AHEAD OF WHAT'S POSSIBLE™

MAX96714/MAX96714F/MAX96714R User Guide

GMSL SerDes Applications Team

Rev 0

05/2023

Table of Contents

Figures

Tables

MAX96714/MAX96714F/MAX96714R Serializers

Device Overview

Use this user guide in conjunction with the MAX96714 data sheets, errata documents, and other user and design guides. It provides explanations, examples, and instructions to set up video configurations and use various features. Within the examples, assume the serializer I^2C address to be 0x80, unless otherwise noted.

The examples may not have errata writes necessary to ensure reliable operation in production. Be sure to contact your Analog Devices field applications engineer or representative for the errata documents. 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 MAX96714 device for testing.

The MAX96714 family consists of the MAX96714 (plain, with no suffix in the base part number), MAX96714F, and MAX96714R. The MAX96714 is the full-feature device. The MAX96714F omits the 6 Gbps mode, and MAX96714R does not have several features, including the 6 Gbps mode. In some cases, the document refers to them all as just MAX96714, unless it is necessary to differentiate the three.

Table 1. Comparing the MAX96714 Family

PART NUMBER	FORWARD LINK RATE	ASIL RATING	FUNCTIONAL SAFETY FEATURES (E.G., CRC)	PACKAGE	I^2C/UART PASS-THROUGH
MAX96714	3 Gbps or 6 Gbps	B	Yes	Wettable	Yes
MAX96714F	3 Gbps	B	Yes	Wettable or Non-Wettable	Yes
MAX96714R	3 Gbps	QM	No	Wettable or Non-Wettable	No

Note: This is a not a complete list of device differences. Refer to the device data sheets for all feature details.

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. The MAX96714 is capable of 3 Gbps or 6 Gbps forward link rate (selectable with register writes) and the MAX96714F and MAX96714R are capable of 3 Gbps. All variants have a 187.5 Mbps reverse direction rate. In addition, all variants are capable of GMSL1 links.

Application Use Case

Compared to other deserializers, the MAX96714 has only one gigabit multimedia serial link (GMSL) input and one camera serial interface (CSI) output. Thus, the main application use case is a single link represented in ***Figure 1***.

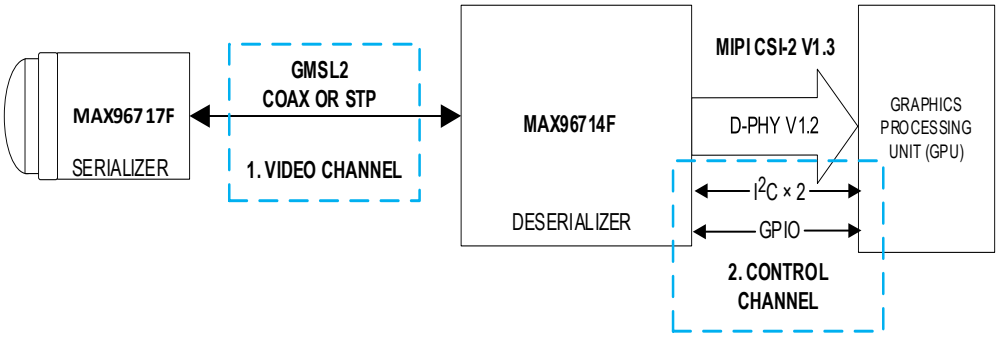

Figure 1. MAX96714, Single Camera Application

In a typical configuration, the serializer is used to support cameras in the 1 MP to 8 MP range. Typically, the camera's image sensor feeds the video into the CSI input port of the serializer. The serializer then takes that data, converts it to GMSL, and sends it out over the link to the MAX96714.

Architecture

Figure 2 shows the video path within the GMSL2 CSI-2 deserializer D-PHY architecture.

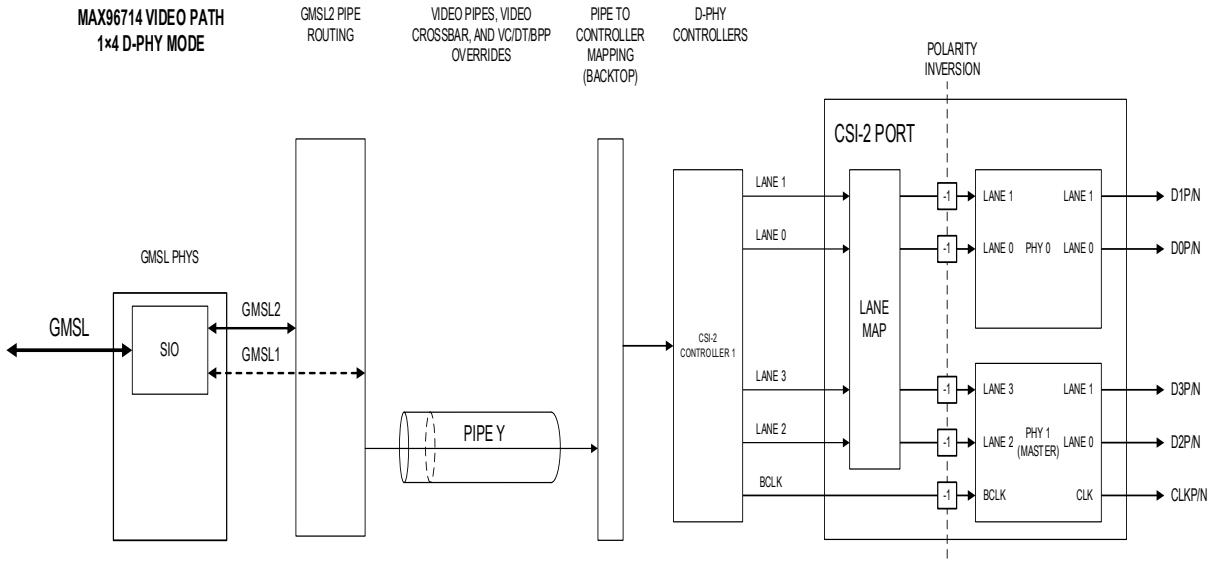

Figure 2. MAX96714 Video Path (1x4 D-PHY Mode)

Start-Up and Programming Sequence

Overview

The MAX96714 family has many application use cases and features that work in conjunction with each other. To avoid feature and system sequencing issues, *Table 2* outlines the preferred order. Features or configuration changes may not be required and may be skipped in the start-up sequence. This depends on system requirements and data configurations.

Recommended Start-Up Sequence

*Table 2. MAX96714/ MAX96714F/MAX96714R Start-Up and Programming Sequence (*Some Features are Not Available on the MAX96714R*)*

GENERAL DEVICE SETTINGS		
SEQUENCE NUMBER	**FEATURE ENABLE ORDER**	**NOTES**
*	**Important note regarding configuration changes.	Perform any configuration before the video runs. Stop the video for a configuration change.
*	Voltage supply ramp order is independent.	No voltage sequencing required.
1	Configuration pins	Starting point (reboot if changed). Device automatically reads configuration pin levels.
2	I^2C/UART wake time	Device automatically powers up and enables communication from local I^2C or UART transaction.
3	GMSL wake up	GMSL automatically locks to GMSL signals from remote serializer.
4	Primary I^2C configuration and CRC	Changes (e.g., CRC ARQ S/H times, bitrate)
5	I^2C pass-through configuration and CRC (MAX96714/MAX96714F only)	Changes (e.g., CRC ARQ S/H times, bitrate)
6	Interrupt handling (ERRB)	Turn on and off error reporting (e.g., line fault, remote errors).
7	RESET_LINK = 1	Used for GMSL2 links only.
8	Retransmission of reverse channel packets during CRC error	Set by default.
9	GPIO configuration and CRC	
10	VSYNC and HSYNC output configuration	Normally used for debug purposes.
11	SSC configuration and enable	
VIDEO ROUTING AND CONFIGURATION		
*	Video pipe configuration (not common)	Bits per pixel (bpp) manipulations. Enable/disable heartbeat mode. Stream ID selection.
12	Line CRC (32-bit)	Enabled by default.
13	Video Pixel CRC (16-bit)	Disabled by default.
14	VID_EN	Enabled by default.
15	VC/DT override	
16	VC/DT mapping	

17	Pixel doubling for efficiency	Dependent on bpp.
18	ERRB forwarding	Disable/enable ERRB forwarding packet.
19	Crossbar switch	
20	MIPI D-PHY setup	Lane map, count, swap, and polarity Inversion.
21	MIPI D-PHY deskew	Required above 1.5 Gbps/lane.
22	MIPI D-PHY Tx configuration	CSIPLL frequency (data rate)
23	FSYNC setup	Configure but do not start.
GMSL2 SETUP (IF USED)		
24	GMSL link configuration	Link rate, Coax/STP
25	RESET_LINK = 0	Used for GMSL2 mode only
26	RoR Handshake	Register configuration can continue beyond this point. However, RESET_LINK or ONESHOT_RESET must not be toggled until Link Lock for fastest GMSL2 Link Lock time.
27	AEQ	
28	SYNC lock	
29	Lock info frames	
30	LINK lock	
31	Eye opening monitor/mapper	
GMSL1 SETUP (IF USED)		
32	Enable PKTCC and HIM mode	
33	GMSL1 specific errors and mapping	
34	Color format mapping to GMSL2	
35	CNTL pin mapping	
36	GMSL1 FSYNC setup	
37	Set I^2C local ACK	
38	Start CLINK from unlocked channel	
GENERAL DEVICE SETTINGS (CONTINUED)		
	Serializer settings	See serializer user guide.
39	Line fault setup	
40	MIPI Video PRBS or pattern generator setup (if used)	
	Image sensor/video source setup	
41	Stat video from source	Start camera or image sensor.
42	Start FSYNC	
43	Start serializer link	(GMSL1 only)
44	Disable I^2C local ACK	(GMSL1 only)
45	Verify PCLK detect (serializer)	
46	Verify video lock detected	
47	No LCRC or video memory errors detected	
48	csi2_tx1_pkt_cnt detected	Verify deserializer output.
49	phy_pkt_cnt detected	Verify deserializer output.

Configuration

Overview

The forward video paths of the MAX96714/MAX96714F/MAX96714R serializers are configured with the following programming:

- Pixel and Tunneling Mode
- Link Initialization
- Link Lock Check
- Video Pipe Selection
- MIPI PHY Settings
- Datatype (DT)/Virtual Channel (VC) Overrides (including extended virtual channels)

Enable the video only after the video path is configured; **dynamic configuration is not supported.** The following sub-sections detail the operation of each of these steps with descriptions of relevant registers and programming examples.

Pixel and Tunneling Mode

The MAX96714/MAX96714F/MAX96714R support both the pixel and tunneling modes. Always ensure to match the serializer and deserializer modes.

The pixel mode helps systems to manipulate data types, bits per pixel, and virtual channels. Use this mode to manipulate the incoming data over the serial link before outputting from the deserializer.

Use the tunneling mode when data integrity is a major system concern as it ensures end-to-end data integrity. End-to-end data protection is a common requirement for advanced driver assistance systems (ADAS), where data may not be altered from the transmitter to the downstream receiver. In the tunneling mode, data may not be changed as it is protected with an end-to-end cyclic redundancy check (CRC) and is passed from serializer to deserializer without any manipulation. In the tunneling mode, any combination of data type, bits per pixel (bpp), and virtual channel may be transmitted if the video bandwidth total does not exceed the link bandwidth.

Note: The tunnel mode requires the serializer to use D-PHY (use the MAX96724 for C-PHY serializers).

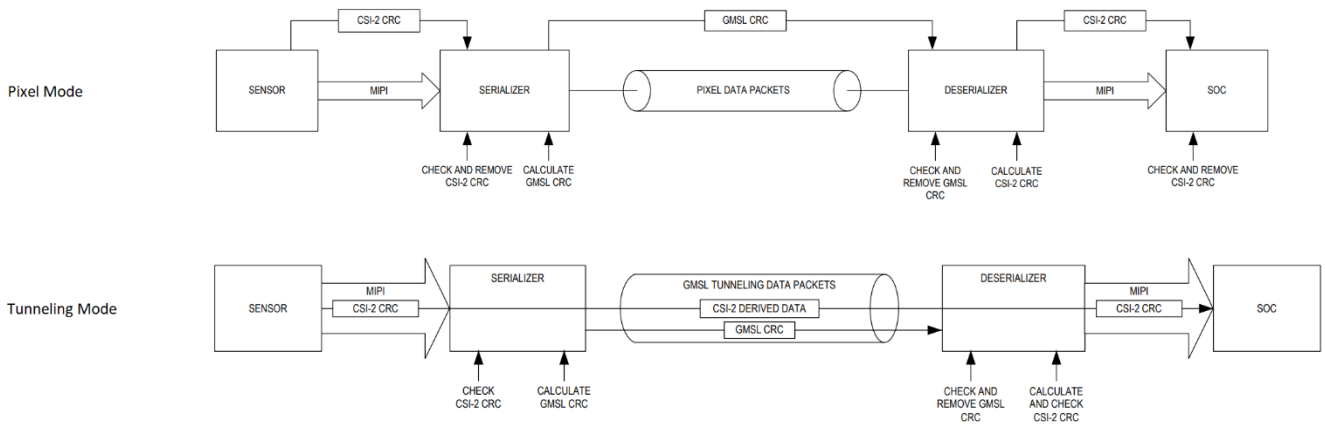

Figure 3. Tunneling and Pixel Mode

Table 3. Features Supported by the Pixel vs. Tunneling Mode

FEATURE	PIXEL MODE	TUNNELING MODE
Virtual Channel Reassignment*	Supported	Not Supported
End-to-End CRC Coverage for Video	Not Supported	Supported
Video Line CRC (LCRC)	Supported	Supported
GMSL Packet CRC (VID_PXL_CRC)	Supported	Supported
Line Length >8k Pixels at 24 BPP	Not Supported	Supported
16-Channel Virtual Channel Support	Supported	Supported

Note*: Video source must set different VCs, as the tunneling mode does not support VC reassignment to ensure data integrity.

Tunnel Mode Configuration Registers

Table 4. Tunnel Mode Register Table

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0474	0	0x08	Tunnel Mode Enable Bit 0: Tunnel Mode Enable.

Tunnel Mode Configuration Programming Examples

```
# Enable Tunnel mode manually. TUN_EN = 1.
0x98, 0x474, 0x0x09
# Reset one-shot to reset the link
0x98, 0x010, 0x31
```

Link Initialization

Link initialization establishes the device link modes and speeds. The MAX96714/MAX96714F/MAX96714R device family is a GMSL1/GMSL2, single-port deserializer that can support coax or shielded-twisted pair (STP) cables. The MAX96714 variant can receive 3 Gbps or 6 Gbps GMSL2 data in the forward direction over the GMSL link while the MAX96714F and MAX96714R variants are only capable of 3 Gbps GMSL2. All variants support GMSL1 data up to 3.125 Gbps. Using the following registers, select the GMSL link rate and coax or STP cabling. Any changes to the GMSL link should be followed by a link reset to reinitialize the link (toggle RESET_LINK HIGH and then LOW). CFG pins are the preferred method of setting up the GMSL rate and transmission mode (*Table 5*). The selected configuration becomes the new default on power-up once the CFG pins are set and the part is power cycled.

Table 5. MAX96714 Basic Settings

CFG1 VALUE	COAX/STP	DATA RATE	TRANSMISSION MODE
0	RSVD	RSVD	RSVD
1	STP	6 Gbps	Tunneling Mode
2	RSVD	RSVD	RSVD
3	STP	6 Gbps	Pixel Mode
4	RSVD	RSVD	RSVD
5	Coax	6 Gbps	Tunneling Mode
6	RSVD	RSVD	RSVD
7	Coax	6 Gbps	Pixel Mode

Note: Use the register settings (Register 0x0001) to use the 3 Gbps mode.

Table 6. MAX96714F/MAX96714R Basic Settings

CFG1 VALUE	COAX/STP	GMSL1/GMSL2	TRANSMISSION MODE (GMSL2)	HIM/LEGACY MODE (GMSL1)
0	Coax	GMSL2	Tunneling Mode	–
1	Coax	GMSL1	–	HIM
2	Coax	GMSL1	–	Legacy
3	STP	GMSL2	Tunneling Mode	–
4	STP	GMSL2	Pixel Mode	–
5	STP	GMSL1	–	HIM
6	STP	GMSL1	–	Legacy
7	Coax	GMSL2	Pixel Mode	–

Link Initialization Registers

Table 7. Link Initialization Registers

REGISTER	BITFIELD NAME	BITS	DEFAULT VALUE	DECODE
0x0001	RX_RATE[1:0] (MAX96714 only)	1:0	0b10	01 = 3 Gbps 10 = 6 Gbps
0x0006	GMSL2_A	6	0b1 (MAX96714) Depends on CFG1 (MAX96714F and MAX96714R)	0 = GMSL1 mode 1 = GMSL2 mode
0x0011	CXTP_A	0	0b1	0 = Shielded twisted pair drive 1 = Coax drive
0x0010	RESET_ALL	7	0b0	0 = No action 1 = Activate chip reset (PWDNB)
0x0010	RESET_LINK	6	0b0	0 = Release link reset 1 = Activate link reset
0x0010	RESET_ONESHOT	5	0b0	0 = No action 1 = Activate one-shot reset on link (bit self clears)

Note: A link reset on CSI-2 deserializers resets the entire data path of any video connected to the reset PHY. Do not use link resets when video is being fed to the device. Doing this may corrupt data and have unintended consequences.

Link Initialization Programming Example

Turn on GMSL1 mode and set link to 3 Gbps GMSL link rate (for MAX96714).

```
# Turn on all links in GMSL2 mode.
0x98, 0x0006 ,0x50
# Set 6Gbps/187.5Mbps
0x98, 0x0001, 0x02
# Reset one-shot to reset the link
0x98, 0x010, 0x31
```

Link Lock Check

If the device configuration is correct, the link automatically locks upon connection. Pin 6 (MFP5) is used as LOCK indication by default. The following are the lock indication register bits:

- **GMSL1 Mode**
  - Register 0x0BCB bit 0 asserts if link is locked in the GMSL1 mode.
- **GMSL2 Mode**
  - Register 0x0013 bit 3 asserts if link is locked in the GMSL2 mode.

When pairing with GMSL2 serializers in the GMSL2 mode, links are automatically locked upon connection. The GMSL2 protocol uses a fixed link rate based on a constant-frequency link clock generated from a 25 MHz crystal. The link clock does not have any relationship with the input data pixel clock. In the GMSL1 mode, the link is not automatically established or locked. If there is no pixel clock present on the serializer side, the reverse control link must be established first (see the GMSL1 sections).

Video Pipe Selection

The video pipe must be configured to match the video streams received from the connected serializer. This programming step, typically performed following link initialization, ensures that the GMSL2 CSI-2 deserializer properly receives video data from the serializers. There is one video pipe (video pipe Y) that can select one of four streams from the serializer. By default, the deserializer is programmed to accept the most common stream from the CSI serializers (stream 0b10), and configuration is not usually needed.

Figure 4 shows the relationship between the GMSL1 and GMSL2 data and video pipe.

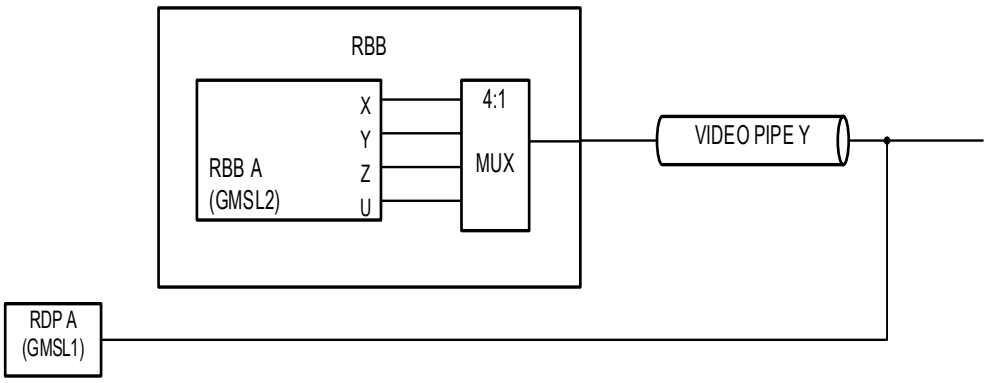

Figure 4. Video Pipe Selection Block Diagram

GMSL1 Mode Video Pipes

In the GMSL1 mode, video pipe selection is not required. Video pipes are enabled by default.

GMSL2 Mode Video Pipes

Select the serializer stream ID to match the video streams (STR_ID) from the serializers for each deserializer video pipe.

Most GMSL2 camera serializers have four video pipes (X, Y, Z, and U). These are annotated as 2 bits representing the stream ID. Pipe X = 0b00, Pipe Y = 0b01, Pipe Z = 0b10, and Pipe U = 0b11.

By default, the deserializer selects stream 0b10, which is the default for most CSI serializers.

Video Pipe Selection Registers (GMSL2 Mode)

Table 8. Video Pipe Selection Register

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0160	0	0x01	Video Pipe Enable Register: Bit 0: Enable video pipe 0 = Disable; 1 = Enable (default)
0x0161	2:0	0x02	**Video Pipe Select Register:** Bits [1:0]: Stream ID Selection for Pipe 2 Serializer Stream ID Selection: 00 = Stream 0 01 = Stream 1 10 = Stream 2 (Default) 11 = Stream 3

Video Pipe Selection Programming Examples (GMSL2 Mode)

This example sets up the deserializer video pipes Y to use stream 0. (e.g., an HDMI device).

```
# Enable pipe Y for stream ID 0
0x98,0x0161,0x00
# Turn on pipe Y (On by default)
0x98,0x0160,0x01
```

Video Lock Check

Register 0x01FC bit 0 (VIDEO_LOCK) asserts if it is receiving valid video data. VIDEO_LOCK in the GMSL1 mode is equivalent to link lock.

Video Pipe to MIPI Controller Mapping (VC/DT Mapping only)

Video pipe to mobile industry processor interface (MIPI) controller mapping is not configured beyond the default state due to the simple one-input-one-output nature of the deserializer. The MIPI controller mapping can also be used to change the VC mapping of individual MIPI data types.

Note: For simpler cases with only one desired virtual channel (VC) or data type, use the VC override features, instead of the controller mapper.

Method to Change the Mapping

Enable the number of mappings required (of a maximum possible 16) by writing the two map enable registers (MAP_EN_L and MAP_EN_H) for each video pipe to be configured. Then, specify the destination for Controller 1 (there is only one controller on this deserializer) for each enabled mapping block (i.e., the MIPI PHY controller destination). The data being mapped is designated by programming two register sets for the source virtual channel

(VC) and data type (DT) as well as the destination VC and DT. By setting the destination values different than the source, the VC and DT can be changed.

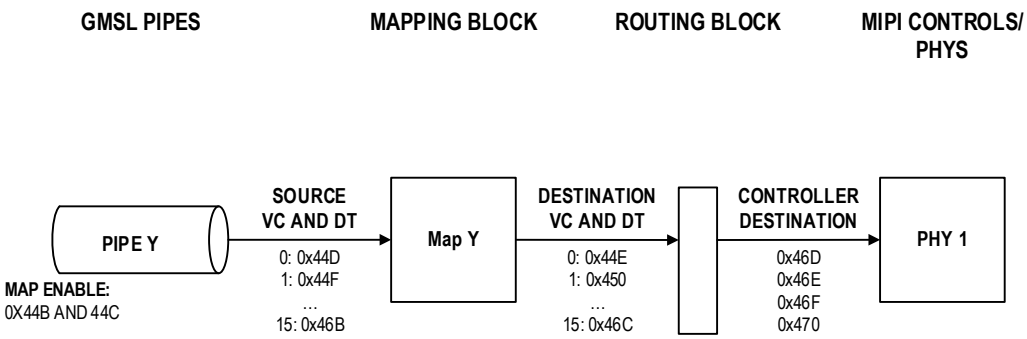

Figure 5. Video Pipe to MIPI Controller Mapping Block Diagram

Pipe to controller mapping register blocks in register block: 0x0440 to 0x047F.

Video Pipe to MIPI PHY Controller Registers

Table 9. Video Pipe Y to MIPI PHY Controller Register Table

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x044B	7:0	0x00	**Mapping Enable Low Byte Register:** 0000_0000 = No mapping enabled. XXXX_XXX1 = Map SRC_0 to DES_0. … 1XXX_XXXX = Map SRC_7 to DES_7 Each bit enables one of maximum 16 mapping and distribution entries for current video stream.
0x044C	7:0	0x00	**Mapping Enable High Byte Register:** 0000_0000 = No mapping enabled. XXXX_XXX1 = Map SRC_8 to DES_8. … 1XXX_XXXX = Map SRC_15 to DES_15 Each bit enables one of maximum 16 mapping and distribution entries for current video stream.
0x044D	7:0	0x00	**MAP_SRC_0 Register:** Bits [7:6]: VC Bits [5:0]: DT
0x044E	7:0	0x00	**MAP_DES_0 Register:** Bits [7:6]: VC Bits [5:0]: DT
…	…	…	…
0x046B	7:0	0x00	**MAP_SRC_15 Register:** Bits [7:6]: VC Bits [5:0]: DT
0x046C	7:0	0x00	**MAP_DST_15 Register:** Bits [7:6]: VC Bits [5:0]: DT
0x046D	7:0	0x00	**MAP Destination Controller Register:** Bits [7:6]: Destination Controller for Map 3

			Bits [5:4]: Destination Controller for Map 2 Bits [3:2]: Destination Controller for Map 1 Bits [1:0]: Destination Controller for Map 0 00 = Do not use 01 = Map to Controller 1 10 = Do not use 11 = Do not use
0x046E	7:0	0x00	**MAP Destination Controller Register:** Bits [7:6]: Destination Controller for Map 7 Bits [5:4]: Destination Controller for Map 6 Bits [3:2]: Destination Controller for Map 5 Bits [1:0]: Destination Controller for Map 4 00 = Do not use 01 = Map to Controller 1 10 = Do not use 11 = Do not use
0x046F	7:0	0x00	**MAP Destination Controller Register:** Bits [7:6]: Destination Controller for Map 11 Bits [5:4]: Destination Controller for Map 10 Bits [3:2]: Destination Controller for Map 9 Bits [1:0]: Destination Controller for Map 8 00 = Do not use 01 = Map to Controller 1 10 = Do not use 11 = Do not use
0x0470	7:0	0x00	**MAP Destination Controller Register:** Bits [7:6]: Destination Controller for Map 15 Bits [5:4]: Destination Controller for Map 14 Bits [3:2]: Destination Controller for Map 13 Bits [1:0]: Destination Controller for Map 12 00 = Do not use 01 = Map to Controller 1 10 = Do not use 11 = Do not use

Video Pipe to MIPI Controller Mapping Programming Examples

Pipe to Controller Mapping

In the deserializer, the RAW12 data is assigned VC1 while RGB888 is assigned VC0.

```
# Enable mapping for pipe Y, 4 mappings enabled for FS, FE, RAW12 and RGB888 data
type
0x98, 0x44B, 0x0F
# Map_SRC_0 - FS, VC = 0
0x98, 0x44D, 0x00
# Map_DST_0 - FS, VC = 0
0x98, 0x44E, 0x00
# Map_SRC_1 - Raw12, VC = 0
0x98, 0x44F, 0x2C
# Map_DST_1 - Raw12, VC = 1
0x98, 0x450, 0x6C
# Map_SRC_2 - FE, VC = 0
0x98, 0x451, 0x01
```

```
# Map_DST_2 - FE, VC = 0
0x98, 0x452, 0x01
# Map_SRC_3 – RGB888, VC = 0
0x98, 0x453, 0x24
# Map_DST_3 – RGB888, VC = 0
0x98, 0x454, 0x24
# Map_D-PHY_DST Sets PHY destination to PHY1 for mappings 0-3
0x98, 0x46D, 0x55
```

MIPI PHY Settings

The MIPI PHY settings contain programming options for output data rate, number of lanes, and port selection. The MAX96714 deserializers have two 2-lane MIPI PHYs (PHY 0, 1) controlled by a single MIPI controller (Ctrl 1) to establish a 4-lane port. *Table 10* contains the MIPI PHY settings registers.

The 1x1, 1x2, 1x3, and 1x4 configurations use lane count settings. See *Table 10* for register write details.

MIPI PHY Settings Registers

Table 10. MIPI PHY Settings Register Table

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0330	7	0x00	**MIPI PHY Mode Select Register:** Bit 7: Set to force all MIPI clocks running. Used with the MAX96714 internal pattern generator.
0x0332	7:4	0xF4	**MIPI PHY Enable Register:** Bit 5: Enable MIPI PHY 1 0 = PHY in standby 1 = PHY enabled.
0x0333	7:0	0xE4	**MIPI PHY 0 and 1 Lane Mapping Register:** Bits [7:6]: Set PHY 1 Data Lane 1 Bits [5:4]: Set PHY 1 Data Lane 0 Bits [3:2]: Set PHY 0 Data Lane 1 Bits [1:0]: Set PHY 0 Data Lane 0 00 = Map D0. 01 = Map D1. 10 = Map D2. 11 = Map D3.
0x0335	5:0	0x00	**MIPI PHY 0 and 1 Lane Polarity Register:** Bit 5: Set Polarity on PHY 1 CLK lane Bit 4: Set Polarity on PHY 1 D1 lane Bit 3: Set Polarity on PHY 1 D0 lane Bit 2: Set Polarity on PHY 0 CLK lane Bit 1: Set Polarity on PHY 0 D1 lane Bit 0: Set Polarity on PHY 0 D0 lane 0 = Normal polarity, 1 = Inversed polarity.
0x033F	1	0x00	**MIPI Controller Reset Register:** Bits 1: Resets for 1 0: Do not reset 1: Reset the MIPI Controller
0x044A	7:6	11	**MIPI PHY 1 Lane Count / C-PHY enable Register:** Bits [7:6]: Set lane count. 00 = 1 data lane 01 = 2 data lane

			10 = 3 data lane 11 = 4 data lane
0x0320	4:0	01111	**MIPI PHY 1 DPLL Freq Register:** Set DPLL frequency on multiples of 100 MHz. Clock frequency is half; data rate is equivalent bps/lane. 00001 = 100 MHz DPLL, 100 Mbps/lane data rate. … 11001 = 2500 MHz DPLL, 2.5 Gbps/lane data rate.
0x1D00	0	1	**Controller 1 DPLL Register:** Set DPLL block into reset mode for controller 1 0 = DPLL block in reset 1 = DPLL block in normal mode (default)

MIPI Data Lane Swap

Lane swapping is available for pins on the same port. See *Table 10* for relevant registers.

The data pins can be swapped within each port, but the clock location is fixed. For example, in the 2 x 4 mode, the default mappings of the D0, D1, D2, and D3 pairs can be swapped to different output pins. Additionally, the polarity of each output data pairs, and the clock lane supports polarity inversion. *Figure 6* shows the lane swap required to match the device pinout. *Figure 7* demonstrates the polarity swap.

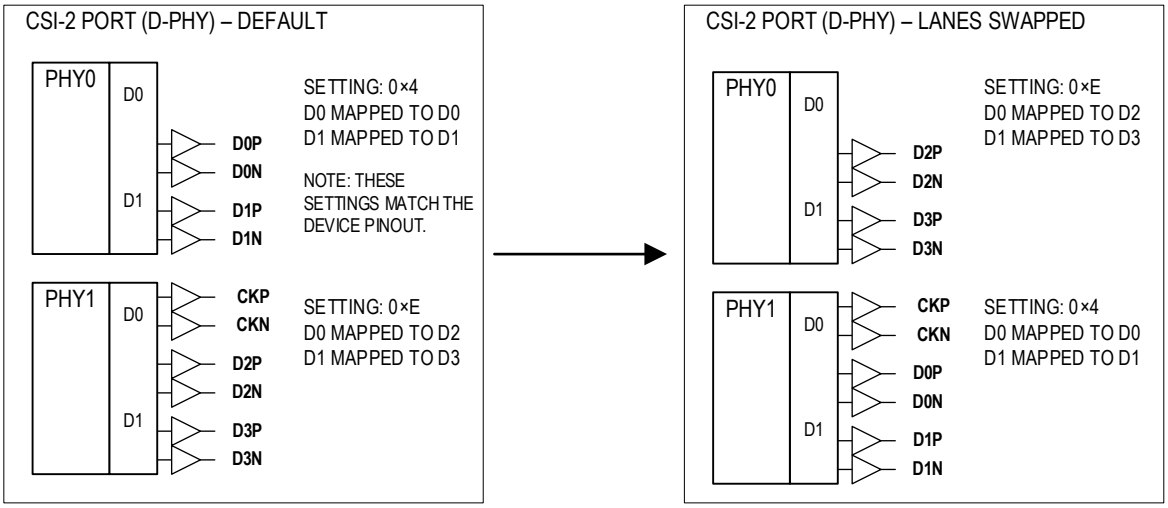

Figure 6. D-PHY Lane Swap Example

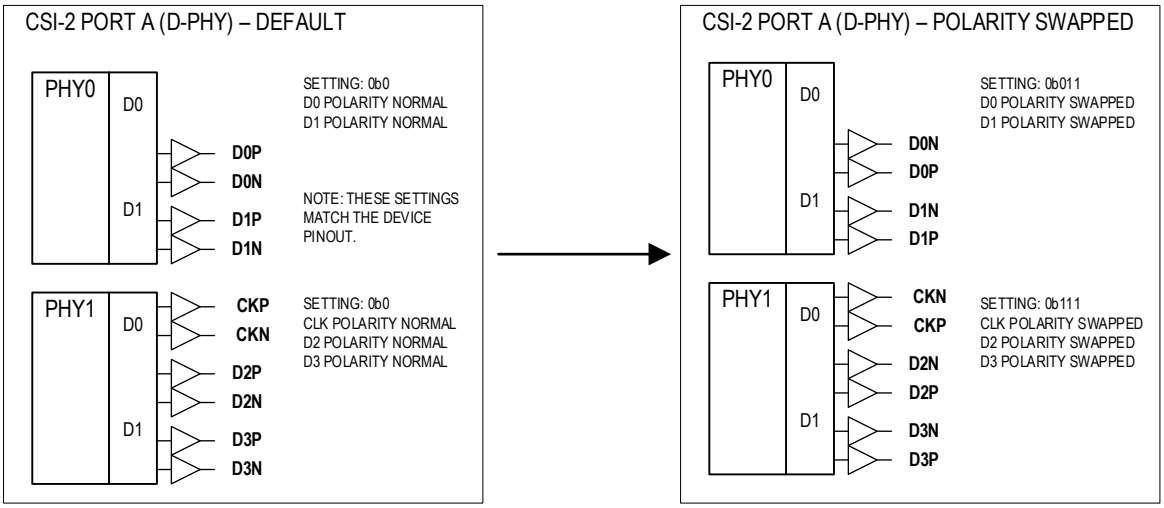

Figure 7. D-PHY Polarity Swap Example

Lane Swap Programming Example

This example programs the MIPI PHY mode, configures the lane mappings, and sets the lane rates.

```
# Set lane mapping for ctrl 1 (MIPI Port A in 1x4 mode). This is written to
completely swap the device pinout from default.
0x98,0x0333,0x4E
# Set 4 lanes for ctrl 1 (MIPI Port A in 1x4 mode)
0x98,0x094A,0xC0
# Set MIPI lane rate to be 2Gbps/lane for ctrl 1 (MIPI Port A in 1x4 mode)
0x98,0x0320,0x34
```

MIPI D-PHY Deskew Settings

In compliance with the MIPI D-PHY v1.2 specification, the deserializers' MIPI D-PHY ports support the mandatory deskew calibration upon initialization. Additionally, they support optional periodic deskew as described by the MIPI alliance. The D-PHY deskew mechanism is only relevant to lane speeds greater than 1.5 Gbps per lane. Periodic deskew requires a continuous clock and the clock lane always in the high-speed mode.

The initial transmit deskew is set with DESKEW_INIT[7:0] in register 0x0443. All settings must be configured before video streams are received. After video lock, the MIPI Tx clock lane starts automatically and an automatic deskew pattern is generated before the first HS data transmission. For system flexibility, an additional initial deskew pattern can be inserted by changing DESKEW_INIT bit 5 any time after the clock lane is enabled. Note that DESKEW_INIT bit 4 must be set to trigger a manual deskew.

The periodic deskew is initiated by setting bit 7 of DESKEW_PER[7:0] in register 0x0444. The period interval has a programmable range from every 1 to 128 frames. *Table 11* contains the relevant MIPI D-PHY deskew registers.

MIPI D-PHY Deskew Registers

Table 11. MIPI D-PHY Deskew Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0443	7:0	0x00	**MIPI TX 1 DESKEW_INIT Register:** Bit 7: Auto initial deskew on/off Bit 6: Reserved Bit 5: When bit 4 = 1, any change of this bit triggers one-time immediate initial skew. Bit 4: Manual initial on/off Bit 3: Reserved Bits [2:0]: Initial deskew width 1, 2, … 8 x (32 K) UI
0x0444	7:0	0x00	**MIPI TX 1 DESKEW_PER Register:** Bit 7: Period deskew calibration on/off Bit 6: Select generation on rising or falling edge of VS Bit [5:3]: Select periodic interval at every: 1, 2, 4, 8, … 128 frames Bit [2:0]: Select periodic deskew width: 1, 2, 3, … 8 x (1 K) UI

MIPI D-PHY Deskew Programming Example

The following programming example shows a typical use of initial and periodic deskew calibration.

```
# Enable initial deskew packets with the minimum width 32K UI on controller 1 for
port A.
0x98,0x0443,0x80
# Enable periodic deskew packets with width 2K UI every 2 frames.
0x98,0x0444,0x91
```

Extended Virtual Channels

GMSL2 deserializers can use CSI-2 extended virtual channels to accommodate larger serial link systems. Virtual channels allow the serial link system to differentiate video inputs by the virtual channel assigned to them. When extended virtual channels are enabled, the standard 2-bit virtual channel selection is extended to 4-bit (D-PHY), increasing the number of available virtual channels to 16 for D-PHY applications. The increase in available virtual channels allows systems to support more camera inputs.

Extended virtual channels are not enabled by default. See *Table 14* for extended virtual channels register details.

Note: The two virtual channel source-to-destination mapping bits referenced in are used with the extended virtual channel configuration. However, these bits become the least significant bits (LSB) of the virtual channel bit selection.

If the input serializer is C-PHY, do not use VCs 16 to 31.

Extended Virtual Channels Register Examples

The following example demonstrates the MSB and LSB of the extended virtual channel for pipe 0, mapping 0. Here, the source is changed from VC = 0 to destination VC = 5, VC = 4, or VC = 2.

Table 12. Source Mapping (Extended VC)

EXTENDED VC SOURCE					VC SOURCE
Register	0x0510 [7:5]				0x044E [7:6]
MIPI	Must be '0'	D-PHY			
Bit Place	4	3	2	1	0
VC = 0	0	0	0	0	0

Table 13. Destination Mapping (Extended VC)

EXTENDED VC DESTINATION					VC DESTINATION
Register	**0x0510 [4:2]**				**0x044E [7:6]**
MIPI	Must be '0'	D-PHY			
Bit Place	4	3	2	1	0
VC = 5	0	0	1	0	1
VC = 4	0	0	1	0	0
VC = 2	0	0	0	1	0

Extended Virtual Channel Registers

Table 14. Extended Virtual Channels Register Table

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x044A	4	0	**CSI2_vcx_en[?]:** 0 = VC extension disabled 1 = VC extension enabled Controller 0,1,2, and 3 virtual channels enable.
0x510	7:2	0x00	**Pipe Y, Extended VC MAP_SRC/DST_0 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel
…	…	…	**…**
0x51F	7:2	0x00	**Pipe Y, Extended VC MAP_SRC/DST_15 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel

Extended Virtual Channels Programming Example

This example assigns the extended virtual channel (VC = 4).

```
# Setup Source to Destination mappings
# Turn on controller 1 csi2_vcx_en
0x98, 0x44A, 0xD8
# Enable pipe Y, mapping 0-3.
0x98, 0x44B, 0x07

# Mapping 0
# Source, VC = 0 and DT = FS
0x98, 0x44D, 0x00
# Destination, VC = 0 and DT = FS, The 2 - VC bits here are the LSB part of the VC
bits for the VC extension.
0x98, 0x44E, 0x00

# Mapping 1
# Source, VC = 0 and DT = Raw12
```

```
0x98, 0x44F, 0x2C
# Destination, VC = 0 and DT = Raw12, The 2 - VC bits here are the LSB part of the
VC bits for the VC extension.
0x98, 0x450, 0x2C

# Mapping 2
# Source, VC = 0 and DT = FE
0x98, 0x451, 0x01
# Destination, VC = 0 and DT = FE, The 2 - VC bits here are the LSB part of the VC
bits for the VC extension.
0x98, 0x452, 0x01

# D-PHY Destination is set to controller 1 for mapping 0-3
0x98, 0x46D, 0x15

# Setup for Extended virtual channels
# Set the 3 MSB for the destination mapping. This is the MSB for mapping 0.
0x98, 0x510, 0x04
# Set the 3 MSB for the destination mapping. This is the MSB for mapping 1.
0x98, 0x511, 0x04
# Set the 3 MSB for the destination mapping. This is the MSB for mapping 2.
0x4E, 0x512, 0x04
```

Software Override

The software override manually overrides the video data type (DT) (i.e., packet header), virtual channel (VC) number, or bits per pixel (bpp). This operation affects the video data between the video pipe(s) and the MIPI controller(s). Overriding the DT and VC information is used for MIPI controller mapping. If the received video data is from a serializer in parallel mode (e.g., GMSL1 serializers), it is necessary to specify the desired DT, VC, and bpp with the software override. See *Table 15* for a list of software override registers.

Note: VC can be changed individually. However, DT and bpp must be adjusted together to ensure settings compatibility.

- DT: soft_dt_y[5:0] E.g., RGB888: DT = 0x24 = 0b100100
- VC: soft_vc_y[3:0] E.g., VC: 3 = 0x03 = 0b0011
- bpp: soft_bpp_y[4:0] E.g., RAW12: bpp = 0xC = 0b01100

The data type, virtual channel, and bpp overrides must be enabled to take effect.

Enable override_bpp_vc_dt_y bit for video streams that require the override of the VC, DT, and bpp.

Software Override Registers

Table 15. Software Override Register Table

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0314	7:4	0x00	**Pipe Y VC Software Override Register:** Bits [7:4]: Pipe Y VC.
0x0316	7:6	0x00	**Pipe Y (H) DT Software Override Register:** Bits [7:6]: Pipe Y DT (High Bits [5:4]).
0x0317	3:0	0x00	**Pipe Y (L) DT Software Override Register:** Bits [3:0]: Pipe Y DT (Low Bits [3:0]).
0x0319	4:0	0x00	**Pipe Y BPP Software Override Register:** Bits [4:0]: Pipe Y BPP.
0x031D	7	0x00	**Pipe Y Software Override Enable:** Bit 7 0 = Override disable (default) 0 = Override enable

Note: Bits are not aligned consistently. E.g., 0x2C is stored in registers as 0x80 and 0x0C.

Software Override Programming Example

The following script performs a series of VC, DT, and bpp software overrides on video pipes 0, 1, 2, and 3.

```
# VC/DT/BPP Software override for Pipe Y.
# VC 0 override for pipe Y
0x98,0x0314,0x00
# RAW12 (0x2C) DT override for pipe Y
0x98,0x0316,0x80
0x98,0x0317,0x0C
# 12 BPP override for pipe Y
0x98,0x0319,0x0C
# Enable the software override for Y
0x98,0x031D,0x80
```

I²C Control Channels

Overview

The deserializer provides three I²C control channels (primary port, and port 1 or port 2). The primary I²C port can access the deserializer, connected serializer, and remote peripherals connected to the serializer's control channel port. The secondary ports, port 1 or port 2, can access the remote-side devices connected through the corresponding pass-through ports but cannot access the serializer or deserializer registers.

Note: Only one pass-through port is intended to be on in a system. Peripheral connections on the serializer side determine if pass-through 1 or 2 is used. The pass-through port and primary port can be accessed simultaneously, as each side is independent within the deserializer.

Primary I²C Control Channel

The Primary I²C control channel provides access to both the serializer and deserializer registers across the GMSL link. This provides flexibility where the registers for both the serializer and deserializer are accessible from whichever side the main microcontroller resides (for MAX96714 applications, the main microcontroller typically resides on the deserializer side).

I²C Pass-Through Channel

Note: The MAX96714R variant features only the primary I²C control channel and does not include the I²C pass-through channel access.

There are two pass-through I²C channels to send I²C data across the GMSL link. These channels prevent multicontroller conflict. Thus, register access of the serializer/deserializer is not possible on the pass-through I²C channels. Pass-through channels share the same set of pins. So, only one passthrough channel can be active at a time.

Port Access and Routing

The multifunction pins (MFPs) shown in *Table 16* are used for the I²C control and pass-through channels. This table does not apply to the MAX96714R variant.

Table 16. Multifunction Pins (MFPs) for I²C

MFP	I²C /UART FUNCTION	DEFAULT FUNCTION	NOTES
MFP1*	SDA1/RX1/SDA2/RX2	RX1	UART pass-through 1 on by default.
MFP2*	SCL1/TX1/SCL2/TX2	TX1	UART pass-through 1 on by default.
MFP3	SDA	SDA_RX	SDA (I²C) or RX (UART) functionality determined by CFG0 status on power-up.
MFP4	SCL	SCL_TX	SCL (I²C) or TX (UART) functionality determined by CFG0 status on power-up.

* Not supported in the MAX96714R. These MFPs only operate as GPIOs.

On power-up, the device should be set to the I²C mode through the CFG0 latch. The function names in *Table 16* and ensuing I²C sections assume the device is configured for the I²C mode.

By default, the primary I²C control channel lines are brought out on MFP3 and MFP4 for SDA and SCL, respectively. Disable the primary control channel's line access by setting field DIS_LOCAL_CC in register 0x1. Also disable access to the remote device control by setting field DIS_REM_CC in register 0x1. Disabling the remote communication channel is a one-sided command, meaning that if a microcontroller is located on the serializer side, it may encounter I²C timeouts if it sends commands to a disabled control channel.

Note: A minimum 10 µs delay is required after enabling/disabling the I²C functionality through the DIS_LOCAL_CC and DIS_REM_CC fields in register 0x1.

Bring out the pass-through I²C channels on MFP1 and MFP2 for SDA1/SDA2 and SCL1/SCL2, respectively. As mentioned above, it is possible to use only one pass-through channel at a time. Pass-through channels are enabled by setting the fields IIC_1_EN and IIC_2_EN in register 0x1.

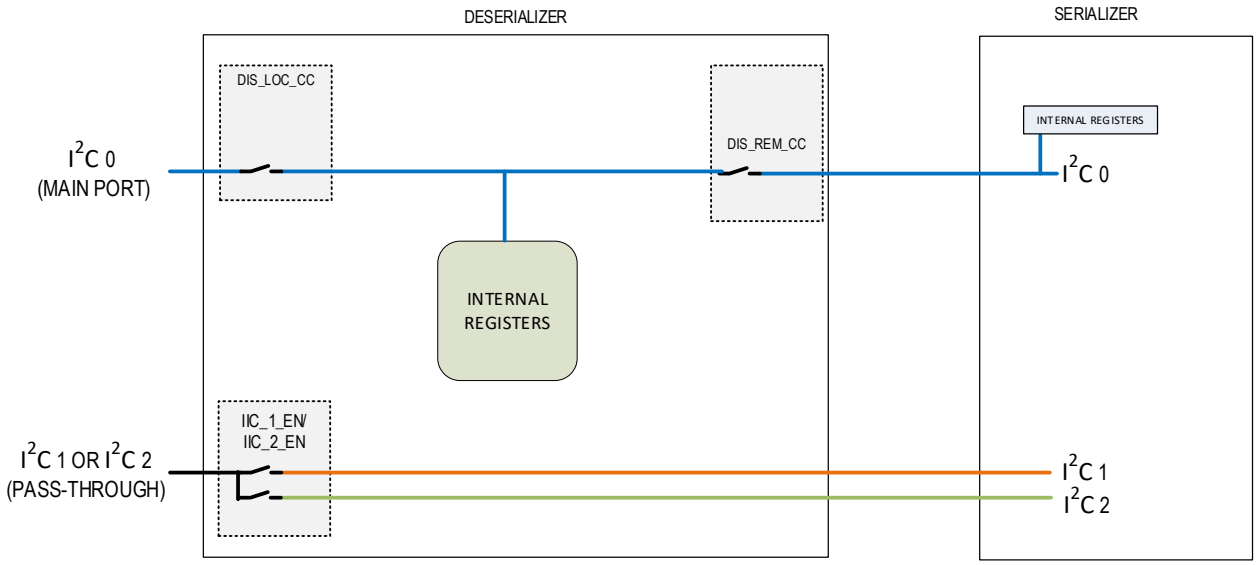

Figure 8. I²C Port Access and Routing

I²C for CRC + Message Counter

Note: This section does not apply to the MAX96714R variant.

The MAX96714 provides two additional functional safety features by adding an optional CRC packet to either side of the link and an optional message counter to monitor both write and read transactions. Both features are available only to the deserializer's primary I²C control channel for register read/write transactions. They are disabled by default and must be enabled at the register level.

CRC over I²C

The CRC feature detects corrupt data written on the deserializer's primary I²C control channel. Each I²C transaction has a corresponding CRC packet associated with it. If a CRC error is detected, the command is not executed, and the transaction is reported as NACK. The CRC feature is enabled by setting fields CC_CRC_EN and CC_CRC_MSGCNTR_OVR in register 0x300F.

Message Counter over I²C

The message counter feature detects missing or repeated I²C transactions. For every transaction, the message counter is incremented accordingly while a copy of the counter is maintained on both ends of the I²C link. If a mismatch between copies is found, the transaction is rejected. The message counter feature is enabled by setting fields CC_MSGCNTR_EN and CC_CRC_MSGCNTR_OVR in register 0x300F.

I²C Registers

Table 17. MAX96714/MAX96714F I²C Registers (Not Applicable to MAX96714R)

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0001	7:4	0x08	**I²C Enable Register:** Bit [7]: Enable pass-through I²C Control Channel 2 (SDA2, SCL2) Bit [6]: Enable pass-through I²C Control Channel 1 (SDA1, SCL1) Bit [5]: Disable primary I²C control channel connection to SDA and SCL pins Bit [4]: Disable access to remote device control channel over GMSL2 connection
0x0006	4	0x80	**I²C Selection Register:** Bit [4]: Enables I²C when set to a 1 or UART when set to a 0. **Note: This bit is set according to the CFG0 pin value on power-up. Writing to this register is not recommended.**
0x3000	3	0x00	**Enable CRC Computation Register:** Bit [3]: Compute register CRC after every I²C register write
0x3008	0	0x00	**Message Counter Reset Register:** Bit [0]: Reset Message Counter value to 0
0x3009	1:0	0x00	**CRC Reset Register:** Bit [1]: Resets Message Counter error count to 0 Bit [0]: Resets CRC error count to 0
0x300A	7:0	0x00	**Read CRC Value Register:** Bits [7:0]: CRC value for the last write transaction
0x300B	7:0	0x00	**Read Message Counter Low Bits Register:** Bits [7:0]: Low bits of current message counter value
0x300C	7:0	0x00	**Read Message Counter High Bits Register:** Bits [7:0]: High bits of current message counter value
0x300D	7:0	0x00	CRC Error Count Register: Bits [7:0]: Number of I2C/UART CRC errors detected
0x300E	7:0	0x00	**Message Counter Error Count Register:** Bits [7:0]: Number of message counter errors detected
0x300F	2:0	0x06	**Enable CRC and Message Counter Register:** Bit [2]: Enable I²C message counter. **Note: Only active when Bit [2] is also set to 1.**

			Bit [1]: Enable I^{2}C CRC packeting. **Note: Only active when Bit [2] is also set to 1.** Bit [0]: Enable manual override of I^{2}C CRC or message counter configuration. If set to a 0, then CRC and message counter features are disabled.

Control Channel Programming Example

This example enables pass-through I^{2}C channel 1 (normally disabled by default).

```
# Disable UART pass-through Control Channel 1
0x98,0x0003,0x40
# Enable pass-through I2C Control Channel 1
0x98,0x0001,0x42
```

Enable CRC and Message Counter on Primary I^{2}C Control Channel

This example enables the CRC and message counter features on the primary I^{2}C control channel.

```
# Enables CRC and Message Counter features along with corresponding override enable
0x98,0x300F,0x07
```

I²C Broadcasting

Overview

When transmitting to a multiple-link input deserializer or multiple deserializers, each device on the serializer side requires a unique address for individual programming and identification. Through I²C translation and address reassignment, each serializer and image sensor can have both a unique address and a broadcasting address. This allows for selective programming of each device and the ability to broadcast commands to all devices at the same time. When broadcasting, if any remote GMSL I²C port ACKs the packet, it ACKs for all remote GMSL I²C ports.

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

An example of I²C broadcasting is discussed in the ensuing section. Two equivalent camera modules, including an image sensor and GMSL2 serializer with the same respective addresses, are connected to two GMSL2 deserializers with different device addresses. Each of the camera modules comprise a serializer at the default I²C address 0x80 and an image sensor at address 0x20.

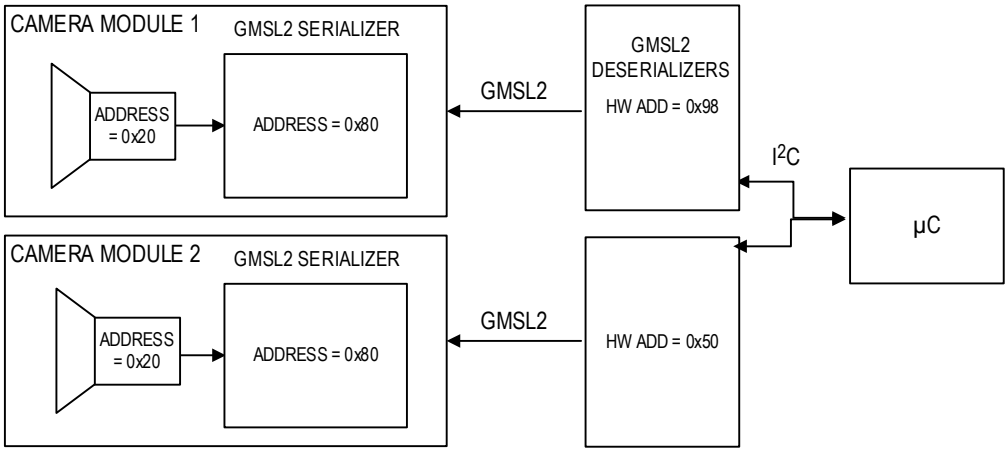

Figure 9. I²C Interfaced Camera-Module System with Default Address Settings

I²C Broadcasting Technique

The I²C broadcasting technique helps to communicate with multiple camera-serializer modules with a single microcontroller, which in-turn streamlines the transmission process.

The general procedure is to:

- Isolate a single camera/serializer module for remote I²C access, meaning no other device with the same address should be connected to the I²C data line.
- Change the serializer address to a unique address.
- Modify the first I²C address translation register with a common source address but the unique destination address. This is to streamline the interface with the serializer.

- Modify the second I^2C address translation register with a unique source address but the default image sensor addresses for the destination address. This is to streamline the interface with the image sensor.
- Repeat this process for each camera serializer module.
- When making changes to any of the serializer or deserializer's I^2C configuration, such as enabling or disabling an I^2C port, at least a 10 µs delay from the write acknowledgement (ACK) to the next transaction is required.

I^2C Broadcasting GMSL2 Use Case Example

The procedure for the I^2C broadcasting example is as follows.

1) Isolate camera module 1 by disabling camera module 2's GMSL link (RESET_LINK = 1).
2) Change the serializer device address in camera module 1 from 0x80 to 0x82. This is done with a register write to DEV_ADDR[6:0] located in REG0.
3) Modify the first address translation register in this serializer to give a broadcast address (0xC4) to the serializer. Program 0xC4 into the source register SRC_A[6:0], and 0x82 in the destination register DST_A[6:0]. Thus, for the serializer in camera module 1, anything sent to address 0xC4 is sent to address 0x82 instead.
4) Modify the second translation register in this serializer to give a unique address to the image sensor. Program 0x22 into the source register SRC_B[6:0] and 0x20 into the destination register DST_B[6:0]. Thus, for the serializer in camera module 1, anything sent to address 0x22 is sent to address 0x20 instead.
5) Isolate camera module 2 by disabling camera module 1's GMSL link (RESET_LINK = 1) and enabling camera 2's GMSL link (RESET_LINK = 0).
6) Change the serializer device address in camera module 2 from 0x80 to 0x84. This is done with a register write to DEV_ADDR[6:0] located in REG0.
7) Modify the first address translation register in this serializer to give a broadcast address (0xC4) to the serializer. Program 0xC4 into the source register SRC_A[6:0] and 0x84 in the destination register DST_A[6:0]. Thus, for the serializer in camera module 2, anything sent to address 0xC4 is sent to address 0x84 instead.
8) Modify the second translation register in this serializer to give a unique address to the image sensor. Program 0x24 into the source register SRC_B[6:0] and 0x20 into the destination register DST_B[6:0]. Thus, for the serializer in camera module 2, anything sent to address 0x24 is sent to address 0x20 instead.
9) Now enable all the links for remote primary I^2C port access.
10) All devices should be present on the I^2C bus. Continue with any additional required system configuration.

Figure 10 shows the same camera module system with translated addresses. *Table 18* and *Table 19* also summarize the changes made.

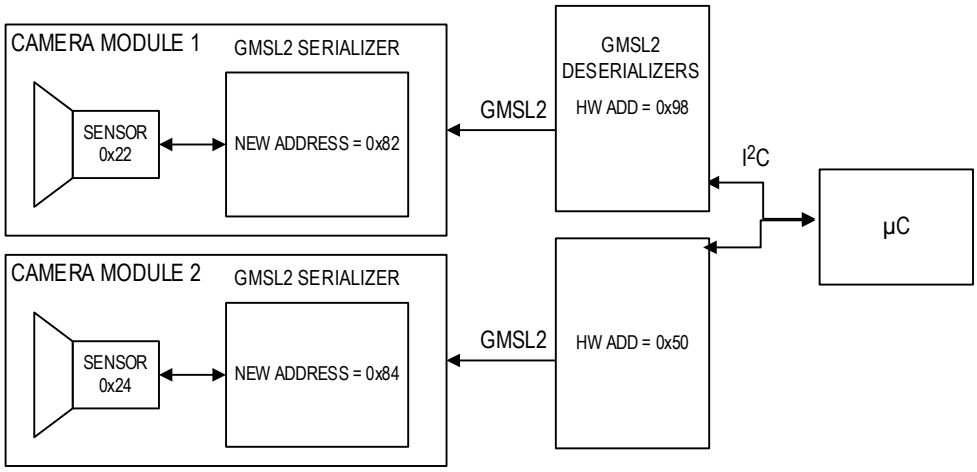

Figure 10. Camera-Module System with Translated Address Settings

The serializers are assigned a single device address to allow writes to all devices as a broadcast.

Table 18. I²C Broadcasting Example (Serializer)

I²C ADDRESS	SRC_A	DST_A	SINK DEVICE(S)
0x82	0xC4	0x82	Serializer in Camera Module 1
0x84	0xC4	0x84	Serializer in Camera Module 2

Each image sensor is assigned a unique device address.

Table 19. I²C Broadcasting Example (Image Sensor)

I²C ADDRESS	SRC_B	DST_B	SINK DEVICE(S)
0x20	0x22	0x20	Image Sensor in Camera Module 1
0x20	0x24	0x20	Image Sensor in Camera Module 2

I²C Broadcasting Programming Examples

This script sets up the I²C broadcasting as shown in *Figure 10*.

```
# Enable Link B remote control channel only
0x50,0x0010,0x51
# Change I2C address for this Link A serializer
0x80,0x0000,0x82
# Set Ser source to 0xC4
0x82,0x0042,0xC4
# Set Ser destination to 0x82
0x82,0x0043,0x82
# Set Image sensor source to 0x22
0x82,0x0044,0x22
# Set Image sensor destination to 0x20
0x82,0x0045,0x20
```

```
# Enable Link B remote control channel
0x50,0x0010,0x31
# Disable Link A remote control channel
0x98,0x0010,0x51
# Change I2C address for this Link B serializer
0x80,0x0000,0x84
# Set Ser source to 0xC4
0x84,0x0042,0xC4
# Set Ser destination to 0x84
0x84,0x0043,0x84
# Set Image sensor source to 0x24
0x84,0x0044,0x24
# Set Image sensor destination to 0x20
0x84,0x0045,0x20
# Enable Link A remote control channel
0x98,0x0010,0x31
```

UART Control Channels

Overview

The MAX96714 and MAX96714F feature one primary UART control channel and two pass-through UART channels. Due to MFP limitations, it is possible to use at most one of the two pass-through channels in one application.

Note: The MAX96714R variant does not feature UART capability and is not applicable to the UART control channels section.

When making changes to any of the serializer or deserializer's UART configuration, such as enabling or disabling a UART port, at least a 10 µs delay from the write acknowledgement (ACK) to the next transaction is required.

Primary UART Control Channel

The Primary UART control channel provides access to both the 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 MAX96714 applications, the main microcontroller usually resides on the deserializer side).

Base Mode

The base mode allows the host microcontroller to access the device registers of both the serializer and deserializer. It is the default mode for the primary UART control channel on power-up.

Bypass Mode

In the 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 GMSL2 link. This mode prevents inadvertent programming of the serializer/deserializer registers and can be switched in and out of during normal operation.

Pass-Through UART Channel

There are two pass-through UART channels to send data across the GMSL2 link. Serializer/deserializer registers are not accessible in this mode but any other peripherals on the link with compatible UART protocol are accessible. This makes either of the pass-through UART channels equivalent to running the primary UART control channel in the bypass mode as described above.

Port Access and Routing

The multifunction pins (MFPs) shown in *Table 20* are used for the UART control and pass-through channels.

Table 20. Multifunction Pins (MFPs) for UART

MFP PIN	PRIMARY UART FUNCTION	DEFAULT FUNCTION	NOTES
MFP0	MS	GPI0	Mode select (MS) is used as a hard trigger to run the primary UART channel (TX/RX) in the bypass mode.

MFP1	RX1/RX2	RX1	Register 0x0003 determines which UART port is active.
MFP2	TX1/TX2	RX1	Register 0x0003 determines which UART port is active.
MFP3	RX	SDA_RX	SDA (I^2C) or RX (UART) functionality determined by the CFG0 status on power-up.
MFP4	TX	SCL_TX	SCL (I^2C) or TX (UART) functionality determined by the CFG0 status on power-up.

On power-up, the device should be set to the UART mode through the CFG0 latch. The function names in *Table 20* and ensuing UART sections assume the device is configured for the UART mode.

By default, the Primary UART control channel lines are brought out on MFP3 and MFP4 for RX and TX, respectively. Disable the primary control channel's line access by setting field DIS_LOCAL_CC in register 0x1. Also, disable access to the remote device control by setting field DIS_REM_CC in register 0x1.

Enabling the UART Bypass Mode Through Register Setting (Soft-Bypass)

Enable the UART bypass mode through register setting by first setting field BYPASS_EN in register 0x48. Next, configure a timeout (2 ms, 8 ms, 32 ms, or no-timeout) by setting the field BYPASS_TO in register 0x48. The bypass mode is active only if there is UART activity. When there are no UART transitions detected for the selected timeout duration, the device exits the bypass mode and re-enters the base mode. The timeout is optional. If field BYPASS_TO is set for no timeout, the device remains in the bypass mode until the next power-cycle.

Enabling the UART Bypass Mode Through Pin Setting (Hard-Bypass)

Also enable the UART bypass mode by the mode select (MS) pin, which can be brought out on MFP8. In this state, a high-voltage level on the MS pin enables the bypass mode while a low voltage level disables the bypass mode. To enable this setting, set field REM_MS_EN in register 0x48.

Additionally, the MS pin can be set to use the GPIO2 pin instead of the function MS on MFP0. To enable this setting, set the field LOC_MS_EN in register 0x48. This setting might be needed if MFP0 is needed for another use.

Enabling the UART Pass-Through Channels

Bring out the first pass-through UART channel on MFP1 and MFP2 for RX1/RX2 and TX1/TX2, respectively. As mentioned above, it is possible to use only one pass-through channel at a time. Both pass-through channels are enabled by setting the fields UART_1_EN and UART_2_EN in register 0x3.

UART for CRC + Message Counter

The MAX96714 provides two additional functional safety features by adding an optional CRC packet to either side of the link and an optional message counter to monitor both the write and read transactions. Both features are available only to the serializer's primary UART control channel (not the pass-through channels). They are disabled by default and must be enabled at the register level.

CRC over UART

The CRC feature detects corrupt data written on the UART control channel. Each UART transaction has a corresponding CRC packet associated with it. The host microcontroller must compute and send a CRC byte after each data byte.

- If the host microcontroller is writing to the serializer registers, the serializer receives the data byte, calculates the CRC using an identical CRC engine, and verifies a match before accepting the data byte. If a mismatch is detected, then the write is not accepted, and the error counter is incremented.
- If the host microcontroller is reading the serializer registers, then the serializer calculates the CRC byte and appends it to the output data stream. The host microcontroller's CRC engine should then calculate its own CRC byte and compare it with the one received from the serializer to determine if there is a mismatch.

The CRC feature is enabled by setting fields CC_CRC_EN and CC_CRC_MSGCNTR_OVR in register 0x4.

Message Counter over UART

The message counter feature detects missing or repeated UART transactions. For every transaction, the message counter is incremented accordingly while a copy of the counter is maintained on both ends of the UART transaction. If a mismatch between copies is found, then the transaction is rejected. The message counter feature is enabled by setting fields CC_MSGCNTR_EN and CC_CRC_MSGCNTR_OVR in register 0x4.

Table 21. MAX96714/MAX96714F UART Registers (Not Applicable to MAX96714R)

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0001	5:4	0x08	**UART Control Channel Enable Register:** Bit [5]: Disable primary UART control channel connection to TX/RX pins Bit [4]: Disable access to remote device control-channel over GMSL2 connection
0x0003	6:4	0x00	**UART Pass-Through Channel Enable Register:** Bit [6[: Swap I2C/UART pass-through device pin assignments. Bit [5]: Enable pass-through UART Channel 2 Bit [4]: Enable pass-through UART Channel 1
0x0006	4	0x80	**UART Selection Register:** Bit [4]: Enables UART when set to a 0. **Note: This bit is set according to the CFG0 pin value on power-up. Writing to this register is not recommended.**
0x0048	5:0	0x42	**UART Bypass Mode Control Register:** Bit[5]: Enable UART bypass mode control by remote GPIO pin (Function **MS** on **MFP8**) Bit[4]: Enable UART bypass mode control by local GPIO pin (**GPIO2)** Bit[3]: Enable or disable parity bit in bypass mode Bit[2]: UART soft-bypass timeout duration Bit[1]: Enable UART soft-bypass mode
0x004F	7:6 3:2	0x00	**UART Pass-Through Channels Config Register:** Bit[7]: Use standard or custom bit rate Bit[6]: Enable parity bit Bit[3]: Use standard or custom bit rate Bit[2]: Enable parity bit
0x3008	0	0x00	**Message Counter Reset Register:** Bit [0]: Reset Message Counter value to 0
0x3009	1:0	0x00	**CRC Reset Register:** Bit [1]: Resets Message Counter error count to 0

			Bit [0]: Resets CRC error count to 0.
0x300A	7:0	0x00	**Read CRC Value Register:** Bits [7:0]: CRC value for the last write transaction
0x300B	7:0	0x00	**Read Message Counter Low Bits Register:** Bits [7:0]: Low bits of current message counter value
0x300C	7:0	0x00	**Read Message Counter High Bits Register:** Bits [7:0]: High bits of current message counter value
0x300D	7:0	0x00	**CRC Error Count Register:** Bits [7:0]: Number of I^2C/UART CRC errors detected
0x300E	7:0	0x00	**Message Counter Error Count Register:** Bits [7:0]: Number of message counter errors detected
0x300F	2:0	0x06	**Enable CRC and Message Counter Register:** Bit [2]: Enable I^2C message counter. **Note: Only active when Bit [2] is also set to 1.** Bit [1]: Enable I^2C CRC packeting. **Note: Only active when Bit [2] is also set to 1.** Bit [0]: Enable manual override of I^2C CRC or message counter configuration. If set to a 0, the CRC and message counter features are disabled.

Enable Pass-Through UART Channel 1

This example enables pass-through UART channel 1.

```
# Enable pass-through UART Channel 1
0x98,0x0003,0x10
```

Enable CRC and Message Counter on Primary UART Control Channel

This example enables the CRC and message counter features on the primary UART control channel.

```
# Enables CRC and Message Counter features along with corresponding override enable
0x98,0x300F,0x07
```

Frame Synchronization (FSYNC)

Note: FSYNC is generally only used if there are multiple SerDes pairs in a system.

Frame synchronization (FSYNC) aligns images sent from multiple sources in surround-view applications and is required for concatenation. In the FSYNC mode, the GMSL2 CSI-2 deserializer sends a sync signal to the serializer; the serializers then send the signal to the connected sensor.

There are two types of FSYNC methods available on the GMSL2 CSI-2 deserializers: internal and external frame syncs. The internal frame sync indicates the GMSL2 CSI-2 deserializer generates the sync signal internally from its internal clock. The sync signal frequency must be specified in terms of the onboard crystal clock (25 MHz) in the FSYNC period registers *0x03E5 to 0x03E7*. The deserializer may be configured as a main that generates the FSYNC and outputs it on an MFP pin (MFP0 or MFP2), or as a subordinate. If configured as a subordinate, it may accept an FSYNC output from another deserializer configured as the main.

With external frame sync, the GMSL2 CSI-2 deserializer forwards a sync signal generated by a system-on-chip (SoC). In the GMSL2 application, any of the serializer or deserializer general purpose inputs/outputs (GPIOs) can be used as a sync signal input/output by using GPIO forwarding. However, in the GMSL1 mode, there are only specific GPI/GPO function pins available for sync signal forwarding (refer to the device data sheet for more information).

The *FSYNC_MODE* bit field configures the selection between the external or internal FSYNC.

Table 22. FSYNC Method and Mode

FEATURE	OFF/GPIO	EXTERNAL (GMSL)	MANUAL
FSYNCMETH	N/A	N/A	00
FSYNC MODE	11	10	00, 01
SOURCE OF FSYNC	None/SoC	Other GMSL	Deserializer
FSYNC CLOCK SOURCE	N/A	External (GMSL)	Controller PCLK/XTAL
WHY USE THIS MODE	SoC wants full control, FSYNC for multiple deserializers. FSYNC is non-periodic.	GPIO is used as FSYNC input driven by a main device.	SoC cannot create FSYNC signal or GPIO is used as FSYNC output and drives a subordinate device.

In the FSYNC mode, the auto-masking feature is enabled by default to mask corrupted images with blank screens. When enabled, the overall output is not impacted if any of the links are dropped.

Table 23 contains frame sync registers.

Note: External frame sync is recommended if the SoC can generate a sync signal.

Table 23. Frame Sync Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x3E0	7:0	0x00	**Frame Sync (FSYNC) Setting Register:** Bit 7: When enabled, memory overflow resets frame sync generation. Bit 6: Select FSYNC falling transition. 0 = Set at the middle of frame. 1 = Set immediately after rising transition. Bit 5: Select GPIO to output FSYNC signal (only works when FSYNC_MODE = 01). 0 = MFP0, 1 = MFP2 Bit 4: EN_VS_GEN Keep it 0 (for most cases) to use VS received from source. 0 = Use VS from source; VS is not generated internally. 1 = VS is generated internally to disregard source VS. Bits [3:2]: FSYNC_MODE 00 = FSYNC on; GPIO is not used as FSYNC input/output; set this mode for internal frame sync mode. 01 = FSYNC on; GPIO is used as FSYNC output and drives a subordinate device; set this mode for the main deserializer when there are multiple. 10 = FSYNC off; GPIO is used as FSYNC input driven by a main device; set this mode for subordinate deserializer(s). 11 = FSYNC off; GPIO is not used as FSYNC input/output; set this mode for external frame sync mode. Bits [1:0]: FSYNC Method. 00 = Manual 01 = Reserved 10 = Reserved 11 = Reserved
0x3E2	7:5	100	**FSYNC Controller Link Selection Register:** 000 = Do not use 00Y = Select pipe Y to be the master 010 = Do not use 011 = Do not use 1xx = Do not use (default)
0x3E5	7:0	0x00	**FSYNC Period Low Register:** Set FSYNC period (Bits [7:0]) in terms of clock cycles. (Effective when FSYNC Method = 00 and FSYNC Mode = 0x.)
0x3E6	7:0	0x00	**FSYNC Period Middle Register:** Set FSYNC period (Bits [15:8]) in terms of clock cycles. (Effective when FSYNC Method = 00 and FSYNC Mode = 0x.)
0x3E7	7:0	0x00	**FSYNC Period High Register:** Set FSYNC period (Bits [23:16]) in terms of clock cycles. (Effective when FSYNC Method = 00 and FSYNC Mode = 0x.)
0x3EF	7:0	0x92	**FSYNC Setting Register:** Bit 7: Select the type of FSYNC signal to output from GPIO. 0 = GMSL1 type, 1 = GMSL2 type. Bit 6: Uses crystal clock for generating frame sync signal. 0 = Disabled, 1 = Enabled. This bit should be enabled when generating FSYNC. Bit 4: Select how links are selected for FSYNC generation. 0 = Include links selected by FS_LINK_x registers. 1 = Include all enabled links.

			Bit 1: FS_EN_Y in FSYNC generation. 0 = Do not include video pipe Y. 1 = Include video pipe.
0x03F0	7:0	0x00	**FSYNC Error Counter Register (read only):** Bits [7:0]: Report FSYNC error counter; reset to 0 when read or when FSYNC_LOCKED asserted.
0x03F1	7:3	0x00	**FSYNC TX ID** Bits [7:3]: Select the GPIO for FSYNC to transmit to.
0x03F5	7:0	0x00	**FSYNC Difference Register (read only):** Bits [7:0]: Report the difference (Low Bits [7:0]) between the fastest and slowest frame in terms of controller PCLK cycles.
0x03F6	7:0	0x00	**FSYNC Difference and Lock Register (read only):** Bit 7: FSYNC_LOSS_OF_LOCK 0 = FSYNC loss of lock not detected. 1 = FSYNC loss of lock detected. Bit 6: FSYNC_LOSS_LOCKED (works only for internal frame sync modes) 0 = FSYNC is not locked. 1 = FSYNC is locked. Bits [5:0]: Report the difference (High Bits [13:8]) between the fastest and slowest frame in terms of controller PCLK cycles.
0x03F7	5	0x00	**FSYNC Reset:** Bit 5: FSYNC_RST_MODE 0 = Legacy Mode 1 = Start frame sync state machine regardless of video locks.
0x0B08	4	0x21	**GMSL1 FSYNC** Bit 4: EN_FSYNC_TX 0 = Disable FSYNC transmission in GMSL1 mode (default) 1 = Enable FSYNC transmission in GMSL1 mode

Note: The main link select (register0x3E2) is incorrect by default, and must be changed to select pipe Y.

Programming Examples

Internal FSYNC (GMSL1)

This example demonstrates the programming sequence required to enable internal FSYNC in the GMSL1 mode.

```
# Set to use 25MHz XTAL; AUTO_FS_LINKS = 1->0, FS_USE_XTAL = 1, FS_LINK_x[3:0] = 1,
GMSL1
#The MST_Link_SEL may need to be set in register 0x4A2 depending on sync generation
#0x98,0x03E2,0x00
0x98,0x03EF,0xDF
0x98,0x03EF,0xCF
# Set FSYNC period to 25M/30 clock cycles. CLK = 25MHz. Sync freq = 30Hz
0x98,0x03E7,0x0C
0x98,0x03E6,0xB7
0x98,0x03E5,0x35
# Enable FSYNC transmission to serializer. Select GPI_1
0x98,0x0B08,0x31
# Enable Internal FSYNC manual mode
0x98,0x03E0,0x04
# A 30Hz sync signal is then expected at Ser (e.g., MAX96705) GPO pin. No
sensor/PCLK is required.
```

Internal FSYNC (GMSL2)

This example demonstrates the programming sequence required to enable internal FSYNC in the GMSL2 mode.

```
# Set to use 25MHz XTAL; AUTO_FS_LINKS = 1->0, FS_USE_XTAL = 1, FS_LINK_x[3:0] = 1,
GMSL2
#The MST_Link_SEL may need to be set in register 0x4A2 depending on sync generation
#0x98,0x03E2,0x00
0x98,0x03EF,0xDF
0x98,0x03EF,0xCF
# Set FSYNC period to 25M/30 clock cycles. CLK = 25MHz. Sync freq = 30Hz
0x98,0x03E7,0x0C
0x98,0x03E6,0xB7
0x98,0x03E5,0x35
# Set FSYNC TX ID to 1 to match MFP1 on Ser side.
0x98,0x03F1,0x08
# Enable GPIO_RX_EN on Ser MFP1
0x80,0x02C1,0x84
# Enable Internal FSYNC manual mode
0x98,0x03E0,0x04
# A 30Hz sync signal is then expected at Ser (e.g., MAX9717) MFP1. No sensor/PCLK
is required.
```

External FSYNC (GMSL1)

The follow programming example enables the external FSYNC in the GMSL1 mode.

```
# This is GMSL1 Ext. FSYNC example; GPI_1/GPO_1 are used.
# Set Internal FSYNC off, GPIO is used for FSYNC, type = GMSL1
0x98,0x03E0,0x08
0x98,0x03EF,0x1F
# Enable GPI/GPO function in GMSL1 mode for each link;
0x98,0x0B08,0x21 (default)
```

External FSYNC (GMSL2)

The following script configures and enables the external FSYNC in the GMSL2 mode.

```
# This is GMSL2 Ext. FSYNC example;
# Update SER MFP1 RX ID = 2 to match MFP2 of MAX96714
0x80,0x02C3,0x02
# Config SER MFP1 to forward GPIO from the MAX96714
0x80,0x02C1,0x84
# Set Internal FSYNC off, GPIO is used for FSYNC, type = GMSL2
0x98,0x03E0,0x08
0x98,0x03EF,0x9F
# Config MAX96714 MFP2 to receive external FSYNC signal for each link
0x98,0x02B6,0x83
```

Power Manager

Overview

The MAX96714/MAX96714F/MAX96714R include an integrated power manager that 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 are validated and the digital core assumes normal operations. In all power modes, the power manger monitors power supplies for undervoltage and overvoltage conditions. Note that the MAX96714 has the power manager but does not have the sleep mode.

Table 24. Power Manager and Sleep Mode Availability for the MAX96714 Family

PART NUMBER	POWER MANAGER	SLEEP MODE
MAX96714	Supported	Not Supported
MAX96714F	Supported	Not Supported
MAX96714R	Supported	Not Supported
Other GMSL2 Device	Supported	Generally Supported

Device Power Operation

The part uses four common power rails (V_{DD}, V_{DD18}, V_{TERM}, and V_{DDIO}) and an integrated internal V_{DD} regulator.

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.

***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

The MAX96714/MAX96714F/MAX96714R 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 as follows.

- **V_{DD}:** The input voltage to the V_{DD} rail can be between 1.0 V and 1.2 V. An internal LDO regulates the voltage to 1.0 V.
- **V_{DD18}:** 1.8 V power rail.
- **V_{TERM}**: 1.2 V power rail.
- **V_{DDIO}**: 1.8 V or 3.3 V I/O power rail for I/O.
- **V_{DD_SW}** (CAP_VDD pin): Internal 1.0 V power rail that powers the digital core logic.

External power is supplied directly to V_{DD}, V_{DD18}, V_{TERM}, and V_{DDIO}, but the V_{DD_SW} (CAP_VDD pin) just has external capacitors connected.

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 11 shows the state diagram for the power manager.

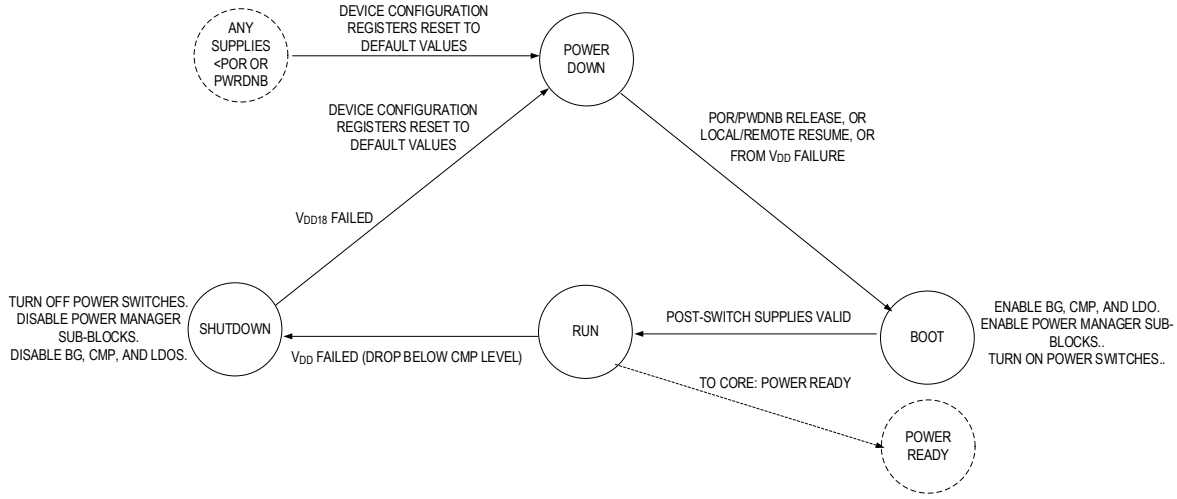

Figure 11. Power Manager State Diagram

Reset (Power Down)

Power down is the reset state. The device enters the power-down state if the PWDNB pin is asserted (low), V_{DD_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. Deasserting PWDNB (high) releases the chip from the power-down state and into the boot state.

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 the 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. On 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.

Shutdown

If the device is ever shut down, all power switches are turned off and the power manager blocks are disabled. The device then enters the reset state.

Sleep Mode

The MAX96714/MAX96714F/MAX96714R do not support the sleep mode.

Register CRC

Overview

This device includes a register CRC to alert if the device is accidentally placed into an undesired state. This is done by calculating a CRC value based on the state of the specific control registers' values. These control registers program certain device and system parameters such as, but not limited to, GMSL link speed, MIPI port configuration, and GPIO configuration. If any of these parameters are changed mid-operation, the device configuration changes, and the ERRB output of the device is activated.

The period between CRC calculations is programmable from 2 ms to ~500 ms.

Usage Models

There are two usage modes to employ register CRC (basic and rolling). There is also a register block to skip specific registers from CRC calculation.

Basic CRC

The basic mode simply calculates a CRC value during each 'CRC_PERIOD' and checks that this value remains unchanged. The calculated CRC value is deposited in the REGCRC_MSB/LSB registers. Errors are indicated on the interrupt pin, which can be enabled or disabled using the standard interrupt mechanism.

Rolling CRC

The rolling CRC mode incorporates a rotating 2-bit counter that is incremented each 'CRC_PERIOD' so that the CRC value changes through four values in a repeating fashion. The calculated CRC value is deposited in the REGCRC_MSB/LSB registers. In this mode, poll the CRC value to verify the CRC value is cycling through these four values and has not stopped working. ERRB is not indicated as the CRC value is changing periodically, and the usage mode is to have the user poll the CRC register.

Skipping Registers from CRC Calculation

If desired, registers can be removed from the CRC calculation using SKIPX_MSB[7:0] and SKIPX_LSB[7:0]; where X = 0 to 7.

Note: The register CRC protection mechanism must be enabled as the final function on the chip (including all interrupt ERRB flags); no register writes to CRC protected registers can occur after enabling the register CRC, or the CRC is corrupted.

System Implementation

Follow these steps to enable the Register CRC feature.

1. Configure SerDes link per use case. Program any 'SKIP' registers to avoid CRC calculation. Example: Skipping GPIO_C register 0x02D8: SKIP0_MSB=0x02, SKIP0_LSB=0xD8.
2. Enable video.
3. Enable the REG CRC feature.
    a. Basic CRC
        i. Set REG_CRC_ERR_OEN = 1b'1.

ii. Set I2C_WR_COMPUTE = 0b'1.
iii. Set CRC_PERIOD[7] = 1'b1 (corresponds to a CRC period of ~250 ms).
iv. Set PERIODIC_COMPUTE = 1b'1.
v. Set CHECK_CRC bit = 1b'1.
vi. Read calculated CRC MSB/LSB and store.
Example: 'REGCRC_Original', REGCRC_MSB[7:0], REGCRC_LSB[7:0]

b. Rolling CRC
i. Set GEN_ROLLING_CRC = 1'b1.
ii. Follow steps for basic CRC mode.
1. Read calculated CRC MSB/LSB and store.
Example: 'REGCRC_Original', REGCRC_MSB[7:0], REGCRC_LSB[7:0]
2. There are four individual CRC calculated automatically that should be checked per CRC_PERIOD.

4. *If REG_CRC_ERR_FLAG only asserts, do the following:
a. Rewrite register values to get back to the original state.
b. Read calculated CRC MSB/LSB.
i. If new calculated CRC MSB/LSB = 'REGCRC_Original', registers are back to the original state.
1. Do RESET_CRC = 1'b1.
2. Continue system operation.
ii. If new calculated CRC MSB/LSB $\neq$ 'REGCRC_Original', a different register must have been written.
1. Restart the link.
2. Reconfigure the SerDes link.

Note: *Events that trigger REG_CRC_FLAG may also simultaneously trigger other ERRB flags. Prioritize error handling accordingly.

CRC-Protected Status Registers

There are video status registers that change logic levels depending on whether there is input video. It is recommended to reset the CRC value when system changes states (that is, with or without video running). Another option is to 'SKIP' these status registers.

Table 25: CRC-Protected Deserializer Video Status Registers

REGISTER	BITFIELD(S)
VPRBS Y	VIDEO_LOCK, Pipe Y
BACKTOP1	CSIPLLY_LOCK, Pipe Y
GPIOX: GPIO_C	GPIO_RECVED

Note: "X" is the GPIO number. It is recommended to skip the GPIO_RECVED register from the CRC calculation.

Table 26: Relevant CRC Registers

DEVICE PART NUMBER	REGISTER NAME	REGISTER FUNCTION	REGISTER ADDRESS
MAX96714/MAX96714F	FS_INTR0	Reg CRC Error Flag to ERRB	0x3010
	FS_INTR1	Reg CRC Error Flag	0x3011
	REGCRC0	Register CRC Feature Enable	0x3000
	REGCRC1	Register CRC Period	0x3001
	REGCRC2	Register CRC LSB	0x3002
	REGCRC3	Register CRC MSB	0x3003
	REGCRC8	Skip Register 0 LSB	0x3030
	REGCRC9	Skip Register 0 MSB	0x3031
	REGCRC10	Skip Register 1 LSB	0x3032
	REGCRC11	Skip Register 1 MSB	0x3033
	REGCRC12	Skip Register 2 LSB	0x3034
	REGCRC13	Skip Register 2 MSB	0x3035
	REGCRC14	Skip Register 3 LSB	0x3036
	REGCRC15	Skip Register 3 MSB	0x3037
	REGCRC16	Skip Register 4 LSB	0x3038
	REGCRC17	Skip Register 4 MSB	0x3039
	REGCRC18	Skip Register 5 LSB	0x303A
	REGCRC19	Skip Register 5 MSB	0x303B
	REGCRC20	Skip Register 6 LSB	0x303C
	REGCRC21	Skip Register 6 MSB	0x303D
	REGCRC22	Skip Register 7 LSB	0x303E
	REGCRC23	Skip Register 7 MSB	0x303F

Reference-over-Reverse (RoR)

Note: RoR is a serializer function. Check the serializer user guide for more information

Crystal or RoR for Basic Function

All GMSL2 parts must receive an external clock for full function (typically through a crystal). Some serializers, such as the MAX96717F, can get a reference clock through reference clock over reverse channel (R0R). In RoR, there is no need to connect a crystal next to the serializer because the MAX96714 is supplying the reference signal on the reverse channel of the GMSL link.

Using RoR instead of the crystal oscillator provides several advantages:

- Reduced system cost
- Increased reliability
- Reduced board area
- Simplified board layout

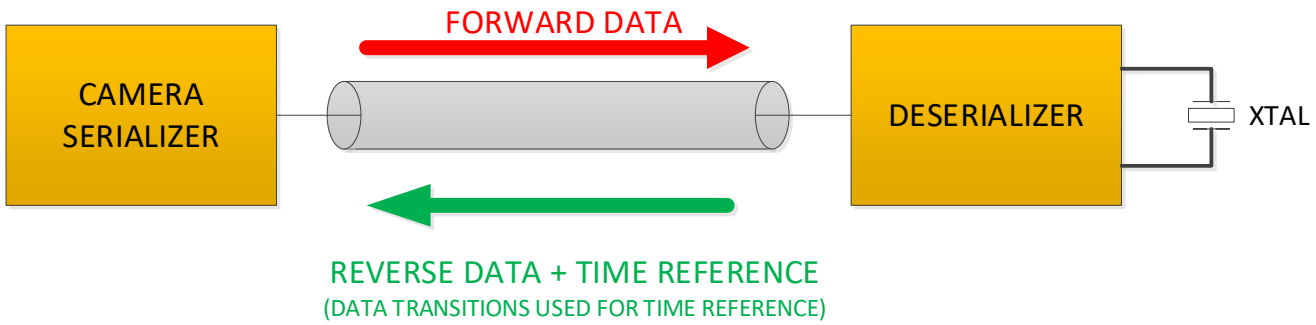

Figure 12. RoR Operation

Enabling the RoR Mode

This device automatically detects RoR. No additional deserializer configuration is required. Serializer configuration may be needed. See the serializer user guide for RoR configuration and status.

GMSL Link Lock Time

GMSL link lock time in the RoR-mode is like it is in the crystal mode. Refer to the device data sheet for lock time specifications.

RoR Jitter Considerations

The deserializer clock source must meet the data sheet requirements for reference clock input jitter. This specification is easily met by typical clock sources.

Spread Spectrum Clocking (SSC) During RoR

Spread spectrum clocking (SSC) is an alternate configuration for the GMSL link that helps with electromagnetic interference (EMI). SSC is supported in both the Xtal and RoR modes. If it is used, it is enabled only in the deserializer. The serializer clock follows the deserializer clock modulation and consequently the serializer operates with a spread spectrum clock as well. The serializer RCLKOUT output reference is also frequency modulated in the same manner.

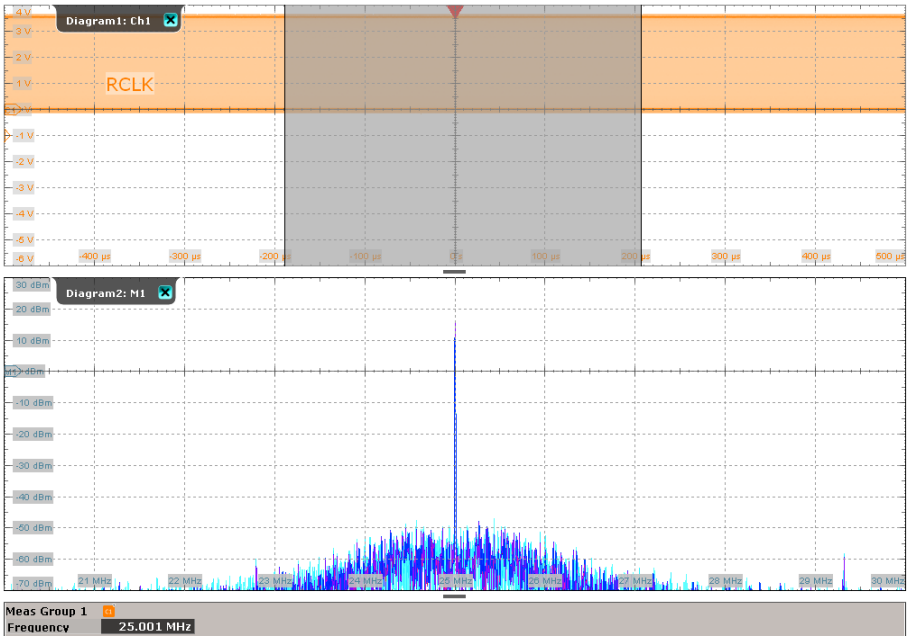

Figure 13. 25 MHz RCLKOUT Signal Using RoR Without SSC Feature Enabled

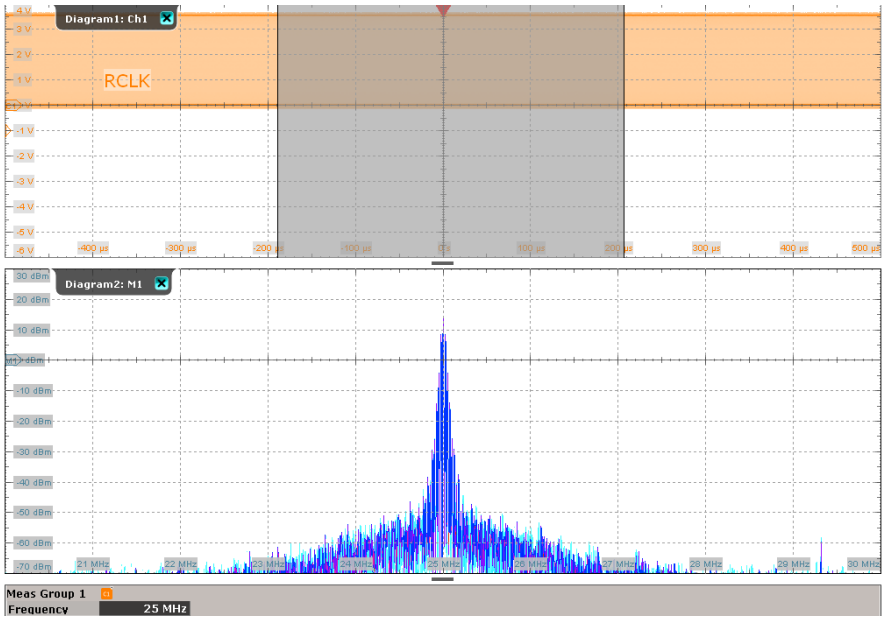

Figure 14. 25 MHz RCLKOUT Signal Using RoR with SSC Feature Enabled

Power-on-Self-Test (POST) (MAX96714/MAX96714F Only)

POST runs on the chip during the power-up sequence and is turned off afterwards. During POST, a portion of the logic is tested using the logic built-in self-test (LBIST) and memories are tested using the memory built-in self-test (MBIST).

The runtime for POST is under 10 ms and cannot be bypassed. Following POST, a status register contains a bit indicating pass/fail for LBIST and a bit for MBIST pass/fail. Although LBIST/MBIST testing may fail, the chip continues the bring-up operation, enabling continued functionality.

As LBIST and MBIST operate on multiple blocks, LBIST and MBIST pass/fail bits indicate that all LBIST sub-blocks or all MBIST memories passed. Therefore, a failure in one of the LBIST sub-blocks or one of the MBIST memories indicates that the entire LBIST or MBIST operation failed.

Operation

POST is not bypassable. There are no configuration bits for this function. During start-up, do not attempt to configure the part until 15 ms after power is applied. The status of the POST step is checked by reading the following register bit fields.

Note: The MAX96714R can be accessed after 5 ms as it does not have POST,

Register: REG_POST0 (0x3020)

D7: POST_DONE: This value should always be ‘1’, for parts that have POST enabled.
D6: POST_MBIST_PASSED: This value should be ‘1’ for passing MBIST.
D5: POST_LBIST_PASSED: This value should be ‘1’ for passing LBIST.
D1: POST_RUN_MBIST: This value should be ‘1’ if the device runs MBIST at start-up.
D0: POST_RUN_LBIST: This value should be ‘1’ if the device runs LBIST at start-up.

If LBIST or MBIST fails, do not use the device (shut down the application).

Bandwidth Efficiency Optimization

Overview

GMSL bandwidth is a limitation of the serializer. Refer to the serializer user guide for details and tips to optimize the bandwidth. The following section mentions some of the bits used during optimization.

Using Double Mode

If the serializer is using the double mode, the following bits should be set:

1. Double pixel mode for 8 bpp
    - Set *bpp8dblZ* = 1, *soft_bppZ_en* = 1, and *soft_bppZ* = 16 in the serializer.
    - Set *bpp8dblY* = 1, *bpp8dblY_mode* = 1, and *ALT_MEM_MAP8* = 1 in the MAX96714.

2. Double pixel mode for 10 bpp
    - Set *bpp10dblZ* = 1, *soft_bppZ_en* = 1, and *soft_bppZ* = 20 in the serializer.
    - Set *ALT_MEM_MAP10* = 1 in the MAX96714.

3. Double pixel mode for 12 bpp
    - Set *bpp12dblZ* = 1, *soft_bppZ_en* = 1, and *soft_bppZ* = 24 in the serializer.
    - Set *ALT_MEM_MAP12* = 1 in the MAX96714.

Note: Serializer bit names may differ from what is shown above. Check the serializer user guide for more information.

Note: The operation for the 8 bpp double mode is different than the 10 bpp/12 bpp and requires two extra writes in the MAX96714 deserializer.

MIPI Packet Counters

The MIPI packet count registers determine whether MIPI data is flowing in the GMSL2 CSI-2 deserializer. The *csi2_tx*_pkt_cnt* registers report the packet count of the associated MIPI controller; the *phy*_pkt_cnt* registers report the packet count of the associated MIPI PHY. Sequential reads returning different values indicate that data is being transmitted by the associated MIPI controller or PHY. The register value does not change if data is not flowing. See *Table 27* for MIPI packet count registers.

Note: While other GMSL deserializers make distinctions between the controllers and PHYs, all data from the controller is passed through the PHY in the MAX96714.

Table 27. MIPI Packet Count Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION	DECODE
0x0342	3:0	0x00	Packet count of CSI-2 Controller 1	**csi2_tx*_pkt_cnt registers:** 0bXXXX: Toggling bits indicate MIPI data is active on the controller.
0x0344	7:4	0x00	Packet count of MIPI PHY1	**phy*_pkt_cnt registers:** 0bXXXx: Toggling bits indicate MIPI data is active on the PHY.

HSYNC, VSYNC, DE (HVD) Outputs and Counters

The GMSL2 CSI-2 deserializers are capable of outputting the VS or DE signals from the video pipe data streams on MFP pins. Each pin can be selected to output from the video pipe.

Pulses can be output on two different GPIOs.

- MFP0, 5 – VS, or DE, or output to GPIO

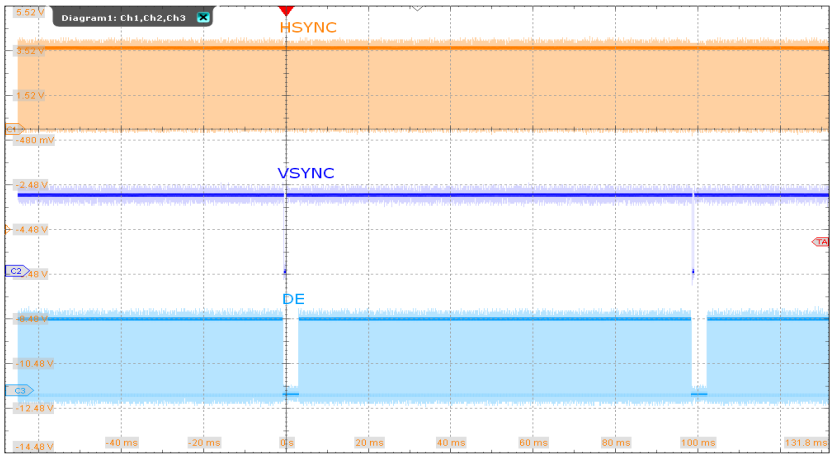

Figure 15. HS, VS, and DE Outputs from GPIOs

GPIOs generally multiplex with multiple functions. Therefore, to output sync pulses, the default or higher-priority GPIO function(s) of the above pins must be disabled. MFP5 is enabled as LOCK by default. So, this feature must be disabled.

See *Table 28* for sync pulse output registers.

Table 28. Sync Pulse Output Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0540	7	0x00	**VS_OUT1 enable register:** Bit 7: VS Enable MFP0
0x0541	7	0x00	**VS_OUT2 enable register:** Bit 7: VS Enable MFP5
0x0542	7	0x00	**HS_OUT1 enable register:** Bit 7: DE Enable MFP0

Alternatively, sync pulse signals can be monitored through registers. Registers *0x575* contain status flags indicating the detection of DE/HS/VS signals for each video pipe. The polarity of the HS/VS signals is accessed in the same register. HVD counters may also be read to verify the pipe count values match the expected video input values. Additionally, comparators may be set up and enabled to flag dropped frames or lines as errors. See *Table 29* for additional information.

Table 29. Sync Signal Status Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0575	3:0	N/A	Bit 6 Indicates if DE signal is detected in video pipe. 0b0: DE is not detected 0b1: DE is detected Bit 5 Indicates if VS signal is detected in video pipe. 0b0: VS is not detected 0b1: VS is detected Bit 4 Indicates if HS signal is detected in video pipe. 0b0: HS is not detected 0b1: HS is detected Bit 1 Indicates the detected VS signal polarity in video pipe. 0b0: Active low 0b1: Active high Bit 0 Indicates the detected HS signal polarity in video pipe. 0b0: Active low 0b1: Active high

Note: HS is generally not used by the MAX96714. Line start and line end (DE) determine the start and end of a CSI long packet.

Programming Examples

The following examples demonstrate the configuration required to output the sync pulses in both the GMSL2 and GMSL1 modes.

Output Pipe 1's DE on MFP0, VS on MFP5, and DE on MFP7

```
# Disable Lock for MFp5
0x98,0x0005,0x40
# Enable VS_OUT for MFP5
0x98,0x0541,0x80
# Enable DE_OUT for MFP0
0x98,0x0542,0x80
```

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. So, it is a convenient way to check for errors. It is available at MFP8. Whether an error is included in the ERRB output depends on if its output-enable (OEN) is high. Most OENs are high by default.

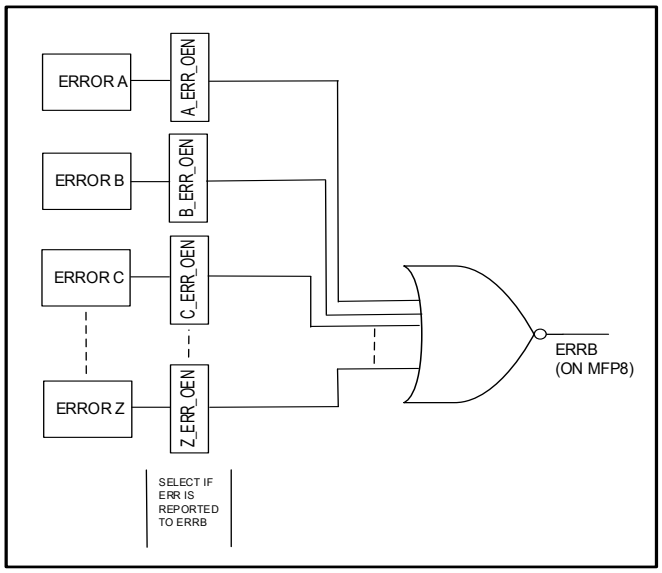

Figure 16. ERRB Reporting Flow

See *Table 30* for a description of each error flag in the MAX96714 and MAX96714F.

Note that some of these are not available in the MAX96714R. Refer to the device data sheets for details.

Table 30. Error Flags Table

ERROR FLAG	ERROR DESCRIPTION
cmd_overflow2	Command FIFO overflow occurred.
CMP_STATUS	V_{DD18}, V_{DDIO}, and CAP_VDD voltage
Csipll1_PLLORangeH_flag	Indicates csipll1 is above range at some point since last cleared.
Csipll1_PLLORangeL_flag	Indicates csipll1 is below range at some point since last cleared.
DEC_ERR_FLAG_A	Errors are detected in a GMSL data packet. This could indicate poor link quality.
EFUSE_CRC_ERR_FLAG	An issue with the device EFUSE occurred. The device should no longer be used.
EOM_ERR_FLAG_A	Indicates that receiver eye-opening is below the configured threshold.
FSYNC_ERR_FLAG	An error in the frame synchronization occurred.
G1_A_ERR_FLAG	An error on the GMSL1 link is detected.

G1_CC_CRC_ERR_FLAG	GMSL1 CC CRC error violation flag.
G1_DET_ERR_FLAG	GMSL1 decode error flag
G1_EOM_ERR_FLAG	GMSL1 EOM eye width minimum width violation flag.
G1_LINE_CRC_ERR_FLAG	GMSL1 video line CRC error violation flag.
G1_MAX_RT_ERR_GPI_FLAG	GMSL1 maximum retransmission on GPI error flag.
G1_MAX_RT_ERR_I2C_FLAG	GMSL1 maximum retransmission on I^2C error flag.
G1_PRBS_ERR_FLAG	GMSL1 PRBS error flag.
I2C_UART_CRC_ERR_INT	CRC error occurred for I^2C packet.
I2C_UART_MSGCNTR_ERR_INT	Error in the I^2C message counter.
IDLE_ERR_FLAG_A	Errors are detected in a GMSL Idle packet. This could indicate poor link quality.
LCRC_ERR	A video line CRC error is detected. This error is for each video pipe.
LCRC_ERR_FLAG	A video line CRC error is detected on any of the video pipes. This is a combined error.
LFLT_INT	Line-fault interrupt indicates at least one line-fault occurred.
LFLT_INT_FLAG	Line-fault error indicator.
LMO_Y	Line memory overflow occurred.
MAX_RT_ERR_GPI	Maximum retransmission error flag. Cleared when read.
MAX_RT_ERR_I2C	Maximum retransmission error flag. Cleared when read.
MAX_RT_FLAG_A	Combined ARQ maximum retransmission limit error flag.
PHY_INT_A	PHY interruption occurred.
PKT_CNT_FLAG	Packet counter reached packet count threshold.
REM_ERR_FLAG	An error on the serializer is flagged. This is the status of the serializer ERRB level.
RT_CNT_FLAG	One or more of the selected channels has done at least one ARQ retransmission.
STATUS	**MIPI Tx Status Register**: The register is split into decode segments: Bit [0] SYNC mode enable. Bit [1] Video sync flag. Bit [2] Loss of video sync flag. Bit [3] Tunneling mode: DPHY ECC (correctable). Bit [4] Tunneling mode: DPHY ECC (uncorrectable). Bit [5] Tunneling mode: DPHY data CRC error.
TUN_DATA_CRC_ERR	For tunneling mode, DPHY data CRC errors.
TUN_ECC_CORR_ERR	For tunneling mode, correctable errors on DPHY ECC.
TUN_ECC_UNCORR_ERR	For tunneling mode, uncorrectable errors on DPHY ECC.
VDDBAD_INT_FLAG	Combined VDDBAD indicator, indicates an issue with V_{DD} level.
VDDCMP_INT_FLAG	Combined V_{DD} comparator output. See CMP_STATUS and VDDCMP_MASK registers.
VID_PXL_CRC_ERR_FLAG	Video pixel CRC error detected.
VID_SEQ_ERR	Video Rx sequence error detection.
VPRBS_ERR_FLAG	Video PRBS errors are detected. This is for testing only.

General-Purpose Input and Output (GPIO)

Overview

The MAX96714 family has nine multifunction pins (MFPs). 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., I^2C, LOCK, ERRB, etc.). This section explains the GPIO function of MFP pins. Refer to the data sheets for additional details on GPIO capabilities and default states after power-up.

The GPIO blocks of the MAX96714 family 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.

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.

The MAX96714 uses 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 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.

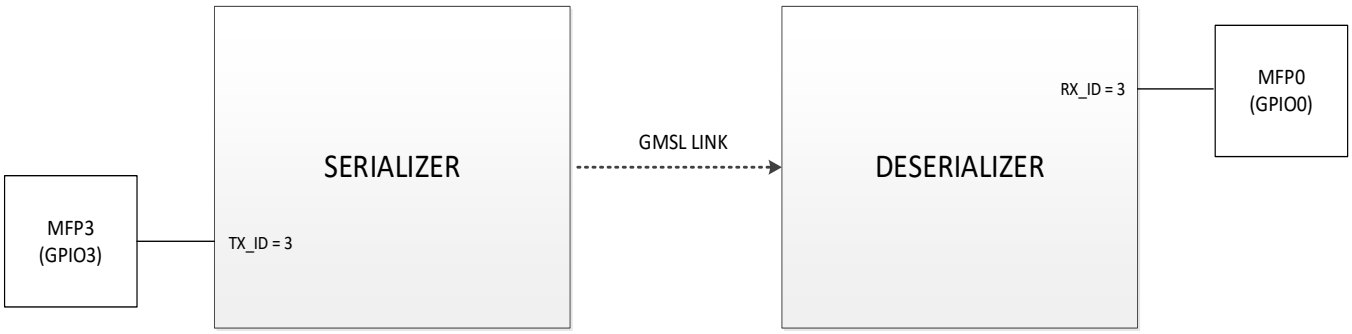

Figure 17. GPIO Forwarding Example with a Transition from MFP3 to MFP0

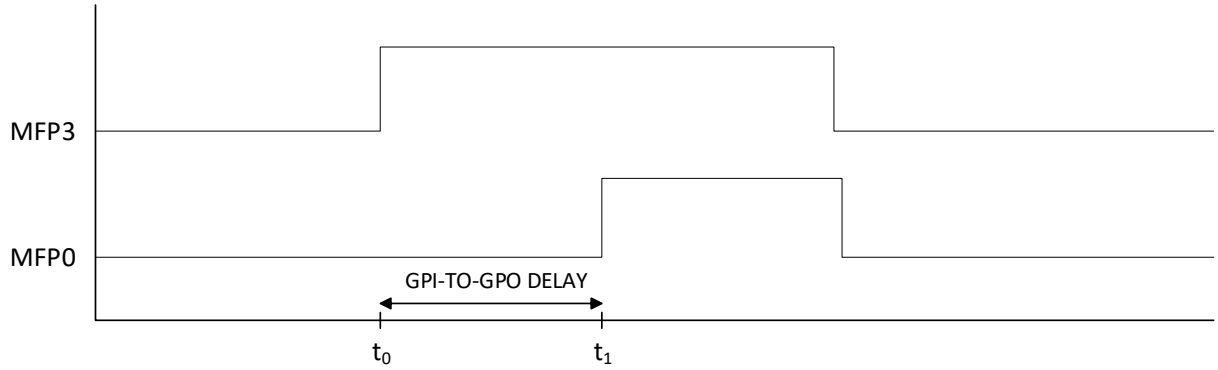

Figure 18. GPIO Forwarding Timing Diagram

MFP Capabilities: GPIO vs. GPI vs. GPO vs. ODO

The MAX96714 have nine MFPs; seven of these MFPs are GPIO (general-purpose input or output), two are ODO_GPI (open-drain output or general-purpose input) and reserved for the Primary control channel. *Table 31* shows the GPIO capabilities of each MFP for the different devices:

Table 31. Multifunction Pin (MFP) Capabilities

PIN NAME	CAPABILITY
MFP0	GPIO0
MFP1	GPIO1
MFP2	GPIO2
MFP3	RDX/SDA
MFP4	RX/SCL
MFP5	GPIO5
MFP6	GPIO6
MFP7	GPIO7
MFP8	GPIO8

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 40 kΩ or 1 MΩ.

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 the 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 (GMSL2 only)

Note: GPIO forwarding is only available in the GMSL2 mode. For forwarding with GMSL1, see the *Pairing with GMSL1 Serializers* section.

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:

1. 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.
2. 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 (GMSL2 Only)

Note: GPIO broadcasting is not available in the GMSL1 mode.

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 19* is an example of this configuration.

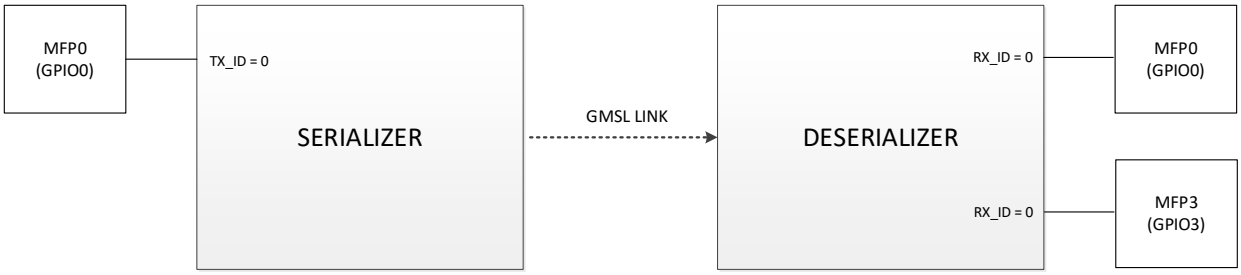

Figure 19. GPIO Broadcasting

GMSL2 GPIO Delay Compensation

Note: See the Pairing with GMSL1 Serializers section for GMSL1 operation.

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 32* and *Table 33* show the typical values and registers to set delay compensation.

Table 32. Delay Values with and without Compensation

DIRECTION	DELAY COMPENSATION	DELAY
GPIO forwarding from serializer to deserializer	0	1 µs
	1	3.5 µs (default)
GPIO forwarding from deserializer to serializer	0	6 µs
	1	15 µs (default)

Table 33. Compensation Delay Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x30	5:0	0b000001	**GPIOA:** Bit [5:0]: GPIO_FWD_CDLY Compensation delay multiplier for the forward direction. This must be the same value as GPIO_FWD_CDLY of the chip on the other side of the link. Total delay is the (value + 1) multiplied by 1.7 µs. Default delay is 3.4 µs.
0x31	5:0	0b001000	**GPIOB:** Bit [5:0]: GPIO_REV_CDLY Compensation delay multiplier for the forward direction. This must be the same value as GPIO_REV_CDLY of the chip on the other side of the link. Total delay is the (value + 1) multiplied by 1.7 µs. Default delay is 3.4 µs.

Toggling GPIO Manually with Registers

GPIO pins can be manually controlled through I^2C 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.

GPIO Registers

Table 34. GPIO Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x2B0	7:0	0x81	**GPIO0 A:** Bit 7: RES_CFG 0 = 40 kΩ, 1 = 1 MΩ Bit 4: GPIO_OUT 0 = Drive output to 0, 1 = Drive output to 1 Bit 3: GPIO_IN, GPIO input level 0 or 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
0x2B1	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 to 31
0x2B2	6:0	0x40	**GPIO0 C:** Bit 6: GPIO_RECVED, Received GPIO Value 0 or 1 Bit [4:0]: GPIO_RX_ID, Address = 0 to 31
0x2B3 to 0x2CA	…	…	**Repeat Registers A, B, C for GPIO1 to GPIO8.**

GPIO Programming Example

In this example, GPIO0 on a MAX96717 serializer is forwarded across the link to GPIO0 on a MAX96714 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.

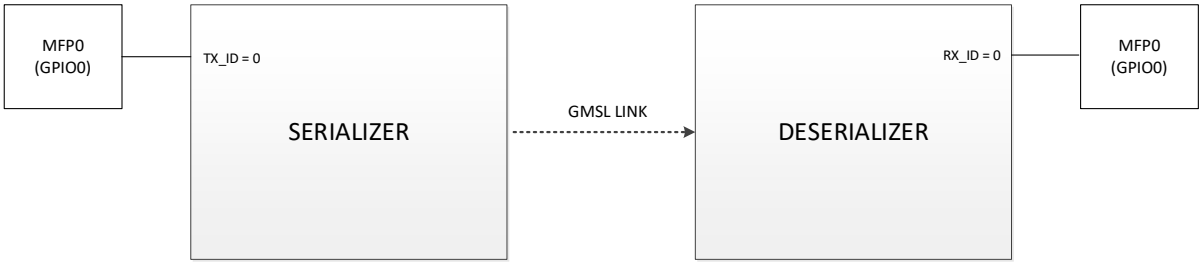

Figure 20. GPIO Forwarding Programming Example

Table 35. GPIO Programming Example

STEP	ACTION	DEVICE	READ/WRITE
1	Set up deserializer GPIO0 Rx enable, enable output driver, and set resistor to 1 MΩ.	DES	W
2	Set up deserializer GPIO output type to push-pull and configure resistor as pull-down.	DES	W
3	Set up deserializer GPIO0 to receive ID GPIO_RX_ID to be 0.	DES	W
4	Set up serializer GPIO0 Tx enable.	SER	W
5	Set up serializer GPIO0 transmit ID GPIO_TX_ID to be 0.	SER	W

MFP Slew Rate

The MAX96714's 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.

MFP Slew Rate Operation

The configurable slew rate applies to the various MFP functions differently:

1. **I²C/UART**: MFP pins operating as an I²C or UART function (i.e., control channel or pass-through) are not affected by the MFP rise/fall setting. The I²C/UART circuitry has a fixed falling-edge slew rate, and the rising-edge slew rate is determined by the external pull-up resistor.
2. **Dedicated Function**: The rise and fall times of MFP pins assigned dedicated functions (e.g., SPI, RCLKOUT, or LOCK and ERR) can be adjusted by the MFP slew rate control registers.
3. **GPIO**: MFP pins operating as GPI or GPO can be adjusted by the MFP rise/fall slew rate control register.

The V_{DDIO} supply voltage affects the I/O slew rate. The impact of the chosen V_{DDIO} voltage must be considered when programming MFP slew rates.

MFP Slew Rate Programming and 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.

The I²C/UART functions are not affected by the MFP slew rate adjustment. If an MFP is used as an I²C or UART pin, the slew rate is automatically adjusted to meet the applicable specification.

The device V_{DDIO} level determines the available range of the slew rate configuration options. For each V_{DDIO} level, the MFP speed groups have four available speed options configured by two speed control bits.

CMU4 is the MFP speed control register for the MAX96714.

Refer to the corresponding device's data sheet "Control- and Side-Channel Typical Rise and Fall Times" section for V_{DDIO} timing details. *Table 36* presents the typical rise and fall times for GMSL2 devices.

Table 36. Typical Rise/Fall Times for GMSL2 Devices

REGISTER VALUE	RISE/FALL TIME	
	V_{DDIO} = 1.8 V	V_{DDIO} = 3.3 V
0x0	1 n/0.8 n	0.6 n/0.5 n
0x1	2.1 n/2 n	1.1 n
0x2	4 n/4.3 n	2.3 n
0x3	9 n/10 n	5 n

MFP Slew Rate Example

Example 1. GPIO Example Using the CSI-2 Deserializer

The MAX96714 PIO_SLEW0 (0x570) **r**egister to configure the MFP0 slew rate. This register has the following mapping.

Table 37. CMU4 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PIO03_SLEW		PIO02_SLEW		PIO01_SLEW		PIO00_SLEW	

Note: Register mappings vary by device family. Refer to the device-specific data sheet for register mapping details.

Video PRBS Generator and Checker

Overview

One way to verify the link is functional and check for issues is to run a pseudorandom binary sequence (PRBS) test. During this testing, the serializer generates the video PRBS signal and the deserializer checks it.

In conjunction with the VPRBS generator/checker, the MAX96717/MAX96717F/MAX96717R's error generator feature can also be used to validate that the errors are seen on the deserializer. The MIPI input to the serializer must be disabled before running this test.

MAX96717 Serializer (Not Deserializer) Video PRBS and Status Registers

Table 38. Serializer Video PRBS Generator and Checker Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION	DECODE
0x110	3	0b1	Select bpp source.	0b0: Use BPP from register (Note 1) 0b1: Use BPP from MIPI RX
0x24F	3:1	0b000	Pattern generator clock source for video PRBS, checkerboard, and gradient patterns.	0b0xx: Use external PCLK 0b100: Use 25 MHz internal CLK 0b101: Use 75 MHz internal CLK 0b110: Use 150 MHz internal CLK 0b111: Use 375 MHz internal CLK
0x26B	7	0b0	Enable video PRBS generator.	0b0: Video PRBS generator disabled 0b1: Video PRBS generator enabled
0x112	7	0b0	PCLKDET	0b0: Video transmit PCLK not detected 0b1: Video transmit PCLK detected

Note 1: When VPRBS is enabled, the default bpp = 24 when register 0x111 is used.

Deserializer Video PRBS Registers

Table 39. Deserializer Video PRBS Generator and Checker Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION	DECODE
0x01FC	5	0b0	Indicates VPRBS check PASS/FAIL.	0b0: PASS 0b1: FAIL
	4	0b0	Enables VPRBS checker.	0b0: Disabled 0b1: Enabled
	0	0b0	Video channel locked and outputting valid video data.	0b0: Unlocked 0b1: Locked
0x01FB	7:0	0x00	Video PRBS error counter, clears on read.	0xXX: Number of video PRBS errors since last read.
0x001E	2	0b1	Enables reporting of video PRBS errors (VPRBS_ERR_FLAG—0x1F) at ERRB pin.	0b0: Disable video PRBS error reporting. 0b1: Enable video PRBS error reporting.
0x001F	2	0b0	Video PRBS Error Flag. Asserted when VPRBS_ERR (0x1FB) > 0.	0b0: VPRBS_ERR = 0 0b1: VPRBS_ERR > 0

Video PRBS Programming Examples
Serializer Generated VPRBS

The following script configures a MAX96717 and a MAX96714 to conduct the standard VPRBS test. In this test, the serializer generates a PRBS pattern and the deserializer checks it. For this test, the serializer must have PCLK and no video must be running into the serializer.

```
# Connect Serializer GMSL link to Deserializer GMSL link B

#Disable auto bpp on serializer
0x80,0x0110,0x60
#Enable serializer internal PCLK generation, PCLK = 150MHz
0x80,0x024F,0x0D
#Enable serializer pattern generator
0x80,0x026B,0x01
#Delay for 3ms
#Enable PRBS checker for deserializer
0x98,0x01FC,0x90
#Delay for 3ms
#Enable serializer VPRBS generator
0x80,0x026B,0x81

#Verify serializer has PCLKDET = 1
0x80,0x0112,0x8A
#Verify deserializer has VIDEO_LOCK = 1 for Pipe 1
0x98,0x01FC,0x91
#Verify deserializer does not have PRBS errors VPBRB_ERR = 0x00
0x98,0x01FB,0x00

#Optional Step: Enable Serializer errors to verify setup and PRBS error detection
#Note: Error generation also creates decode and idle errors, so these must be
cleared as well.
#Enable serializer error generator
0x80,0x0029,0x18
#Delay for a short time to accumulate errors.
#Disable serializer error generator
0x80,0x0029,0x08
#Verify deserializer has PRBS errors VPBRB_ERR > 0x00, 0xFF is used as the read
value in this example, but errors may vary. VPRBS Errors should clear after reading
this register.
0x98,0x01FB,0xFF
```

Video Pattern Generator (VPG)

The video pattern generator (VPG) creates either a checkerboard or gradient pattern with programable 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 deserializer has an internal VPG that accommodates a wide range of resolutions and frame rates. The VPG does not require an external PCLK source from the CSI input and uses the external 25 MHz crystal clock (i.e., REFCLK input) to derive four different pixel clock (PCLK) options. Link lock is not required to use the VPG.

The VPG has its own register block settings for timing configurations. *Table 40* contains video pattern register addresses for the VPG.

There are two clock configuration registers that set the PCLK value for the video pipes. The video pattern PCLK frequency can optionally be set to 25 MHz/75 MHz/150 MHz/375 MHz. This internal PCLK is not related to the MIPI CSI port clock rate, which must be set to accommodate the VPG data stream. See *Table 41* for configuration details. *Table 42* contains reference information to select the VPG PCLK.

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

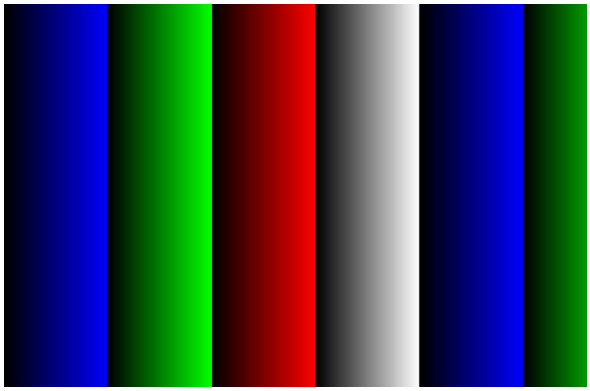

Figure 21. VPG (Gradient Pattern)

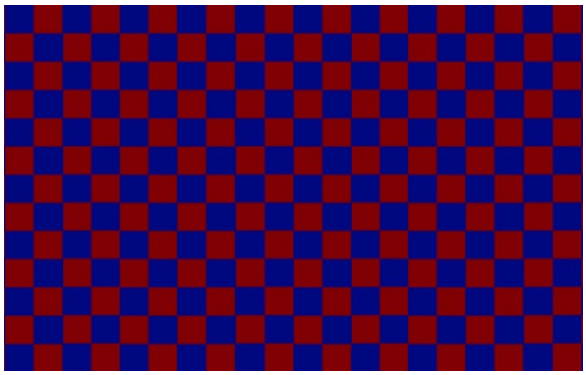

Figure 22. VPG (Checkerboard Pattern)

Table 40. Video Pattern Registers

REGISTER (VPG0/VPG1)	BITS	DEFAULT VALUE	DESCRIPTION
0x0240	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.
0x0241	5:4	00	**Pattern Generator Mode Register:** Bits [5:4]: Pattern selection. 01 = Color Gradient 10 = Checkerboard
0x0242	7:0	0x00	**VS Delay Register 2:** VS Delay in terms of PCLK cycles. (Bits [23:16])
0x0243	7:0	0x00	**VS Delay Register 1:** VS Delay in terms of PCLK cycles. (Bits [15:8])
0x0244	7:0	0x00	**VS Delay Register 0:** VS Delay in terms of PCLK cycles. (Bits [7:0])
0x0245	7:0	0x00	**VS High Register 2:** VS High Period in terms of PCLK cycles. (Bits [23:16])
0x0246	7:0	0x00	**VS High Register 1:** VS High Period in terms of PCLK cycles. (Bits [15:8])
0x0247	7:0	0x00	**VS High Register 0:** VS High Period in terms of PCLK cycles. (Bits [7:0])
0x0248	7:0	0x00	**VS Low Register 2:** VS Low Period in terms of PCLK cycles. (Bits [23:16])
0x0249	7:0	0x00	**VS Low Register 1:** VS Low Period in terms of PCLK cycles. (Bits [15:8])
0x024A	7:0	0x00	**VS Low Register 0:** VS Low Period in terms of PCLK cycles. (Bits [7:0])
0x024B	7:0	0x00	**V2H Register 2:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [23:16])
0x024C	7:0	0x00	**V2H Register 1:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [15:8])
0x024D	7:0	0x00	**V2H Register 0:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [7:0])
0x024E	7:0	0x00	**HS High Register 1:** HS High Period in terms of PCLK cycles. (Bits [15:8])
0x024F	7:0	0x00	**HS High Register 0:** HS High Period in terms of PCLK cycles. (Bits [7:0])
0x0250	7:0	0x00	**HS Low Register 1:** HS Low Period in terms of PCLK cycles. (Bits [15:8])
0x0251	7:0	0x00	**HS Low Register 0:** HS Low Period in terms of PCLK cycles. (Bits [7:0])
0x0252	7:0	0x00	**HS Count Register 1:**

			HS pulses per frame. (Bits [15:8])
0x0253	7:0	0x00	**HS Count Register 0:** HS pulses per frame. (Bits [7:0])
0x0254	7:0	0x00	**V2D Register 2:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [23:16])
0x0255	7:0	0x00	**V2D Register 1:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [15:8])
0x0256	7:0	0x00	**V2D Register 0:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [7:0])
0x0257	7:0	0x00	**DE High Register 1:** DE High Period in terms of PCLK cycles. (Bits [15:8])
0x0258	7:0	0x00	**DE High Register 0:** DE High Period in terms of PCLK cycles. (Bits [7:0])
0x0259	7:0	0x00	**DE Low Register 1:** DE Low Period in terms of PCLK cycles. (Bits [15:8])
0x025A	7:0	0x00	**DE Low Register 0:** DE Low Period in terms of PCLK cycles. (Bits [7:0])
0x025B	7:0	0x00	**DE Count Register 1:** DE pulses per frame. (Bits [15:8])
0x025C	7:0	0x00	**DE Count Register 0:** DE pulses per frame. (Bits [7:0])
0x025D	7:0	0x00	**Gradient Increment Register:** Set color gradient increment.
0x025E	7:0	0x00	**CHRK_COLOR_A_L Register:** Checkerboard mode color A low byte. RGB888 color Red (0-255).
0x025F	7:0	0x00	**CHRK_COLOR_A_M Register:** Checkerboard mode color A mid byte. RGB888 color Green (0-255).
0x0260	7:0	0x00	**CHRK_COLOR_A_H Register:** Checkerboard mode color A high byte. RGB888 color Blue (0-255).
0x0261	7:0	0x00	**CHRK_COLOR_B_L Register:** Checkerboard mode color B low byte. RGB888 color Red (0-255).
0x0262	7:0	0x00	**CHRK_COLOR_B_M Register:** Checkerboard mode color B mid byte. RGB888 color Green (0-255).
0x0263	7:0	0x00	**CHRK_COLOR_B_H Register:** Checkerboard mode color B high byte. RGB888 color Blue (0-255).
0x0264	7:0	0x00	**CHRK_RPT_A Register:** Checkerboard mode color A repeat count.
0x0265	7:0	0x00	**CHRK_RPT_B Register:** Checkerboard mode color B repeat count.
0x0266	7:0	0x00	**CHRK_ALT Register:** Checkerboard mode alternate line count.
0x0330	7	0x00	**MIPI PHY Mode Select Register:** Bit 7: Set to force all MIPI clocks running. Used with the MAX96714 internal pattern generator.

Table 41. PLCK Settings Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION
0x0038	1:0	00	**PIN_DRV_EN_0 Register:** Bit [1:0]: Set video pattern PCLK frequency. 00 = 25 MHz default 01 = 75 MHz 10 = PATGEN_CLK_SRC determines PCLK frequency 11 = PATGEN_CLK_SRC determines PCLK frequency
0x01FC	7	1	**PATGEN_CLK_SRC Register:** Select clock source for video pattern on pipe Y. 0 = 150 MHz, 1 = 375 MHz Work together with Pattern CLK Freq register.

Table 42 is a reference table for VPG PCLK selection. This is an alternative method of setting the pattern generator clock frequency with additional options for the PCLK frequency.

Table 42. Video Pattern PCLK Selection

BITFIELD NAME (REG #)		PCLK FREQUENCY
PIN_DRV_EN_0 (0x38 [1:0])	PATGEN_CLK_SRC (0x1FC [7])	
00	X	25 MHz
01	X	75 MHz
1x	0	150 MHz
1x	1 (Default)	375 MHz

Pairing with GMSL1 Serializers

Overview

The MAX96714 are GMSL1 backwards compatible and may be paired with GMSL1 serializers. Unlike GMSL2, the GMSL1 link does not automatically establish, and the forward video channel requires an external video pixel clock at the serializer input to be established.

The following procedure is used to establish a GMSL1 link:

- Build the reverse control channel (the CLINK).
- Program the serializer and image sensor through the reverse channel.
- Program the deserializer following the steps described in the *Configuration* section.
- Enable image sensor streaming and the forward channel.

When using the GMSL1 mode, use only GMSL1 camera serializers equipped with high immunity mode (HIM). HIM provides robust control channel electromagnetic compatibility (EMC) tolerance, which reduces the effects of bit errors and the potential to corrupt reverse channel communication. Devices operating without HIM can fail bulk current injection (BCI) tests and should not be used in applications requiring power-over-coax (PoC).

Other GMSL1 legacy mode serializers without HIM support (e.g., MAX9271) can be used, but require additional design considerations. Building the CLINK without HIM enabled requires more configuration and is sensitive to hardware setup if PoC is used. This is not supported for new designs but may be supported for legacy systems.

A robust reverse communication channel is important when pairing the GMSL2 CSI-2 deserializer with GMSL1 devices. Enable packet-based control channel mode by setting the deserializer *PKTCC_EN* bits high for each link. If there is no I^2C main on the remote side of the link, set *NO_REM_MST* high to reduce unnecessary timeouts on the communication channels.

Pairing with a High Immunity Mode Capable GMSL1 Serializer

A GMSL1 serializer with HIM mode enabled upon power-up (e.g., MAX96705A) is recommended for GMSL1 camera applications with the MAX96714. This is achieved by adding a pull-up resistor at the HIM pin of the serializer. Refer to the MAX96705A data sheet for additional details.

There are two methods to build the CLINK. Power up the GMSL2 CSI-2 deserializer with GMSL1 HIM mode enabled by setting the CFG pins or enable HIM mode with register write *HIGHIMM* (bit 7 in register *0x0B06*).

If a MAX96705A is powered up without HIM enabled, the communication link, CLINK, must be built under non-HIM mode. After the CLINK is built, HIM mode can then be enabled through register writes.

The procedure is listed in *Figure 23* and *Table 43*. Note that in systems with multiple GMSL1 deserializers, before HIM mode is established on both sides, it is recommended to individually enable and program each link to avoid potential I^2C command race conditions. This can be done by only having one deserializer active at a time.

Programming Examples

This example demonstrates how to set up the communication channel for a GMSL 1 link. Note that this requires an I^2C interface capable of writing 8-bit or 16-bit register addresses.

```
# Turn on HIM on MAX96714 Link A
0x98,0x0B06,0xEF
# Turn on local I2C acknowledge as link is not established yet
0x98,0x0B0D,0x80
# Enable CLINK in MAX96705A that is connected on Link A
0x80,0x04,0x43
# Delay 5ms
# Turn off local I2C acknowledge
0x98,0x0B0D,0x00
```

This example shows how to set up a complete GMSL1 link and the required registers for the deserializer. This example assumes that the GMSL1 serializer is in HIM mode at power-up.

```
# Enable GMSL 1
0x98,0x0006,0x40
# Set deserializer link A: HIM Mode Enable
0x98,0x0B06,0xEF
# Turn on local I2C acknowledge as link is not established yet
0x98,0x0B0D,0x80
# At this point users should have access to the serializer I2C communication. If
using the GMSL GUI users need to hit identify devices under options tab.

# Turn on CLINK link and turn video link off.
0x80,0x04,0x43

# Serializer Enable double and HVEN
0x80,0x07,0x84
# Deserializer Enable double and HVEN
0x98,0x0B07,0x84
# Deserializer Disable Remote master
0x98,0x0B05,0x79
# Deserializer Enable SHIFT_VID_HVD
0x98,0x0BA7,0x45
# Disable I2C_LOC_ACK: This should be disabled once the forward channel is
established.
0x98,0x0B0D,0x00
# Deserializer, Set datatype to YUV422. This can be changed depending on the data
type used.
0x98,0x0B96,0x1B
# Deserializer, DE_EN = 0; This is dependent on the camera used.
0x98,0x0B0F,0x01
```

Configuration Flowchart and Detailed Steps

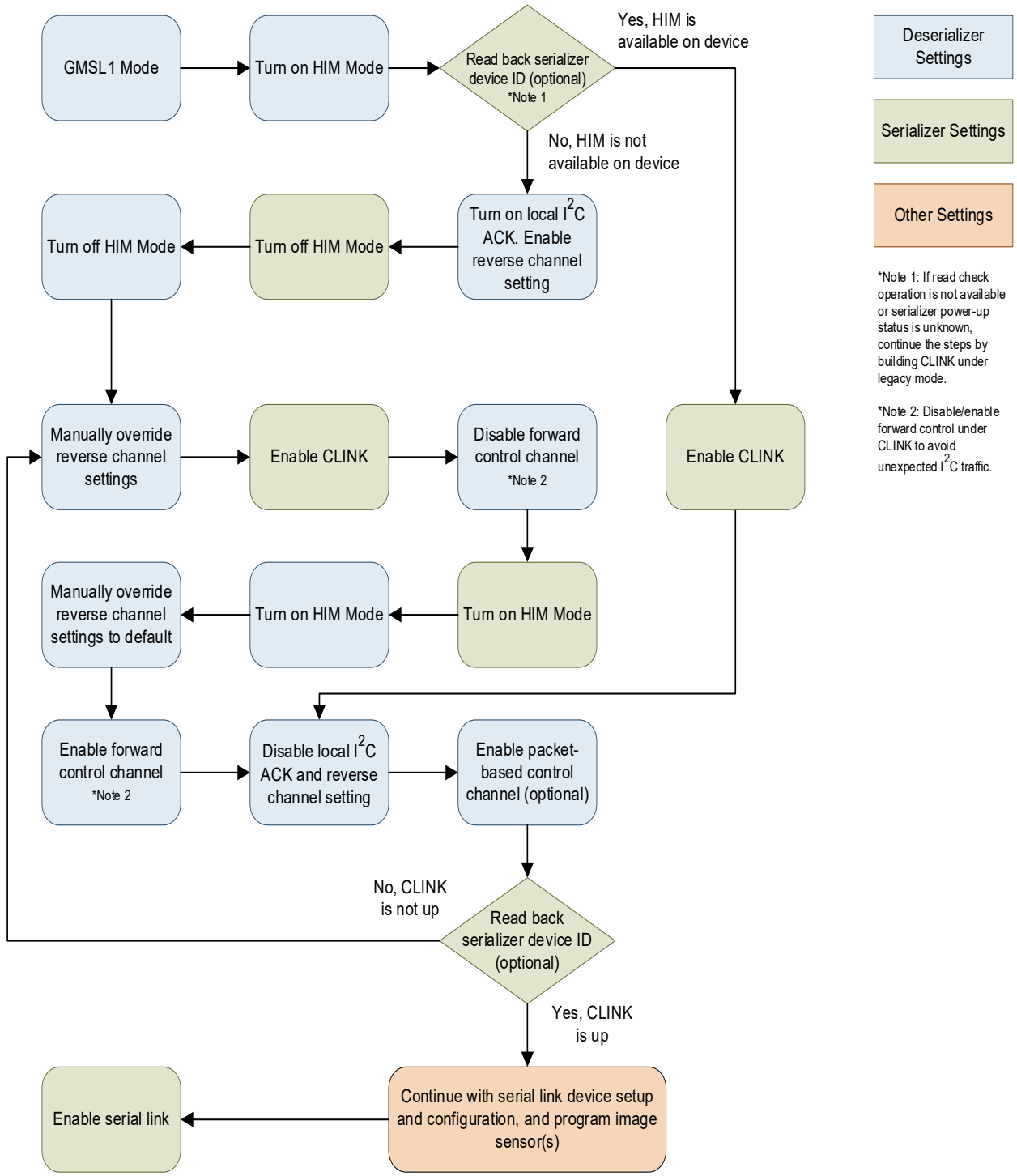

Figure 23 Building the CLINK on HIM-Capable GMSL1 Serializer

Table 43. Configuration Steps for Pairing HIM-Capable GMSL1 Serializers

STEP	DEVICE	SLAVE ID	REGISTER(S)	VALUE	DESCRIPTION
0	DES	0x98	0x0006	0x40	Enable Link A GMSL1 mode.
Delay 5 ms					
1	DES	0x98	0x0313	0x00	Turn off CSI output
2	DES	0x98	0x0B06	0xEF	Turn on HIM mode on Link A *Skip this step if the serializer is not in HIM mode.
Note					If Ser is powered up with HIM by default, CLINK is established with one more write on Ser (0x04=0x43). Proceed to step 17. Otherwise continue.
3	DES	0x98	0x0B0D	0x81	Enable reverse channel configuration on Link A. Turn on local I^2C acknowledge on Link A.
4	SER	0x80	0x4D	0x40	Turn off HIM on Ser.
Delay 10 ms					
5	DES	0x98	0x0B06	0x6F	Turn off HIM mode on Link A.
6	DES	0x98	0x14C5	0xAA	Manually override reverse channel pulse length for Link A.
7	Des	0x98	0x14C4	0x80	Manually override reverse channel rise/fall time for Link A.
8	Des	0x98	0x1495	0xC8	Manually override reverse channel Tx amplitude for Link A.
9	SER	0x80	0x04	0x43	Enable control link only. Disable serial link.
Delay 5 ms					Control link should be built up successfully.
10	DES	0x98	0x0B04	0x02	Disable forward control channel transmitter for Link A.
11	SER	0x80	0x4D	0xC0	Enable HIM mode on Ser (i.e., remote side).
Delay 5 ms					
12	DES	0x98	0x0B06	0xEF	Turn on HIM mode on Link A.
CLINK should be established with HIM enabled.					
13	DES	0x98	0x14C5	0x40	Manually override reverse channel pulse length to default value for Link A.
14	DES	0x98	0x14C4	0x40	Manually override reverse channel rise/fall time to default value for Link A.
15	DES	0x98	0x1495	0x69	Manually override reverse channel Tx amplitude to default value for Link A.
16	DES	0x98	0x0B04	0x03	Enable forward control channel transmitter for Link A.

17	DES	0x98	0x0B0D	0x00	Disable local I^2C acknowledge. Disable reverse channel setting.
18 (Optional)	SER	0x80	0x13	0x42	Allow PKT CC in serializer (MAX96701A/MAX96705A/MAX96715).
Optional	DES	0x98	0x0B08	0x25	Enable packet-based control channel mode for Link A.
Delay 10 ms					
Continue programming the deserializer configuration.					
Continue programming the serializer configuration and camera.					
Delay 10 ms					
19	SER	0x80	0x04	0x83	Enable serial link.
(If HIBW)	SER	0x80	0x07	0bX1XXXXXX	Enable HIBW mode
Delay 10 ms					
(If HIBW)	DES	0x98	0x0B07	0bXXXX1XXX	Enable HIBW mode
Delay 10 ms					
(If HIBW and PKTCC)	DES	0x98	0x0B19	Read	Clear out CC CRC errors due to HIBW switch
Last	DES	0x98	0x0313	0x02	Enable CSI output.

GPIO Forwarding with GPI-GPO and CNTL Pins

In the GMSL1 mode, GPIO forwarding is done using GPI-GPO for the reverse direction and CNTL pins in the forward direction.

Reverse Direction with GPI-GPO

GPI is located at MFP1 and is enabled by default. Disable GPI through register 0xB08. Do not use GPI-GPO forwarding when FSYNC is enabled.

GPI-GPO takes priority on the control channel. I^2C communication is paused through clock stretching, while UART communication gets overwritten and requires the microcontroller to retransmit.

Forward Direction with CNTL

GPIO tunneling in the forward direction is done through the CNTL outputs. The number of available CNTL signals depends on channel settings of the GMSL1 link and is detailed in *Table 44*. Note that while most CNTL outputs of the MAX96714 correspond with the same input on the GMSL1 part, CNTL4 corresponds with the GMSL1 audio bit

Table 44. CNTL Availability and Bandwidth

MODE	CNTL0	CNTL1	CNTL2	CNTL3	CNTL4
BWS = 0, HIBW = 0	None	None	None	None	Full
BWS = 1	None	Full	Full	None	Full
BWS = 0 HIBW = 1	Blank*	Blank*	Blank*	Blank*	None

*In this mode, CNTL pins only change during the blanking period and are intended for slow signals only. Set the triggering modes in the serializer

Note : CNTL operation requires an active video link, and certain register settings to work. The following bits must be set: DBL = 0, HVEN = 0, SEREN = 1, CLINKEN = 0, and PCLK is applied.

The CNTL signals can be reordered to change the output that sent a CNTL signal. By default, the signals are assigned in a straight through manner, that is, the CNTL0 signal outputs from CNTL0, CNTL1 outputs from CNTL1, etc., to change the output order, and change the CNTL OUT ORD bits. E.g., when CNTL_OUT_ORD = 0b001, CNTL3 outputs from the CNTL4 pin, CNTL2 outputs from the CNTL3 pin, etc.

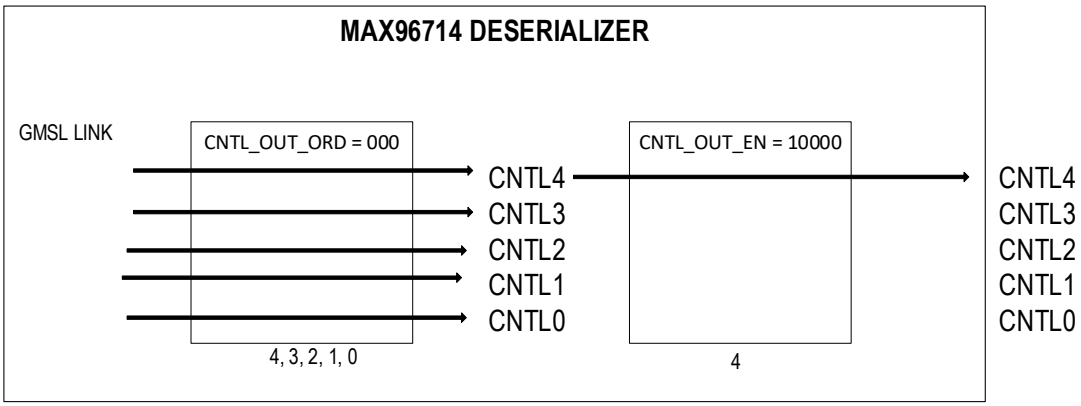

Figure 24. Default CNTL Order, CNTL4 Enabled

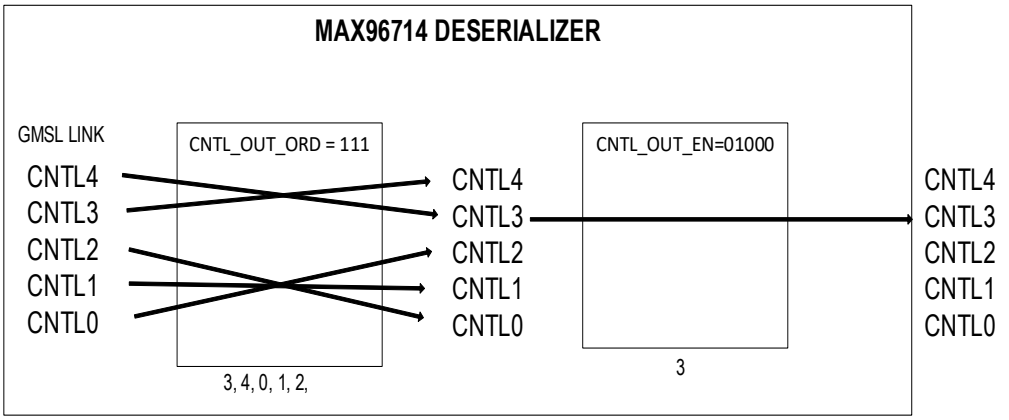

Figure 25. Outputting CNTL4 Signal from CNTL3 Pin

GMSL1 Delay Compensation

By default, the GMSL1 control channel (and thus GPI-GPO) is not a packet-based channel and does not need delay compensation.

If packet-based control channel is used, set GPI_COMP_EN = 1 (in the GMSL1 block) to turn on packet-based GPI-GPO delay compensation.

The CNTL pins are sent on the video path (not the control channel), and do not require delay compensation.

GMSL1 GPI-GPO and CNTL-Related Register Bits

Table 45. Serializer Video PRBS Generator and Checker Registers

REGISTER	BITS	DEFAULT VALUE	DESCRIPTION	DECODE
0xB06	4	0b0	GPI skew compensation enable (when using packet-based control channel).	0b0: GPI Skew compensation not enabled 0b1: GPI Skew compensation
0xBD1	7:5	0b000	Internal CNTL_OUT order to pins CNTL4 to CNTL0. By default, CNTL pins are assigned as shown in the pin description table.	**0b000 4, 3, 2, 1, 0 (Default)** 0b001 3, 2, 1, 0, 4 0b010 2, 1, 0, 4, 3 0b011 1, 0, 4, 3, 2 0b100 0, 1, 2, 3, 4 0b101 1, 2, 3, 4, 0 0b110 2, 3, 4, 0, 1 0b111 3, 4, 0, 1, 2
	4:0	0b00000	CNTL output enable for CNTL[4:0]	0b0XXXX: Disable CNTL4 0b1XXXX: Enable CNTL4 0bX0XXX: Disable CNTL3 0bX1XXX: Enable CNTL3 0bXX0XX: Disable CNTL2 0bXX1XX: Enable CNTL2 0bXXX0X: Disable CNTL1 0bXXX1X: Enable CNTL1 0bXXXX0: Disable CNTL0 0bXXXX1: Enable CNTL0

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 to expect to get a system working with the MAX96714 deserializer and MAX96717 or MAX96705A serializers.

The format of the programming examples throughout this section follow the format allowable by the GMSL GUI for .CPP files to copy them for use immediately.

Table 46. Explanation of GUI Programming for .CPP Files

NUMBER OF I^2C TRANSACTIONS	DEVICE ADDRESS	HIGH REGISTER ADDRESS	LOW REGISTER ADDRESS	REGISTER VALUE	DESCRIPTION
0x04	0x80	0x03	0x83	0x00	//Description

Note: Commands to GMSL2 devices use four transactions (16-bit register address), while GMSL1 devices use three transactions (8-bit register address).

Serializer RCLKOUT to send the clock signal to the image sensor is not included in the scripts. See the *Reference-over-Reverse (RoR)* section for information on how to enable it.

Use Case Example 1 (Pixel Mode)

This example has these characteristics:

- MAX96717 connected to MAX96714.
- Pixel mode
- I^2C address of serializer is 0x80.
- I^2C address of deserializer is 0x98.
- Link rate = 6 Gbps
- Serializer receives RAW12 and Embedded 8 on its MIPI input port with VC = 0.
- The deserializer outputs the data at 1300 Mbps.

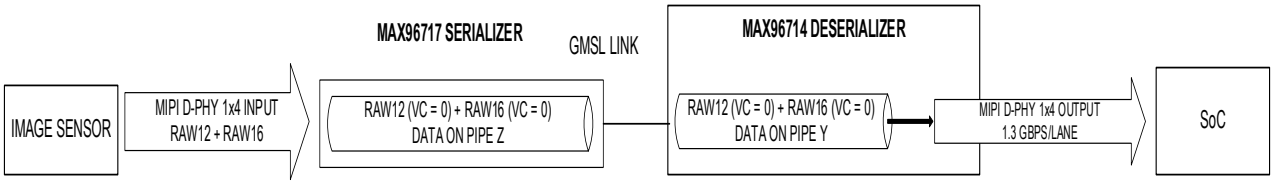

Figure 26. Use Case Example 1

Use Case Example 1 Script

```
// GMSL-A / Serializer: MAX96717 (Pixel Mode) / Mode: 1x4 / Device Address: 0x80 / Multiple-VC Case:
Single VC / Multiple-VC Pipe Sharing: N/A
// PipeZ:
// Input Stream: VC0 RAW12 PortB (D-PHY)
// Input Stream: VC0 EMB8 PortB (D-PHY)

// Deserializer: MAX96714 / Mode: 1x4 / Device Address: 0x98
// PipeY:
// GMSL-A Input Stream: VC0 RAW12 PortB - Output Stream: VC0 RAW12 PortA (D-PHY)
// GMSL-A Input Stream: VC0 EMB8 PortB - Output Stream: VC0 EMB8 PortA (D-PHY)

0x04,0x98,0x03,0x13,0x00, // (CSI_OUT_EN): CSI output disabled
// Link Initialization for Deserializer
0x04,0x98,0x0F,0x00,0x01, // (Default) (LINK_EN_A): Enabled
0x04,0x98,0x00,0x10,0x31, // (Default) (LINK_CFG): 0x1 | (RESET_ONESHOT LINK A): Activated
0x00,0x78,
// Video Transmit Configuration for Serializer(s)
0x04,0x80,0x00,0x02,0x03, // DEV : REG2 | VID_TX_EN_Z (VID_TX_EN_Z): Disabled
//
// INSTRUCTIONS FOR GMSL-A SERIALIZER MAX96717
//
// MIPI DPHY Configuration
0x04,0x80,0x03,0x30,0x00, // MIPI_RX : MIPI_RX0 | (Default) RSVD (Port Configuration): 1x4
0x04,0x80,0x03,0x83,0x00, // MIPI_RX_EXT : EXT11 | Tun_Mode (Tunnel Mode): Disabled
0x04,0x80,0x03,0x31,0x30, // MIPI_RX : MIPI_RX1 | (Default) ctrl1_num_lanes (Port B - Lane Count): 4
0x04,0x80,0x03,0x32,0xE0, // MIPI_RX : MIPI_RX2 | (Default) phy1_lane_map (Lane Map - PHY1 D0): Lane 2 |
(Default) phy1_lane_map (Lane Map - PHY1 D1): Lane 3
0x04,0x80,0x03,0x33,0x04, // MIPI_RX : MIPI_RX3 | (Default) phy2_lane_map (Lane Map - PHY2 D0): Lane 0 |
(Default) phy2_lane_map (Lane Map - PHY2 D1): Lane 1
0x04,0x80,0x03,0x34,0x00, // MIPI_RX : MIPI_RX4 | (Default) phy1_pol_map (Polarity - PHY1 Lane 0):
Normal | (Default) phy1_pol_map (Polarity - PHY1 Lane 1): Normal
0x04,0x80,0x03,0x35,0x00, // MIPI_RX : MIPI_RX5 | (Default) phy2_pol_map (Polarity - PHY2 Lane 0):
Normal | (Default) phy2_pol_map (Polarity - PHY2 Lane 1): Normal | (Default) phy2_pol_map (Polarity -
PHY2 Clock Lane): Normal
// Controller to Pipe Mapping Configuration
0x04,0x80,0x03,0x08,0x64, // FRONTTOP : FRONTTOP_0 | (Default) RSVD (CLK_SELZ): Port B | (Default)
START_PORTB (START_PORTB): Enabled
0x04,0x80,0x03,0x11,0x40, // FRONTTOP : FRONTTOP_9 | (Default) START_PORTBZ (START_PORTBZ): Start Video
0x04,0x80,0x03,0x18,0x6C, // FRONTTOP : FRONTTOP_16 | mem_dt1_selz (mem_dt1_selz): 0x6C
0x04,0x80,0x03,0x19,0x52, // FRONTTOP : FRONTTOP_17 | mem_dt2_selz (mem_dt2_selz): 0x52
0x04,0x80,0x03,0x15,0x80, // (independent_vs_mode): Enabled
0x04,0x80,0x03,0x0D,0x01, // FRONTTOP : FRONTTOP_5 | VC_SELZ_L (VC_SELZ_L): 0x1
// Double Mode Configuration
0x04,0x80,0x01,0x10,0x60, // VID TX  Z : VIDEO TX0 | AUTO BPP (AUTO BPP Pipe Z): Use bpp from bpp
register
0x04,0x80,0x01,0x11,0x50, // VID_TX__Z : VIDEO_TX1 | BPP (BPP Pipe Z): 0x10
0x04,0x80,0x01,0x12,0x08, // VID_TX__Z : VIDEO_TX2 | RSVD (DRIFT_DET_EN Pipe Z): Disable PCLK frequency
drift detection
0x04,0x80,0x03,0x12,0x04, // FRONTTOP : FRONTTOP_10 | bpp8dblz (bpp8dblz): Send 8-bit pixels as 16-bit
0x04,0x80,0x03,0x1E,0x2C, // FRONTTOP : FRONTTOP_22 | soft_bppz (soft_bppz): 0xC | soft_bppz_en
(soft_bppz_en): Software override enabled
// Pipe Configuration
0x04,0x80,0x00,0x5B,0x00, // CFGV__VIDEO_Z : TX3 | TX_STR_SEL (TX_STR_SEL Pipe Z): 0x0

// INSTRUCTIONS FOR DESERIALIZER MAX96714

// Video Pipes And Routing Configuration
0x04,0x98,0x01,0x61,0x00, // (STR_SELY): Link A Stream Id 0
// Pipe to Controller Mapping Configuration
0x04,0x98,0x04,0x4B,0x0F, // (MAP_EN_L Pipe Y): 0xF
0x04,0x98,0x04,0x4C,0x00, // (Default) (MAP_EN_H Pipe Y): 0x0
0x04,0x98,0x04,0x4D,0x2C, // (MAP_SRC_0 Pipe Y DT): 0x2C | (Default) (MAP_SRC_0 Pipe Y VC): 0x0
0x04,0x98,0x04,0x4E,0x2C, // (MAP_DST_0 Pipe Y DT): 0x2C | (Default) (MAP_DST_0 Pipe Y VC): 0x0
0x04,0x98,0x04,0x4F,0x00, // (Default) (MAP_SRC_1 Pipe Y DT): 0x0 | (Default) (MAP_SRC_1 Pipe Y VC): 0x0
0x04,0x98,0x04,0x50,0x00, // (Default) (MAP_DST_1 Pipe Y DT): 0x0 | (Default) (MAP_DST_1 Pipe Y VC): 0x0
0x04,0x98,0x04,0x51,0x01, // (MAP_SRC_2 Pipe Y DT): 0x1 | (Default) (MAP_SRC_2 Pipe Y VC): 0x0
0x04,0x98,0x04,0x52,0x01, // (MAP_DST_2 Pipe Y DT): 0x1 | (Default) (MAP_DST_2 Pipe Y VC): 0x0
0x04,0x98,0x04,0x53,0x12, // (MAP_SRC_3 Pipe Y DT): 0x12 | (Default) (MAP_SRC_3 Pipe Y VC): 0x0
```

```
0x04,0x98,0x04,0x54,0x12, // (MAP_DST_3 Pipe Y DT): 0x12 | (Default) (MAP_DST_3 Pipe Y VC): 0x0
0x04,0x98,0x04,0x6D,0x55, // (MAP_DPHY_DST_0 Pipe Y): 0x1 | (MAP_DPHY_DST_1 Pipe Y): 0x1 |
(MAP_DPHY_DST_2 Pipe Y): 0x1 | (MAP_DPHY_DST_3 Pipe Y): 0x1
// Double Mode Configuration
0x04,0x98,0x04,0x73,0x10, // (ALT2_MEM_MAP8 CTRL1): Alternate memory map enabled
// MIPI DPHY Configuration
0x04,0x98,0x04,0x4A,0xD0, // (Default) (Port A - Lane Count): 4
0x04,0x98,0x03,0x33,0x4E, // (Default) (Lane Map - PHY0 D0): Lane 2 | (Default) (Lane Map - PHY0 D1):
Lane 3 | (Default) (Lane Map - PHY1 D0): Lane 0 | (Default) (Lane Map - PHY1 D1): Lane 1
0x04,0x98,0x03,0x35,0x00, // (Default) (Polarity - PHY0 Lane 0): Normal | (Default) (Polarity - PHY0
Lane 1): Normal | (Default) (Polarity - PHY1 Lane 0): Normal | (Default) (Polarity - PHY1 Lane 1):
Normal | (Default) (Polarity - PHY1 Clock Lane): Normal
// This is to set predefined (coarse) CSI output frequency
// CSI Phy 1 is 1300 Mbps/lane.
0x04,0x98,0x1D,0x00,0xF4,
0x04,0x98,0x03,0x20,0x2D,
0x04,0x98,0x1D,0x00,0xF5,
0x04,0x98,0x03,0x32,0xF4, // (Default) Leave Phy 1 powerd up
0x04,0x98,0x03,0x13,0x02, // (CSI_OUT_EN): CSI output enabled
// Video Transmit Configuration for Serializer(s)
0x04,0x80,0x00,0x02,0x43, // DEV : REG2 | VID_TX_EN_Z (VID_TX_EN_Z): Enabled
```

Use Case Example 2 (Tunnel Mode)

This example has these characteristics:

- One MAX96717 connected to one MAX96714.
- Tunnel mode
- I^2C address of serializers is 0x80.
- I^2C address of deserializer is 0x98.
- Link rate = 6 Gbps
- Serializer receives RAW12 with VC = 0 and RAW16 with VC = 1 on its MIPI input port.
- The deserializer outputs the data on port A at 1300 Mbps.

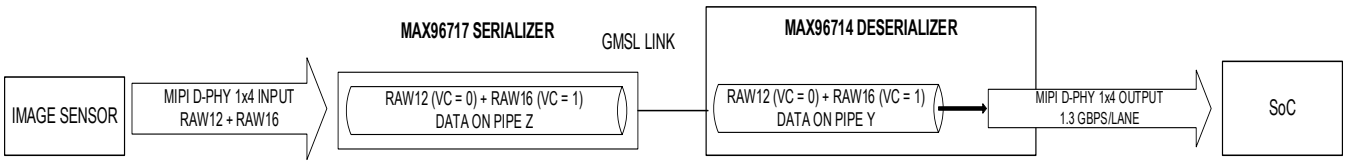

Figure 27. Use Case Example 2

Use Case 2 Example Script

```
// GMSL-A / Serializer: MAX96717 (Tunnel Mode) / Mode: 1x4 / Device Address: 0x80 / Multiple-VC Case:
Frame - Syncronized Line - Interleaved / Multiple-VC Pipe Sharing: Shared Pipe
// PipeZ:
// Input Stream: VC0 RAW12 PortB (D-PHY)
// Input Stream: VC1 RAW16 PortB (D-PHY)

// Deserializer: MAX96714 / Mode: 1 (1x4) / Device Address: 0x98
// PipeY:
// GMSL-A Input Stream: VC0 RAW12 PortB - Output Stream: VC0 RAW12 PortA (D-PHY)
// GMSL-A Input Stream: VC1 RAW16 PortB - Output Stream: VC1 RAW16 PortA (D-PHY)

0x04,0x98,0x03,0x13,0x00, // (CSI_OUT_EN): CSI output disabled
0x04,0x80,0x00,0x02,0x03, // DEV : REG2 | VID_TX_EN_Z (VID_TX_EN_Z): Disabled
// Link Initialization for Deserializer
0x04,0x98,0x00,0x10,0x31, // (RESET_ONESHOT):
0x00,0x78,
//
```

```
// INSTRUCTIONS FOR GMSL-A SERIALIZER MAX96717
//
// MIPI DPHY Configuration
0x04,0x80,0x03,0x30,0x00, // MIPI_RX : MIPI_RX0 | (Default) RSVD (Port Configuration): 1x4
0x04,0x80,0x03,0x83,0x80, // MIPI_RX_EXT : EXT11 | (Default) Tun_Mode (Tunnel Mode): Enabled
0x04,0x80,0x03,0x31,0x30, // MIPI_RX : MIPI_RX1 | (Default) ctrl1_num_lanes (Port B - Lane Count): 4
0x04,0x80,0x03,0x32,0xE0, // MIPI_RX : MIPI_RX2 | (Default) phy1_lane_map (Lane Map - PHY1 D0): Lane 2 |
(Default) phy1_lane_map (Lane Map - PHY1 D1): Lane 3
0x04,0x80,0x03,0x33,0x04, // MIPI_RX : MIPI_RX3 | (Default) phy2_lane_map (Lane Map - PHY2 D0): Lane 0 |
(Default) phy2_lane_map (Lane Map - PHY2 D1): Lane 1
0x04,0x80,0x03,0x34,0x00, // MIPI_RX : MIPI_RX4 | (Default) phy1_pol_map (Polarity - PHY1 Lane 0):
Normal | (Default) phy1_pol_map (Polarity - PHY1 Lane 1): Normal
0x04,0x80,0x03,0x35,0x00, // MIPI_RX : MIPI_RX5 | (Default) phy2_pol_map (Polarity - PHY2 Lane 0):
Normal | (Default) phy2_pol_map (Polarity - PHY2 Lane 1): Normal | (Default) phy2_pol_map (Polarity -
PHY2 Clock Lane): Normal
// Controller to Pipe Mapping Configuration
0x04,0x80,0x03,0x08,0x64, // FRONTTOP : FRONTTOP_0 | (Default) RSVD (CLK_SELZ): Port B | (Default)
START_PORTB (START_PORTB): Enabled
0x04,0x80,0x03,0x11,0x40, // FRONTTOP : FRONTTOP_9 | (Default) START_PORTBZ (START_PORTBZ): Start Video
0x04,0x80,0x03,0x15,0x00, // (Default) (independent_vs_mode): Disabled
// Pipe Configuration Configuration
0x04,0x80,0x00,0x5B,0x02, // CFGV__VIDEO_Z : TX3 | (Default) TX_STR_SEL (TX_STR_SEL Pipe Z): 0x2

// INSTRUCTIONS FOR DESERIALIZER MAX96714

// Video Pipes And Routing Configuration
0x04,0x98,0x01,0x61,0x02, // (Default) (STR_SELY): Link A Stream Id 2
// Double Mode Configuration
// MIPI DPHY Configuration
0x04,0x98,0x04,0x74,0x09, // (Port A Tunnel Mode): Enabled
0x04,0x98,0x04,0x4A,0xD0, // (Default) (Port A - Lane Count): 4
0x04,0x98,0x03,0x33,0x4E, // (Default) (Lane Map - PHY0 D0): Lane 2 | (Default) (Lane Map - PHY0 D1):
Lane 3 | (Default) (Lane Map - PHY1 D0): Lane 0 | (Default) (Lane Map - PHY1 D1): Lane 1
0x04,0x98,0x03,0x35,0x00, // (Default) (Polarity - PHY0 Lane 0): Normal | (Default) (Polarity - PHY0
Lane 1): Normal | (Default) (Polarity - PHY1 Lane 0): Normal | (Default) (Polarity - PHY1 Lane 1):
Normal | (Default) (Polarity - PHY1 Clock Lane): Normal
// This is to set predefined (coarse) CSI output frequency
// CSI Phy 1 is 1300 Mbps/lane.
0x04,0x98,0x1D,0x00,0xF4,
0x04,0x98,0x03,0x20,0x2D,
0x04,0x98,0x1D,0x00,0xF5,
0x04,0x98,0x03,0x13,0x02, // (CSI_OUT_EN): CSI output enabled
0x04,0x80,0x00,0x02,0x43, // DEV : REG2 | VID_TX_EN_Z (VID_TX_EN_Z): Enabled
```

Use Case Example 3 (GMSL1)

This example has these characteristics:

- One MAXCAM with OV10640 connected to one MAX96714.
- GMSL1 mode
- I^2C address of serializer is 0x80.
- I^2C address of deserializer is 0x98.
- Each serializer receives 12-bit parallel data.
- The deserializer assigns a data type of RAW12 and a VC or 0, and outputs the data at 300 Mbps.

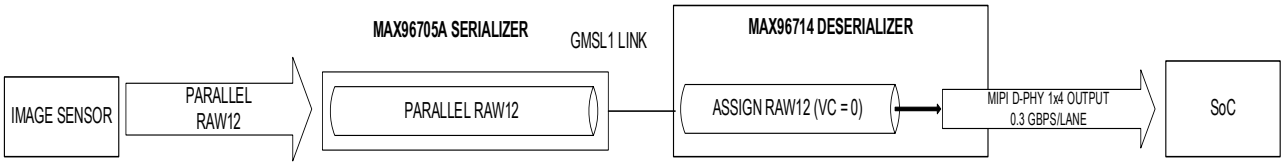

Figure 28. Use Case Example 3

Use Case 3 Example Script

```
0x04,0x98,0x14,0x3F,0x3D,//Errata write 6G
0x04,0x98,0x14,0x3E,0xFD,//Errata write 6G
0x04,0x98,0x14,0xA3,0x30,//Errata write 6G
0x04,0x98,0x14,0xD8,0x07,//Errata write 6G
0x04,0x98,0x14,0xA5,0x70,//Errata write 6G
0x04,0x98,0x14,0x45,0x80,//Errata write SCC disable

0x04,0x99,0x03,0x13,0x00,0x02,//turn off output
0x04,0x99,0x00,0x06,0x00,0xC0,//set GMSL1 mode

0x04,0x99,0x0F,0x05,0x03,0x07,//reduce timeout

0x04,0x99,0x0B,0x05,0x40,0x40,//No masters
0x04,0x99,0x0B,0x0D,0x80,0x80,//Enable LOC ACK
0x04,0x99,0x0B,0x06,0x00,0x80,//Turn off HIM
0x04,0x98,0x14,0xC5,0xAA,//Change pulse length
0x04,0x99,0x14,0xC4,0x80,0xF0,//Change tRF
0x04,0x99,0x14,0x95,0xC8,0xBF,//change amplitude

0x03,0x80,0x04,0x43,//enable CLINK
0x00,0x0A,//delay A
0x04,0x99,0x0B,0x04,0x02,0x03,//disable return CC
0x03,0x80,0x4D,0xC0,//set HIGHIMM
0x04,0x99,0x0B,0x06,0x80,0x80,//set HIGHIMM
0x04,0x99,0x14,0xC5,0x00,0x80,//Set to default
0x04,0x99,0x14,0xC4,0x00,0x80,//Set to default
0x04,0x99,0x14,0x95,0x00,0x80,//Set to default
0x04,0x99,0x0B,0x04,0x03,0x03,//enable return CC
0x04,0x99,0x0B,0x0D,0x00,0x80,//disable LOC ACK
0x03,0x81,0x13,0x40,0x40,//allow PKT CC
0x04,0x99,0x0B,0x08,0x04,0x04,//set PKTCC
0x00,0x0A,//delay A

0x03,0x81,0x07,0x84,0xA4,//set link A SER (need to clear det errors and CC CRC errors)
0x00,0x05,//delay 5
0x04,0x99,0x0B,0x07,0x84,0xA4,//set link A DES
0x00,0x0A,//delay A

0x04,0x99,0x0B,0xA7,0x40,0x40,//SHIFT_VID_HVD = 1 for GMSL1
0x04,0x99,0x0B,0x06,0x07,0x07,//SET HVSRC for the 96705
0x04,0x99,0x0B,0x0F,0x00,0x08,//SET DE_EN 96705
0x04,0x99,0x03,0x11,0x00,0x20,//SET DE_SEL 0
0x04,0x99,0x0B,0x96,0x3A,0xFA,//SET mapping for DT 0x2C
0x04,0x99,0x03,0x22,0x00,0x20,//SET MUX MODE 0

0x04,0x99,0x03,0x19,0x0C,0x1F,//BPP = 0xC in [4:0]
0x04,0x99,0x03,0x14,0x00,0xF0,//VC = 0 in [7:4]
0x04,0x99,0x03,0x16,0x80,0xC0,//DT = in [7:6]
0x04,0x99,0x03,0x17,0x0C,0x0F,//DT = in [3:0]
0x04,0x99,0x03,0x1D,0x80,0x80,//Enable override Y

0x04,0x98,0x04,0x4B,0x07,//turn on mappers
0x04,0x98,0x04,0x4D,0x00,//FS
0x04,0x98,0x04,0x4E,0x00,
0x04,0x98,0x04,0x4F,0x01,//FE
0x04,0x98,0x04,0x50,0x01,
0x04,0x98,0x04,0x51,0x2C,//DT = 2C
0x04,0x98,0x04,0x52,0x2C,
0x04,0x98,0x04,0x6D,0x15,//To PHY 1

0x04,0x99,0x1D,0x00,0x00,0x01,//disable PLL 1
0x04,0x99,0x03,0x20,0x03,0x1F,//set Predef

0x04,0x99,0x1D,0x00,0x01,0x01,//Enable PLL 1

0x04,0x99,0x03,0xE2,0x81,0xE0,//set FSYNC Controller for single
0x04,0x98,0x03,0xE5,0x78,//Set FSYNC period 29.97Hz LSB
0x04,0x98,0x03,0xE6,0xBA,//Set FSYNC period 29.97Hz
```

```
0x04,0x98,0x03,0xE7,0x0C,//Set FSYNC period 29.97Hz MSB
0x04,0x98,0x03,0xEA,0x00,//disable FSYNC overlap window
0x04,0x99,0x03,0xEB,0x00,0x1F,//disable FSYNC overlap window
0x04,0x99,0x03,0xEF,0x41,0xDF,//FSYNC for some pipes, use XTAL, GMSL1 FSYNC
0x04,0x99,0x0B,0x08,0x30,0x10,//Enable FSYNC on A

0x04,0x99,0x03,0xE0,0x04,0x0F,//enable manual FSYNC

#****************************************************************/
#           Image Sensor Setup              */
#****************************************************************/
//program image sensor here
//enable image sensor

0x03,0x81,0x04,0x80,0xC0,//Start links set (need to clear CC CRC errors)
0x00,0x0A,//delay A
0x03,0x81,0x07,0x40,0x40,//set link A SER HIBW(need to clear CC CRC errors)
0x00,0x05,//delay 5
0x04,0x99,0x0B,0x07,0x08,0x08,//set link A DES HIBW (need to clear CC CRC errors)
0x00,0x0A,//delay A

0x04,0x99,0x0B,0x19,0x00,0x00,//Clear CC CRC errors

0x04,0x99,0x03,0x13,0x02,0x02,//turn on output
```

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION
0	05/2023	Initial release