MAX96751 User Guide

UG-2228

Revision 0

Rev 0; 4/24

TABLE OF CONTENTS

About the User Guide

This user guide is intended to be used in conjunction with other documents such as the device specific data sheets, errata documents, and other user and design guides. It provides explanations, examples, and instructions to help setup video configurations and use various features.

Examples shown in this guide do not consider errata fixes that may be necessary to ensure reliable operation in production. To obtain errata documents, on the Analog Devices website, search for a specific part, and review the Documentation section. Make sure to include any relevant errata writes in the final production software. In addition to the errata, it is also important to have the latest revision of the GMSL device for testing.

Device Overview

GMSL2 serial links use packet-based, bidirectional architecture with forward and reverse channels. The forward channel transfers data from the serializer to the deserializer; the reverse channel transfers data from the deserializer to the serializer. GMSL2 devices have a forward serial bit rate of 6Gbps or 3Gbps and reverse channel serial bit rate of 187.5Mbps. See below for supported data rates/features by part number.

The MAX96751 is a full-featured GMSL2 serializer device. The MAX96751 is capable of 6Gbps or 3Gbps forward link rate (selectable by pulling the CXTP pin up or down and selectable with register writes). This device has a 187.5Mbps reverse direction rate.

Table 1. Comparison of the MAX96751 Family High Level Features

Part Number	Forward Link Rate	Coax/STP	Input	GMSL Links	ASIL Rating
MAX96751	3 or 6Gbps	Coax or STP	HDMI v1.4b/v2.0a	2	B

Note: This is a not a complete list of device features. See the device data sheets for all feature details.

The MAX96751 has 5 two-state configuration pins for bootstrapping the device address, GMSL link rate, STP/COAX modes, and I2C/UART Control Channel modes.

CXTP pin selects coaxial operation at 6Gbps when set high or selects shielded-twisted pair operational mode at 3Gbps when set low.

I2CSEL pin selects I2C operation of the Primary Control Channel when set high or selects UART operation of the Primary Control Channel when set low.

ADD0, ADD1, and ADD2 pins allow configuration of different device addresses.

Start-up and Programming Sequence

Register Configuration Sequencing – MAX96751

Aside from the sequencing required for the GMSL link and video throughput, there are sequencing requirements for the register configurations of features in GMSL devices. The register configuration sequences are divided into groups. The configuration groups must be done in sequence as described below, but the sequence of configurations within each group is flexible.

Table 2. Configuration Sequencing

Feature/Event	Notes	Sequence Group
Power Up*	Power the device and raise PWDNB.	0
HDMI Source Power Detect (HSPD)*	HSPD is an HDMI port "connected" pin. The HDMI RX IP in the serializer does not run until it sees HSPD=1.	0
Link Reset*	Hold link in reset during configuration to prevent multi-master I2C conflicts with remote side of the GMSL link. RESET_LINK = 1	1
HDMI RX IP HPD = 0	HPD should be driven low by writing 0x20F5=0 in Consumer Port applications, before initializing the EDID memory.	2
Initialize HDMI EDID memory, if needed	For applications where the external HDMI source needs to read the serializer EDID to determine its capabilities, initialize the EDID memory at this time, while HPD=0.	3
HDMI 1.4 processing	HDMI RX IP by default expects HDMI 1.4 signaling on TMDS.	5
HDMI RX IP asserts HPD = 1	Reassert HPD high to HDMI source by writing 0x20F5=0x01, which will initialize HDMI TX to read EDID from serializer.	7
Disable SER VID_EN_X	SER ECU sequence is required to include this before enabling video input into SER. This is required to not inject unstable video into eDP deserializer. Can be done ahead of setting RESET_LINK = 1.	8
Video Source: Enable Video into SER	Video source can be enabled at any time and is required to be stable before enabling SER video transmission over GMSL link (VID_EN_X=1).	9
HDMI 2.0 processing	For HDMI 2.0, configure set of HDMI RX registers.	10
Set HDMI_AUTOS = 1*	Write HDMI_AUTOS = 1 to allow HDMI RX IP to start processing HDMI TMDS stream. For HDMI 2.0, set HDMI_AUTOS=1 after set of configuration registers in HDMI RX IP programmed for HDMI 2.0. HDMI RX IP outputs erroneous video signals for up to 33ms whenever HDMI_AUTOS changes to 1.	11
Configure Main I2C / UART Config and CRC	Configuration changes from default (e.g., CRC ARQ)	12
Configure I2C / UART PT Config	Configuration and enabling of passthrough channels.	13
GPIO Config		14
I2S Config	Audio routing, Stream ID, enable, etc.	15
SPI Config and CRC		15
V-sync and H-sync Output Config	Debug feature route VS and HS (after HDMI RX, VTX) to respective GPIO pins – enable if needed. Should not be used in production.	15
GMSL PHY Config		15
Video Pipe Config	(Stream ID Config, Video Duplication)	15
Watermark	Should be configured and enabled before video.	15

Interrupt handling (ERRB)	Location should be chosen by the customer depending on what ERRB outputs are being monitored. Enable selected errors to the ERRB pin for system error handling and ASIL requirements Depending on the configuration chosen, may need to clear initial conditions (may need to be done after links are locked).	19
Release Link Reset	RESET_LINK = 0	20
GMSL Link Lock obtained*		21
Wait for 2 frame periods from HDMI_AUTOS=1, or starting video source, whichever is later. Enable VID_EN_X*	eDP deserializer DPTX may not recover if it is exposed to unstable video. SER side ECU needs to make sure that the provided video is stable. Set VID_EN_X only after the previous step (GMSL link locked and FEC enabled, if needed).	23

*These are the minimum configurations that need to be done to achieve video throughput through the MAX96751 deserializer.

Dual-View

Dual-view is a configurable feature that enables a single serializer to support two different video streams. A serializer with dual-view enabled can receive a video stream comprising two videos combined into a single frame. This combined frame is called a superframe.

The dual-view block takes the incoming superframe video stream, splits it into two video streams according to the enabled mode, and outputs the video onto two independent GMSL stream IDs. The output video streams can then be routed to one or more deserializers depending on the application.

In the example below, a superframe of 3840x1200 at 60Hz is created in the SoC by combining two videos side-by-side into a single frame. The superframe is output from the SoC and received by the serializer. The serializer, with symmetric dual-view enabled, splits the superframe into two separate video streams marked as Video A and Video B in *Figure 1*. Each of these streams is 1920x1200 at 60Hz.

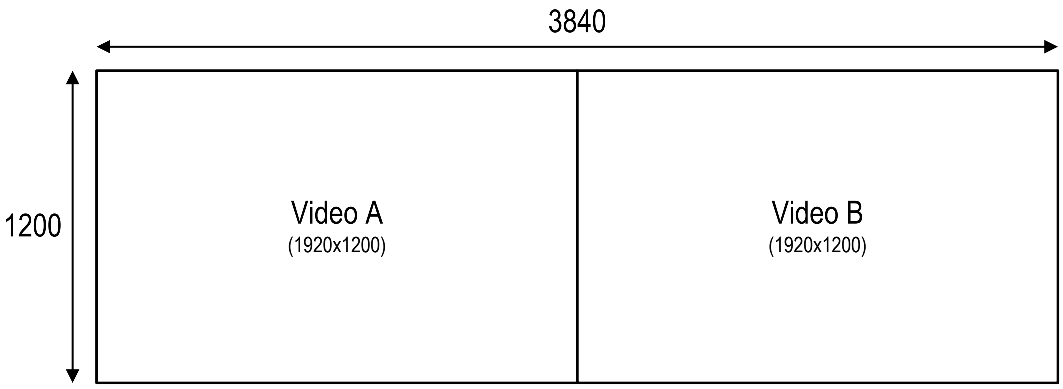


Figure 1. Example of Side-by-Side Symmetric Dual-View

Symmetric Dual-view modes require that constituent videos (i.e., the subframes) of the incoming superframe have identical timing (i.e., active resolution, blanking interval, and frame rate). Three types of symmetric dual-view superframes are supported:

- Pixel-interleaved Dual-view mode
- Side-by-side Dual-view mode
- Line-interleaved Dual-view mode

The choice of Dual-view mode is dependent on the superframe timing as output by the SoC and received by the serializer. In each mode, the serializer should be configured to match the superframe that was created by the SoC. When configured, the superframe is split into two symmetrical video frames by the dual-view block of the serializer and output onto two independent GMSL stream IDs. *Figure 2* shows the timings for each subframe, Video A and Video B.

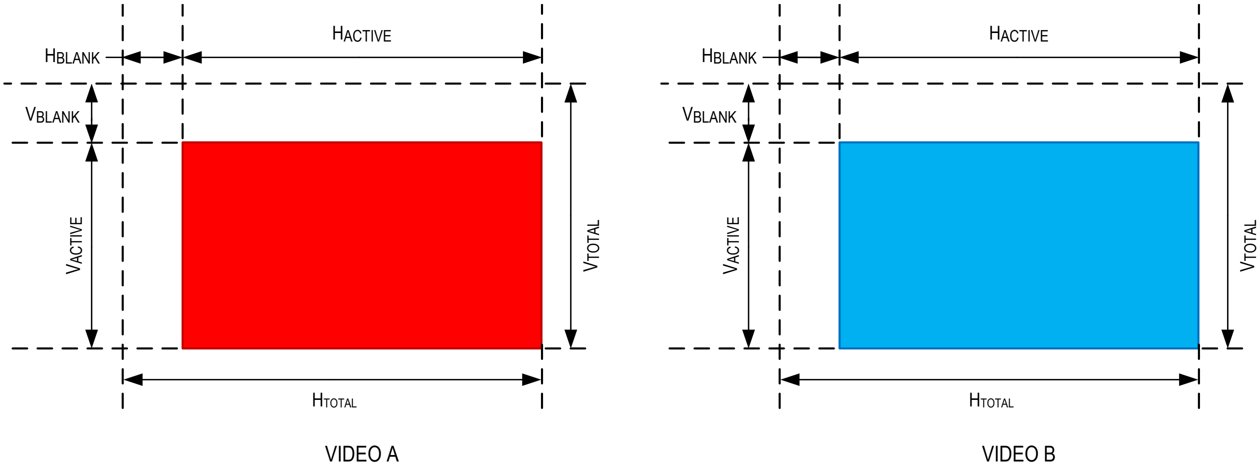


Figure 2.Single Frame Timing

Pixel-Interleaved Dual-View Mode

Pixel-interleaved mode (also known as Column-interleaved mode) separates the video stream by assigning every odd pixel to one stream and every even pixel to another stream (*Figure 3*). This includes the horizontal blanking, which is split in half and symmetrically distributed to the two video streams. Vertical blanking timing is maintained from superframe to each subframe video stream. The PCLK is divided by two for each subframe.

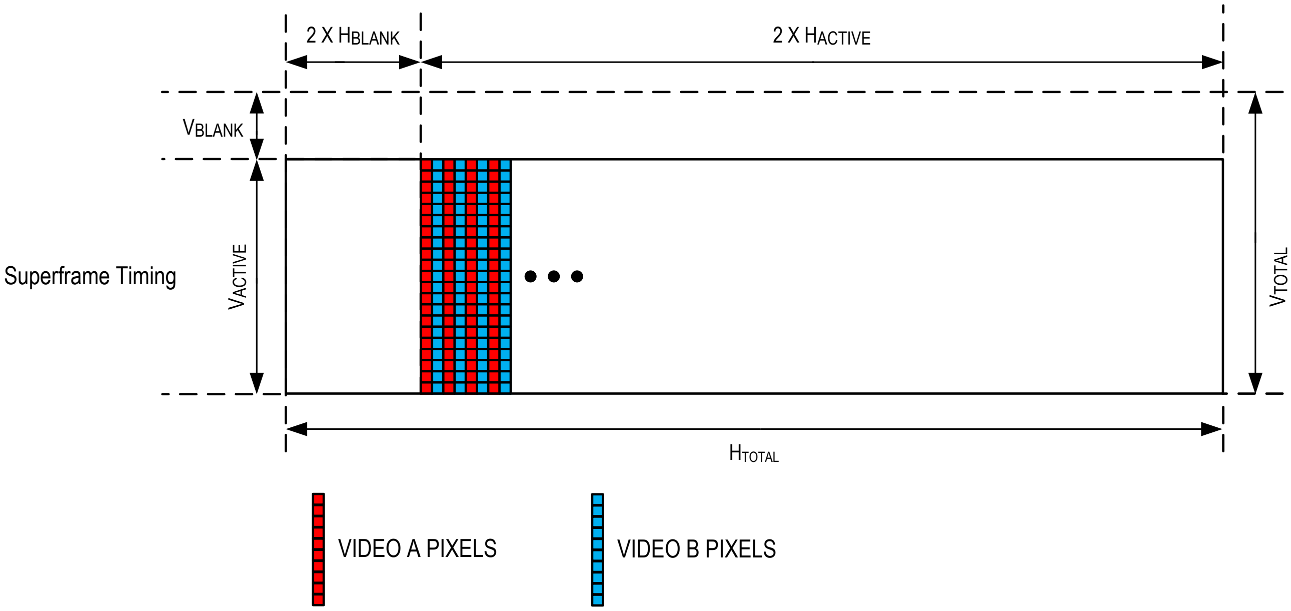


Figure 3.Pixel-Interleaved Dual-View Mode

Side-by-Side Dual-View Mode

Side-by-side mode is used to split the superframe in half by creating subframes from the left and right halves of the superframe as shown in *Figure 4*. Side-by-side Dual-view mode works by first converting the superframe data into pixel-interleaved data. Then, the pixel-interleaved data is separated by assigning every odd pixel to one stream and every even pixel to another stream. When side-by-side Dual-view mode is enabled, the dual-view block measures the length of the superframe line that contains the side-by-side image format. Assuming that the pixel-length of the line is 2N, the pixels are reordered in the following sequencing:

0,N,1,N+1,……………………………………………..N-2,2N-2,N-1,2N-1

The odd pixels are sent to one video stream and even are sent to another stream.

Horizontal blanking is split symmetrically between streams, and vertical timing is maintained from superframe to the two subframe video streams. The PCLK is divided by two for each subframe.

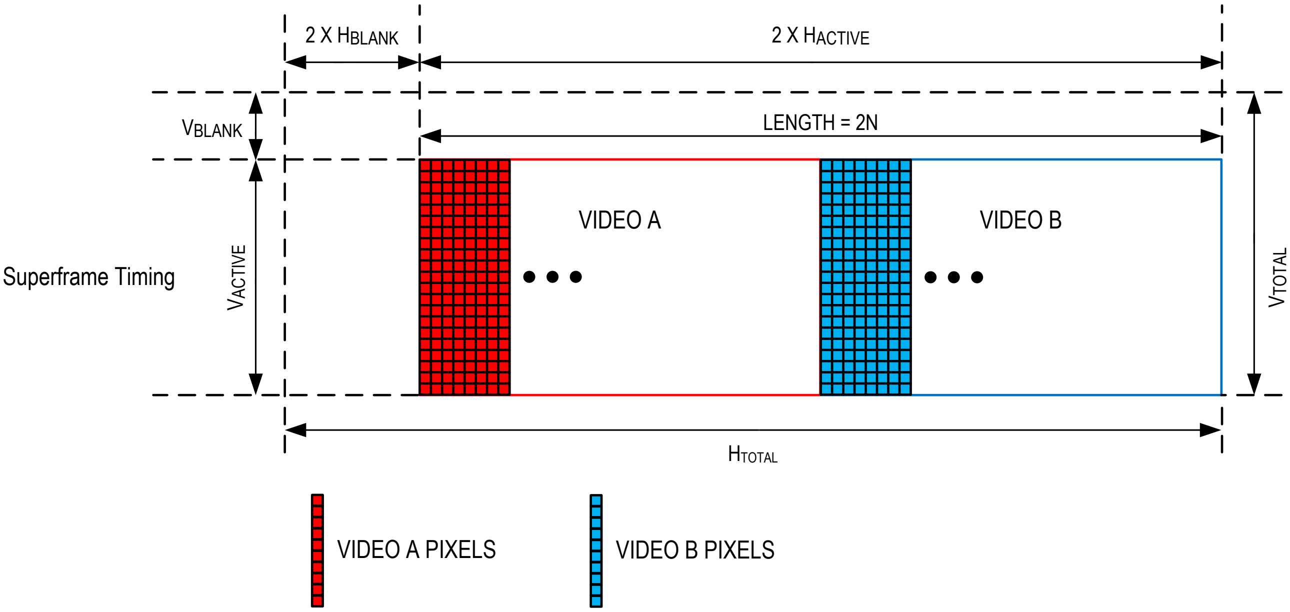


Figure 4. Side-by-Side Dual-View Mode

Line Interleaved Dual-View Mode

In Line-interleaved mode, the incoming data is split horizontally line-by-line (*Figure 5*). Odd lines (beginning with the first active line of a frame) contain Video A; even lines contain Video B. When Line-interleaved mode is enabled, odd lines are split into one video stream and even into the other. Horizontal blanking timing is maintained from superframe to subframe, but vertical timing is symmetrically split between streams. The PCLK is divided by two for the subframes.

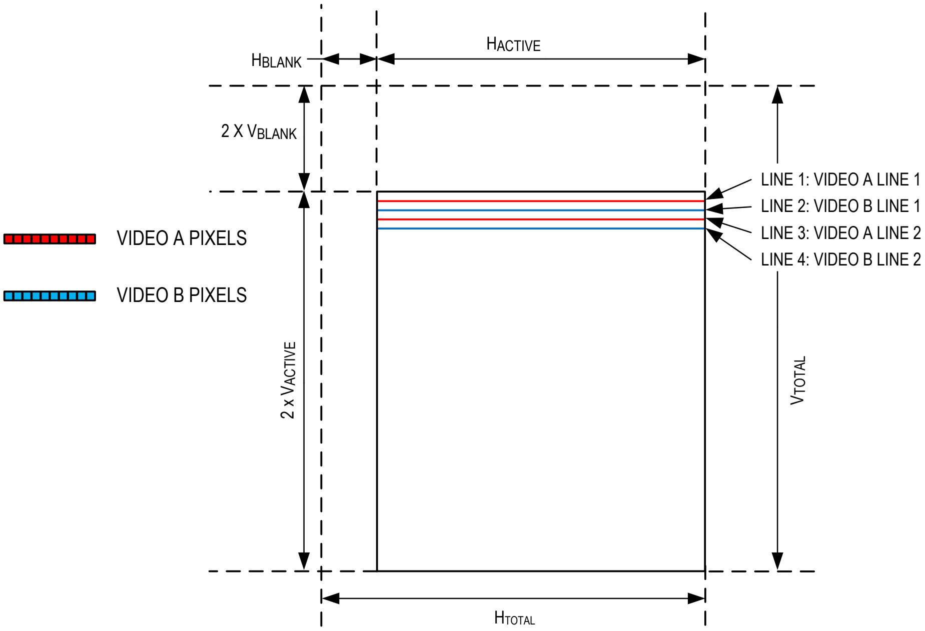


Figure 5. Line-Interleaved Dual-View Mode

Dual-View Setup Registers

Table 3. MAX96751 Dual View Setup Registers

Register Address	Bitfield Name	Bits	POR	Decode
0x1A0	DV_EN	0	0b0	0b0 Dual-view processing disabled 0b1 Dual-view processing enabled
0x1A0	DV_SPL	1	0b0	0b0 Dual-view split disabled 0b1 Dual-view split enabled
0x1A0	DV_CONV	2	0b1	0b0 Bypass 0b1 Convert side-by-side to pixel interleaved
0x1A0	DV_SWAP_AB	5	0b0	0b0 Normal video split between SIOA and SIOB 0b1 Inverts video split between SIOA and SIOB
0x1A0	LINE_ALT	6	0b0	0b0 Line alternating mode disabled 0b1 Line alternating mode enabled
0x1A2	VID_EN_X	5	0b1	0b0 Split video on SIOA disabled 0b1 Split video on SIOA enabled
0x1A2	VID_EN_Y	6	0b0	0b0 Split video on SIOB disabled 0b1 Split video on SIOB enabled

Configuration

The configuration process for enabling Pixel-interleaved Dual-view mode is presented in *Table 4*.

Table 4. Pixel-Interleaved Dual-View Configuration Steps

Step	Description	Register Name	Register Write
1	Enable dual-view block	DV0	DV_EN = 1
2	Split the video into two streams, outputting on pipe X and pipe Y	DV0	DV_SPL = 1
3	Select which pipe receives even/odd pixels	DV0	DV_SWP_AB = 0 sends odd pixels to pipe X and even pixels to pipe Y DV_SWP_AB = 1 sends even pixels to pipe X and odd pixels to pipe Y
4	Enable video stream on pipe X (default)	DV2	VID_EN_X = 1
5	Enable video stream on pipe Y	DV2	VID_EN_Y = 1
6	Assign pipe X stream ID and select which PHY to send video	VIDEO_X TX3	set TX_STR_SEL, TX_SPLT_MASK_A and TX_SPLT_MASK_B
7	Assign pipe Y stream ID and select which PHY to send video	VIDEO_Y TX3	set TX_STR_SEL, TX_SPLT_MASK_A and TX_SPLT_MASK_B

The configuration process for enabling side-by-side Dual-view mode is presented in *Table 5*.

Table 5. Side-by-Side Dual-View Configuration Steps

Step	Description	Register Name	Register Write
1	Enable dual-view block	DV0	DV_EN = 1
2	Convert side-by-side to pixel interleaved video stream	DV0	DV_CONV = 1
3	Split the video into two streams, outputting on pipe X and pipe Y	DV0	DV_SPL = 1
4	Select which pipe receives even/odd pixels	DV0	DV_SWP_AB = 0 sends odd pixels to pipe X and even pixels to pipe Y DV_SWP_AB = 1 sends even pixels to pipe X and odd pixels to pipe Y
5	Enable video stream on pipe X (default)	DV2	VID_EN_X = 1
6	Enable video stream on pipe Y	DV2	VID_EN_Y = 1
7	Assign pipe X stream ID and select which PHY to send video	VIDEO_X TX3	set TX_STR_SEL, TX_SPLT_MASK_A and TX_SPLT_MASK_B
8	Assign pipe Y stream ID and select which PHY to send video	VIDEO_Y TX3	set TX_STR_SEL, TX_SPLT_MASK_A and TX_SPLT_MASK_B

Configuring Line-interleaved Dual-view requires configuring the dual-view block as well as programming the *Video Timing Generator (VTG)*. These settings are defined in the table below.

Table 6. Line-Interleaved Dual-View Configuration Steps

Step	Description	Register Name	Register Write
1	Enable Line-interleaved Dual-view mode	DV0	LINE_ALT = 1
2	Split the video into 2 streams, outputting on pipe X and pipe Y	DV0	DV_SPL = 1
3	Select which pipe receives even/odd pixels	DV0	DV_SWP_AB = 0 sends odd lines to pipe X and even pixels to pipe Y DV_SWP_AB = 1 sends even lines to pipe X and odd pixels to pipe Y
4	Enable video stream on pipe X (default)	DV2	VID_EN_X = 1
5	Enable video stream on pipe Y	DV2	VID_EN_Y = 1

Step	Description	Register Name	Register Write
6	Assign pipe X stream ID and select which PHY to send video	VIDEO_X TX3	set TX_STR_SEL, TX_SPLT_MASK_A and TX_SPLT_MASK_B
7	Assign pipe Y stream ID and select which PHY to send video	VIDEO_Y TX3	set TX_STR_SEL, TX_SPLT_MASK_A and TX_SPLT_MASK_B

8	**VTG**: Enable VTG, set to VS one-trigger mode, and generate DE, HS, and VS signals	VTX0	VTG_MODE[1:0] = 01
			GEN_DE = 1
			GEN_HS = 1
			GEN_VS = 1

9	**VTG**: Trigger VS on rising edge	VTX1	VS_TRIG = 1
10	**VTG**: Time between VS edge of superframe and VTX VS generation. Used as a buffer for making sure first line of superframe is written to the memory before starting to read. (1 line + 40 pixels)	VTX2 - VTX4	VS_DLY = Htot + 40
11	**VTG**: VS high time of the superframe in terms of superframe pixels	VTX5 - VTX7	VS_HIGH = Vsw * Htot
12	**VTG**: VS low time of the superframe in terms of superframe pixels	VTX8 - VTX10	VS_LOW = (Vtot - Vsw) * Htot
13	**VTG**: Time between VS edge of superframe and first VTX HS generation. Must be equal to VS_DLY to keep VS and HS aligned.	VTX11 - VTX13	V2H = Htot + 40
14	**VTG**: Double HS high time in terms of superframe pixels	VTX14 - VTX15	HS_HIGH = 2 * Hsw
15	**VTG**: Double HS low time in terms of superframe pixels	VTX16 - VTX17	HS_LOW = 2 * (Htot - Hsw)
16	**VTG**: Half the number of HS pulses in the superframe	VTX18 - VTX19	HS_CNT = Vtot / 2
17	**VTG**: Time between VS rising edge of superframe and first VTG DE generation. This value is equal to VS_DLY + vertical sync width plus back porch time + two horizontal sync width plus horizontal back porch time	VTX20 - VTX22	V2D = (Htot + 40) + Htot * (Vsw + Vbp) + 2 * (Hsw + Hbp)

Step	Description	Register Name	Register Write
18	**VTG**: Two times the DE high time in a superframe line in terms of superframe pixels	VTX23 - VTX24	DE_HIGH = 2 * Ha
19	**VTG**: Two times the DE low time in a superframe line in terms of superframe pixels	VTX25 - VTX26	DE_LOW = 2 * (Hfp + Hsw + Hbp)
20	**VTG**: Half the number of active lines in the superframe	VTX27 - VTX28	DE_CNT = Vactive / 2
21	Enable dual-view block	DV0	DV_EN = 1

Line Alternating Dual-View Splitter Can Cause Display Swap

For use cases where dual-view line alternating mode splitter is used, the video in displays might be swapped intermittently with a probability of 1/Htotal.

The following method should be applied every time the video pipeline restarts inside the serializer. Note that the script format below is defined as [device address, register address, data] and assumes the serializer is set to I^{2}C address 0x80.

```
# Configure the system
# Start HDMI video
# Wait for HDMI_SCDT,HDMI_CKDT,PCLKDET to be asserted
# Write sel_ext_pclk=1
0x80,0x01C9,0xF1
# force vs=0, de=0
0x80,0x01F2,0x3A
0x80,0x01F1,0x39
0x80,0x2A0C,0x02
# turn off dualview
0x80,0x01A0,0x26
# write sel_ext_pclk=0
0x80,0x01C9,0xE1
# release force on vs and de
0x80,0x01F1,0x19
0x80,0x01F2,0x1A
0x80,0x2A0C,0x00
# turn on dualview
0x80,0x01A0,0x27
```

Video Configuration and Routing

Video Pipe Selection

Video pipes must be configured to match video streams between the deserializer and serializer. This programming step is typically done following link initialization and ensures that the deserializer properly receives video data from the serializer. By default, the deserializer is programmed to accept the most common stream from the serializers, and configuration is not usually needed.

Video Pipe Routing Registers

Select the serializer stream ID to match the video streams (STR_ID) from the serializers for each deserializer video pipe. GMSL serializers can have up to four video pipes (X, Y, Z, and U). These are annotated as 2 bits representing the stream ID (Ex. Pipe X = 0b00, Pipe Y = 0b01, Pipe Z = 0b10, and Pipe U = 0b11). GMSL display deserializers have usually one video pipe which is also annotated as 2 bits representing the stream ID.

By default, the MAX96751 serializer video pipes are turned on and stream ID 0b00 in Pipe X and 0b01 in Pipe Y are selected.

Table 7. MAX96751 Video Pipe Routing Registers

Register Address	Bitfield Name	Bits	POR	Decode
0x53	TX_STR_SEL[1:0] (Pipe X)	1:0	0b00	0b00: str_id=00 0b01: str_id=01 0b10: str_id=10 0b11: str_id=11
0x57	TX_STR_SEL[1:0] (Pipe Y)	1:0	0b01	0b00: str_id=00 0b01: str_id=01 0b10: str_id=10 0b11: str_id=11
0x53	TX_SPLT_MASK_A[0] (Pipe X)	4	0b1	0b0 Packets are not transmitted over GMSLA in Splitter mode 0b1 Packets are transmitted over GMSLA in Splitter mode
0x53	TX_SPLT_MASK_B[0] (Pipe X)	5	0b0	0b0 Packets are not transmitted over GMSLB in Splitter mode 0b1 Packets are transmitted over GMSLB in Splitter mode
0x57	TX_SPLT_MASK_A[0] (Pipe Y)	4	0b0	0b0 Packets are not transmitted over GMSLA in Splitter mode 0b1 Packets are transmitted over GMSLA in Splitter mode
0x57	TX_SPLT_MASK_B[0] (Pipe Y)	5	0b1	0b0 Packets are not transmitted over GMSLB in Splitter mode 0b1 Packets are transmitted over GMSLB in Splitter mode

HDMI Serializer Video Configuration and Routing

All GMSL HDMI serializers convert audio and RGB888 video from HDMI 1.4 or 2.0 input to single or dual GMSL output. HDMI audios receive, and transmission over the GMSL forward channel is also supported. To enable the HDMI video receiver on the GMSL HDMI serializer, set HDMI_AUTOS = 1 in REG1 (0x01).

The functional block diagram (*Figure 6*) shows the video path from the HDMI input pins to the video pipes for MAX96751.

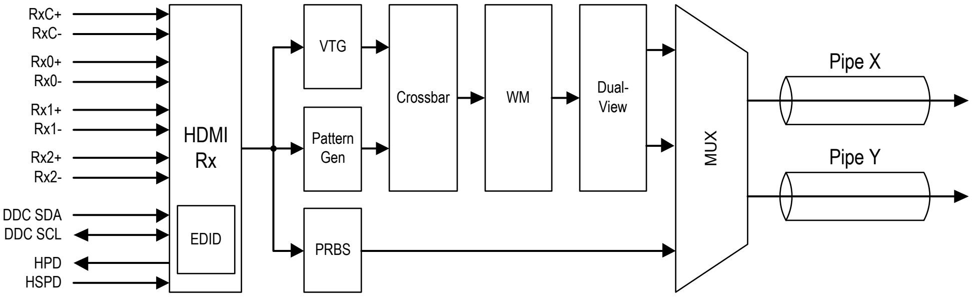


Figure 6.Block Diagram of the Video Path Through GMSL2 HDMI Serializer

Embedded Applications

GMSL2 HDMI serializers are most used in embedded applications where the video source (typically an SoC) is installed on the same PCB as the GMSL2 serializer (*Figure 7*). In these implementations, a PCB trace provides the HDMI connection from the video source to the serializer; the subsystem is contained within the PCB, and there are no external off-board connections. This use case requires careful consideration of PCB design and layout to ensure good signal quality on the HDMI connection. HDMI compliance testing is not required for this application.

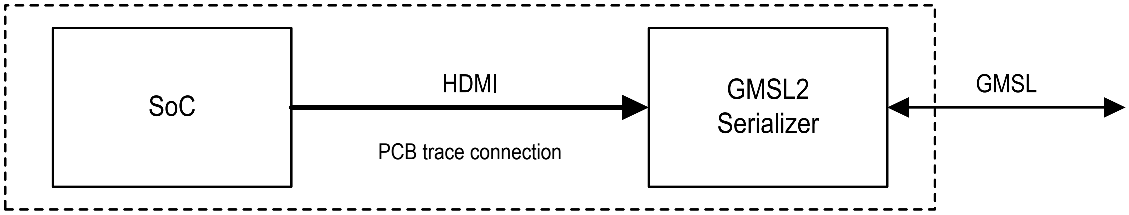


Figure 7.Embedded HDMI Application

Programming HDMI EDID Table

GMSL2 serializers can store EDID data and share it with the source device (*Figure 8*).

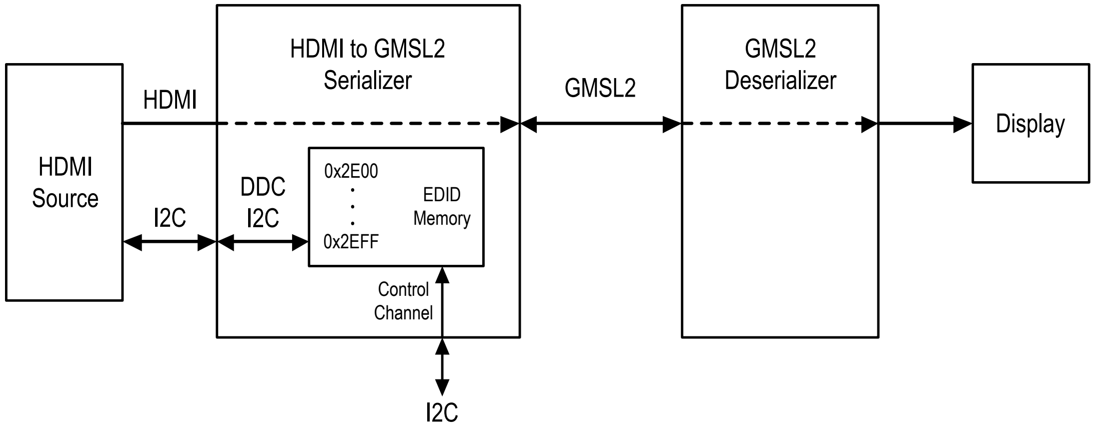


Figure 8. EDID Memory in the Serializer HDMI Rx Block

The HDMI source must access the EDID data through the display data channel (DDC) at the HDMI interface. The GMSL2 HDMI serializer acts as the sync device and functionally represents the display to the HDMI source. The GMSL2 HDMI serializer provides 256 registers for EDID storage (accommodating EDID structure v2.0). The EDID data is written to volatile memory space in the serializer through the device's primary I^{2}C or UART interface. The EDID register block is 0x2E00–0x2EFF. EDID data must be written to the serializer whenever the device is power cycled. When the HDMI source device in the HDMI link detects hot plug detect high (HPD), it reads the EDID table through the DDC_SDA and DDC_SCL lines of the HDMI interface.

Configuration steps are:

1. Write register to drive HPD pin low (register 0x20F5 set to 0x00).
2. Write EDID table (registers 0x2E00–0x2EFF).
3. Write register to drive HPD pin high (register 0x20F5 set to 0x01).

By toggling the HPD pin, the source recognizes that there is a new EDID table to read. If HPD is not cycled low then high, the source device may not re-read the new EDID table.

HDMI source devices do not use a universal method of polling the status of the HPD pin. Some HDMI source devices detect an edge change on the pin and immediately react; some devices poll the pin level at a set time interval. For certain devices, this polling interval can be up to multiple seconds. Ensure the delay between steps 1 and 3 (above) is sufficient for the target source device to recognize the HPD level change.

HDMI Mode Configuration

HDMI serializers receive video input from an HDMI source via the TMDS (transition-minimized differential signaling) interface.

In HDMI 1.4 mode, the three sets of data channels operate at 10x of the pixel clock frequency (up to 3.4Gbps). The TMDS clock signal pair has the same frequency as the pixel clock. There is no phase requirement between the clock signals and the data channels. In HDMI 2.0 mode, the frequency of the TMDS clock signal pair is divided by four. The data channels' rate is 40x the TMDS clock rate. Data scrambling is required to reduce EMI/RFI.

- HDMI 1.4 mode is used for PCLK values under 340MHz.
- HDMI 2.0 mode is used for PCLK values between 340MHz and 600MHz.

HDMI 1.4 Configuration

This mode MUST be used if the source PCLK is 340MHz or below. Enable HDMI 1.4 mode using the following register writes:

```
# Script to set up HDMI 1.4 RX EQ Optimizations

0x80,0x3D0A,0x1F

0x80,0x3E5C,0x0D

# CTS EQ Lookup table - PEQ_VAL0-7

0x80,0x3E08,0x18

0x80,0x3E09,0x38

0x80,0x3E0A,0x58

0x80,0x3E0B,0x78

0x80,0x3E0C,0x98

0x80,0x3E0D,0xB8

0x80,0x3E0E,0xD8

0x80,0x3E0F,0x78

# cdr bypass

0x80,0x3D0E,0x82

# bp_fix

0x80,0x3E15,0x40

# auto mode and auto engine decision enable

0x80,0x3E00,0x20
```

HDMI 2.0 Configuration

HDMI source devices utilize HDMI 2.0 to transmit videos with a PCLK value above 340MHz. HDMI serializers MUST be configured to HDMI 2.0 mode to receive this input. Enable HDMI 2.0 mode using the following register writes:

```
# Enable HDMI 2.0 Mode

# Hold HDMI receiver digital logic in reset

0x80,0x2005,0x01

# Enable all 3 TMDS data channels

0x80,0x3D0C,0x20

0x80,0x3E8D,0x40

# HDMI2 overwrite, scdt on

0x80,0x3E50,0x23

# 5MHz OSC for Zone detect reset

0x80,0x3C04,0x04

# Release reset of 5MHz OSC

0x80,0x3C04,0x00

0x80,0x230E,0x24

0x80,0x230F,0x00

0x80,0x233B,0x80

0x80,0x2301,0x00

# Scramble-on overwrite, HDMI2-on overwrite

0x80,0x2040,0x33

0x80,0x2313,0x20

# Release HDMI receiver digital reset

0x80,0x2005,0x00
```

HDMI Pixel Repetition Mode

HDMI video sources use a pixel repetition (replication) scheme when the selected video format produces a pixel clock slower than 25MHz. This scheme duplicates pixels and increases the pixel clock frequency. GMSL2 HDMI serializers can be programmed to remove the replicated pixels in multiples of two, four, or eight.

Pixel Repetition Removal

GMSL2 HDMI serializers have a pixel repetition mode that can return the received video to the original video format and rate if the HDMI source device uses the pixel repetition scheme. Refer to *Table 8* for pixel repetition mode details.

Table 8. Pixel Repetition Removal

Register Address	Bitfield	Description	Decode
0x2A15	Pixel_Rate[1:0]	Remove replicated pixels	0x0: Pixel Replication 1x 0x1: Pixel Replication 2x 0x2: Pixel Replication 4x 0x3: Pixel Replication 8x

I²C Control Channel

Overview

The primary I²C control channel is used to provide access to both serializer and deserializer registers across the GMSL link. This provides flexibility where the registers for both serializer and deserializer are accessible from whichever side the main microcontroller resides. For camera applications, the main microcontroller typically resides on the deserializer side, while for display applications, the main microcontroller typically resides on the serializer side.

The pass-through I²C channels are used to send I²C data across the GMSL link. The pass-through channels can access the remote-side devices connected through the corresponding pass-through ports but cannot access the serializer or deserializer registers.

When making changes to any of the serializer or deserializer I²C configuration, such as enabling or disabling an I²C channel, a 10µs delay from the write acknowledgement (ACK) to the next transaction is required.

Port Access and Routing

The MFPs shown in the table below are used for the I2C primary and pass-through channels.

Table 9. MAX96751 Pins for I2C

Pin	Default Function	I2C Function #1	Notes
28	GPIO00	SDA1	Enable I2C pass-through via register
29	GPIO01	SCL1	Enable I2C pass-through via register
30	GPIO02	SDA2	Enable I2C pass-through via register
32	GPIO04	SCL2	Enable I2C pass-through via register
10	SDA_RX	N/A	I2C or UART function selected via CFG0 pin
11	SCL_TX	N/A	I2C or UART function selected via CFG0 pin
1	I2CSEL	I2CSEL	Selects I2C or UART on power-up. Latch low to select I2C.

On power-up, the device should be set to I2C mode via the CFG0 latch. The function names in the above MFP table and ensuing I2C sections assume the device has been configured for I2C mode.

By default, the primary I2C control channel lines are brought out on MFP11 and MFP12 for SDA and SCL, respectively. One can disable the primary control channel's line access by setting field DIS_LOCAL_CC in register 0x1. One can also disable access to remote device control by setting field DIS_REM_CC in register 0x1.

I²C Registers

Below table has registers that are needed to enable/disable primary and pass-through I²C channels.

Enabling I^{2}C Pass-Through Channels

When enabling a I^{2}C pass-through channel, other MFP functions must be disabled first.

With MAX96751 the user can bring out the first pass-through I2C channel (SDA1_RX1/SCL1_TX1) on pin 28/pin 29. The second pass-through I2C channel (SDA2_RX2/SCL2_TX2) can also be programmed on pin 30/ pin 32. Pass-through channels are enabled by setting the fields IIC_1_EN and IIC_2_EN in register 0x1.

Control Channel Programming Example

Below example shows the register writes needed to enable I^{2}C pass-through channel 1 on MAX96751 and MAX96752.

#DES I2C Address=0x90

#SER I2C Address=0x80

#Enable I2C Pass-through channel 1 on DES

0x90,0x0001,0x12

#Enable I2C Pass-through channel 1 on SER

0x80,0x0001,0x58

UART Control Channel

Overview

The primary UART control channel is used to provide access to both serializer and deserializer registers across the GMSL link. This provides flexibility where the registers for both serializer and deserializer are accessible from whichever side the main microcontroller resides (for camera applications, the main microcontroller typically resides on the deserializer side).

The pass-through UART channels are used to send UART data across the GMSL Link. The pass-through channels can access the remote-side devices connected through the corresponding pass-through ports but cannot access the serializer or deserializer registers.

Note: When making changes to any of the serializer or deserializer UART configuration, such as enabling or disabling an UART channels, a 10µs delay from the write acknowledgement (ACK) to the next transaction is required.

Base Mode

Base mode allows the device registers of both the serializer and deserializer to be accessed by the host microcontroller. It is the default mode for the primary UART control channel on power-up.

Bypass Mode

In Bypass mode, both the serializer and deserializer ignore all UART commands from the microcontroller. The serializer/deserializer registers are not accessible, and the microcontroller can freely communicate with any peripherals using its defined UART protocol. In this mode, the UART commands are still sent over the GMSL link. This mode prevents inadvertent programming of the serializer/deserializer registers and can be switched in and out of during normal operation.

Port Access and Routing

The MFPs shown in the table below are used for the UART primary and pass-through channels.

Table 10. MAX96751 MFP pins for UART

Pin	Default Function	UART Function #1	Notes
28	GPIO00	RX1	Enable UART pass-through via register
29	GPIO01	TX1	Enable UART pass-through via register
30	GPIO02	RX2	Enable UART pass-through via register
32	GPIO04	TX2	Enable UART pass-through via register
10	SDA _RX	RX	I2C or UART function selected via I2CSEL pin
11	SCL_TX	TX	I2C or UART function selected via I2CSEL pin
1	I2CSEL	I2CSEL	Selects I2C or UART on power-up. Latch low to select UART.

On power-up, the device should be set to UART mode via the I2CSEL pin state. The function names in the above MFP table and ensuing UART sections assume the device has been configured for UART mode.

By default, the primary UART control channel lines are brought out on MFP11 and MFP12 for RX and TX, respectively. The user can disable the primary control channel's line access by setting field DIS_LOCAL_CC in register 0x01. One can also disable access to remote device control by setting field DIS_REM_CC in register 0x01.

UART Registers

Table 11. MAX96751 UART Registers

Register Address	Bits	POR	Decode
0x02	5	0b0	DIS_LOCAL_CC, Bit [5]: Disable primary UART Control Channel connection to RX and TX pins
0x02	4	0b0	DIS_REM_CC, Bit [4]: Disable access to remote device control-channel over GMSL link
0x03	2	0b0	UART_1_EN, Bit [2] Enable UART pass-through channel 1
0x03	3	0b0	UART_2_EN, Bit [3] Enable UART pass-through channel 2
0x03	4	0b0	I2CSEL, Bit [4] select between UART and I^{2}C for primary Control Channel. Bootstrapped with the I2CSEL pin

Serial Peripheral Interface

Overview

SPI interface is available on the GMSL devices. Unlike I^{2}C and UART, SPI cannot be used to modify any registers in either serializer or deserializer; it is only used to transfer SPI data across the GMSL link. Typical SPI use cases are to send commands for other devices or to stream data other than video data (e.g. for sensors). GMSL devices support SPI transmission rate up to 25MHz.

The figure below shows the GMSL SPI architecture. On each side of the link, the GMSL devices become part of a main-subordinate pair and have transmit and receive buffers inside. On the local side, an internal SPI slave receives data from an external SPI main or micro-controller and transmits it across the serial link. On the remote side, the device receives the data from GMSL link and uses an internal SPI master to transmit the data to the external SPI peripheral/subordinate devices.

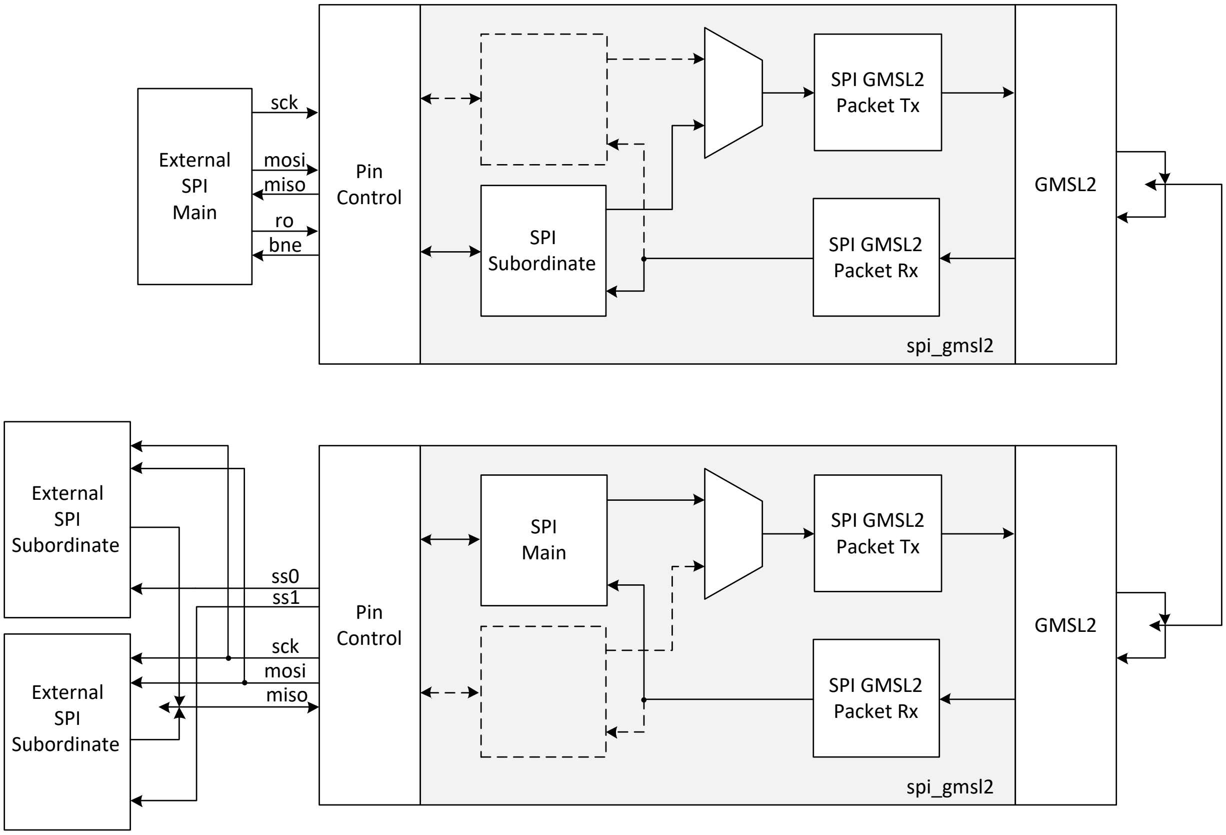


Figure 9. GMSL2 SPI Architecture

MFP/CFG Pin Setup for SPI

- Refer to the latest device data sheets for SPI MFP pins. Some MFP pins may have default alternate functions that must be disabled before enabling SPI. MFP status tool in the GMSL GUI can be used to verify the MFP functions that are enabled/disabled. If any of the SPI pins are also used as CFG pins, do not let any external SPI devices pull the CFG pins up or down until the GMSL devices power up and the CFG pins are latched. Power on the GMSL parts with the external SPI main device not connected or not pulling on the CFG pins; otherwise, the GMSL part will boot-up into an unwanted configuration.
- RO (Read Only) is an input bit that determines if the SPI subordinate is in Read or Write mode.
- BNE (Buffer Not Empty) is an output bit that shows the receive FIFO state. BNE is low when the buffer is empty. BNE is high when there is data in the buffer. This bit is used to determine the status of the buffer for data transfers and avoiding buffer overflow.

SPI Initialization

Configure the serializer and deserializer in the following order to initialize SPI (starting from the default values) on GMSL devices:

1. Configure SPI mode 0 or 3 on the serializer and deserializer.
2. Set SS output polarity (remote side).

3. Set the clock delay and SCLK rate high/low times (in number of 300MHz clocks).
4. Program the IO pin enables (BNE/RO/SS1/SS2).
5. Configure internal GMSL Main/Subordinate mode, SPI ID (if needed), and enable SPI.

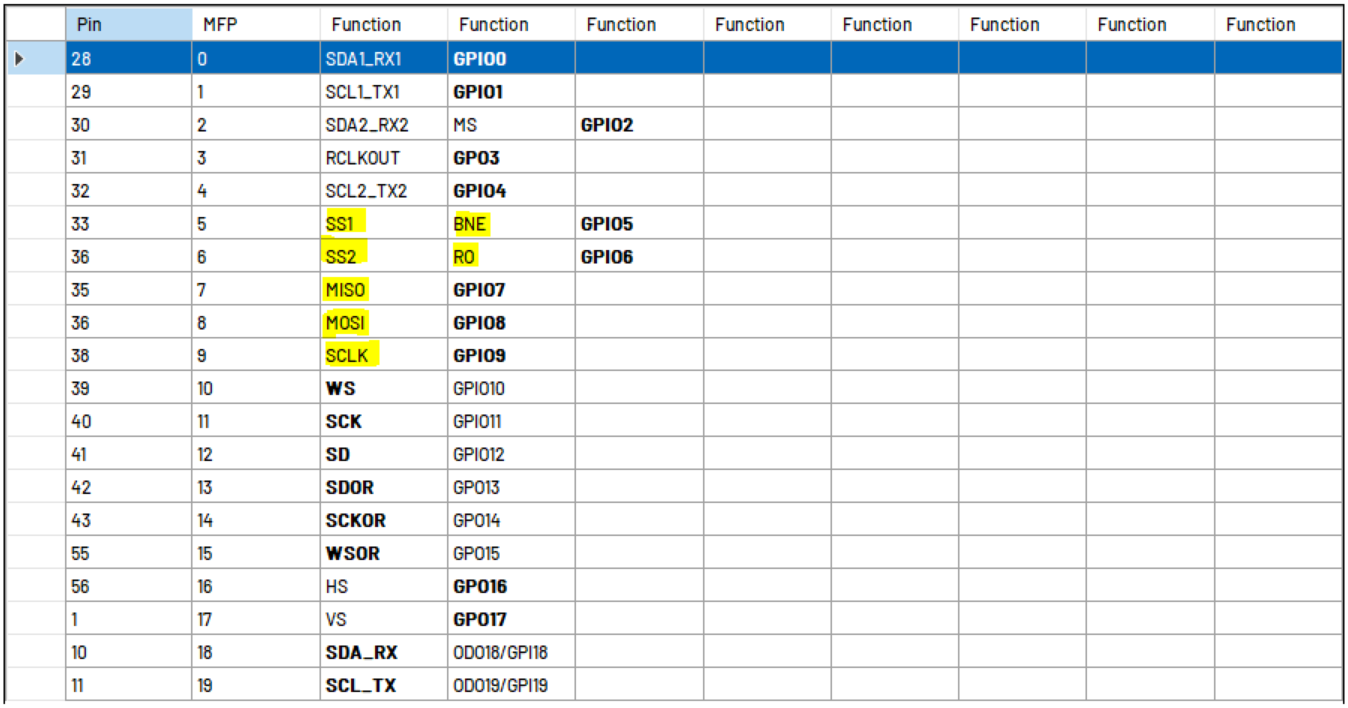
SPI Example with Register Writes

Below is the SPI setup example script for MAX96751 with its accompanying device MAX96752 (0x80 is the serializer address, 0x90 is the deserializer address), assuming micro-controller or external SPI main is on the serializer side*.

```
0x80,0x160,0x09  #enable SPI, default set to subordinate, and ignore the SPI header ID.

0x80,0x161,0x1D #default, sets SPI packet size and GMSL link scheduler priority.

0x80,0x162,0x00  #subordinate select (SS) is active low, SPI mode is 0.

0x80,0x163,0x00  #default, delay between assertion of subordinate select (SS) and SPI clock (SCLK) start.

0x80,0x166,0x03  #enable RO and BNE.

0x80,0x168,0x00  #default, timeout delay.

0x90,0x160,0x0B  #Enable SPI channel, set as main.

0x90,0x161,0x1D #default, sets SPI packet size and GMSL link scheduler priority.

0x90,0x162,0x00  #default, subordinate select is active low, SPI mode is 0.

0x90,0x163,0x00 #default, delay between assertion of subordinate select and clock start.

0x90,0x164,0x1E #SPI clock low time.

0x90,0x165,0x1E #SPI clock high time.

0x90,0x166,0x0C  #Enable subordinate select 1 (SS1) and 2 (SS2), RO and BNE not enabled.

0x90,0x168,0x00  #SPI timeout delay.
```

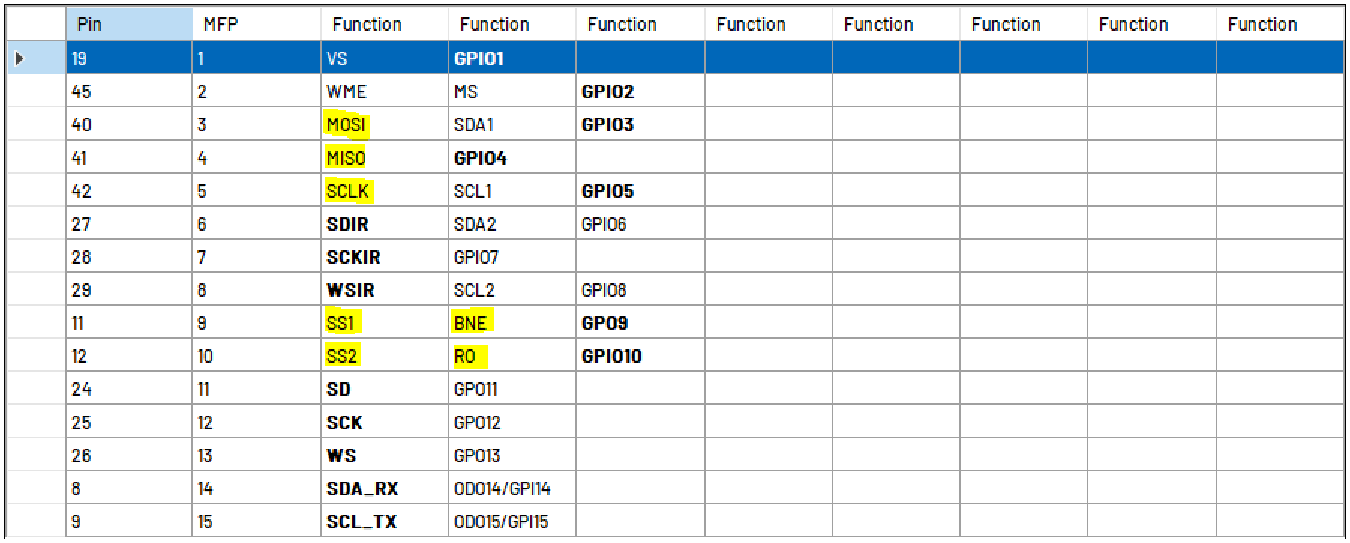
*Note: If micro-controller or external SPI main is on the deserializer side, swap the Ser and Des register writes in the script.

Figure 10 and *Figure 11* show MFP pins being used for SPI after running the above script.



	Pin	MFP	Function	Function	Function	Function	Function	Function	Function	Function
▶	28	0	SDA1_RX1	**GPIO0**						
	29	1	SCL1_TX1	**GPIO1**						
	30	2	SDA2_RX2	MS	**GPIO2**					
	31	3	RCLKOUT	**GPO3**						
	32	4	SCL2_TX2	**GPIO4**						
	33	5	SS1	BNE	**GPIO5**					
	36	6	SS2	RO	**GPIO6**					
	35	7	MISO	**GPIO7**						
	36	8	MOSI	**GPIO8**						
	38	9	SCLK	**GPIO9**						
	39	10	**WS**	GPIO10						
	40	11	**SCK**	GPIO11						
	41	12	**SD**	GPIO12						
	42	13	**SDOR**	GPO13						
	43	14	**SCKOR**	GPO14						
	55	15	**WSOR**	GPO15						
	56	16	HS	**GPO16**						
	1	17	VS	**GPO17**						
	10	18	**SDA_RX**	OD018/GPI18						
	11	19	**SCL_TX**	OD019/GPI19						

Figure 10. SPI MFP Pin Settings for MAX95751



	Pin	MFP	Function	Function	Function	Function	Function	Function	Function	Function
▶	19	1	VS	**GPIO1**						
	45	2	WME	MS	**GPIO2**					
	40	3	MOSI	SDA1	**GPIO3**					
	41	4	MISO	**GPIO4**						
	42	5	SCLK	SCL1	**GPIO5**					
	27	6	**SDIR**	SDA2	GPIO6					
	28	7	**SCKIR**	GPIO7						
	29	8	**WSIR**	SCL2	GPIO8					
	11	9	SS1	BNE	**GPO9**					
	12	10	SS2	RO	**GPIO10**					
	24	11	**SD**	GPO11						
	25	12	**SCK**	GPO12						
	26	13	**WS**	GPO13						
	8	14	**SDA_RX**	OD014/GPI14						
	9	15	**SCL_TX**	OD015/GPI15						

Figure 11. SPI MFP Pin Settings for MAX96752

SPI Example using GMSL GUI and Evaluation Boards

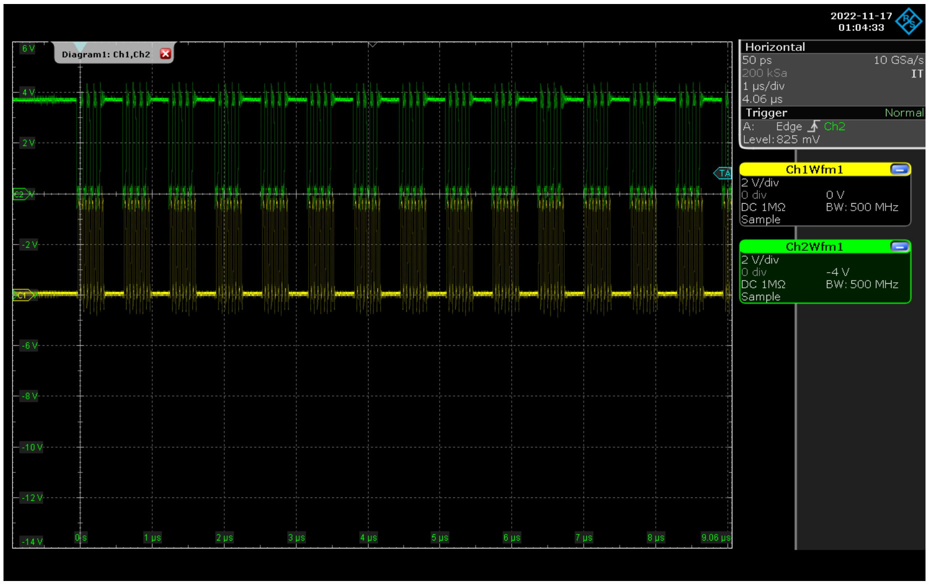
- It is recommended to set the SCLK output rate equal or more than the SCLK input rate to avoid buffer overflow in the SER or DES. The SCLK rate can be set using register writes.
- Disconnect external SPI main device or do not pull CFG pins on SER and DES.

Power up EV boards (ensure that V_{DDIO} on the external SPI main device matches the V_{DDIO} on GMSL SER and DES).

- Start GMSL GUI.
- Load GMSL script to enable SPI interface.
- Reconnect external SPI main device.
- Set RO (Read Only) high and write 0xA0, 0xA4(0xA0-0xA3 are used for SPI ID selection, 0xA4 asserts SS1, 0xA5 asserts SS2, and 0xA6 de-asserts both SS1 and SS2).
- As a best practice, before starting SPI data transfer, check BNE (Buffer Not Empty) to ensure that the buffer is empty. If BNE is high, there is data in the RX buffer that is ready to be read by the external SPI main device. Set RO high and write FF until BNE = 0.
- Set RO low and start the SPI data transfer.

SPI With and Without Video Running

The SPI Tx FIFO is 16 bytes and Rx FIFO is 32 bytes. The below screenshots show the SPI oscilloscope probes on the SPI clock and data output (the receiving end at the deserializer). When there is no video through the GMSL link, the SPI data is transferred consistently without any delay; however, when the GMSL link is utilized 90% by video data, there may be some intermittent pauses during the SPI data transmission. This is because video data has higher priority compared to SPI data transfer. The 32-byte buffer compensates for this scheduling delay and makes continuous streaming of the data possible.



Horizontal
50 ps 10 GSa/s
200 kSa IT
1 µs/div
4.06 µs
Trigger Normal
A: Edge Ch2
Level: 825 mV

Ch1Wfm1
2 V/div
0 div 0 V
DC 1MΩ BW: 500 MHz
Sample

Ch2Wfm1
2 V/div
0 div -4 V
DC 1MΩ BW: 500 MHz
Sample

Figure 12. SPI Clock and Data at Final Output (at External SPI Subordinate), No Video on GMSL Link

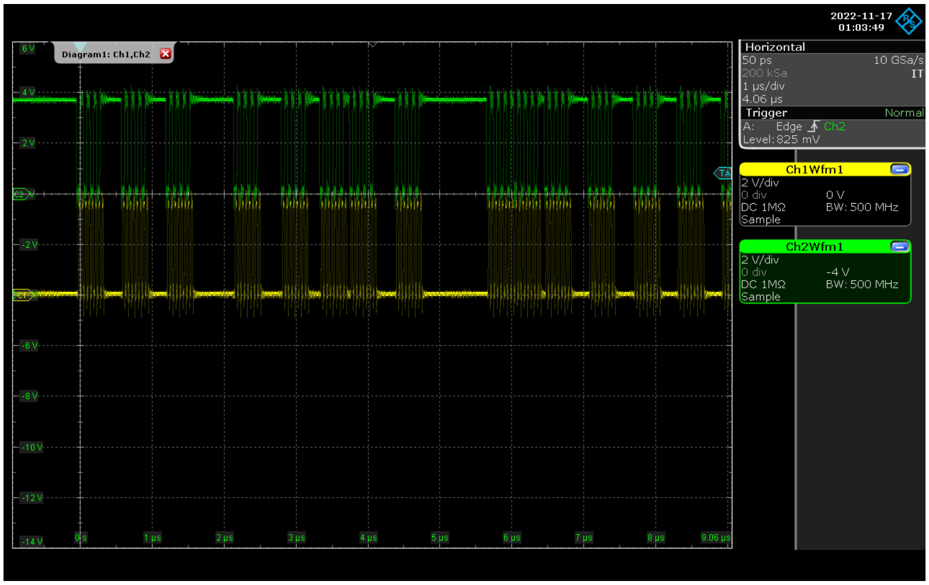


Figure 13. SPI Clock and Data at Final Output (at External SPI Subordinate), 92% Video on GMSL Link

Data Integrity and Avoiding Buffer Overflow

In general, SPI will stream continuously and without having the SPI external main/generator read back any of the values. However, the techniques in this section are additional steps that are recommended to be implemented in the system to ensure that all the data is sent correctly.

After a SPI data byte is sent across the GMSL link, the GMSL device on the remote side will send the data out on the MFP pins. It will also send the data back across the link so that the external SPI main device can read back the data.

State of RO (Read Only) pin dictates direction of data movement.

• RO = 0: Data transmitted between main and subordinate via MOSI

• RO = 1: Data transmitted between subordinate and main via MISO, BNE will be high if there are bytes in this buffer to be read back.

Note that the word “read” in the name of the RO pin does not mean that it is an output pin; it is, in fact, an input pin that is toggled externally high or low depending on read or write operation.

It is recommended to limit the amount of “bytes in transit” (bytes that have been sent but not received back) to 16. The external SPI main device can compute this value (= valid bytes sent – valid bytes read).

One way of doing this is to send the data in a group of 16 bytes or less. If more than 16 bytes are sent at a time, it is still possible (depending on timing) that all the data will be sent properly, but it will not be possible to easily be sure that the data was sent properly.

Below is an example of transferring 4 bytes of SPI data across the GMSL link:

- RO is pulled high, and the A0 and A4 control commands are sent.
- RO is pulled low, and 4 bytes of SPI data (0x80, 0x04, 0x01, 0x47) are sent from the external SPI main device to the GMSL input side (serializer).

- The last three signals in the graph show SPI data output from the remote side of the link (deserializer), there may be some delay between the input and output of the SPI data across the GMSL link.
- RO is high, and the bytes are read back by the external SPI main device (BNE is high, which indicates that SPI data bytes from the external SPI subordinate are available to be read.).

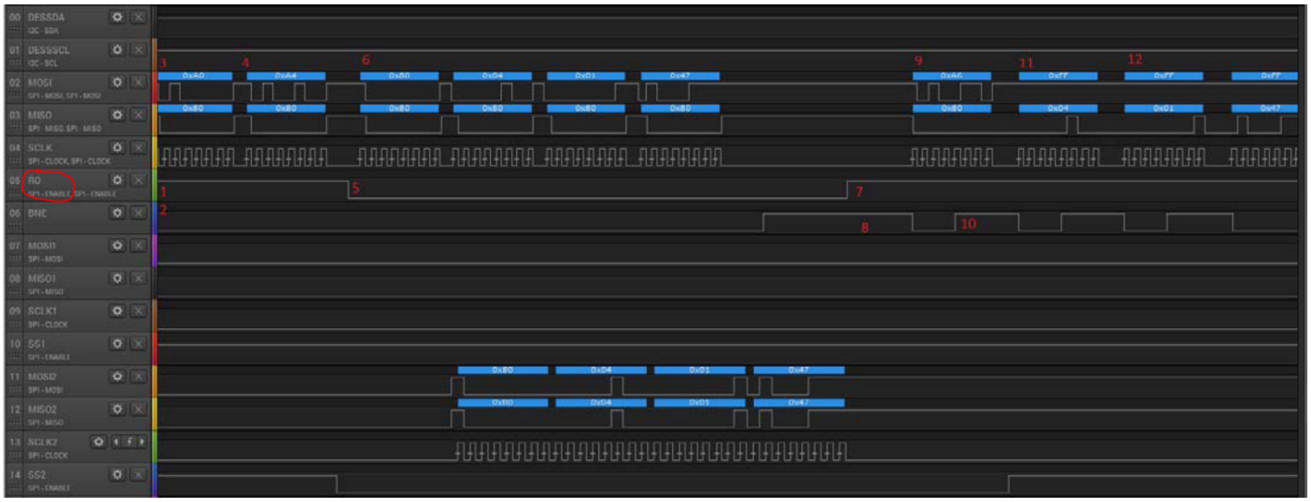


Figure 14. SPI Transmission Example

Each Rx and Tx SPI buffer has overflow detection logic with status bits that can be monitored by registers SPI_TX_OVRFLW and SPI_RX_OVRFLW.

SPI Setup Registers

The table below shows some of the important setup registers for enabling SPI interface. Register block in the data sheet will have additional details for configuring the SPI main, subordinate, SCLK timings, etc.). See the programming script in the section below as an example.

Table 12. MAX96751 SPI Register Settings

Register	Bitfield Name	Bits	Default Value	Description
0x160	SPI_EN	0	0	0 = SPI not enabled 1 = SPI enabled
0x160	MST_SLVN	1	0	0 = SPI subordinate 1 = SPI main
0x160	SPI_IGNR_ID	2	1	0 = Accept packets with proper ID 1 = Ignore ID and accept all packets (recommended)
0x162	SPIM_SS1_ACT_H	0	1	0 = SS1 is active low 1 = SS1 is active high
0x162	SPIM_SS2_ACT_H	1	1	0 = SS2 is active low 1 = SS2 is active high
0x166	RWN_IO_EN	0	0	0 = Do not bring RO out to MFP pin 1 = Bring out RO to MFP pin
0x166	BNE_IO_EN	1	0	0 = Do not bring BNE out to MFP pin 1 = Bring out BNE to MFP pin
0x166	BNE	5	0	0 = No bytes to read 1 = Bytes ready to read

0x167	SPI_TX_OVRFLW	6	0	0 = No overflow 1 = Overflow
0x167	SPI_RX_OVRFLW	7	0	0 = No overflow 1 = Overflow

General-Purpose Input and Output (GPIO)

Overview

This section explains the GPIO function of MFP pins. The GPIO blocks communicate and regenerate state changes of GPIO pins from one side of the serial link to the other. An input GPIO value on one side of the GMSL link may be sent to any of the GPIO outputs on the opposite side of the link.

Depending on the pin, they can be used as either full or partial general-purpose input and output (GPIO) pins or for other functionality (e.g., I²C, LOCK, ERRB, etc.). Refer to the data sheets for additional details on GPIO capabilities and default states after power-up.

The MAX96751 family has 19 multifunction pins (MFPs), refer to the data sheet for full details.

Operation

GPIO pin mapping is coordinated across the serial link through GPIO “pin ID” assignments. Each GPIO input is assigned a pin ID that is included in the packet sent across the serial link and corresponds with a GPIO output. By default, the GPIO mapping is GPIO0 to GPIO0, GPIO1 to GPIO1, GPIO2 to GPIO2, etc. The GPIO mappings can be changed through registers.

GMSL devices use 5-bit pin IDs that can support mapping up to 32 GPIO pins. Note that the usable number of GPIOs is limited by the specific GPIO pinout. Each GPIO is controlled by three registers: GPIO_A, GPIO_B, and GPIO_C. In the register documentation, the GPIO mapping is sequential (i.e., the first three GPIO registers correspond to GPIO0, then the next three to GPIO1, etc.). Additional details related to these registers can be found in the “GPIO Registers” section of the respective data sheet.

When programming GPIOs, it is important to program the GPIO Rx before the GPIO Tx to avoid asynchronous initial states. For example, if Tx is low but Rx is high, the first transition of Tx from low to high is ignored by Rx as Rx is already high. All subsequent transitions are correctly observed.

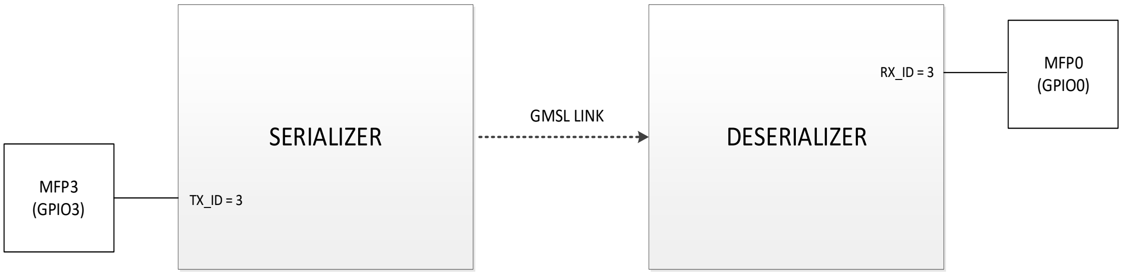


Figure 15. GPIO Forwarding Example with a Transition from MFP3 to MFP0

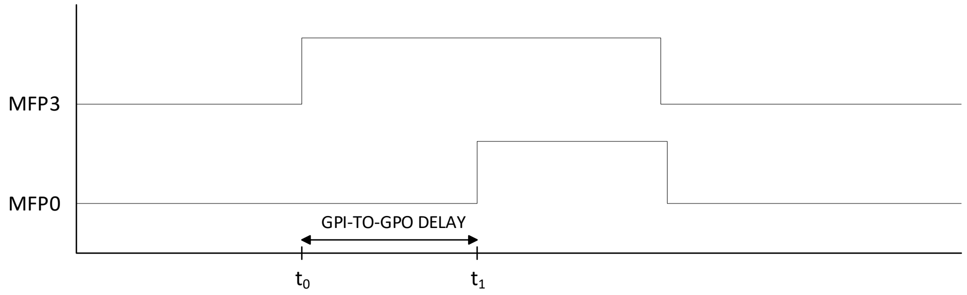


Figure 16. GPIO Forwarding Timing Diagram

GPIO, GPI, GPO and ODO

The MAX96751 has 20 GPIOs; 15 of these MFPs are GPIO (general-purpose input or output) and 5 are GPO (general-purpose-output). *Table 13* shows the GPIO capabilities of each GPIO pin.

Table 13. MAX96751 GPIO Capabilities

Pin	Capability
28	GPIO00
29	GPIO01
30	GPIO02
31	GPIO03
32	GPIO04
33	GPIO05
34	GPIO06
35	GPIO07
36	GPIO08
38	GPIO09
39	GPIO10
40	GPIO11
41	GPIO12
42	GPO13
43	GPO14
55	GPO15
56	GPO16
1	GPO17
10	GPIO18
11	GPIO19

GPIO Pull-up and Pull-Down Resistor Setup

Each GPIO can be programmed to have either a pull-up, pull-down, or no resistor. The pull-up or pull-down resistance can be set to either 40kΩ or 1MΩ.

The resistor is configured with the PULL_UPDN_SEL[1:0] register:

- 00: No resistor
- 01: Pull-up resistor

- 10: Pull-down resistor
- 11: Reserved

The resistance value of the resistor is set using the RES_CFG register:

- 0: 40 kΩ
- 1: 1 MΩ

GPIO Output Driver Setup

The GPIO output driver can be enabled or disabled. When enabled, the output driver can be configured to be either open-drain or push-pull. The output driver is enabled by writing GPIO_OUT_DIS = 0 and disabled by writing GPIO_OUT_DIS = 1. The output driver is configured for open-drain mode (i.e., NMOS output driver enabled) by writing OUT_TYPE = 0 and for push-pull mode (i.e., both NMOS and PMOS output driver enabled) by writing OUT_TYPE = 1.

Configuring GPIO Forwarding

GPIO forwarding is the transmission and regeneration of state changes of GPIO pins on the local side of the serial link to the corresponding GPIO pins on the remote side. To forward the pin value, the local and remote side GPIOs must be properly configured. Each GPIO has configurable registers GPIO_TX_ID and GPIO_RX_ID used for mapping GPIO pins across the serial link. Note that this configuration applies to both the serializer-to-deserializer and deserializer-to-serializer communications.

Configuring Input GPIO:

- Set GPIO_TX_ID with a value from 0 to 31 to assign the GPIO pin ID.
- Write GPIO_TX_EN = 1 to enable the GPIO transmit block.

Configuring Output GPIO:

- Set GPIO_RX_ID with a value from 0 to 31 to assign the GPIO pin ID. This must be the same value used for GPIO_TX_ID to *map* the input and output GPIO pins.
- Write GPIO_RX_EN = 1 to enable the GPIO receive block for the GPIO pin.

By default, the GPIO_TX_ID and GPIO_RX_ID are the same value as the GPIO number. For example, the default GPIO_TX_ID and GPIO_RX_ID values for GPIO1 is 1. Accordingly, GPIO1 is mapped to GPIO1 on the opposite side of the serial link by default.

GPIO Broadcasting

The same concept of GPIO forwarding can be configured so that a transition on a single GPIO input is mapped to multiple GPIO outputs (broadcasting). To do this, set the GPIO_TX_ID of the input GPIO to the same GPIO_RX_ID of multiple output GPIO pins. *Figure 17* is an example of this configuration.

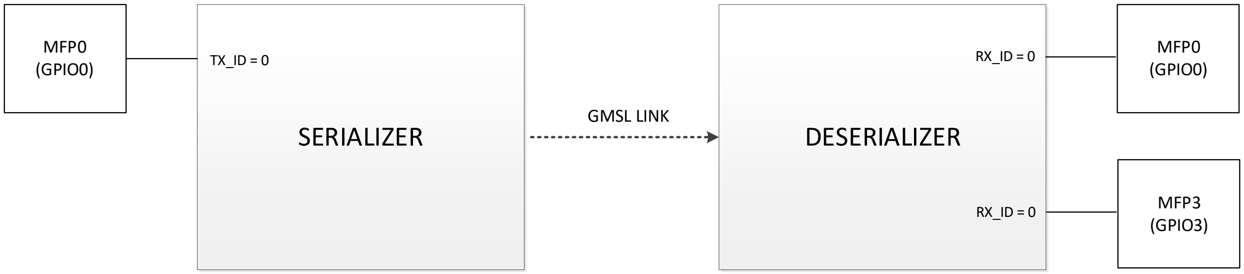


Figure 17. GPIO Broadcasting

GPIO Delay Compensation

In the non-delay-compensated mode (default), the GPI transition is sent as fast as possible across the link, based on priority and available link bandwidth. As a result, there is a variable delay between an input transition and the subsequent transition on the other side of the GMSL2 link. Delay compensation can be used to ensure that the timing delay between input transition and output transition is constant. *Table 14* shows the typical values and below registers show how to set delay compensation.

Table 14. GPIO (with/without) Delay Compensation Values

Direction	Delay Compensation	Delay
GPIO Forwarding from Serializer to Deserializer	0	1us
	1	3.5us (default)
GPIO Forwarding from Deserializer to Serializer	0	6us
	1	15us (default)

Toggling GPIO Manually with Registers

GPIO pins can be manually controlled through I^{2}C or UART register writes. Write to the local device to toggle local GPIO pins; write to the remote device using the control channel to toggle remote GPIO pins.

- Set GPIO_OUT_DIS = 0 to enable the output driver and configure OUT_TYPE to the desired output mode (open-drain or push-pull).
- Set GPIO_RX_EN = 0 to disable the GPIO receive block for the GPIO pin. This sets the GPIO to receive its value from the bitfield GPIO_OUT instead of from the value being transmitted across the GMSL2 link.
- Set GPIO_OUT to the desired value.

Table 15. MAX96751 GPIO Registers

Register	Bits	Default Value	Decode
0x200	7:0	0x83	**GPIO0 A:** Bit 7: RES_CFG 0=40kΩ 1=1MΩ Bit 6: RSVD

Register	Bits	Default Value	Decode
			Bit 5: TX_COMP_EN 0=Jitter compensation disabled 1=Jitter compensation enabled Bit 4: GPIO_OUT 0=Drive output to LOW (0) 1=Drive output to HIGH (1) Bit 3: GPIO_IN 0=GPIO input level LOW (0) 0=GPIO input level HIGH (1) Bit 2: GPIO_RX_EN 0=Disable receiving from the link. 1=Enable receiving from the link Bit 1: GPIO_TX_EN 0=Disable transmitting to the link. 1=Enable transmitting to the link Bit 0: GPIO_OUT_DIS 0=Output driver enabled 1=Output driver disabled
0x201	7:0	0xA0	**GPIO0 B:** Bit [7:6]: Resistor Configuration 00=None 01=Pull-up 10=Pull-down Bit 5: OUT_TYPE 0 = Open-drain 1 = Push-pull Bit [4:0]: GPIO_TX_ID Address=0-31
0x202	7:0	0x40	**GPIO0 C:** Bit 7: OVR_RES_CFG[0] 0=Received GPIO Value 0 1=Received GPIO Value 1 Bit [4:0]: GPIO_RX_ID Address=0 to 31
0x203 to 0x23B	…	…	Repeat Registers A, B, C for GPIO1 to GPIO19.

GPIO Programming Example

In this example, GPIO0 on a serializer is forwarded across the link to GPIO1 on a deserializer. This example can be adjusted to use different GPIO pins or forward a GPIO on the local side to the remote side, depending on the desired application. An important note is to set up the GPIO Rx side before setting up the GPIO Tx side to prevent asynchronous states.

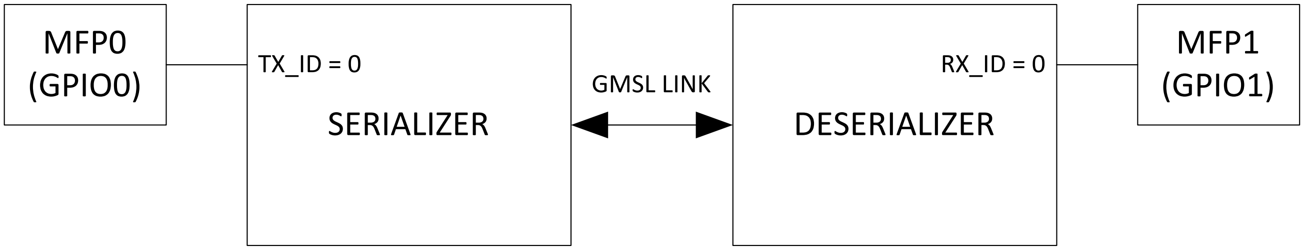


Figure 18. GPIO Forwarding Programming Example

```
#DES I2C Address=0x90
#SER I2C Address=0x80
#Setup SER MFP0 Pin
#1M ohm Pull-Up, Open Drain
0x80,0x0200,0x83
0x80,0x0201,0x40
0x80,0x0202,0x40
#Setup DES MFP0 Pin
#1M ohm Pull-Up, Open Drain
0x90,0x0203,0x84
0x90,0x0204,0x40
0x90,0x0205,0x40
```

MFP Slew Rate

The MFPs have configurable rise and fall times (slew rate). This parameter may be referred to as the I/O “speed (control),” “slew (rate),” or “edge rate” in register control bit names. Note that the MFP slew rate cannot be adjusted independently on a per-pin basis. MFPs are divided into separate speed groups; the slew rate adjustment register contains a bitfield for each group that configures the rise and fall time to all pins in the group. Refer to the data sheet for the relevant register map and MFP speed grouping details.

The MFP edge transitions must be fast enough to meet the application’s requirement; however, the high-speed I/O of the GMSL link and video protocols (e.g., MIPI) are sensitive to coupling and crosstalk from MFP transitions. Take care at a system level to prevent high edge rates and high frequencies on the MFP inputs close to these I/O. In general, the MFP pins should be configured to the slowest slew rate that allows proper function to mitigate I/O interference.

Note: Coupling refers to both inductive and capacitive coupling. Higher V_{DDIO} supply values increase the MFP edge rate and energy. This can introduce additional noise into the high-speed I/O.

High MFP slew rates, especially combined with high toggle frequencies, near the GMSL or high-speed video pins may adversely affect performance of the data path, including CRC errors, 9b10b code or disparity errors, reduction of link margin, and/or loss of link lock.

Configuration

MFPs are divided into speed groups by digital function. The slew rate adjustment register configures the rise and fall times for each MFP in the group simultaneously. The MFP slew rate can be adjusted at any time, and the changes are applied immediately.

The MFP slew rate configuration applies to all pins in the speed group regardless of the enabled function of the pin. For example, the speed setting is applied to a GPIO and a dedicated pin function if both are in the same MFP speed group.

Refer to the corresponding device's data sheet "Control- and Side-Channel Typical Rise and Fall Times" section for V_{DDIO} timing details. The table below represents slew rate registers and the typical rise and fall times.

Table 16. MAX96751 MFP Slew Rate Registers

Register	Bitfield Name	Bits	Default Value	Description
0x245	GPIO_SPEED_A[1:0]	1:0	0x10	Controls GPIO Speed Group A (GPIO 14-17, LOCK, ERRB, HSPD) transition time. First value is for V_{DDIO} = 1.8V, second value is for V_{DDIO} = 3.3V 0 2ns, 1ns 1 4ns, 2ns 2 8ns, 4ns 3 16ns, 8ns
	GPIO_SPEED_B[1:0]	3:2	0x11	Controls GPIO Speed Group B (GPIO 0-13, HPD) transition time. First value is for V_{DDIO} = 1.8V, second value is for V_{DDIO} = 3.3V 0 2ns, 1ns 1 4ns, 2ns 2 8ns, 4ns 3 16ns, 8ns

Audio

Overview

I²S or TDM is available on GMSL devices. Typical use case is to send audio data from one end to the other via the GMSL link. GMSL2 devices support bidirectional transmission of audio (i.e., I²S or TDM) over the forward and reverse channels of the serial link.

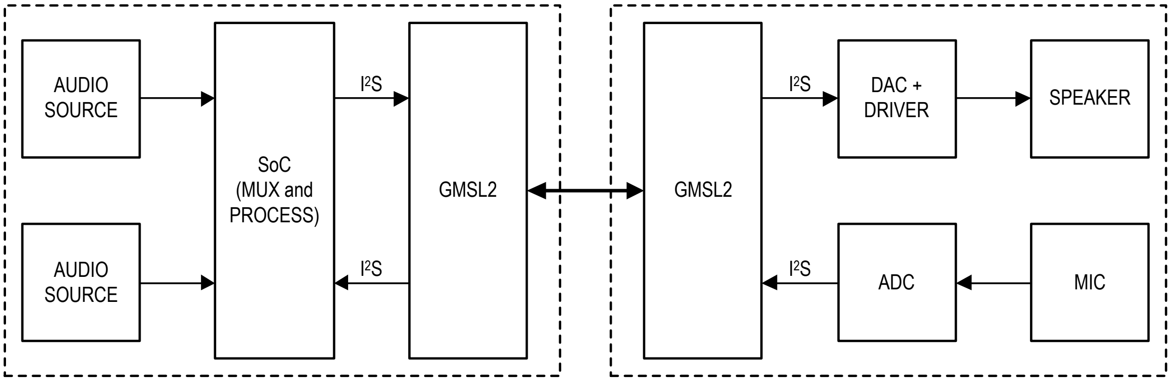


Figure 19. Audio Architecture Example

Pin Access and Routing

Table 17. MAX96751 Pins for Audio

Pin	Default Function	Audio Function #1	Notes
39	WS	Word Select	Word-Select Input with Internal 1MΩ Pull-Down to Ground. Supports forward audio from serializer to deserializer
40	SCK	Serial Clock	I2S/TDM Serial Clock Input with Internal 1MΩ Pull-Down to Ground. Supports forward audio from serializer to deserializer.
41	SD	Serial Data	I2S/TDM Serial-Data Input with an Internal 1MΩ Pull-Down to Ground. Supports forward audio from serializer to deserializer.
42	SDOR	Serial Data Output Reverse	I2S/TDM Serial Data Push-Pull Output. Supports reverse-channel audio from deserializer to serializer.
43	SCKOR	Serial Clock Output Reverse	I2S/TDM Clock Push-Pull Output. Supports reverse-channel audio from deserializer to serializer.
55	WSOR	Word Select Output Reverse	I2S/TDM Word Select Push-Pull Output. Supports reverse-channel audio from deserializer to serializer.

Audio Setup Registers

Table 18. MAX96751 Audio Register Settings

Register	Bitfield Name	Bits	Default Value	Description
0x02	*AUD_TX_EN_X*	2	1	0 = Audio Transmit Channel X Disabled 1 = Audio Transmit Channel X Enabled
0x02	*AUD_TX_EN_Y*	3	0	0 = Audio Transmit Channel Y Disabled 1 = Audio Transmit Channel Y Enabled
0x140	*AUD_EN_RX*	0	1	0 = Audio Receiver Disabled 1 = Audio Receiver Enabled
0x121	*AUD_STR_TX[1:0]*	4, 5	0b00	0b00 = Transmit Channel X Use Audio Stream ID 0 0b01 = Transmit Channel X Use Audio Stream ID 1 0b10 = Transmit Channel X Use Audio Stream ID 2 0b11 = Transmit Channel X Use Audio Stream ID 3
0x131	*AUD_STR_TX[1:0]*	4, 5	0b01	0b00 = Transmit Channel Y Use Audio Stream ID 0 0b01 = Transmit Channel Y Use Audio Stream ID 1

				0b10 = Transmit Channel Y Use Audio Stream ID 2 0b11 = Transmit Channel Y Use Audio Stream ID 3
0x146	*AUD_STRM[1:0]*	2,3	0b00	0b00 = Receive Channel receive Audio Stream ID 0 0b01 = Receive Channel receive Audio Stream ID 1 0b10 = Receive Channel receive Audio Stream ID 2 0b11 = Receive Channel receive Audio Stream ID 3

Power Manager and Sleep Mode

Overview

The power manager ensures the reliable and efficient operation of various power functions. The power manager controls the internal switched supply domains during the full sequence of power states so that the device powers up and down smoothly. During power-up, the power manager guards the device until the internal supplies have been validated and the digital core assumes normal operations. In all power modes, the power manager monitors power supplies for under- and over-voltage conditions. In Sleep mode, the power manager minimizes current consumption and can quickly restore device configurations after waking up.

Table 19. MAX96751 Power Manager and Sleep Mode Availability

Part Number	Power Manager	Sleep Mode
MAX96751	Supported	Supported

Device Power Operation

The power manager block minimizes required user interaction while providing extensive diagnostic indicators. Power manager status registers can be polled for valid supply levels, and a system-level interrupt (ERRB=0) can be generated in the case of a device power failure.

MAX96751 uses power rails (V_{DDD}, V_{DDA}, V_{DD33}, V_{DD18}, and V_{DDIO}).

Note: If the power manager sends an ERRB interrupt due to a power fail condition, check PWR0, PWR1, and other diagnostic registers to identify the source of the failure. Refer to the voltage monitoring section of the respective data sheet for additional details.

Power Supplies

GMSL devices share a common set of power supply voltages that power universal functions such as the digital core, GMSL link circuitry, and GPIO. These power supplies are summarized below.

- V_{DDD}: The input voltage to the V_{DDD} rail can be between 0.95V and 1.05V.
- V_{DD18}: 1.8 V power rail.
- V_{DDA}: The input voltage to the V_{DDA} rail can be between 0.95V and 1.05V.
- V_{DDIO}: 1.8 V or 3.3 V I/O power rail for I/O.
- V_{DD33}: The input voltage to the V_{DD33} rail can be between 3.14V and 3.47V.

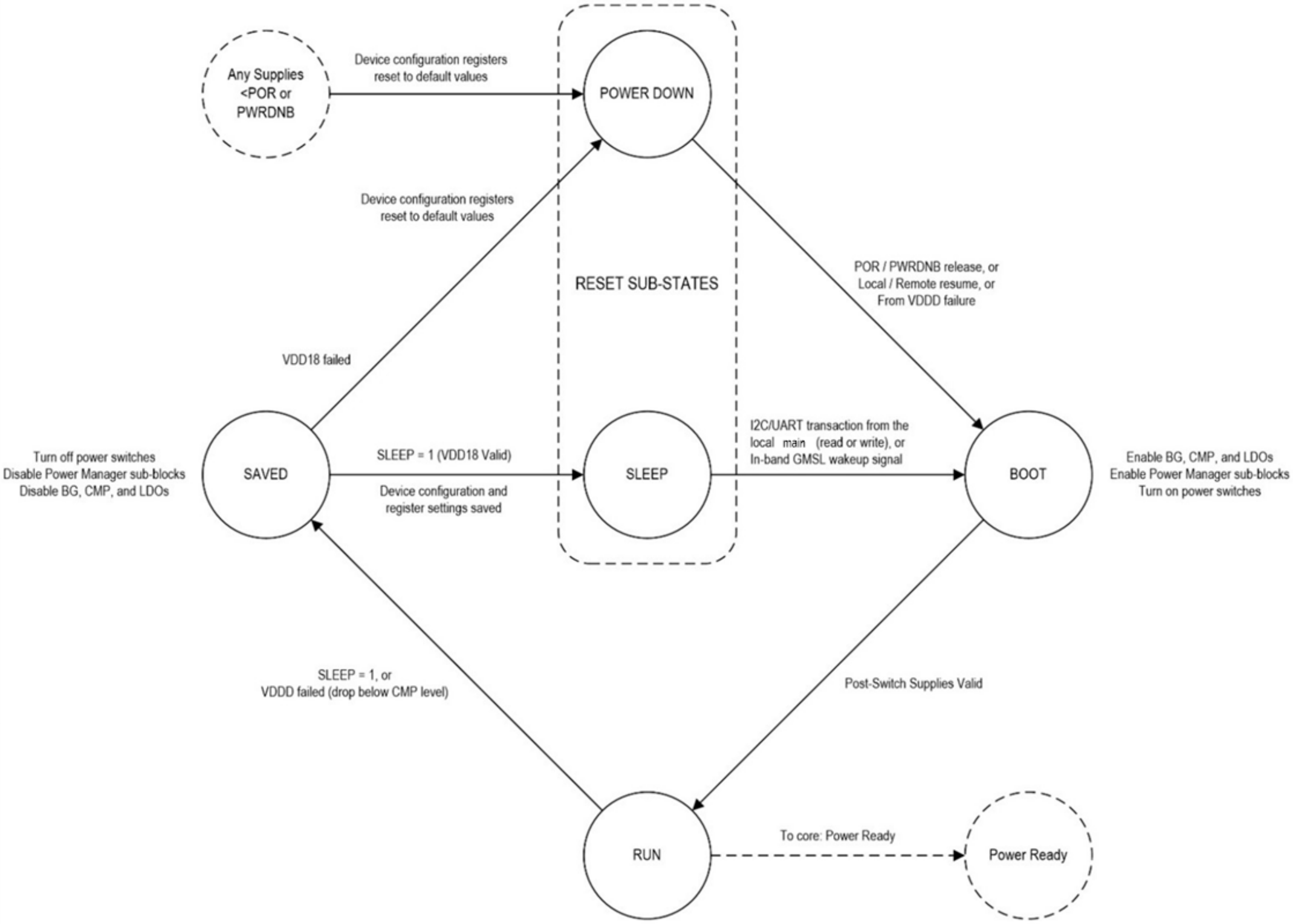
External power is supplied directly to V_{DDD}, V_{DD18}, V_{DDA}, V_{DDIO}, and V_{DD33}

Power Manager States

At device power-up, the power manager block automatically controls the power sequencing process. Power supplies can ramp in any order and do not need to be externally sequenced. When power is applied, the power manager senses the presence of each domain. When the voltage threshold is reached for all supplies, the power manager signals to the other device domains that power is stable and begins to transition into Run mode.

The power manager state machine has four power states: boot, run, saved, and reset (power down/sleep). The power manager circuitry is in the “always-on” V_{DD18} domain so that all power domains may be managed and monitored during the full sequence of power states. This architecture allows for a seamless resume from the Sleep to Run mode and draws minimal current. Retention memories are also powered by the V_{DD18} domain so that device configuration and register settings can be saved and restored.

Figure 20 shows the state diagram for the power manager.



Any Supplies <POR or PWRDNB
Device configuration registers reset to default values
POWER DOWN
Device configuration registers reset to default values
RESET SUB-STATES
POR / PWRDNB release, or Local / Remote resume, or From VDDD failure
VDD18 failed
Turn off power switches
Disable Power Manager sub-blocks
Disable BG, CMP, and LDOs
SAVED
SLEEP = 1 (VDD18 Valid)
Device configuration and register settings saved
SLEEP
I2C/UART transaction from the local main (read or write), or In-band GMSL wakeup signal
BOOT
Enable BG, CMP, and LDOs
Enable Power Manager sub-blocks
Turn on power switches
SLEEP = 1, or VDDD failed (drop below CMP level)
Post-Switch Supplies Valid
RUN
To core: Power Ready
Power Ready

Figure 20. Power Manager State Diagrams

Reset (Power Down/Sleep)

Power down and sleep are two sub-states of the reset state.

The device will enter the power down state if the PWDNB pin is asserted (low), VDD_sw falls below the internally set threshold, or if any other supply falls below the associated POR value. In power down, all registers in the digital core revert to default reset values. Power failure latches are retained unless V_{DD18} falls too low. De-asserting PWDNB (high) releases the chip from the power down state and into the boot state.

Sleep is a low-power consumption state that preserves the configurations and settings saved in the previous state and enables a much faster return to running operation than from power down. When the device is in the run state, the system (µC/SoC) can initiate sleep state with an I^2C/UART command (SLEEP = 1). Sleep mode is entered automatically after the retention memory is loaded following the SLEEP=1 command. In the sleep state, the V_{DD18} supply must be continuously maintained to ensure that previous configurations and settings are preserved. It is recommended that all other supplies be maintained during Sleep mode to simplify the sleep and wakeup sequences.

BOOT

The device can enter the boot state from reset after external supplies have ramped up or the device has resumed operation from sleep. In boot, all power switches are turned on, and all power manager sub-blocks are enabled. When all post-switch supplies are valid, the chip enters run state. The power manager has an inrush current control feature. In boot state, the core supply switches are turned on gradually.

RUN

The run state is the normal operating mode of the device. The device enters run when all power supplies to the chip are valid. Once entering this state, the crystal begins to warm up, on-board calibration is initiated, and the GMSL handshake begins the process of establishing link lock.

SAVED

Saved mode is initiated with an I^2C/UART command (SLEEP = 1) while the device is in run. Before the power manager enters the saved state, the core saves the current device configuration and register values to retention memory. In the saved state, all power switches are turned off and the power manager blocks are disabled. The device enters the sleep state.

Sleep Mode

Sleep mode provides a low power state from which prior configuration information is automatically loaded upon wakeup. This enables very fast recovery from low power sleep to full run operation by eliminating the need for the user to reprogram configuration registers as is required after a full power cycle.

In Run mode (normal operation), writing (SLEEP=1) starts the process of saving device configuration and register settings. The power manager shuts down all internal power supplies, the clocks are disabled, and the chip enters the very low power consumption sleep state. V_{DD18} must remain stable to provide continuous power to the data retention memory, and it is recommended that all supplies be maintained in their nominal operating range.

All GPIO pins are Hi-Z in power-down and sleep states.

Wake-up Sequences

There are two ways to wake up from Sleep mode:

- Local Wake-Up: Local wake-up entails the local host processor initiating a dummy control channel transaction, which briefly wakes up the device. In I2C mode, the initial temporary wake-up requires four falling edges on SDA. Depending on the device address used, there may need to be multiple consecutive dummy transactions issued by the host to achieve the required four SDA falling edges. The dummy transaction(s) must immediately be followed by the host processor setting SLEEP=0 to permanently exit Sleep mode. The device automatically returns to Sleep mode if SLEEP remains set to 1.

- Remote Wake-Up: All GMSL devices include wake-up detectors to observe GMSL link activity and briefly turn on the device when activity is detected. The link can then lock, providing an opportunity for a remote host to disable Sleep mode (setting SLEEP=0). The device automatically reverts to Sleep mode if SLEEP remains set to 1.

Sleep Mode Limitations

Sleep mode should not be used in conjunction with RESET_ALL.

The GMSL2 family includes a global soft reset function called RESET_ALL. This is a self-clearing reset command that is intended to reset all sub-systems to their default configurations. However, if a device has previously gone through a sleep/wake cycle, issuing a RESET_ALL resets the device and erroneously loads the contents of the retention memory that had been stored when the most recent SLEEP command was executed. As a result, the device will reset to the state that had been configured prior to entering Sleep mode previously rather than recovering in a clean power-up default state. The most severe implication of this is that the (SLEEP=1) state is saved in the retention memory, so the device will recover from reset and immediately enter Sleep mode.

Due to the above-described behavior, RESET_ALL and sleep should never be used together.

Not All Registers are Saved

Most key registers corresponding to common device configuration are saved in retention memory. However, applications requiring extensive low-level configuration or infrequently used features may require writing to registers that are not saved in retention after resume from sleep. In these cases, full recovery from Sleep mode to the pre-sleep device state requires some repeated register configuration following resume from sleep. Registers that are stored in retention memory are marked with “*” in the register map in the data sheet.

Voltage Monitoring/ADC

Overview

GMSL2 devices monitor various onboard supply voltages and provide alerts for over- or undervoltage conditions. Dedicated status register flags are driven by internal comparators that measure each supply voltage relative to an internal voltage reference. These registers are latching status flags. Supply voltages with latching status flags retain alerts and can be used to capture temporary conditions and be read back later, at which time the previously latched state is cleared. Many status flags have user-configurable parameters including custom voltage thresholds and the option to report status registers to the ERRB pin, which will assert low when an errant condition is sensed. Reference the GMSL2 Error Reporting (ERRB Pin) section for additional information.

Architecture

Most GMSL2 devices include a common set of power supplies that provide power to the digital core (V_{DD}), GMSL link circuitry (V_{DD18}), and general GPIO pins (V_{DDIO}). All devices include undervoltage monitoring on the common set of power supplies with some providing extended monitoring capabilities. Each GMSL2 product family has unique, family-specific power supplies with dedicated functionality. Examples here include HDMI power (V_{DD33}). These family-specific power supplies are not typically monitored.

The following table details the power supplies for which voltage monitoring is available.

Table 20. Power Supply Monitoring Functions Available for MAX96751

Supply Name	Description	Monitor Functions Available
VDD	Internal 1V digital core supply	Undervoltage
VDD18	GMSL 1.8V supply	Undervoltage
VDDIO	1.8V to 3.3V I/O supply	Undervoltage
VDD33, VDDA	Misc. special function supplies	Not Monitored

Operation

V_{DD} Monitoring Details

Undervoltage (UV) and overvoltage (OV) monitoring of VDD_SW (the primary 1V core supply) is included on nearly all GMSL2 devices (V_{DD} OV monitoring is not available on HDMI and advanced HDMI serializers). VDD_sw is typically derived from VDD/VREG either via an internal LDO or series switches.

Undervoltage Monitoring of V_{DD}

An undervoltage condition on V_{DD} nominally occurs when VDD < 0.82V. Note that the precise threshold varies some between devices. Refer to the data sheet specific to the part number to verify the specified V_{DD} undervoltage threshold for a given device.

In the case of an undervoltage condition, the power manager triggers a reset of the digital core and resets all registers powered by V_{DD}. When power has recovered, the reset is released. The reset of the digital core ensures that a brownout does not result in corruption of the registers.

The presence of an undervoltage event is recorded via two status flags:

Table 21. V_{DD} Undervoltage Status Flags

Bitfield	Description
VDDBAD_STATUS[1], VDDBAD_STATUS[0]	Latched high following undervoltage event.
CMP_STATUS[2]	Latched low following undervoltage.

The memory that maintains the VDDBAD_STATUS and CMP_STATUS[2] bits is powered by the 1.8V power supply; as a result, they are persistent following a reset triggered by a V_{DD} undervoltage event. This enables a V_{DD} undervoltage event to be reported following the recovery of the V_{DD} power supply, and the associated interrupt flags, VDDBAD_INT_FLAG and VDDCMP_INT_FLAG, can drive ERRB low to alert the system of a brownout.

To clear VDDBAD_INT_FLAG, the associated VDDBAD_STATUS bits must be read first to clear. After the VDDBAD_STATUS bits are cleared, VDDBAD_INT_FLAG can then be read, at which point the flag will be cleared. The process to clear VDDCMP_INT_FLAG is similar (i.e., the associated CMP_STATUS bit must be cleared first via reading).

V_{DD18} Monitoring Details

All GMSL2 devices include undervoltage monitoring of V_{DD18}.

Undervoltage Monitoring of V_{DD18}

An undervoltage condition on V_{DD18} is nominally flagged when V_{DD18} < 1.625V. Note that the precise threshold varies some between devices. Refer to the data sheet specific to the part number to verify the specified V_{DD18} undervoltage threshold for a given device.

In the case of an undervoltage condition, the status register bit CMP_STATUS[0] is latched low. The error can be flagged using the VDDCMP_INT_FLAG, which can be configured to drive the fault condition to the ERRB pin. The error status is cleared by first reading CMP_STATUS[0] and then reading VDDCMP_INT_FLAG.

V_{DDIO} Monitoring Details

V_{DDIO} includes undervoltage monitoring only; overvoltage monitoring is not available. This monitoring is available to all GMSL2 devices. An undervoltage condition on V_{DDIO} is nominally flagged when $V_{DDIO} < 1.625V$. Note that the precise threshold varies some between devices. Refer to the data sheet specific to the part number to verify the specified V_{DDIO} undervoltage threshold for a given device.

In the case of an undervoltage condition, the status register bit CMP_STATUS[1] is latched low. The error can be flagged using the VDDCMP_INT_FLAG, which can be configured to drive the fault condition to the ERRB pin. The error status is cleared by first reading CMP_STATUS[1] and then reading VDDCMP_INT_FLAG.

Error Reporting and Status

ERRB Configuration

The ERRB pin can be used to notify the system of undervoltage and overvoltage conditions. Each of the available monitor functions can be separately routed to the ERRB pin as described in the above sections detailing the various voltage monitors. The user-configurable register fields are listed below. These register fields, when set to 1, enable the reporting of the associated undervoltage/overvoltage condition to the ERRB pins (i.e., the error condition will assert ERRB low. Reference the GMSL2 Error Reporting (ERRB Pin) section for additional information.

Table 22. ERRB Pin Reporting Enable for Undervoltage and Overvoltage Conditions for MAX96751

Bitfield	Description
VDDBAD_INT_OEN	Asserts ERRB low when VDDBAD_INT_FLAG = 1 (VDD_sw UV)
VDDCMP_INT_OEN	Asserts ERRB low when VDDCMP_INT_FLAG = 1 (V_{DD18}, V_{DDIO}, V_{TERM}, and/or VDD_sw UV)

Latching Status Bits and Clearing Errors

The status flags that drive ERRB are latching. They are set in the event of an error condition; the error indication is persistent following recovery of the error. The flags and other associated bits are cleared automatically when read. Note that some of the flags are driven by status bits that are also latched and must be cleared prior to clearing the flag. Flags associated with overvoltage conditions only become active when the video path is active. If the video path is not active, the flag bits will not be set in the event of an overvoltage.

Bandwidth Efficiency Optimization

Overview

Before implementing a new design, it is critical to do bandwidth (BW) calculations to verify that the right devices and settings are used. If this is not done, it is possible that data will be lost or corrupted. Although the MAX96751 family can transmit at 6Gbps/187Mbps or 3Gpbs/187Mbps depending on part number and configuration, the maximum allowable video payload is smaller due to the overhead of the GMSL link. The video payload should not exceed the values shown in this table.

Table 23. GMSL2 Maximum Video Payloads

GMSL2 Mode	Maximum Video Payload
3Gbps Mode (NRZ)	2.6Gbps (2600Mbps)
6Gbps Mode (NRZ)	5.2Gbps (5200Mbps)

Calculating Bandwidth

The basic video payload can be calculated using the equations below.

$$PCLK = H_{total} * V_{total} * frame\ rate \quad OR \quad PCLK = \frac{LANE\_CNT * LANE\_RATE}{bpp}$$

$$Video\ Payload = PCLK * bpp$$

Note: (1) H_{total} and V_{total} must include the horizontal and vertical blanking. (2) Use a BPP of 9 when calculating the BW for 8 BPP datatypes. This is the minimum BPP required for the video pipe.

If the lane speeds on the MIPI receiver are fast enough to handle the video payload, the part should not overflow. After calculating the video payload, overhead is added to calculate the total GMSL bandwidth.

$$Video\ BW = [(video\ payload) + (video\ packet\ header) + (video\ pixel\ CRC)] * (9b10b\ encoding) * (sync\ words) * (forward\ error\ correction)^{*}$$

$$Video\ BW = PCLK * \left[(bpp) + \left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\right] * \left(\frac{10}{9}\right) * \left(\frac{2048}{2047}\right) * \left(\frac{128}{120}\right)^{*}$$

The majority of GMSL2 serial link bandwidth comprises video transmission. The total link bandwidth consumed by video is derived from the incoming video stream and calculated by multiplying the pixel clock (PCLK) expressed in MHz, bits per pixel (bpp), and GMSL2 link overhead factors. Note that control channel features (e.g., GPIO, SPI) affect link bandwidth consumption and must be considered if enabled.

Note: “*” only applies when GMSL FEC is enabled. Bandwidth utilization for all control channel and side channel communication increases by 6.7% when forward-error correction (FEC) is enabled. FEC is optional for GMSL2 operation.

Error Flags

Overview

The device contains many internal error detection mechanisms to alert the system or user of any issues. Error flags are register bits that can be checked to see if an error has occurred.

The error bar (ERRB) pin is an MFP pin that logically NORs many of the errors. Whether an error is included in the ERRB output depends on if its output-enable (OEN) bitfield is set high. Most OENs are high by default.

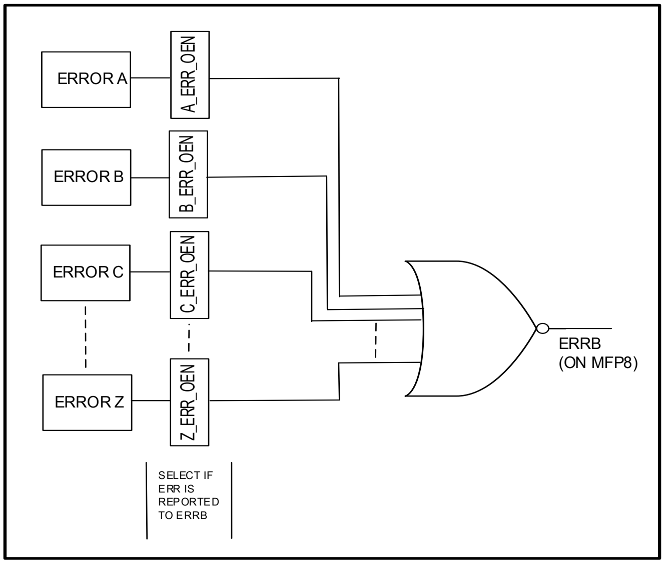


Figure 21. ERRB Reporting Flow

Device Error Flags

MAX96751 ERRB pin (active low) is available on pin 14 and is enabled by default. In addition, register bitfield ERROR (register 0x13 bit 2) goes HIGH when device has an error. *Table 24* below lists all the error flags in this device.

Table 24. MAX96751 Error Flags Table

Error Flag	Error Description
DEC_ERR_FLAG_A	Errors detected in GMSL packet on Link A
DEC_ERR_FLAG_B	Errors detected in GMSL packet on Link B
IDLE_ERR_FLAG	Idle-Word Error Flag on Link A and B
LFLT_INT	Line-Fault Interrupt asserted when either one of line-fault monitors indicates a fault status
HDMI_INT	HDMI interrupt
REM_ERR_FLAG	Received remote side error status
WM_ERR_FLAG	Watermark Error Flag
PKT_CNT_FLAG	Packet Count Error Flag on Link A
RT_CNT_FLAG	ARQ Retransmission Event Flag for Link A
MAX_RT_FLAG	ARQ Maximum Retransmission Event Flag for Link A and B
MEM_INT_ERR_FLAG	Memory Error Flag
VDDBAD_INT_FLAG	V_{DD} supply fault flag
APRBS_ERR_FLAG	Flag is asserted when the Audio PRBS checker detects at least one error (APRBS_ERR > 0) while the Audio PRBS test is running
VDDCMP_INT_FLAG	Flag is asserted when the internal voltage comparator detects that selected internal V_{DD} node is below a threshold.

The HDMI_INT flag is driven by a subset of various HDMI interrupts. These interrupt signals, readable in status registers, can be further configured to output to the ERRB pin to create a hardware interrupt, as shown in *Figure 22*.

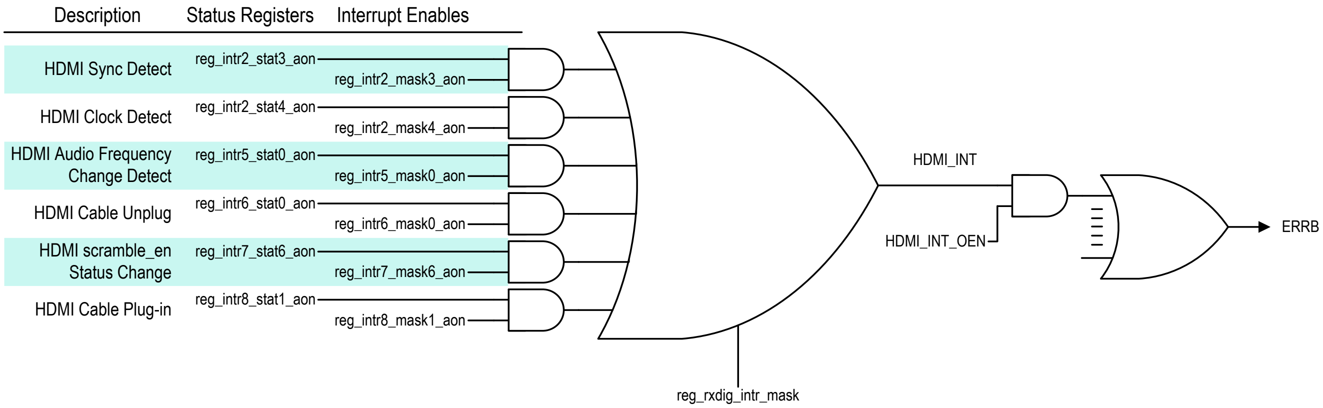


Figure 22. HDMI Fault/Error Register Architecture

To clear the HDMI_INT status, clear the interrupts enabled in registers 0x2080, 0x2081, 0x2083 by writing each status bitfield high (*Table 25*). Reference *Table 26* for the HDMI serializer interrupt enable registers.

To enable an interrupt to be output to the ERRB pin:

- Set the interrupt enable bit given in *Table 25*.
- Enable HDMI receiver "Rx digital interrupt" by setting reg_rxdig_intr_mask to 1 (0x3C11).
- Set HDMI_INT_OEN (0x001A) to 1.

Table 25. HDMI Rx Interrupt Status Registers

Register Address	Register Name	Bitfield	Decode
0x2090	RX_INTR2_MASK_AON	reg_intr2_mask3_aon	Bit 3: HDMI sync detect change interrupt
0x2090	RX_INTR2_MASK_AON	reg_intr2_mask4_aon	Bit 4: TMDS clock detect change interrupt
0x2354	RX_INTR5_MASK	reg_intr5_mask0	Bit 0: Audio sample frequency change event interrupt
0x2091	RX_INTR6_MASK_AON	reg_intr6_mask0_aon	Bit 0: HDMI cable unplug interrupt
0x2092	TX_INTR7_MASK_AON	reg_intr7_mask6_aon	Bit 6: HDMI TMDS register scramble_en change event interrupt (indicates a change between HDMI 1.4 and HDMI 2.0 mode)
0x2093	RX_INTR8_MASK_AON	reg_intr8_mask1_aon	Bit 1: HDMI cable plug-in interrupt

Table 26. GMSL2 HDMI Serializer Interrupt Enable Registers

Register Address	Register Name	Bitfield	Decode
0x3C11	INTR_STAT0	reg_intr2_mask3_aon	Bit 3: HDMI sync detect change interrupt
0x001B	INTR3	reg_intr2_mask4_aon	Bit 4: TMDS clock detect change interrupt
0x001A	INTR2	reg_intr5_mask0	Bit 0: Audio sample frequency change event interrupt

The HDMI_INT flag is driven by a subset of various HDMI interrupts. These interrupt signals, readable in status registers, can be further configured to output to the ERRB pin to create a hardware interrupt, as shown in *Figure 23*.

To clear the HDMI_INT status, clear the interrupts enabled in registers 0x2080, 0x2081, 0x2083 by writing each status bitfield high (*Table 28*). Reference *Table 29* for the HDMI serializer interrupt enable registers.

To enable an interrupt to be output to the ERRB pin:

- Set the interrupt enable bit given in *Table 29*.
- Enable HDMI receiver "Rx digital interrupt" by setting reg_rxdig_intr_mask to 1 (0x3C11).
- Set HDMI_INT_OEN (0x001A) to 1.

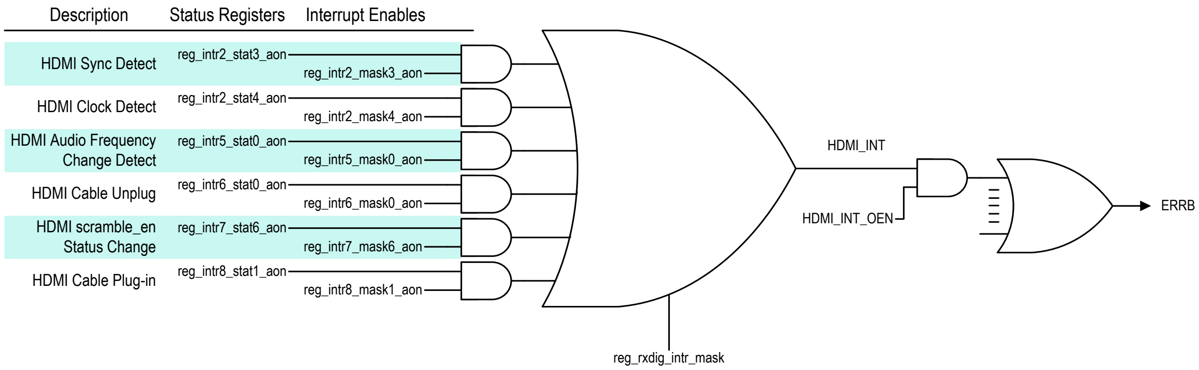


Figure 23. HDMI Fault/Error Register Architecture

Table 27. GMSL2 HDMI Serializer Interrupt Enable Registers

Register Address	Register Name	Bitfield	Decode
0x3C11	INTR_STAT0	reg_rxdig_intr_mask	Bit 7: HDMI error enable (direct HDMI interrupts to serializer error handling)

Each interrupt has a latched status register (*Table 28*). Write 0x1 to clear the bit. The following interrupts are available to route to HDMI_INT:

Table 28. HDMI Rx Interrupt Status Registers

Register Address	Register Name	Bitfield	Decode
0x2080	*RX_INTR2_AON*	reg_intr2_stat3_aon	Bit 3: HDMI sync detect change interrupt
0x2080	*RX_INTR2_AON*	reg_intr2_stat4_aon	Bit 4: TMDS clock detect change interrupt
0x2344	*RX_INTR5*	reg_intr5_stat0	Bit 0: Audio sample frequency change event interrupt
0x2081	*RX_INTR6_AON*	reg_intr6_stat0_aon	Bit 0: HDMI cable unplug interrupt
0x2082	*RX_INTR7_AON*	reg_intr7_stat6_aon	Bit 6: HDMI TMDS register scramble_en change event interrupt (indicates a change between HDMI 1.4 and HDMI 2.0 mode)
0x2083	*RX_INTR8_AON*	reg_intr8_stat1_aon	Bit 1: HDMI cable plug-in interrupt

Table 29. GMSL2 HDMI Rx Interrupt Enable

Register Address	Register Name	Bitfield	Decode
0x2090	RX_INTR2_MASK_AON	reg_intr2_mask3_aon	Bit 3: HDMI sync detect change interrupt enable
0x2090	RX_INTR2_MASK_AON	reg_intr2_mask4_aon	Bit 4: TMDS clock detect change interrupt enable
0x2354	RX_INTR5_MASK	reg_intr5_mask0	Bit 0: Enable audio sample frequency change event interrupt
0x2091	RX_INTR6_MASK_AON	reg_intr6_mask0_aon	Bit 0: HDMI cable unplug interrupt enable
0x2092	TX_INTR7_MASK_AON	reg_intr7_mask6_aon	Bit 6: Enable interrupt for scramble_en change event (indicates a change between HDMI 1.4 and HDMI 2.0 mode)
0x2093	RX_INTR8_MASK_AON	reg_intr8_mask1_aon	Bit 1: HDMI cable plug-in interrupt enable

Video Pattern Generator (VPG)

Overview

The video pattern generator (VPG) creates either a checkerboard or gradient pattern with programmable parameters. These patterns can be used to replace the incoming video or in conjunction with the VTG to create an RGB888 video pattern when no video is present on the serializer input.

The serializer and deserializer have an internal VPG that accommodates a wide range of resolutions and frame rates. Link lock is not required to use the VPG when being outputted on the deserializer, link lock is required if using serializer as VPG source.

The VPG has its own register block settings for timing configurations. See below tables for register settings required.

The GMSL SerDes GUI can be used to set up the VPG and to generate VPG register write example codes.

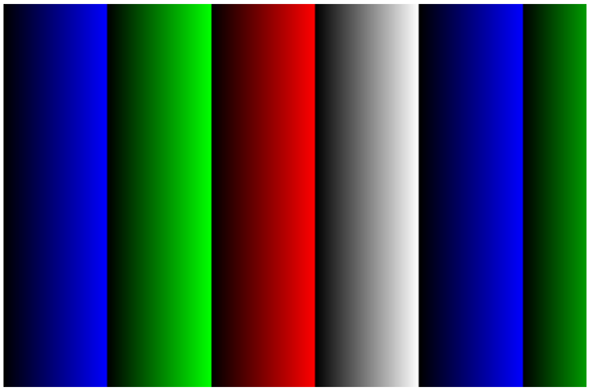


Figure 24. VPG - Gradient Pattern

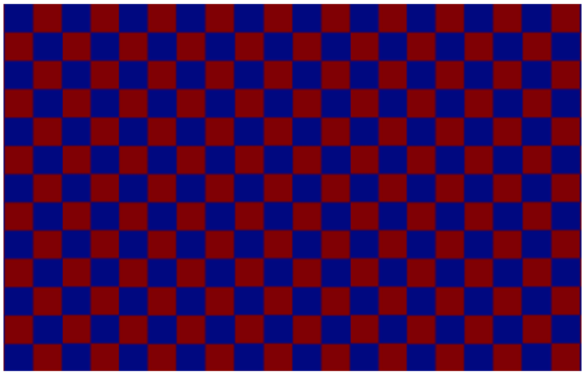


Figure 25. VPG - Checkerboard Pattern

VPG is available on Pipe X, and external PCLK can be provided via GPIO4.

Table 30. MAX96751 Video Pattern Registers

Register	Bits	Default Value	Description
0x1C8	7:0	0x03	**Pattern Generator 0/1 Register:** Bit 7: Generate VS according to the timing setting. Bit 6: Generate HS according to the timing setting. Bit 5: Generate DE according to the timing setting. Bit 4: Invert VSYNC of Video Timing Generator. Bit 3: Invert HSYNC of Video Timing Generator. Bit 2: Invert DE of Video Timing Generator. Bits [1:0]: Video Interface Timing Generation Mode.
0x1E5	0:1	0x00	**Pattern Generator Mode Register:** Bits [5:4]: Pattern selection. 01 = Color Gradient 10 = Checkerboard
0x1CA	7:0	0x00	**VS Delay Register 2:** VS Delay in terms of PCLK cycles. (Bits [23:16])

0x1CB	7:0	0x00	**VS Delay Register 1:** VS delay in terms of PCLK cycles. (Bits [15:8])
0x1CC	7:0	0x00	**VS Delay Register 0:** VS delay in terms of PCLK cycles. (Bits [7:0])
0x1CD	7:0	0x00	**VS High Register 2:** VS high period in terms of PCLK cycles. (Bits [23:16])
0x1CE	7:0	0x00	**VS High Register 1:** VS high period in terms of PCLK cycles. (Bits [15:8])
0x1CF	7:0	0x00	**VS High Register 0:** VS high period in terms of PCLK cycles. (Bits [7:0])
0x1D0	7:0	0x00	**VS Low Register 2:** VS low period in terms of PCLK cycles. (Bits [23:16])
0x1D1	7:0	0x00	**VS Low Register 1:** VS low period in terms of PCLK cycles. (Bits [15:8])
0x1D2	7:0	0x00	**VS Low Register 0:** VS low period in terms of PCLK cycles. (Bits [7:0])
0x1D3	7:0	0x00	**V2H Register 2:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [23:16])
0x1D4	7:0	0x00	**V2H Register 1:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [15:8])
0x1D5	7:0	0x00	**V2H Register 0:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [7:0])
0x1D6	7:0	0x00	**HS High Register 1:** HS high period in terms of PCLK cycles. (Bits [15:8])
0x1D7	7:0	0x00	**HS High Register 0:** HS high period in terms of PCLK cycles. (Bits [7:0])
0x1D8	7:0	0x00	**HS Low Register 1:** HS low period in terms of PCLK cycles. (Bits [15:8])
0x1D9	7:0	0x00	**HS Low Register 0:** HS low period in terms of PCLK cycles. (Bits [7:0])
0x1DA	7:0	0x00	**HS Count Register 1:** HS pulses per frame. (Bits [15:8])
0x1DB	7:0	0x00	**HS Count Register 0:** HS pulses per frame. (Bits [7:0])
0x1DC	7:0	0x00	**V2D Register 2:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [23:16])
0x1DD	7:0	0x00	**V2D Register 1:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [15:8])
0x1DE	7:0	0x00	**V2D Register 0:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [7:0])
0x1DF	7:0	0x00	**DE High Register 1:** DE high period in terms of PCLK cycles. (Bits [15:8])

0x1E0	7:0	0x00	**DE High Register 0:** DE high period in terms of PCLK cycles. (Bits [7:0])
0x1E1	7:0	0x00	**DE Low Register 1:** DE low period in terms of PCLK cycles. (Bits [15:8])
0x1E2	7:0	0x00	**DE Low Register 0:** DE low period in terms of PCLK cycles. (Bits [7:0])
0x1E3	7:0	0x00	**DE Count Register 1:** DE pulses per frame. (Bits [15:8])
0x1E4	7:0	0x00	**DE Count Register 0:** DE pulses per frame. (Bits [7:0])
0x1E6	7:0	0x04	**Gradient Increment Register:** Set color gradient increment.
0x1E7	7:0	0x00	**CHRK_COLOR_A_L Register:** Checkerboard mode color A low byte. RGB888 color Red (0-255).
0x1E8	7:0	0x00	**CHRK_COLOR_A_M Register:** Checkerboard mode color A mid byte. RGB888 color Green (0-255).
0x1E9	7:0	0x00	**CHRK_COLOR_A_H Register:** Checkerboard mode color A high byte. RGB888 color Blue (0-255).
0x1EA	7:0	0x00	**CHRK_COLOR_B_L Register:** Checkerboard mode color B low byte. RGB888 color Red (0-255).
0x1EB	7:0	0x00	**CHRK_COLOR_B_M Register:** Checkerboard mode color B mid byte. RGB888 color Green (0-255).
0x1EC	7:0	0x00	**CHRK_COLOR_B_H Register:** Checkerboard mode color B high byte. RGB888 color Blue (0-255).
0x1ED	7:0	0x00	**CHRK_RPT_A Register:** Checkerboard mode color A repeat count.
0x1EE	7:0	0x00	**CHRK_RPT_B Register:** Checkerboard mode color B repeat count.
0x1EF	7:0	0x00	**CHRK_ALT Register:** Checkerboard mode alternate line count.

Table 31. MAX96751 VPG PCLK Setting Registers

Register	Bits	Default Value	Description
0x1C9	4	0b0	**SEL_EXT_PCLK:** whether to use GPIO4 as PCLK input instead of the clock received from HDMI input Decode: 0b0 = Use clock received from HDMI input 0b1 = Use GPIO4 as PCLK

Video Timing Generator (VTG)

Overview

The video timing generator (VTG) generates or adjusts video sync signals. It may be used to modify the timing details of incoming video streams, or it may be used with the video pattern generator (VPG) to generate test images. The VTG is implemented within the video pipe of the serializer. The MAX96751 all have the same VTG functionality.

The VTG provides user-configurable options for the video sync signals: vertical sync (VS), horizontal sync (HS), and data enable (DE). If enabled, the VTG regenerates the video sync signals in accordance with user defined timing parameters. These parameters offer flexibility to customize the sync polarity, pulse width, and timing of the regenerated video sync signals.

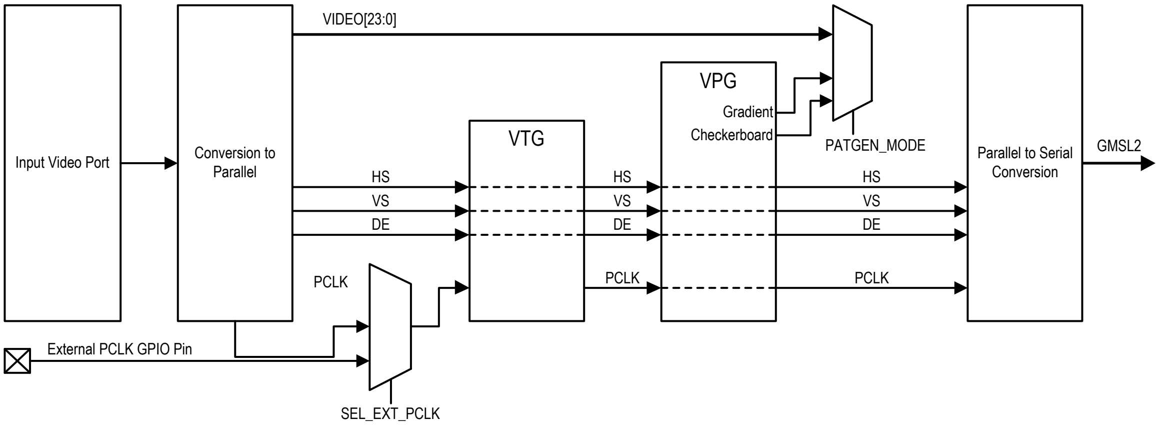


Figure 26. MAX96751 Video Timing Generator Block Diagram

VTG Operation

The core function of the VTG block is to generate VS, HS, and DE signals based on a trigger. The VTG can be configured to generate these signals internally or, if the VTG is not enabled, then the VS/HS/DE signals at the input of the VTG drive the output signals. This selection is made individually for each sync signal via the GEN_VS, GEN_HS, and GEN_DE register bits.

The VTG can be triggered by either a VS input transition (i.e., the external trigger) or an internally generated VS trigger (i.e., the tracking VS signal). In the case of the external trigger, the VS transition trigger is selected to be either the rising edge or the falling edge of the input VS signal with the VS_TRIG bit. Note that the selected edge for the input transition is referred to as the active edge. The polarity, start timing (i.e., delay from the trigger), periodicity, duty-cycle, and the number of HS and DE pulses per frame are all programmable.

Note: After the VTG is enabled, the first and/or second frame sync output pulses (VS/HS/DE) may be invalid.

VTG Configuration

Configuring the VTG consists of two steps: selecting the VTG operation mode and configuring the timing parameters for the VS, HS, and DE generation as shown in the table below.

Table 32. MAX96751 VTG Operation Mode

Parameter	Register	Bitfield	Decode and Description
VS Generation Enable	VTX0	GEN_VS	0: Bypass VTG and use the VS signal from the video input 1: Generate VS signal from the VTG with specified timing
HS Generation Enable	VTX0	GEN_HS	0: Bypass VTG and use the HS signal from the video input 1: Generate HS signal from the VTG with specified timing

DE Generation Enable	VTX0	GEN_DE	0: Bypass VTG and use the DE signal from the video input 1: Generate DE signal from the VTG with specified timing
VS trigger mode	VTX0	VTG_MODE	00: VS Tracking mode 01: VS One-trigger mode 10: Auto-repeat mode 11: Free-running mode (default)
VS trigger polarity	VTX0	VS_TRIG	0: Falling edge 1: Rising edge (default)
Sync signal polarity	VTX0	VS_INV	The VTG timing configuration instructions assume positive sync polarity (sync pulse active high), so if negative VS polarity is desired, use this bit to invert 0: Do not invert VS signal 1: Invert VS signal Note: This bit is active even when GEN_VS=0 and can be used to invert the VS polarity without configuring the VTG timing parameters
Sync signal polarity	VTX0	HS_INV	The VTG timing configuration instructions assume positive sync polarity (sync pulse active high), so if negative HS polarity is desired, use this bit to invert 0: Do not invert HS signal 1: Invert HS signal Note: This bit is active even when GEN_HS=0 and can be used to invert the HS polarity without configuring the VTG timing parameters
Sync signal polarity	VTX0	DE_INV	The VTG timing configuration instructions assume positive sync polarity (sync pulse active high), so if negative DE polarity is desired, use this bit to invert 0: Do not invert DE signal 1: Invert DE signal Note: This bit is active even when GEN_DE=0 and can be used to invert the DE polarity without configuring the VTG timing parameters

VTG Trigger Modes

There are four different VTG trigger modes:

- VS Tracking mode – This mode is used to reduce the glitches and jitters of the input VS signal. In this mode, the input VS period (VS_HIGH + VS_LOW) is tracked. After the VS tracking has locked, any VS input edge not in the expected PCLK cycle (e.g., glitch) is ignored. VS tracking is locked upon three consecutive periodic matches; VS tracking is unlocked following three consecutive periodic mismatches. At power-up or if VS tracking is unlocked, the next VS input edge is assumed to be the correct VS edge.
- VS One-trigger mode – In this mode, only one frame of VS, HS, and DE output signals is generated per VS input trigger. The polarity, timing (delay from the VS input trigger), and period/duty-cycle of the generated VS, HS, and DE signals are in accordance with the user-programmed parameters.
- Auto-repeat mode – This mode uses the VS input trigger to generate VS, HS, and DE signals as with VS One-trigger mode. However, instead of one frame per VS input trigger, Auto-repeat mode generates continuous frames of VS, HS, and DE output signals following a VS input trigger. If the next VS input edge occurs earlier or

later than expected by the VS period (VS_HIGH + VS_LOW), the newly generated frame will be considered correct. The previous VS/HS/DE signals will be cut or extended at the time point of the rising edge of the newly generated VS, HS, and DE signals.

- Free-running mode (default) – This mode is based on Auto-repeat mode. In this mode, the VS input signal is not needed to generate continuous frames of VS, HS, and DE output signals. The VTG automatically starts generating a continuous stream of frames, consisting of VS, HS, and DE signals in accordance with the user-programmed parameters.

VTG Timing Parameters

The sync pulse timing parameters for the VTG are shown in the figure below. Configuring the timing parameters for the VS, HS, and DE generation is shown in the *Table 33*.

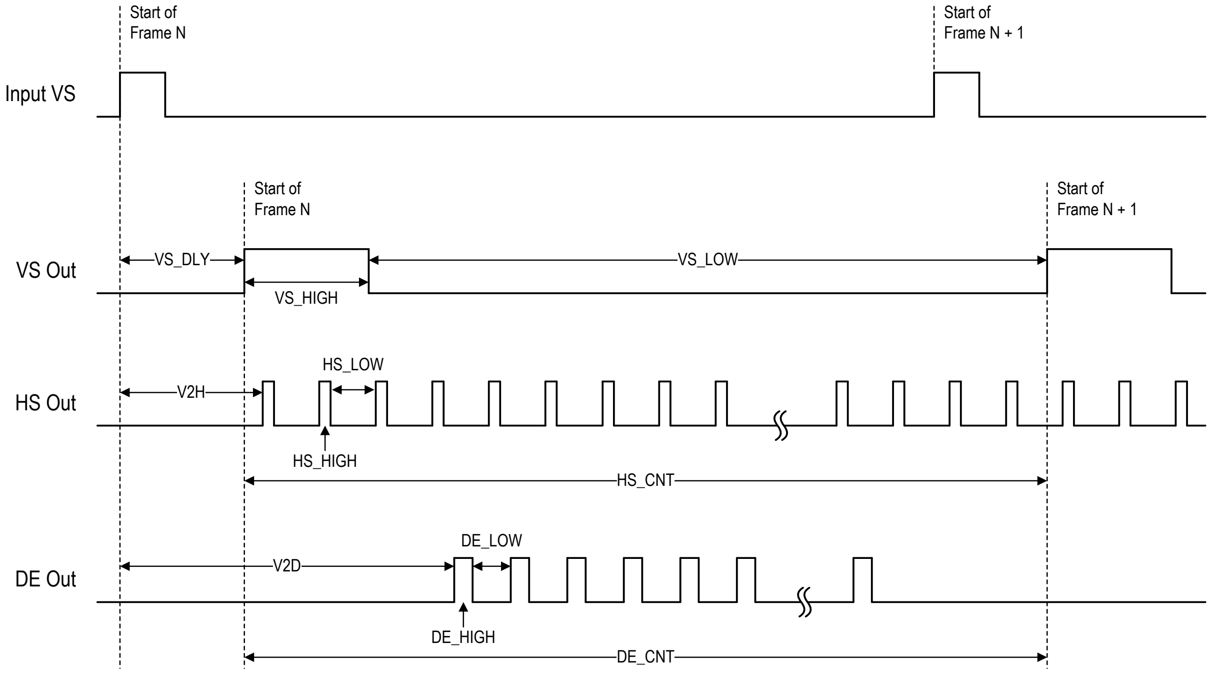


Figure 27. VTG Timing Parameters Diagram

All parameters are defined in terms of PCLKs. Vertical timing parameters must be converted from lines to pixel clocks by multiplying by H_{total}.

Only the parameters associated with the sync pulses that are enabled must be configured. For example, if only VS is being generated by the VTG (GEN_VS=1, GEN_HS=0, GEN_DE=0), only VS_DLY, VS_HIGH, and VS_LOW must be configured.

Note: In cases where the VTG is used to regenerate DE in combination with incoming video data, the incoming data is not retimed to DE if it is adjusted from the original timing. This will result in lost video data. For example, if DE is delayed four PCLKs, the first four pixels from the incoming data will be lost and the last four will be converted to zeros (i.e., padded).

Table 33. MAX96751 Timing Parameter Configuration Registers

Parameter	Register	Bitfield	Description
VS_DLY	VTX2 VTX3 VTX4	VS_DLY_2[7:0] VS_DLY_1[7:0] VS_DLY_0[7:0]	The delay from the VS trigger to the generated VS signal in terms of PCLKs. Note: if using the input video stream, this will delay the output sync signals relative to the video data.
VS_HIGH	VTX5 VTX6 VTX7	VS_HIGH_2[7:0] VS_HIGH_1[7:0] VS_HIGH_0[7:0]	The high duration of the generated VS output signal in terms of PCLKs
VS_LOW	VTX8 VTX9 VTX10	VS_LOW_2[7:0] VS_LOW_1[7:0] VS_LOW_0[7:0]	The low duration of the generated VS output signal in terms of PCLKs
V2H	VTX11 VTX12 VTX13	V2H_2[7:0] V2H_1[7:0] V2H_0[7:0]	The delay from the VS trigger to the rising edge of the generated HS signal
HS_HIGH	VTX14 VTX15	HS_HIGH_1[7:0] HS_HIGH_0[7:0]	The high duration of the generated HS output signal in terms of PCLKs
HS_LOW	VTX16 VTX17	HS_LOW_1[7:0] HS_LOW_0[7:0]	The low duration of the generated HS output signal in terms of PCLKs
HS_CNT	VTX18 VTX19	HS_CNT_1[7:0] HS_CNT_0[7:0]	The number of HS output pulses generated per video frame
V2D	VTX20 VTX21 VTX22	V2D_2[7:0] V2D_1[7:0] V2D_0[7:0]	The delay from the VS trigger to the rising edge of the generated DE signal
DE_HIGH	VTX23 VTX24	DE_HIGH_1[7:0] DE_HIGH_0[7:0]	The high duration of the generated DE output signal in terms of PCLKs
DE_LOW	VTX25 VTX26	DE_LOW_1[7:0] DE_LOW_0[7:0]	The low duration of the generated DE output signal in terms of PCLKs
DE_CNT	VTX27 VTX28	DE_CNT_1[7:0] DE_CNT_0[7:0]	The number of DE pulses generated per video frame

Complete Use Case Programming Examples

The following use case examples demonstrate how many of the features described throughout this document can be used together to program a SerDes system. These examples may need to be manipulated or completely changed for more specific use cases. The basic flow of programming and important steps is annotated to give a broad picture of the requirements users can expect to get a system working with the MAX96751 serializer and MAX96752 deserializer. The format of the programming examples throughout this section will follow the format allowable by the GMSL GUI for .csv files, so that users may copy them for use immediately.

Use Case Example 1

This example has the following characteristics:

- One MAX96751 connected to one MAX96752 using GMSL link A only
- Initial I2C address of serializer is 0x80

- Initial I2C address of deserializer is 0x90
- Link rate = 6 Gbps
- The serializer receives HDMI data on its HDMI input port
- The deserializer outputs dual oLDI data on oLDI ports A and B

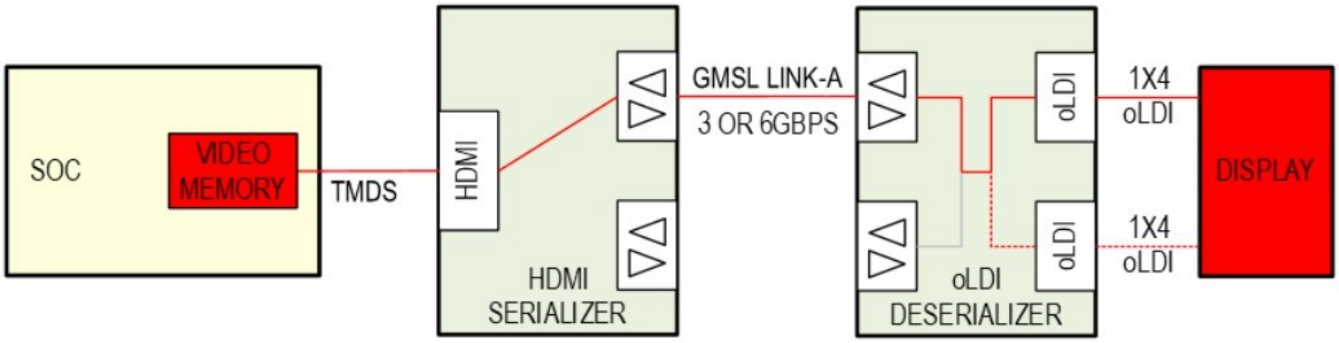


Figure 28. Use Case Example 1 Diagram

Use Case Example 1 Script

#Single Video Stream, Single GMSL2 Link at Either 3Gbps or 6Gbps, OLDI Deserializer with Single or Dual OLDI Outputs

#PHY A, Dual OLDI

#SER 0x80

#DES 0x90

#Video Start

0x80,0x01,0xC8

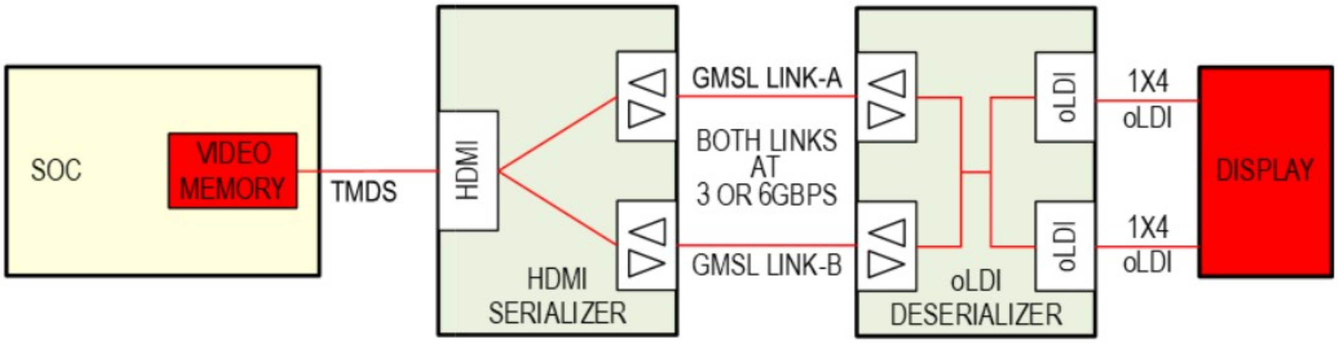
#DES

0x90,0x1CE,0x5E

Use Case Example 2

This example has the following characteristics:

- One MAX96751 connected to one MAX96752 using GMSL Links A and B
- Initial I2C address of serializer is 0x80
- Initial I2C address of deserializers is 0x90
- Link rate = 6 Gbps (on each PHY)
- The serializer receives HDMI data on its HDMI input port
- The deserializer outputs dual OLDI data on OLDI ports A and B



Use Case Example 2 Script

```
#Single Video Stream, Dual GMSL2 Links with Both at Either 3Gbps or 6Gbps. OLDI Deserializer with Single or Dual OLDI

#DUAL LINK, Dual OLDI

#SER 0x80

#DES 0x90

#DES

0x90,0x10,0x00

0x90,0x10,0x20

#SER

0x80,0x10,0x00

0x80,0x10,0x20

0x80,0x53,0x30

#Video Start

0x80,0x01,0xC8

#DES

0x90,0x1CE,0x5E
```