancement and selective color correction
- Programmable brightness, contrast, tint and saturation control
- 2-layer OSD with 8 sprites and 2/4/16/256/full colors (bitmap and character mode)
- Multi-channel video processing with 3-D deinterlacing, 3-D NR, etc.
- Cost and size effective embedded DDR SDRAM for frame buffer

MDIN-325 / MDIN-325A(*) Features

- 24-pin digital input port and 24-pin digital output port
- NTSC/PAL video encoder
- Serial I^2C interface
- 16MB (MDIN-325) or 32MB (MDIN-325A) DDR SDRAM
- 144-pin FBGA package

(Note *) MDIN-325A has the same features as MDIN-325 except for DDR SDRAM size.

MDIN-340 Features

- 24-pin digital input port
- HDMI 1.3 and DVI 1.0 compliant HDMI transmitter
- Serial I^2C interface
- 16MB DDR SDRAM
- 144-pin FBGA package (pin-compatible with MDIN-240)

MDIN-380 Features

- Up to 36-pin digital input port and up to 30-pin digital output port (shared pins)
- HDMI 1.3 and DVI 1.0 compliant internal HDMI transmitter
- NTSC/PAL video encoder
- Serial I^2C interface, parallel (8/16bits), PCI interface
- 32MB DDR SDRAM
- 240-pin FBGA package

Specifications

- Input Formats
  - Configurable two input ports : Total 36-pin (MDIN-380) / 24-pin (MDIN-325/340) configurable single 30/24-bit or 24/20/10-bit + 10-bit with 10-bit or 8-bit per color component
  - Video sources : 30-bit(MDIN-380 only) or 24-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2, 20-bit Y/C seperated digital format (ITU-R BT.1120) 10-bit Y/C multiplexed digital format (ITU-R BT.656)
  - RGB-to-YCbCr color space conversion with programmable 3x3 matrix
  - Interlaced input : various standard video format up to 1920x1080@60i
  - Progressive input : various standard video format up to 1920x1080@60p
  - Multiplexed input : up to 4-channel ITU-R BT.656 or BT.1120 time-multiplexed format
  - Maximum input size : 8190 pixels, 8191 lines (limited by clock freq & internal memories)
  - Maximum pixel rate : 165MHz

- Output Formats
  - Programmable standard or non-standard video format up to 1920x1080@60p
  - Progressive or interlaced scanning
  - Digital output (MDIN-325) : 24-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2, 10-bit Y/C multiplexed 4:2:2 with separate sync or embedded timing references (SAV/EAV)
  - Digital output (MDIN-380) : 30-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2, 10-bit Y/C multiplexed 4:2:2 with separate sync or embedded timing references (SAV/EAV)
  - HDMI Output (MDIN-340) : 30-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2
  - HDMI Output (MDIN-380) : 36-bit(bypass only) or 30-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2
  - Analog output : 10-bit RGB/YPbPr with separate sync or sync on G/Y (bi-level/tri-level)
  - YCbCr-to-RGB color space conversion with programmable 3x3 matrix
  - Multiplexed output (MDIN-325/380) : up to 4-channels ITU-R BT.656 or BT.1120 time-multiplexed
  - Maximum output size : 4094 pixels, 4095 lines (limited by clock freq & internal memories)
  - Maximum pixel rate : 165MHz
  - NTSC/PAL output (MDIN-325/380) : CVBS or Y/C(S-Video)

- Format Conversion
  - Independent horizontal and vertical scaling
  - Format conversion from one format to another format at an arbitrary scaling ratio
  - Horizontal 8-tap and vertical 4-tap anti-aliasing interpolation filters for graceful format conversion

- Frame Rate Conversion
  - Frame rate conversion from 3~250Hz to 3~250Hz
  - Arbitrary conversion ratio : x1/31 to x31

- Deinterlacing
  - Motion adaptive 3-D deinterlace using 5 fields on a per-pixel basis
  - Programmable motion detection and adaptation control
  - Adaptive motion-weighted interpolation for eliminating non-motion artifacts
  - Multi-directional edge preserving interpolation
  - Fast motion detection and handling using recent 30-field motion information
  - Motion boundary preservation using recent 30-field motion information
  - Still Mode for crisp image viewing
  - 3:2 or 2:2 pull-down film mode detection and handling
  - Bad edit detection and handling
  - Still and moving subtitle detection and handling

- Noise Reduction and Signal Enhancement
  - Programmable 15-tap and 7-tap front NR filter for luminance and chrominance input
  - 3-D noise reduction in both temporal and spatial domain
  - One dimensional MNR (mosquito noise reduction)
  - Vertical CUE (color upsampling error) correction
  - One dimensional LTI and CTI for edge enhancement with gain and coreing control
  - Programmable high order peaking filter for horizontal sharpness control
  - Piecewise linear transformation both for luminance and chrominance
  - Adaptive contrast enhancement with up to 12-point piecewise linear transformation
  - Selective color control
  - Dithering down to 5-bits by error diffusion

- Display Functions
  - Programmable sizing & positioning
  - PIP or POP display
  - 4-channel video processing of SD video with deinterlacing, NR and signal enhancement
  - Multi-window display (2x2, 4x1 or 1x4) on multi-channel mode
  - Dual format video output with main and auxiliary video paths
  - Programmable brightness, contrast, tint and saturation control

- OSD Functions
  - 2 layers with 4 sprites per layer with bitmap and character mode OSD
  - 2/4/16/256-color indexed mode or 16/24/32-bit full color mode with 32-level alpha blending
  - 2/4/16/256-color indexed color or 16/24/32-bit full color OSD with 32-level alpha blending
  - Cursor layer with 16-color indexed mode
  - Up to 2047x2047 of bitmap size (depending on available memory size)
  - Up to 32x63 of font size with 2/16/256-color
  - 32-row x 16-col or 16-row x 32-col character display map supported
  - Accelerated bitmap filling and copying
  - Support for decoding of run-length encoded bitmap and font data
  - One 2/4/16-color bitmap layer with 4 sprites and cursor layer on auxiliary video output
  - Box OSD for boundary display of highlight area or windows

- Auxiliary Format Conversion
  - Independent format conversion from main video path
  - Format conversion at an arbitrary scaling ratio
  - Horizontal 9-tap and vertical 5-tap anti-aliasing filters
  - Input and output CSC
  - Video input : up to 1080p of 8-bit from input ports or main format converter
  - Video output : up to 1080p of 8-bit to analog output, NTSC/PAL output, digital output port

- Frame Buffer Memory
  - Space and cost effective internal SDRAM
  - 16MB (MDIN-325/340) or 32MB (MDIN-325A/380) low power DDR SDRAM

- Host Interface
  - I^2C slave interface capable of running up to 400kHz
  - Pin-configurable parallel interface with 8 or 16-bit data bus (MDIN-380)
  - DMA slave capability through parallel interface (MDIN-380)
  - 32-bit, 33MHz PCI slave interface (MDIN-380)
  - Flexible Interrupt output pin

- Miscellaneous
  - Auto detection of input video/sync type and format
  - Auto detection of input video/sync changing and lost
  - Sync detection for composite and non-standard input sync
  - Input-frame-locking mode and free-run mode
  - Frame-locking and clock-locking to an external sync source
  - Programmable output sync generation
  - Built-in input/output test pattern generator
  - Video overlay on background video
  - Programmable output sync generation

- HDMI transmitter (MDIN-340/380)
  - HDMI 1.3, HDCP 1.2 and DVI 1.0 compliant transmitter
  - Master I^2C interface for DDC connection
  - Integrated HDCP encryption engine and pre-programmed HDCP keys
  - Monitor detection supported through hot plug and receiver detection
  - Deep color and xvYCC supported
  - IEC 60958/61937 compatible audio input interface
  - Four I^2S audio inputs : 2-channel 192kHz, 8-channel 96kHz
  - One S/PDIF input : PCM, Dolby Digital, DTS (32-192kHz)
  - Flexible, programmable I^2S channel mapping
  - Adjustable audio delay (up to 680ms at 48kHz) for A/V synchronization
  - Built in Consumer Electronics Control (CEC) support

- Electrical and Mechanical Characteristics
  - 1.2V, 1.8V and 3.3V supply voltage, 3.3V I/O (5V tolerant I/O for I^2C and DDC)
  - Power consumption : (Refer to 4.3.2)
  - MDIN-325/325A : 144-pin FBGA package (12mm x 12mm)
    MDIN-340 : 144-pin FBGA package (12mm x 12mm)
    MDIN-380 : 240-pin FBGA package (12mm x 16mm)

(This page is intentionally blanked)

Contents

1. Pin Configuration

1.1 Pinout Diagram

	1	2	3	4	5	6	7	8	9	10	11	12
A	DATA_ OUT(23)	DATA_ OUT(20)	DATA_ OUT(17)	DATA_ OUT(14)	DATA_ OUT(11)	VCLK_ OUT_B	DATA_ OUT(6)	VCLK_ OUT	DATA_ OUT(2)	INT_OUT	VENC_ AGND	VENC_ VREF
B	DATA_ OUT(22)	DATA_ OUT(19)	DATA_ OUT(16)	DATA_ OUT(13)	DATA_ OUT(10)	DATA_ OUT(8)	DATA_ OUT(5)	DE_OUT	DATA_ OUT(1)	I2C_ MODE	VENC_ AVDD33	VENC_ OUT
C	DATA_ OUT(21)	DATA_ OUT(18)	DATA_ OUT(15)	DATA_ OUT(12)	DATA_ OUT(9)	DATA_ OUT(7)	DATA_ OUT(4)	DATA_ OUT(3)	DATA_ OUT(0)	GND	GND	VENC_ FSADJ
D	CLK_B	DATA_IN (0)	DATA_IN (1)	VDD18	VDD18	GND	VDD33	GND	GND	VDD18	GND	GND
E	DATA_IN (2)	DATA_IN (3)	DATA_IN (4)	VDD18	VDD12	GND	VDD33	GND	VDD12	VDD18	DAC_ AVDD33	DAC_ ROUT
F	DATA_IN (5)	DATA_IN (6)	DATA_IN (7)	VDD18	VDD12	GND	VDD33	GND	VDD12	VDD18	DAC_ VREF	DAC_ AVDD33
G	DATA_IN (8)	DATA_IN (9)	DATA_IN (10)	VDD18	VDD12	GND	VDD33	GND	VDD12	VDD18	DAC_ AGND	DAC_ FSADJ
H	DATA_IN (11)	DATA_IN (12)	DATA_IN (13)	VDD18	VDD12	GND	VDD33	GND	VDD12	VDD18	DAC_ AGND	DAC_ GOUT
J	DATA_IN (14)	DATA_IN (15)	DATA_IN (16)	VDD18	VDD18	GND	VDD33	GND	VDD18	VDD18	DAC_ AGND	DAC_ AVDD33
K	DATA_IN (17)	DATA_IN (18)	DATA_IN (19)	GND	GND	GND	GND	RESET_N	TEST_ MODE(3)	TEST_ MODE(2)	TEST_ MODE(1)	DAC_ BOUT
L	DATA_IN (20)	DATA_IN (21)	DATA_IN (22)	GND	MCLK_ PLL_ AGND12	VCLK_ PLL_ AVDD33	VCLK_ PLL_ AVDD12	SDA	FIELDID _A	TEST_ MODE(0)	HSYNC_ OUT	VSYNC_ OUT
M	DATA_IN (23)	CLK_A	GND	MCLK_ PLL_ AVDD12	VCLK_ PLL_ AGND33	VCLK_ PLL_LF	VCLK_ PLL_ AGND12	SCL	HACTIVE_ A	VACTIVE_ A	XTAL_ IN	XTAL_ OUT

Figure 1-1 (a). MDIN-325 Pinout Diagram – Top view

	1	2	3	4	5	6	7	8	9	10	11	12
A	VCLK_ OUT	GND	HDMI_ GND	HDMI_ EXN(2)	HDMI_ EXN(1)	HDMI_ EXN(0)	HDMI_ EXCN	HDMI_ VDN12	HDMI_ GND	GND	SCK_IN	WS_IN
B	HDMI_ TEST_ MODE	GND	HDMI_ GND	HDMI_ EXP(2)	HDMI_ EXP(1)	HDMI_ EXP(0)	HDMI_ EXCP	HDMI_ VDN12	HDMI_ GND	GND	SD_IN(0)	SD_IN(1)
C	INT_OUT	HDMI_ HPD	HDMI_ TEST_ XRST	HDMI_ VDP33	HDMI_ GND	HDMI_ VDN12	HDMI_ GND	HDMI_ VDP33	HDMI_ VDN12	GND	SD_IN(2)	SD_IN(3)
D	CLK_B	DATA_IN (0)	DATA_IN (1)	VDD18	GND	GND	GND	GND	VDD18	VDD18	MCLK_IN	SPDIF_IN
E	DATA_IN (2)	DATA_IN (3)	DATA_IN (4)	VDD18	VDD12	GND	VDD33	GND	VDD12	SD_DQM	HDMI_ CEC	DAC_ AGND
F	DATA_IN (5)	DATA_IN (6)	DATA_IN (7)	VDD18	VDD12	GND	GND	GND	VDD12	SD_CKE	I2C_ MODE	DAC_ ROUT
G	DATA_IN (8)	DATA_IN (9)	DATA_IN (10)	VDD18	VDD18	GND	VDD33	GND	VDD12	VDD18	DAC_ VREF	DAC_ AVDD33
H	DATA_IN (11)	DATA_IN (12)	DATA_IN (13)	VDD18	VDD18	GND	GND	GND	GND	VDD18	NC	DAC_ FSADJ
J	DATA_IN (14)	DATA_IN (15)	DATA_IN (16)	VDD18	VDD12	GND	VDD33	GND	TEST_ MODE(3)	VDD18	DAC_ AGND	DAC_ GOUT
K	DATA_IN (17)	DATA_IN (18)	DATA_IN (19)	VDD12	VDD12	RESET_N	FIELDID _A	HACTIVE _A	VACTIVE _A	TEST_ MODE(2)	DAC_ AVDD33	NC
L	DATA_IN (20)	DATA_IN (21)	DATA_IN (22)	VCLK_ PLL_ AGND12	VCLK_ PLL_ AVDD33	MCLK_ PLL_ AVDD12	SDA	HDMI_ DSCL	XTAL_ IN	TEST_ MODE(1)	DAC_ AGND	DAC_ BOUT
M	DATA_IN (23)	CLK_A	VCLK_ PLL_ AVDD12	VCLK_ PLL_ AGND33	VCLK_ PLL_LF	MCLK_ PLL_ AGND12	SCL	HDMI_ DSDA	XTAL_ OUT	TEST_ MODE(0)	HSYNC_ OUT	VSYNC_ OUT

Figure 1-1 (b). MDIN-340 Pinout Diagram – Top view

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A		HDMI_VDN12	HDMI_GND	GND	GND	VENC_AGND	VENC_OUT	GND	DAC_ROUT	DAC_AGND	DAC_FSADJ	DAC_GOUT	DAC_AGND	
B	HDMI_EXCN	HDMI_EXCP	HDMI_VDN12	GND	VENC_VREF	VENC_FSADJ	VENC_AVDD33	GND	DAC_AVDD33	DAC_AVDD33	DAC_VREF	DAC_AVDD33	DAC_AGND	DAC_BOUT
C	HDMI_EXN(0)	HDMI_EXP(0)	HDMI_VDP33	GND	HOST_SYNC_BUS	TEST_MODE(0)	TEST_MODE(1)	TEST_MODE(2)	TEST_MODE(3)	HDMI_TEST_XRST	HDMI_TEST_MODE	HDMI_HPD	HSYNC_OUT	VSYNC_OUT
D	HDMI_EXN(1)	HDMI_EXP(1)	HDMI_GND	GND	HOST_PCI_BUS	HOST_BUS_MODE(3)	I2C_MODE	GND	GND	GND	HDMI_DSDA	HDMI_DSCL	XTAL_IN	XTAL_OUT
E	HDMI_EXN(2)	HDMI_EXP(2)	HDMI_VDN12	GND	GND	GND	VDD12	VDD12	VDD12	GND	GND	VACTIVE_A	SDA	RESET_N
F	GND	HDMI_GND	HDMI_VDP33	GND	SD_CKE	VDD18	VDD18	VDD18	VDD18	VDD18	VDD18	FIELDID_A	HACTIVE_A	SCL
G	HDMI_CEC	GND	GND	GND	VDD18	VDD18	VDD18	VDD18	VDD18	VDD18	VDD18	GND	GND	GND
H	SPDIF_IN	MCLK_IN	VDD12									GND	VCLK_PLL_AGND12	VCLK_PLL_AVDD12
J	SD_IN(0)	SD_IN(1)	VDD12		VDD33	GND	VDD33	GND	GND	VDD33		MCLK_PLL_AGND12	VCLK_PLL_AGND33	VCLK_PLL_AVDD33
K	WS_IN	SD_IN(3)	SD_IN(2)		VDD33	GND	GND	GND	GND	VDD33		VDD12	MCLK_PLL_AVDD12	VCLK_PLL_LF
L	HOST_CLK	HOST_BUS_MODE(0)	SCK_IN		VDD33	GND	GND	GND	GND	VDD33		VDD12	GND	GND
M	HOST_AD(0)	HOST_BUS_MODE(2)	HOST_BUS_MODE(1)									D_IN(33)	D_IN(34)	D_IN(35)
N	HOST_AD(2)	HOST_AD(1)	VDD18	VDD18	VDD18	VDD18	VDD18	VDD18	VDD18	VDD18	D_IN(30)	D_IN(31)	D_IN(32)	CLK_A
P	HOST_AD(5)	HOST_AD(4)	HOST_AD(3)	VDD18	GND	GND	GND	GND	VDD18	VDD18	D_IN(26)	D_IN(27)	D_IN(28)	D_IN(29)
R	HOST_AD(8)	HOST_AD(7)	HOST_AD(6)	GND	GND	SD_DQM	VDD12	VDD12	VDD12	VDD18	D_IN(22)	D_IN(23)	D_IN(24)	D_IN(25)
T	HOST_AD(11)	HOST_AD(10)	HOST_AD(9)	HOST_ALE	HOST_BLAST_N	INT_OUT	D_OUT(1)	D_OUT(4)	D_IN(0)	D_IN(3)	D_IN(18)	D_IN(19)	D_IN(20)	D_IN(21)
U	HOST_AD(14)	HOST_AD(13)	HOST_AD(12)	HOST_CS_N	HOST_RDY_N	DREQ_N	D_OUT(2)	D_OUT(5)	D_IN(1)	D_IN(4)	D_IN(14)	D_IN(15)	D_IN(16)	D_IN(17)
V	HOST_AD(17)	HOST_AD(16)	HOST_AD(15)	HOST_DS_N	DACK_N	DE_OUT	D_OUT(3)	D_OUT(6)	D_IN(2)	D_IN(5)	D_IN(10)	D_IN(11)	D_IN(12)	D_IN(13)
W		HOST_AD(19)	HOST_AD(18)	HOST_WR_N	VCLK_OUT_B	D_OUT(0)	VCLK_OUT	D_OUT(7)	CLK_B	D_IN(6)	D_IN(7)	D_IN(8)	D_IN(9)	

Figure 1-1 (c). MDIN-380 Pinout Diagram – Top view

1.2 I/O Signal Descriptions

1.2.1 MDIN-325 I/O Signal Descriptions

Group	Signal Name	I/O	Description
Global	RESET_N	I	Global reset, Active low (5V tolerant & Schmitt trigger) All the internal registers are set to default values.
Video Input	CLK_A	I	Input port A video clock
	HACTIVE_A	I	Input port A horizontal active or sync signal Both active low and active high horizontal sync signal can be acceptable. This signal can be composite sync.
	VACTIVE_A	I	Input port A vertical active or sync signal Both active low and active high vertical sync signal can be acceptable.
	FIELDID_A	I	Input port A field ID signal Both active low and active high field identification signal can be acceptable.
	DATA_IN(23:0)	I	Video data Input for input port A and B These bits are configurable, 24-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2 or dual 10-bit YC-multiplexed 4:2:2 data can be input.
	CLK_B	I	Input port B video clock
Video Output	VCLK_OUT	O	Output video pixel clock This pin outputs the same clock as the internal video output clock with certain amount of delay. The amount of delay can be programmable by internal registers.
	VCLK_OUT_B	O	Additional output video pixel clock The polarity and the amount of delay can be programmable by internal registers. This clock can be used to output an inverted video clock.
	HSYNC_OUT	O	Horizontal sync or active for output video data Active high or active low can be selected by internal register.
	VSYNC_OUT	O	Vertical sync or active for output video data Active high or active low can be selected by internal register.
	DE_OUT	O	Data enable for output video data This signal can be horizontal active or composite sync. Active high or active low can be selected by internal register.
	DATA_OUT(23:0)	O	Digital video data output 24-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2 or 10-bit YC-multiplexed 4:2:2 data can be output.
Host Interface	SCL	I	I^2C bus clock (5V tolerant & Schmitt trigger)
	SDA	I/O	I^2C bus data (5V tolerant & Schmitt trigger)
	INT_OUT	O	Interrupt output (active low or high)
	I2C_MODE	I	I^2C bus 16-bit mode selection input When open or high, the 16-bit mode (MDIN-2xx compatible) is selected. When low, the 8-bit mode is selected.
Test Mode	TEST_MODE(3)	I	Test mode control input For normal operation this pin should be low.
	TEST_MODE(2:1)	I	Used to select the host interface clock. Normally, connect both pins to GND and **XTAL_IN** will be used as the host interface clock.
	TEST_MODE(0)	I	Used as the LSB of I^2C slave device address.

Table 1-1. MDIN-325 I/O Signal Descriptions

Group	Signal Name	I/O	Description
GPIO (*1)	GPIO(36:0)	I/O	General purpose input and output
	GPO(2:0)	O	General purpose output on sync output pins
SPI Interface (*2)	SPI_CLK	O	Clock output to an SPI slave device
	SPI_CE_N	O	Chip enable output to an SPI slave device
	SPI_DO	O	Data output to an SPI slave device
	SPI_DI	I	Data input from an SPI slave device
PLL	XTAL_IN	I	External crystal input Either crystal or oscillator can be used for this input pin.
	XTAL_OUT	O	External crystal output
	VCLK_PLL_LF	AI	VCLK PLL loop filter This pin must be connected to external low pass filter.
	VCLK_PLL_AVDD33	VDD	VCLK PLL analog power (3.3V)
	VCLK_PLL_AGND33	GND	VCLK PLL analog ground (3.3V)
	VCLK_PLL_AVDD12	VDD	VCLK PLL analog power (1.2V)
	VCLK_PLL_AGND12	GND	VCLK PLL analog ground (1.2V)
	MCLK_PLL_AVDD12	VDD	MCLK PLL analog power (1.2V)
	MCLK_PLL_AGND12	GND	MCLK PLL analog ground
DAC	DAC_GOUT	AO	Analog green or Y output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_BOUT	AO	Analog blue or Pb output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_ROUT	AO	Analog red or Pr output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_FSADJ	AI	Full-scale output current adjust The full-scale current on each output channel is determined by the value of a resistor connected between this pin and analog ground.
	DAC_VREF	AI	Reference voltage input Well regulated 1.2V reference voltage must be externally supplied. A 0.1uF ceramic capacitor must be connected between this pin and analog ground.
	DAC_AVDD33	VDD	Analog 3.3V power supply for each internal DAC channel
	DAC_AGND	GND	Analog ground for each internal DAC channel
Video Encoder DAC	VENC_OUT	AO	Analog output from internal video encoder DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	VENC_FSADJ	AI	Full-scale output current adjust The full-scale current on the output is determined by the value of a resistor connected between this pin and analog ground.
	VENC_VREF	AI	Reference voltage input Well regulated 1.2V reference voltage must be externally supplied. A 0.1uF ceramic capacitor must be connected between this pin and analog ground.
	VENC_AVDD33	VDD	Analog 3.3V power supply for VENC DAC
	VENC_AGND	GND	Analog ground for VENC DAC
Power & Ground	VDD12	VDD	Digital 1.2V power supply for the core logic
	VDD18	VDD	Digital 1.8V power supply for the SDRAM interface
	VDD33	VDD	Digital 3.3V power supply for the I/O
	GND	GND	Digital ground

Note 1) GPIO signals are shared with other pins.
Note 2) SPI interface signals are shared with other pins.

Table 1-1. MDIN-325 I/O Signal Descriptions (Continued)

1.2.2 MDIN-340 I/O Signal Descriptions

Group	Signal Name	I/O	Description
Global	RESET_N	I	Global reset, Active low (5V tolerant & Schmitt trigger) All the internal registers are set to default values.
Video Input	CLK_A	I	Input port A video clock
	HACTIVE_A	I	Input port A horizontal active or sync signal Both active low and active high horizontal sync signal can be acceptable. This signal can be composite sync.
	VACTIVE_A	I	Input port A vertical active or sync signal Both active low and active high vertical sync signal can be acceptable.
	FIELDID_A	I	Input port A field ID signal Both active low and active high field identification signal can be acceptable.
	DATA_IN(23:0)	I	Input port for video data These bits are configurable, 24-bit RGB/YCbCr 4:4:4, 20-bit YCbCr 4:2:2 or 10-bit YC-multiplexed 4:2:2 data can be input.
	CLK_B	I	Input port B video clock
Audio Input	MCLK_IN	I	Audio input master clock for HDMI transmitter
	SCK_IN	I	I^2S serial clock for HDMI transmitter
	WS_IN	I	I^2S serial word select for HDMI transmitter
	SD_IN(3:0)	I	I^2S serial audio data for HDMI transmitter
	SPDIF_IN	I	S/PDIF audio input for HDMI transmitter
Sync Output	HSYNC_OUT	O	Horizontal sync or active for output video data Active high or active low can be selected by internal register.
	VSYNC_OUT	O	Vertical sync or active for output video data Active high or active low can be selected by internal register.
Host Interface	SCL	I	I^2C bus clock (5V tolerant & Schmitt trigger)
	SDA	I/O	I^2C bus data (5V tolerant & Schmitt trigger)
	INT_OUT	O	Interrupt output (active low or high)
	I2C_MODE	I	I^2C bus 16-bit mode selection input When open or high, the 16-bit mode (MDIN-2xx compatible) is selected. When low, the 8-bit mode is selected.
Test Mode	TEST_MODE(3)	I	Test mode control input For normal operation this pin should be low.
	TEST_MODE(2:1)	I	Used to select the host interface clock. Normally, connect both pins to GND and **XTAL_IN** will be used as the host interface clock.
	TEST_MODE(0)	I	Used as the LSB of I^2C slave device address.
SDRAM	SD_DQM	I	SDRAM DQM input (should be tied to GND)
	SD_CKE	I	SDRAM CKE input (should be tied to VDD18)
GPIO (*1)	GPIO(36:20)	I/O	General purpose input and output
	GPO(2:1)	O	General purpose output on sync output pins
SPI Interface (*2)	SPI_CLK	O	Clock output to an SPI slave device
	SPI_CE_N	O	Chip enable output to an SPI slave device
	SPI_DO	O	Data output to an SPI slave device
	SPI_DI	I	Data input from an SPI slave device

Note 1) GPIO signals are shared with other pins.
Note 2) SPI interface signals are shared with other pins.

Table 1-2. MDIN-340 I/O Signal Descriptions

Group	Signal Name	I/O	Description
PLL	XTAL_IN	I	External crystal input Either crystal or oscillator can be used for this input pin.
	XTAL_OUT	O	External crystal output
	VCLK_PLL_LF	AI	VCLK PLL loop filter This pin must be connected to external low pass filter.
	VCLK_PLL_AVDD33	VDD	VCLK PLL analog power (3.3V)
	VCLK_PLL_AGND33	GND	VCLK PLL analog ground (3.3V)
	VCLK_PLL_AVDD12	VDD	VCLK PLL analog power (1.2V)
	VCLK_PLL_AGND12	GND	VCLK PLL analog ground (1.2V)
	MCLK_PLL_AVDD12	VDD	MCLK PLL analog power (1.2V)
	MCLK_PLL_AGND12	GND	MCLK PLL analog ground
DAC	DAC_GOUT	AO	Analog green or Y output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_BOUT	AO	Analog blue or Pb output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_ROUT	AO	Analog red or Pr output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_FSADJ	AI	Full-scale output current adjust The full-scale current on each output channel is determined by the value of a resistor connected between this pin and analog ground.
	DAC_VREF	AI	Reference voltage input Well regulated 1.2V reference voltage must be externally supplied. A 0.1uF ceramic capacitor must be connected between this pin and analog ground.
	DAC_AVDD33	VDD	Analog 3.3V power supply for each internal DAC channel
	DAC_AGND	GND	Analog ground for each internal DAC channel
HDMI Interface	HDMI_EXP(2:0), HDMI_EXN(2:0)	O	HDMI output data pair
	HDMI_EXCP, HDMI_EXCN	O	HDMI output clock pair
	HDMI_HPD	I	Hot plug detect input (5V tolerant)
	HDMI_DSCL	O	DDC bus clock (5V tolerant)
	HDMI_DSDA	I/O	DDC bus data (5V tolerant & Schmitt trigger)
	HDMI_CEC	I/O	CEC bus (Schmitt trigger & open drain)
	HDMI_TEST_MODE	I	Test mode control input for HDMI PHY For normal operation this pin should be low.
	HDMI_TEST_XRST	I	Test mode reset input for HDMI PHY For normal operation this pin should be high.
	HDMI_VDN12	VDD	HDMI analog power supply (1.2V)
	HDMI_VDP33	VDD	HDMI analog power supply (3.3V)
	HDMI_GND	GND	HDMI analog ground
Test Output	VCLK_OUT	O	Test output of internal video pixel clock
Power & Ground	VDD12	VDD	Digital 1.2V power supply for the core logic
	VDD18	VDD	Digital 1.8V power supply for the SDRAM interface
	VDD33	VDD	Digital 3.3V power supply for the I/O
	GND	GND	Digital ground

Table 1-2. MDIN-340 I/O Signal Descriptions (Continued)

1.2.3 MDIN-380 I/O Signal Descriptions

Group	Signal Name	I/O	Description
Global	RESET_N	I	Global reset, Active low (5V tolerant & Schmitt trigger) All the internal registers are set to default values.
Video Input	CLK_A	I	Input port A video clock
	HACTIVE_A	I	Input port A horizontal active or sync signal Both active low and active high horizontal sync signal can be acceptable. This signal can be composite sync.
	VACTIVE_A	I	Input port A vertical active or sync signal Both active low and active high vertical sync signal can be acceptable.
	FIELDID_A	I	Input port A field ID signal Both active low and active high field identification signal can be acceptable.
	D_IN(35:0)	I	Input port for video data These bits are configurable, 36/30/24-bit RGB/YCbCr 4:4:4, 24/20/16-bit YCbCr 4:2:2 or 12/10/8-bit YC-multiplexed 4:2:2 data can be input. D_IN(25:0) are multi-function pins. D_IN(2), D_IN(1), D_IN(0) can also be used as HACT_B, VACT_B, FID_B respectively.
	CLK_B	I	Input port B video clock
Audio Input	MCLK_IN	I	Audio input master clock for HDMI transmitter
	SCK_IN	I	I^2S serial clock for HDMI transmitter
	WS_IN	I	I^2S serial word select for HDMI transmitter
	SD_IN(3:0)	I	I^2S serial audio data for HDMI transmitter
	SPDIF_IN	I	S/PDIF audio input for HDMI transmitter
Video Output	VCLK_OUT	O	Output video pixel clock This pin outputs the same clock as the internal video output clock with certain amount of delay. The amount of delay can be programmable by internal registers.
	VCLK_OUT_B	O	Additional output video pixel clock The polarity and the amount of delay can be programmable by internal registers. This clock can be used to output an inverted video clock.
	HSYNC_OUT	O	Horizontal sync or active for output video data Active high or active low can be selected by internal register.
	VSYNC_OUT	O	Vertical sync or active for output video data Active high or active low can be selected by internal register.
	DE_OUT	O	Data enable for output video data This signal can be horizontal active or composite sync. Active high or active low can be selected by internal register.
	D_OUT(7:0)	O	Digital video data output 30/24-bit RGB/YCbCr 4:4:4, 20/16-bit YCbCr 4:2:2 or 10/8-bit YC-multiplexed 4:2:2 data can be output from D_OUT(7:0) and multi-function pins D_IN(21:0), HOST_AD(19:0). See Table TBD. D_OUT(3:0) are multi-function pins.
Host Interface	SCL	I	I^2C bus clock (5V tolerant & Schmitt trigger)
	SDA	I/O	I^2C bus data (5V tolerant & Schmitt trigger)
	INT_OUT	O	Interrupt output (active low or high)
	I2C_MODE	I	I^2C bus 16-bit mode selection input When open or high, the 16-bit mode (MDIN-2xx compatible) is selected. When low, the 8-bit mode is selected.

Table 1-3. MDIN-380 I/O Signal Descriptions

Group	Signal Name	I/O	Description
Host Interface	HOST_SYNC_BUS	I	Synchronous bus selection input Synchronous bus mode on high, asynchronous bus mode on low
	HOST_BUS_MODE(3:0)	I	Host bus mode selection input
	HOST_PCI_BUS	I	PCI bus selection input PCI bus mode on high, parallel bus mode on low
	HOST_CLK	I	Host bus clock input This can be used as the host interface clock instead of **XTAL_IN** on the synchronous bus mode.
	HOST_AD(19:0)	I/O	Host address input and data bus
	HOST_ALE	I	Address latch enable input (active high)
	HOST_CS_N	I	Chip select input (active low)
	HOST_DS_N	I	Data strobe input (active low) when **HOST_BUS_MODE(3)** is low Write strobe input (active low) when **HOST_BUS_MODE(3)** is high
	HOST_WR_N	I	Write enable input (active low) when **HOST_BUS_MODE(3)** is low Read strobe input (active low) when **HOST_BUS_MODE(3)** is high
	HOST_BLAST_N	I	Burst last (active low) This indicates the last transfer in a synchronous bus access.
	HOST_RDY_N	O	Host bus ready output (active low) when **HOST_BUS_MODE(2)** is low Host bus wait output (active low) when **HOST_BUS_MODE(2)** is high
	DREQ_N	O	DMA request output (active low)
	DACK_N	I	DMA acknoledge input (active low)
PCI Interface (*1)	PCI_CLK	I	Clock input from PCI bus (33MHz)
	PCI_RST#	I	Reset input from PCI bus (active low)
	PCI_AD(31:0)	I/O	PCI address input and data bus
	PCI_CBE#(3:0)	I/O	Bus command and byte enable (active low)
	PCI_PAR	I/O	Even parity across PCI_AD(31:0) and PCI_CBE#(3:0)
	PCI_FRAME#	I	Cycle frame input (active low)
	PCI_IRDY#	I	Initiator ready input (active low)
	PCI_TRDY#	O	Target ready output (active low)
	PCI_STOP#	O	Transaction stop request output (active low)
	PCI_IDSEL	I	Initialization device select
	PCI_DEVSEL#	O	Device select output (active low)
	PCI_PERR#	O	Parity error output (active low)
	PCI_SERR#	O	System error output (active low)
	PCI_INTA#	O	Interrupt A output (open drain, active low)
Test Mode	TEST_MODE(3)	I	Test mode control input For normal operation this pin should be low.
	TEST_MODE(2:1)	I	Used to select the host interface clock. Normally, connect both pins to GND and **XTAL_IN** will be used as the host interface clock. For synchronous host bus modes such as PCI interface, connect **TEST_MODE(2)** to GND and connect **TEST_MODE(1)** to VDD33, and **HOST_CLK** will be used as the host interface clock.
	TEST_MODE(0)	I	Used as the LSB of I^2C slave device address.
SDRAM	SD_DQM	I	SDRAM DQM input (should be tied to GND)
	SD_CKE	I	SDRAM CKE input (should be tied to VDD18)

Note 1) PCI interface signals are shared with other pins.

Table 1-3. MDIN-380 I/O Signal Descriptions (Continued)

Group	Signal Name	I/O	Description
GPIO (*2)	GPIO(36:0)	I/O	General purpose input and output
	GPO(2:0)	O	General purpose output on sync output pins
SPI Interface (*3)	SPI_CLK	O	Clock output to an SPI slave device
	SPI_CE_N	O	Chip enable output to an SPI slave device
	SPI_DO	O	Data output to an SPI slave device
	SPI_DI	I	Data input from an SPI slave device
PLL	XTAL_IN	I	External crystal input Either crystal or oscillator can be used for this input pin.
	XTAL_OUT	O	External crystal output
	VCLK_PLL_LF	AI	VCLK PLL loop filter This pin must be connected to external low pass filter.
	VCLK_PLL_AVDD33	VDD	VCLK PLL analog power (3.3V)
	VCLK_PLL_AGND33	GND	VCLK PLL analog ground (3.3V)
	VCLK_PLL_AVDD12	VDD	VCLK PLL analog power (1.2V)
	VCLK_PLL_AGND12	GND	VCLK PLL analog ground (1.2V)
	MCLK_PLL_AVDD12	VDD	MCLK PLL analog power (1.2V)
	MCLK_PLL_AGND12	GND	MCLK PLL analog ground
DAC	DAC_GOUT	AO	Analog green or Y output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_BOUT	AO	Analog blue or Pb output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_ROUT	AO	Analog red or Pr output from 10-bit internal DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	DAC_FSADJ	AI	Full-scale output current adjust The full-scale current on each output channel is determined by the value of a resistor connected between this pin and analog ground.
	DAC_VREF	AI	Reference voltage input Well regulated 1.2V reference voltage must be externally supplied. A 0.1uF ceramic capacitor must be connected between this pin and analog ground.
	DAC_AVDD33	VDD	Analog 3.3V power supply for each internal DAC channel
	DAC_AGND	GND	Analog ground for each internal DAC channel
Video Encoder DAC	VENC_OUT	AO	Analog output from internal video encoder DAC It can drive a doubly terminated 75Ω load (37.5Ω equivalent load)
	VENC_FSADJ	AI	Full-scale output current adjust The full-scale current on the output is determined by the value of a resistor connected between this pin and analog ground.
	VENC_VREF	AI	Reference voltage input Well regulated 1.2V reference voltage must be externally supplied. A 0.1uF ceramic capacitor must be connected between this pin and analog ground.
	VENC_AVDD33	VDD	Analog 3.3V power supply for VENC DAC
	VENC_AGND	GND	Analog ground for VENC DAC

Note 2) GPIO signals are shared with other pins.
Note 3) SPI interface signals are shared with other pins.

Table 1-3. MDIN-380 I/O Signal Descriptions (Continued)

Group	Signal Name	I/O	Description
HDMI Interface	HDMI_EXP(2:0), HDMI_EXN(2:0)	O	HDMI output data pair
	HDMI_EXCP, HDMI_EXCN	O	HDMI output clock pair
	HDMI_HPD	I	Hot plug detect input (5V tolerant)
	HDMI_DSCL	O	DDC bus clock (5V tolerant)
	HDMI_DSDA	I/O	DDC bus data (5V tolerant & Schmitt trigger)
	HDMI_CEC	I/O	CEC bus (Schmitt trigger & open drain)
	HDMI_TEST_MODE	I	Test mode control input for HDMI PHY For normal operation this pin should be low.
	HDMI_TEST_XRST	I	Test mode reset input for HDMI PHY For normal operation this pin should be high.
	HDMI_VDN12	VDD	HDMI analog power supply (1.2V)
	HDMI_VDP33	VDD	HDMI analog power supply (3.3V)
	HDMI_GND	GND	HDMI analog ground
Power & Ground	VDD12	VDD	Digital 1.2V power supply for the core logic
	VDD18	VDD	Digital 1.8V power supply for the SDRAM interface
	VDD33	VDD	Digital 3.3V power supply for the I/O
	GND	GND	Digital ground

Table 1-3. MDIN-380 I/O Signal Descriptions (Continued)

1.3 Pin Number List

1.3.1 Pin Number List for MDIN-325

Signal Name	Pin Number	I/O	Signal Name	Pin Number	I/O
RESET_N	K8	I	DATA_OUT(12) / GPIO(8)	C4	I/O
CLK_A	M2	I	DATA_OUT(11) / GPIO(7)	A5	I/O
HACTIVE_A / GPIO(36)	M9	I/O	DATA_OUT(10) / GPIO(6)	B5	I/O
VACTIVE_A / GPIO(35)	M10	I/O	DATA_OUT(9) / GPIO(5)	C5	I/O
FIELDID_A / GPIO(34)	L9	I/O	DATA_OUT(8) / GPIO(4)	B6	I/O
DATA_IN(23)	M1	I	DATA_OUT(7) / GPIO(3)	C6	I/O
DATA_IN(22)	L3	I	DATA_OUT(6) / GPIO(2)	A7	I/O
DATA_IN(21)	L2	I	DATA_OUT(5) / GPIO(1)	B7	I/O
DATA_IN(20)	L1	I	DATA_OUT(4) / GPIO(0)	C7	I/O
DATA_IN(19)	K3	I	DATA_OUT(3)	C8	O
DATA_IN(18)	K2	I	DATA_OUT(2)	A9	O
DATA_IN(17)	K1	I	DATA_OUT(1)	B9	O
DATA_IN(16)	J3	I	DATA_OUT(0)	C9	O
DATA_IN(15)	J2	I	SCL	M8	I
DATA_IN(14)	J1	I	SDA	L8	I/O
DATA_IN(13) / GPIO(33)	H3	I/O	INT_OUT	A10	O
DATA_IN(12) / GPIO(32)	H2	I/O	I2C_MODE	B10	I
DATA_IN(11) / GPIO(31)	H1	I/O	XTAL_IN	M11	I
DATA_IN(10) / GPIO(30)	G3	I/O	XTAL_OUT	M12	O
DATA_IN(9) / GPIO(29)	G2	I/O	VCLK_PLL_LF	M6	AI
DATA_IN(8) / GPIO(28)	G1	I/O	VCLK_PLL_AVDD33	L6	VDD
DATA_IN(7) / GPIO(27)	F3	I/O	VCLK_PLL_AGND33	M5	GND
DATA_IN(6) / GPIO(26)	F2	I/O	VCLK_PLL_AVDD12	L7	VDD
DATA_IN(5) / GPIO(25)	F1	I/O	VCLK_PLL_AGND12	M7	GND
DATA_IN(4) / GPIO(24)	E3	I/O	MCLK_PLL_AVDD12	M4	VDD
DATA_IN(3) / SPI_CLK / GPIO(23)	E2	I/O	MCLK_PLL_AGND12	L5	GND
DATA_IN(2) / SPI_CE_N / GPIO(22)	E1	I/O	DAC_GOUT	H12	AO
DATA_IN(1) / SPI_DO / GPIO(21)	D3	I/O	DAC_BOUT	K12	AO
DATA_IN(0) / SPI_DI / GPIO(20)	D2	I/O	DAC_ROUT	E12	AO
CLK_B	D1	I	DAC_FSADJ	G12	AI
VCLK_OUT	A8	O	DAC_VREF	F11	AI
VCLK_OUT_B	A6	O	DAC_AVDD33(2)	E11	VDD
HSYNC_OUT / GPO(2)	L11	O	DAC_AVDD33(1)	F12	VDD
VSYNC_OUT / GPO(1)	L12	O	DAC_AVDD33(0)	J12	VDD
DE_OUT / GPO(0)	B8	O	DAC_AGND(2)	G11	GND
DATA_OUT(23) / GPIO(19)	A1	I/O	DAC_AGND(1)	H11	GND
DATA_OUT(22) / GPIO(18)	B1	I/O	DAC_AGND(0)	J11	GND
DATA_OUT(21) / GPIO(17)	C1	I/O	VENC_OUT	B12	AO
DATA_OUT(20) / GPIO(16)	A2	I/O	VENC_FSADJ	C12	AI
DATA_OUT(19) / GPIO(15)	B2	I/O	VENC_VREF	A12	AI
DATA_OUT(18) / GPIO(14)	C2	I/O	VENC_AVDD33(0)	B11	VDD
DATA_OUT(17) / GPIO(13)	A3	I/O	VENC_AGND(0)	A11	GND
DATA_OUT(16) / GPIO(12)	B3	I/O	TEST_MODE(3)	K9	I
DATA_OUT(15) / GPIO(11)	C3	I/O	TEST_MODE(2)	K10	I
DATA_OUT(14) / GPIO(10)	A4	I/O	TEST_MODE(1)	K11	I
DATA_OUT(13) / GPIO(9)	B4	I/O	TEST_MODE(0)	L10	I

Table 1-4. MDIN-325 Pin Descriptions

Signal Name	Pin Number	I/O	Signal Name	Pin Number	I/O
VDD12(7)	E5	VDD	GND(22)	M3	GND
VDD12(6)	F5	VDD	GND(21)	K4	GND
VDD12(5)	G5	VDD	GND(20)	L4	GND
VDD12(4)	H5	VDD	GND(19)	K5	GND
VDD12(3)	E9	VDD	GND(18)	D6	GND
VDD12(2)	F9	VDD	GND(17)	E6	GND
VDD12(1)	G9	VDD	GND(16)	F6	GND
VDD12(0)	H9	VDD	GND(15)	G6	GND
VDD18(14)	D4	VDD	GND(14)	H6	GND
VDD18(13)	E4	VDD	GND(13)	J6	GND
VDD18(12)	F4	VDD	GND(12)	K6	GND
VDD18(11)	G4	VDD	GND(11)	K7	GND
VDD18(10)	H4	VDD	GND(10)	D8	GND
VDD18(9)	J4	VDD	GND(9)	E8	GND
VDD18(8)	D5	VDD	GND(8)	F8	GND
VDD18(7)	J5	VDD	GND(7)	G8	GND
VDD18(6)	J9	VDD	GND(6)	H8	GND
VDD18(5)	D10	VDD	GND(5)	J8	GND
VDD18(4)	E10	VDD	GND(4)	D9	GND
VDD18(3)	F10	VDD	GND(3)	C10	GND
VDD18(2)	G10	VDD	GND(2)	C11	GND
VDD18(1)	H10	VDD	GND(1)	D11	GND
VDD18(0)	J10	VDD	GND(0)	D12	GND
VDD33(5)	D7	VDD			
VDD33(4)	E7	VDD			
VDD33(3)	F7	VDD			
VDD33(2)	G7	VDD			
VDD33(1)	H7	VDD			
VDD33(0)	J7	VDD			

Table 1-4. MDIN-325 Pin Descriptions (Continued)

1.3.2 Pin Number List for MDIN-340

Signal Name	Pin Number	I/O	Signal Name	Pin Number	I/O
RESET_N	K6	I	HDMI_EXN(1)	A5	O
CLK_A	M2	I	HDMI_EXP(0)	B6	O
HACTIVE_A / GPIO(36)	K8	I/O	HDMI_EXN(0)	A6	O
VACTIVE_A / GPIO(35)	K9	I/O	HDMI_EXCP	B7	O
FIELDID_A / GPIO(34)	K7	I/O	HDMI_EXCN	A7	O
DATA_IN(23)	M1	I	HDMI_HPD	C2	I
DATA_IN(22)	L3	I	HDMI_DSCL	L8	O
DATA_IN(21)	L2	I	HDMI_DSDA	M8	I/O
DATA_IN(20)	L1	I	HDMI_CEC	E11	I/O
DATA_IN(19)	K3	I	HDMI_TEST_MODE	B1	I
DATA_IN(18)	K2	I	HDMI_TEST_XRST	C3	I
DATA_IN(17)	K1	I	SCL	M7	I
DATA_IN(16)	J3	I	SDA	L7	I/O
DATA_IN(15)	J2	I	INT_OUT	C1	O
DATA_IN(14)	J1	I	I2C_MODE	F11	I
DATA_IN(13) / GPIO(33)	H3	I/O	XTAL_IN	L9	I
DATA_IN(12) / GPIO(32)	H2	I/O	XTAL_OUT	M9	O
DATA_IN(11) / GPIO(31)	H1	I/O	VCLK_PLL_LF	M5	AI
DATA_IN(10) / GPIO(30)	G3	I/O	VCLK_PLL_AVDD33	L5	VDD
DATA_IN(9) / GPIO(29)	G2	I/O	VCLK_PLL_AGND33	M4	GND
DATA_IN(8) / GPIO(28)	G1	I/O	VCLK_PLL_AVDD12	M3	VDD
DATA_IN(7) / GPIO(27)	F3	I/O	VCLK_PLL_AGND12	L4	GND
DATA_IN(6) / GPIO(26)	F2	I/O	MCLK_PLL_AVDD12	L6	VDD
DATA_IN(5) / GPIO(25)	F1	I/O	MCLK_PLL_AGND12	M6	GND
DATA_IN(4) / GPIO(24)	E3	I/O	DAC_GOUT	J12	AO
DATA_IN(3) / SPI_CLK / GPIO(23)	E2	I/O	DAC_BOUT	L12	AO
DATA_IN(2) / SPI_CE_N / GPIO(22)	E1	I/O	DAC_ROUT	F12	AO
DATA_IN(1) / SPI_DO / GPIO(21)	D3	I/O	DAC_FSADJ	H12	AI
DATA_IN(0) / SPI_DI / GPIO(20)	D2	I/O	DAC_VREF	G11	AI
CLK_B	D1	I	DAC_AVDD33(1)	K11	VDD
MCLK_IN	D11	I	DAC_AVDD33(0)	G12	VDD
SCK_IN	A11	I	DAC_AGND(2)	L11	GND
WS_IN	A12	I	DAC_AGND(1)	J11	GND
SD_IN(3)	C12	I	DAC_AGND(0)	E12	GND
SD_IN(2)	C11	I	TEST_MODE(3)	J9	I
SD_IN(1)	B12	I	TEST_MODE(2)	K10	I
SD_IN(0)	B11	I	TEST_MODE(1)	L10	I
SPDIF_IN	D12	I	TEST_MODE(0)	M10	I
HSYNC_OUT / GPO(2)	M11	O	SD_DQM	E10	I
VSYNC_OUT / GPO(1)	M12	O	SD_CKE	F10	I
HDMI_EXP(2)	B4	O	VCLK_OUT	A1	O
HDMI_EXN(2)	A4	O	NC(1)	H11	-
HDMI_EXP(1)	B5	O	NC(0)	K12	-

Table 1-5. MDIN-340 Pin Descriptions

Signal Name	Pin Number	I/O	Signal Name	Pin Number	I/O
VDD12(7)	E5	VDD	GND(16)	D5	GND
VDD12(6)	E9	VDD	GND(15)	D6	GND
VDD12(5)	F5	VDD	GND(14)	D7	GND
VDD12(4)	F9	VDD	GND(13)	D8	GND
VDD12(3)	G9	VDD	GND(12)	E6	GND
VDD12(2)	J5	VDD	GND(11)	E8	GND
VDD12(1)	K4	VDD	GND(10)	F6	GND
VDD12(0)	K5	VDD	GND(9)	F7	GND
VDD18(12)	D4	VDD	GND(8)	F8	GND
VDD18(11)	D9	VDD	GND(7)	G6	GND
VDD18(10)	D10	VDD	GND(6)	G8	GND
VDD18(9)	E4	VDD	GND(5)	H6	GND
VDD18(8)	F4	VDD	GND(4)	H7	GND
VDD18(7)	G4	VDD	GND(3)	H8	GND
VDD18(6)	G5	VDD	GND(2)	H9	GND
VDD18(5)	G10	VDD	GND(1)	J6	GND
VDD18(4)	H4	VDD	GND(0)	J8	GND
VDD18(3)	H5	VDD	HDMI_VDN12(3)	A8	VDD
VDD18(2)	H10	VDD	HDMI_VDN12(2)	B8	VDD
VDD18(1)	J4	VDD	HDMI_VDN12(1)	C6	VDD
VDD18(0)	J10	VDD	HDMI_VDN12(0)	C9	VDD
VDD33(2)	E7	VDD	HDMI_VDP33(1)	C4	VDD
VDD33(1)	G7	VDD	HDMI_VDP33(0)	C8	VDD
VDD33(0)	J7	VDD	HDMI_GND(5)	A3	GND
GND(21)	A2	GND	HDMI_GND(4)	B3	GND
GND(20)	A10	GND	HDMI_GND(3)	C5	GND
GND(19)	B2	GND	HDMI_GND(2)	C7	GND
GND(18)	B10	GND	HDMI_GND(1)	A9	GND
GND(17)	C10	GND	HDMI_GND(0)	B9	GND

Table 1-5. MDIN-340 Pin Descriptions (Continued)

1.3.3 Pin Number List for MDIN-380

Signal Name	Pin Number	I/O	Signal Name	Pin Number	I/O
RESET_N / PCI_RST#	E14	I	SD_IN(1)	J2	I
CLK_A	N14	I	SD_IN(0)	J1	I
HACTIVE_A / GPIO(36)	F13	I/O	SPDIF_IN	H1	I
VACTIVE_A / GPIO(35)	E12	I/O	VCLK_OUT	W7	O
FIELDID_A / GPIO(34)	F12	I/O	VCLK_OUT_B	W5	O
D_IN(35)	M14	I	HSYNC_OUT / GPO(2)	C13	O
D_IN(34)	M13	I	VSYNC_OUT / GPO(1)	C14	O
D_IN(33)	M12	I	DE_OUT / GPO(0)	V6	O
D_IN(32)	N13	I	D_OUT(7) / GPIO(7)	W8	I/O
D_IN(31)	N12	I	D_OUT(6) / GPIO(6)	V8	I/O
D_IN(30)	N11	I	D_OUT(5) / GPIO(5)	U8	I/O
D_IN(29)	P14	I	D_OUT(4) / GPIO(4)	T8	I/O
D_IN(28)	P13	I	D_OUT(3) / PCI_AD(23) / GPIO(3)	V7	I/O
D_IN(27)	P12	I	D_OUT(2) / PCI_AD(22) / GPIO(2)	U7	I/O
D_IN(26)	P11	I	D_OUT(1) / PCI_AD(21) / GPIO(1)	T7	I/O
D_IN(25) / GPIO(33)	R14	I/O	D_OUT(0) / PCI_AD(20) / GPIO(0)	W6	I/O
D_IN(24) / GPIO(32)	R13	I/O	SCL	F14	I
D_IN(23) / GPIO(31)	R12	I/O	SDA	E13	I/O
D_IN(22) / GPIO(30)	R11	I/O	INT_OUT	T6	O
D_IN(21) / GPIO(29)	T14	I/O	I2C_MODE	D7	I
D_IN(20) / GPIO(28)	T13	I/O	HOST_SYNC_BUS	C5	I
D_IN(19) / GPIO(27)	T12	I/O	HOST_BUS_MODE(3)	D6	I
D_IN(18) / GPIO(26)	T11	I/O	HOST_BUS_MODE(2) / PCI_PERR#	M2	I/O
D_IN(17) / GPIO(25)	U14	I/O	HOST_BUS_MODE(1) / PCI_SERR#	M3	I/O
D_IN(16) / GPIO(24)	U13	I/O	HOST_BUS_MODE(0) / PCI_INTA#	L2	I/O
D_IN(15) / SPI_CLK / GPIO(23)	U12	I/O	HOST_PCI_BUS	D5	I
D_IN(14) / SPI_CE_N / GPIO(22)	U11	I/O	HOST_CLK / PCI_CLK	L1	I
D_IN(13) / SPI_DO / GPIO(21)	V14	I/O	HOST_AD(19) / PCI_AD(19)	W2	I/O
D_IN(12) / SPI_DI / GPIO(20)	V13	I/O	HOST_AD(18) / PCI_AD(18)	W3	I/O
D_IN(11) / PCI_CBE#(3) / GPIO(19)	V12	I/O	HOST_AD(17) / PCI_AD(17)	V1	I/O
D_IN(10) / PCI_CBE#(2) / GPIO(18)	V11	I/O	HOST_AD(16) / PCI_AD(16)	V2	I/O
D_IN(9) / PCI_CBE#(1) / GPIO(17)	W13	I/O	HOST_AD(15) / PCI_AD(15)	V3	I/O
D_IN(8) / PCI_CBE#(0) / GPIO(16)	W12	I/O	HOST_AD(14) / PCI_AD(14)	U1	I/O
D_IN(7) / PCI_AD(31) / GPIO(15)	W11	I/O	HOST_AD(13) / PCI_AD(13)	U2	I/O
D_IN(6) / PCI_AD(30) / GPIO(14)	W10	I/O	HOST_AD(12) / PCI_AD(12)	U3	I/O
D_IN(5) / PCI_AD(29) / GPIO(13)	V10	I/O	HOST_AD(11) / PCI_AD(11)	T1	I/O
D_IN(4) / PCI_AD(28) / GPIO(12)	U10	I/O	HOST_AD(10) / PCI_AD(10)	T2	I/O
D_IN(3) / PCI_AD(27) / GPIO(11)	T10	I/O	HOST_AD(9) / PCI_AD(9)	T3	I/O
D_IN(2) / PCI_AD(26) / GPIO(10) / VACT_B	V9	I/O	HOST_AD(8) / PCI_AD(8)	R1	I/O
D_IN(1) / PCI_AD(25) / GPIO(9) / HACT_B	U9	I/O	HOST_AD(7) / PCI_AD(7)	R2	I/O
D_IN(0) / PCI_AD(24) / GPIO(8) / FID_B	T9	I/O	HOST_AD(6) / PCI_AD(6)	R3	I/O
CLK_B	W9	I	HOST_AD(5) / PCI_AD(5)	P1	I/O
MCLK_IN	H2	I	HOST_AD(4) / PCI_AD(4)	P2	I/O
SCK_IN	L3	I	HOST_AD(3) / PCI_AD(3)	P3	I/O
WS_IN	K1	I	HOST_AD(2) / PCI_AD(2)	N1	I/O
SD_IN(3)	K2	I	HOST_AD(1) / PCI_AD(1)	N2	I/O
SD_IN(2)	K3	I	HOST_AD(0) / PCI_AD(0)	M1	I/O

Table 1-6. MDIN-380 Pin Descriptions

Signal Name	Pin Number	I/O	Signal Name	Pin Number	I/O
HOST_ALE / PCI_FRAME#	T4	I	HDMI_DSCL	D12	O
HOST_CS_N / PCI_IRDY#	U4	I	HDMI_DSDA	D11	I/O
HOST_DS_N / PCI_PAR	V4	I/O	HDMI_CEC	G1	I/O
HOST_WR_N / PCI_STOP#	W4	I/O	HDMI_TEST_MODE	C11	I
HOST_BLAST_N / PCI_IDSEL	T5	I	HDMI_TEST_XRST	C10	I
HOST_RDY_N / PCI_TRDY#	U5	O	HDMI_VDN12(2)	A2	VDD
DREQ_N / PCI_DEVSEL#	U6	O	HDMI_VDN12(1)	B3	VDD
DACK_N	V5	I	HDMI_VDN12(0)	E3	VDD
TEST_MODE(3)	C9	I	HDMI_VDP33(1)	C3	VDD
TEST_MODE(2)	C8	I	HDMI_VDP33(0)	F3	VDD
TEST_MODE(1)	C7	I	HDMI_GND(2)	A3	GND
TEST_MODE(0)	C6	I	HDMI_GND(1)	D3	GND
SD_DQM	R6	I	HDMI_GND(0)	F2	GND
SD_CKE	F5	I	VDD12(9)	E7	VDD
XTAL_IN	D13	I	VDD12(8)	E8	VDD
XTAL_OUT	D14	O	VDD12(7)	E9	VDD
VCLK_PLL_LF	K14	AI	VDD12(6)	H3	VDD
VCLK_PLL_AVDD33	J14	VDD	VDD12(5)	J3	VDD
VCLK_PLL_AGND33	J13	GND	VDD12(4)	K12	VDD
VCLK_PLL_AVDD12	H14	VDD	VDD12(3)	L12	VDD
VCLK_PLL_AGND12	H13	GND	VDD12(2)	R7	VDD
MCLK_PLL_AVDD12	K13	VDD	VDD12(1)	R8	VDD
MCLK_PLL_AGND12	J12	GND	VDD12(0)	R9	VDD
DAC_GOUT	A12	AO	VDD18(24)	F10	VDD
DAC_BOUT	B14	AO	VDD18(23)	F11	VDD
DAC_ROUT	A9	AO	VDD18(22)	F6	VDD
DAC_FSADJ	A11	AI	VDD18(21)	F7	VDD
DAC_VREF	B11	AI	VDD18(20)	F8	VDD
DAC_AVDD33(2)	B9	VDD	VDD18(19)	F9	VDD
DAC_AVDD33(1)	B10	VDD	VDD18(18)	G10	VDD
DAC_AVDD33(0)	B12	VDD	VDD18(17)	G11	VDD
DAC_AGND(2)	A10	GND	VDD18(16)	G5	VDD
DAC_AGND(1)	A13	GND	VDD18(15)	G6	VDD
DAC_AGND(0)	B13	GND	VDD18(14)	G7	VDD
VENC_OUT	A7	AO	VDD18(13)	G8	VDD
VENC_FSADJ	B6	AI	VDD18(12)	G9	VDD
VENC_VREF	B5	AI	VDD18(11)	N10	VDD
VENC_AVDD33	B7	VDD	VDD18(10)	N3	VDD
VENC_AGND	A6	GND	VDD18(9)	N4	VDD
HDMI_EXP(2)	E2	O	VDD18(8)	N5	VDD
HDMI_EXN(2)	E1	O	VDD18(7)	N6	VDD
HDMI_EXP(1)	D2	O	VDD18(6)	N7	VDD
HDMI_EXN(1)	D1	O	VDD18(5)	N8	VDD
HDMI_EXP(0)	C2	O	VDD18(4)	N9	VDD
HDMI_EXN(0)	C1	O	VDD18(3)	P10	VDD
HDMI_EXCP	B2	O	VDD18(2)	P4	VDD
HDMI_EXCN	B1	O	VDD18(1)	P9	VDD
HDMI_HPD	C12	I	VDD18(0)	R10	VDD

Table 1-6. MDIN-380 Pin Descriptions (Continued)

Signal Name	Pin Number	I/O	Signal Name	Pin Number	I/O
VDD33(6)	J10	VDD	GND(24)	G13	GND
VDD33(5)	J5	VDD	GND(23)	G14	GND
VDD33(4)	J7	VDD	GND(22)	G2	GND
VDD33(3)	K10	VDD	GND(21)	G3	GND
VDD33(2)	K5	VDD	GND(20)	G4	GND
VDD33(1)	L10	VDD	GND(19)	H12	GND
VDD33(0)	L5	VDD	GND(18)	J6	GND
GND(42)	A4	GND	GND(17)	J8	GND
GND(41)	A5	GND	GND(16)	J9	GND
GND(40)	A8	GND	GND(15)	K6	GND
GND(39)	B4	GND	GND(14)	K7	GND
GND(38)	B8	GND	GND(13)	K8	GND
GND(37)	C4	GND	GND(12)	K9	GND
GND(36)	D10	GND	GND(11)	L13	GND
GND(35)	D4	GND	GND(10)	L14	GND
GND(34)	D8	GND	GND(9)	L6	GND
GND(33)	D9	GND	GND(8)	L7	GND
GND(32)	E10	GND	GND(7)	L8	GND
GND(31)	E11	GND	GND(6)	L9	GND
GND(30)	E4	GND	GND(5)	P5	GND
GND(29)	E5	GND	GND(4)	P6	GND
GND(28)	E6	GND	GND(3)	P7	GND
GND(27)	F1	GND	GND(2)	P8	GND
GND(26)	F4	GND	GND(1)	R4	GND
GND(25)	G12	GND	GND(0)	R5	GND

Table 1-6. MDIN-380 Pin Descriptions (Continued)

2. Functional Descriptions

2.1 Input Port Pin Mapping

2.1.1 MDIN-325/340 Input Port Pin Mapping

MDIN-325/340 has 24 configurable input pins and three pin mapping modes as shown in the following tables. The pin mapping mode is selected by ***in_data_map_mode*** of the register ***in_format_ctrl*** and the input port pin mapping is determined by ***in_mode_a*** and ***in_mode_b*** of the register ***in_sync_control*** for each pin mapping mode. Since MDIN-325/340 only supports the 24-pin input mode, ***in_data_24pin_mode*** should always be set to '1'.

Mode / Pin Name	Dual YC-muxed ITU-R656 10-bit	Dual YC-muxed ITU-R656 8-bit	Single YC 4:2:2 20-bit		Single YC 4:2:2 16-bit		Single YCbCr 4:4:4 24-bit	Single RGB 4:4:4 24-bit
DATA_IN0	-	-	-		-		Cb_A0	B_A0
DATA_IN1	-	-	-		-		Cb_A1	B_A1
DATA_IN2	-	-	-		-		Cb_A2	B_A2
DATA_IN3	-	-	-		-		Cb_A3	B_A3
DATA_IN4	YC_B0	-	C_A0		-		Cb_A4	B_A4
DATA_IN5	YC_B1	-	C_A1		-		Cb_A5	B_A5
DATA_IN6	YC_B2	YC_B0	C_A2		C_A0		Cb_A6	B_A6
DATA_IN7	YC_B3	YC_B1	C_A3		C_A1		Cb_A7	B_A7
DATA_IN8	YC_B4	YC_B2	C_A4		C_A2		Y_A0	G_A0
DATA_IN9	YC_B5	YC_B3	C_A5		C_A3		Y_A1	G_A1
DATA_IN10	YC_B6	YC_B4	C_A6		C_A4		Y_A2	G_A2
DATA_IN11	YC_B7	YC_B5	C_A7		C_A5		Y_A3	G_A3
DATA_IN12	YC_B8	YC_B6	C_A8		C_A6		Y_A4	G_A4
DATA_IN13	YC_B9	YC_B7	C_A9		C_A7		Y_A5	G_A5
DATA_IN14	YC_A0	-	Y_A0		-		Y_A6	G_A6
DATA_IN15	YC_A1	-	Y_A1		-		Y_A7	G_A7
DATA_IN16	YC_A2	YC_A0	Y_A2		Y_A0		Cr_A0	R_A0
DATA_IN17	YC_A3	YC_A1	Y_A3		Y_A1		Cr_A1	R_A1
DATA_IN18	YC_A4	YC_A2	Y_A4		Y_A2		Cr_A2	R_A2
DATA_IN19	YC_A5	YC_A3	Y_A5		Y_A3		Cr_A3	R_A3
DATA_IN20	YC_A6	YC_A4	Y_A6		Y_A4		Cr_A4	R_A4
DATA_IN21	YC_A7	YC_A5	Y_A7		Y_A5		Cr_A5	R_A5
DATA_IN22	YC_A8	YC_A6	Y_A8		Y_A6		Cr_A6	R_A6
DATA_IN23	YC_A9	YC_A7	Y_A9		Y_A7		Cr_A7	R_A7
HACTIVE_A	-	-	-	HACT_A	-	HACT_A	HACT_A	HACT_A
VACTIVE_A	-	-	-	VACT_A	-	VACT_A	VACT_A	VACT_A
FIELDID_A	-	-	-	FID_A	-	FID_A	FID_A	FID_A
in_mode_b	'10'		-		-		-	
in_mode_a	'10'		'11'	'01'	'11'	'01'	'00'	
main_a_sel	'1' or '0'	'1' or '0'	'1' (Only port A can be used.)					

(a) When ***in_data_map_mode*** = '00'

Table 2-1. MDIN-325/340 Input Port Pin Mapping

Mode / Pin Name	Dual YC-muxed ITU-R656 10-bit	Dual YC-muxed ITU-R656 8-bit	Single YC 4:2:2 20-bit		Single YC 4:2:2 16-bit		Single YCbCr 4:4:4 24-bit	Single RGB 4:4:4 24-bit
DATA_IN0	YC_B9	YC_B7	C_A9		C_A7		Cr_A7	R_A7
DATA_IN1	YC_B8	YC_B6	C_A8		C_A6		Cr_A6	R_A6
DATA_IN2	YC_B7	YC_B5	C_A7		C_A5		Cr_A5	R_A5
DATA_IN3	YC_B6	YC_B4	C_A6		C_A4		Cr_A4	R_A4
DATA_IN4	YC_B5	YC_B3	C_A5		C_A3		Cr_A3	R_A3
DATA_IN5	YC_B4	YC_B2	C_A4		C_A2		Cr_A2	R_A2
DATA_IN6	YC_B3	YC_B1	C_A3		C_A1		Cr_A1	R_A1
DATA_IN7	YC_B2	YC_B0	C_A2		C_A0		Cr_A0	R_A0
DATA_IN8	YC_A9	YC_A7	Y_A9		Y_A7		Y_A7	G_A7
DATA_IN9	YC_A8	YC_A6	Y_A8		Y_A6		Y_A6	G_A6
DATA_IN10	YC_A7	YC_A5	Y_A7		Y_A5		Y_A5	G_A5
DATA_IN11	YC_A6	YC_A4	Y_A6		Y_A4		Y_A4	G_A4
DATA_IN12	YC_A5	YC_A3	Y_A5		Y_A3		Y_A3	G_A3
DATA_IN13	YC_A4	YC_A2	Y_A4		Y_A2		Y_A2	G_A2
DATA_IN14	YC_A3	YC_A1	Y_A3		Y_A1		Y_A1	G_A1
DATA_IN15	YC_A2	YC_A0	Y_A2		Y_A0		Y_A0	G_A0
DATA_IN16	YC_B1	-	C_A1		-		Cb_A7	B_A7
DATA_IN17	YC_B0	-	C_A0		-		Cb_A6	B_A6
DATA_IN18	-	-	-		-		Cb_A5	B_A5
DATA_IN19	-	-	-		-		Cb_A4	B_A4
DATA_IN20	YC_A1	-	Y_A1		-		Cb_A3	B_A3
DATA_IN21	YC_A0	-	Y_A0		-		Cb_A2	B_A2
DATA_IN22	-	-	-		-		Cb_A1	B_A1
DATA_IN23	-	-	-		-		Cb_A0	B_A0
HACTIVE_A	-	-	-	HACT_A	-	HACT_A	HACT_A	HACT_A
VACTIVE_A	-	-	-	VACT_A	-	VACT_A	VACT_A	VACT_A
FIELDID_A	-	-	-	FID_A	-	FID_A	FID_A	FID_A
in_mode_b	'10'		-		-		-	
in_mode_a	'10'		'11'	'01'	'11'	'01'	'00'	
main_a_sel	'1' or '0'		'1' (Only port A can be used.)					

(b) When ***in_data_map_mode*** = '10'

Table 2-1. MDIN-325/340 Input Port Pin Mapping (Continued)

Pin Name \ Mode	Dual YC-muxed ITU-R656 10-bit	Dual YC-muxed ITU-R656 8-bit	Single YC 4:2:2 20-bit		Single YC 4:2:2 16-bit		Single YCbCr 4:4:4 24-bit	Single RGB 4:4:4 24-bit
DATA_IN0	-	-	-		-		Cb_A0	B_A0
DATA_IN1	-	-	-		-		Cb_A1	B_A1
DATA_IN2	YC_A0	-	Y_A0		-		Cb_A2	B_A2
DATA_IN3	YC_A1	-	Y_A1		-		Cb_A3	B_A3
DATA_IN4	-	-	-		-		Cb_A4	B_A4
DATA_IN5	-	-	-		-		Cb_A5	B_A5
DATA_IN6	YC_B0	-	C_A0		-		Cb_A6	B_A6
DATA_IN7	YC_B1	-	C_A1		-		Cb_A7	B_A7
DATA_IN8	YC_A2	YC_A0	Y_A2		Y_A0		Y_A0	G_A0
DATA_IN9	YC_A3	YC_A1	Y_A3		Y_A1		Y_A1	G_A1
DATA_IN10	YC_A4	YC_A2	Y_A4		Y_A2		Y_A2	G_A2
DATA_IN11	YC_A5	YC_A3	Y_A5		Y_A3		Y_A3	G_A3
DATA_IN12	YC_A6	YC_A4	Y_A6		Y_A4		Y_A4	G_A4
DATA_IN13	YC_A7	YC_A5	Y_A7		Y_A5		Y_A5	G_A5
DATA_IN14	YC_A8	YC_A6	Y_A8		Y_A6		Y_A6	G_A6
DATA_IN15	YC_A9	YC_A7	Y_A9		Y_A7		Y_A7	G_A7
DATA_IN16	YC_B2	YC_B0	C_A2		C_A0		Cr_A0	R_A0
DATA_IN17	YC_B3	YC_B1	C_A3		C_A1		Cr_A1	R_A1
DATA_IN18	YC_B4	YC_B2	C_A4		C_A2		Cr_A2	R_A2
DATA_IN19	YC_B5	YC_B3	C_A5		C_A3		Cr_A3	R_A3
DATA_IN20	YC_B6	YC_B4	C_A6		C_A4		Cr_A4	R_A4
DATA_IN21	YC_B7	YC_B5	C_A7		C_A5		Cr_A5	R_A5
DATA_IN22	YC_B8	YC_B6	C_A8		C_A6		Cr_A6	R_A6
DATA_IN23	YC_B9	YC_B7	C_A9		C_A7		Cr_A7	R_A7
HACTIVE_A	-	-	-	HACT_A	-	HACT_A	HACT_A	HACT_A
VACTIVE_A	-	-	-	VACT_A	-	VACT_A	VACT_A	VACT_A
FIELDID_A	-	-	-	FID_A	-	FID_A	FID_A	FID_A
in_mode_b	'10'		-		-		-	
in_mode_a	'10'		'11'	'01'	'11'	'01'	'00'	
main_a_sel	'1' or '0'		'1' (Only port A can be used.)					

(c) When ***in_data_map_mode*** = '11'

Table 2-1. MDIN-325/340 Input Port Pin Mapping (Continued)

2.1.2 MDIN-380 Input Port Pin Mapping

MDIN-380 has 36 configurable input pins, and it has two input modes – 24-pin input mode and 36-pin input mode. On the 24-pin input mode, only upper 24 pins **D_IN(35:12)** of 36 pins are used as input pins and the three pin mapping modes are the same as those of MDIN-325/340 as shown in the following tables. The pin mapping mode is selected by ***in_data_map_mode*** of the register ***in_format_ctrl*** and the input port pin mapping is determined by ***in_mode_a*** and ***in_mode_b*** of the register ***in_sync_control*** for each pin mapping mode.
For using the 24-pin input mode, ***in_data_24pin_mode*** should be set to '1'.

Mode / Pin Name	Dual YC-muxed ITU-R656 10-bit	Dual YC-muxed ITU-R656 8-bit	Single YC 4:2:2 20-bit	Single YC 4:2:2 16-bit	Single YCbCr 4:4:4 24-bit	Single RGB 4:4:4 24-bit
D_IN12	-	-	-	-	Cb_A0	B_A0
D_IN13	-	-	-	-	Cb_A1	B_A1
D_IN14	-	-	-	-	Cb_A2	B_A2
D_IN15	-	-	-	-	Cb_A3	B_A3
D_IN16	YC_B0	-	C_A0	-	Cb_A4	B_A4
D_IN17	YC_B1	-	C_A1	-	Cb_A5	B_A5
D_IN18	YC_B2	YC_B0	C_A2	C_A0	Cb_A6	B_A6
D_IN19	YC_B3	YC_B1	C_A3	C_A1	Cb_A7	B_A7
D_IN20	YC_B4	YC_B2	C_A4	C_A2	Y_A0	G_A0
D_IN21	YC_B5	YC_B3	C_A5	C_A3	Y_A1	G_A1
D_IN22	YC_B6	YC_B4	C_A6	C_A4	Y_A2	G_A2
D_IN23	YC_B7	YC_B5	C_A7	C_A5	Y_A3	G_A3
D_IN24	YC_B8	YC_B6	C_A8	C_A6	Y_A4	G_A4
D_IN25	YC_B9	YC_B7	C_A9	C_A7	Y_A5	G_A5
D_IN26	YC_A0	-	Y_A0	-	Y_A6	G_A6
D_IN27	YC_A1	-	Y_A1	-	Y_A7	G_A7
D_IN28	YC_A2	YC_A0	Y_A2	Y_A0	Cr_A0	R_A0
D_IN29	YC_A3	YC_A1	Y_A3	Y_A1	Cr_A1	R_A1
D_IN30	YC_A4	YC_A2	Y_A4	Y_A2	Cr_A2	R_A2
D_IN31	YC_A5	YC_A3	Y_A5	Y_A3	Cr_A3	R_A3
D_IN32	YC_A6	YC_A4	Y_A6	Y_A4	Cr_A4	R_A4
D_IN33	YC_A7	YC_A5	Y_A7	Y_A5	Cr_A5	R_A5
D_IN34	YC_A8	YC_A6	Y_A8	Y_A6	Cr_A6	R_A6
D_IN35	YC_A9	YC_A7	Y_A9	Y_A7	Cr_A7	R_A7
HACTIVE_A	-	-	- / HACT_A	- / HACT_A	HACT_A	HACT_A
VACTIVE_A	-	-	- / VACT_A	- / VACT_A	VACT_A	VACT_A
FIELDID_A	-	-	- / FID_A	- / FID_A	FID_A	FID_A
in_mode_b	'10'		-	-	-	
in_mode_a	'10'		'11' / '01'	'11' / '01'	'00'	
main_a_sel	'1' or '0'	'1' or '0'	'1' (Only port A can be used.)			

(a) When ***in_data_map_mode*** = '00'

Table 2-2. MDIN-380 Input Port Pin Mapping on 24-pin Input Mode

Pin Name \ Mode	Dual YC-muxed ITU-R656 10-bit	Dual YC-muxed ITU-R656 8-bit	Single YC 4:2:2 20-bit		Single YC 4:2:2 16-bit		Single YCbCr 4:4:4 24-bit	Single RGB 4:4:4 24-bit
D_IN12	YC_B9	YC_B7	C_A9		C_A7		Cr_A7	R_A7
D_IN13	YC_B8	YC_B6	C_A8		C_A6		Cr_A6	R_A6
D_IN14	YC_B7	YC_B5	C_A7		C_A5		Cr_A5	R_A5
D_IN15	YC_B6	YC_B4	C_A6		C_A4		Cr_A4	R_A4
D_IN16	YC_B5	YC_B3	C_A5		C_A3		Cr_A3	R_A3
D_IN17	YC_B4	YC_B2	C_A4		C_A2		Cr_A2	R_A2
D_IN18	YC_B3	YC_B1	C_A3		C_A1		Cr_A1	R_A1
D_IN19	YC_B2	YC_B0	C_A2		C_A0		Cr_A0	R_A0
D_IN20	YC_A9	YC_A7	Y_A9		Y_A7		Y_A7	G_A7
D_IN21	YC_A8	YC_A6	Y_A8		Y_A6		Y_A6	G_A6
D_IN22	YC_A7	YC_A5	Y_A7		Y_A5		Y_A5	G_A5
D_IN23	YC_A6	YC_A4	Y_A6		Y_A4		Y_A4	G_A4
D_IN24	YC_A5	YC_A3	Y_A5		Y_A3		Y_A3	G_A3
D_IN25	YC_A4	YC_A2	Y_A4		Y_A2		Y_A2	G_A2
D_IN26	YC_A3	YC_A1	Y_A3		Y_A1		Y_A1	G_A1
D_IN27	YC_A2	YC_A0	Y_A2		Y_A0		Y_A0	G_A0
D_IN28	YC_B1	-	C_A1		-		Cb_A7	B_A7
D_IN29	YC_B0	-	C_A0		-		Cb_A6	B_A6
D_IN30	-	-	-		-		Cb_A5	B_A5
D_IN31	-	-	-		-		Cb_A4	B_A4
D_IN32	YC_A1	-	Y_A1		-		Cb_A3	B_A3
D_IN33	YC_A0	-	Y_A0		-		Cb_A2	B_A2
D_IN34	-	-	-		-		Cb_A1	B_A1
D_IN35	-	-	-		-		Cb_A0	B_A0
HACTIVE_A	-	-	-	HACT_A	-	HACT_A	HACT_A	HACT_A
VACTIVE_A	-	-	-	VACT_A	-	VACT_A	VACT_A	VACT_A
FIELDID_A	-	-	-	FID_A	-	FID_A	FID_A	FID_A
in_mode_b	‘10’		-		-		-	
in_mode_a	‘10’		‘11’	‘01’	‘11’	‘01’	‘00’	
main_a_sel	‘1’ or ‘0’		‘1’ (Only port A can be used.)					

(b) When ***in_data_map_mode*** = ‘10’

Table 2-2. MDIN-380 Input Port Pin Mapping on 24-pin Input Mode (Continued)

Pin Name \ Mode	Dual YC-muxed ITU-R656 10-bit	Dual YC-muxed ITU-R656 8-bit	Single YC 4:2:2 20-bit		Single YC 4:2:2 16-bit		Single YCbCr 4:4:4 24-bit	Single RGB 4:4:4 24-bit
D_IN12	-	-	-		-		Cb_A0	B_A0
D_IN13	-	-	-		-		Cb_A1	B_A1
D_IN14	YC_A0	-	Y_A0		-		Cb_A2	B_A2
D_IN15	YC_A1	-	Y_A1		-		Cb_A3	B_A3
D_IN16	-	-	-		-		Cb_A4	B_A4
D_IN17	-	-	-		-		Cb_A5	B_A5
D_IN18	YC_B0	-	C_A0		-		Cb_A6	B_A6
D_IN19	YC_B1	-	C_A1		-		Cb_A7	B_A7
D_IN20	YC_A2	YC_A0	Y_A2		Y_A0		Y_A0	G_A0
D_IN21	YC_A3	YC_A1	Y_A3		Y_A1		Y_A1	G_A1
D_IN22	YC_A4	YC_A2	Y_A4		Y_A2		Y_A2	G_A2
D_IN23	YC_A5	YC_A3	Y_A5		Y_A3		Y_A3	G_A3
D_IN24	YC_A6	YC_A4	Y_A6		Y_A4		Y_A4	G_A4
D_IN25	YC_A7	YC_A5	Y_A7		Y_A5		Y_A5	G_A5
D_IN26	YC_A8	YC_A6	Y_A8		Y_A6		Y_A6	G_A6
D_IN27	YC_A9	YC_A7	Y_A9		Y_A7		Y_A7	G_A7
D_IN28	YC_B2	YC_B0	C_A2		C_A0		Cr_A0	R_A0
D_IN29	YC_B3	YC_B1	C_A3		C_A1		Cr_A1	R_A1
D_IN30	YC_B4	YC_B2	C_A4		C_A2		Cr_A2	R_A2
D_IN31	YC_B5	YC_B3	C_A5		C_A3		Cr_A3	R_A3
D_IN32	YC_B6	YC_B4	C_A6		C_A4		Cr_A4	R_A4
D_IN33	YC_B7	YC_B5	C_A7		C_A5		Cr_A5	R_A5
D_IN34	YC_B8	YC_B6	C_A8		C_A6		Cr_A6	R_A6
D_IN35	YC_B9	YC_B7	C_A9		C_A7		Cr_A7	R_A7
HACTIVE_A	-	-	-	HACT_A	-	HACT_A	HACT_A	HACT_A
VACTIVE_A	-	-	-	VACT_A	-	VACT_A	VACT_A	VACT_A
FIELDID_A	-	-	-	FID_A	-	FID_A	FID_A	FID_A
in_mode_b	‘10’		-		-		-	
in_mode_a	‘10’		‘11’	‘01’	‘11’	‘01’	‘00’	
main_a_sel	‘1’ or ‘0’		‘1’ (Only port A can be used.)					

(c) When ***in_data_map_mode*** = ‘11’

Table 2-2. MDIN-380 Input Port Pin Mapping on 24-pin Input Mode (Continued)

On the 36-pin input mode, all the 36 pins can be used as input pins and there are four pin mapping modes as shown in the following tables. For using the 36-pin input mode, ***in_data_24pin_mode*** should be set to '0'.

Mode / Pin Name	Dual YC-muxed ITU-R656 10-bit	Dual YC 4:2:2 20-bit / 16-bit		Single YCbCr 4:4:4 30-bit	Single RGB 4:4:4 30-bit
D_IN0	-	C_B0		-	-
D_IN1	-	C_B1		-	-
D_IN2	-	C_B2		-	-
D_IN3	-	C_B3		-	-
D_IN4	-	C_B4		-	-
D_IN5	-	C_B5		-	-
D_IN6	YC_B0	C_B6		Cr_A0	R_A0
D_IN7	YC_B1	C_B7		Cr_A1	R_A1
D_IN8	YC_B2	Y_B0		Cr_A2	R_A2
D_IN9	YC_B3	Y_B1		Cr_A3	R_A3
D_IN10	YC_B4	Y_B2		Cr_A4	R_A4
D_IN11	YC_B5	Y_B3		Cr_A5	R_A5
D_IN12	YC_B6	Y_B4		Cr_A6	R_A6
D_IN13	YC_B7	Y_B5		Cr_A7	R_A7
D_IN14	YC_B8	Y_B6		Cr_A8	R_A8
D_IN15	YC_B9	Y_B7		Cr_A9	R_A9
D_IN16	-	C_A0		Cb_A0	B_A0
D_IN17	-	C_A1		Cb_A1	B_A1
D_IN18	-	C_A2		Cb_A2	B_A2
D_IN19	-	C_A3		Cb_A3	B_A3
D_IN20	-	C_A4		Cb_A4	B_A4
D_IN21	-	C_A5		Cb_A5	B_A5
D_IN22	-	C_A6		Cb_A6	B_A6
D_IN23	-	C_A7		Cb_A7	B_A7
D_IN24	-	C_A8		Cb_A8	B_A8
D_IN25	-	C_A9		Cb_A9	B_A9
D_IN26	YC_A0	Y_A0		Y_A0	G_A0
D_IN27	YC_A1	Y_A1		Y_A1	G_A1
D_IN28	YC_A2	Y_A2		Y_A2	G_A2
D_IN29	YC_A3	Y_A3		Y_A3	G_A3
D_IN30	YC_A4	Y_A4		Y_A4	G_A4
D_IN31	YC_A5	Y_A5		Y_A5	G_A5
D_IN32	YC_A6	Y_A6		Y_A6	G_A6
D_IN33	YC_A7	Y_A7		Y_A7	G_A7
D_IN34	YC_A8	Y_A8		Y_A8	G_A8
D_IN35	YC_A9	Y_A9		Y_A9	G_A9
HACTIVE_A	-	-	HACT_A	HACT_A	HACT_A
VACTIVE_A	-	-	VACT_A	VACT_A	VACT_A
FIELDID_A	-	-	FID_A	FID_A	FID_A
in_mode_b	'10'	'11'		-	
in_mode_a	'10'	'11'	'01'	'00'	
main_a_sel	'1' or '0'	'1' (Only port A can be used.)			

(a) When ***in_data_map_mode*** = '00'

Table 2-3. MDIN-380 Input Port Pin Mapping on 36-pin Input Mode

Pin Name \ Mode	Dual YC-muxed ITU-R656 8-bit	Dual YC 4:2:2 16-bit	
D_IN0	-	-	FID_B
D_IN1	-	-	HACT_B
D_IN2	-	-	VACT_B
D_IN3	-	-	
D_IN4	-	C_B0	
D_IN5	-	C_B1	
D_IN6	-	C_B2	
D_IN7	-	C_B3	
D_IN8	-	C_B4	
D_IN9	-	C_B5	
D_IN10	-	C_B6	
D_IN11	-	C_B7	
D_IN12	YC_B0	Y_B0	
D_IN13	YC_B1	Y_B1	
D_IN14	YC_B2	Y_B2	
D_IN15	YC_B3	Y_B3	
D_IN16	YC_B4	Y_B4	
D_IN17	YC_B5	Y_B5	
D_IN18	YC_B6	Y_B6	
D_IN19	YC_B7	Y_B7	
D_IN20	-	C_A0	
D_IN21	-	C_A1	
D_IN22	-	C_A2	
D_IN23	-	C_A3	
D_IN24	-	C_A4	
D_IN25	-	C_A5	
D_IN26	-	C_A6	
D_IN27	-	C_A7	
D_IN28	YC_A0	Y_A0	
D_IN29	YC_A1	Y_A1	
D_IN30	YC_A2	Y_A2	
D_IN31	YC_A3	Y_A3	
D_IN32	YC_A4	Y_A4	
D_IN33	YC_A5	Y_A5	
D_IN34	YC_A6	Y_A6	
D_IN35	YC_A7	Y_A7	
HACTIVE_A	-	-	HACT_A
VACTIVE_A	-	-	VACT_A
FIELDID_A	-	-	FID_A
in_mode_b	‘10’	‘11’	‘01’
in_mode_a	‘10’	‘11’	‘01’
main_a_sel	‘1’ or ‘0’	‘1’ or ‘0’	

(b) When ***in_data_map_mode*** = ‘01

Table 2-3. MDIN-380 Input Port Pin Mapping on 36-pin Input Mode (Continued)

Mode / Pin Name	Single YCbCr 4:4:4 36-bit	Single RGB 4:4:4 36-bit
D_IN0	Cr_A11	R_A11
D_IN1	Cr_A10	R_A10
D_IN2	Cr_A9	R_A9
D_IN3	Cr_A8	R_A8
D_IN4	Cr_A7	R_A7
D_IN5	Cr_A6	R_A6
D_IN6	Cr_A5	R_A5
D_IN7	Cr_A4	R_A4
D_IN8	Cr_A3	R_A3
D_IN9	Cr_A2	R_A2
D_IN10	Cr_A1	R_A1
D_IN11	Cr_A0	R_A0
D_IN12	Y_A11	G_A11
D_IN13	Y_A10	G_A10
D_IN14	Y_A9	G_A9
D_IN15	Y_A8	G_A8
D_IN16	Y_A7	G_A7
D_IN17	Y_A6	G_A6
D_IN18	Y_A5	G_A5
D_IN19	Y_A4	G_A4
D_IN20	Y_A3	G_A3
D_IN21	Y_A2	G_A2
D_IN22	Y_A1	G_A1
D_IN23	Y_A0	G_A0
D_IN24	Cb_A11	B_A11
D_IN25	Cb_A10	B_A10
D_IN26	Cb_A9	B_A9
D_IN27	Cb_A8	B_A8
D_IN28	Cb_A7	B_A7
D_IN29	Cb_A6	B_A6
D_IN30	Cb_A5	B_A5
D_IN31	Cb_A4	B_A4
D_IN32	Cb_A3	B_A3
D_IN33	Cb_A2	B_A2
D_IN34	Cb_A1	B_A1
D_IN35	Cb_A0	B_A0
HACTIVE_A	HACT_A	HACT_A
VACTIVE_A	VACT_A	VACT_A
FIELDID_A	FID_A	FID_A
in_mode_a	'00'	
main_a_sel	'1' (Only port A can be used.)	

(c) When ***in_data_map_mode*** = '10'

Table 2-3. MDIN-380 Input Port Pin Mapping on 36-pin Input Mode (Continued)

Mode / Pin Name	Single YCbCr 4:4:4 36-bit	Single RGB 4:4:4 36-bit
D_IN0	Cb_A0	B_A0
D_IN1	Cb_A1	B_A1
D_IN2	Cb_A2	B_A2
D_IN3	Cb_A3	B_A3
D_IN4	Cb_A4	B_A4
D_IN5	Cb_A5	B_A5
D_IN6	Cb_A6	B_A6
D_IN7	Cb_A7	B_A7
D_IN8	Cb_A8	B_A8
D_IN9	Cb_A9	B_A9
D_IN10	Cb_A10	B_A10
D_IN11	Cb_A11	B_A11
D_IN12	Y_A0	G_A0
D_IN13	Y_A1	G_A1
D_IN14	Y_A2	G_A2
D_IN15	Y_A3	G_A3
D_IN16	Y_A4	G_A4
D_IN17	Y_A5	G_A5
D_IN18	Y_A6	G_A6
D_IN19	Y_A7	G_A7
D_IN20	Y_A8	G_A8
D_IN21	Y_A9	G_A9
D_IN22	Y_A10	G_A10
D_IN23	Y_A11	G_A11
D_IN24	Cr_A0	R_A0
D_IN25	Cr_A1	R_A1
D_IN26	Cr_A2	R_A2
D_IN27	Cr_A3	R_A3
D_IN28	Cr_A4	R_A4
D_IN29	Cr_A5	R_A5
D_IN30	Cr_A6	R_A6
D_IN31	Cr_A7	R_A7
D_IN32	Cr_A8	R_A8
D_IN33	Cr_A9	R_A9
D_IN34	Cr_A10	R_A10
D_IN35	Cr_A11	R_A11
HACTIVE_A	HACT_A	HACT_A
VACTIVE_A	VACT_A	VACT_A
FIELDID_A	FID_A	FID_A
in_mode_a	'00'	
main_a_sel	'1' (Only port A can be used.)	

(d) When ***in_data_map_mode*** = '11'

Table 2-3. MDIN-380 Input Port Pin Mapping on 36-pin Input Mode (Continued)

2.2 Output Port Pin Mapping

2.2.1 MDIN-325 Output Port Pin Mapping

MDIN-325 has 24 configurable digital output pins as shown in the following table. The output port pin mapping is determined by ***digital_out_format*** and ***digital_8bit_out_en*** of the register ***out_control***. To enable the digital output of MDIN-320, ***out_data_mode*** of the register ***out_mux_ctrl*** should be set to 5 ('0101').

Mode / Pin Name	4:4:4 24-bit	YC 4:2:2 20-bit	YC 4:2:2 16-bit	YC-muxed 10-bit	YC-muxed 8-bit
DATA_OUT23	R7 / Cr7	C9	C7	-	-
DATA_OUT22	R6 / Cr6	C8	C6	-	-
DATA_OUT21	R5 / Cr5	C7	C5	-	-
DATA_OUT20	R4 / Cr4	C6	C4	-	-
DATA_OUT19	R3 / Cr3	C5	C3	-	-
DATA_OUT18	R2 / Cr2	C4	C2	-	-
DATA_OUT17	R1 / Cr1	C3	C1	-	-
DATA_OUT16	R0 / Cr0	C2	C0	-	-
DATA_OUT15	G7 / Y7	Y9	Y7	YC9	YC7
DATA_OUT14	G6 / Y6	Y8	Y6	YC8	YC6
DATA_OUT13	G5 / Y5	Y7	Y5	YC7	YC5
DATA_OUT12	G4 / Y4	Y6	Y4	YC6	YC4
DATA_OUT11	G3 / Y3	Y5	Y3	YC5	YC3
DATA_OUT10	G2 / Y2	Y4	Y2	YC4	YC2
DATA_OUT9	G1 / Y1	Y3	Y1	YC3	YC1
DATA_OUT8	G0 / Y0	Y2	Y0	YC2	YC0
DATA_OUT7	B7 / Cb7	C1	-	-	-
DATA_OUT6	B6 / Cb6	C0	-	-	-
DATA_OUT5	B5 / Cb5	-	-	-	-
DATA_OUT4	B4 / Cb4	-	-	-	-
DATA_OUT3	B3 / Cb3	Y1	-	YC1	-
DATA_OUT2	B2 / Cb2	Y0	-	YC0	-
DATA_OUT1	B1 / Cb1	-	-	-	-
DATA_OUT0	B0 / Cb0	-	-	-	-
digital_out_format	'00'	'01'	'01'	'10'	'10'
digital_8bit_out_en	don't care	'0'	'1'	'0'	'1'

Table 2-4. MDIN-325 Output Port Pin Mapping

2.2.2 MDIN-380 Output Port Pin Mapping

MDIN-380 has only 8 dedicated digital output pins **D_OUT(7:0)**, but up to 30 bits of digital video data can be output using additional multi-function pins – **D_IN(21:0)**, **HOST_AD(19:0)**. MDIN-380 has 15 pin configurations on the digital output port as shown in Table 2-5. The pin configuration is determined by ***out_data_mode*** of the register ***out_mux_ctrl***.

out_data_mode / Pin Name	0 (none)	1 (10-pin)	2 (20-pin)	3 (24-pin)	4 (30-pin)	5 (24-pin)	6 (30-pin)	7 (16-pin)
D_IN21	-	-	-	-	D29	-	-	-
D_IN20	-	-	-	-	D28	-	-	-
D_IN19	-	-	-	-	D27	-	-	-
D_IN18	-	-	-	-	D26	-	-	-
D_IN17	-	-	-	-	D25	-	-	-
D_IN16	-	-	-	-	D24	-	-	-
D_IN15	-	-	-	D23	D23	-	-	-
D_IN14	-	-	-	D22	D22	-	-	-
D_IN13	-	-	-	D21	D21	-	D29	-
D_IN12	-	-	-	D20	D20	-	D28	-
D_IN11	-	-	D19	D19	D19	D23	D27	-
D_IN10	-	-	D18	D18	D18	D22	D26	-
D_IN9	-	-	D17	D17	D17	D21	D25	-
D_IN8	-	-	D16	D16	D16	D20	D24	-
D_IN7	-	-	D15	D15	D15	D19	D23	-
D_IN6	-	-	D14	D14	D14	D18	D22	-
D_IN5	-	-	D13	D13	D13	D17	D21	-
D_IN4	-	-	D12	D12	D12	D16	D20	-
D_IN3	-	-	D11	D11	D11	D15	D19	-
D_IN2	-	-	D10	D10	D10	D14	D18	-
D_IN1	-	D9	D9	D9	D9	D13	D17	-
D_IN0	-	D8	D8	D8	D8	D12	D16	-
D_OUT7	-	D7	D7	D7	D7	D11	D15	D15
D_OUT6	-	D6	D6	D6	D6	D10	D14	D14
D_OUT5	-	D5	D5	D5	D5	D9	D13	D13
D_OUT4	-	D4	D4	D4	D4	D8	D12	D12
D_OUT3	-	D3	D3	D3	D3	D7	D11	D11
D_OUT2	-	D2	D2	D2	D2	D6	D10	D10
D_OUT1	-	D1	D1	D1	D1	D5	D9	D9
D_OUT0	-	D0	D0	D0	D0	D4	D8	D8
HOST_AD19	-	-	-	-	-	-	-	-
HOST_AD18	-	-	-	-	-	-	-	-
HOST_AD17	-	-	-	-	-	-	-	-
HOST_AD16	-	-	-	-	-	-	-	-
HOST_AD15	-	-	-	-	-	-	-	D7
HOST_AD14	-	-	-	-	-	-	-	D6
HOST_AD13	-	-	-	-	-	-	-	D5
HOST_AD12	-	-	-	-	-	-	-	D4
HOST_AD11	-	-	-	-	-	D3	-	D3
HOST_AD10	-	-	-	-	-	D2	-	D2
HOST_AD9	-	-	-	-	-	D1	-	D1
HOST_AD8	-	-	-	-	-	D0	-	D0
HOST_AD7	-	-	-	-	-	-	D7	-
HOST_AD6	-	-	-	-	-	-	D6	-
HOST_AD5	-	-	-	-	-	-	D5	-
HOST_AD4	-	-	-	-	-	-	D4	-
HOST_AD3	-	-	-	-	-	-	D3	-
HOST_AD2	-	-	-	-	-	-	D2	-
HOST_AD1	-	-	-	-	-	-	D1	-
HOST_AD0	-	-	-	-	-	-	D0	-

Table 2-5. MDIN-380 Output Port Pin Configurations

out_data_mode / Pin Name	8 (30-pin)	9 (8-pin)	10 (20-pin)	11 (30-pin)	12 (16-pin)	13 (16-pin)	14 (24-pin)	15 (16-pin)
D_IN21	-	-	-	-	-	-	-	-
D_IN20	-	-	-	-	-	-	-	-
D_IN19	-	-	-	-	-	-	-	-
D_IN18	-	-	-	-	-	-	-	-
D_IN17	-	-	-	-	-	-	-	-
D_IN16	-	-	-	-	-	-	-	-
D_IN15	D29	-	-	-	-	-	-	-
D_IN14	D28	-	-	-	-	-	-	-
D_IN13	D27	-	-	-	-	-	-	-
D_IN12	D26	-	-	-	-	-	-	-
D_IN11	D25	-	-	-	-	-	-	-
D_IN10	D24	-	-	-	-	-	-	-
D_IN9	D23	-	-	-	-	-	-	-
D_IN8	D22	-	-	-	-	-	-	-
D_IN7	D21	-	-	-	-	-	-	D15
D_IN6	D20	-	-	-	-	-	-	D14
D_IN5	D19	-	D19	D29	-	-	-	D13
D_IN4	D18	-	D18	D28	-	-	-	D12
D_IN3	D17	-	D17	D27	-	-	-	D11
D_IN2	D16	-	D16	D26	-	-	-	D10
D_IN1	D15	-	D15	D25	-	-	-	D9
D_IN0	D14	-	D14	D24	-	-	-	D8
D_OUT7	D13	D7	D13	D23	D15	D15	D23	D7
D_OUT6	D12	D6	D12	D22	D14	D14	D22	D6
D_OUT5	D11	D5	D11	D21	D13	D13	D21	D5
D_OUT4	D10	D4	D10	D20	D12	D12	D20	D4
D_OUT3	D9	D3	D9	D19	D11	D11	D19	D3
D_OUT2	D8	D2	D8	D18	D10	D10	D18	D2
D_OUT1	D7	D1	D7	D17	D9	D9	D17	D1
D_OUT0	D6	D0	D6	D16	D8	D8	D16	D0
HOST_AD19	-	-	-	-	D7	-	-	-
HOST_AD18	-	-	-	-	D6	-	-	-
HOST_AD17	-	-	-	-	D5	-	-	-
HOST_AD16	-	-	-	-	D4	-	-	-
HOST_AD15	-	-	-	D15	D3	-	D15	-
HOST_AD14	-	-	-	D14	D2	-	D14	-
HOST_AD13	D5	-	D5	D13	D1	-	D13	-
HOST_AD12	D4	-	D4	D12	D0	-	D12	-
HOST_AD11	D3	-	D3	D11	-	-	D11	-
HOST_AD10	D2	-	D2	D10	-	-	D10	-
HOST_AD9	D1	-	D1	D9	-	-	D9	-
HOST_AD8	D0	-	D0	D8	-	-	D8	-
HOST_AD7	-	-	-	D7	-	D7	D7	-
HOST_AD6	-	-	-	D6	-	D6	D6	-
HOST_AD5	-	-	-	D5	-	D5	D5	-
HOST_AD4	-	-	-	D4	-	D4	D4	-
HOST_AD3	-	-	-	D3	-	D3	D3	-
HOST_AD2	-	-	-	D2	-	D2	D2	-
HOST_AD1	-	-	-	D1	-	D1	D1	-
HOST_AD0	-	-	-	D0	-	D0	D0	-

Table 2-5. MDIN-380 Output Port Pin Configurations (Continued)

When the pins are used as digital output pins, other functions of the pins cannot be used. It is recommended to select a pin configuration whose pin count is just matched to the required number of digital output pins so that the other pins cane be used as other fuctions. When no digital output pins are used, ***out_data_mode*** should be set to 0 ('0000') to use the pins as other functions.
In each pin configuration, the mappings of bit names D(29:0) to video data are determined by ***digital_out_format*** of the register ***out_control*** as shown in Table 2-6.

Mode / Bit Name	30-pin Mode			24-pin Mode		
	4:4:4 30-bit	YC 4:2:2 20-bit	YC-muxed 10-bit	4:4:4 24-bit	YC 4:2:2 20-bit	YC-muxed 10-bit
D29	R9 / Cr9	C9	-			
D28	R8 / Cr8	C8	-			
D27	R7 / Cr7	C7	-			
D26	R6 / Cr6	C6	-			
D25	R5 / Cr5	C5	-			
D24	R4 / Cr4	C4	-			
D23	R3 / Cr3	C3	-	R7 / Cr7	C9	-
D22	R2 / Cr2	C2	-	R6 / Cr6	C8	-
D21	R1 / Cr1	-	-	R5 / Cr5	C7	-
D20	R0 / Cr0	-	-	R4 / Cr4	C6	-
D19	G9 / Y9	Y9	YC9	R3 / Cr3	C5	-
D18	G8 / Y8	Y8	YC8	R2 / Cr2	C4	-
D17	G7 / Y7	Y7	YC7	R1 / Cr1	C3	-
D16	G6 / Y6	Y6	YC6	R0 / Cr0	C2	-
D15	G5 / Y5	Y5	YC5	G7 / Y7	Y9	YC9
D14	G4 / Y4	Y4	YC4	G6 / Y6	Y8	YC8
D13	G3 / Y3	Y3	YC3	G5 / Y5	Y7	YC7
D12	G2 / Y2	Y2	YC2	G4 / Y4	Y6	YC6
D11	G1 / Y1	-	-	G3 / Y3	Y5	YC5
D10	G0 / Y0	-	-	G2 / Y2	Y4	YC4
D9	B9 / Cb9	C1	YC1	G1 / Y1	Y3	YC3
D8	B8 / Cb8	C0	YC0	G0 / Y0	Y2	YC2
D7	B7 / Cb7	-	-	B7 / Cb7	C1	-
D6	B6 / Cb6	-	-	B6 / Cb6	C0	-
D5	B5 / Cb5	Y1	-	B5 / Cb5	-	-
D4	B4 / Cb4	Y0	-	B4 / Cb4	-	-
D3	B3 / Cb3	-		B3 / Cb3	Y1	YC1
D2	B2 / Cb2	-		B2 / Cb2	Y0	YC0
D1	B1 / Cb1	-	-	B1 / Cb1	-	-
D0	B0 / Cb0	-	-	B0 / Cb0	-	-
digital_out_format	'00'	'01'	'10'	'00'	'01'	'10'
digital_8bit_out_en	'0'	'0'	'0'	don't care	'0'	'0'

Table 2-6. MDIN-380 Output Port Bit Mapping

Mode / Bit Name	20-pin Mode		10-pin Mode	16-pin Mode		8-pin Mode
	YC 4:2:2 20-bit	YC-muxed 10-bit	YC-muxed 10-bit	YC 4:2:2 16-bit	YC-muxed 8-bit	YC-muxed 8-bit
D19	Y9	YC9				
D18	Y8	YC8				
D17	Y7	YC7				
D16	Y6	YC6				
D15	Y5	YC5		Y7	YC7	
D14	Y4	YC4		Y6	YC6	
D13	Y3	YC3		Y5	YC5	
D12	Y2	YC2		Y4	YC4	
D11	Y1	YC1		Y3	YC3	
D10	Y0	YC0		Y2	YC2	
D9	C9	-	YC9	Y1	YC1	
D8	C8	-	YC8	Y0	YC0	
D7	C7	-	YC7	C7	-	YC7
D6	C6	-	YC6	C6	-	YC6
D5	C5	-	YC5	C5	-	YC5
D4	C4	-	YC4	C4	-	YC4
D3	C3	-	YC3	C3	-	YC3
D2	C2	-	YC2	C2	-	YC2
D1	C1	-	YC1	C1	-	YC1
D0	C0	-	YC0	C0	-	YC0
digital_out_format	‘01’	‘10’	‘10’	‘01’	‘10’	‘10’
digital_8bit_out_en	‘0’	‘0’	‘0’	don’t care	don’t care	don’t care

Table 2-6. MDIN-380 Output Port Bit Mapping (Continued)

2.3 Active Window Size and Position of Input Video

MDIN-3xx can be programmed to define the size and position of the received video by register setttings. It is necessary to adjust active window size and position of input video when horizontal and vertical timing signals received through input port A are sync signal, not active signal. In this case, the function of adjusting size and position that MDIN-3xx provides is very helpful to extract the desired active video data without glue logic within a system. Figure 2-1 shows how this function works. The horizontal size is determined by ***in_size_h*** register, and the result is the minimum value between the register value and duration of the received horizontal sync signal. The vertical size is determined by ***in_size_v*** register, and the result is the minimum value between the register value and duration of the received vertical sync signal. The offset of horizontal position is determined by ***in_hact_offset*** register. As a result, active window is shifted by the register value. The offset of vertical position is also controlled as that of horizontal position by ***in_vact_offset*** register.

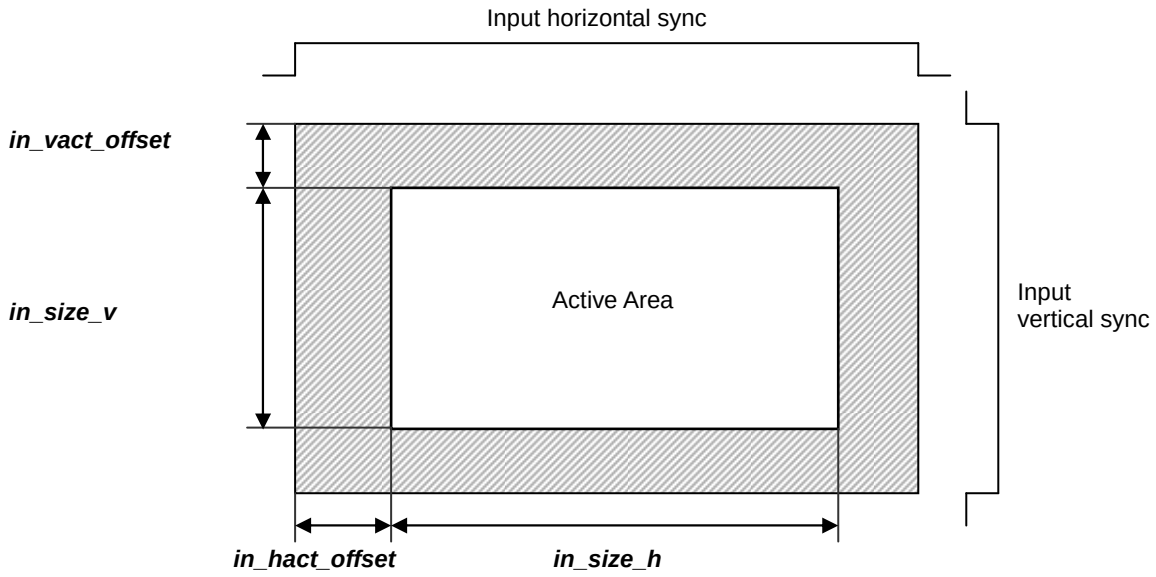


Figure 2-1. Active Window Size and Position of Input Video

2.4 Input Color Space Conversion

MDIN-3xx has one set of color space conversion in video input block, which can be used to convert RGB input video to YCbCr format for internal 4:2:2 processing. This color space conversion logic is implemented as 3x3 matrix scheme to provide full flexibility. Functional block diagram of 3x3 matrix color space conversion logic in input block is shown in Figure 2-2. For proper color space conversion, there are nine coefficient registers for 3x3 matrix calculation and one control register for preprocessing of video data value. The coefficient registers of ***csc_f_coef0***, ***csc_f_coef1***, …, ***csc_f_coef7*** and ***csc_f_coef8*** in the input block are user programmable. All input components and output components can be added or substracted by setting ***csc_f_offset_y***, ***csc_f_offset_cb, csc_f_offset_cr, csc_f_offset_g, csc_f_offset_b*** and ***csc_f_offset_r*** registers as user defined values. And it is possible to make the input and output value clipped to a certain range. One way is clipping for Cb, Cr, B, R to be ranging from 0 to 1023 and the other is from -512 to 511. User can select each clipping mode for each Cb, Cr, B, R. And Y and G signals are clipped to be ranging from 0 to 1023. The CSC can be bypassed by setting ***csc_f_bypass*** bit of ***csc_f_control*** register. All the coefficient and offset registers are 12-bit 2's complement values, and 512 corresponds to 1.0 (unity).

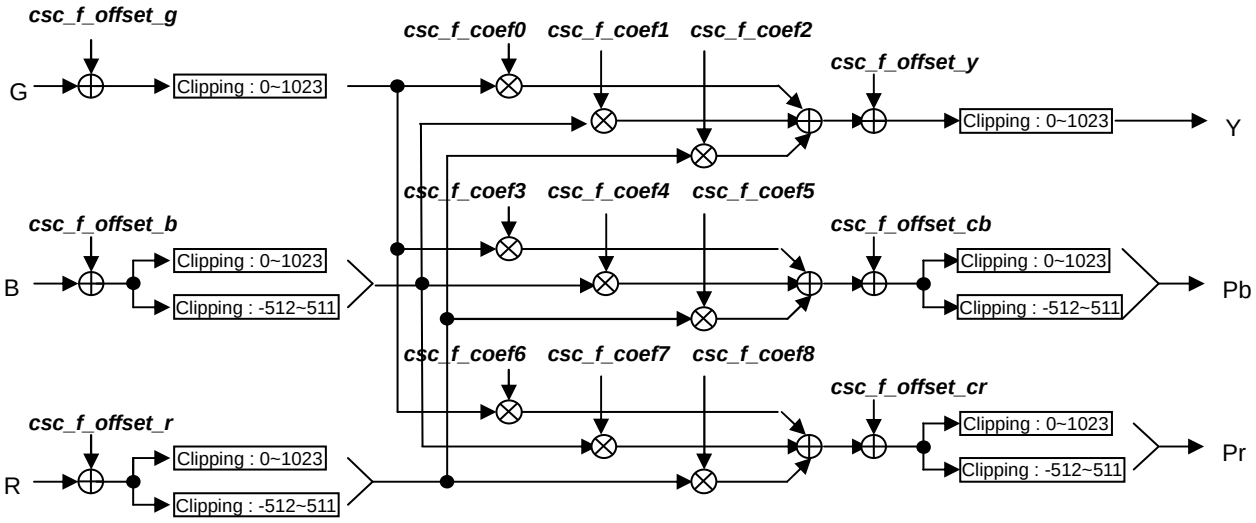


Figure 2-2. Functional Block Diagram of Input Color Space Conversion

The register values in the following example can be used to convert RGB video to the ITU-R BT.601 YCbCr format. The RGB and YCbCr data are assumed a range of 0 to 1023. If the range is 64 to 940, the coefficient and offset registers should be adjusted properly.

$$Y = 0.587\times G + 0.114\times B + 0.299\times R$$
$$Cb = -0.339\times G + 0.511\times B - 0.172\times R$$
$$Cr = -0.428\times G - 0.083\times B + 0.511\times R$$

csc_f_coef0 = 0x12D, ***csc_f_coef1*** = 0x03A, ***csc_f_coef2*** = 0x099
csc_f_coef3 = 0xF52, ***csc_f_coef4*** = 0x106, ***csc_f_coef5*** = 0xFA8
csc_f_coef6 = 0xF25, ***csc_f_coef7*** = 0xFD6, ***csc_f_coef8*** = 0x106
csc_f_offset_g = 0x000, ***csc_f_offset_b*** = 0x000, ***csc_f_offset_r*** = 0x000
csc_f_offset_y = 0x000, ***csc_f_offset_cb*** = 0x200, ***csc_f_offset_cr*** = 0x200
csc_f_control = 0x1F8

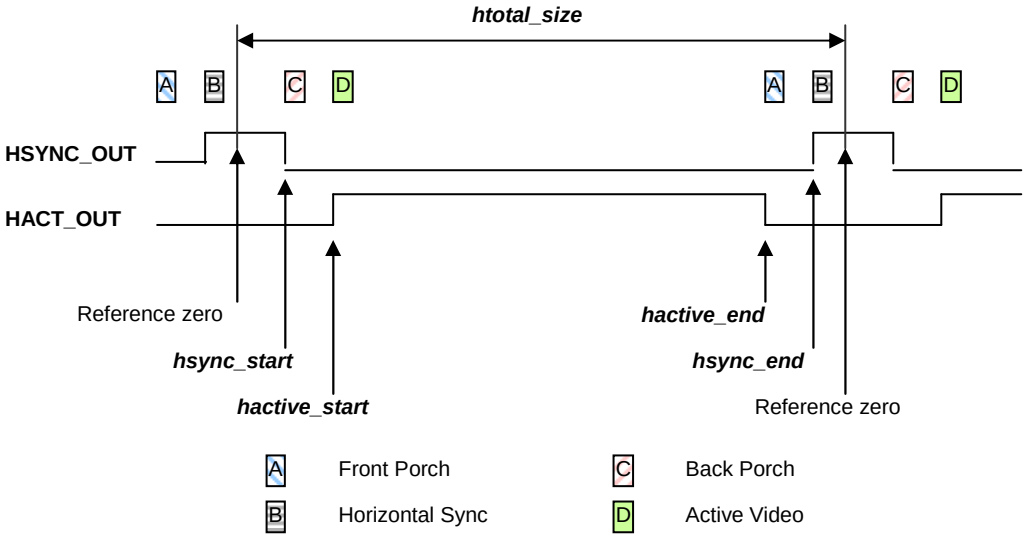
2.5 Auto Detection and Compensation of Input Video Sync

MDIN-3xx provides input auto detection capability. Auto detection of the follwing parameters is supported - horizontal sync period, horizontal size or the number of horizontal pixels for both the high and low period of horizontal sync, vertical size or the number of vertical lines for both high and low period of vertical sync, and scan type. Auto detection on the input port B is not necessary since it always receives video data with embedded timing references.
MDIN-3xx also detects the existence of video sync signal and the change in the horizontal and vertical sync period. Auto detection of sync existence or sync change is always performed and the other auto detection features are performed by setting ***in_detect_en*** register. The results of auto detection are stored in input status registers (***in_sync_lost, in_hsync, in_scan_i, in_hsync_high, in_hsync_low, in_vsync_high, in_vsync_low***) and they can be read and used by an external host processor for the detection of exact input video format.
MDIN-3xx supports composite sync input. It can extract the hsync and vsync from the composite sync signal which can be obtained by slicing the bi-level or tri-level sync tips of the input video signal. MDIN-3xx can receive a composite sync signal through **HACTIVE_A** pin by setting ***in_sync_ctrl*** register.
And MDIN-3xx can extract the sync information even though the input sync signals are in non-standard format. When the timing of vsync-transition to hsync is different from the standard timing it can be compensated by setting ***vsync_forced_rising*** register**.** In this case, the horizontal total size of input signal needs to be specified in

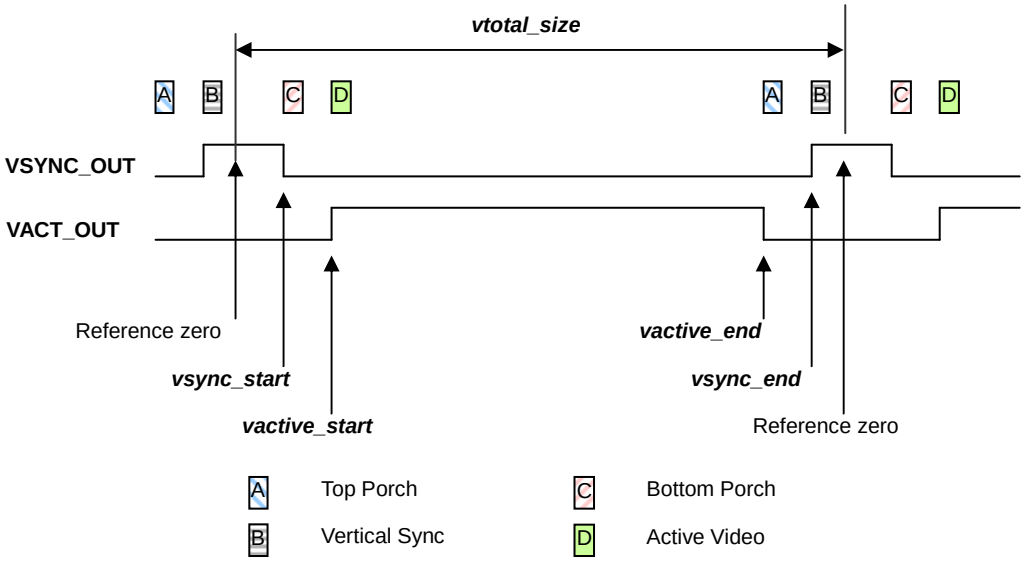
in_fieldid_ctrl register. And as prevention against unstable vsync-rising, the internal vsync signal can be forced to rise at any position by setting ***vsync_forced_rising*** register**.** ***in_sync_sel*** register determines which input sync will be selected as the auto detection input between the original and compensated input sync signals**.**

2.6 Output Sync Generation

MDIN-3xx has two kinds of sync control mode. The first one is free-run mode. This is the case the output and input video is not synchronized and frame rate conversion may be required. The second mode is frame-lock mode. This is the case the output vertical sync is locked to input vertical sync where the frame rate of input and output video is exactly same.
User can make the output video sync locked to the input video on free-run mode by setting the PLL parameters and sync control registers properly. The sync control mode is selected by ***sync_ctrl*** register. The timings of output sync signals are fully user-programmable by setting ***htotal_size, hactive_start, hactive_end, hsync_start, hsync_end, vactive_start, vactive_end, vsync_start, vsync_end*** registers. The polarities of all output sync signals are programmable and the composite sync is also supported. Figure 2-3 shows the details of output sync generation.



(a) Horizontal Output Sync Generation



(b) Vertical Output Sync Generation

Figure 2-3. Output Sync Generation

2.7 Deinterlacing

MDIN-3xx provides deinterlacing function for interlaced input video. It supports 3-D motion adaptive deinterlacing of standard and high deifinition videos such as 480i, 576i or 1080i. The supported maximum horizontal size is 2047 pixels for 8-bit pixel precision mode and 1536 pixels for 10-bit pixel precision mode. The supported vertical size is up to 2047 lines.

2.7.1 Motion Adaptive Deinterlacing

MDIN-3xx uses 6 fields of frame memory to get motion information and performs motion-adaptive interpolation. MDIN-3xx supports three kinds of deinterlacing modes - intra-only mode, inter-only mode and adaptive mode. The intra-only mode of deinterlacing uses only current field to calculate and generate the deinterlaced pixel output (known as bob mode). The inter-only mode of deinterlacing always generates the deinterlaced pixel output by inter-interpolation using current field and previous field (known as weave mode). The adaptive mode performs deinterlacing using weighted combination of intra-field interpolation and inter-field interpolation on a per pixel basis. MDIN-3xx detects motion information using 6 fields so that the motion detection errors will be minimized.

2.7.2 Edge Preserving

MDIN-3xx provides high quality deinterlacing with detecting and preserving edges of various angles when the intra-field interpolation should be used. If a simple deinterlacing algorithm that does not support edge preservation is used, there should be much jagged edges on the output video after deinterlacing. MDIN-3xx uses a proprietary algorithm in detecting the existence and direction of edges for interlaced input video, and it performs deinterlacing according to the edge detection results so that a smooth and graceful deinterlaced result will be provided. MDIN-3xx also detects the frequency characteristics of the video area to deinterlace and uses them to determine the edge direction. So the artifacts for high frequency area in the video can be minimized.

2.7.3 Fast Motion Detect and Handling

MDIN-3xx supports graceful deinterlacing for interlaced input video with fast motion. If a deinterlacing algorithm that supports motion-adaptive deinterlacing without fast motion support, there must be artifacts of undesired inter-field interpolation. MDIN-3xx detects fast motion using 6 fields of frame memory and the motion information of the recent 30 fields and perfoms appropriate deinterlacing algorithm to remove those artifacts.

2.7.4 Motion Boundary Preserving

There can be a still object on the moving area in the scene. The boundary of this object is hard to be preserved due to the surrounding motions. MDIN-3xx can detect the motion boundary and preserve it even though the object has one line size in vertical direction. For this operation MDIN-3xx uses motion information of recent 30 fields.

2.7.5 Film Mode and Bad Edit Detection

MDIN-3xx supports film mode detection and proper processing to deinterlace the video from film source without artifacts. In a normally interlaced video, successive fields are captured at different time instances so that they have different pictures if the objects in the video are moving. But when film of 24Hz or 30Hz frame rate is converted to an interlaced video, 2 or 3 successive fields may be captured from one film frame due to 3:2 or 2:2 pull-down processing of telecine machine. So the original film frames can be recovered by a proper inverse pull-down process.
If a deinterlacing algorithm does not support proper film mode for the interlaced video which is converted from film by 3:2 or 2:2 pull-down process, there can be jagging artifacts on edges after deinterlacing when there are motions. There can be flikers especially in high frequency area. MDIN-3xx does examine whether the source of the interlaced input video is film or not, and performs proper deinterlacing process according to the detection result. MDIN-3xx can also detect bad editing scene for 3:2 pull-down or 2:2 pull-down mode and does process properly to eliminate the artifact due to bad edit. MDIN-3xx can also properly handle subtitles in film mode so that tearing or combing artifacts in subtitle area do not occur even when there are moving subtitles in the film mode. And MDIN-3xx does not generate artifacts even when the still subtitles appear and disappear in the film mode.

2.7.6 Still Mode

As well as pixel-by-pixel basis deinterlacing, MDIN-3xx provides field basis still mode. If there are little motions in the entire picture, the deinterlace process can go to the still mode and the internal control parameters vary adaptively. This makes still images look crisp.

2.7.7 Inter-only Deinterlacing Area

MDIN-3xx supports inter-only mode deinterlacing on specified areas while adaptive mode deinterlacing on other areas. This feature is useful to display translucent OSD graphics over moving background video without flickering.
Up to 5 rectangle areas can be specified by register. A rectangle is specified by the registers ***inter_areaX_start_h*** and ***inter_areaX_start_v***, ***inter_areaX_end_h*** and ***inter_areaX_end_v***. (where ***X*** = 0, 1, 2, .. 4) Each inter-only area can be enabled or disabled by its own enable bit ***inter_areaX_en***.
The registers for inter-only area can be written or read using ***inter_area_data***, ***inter_area_wen*** and ***inter_area_addr***. Their address range is 0x00 to 0x20. Refer to 3.14.1 in MDIN-3xx Register Manual for the description of the inter-only area registers.

2.7.8 10-bit processing

MDIN-3xx supports 10-bit precision video input and output, and processes video internally with maximum 12-bit precision. This 10-bit precision processing works only when the horizontal size of input video is less than or equal to 1536 pixels per line. 8-bit precision should be used if the horizontal size is greater than 1536.
The auxiliary video path supports 8-bit precision processing only.

2.8 Video Scaling – Format Conversion

MDIN-3xx supports both up-conversion and down-conversion. MDIN-3xx's main format converter consists of a horizontal format converter and a vertical format converter. The main horizontal format converter uses a polyphase interpolation filter with 8 taps and 64 phases, and the main vertical format converter uses a polyphase interpolation filter with 4 taps and 64 phases. In chrominace conversion, horizontal and vertical 4-tap polyphase interpolation filters with 64 phases are used. The coeffients of polyphase filters are programmable and they can be used to control the frequency response of the interpolation filter.
The vertical format converter can do up-conversion and down-conversion, but the horizontal format converter does up-conversion only. The scaling ratio can be arbitrary and it is determined by specifying source size and destination size. The scaling ratio has no upper limit, but the lower limit of the scaling ratio in vertical down-conversion might be constrained by the SDRAM bandwidth.

MDIN-3xx has a front format converter as well as the main format converter to scale down the input video when the horizontal size of input video exceeds the maximum size which MDIN-3xx can process internally. The maximum horizontal size is 2048 pixels per line on 8-bit pixel precision mode and 1536 pixels on 10-bit precision mode. The front format converter uses linear interpolation filters with 2 taps in both luminance and chrominance conversions. To prevent aliasing in down-conversion, the front noise reduction filter can be used as an anti-aliasing filter. It has 15 taps for liminance and 7 taps for chrominace with programmable coefficients. The front format converter does not support horizontal up-conversion or any vertical format conversion.
The front format converter does not support vertical scaling and the input video is written to the frame buffer without vertical down-conversion. The maximum vertical size in the frame buffer is limited by the capacity and bandwidth of the frame buffer. If the vertical size exceeds the capacity or the bandwidth of the MDIN-3xx device, the video can be down-scaled horizontally to less than the output horizontal size by the front format converter and up-scaled to the output horizontal size by the main format converter with sacrificing some horizontal resolutions.

The format conversion of the main video path can be performed in two-step procedure as explained in Figure 2-4. The first step which is performed by the front format converter is the down-conversion of the input video in horizontal direction when the video needs to be scaled down. This can be bypassed when it is not required. Then the result is written into the frame memory. The second step which is performed by the main format converter is to convert the desired part of the video in the frame memory to the destination area.
The vertical size of source video and destination video can be specified in 1-line step but the horizontal size of source video and destination video can be specified with some limitations depending on the pixel precision. The position of source video and destination video is arbitrarily programmable in 2-pixel step horizontally and in 1-line step vertically.

Figure 2-4 shows the concept of the format conversion process and the actual registers might be different from those in the figure. ***src_size_v2*** and ***dest_size_v2*** are the same size and the value is specified by the register ***mem_size_v*** which should be the same as the vertical size of the selected input video (***in_size_v_a*** or ***in_size_v_b***), since the front format converter does not support vertical format conversion. ***dest_size_h2*** must be less than or equal to ***src_size_h2*** and ***dest_size_h_cr*** must be greater than or equal to ***src_size_h_cr***, since the front format converter supports only down-scaling and the main format converter supports only up-scaling in horizontal direction. The source area of the main format converter is specified by the register ***src_size_h*** and ***src_size_v*** but the scaling factors are defined by the Main Format Converter Size registers. Assuming uniform format conversion, the horizontal scaling factor is specified by the ratio of ***src_size_h_cr*** to ***dest_size_h_cr*** and the vertical scaling factor is specified by the ratio of ***src_size_v_cr*** to ***dest_size_v_cr***. The main window size is specified by the registers ***win_size_h*** and ***win_size_v***, and its size is specified by the registers ***win_posi_h*** and ***win_posi_v***. Normally the main window size is the same as the destination size of the main format converter which is specified by ***dest_size_h_cr*** and ***dest_size_v_cr***, but it can be specified differently from the destination size to hide garbage data on video edges.

The main format converter supports non-uniform format conversion as well as uniform format conversion as shown in Figure 2-5. On the non-uniform format conversion mode, each window is divided by two regions – central region and outside region. Each region can be scaled with its own scaling factor. This is useful for panorama mode. On the uniform format conversion mode, only the central region is available and the whole window is scaled with a single scaling factor.

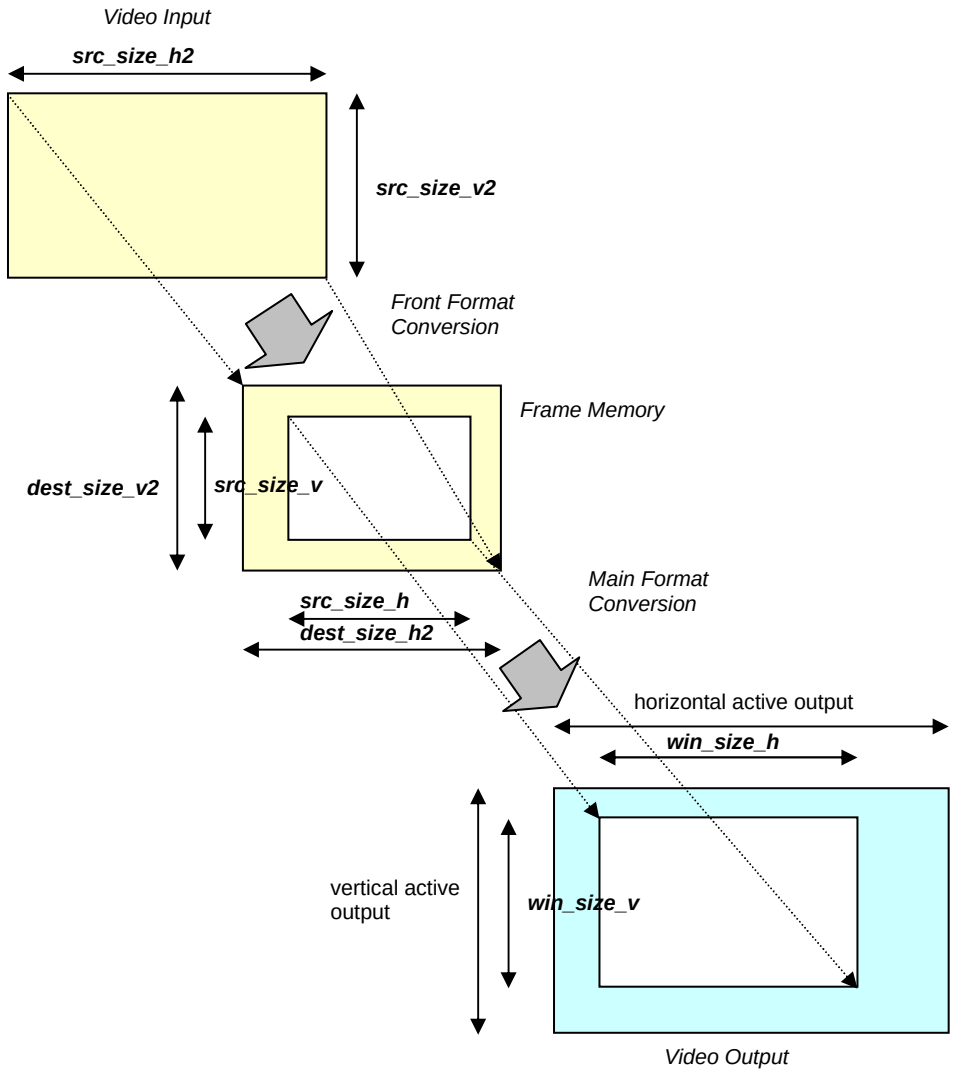
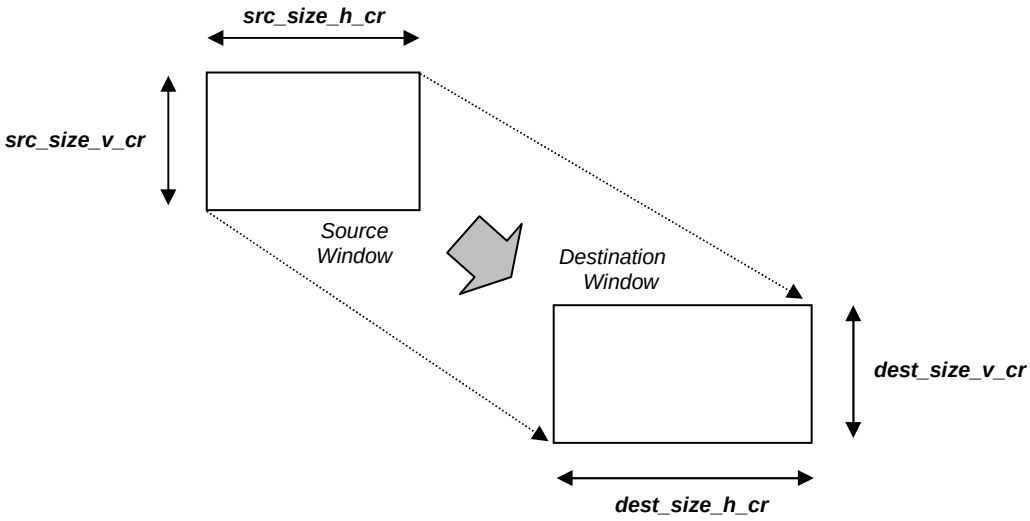
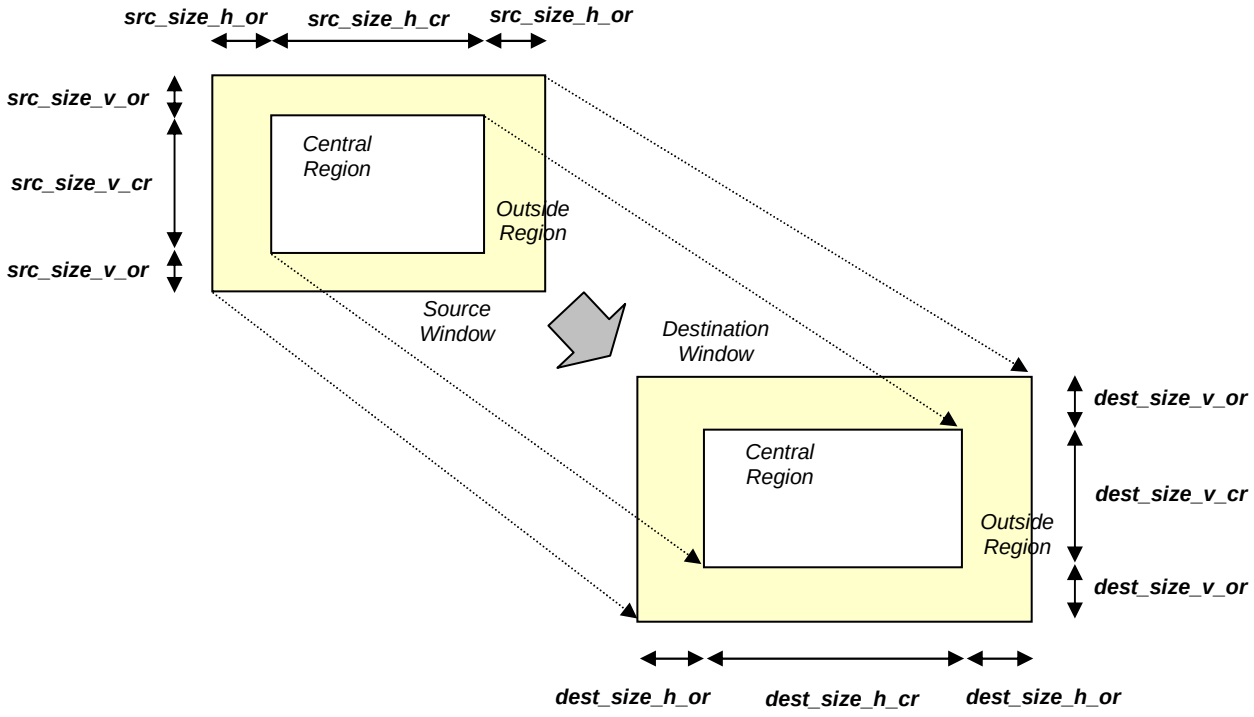


Figure 2-4. Scaling for Main Video



(a) Uniform Format Conversion



(b) Non-uniform Format Conversion

Figure 2-5. Uniform and Non-uniform Format Conversion

2.8.1 MFC Interpolation Filter

MDIN-3xx uses high order interpolation filters to resample pixels on the sampling points between input pixels for scaling in the main format converter. These interpolation filters are implemented using the polyphase filter architecture.

2.8.1.1 Polyphase Filter

When the original filter coefficients are *h(k)*, k = 0,1,2,..,N, the convolution is expressed as follows :

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

Assuming N = KM, this filter can be decomposed to M subphase filters with K-tap order :

$$y(n) = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} h(kM+m)x(n-kM-m)$$

M is the number of subphase filters and K is the filter tap order of each subphase filter. The following table show examples for K=8 and 4. Each row represents a subphase filter and each subphase filter acts as an 8-tap or 4-tap FIR filter. In MDIN-3xx, the interpolation filter for horizontal luminance conversion has 8 taps and other interpolation filters have 4 taps. M can be 32 or 64.

Tap / Phase	0	1	2	3	4	5	6	7
0	*h(0)*	*h(M)*	*h(2M)*	*h(3M)*	*h(4M)*	*h(5M)*	*h(6M)*	*h(7M)*
1	*h(1)*	*h(M+1)*	*h(2M+1)*	*h(3M+1)*	*h(4M+1)*	*h(5M+1)*	*h(6M+1)*	*h(7M+1)*
2	*h(2)*	*h(M+2)*	*h(2M+2)*	*h(3M+2)*	*h(4M+2)*	*h(5M+2)*	*h(6M+2)*	*h(7M+2)*
3	*h(3)*	*h(M+3)*	*h(2M+3)*	*h(3M+3)*	*h(4M+3)*	*h(5M+3)*	*h(6M+3)*	*h(7M+3)*
…								
M-1	*h(M-1)*	*h(2M-1)*	*h(3M-1)*	*h(4M-1)*	*h(5M-1)*	*h(6M-1)*	*h(7M-1)*	*h(8M-1)*

(a) K = 8

Tap / Phase	0	1	2	3
0	*h(0)*	*h(M)*	*h(2M)*	*h(3M)*
1	*h(1)*	*h(M+1)*	*h(2M+1)*	*h(3M+1)*
2	*h(2)*	*h(M+2)*	*h(2M+2)*	*h(3M+2)*
3	*h(3)*	*h(M+3)*	*h(2M+3)*	*h(3M+3)*
…				
M-1	*h(M-1)*	*h(2M-1)*	*h(3M-1)*	*h(4M-1)*

(b) K = 4

Table 2-7. Polyphase coefficients for K = 8 and 4

For linear phase response, the coefficients of the original N-tap filter should be symmetric, i.e., *h(k) = h(N-1-k)*. So only half taps of coefficients are necessary for the filter operation. In MDIN-3xx, the first half taps in the above table are used to program the interpolation filters, i.e., tap 0 to 3 for K=8 and tap 0 to 1 for K=4. They can be downloaded using the MFC

coefficient data registers ***mfc_itp_coef_data0*** to ***mfc_itp_coef_data3*** with the relationship between the MFC coefficient data registers and the subphase filter coefficients shown in the following table.

MFC coefficient data register	K=8	K=4
mfc_itp_coef_data0	*h(3M+m)*	*h(M+m)*
mfc_itp_coef_data1	*h(2M+m)*	*h(m)*
mfc_itp_coef_data2	*h(M+m)*	*not used*
mfc_itp_coef_data3	*h(m)*	*not used*

Table 2-8. Relationship between MFC coefficient data register and subphase filter coefficient
(where m = 0,1,2, … , M-1 = ***mfc_itp_coef_addr***)

The phase indicator m is used as the address of a coefficient entry for the subphase filter of phase m.
If the filter taps are shorter than the tap order of an interpolation filter, the coefficients from the center tap should be positioned to the data register from ***mfc_itp_coef_data0*** and the unused taps should be filled with zeros.

2.8.1.2 Interpolation Function

The interpolation filter coefficients are generated from interpolation function. In case of K=4, cubic spline functions are popularly used. For higher order filters, sinc functions such as Lanczos function are used.

- Cubic Spline Function

A generalized cubic spline function can be expressed as follows :

$$h(s) = \begin{cases} (2 - \frac{3}{2}B - C)|s|^3 \\ +(-3 + 2B + C)|s|^2 \\ +(1 - \frac{1}{3}B), & |s| \le 1 \\ \\ (-\frac{1}{6}B - C)|s|^3 \\ +(B + 5C)|s|^2 \\ +(-2B - 8C)|s| \\ +(\frac{4}{3}B + 4C), & 1 < |s| \le 2 \\ \\ 0, & \text{otherwise} \end{cases}$$

The polyphase filter coefficients are generated as follows :

h(kM+m) = h(s), where s = (kM+m)/M

where k=0 for ***mfc_itp_coef_data1***, k=1 for ***mfc_itp_coef_data0***
and m=0,1,2, …, M-1 for each subphase filter

The characteristics of the cubic spline function are determined by controlling the parameter B and C between 0 and 1. The following spline functions are proposed in various papers :

- B-spline : B = 1, C = 0
- Cardinal spline : B = 0, C = 1(most sharp), 3/4, 1/2
- Mitchell & Netravali : 2C + B = 1 (most sharp when B=0)

- Lanczos Function

Lanczos function is a sinc-windowed sinc function which is expressed as follows :

$$L(x) = \begin{cases} \operatorname{sinc}(x)\operatorname{sinc}\left(\frac{x}{a}\right) & -a < x < a, x \neq 0 \\ 1 & x = 0 \\ 0 & \text{otherwise} \end{cases}$$

where α is a positive integer, 3 for 6-tap filter and 4 for 8-tap filter.

Assuming 8-tap filter (K=8), the polyphase filter coefficients are generated as follows :

h(kM+m) = L(x), where x = (kM+m)/M and $\alpha = 4$

where k=0 for ***mfc_itp_coef_data3***, k=1 for ***mfc_itp_coef_data2***, …, k=3 for ***mfc_itp_coef_data0***
and m=0,1,2, …, M-1 for each subphase filter

2.8.1.3 Downloading Filter Coefficients

The interpolation filter coefficients must be downloaded as the following procedure :

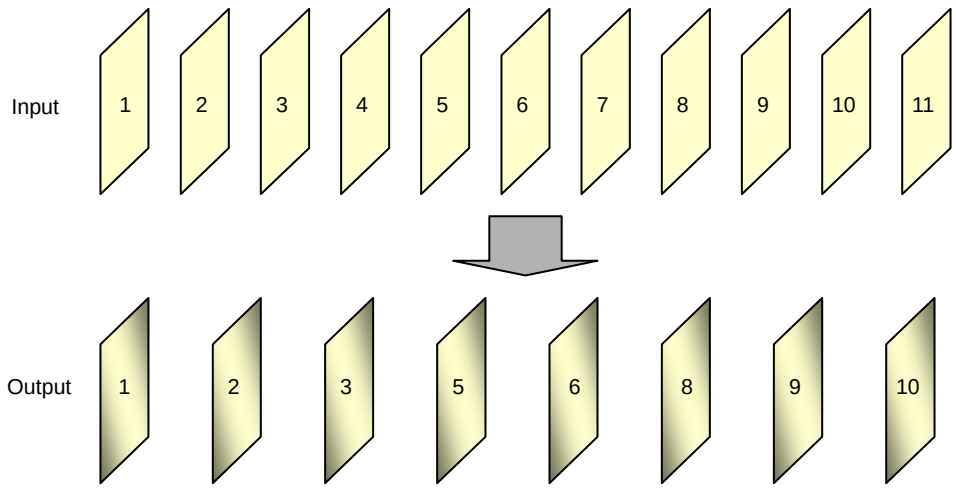
(1) Write coefficient data of a subphase filter to ***mfc_itp_coef_data#*** (# = 0,1,2,3)
(2) Write command data to ***mfc_itp_coef_cmd***
mfc_itp_coef_sel = filter selection (HY='000', HC='001', VY='010', VC='011, ALL='100')
mfc_itp_coef_wen = '1' for writing coefficients
mfc_itp_coef_addr = address of subphase filter (0,1,2, …, M-1)
(3) Repeat (1) & (2) until all the phases are downloaded

2.8.1.4 Dual Filter Mode

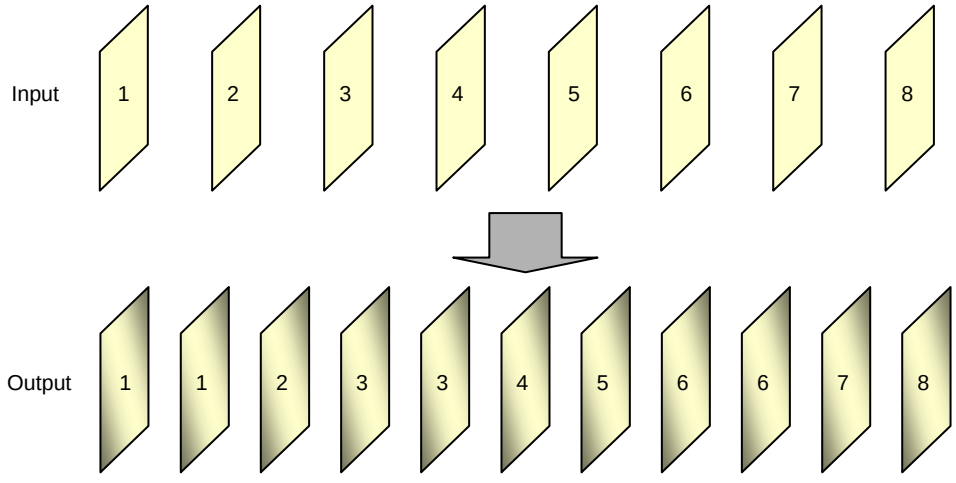
Each interpolation filter has 64-phase coefficients. On the single filter mode, the whole 64-phase coefficients are used as a single filter. On the dual filter mode, the 64 phases are divided into two 32-phase filters and two filters can be programmed to have different coefficient sets. Each filter can be selected by the register ***mfc_itp_mode_XX***. (***XX*** = ***hy***, ***hc***, ***vy***, ***vc***) When this register is set to '01', the dual filter mode can be used as 2-D noise reduction filtering. In this case, the lower filter has normal coefficents and the upper filter has low-pass coefficients with narrow bandwidth.

2.9 Frame Rate Conversion

MDIN-3xx supports frame rate conversion with arbitrary frame rate ratio between 1/31 to 31 times for progressive video input. Frame memory has two frame buffers for progressive input video and the stored frames are repeated or the writing of the incoming frame is skipped when generating output frames according to output sync as shown in Figure 2-6. Output sync generation has two modes - lock mode and free-run mode. When frame rate conversion is necessary, free-run mode should be selected by properly setting ***sync_ctrl*** register.
For interlaced input video, frame rate up-conversion can be performed without any problem. If some fields are skipped by frame rate down-conversion, however, deinterlacing cannot be performed gracefully because one or more reference fields which are required in 3-D deinterlacing might be missing. MDIN-3xx's deinterlacer can handle the skipping of a field without artifacts but it should be avoided to maintain a good picture quality.



(a) Down-conversion



(b) Up-conversion

Figure 2-6. Frame rate conversion

2.10 Frame Buffer Memory

MDIN-3xx uses the internal SDRAM as frame buffer memory. It consists of up to four frame buffers and the allocation of frame buffers is programmable via the FC Memory Control registers. Each frame buffer can be allocated with four parameters – start row address, rows per frame, columns per line and bits per pixel.

- Start row address : SDRAM row address of each frame buffer
  The gap between a frame buffer and the next frame buffer should be equal or greater than the size of the frame buffer (total number of rows).
  (size of a frame buffer) = (rows per frame) * (frames per buffer)

- Rows per frame : the number of rows which are occupied by one frame of a frame buffer
  (rows per frame) * (columns per row) >= (columns per line) * (lines per frame)
  where (columns per row) = 256 columns (128Mb SDRAM) or 512 columns (256Mb SDRAM)

- Columns per line : the numer of columns which are occupied by one line of a frame buffer
  (columns per line) * (bits per column) >= (bits per pixel) * (pixels per line)
  where (bits per column) = 64 bits (8bpp) or 60 bits (10bpp)

- Bits per pixel : pixel bit precision
  8 or 10 bits per pixel for video data frame
  4 or 5 bits per pixel for MH data frame

The data in the frame buffers can be accessed via the host interface.

2.11 Signal Enhancement

2.11.1 Front Noise Reduction Filter

MDIN-3xx has a one-dimensional noise reduction filter in the input block of the main video path, which consists of a median filter and an FIR filter.
The median filter has a 3-tap order-statistic filter which outputs median values within its window. It can be used to reduce impulse noises on the luminance component of the input video. The chrominance component bypasses the median filter without any filtering. The median filter is enabled by the register ***front_median_flt_on***.
The FIR filter has 15 taps for luminance and 7 taps for chrominance and their coefficients are programmable for any type of FIR filters such as a low pass filter for high frequency noise reduction. The FIR filters for luminance and chrominance are enabled by the registers ***front_nr_y_flt_on*** and ***front_nr_c_flt_on*** respectively. The FIR filter, alternatively, can be used as an anti-aliasing filter. When the input video is scaled down in the horizontal direction, the FIR filter can be used as a low pass filter to avoid aliasing.
The difference between input and output of the median filter is stored in ***front_median_flt_diff*** register and the difference between input and output of the FIR low pass filter is stored in ***front_nr_flt_diff*** register. The difference valuses are used in post noise reduction of deinterlacing process to reduce artifacts caused by noise, and user can select which result to be used by setting ***front_nr_flt_diff_sel*** register.

2.11.2 Main Noise Reduction Filter

MDIN-3xx has a one-dimensional noise reduction filter in the main format converter, which consists of an FIR filter.
The FIR filter has 7 taps for luminance only and their coefficients are programmable for any type of FIR filters such as a low pass filter for high frequency noise reduction. The filter can be used in serveral modes – fixed FIR filter or 1-D NR surface filter, which can be selected by the register ***nr_flt_mode***. In the fixed FIR filter mode, filtering is performed on the whole area. In the NR surface filter, filtering is performed on surface areas, respectively.

2.11.3 Horizontal Peaking Filter

MDIN-3xx has a high order peaking filter. It is a 15-tab order filter for horizontal peaking and has fully programmable parameters and 256-level gain control. It also has programmable coring and saturation control. There are three methods to impliment the peaking filter. But generally band pass filter method is recommended.

2.11.3.1 Band Pass Filter Method

- To get a peaking component a band pass filter is used.
- First, construct a band pass filter using from ***h_peaking_flt0*** to ***h_peaking_flt7***. This filter is a symmetric FIR filter with ***h_peaking_flt0*** as a centered coefficient. These coefficients are signed, 11-bit two's complement integer and 512(0x200) is corresponding to 1.0(unity).
- The band pass filter output is peaking component.
- Coring will be done for the peaking component with ***h_peak_flt_cor_value***.
- Clipping will be done for the cored output with ***h_peak_flt_sat_value***.
- The output will be gained with ***h_peak_flt_gain***. 0x40 is default value and means 100% gain.
- The final output is got to add the gained output to the intput.
- In this method, both ***h_peak_no_sum*** and ***h_peak_inverse*** should be 0.

2.11.3.2 Low Pass Filter Method

- To get a peaking component a low pass filter is used.
- First, construct a low pass filter using from ***h_peaking_flt0*** to ***h_peaking_flt7***. This filter is a symmetric FIR filter with ***h_peaking_flt0*** as a centered coefficient. These coefficients are signed, 11-bit two's complement integer and 512(0x200) is corresponding to 1.0(unity).
- The component to subtract the low pass filter output from input is a peaking component.
- Coring will be done for the peaking component with ***h_peak_flt_cor_value***.

- Clipping will be done for the cored output with ***h_peak_flt_sat_value***.
- The output will be gained with ***h_peak_flt_gain*** and 0x40 is default value and means 100% gain.
- The final output is got to add the gained output to the intput.
- In this method, both ***h_peak_no_sum*** and ***h_peak_inverse*** should be 1.

2.11.3.3 High Pass Filter Method

- To get a peaking component a high pass filter is used.
- First, construct a high pass filter using from ***h_peaking_flt0*** to ***h_peaking_flt7***. This filter is a symmetric FIR filter with ***h_peaking_flt0*** as a centered coefficient. These coefficients are signed, 11-bit two's complement integer and 512(0x200) is corresponding to 1.0(unity). DC gain of the high pass filter should be 1.
- The component to subtract the low pass filter output from input is a peaking component.
- Coring will be done for the peaking component with ***h_peak_flt_cor_value***.
- Clipping will be done for the cored output with ***h_peak_flt_sat_value***.
- The output will be gained with ***h_peak_flt_gain*** and 0x40 is default value and means 100% gain.
- The final output is got to add the gained output to the intput.
- In this method, ***h_peak_no_sum*** should be 1and ***h_peak_inverse*** should be 0.

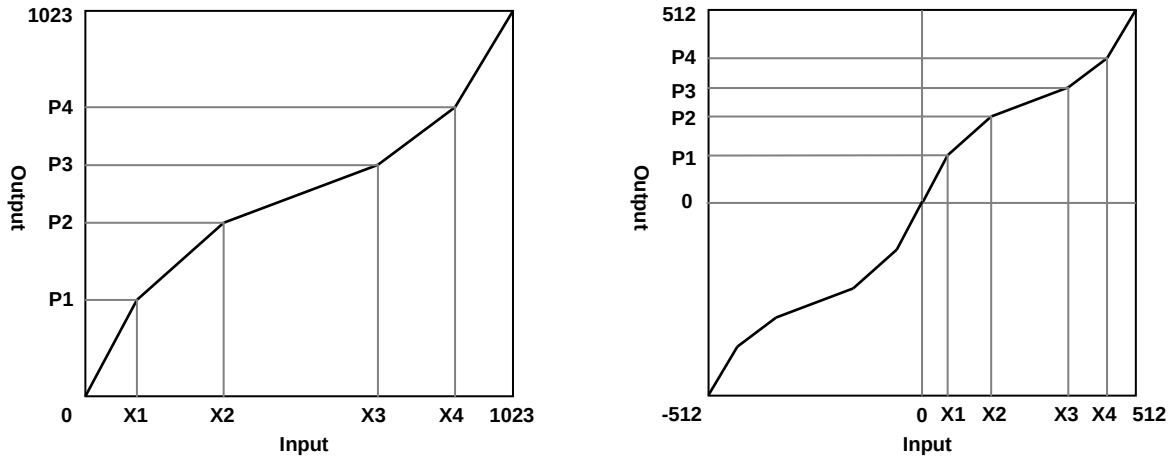
2.11.4 LTI and CTI

MDIN-3xx has one-dimensiontal LTI(Luminance Transition Improvement) and CTI(Chrominance Transition Improvement) function. The edges of luminance and chrominance component can be improved and the gain and coring levels are fully programmable. First, to get the transient value of edge, a high pass filter is used. The high pass filter output is cored by ***lti_cor_prm*** value and the cored output is gained by ***lti_gain_ctrl*** value. They are unsigned integer values. If ***lti_gain_ctrl*** value is zero, then the function is off. The final output is got to add the gained output to input. CTI is same function as LTI, but for chrominance components. For the Cb component, ***cti_b_cor_prm*** and ***cti_b_gain_ctl*** are used. For the Cr component, ***cti_r_cor_prm*** and ***cti_r_gain_ctl*** are used.

2.11.5 Piecewise Linear Transformation

MDIN-3xx provides a piecewise linear transformation both for luminance and chrominance data. The input and output ranges are divided into five segments respectively, and each input segment is mapped to an output segment based on a linear transformation for the corresponding segment as shown in Figure 2-7. The input segment points (X1, X2, X3, X4) are given through the registers ***bw_ext_x#_pos*** and ***ch_ext_x#_pos*** (# = 1,2,3,4) for luminance and chrominance data respectively. The output segment points (P1, P2, P3, P4) can be also given through the registers ***bw_ext_p#_offset*** and ***ch_ext_p#_offset*** (# = 1,2,3,4). In this manner, the contrast or color saturation of input video can be enhanced by compressing or extending the dynamic range of each segment separately.
The output segment points of luminance data can be automatically generated from the histogram of input video data, which is useful on implementing an adaptive constrast enhancement feature. Furthermore, the histogram values of each input segment can be read from the status registers. From these results, more elaborate transform curve can be programmed by the software of an external host processor.
The segment points for chrominance data can be used as the segment points for luminance data for the luminance only mode if ***luma_ext_only*** is set to '1'. In this case, up to 12 segment points can be defined.

If you want to transform Cb and Cr channel separately, set ***ch2_ext_en*** to '1' then you can transform Cb channel using ***ch_ext_x#_pos***, ***ch_ext_p#_offset***, ***ch_ext_p#_slope*** and Cr channel using ***ch2_ext_x#_pos***, ***ch2_ext_p#_offset***, ***ch2_ext_p#_slope***.
In this case, you can also transform G, B and R channels separately after color space conversion with setting ***bw_ext_after_csc*** to '1' and ***ch_ext_unsigned*** to '1'. Transforms of G, B and R channels are controlled by ***bw_ext***, ***ch_ext*** and ***ch2_ext*** settings respectively. All the channels are transformed with unsigned processing.



(a) For luminance data (b) For chrominance data

Figure 2-7. Piecewise linear transformation

2.11.6 Color Signal Enhancing Filter

MDIN-3xx has a high order FIR filter to enhance the color component signal when converting the chrominance component from 4:2:2 to 4:4:4 at the output stage. The filter has 7-tab order and the coefficients are programmable. The color component signal can have more smooth and crisp characteristics through this filter. This processing is performed in CbCr domain.
If the color signal enhancing filter is disabled, color component will be repeated for successive two pixels when the 4:2:2 video is converted to 4:4:4 format.

2.11.7 3-D Noise Reduction

MDIN-3xx can support 3-dimensional noise reduction. It includes both temporal and spatial domain noise reduction. The noise reduction performance is very high and the gain is programmable. The noise reduction is very powerful for the random and gaussian noise, especially videos from RF signal with noise.

2.11.8 Cross Color Suppression

MDIN-3xx supports cross color suppression function. When a front-end video decoder does not fully separate color component and luminance component, this function can be very useful. MDIN-3xx detects pixel level motions and suppresses the cross color component in non-motion areas of fine detail so that a coarse rainbow pattern or random colors can be reduced. It is very useful when using video decoders without supporting 3-D comb filter. This cross color suppression works only for interlaced input video of standard definition. It does not work for progressive video.

2.11.9 Mosquito Noise Reduction Filter

MDIN-3xx can reduce mosquito noises which are caused by most video compression methods. Mosquito noise is most apparent around artificial or CG (Computer Generated) objects or scrolling credits (lettering) on a plain coloured background. It occurs when reconstructing the image and approximating discarded data by inversing the transform model (iDCT). MDIN-3xx has one-dimensional mosquito noise reduction filter on the main input path, which is controllable by register settings.

2.11.10 Chroma Upsampling Error Correction

MDIN-3xx can detect and correct the chroma upsampling error (chroma bug) which is caused when MPEG decoders incorrectly upsample the chroma component of 4:2:0 interlaced video data. In each field of an MPEG frame, the chrominance samples do not lie (vertically) mid way between the luminance samples of the field, this is so that the spatial location of the chrominance samples in the frame is the same whether the frame is represented as a single frame-picture or two field-pictures. Most MPEG decoders always decode the interlaced video data as interlaced. The color for scan lines 1 and 2 in a frame is decoded and applied to scan lines 1 and 3 in the output frame. The color for scan lines 3 and 4 in a frame is decoded and applied to scan lines 2 and 4 in the output frame. So color upsampling errors occur. Sometimes these errors are conspicuous on vertical edges in chroma data, such as where a dark blue patch meets a bright red patch.
MDIN-3xx can automatically detect the presense of these chroma upsampling errors and restore the correct chroma samples when deinterlacing interlaced video fields to progressive frames.

2.12 Output Color Space Conversion

MDIN-3xx has one set of color space conversion in video output block, which can be used to convert YCbCr internal video to RGB format. This color space conversion logic is implemented as 3x3 matrix scheme to provide full flexibility. Functional block diagram of 3x3 matrix color space conversion logic in output block is shown in Figure 2-8. For proper color space conversion, there are nine coefficient registers for 3x3 matrix calculation and one control register for preprocessing of video data value. The coefficient registers of ***csc_coef0***, ***csc_coef1***, …, ***csc_coef7*** and ***csc_coef8*** in output block are user programmable. All input components and output components can be added or substracted by setting ***csc_offset_y***, ***csc_offset_cb, csc_offset_cr, csc_offset_g, csc_offset_b*** and ***csc_offset_r*** registers as user defined values. And it is possible to make the input and output value clipped to a certain range. One way is clipping for Cb, Cr, B, R to be ranging from 0 to 1023 and the other is from -512 to 511. User can select each clipping mode for each Cb, Cr, B, R. And Y and G signals are clipped to be ranging from 0 to 1023. The brightness, contrast, hue, saturation and color temperature controls can be performed using these registers. The output CSC can be bypassed by setting ***csc_bypass*** bit of ***csc_control*** register. All the coefficient and offset registers are 12-bit 2's complement values and 512 corresponds to 1.0 (unity).

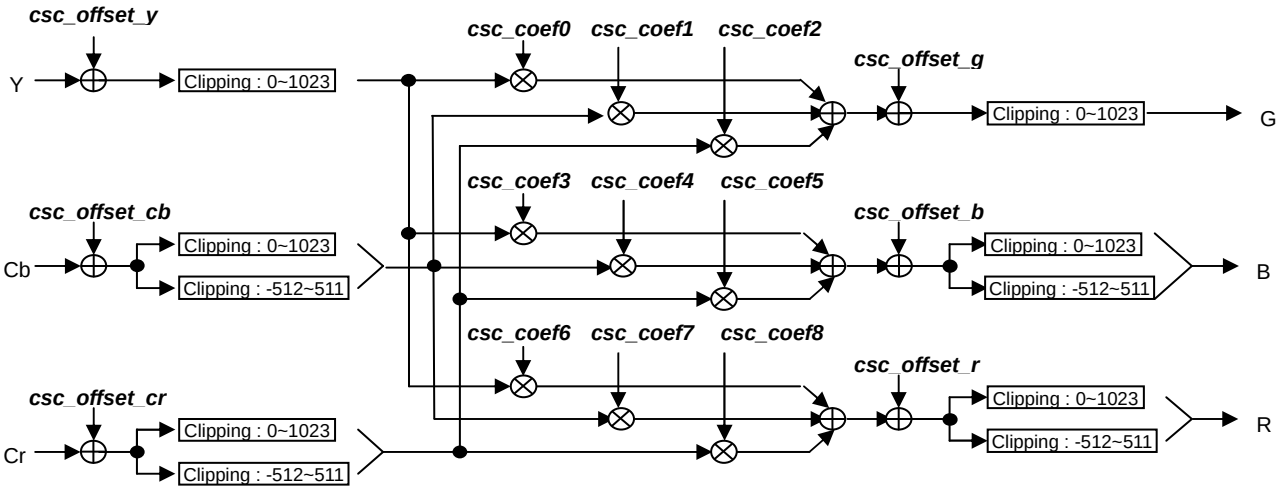


Figure 2-8. Functional block diagram of Output Color Space Conversion

The register values in the following example can be used to convert the internal ITU-R BT.601 YCbCr video to RGB format. **c0**=1.000, **c1**=–0.336, **c2**=–0.698, **c3**=1.000, **c4**=1.732, **c5**=0.000, **c6**=1.000, **c7**=0.000, **c8**=1.371 are used.

$$G'/Y' = c0 \times Y + c1 \times Cb + c2 \times Cr$$
$$B'/Cb' = c3 \times Y + c4 \times Cb + c5 \times Cr$$
$$R'/Cr' = c6 \times Y + c7 \times Cb + c8 \times Cr$$

csc_coef0 = 0x200, ***csc_coef1*** = 0xF54, ***csc_coef2*** = 0xE9B
csc_coef3 = 0x200, ***csc_coef4*** = 0x377, ***csc_coef5*** = 0x000
csc_coef6 = 0x200, ***csc_coef7*** = 0x000, ***csc_coef8*** = 0x2BE
csc_offset_y = 0x000, ***csc_offset_cb*** = 0xE00, ***csc_offset_cr*** = 0xE00
csc_offset_g = 0x000, ***csc_offset_b*** = 0x000, ***csc_offset_r*** = 0x000
csc_control = 0x1FC

Brightness, contrast, hue and saturation can be controlled using the output CSC converter. Assuming the above conversion equations, brightness, contrast, hue and saturation can be controlled by setting the CSC registers as follows. (INT12[ ] means the conversion to a 12-bit 2's complement value. **contrast** and **saturation** are floating point values with the default values of 1.0 and 1.0 respectively. **brightness** is an integer value with the default value of 0. **hue** is an angle value with the default value of 0 degree.)

csc_coef0 = INT12[**c0** * **contrast**]
csc_coef1 = INT12[{**c1** * cos(**hue**) + **c2** * sin(**hue**)} * **saturation**]
csc_coef2 = INT12[{**c2** * cos(**hue**) – **c1** * sin(**hue**)} * **saturation**]
csc_coef3 = INT12[**c3** * **contrast**]
csc_coef4 = INT12[{**c4** * cos(**hue**) + **c5** * sin(**hue**)} * **saturation**]
csc_coef5 = INT12[{**c5** * cos(**hue**) – **c4** * sin(**hue**)} * **saturation**]
csc_coef6 = INT12[**c6** * **contrast**]
csc_coef7 = INT12[{**c7** * cos(**hue**) + **c8** * sin(**hue**)} * **saturation**]
csc_coef8 = INT12[{**c8** * cos(**hue**) – **c7** * sin(**hue**)} * **saturation**]
csc_offset_y = INT12[**brightness**]
csc_offset_cb = 0xE00 = INT12[–512]
csc_offset_cr = 0xE00 = INT12[–512]
csc_offset_g = ***csc_offset_b*** = ***csc_offset_r*** = 0x000
csc_control = 0x1FC

2.13 Video Output

2.13.1 HDMI Output

MDIN-340/380 has an internal HDMI transmitter. The HDMI transmitter can carry high quality multi-channel audio data, and all standard and high-definition consumer electronics video formats. HDCP protection technology is also available using the pre-programmed internal HDCP key.
The HDMI transmitter receives video data from the video formatter and audio data either from the S/PDIF input port or from the I^2S audio input ports. If the I^2S audio input port is selected, maximum 8 channels of audio data can be carried and they can be delayed via the internal audio buffer to compensate the frame delay of video data. The video data format can be RGB/YCbCr 4:4:4 or YCbCr 4:2:2 with up to 12 bits per pixel precision.

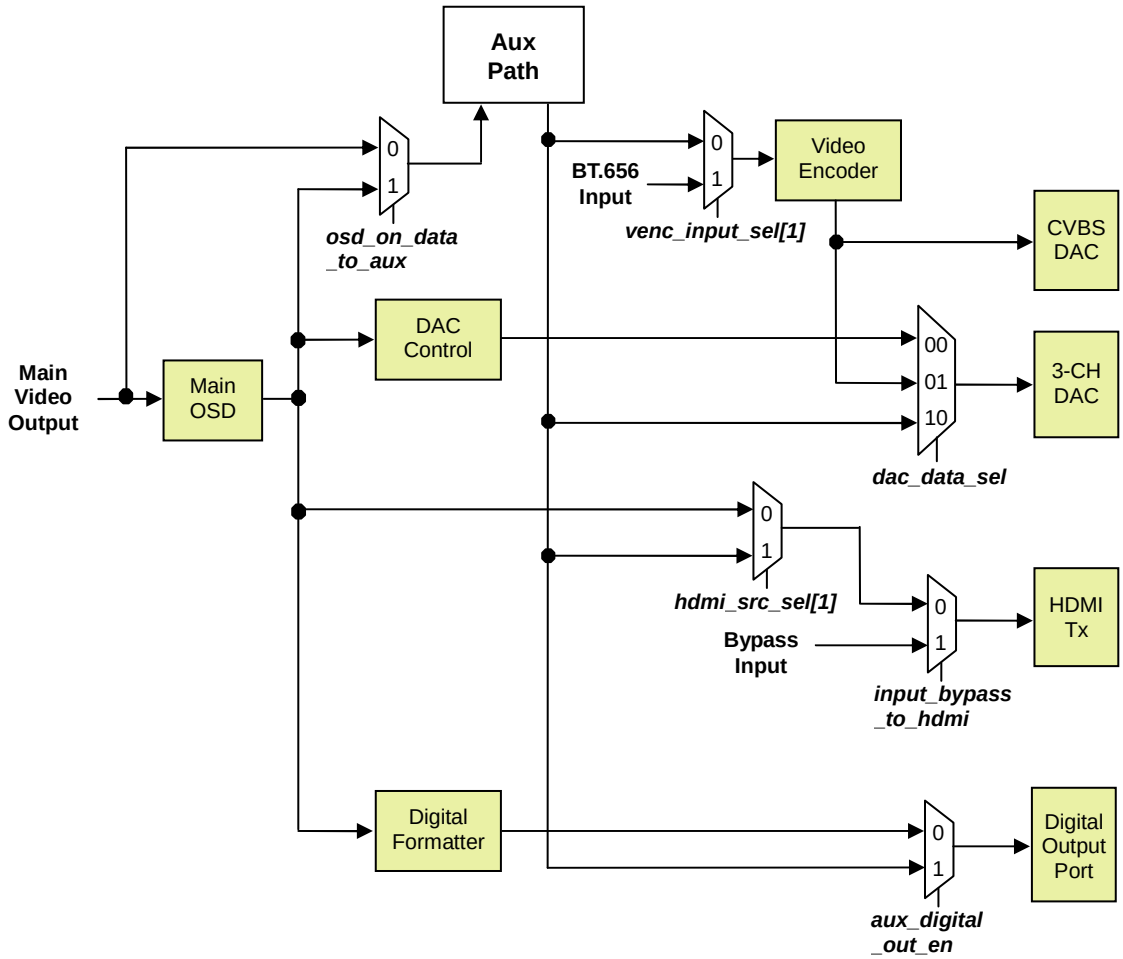
2.13.2 Analog Video Output

MDIN-3xx has an internal 10-bit triple video DAC for analog video output. It can output analog video with separate sync signals or with embedded sync signals on Y/G output :

1. **Sync separated output mode**
   MDIN-3xx can generate sync separated output mode. The output sync signals are horizontal and vertical sync. Horizontal and vertical active signal also can be output. MDIN-325/380 has separate data enable output, which can be also used to output composite sync signal.
2. **Analog sync on Green / Luminance(Y) mode**
   The internal DAC can insert a bi-level sync or a tri-level sync on the green or Y channel. In RGB mode, the sync can be inserted on all three channels. MDIN-3xx's sync generator supports various kinds of sync formats by programming the registers. The sync levels are also programmable and the smooth transition of the sync signal is supported to suppress undesired high frequency components.

2.13.3 Digital Video Output

MDIN-325 has 24 dedicated digital video output pins, while MDIN-380 can support up to 30-bit digital video output with multi-function pins. The digital video can be generated with separate sync signals and a data enable signal, or can be generated with embeded sync informations (SAV, EAV) which are embedded in the blanking area of the digital video data. The sync embedded digital format can be YCbCr 4:2:2 (ITU-R BT.1120) or YC-multiplexed 4:2:2 (ITU-R BT.656).



Note 1) BT.656 input to Video Encoder is selected by ***venc_src_sel***.
Note 2) Bypass input to HDMI Tx is determined by the input port pin mapping.

Figure 2-9. Video output block diagram

2.14 Test Pattern Generation

MDIN-3xx has two sets of test pattern generator. One is at input interface block and the other is at output block. Input test pattern generator has its own sync generator so that it is possible to perform chip or system level test without input video source only if the input clock is supplied. The video formats for input test pattern generator are 480i, 480p, 1024p and 1080i. Output test pattern is generated according to the specified output sync paramters. There are several kinds of test patterns including white image, cross pattern, cross hatch, color bar, 16 level gray scale, white window, narrow horizontal ramp, wide horizontal ramp and wide vertical ramp. Figure 2-10 and Figure 2-11 show test patterns that MDIN-3xx provides.

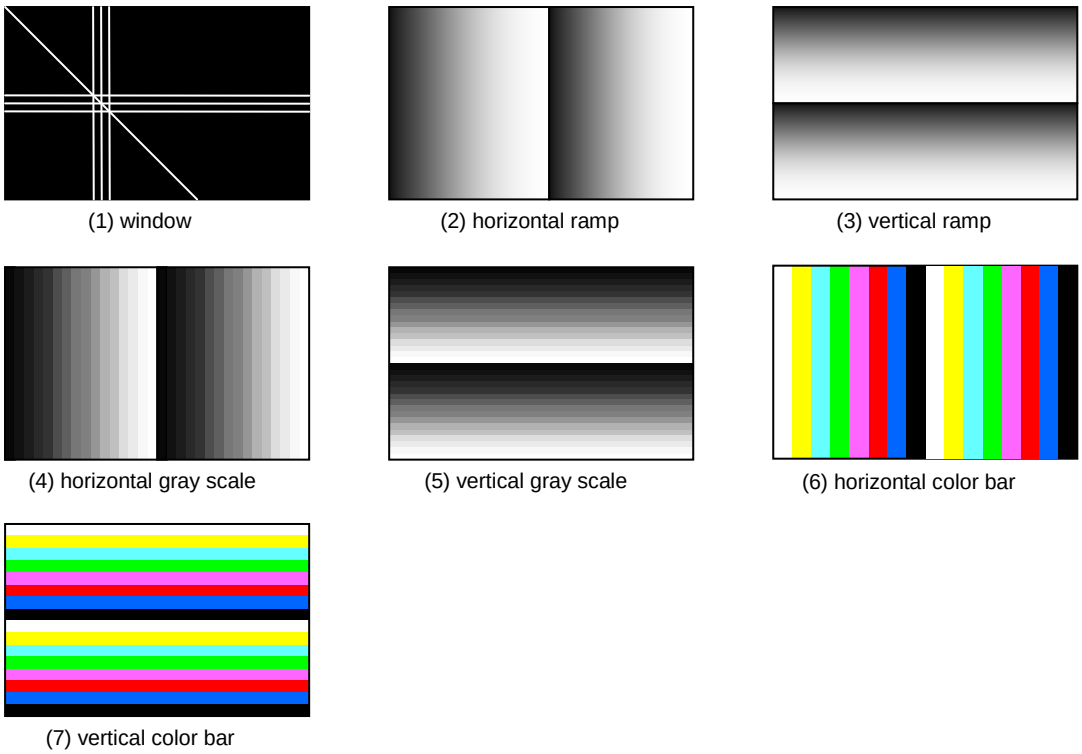


Figure 2-10. Test patterns generated in the input interface block

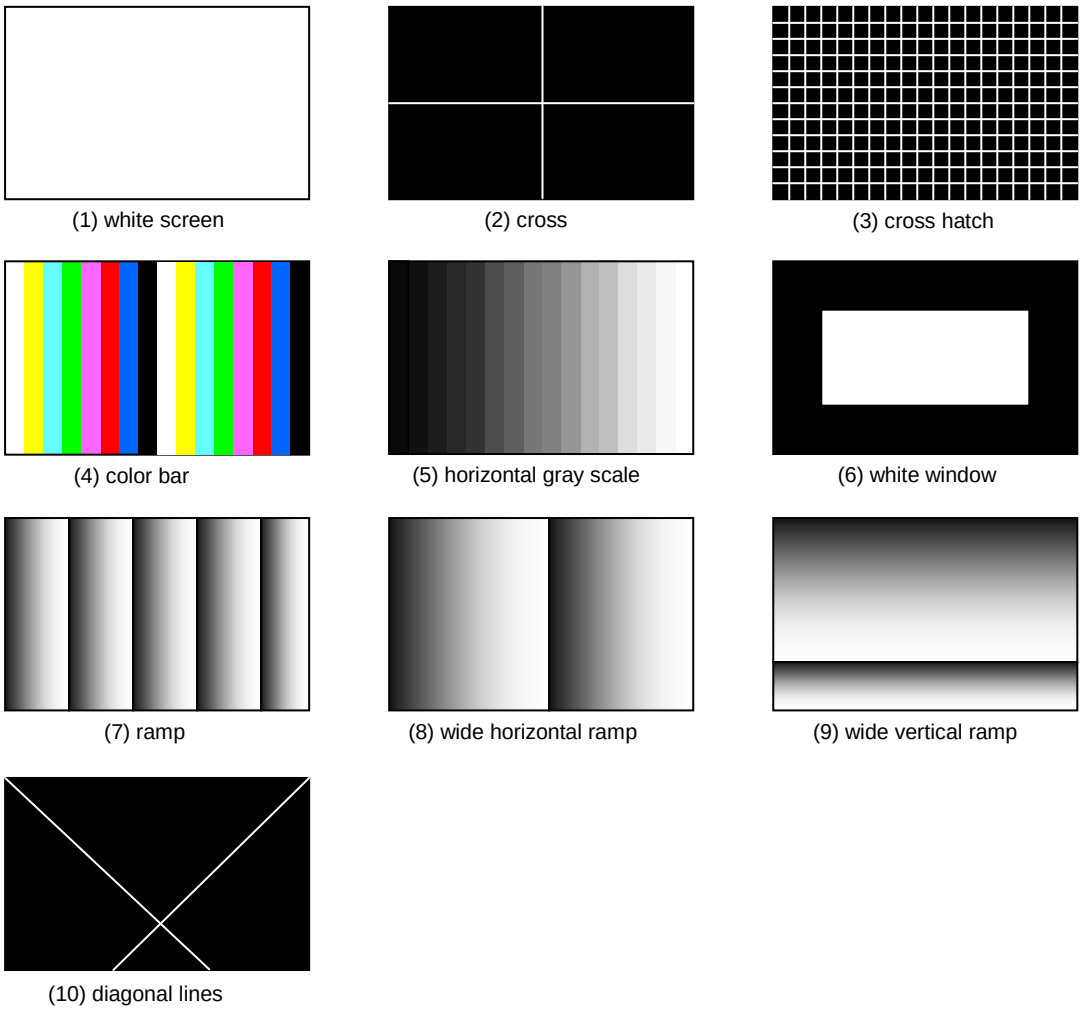


Figure 2-11. Test patterns generated in the output interface block

2.15 OSD (On Screen Display)

MDIN-3xx has a sprite-based graphic engine. It supports five layers – one background layer, one video layer, two OSD layer and one cursor layer. The background layer is the lowest one among three layers. It uses 24-bit one color, which is set by ***bg_color_y, bg_color_cb*** and ***bg_color_cr*** registers. The video layer is the layer displaying the video. The video layer is displayed over the background layer. The OSD layers are placed over the video layer. The order of two OSD layers is selectable. The cursor layer is placed over the OSD layers. Each OSD layer consists of four sprites which display bitmaps on the layer.
The OSD engine has its own background layer which is placed between the OSD layer and the video layer. This can be useful when you want to draw OSD over a solid background without displaying any video. The OSD background layer is composed of one or more rectangle solid blocks. Refer to section 2.15.4 for details.
The SDRAM memory is used to store the bitmap data of the OSD layer. It can be placed anywhere except for the video frame memory space. The available memory size for the bitmap data varies depending on the input video (size, interlaced/progressive) and the operation mode (3-D deinterlacing, 3-D NR).
MDIN-3xx provides 2, 4, 16 and 256-color palette modes and 16, 24 and 32-bit full color modes. The graphic engine supports alpha-blending of OSD layer with video layer. The OSD layer can be scaled up by 2 in horizontal and/or vertical direction through pixel duplication. The process of overlaying the video and graphics data, and the display result are shown in Figure 2-12.

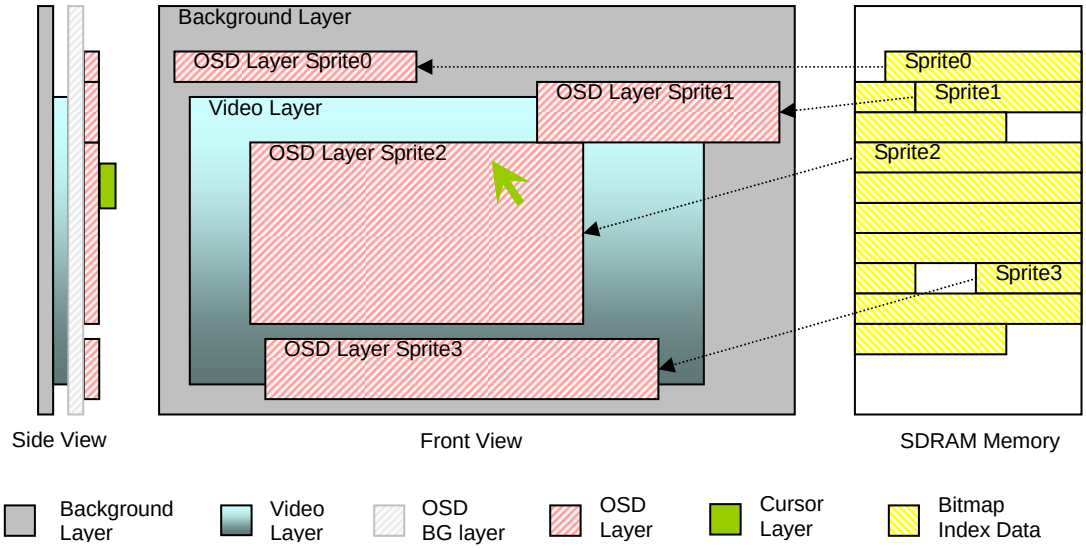


Figure 2-12. Graphics processing

2.15.1 OSD Overlay

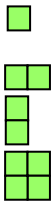
2.15.1.1 OSD Layer and Sprite

MDIN-3xx provides two OSD layers. The OSD layers are displayed over the video layer. The graphics engine provides sprite mode. A graphic image to be displayed is drawn as a sprite in an area of the SDRAM, where maximum 4 sprites are allowed in a layer. Each sprite is related to one graphic image, and its display is enabled by setting the register ***osd_sX_en*** (***X*** = 0, 1, 2, …, 7 for sprite***X***). This is useful when the parts of screen will be filled with several graphic images. sprite0 to sprite3 belong to the OSD layer0 and sprite4 to sprite7 belong to the OSD layer1. The OSD layers can be blended with the video layer using a per-sprite alpha and a per-pixel alpha. The per-sprite alpha is given by the OSD registers, and the per-pixel alpha is given in the palette of each sprite or in the alpha channel of the full color pixel data.
The graphics engine provides indexed color mode and full color mode. The indexed color mode supports 2, 4, 16 and 256-color palette modes. Each pixel is represented by 1, 2, 4 or 8-bit bitmap index which indicates a palette color entry, and each palette color entry is made up of 24-bit color and 4-bit blending informations. The blending value in the palette

entry is specified for each palette entry, and the blending level can be different among the palette entries. The full color mode supports 16, 24 and 32-bit bitmap data. 16 or 32-bit bitmap data can include up to 4-bit alpha value for per-pixel alpha blending. The graphics engine also supports 32-level per-sprite alpha blending and its level is determined by the register ***osd_sX_alpha*** (***X*** = 0 to 7 for sprite***X***) of each sprite.
The video layer is combined with the OSD layer after the video is converted to a desired color space, and the OSD colors are not affected by the output color space converter. The OSD colors can be RGB or YCbCr depending on the output color space. The graphics engine has its own color space converter for YCbCr-to-RGB conversion of 16-bit YCbCr 4:2:2 bitmap data.

2.15.1.2 Horizontal and Vertical Duplication

The OSD layer can be displayed with 1 pixel, 2 pixels or 4 pixels per one color as Figure 2-13. In 1 pixel per one color, every pixel will be display with different color data. In 2 pixels per one color, adjacent 2 pixels in a row or a column will be displayed with the same color data, where horizontal or vertical duplication scheme is used. In 4 pixels per one color, adjacent 2x2 pixels will be displayed with the same color data, where both horizontal and vertical duplication are performed for display. This feature is set by the registers ***osd_lX_rpt_h, osd_lX_rpt_v*** (***X*** = 0, 1 for layer***X***) for each layer as shown in Figure 2-13 and all the sprites in a layer will be duplicated together.



(a) 1 pixel /color : ***osd_rpt_h*** = '0' and ***osd_rpt_v*** = '0'

(b) 2 pixels/color : ***osd_rpt_h*** = '1' and ***osd_rpt_v*** = '0'

(c) 2 pixels/color : ***osd_rpt_h*** = '0' and ***osd_rpt_v*** = '1'

(d) 4 pixels/color : ***osd_rpt_h*** = '1' and ***osd_rpt_v*** = '1'

Figure 2-13. Horizontal and Vertical Duplication

2.15.1.3 Sprite Position and Size

Each sprite has position registers. ***osd_sX_disp_posi_x*** and ***osd_sX_disp_posi_y*** (***X*** = 0 to 7 for sprite***X***) determine the position of the upper left corner of a sprite image to be displayed. The vertical position must be an even number when the output is interlaced mode. Each sprite has size registers. ***osd_sX_img_size_x*** (***X*** = 0 to 7 for sprite***X***) determines the horizontal size of a sprite image and ***osd_sX_img_size_y*** (***X*** = 0 to 7 for sprite***X***) determines the vertical size of a sprite image. The rows of sprites cannot be overlapped with those of other sprites in the same layer. If they are overlapped, only the sprite whose vertical position is less will be displayed.

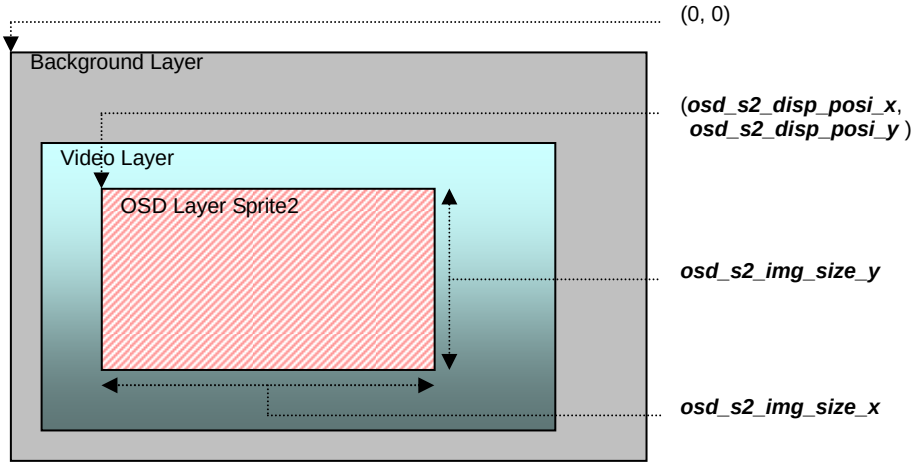


Figure 2-14. Sprite Position and Size

2.15.2 Bitmap Palette data

2.15.2.1 Palette color mode

There are four palette color modes on representing one pixel data – 2, 4, 16 and 256-color mode, which can be selected by the register ***osd_sX_pal_color*** (***X*** = 0, 1 for layer***X***). One pixel is represented by 1, 2, 4 or 8-bit palette entry address in 2, 4, 16 or 256-color mode respectively when the pixel duplication is not applied. The palette color modes are shown in Figure 2-15. Each blue box represents one bit of SDRAM data.
Every line of the bitmap data in SDRAM should start on a 64-bit word boundary. The start address and width of each line of the bitmap data in SDRAM should be a multiple of 8 bytes. The redundant bits in the end of a line whose positions exceed the horizontal size of a sprite are not displayed on the OSD layer.

3F	3E	3D	3C	3B	3A	39	38	37	36	35	34	33	32	31	30	2F	2E	2D	2C	2B	2A	29	28	…
Pixel 0								Pixel 1								Pixel 2								…

(a) 256-color mode

3F	3E	3D	3C	3B	3A	39	38	37	36	35	34	33	32	31	30	2F	2E	2D	2C	2B	2A	29	28	27	26	25	24	…
Pixel 0				Pixel 1				Pixel 2				Pixel 3				Pixel 4				Pixel 5				Pixel 6				…

(b) 16-color mode

3F	3E	3D	3C	3B	3A	39	38	37	36	35	34	33	32	31	30	2F	2E	2D	2C	2B	2A	29	28	27	26	25	24	…
Pxl 0				Pxl 1				Pxl 2				Pxl 3				Pxl 4				Pxl 5				Pxl6				Pxl 7				Pxl 8				Pxl 9				Pxl 10				Pxl 11				Pxl 12				Pxl 13				…

(c) 4-color mode

3F	3E	3D	3C	3B	3A	39	38	37	36	35	34	33	32	31	30	2F	2E	2D	2C	2B	2A	29	28	27	26	25	24	…
P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	…

(d) 2-color mode

Figure 2-15. Palette color mode

2.15.2.2 Palette color entry

MDIN-3xx has 256 palette entries in a palette memory, and all the four sprites in a layer share this palette memory. Each layer has its own palette memory. Each palette color entry consists of 24-bit color and 4-bit blending information. The final alpha blending ratio of a sprite pixel is the product of the 4-bit blending ratio of a palette color entry and 5-bit blending ratio of a sprite. In case that the 4-bit blend ratio of a palette color entry is set to zero, that palette color is completely transparent no matter what 24-bit color information may be. The construction of the color palette is shown in Figure 2-16. Each blue box represents one bit of palette SRAM.

	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Row 0	Palette color entry 0																															
Row 1	Palette color entry 1																															
…	…																															
Row FE	Palette color entry 254																															
Row FF	Palette color entry 255																															

(a) Structure of color palette memory

1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Alpha								R(Cr)								G(Y)								B(Cb)							

(b) Palette color entry

Figure 2-16. Construction of color palette

2.15.2.3 Palette Addressing

Each sprite uses 2, 4, 16 or 256 palette entries among the 256 palette entries according to its palette mode. The start entry address of each sprite is specified by the 4-bit register ***osd_sX_pal_addr*** (***X*** = 0, 1 for layer***X***). The actual address is the multiple of 16 and can be calculated by multiplying ***osd_sX_pal_addr*** by 16. If the start address plus the number of palette colors exceeds the last entry address (255), the entries from 0 will be addressed. And a sprite shares some or the whole palette entries with other sprites. An example is shown in Figure 2-17.

Palette Address	Sprite0 (16 colors)	Sprite1 (4 colors)	Sprite2 (256 colors)	Sprite3 (256 colors)
0	Color0		Color0	Color240
1	Color1		Color1	Color241
2	Color2		Color2	Color242
• • •	• • •		• • •	• • •
14	Color14		Color14	Color254
15	Color15		Color15	Color255
16		Color0	Color16	Color0
17		Color1	Color17	Color1
18		Color2	Color18	Color2
19		Color3	Color19	Color3
20			Color20	Color4
			Color21	Color5
			Color22	Color6
			Color23	Color7
			Color24	Color8
• • •			• • •	• • •
254			Color254	Color238
255			Color255	Color239
osd_sX_pal_addr	‘0000’	‘0001’	‘0000’	‘0001’

Figure 2-17. Palette Addressing Example

2.15.2.4 Palette Write and Read

MDIN-3xx uses the OSD Palette registers to write and read palette entries in the palette memory. ***osd_lX_pal_data_a***, ***osd_lX_pal_data_r***, ***osd_lX_pal_data_g***, ***osd_lX_pal_data_b*** (***X*** = 0, 1 for layer***X***) are used to specify the alpha-blending level, red/Cr, green/Y and blue/Cr component of a palette entry to be written or read, respectively. ***osd_lX_pal_wen*** (***X*** = 0, 1 for layer***X***) is used to select writing or reading operation mode and ***osd_lX_pal_addr*** (***X*** = 0, 1 for layer***X***) determines the address of a palette entry to be written or read.
Palette writing operation can be done in two steps. The first step is writting a palette color entry value to ***osd_lX_pal_data_a***, ***osd_lX_pal_data_r***, ***osd_lX_pal_data_g*** and ***osd_lX_pal_data_b***. The second step is writting the desired palette entry address to ***osd_lX_pal_addr*** with setting '1' to ***osd_lX_pal_wen***. Also, palette reading operation can be done in two steps. The first step is writting the desired color entry address to ***osd_lX_pal_addr*** with setting '0' to ***osd_lX_pal_wen***. The second step is reading palette color entry value from ***osd_lX_pal_data_a***, ***osd_lX_pal_data_r***, ***osd_lX_pal_data_g*** and ***osd_lX_pal_data_b*** registers.

2.15.3 Bitmap Index Data

Bitmap index data structure and memory configurations will be described.

2.15.3.1 Index Data Structure

Bitmap index data indicate a color entry address in the palette of a sprite which is corresponding to each OSD color to be displayed. Data construction of index data in frame buffer memory for OSD is shown in Figure 2-18. The internal bus width of MDIN-3xx for SDRAM data is 64-bit wide. OSD logic regards most-significant-bit (bit 63) as the start of first pixel data. Figure 2-18 shows an example of OSD bitmap index data construction of 16-color palette mode, where P0 represents the upper-left corner pixel of a picture.
Every line of the bitmap index data in SDRAM should start on a 64-bit word boundary. The start address and width of each line of the bitmap data in SDRAM should be a multiple of 8 bytes. The redundant bits in the end of a line whose positions exceed the horizontal size of a sprite are not displayed on the OSD layer.

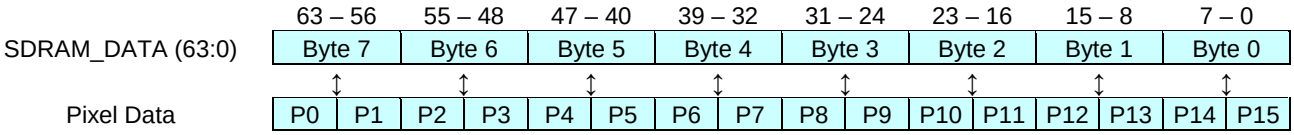
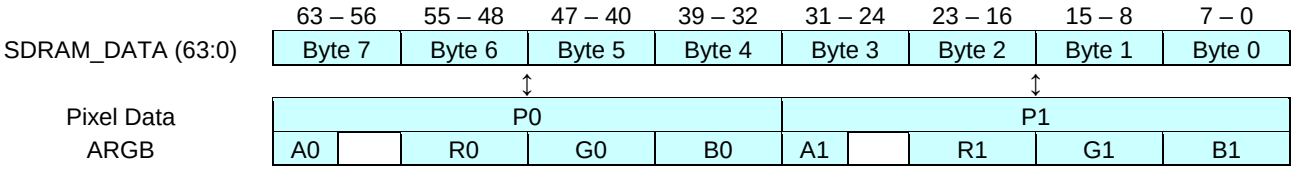


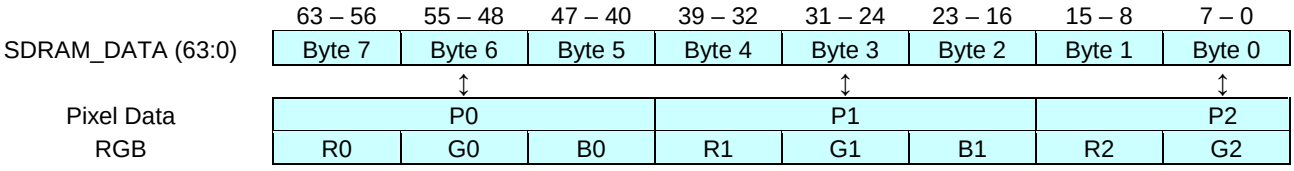
Figure 2-18. 16-color Index Data in SDRAM

2.15.3.2 Full Color Data Structure

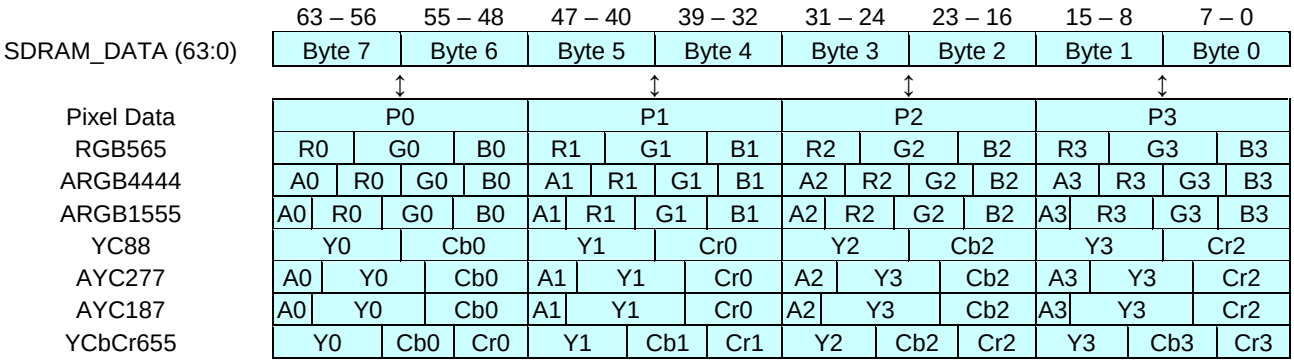
The full color bitmap is displayed without mapping from color index to color data. The bitmap data for each pixel represent the OSD color to be displayed. Data construction of full color bitmap data in frame buffer memory is shown in Figure 2-19. The internal bus width of MDIN-3xx for SDRAM data is 64-bit wide. OSD logic regards most-significant-bit (bit 63) as the start of first pixel data and P0 represents the upper-left corner pixel of a picture.
The 32-bit ARGB color consists of 4-bit alpha, 8-bit R/Cr, 8-bit G/Y and 8-bit B/Cb. The 24-bit RGB color consists of 8-bit R/Cr, 8-bit G/Y and 8-bit B/Cb. The 16-bit RGB565 color consists of 5-bit R/Cr, 6-bit G/Y and 5-bit B/Cb. The 16-bit ARGB4444 color consists of 4-bit alpha, 4-bit R/Cr, 4-bit G/Y and 4-bit B/Cb. The 16-bit ARGB1555 color consists of 1-bit alpha, 5-bit R/Cr, 5-bit G/Y and 5-bit B/Cb. The 16-bit YC88 color is 4:2:2 format and consists of 8-bit Y and 8-bit Cb/Cr. The 16-bit AYC277 color is 4:2:2 format and consists of 2-bit alpha, 7-bit Y and 7-bit Cb/Cr. The 16-bit AYC187 color is 4:2:2 format and consists of 1-bit alpha, 8-bit Y and 7-bit Cb/Cr. The 16-bit YCbCr655 color is 4:4:4 format and consists of 6-bit Y, 5-bit Cb and 5-bit Cr.
Every line of the bitmap data in SDRAM should start on a 64-bit word boundary. The start address and width of each line of the bitmap data in SDRAM should be a multiple of 8 bytes. The redundant bits in the end of a line whose positions exceed the horizontal size of a sprite are not displayed on the OSD layer.



(a) 32-bit full color bitmap data



(a) 24-bit full color bitmap data



(c) 16-bit full color bitmap data

Figure 2-19. Full Color Bitmap Data in SDRAM

2.15.3.3 Memory Configuration

MDIN-3xx utilizes the internal SDRAM as frame buffers and OSD bitmap data. The OSD bitmap data can be placed at any location in the SDRAM which is not used as video frame buffer, but the bitmap data of one sprite should be a consecutive block. Each sprite has a memory start address, which is set by ***osd_sX_bmp_row, osd_sX_bmp_bank, osd_sX_bmp_col*** (***X*** = 0 to 7 for sprite***X***).

2.15.4 Block OSD

The OSD background layer is composed of one or more rectangle solid blocks. A single full screen block can fill the entire background layer, or multiple blocks can be displayed on the background layer. Up to 8 solid rectangle blocks with different sizes and positions can be displayed at the same time. Their size can be given to any arbitrary size and they can be placed at any positions of the screen. The rectangles can be overlapped without any restrictions. The OSD background layer is always opaque and it cannot be blended with the video layer.
The size of a rectangle is specified by the registers ***block_osdX_size_h*** and ***block_osdX_size_v***, and the postion is specified by the registers ***block_osdX_posi_h*** and ***block_osdX_posi_v***. (where ***X*** = 0, 1, 2, .. 7) Each block OSD can be displayed or hidden by its own enable bit ***block_osdX_en*** and global enable bit ***osd_backgnd_en***. All the block OSD must have the same color, which is given by the register ***osd_backgnd_r, osd_backgnd_g*** and ***osd_backgnd_b***.

The registers for block OSD can be written or read using ***block_osd_data***, ***block_osd_wen*** and ***block_osd_addr***. Their address range is 0x00 to 0x20. Refer to 3.14.2 in MDIN-3xx Register Manual for the description of the block OSD registers.

2.16 Graphic Acceleration

MDIN-3xx supports several graphic acceleration functions – rectangle filling, rectangle copying, run-length decoding and character drawing. These graphic acceleration functions are performed by the internal hardware without requiring any processing power of an external host so that you can fill or copy the OSD rectangle and drawing characters on OSD sprites at high speed even with 100kbps or 400kbps I^2C bus.

2.16.1 Rectangle Filling

This function is to draw a rectangle and fill it with a selected color. The rectangle window is in a bitmap, which can be used as a sprite of OSD. 2, 4 and 8-bit indexed bitmap and 16 and 32-bit full color bitmap are supported.
At first, the information on the target bitmap should be set – the "row + bank" address in the SDRAM buffer, horizontal size and palette mode of the bitmap to the register ***ga_bitmap_start_rb*** and ***ga_bitmap_x_size***. The column address of the bitmap to be used in the rectangle filling function should be zero. Next, the position and size of the rectangle window to be filled should be set. They are given by the registers ***ga_window_x_pos***, ***ga_window_y_pos***, ***ga_window_x_size*** and ***ga_window_y_size***. The horizontal position and size values should be specified in order that the rectangle window starts and ends on 64-bit word boundaries. Finally, the acceleration process is started by setting the register bit ***ga_process_en*** to '1' with specifying the bit precision to ***ga_sprite_mode***.
The bitmap data to fill the rectangle should be given to the register ***ga_bmp_data*** as a 32-bit word value regardless of the bit precision of the target bitmap. That is, the bitmap data of consecutive 8 pixels should be given as a 32-bit word for 4-bit bitmap, and 16 pixels for 2-bit bitmap. In this manner, the target rectangle window can be filled with a vertical stripe pattern as well as a solid patten.

2.16.2 Rectangle Copying

This function is to copy the rectangle area of a bitmap to another position. The source bitmap and target bitmap need not to be a sprite of MDIN-3xx. The source rectangle and the target rectangle can be in the same bitmap or can belong to different bitmaps. They can be any rectangle area in the SDRAM space.

At first, the parameters in the following table must be set :

Register	Description
gac_mode	Set **gac_mode** to 1 for rectangle copying
gac_src_start_rb, gac_src_start_cl	Start address of source bitmap **gac_src_start_rb** is (row + bank) address and **gac_src_start_cl** is column address.
gac_dst_start_rb, gac_dst_start_cl	Start address of destination bitmap **gac_dst_start_rb** is (row + bank) address and **gac_dst_start_cl** is column address.
gac_src_precision, gac_dst_precision	Bit precision of source and destination bitmap. 4, 8, 16, 24 and 32-bit modes are supported.
gac_src_byte_pitch, gac_dst_byte_pitch	The number of bytes in a line for source and destination bitmap They should be a multiple of 8.
gac_src_start_xpos, gac_src_start_ypos	Start position within source bitmap **gac_src_start_xpos** should be a multiple of 2 for 4-bit bitmap.
gac_dst_start_xpos, gac_dst_start_ypos	Start position within destination bitmap **gac_dst_start_xpos** should be a multiple of 2 for 4-bit bitmap.
gac_src_pixel_count, gac_src_line_count	The size of the rectangle to copy The horizontal size and the vertical size of the rectangle, respectively

Table 2-9. Rectangle copying parameters

The execution is started by setting the register bit ***gac_process_en*** to '1'.
The 1 or 2-bit precision mode can be supported as 4-bit bitmap precision. In this case, one 4-bit bitmap datum is

corresponding to to four 1-bit precision data or two 2-bit precision data so that the horizontal position and size values should be scaled down by this ratio.
The rectangle filling can be performed using this function if ***gac_fill_en*** is set to '1'. The bitmap value to fill the bitmap is given to ***ga_bmp_data*** as a 32-bit word value except for 24-bit precision mode. In this case, it is not necessary to set the parameters for the source bitmap.

2.16.3 Run-length decoding

It takes quite a little time to load large bitmaps to the SDRAM of MDIN-3xx through its I^2C host interface. MDIN-3xx provides a run-length decoder to speed up the bitmap loading time. Both the source and destination of the decoder should belong to SDRAM memory. You can load the compressed bitmap data to a certain SDRAM area and decode it to the desired bitmap area.
The two encoding modes are used to encode bitmap data to a compressed format – Run-length mode and uncompressed mode. The run-length mode consists of two bytes. The first byte is the length of a run – the number of consecutive bytes to be drawn using the bitmap data in the second byte. The run can be any length from 1 to 255. The run of zero length is reserved for escape modes. The bitmap data in the second byte can be any bitmap mode data. It should consist of 8, 4, 2 or 1 index data for 1, 2, 4 or 8 color palette mode respectively.
The uncompressed mode can be started with the first byte of zero which indicates the escape mode, and the second byte is the number of bytes that follow, which are the bitmap data to be drawn. The second byte should be from 2 to 255. Every sequence of the encode data should be aligned on a 2-byte word boundary. One dummy byte should be padded if an uncompressed mode sequence terminates with odd number of bytes.
If the second byte which follows the first byte of zero in the escaped mode is less than 2, it has special meanings. "0" means the end of line and "1" means the end of bitmap. If the end of line appears, the next byte should be the start of next line or the end of bitmap. If the end of bitmap is encountered, the run-length decoder goes to the idle state.
Summarizing the format of run-length bitstream:

- 2-byte run-length coding (run = 1 ~ 255) : 1st = run(# of bytes), 2nd = data byte
- Uncompressed mode (run=0) : 2nd byte = 2 ~ 255 (# of uncompressed data bytes that follow)
- Escape code (run=0) : 2nd byte = 0(end of line), 1(end of bitmap)

The run-length decoding is started by setting the register bit ***gac_process_en*** to '1' after the parameters in the following table are set.

Register	Description
gac_mode	Set **gac_mode** to 0 for run-length decoding mode
gac_src_start_rb, gac_src_start_cl	Start address of compressed stream **gac_src_start_rb** is (row + bank) address and **gac_src_start_cl** is column address.
gac_dst_start_rb, gac_dst_start_cl	Start address of destination bitmap **gac_dst_start_rb** is (row + bank) address and **gac_dst_start_cl** is column address.
gac_dst_precision	Color palette mode of destination bitmap. 4, 8, 16, 24 and 32-bit modes are supported.
gac_dst_byte_pitch	The number of bytes in a line for destination bitmap They should be a multiple of 8.
gac_dst_start_xpos, gac_dst_start_ypos	Start position within destination bitmap **gac_dst_start_xpos** should be a multiple of 2 for 4-bit bitmap.
gac_src_byte_pitch, gac_src_line_count	**gac_src_byte_pitch** and **gac_src_line_count** indicate the maximum size of the data to decode. Even though the compressed stream does not terminate by the end of bitmap code, the decoding will be terminated if the number of decoded data reaches the maximum size. (**gac_src_byte_pitch** * **gac_src_line_count**) should be greater than or equal to the size of compressed bitmap in bytes.

Table 2-10. Run-length decoding parameters

The 1 or 2-bit precision mode can be supported as 4-bit bitmap precision. In this case, one 4-bit bitmap datum is

corresponding to to four 1-bit precision data or two 2-bit precision data so that the horizontal position and size values should be scaled down by this ratio.

2.16.4 Character drawing

MDIN-3xx supports character-based OSD as well as bitmap-based OSD. Characters can be drawn on a bitmap using the font data pre-loaded in the SDRAM memory. There are two modes to specify the location in a bitmap to draw characters – character position mode and XY position mode. In the character position mode, the whole rectangle of a bitmap is divided into 32x16 or 16x32 blocks of a character size and each block is mapped to a register which can be programmed with a font index and a palette index. So, you can draw a character just by writing the index of a font with its palette index. In the XY position mode, the position of a character to be drawn is given by the horizontal and vertical positon in a bitmap. Auto-increment of the horizontal or vertical position in a character size is possible without re-writing a new position when contiguous characters are drawn successively.
2, 16 or 256-color bitmap can be used as a font bitmap. When 2-color fonts are used, each bit of font bitmap corresponds to a pixel in the 16-color destination bitmap. The pixel is rendered with the given foreground color index if the bit is one. Otherwise, it is rendered with the given background color index. The foreground and background color indexes are not used when 16 or 256-color fonts are used. The color index of the font bitmap will be copied to the destination bitmap without any conversion.
A font bitmap consists of consecutive bitmap blocks, each of which is corresponding to a character in the font data. Each font block is accessed by its index. The first bitmap block has the index of zero. The maximum number of blocks in a font bitmap is 256. If more than 256 kinds of characters are needed, multiple font bitmaps in different memory areas can be used.

2.16.4.1 Character position mode

At first, the parameters in the following table must be set :

Register	Description
gac_mode	Set **gac_mode** to 2 for character position mode drawing
gac_src_start_rb, gac_src_start_cl	Start address of font bitmap **gac_src_start_rb** is (row + bank) address and **gac_src_start_cl** is column address.
gac_dst_start_rb, gac_dst_start_cl	Start address of destination bitmap **gac_dst_start_rb** is (row + bank) address and **gac_dst_start_cl** is column address.
gac_src_precision	Bit precision of font bitmap. 1, 4 and 8-bit precisions are supported.
gac_dst_precision	Bit precision of destination bitmap. It should be 4-bit precision for 1-bit precision font, and should be the same as the source bit precesion.
gac_char_xsize, gac_char_ysize	The font size in horizontal and vertical direction, respectively **gac_char_xsize** should be an even number less than or equal to 32 and **gac_char_ysize** can be any integer less than 64.
gac_src_byte_pitch	The number of bytes in a line for font bitmap It should be as follows : **gac_char_xsize** — **gac_src_byte_pitch** 0 ~ 16 (2-color) — 2 18 ~ 32 (2-color) — 4 0 ~ 16 (16-color) — 8 18 ~ 32 (16-color) — 16 0 ~ 16 (256-color) — 16 18 ~ 32 (256-color) — 32
gac_dst_byte_pitch	The number of bytes in a line for destination bitmap They should be a multiple of 8.
gac_char_grid_mode	The grid mapping mode in character position mode The 9 LSB address bits (8:0) of **gac_char_mode_dataX** can be mapped to the horizontal position (x_pos) and the vertical position (y_pos) of a 32x16 or 16x32 grid as follows : **gac_char_grid_mode** — x_pos — y_pos 0 (32x16 grid) — (4:0) — (8:5) 1 (16x32 grid) — (3:0) — (8:4) 2 (32x16 grid) — (8:4) — (3:0) 3 (16x32 grid) — (8:5) — (4:0)
gac_dst_start_xpos, gac_dst_start_ypos	Position offset of the top-left character within destination bitmap **gac_dst_start_xpos** should be a multiple of 2 for 4-bit bitmap.

Table 2-11. Character position mode parameters

The execution is started by writing anyone of ***gac_char_mode_dataX*** (where ***X*** is 0 ~ 511, and this address value of each register is mapped to a position in 32x16 or 16x32 character grid) with the values of ***gac_fg_color_index***, ***gac_bg_color_index*** and ***gac_font_index***, after setting the registers bit ***gac_auto_process_en*** to '1'. ***gac_fg_color_index*** and ***gac_bg_color_index*** are palette indexes for the foreground and background color of the character, respectively, to be drawn. They are 4-bit values since the destination bitmap must be 16-color mode, and have no meaning for 16 or 256-color font. ***gac_font_index*** is the zero-based index of the font which is selected to draw the character.
Since registers can be written in burst via I^2C bus interface, contiguous characters in a row or column can be drawn by writing their font index data and palette color index data to ***gac_char_mode_dataX*** in burst with one I^2C transaction.

gac_char_mode_data0, **gac_char_mode_data1,** **gac_char_mode_data2,** **gac_char_mode_data3,** **…** **…** **gac_char_mode_data510,** **gac_char_mode_data511**	These 512 16-bit registers in the offset 0x200 ~ 0x3FF include the font index data and palette color index data of characters which are mapped to the character positions of 32x16 or 16x32 grid within a bitmap. Each register has the informations on a character to be drawn as follows (Palette indexes are not used for 16-color font.) : (15:12) = **gac_fg_color_index** (foreground color palette index) (11:8) = **gac_bg_color_index** (background color palette index) (7:0) = **gac_font_index** (zero-based font index)

Table 2-12. Character position mode data registers

2.16.4.2 XY position mode

At first, the parameters in the following table must be set :

Register	Description
gac_mode	Set **gac_mode** to 3 for XY position mode drawing
gac_src_start_rb, **gac_src_start_cl**	Start address of font bitmap **gac_src_start_rb** is (row + bank) address and **gac_src_start_cl** is column address.
gac_dst_start_rb, **gac_dst_start_cl**	Start address of destination bitmap **gac_dst_start_rb** is (row + bank) address and **gac_dst_start_cl** is column address.
gac_src_precision	Bit precision of font bitmap. 1, 4 and 8-bit precisions are supported.
gac_dst_precision	Bit precision of destination bitmap. It should be 4-bit precision for 1-bit precision font, and should be the same as the source bit precesion.
gac_char_xsize, **gac_char_ysize**	The font size in horizontal and vertical direction, respectively **gac_char_xsize** should be an even number less than or equal to 32 and **gac_char_ysize** can be any integer less than 64.
gac_src_byte_pitch	The number of bytes in a line for font bitmap Same meaning as the character position mode
gac_dst_byte_pitch	The number of bytes in a line for destination bitmap They should be a multiple of 8.
gac_xy_auto_inc	Auto position increment on XY position mode The destination position will be incremented automatically after drawing the current chararcter as follows : 0 no auto-increment 1 auto-increment in horizontal direction 2 auto-increment in vertical direction 3 auto-increment in both directions
gac_dst_start_xpos, **gac_dst_start_ypos**	Position offset of the top-left character within destination bitmap **gac_dst_start_xpos** should be a multiple of 2 for 4-bit bitmap.

Table 2-13. XY position mode parameters

The execution is started by writing the values of ***gac_fg_color_index***, ***gac_bg_color_index*** and ***gac_font_index*** to the register ***gac_xy_mode_data***, after setting the registers bit ***gac_auto_process_en*** to '1'. Any register of ***gac_char_mode_dataX*** (where ***X*** is 0 ~ 511) can be used instead of ***gac_xy_mode_data*** for the burst write mode. The register offset ***X*** has no meaning on XY position mode. Consecutive characters in a row or column can be drawn if the desired font index and color index data are written to any range of these registers in burst with enabling the auto-increment feature. ***gac_fg_color_index*** and ***gac_bg_color_index*** have no meaning for 16 or 256-color font.

gac_xy_mode_data	Font index and palette index of a character to be drawn Palette indexes are not used for 16-color font. (15:12) = **gac_fg_color_index** (foreground color palette index) (11:8) = **gac_bg_color_index** (background color palette index) (7:0) = **gac_font_index** (zero-based font index)

Table 2-14. XY position mode data register

2.17 Auxiliary Video Path

MDIN-3xx has two video paths – main video path and auxiliary video path. The auxiliary video path receives the input video data from the input ports or the output video data of the main video path. It has its own format converter which can scale up and down videos with an arbitrary ratio. The auxiliary video output can be fed to the triple DAC, the video encoder and the digital output port.

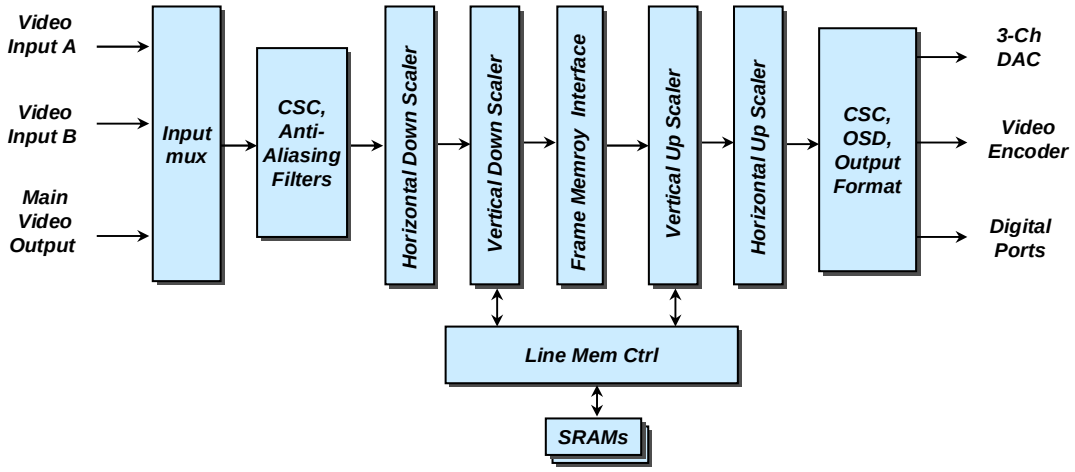


Figure 2-20. Block Diagram of Auxiliary Video Path

The aux video path can operate as follows :

1) Alternative Format Converter
The aux video path receives a video signal from one of the two input ports or the main video path and converts it to a different video format. In this case, the sync generator of the aux video path operates independent of the main video path. MDIN-3xx can be configured to output the main video on the HDMI transmitter and the aux video on the DAC. The output stages of both paths can operate at different video clocks. Refer to “3.3 Internal PLL and Clock” for details.

2) PIP or POP Mode
The aux video path receives a video signal from one of the two input ports and converts it to a different video format. In this case, the sync generator of the aux video path is locked to the sync of the main video path, and the aux video is overlaied on the main video. The output stages of both paths should operate at the same video clock and the output sync timings should be identical to those of the main path.

3) Multi-channel Processing Mode
The main video path receives up to four channels of video and performs all the video processings in a frame-multiplexed fashion. The processed video frames are transferred to the aux video path. The aux video path gathers the frames of the video channels and outputs them on the digital output port with a time-multiplexed format. Alternatively, the aux video path can display the video channels as multiple windows on a single screen.

2.17.1 Color Space Conversion

The aux video path has its own color space converters on both the input and output stages. Normally the input CSC is used to convert an RGB video into YCbCr format for internal 4:2:2 processing, and the output CSC is used to convert the internal YCbCr video to RGB format. Both the color space converters have the same architecture as those of the main path. They have selectable sets of preset matrices (shown in Table 2-15) as well as fully programmable CSC coefficients.

CSC coefficients	RGB (16-235) to BT.601	RGB (0-255) to BT.601	RGB (16-235) to BT.709	RGB (0-255) to BT.709	BT.601 to RGB (16-235)	BT.601 to RGB (0-255)	BT.709 to RGB (16-235)	BT.709 to RGB (0-255)	BT.601 to BT.709	BT.709 to BT.601
csc_coef0	0.587	0.504	0.715	0.614	1.000	1.164	1.000	1.164	1.000	1.000
csc_coef1	0.114	0.098	0.072	0.062	-0.336	-0.391	-0.183	-0.213	-0.115	0.100
csc_coef2	0.299	0.257	0.213	0.183	-0.698	-0.813	-0.459	-0.534	-0.207	0.192
csc_coef3	-0.339	-0.291	-0.394	-0.338	1.000	1.164	1.000	1.164	0.000	0.000
csc_coef4	0.511	0.439	0.511	0.439	1.732	2.018	1.816	2.115	1.018	0.990
csc_coef5	-0.172	-0.148	-0.117	-0.101	0.000	0.000	0.000	0.000	0.114	-0.110
csc_coef6	-0.428	-0.368	-0.464	-0.399	1.000	1.164	1.000	1.164	0.000	0.000
csc_coef7	-0.083	-0.071	-0.047	-0.040	0.000	0.000	0.000	0.000	0.075	-0.072
csc_coef8	0.511	0.439	0.511	0.439	1.371	1.596	1.540	1.793	1.025	0.984

Table 2-15. Preset Coefficients of CSCs in Auxiliary Path

2.17.2 Anti-aliasing Filter

The aux path has four FIR filters – horizontal and vertical filters for both the luminance and chrominance channels. The horizontal filter for luminance is 9-tap, the horizontal filter for chrominance is 7-tap, the vertical filter for luminance is 5-tap and the vertical filter for chrominance is 3-tap. These filters can be used as anti-aliasing filter when the input video is down-scaled. They can also be used to reduce the noises. When a filter is not necessary, it needs to be disabled and bypassed to save the power consumption.
The vertical filters can be enabled only when the horizontal video size in the line memories which is specified by ***aux_mem_size_h*** is not greater than 1024. If the horizontal size is greater than 1024, the vertical filters should be bypassed and ***aux_linemem_cfg*** should be set to '1'.

2.17.3 Video Format Conversion

The format converter of the aux path supports both up-conversion and down-conversion. The down-conversion is performed before the video is written to the frame memory, and the up-conversion is performed after the video is read from the frame memory. This architecture is helpful to reduce the size and bandwidth of the frame memory.
The vertical up-conversion and down-conversion cannot be performed at the same time, since they are implemented by a single set of line memories. So, they should be enabled mutually exclusively. The vertical up-conversion or down-conversion is selected by setting the register ***aux_vfc_mode***. If it is set to '1', up-conversion is selected. Otherwise, down-conversion is selected. The horizontal up-conversion and down-conversion can be performed at the same time.
When a format conversion is not necessary, the corresponding format converter needs to be disabled and bypassed to save the power consumption.
Figure 2-21 shows the concept of the format conversion process in the aux video path. The vertical scaling factor is determined by the ratio of ***aux_src_size_v*** to ***aux_mem_size_v*** on down-conversion mode, and it is determined by the ratio of ***aux_mem_size_v*** to ***aux_dest_size_v*** on up-conversion mode.

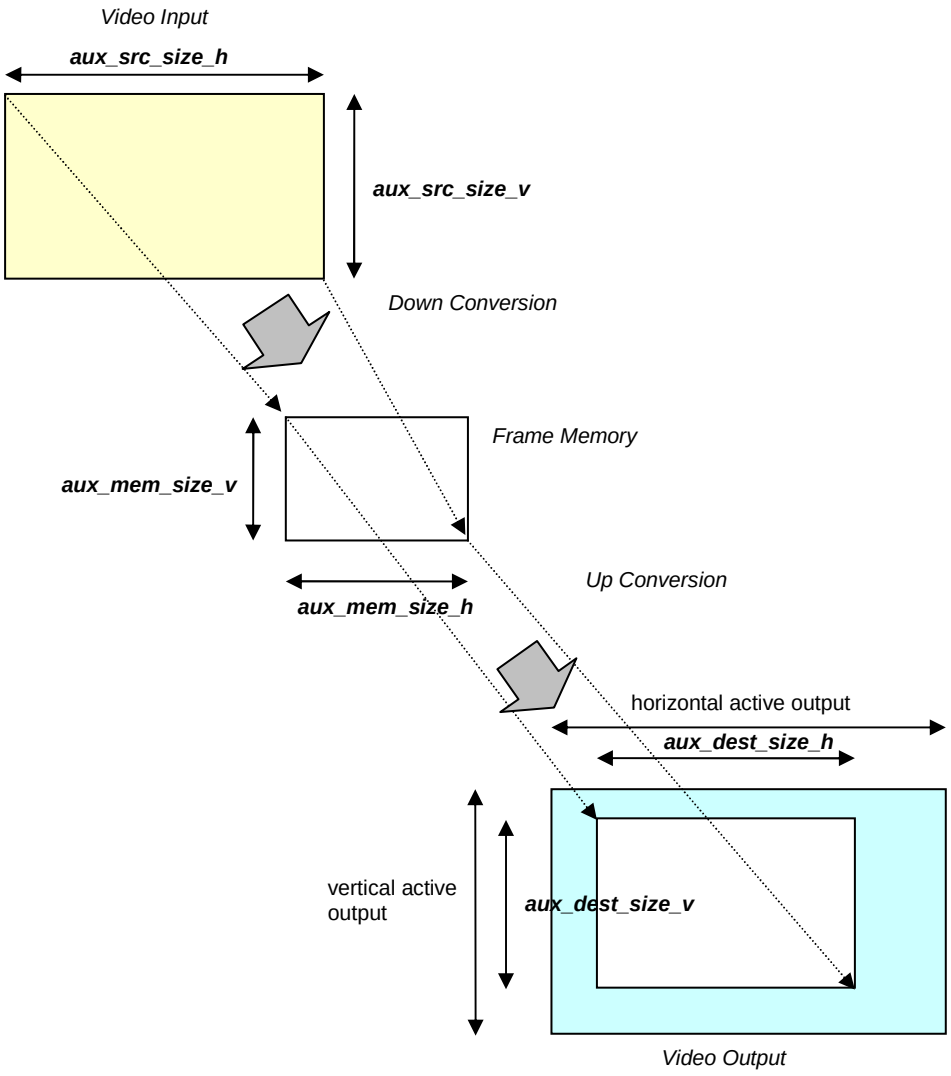


Figure 2-21. Scaling of Auxiliary Video

2.17.4 OSD

The graphic engine of the aux path operates similarly to that of the main path, but it has one bitmap layer with four sprites and one cursor layer. Both the layers support 2/4/16-color indexed mode. Each layer has its own palette, and the palette entries are directly written to or read from the registers without using a palette memory.
The bitmap data can be allocated in the SDRAM. Alpha blending and pixel repetition are also supported.

2.17.5 Video Output

The aux video can be output as an RGB or YCbCr component video to the triple DAC. MDIN-325/380 can output the aux video as a composite video to its CVBS DAC after encoding it in the video encoder. When S-Video output (Y/C) is necessary, it can be output using two channels of the triple DAC.
MDIN-325/380 can also output the aux video on the digital output port by setting the register ***aux_digital_out_en***. The digital output pin mapping of MDIN-325 is shown in Table 2-16. The register bit ***aux_out_data_mode*** which controls the

pin configuration of the aux digital output should be fixed to '0101' for MDIN-325. For MDIN-380, the pin configuration can be selected by ***aux_out_data_mode*** as shown in Table 2-17.

Mode / Pin Name	4:4:4 24-bit	YC 4:2:2 16-bit	YC-muxed 8-bit
DATA_OUT23	R7 / Cr7	C7	YC_B7
DATA_OUT22	R6 / Cr6	C6	YC_B6
DATA_OUT21	R5 / Cr5	C5	YC_B5
DATA_OUT20	R4 / Cr4	C4	YC_B4
DATA_OUT19	R3 / Cr3	C3	YC_B3
DATA_OUT18	R2 / Cr2	C2	YC_B2
DATA_OUT17	R1 / Cr1	C1	YC_B1
DATA_OUT16	R0 / Cr0	C0	YC_B0
DATA_OUT15	G7 / Y7	Y7	YC_A7
DATA_OUT14	G6 / Y6	Y6	YC_A6
DATA_OUT13	G5 / Y5	Y5	YC_A5
DATA_OUT12	G4 / Y4	Y4	YC_A4
DATA_OUT11	G3 / Y3	Y3	YC_A3
DATA_OUT10	G2 / Y2	Y2	YC_A2
DATA_OUT9	G1 / Y1	Y1	YC_A1
DATA_OUT8	G0 / Y0	Y0	YC_A0
DATA_OUT7	B7 / Cb7	-	-
DATA_OUT6	B6 / Cb6	-	-
DATA_OUT5	B5 / Cb5	-	-
DATA_OUT4	B4 / Cb4	-	-
DATA_OUT3	B3 / Cb3	-	-
DATA_OUT2	B2 / Cb2	-	-
DATA_OUT1	B1 / Cb1	-	-
DATA_OUT0	B0 / Cb0	-	-
aux_out_mux_en	'0'	'0'	'1'
aux_out_444	'1'	'0'	'0'

Table 2-16. MDIN-325 Output Port Pin Mapping for Auxiliary Video Output

aux_out_data_mode / Pin Name	0 (none)	1 (8-pin)	2 (16-pin)	3 (24-pin)	4 (32-pin)	5 (24-pin)	6 (32-pin)	7 (16-pin)
D_IN23	-	-	-	-	D31	-	-	-
D_IN22	-	-	-	-	D30	-	-	-
D_IN21	-	-	-	-	D29	-	-	-
D_IN20	-	-	-	-	D28	-	-	-
D_IN19	-	-	-	-	D27	-	-	-
D_IN18	-	-	-	-	D26	-	-	-
D_IN17	-	-	-	-	D25	-	-	-
D_IN16	-	-	-	-	D24	-	-	-
D_IN15	-	-	-	D23	D23	-	D31	-
D_IN14	-	-	-	D22	D22	-	D30	-
D_IN13	-	-	-	D21	D21	-	D29	-
D_IN12	-	-	-	D20	D20	-	D28	-
D_IN11	-	-	-	D19	D19	D23	D27	-
D_IN10	-	-	-	D18	D18	D22	D26	-
D_IN9	-	-	-	D17	D17	D21	D25	-
D_IN8	-	-	-	D16	D16	D20	D24	-
D_IN7	-	D7	-	D15	D15	D19	D23	-
D_IN6	-	D6	-	D14	D14	D18	D22	-
D_IN5	-	D5	-	D13	D13	D17	D21	-
D_IN4	-	D4	-	D12	D12	D16	D20	-
D_IN3	-	D3	D15	D11	D11	D15	D19	-
D_IN2	-	D2	D14	D10	D10	D14	D18	-
D_IN1	-	D1	D13	D9	D9	D13	D17	-
D_IN0	-	D0	D12	D8	D8	D12	D16	-
D_OUT7	-	-	D11	D7	D7	D11	D15	D15
D_OUT6	-	-	D10	D6	D6	D10	D14	D14
D_OUT5	-	-	D9	D5	D5	D9	D13	D13
D_OUT4	-	-	D8	D4	D4	D8	D12	D12
D_OUT3	-	-	D7	D3	D3	D7	D11	D11
D_OUT2	-	-	D6	D2	D2	D6	D10	D10
D_OUT1	-	-	D5	D1	D1	D5	D9	D9
D_OUT0	-	-	D4	D0	D0	D4	D8	D8
HOST_AD19	-	-	-	-	-	-	-	-
HOST_AD18	-	-	-	-	-	-	-	-
HOST_AD17	-	-	-	-	-	-	-	-
HOST_AD16	-	-	-	-	-	-	-	-
HOST_AD15	-	-	-	-	-	-	-	D7
HOST_AD14	-	-	-	-	-	-	-	D6
HOST_AD13	-	-	-	-	-	-	-	D5
HOST_AD12	-	-	-	-	-	-	-	D4
HOST_AD11	-	-	D3	-	-	D3	-	D3
HOST_AD10	-	-	D2	-	-	D2	-	D2
HOST_AD9	-	-	D1	-	-	D1	-	D1
HOST_AD8	-	-	D0	-	-	D0	-	D0
HOST_AD7	-	-	-	-	-	-	D7	-
HOST_AD6	-	-	-	-	-	-	D6	-
HOST_AD5	-	-	-	-	-	-	D5	-
HOST_AD4	-	-	-	-	-	-	D4	-
HOST_AD3	-	-	-	-	-	-	D3	-
HOST_AD2	-	-	-	-	-	-	D2	-
HOST_AD1	-	-	-	-	-	-	D1	-
HOST_AD0	-	-	-	-	-	-	D0	-

Table 2-17. MDIN-380 Output Port Pin Configurations for Auxiliary Video Output

aux_out_data_mode / Pin Name	8 (32-pin)	9 (8-pin)	10 (16-pin)	11 (32-pin)	12 (16-pin)	13 (16-pin)	14 (24-pin)	15 (16-pin)
D_IN23	-	-	-	-	-	-	-	-
D_IN22	-	-	-	-	-	-	-	-
D_IN21	-	-	-	-	-	-	-	-
D_IN20	-	-	-	-	-	-	-	-
D_IN19	D31	-	-	-	-	-	-	-
D_IN18	D30	-	-	-	-	-	-	-
D_IN17	D29	-	-	-	-	-	-	-
D_IN16	D28	-	-	-	-	-	-	-
D_IN15	D27	-	-	-	-	-	-	-
D_IN14	D26	-	-	-	-	-	-	-
D_IN13	D25	-	-	-	-	-	-	-
D_IN12	D24	-	-	-	-	-	-	-
D_IN11	D23	-	-	-	-	-	-	-
D_IN10	D22	-	-	-	-	-	-	-
D_IN9	D21	-	-	-	-	-	-	-
D_IN8	D20	-	-	-	-	-	-	-
D_IN7	D19	-	-	-	-	-	-	D15
D_IN6	D18	-	-	-	-	-	-	D14
D_IN5	D17	-	D15	D31	-	-	-	D13
D_IN4	D16	-	D14	D30	-	-	-	D12
D_IN3	D15	-	D13	D29	-	-	-	D11
D_IN2	D14	-	D12	D28	-	-	-	D10
D_IN1	D13	-	D11	D27	-	-	-	D9
D_IN0	D12	-	D10	D26	-	-	-	D8
D_OUT7	D11	D7	D9	D25	D15	D15	D23	D7
D_OUT6	D10	D6	D8	D24	D14	D14	D22	D6
D_OUT5	D9	D5	D7	D23	D13	D13	D21	D5
D_OUT4	D8	D4	D6	D22	D12	D12	D20	D4
D_OUT3	D7	D3	D5	D21	D11	D11	D19	D3
D_OUT2	D6	D2	D4	D20	D10	D10	D18	D2
D_OUT1	D5	D1	D3	D19	D9	D9	D17	D1
D_OUT0	D4	D0	D2	D18	D8	D8	D16	D0
HOST_AD19	-	-	D1	-	D7	-	-	-
HOST_AD18	-	-	D0	-	D6	-	-	-
HOST_AD17	-	-	-	D17	D5	-	-	-
HOST_AD16	-	-	-	D16	D4	-	-	-
HOST_AD15	-	-	-	D15	D3	-	D15	-
HOST_AD14	-	-	-	D14	D2	-	D14	-
HOST_AD13	-	-	-	D13	D1	-	D13	-
HOST_AD12	-	-	-	D12	D0	-	D12	-
HOST_AD11	D3	-	-	D11	-	-	D11	-
HOST_AD10	D2	-	-	D10	-	-	D10	-
HOST_AD9	D1	-	-	D9	-	-	D9	-
HOST_AD8	D0	-	-	D8	-	-	D8	-
HOST_AD7	-	-	-	D7	-	D7	D7	-
HOST_AD6	-	-	-	D6	-	D6	D6	-
HOST_AD5	-	-	-	D5	-	D5	D5	-
HOST_AD4	-	-	-	D4	-	D4	D4	-
HOST_AD3	-	-	-	D3	-	D3	D3	-
HOST_AD2	-	-	-	D2	-	D2	D2	-
HOST_AD1	-	-	-	D1	-	D1	D1	-
HOST_AD0	-	-	-	D0	-	D0	D0	-

Table 2-17. MDIN-380 Output Port Pin Configurations for Auxiliary Video Path (Continued)

When the pins are used as digital output pins, other functions of the pins cannot be used. It is recommended to select a pin configuration whose pin count is just matched to the required number of digital output pins so that the other pins cane be used as other fuctions. When no digital output pins are used, ***aux_out_data_mode*** should be set to 0 ('0000') to use the pins as other functions.

In each pin configuration, the mappings of bit names D(31:0) to video data are determined by ***aux_out_444*** and ***aux_out_mux_en*** as shown in Table 2-18. The aux digital output can send multiple video channels without using time-multiplexing on the multi-channel mode.

Mode / Bit Name	32-pin Mode		24-pin Mode		
	YC 4:2:2 16-bit	YC-muxed 8-bit	4:4:4 24-bit	YC 4:2:2 16-bit	YC-muxed 8-bit
D31	Y_A7	YC_A7			
D30	Y_A6	YC_A6			
D29	Y_A5	YC_A5			
D28	Y_A4	YC_A4			
D27	Y_A3	YC_A3			
D26	Y_A2	YC_A2			
D25	Y_A1	YC_A1			
D24	Y_A0	YC_A0			
D23	C_A7	YC_B7	R7 / Cr7	C7	YC_B7
D22	C_A6	YC_B6	R6 / Cr6	C6	YC_B6
D21	C_A5	YC_B5	R5 / Cr5	C5	YC_B5
D20	C_A4	YC_B4	R4 / Cr4	C4	YC_B4
D19	C_A3	YC_B3	R3 / Cr3	C3	YC_B3
D18	C_A2	YC_B2	R2 / Cr2	C2	YC_B2
D17	C_A1	YC_B1	R1 / Cr1	C1	YC_B1
D16	C_A0	YC_B0	R0 / Cr0	C0	YC_B0
D15	Y_B7	YC_C7	G7 / Y7	Y7	YC_A7
D14	Y_B6	YC_C6	G6 / Y6	Y6	YC_A6
D13	Y_B5	YC_C5	G5 / Y5	Y5	YC_A5
D12	Y_B4	YC_C4	G4 / Y4	Y4	YC_A4
D11	Y_B3	YC_C3	G3 / Y3	Y3	YC_A3
D10	Y_B2	YC_C2	G2 / Y2	Y2	YC_A2
D9	Y_B1	YC_C1	G1 / Y1	Y1	YC_A1
D8	Y_B0	YC_C0	G0 / Y0	Y0	YC_A0
D7	C_B7	YC_D7	B7 / Cb7	-	-
D6	C_B6	YC_D6	B6 / Cb6	-	-
D5	C_B5	YC_D5	B5 / Cb5	-	-
D4	C_B4	YC_D4	B4 / Cb4	-	-
D3	C_B3	YC_D3	B3 / Cb3	-	-
D2	C_B2	YC_D2	B2 / Cb2	-	-
D1	C_B1	YC_D1	B1 / Cb1	-	-
D0	C_B0	YC_D0	B0 / Cb0	-	-
aux_out_mux_en	'0'	'1'	'0'	'0'	'1'
aux_out_444	'0'	'0'	'1'	'0'	'0'

Table 2-18. MDIN-380 Output Port Bit Mapping for Auxiliary Video Path

Mode / Bit Name	16-pin Mode		8-pin Mode
	YC 4:2:2 16-bit	YC-muxed 8-bit	YC-muxed 8-bit
D15	Y7	YC_A7	
D14	Y6	YC_A6	
D13	Y5	YC_A5	
D12	Y4	YC_A4	
D11	Y3	YC_A3	
D10	Y2	YC_A2	
D9	Y1	YC_A1	
D8	Y0	YC_A0	
D7	C7	YC_B7	YC7
D6	C6	YC_B6	YC6
D5	C5	YC_B5	YC5
D4	C4	YC_B4	YC4
D3	C3	YC_B3	YC3
D2	C2	YC_B2	YC2
D1	C1	YC_B1	YC1
D0	C0	YC_B0	YC0
aux_out_mux_en	'0'	'1'	'1'
aux_out_444	'0'	'0'	'0'

Table 2-18. MDIN-380 Output Port Bit Mapping for Auxiliary Video Path (Continued)

2.18 Video Encoder (MDIN-325/380)

MDIN-325/380 has a video encoder to encode input video data into any of the NTSC and PAL video standards. Supported video output formats are NTSC-M/J/433, PAL-B/D/G/H/I/Nc, PAL-M/N. The video encoder can receive ITU-R656 format video data. The input to video encoder can be input port A, B and the output of aux video scaler. The sync and color burst timings can be tuned by register settings. The low pass filters for the luminance and chrominance are selectable. It also support brightness, contrast, saturation and sharpness control.

2.18.1 Features

- 525-line system and 625-line system
- Modulation format : NTSC-M/J/433, PAL-B/D/G/H/I/Nc, PAL-M/N
- Input data format : 4:2:2 Y/C-multiplexed (ITU-R656)
- Selecable input data : input port A, input port B and aux video scaler output
- CVBS and S-Video output
- Sync and color burst timing control
- Low pass filter for luminance and chrominance
- Brightness, contrast, saturation and sharpness control
- Auto and manual color modulation control

2.18.2 Internal block diagram

Figure 2-22 shows the internal block diagram of the video encoder. It selects one of 3 video input and decodes ITU-R656 format signal. The decoded video signals are converted to 4:4:4 format of YUV color space. There is a test pattern generation logic to output test patterns without input video. Timing generation logic makes all timing information signals and they are used to the other blocks. The video signals are low-pass filtered and the sync and color burst information are inserted to the video signal. Finally, color signal is modulated to generate composite video signal. The output of video encoder is CVBS, Y/C of S-video. They can be output through internal DACs.

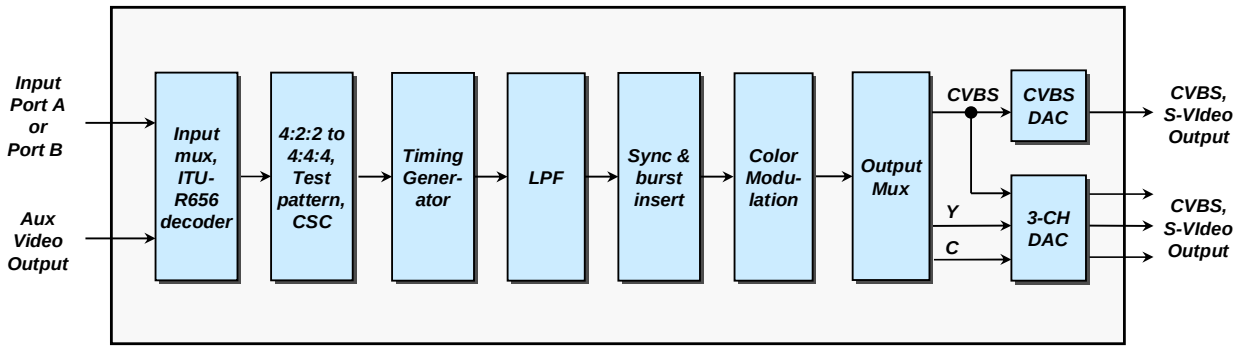


Figure 2-22. Internal block diagram of the video encoder

2.18.3 Input mux and ITU-R656 decoder

There are two input ports in the video encoder. One of them are connected to the input block. The other port is connected to the output of the aux video path. The input can be selected by ***venc_input_sel***. The input format should be ITU-R656 format with 27MHz video clock. If the aux video output is selected as an input to the video encoder, the aux video output needs to be configured to output ITU-R656 format video. The ITU-R656 decoder extracts video and sync informations from the input signal. The decoded video has 4:2:2 format.

2.18.4 4:2:2 to 4:4:4 conversion, Test pattern and CSC

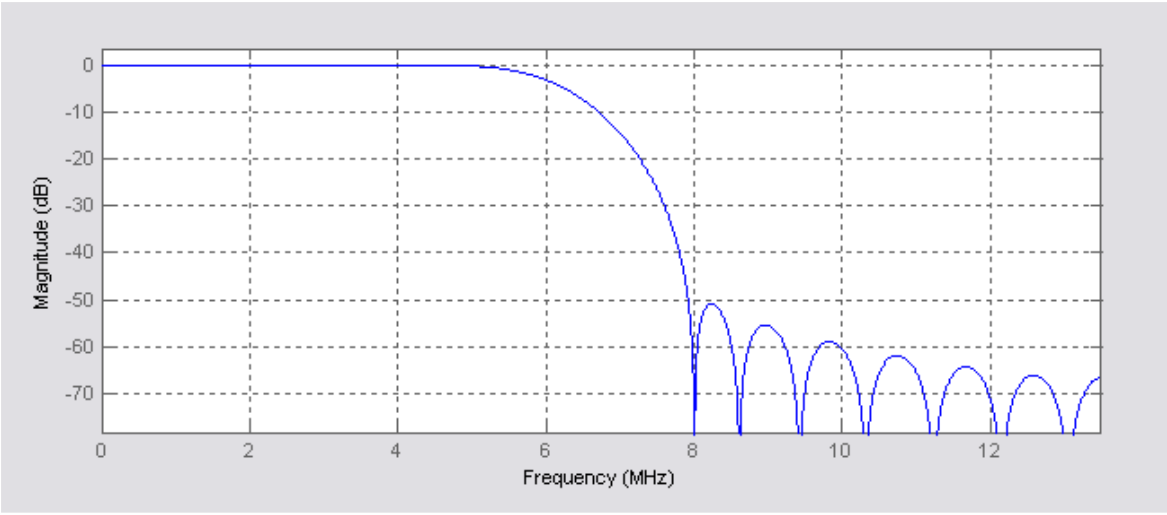
The 4:2:2 format video will be converted to 4:4:4 for color space conversion. And there is a test pattern generation logic for test. The test pattern has both 525 and 625-line systems and it can be selected by ***venc_test_pattern_pal***. There are 7 kinds of test patterns which can be selected by ***venc_test_pattern***.
The video data with YCbCr color space needs to be converted to YUV color space for modulation. This color space conversion is perform in the CSC block. The conversion rules depends on the input video system and it is controlled by ***venc_csc_sel***.

2.18.5 Timing generator

With the extracted sync informations from the ITU-R656 decoder, the timing generator creates various timing signals required on the other blocks. They are composite sync, pedestal, color burst, active video sync and etc. The selection between 525 and 625-line system can be done by ***venc_pal_sequence***. The reference of timing generation - edge of vactive - can be controlled by ***enc_vact_fall_ref***. And the polarity of field and period for timing reference can be controlled by ***venc_sync_reset_fid_pol*** and ***venc_sync_reset_vcnt***. The color burst timing can be controlled by ***venc_burst_start_val*** and ***venc_burst_end_val***. The timing between ITU-R656 input sync and the output sync can be controlled by ***venc_sync_del_val***.

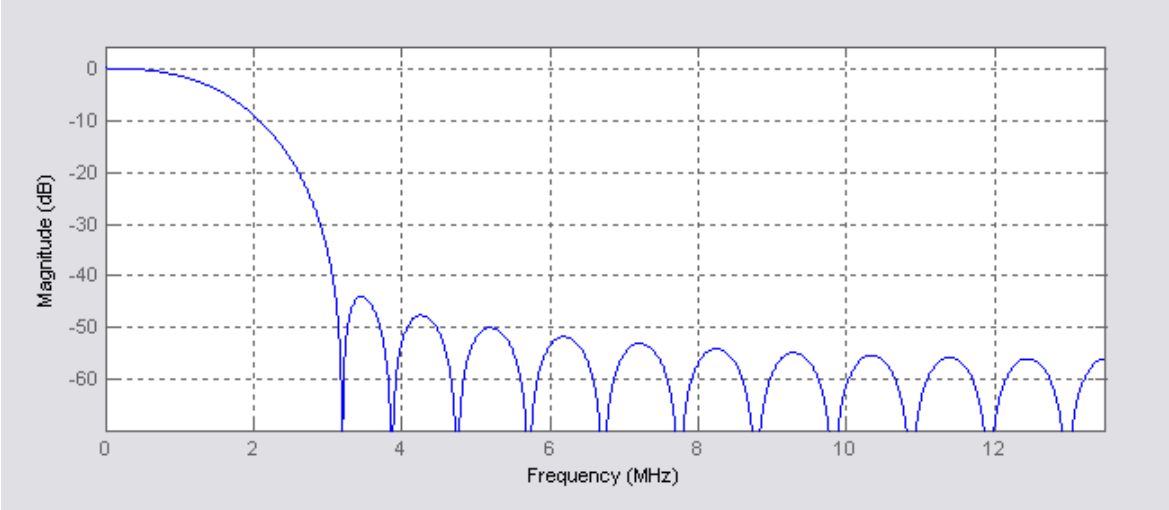
2.18.6 Low pass filter

The input video data needs to be filtered to remove high frequency components. Figure 2-23 shows the frequency response of 25-tap low pass filter for luminance, and Figure 2-24 shows the frequency response of 25-tap low pass filter for chrominance. They show the case of ***venc_y_lpf_sel*** = '001' and ***venc_c_lpf_sel*** = '001' which are recommended filter selections. The filtering for Y, U and V components can be enabled or disabled by ***venc_y_lpf_on***, ***venc_u_lpf_on*** and ***venc_v_lpf_on***, respectively.



Filter characteristics :
- 25-tab FIR filter
- 3dB bandwidth : 6.01 MHz
- Pass band ripple : 0.05 dB
- Stop band cut off frequency : 8.00 MHz
- Stop band attenuation : -50.88 dB

Figure 2-23. Frequency response of low pass filter for luminance



Filter characteristics :
- 25-tab FIR filter
- 3dB bandwidth : 1.363 MHz
- Pass band ripple : 0.15 dB
- Stop band cut off frequency : 3.20 MHz
- Stop band attenuation : -44.00 dB

Figure 2-24. Frequency response of low pass filter for chrominance

2.18.7 Sync and burst insert

The sync and pedestal are inserted to the luminance signal Y, and the color burst is inserted to the chrominance signal U. The slope of sync and pedestal can be controlled by ***venc_sync_slope***, ***venc_ped_slope***. The slope of color burst envelope can be controlled by ***venc_burst_slope***. For the blank and overscan period, the luminance can be forced to black level by setting ***venc_os_black_en*** and ***venc_blank_black_en***. And the level of sync and pedestal can be controlled by ***venc_y_sync_val*** and ***venc_pedestal_val***. The unit of ***venc_y_sync_val*** and ***venc_pedestal_val*** is 1/1024 of the full-scale DAC amplitude. ***venc_y_sync_val*** is based on zero level and ***venc_pedestal_val*** is based on the blank level. Amplitude of color burst can be controlled by ***venc_burst_val***.

2.18.8 Brightness, contrast, saturation and sharpness control

The brightness can be controlled by ***venc_y_offset***. The unit of this reigster is 1/1024 of the full-scale DAC amplitude. The reference value depends on the video system. The contrast can be cotrolled by ***venc_y_gain*** and the saturation can be controlled by ***venc_c_gain***. The register value which is corresponding to unity (1.0) is 512 for both ***venc_y_gain*** and ***venc_c_gain***. The sharpness can be controlled by ***venc_y_peak_gain*** and ***venc_y_peak_on***. The strength of sharpness has 7 steps.

2.18.9 Color modulation

To generate color subcarrier frequency, discrete timing oscillator (DTO) is used. The color modulation depends on the video system and can be selected by ***venc_mod_format***. If the specific color system is selected, all color modulation processes will be done automatically. In the manual modulation mode, ***venc_dto_value***, ***venc_pal_switch***, ***venc_max_frame*** and all the phase control registers should be set properly.

2.18.10 Output mux

The output of video encoder has 3 ports – CVBS, Y and C – which can be used generate CVBS (composite), Y (lumiance) and C (chrominance) signals. Any port can output all type of signals. The output signal can be selected by ***venc_cvbs_out_sel***, ***venc_y_out_sel*** and ***venc_c_out_sel*** respectively. The video encoder DAC port **VENC_OUT** is connected to the CVBS output port of the video encoder. The component DAC ports **DAC_BOUT**, **DAC_GOUT** and **DAC_ROUT** can be connected to the CVBS, Y and C output ports of the video encoder respectively if both ***dac_data_sel*** and ***dac_clk_sel*** are set to '01'.

2.19 Multi-channel Mode

MDIN-3xx can receive up to four channels of video and performs all the video processings in a frame-multiplexed fashion. The processed video channels can be output to the digital video port in multi-channel format (MDIN-325/380) or can be displayed in multi-window mode on a single video screen. On multi-channel mode, both the main and aux video paths are used so that some features which require dual-scaling such as PIP and dual video output cannot be used.
The multi-channel input video fields/frames are captured to the SDRAM via the main input video path, and then the main format converter reads back one field/frame from each channel by turns in a frame-multiplexed fashion with a higher frame rate. Each channel can be deinterlaced and/or scaled if necessary. The output of the main video path is transferred to the input of the aux video path and the video channels are stored again to the SDRAM. The aux format converter gathers the frames of the video channels and outputs them on the digital output port with a time-multiplexed format. Alternatively, the aux video path can display the video channels as multiple windows on a single screen.

2.19.1 Multi-channel Input

MDIN-3xx can receive up to 4 video streams on the multi-channel input mode but it has only two input video ports. The input video data should be time-multiplexed for more than two video channels. For multiplexing of multiple video streams, sync embedded digital formats such as BT.656 or BT.1120 should be used. Up to 4 video channels can be multiplexed on a single input port, but MDIN-3xx can handle up to 4 video channels at the same time. Example input configurations are (1) 4 channels on the port A or B, (2) 2 channels on the port A, 2 channels on the port B.

4 DTR(BT.656/BT.1120) decoders can be used to decode the multi-channel input video streams. The input channels are mapped to the DTR decoders as follows :

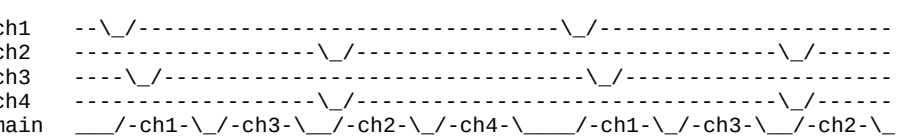
- A1 : clocked by **clk_a1** (mapped to channel1)
- A2 : clocked by **clk_a2** (mapped to channel2)
- B1 : clocked by **clk_b1** (mapped to channel3)
- B2 : clocked by **clk_b2** (mapped to channel4)

A1/A2 output data are clocked by **clk_m1** and captured to SDRAM, and A1/A2 paths are affected by the register settings for the port A. B1/B2 output data are clocked by **clk_m2** and captured to SDRAM, and B1/B2 paths are affected by the register settings for the port B.

For the identification of the channels in multiplexed streams, the channel ID can be inserted into the protection bits of SAV/EAV codes or the horizontal blanking data in the video stream. The channel ID of each input channel can be read from Input Channel ID Status Registers. MDIN-3xx supports the channel ID data in both the protection bits of SAV/EAV codes or horizontal blanking data.

2.19.2 Main Video Processing

If the multi-channel mode is enabled, the main sync generator generates one frame of synchronization signals according to Main Output Sync Control register settings after every field/frame of input streams is captured to the SDRAM. With synchronized to the frame synchronization signals, the captured fields/frames are read and processed by the deinterlacer and the main format converter. The processed output frames are transferred to the input of the aux video path. This frame-by-frame operation is repeated in frame-multiplexed fashion for all the input channels.
One frame period of the main video should be equal to or less than 1/Nc of the input field/frame period, where Nc is the number of input channels. For example, when four 480i@60fps channels are deinterlaced to 480p@60fps the frame rate of the main video timing should be equal or higher than 240fps so that all four channels can be processed within one output frame period. In this case, the input channels will be processed with the following timings :



Most register settings for the main video path are common to all channels - CSC, NR, deinterlacing, video enhancement filters, etc. Some features can be controlled independently for each channel - input offset, scaling factor, OSD. Some

features in the main input block – input CSC, front NR filter, front format converter – cannot be used on the multi-channel mode. When the horizontal video size needs to be reduced before the channels are written to the frame buffer, all the video channels can be decimated to a half size at the same time.
The main OSD can be enabled on or off for the video data to the aux path using ***osd_on_data_to_aux*** ('0' : off, '1' : on). The sprites of the main OSD are displayed on the all video channels if ***osd_mch_en*** is set to '0'. Otherwise, sprite0 and sprite4 are displayed on the channel1, sprite1 and sprite5 on the channel2, sprite2 and sprite6 on the channel3 and sprite3 and sprite7 on the channel4.

2.19.3 Video Output

The input of the aux video path is connected to the output of the main video path on multi-channel mode. Multiplexed video frames from the main video path are captured into the separate areas of SDRAM. Before capturing to the SDRAM, the aux scaler can adjust the input offset and size, and scale down the video if necessary. In this case, the offset and size parameters are common to all the channels. The aux scaler reads back all the stored channels simultaneously from the SDRAM and outputs them as a multi-channel format or displays them as a multi-window style.

1) Multi-channel output mode
The stored video frames are output to the digital port in a time-multiplexed digital format. All the channels are frame-synchronized and clocked by the same output clock. The digital format can be Y/C multiplexed format (BT.656) or Y/C separate format (BT.1120). Up to 4 channels can be multiplexed in a single video port. Up to four output ports are supported in the Y/C multiplexed format, and up to two output ports are supported in the Y/C separte format.
The channel ID can be inserted into the protection bits of SAV/EAV codes or the horizontal blanking data in the output video streams for the identification of the channels.
In the multi-channel output mode, the stored video frames in the frame buffer are transferred to the output ports without further video processing.

2) Multi-window output mode
The multiple channels are displayed as multiple windows on a single video screen. The multi-windowed video can be output to the digital port or the component DAC. The stored video in the frame buffer can be scaled up to be fit on the window. The color space conversion and OSD overlay can be performed before transferred to the output ports.
The supported window layouts are 2x2, 1x4, 4x1, 1x2 and 2x1.

3. Input and Output Interface

3.1 Digital Video Input and Output

3.1.1 Digital Video Input

MDIN-3xx receives various formats of video source including RGB/YCbCr 4:4:4, 8/10-bit Y/C multiplexed format and 16/20-bit Y/C seperated format. MDIN-3xx supports both interlaced and progressive scanned video up to 1920x1080i and 1920x1080p. It handles both standard and non-standard format videos. Figure 3-1 to Figure 3-3 shows various formats of video input. Refer to Table 2-1 for the input pin mapping in detail.
Unused clock input should be tied to ground.

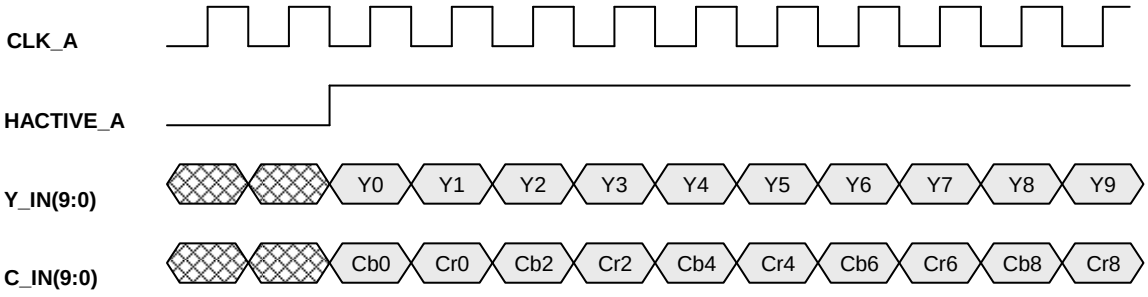
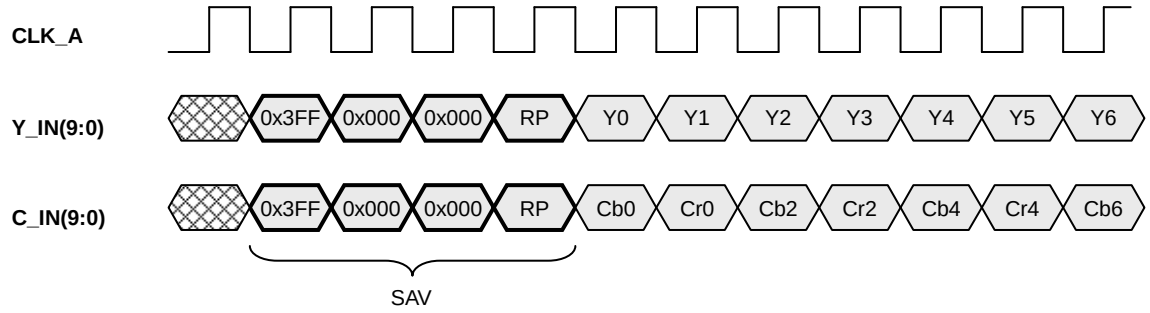
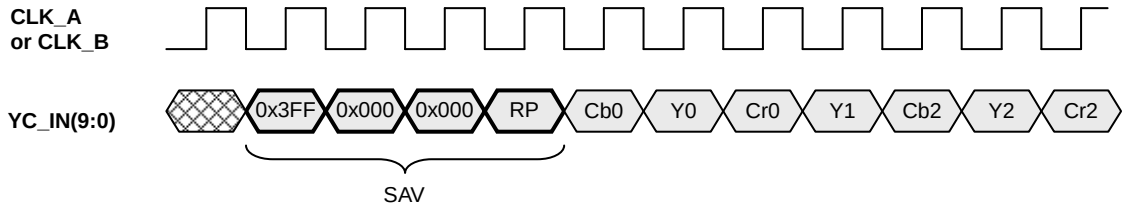


Figure 3-1. Video Input of YCbCr 4:2:2 (10-bit precision)



(a) 20-bit Y/C Seperated



(b) 10-bit Y/C Multiplexed

Figure 3-2. Video Input of Digital Format (10-bit precision)

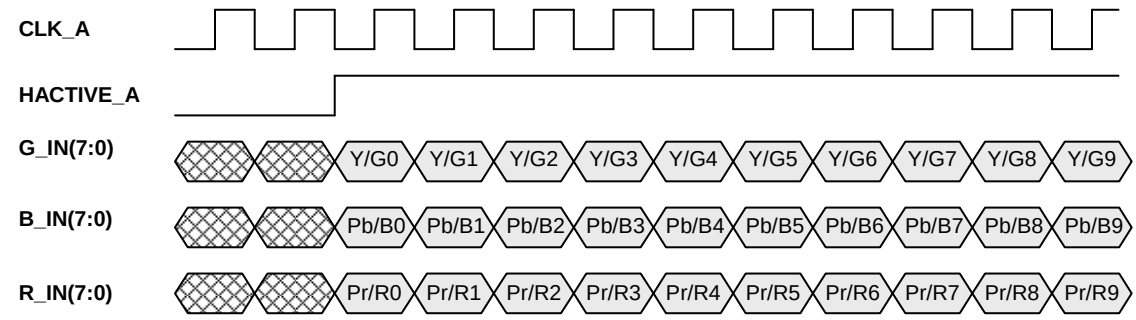


Figure 3-3. Video Input of RGB/YCbCr 4:4:4

3.1.2 Digital Video Output

MDIN-325/380 provides various formats of video output including RGB/YCbCr 4:4:4, YCbCr 4:2:2 and 20-bit Y/C seperated digital format. MDIN-325/380 provides progressive scanned video output up to 1920x1080p and also provides non-standard format with flexible sync generation capability. Figure 3-4 to Figure 3-7 shows the various format of video output.

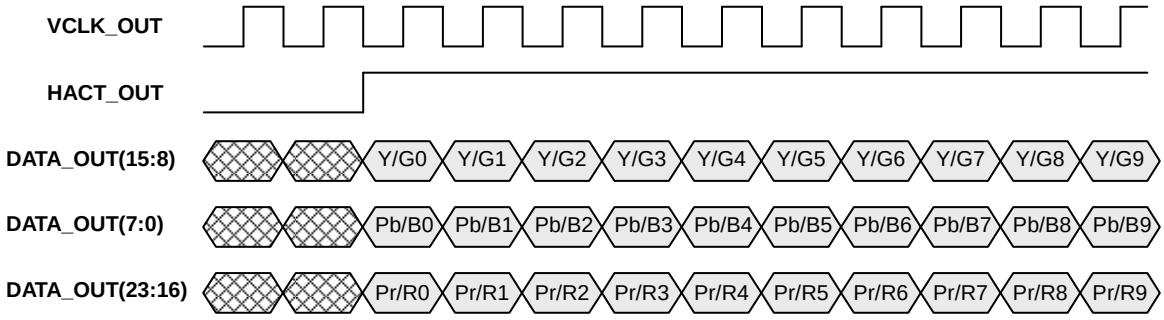


Figure 3-4. Video Output of RGB/YCbCr 4:4:4 (8-bit precision)

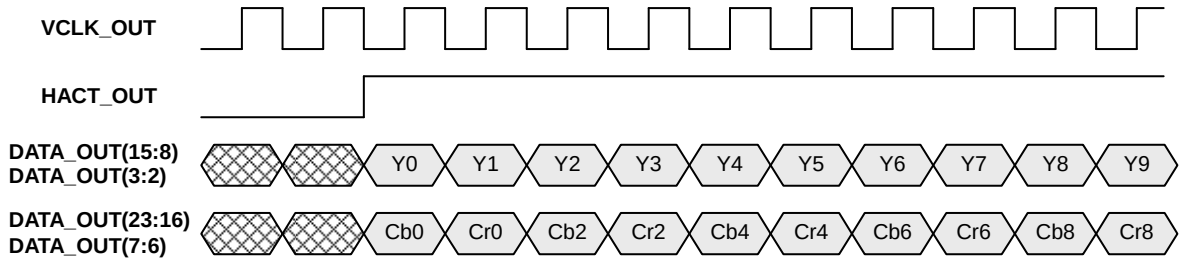
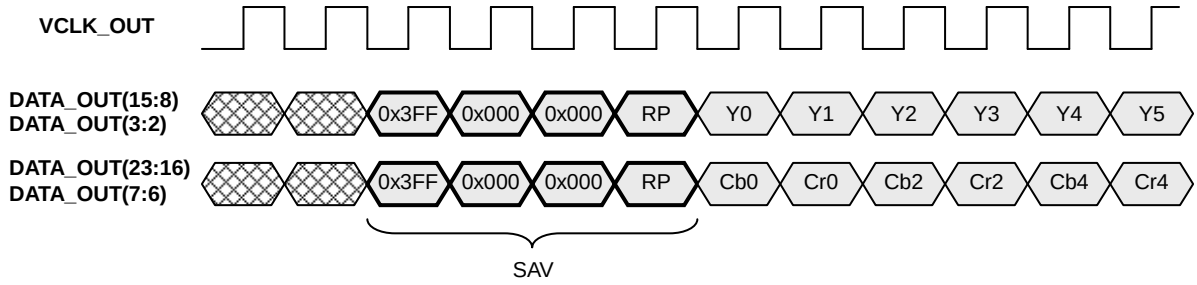
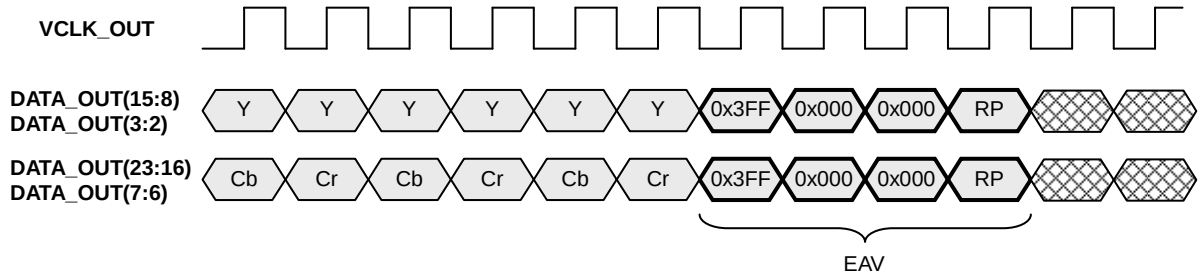


Figure 3-5. Video Output of YCbCr 4:2:2 (10-bit precision)



(a) Start of Active Video



(b) End of Active Video

Figure 3-6. Video Output of 20-bit Y/C Seperated Digital Format

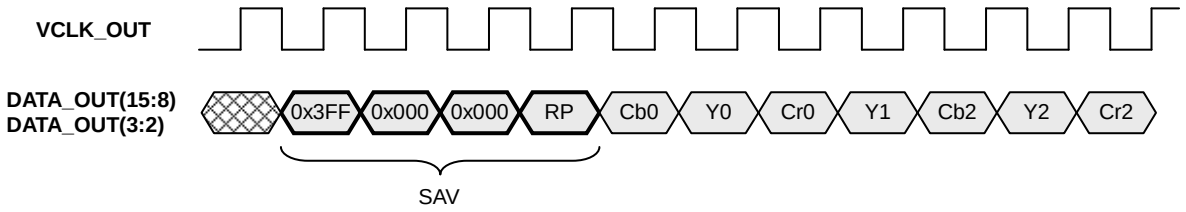


Figure 3-7. Video Output of 10-bit Y/C Multiplexed Digital Format

3.2 Host Bus Interface

MDIN-3xx provides host bus interfaces for accessing the internal registers and frame buffer memory of MDIN-3xx from an external host processor. Three types of host bus interfaces are provided – I^2C bus interface, parallel bus interface and PCI bus interface. The I^2C interface is provided by all the devices while the parallel interface and the PCI interface are provided by MDIN-380 only. The I^2C bus interface and the parallel bus interface support 16-bit mode and 8-bit mode. The PCI interface supports 32-bit mode only. The I^2C interface is always enabled by default. The parallel interface and the PCI interface of MDIN-380 can be enabled mutually exclusive, which can be selectable by the pin **HOST_PCI_BUS**. The two interfaces share several I/O pins. It should be noted that the I^2C interface and another host interface should not be accessed at the same time even though they are enabled together.

The host bus interfaces are configured by the host bus configuration pins as follows :

Signal Name	Description
I2C_MODE	I^2C bus 16-bit mode selection 'Low' : 8-bit mode 'High' : 16-bit mode (MDIN-2xx compatible)
HOST_BUS_MODE(3)	Operation mode selection of **HOST_DS_N** and **HOST_WR_N** 'Low' : **HOST_DS_N** = data strobe (DS_N), **HOST_WR_N** = write enable (WR_N) 'High' : **HOST_DS_N** = write strobe (WS_N), **HOST_WR_N** = read strobe (RS_N)
HOST_BUS_MODE(2)	'Low' : **HOST_RDY_N** = host bus ready output (READY_N) 'High' : **HOST_RDY_N** = host bus wait output (WAIT_N)
HOST_BUS_MODE(1)	Parallel bus 16-bit mode selection 'Low' : 8-bit mode is selected 'High' : 16-bit mode is selected
HOST_BUS_MODE(0)	Address/data bus mode selection 'Low' : address/data multiplexed using **HOST_ALE** 'High' : address/data separated
HOST_SYNC_BUS	Synchronous bus selection 'Low' : asynchronous bus mode 'High' : synchronous bus mode
HOST_PCI_BUS	PCI bus selection input 'Low' : parallel bus mode 'High' : PCI bus mode

Table 3-1. Host Bus Interface Configuration Pins

3.2.1 Address Space

The address space of MDIN-3xx is divided into 4 regions – host register region, local register region, HDMI register region, SDRAM region. The host register region includes host interface control registers, clock control registers, DAC control registers and GAC control registers. The local register region includes the local registers of each sub-block. The HDMI register region is used to access the local registers in the internal HDMI transmitter. The SDRAM region is used to access the internal SDRAM which is used for frame buffers and OSD bitmap buffers.

The address space of MDIN-3xx is accessed in different ways depending on the host interface type. Table 3-2 shows the address space of MDIN-3xx on the I^2C interfaces and the parallel interfaces. The page-addressing is used on these interface types to access the whole address space with limited address bits, i.e., each region is mapped to a page which can be selected by setting the page register ***page_sel***.
The registers in MDIN-3xx are 16-bit wide data, and each 16-bit register has its own index **Offset**. When a register is accessed via the host interface, the address value can be calculated from its **Offset** value. The mapping from **Offset** to a register address is different on host interface types. The register address on the 16-bit I^2C interface is the same as the **Offset** value of the register, since the 16-bit I^2C interface uses a special word-based addressing for 16-bit word data. The 8-bit I^2C interface and both the parallel interfaces use normal byte-based addressing, and the **Offset** value needs to be mutlplied by 2 to be used as the register address. On the 8-bit bus interface, the Bit0 of the address can be used to address the upper byte or the lower byte within 16-bit data. On the16-bit bus interface, the Bit0 of the address is ignored and it is assumed as '0', so the data access should be aligned on 16-bit boundaries.
The SDRAM region can be accessed by setting the row and bank address of an SDRAM word to the register ***cpu_base_addr*** at first. After then, any data within the specified SDRAM bank can be accessed with the address within the SDRAM region. The address consists of the column address in the bank and the offset in the 64-bit SDRAM word.

The least significant 3 bits are corresponding to the offset in the SDRAM word, and the other bits are corresponding to the column address.

Region \ Host I/F	16-bit I²C	8-bit I²C	16-bit parallel	8-bit parallel	*page_sel* value	Offset range
Host register region	0x000 ~ 0x3FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	'00'	0x000 ~ 0x3FF
Local register region	0x000 ~ 0x3FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	'01'	0x000 ~ 0x3FF
HDMI register region	0x000 ~ 0x3FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	'10'	0x000 ~ 0x3FF
SDRAM region	0x000 ~ 0x3FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	0x000 ~ 0x7FF	'11'	-
Page register address	0x400	0x800	0x800	0x800	-	-

1) The address range of each region is the address within the region.
2) Each region can be selected by setting ***page_sel*** before accessing the region.
3) The address for the 16-bit I²C interface is word-based, others are byte-based.

Table 3-2. Address Space of MDIN-3xx by Host Interface Type

Table 3-3 shows the address space of MDIN-3xx on the PCI interface. All the regions in the address space can be accessed by direct addressing without using the page register. The PCI interface uses byte-based addressing, but all the 16-bit registers are accessed as 32-bit wide data with dummy upper 16 bits on each register while the SDRAM data are accessed as normal 32-bit wide data. The **Offset** value of the register needs to be multiplied by 4, and then the base address of the region is added to get the PCI address of the register. When a register is read from the PCI interface, the upper 16 bits are always 0x0000 and only the lower 16 bits are valid. When a register is written to the PCI interface, the upper 16 bits are ignored and the lower 16 bits are written to the corresponding register bits.
The SDRAM region is accessed without using ***cpu_base_addr***. The row, bank and column addresses are included in the PCI address bits as shown in the table. The least significant 3 bits of the address are corresponding to the offset in the SDRAM word.

Region	PCI address range	Offset range	SDRAM row + bank address	SDRAM column address
Host register region	0x00000000 ~ 0x00000FFF	0x000 ~ 0x3FF	-	-
Local register region	0x00800000 ~ 0x00800FFF	0x000 ~ 0x3FF	-	-
HDMI register region	0x01000000 ~ 0x01000FFF	0x000 ~ 0x3FF	-	-
SDRAM region (256Mb)	0x02000000 ~ 0x03FFFFFF	-	A(24:11)	A(10:3)
SDRAM region (128Mb)	0x02000000 ~ 0x02FFFFFF	-	A(23:10)	A(9:3)

1) All addresses are based on the whole address space, not the address within a range.
All regions can be accessed directly by PCI address without using the page register.
2) The addresses are byte-based, and they correspond to the offsets from the PCI base address.
3) All the 16-bit registers are treated as 32-bit data with dummy upper 16 bits.
3) The address range of SDRAM region includes the row, bank and column addresses of SDRAM.

Table 3-3. Address space of MDIN-3xx on PCI mode

3.2.2 I²C Bus Interface

All the MDIN-3xx devices provide serial I²C bus interface with 16-bit mode and 8-bit mode. The 16-bit I²C mode is fully compatible with the I²C bus interface of the MDIN-2xx devices. In this mode, I²C read and write operations are performed with 16-bit word data and the register address should be given as word address. In the 8-bit I²C mode, byte data and byte address are used.
The I²C bus interface consists of a clock pin (**SCL**) and a bi-directional data pin (**SDA**). MDIN-3xx acts as a slave device on I²C bus. Data received or transmitted on **SDA** line should be stable when **SCL** is high and it may change only when SCL is low. If **SDA** changes its state while **SCL** is high, it is interpreted as a START or a STOP condition. As defined in the I²C specification, the START condition is defined as a high-to-low transition on **SDA** while **SCL** is high. The STOP condition is defined as a low-to-high transition on **SDA** while **SCL** is high. Figure 3-8 shows the waveforms of these conditions. The first eight bit of data transferred after a START condition comprises a 7-bit slave address and one R/W bit. The R/W bit indicates the direction of data transfer. When the R/W bit is high, it is to read data from a slave device. When the R/W bit is low, it is to write data to a slave device. If the received slave address matches the address of the device address of MDIN-3xx, MDIN-3xx acknowledges it by setting **SDA** signal low on the ninth **SCL** pulse as shown in Figure 3-9. The I²C slave device address of MDIN-3xx is shown in Figure 3-10. The LSB of the address is determined by **TEST_MODE(0)** input pin, and one of two addresses can be selected for MDIN-3xx.

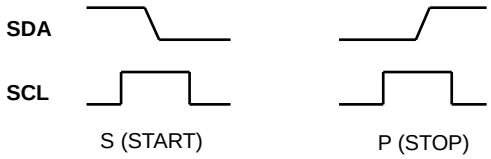


Figure 3-8. I²C START, STOP Condition

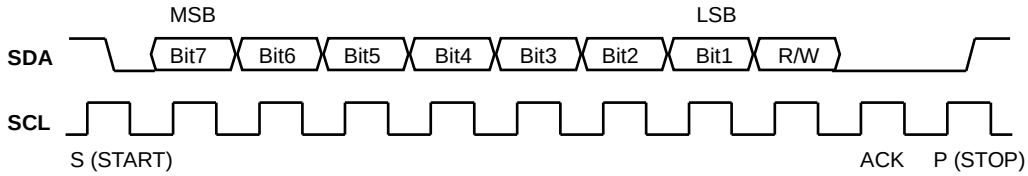


Figure 3-9. I²C Byte Transfer Cycle

Bit number on SDA	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Slave Address	A6 (MSB)	A5	A4	A3	A2	A1	A0
Value	1	1	0	1	1	1	0 or 1

Figure 3-10. I²C Slave Device Address of MDIN-3xx

3.2.2.1 16-bit I²C Mode Operation

For the 16-bit I²C mode, the **I2C_MODE** pin should set to high or be left open.
The I²C read and write sequences on the 16-bit I²C mode are shown in Figure 3-11 and Figure 3-12. For an I²C read operation, a repeated START condition which is followed by a slave address byte with R/W bit setting to high (sequence number 5 and 6 in Figure 3-11(a)) should be sent to reverse the direction of **SDA** line. For an I²C write operation, a repeated START condition (sequence number 5 and 6 in Figure 3-11(b)) is optional. Data can be written by sending data bytes just after address bytes without sending a repeated START condition.
The data width and address width on the 16-bit I²C mode are 16 bits, so that I²C byte transfer cycles of the host address and the read/write data should be performed in pairs. The host address A(10:0) is an address for 16-bit word registers,

so the register offset values can be used without any conversion. You may access successive registers in burst as many as required because there is no limit in the number of read/write data words (2-byte). In this burst mode, the internal register address is automatically incremented.
Host address A(10:8) is set by bit(2:0) of Host Address Byte 1 in the I^2C sequence number (3) and host address A(7:0) is set by bit(7:0) of Host Address Byte 0 in the I^2C sequence number (4) as shown in Figure 3-1.

Number	Phase	Transmitter	Receiver	Remarks
1	START condition (S)	Host CPU	MDIN	
2	I^2C Slave Address Byte	Host CPU	MDIN	R/W bit = Low
3	Host Address Byte 1 (MSB)	Host CPU	MDIN	A(10:8)
4	Host Address Byte 0 (LSB)	Host CPU	MDIN	A(7:0)
5	Repeated START condition (Sr)	Host CPU	MDIN	
6	I^2C Slave Address Byte	Host CPU	MDIN	R/W bit = High
7	Read Data Byte 1 (MSB)	MDIN-3xx	Host CPU	D(15:8)
8	Read Data Byte 0 (LSB)	MDIN-3xx	Host CPU	D(7:0)
9	Optional repeat of (7) & (8) in pairs	MDIN-3xx	Host CPU	
10	STOP condition (P)	Host CPU	MDIN	

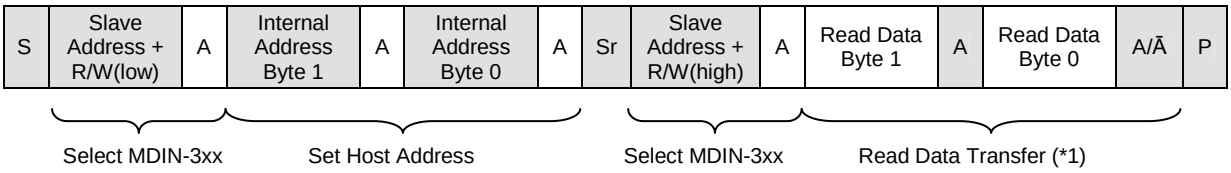
(a) 16-bit I^2C Read Sequence

Number	Phase	Transmitter	Receiver	Remarks
1	START condition (S)	Host CPU	MDIN	
2	I^2C Slave Address Byte	Host CPU	MDIN	R/W bit = Low
3	Host Address Byte 1	Host CPU	MDIN	A(10:8)
4	Host Address Byte 0	Host CPU	MDIN	A(7:0)
5 (*)	Repeated START condition (Sr)	Host CPU	MDIN	
6 (*)	I^2C Slave Address Byte	Host CPU	MDIN	R/W bit = Low
7	Write Data Byte 1 (MSB)	Host CPU	MDIN	D(15:8)
8	Write Data Byte 0 (LSB)	Host CPU	MDIN	D(7:0)
9	Optional repeat of (7) & (8) in pairs	Host CPU	MDIN	
10	STOP condition (P)	Host CPU	MDIN	

Note (*) : Phase 5 & 6 are optional for write sequence. Data can be written by sending data bytes just after address bytes.

(b) 16-bit I^2C Write Sequence

Figure 3-11. 16-bit I^2C Read and Write Sequence



(a) 16-bit I^{2}C Read Cycle

Note (*) : Repeated start condition is optional for write. Data can be written by sending data bytes just after address bytes.

(b) 16-bit I^{2}C Write Cycle

From Host CPU to MDIN-3xx
From MDIN-3xx to Host CPU

S = START condition
A = Acknowledge (SDA low)
Ā = Not acknowledge (SDA high)
P = STOP condition
Sr = Repeated START condition
Note : (*1) read cycles may be repeated with sending an acknowledge, and a cycle is terminated with sending a not acknowledge.
Note : (*2) write cycles may be repeated.

Figure 3-12. 16-bit I^{2}C Read and Write Cycle

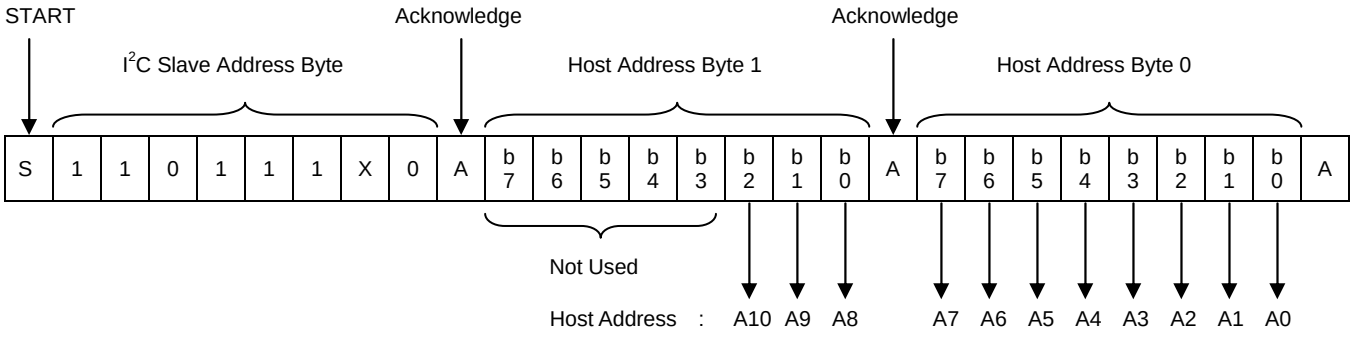


Figure 3-13. Host Address on 16-bit I^{2}C Mode

3.2.2.2 8-bit I²C Mode Operation

For the 8-bit I²C mode, the **I2C_MODE** pin should set to low.
The I²C read and write sequences on the 8-bit I²C mode are shown in Figure 3-14 and Figure 3-15. For an I²C read operation, a repeated START condition which is followed by a slave address byte with R/W bit setting to high (sequence number 5 and 6 in Figure 3-14(a)) should be sent to reverse the direction of **SDA** line.
The data width on the 8-bit I²C mode is 8 bits, so that I²C byte transfer cycles of the read/write data can be performed in bytes. The address consists of two bytes and the MSB is transferred frist. Since the host address is a byte address and the MDIN-3xx registers are 16-bit words, the offset value of the registers should be converted to a byte address by doubling the offset value. If the bit(0) of the address is '0', the lower byte of a register is accessed. Otherwise, the upper byte is accessed. You may access successive bytes in burst as many as required because there is no limit in the number of read/write data bytes. In this burst mode, the internal register address is automatically incremented.
Host address A(11:8) is set by bit(3:0) of Host Address Byte 1 in the I²C sequence number (3) and host address A(7:0) is set by bit(7:0) of Host Address Byte 0 in the I²C sequence number (4) as shown in Figure 3-14.

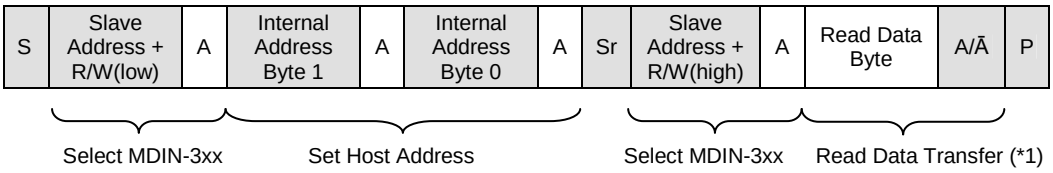
Number	Phase	Transmitter	Receiver	Remarks
1	START condition (S)	Host CPU	MDIN	
2	I²C Slave Address Byte	Host CPU	MDIN	R/W bit = Low
3	Host Address Byte 1 (MSB)	Host CPU	MDIN	A(11:8)
4	Host Address Byte 0 (LSB)	Host CPU	MDIN	A(7:0)
5	Repeated START condition (Sr)	Host CPU	MDIN	
6	I²C Slave Address Byte	Host CPU	MDIN	R/W bit = High
7	Read Data Byte	MDIN-3xx	Host CPU	D(7:0)
8	Optional repeat of (7)	MDIN-3xx	Host CPU	
9	STOP condition (P)	Host CPU	MDIN	

(a) 8-bit I²C Read Sequence

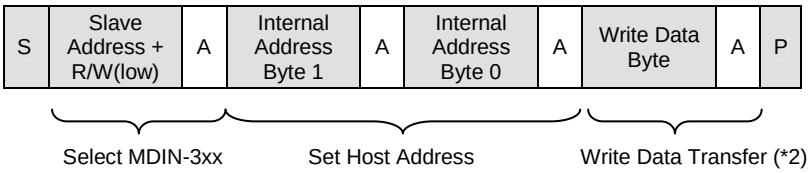
Number	Phase	Transmitter	Receiver	Remarks
1	START condition (S)	Host CPU	MDIN	
2	I²C Slave Address Byte	Host CPU	MDIN	R/W bit = Low
3	Host Address Byte 1	Host CPU	MDIN	A(11:8)
4	Host Address Byte 0	Host CPU	MDIN	A(7:0)
5	Write Data Byte	Host CPU	MDIN	D(7:0)
6	Optional repeat of (7)	Host CPU	MDIN	
7	STOP condition (P)	Host CPU	MDIN	

(b) 8-bit I²C Write Sequence

Figure 3-14. 8-bit I²C Read and Write Sequence



(a) 8-bit I^{2}C Read Cycle



(b) 8-bit I^{2}C Write Cycle

From Host CPU to MDIN-3xx
From MDIN-3xx to Host CPU

S = START condition
A = Acknowledge (SDA low)
Ā = Not acknowledge (SDA high)
P = STOP condition
Sr = Repeated START condition
Note : (*1) read cycles may be repeated with sending an acknowledge, and a cycle is terminated with sending a not acknowledge.
Note : (*2) write cycles may be repeated.

Figure 3-15. 8-bit I^{2}C Read and Write Cycle

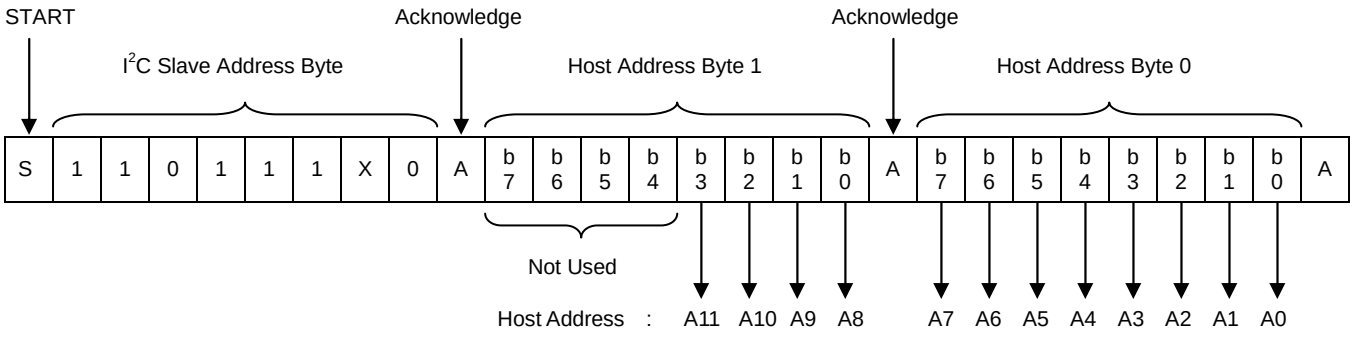


Figure 3-16. Host Address on 8-bit I^{2}C Mode

3.2.3 Parallel Bus Interface (MDIN-380)

MDIN-380 provides parallel bus interface with 16-bit mode and 8-bit mode. The parallel bus interface can be operated on asynchronous or synchronous mode, and the address bus and the data bus can be multiplexed or separated. These features are configured using the host bus configuration pins. Besides, the bus operation parameters can be changed from the default settings using the register ***host_pin_config*** via the I^2C bus interface so that various types of host controllers can be supported.

3.2.3.1 Address/Data Pin Assignment

The address/data pins **HOST_AD(19:0)** can be assigned to the internal address signal A(11:0) and the data signal D(15:0) as shown in Table 3-4 to support various address/data bus configurations. The register bits ***host_adata_config*** should be changed via the I^2C bus interface to select the configurations except for the default setting.

host_adata_config	Address/Data Pin Assignment		
	A(11:0)	D(15:8)	D(7:0)
'00' (default)	HOST_AD(19:16, 7:0)	N.A.	HOST_AD(7:0)
'01'	HOST_AD(11:0)	N.A.	HOST_AD(7:0)
'10'	HOST_AD(19:8)	N.A.	HOST_AD(15:8)
'11'	HOST_AD(19:8)	N.A.	HOST_AD(15:8)

(a) 8-bit Data Width, Multiplexed Mode (**HOST_BUS_MODE(1:0)** = '00')

host_adata_config	Address/Data Pin Assignment		
	A(11:0)	D(15:8)	D(7:0)
'00' (default)	HOST_AD(19:8)	N.A.	HOST_AD(7:0)
'01'	HOST_AD(19:16, 7:0)	N.A.	HOST_AD(15:8)
'10'	HOST_AD(11:0)	N.A.	HOST_AD(19:12)
'11'	HOST_AD(11:0)	N.A.	HOST_AD(19:12)

(b) 8-bit Data Width, Separated Mode (**HOST_BUS_MODE(1:0)** = '01')

host_adata_config	Address/Data Pin Assignment		
	A(11:0)	D(15:8)	D(7:0)
'00' (default)	HOST_AD(19:16, 7:0)	HOST_AD(15:8)	HOST_AD(7:0)
'01'	HOST_AD(19:16, 7:0)	HOST_AD(16:9)	HOST_AD(8:1)
'10'	HOST_AD(11:0)	HOST_AD(15:8)	HOST_AD(7:0)
'11'	HOST_AD(11:0)	HOST_AD(16:9)	HOST_AD(8:1)

(c) 16-bit Data Width, Multiplexed Mode (**HOST_BUS_MODE(1:0)** = '10')

Table 3-4. Address/Data Pin Assignment

On the multiplexed mode, the address signal is latched on the rising edge of **HOST_ALE** by default. The polarity of **HOST_ALE** can be inverted by setting ***host_ale_pol*** to '1'.

3.2.3.2 Bus Operation Mode

The host interface of MDIN-380 stays disabled as long as **HOST_CS_N** is not asserted. The polarity of **HOST_CS_N** is

active low by default, and it can be inverted by setting ***host_cs_n_pol*** to '1'. **HOST_DS_N**, **HOST_WR_N** and **HOST_RDY_N** are multi-function signals, and their functions can be configured by **HOST_BUS_MODE(3:2)** as shown in Table 3-5. The polarities of these signals are active low by default, and they can be inverted by setting the corresponding bits of ***host_pin_config***. The operation modes of these signals are identical for both the asynchronous and synchronous bus modes.

HOST_BUS_MODE (3:2)	Pin Functions		
	HOST_DS_N	**HOST_WR_N**	**HOST_RDY_N**
'00'	Data Strobe Input	Write Enable Input	Ready Output
'01'	Data Strobe Input	Write Enable Input	Wait Output
'10'	Write Strobe Input	Read Strobe Input	Ready Output
'11'	Write Strobe Input	Read Strobe Input	Wait Output

Note : All the signals are active low. The polarities of **HOST_DS_N**, **HOST_WR_N**, **HOST_RDY_N** can be inverted by setting the corresponding bits of ***host_pin_config***.

Table 3-5. Functions of **HOST_DS_N**, **HOST_WR_N**, **HOST_RDY_N**

The typical write and read bus cycles for the asynchronous bus mode are shown in Figure 3-17 to 3-20.

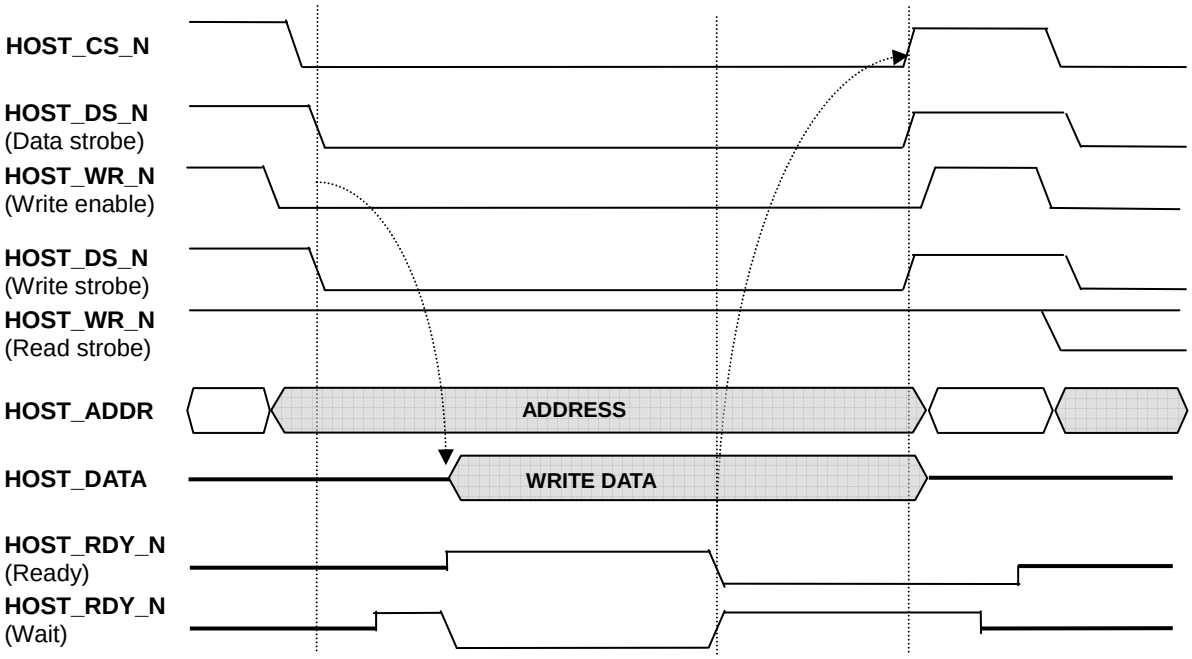


Figure 3-17. Address/Data Separated Parellel Bus Write Cycle

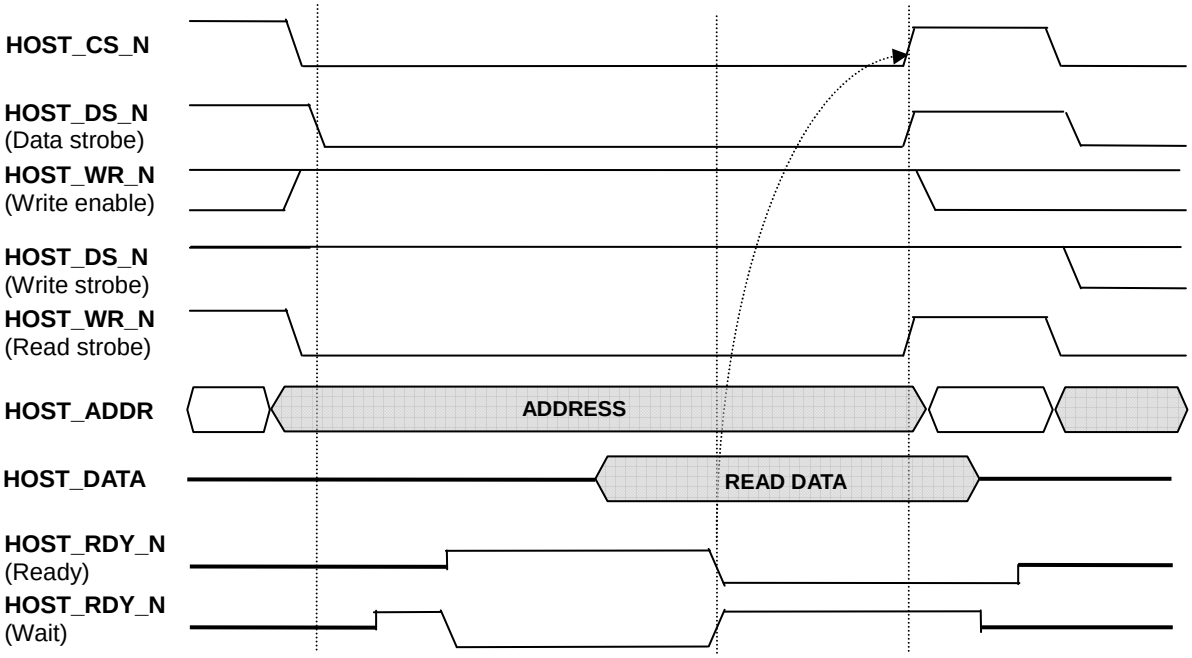


Figure 3-18. Address/Data Separated Parellel Bus Read Cycle

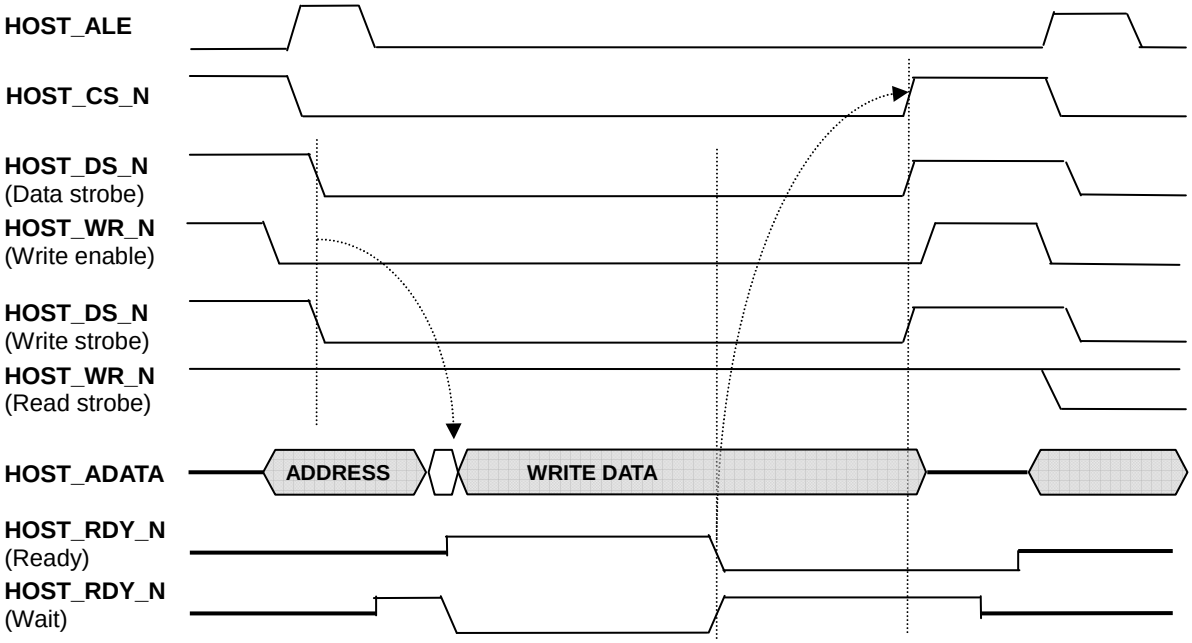


Figure 3-19. Address/Data Multiplexed Parellel Bus Write Cycle

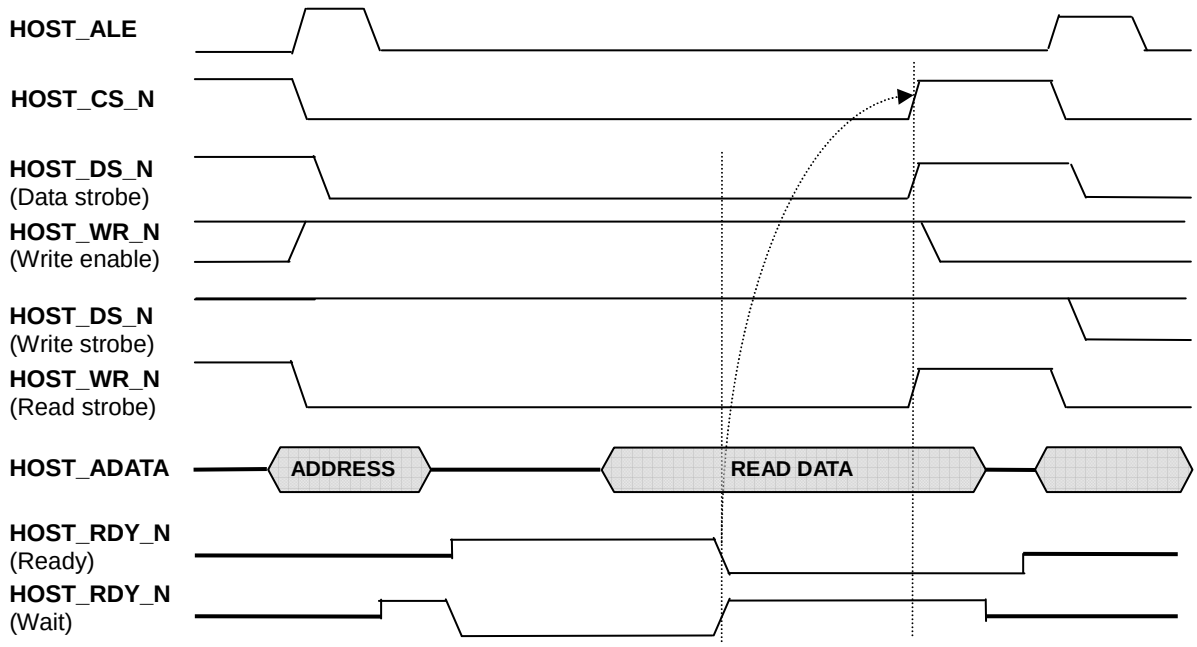


Figure 3-20. Address/Data Multiplexed Parellel Bus Read Cycle

3.2.3.3 DMA Operation

When higher data transfer rate is required for the larger amount of OSD bitmap data, data can be transferred using the DMA controller of MDIN-380. The following steps are recommended to access SDRAM using the DMA operation :

(1) Setup the DMA related CPU registers. (DMA size registers, source/destination address registers and etc.)
(2) Set row and bank address of SDRAM base address by the register ***dma_base_addr***.
(3) Set byte offset from ***dma_base_addr*** to start the DMA by the register ***dma_offset_addr***.
(4) Set DMA transfer count by the register ***dma_sb_size*** or ***dma_mb_size*** according to DMA mode.
(5) Set ***dma_start*** to '1' with setting ***dma_mb_en***, ***dma_rd*** and ***dma_single*** to appropriate values.

DMA transfer cycles are shown in Figure 3-17. The timing for deasserting **DREQ_N** to terminate DMA transfer is programmable by ***dma_ack_count***. It may be from trailing edge of (N-1)th **DACK_N** to rising edge of last Nth **DACK_N**.

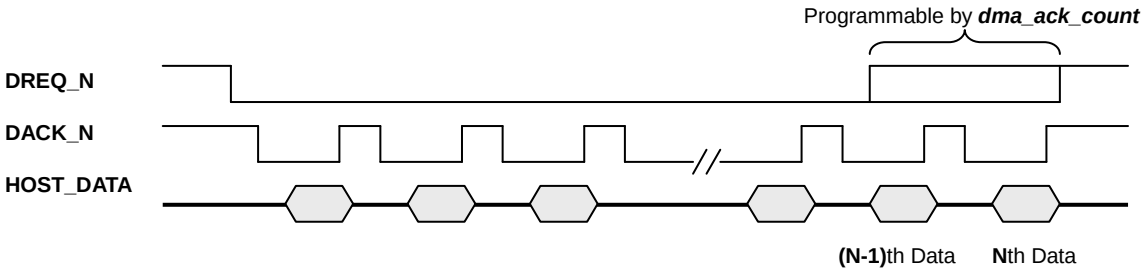


Figure 3-21. DMA Transfer Cycles

3.2.4 PCI Bus Interface (MDIN-380)

The PCI interface of MDIN-380 supports version 2.2 of the PCI bus specification at 33MHz. It can a slave device on the PCI bus. It responds only to memory bus transactions. I/O PCI transactions are ignored. Internal registers are accessed as memory mapped registers, the internal SDRAM region is accessed directly via the PCI interface. PCI Configuration registers are implemented as required in the PCI specification.

3.2.4.1 Configuration

The PCI Specification describes the Configuration Space used by the system to identify and configure each device attached to the bus. The Configuration Space contains a 256-byte address space for each device and has sufficient information for the system to identify the capabilities of the device. The software executes a special configuration cycle to allow access to these registers.
MDIN-380 implements a standard PCI configuration space, accessed in configuration cycles in which IDSEL is also asserted. Fields such as Vendor ID, Subvendor ID are included in the configuration space. The PCI configuration registers within MDIN-380 are hard-coded except for some reprogrammable registers, so they cannot be loaded from an external ROM.
MDIN-380 supports one address space, which includes all the register regions and the SDRAM region. The addressing range of the address space is 64 Mbytes. The address ranges of the regions are explained in Table 3-3.

3.2.4.2 Register Access

The internal registers in MDIN-380 have 16-bit data width and they are written or read via the lower 16 bits of the PCI data bus. The 16 bits of a register are mapped to the lower 16 bits of 32-bit PCI data bus and the upper 16 bits of 32-bit PCI data bus are not used. So the address range of each register region occupies double the size of the parallel bus interfaces. Each register region has the address range of 8 Mbytes but only the lowest 4 kbytes are used.

3.2.4.3 SDRAM Access

The SDRAM region is located at the upper 32 Mbyte area. When an SDRAM word is accessed, byte ordering can be changed using the registers ***pci_wr_byte_swap***, ***pci_wr_32bit_swap***, ***pci_rd_byte_swap*** and ***pci_rd_32bit_swap***.

3.2.4.4 Interrupts

All interrupts are mapped to the INTA# pin.

3.2.5 SDRAM Addressing

The construction of the internal SDRAM of MDIN-3xx is shown in Figure 3-22.

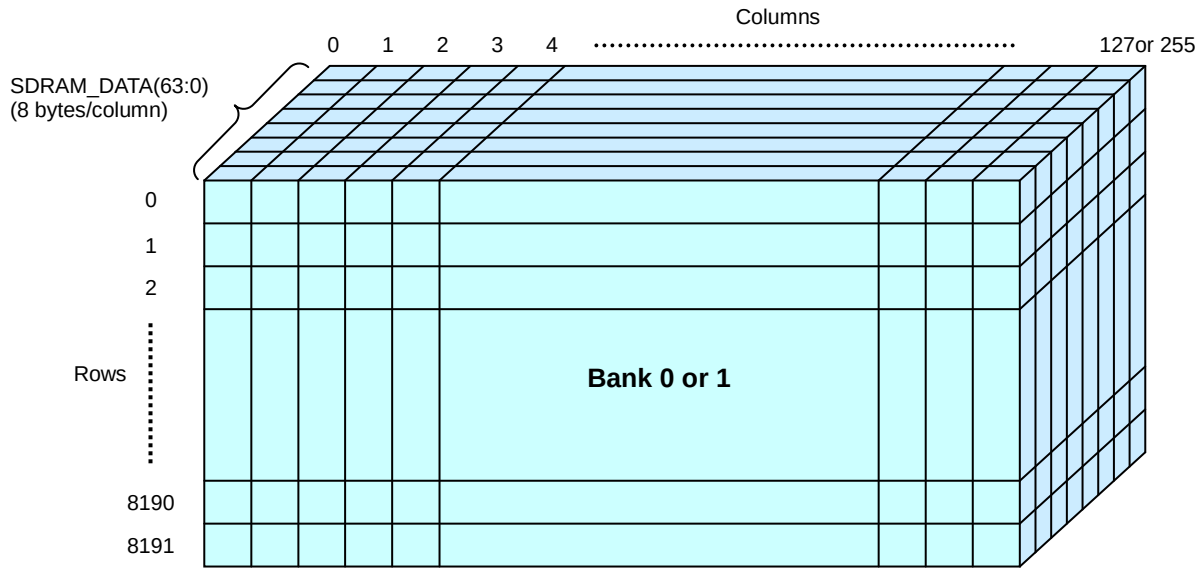


Figure 3-22. Construction of Internal SDRAM

Accessing the SDRAM data on the I^2C interface or the parallel interface requires two steps. At first, the row address and bank address should be set to the register ***cpu_base_addr***. And then, SDRAM data can be accessed with specifying the column address and the word/byte offset as the address.
On the PCI interface, the whole SDRAM region is mapped to a specific PCI memory range and any SDRAM data can be directly accessed by specifying the row, bank, column address and the byte offset to the PCI address bus.

The internal memory data bus of MDIN-3xx is 64-bit wide regardless of the host interface type, and all the explanations for SDRAM data structure are based on 64-bit wide SDRAM words. Data construction of video pixel data in the frame buffer memory is shown in Figure 3-23. The pixel 0 represents the left-most pixel in successive pixels.

SDRAM_DATA(63:0)	63 – 56	55 – 48	47 – 40	39 – 32	31 – 24	23 – 16	15 – 8	7 – 0
Pixel Data	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	Pixel 0

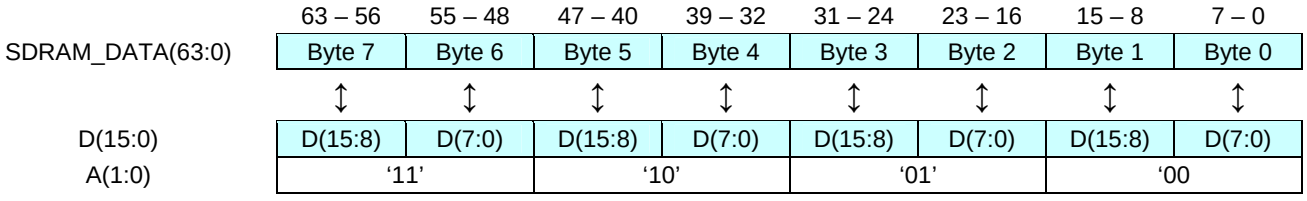
(a) 8-bit Precision Mode

SDRAM_DATA(63:0)	63 – 60	59 – 50	49 – 40	39 – 30	29 – 20	19 – 10	9 – 0
Pixel Data	-	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	Pixel 0

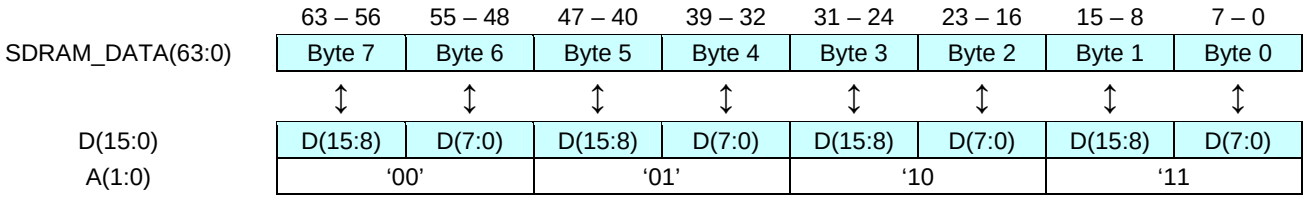
(b) 10-bit Precision Mode

Figure 3-23. Video Pixel Data in Frame Buffer Memory

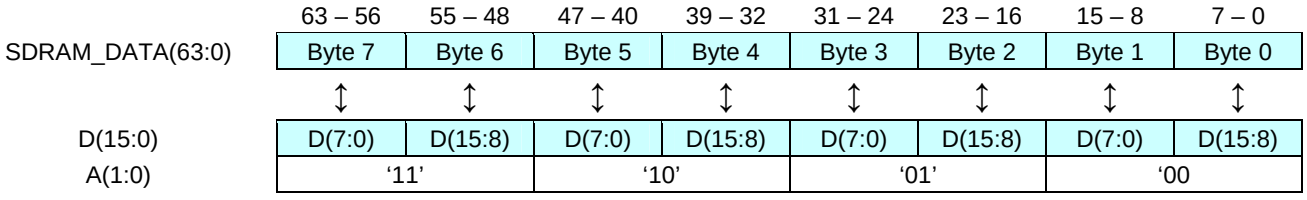
For a flexible interface to any external host processor, MDIN-3xx supports programmable endian conversion in accessing SDRAM data. The operation of endian conversions depends on the host interface type as shown in Figure 3-24 to Figure 3-27. The endian conversion on the I^2C interface and the parallel interface is controlled by the register ***endian_swap***, and the endian conversion on the PCI interface is controlled by the registers ***pci_wr_endian_swap*** and ***pci_rd_endian_swap***.



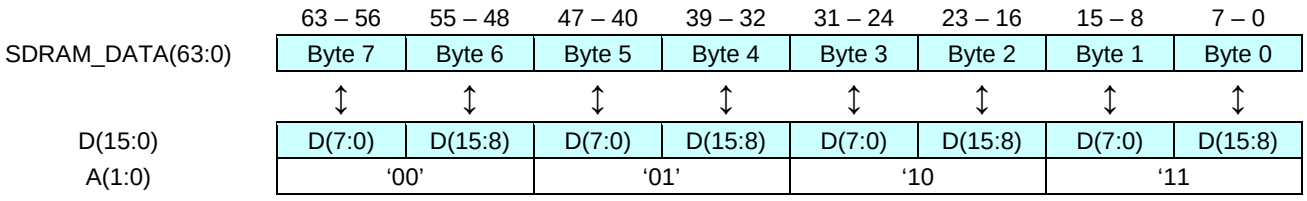
(a) ***endian_swap*** = ‘00’



(b) ***endian_swap*** = ‘01’



(c) ***endian_swap*** = ‘10’



(d) ***endian_swap*** = ‘11’

Figure 3-24. Endian Conversion of **SDRAM_DATA** on 16-bit I^2C Interface
(Note: A(1:0) is word-based address)

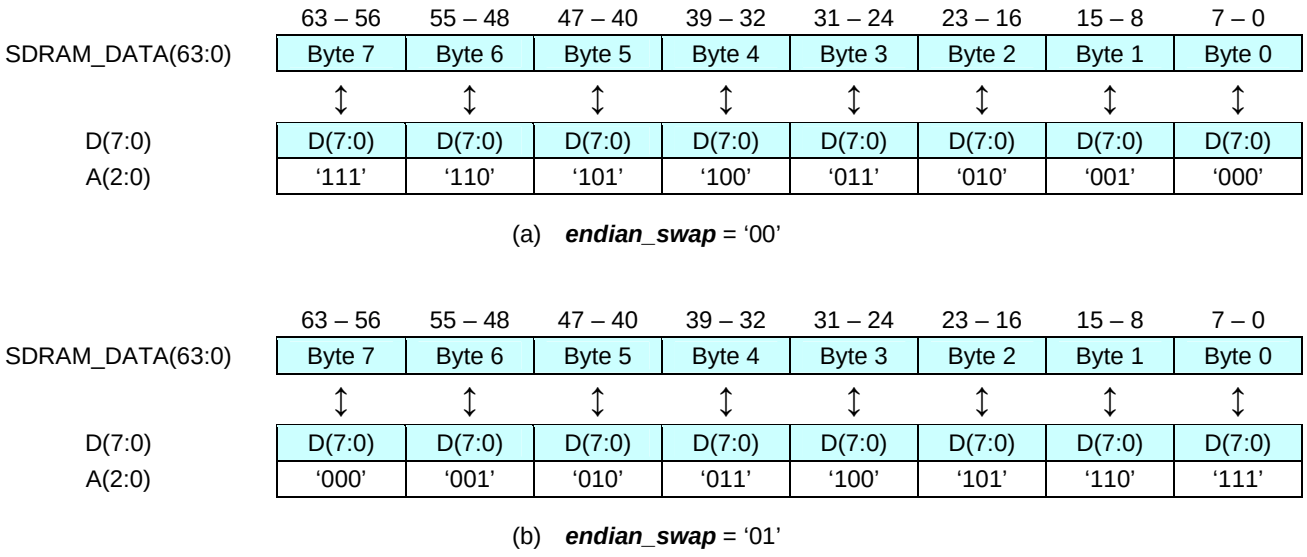
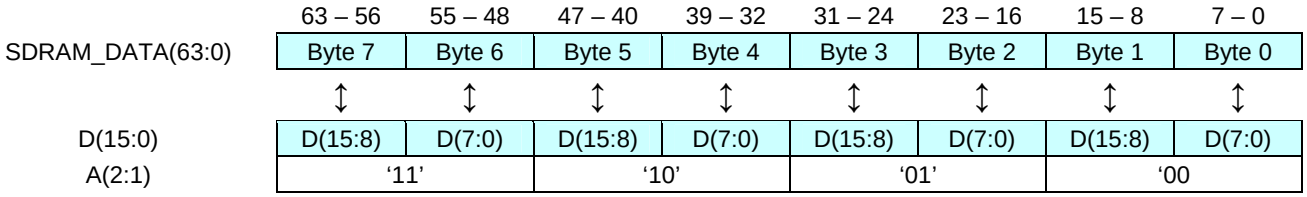
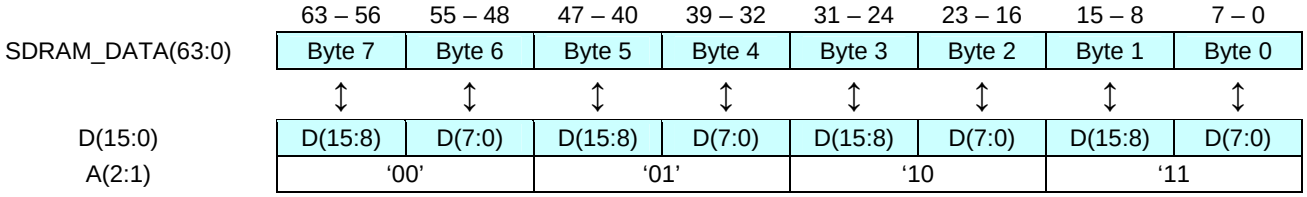


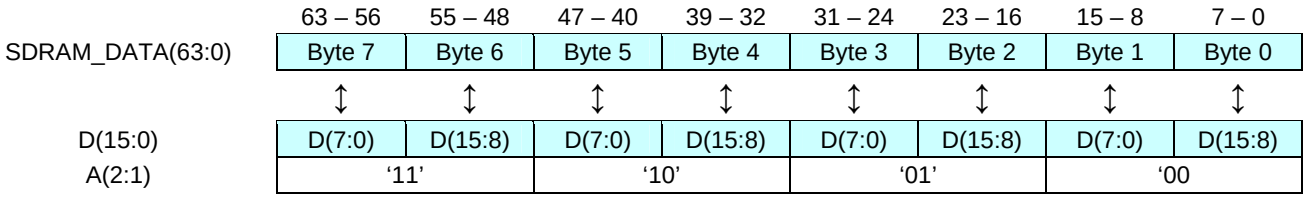
Figure 3-25. Endian Conversion of **SDRAM_DATA** on 8-bit I^2C and Parallel Interface



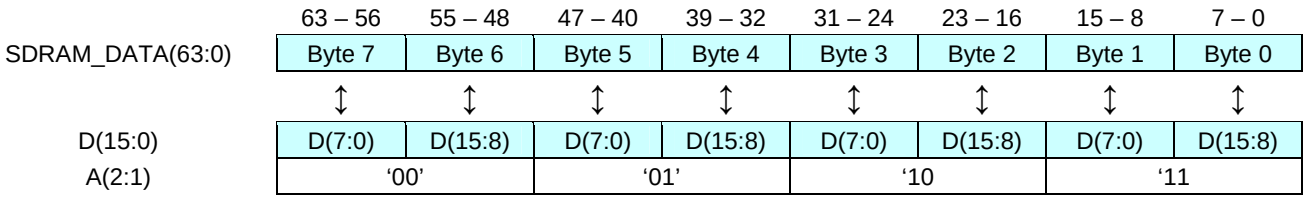
(a) ***endian_swap*** = ‘00’



(b) ***endian_swap*** = ‘01’

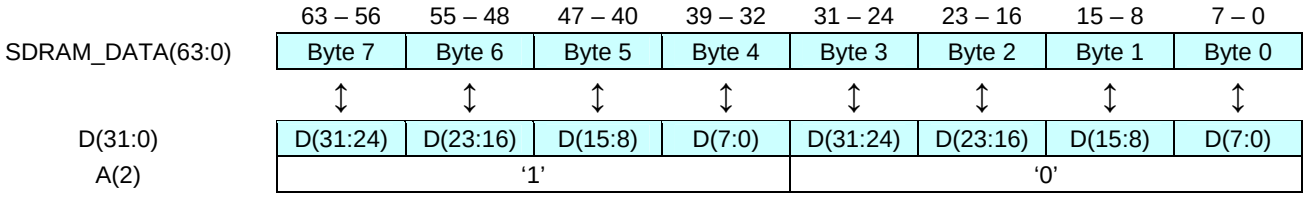


(c) ***endian_swap*** = ‘10’

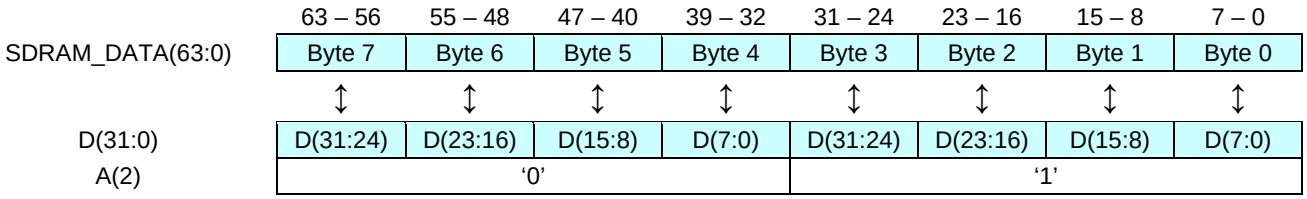


(d) ***endian_swap*** = ‘11’

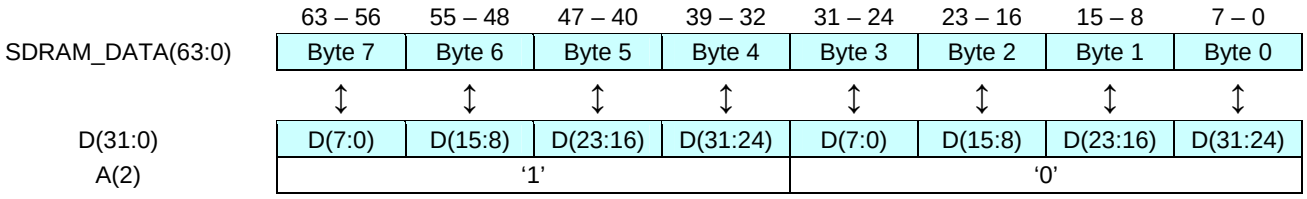
Figure 3-26. Endian Conversion of **SDRAM_DATA** on 16-bit Parallel Interface
(Note: A(0) is not used)



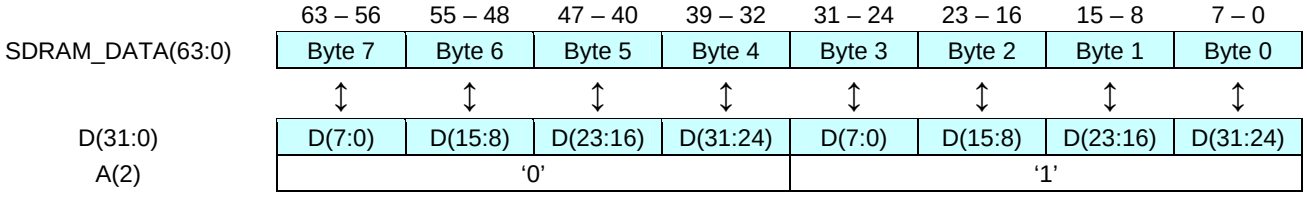
(a) ***pci_wr_endian_swap*** or ***pci_rd_endian_swap*** = '00'



(b) ***pci_wr_endian_swap*** or ***pci_rd_endian_swap*** = '01'



(c) ***pci_wr_endian_swap*** or ***pci_rd_endian_swap*** = '10'



(d) ***pci_wr_endian_swap*** or ***pci_rd_endian_swap*** = '11'

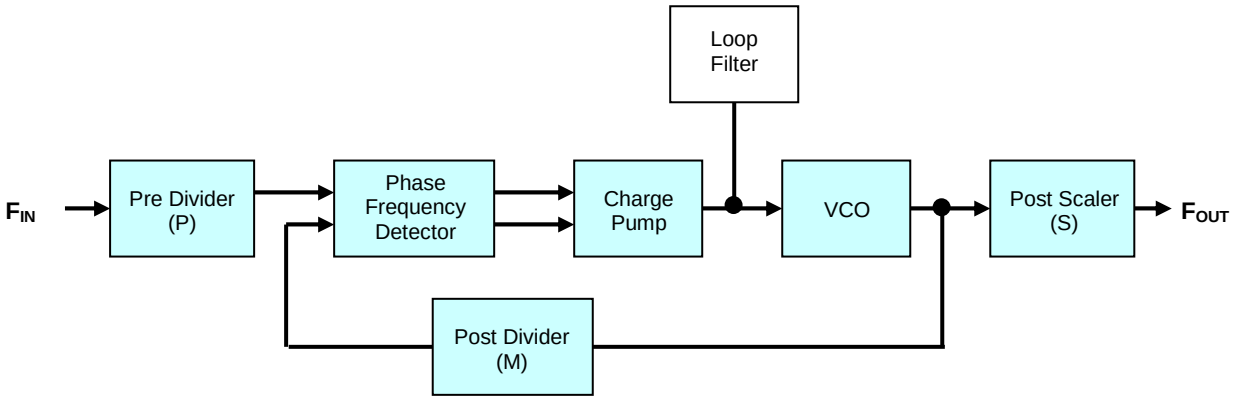
Figure 3-27. Endian Conversion of **SDRAM_DATA** on PCI Interface

3.3 Internal PLL and Clock

MDIN-3xx operates with several clock sources including video clock (**vclk**), memory clock (**mclk**), host interface clock (**hclk**), main input clock (**clk_m**) and internal DAC clock (**dac_clk**). MDIN-3xx can use input video clocks (**CLK_A, CLK_B**) as well as the crystal input (**XTAL_IN**) as a source of the internal PLL for the video clock. The crystal input can only be used as a source of the internal PLL for the memory clock. The crystal input and the divided memory clock can be used as the host interface clock.
MDIN-3xx generates delay-programmable SDRAM clock (**SDRAM_OUT_CLK**), output video clock (**VCLK_OUT**) for proper external interfaces. MDIN-3xx provides several ways in selecting one of various clock sources for default operation and for test purpose. The clock selection and distribution will be explained in the following sections

3.3.1 Internal PLL

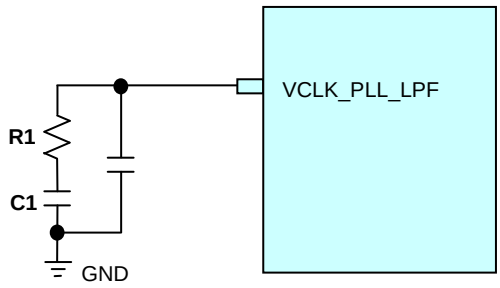
MDIN-3xx has two internal PLL's. One generates the video clock (**vclk**) for video processing, the other generates the memory clock (**mclk**) for SDRAM interface. Figure 3-28 shows the block diagram of a PLL block. The required clock frequency for each PLL is obtained by setting ***pre_div_vclk***, ***post_div_vclk***, ***post_scale_vclk*** registers for the video clock and ***pre_div_mclk***, ***post_div_mclk***, ***post_scale_mclk*** registers for the memory clock.
The pre-divider value (P) in Figure 4-19 is determined by ***pre_div_vclk*** or ***pre_div_mclk*** registers. The post-divider value (M) is determined by ***post_div_vclk*** or ***post_div_mclk*** registers. The post scaler value (S) is determined by ***post_scale_vclk*** or ***post_scale_mclk*** registers. The PLL for the memory clock requires no external loop filter.



$$F_{OUT} = (F_{IN} * M) / (P * S)$$

Figure 3-28. Structure of Internal PLL

The PLL for the video clock requires a loop filter as shown in Figure 3-29(a). The recommended values for R1, C1, C2 are also shown in Figure 3-29(b). They vary depending on the input and output frequency and the frequency multiplication. If lower phase jitter is needed, it is recommended to use smaller C1.



(a) Loop Filter Circuit for Video PLL

F_{IN} / P (MHz)	M	**F_{OUT}** * S (MHz)	**R1** (kΩ)	**C1** (uF)	**C2** (pF)
0.5172 ~ 150	2 ~ 116	60 ~ 300	0.56	0.015	150
0.2343 ~ 100	3 ~ 256	60 ~ 300	0.56	0.033	330
0.1136 ~ 50	6 ~ 528	60 ~ 300	0.56	0.068	680
0.0513 ~ 25	12 ~ 1168	60 ~ 300	0.56	0.150	1500
0.0233 ~ 12	25 ~ 2572	60 ~ 300	0.56	0.330	3300
0.0146 ~ 7.143	42 ~ 4096	60 ~ 300	0.56	0.560	5600

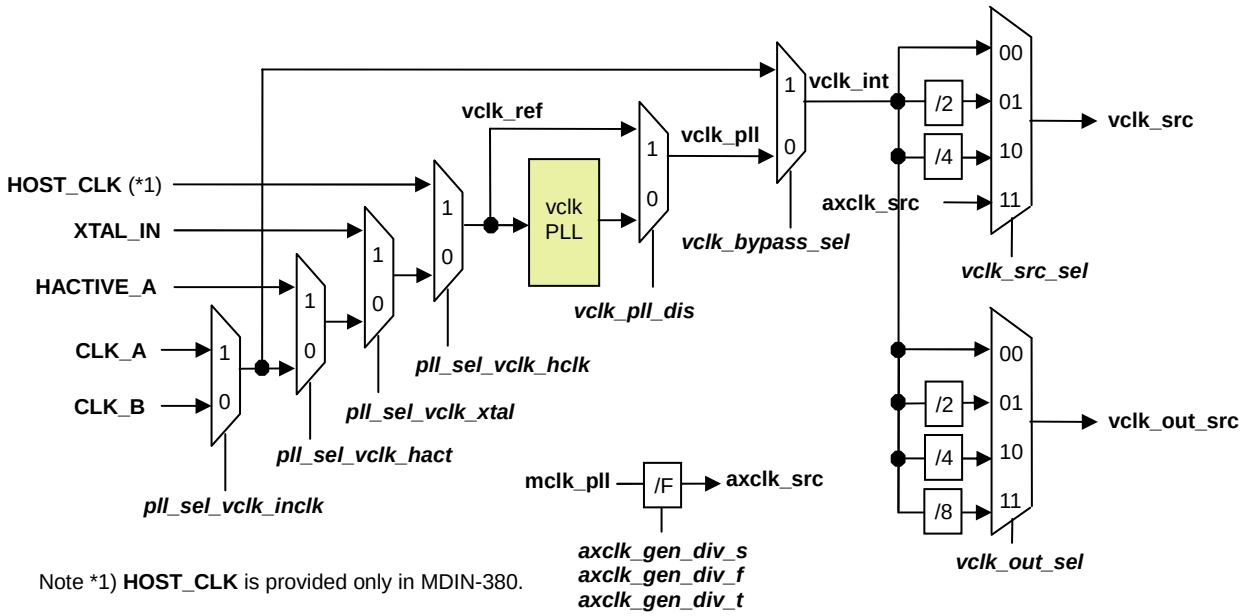
(b) Recommended Values of Loop Filter Constants

Figure 3-29. Loop Filter for Video PLL

3.3.2 Video Clock

The video clocks in MDIN-3xx are generated using the PLL from several clock sources as shown in Figure 3-30. The clock signals from the input pins are used as the reference clock of the PLL. After reset, the clock signal **XTAL_IN** from the crystal oscillator is selected by default, and the PLL is disabled and bypassed. If the PLL is not necessary, it is recommended to disable it to save the power consumption. When the PLL is disabled, the clock input to it should be bypassed to the output path.
The video clock PLL has the capability of generating the clock of most standard video formats from the single frequency of the crystal oscillator. When the video clock needs to be locked to the input video clock, **CLK_A** or **CLK_B** can be selected as the reference of the PLL. When the video clock needs to be line-locked to the input sync signal, **HACTIVE_A** is selected. The clock for the host interface also can be selected as the reference clock in MDIN-380. This clock should not be selected in MDIN-325/340. When an external clock source needs to be used as the video clock, it can be connected to one of the reference clock inputs with bypassing the PLL.

Normally, the output clock of the PLL is used as the main video clock (**vclk**) by setting ***vclk_src_sel*** to '00'. When the pixel repetition is required for the HDMI transmission, the main video clock can be divided by 2 or 4 with setting ***vclk_src_sel*** to '01' or '10' respectively, and the output clock of the PLL is fed to the HDMI transmitter. The auxiliary video clock (**axclk**) which is generated from the memory clock can also be used by setting ***vclk_src_sel*** to '11'. The auxiliary video clock is generated by the fractional divider. When the division ratio is a fractional number, it should be noted that a large amount of clock jitter will be generated.
Various kinds of video clock signals are used in MDIN-3xx as shown in Figure 3-31. Each clock can be selected from serveral clock sources and its propagation delay can be tuned independently using the delay control logic.



Note *1) **HOST_CLK** is provided only in MDIN-380.

(a) Block diagram of the video clock generation logic

Reference of vclk PLL	***pll_sel_vclk_hclk***	***pll_sel_vclk_xtal***	***pll_sel_vclk_hact***	***pll_sel_vclk_inclk***
HOST_CLK (*1)	'1'	don't care	don't care	don't care
XTAL_IN	'0'	'1'	don't care	don't care
HACTIVE_A	'0'	'0'	'1'	don't care
CLK_A	'0'	'0'	'0'	'1'
CLK_B	'0'	'0'	'0'	'0'

Note *1) **HOST_CLK** is provided only in MDIN-380. ***pll_sel_vclk_hclk*** should be '0' for MDIN-325/340.

(b) Reference source selection of the video clock PLL

Figure 3-30. Video Clock Generation

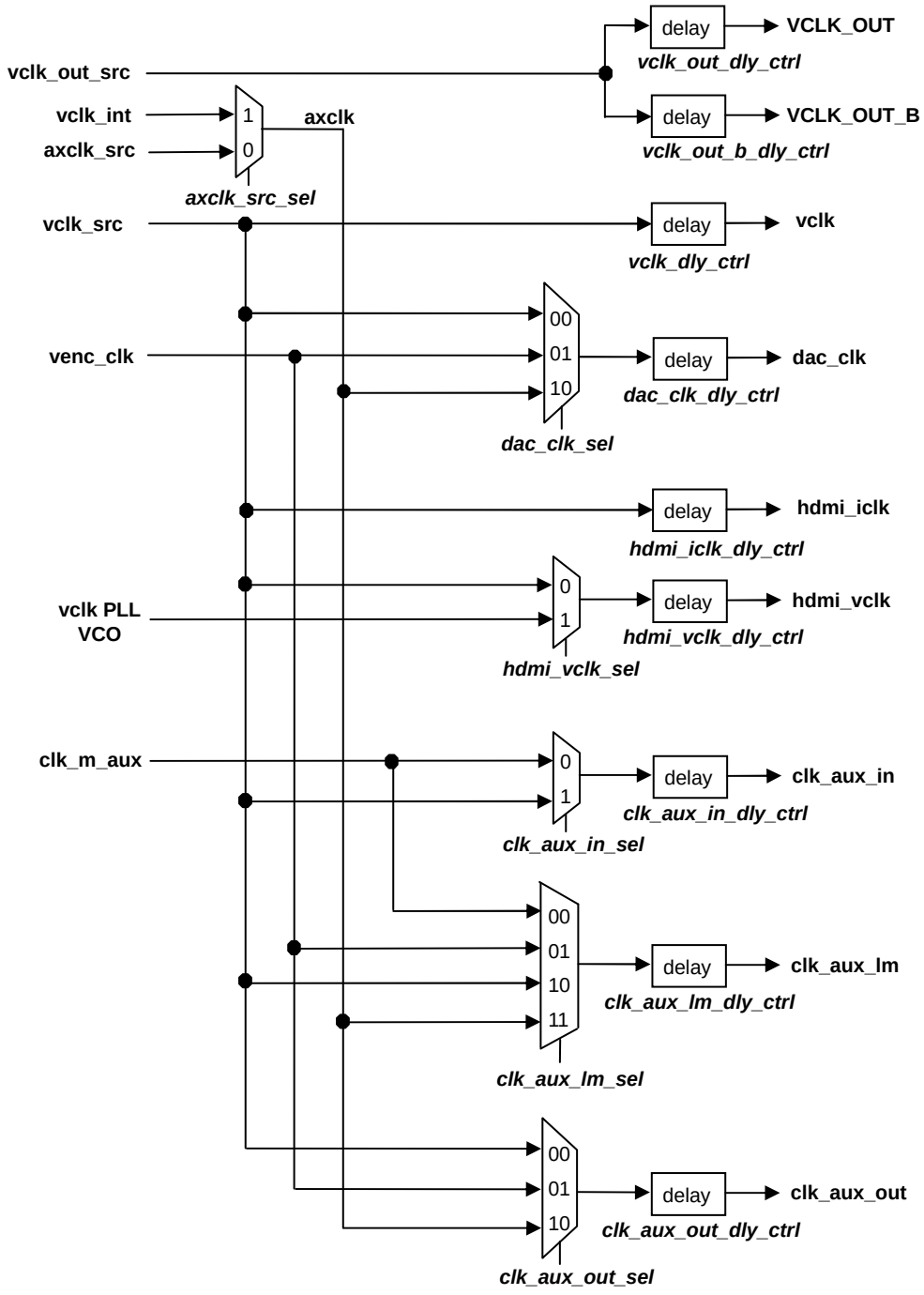
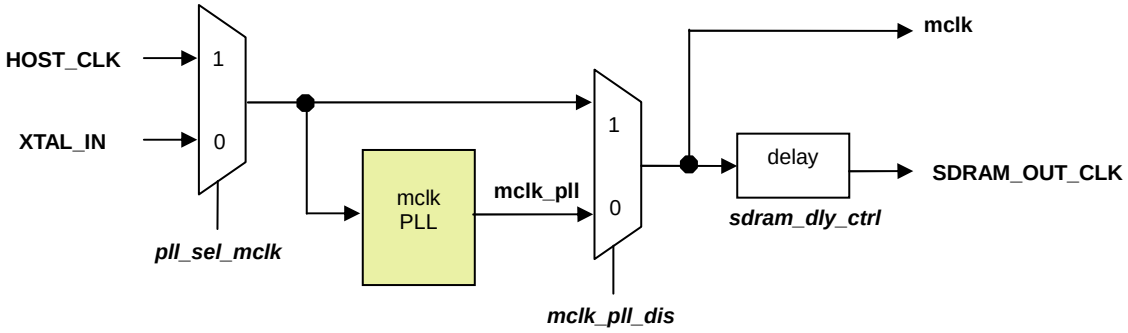


Figure 3-31. Video Clock Network

3.3.3 Memory Clock

Memory clock **(mclk**) is used for internal SDRAM control logic and SDRAM interface. Normally, it is generated by the PLL from **XTAL_IN** input clock. After reset, the PLL is disabled and ***mclk_pll_dis*** is high, so that the clock from **XTAL_IN** is bypassed to the memory clock as shown in Figure 3-32. For normal operation, the PLL generates the memory clock from **XTAL_IN**. **HOST_CLK** also can be selected as the reference clock to the memory clock PLL in MDIN-380. This clock should not be selected in MDIN-325/340. Figure 3-32 shows the memory clock generation logic. The frequency setting of the PLL should be double the frequency of **mclk** because the clock is divided by 2 in the PLL block.



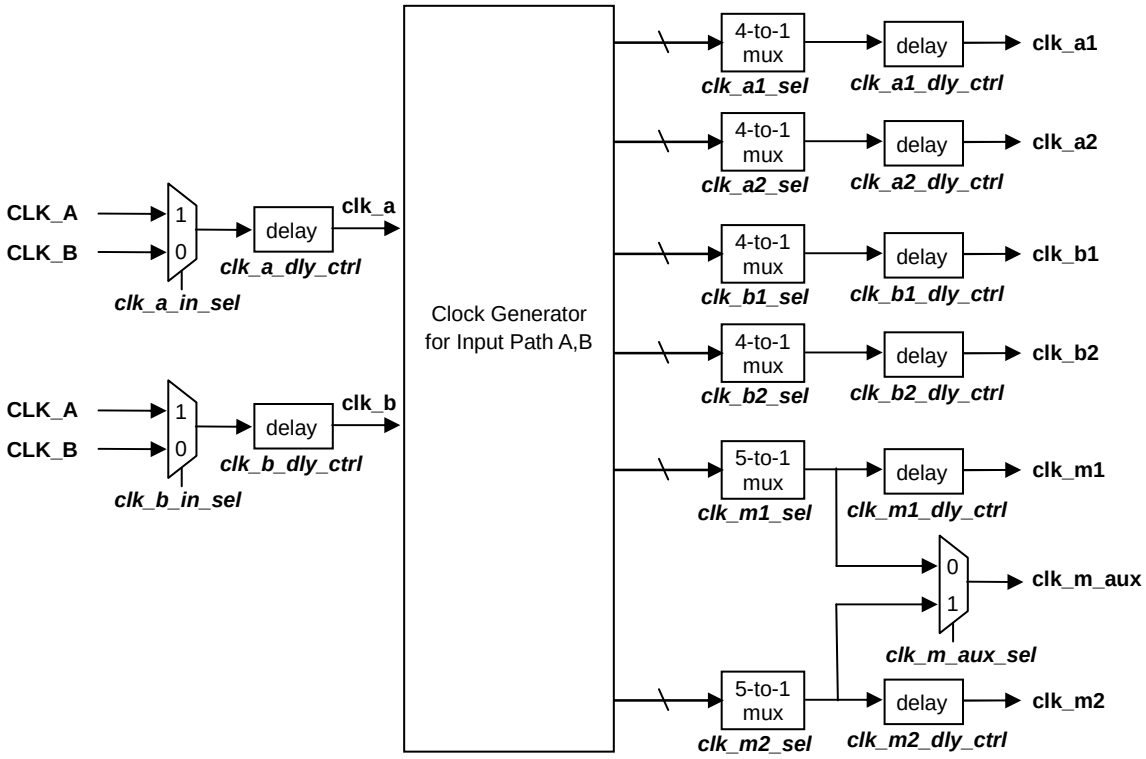
mclk_pll_dis	***pll_sel_mclk***	Input of PLL	**mclk**
'0'	'0'	**XTAL_IN**	**mclk_pll**
'0'	'1'	**HOST_CLK** (*1)	**mclk_pll**
'1'	'0'	-	**XTAL_IN**
'1'	'1'	-	**HOST_CLK** (*1)

Note *1) **HOST_CLK** is provided only in MDIN-380. ***pll_sel_mclk*** should be '0' for MDIN-325/340.

Figure 3-32. Memory Clock Generation Logic

3.3.4 Video Input Clock

MDIN-3xx have serveral internal video clocks for input video processing as shown in Figure 3-33(a).



(a) Clock Generation Logic of Input Path

Clock	4-to-1 Mux	'00'	'01'	'10'	'11'
clk_a1	***clk_a1_sel***	**clk_a**, 0°	**clk_a**/2, 0°	**clk_a**/2, 0°	**clk_a**/4, 0°
clk_a2	***clk_a2_sel***	**clk_a**, 180°	**clk_a**/2, 180°	**clk_a**/2, 180°	**clk_a**/4, 90°
clk_b1	***clk_b1_sel***	**clk_b**, 0°	**clk_b**/2, 0°	**clk_a**/2, 90°	**clk_a**/4, 180°
clk_b2	***clk_b2_sel***	**clk_b**, 180°	**clk_b**/2, 180°	**clk_a**/2, 270°	**clk_a**/4, 270°

(b) Input Clock Configurations

Clock	5-to-1 Mux	'000'	'001'	'010'	'011'	'100'
clk_m1	***clk_m1_sel***	**clk_a**, 0°	**clk_a**/2, 90°	**clk_a**/4, 90°	**clk_a**/8, 180°	**clk_a**/4, 90°
clk_m2	***clk_m2_sel***	**clk_b**, 0°	**clk_b**/2, 90°	**clk_a**/4, 135°	**clk_a**/8, 0°	**clk_b**/4, 135°

(c) Input Pixel Clock Configurations

Figure 3-33. Clock Generation of Input Path

In normal operations, **CLK_A** is selected as the input clock of the input path A (**clk_a1**) (***clk_a_in_sel*** = '1', ***clk_a1_sel*** = '00'), and it is also selected as the input pixel clock **clk_m1** of the main video path (***clk_m1_sel*** = '000'). The input clock is divided by 2 and used as the input pixel clock (***clk_m1_sel*** = '001') when a Y/C-multiplexed video is demultiplexed in

the input path. In similar fashion, **CLK_B** is used as the source clock of **clk_b1** and **clk_m2**. The aux video path can use the same clock path as the main video path or the other clock path depending on the video processing architecture. It should be noted that **CLK_A** cannot be the clock source of the aux video path when **CLK_B** is selected as the clock source of the main video path.
When the input path A receives 2-channel time-multiplexed video data, **clk_a1** and **clk_a2** can be configured to latch one of 2 channels respectively. **clk_b1** and **clk_b2** also can be configured to receive 2-channel video data on the input video path B. When the input path A or B receives 4-channel time-multiplexed video data, all four input clocks are used and they needs to be configured to latch one of 4 channels respectively. These clock configurations are explained in Figure 3-33(b).
clk_m1 is connected to the input path A and mainly used as the input pixel clock of the main video path. **clk_m2** can be connected to the input path A or B and used as the input pixel clock of the aux video path or the input pixel clock of the second video path on multi-channel mode. The input pixel clocks are configured as shown in Figure 3-33(c).
Figure 3-34 shows the clock waveforms in the input path.

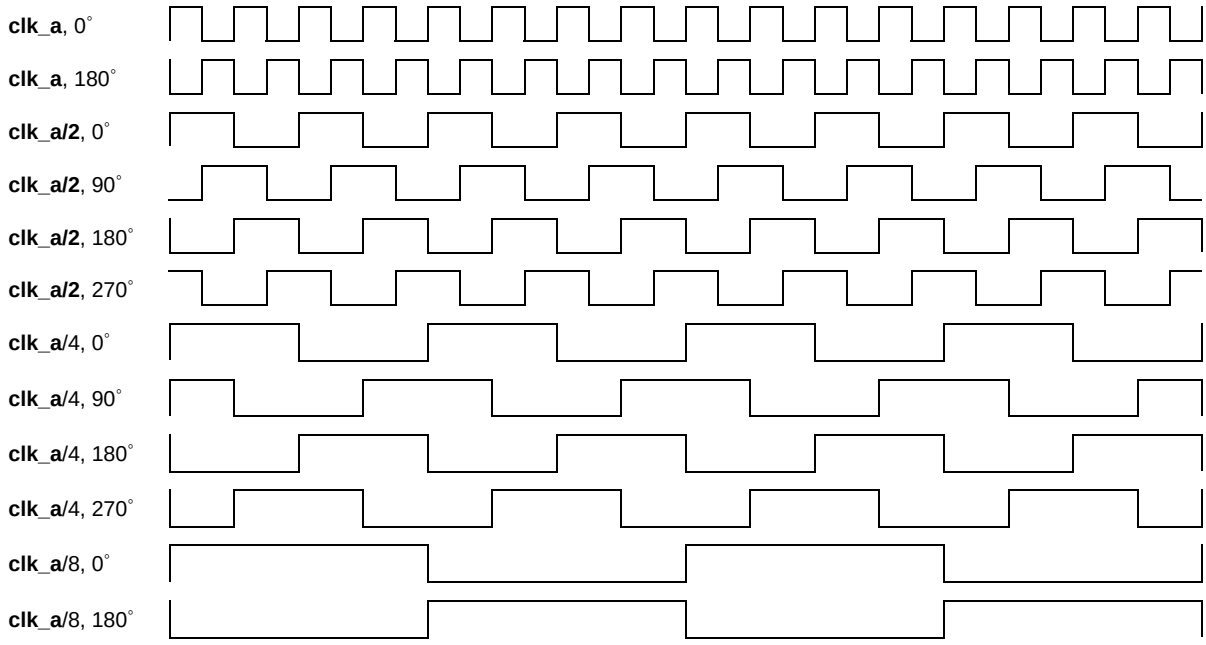


Figure 3-34. Input Clock Waveforms

Table 3-6 shows example configurations of the input clocks and input pixel clocks for single or multi-channel input.

Y/C muxed	Channels on A / B	Data Rate	Clock Edge	***clk_in_a_sel***	***clk_in_b_sel***	***clk_a1_sel***	***clk_a2_sel***	***clk_b1_sel***	***clk_b2_sel***	***clk_m1_sel***	***clk_m2_sel***
No	1 / 0	1	Single	'1'	-	'00'	-	-	-	'000'	-
	2 / 2	1	Dual	'1'	'0'	'00'	'00'	'00'	'00'	'000'	'000'
	2 / 2	2	Single	'1'	'0'	'01'	'01'	'01'	'01'	'001'	'001'
	4 / 0	2	Dual	'1'	'1'	'10'	'10'	'10'	'10'	'001'	'001'
	4 / 0	4	Single	'1'	'1'	'11'	'11'	'11'	'11'	'100'	'100'
Yes	1 / 0	1	Single	'1'	-	'00'	-	-	-	'001'	-
	2 / 2	1	Dual	'1'	'0'	'00'	'00'	'00'	'00'	'001'	'001'
	2 / 2	2	Single	'1'	'0'	'01'	'01'	'01'	'01'	'100'	'100'
	4 / 0	2	Dual	'1'	'1'	'10'	'10'	'10'	'10'	'100'	'100'
	4 / 0	4	Single	'1'	'1'	'11'	'11'	'11'	'11'	'011'	'011'

Table 3-6. Example Configurations of Input Clocks

3.3.5 Host Interface Clock

Host interface clock **(hclk**) is used for the internal host interface logic. It can be selected by the test mode pins **TEST_MODE(2)** and **TEST_MODE(1)** as shown in Table 3-7. Normally, **XTAL_IN** is used as **hclk** by connecting both **TEST_MODE(2)** and **TEST_MODE(1)** to low. When the synchronous host interface is used in MDIN-380, **HOST_CLK** should be used as **hclk**. The frequency of **hclk** should not be less than 1/6 of **mclk** frequency if reading or writing SDRAM data over the host interface is required.

TEST_MODE(2:1)	hclk
'00'	**XTAL_IN**
'01'	**HOST_CLK**
'10'	reserved
'11'	**mclk** / 2

Table 3-7. Host Interface Clock Selection

3.3.6 Video Encoder and DAC Clock (MDIN-325/380)

The video encoder in MDIN-325/380 receives a video from three different sources – Input port A, Input port B and Aux video output. Various clock signals are multiplexed to the clock of the video encoder and DAC. When the video source is the aux video, the video encoder clock should be **XTAL_IN** and the aux output clock (**clk_aux_out**) should be **vclk_enc**.

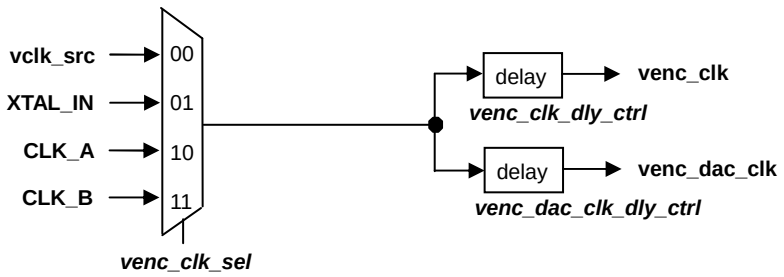


Figure 3-35. Video Encoder and DAC Clock Generation Logic

3.4 Crystal Oscillator Interface

Figure 3-36 shows a reference circuit diagram for crystal oscillation circuit. The frequence range of the crystal oscillator interface is 1 to 54MHz. Normally a 27MHz crystal with fundamental mode is used to generate the frequencies of host clock, memory clock and video clock for standard video formats.

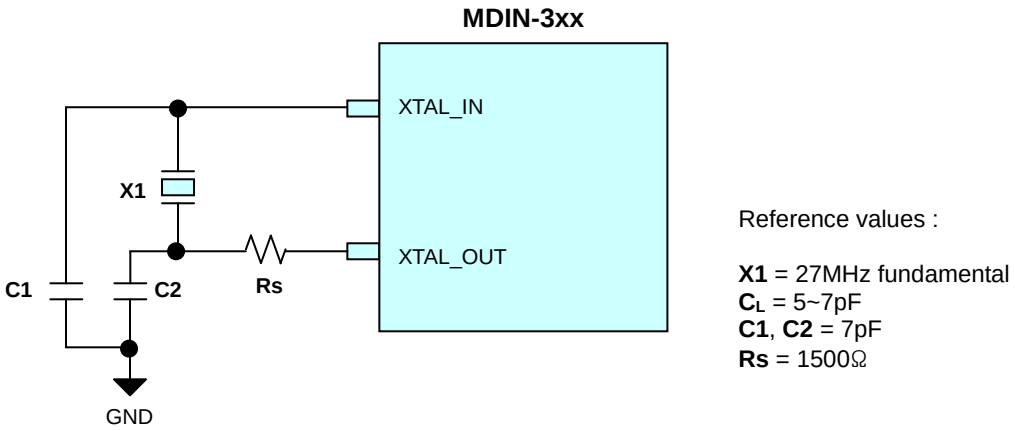


Figure 3-36. Reference Circuit for External Crystal

The value of external load capacitors (**C1**, **C2**) should be chosen to meet the equation $C_L = (C1||C2) + C_{stray}$, where C_L is the load capacitance of **X1** and C_{stray} is the stray capacitance by the pin capacitance and board or trace PCB-related capacitance (typically 2 to 7 pF).
Rs resistor is used in order to limit the output of **XTAL_OUT** so that the crystal is not over driven. The minimum value recommended depends on the crystal characteristics. Note that over-driving of the crystal can be observed on the oscillator output signal (where the signal is non-symmetrical or is oscillating at the harmonic frequency). Crystal over-driving shortens its life and in some cases, permanently damages it. Ideally the inverter in the crystal oscillator interface provides 180° phase shift, but the inherent delay of the inverter provides additional phase shift. In order to ensure the total 360° shift delay, **Rs** can be used to decrease the shift delay in the feedback loop. **Rs** and **C2** form a voltage dividing circuit. Acceptable results can be reached by choosing the value of **Rs** equal to the **C2** capacitive reactance.

If there is an LVTTL-level clock signal with the required frequency in system, it can be directly input to **XTAL_IN** pin without a crystal. If this signal has small amplitude and its offset voltage does not fall around the threshold voltage of the oscillatior interface (≈VDD33/2), a capacitor needs to be inserted in series to adjust the offset voltage to be around the threshold voltage. Care must be taken in choosing an external clock signal, if **XTAL_IN** is used as a source of video clock PLL. The clock must have low jitter characteristics to avoid jitter on the output video signal.

3.5 Internal DAC

3.5.1 Triple DACs

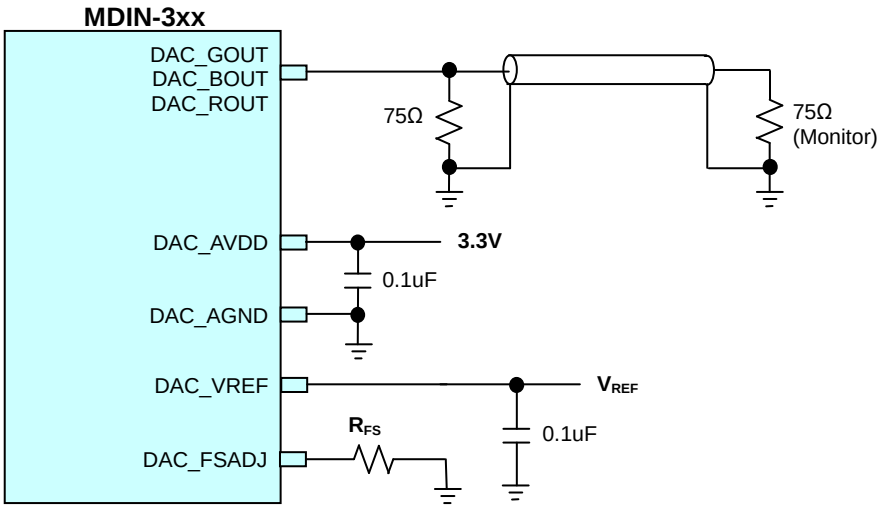
MDIN-3xx can generate RGB or YPbPr analog video using 3 internal 10-bit DACs. Various video formats including VESA and DTV formats are supported with or without embedded sync signals. The embedded sync can be conventional bi-level sync or tri-level sync.
The typical external circuit for the DAC is shown in Figure 3-37(a). Each DAC channel is capable of driving a 37.5Ω load (75Ω doubly terminated). The full-scale output current is controlled by the reference voltage on **DAC_VREF** pin and an external resistor connected on **DAC_FSADJ** pin, and it is defined as $I_{FS} = (V_{REF}/R_{FS})*I_{REF}$, where I_{REF} = 76.725. The nominal value of R_{FS} is 3.3kΩ corresponding to 27.9mA full-scale current with V_{REF} = 1.2V. MDIN-3xx has no internal voltage reference. The reference voltage V_{REF} should be supplied from a clean and stable external voltage reference such as a bandgap voltage reference or a zener diode to gererate a high quality video output. However, it might be generated by dividing the supply voltage with a resistor divider if the supply voltage is well regulated.
When the separate sync signals (**HSYNC_OUT**, **VSYNC_OUT**) are used and 5V signaling is required, 5V digital buffers can be used for level-shifting because MDIN-3xx's output level is 3.3V. In this case, the input level of the buffers should be should be LVTTL compatible. Serial damping registers can be used to reduce the slew rate of the sync signals.

3.5.2 Video Encoder DAC (MDIN-325/380)

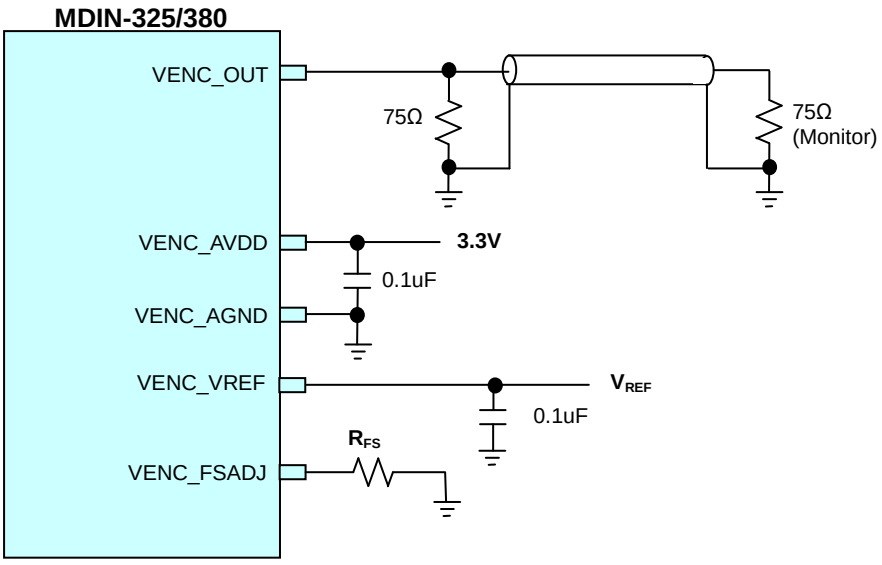
MDIN-325/380 can generate NTSC/PAL analog video using the video encoder and an internal 10-bit DAC. Y/C separate (S-Video) video can also be generated through triple DACs.
The typical external circuit for the video encoder DAC is shown in Figure 3-37(b). The DAC is capable of driving a 37.5Ω load (75Ω doubly terminated). The full-scale output current is controlled by the reference voltage on **VENC_VREF** pin and an external resistor connected on **VENC_FSADJ** pin, and it is defined as $I_{FS} = (V_{REF}/R_{FS})*I_{REF}$, where $I_{REF} = 76.725$. The nominal value of R_{FS} is 3.3kΩ corresponding to 27.9mA full-scale current with $V_{REF} = 1.2V$. MDIN-3xx has no internal voltage reference. The reference voltage V_{REF} should be supplied from a clean and stable external voltage reference such as a bandgap voltage reference or a zener diode to gererate a high quality video output. However, it might be generated by dividing the supply voltage with a resistor divider if the supply voltage is well regulated.

3.5.3 Low-power Consideration

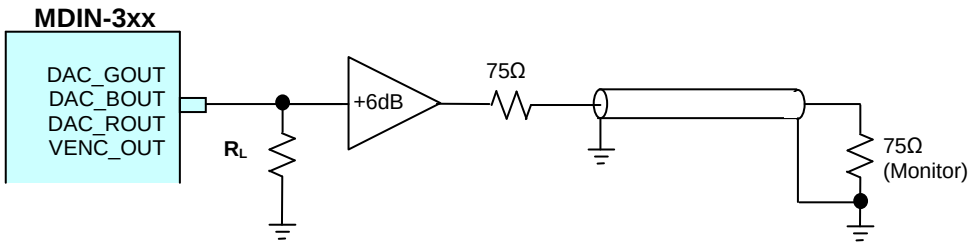
The doubly terminated architecture shown in Figure 3-37(a) is quite simple and cost effective but the power consumption is quite high due to the low impedance 37.5Ω loads. If the low power consumption is required, the full-scale current can be lowered by increasing the value of R_{FS} with increasing the values of load resistor R_L proportially. In this case, the termination resistors at the monitor input can not be driven properly by the lowered output current of DAC, so video buffers should be used between DAC and cables as shown in Figure 3-37(c). The output load for the video buffer is 150Ω, so the output current to the load will be lowered considerably.



(a) Triple DACs



(b) Video Encoder DAC

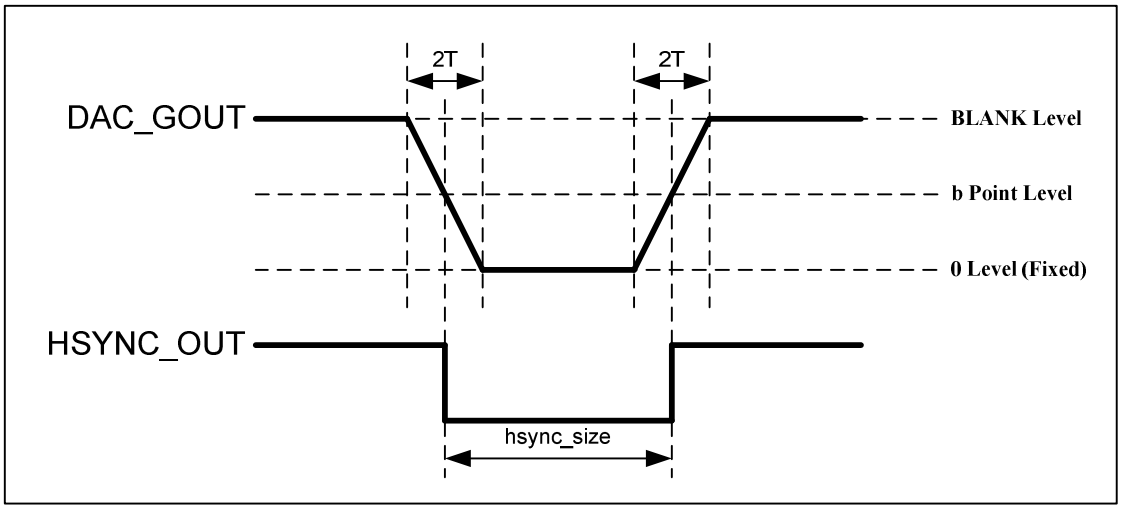


(c) Low Power Consumption on DAC

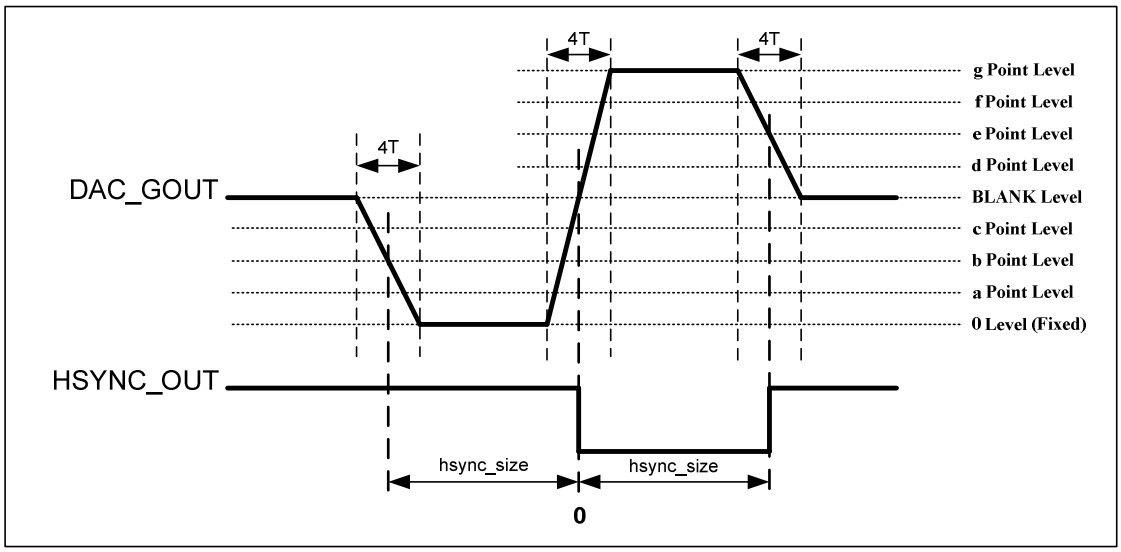
Figure 3-37. Reference Circuit for Internal DACs

3.5.4 DAC Waveform Generation

MDIN-3xx has a special register set to control the operation of the internal DAC. Sync level, sync width, sync type, blank level, output level scaling and clipping are controlled by there registers. They can be written or read using ***dac_ctrl_data*** , ***dac_ctrl_wen*** and ***dac_ctrl_addr***. They are also 16-bit wide and their address range is 0x00 to 0x24. Refer to 3.14.3 in MDIN-3xx Register Manual for the descriptions of the DAC control registers.
Figure 3-38 shows the meaning of sync level and blank level registers.



(a) Bi-level Sync



(b) Tri-level Sync

Figure 3-38. Embedded Sync Waveform from Internal DAC

The waveforms in the blanking and sync intervals are determined by ***dac_line_formatX*** registers. From the given vertical start position to the next vertical position the waveforms are generated with the specified line format. Table 3-8 lists the pre-defined line formats and Figure 3-39 and Figure 3-40 show the actual waveforms of the line formats.

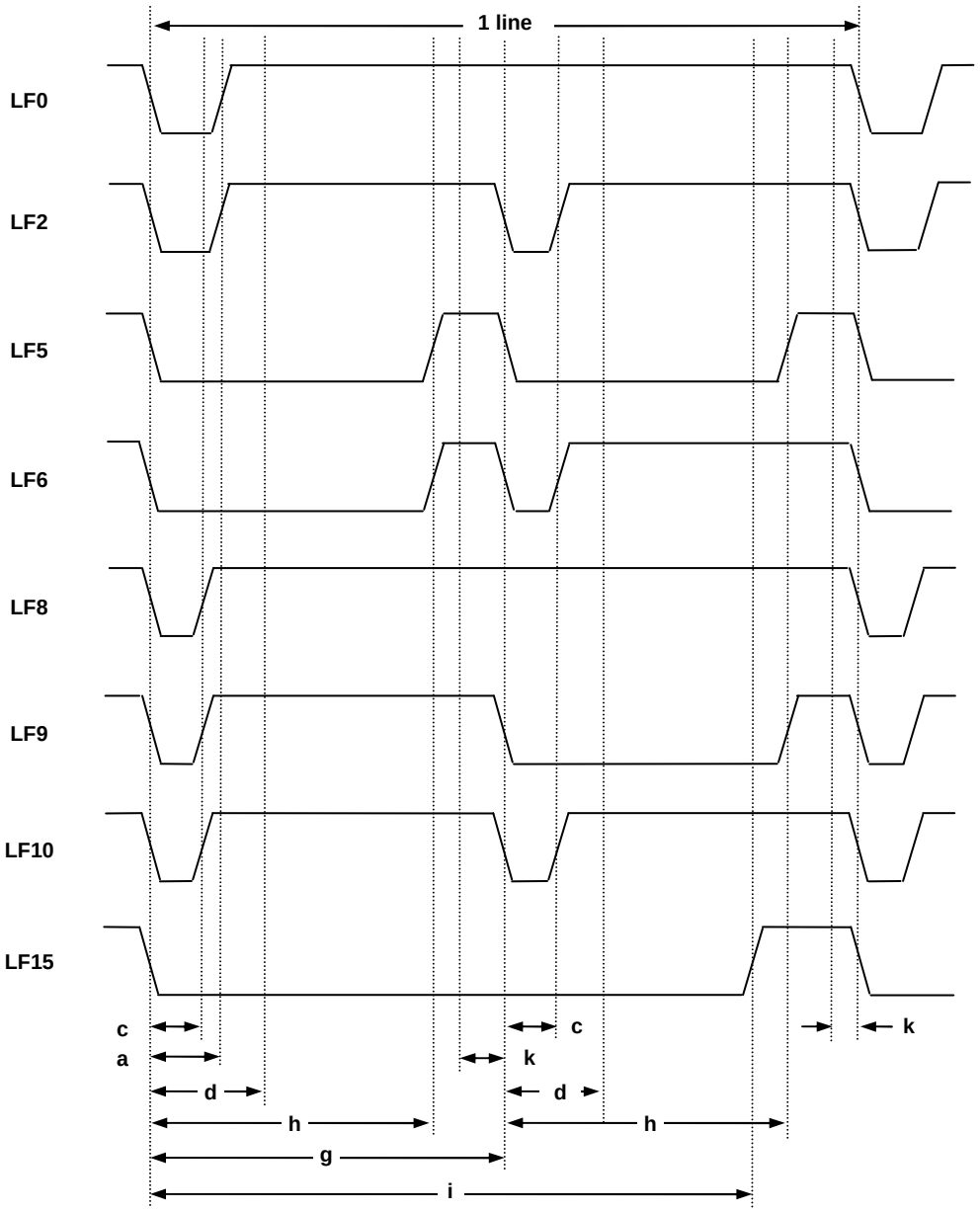
Index to Line Format	Line format of 1st half line	Line format of 2nd half line
0	blanking (full line)	
2	blanking	equalizing pulse
5	broad pulse	broad pulse
6	broad pulse	equalizing pulse
8	equalizing pulse (full line)	
9	equalizing pulse	broad pulse
10	equalizing pulse	equalizing pulse
15	broad pulse (full line)	

(a) Line Format for Bi-level Sync Mode

Index to Line Format	Line format of 1st half line	Line format of 2nd half line
0	blanking (full line)	
5	broad pulse	broad pulse
6	broad pulse	blanking
9	blanking	broad pulse
10	blanking	blanking
15	broad pulse (full line)	

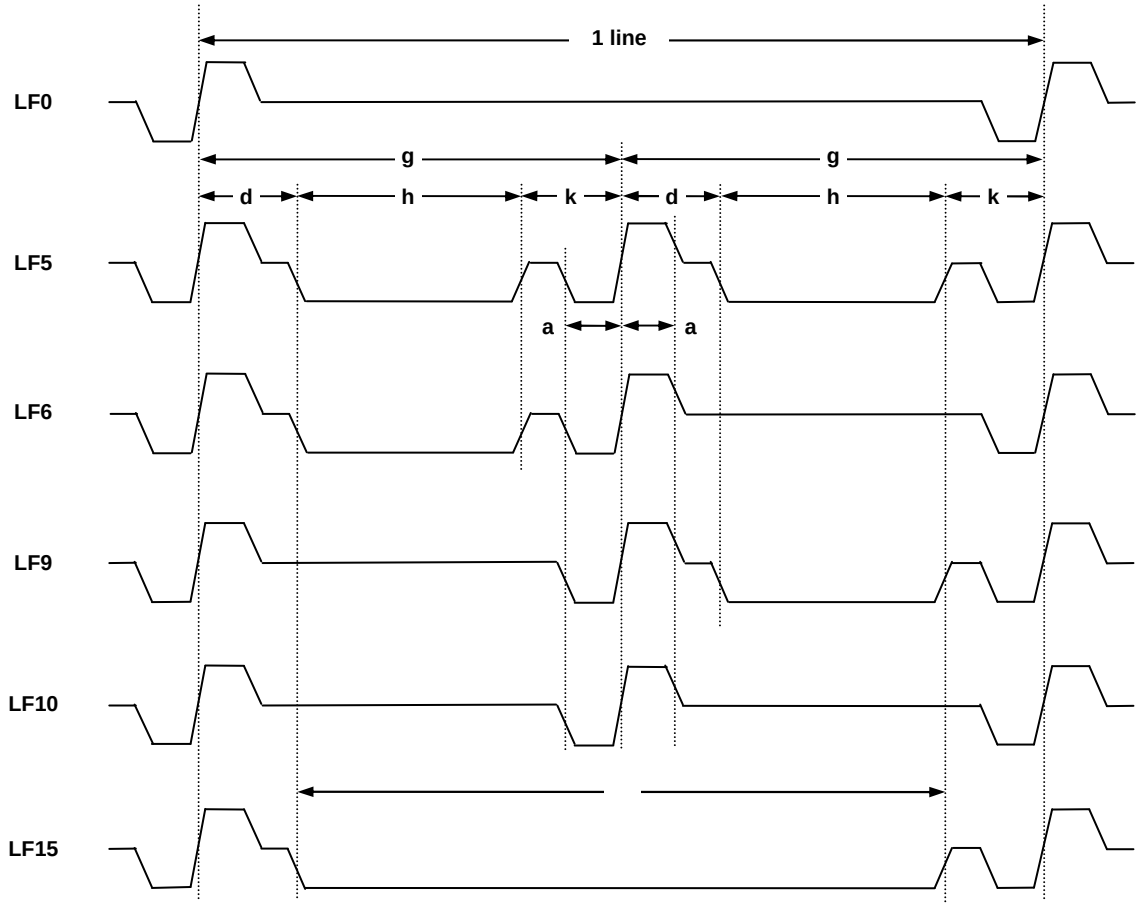
(b) Line Format for Tri-level Sync Mode

Table 3-8. Description of Line Formats for internal DAC



Legend	Descriptions
a	***dac_hsync_size***
c	***dac_heq_pulse_size***
d	***hactive_start***
g	***dac_htotal_half_size***
h	***dac_htotal_half_size*** - ***dac_heq_pulse_size***
i	***htotal_size*** - ***dac_hsync_size***
k	***htotal_size*** - ***hactive_end***

Figure 3-39. Line Format Waveforms for Bi-level Sync Mode



Legend	Descriptions
a	***dac_hsync_size***
d	***dac_hbp_interval***
g	***dac_htotal_half_size***
h	***dac_htotal_half_size - dac_hsync_size - dac_hbp_interval***
i	***htotal_size - dac_hbp_interval - dac_hfp_interval***
k	***dac_hfp_interval***

Figure 3-40. Line Format Waveforms for Tri-level Sync Mode

4. Electrical Characteristics

4.1 Absolute Maximum Ratings

CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Supply Voltage	VDD12	-0.5		1.8	V
	MCLK_PLL_AVDD12	-0.5		1.8	V
	VCLK_PLL_AVDD12	-0.5		1.8	V
	VDD18	-0.5		2.6	V
	VDD33	-0.5		4.6	V
	DAC_AVDD	-0.5		4.6	V
	VENC_AVDD (*1)	-0.5		4.6	V
	VCLK_PLL_AVDD33	-0.5		4.6	V
	HDMI_VDN12 (*2)	-0.5		1.8	V
	HDMI_VDP33 (*2)	-0.5		4.0	V
Input voltage	Vin(3.3V)	-0.5		4.6	V
	Vin(5V-tolerant)	-0.5		6.0	V
Storage Temperature Range	Tstrg	-40		125	°C

Note *1) MDIN-325/380 only
Note *2) MDIN-340/380 only

4.2 Recommended Operation Conditions

CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Operating Supply Voltage	VDD12	1.1	1.2	1.3	V
	MCLK_PLL_AVDD12	1.1	1.2	1.3	V
	VCLK_PLL_AVDD12	1.1	1.2	1.3	V
	VDD18	1.70	1.8	1.95	V
	VDD33	3.0	3.3	3.6	V
	DAC_AVDD	3.0	3.3	3.6	V
	VENC_AVDD (*1)	3.0	3.3	3.6	V
	VCLK_PLL_AVDD33	3.0	3.3	3.6	V
	HDMI_VDN12 (*2)	1.1	1.2	1.3	V
	HDMI_VDP33 (*2)	3.0	3.3	3.6	V
Input voltage	Vin(3.3V)	0		3.6	V
	Vin(5V-tolerant)	0		5.5	V
Operating Temperature Range	Top	0	25	70	°C

Note *1) MDIN-325/380 only
Note *2) MDIN-340/380 only

4.3 DC Electrical Characteristics

4.3.1 Digital I/O Characteristics

CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
High Level Input Voltage	Vih	2.0		VDD33 + 0.3	V
Low Level Input Voltage	Vil	-0.3		0.8	V
High Level Output Voltage	Voh	VDD33 - 0.2			V
Low Level Output Voltage	Vol			0.2	V
High Level Output Current	Ioh			-6	mA
Low Level Output Current	Iol			6	mA
High Level Output Current (clock, sync pins)	Ioh1			-8	mA
Low Level Output Current (clock, sync pins)	Iol1			8	mA
High Level Output Current (I_2C, DDC pins)	Ioh2			-3	mA
Low Level Output Current (I_2C, DDC pins)	Iol2			3	mA

4.3.2 Power Supply Pin Characteristics

POWER DOMAIN	VOLTAGE	INPUT	OUTPUT	MIN	TYP	MAX	UNIT
VDD12	1.2V	480i	480p		114		mA
		480i	720p		162		mA
		480i	1080p		232		mA
		1080i	1080p		265		mA
VDD33	3.3V	480i	480p		29		mA
		480i	720p		32		mA
		480i	1080p		41		mA
		1080i	1080p		43		mA
VDD18	1.8V	480i	> 480i		20		mA
		1080i	> 1080i		58		mA
DAC_AVDD33	3.3V				80		mA
VENC_AVDD33	3.3V				28		mA
HDMI_VDN12	1.2V		480p		8		mA
			720p		20		mA
			1080p		33		mA
HDMI_VDP33	3.3V		480p		2		mA
			720p		8		mA
			1080p		16		mA

4.4 DAC Characteristics

4.4.1 Triple DACs

CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Reference Voltage (DAC_VREF)	V_{REF}	1.15	1.2	1.25	V
Supply Current (DAC_AVDD)	IDDA		40	43	mA
Supply Current (Power-down)	IDDAS		0.3	10	uA
Resolution		-	10	-	bit
Integral Non-linearity	INL	-2.0		+2.0	LSB
Differential Non-linearity	DNL	-1.0		+1.0	LSB
Output Current (V_{REF}=1.2V, R_{FS}=3.3kΩ) - Full-scale (*1) - Zero-scale - LSB Size		 25.2 0	 27.9 4 27.3	 30.6 40	 mA uA uA
Output Compliance (DAC_AVDD = 3.3V)	VOC	-0.1		1.3	V
Output Impedance (V_O<1V, D9-0=all "1")	Z_O		90		kΩ
DAC-to-DAC Matching			3		%
Clock Frequency				165	MHz

Note *1) Full-scale current = (V_{REF}/R_{FS})*76.725

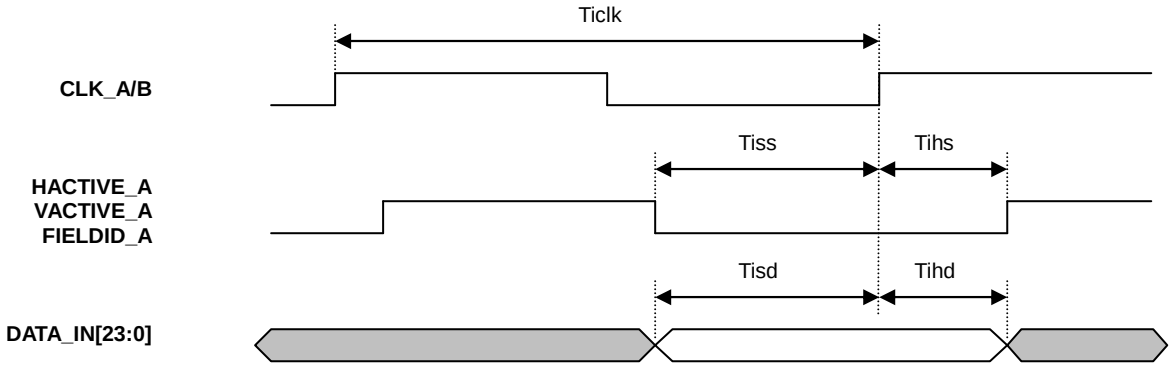
4.4.2 Video Encoder DAC (MDIN-325/380)

CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Reference Voltage (VENC_VREF)	V_{REF}	1.15	1.2	1.25	V
Supply Current (VENC_AVDD)	IDDA		36	40	mA
Supply Current (Power-down)	IDDAS		0.3	10	uA
Resolution		-	10	-	bit
Integral Non-linearity	INL	-2.0		+2.0	LSB
Differential Non-linearity	DNL	-1.0		+1.0	LSB
Output Current (V_{REF}=1.2V, R_{FS}=3.3kΩ) - Full-scale (*1) - Zero-scale - LSB Size		 25.2 0	 27.9 4 27.3	 30.6 40	 mA uA uA
Output Compliance (VENC_AVDD = 3.3V)	VOC	-0.1		1.3	V
Output Impedance (V_O<1V, D9-0=all "1")	Z_O		90		kΩ
DAC-to-DAC Matching			3		%
Clock Frequency				54	MHz

Note *1) Full-scale current = (V_{REF}/R_{FS})*76.725

4.5 AC Electrical Characteristics

4.5.1 Video Input

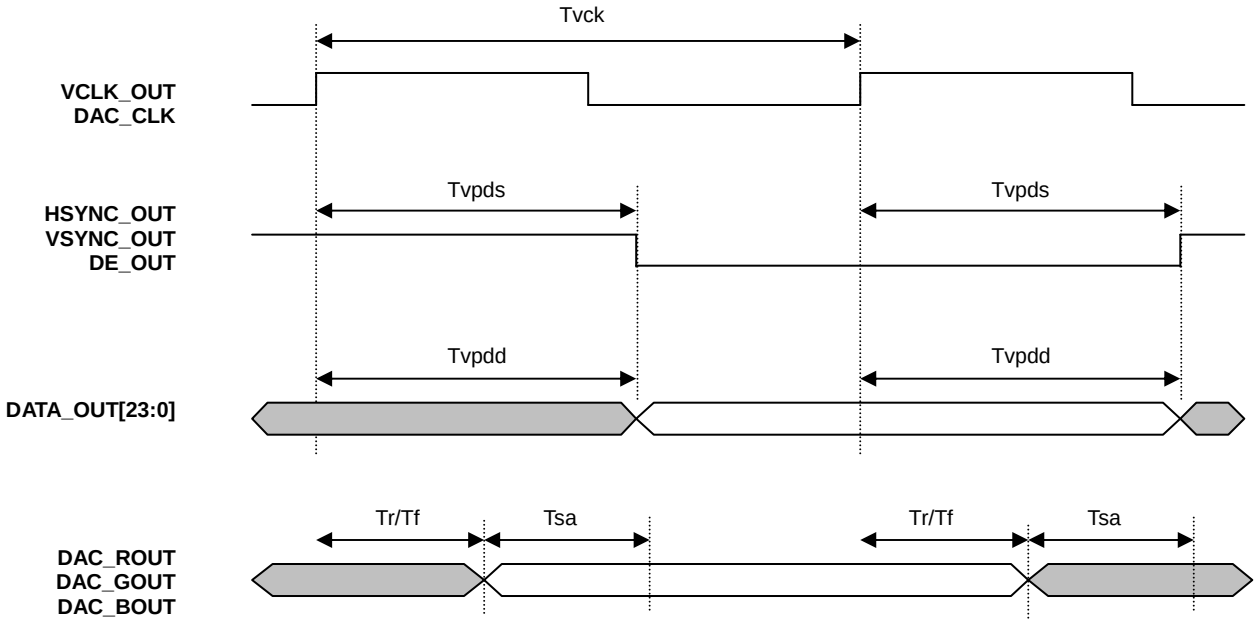


CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Input video pixel clock period	Ticlk	6.1		74	ns
Input video sync signal setup time (*1)	Tiss	1.2			ns
Input video sync signal hold time (*1)	Tihs	0.5			ns
Input video data signal setup time (*1)	Tisd	1.2			ns
Input video data signal hold time (*1)	Tihd	0.5			ns

Note *1) These values correspond to the default setting of the register ***clk_a_dly_ctrl*** (0x044) and ***clk_b_dly_ctrl*** (0x046). They can be adjusted by changing the register value.

Figure 4-1. Video Input Timing Diagram

4.5.2 Video Output



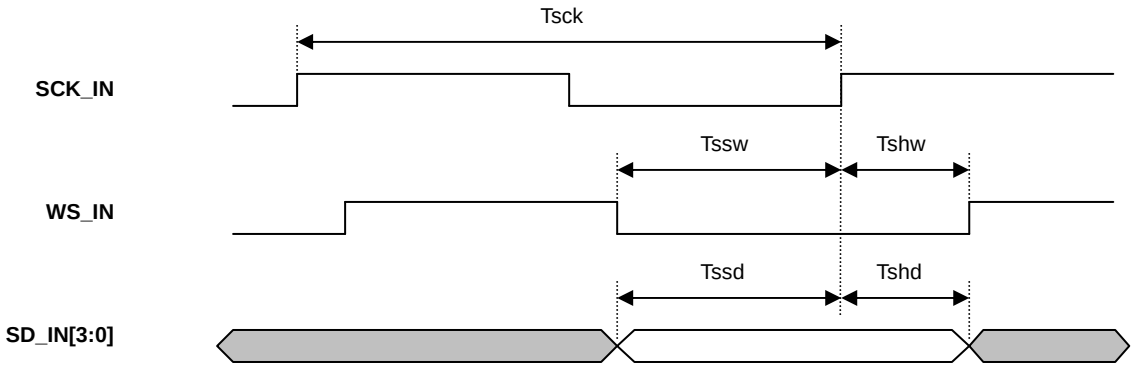
CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Video clock period	Tvclk	6.1		74.0	ns
Vclk to output sync signal propagation delay (*1)	Tvpds	1.3		4.6	ns
Vclk to output data signal propagation delay (*1)	Tvpdd	1.1		3.9	ns
Analog output delay time (*2) (to 50% point of full-scale transition)	Tda		1		ns
Analog output settling time (to point of settling within ±1LSB)	Tsa		5		ns

Note *1) These values correspond to the default setting of the register ***vclk_out_dly_ctrl*** (0x024). They can be adjusted by changing the register value.

Note *2) These values correspond to the default setting of the register ***dac_clk_dly_ctrl*** (0x028). They can be adjusted by changing the register value.

Figure 4-2. Video Output Timing Diagram

4.5.3 Audio Input

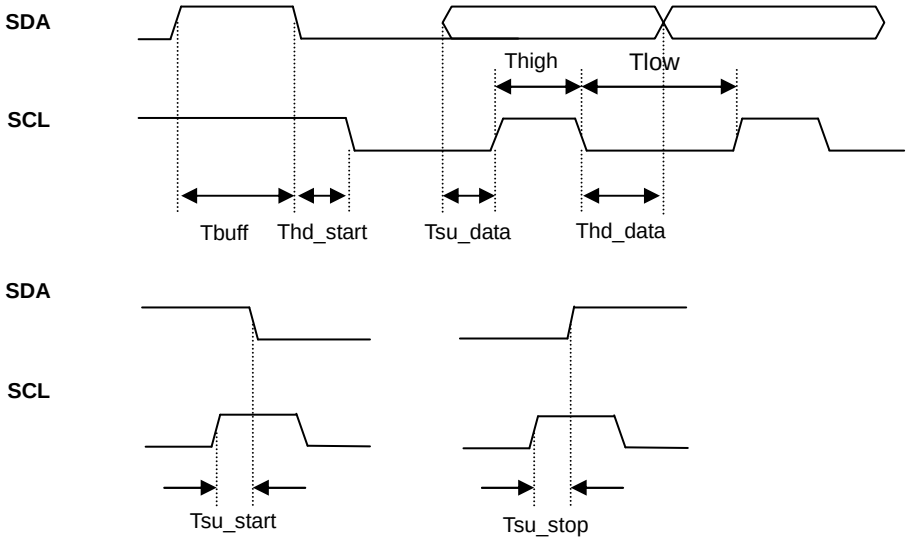


CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Input audio sample rate		32		192	kHz
I^2S cycle time (*1)	Tsck			1.0	UI
I^2S clock duty cycle		45		55	%
I^2S word select signal setup time (*2)	Tssw	15			ns
I^2S word select signal hold time (*2)	Tshw	0			ns
I^2S audio data signal setup time (*2)	Tssd	15			ns
I^2S audio data signal hold time (*2)	Tshd	0			ns
S/PDIF cycle time (*1)				1.0	UI
Master clock (MCLK) cycle time		13.3			ns
Master clock (MCLK) frequency				75	MHz
Master clock (MCLK) duty cycle		40		60	%
Audio pipeline delay			30	70	us

Note *1) Proportional to unit time (UI) according to sample rate.
Note *2) These values correspond to the default setting of the register ***hdmi_sck_dly_sel*** (0x03E). They can be adjusted by changing the register value.
Note *3) Audio pipeline delay measured from transmitter input to TMDS output.

Figure 4-3. Audio Input Timing Diagram

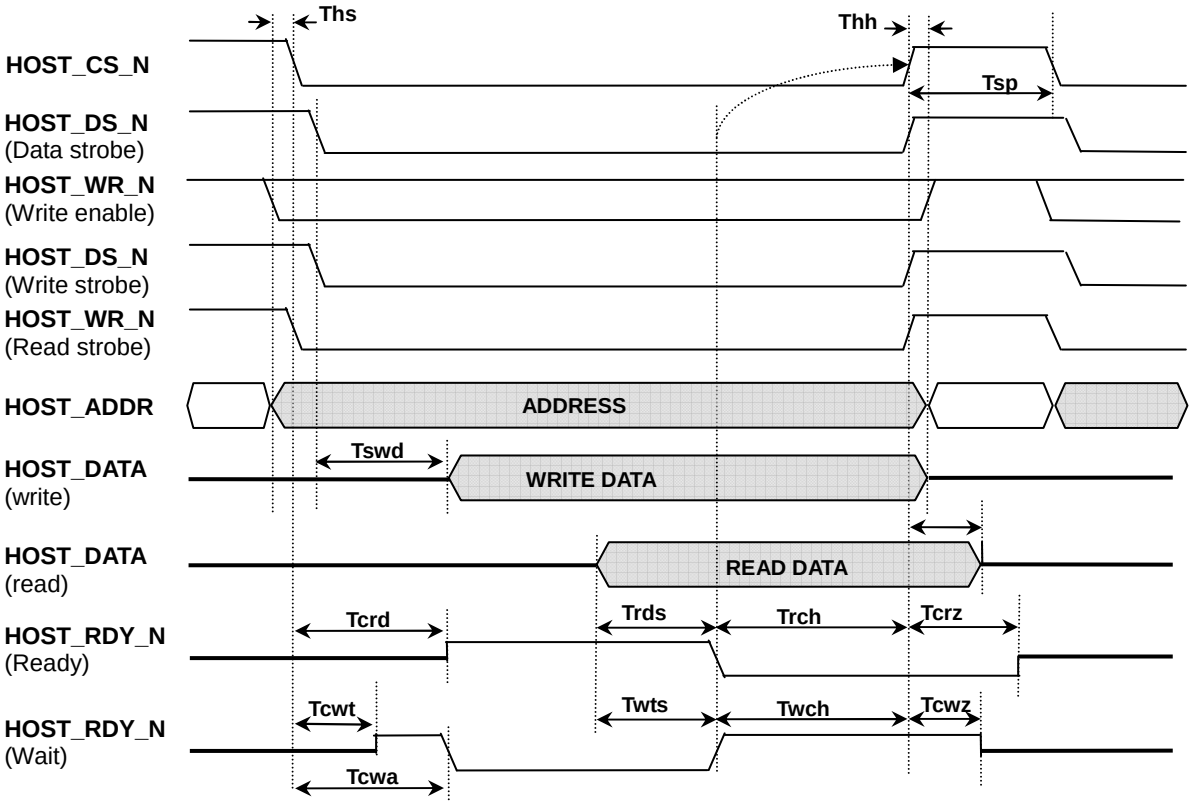
4.5.4 I²C Read/Write



CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
High period of the SCL clock	Thigh	0.6			us
Low period of the SCL clock	Tlow	1.3			us
Bus free time between STOP and START condition	Tbuff	1.3			us
Set-up time for START condition	Tsu_start	0.6			us
Hold time for START condition	Thd_start	0.6			us
Set-up time for STOP condition	Tsu_stop	0.6			us
Data set-up time	Tsu_data	100			ns
Data hold time	Thd_data	0.0		0.9	us
Clock frequency (standard mode)			100		kHz
Clock frequency (fast mode)			400		kHz

Figure 4-4. I²C Read/Write Timing Diagram

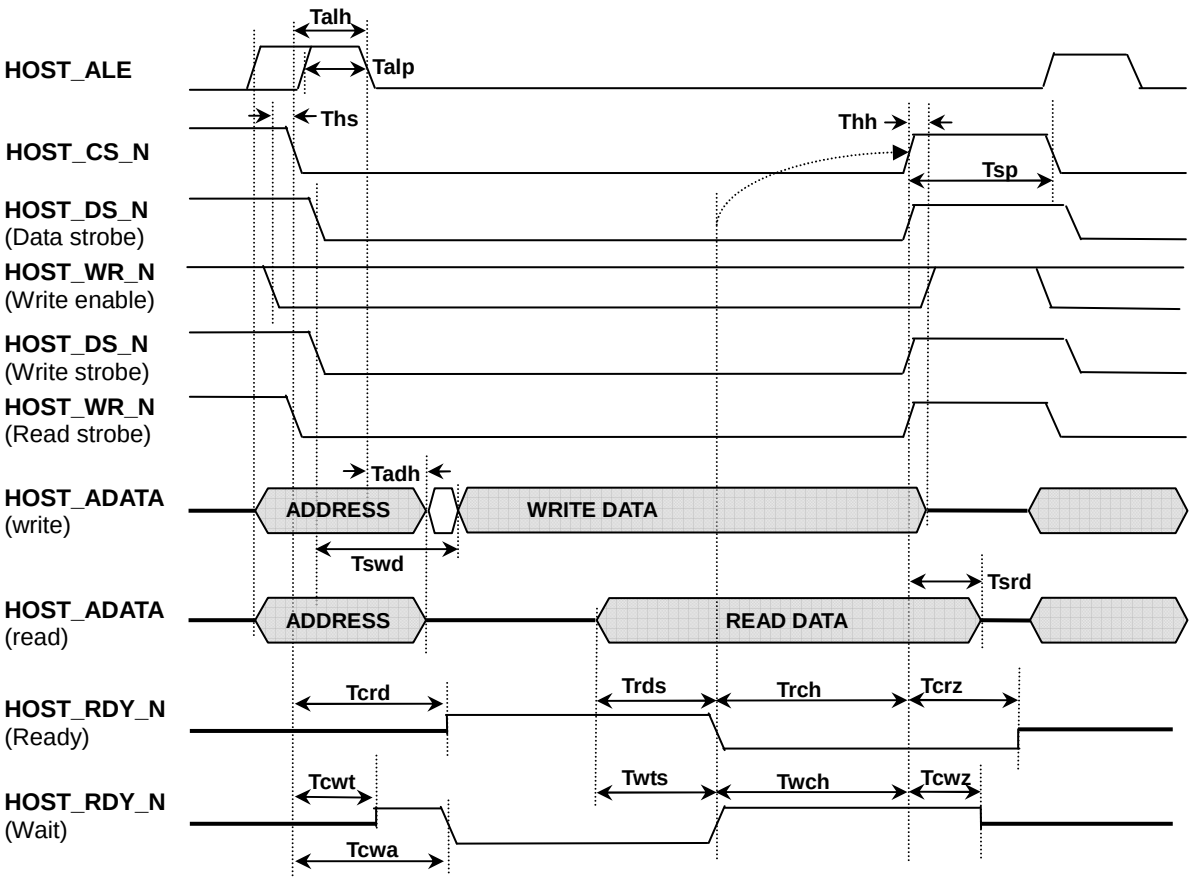
4.5.5 Parallel Host Bus Read/Write



CHARATERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
High period of **HOST_CS_N** signal	Tsp	2			Thclk
Address/Write enable setup time to **HOST_CS_N** Active	Ths	0.0			ns
Address/Write enable/Data hold time from **HOST_CS_N** Inactive	Thh	0.0			ns
Write data valid time after **HOST_DS_N** Active	Tswd			4	Thclk
Read data hold time from **HOST_CS_N** Inactive	Tsrd	2		3	Thclk
HOST_CS_N Active to Ready Enable	Tcrd	1		2	Thclk
Read data valid before Ready Active	Trds		2		Thclk
HOST_CS_N Inactive after Ready Active	Trch	0			ns
HOST_CS_N Inactive to Ready Disable	Tcrz	1		2	Thclk
HOST_CS_N Active to Wait Enable	Tcwt	0		1	Thclk
HOST_CS_N Active to Wait Active	Tcwa	1		2	Thclk
Read data valid before Wait Inactive	Twts		2		Thclk
HOST_CS_N Inactive after Wait Inactive	Twch	0			ns
HOST_CS_N Inacitve to Wait Disable	Tcwz	0		1	Thclk

Note: Thclk means the period of the host inferface clock (**hclk**).

Figure 4-5. Parallel Host Bus Read/Write Timing Diagram (Address/Data Separated Mode)

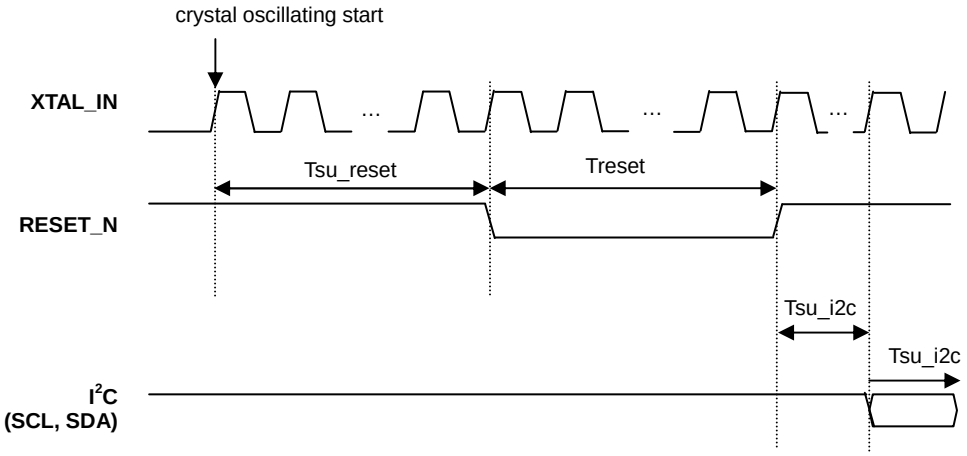


CHARATERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
High period of **HOST_CS_N** signal	Tsp	2			Thclk
Address/Write enable setup time to **HOST_CS_N** Active	Ths	0.0			ns
Address/Write enable/Data hold time from **HOST_CS_N** Inactive	Thh	0.0			ns
HOST_ALE hold time from **HOST_CS_N** Active	Talh	2			ns
High period of **HOST_ALE** signal	Talp	1.5			Thclk
Address hold time from **HOST_ALE** Inactive	Tadh	1.5			Thclk
Write data valid time after **HOST_DS_N** Active	Tswd			4	Thclk
Read data hold time from **HOST_CS_N** Inactive	Tsrd	2		3	Thclk
HOST_CS_N Active to Ready Enable	Tcrd	1		2	Thclk
Read data valid before Ready Active	Trds		2		Thclk
HOST_CS_N Inactive after Ready Active	Trch	0			ns
HOST_CS_N Inactive to Ready Disable	Tcrz	1		2	Thclk
HOST_CS_N Active to Wait Enable	Tcwt	0		1	Thclk
HOST_CS_N Active to Wait Active	Tcwa	1		2	Thclk
Read data valid before Wait Inactive	Twts		2		Thclk
HOST_CS_N Inactive after Wait Inactive	Twch	0			ns
HOST_CS_N Inacitve to Wait Disable	Tcwz	0		1	Thclk

Note: Thclk means the period of the host inferface clock (**hclk**).

Figure 4-6. Parallel Host Bus Read/Write Timing Diagram
(Address/Data Multiplexed Mode)

4.5.6 Reset Timing



CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Setup time for Reset	Tsu_reset	10	-	-	us
Reset period	Treset	100	-	-	us
Setup time for I^2C valid	Tsu_i2c	10	-	-	us

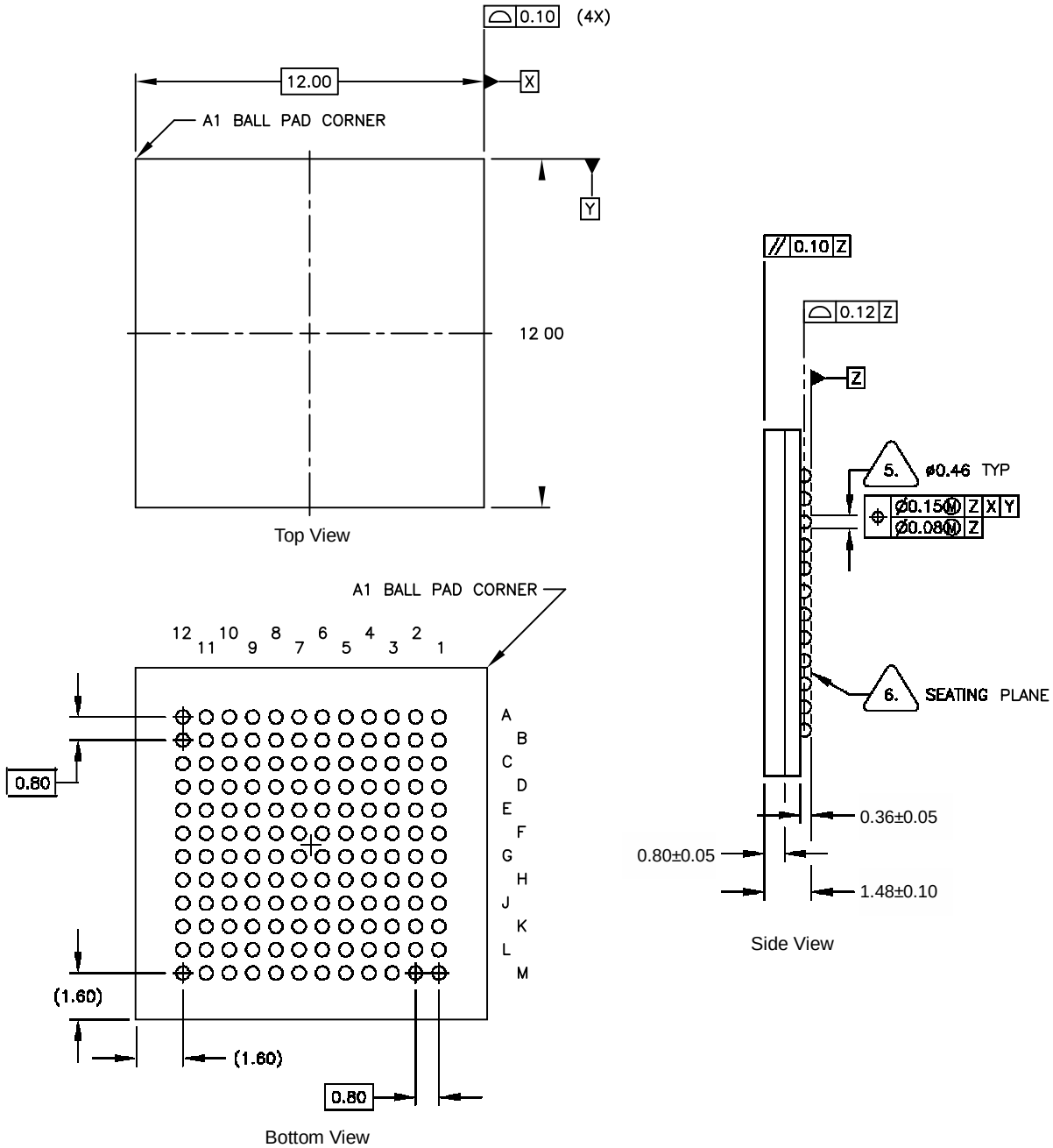
Figure 4-7. Reset Timing Diagram

4.5.7 Internal DDR SDRAM

CHARACTERISTICS	SYMBOL	MIN	TYP	MAX	UNIT
Clock Frequency (MDIN-325/340)	Fmax_ddr1	-	-	166	MHz
Clock Frequency (MDIN-325A/380)	Fmax_ddr2	-	-	200	MHz

5. Package Dimensions

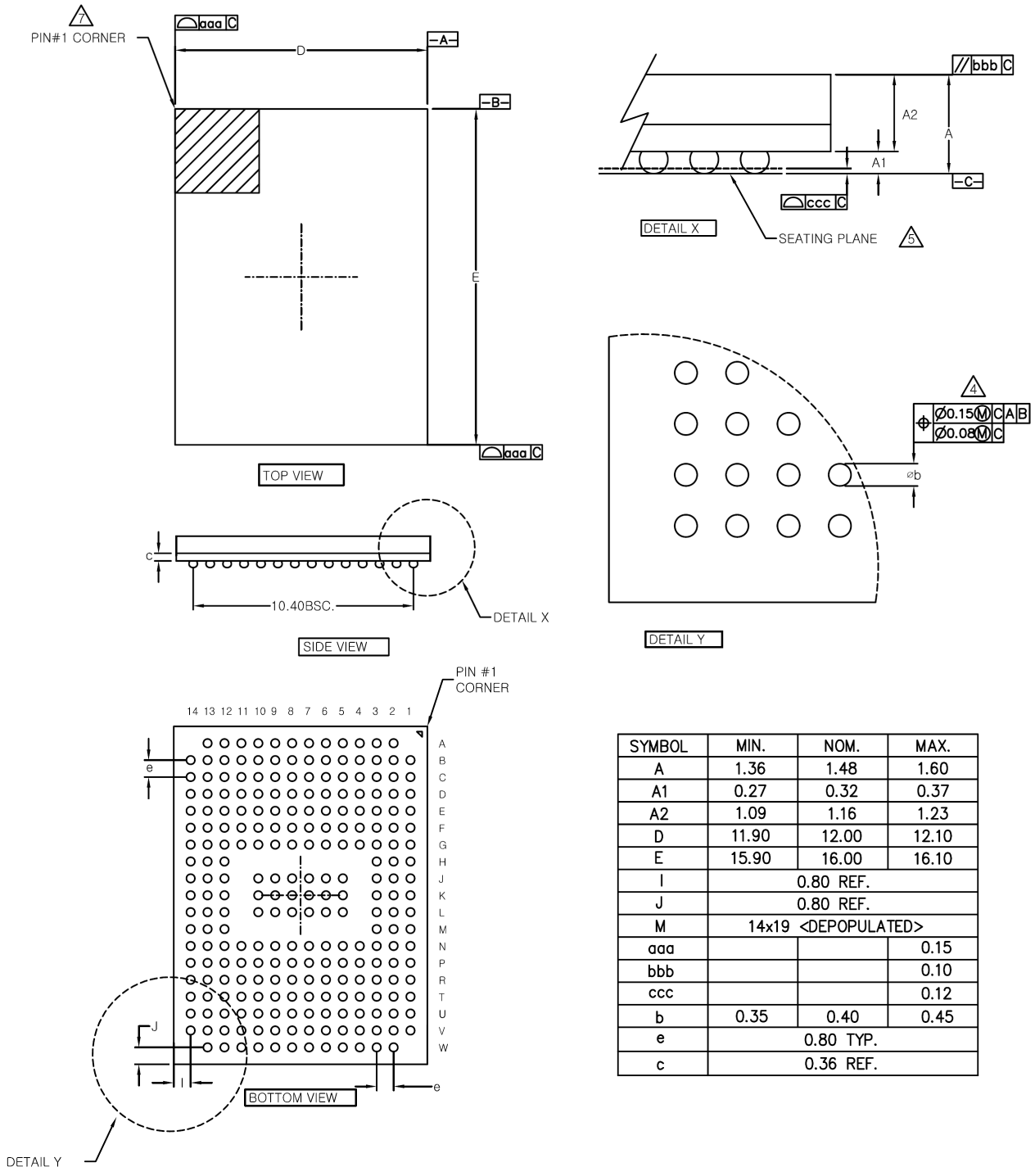
◆ 12x12 FBGA (MDIN-325/340)



Note 1) All dimensions are in millimeters unless otherwise specified.
Note *5) Dimension is measured at the maximum solder ball diameter, parallel to primary datum Z.
Note *6) Primary datum Z and seating plane are defined by the spherical crowns of the solder balls.

Figure 5-1. MDIN-325/340 Package Dimension

◆ 14x19 FBGA (MDIN-380)



SYMBOL	MIN.	NOM.	MAX.
A	1.36	1.48	1.60
A1	0.27	0.32	0.37
A2	1.09	1.16	1.23
D	11.90	12.00	12.10
E	15.90	16.00	16.10
I	0.80 REF.		
J	0.80 REF.		
M	14x19 <DEPOPULATED>		
aaa			0.15
bbb			0.10
ccc			0.12
b	0.35	0.40	0.45
e	0.80 TYP.		
c	0.36 REF.		

Note 1) All dimensions are in millimeters unless otherwise specified.
Note *4) Dimension “b” is measured at the maximum solder ball diameter, parallel to primary datum -C-.
Note *5) Primary datum -C- and seating plane are designed by the spherical crowns of the solder balls.

Figure 5-2. MDIN-380 Package Dimension

6. PCB Layout Design Guideline

MDIN-3xx is a highly integrated circuit containing both precision analog and high speed digital circuitry. The system should be designed to minimize interference effects on the integrity of the analog circuitry by the high speed digital circuitry so that high speed, accurate performance is achieved.
The PCB layout should be optimized for lowest noise on the MDIN-3xx power and ground lines by providing good decoupling. The lead length between groups of VDD and GND pins should be minimized to minimize inductive ringing.

Power Supplies

The analog power pins of MDIN-3xx should be connected to well-regulated power sources at a single point through a ferrite bead. This bead should not be located far from MDIN-3xx. If a high frequency switching power supply is used, the designer should pay close attention to reducing power supply noise and consider using additional three-terminal linear voltage regulators for supplying power to the analog power pins.
The digital power pins also should be connected to low ripple power sources to avoid the coupling of the power noise between the digital power pins and the analog power pins of MDIN-3xx.

Ground Planes

The ground plane should encompass all MDIN-3xx ground pins, voltage reference circuitry, power supply bypass circuitry for MDIN-3xx, the analog output traces, and all the digital signal traces leading up to MDIN-3xx. The ground plane is the board's common ground plane. This should be as substantial as possible to maximize heat spreading and power dissipation on the board.
If the ground planes for the digital circuit and the analog circuit are separate, both ground areas should be connected tightly at one point under MDIN-3xx to avoid the noise from ground bouncing.

Supply Decoupling

Each group of power pins on MDIN-3xx must have at least one decoupling capacitor to the ground plane. Normally, best performance is obtained with 0.1 uF ceramic capacitor decoupling. These capacitors should be placed as close to the device as possible. For optimum performance, bypass capacitors should be connected using as short as traces possible without any vias to reduce the trace inductance.

Digital Signal Interconnect

The digital inputs, the digital output signals of MDIN-3xx should be isolated as much as possible from the analog outputs and other analog circuitry. Also, these signals should not overlay the analog power or ground plane.
When the data rate of the video output is very high, an attention should be paied to the clock and the data signals of the connection between MDIN-3xx and the connected device.

DAC Signal Interconnect

MDIN-3xx should be located as close to the video output connectors as possible to minimize noise pickup and reflections due to impedance mismatch. The video output signals should overlay the ground plane, not the analog power plane, to maximize the high frequency power supply rejection. Digital signals, especially video, and clocking signals, should never overlay any of the analog signal circuitry and should be kept as far away as possible.
For best performance, the DAC outputs should each have a proper load resistor connected to GND. These resistors should be placed as close as possible to MDIN-3xx to minimize reflections.

PLL Signal Interconnect

The noise on the power supply to the PLL circuitry should be minimized to minimize the jitter on the clock signal. The jitter on the video output clock will have a great impact on the video quality. Close attention should be paied on suppressing the ripple noise and decoupling the PLL power supply pins.
Digital signals, especially video, and clocking signals, should never overlay the PLL circuitry and should be kept as far away as possible.

Revision History

Version	Date	Description
V0.1	2010. 09. 15	First preliminary version
V0.2	2010. 09. 27	In Table 1-2 & 1-3 - Add 5V-tolerant to HDMI_HPD pin - Add Schmitt trigger & open drain to HDMI_CEC pin In Table 1-5 Fix the I/O type of HDMI_HPD from I to O In Table 1-6 Fix the I/O type of DAC_AVDD33(0) from GND to VDD Add the I/O type to DAC_AGND(0) Fix the I/O type of HDMI_HPD from I to O Delete “This 3-D noise reduction works only for interlaced input video of standard definition. It does not work for progressive video.” from 2.11.7 Fix 8-bit alpha to 4-bit alpha for 32-bit ARGB in 2.15.3.2 Fix “output of main video scaler” to “output of aux video scaler” in 2.18 Fix “Figure 4-19” to “Figure 3-24” in 3.3.1 Fix “Main video output” to “Aux video output” in 3.3.6
V0.3	2010. 10. 20	Add bus cycle diagrams for parallel host interface.
V0.4	2011. 03. 11	Fix frame buffer calculation in 2.10 Frame Buffer Memory Add new section 2.19 Multi-channel Mode Add restrictions on CLK_A/B for main & aux path in 3.3.4 Change the numbering of Figure 5-1 to 5-4 Add vclk_int to Figure 3-30 & 3-31 Fix 1.1V to 1.2V in the signal description of DAC_VREF & VENC_VREF Fix type error Single for dual input modes in Table 2-1, 2-2 Fix bit mapping of 20-bit & 16-bit modes in Table 2-6 Add the pin functions of HACT_B, VACT_B and FID_B to the MDIN-380 pin descriptions.
V0.5	2011. 03. 17	Fix aux output port bit mapping of 16-bit mode in Table 2-16
V0.6	2011. 05. 30	Add new device MDIN-325A
V0.7	2012. 04. 01	Delete 4:4:4 processing & BNR Add maximum input & output video size Add Power Supply Pin Characteristics Add AC charateristics of DDR SDRAM Add Video Output Block Diagram Remove inter-only deinterlacing registers descriptions Remove block OSD registers descriptions Remove DAC control registers descriptions, add line format waveforms Add description for TEST_MODE(2:1) = ‘11’ Fix F_{IN} => (F_{IN} / P), F_{OUT} => (F_{OUT} * S) in Figure 3-29 Fix dac_input_sel => dac_clk_sel & dac_data_sel in 2.18.10 Fix typo in Figure 3-33(a) Fix errors in DC Electrical Characteristics Delete HDMI_VDU12 in Electrical Characteristics Fix vclk_pll => “vclk PLL VCO” in Figure 3-31.