AHEAD OF WHAT'S POSSIBLE™

MAX96724/F/R Users Guide

GMSL SerDes Applications Team

Revision 2

MAR 1 , 2023

Table of Contents

MAX96724/F/R Quad Deserializers

Document Overview

The “MAX96724/F/R Deserializer Users Guide” is meant to be used as an overview of MAX96724 deserializer and to facilitate using the device one function at a time. Each section contains a topic or feature overview, a list of relevant registers, and finally an example if how to program the feature by register writes. In addition to this document users may need to refer to other Gigabit Multimedia Serial Link (GMSL) documentation to successfully bring up a complex system. Important documentation resources include the following: “GMSL2 Hardware Design Guide”, “GMSL2 Users Guide”, “GMSL2 Channel Specification”, device datasheets, and companion serializer users guides.

Please reference these documents for additional GMSL information or features that are not covered by this document. Throughout this document users will find programming examples to help setup specific features. Within these examples the deserializer address is assumed to be 0x4E unless it is specifically changed as part of the example.

Device Overview

GMSL2 CSI-2 quad deserializers receive data from up to four separate GMSL1 or GMSL2 serializers over 50Ω coax or 100Ω shielded twisted pair (STP) cables. The combined image data is converted to MIPI CSI-2 (D-PHY v1.2 up to 2.5Gbps/lane or C-PHY v1.0 up to 5.7Gbps/lane) and can be aggregated to any of the output MIPI ports.

MAX96724 quad deserializers operate each GMSL2 link at a fixed rate of 3Gbps or 6Gbps in the forward direction and 187.5Mbps in the reverse direction; the MAX96724F and MAX96724R can operate GMSL2 links at 3Gbps in the forward direction and 187.5Mbps in the reverse direction. All devices are GMSL1 backwards compatible and can pair with first-generation GMSL1 serializers running at rates up to 3.12Gbps (GMSL1 serial link data rate is variable and application-dependent). All MAX96724 variants support the reference over reverse channel (RoR) feature with compatible GMSL2 serializers. In reference over reverse channel mode the deserializer sends a reference clock across the GMSL2 link to each connected serializer. This removes the need for an external crystal for each serializer.

Application Use Case

Each quad deserializer link is completely independent. This independence allows for flexibility in mixing GMSL data rates, GMSL generations, cable types, and GMSL transmit modes. All links may be configured differently, or identically based on system requirements. The simplest and most common application for quad deserializers is the four-camera link application. Figure 1 represents the four-camera application with mixed GMSL links.

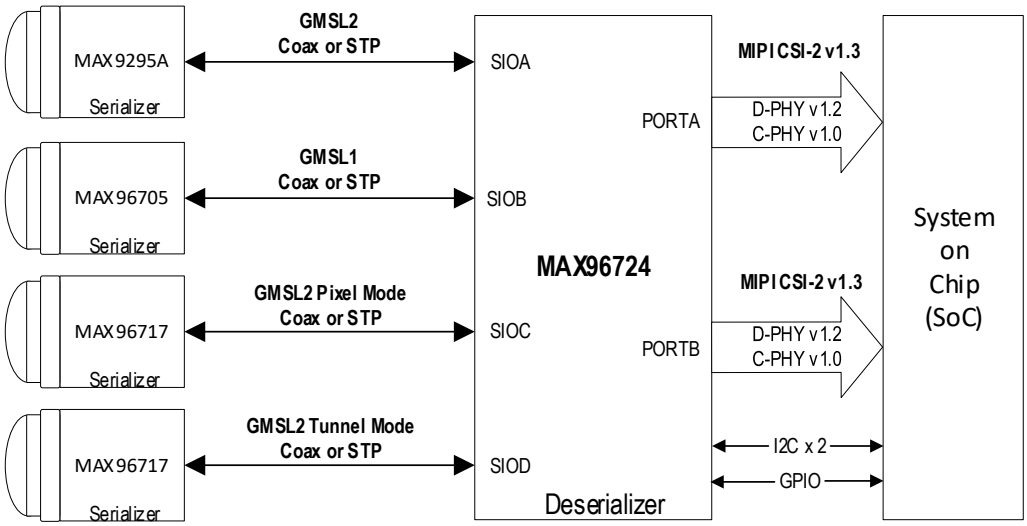

Figure 1. MAX96724, Four Camera Application

Architecture

The following block diagrams show the video path within the GMSL2 CSI-2 quad deserializer D-PHY (Figure 2 and Figure 3) and C-PHY (Figure 4 and Figure 5) architectures.

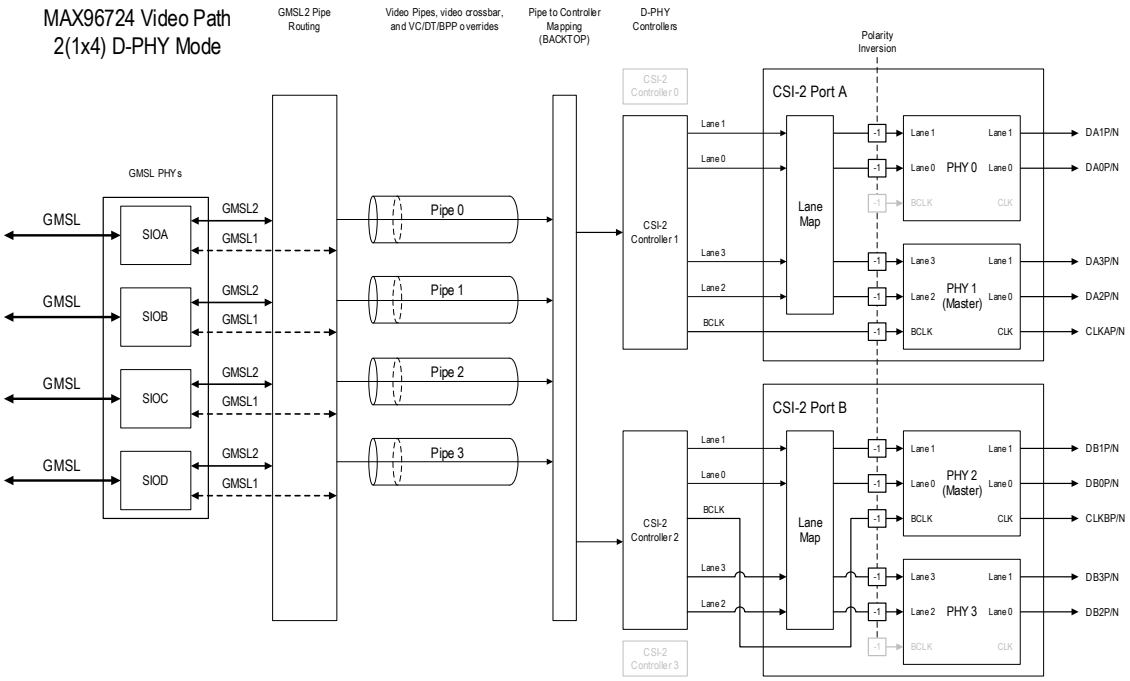

Figure 2. MAX96724 Video Path - 2x4 D-PHY Mode

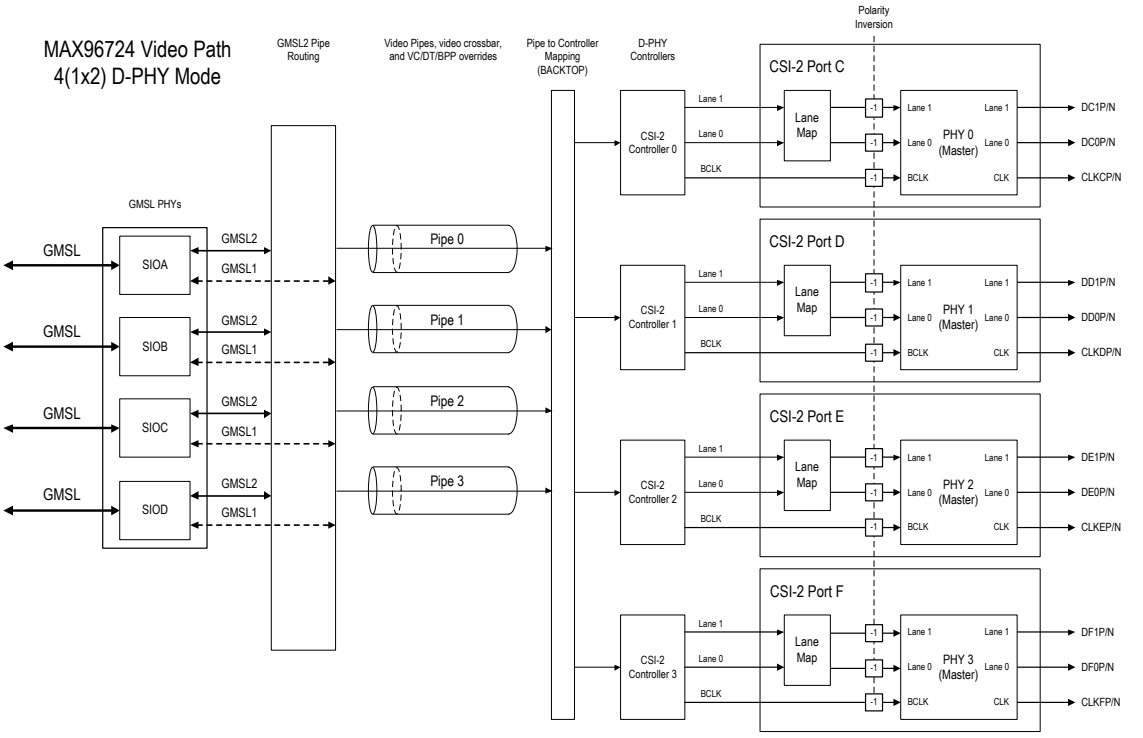

Figure 3. MAX96724 Video Path – 4x2 D-PHY Mode

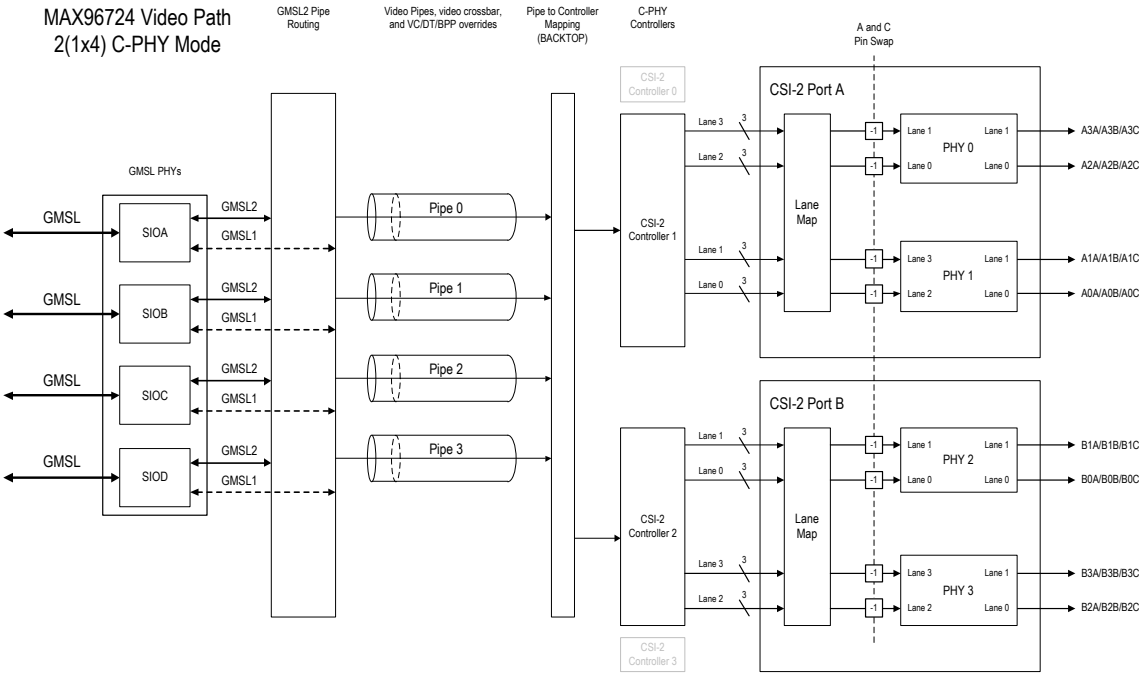

Figure 4. MAX96724 Video Path – 2x4 C-PHY Mode

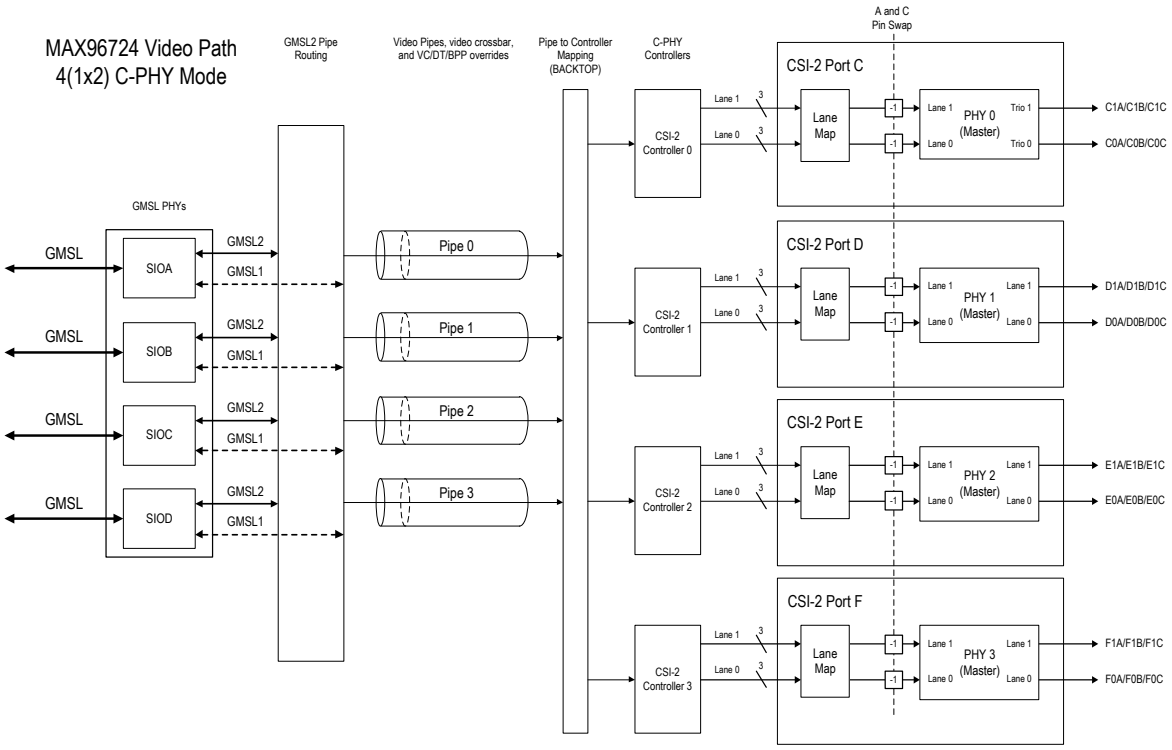

Figure 5. MAX96724 Video Path – 4x2 C-PHY Mode

Start up and Programming Sequence

Overview

The quad deserializers cover many use cases and features that work in conjunction with each other. To avoid feature and system sequencing issues, the best-case startup practices are outlined in Table 1. Features or configuration changes may not be required and may be skipped in the startup sequence. This will depend on system requirements and data configurations.

Recommended Startup Sequence

Table 1. MAX96724/F/R Startup Sequence

MAX96724/F/R Startup and Programming Sequence		
General Device Settings		
Sequence Number	**Feature Enable Order**	**Notes**
*	**Important note regarding configuration changes.	Any configuration should be done before video is running. Video must be stopped if a configurations change is required.
*	Voltage supply ramp order is independent.	No voltage sequencing required
1	Configuration Pins	Starting point (reboot if changed)

2	I2C wake time	From local I2C transaction
3	GMSL wake up	From remote Serializer
4	Main I2C configuration and CRC	Changes (e.g., CRC ARQ)
5	I2C pass through config. and CRC	Changes (e.g., CRC ARQ)
6	Interrupt handling (ERRB)	Turn on and off error reporting.
7	I2C crossover setup	
8	RESET_LINK = 1	Used for GMSL2 links only
9	Set GMSL profile registers	
10	Set MIPI profile registers	
11	Retransmission of reverse channel packets during CRC error	
12	GPIO configuration and CRC	
13	V-sync and H-sync output config.	
14	SSC configuration and enable	
Video Routing and Configuration		
Sequence Number	**Feature Enable Order**	**Notes**
*	Video pipe configuration	Controller-to-Pipe Mapping bpp manipulations Pixel Datatype and Virtual Channel Routing Enable/Disabling Heartbeat Mode Stream ID selection Video Duplication
15	Video pipe to link selection	8:1 Mux mapping Serializer X, Y, Z, U
16	Line CRC (32-bit)	Enabled by default
17	Video Pixel CRC (16-bit)	Disabled by default
18	VID_EN	Enabled by default
19	VC/DT override	
20	VC/DT mapping	
21	Pixel doubling for efficiency	Dependent on bpp
22	ERRB Forwarding	Disable/Enable ERRB Forwarding packet
23	MIPI Error Packet	Disable/Enable MIPI Error packet
24	Crossbar switch	
25	Aggregation (FCFS)	
26	Aggregation (W x 4H, W x 4H)	
27	MIPI Replication Setup	
28	MIPI C-PHY setup	Lane Map, count, swap, and polarity Inversion
29	MIPI D-PHY setup	Lane Map, count, swap, and polarity Inversion
30	MIPI D-PHY Deskew	(Required above 1.5Gbps/lane)
31	MIPI D-PHY Tx Config	CSIPLL frequency (Data Rate)
32	MIPI C-PHY Tx Config	CSIPLL frequency (Data Rate)

33	FSYNC setup	Configure but do not start.
GMSL2 Setup		
Sequence Number	**Feature Enable Order**	**Notes**
34	GMSL link configuration	Link Rate, Coax/STP
35	RESET_LINK = 0	Used for GMSL2 mode only
36	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.
37	AEQ	
38	SYNC Lock	
39	Lock Info Frames	
40	LINK Lock	
41	Eye opening monitor/mapper	
GMSL1 Setup		
Sequence Number	**Feature Enable Order**	**Notes**
42	Enable PKTCC and HIM mode	
43	GMSL1 specific errors and mapping.	
44	Color Format Mapping to GMSL2	
45	CNTL pin mapping	
46	GMSL1 FSYNC setup	
47	Set I2C local ACK	
48	Start CLINK from unlocked channel	
General Device Settings Continued		
Sequence Number	**Feature Enable Order**	**Notes**
49	Line fault setup	
50	MIPI Video PRBS or Pattern Generator Setup (if used)	
51	START VIDEO FROM SOURCE	Start camera or image sensor
52	Start FSYNC	
53	Start Serializer link	(GMSL1 only)
54	Disable I2C Local ACK	(GMSL1 only)
55	Verify PCLK detect (Serializer)	
56	Verify video lock detected	
57	No LCRC or video memory errors detected	
58	csi2_tx1_pkt_cnt detected	Verify deserializer output
59	phy_pkt_cnt detected	Verify deserializer output

Configuration

Overview

The GMSL2 CSI-2 quad deserializer must be configured before receiving input from the serial link. **Dynamic configuration is not supported.**

The forward video path of GMSL2 CSI-2 quad deserializers is configured with the following programming:

- Pixel and Tunnel Mode
- Link Initialization
- Link Lock Check
- Video Pipe Selection

Only after the video path is configured should video be enabled. The following sub-sections detail the operation of each of these steps with descriptions of relevant registers and programming examples.

Pixel and Tunnel Mode

The devices support both pixel and tunnel modes for GMSL2 links. Each link may be configured separately to either pixel or tunneling mode. Pixel mode and tunnel mode both support data aggregation, but data transmitted with pixel mode cannot be aggregated with data transmitted with tunnel mode.

Pixel mode provides the ability for systems to manipulate data types, bits per pixel, and virtual channels. This mode can be used when the incoming data must be manipulated before outputting to the selected MIPI transmitters.

Tunnel mode can be used 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 tunnel mode, data may not be changed as it is protected with an end-to-end CRC and is passed from serializer to deserializer without any manipulation. In tunnel mode, any combination of data type, bpp, and virtual channel may be transmitted if the video bandwidth total does not exceed the link bandwidth.

By default, the devices will match the mode of the connected serializer for each link automatically through the automatic tunnel detection feature. While in tunneling mode it can also automatically detect and convert D-PHY signal inputs on the serializer to C-PHY signal outputs on the deserializer. Users may additionally turn off the tunneling mode auto detection features if they wish to program the device manually.

Tunnel Mode Configuration Registers

Table 2. Tunnel Mode Register Table

Register	Bits	Default Value	Description
0x0936, 0x0976, 0x09B6, 0x09F6	4:3 and 0	0x08	**Tunnel Mode Enable and CPHY Lane Count Register:** Bits [4:3]: Manual selection for CPHY Lane count in tunnel mode. Bit 0: Tunnel Mode Enable.

0x0939, 0x0979, 0x09B9, 0x09F9	7:1	0x10	**Tunnel Mode Auto Detections and Tunnel Mode Pipe to PHY Mapping:** Bit 7: Disable Automatic detection of SER lane count. 0 = Automatic detection Enabled. 1 = Automatic detection Disabled. Bit 6: Disable Automatic detection of tunneling mode. 0 = Automatic detection enabled. 1 = Automatic detection disabled. Bit [5:4]: TUN_DEST, Pipe to PHY mapping. 00 = PHY0 01 = PHY1 10 = PHY2 11 = PHY3 Bit 2: Only used when TUN_DPHY_TO_CPHY_CONV_OVRD = 1. Manual override bit to enable DPHY to CPHY conversion in tunneling mode (must also disable auto tunneling mode detect DIS_AUTO_TUN_DET = 1 and manually set tunneling mode TUN_EN = 1). Bit 1: TUN_DPHY_TO_CPHY_CONV register to override automatic detection.
0x1260	7 and 3:0	0x00	**Auto Tunnel Detection Flags:** Bit 7: CPHY_MODE_OVRD_EN, in Wx4H mode each pipe uses values set in BACKTOPx_CPHY_MODE_OVRD register instead of detected (BACKTOPx_CPHY_MODE_DET). Bit [3:0]: Auto-Tunneling mode detection flags. 1 = Tunnel mode detected. 0 = Tunnel mode not detected.

Tunnel Mode Configuration Programming Examples

```
# Enable Tunnel mode manually. TUN_EN = 1.
0x4E, 0x936, 0x09
0x4E, 0x976, 0x09
0x4E, 0x9B6, 0x09
0x4E, 0x9F6, 0x09
# Disable auto tunneling detection and set tunneling destination to PHY1.
0x4E, 0x939, 0x50
0x4E, 0x979, 0x50
0x4E, 0x9B9, 0x50
0x4E, 0x9F9, 0x50
# Reset one-shot to reset the link
0x4E, 0x018, 0x0F
```

Link Initialization

Link initialization establishes the device link modes and link speeds. The GMSL2 CSI-2 quad deserializers have four independent GMSL links A, B, C, and D and each may be configured differently. They accept any combination of GMSL1 and GMSL2 inputs and link rates. The deserializers are powered up in either GMSL1 or GMSL2 mode for all links through CFG1 pin. After initial power-up, link modes can be individually reconfigured by the GMSL profiles from the GMSL and MIPI Profile Tables. They may additionally be adjusted without the use of profiles using the link configuration register annotated in Table 3. The GMSL link rate, GMSL type, and COAX or STP may be selected. Any changes to the GMSL link should be followed by a link RESET_ONESHOT to reinitialize the link.

A link reset on CSI-2 quad deserializers resets the entire data path of any video connected to the reset PHY. This behavior is different than other GMSL2 deserializers. Link resets should not be used when data is running through the deserializer.

Link Initialization Registers

Table 3. Link Initialization Register Table

Register	Bits	Default Value	Description
0x0006	7:0	0xFF	**GMSL Link/PHY Enable and Mode Select Register:** Bit 7: PHY D, 0 = GMSL1, 1 = GMSL2. Bit 6: PHY C, 0 = GMSL1, 1 = GMSL2 Bit 5: PHY B, 0 = GMSL1, 1 = GMSL2 Bit 4: PHY A, 0 = GMSL1, 1 = GMSL2 Bit 3: PHY D, 0 = Link Disable, 1 = Link Enable Bit 2: PHY C, 0 = Link Disable, 1 = Link Enable Bit 1: PHY B, 0 = Link Disable, 1 = Link Enable Bit 0: PHY A, 0 = Link Disable, 1 = Link Enable
0x0010	7:0	0x22 (Depends on CFG pins)	**GMSL Link/PHY Rate Select Register:** Bits [7:6]: Tx Rate on Link B Bits [5:4]: Rx Rate on Link B Bits [3:2]: Tx Rate on Link A Bits [1:0]: Rx Rate on Link A Tx Rate (Transmitter rate; i.e., reverse channel): 00 = 187.5Mbps Rx Rate (Receiver rate; i.e., forward channel): 01 = 3Gbps 10 = 6Gbps

0x0011	7:0	0x22 (Depends on CFG pins)	**GMSL Link/PHY Rate Select Register:** Bits [7:6]: Tx Rate on Link D Bits [5:4]: Rx Rate on Link D Bits [3:2]: Tx Rate on Link C Bits [1:0]: Rx Rate on Link C Tx Rate (Transmitter rate; i.e., reverse channel): 00 = 187.5Mbps Rx Rate (Receiver rate; i.e., forward channel): 01 = 3Gbps 10 = 6Gbps
0x0018	7:0	0x00	**GMSL Link Reset Register:** * Bits [7:4]: Link reset register for each link D/C/B/A 0 = Release link reset 1 = Activate link reset Bits [3:0]: One-shot link reset for each link D/C/B/A 0 = No action 1 = Reset data path (self-clear)
0x0022	7:0	0xFF	**GMSL Cable Type and Polarity:** Bit 6: CXTP_D 0 = Shielded Twisted Pair 1 = Coax cable Bit 4: CXTP_C 0 = Shielded Twisted Pair 1 = Coax cable Bit 2: CXTP_B 0 = Shielded Twisted Pair 1 = Coax cable Bit 0: CXTP_A 0 = Shielded Twisted Pair 1 = Coax cable

Note: A link reset on CSI-2 quad deserializers resets the entire data path of any video connected to the reset PHY. Link resets should not be used when data is running in the device. Doing this may corrupt data and have unintended consequences.

Link Initialization Programming Example

Turn on all links and set links A&B to 6Gbps and links C&D to 3Gbps GMSL link rate.

```
# Turn on all links in GMSL2 mode.
0x4E, 0x0006 ,0xFF
# Set 6Gbps/187.5Mbps for link A and B
0x4E, 0x0010, 0x22
# Set 3Gbps/187.5Mbps for link C and D
0x4E, 0x0011, 0x11
# Reset one-shot to reset the link
0x4E, 0x018, 0x0F
```

Link Lock Check

Pin #13 (MFP4) is used as LOCK indication by default. It is open-drain and will assert only when all enabled GMSL2 links are locked. Unused links should be disabled on order to obtain the lock indicator and to facilitate data output (set by LINK_EN_A, LINK_EN_B, LINK_EN_C, and LINK_EN_D bits in register 0x0006). The following are lock indication register bits for each individual link:

- **GMSL1 Mode**
  - Register 0x0BCB, 0x0CCB, 0x0DCB or 0x0ECB bit 0 will assert if link A, B, C, or D is locked in GMSL1 mode.
- **GMSL2 Mode**
  - Register 0x001A, 0x000A, 0x000B, or 0x000C bit 3 will assert if link A, B, C, or D is locked in GMSL2 mode.

When pairing with GMSL2 serializers in 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 25MHz crystal. The link clock does not have any relationship with the input data pixel clock. In 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.

Video Pipe Selection and Replication

Video pipes must be configured to match the video streams received from connected serializers. This programming step, typically performed following link initialization, ensures that the GMSL2 CSI-2 quad deserializer properly receives video data from serializers. There are four video pipes in total (video pipes 0–3). Video pipe replication is possible by routing the same serializer pipe data to multiple deserializer pipes. GMSL2 CSI-2 quad deserializers can also receive video streams from a serializer in splitter mode; however, video streams must be routed to separate links. Figure 6 shows the relationship between GMSL1 and GMSL2 data and the video pipes.

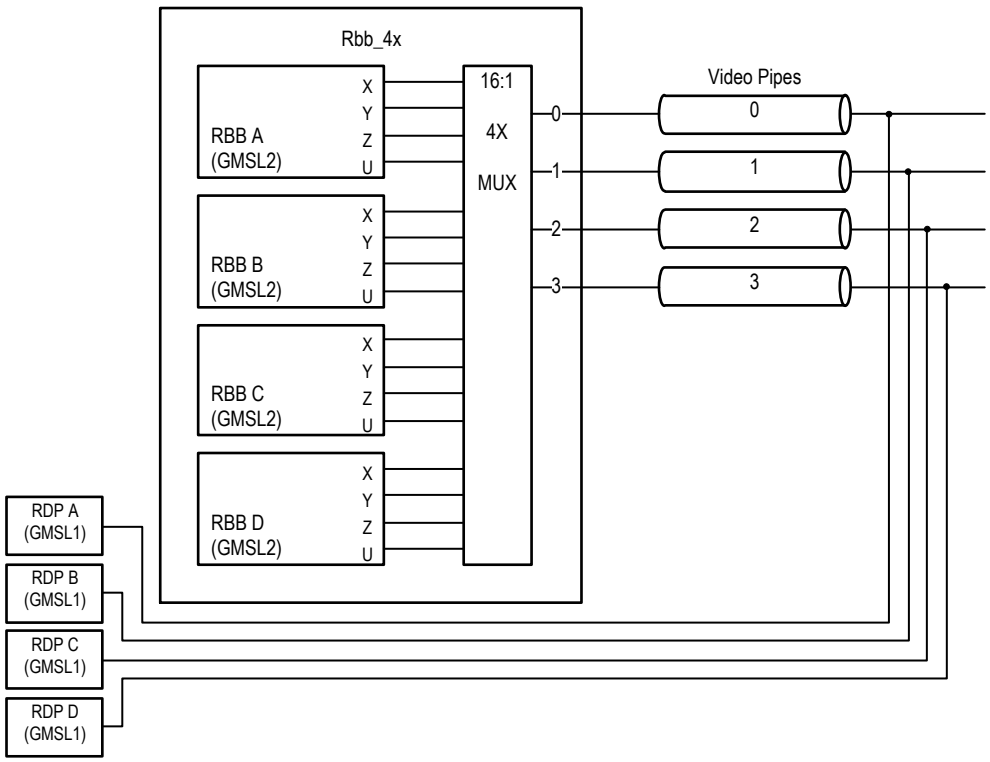

Figure 6. Video Pipe Selection Block Diagram

GMSL1 Mode Video Pipes

By default, video data received from PHY A, B, C, and D is output after video pipes 0, 1, 2, and 3, respectively. In GMSL1 mode, video pipe selection is not required and are enabled by default; however, at least one video pipe must be enabled if others are turned off.

GMSL2 Mode Video Pipes

Each deserializer video pipe may select the GMSL link A-D, and the serializer stream ID to grab the video data from. The 16:1 MUX featured in Figure 6. Video Pipe Selection Block Diagram must be configured to match the video streams (STR_ID) from 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. If the same GMSL link and pipe is selected for multiple deserializer pipes, replication of the video stream will occur.

Stream Select All

The stream select all feature of the devices automatically selects the active serializer pipes once data video is streaming. This will not only select one pipe as a serializer video source, but all active pipes. This feature can be disabled by the STREAM_SEL_ALL bit in register 0x00F4. For complex video streams containing multiple video pipes per serializer, it is required to manually select the streams for each pipe individually.

Video Pipe Selection Registers

Table 4. Video Pipe Selection Register Table

Register	Bits	Default Value	Description
0x00F0	7:0	0x62	**Video Pipe Select Register:** Bits [7:6]: Link Selection for Pipe 1 Bits [5:4]: Stream ID Selection for Pipe 1 Bits [3:2]: Link Selection for Pipe 0 Bits [1:0]: Stream ID Selection for Pipe 0 GMSL Link Selection: 00 = Link A 01 = Link B 10 = Link C 11 = Link D Serializer Stream ID Selection: 00 = Stream 0, default serializer Pipe X 01 = Stream 1, default serializer Pipe Y 10 = Stream 2, default serializer Pipe Z 11 = Stream 3, default serializer Pipe U
0x00F1	7:0	0xEA	**Video Pipe Select Register:** Bits [7:6]: Link Selection for Pipe 3 Bits [5:4]: Stream ID Selection for Pipe 3 Bits [3:2]: Link Selection for Pipe 2 Bits [1:0]: Stream ID Selection for Pipe 2 GMSL Link Selection: 00 = Link A 01 = Link B 10 = Link C 11 = Link D Serializer Stream ID Selection: 00 = Stream 0, default serializer Pipe X 01 = Stream 1, default serializer Pipe Y 10 = Stream 2, default serializer Pipe Z 11 = Stream 3, default serializer Pipe U
0x00F4	4:0	0x1F	**Video Pipe Enable Register:** Bit 4: Stream Select All 0 = Disable; 1 = Enable Bit 3: Enable video pipe 3 Bit 2: Enable video pipe 2 Bit 1: Enable video pipe 1 Bit 0: Enable video pipe 0 0 = Disable; 1 = Enable

Video Pipe Selection Programming Examples

This example sets up the deserializer video pipes 0-3 to have link A and B video streams.

```
# Enable pipe 0 for link A stream ID 0, pipe 1 for link B stream ID 0
0x4E,0x00F0,0x40
# Enable pipe 2 for link A stream ID 1, pipe 3 for link B stream ID 1
0x4E,0x00F1,0x51
# Turn on pipe 0, 1, 2 and 3
0x4E,0x00F4,0x0F
```

Replicating PHY Data Using PHY Copy

GMSL2 CSI-2 quad deserializers also support MIPI PHY-to-PHY replication. This is configured by enabling PHY copy and setting the source and destination PHYs with phy_cp0[4:0] and phy_cp1[4:0] in registers 0x08A9 and 0x08AA, respectively. Table 5 contains MIPI PHY replication registers. All PHY settings including lane count, MIPI CSI-2 type, and MIPI rate must be the same from the source to the destination. The entire PHY and data will be copied from the source PHY when this feature is enabled.

Note: Two replications can be enabled at the same time in x2 mode (see *MIPI PHY Settings* for details).

Table 5. MIPI PHY Replication Registers

Register	Bits	Default Value	Description
0x08A9	7:3	00000	**PHY copy Register:** Bit 7: 0 = Disable, 1 = Enable PHY0 copy Bits [6:5]: PHY copy destination. Bits [4:3]: PHY copy source.
0x08AA	7:3	00000	**PHY copy Register:** Bit 7: 0 = Disable, 1 = Enable PHY1 copy Bits [6:5]: PHY copy destination. Bits [4:3]: PHY copy source.

Video Lock Check

Register 0x01DC, 0x01FC, 0x021C, 0x023C bit 0 (VIDEO_LOCK) for each of the 4 video pipes will assert if it is receiving valid video data. VIDEO_LOCK in GMSL1 mode is equivalent to Link Lock.

GMSL and MIPI Profiles

Overview

The quad deserializers provide both GMSL and MIPI profiles that assist users to setup common usages quicky and efficiently. The use of these profiles can reduce device setup time, device register accesses, and device programming complexity. Each GMSL profile will configure many device registers when selected, and additional configuration is permitted after a profile selection. They may be used to get close to a final configuration with minimal additional register writes.

GMSL profiles can be selected for each link independently and may change the link rate, GMSL link types, and cable connection type. They are most effective in systems that use mixed links, such as GMSL1 and GMSL2, or links of different GMSL data rates.

MIPI profiles may be used to program common use cases for the deserializer or to reduce the deserializer programming for complex use cases. The devices each contain 16 unique MIPI profiles that can set the MIPI physical configuration, the MIPI type (D-PHY or C-PHY), the MIPI data rate, frame synchronization, video data replication, aggregation, and GMSL1 data type conversion.

During deserializer programming, the GMSL profiles for each link should be set before proceeding to the MIPI profile selection. The MIPI profile should be selected to directly match the required use case or to get as close as possible to the final use case settings.

GMSL and MIPI Profile Tables

Table 6. GMSL Profiles

GMSL Profiles				
GMSL Profile #	**CXTP**	**GMSL Type**	**MAX96724F/R GMSL Rate/HIM Mode**	**MAX96724 GMSL Rate/HIM Mode**
0	COAX	GMSL2	3Gbps	6Gbps
1	COAX	GMSL2	3Gbps	3Gbps
2	COAX	GMSL1	HIM Disabled	HIM Disabled
3	STP	GMSL2	3Gbps	6Gbps
4	STP	GMSL2	3Gbps	3Gbps
5	STP	GMSL1	HIM Enabled	HIM Enabled
6	STP	GMSL1	HIM Disabled	HIM Disabled
7	COAX	GMSL1	HIM Enabled	HIM Enabled

Table 7. MIPI Profiles MAX96724R

MIPI Profiles – MAX96724R									
Profile	**Links**	**Lanes**	**MIPI Type**	**MIPI Rate**	**Link to Port**	**Replication**	**Frame Sync**	**Aggregation Mode**	**Data Type for GMSL1**
1	1 (A)	1x4	DPHY	1.5	A -> A	No	No	FCFS - Ctrl 1	RAW12
2	1 (A)	2x2	DPHY	1.5	A -> C, D	C -> D	No	FCFS - Ctrl 0	RAW12
3	1 (A)	1x4	CPHY	1.5	A -> A	No	No	FCFS - Ctrl 1	RAW12
4	1 (A)	2x2	CPHY	1.5	A -> C, D	C -> D	No	FCFS - Ctrl 0	RAW12
5	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	FCFS - Ctrl 1	RAW12
6	4 (A-D)	2x2	DPHY	2.5	All -> C, D	C -> D	30 fps	FCFS - Ctrl 0	RAW12
7	4 (A-D)	2x2	DPHY	1.5	A+B -> C C+D -> D	No	No	FCFS A+B -> Ctrl 0, C+D -> Ctrl 1	RAW12
8	4 (A-D)	1x4	DPHY	1.5	All -> A	No	No	FCFS - Ctrl 1	RAW12
9	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	FCFS - Ctrl 1	YUV422
10	4 (A-D)	2x2	DPHY	2.5	All -> C, D	C -> D	30 fps	FCFS - Ctrl 0	YUV422
11	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	Wx4H (tunnel mode)	RAW12
12	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	4WxH (pixel mode)	RAW12
13	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	Wx4H (tunnel mode)	YUV422
14	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	4WxH (pixel mode)	YUV422
15	4 (A-D)	2x2	DPHY	2.5	All -> C, D	C -> D	30 fps	FCFS - Ctrl 0	RAW12
16	4 (A-D)	2x2	DPHY	2.5	All -> C, D	C -> D	30 fps	FCFS - Ctrl 0	YUV422

Table 8. MIPI Profiles MAX96724F

MIPI Profiles – MAX96724F									
Profile	**Links**	**Lanes**	**MIPI Type**	**MIPI Rate**	**Link to Port**	**Replication**	**Frame Sync**	**Aggregation Mode**	**Data Type for GMSL1**
1	1 (A)	1x4	DPHY	1.5	A -> A	No	No	FCFS - Ctrl 1	RAW12
2	1 (A)	2x2	DPHY	1.5	A -> C, D	C -> D	No	FCFS - Ctrl 0	RAW12
3	1 (A)	1x4	CPHY	1.5	A -> A	No	No	FCFS - Ctrl 1	RAW12
4	1 (A)	2x2	CPHY	1.5	A -> C, D	C -> D	No	FCFS - Ctrl 0	RAW12
5	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	FCFS - Ctrl 1	RAW12
6	4 (A-D)	2x4	DPHY	2.5	All -> A, B	A -> B	30 fps	FCFS - Ctrl 1	RAW12
7	4 (A-D)	2x4	DPHY	1.5	A+B -> A C+D -> B	No	No	FCFS A+B -> Ctrl 1, C+D -> Ctrl 2	RAW12
8	4 (A-D)	1x4	DPHY	1.5	All -> A	No	No	FCFS - Ctrl 1	RAW12
9	4 (A-D)	1x4	DPHY	1.5	All -> A	No	30 fps	FCFS - Ctrl 1	YUV422
10	4 (A-D)	2x4	DPHY	2.5	All -> A, B	A -> B	30 fps	FCFS - Ctrl 1	YUV422
11	4 (A-D)	2x4	DPHY	1.5	All -> A, B	A -> B	30 fps	Wx4H (tunnel mode)	RAW12
12	4 (A-D)	2x4	DPHY	1.5	All -> A, B	A -> B	30 fps	4WxH (pixel mode)	RAW12
13	4 (A-D)	2x4	DPHY	1.5	All -> A, B	A -> B	30 fps	Wx4H (tunnel mode)	YUV422
14	4 (A-D)	2x4	DPHY	1.5	All -> A, B	A -> B	30 fps	4WxH (pixel mode)	YUV422
15	4 (A-D)	2x2	CPHY	2.5	All -> C, D	C -> D	30 fps	FCFS - Ctrl 0	RAW12
16	4 (A-D)	2x2	CPHY	2.5	All -> C, D	C -> D	30 fps	FCFS - Ctrl 0	YUV422

Table 9. MIPI Profiles MAX96724

MIPI Profiles – MAX96724									
Profile	**Links**	**Lanes**	**MIPI Type**	**MIPI Rate**	**Link to Port**	**Replication**	**Frame Sync**	**Aggregation Mode**	**Data Type for GMSL1**
1	1 (A)	2x4	DPHY	1.5	A -> A	A -> B	No	FCFS - Ctrl 1	RAW12
2	1 (A)	4x2	DPHY	1.5	A -> C, D	C -> D	No	FCFS - Ctrl 0	RAW12
3	1 (A)	2x4	CPHY	1.5	A -> A	A -> B	No	FCFS - Ctrl 1	RAW12
4	1 (A)	4x2	CPHY	1.5	A -> C, D	C -> D	No	FCFS - Ctrl 0	RAW12
5	4 (A-D)	2x4	DPHY	1.5	All -> A	No	30 fps	FCFS - Ctrl 1	RAW12
6	4 (A-D)	2x4	DPHY	2.5	All -> A, B	A -> B	30 fps	FCFS - Ctrl 1	RAW12
7	4 (A-D)	2x4	DPHY	2.5	A+B -> A C+D -> B	No	No	FCFS A+B -> Ctrl 1, C+D -> Ctrl 2	RAW12
8	4 (A-D)	2x4	CPHY	2.5	All -> A	No	No	FCFS - Ctrl 1	RAW12
9	4 (A-D)	2x4	CPHY	2.5	All -> A	No	30 fps	FCFS - Ctrl 1	YUV422
10	4 (A-D)	2x4	DPHY	2.5	All -> A, B	A -> B	30 fps	FCFS - Ctrl 1	YUV422

11	4 (A-D)	2x4	CPHY	2.5	All -> A, B	A -> B	30 fps	Wx4H (tunnel mode)	RAW12
12	4 (A-D)	2x4	CPHY	2.5	All -> A, B	A -> B	30 fps	4WxH (pixel mode)	RAW12
13	4 (A-D)	2x4	CPHY	2.5	All -> A, B	A -> B	30 fps	Wx4H (tunnel mode)	YUV422
14	4 (A-D)	2x4	CPHY	2.5	All -> A, B	A -> B	30 fps	4WxH (pixel mode)	YUV422
15	4 (A-D)	2x4	CPHY	2.5	All -> A, B	A -> B	30 fps	FCFS - Ctrl 1	RAW12
16	4 (A-D)	2x4	CPHY	2.5	All -> A, B	A -> B	30 fps	FCFS - Ctrl 1	YUV422

GMSL and MIPI Profile Registers

Table 10. MIPI Profile Registers

Register	Bits	Default Value	Description
0x6E1	5:0	0x00	MIPI Profile Select: 0-16
0x6EA	6:4 and 2:0	0x00	GMSL Profile Select Links A and B
0x6EB	6:4 and 2:0	0x00	GMSL Profile Select Links C and D

GMSL and MIPI Profile Programming Examples

GMSL Profile Programming Examples

```
# Set GMSL Profile to COAX, GMSL2, 3Gbps mode for all links A and B.
0x4E, 0x6EA, 0x11
# Set GMSL Profile to COAX, GMSL1, HIMM Enabled for all links C and D.
0x4E, 0x6EB, 0x77
```

MIPI Profile Programming Examples

```
# Set MIPI Profile to profile 16.
0x4E, 0x6E1, 0x10
# Set MIPI Profile to profile 5.
0x4E, 0x6E1, 0x05
```

Video Pipe to MIPI Controller Mapping

Video pipe to MIPI controller mapping is used to reconfigure the video pipe data paths and route streams to different MIPI ports. GMSL2 CSI-2 quad deserializers have four MIPI controllers (Ctrl 0, 1, 2, and 3) that control four different MIPI PHYs (MIPI PHY 0, 1, 2, and 3).

The necessity of this step is dependent upon the serial link connections and the specific use case. For example, if four video streams are routed through the deserializer to 4 separate 2 lane MIPI outputs ports, then the video pipe to controller mapping may not need to be adjusted. In the case that 4 cameras must be aggregated to a single, 4 lane MIPI output port, then the default pipe to controller mapping must be changed. If aggregating or concatenating multiple data streams, tunnel and pixel mode streams may not be aggregated together. The default video pipe MIPI controller connections are shown below for GMSL1 and GMSL2 modes.

Default Mapping

- GMSL1/2 PHY A → (Video Pipe 0) → Controller 0 / MIPI PHY 0
- GMSL1/2 PHY B → (Video Pipe 1) → Controller 1 / MIPI PHY 1
- GMSL1/2 PHY C → (Video Pipe 2) → Controller 2 / MIPI PHY 2
- GMSL1/2 PHY D → (Video Pipe 3) → Controller 3 / MIPI PHY 3

PHY Selection for Mapping

Video data must be mapped to a PHY that is the master of the CSI-2 port. In 2x4 mode, CSI-2 port A's master PHY is PHY1, and CSI-2 port B's master PHY is PHY2. When in 4x2 mode, each CSI-2 port C, D, E, and F become a master and video data may be routed to any PHY. Video data should not be mapped to PHY's that are currently in the slave state as they do not arbitrate the incoming data.

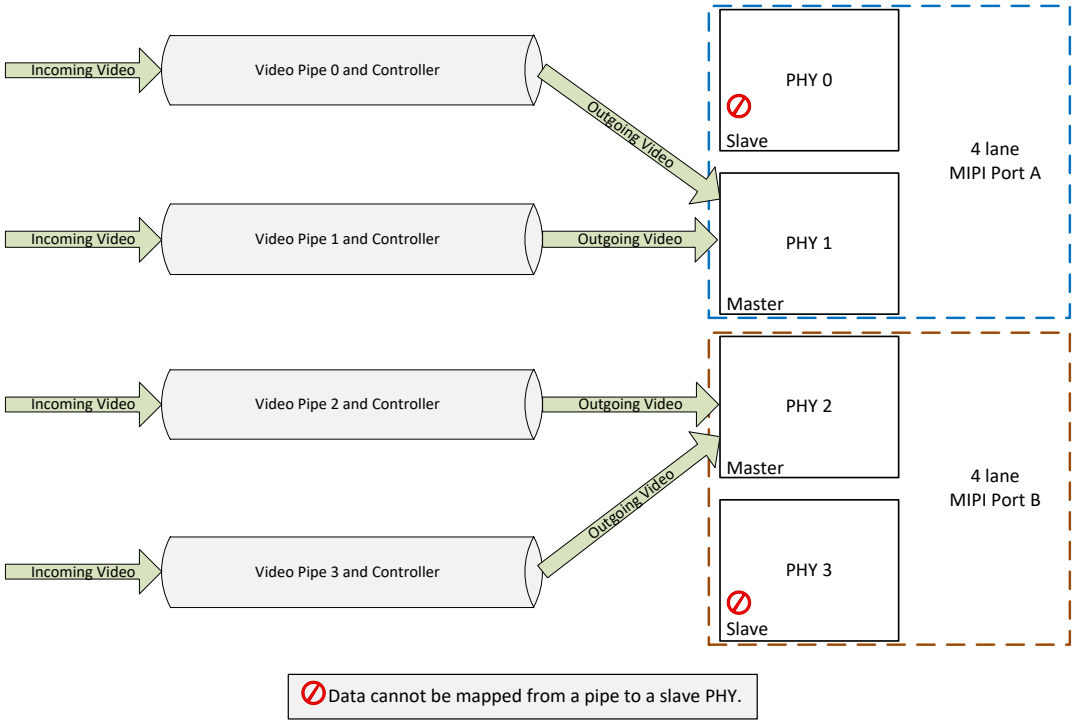

Figure 7. 2x4 Mode PHY Selection

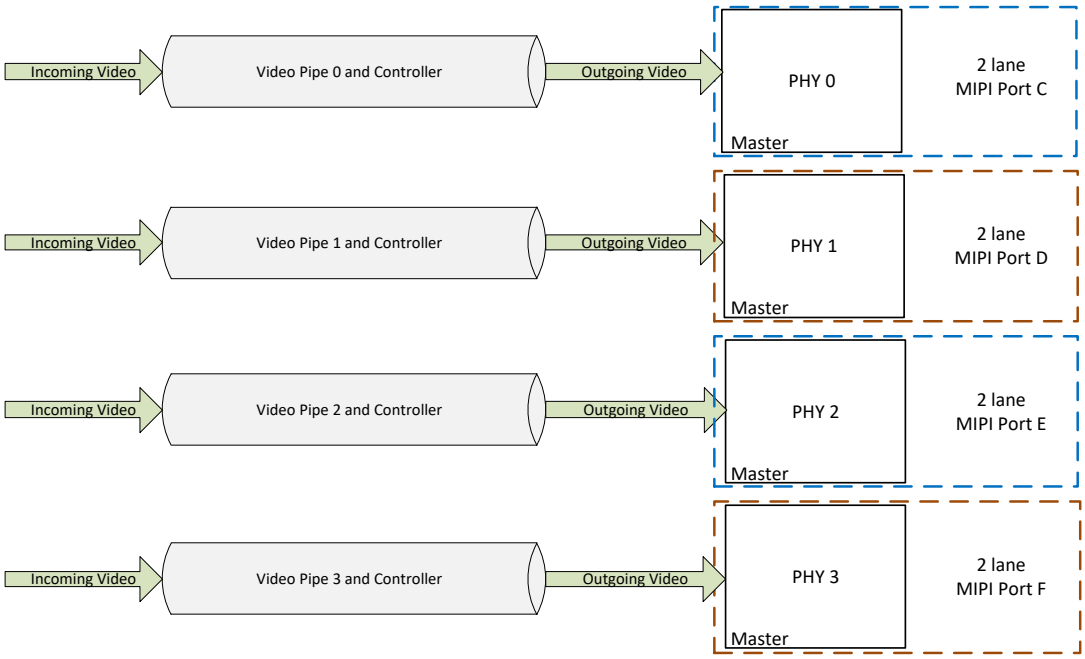

Figure 8. 4x2 Mode PHY Selection

Methods to Change the Mapping

The following steps describe the methods of configuring video pipe to MIPI controller mapping. Method 2 is used to change the mapping for complex mapping scenarios that requires virtual channel reassignments and is especially effective for pipes that contain multiple data types or video streams. Method 1 is a convenient way to adjust mappings for simple routing scenarios.

Method 1

For simple mappings that do not require complex virtual channel reassignments, the default pipe to controller mappings may be changed with a single register write. This is a convenient method to change the mapping if all data on a pipe must be routed to the same MIPI PHY locations. This can be done with MIPI_CTRL_SEL_x, bits in register 0x08CA. This method also allows for a simple way to aggregate data streams. If all multiple pipes are sent to the same controller, aggregation naturally occurs. It is important to consider that in tunneling mode the mappings in the MIPI_CTRL_SEL_x bits must match the TUN_DEST selections.

Method 2

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 each enabled mapping block (i.e., the MIPI PHY controller destination). The data being mapped is designated via 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. This method is required when dealing with parallel format data that is not using the MIPI CSI-2 packet structure.

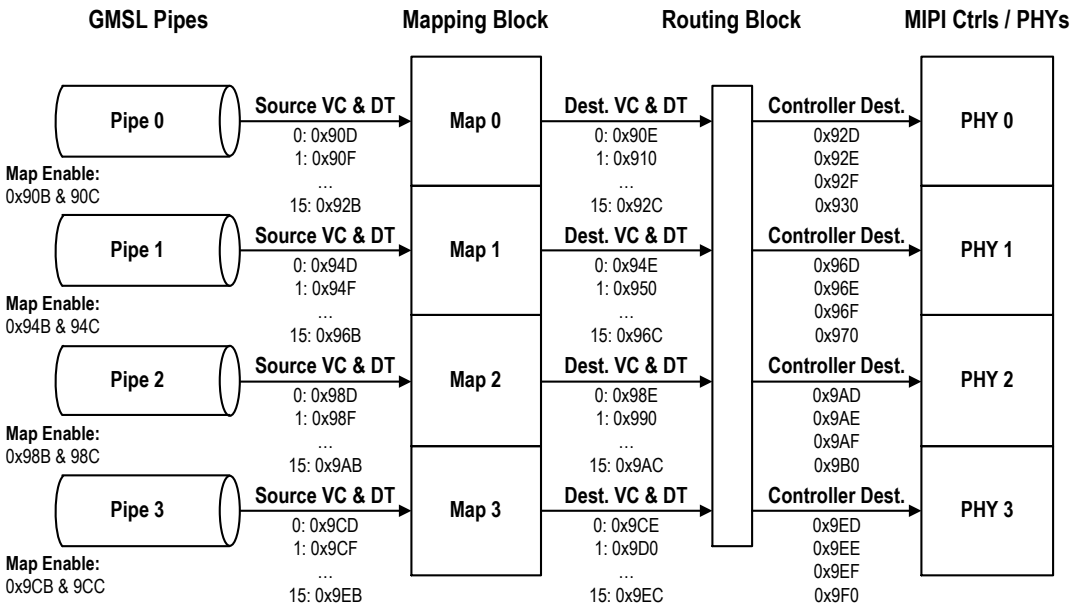

Figure 9. Video Pipe to MIPI Controller Mapping Block Diagram

Pipe to controller mapping register blocks:

- Video Pipe 0 mapping register block: 0x090B – 0x0930
- Video Pipe 1 mapping register block: 0x094B – 0x0970
- Video Pipe 2 mapping register block: 0x098B – 0x09B0
- Video Pipe 3 mapping register block: 0x09CB – 0x09F0

Video Pipe to MIPI PHY Controller Registers

Note: Video Pipe 0 registers are shown as an example due to quantity of registers. Please see device register maps for pipes 1-3.

Table 11. Video Pipe 0 to MIPI PHY Controller Register Table

Register	Bits	Default Value	Description
0x08CA	7:0	0xE4	**MIPI_CTRL_SEL Mapping:** Bits [7:6]: MIPI_CTRL_SEL_3, for Pipe 3 Bits [5:4]: MIPI_CTRL_SEL_2, for Pipe 2 Bits [3:2]: MIPI_CTRL_SEL_1, for Pipe 1 Bits [1:0]: MIPI_CTRL_SEL_0, for Pipe 0 00 = Map to Controller 0 01 = Map to Controller 1 10 = Map to Controller 2 11 = Map to Controller 3
0x090B	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 1 of max 16 mapping and distribution entries for current video stream.
0x090C	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 1 of max 16 mapping and distribution entries for current video stream.
0x090D	7:0	0x00	**MAP_SRC_0 Register:** Bits [7:6]: VC Bits [5:0]: DT
0x090E	7:0	0x00	**MAP_DES_0 Register:** Bits [7:6]: VC Bits [5:0]: DT
…	…	…	…
0x092B	7:0	0x00	**MAP_SRC_15 Register:** Bits [7:6]: VC Bits [5:0]: DT
0x092C	7:0	0x00	**MAP_DST_15 Register:** Bits [7:6]: VC Bits [5:0]: DT
0x092D	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 = Map to Controller 0 01 = Map to Controller 1 10 = Map to Controller 2 11 = Map to Controller 3
0x092E	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 = Map to Controller 0 01 = Map to Controller 1 10 = Map to Controller 2 11 = Map to Controller 3
0x092F	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 = Map to Controller 0 01 = Map to Controller 1 10 = Map to Controller 2 11 = Map to Controller 3
0x0930	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 = Map to Controller 0

			01 = Map to Controller 1 10 = Map to Controller 2 11 = Map to Controller 3

Video Pipe to MIPI Controller Mapping Programming Examples

Pipe to Controller Mapping using Method 1

The deserializer receives uniquely identifiable video data on each pipe and aggregates it to MIPI PHY1 in this example. This utilizes method 2 to change the mapping.

```
# Using MIPI_CTRL_SEL_x, set all pipes to rout to controller 1.
0x4E, 0x8CA, 0x55
```

Pipe to Controller Mapping using Method 2

The deserializer receives data on links A and B from serializer video pipe X in this example. In the deserializer, the RAW12 data in video pipes 1 and 2 are aggregated to PHY1 output. This utilizes method 2 to change the mapping.

```
# Selects Link A and Link B, pipe X, for pipe 0 and pipe 1.
0x4E, 0xF0, 0x40
# Only turn on pipes 0-1
0x4E, 0xF4, 0x03
# Enable mapping for pipe 0, 3 mappings enabled for FS, FE, and data type
0x4E, 0x90B, 0x07
# Map_SRC_0 - FS, VC = 0
0x4E, 0x90D, 0x00
# Map_DST_0 - FS, VC = 0
0x4E, 0x90E, 0x00
# Map_SRC_1 - Raw12, VC = 0
0x4E, 0x90F, 0x2C
# Map_DST_1 - Raw12, VC = 0
0x4E, 0x910, 0x2C
# Map_SRC_2 - FE, VC = 0
0x4E, 0x911, 0x01
# Map_DST_2 - FE, VC = 0
0x4E, 0x912, 0x01
# Map_D-PHY_DST Sets PHY destination to PHY1 for mappings 0-2
0x4E, 0x92D, 0x15
# Enable mapping for pipe1, 3 mappings enabled for FS, FE, and data type
0x4E, 0x94B, 0x07
# Map_SRC_0 - FS, VC = 0
0x4E, 0x94D, 0x00
# Map_DST_0 - FS, VC = 1
0x4E, 0x94E, 0x40
# Map_SRC_1 - Raw12, VC = 0
0x4E, 0x94F, 0x2C
# Map_DST_1 - Raw12, VC = 1
0x4E, 0x950, 0x6C
# Map_SRC_2 - FE, VC = 0
0x4E, 0x951, 0x01
# Map_DST_2 - FE, VC = 1
0x4E, 0x952, 0x41
# Map_D-PHY_DST Sets PHY destination to PHY1 for mappings 0-2
0x4E, 0x96D, 0x15
```

MIPI PHY Settings

The MIPI PHY settings contain programming options for output data rate, number of lanes, and port selection. The GMSL2 CSI-2 quad deserializers have four independent two-lane MIPI PHYs (PHY 0, 1, 2, and 3) controlled by four respective MIPI controllers (Ctrl 0, 1, 2, and 3). Table 12 contains the MIPI PHY settings registers.

There are four MIPI PHY modes available for either D-PHY or C-PHY applications depending on the device version. Please refer to the datasheet for information regarding the version of the device and what MIPI output modes are supported. Set one of the bits (exclusively) in register 0x08A0[4:0] to select the mode.

- 0x08A0 bit 0: 4 x 2 Mode
- 0x08A0 bit 2: 2 x 4 Mode (Ctrl 1 is the master for port A; Ctrl 2 is the master for port B). This is used for 1x4 mode on the MAX96724R variant.
- 0x08A0 bit 3: 1 x 4a + 2 x 2 Mode (Ctrl 1 is the master for port A)
- 0x08A0 bit 4: 1 x 4b + 2 x 2 Mode (Ctrl 2 is the master for port B)

In **x2 Mode**, four MIPI PHYs work independently. Each corresponding port (C, D, E, and F) has its own clock and two data lanes in D-PHY mode. Similarly, each port has two trios in C-PHY mode.

In **x4 Mode**, two of the MIPI PHYs are combined to establish 4-lane mode. In D-PHY mode, by default, PHY 1 and PHY 2 are the master PHYs providing the MIPI clock for ports A and B. The clock from slave PHYs can be used as the master clock by forcing the PHY 0 or 3 clock output on register 0x08A0. In C-PHY mode, the two-trio PHYs are combined into a four-trio.

The 1x1, 1x2, 1x3, and 1x4 configurations use lane count settings. Refer to the Table 12 below for register write details.

MIPI PHY Settings Registers

Table 12. MIPI PHY Settings Register Table

Register	Bits	Default Value	Description
0x08A0	7:0	0x04	**MIPI PHY Mode Select Register:** Bit 7: Set to force all MIPI clocks running. Bit 6: Set to force PHY 3 MIPI clock running. Bit 5: Set to force PHY 0 MIPI clock running. Bit 4: MIPI PHY 1x4b + 2x2 Mode. Bit 3: MIPI PHY 1x4a + 2x2 Mode. Bit 2: MIPI PHY 2x4 Mode. Bit 1: MIPI PHY 1x4 Mode. Bit 0: MIPI PHY 4x2 Mode.
0x08A2	7:4	0xF4	**MIPI PHY Enable Register:** Bit 7: Enable MIPI PHY 3 Bit 6: Enable MIPI PHY 2 Bit 5: Enable MIPI PHY 1 Bit 4: Enable MIPI PHY 0 0 = PHY in standby 1 = PHY enabled.
0x08A3	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.
0x08A4	7:0	0xE4	**MIPI PHY 3 and 2 Lane Mapping Register:** Bits [7:6]: Set PHY 3 Data Lane 1 Bits [5:4]: Set PHY 3 Data Lane 0 Bits [3:2]: Set PHY 2 Data Lane 1 Bits [1:0]: Set PHY 2 Data Lane 0 00 = Map D0. 01 = Map D1. 10 = Map D2. 11 = Map D3.
0x08A5	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 / Swap PHY 1 D1 lane A & C pins in C-PHY mode Bit 3: Set Polarity on PHY 1 D0 lane / Swap PHY 1 D0 lane A & C pins in C-PHY mode Bit 2: Set Polarity on PHY 0 CLK lane Bit 1: Set Polarity on PHY 0 D1 lane / Swap PHY 0 D1 lane A & C pins in C-PHY mode Bit 0: Set Polarity on PHY 0 D0 lane / Swap PHY 0 D0 lane A & C pins in C-PHY mode 0 = normal polarity, 1 = inversed polarity.
0x08A6	5:0	0x00	**MIPI PHY 3 and 2 Lane Polarity Register:** Bit 5: Set Polarity on PHY 3 CLK lane Bit 4: Set Polarity on PHY 3 D1 lane/ Swap PHY 3 D1 lane A & C pins in C-PHY mode Bit 3: Set Polarity on PHY 3 D0 lane/ Swap PHY 3 D0 lane A & C pins in C-PHY mode Bit 2: Set Polarity on PHY 2 CLK lane Bit 1: Set Polarity on PHY 2 D1 lane/ Swap PHY 2 D1 lane A & C pins in C-PHY mode Bit 0: Set Polarity on PHY 2 D0 lane/ Swap PHY 2 D0 lane A & C pins in C-PHY mode 0 = normal polarity, 1 = inversed polarity.
0x08C9	3:0	0x00	**MIPI Controller Reset Register:** Bits [3:0]: resets for each controller 0/1/2/3 0: Do not reset 1: Reset the MIPI Controller
0x090A	7:5	110	**MIPI PHY 0 Lane Count / C-PHY enable Register:** Bits [7:6]: Set lane count. 00 = 1 data lane 01 = 2 data lanes 10 = 3 data lanes 11 = 4 data lanes Bit 5: Enable C-PHY; D-PHY mode by default. 0 = C-PHY disabled, 1 = C-PHY enabled.

0x094A	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 Bit 5: Enable C-PHY; D-PHY mode by default. 0 = C-PHY disabled, 1 = C-PHY enabled.
0x098A	7:6	11	**MIPI PHY 2 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 Bit 5: Enable C-PHY; D-PHY mode by default. 0 = C-PHY disabled, 1 = C-PHY enabled.
0x09CA	7:6	11	**MIPI PHY 3 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 Bit 5: Enable C-PHY; D-PHY mode by default. 0 = C-PHY disabled, 1 = C-PHY enabled.
0x0415	4:0	01111	**MIPI PHY 0 DPLL Freq Register:** Set DPLL frequency on multiple of 100MHz. **D-PHY** Clock freq is half; Data rate is equivalent bps/lane. 00010 = 200MHz DPLL, 200Mbps/lane data rate. … 11001 = 2500MHz DPLL, 2.5Gbps/lane data rate. **C-PHY** 2.28bits/symbol. 00010 = 200MHz DPLL, 456Mbps/lane data rate. … 11001 = 2500MHz DPLL, 5.7Gbps/lane data rate.
0x0418	4:0	01111	**MIPI PHY 1 DPLL Freq Register:** Set DPLL frequency on multiple of 100MHz. **D-PHY** Clock freq is half; Data rate is equivalent bps/lane. 00001 = 100MHz DPLL, 100Mbps/lane data rate. … 11001 = 2500MHz DPLL, 2.5Gbps/lane data rate. **C-PHY** 2.28bits/symbol. 00010 = 200MHz DPLL, 456Mbps/lane data rate. … 11001 = 2500MHz DPLL, 5.7Gbps/lane data rate.
0x041B	4:0	01111	**MIPI PHY 2 DPLL Freq Register:** Set DPLL frequency on multiple of 100MHz. **D-PHY** Clock freq is half; Data rate is equivalent bps/lane.

			00001 = 100MHz DPLL, 100Mbps/lane data rate. … 11001 = 2500MHz DPLL, 2.5Gbps/lane data rate. **C-PHY** 2.28bits/symbol. 00010 = 200MHz DPLL, 456Mbps/lane data rate. … 11001 = 2500MHz DPLL, 5.7Gbps/lane data rate.
0x041E	4:0	01111	**MIPI PHY 3 DPLL Freq Register:** Set DPLL frequency on multiple of 100MHz. **D-PHY** Clock freq is half; Data rate is equivalent bps/lane. 00001 = 100MHz DPLL, 100Mbps/lane data rate. … 11001 = 2500MHz DPLL, 2.5Gbps/lane data rate. **C-PHY** 2.28bits/symbol. 00010 = 200MHz DPLL, 456Mbps/lane data rate. … 11001 = 2500MHz DPLL, 5.7Gbps/lane data rate.

MIPI Lane Swap

Lane swapping is available for pins on the same port in both C-PHY and D-PHY modes. Reference Table 12 for relevant registers.

D-PHY Lane Swap Example

For D-PHY output, the data pins can be swapped within each port, but the clock location is fixed. For example, in 2x4 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 support polarity inversion. Figure 10 shows the lane swap required to match the device pinout in D-PHY mode. Figure 11 demonstrates polarity swap in D-PHY mode.

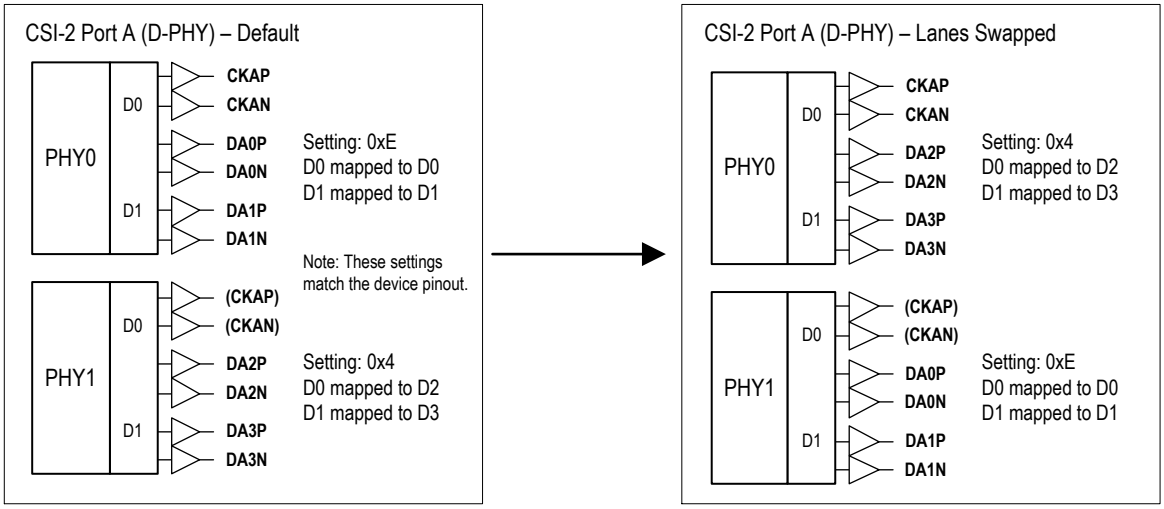

Figure 10. D-PHY Lane Swap Example

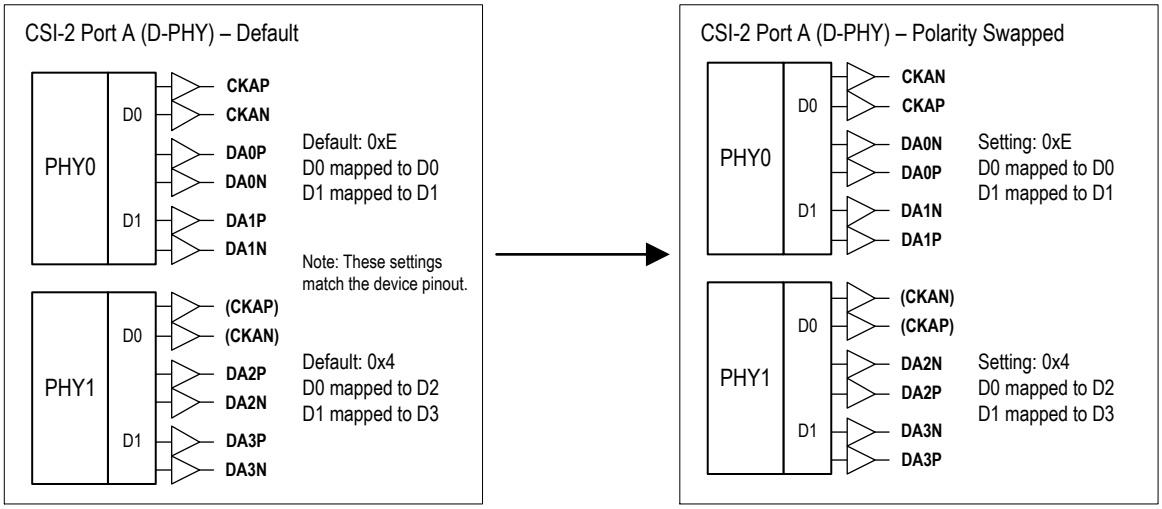

Figure 11. D-PHY Polarity Swap Example

C-PHY Lane Swap Example

The location of each trio may be swapped within the same PHY in C-PHY mode. The default for all three output pins (i.e., A, B, and C) can be swapped from the default pinout. The polarity inversion function in C-PHY mode swaps the A and C pins of a trio. Figure 12 shows the lane swap required to match the device pinout in C-PHY mode. Figure 13 demonstrates polarity swap in C-PHY mode.

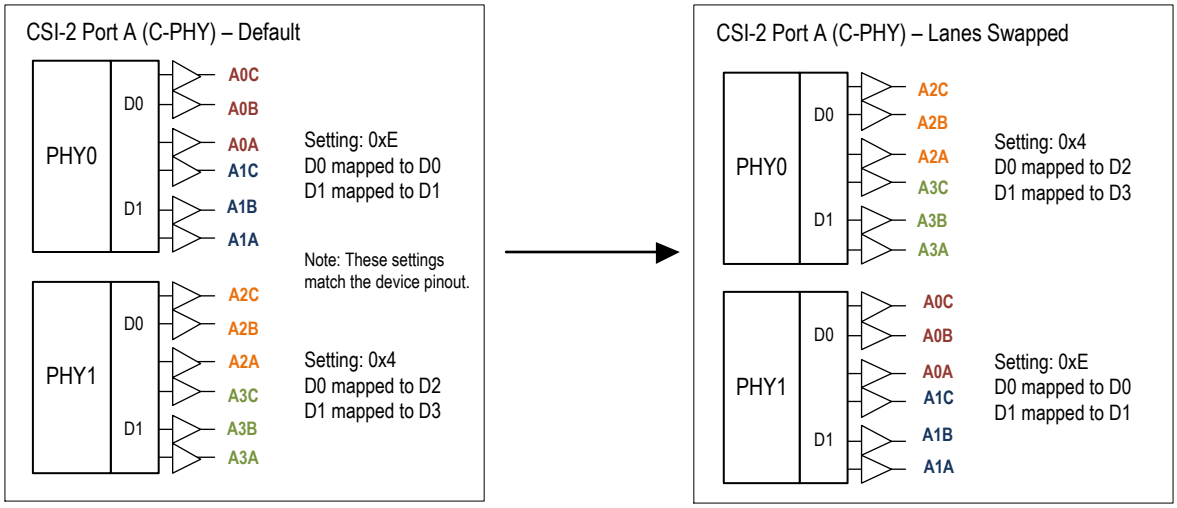

Figure 12. C-PHY Lane Swap Example

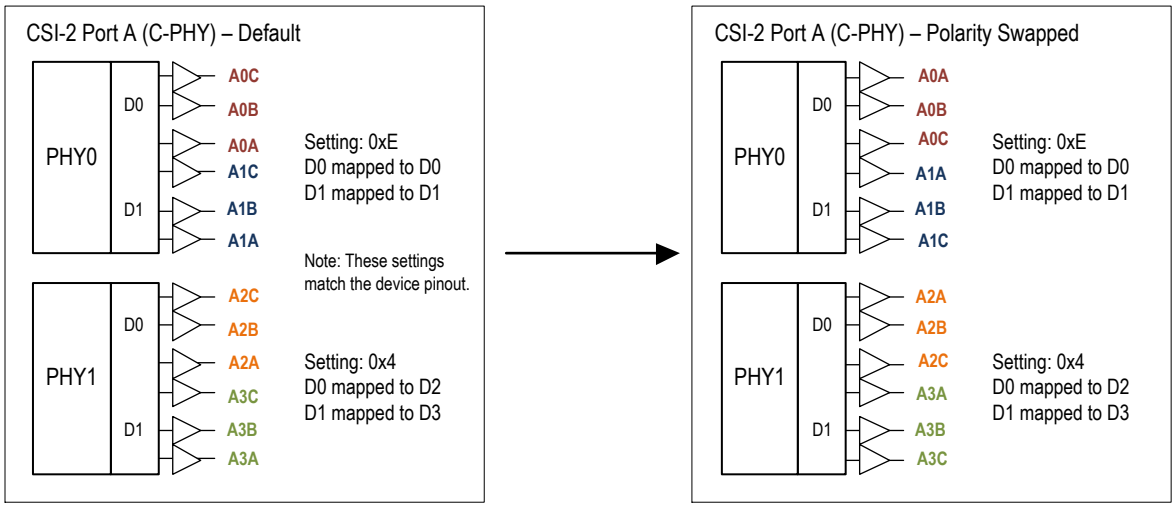

Figure 13. C-PHY Polarity Swap Example

Lane Swap Programming Example

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

```
# Set MIPI PHY 2x4 mode
0x4E,0x08A0,0x04
# Set lane mapping for ctrl 1 and 2 (MIPI Port A and B in x4 mode). This is written
to completely swap the device pinout from default.
0x4E,0x08A3,0x4E
0x4E,0x08A4,0x4E
# Set 4 lanes for ctrl 1 and 2 (MIPI Port A and B in x4 mode) in D-PHY mode
0x4E,0x094A,0xC0
0x4E,0x098A,0xC0
# Set MIPI lane rate to be 2Gbps/lane for ctrl 1 and 2 (MIPI Port A and B in x4
mode)
0x4E,0x0418,0x34
0x4E,0x041B,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 1.5Gbps per lane. Periodic deskew requires a continuous clock and that the clock lane is always in high-speed mode.

The initial transmit deskew is set with DESKEW_INIT[7:0] in registers 0x0903/0x0943/0x0983/0x09C3, which correspond to each MIPI PHY. 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 has been 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 registers 0x0904/0x0944/0x0984/0x09C4, which correspond to each MIPI PHY/Controller. The period interval has a programmable range from every 1 to 128 frames. Table 13 contains the relevant MIPI D-PHY deskew registers.

MIPI D-PHY Deskew Registers

Table 13 MIPI D-PHY Deskew Registers

Register	Bits	Default Value	Description
0x0903	7:0	0x00	**MIPI TX 0 DESKEW_INIT Register:** Bit 7: Auto initial deskew on/off Bit 6: Reserved Bit 5: when bit 4 = 1, any change of this bit will trigger one-time immediate initial skew Bit 4: Manual initial on/off Bit 3: Reserved Bits [2:0]: Select initial deskew width: 1, 2, 3, … 8 * (32K) UI
0x0904	7:0	0x00	**MIPI TX 0 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 * (1K) UI
0x0943	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 will trigger one-time immediate initial skew. Bit 4: Manual initial on/off Bit 3: Reserved Bits [2:0]: Initial Deskew width 1, 2, … 8*(32K) UI
0x0944	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 * (1K) UI
0x0983	7:0	0x00	**MIPI TX 2 DESKEW_INIT Register:** Bit 7: Auto initial deskew on/off Bit 6: Reserved Bit 5: when bit 4 = 1, any change of this bit will trigger one-time immediate initial skew. Bit 4: Manual initial on/off Bit 3: Reserved Bits [2:0]: Initial Deskew width 1, 2, … 8*(32K) UI
0x0984	7:0	0x00	**MIPI TX 2 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 * (1K) UI
0x09C3	7:0	0x00	**MIPI TX 3 DESKEW_INIT Register:** Bit 7: Auto initial deskew on/off

			Bit 6: Reserved Bit 5: when bit 4 = 1, any change of this bit will trigger one-time immediate initial skew. Bit 4: Manual initial on/off Bit 3: Reserved Bits [2:0]: Initial Deskew width 1, 2, … 8*(32K) UI
0x09C4	7:0	0x00	**MIPI TX 3 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 * (1K) 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.
0x4E,0x0943,0x80
# Enable periodic deskew packets with width 2K UI every 2 frames.
0x4E,0x0944,0x91
```

MIPI C-PHY Settings

The quad deserializers conform with the MIPI C-PHY v1.0 standard. This physical interface uses three-phase symbol encoding that delivers 2.28 bits per symbol over three-wire trios operating at 2.5GSym/s per trio. Four trios have a peak bandwidth of 22.8Gbps. C-PHY and D-PHY are intended to be able to share device hardware; refer to MIPI specifications for design rules to ensure ideal system performance.

The deserializers supports 2- and 4-lane C-PHY modes; further programming allows 1- and 3-lane C-PHY arrangements similar to the D-PHY configurations. C-PHY mode is enabled by setting CSI2_CPHY_EN = 1 bit 5 in registers 0x090A/0x094A/0x098A/0x09CA. Each PHY is individually configured with a dedicated register. Refer to *MIPI PHY Settings* for more information.

According to the C-PHY v1.0 protocol, HS data transmissions have two programmable fields that may be adjusted to ensure that the C-PHY receiver has a more stable capture *Figure 14*.
The t3-PREBEGIN field of the preamble comprises multiple groups of seven “3” symbols. The purpose of the t3-PREBEGIN field is to provide enough clocks to the upper layer protocol so that pipeline stages are initialized prior to receiving data. Similarly, the t3-POST field at the end of the packet data is identified by a unique sequence of “4” symbols.

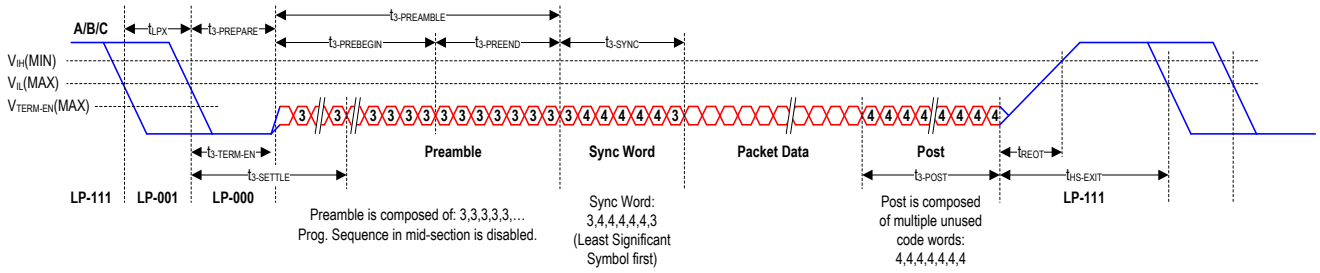

Figure 14 C-PHY v1.0 HS Burst Data Transmission

MIPI C-PHY Timing Registers

Table 14. MIPI C-PHY Timing Registers

Register	Bits	Default Value	Description
0x08AD	5:0	0x1F	**$t_{3\text{-PREBEGIN}}$ Register:** C-PHY Pre-begin phase of the preamble length = ($t_{3\text{-PREBEGIN}}$ + 1) * 7 UI.
0x08AE	6:2	0x5D	**$t_{3\text{-POST}}$ Register:** C-PHY Post length = ($t_{3\text{-POST}}$ + 1) * 7 UI.

MIPI C-PHY Timing Programming Example

The following programming example shows a commonly acceptable C-PHY $t_{3\text{-PREBEGIN}}$ and $t_{3\text{-POST}}$ setting, which matches the deserializer default values. This may need to be adjusted if the MIPI receiver requires different values.

```
# Set 4 lane C-PHY for MIPI PHY 1
0x4E,0x094A,0xE0
# C-PHY timing adjustment
0x4E,0x08AD,0x1F
0x4E,0x08AE,0x5D
```

MIPI PHY Copy

GMSL2 CSI-2 quad deserializers also support MIPI PHY-to-PHY replication. This is configured by enabling PHY copy and setting the source and destination PHYs with phy_cp0[4:0] and phy_cp1[4:0] in registers 0x08A9 and 0x08AA, respectively. When using the PHY copy function the MIPI rate, MIPI type, and lane count must be the same between the two ports. All video data is also copied from the source PHY to the destination PHY for the copy. Table 15 contains MIPI PHY replication registers.

Table 15. MIPI PHY Replication Registers

Register	Bits	Default Value	Description
0x08A9	7:3	00000	**PHY copy Register:** Bit 7: 0 = Disable, 1 = Enable PHY0 copy Bits [6:5]: PHY copy destination. Bits [4:3]: PHY copy source.
0x08AA	7:3	00000	**PHY copy Register:** Bit 7: 0 = Disable, 1 = Enable PHY1 copy

			Bits [6:5]: PHY copy destination. Bits [4:3]: PHY copy source.

PHY Copy Programming Example

This example programs the deserializer to copy MIPI PHY 1 to MIPI PHY2, for 2x4 mode setup. This copies CSI-2 port A to CSI-2 port B.

```
# Enable PHY_CP0 and make the source PHY1 “0b01” and destination PHY2 “0b10”.
0x4E,0x08A9,0xC8
```

Extended Virtual Channels

GMSL2 quad 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) or 5-bit (C-PHY), increasing the number of available virtual channels to 16 for D-PHY applications and 32 for C-PHY applications. The increase in available virtual channels allows systems to support more camera inputs.

Extended virtual channels are enabled individually for each of the four controllers (0-3) on the quad deserializer. Any video pipes routed to a controller with virtual channel extension enabled will use the extended virtual channels.

Reference Table 18. Extended Virtual Channels Register Table for extended virtual channels registers details.

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

Extended Virtual Channels Register Examples

This 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 16. Source Mapping (Extended VC)

	EXT VC Source			VC Source	
Register	**0x0800 [7:5]**			**0x090D [7:6]**	
MIPI	**C-PHY Only**	**D-PHY and C-PHY**			
Bit Place	**4**	**3**	**2**	**1**	**0**
VC = 0	0	0	0	0	0

Table 17. Destination Mapping (Extended VC)

	EXT VC Destination			VC Destination	
Register	**0x0800 [4:2]**			**0x090E [7:6]**	
MIPI	**C-PHY Only**	**D-PHY and C-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 18. Extended Virtual Channels Register Table

Register	Bits	Default Value	Description
0x090A/0x094A/0x098A/0x09CA	4	0	**CSI2_vcx_en[?]:** 0 = VC extension Disabled 1 = VC extension Enabled Controller 0,1,2, and 3 virtual channels enable.
0x800	7:2	0x00	**Pipe 0, Extended VC MAP_SRC/DST_0 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel
…	…	…	**…**
0x80F	7:2	0x00	**Pipe 0, Extended VC MAP_SRC/DST_15 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel
0x810	7:2	0x00	**Pipe 1, Extended VC MAP_SRC/DST_0 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel
…	…	…	**…**
0x81F	7:2	0x00	**Pipe 1, Extended VC MAP_SRC/DST_15 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel
0x820	7:2	0x00	**Pipe 2, Extended VC MAP_SRC/DST_0 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel
…	…	…	**…**
0x82F	7:2	0x00	**Pipe 2, Extended VC MAP_SRC/DST_15 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel
0x830	7:2	0x00	**Pipe 3, Extended VC MAP_SRC/DST_0 Register:** Bits [7:5]: SRC Virtual Channel Bits [4:2]: DST Virtual Channel

…	…	…	…
0x83F	7:2	0x00	**Pipe 3, 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 routes video pipe 3 to controller 1 and assigns the extended virtual channel (VC = 4).

```
# Setup Source to Destination mappings
# Turn on controller 1 csi2_vcx_en
0x4E, 0x94A, 0xD0
# Enable pipe 3, mapping 0-3.
0x4E, 0x9CB, 0x07

# Mapping 0
# Source, VC = 0 and DT = FS
0x4E, 0x9CD, 0x00
# Destination, VC = 0 and DT = FS, The 2 - VC bits here are the LSB part of the VC
bits for the VC extension.
0x4E, 0x9CE, 0x00

# Mapping 1
# Source, VC = 0 and DT = Raw12
0x4E, 0x9CF, 0x2C
# Destination, VC = 0 and DT = Raw12, The 2 - VC bits here are the LSB part of the
VC bits for the VC extension.
0x4E, 0x9D0, 0x2C

# Mapping 2
# Source, VC = 0 and DT = FE
0x4E, 0x9D1, 0x01
# Destination, VC = 0 and DT = FE, The 2 - VC bits here are the LSB part of the VC
bits for the VC extension.
0x4E, 0x9D2, 0x01

# D-PHY Destination is set to controller 1 for mapping 0-3
0x4E, 0x9ED, 0x15

# Setup for Extended virtual channels
# Set the 3 MSB for the destination mapping. This is the MSB for mapping 0.
0x4E, 0x830, 0x04
# Set the 3 MSB for the destination mapping. This is the MSB for mapping 1.
0x4E, 0x831, 0x04
# Set the 3 MSB for the destination mapping. This is the MSB for mapping 2.
0x4E, 0x832, 0x04
```

Software Override

The software override is used to manually override the video data type (DT) (i.e., packet header), virtual channel number (VC), 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 desired DT, VC, and bpp with the software override. See Table 19 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_x[5:0] E.g., RGB888: DT = 0x24 = 0b100100
- VC: soft_vc_x[3:0] E.g., VC: 3 = 0x03 = 0b0011
- bpp: soft_bpp_x[4:0] E.g., RAW12: bpp = 0xC = 0b01100

The data type, virtual channel, and bpp overrides must be enabled for them to take effect. There are multiple options to accomplish this within the devices:

The first option:
Enable override_bpp_vc_dt_x bits for video streams that require VC, DT, and bpp to all be overridden.

The second option:
Enable OVERRIDE_BPP_DT_x register 0x456. This is used for video streams that require the DT and bpp, or VC to be overridden independently. Bits [7:4], are used to enable the overrides for virtual channels only for each pipe. Bits [3:0], are used to enable the overrides for DT and bpp only for each pipe.

The third option:
Enable OVERRIDE_VC_BITS_2_and_3. This bit enables a unique virtual channel override for each pipe data stream. This is meant to be a convenient method to reduce programming for customers that require unique virtual channels. Pipes 0-3 will have the unique virtual channels, VC0, VC4, VC8, and VC12.

Software Override Registers

Table 19. Software Override Register Table

Register	Bits	Default Value	Description
0x0457	0	0x00	**OVERRIDE_VC_BITS_2_and_3:** Bit 0: Enable Override to force each pipe to a unique virtual channel. Unique virtual channels are VC0, VC4, VC8, and VC12.
0x0456	7:0	0x00	**Override VC, or DT and bpp Independently:** Bit 7: Enable VC override for pipe 3 Bit 6: Enable VC override for pipe 2 Bit 5: Enable VC override for pipe 1 Bit 4: Enable VC override for pipe 0 Bit 3: Enable DT and bpp override for pipe 3 Bit 2: Enable DT and bpp override for pipe 2 Bit 1: Enable DT and bpp override for pipe 1 Bit 0: Enable DT and bpp override for pipe 0 0 = Override Disabled 1 = Override Enabled
0x040B	7:3	0x00	**Pipe 0 BPP Software Override Register:** Software override video data BPP.
0x040C	7:0	0x00	**Pipe 1 and 0 VC Software Override Register:** Bits [7:4]: Pipe 1 VC. Bits [3:0]: Pipe 0 VC.
0x040D	7:0	0x00	**Pipe 3 and 2 VC Software Override Register:** Bits [7:4]: Pipe 3 VC. Bits [3:0]: Pipe 2 VC.
0x040E	7:0	0x00	**Pipe 1 (H) and 0 DT Software Override Register:** Bits [7:6]: Pipe 1 DT (High Bits [5:4]). Bits [5:0]: Pipe 0 DT.
0x040F	7:0	0x00	**Pipe 2 (H) and 1 (L) DT Software Override Register:** Bits [7:4]: Pipe 2 DT (High Bits [5:2]). Bits [3:0]: Pipe 1 DT (Low Bits [3:0]).
0x0410	7:0	0x00	**Pipe 3 and 2 (L) DT Software Override Register:** Bits [7:2]: Pipe 3 DT. Bits [1:0]: Pipe 2 DT (Low Bits [1:0]).
0x0411	7:0	0x00	**Pipe 2 (H) and Pipe 1 BPP Software Override Register:** Bits [7:5]: Pipe 2 BPP (High Bits [4:2]). Bits [4:0]: Pipe 1 BPP.
0x0412	6:0	0x00	**Pipe 3 and Pipe 2 (L) BPP Software Override Register:** Bit 7: Reserved. Bits [6:2]: Pipe 3 BPP. Bits [1:0]: Pipe 2 BPP (Low Bits [1:0]).
0x0415	7:6	0x00	**Pipe 1 and 0 Software Override Enable Register:** Bit 7: Pipe 1. Bit 6: Pipe 0. 0 = Disable, 1 = Enable.
0x0418	7:6	0x00	**Pipe 3 and 2 Software Override Enable Register:** Bit 7: Pipe 3. Bit 6: Pipe 2. 0 = Disable, 1 = Enable.

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 0, 1, 2, and 3.
# 12 BPP override for pipe 0
0x4E,0x040B,0x62
# VC 0 override for pipe 0/1/2/3
0x4E,0x040C,0x00
0x4E,0x040D,0x00
# RAW12 (0x2C) DT override for pipe 0/1/2/3
0x4E,0x040E,0xAC
0x4E,0x040F,0xBC
0x4E,0x0410,0xB0
# 12 BPP override for pipe 1/2/3
0x4E,0x0411,0x6C
0x4E,0x0412,0x30
# Enable the software override for 0/1/2/3; Data rate = 600Mbps/lane
0x4E,0x0415,0xE6
0x4E,0x0418,0xE6
```

I2C Control Channels

Overview

The quad deserializer provide two independent, fully featured I2C control channels (Port 0, and Port 1). Device registers can be accessed locally from any of the CC ports. The primary I2C, Port 0, can access the deserializer, connected serializers, and the remote peripherals connected to the serializer's control channel port. The secondary, Port 1, has access to the deserializer and remote-side devices but cannot access serializer registers. All ports may be utilized at the same time, if only one port is accessing the deserializer registers.

Arbitration between the ports is performed on a first come first served basis; lower indexed ports are given highest priority in the case of a simultaneous local access. The deserializer will suspend I2C accesses with clock stretching until the current I2C host access is complete and an I2C stop condition is received. To remove local port register access, the port must be disabled. When making changes to any of the serializer or deserializers I2C configuration, such as enabling or disabling an I2C port, a 10us delay is required between the write acknowledgement and next I2C transaction.

Note: Each I2C port can be simultaneously routed to any combination of GMSL links.

Port Access and Routing

By default, Port 0 is the primary, full-featured I2C port with remote device programming access across the GMSL2 link. Port 1 (i.e., the secondary I2C) can be tunneled to the remote side of the link and used as a pass-through channel for peripheral communication. However, the control channel crossover feature (programmed with CC_CROSSOVER_SEL) allows Port 1 to be configured as the primary (main) control channel and enables crossover access to the remote serializer registers.

The two I2C ports have independently configured access to the control channel and pass-through channels for each of the four GMSL links (Figure 15). Each link's pass-through channels can be separately enabled or disabled, providing extensive routing flexibility between the I2C ports and the remote devices. Up to four serializer devices can connect and maintain I2C access to the quad deserializers via four independent serial links. Remote and downstream device I2C access can be modified with the DIS_REM_CC bits in register 0x03, but only crossover selection can change the port access to the serializer. Disabling the remote communication channel is a one-sided command, meaning that if a microcontroller is located on the serializer side may encounter I2C timeouts if it sends commands to a disabled control channel.

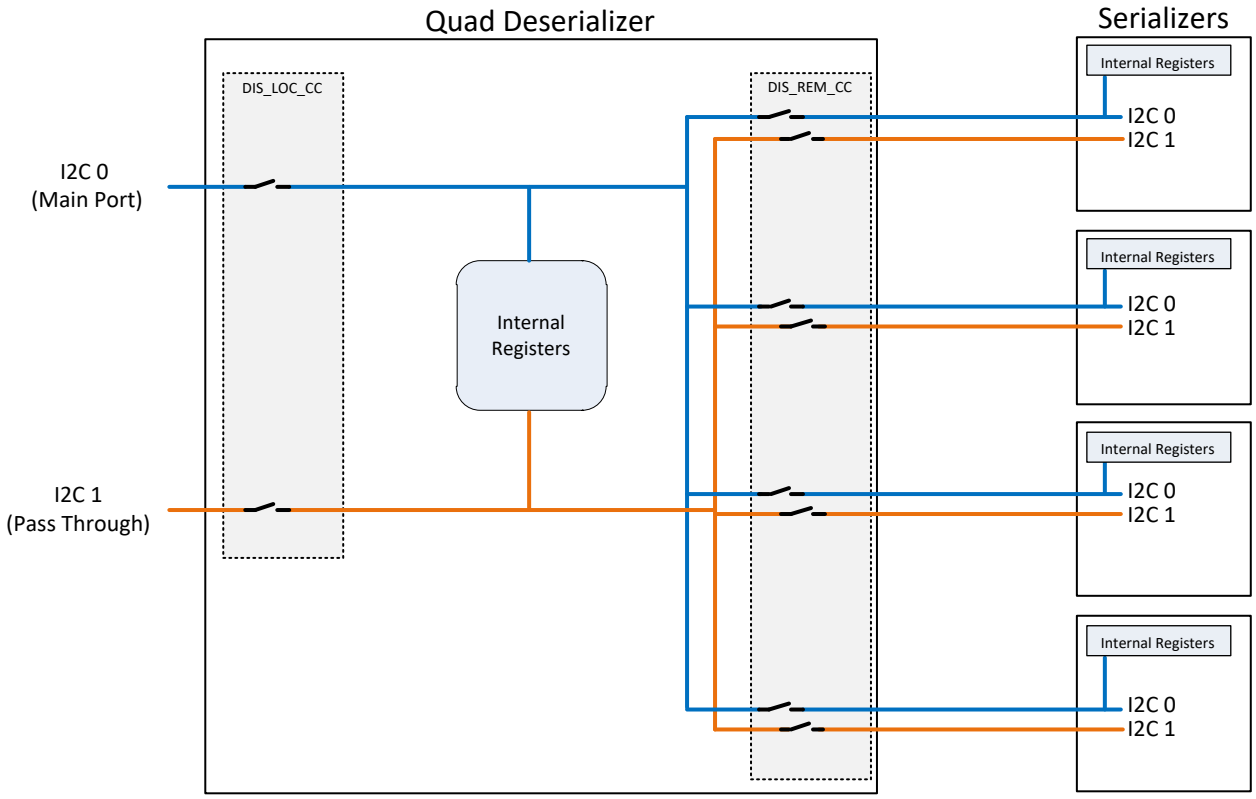

Figure 15. I2C Port Access and Routing

I2C Crossover Function

GMSL2 CSI-2 quad deserializers features a control channel crossover switch that allows the selection of a new primary port, both for redundancy and multi-master operation. It can be enabled in either GMSL1 or GMSL2 mode.

By default, Port 0 is the primary, full-featured I2C with remote device programming access across the serial link. I2C Port 1 can become the primary port and gain access to the downstream serializer and devices by enabling the crossover function.

- Port 0: I2C_0 (SDA pin 28/SCL pin 29)
- Port 1: I2C_1 (MFP7 pin 26/ MFP8 pin 27)

Note: All I2C ports have access to the internal deserializer registers and can change the device operation. Use care to limit deserializer overwrites. It is the responsibility of the system designer to prevent a single I2C host from monopolizing I2C accesses with repeated start conditions. I2C accesses must be balanced using timely stop conditions by each host controller.

The control channel crossover feature allows port 0 or 1 to be configured as the main control channel port and enables crossover access to the primary serializer I2C channel and serializer registers. When the crossover is active, the selected crossover port is switched internally with primary Port 0. See Figure 16 for an example of crossover switching. Relevant control channel registers are presented in Table 20.

Note: Only one primary I2C control channel can be active at a time.

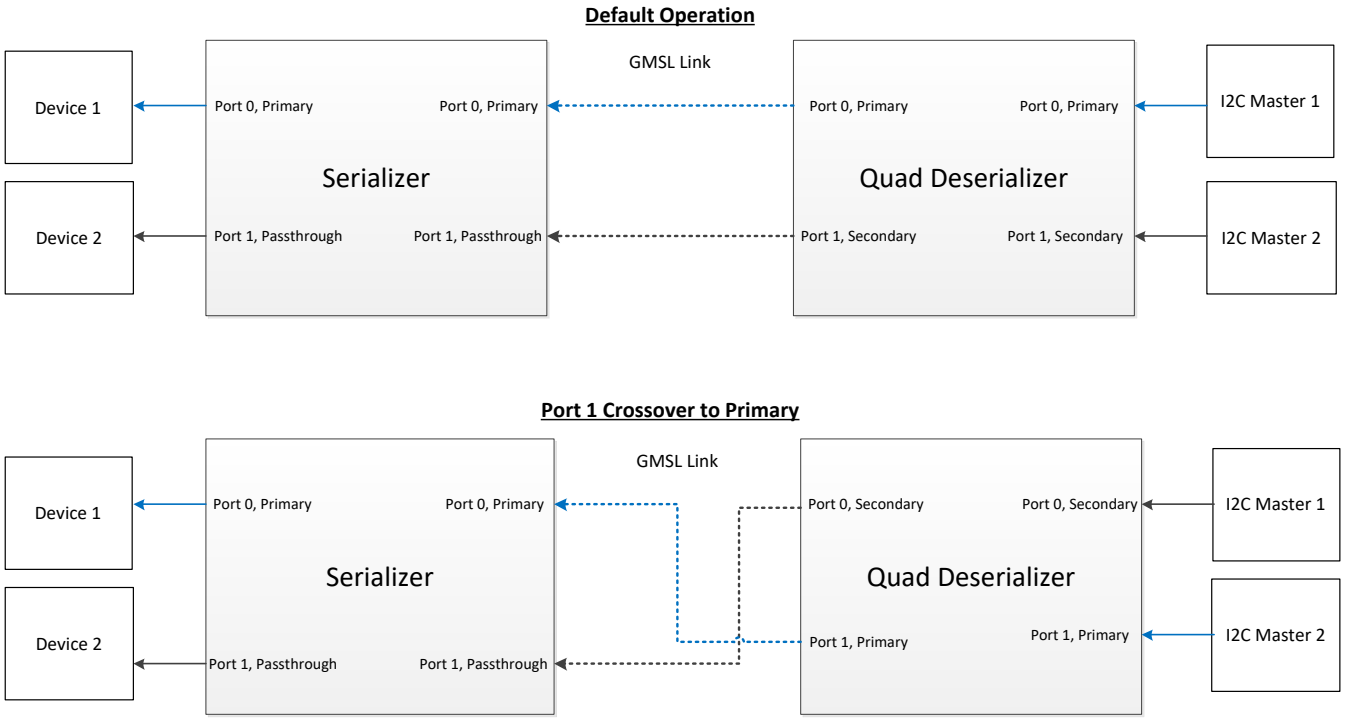

Figure 16. I2C Crossover Function

Control Channel Register

Table 20. Control Channel Registers

Register	Bits	Default Value	Description
0x0001	5:4	0xC0	**I2C CC Register:** Bits [5:4]: Disable local control channel for port 0 and 1. Bit 5 for port 1 Bit 4 for port 0 0 = Connected, 1 = Disabled
0x0003	7:0	0xAA	**Disable Remote Control Channel Register:** Bits [7:6]: Remote control channel for link D. Bits [5:4]: Remote control channel for link C. Bits [3:2]: Remote control channel for link B. Bits [1:0]: Remote control channel for link A. x0 = Enable remote control channel from port 0. x1 = Disable remote control channel from port 0. 0x = Enable remote control channel from port 1. 1x = Disable remote control channel from port 1.
0x0007	7:4	0x00	**I2C Crossover Register:** Bit [7]: Crossover for link D Bit [6]: Crossover for link C Bit [5]: Crossover for link B Bit [4]: Crossover for link A 00 = No crossover (port 0 by default) 01 = Enable crossover for port 1
0x0B04	3	0	**GMSL1 Link A Register:** Bit 3: CC_PORT_SEL Selects which I2C port is connected as the main control channel in GMSL1 mode. 0: Port 0 (RX0_SDA0, TX0_SCL0) 1: Port 1 (RX1_SDA1, TX1_SCL1)
0x0C04	3	0	**GMSL1 Link B Register:** Bit 3: CC_PORT_SEL Selects which I2C port is connected as the main control channel in GMSL1 mode. 0: Port 0 (RX0_SDA0, TX0_SCL0) 1: Port 1 (RX1_SDA1, TX1_SCL1)
0x0D04	3	0	**GMSL1 Link C Register:** Bit 3: CC_PORT_SEL Selects which I2C port is connected as the main control channel in GMSL1 mode. 0: Port 0 (RX0_SDA0, TX0_SCL0) 1: Port 1 (RX1_SDA1, TX1_SCL1)
0x0E04	3	0	**GMSL1 Link D Register:** Bit 3: CC_PORT_SEL Selects which I2C port is connected as the main control channel in GMSL1 mode. 0: Port 0 (RX0_SDA0, TX0_SCL0) 1: Port 1 (RX1_SDA1, TX1_SCL1)

Control Channel Programming Example

Disable Port 1 I2C

This example disables port 1 I2C control channel.

```
# Disable port 1 local control channel
0x4E,0x0001,0xE0
```

Enable Port 1 Crossover on Link A in GMSL2 Mode

This example enables the crossover of I2C Port 1, making it the main I2C channel.

```
# Switch I2C port 0 to have the local access only
# Set I2C port 1 to be the main control channel for both local and remote
# Disable link A remote access for port 0.
0x4E,0x0003,0xA9
# Enable I2C port 1 crossover on link A
0x4E,0x0007,0x10
```

I2C Broadcasting

GMSL2 CSI-2 quad deserializers can broadcast to all addresses by default, but it is often the case that each device will require 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 will ACK for all remote GMSL I2C ports.

An example of I2C broadcasting is shown in the block diagram below (Figure 17). A GMSL2 CSI-2 quad deserializer is connected to four identical camera modules. Each camera module comprises a GMSL2 serializer at I2C address 0x80 and an image sensor at address 0x20. See *Table 21* and *Table 22* below for addressing information.

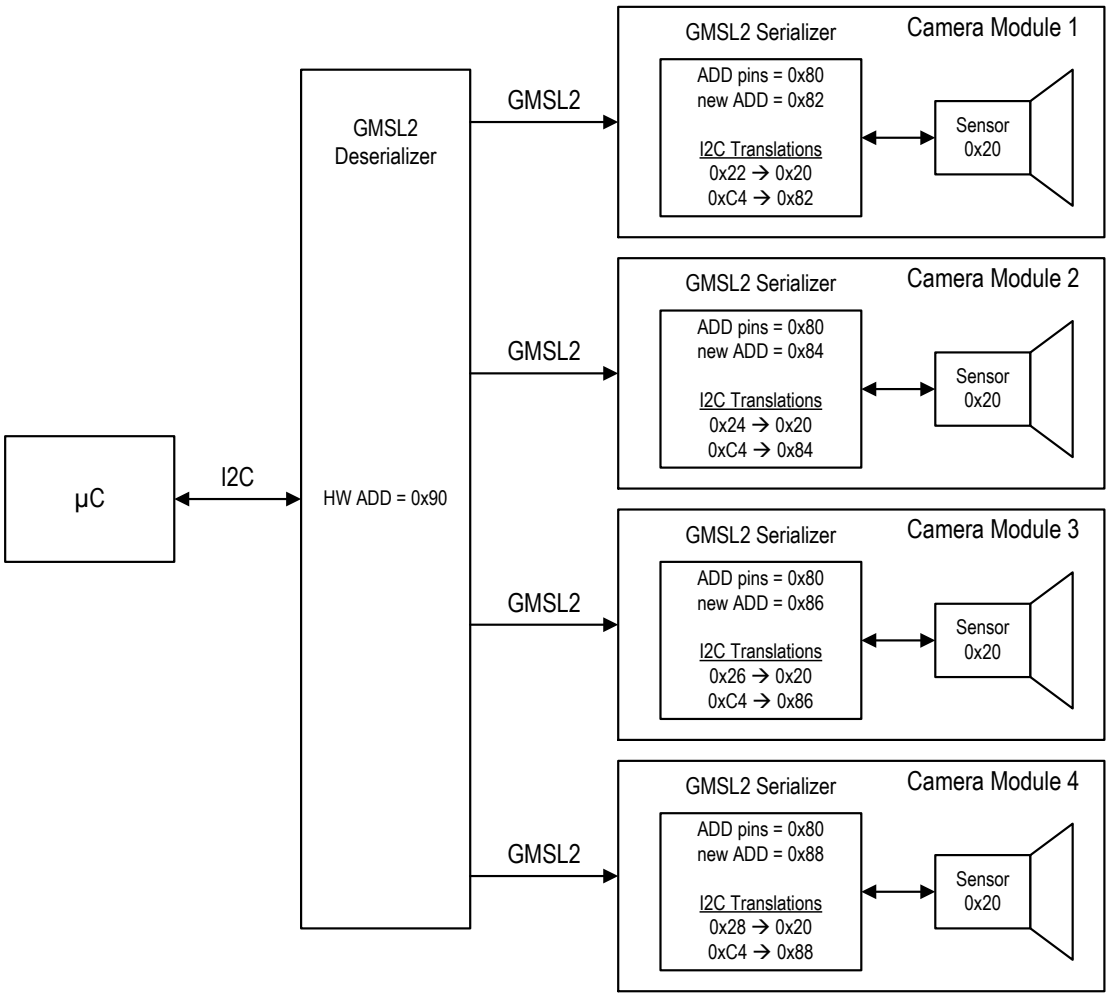

Figure 17. I2C Broadcasting MAX96724

I2C Broadcasting GMSL2 Use Case Example

The procedure for the I2C broadcasting example (Figure 17) is described below.

1) Isolate the camera module 1, by enabling link A only for remote I2C access.
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 will be 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 will be sent to address 0x20 instead.
5) Isolate the camera module 2, by enabling link B only for remote I2C access.
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 will be 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 will be sent to address 0x20 instead.
9) Isolate the camera module 3, by enabling link C only for remote I2C access.
10) Change the serializer device address in camera module 2 from 0x80 to 0x86. This is done with a register write to DEV_ADDR[6:0], located in REG0.
11) 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 0x86 in the destination register DST_A[6:0]. Thus, for the serializer in camera module 3, anything sent to address 0xC4 will be sent to address 0x86 instead.
12) Modify the second translation register in this serializer to give a unique address to the image sensor. Program 0x26 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 3, anything sent to address 0x26 will be sent to address 0x20 instead.
13) Isolate the camera module 4, by enabling link D only for remote I2C access.
14) Change the serializer device address in camera module 4 from 0x80 to 0x88. This is done with a register write to DEV_ADDR[6:0], located in REG0.
15) 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 0x88 in the destination register DST_A[6:0]. Thus, for the serializer in camera module 4, anything sent to address 0xC4 will be sent to address 0x88 instead.
16) Modify the second translation register in this serializer to give a unique address to the image sensor. Program 0x28 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 4, anything sent to address 0x28 will be sent to address 0x20 instead.

17) Now enable all the links remote main I2C port access.
18) All devices should be present on the I2C bus. Continue with any additional required system configuration.

The serializers are assigned a single device address to allow writes to all devices as a broadcast (Table 21).

Table 21. I2C Broadcasting (Quad) 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
0x86	0xC4	0x86	Serializer in Camera Module 3
0x88	0xC4	0x88	Serializer in Camera Module 4

Each image sensor is assigned a unique device address (Table 22).

Table 22. I2C Broadcasting (Quad) 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
0x20	0x26	0x20	Image Sensor in Camera Module 3
0x20	0x28	0x20	Image Sensor in Camera Module 4

I2C Broadcasting Programming Examples

I2C Broadcasting GMSL2 Programming Examples

This example sets up the use case from Figure 17 for GMSL2 links.

```
# Enable Link A remote control channel only
0x4E,0x0003,0xFE
# 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 only
0x4E,0x0003,0xFB
# 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 C remote control channel only
0x4E,0x0003,0xEF
# Change I2C address for this Link C serializer
0x80,0x0000,0x86
# Set Ser source to 0xC4
0x86,0x0042,0xC4
# Set Ser destination to 0x86
0x86,0x0043,0x86
# Set Image sensor source to 0x26
0x86,0x0044,0x26
# Set Image sensor destination to 0x20
0x86,0x0045,0x20
# Enable Link D remote control channel only
0x4E,0x0003,0xBF
# Change I2C address for this Link D serializer
0x80,0x0000,0x88
# Set Ser source to 0xC4
0x88,0x0042,0xC4
# Set Ser destination to 0x88
0x88,0x0043,0x88
# Set Image sensor source to 0x28
0x88,0x0044,0x28
# Set Image sensor destination to 0x20
0x88,0x0045,0x20
# Enable All Links A-D remote control channel
0x4E,0x0003,0xAA
```

I2C Broadcasting GMSL1 Programming Examples

This example sets up GMSL1 serializers and downstream devices to have I2C broadcasting addresses.

```
# Enable Link A only in GMSL1 Mode
0x4E,0x0006,0x01
# Delay ~5ms
# Change I2C address for this Link A serializer
0x80,0x0000,0x82
# Set Ser source to 0xC4
0x82,0x0009,0xC4
# Set Ser destination to 0x82
0x82,0x000A,0x82
# Set Image sensor source to 0x22
0x82,0x000B,0x22
# Set Image sensor destination to 0x20
0x82,0x000C,0x20
# Enable Link B only in GMSL1 Mode
0x4E,0x0006,0x02
# Delay ~5ms
# Change I2C address for this Link B serializer
0x80,0x0000,0x84
# Set Ser source to 0xC4
0x84,0x0009,0xC4
# Set Ser destination to 0x84
0x84,0x000A,0x84
# Set Image sensor source to 0x24
0x84,0x000B,0x24
```

```
# Set Image sensor destination to 0x20
0x84,0x000C,0x20
# Enable Link C only in GMSL1 Mode
0x4E,0x0006,0x04
# Delay ~5ms
# Change I2C address for this Link C serializer
0x80,0x0000,0x86
# Set Ser source to 0xC4
0x86,0x0009,0xC4
# Set Ser destination to 0x86
0x86,0x000A,0x86
# Set Image sensor source to 0x26
0x86,0x000B,0x26
# Set Image sensor destination to 0x20
0x86,0x000C,0x20
# Enable Link D only in GMSL1 Mode
0x4E,0x0006,0x08
# Delay ~5ms
# Change I2C address for this Link D serializer
0x80,0x0000,0x88
# Set Ser source to 0xC4
0x88,0x0009,0xC4
# Set Ser destination to 0x88
0x88,0x000A,0x88
# Set Image sensor source to 0x28
0x88,0x000B,0x28
# Set Image sensor destination to 0x20
0x88,0x000C,0x20
# Enable All Links in GMSL1 mode
0x4E,0x0006,0x0F
```

Concatenation

Typically, video data from multiple video pipes can be aggregated via mapping to one MIPI port through a virtual channel (VC) on a first-come, first-served (FCFS) basis. However, in some cases the SoC cannot parse VCs, and it is recommended to combine multiple streams into a single super frame (Figure 18). The SoC can then manipulate the data and separate the images if needed. This operation requires that pipe-to-controller mapping be disabled and that video sources be frame-synchronized. In this mode, all video streams share a single VC. This is also referred to as synchronous aggregation. Table 23 contains frame concatenation registers.

GMSL2 CSI-2 quad deserializers support either 4WxH (side-by-side) or Wx4H (line-interleaved) concatenation (Figure 19). Note that concatenation can be used with fewer than four streams. The super frame aggregated output is shown in Figure 20. In pixel mode both forms of concatenation are supported, but in tunnel mode only Wx4H concatenation may be used.

- **4WxH mode**: image pixel data is concatenated as a super line; number of lines remains the same.
- **Wx4H mode**: pixel resolution remains the same; lines are concatenated.

In 4WxH mode, the expected number of pixels for each stream is based on the bpp rate:

- If bpp ≤ 16, the number of pixels in a line must be a multiple of 8.

- If bpp > 16, the number of pixels in a line must be a multiple of 4.

If the above criteria are not met, black pixels will fill the remainder. For example, if four RAW12 (bpp = 12) camera streams (1026x800 each) are combined into a 4WxH super frame, the total number of pixels in a line is expected to be 4128 instead of 4104. This is because 1026 is not divisible by 8, and the closest available multiple of 8 is 1032.

In either in FCFS or concatenation mode, the MIPI output will continue if a link loses lock, and the LOCK pin is de-asserted. However, the MIPI output stops if all four links lose lock.

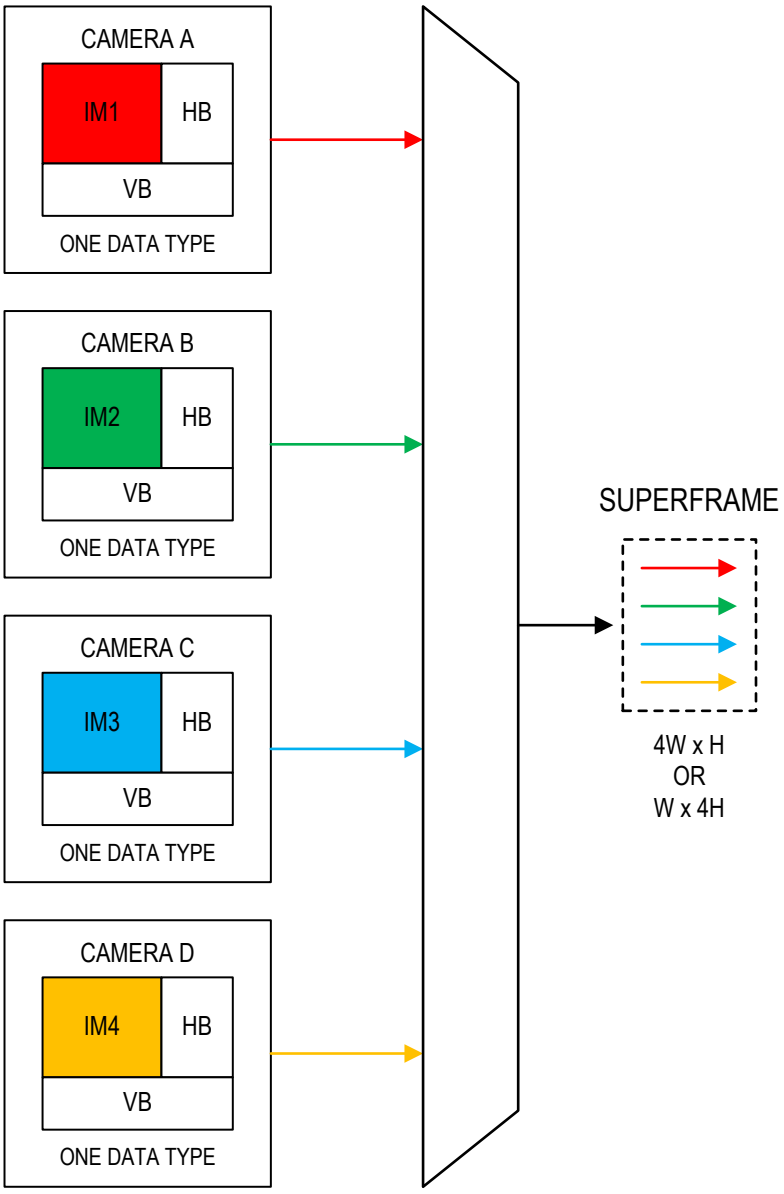

Figure 18. Video Pipe Input and Concatenated Super frame Output

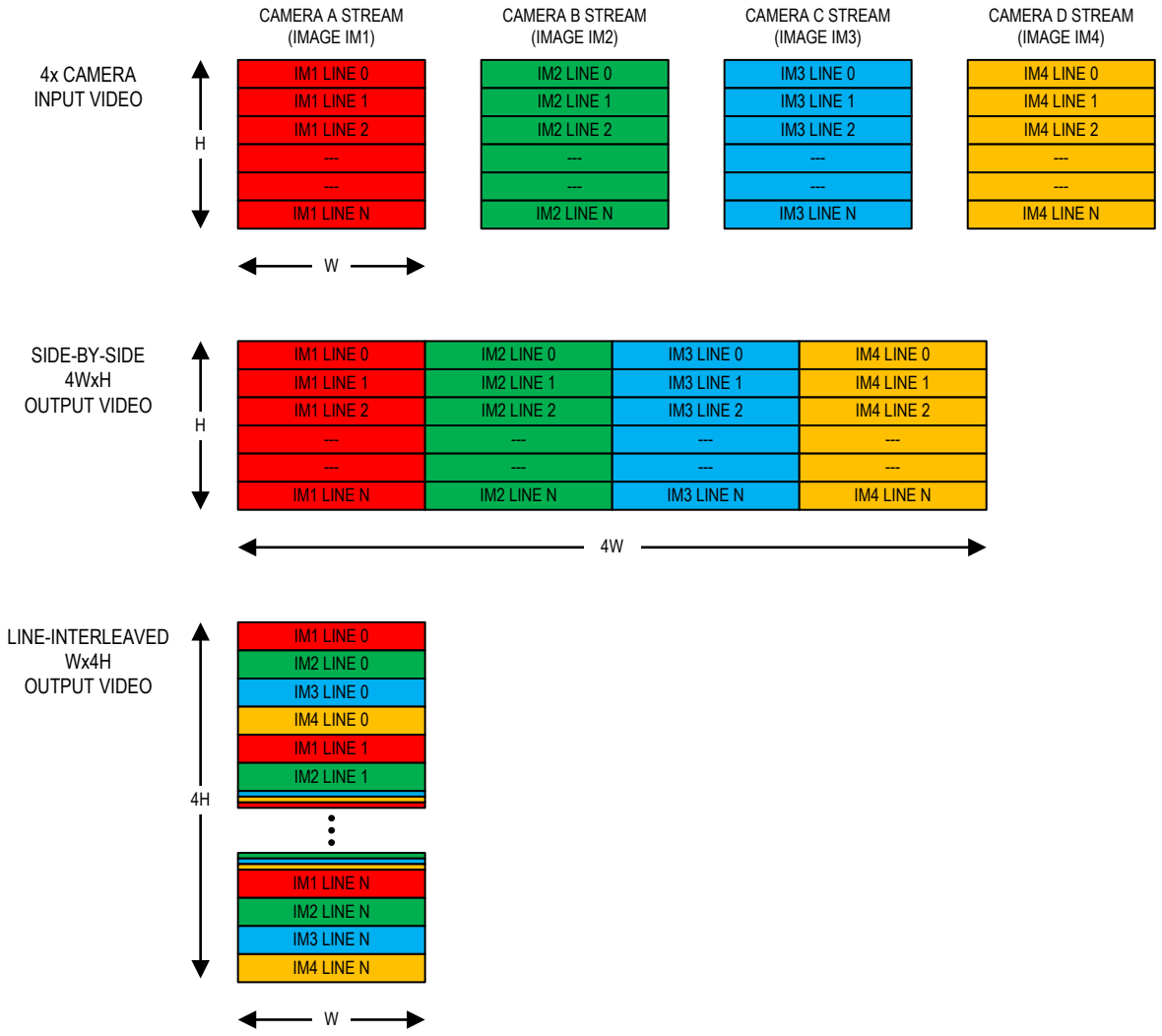

Figure 19. Concatenation (4WxH and Wx4H)

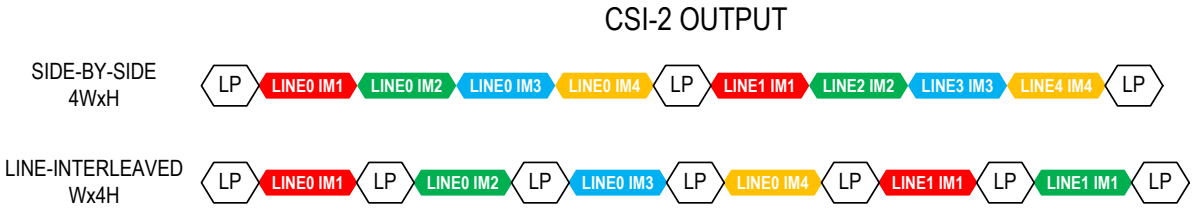

Figure 20. Super Frame Aggregation

Table 23. Frame Concatenation Registers

Register	Bits	Default Value	Description
0x0931	7:0	0x00	**MIPI Controller 0 Concatenation Register:** Bit 7: 0 = Wx4H mode, 1 = 4WxH mode. Bit 6: Reserved. Bits [5:4]: Select for the first line to concatenate; All others follow in order bit [3:0]. 00 = Select pipe 0 to be the master 01 = Select pipe 1 to be the master 10 = Select pipe 2 to be the master 11 = Select pipe 3 to be the master Bit 3: 0 = disable, 1 = concatenate video pipe 3. Bit 2: 0 = disable, 1 = concatenate video pipe 2. Bit 1: 0 = disable, 1 = concatenate video pipe 1. Bit 0: 0 = disable, 1 = concatenate video pipe 0.
0x0971	7:0	0x00	**MIPI Controller 1 Concatenation Register:** Bit 7: 0 = Wx4H mode, 1 = 4WxH mode. Bit 6: Reserved. Bits [5:4]: Select for the first line to concatenate; All others follow in order bit [3:0]. 00 = Select pipe 0 to be the master 01 = Select pipe 1 to be the master 10 = Select pipe 2 to be the master 11 = Select pipe 3 to be the master Bit 3: 0 = disable, 1 = concatenate video pipe 3. Bit 2: 0 = disable, 1 = concatenate video pipe 2. Bit 1: 0 = disable, 1 = concatenate video pipe 1. Bit 0: 0 = disable, 1 = concatenate video pipe 0.
0x09B1	7:0	0x00	**MIPI Controller 2 Concatenation Register:** Bit 7: 0 = Wx4H mode, 1 = 4WxH mode. Bit 6: Reserved. Bits [5:4]: Select for the first line to concatenate; All others follow in order bit [3:0]. 00 = Select pipe 0 to be the master 01 = Select pipe 1 to be the master 10 = Select pipe 2 to be the master 11 = Select pipe 3 to be the master Bit 3: 0 = disable, 1 = concatenate video pipe 3. Bit 2: 0 = disable, 1 = concatenate video pipe 2. Bit 1: 0 = disable, 1 = concatenate video pipe 1. Bit 0: 0 = disable, 1 = concatenate video pipe 0.
0x09F1	7:0	0x00	**MIPI Controller 3 Concatenation Register:** Bit 7: 0 = Wx4H mode, 1 = 4WxH mode. Bit 6: Reserved. Bits [5:4]: Select for the first line to concatenate; All others follow in order bit [3:0]. 00 = Select pipe 0 to be the master 01 = Select pipe 1 to be the master 10 = Select pipe 2 to be the master 11 = Select pipe 3 to be the master Bit 3: 0 = disable, 1 = concatenate video pipe 3. Bit 2: 0 = disable, 1 = concatenate video pipe 2.

			Bit 1: 0 = disable, 1 = concatenate video pipe 1. Bit 0: 0 = disable, 1 = concatenate video pipe 0.

Programming Example

The following example configures 4WxH concatenation on Controller 1 for port A MIPI output in 2x4 mode.

```
# Enable 4WxH on Controller 1 for port A MIPI output in 2x4 mode
# All pipes are enabled and pipe 1 is the master
0x4E,0x0971,0x9F
```

Frame Synchronization (FSYNC)

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

There are two types of FSYNC methods available on the GMSL2 CSI-2 quad deserializers: Internal frame sync and external frame sync. Internal frame sync indicates the GMSL2 CSI-2 quad deserializer generates the sync signal internally from its internal clock. The sync signal frequency must be specified in terms of the onboard crystal clock (25MHz) in the FSYNC Period registers *0x04A5-0x04A7*. The deserializer may be configured as a master, that generates the FSYNC and outputs it on an MFP pin (MFP0 or MFP7), or as a slave. If configured as a slave, it may accept an FSYNC output from another deserializer that is configured as the master.

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

The selection between the external or internal FSYNC is configured with the *FSYNC_MODE* bit field.

Table 24 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)	Master 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 master device	SoC cannot create FSYNC signal or GPIO is used as FSYNC output and drives a slave device.

In 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 25 contains frame sync registers.

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

Table 25. Frame Sync Registers

Register	Bits	Default Value	Description
0x04A0	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 = MFP7 Bit 4: EN_VS_GEN Keep it 0 (for most of cases) to use VS received from source. 0 = Use VS from source; VS is not generated internally. 1 = VS is generated internally 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 slave device; Set this mode for the master deserializer when there are multiple. 10 = FSYNC off; GPIO is used as FSYNC input driven by a master device; Set this mode for slave 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
0x04A2	7:5	100	**FSYNC Master Link Selection Register:** 000 = Select pipe 0 to be the master 001 = Select pipe 1 to be the master 010 = Select pipe 2 to be the master 011 = Select pipe 3 to be the master 1xx = Auto Select
0x04A5	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.)
0x04A6	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.)
0x04A7	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.)

0x04AF	7:0	0x9F	**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 3:** FS_LINK_3 in FSYNC generation. 0 = Do not include video pipe 3. 1 = Include video pipe. **Bit 2:** FS_LINK_2 in FSYNC generation. 0 = Do not include video pipe 2. 1 = Include video pipe. **Bit 1:** FS_LINK_1 in FSYNC generation. 0 = Do not include video pipe 1. 1 = Include video pipe. **Bit 0:** FS_LINK_0 in FSYNC generation. 0 = Do not include video pipe 0. 1 = Include video pipe.
0x04B0	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.
0x04B1	7:3	0x00	**FSYNC TX ID** Bits [7:3]: Select the GPIO for FSYNC to transmit to.
0x04B5	7:0	0x00	**FSYNC Difference Register (read only):** Bits [7:0]: Report the difference (Low Bits [7:0]) between the fastest and the slowest frame in terms of master PCLK cycles.
0x04B6	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 the slowest frame in terms of master PCLK cycles.
0x04B7	5	0x00	**FSYNC Reset:** Bit 5: FSYNC_RST_MODE 0 = Legacy Mode 1 = Start frame sync state machine regardless of video locks.

Programming Examples

Internal FSYNC (GMSL1)

This example demonstrates the programming sequence required to enable internal FSYNC in 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
#0x4E,0x04A2,0x00
0x4E,0x04AF,0xCF
# Set FSYNC period to 25M/30 clock cycles. CLK = 25MHz. Sync freq = 30Hz
0x4E,0x04A7,0x0C
0x4E,0x04A6,0xB7
0x4E,0x04A5,0x35
# Enable FSYNC transmission to serializer. Select GPI_1 (MAX96724 MFP2)
0x4E,0x0B08,0x71
0x4E,0x0C08,0x71
0x4E,0x0D08,0x71
0x4E,0x0E08,0x71
# Enable Internal FSYNC manual mode
0x4E,0x04A0,0x04
# 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 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
#0x4E,0x04A2,0x00
0x4E,0x04AF,0xCF
# Set FSYNC period to 25M/30 clock cycles. CLK = 25MHz. Sync freq = 30Hz
0x4E,0x04A7,0x0C
0x4E,0x04A6,0xB7
0x4E,0x04A5,0x35
# Set FSYNC TX ID to 1 to match MFP1 on Ser side.
0x4E,0x04B1,0x08
# Enable GPIO_RX_EN on Ser MFP1
0x80,0x02C1,0x84
# Enable Internal FSYNC manual mode
0x4E,0x04A0,0x04
# 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 external FSYNC in 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
0x4E,0x04A0,0x08
0x4E,0x04AF,0x1F
# Enable GPI/GPO function in GMSL1 mode for each link; Select GPI_1 (MFP2) for
MAX96724
0x4E,0x0B08,0x61
0x4E,0x0C08,0x61
0x4E,0x0D08,0x61
0x4E,0x0E08,0x61
```

External FSYNC (GMSL2)

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

```
# This is GMSL2 Ext. FSYNC example; MAX96724 MFP2/SER MFP1 are used for GPI/GPO.
# Update SER MFP1 RX ID = 2 to match MFP2 of MAX96724
0x80,0x02C3,0x02
# Config SER MFP1 to forward GPIO from the MAX96724
0x80,0x02C1,0x84
# Set Internal FSYNC off, GPIO is used for FSYNC, type = GMSL2
0x4E,0x04A0,0x08
0x4E,0x04AF,0x9F
# Config MAX96724 MFP2 to receive external FSYNC signal for each link
0x4E,0x0306,0x83
0x4E,0x033D,0x22
0x4E,0x0374,0x22
0x4E,0x03AA,0x22
```

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 video pattern generators (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 25MHz crystal clock (i.e., REFCLK input) to derive four different pixel clock (PCLK) options. Link lock is not required for the VPG to be used.

The VPG has its own register block settings for timing configurations. *Table 26* 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 25/75MHz for all pipes or 150/375MHz on a per pipe basis. This internal PCLK is not related to the MIPI CSI port clock rate, which will need to be set to accommodate the VPG data stream. See *Table 27* for configuration details. *Table 28* contains reference information for VPG PCLK selection.

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

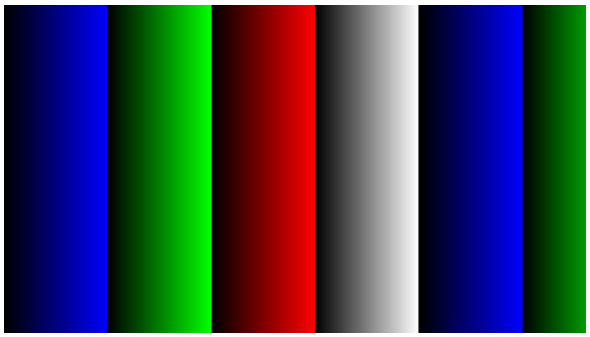

Figure 21. VPG – Gradient Pattern

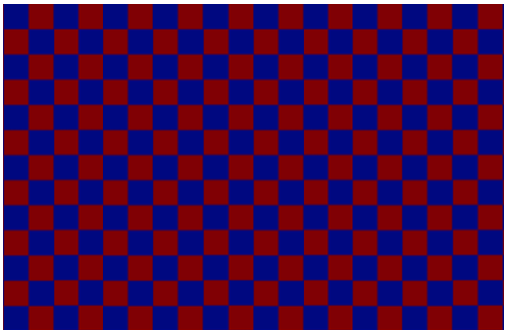

Figure 22. VPG – Checkerboard Pattern

Table 26. Video Pattern Registers

Register (VPG0/VPG1)	Bits	Default Value	Description
0x1050	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.
0x1051	5:4	00	**Pattern Generator Mode Register:** Bits [5:4]: Pattern selection. 01 = Color Gradient 10 = Checkerboard
0x1052	7:0	0x00	**VS Delay Register 2:** VS Delay in terms of PCLK cycles. (Bits [23:16])
0x1053	7:0	0x00	**VS Delay Register 1:** VS Delay in terms of PCLK cycles. (Bits [15:8])
0x1054	7:0	0x00	**VS Delay Register 0:** VS Delay in terms of PCLK cycles. (Bits [7:0])
0x1055	7:0	0x00	**VS High Register 2:** VS High Period in terms of PCLK cycles. (Bits [23:16])
0x1056	7:0	0x00	**VS High Register 1:** VS High Period in terms of PCLK cycles. (Bits [15:8])
0x1057	7:0	0x00	**VS High Register 0:** VS High Period in terms of PCLK cycles. (Bits [7:0])
0x1058	7:0	0x00	**VS Low Register 2:** VS Low Period in terms of PCLK cycles. (Bits [23:16])
0x1059	7:0	0x00	**VS Low Register 1:** VS Low Period in terms of PCLK cycles. (Bits [15:8])
0x105A	7:0	0x00	**VS Low Register 0:** VS Low Period in terms of PCLK cycles. (Bits [7:0])
0x105B	7:0	0x00	**V2H Register 2:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [23:16])
0x105C	7:0	0x00	**V2H Register 1:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [15:8])

0x105D	7:0	0x00	**V2H Register 0:** VS edge to the rising edge of the first HS in terms of PCLK cycles. (Bits [7:0])
0x105E	7:0	0x00	**HS High Register 1:** HS High Period in terms of PCLK cycles. (Bits [15:8])
0x105F	7:0	0x00	**HS High Register 0:** HS High Period in terms of PCLK cycles. (Bits [7:0])
0x1060	7:0	0x00	**HS Low Register 1:** HS Low Period in terms of PCLK cycles. (Bits [15:8])
0x1061	7:0	0x00	**HS Low Register 0:** HS Low Period in terms of PCLK cycles. (Bits [7:0])
0x1062	7:0	0x00	**HS Count Register 1:** HS pulses per frame. (Bits [15:8])
0x1063	7:0	0x00	**HS Count Register 0:** HS pulses per frame. (Bits [7:0])
0x1064	7:0	0x00	**V2D Register 2:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [23:16])
0x1065	7:0	0x00	**V2D Register 1:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [15:8])
0x1066	7:0	0x00	**V2D Register 0:** VS edge to the rising edge of the first DE in terms of PCLK cycles. (Bits [7:0])
0x1067	7:0	0x00	**DE High Register 1:** DE High Period in terms of PCLK cycles. (Bits [15:8])
0x1068	7:0	0x00	**DE High Register 0:** DE High Period in terms of PCLK cycles. (Bits [7:0])
0x1069	7:0	0x00	**DE Low Register 1:** DE Low Period in terms of PCLK cycles. (Bits [15:8])
0x106A	7:0	0x00	**DE Low Register 0:** DE Low Period in terms of PCLK cycles. (Bits [7:0])
0x106B	7:0	0x00	**DE Count Register 1:** DE pulses per frame. (Bits [15:8])
0x106C	7:0	0x00	**DE Count Register 0:** DE pulses per frame. (Bits [7:0])
0x106D	7:0	0x00	**Gradient Increment Register:** Set color gradient increment.
0x106E	7:0	0x00	**CHRK_COLOR_A_L Register:** Checkerboard mode color A low byte. RGB888 color Red (0-255).
0x106F	7:0	0x00	**CHRK_COLOR_A_M Register:** Checkerboard mode color A mid byte. RGB888 color Green (0-255).
0x1070	7:0	0x00	**CHRK_COLOR_A_H Register:** Checkerboard mode color A high byte. RGB888 color Blue (0-255).
0x1071	7:0	0x00	**CHRK_COLOR_B_L Register:** Checkerboard mode color B low byte. RGB888 color Red (0-255).
0x1072	7:0	0x00	**CHRK_COLOR_B_M Register:**

			Checkerboard mode color B mid byte. RGB888 color Green (0-255).
0x1073	7:0	0x00	**CHRK_COLOR_B_H Register:** Checkerboard mode color B high byte. RGB888 color Blue (0-255).
0x1074	7:0	0x00	**CHRK_RPT_A Register:** Checkerboard mode color A repeat count.
0x1075	7:0	0x00	**CHRK_RPT_B Register:** Checkerboard mode color B repeat count.
0x1076	7:0	0x00	**CHRK_ALT Register:** Checkerboard mode alternate line count.

Table 27. PLCK Settings Registers

Register	Bits	Default Value	Description
0x0009	1:0	00	**PCLK Freq Register:** Bit [1:0]: Set video pattern PCLK frequency. 00 = 25MHz 01 = 75MHz 10 = 150MHz (PATGEN_CLK_SRC = 0) 11 = 375MHz (PATGEN_CLK_SRC = 1)
0x01DC	7	1	**PATGEN_CLK_SRC Register:** Select clock source for video pattern on pipe 0. 0 = 150MHz, 1 = 375MHz Work together with Pattern CLK Freq register.
0x01FC	7	1	**PATGEN_CLK_SRC Register:** Select clock source for video pattern on pipe 1. 0 = 150MHz, 1 = 375MHz Work together with Pattern CLK Freq register.
0x021C	7	1	**PATGEN_CLK_SRC Register:** Select clock source for video pattern on pipe 2. 0 = 150MHz, 1 = 375MHz Work together with Pattern CLK Freq register.
0x023C	7	1	**PATGEN_CLK_SRC Register:** Select clock source for video pattern on pipe 3. 0 = 150MHz, 1 = 375MHz Work together with Pattern CLK Freq register.

Below is a reference table for VPG PCLK selection (*Table 28*). This is an alternative method of setting the pattern generator clock frequency with additional options for the PCLK frequency.

Table 28. Video Pattern PCLK Selection

Bitfield Name (Reg #)		PCLK Frequency
DEBUG_EXTRA (0x09 [1:0])	**PATGEN_CLK_SRC (0x1DC [7])**	
00	X	25MHz
01	X	75MHz
1x	0	150MHz
1x	1 (Default)	375MHz

Video Mask Output

GMSL2 CSI-2 quad deserializers can automatically mask video pipe outputs if a video pipe's video lock is lost. When enabled, zeroes are automatically inserted into the synchronized aggregated video output(s) of a video pipe that has lost video lock. This allows the other video streams to continue being transmitted uninterrupted on the MIPI interface. Video pipes can also be manually forced to mask video output streams regardless of video lock status. Reference Table 29 for configuration details.

In Figure 23, the video stream output from three cameras is aggregated via side-by-side concatenation. In Figure 24, the 'Camera 1' video pipe output is masked while the video data from the other two cameras continues to be transmitted.

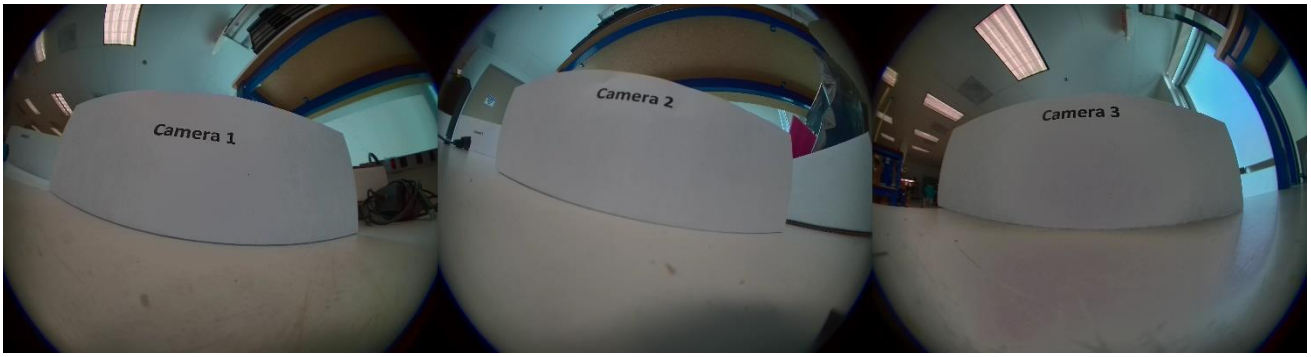

Figure 23. Concatenated Video Stream from Three Cameras

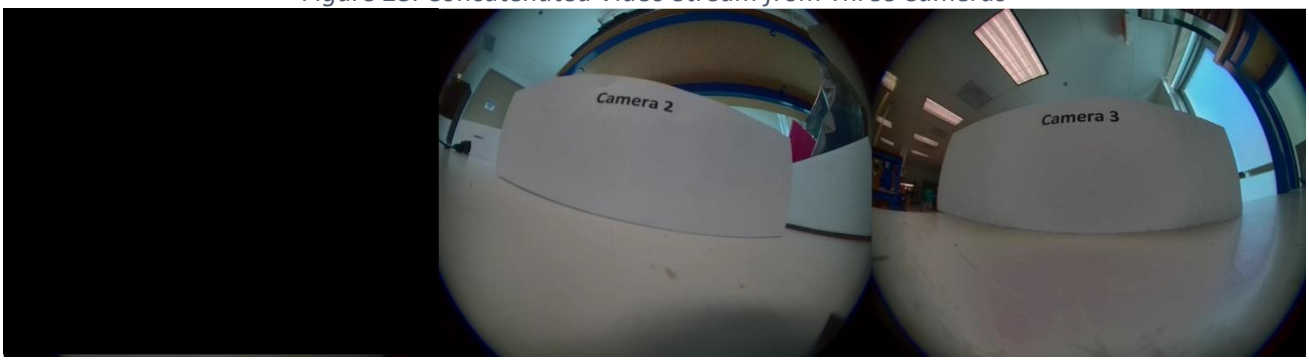

Figure 24. Concatenated Video Stream with Masked Video

Table 29. Video Mask Output Registers

Register	Bits	Default Value	Description
0x08C6	3:0	0x0000	**Auto_Mask_En register:** Automatically insert 0's into synchronized aggregated video outputs if a Video Pipe 0-3 video lock is lost. This allows the other video streams to continue being transmitted on the MIPI interface. 0bXXX1: Auto video mask enabled for Video Pipe 0 0bXX1X: Auto video mask enabled for Video Pipe 1 0bX1XX: Auto video mask enabled for Video Pipe 2 0b1XXX: Auto video mask enabled for Video Pipe 3
0x08C6	7:4	0x0000	**Force_Video_Mask register:** Forces video output to be masked (send all 0's) in 4WxH or Wx4H synchronous aggregation modes.

			0bXXX1: Force video from Video Pipe 0 to be masked. 0bXX1X: Force video from Video Pipe 1 to be masked. 0bX1XX: Force video from Video Pipe 2 to be masked. 0b1XXX: Force video from Video Pipe 3 to be masked.
0x08C6	4:0	0x0F	**Video_Mask restart:** Automatically restarts video streams that were previously masked off due to loss of video lock Bit 0: Restart video from Video Pipe 0 Bit 1: Restart video from Video Pipe 1 Bit 2: Restart video from Video Pipe 2 Bit 3: Restart video from Video Pipe 3

Bandwidth Efficiency Optimization

The 6Gbps GMSL2 forward link rate has an effective maximum video payload of approximately 5.2Gbps after accounting for the 9b/10b encoding and link protocol overhead. Similarly, the 3Gbps GMSL2 forward link rate has an approximate effective video bandwidth of 2.6Gbps. The incoming payload at serializer input must be below the relevant figure to ensure proper operation.

If an image sensor outputs MIPI CSI-2 data, the payload can be calculated by multiplying the lane rate by the number of lanes. For example, an image sensor outputting data with a lane rate of 450Mbps on four lanes:

$$Payload = 450Mbps * 4lanes = 1.8Gbps$$

Alternatively, image details may be used to calculate the payload. For example, a 2.3MP image sensor with a resolution of 1928 x 1208 running at 30fps outputs RAW12 datatype:

$$Payload \cong (1928 * 1208) * 1.3(blanking) * 30(fps) * 12(BPP) \cong 1.1Gbps$$

High-resolution cameras may require additional configuration to optimize bandwidth usage and ensure proper function of the serial link. For instance, an 8MP image sensor (RAW12) running at 30fps has a typical output of less than 4.5Gbps. This is within the data throughput limit of a GMSL2 6Gbps link, however, it will exceed the maximum allowed PCLK limit of 375MHz. Two settings (described below) provide configuration options for optimizing bandwidth efficiency.

The following equation is used to calculate the PCLK value:

$$PCLK = \frac{LANE\_CNT * LANE\_RATE}{bpp}$$

Disabling "Heartbeat" Mode

Heartbeat mode is enabled by default. This mode is designed for GMSL display video links where HS, VS, and DE(HVD) values are transmitted using GMSL short packets during video blanking. The CSI-2 protocol uses its own long data packets for video data as well as FS/FE short packets for VS, so heartbeat mode is not necessary for CSI-2 data transmission and can be turned off to minimize serial link bandwidth usage.

Heartbeat mode is configured individually for each video pipe. It should only be disabled for pipes carrying CSI-2 image data; it should not be disabled if a pipe is only carrying embedded data, which usually appears as a few lines of each frame. A video pipe must continuously receive at least 16 valid video packets within a moving time window for VIDEO_LOCK to be asserted. This window becomes approximately 100 times larger when heartbeat mode is disabled.

When the heartbeat mode is disabled LIM_HEART = 1, the timeout to lose video lock is programmable between 1-127ms. The default timeout with LIM_HEART = 1 is 10ms and can be adjusted for image sensors that utilize slow refresh rates. The timeout may be adjusted for each pipe by LIM_HEART_TIMEOUT registers 0x0160 - 0x0163. When *LIM_HEART* = 0 the timeout is 100us and it is not programmable.

To disable heartbeat mode:

- Set LIM_HEART = 1 in serializer.
- Set SEQ_MISS_EN = 0, DIS_PKT_DET = 0, and LIM_HEART = 1 in the deserializer.

Programming Example

This configuration disables heartbeat mode in both the serializer and in the quad deserializer.

```
# Efficiency updates for pipe X in MAX96717 that carries image data. LIM_HEART = 1
0x80,0x112,0x0E
# Efficiency updates for pipe 0-3 in MAX96724 that carry image data.
#SEQ_MISS_EN = 0
0x4E,0x0100,0x22
0x4E,0x0112,0x22
0x4E,0x0124,0x22
0x4E,0x0136,0x22
# Efficiency updates for pipe 0-3 in MAX96724 that carry image data. LIM_HEART = 1
0x4E,0x0106,0x0A
0x4E,0x0118,0x0A
0x4E,0x012A,0x0A
0x4E,0x013C,0x0A
```

Using Double Mode

GMSL2 CSI-2 quad deserializers have a maximum allowed internal PCLK value of 375MHz when processing image data. For a MIPI CSI-2 input, the PCLK calculation for incoming video is the total payload divided by the bpp rate. For example, an 8MP RAW8 image:

$$PCLK = \frac{4.4Gbps}{8bpp} = 550MHz$$

This exceeds the maximum allowed value (375MHz). This can be addressed by enabling double pixel mode, which doubles the bpp to save bandwidth. GMSL protocol allocates 24 bits of each packet for video content. Therefore, double pixel mode can be used for any data type less than or equal to 12bpp. In the above example, doubling 8bpp would yield a 275MHz PCLK and meet the internal PCLK requirement for the GMSL2 CSI-2 quad deserializers.

There are three bpp values that can be doubled: 8, 10, and 12bpp. The settings for each are listed below. All bpp configurations are individually programmed for each pipe with dedicated ALT_MEM_MAP registers. The * in the

settings below should be replaced with the desired pipe names (i.e., X/Y/Z/U for serializers and 0–7 for GMSL2 CSI-2 quad deserializers).

1. Double pixel mode for 8bpp
    - Set bpp8dbl* = 1, soft_bpp*_en = 1, and soft_bpp* = 16 in the serializer.
    - Set *bpp8dbl** = 1, bpp8dbl*_mode = 1, ALT_MEM_MAP8 = 1 in MAX96724.

2. Double pixel mode for 10bpp
    - Set bpp10dbl* = 1, soft_bpp*_en = 1, soft_bpp* = 20 in the serializer.
    - Set ALT_MEM_MAP10 = 1 in MAX96724.

3. Double pixel mode for 12bpp
    - Set bpp12dbl* = 1, soft_bpp*_en = 1, soft_bpp* = 24 in the serializer.
    - Set ALT_MEM_MAP12 = 1 In MAX96724.

Note: The operation for 8bpp double mode is different than 10/12bpp and requires two extra writes in the GMSL2 CSI-2 MAX96724 deserializer.

MIPI Packet Counters

The MIPI packet count registers are used to determine whether MIPI data is flowing in the GMSL2 CSI-2 quad 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 will not change if data is not flowing. Reference *Table 30* for MIPI packet count registers.

Table 30. MIPI Packet Count Registers

Register	Bits	Default Value	Description	Decode
0x08D0	3:0	0x00	Packet count of CSI-2 Controller 0	**csi2_tx*_pkt_cnt registers:** 0bXXXX: Toggling bits indicate MIPI data is active on the controller.
0x08D0	7:4	0x00	Packet count of CSI-2 Controller 1	
0x08D1	3:0	0x00	Packet count of CSI-2 Controller 2	
0x08D1	7:4	0x00	Packet count of CSI-2 Controller 3	
0x08D2	3:0	0x00	Packet count of MIPI PHY0	**phy*_pkt_cnt registers:** 0bXXXx: Toggling bits indicate MIPI data is active on the PHY.
0x08D2	7:4	0x00	Packet count of MIPI PHY1	
0x08D3	3:0	0x00	Packet count of MIPI PHY2	
0x08D3	7:4	0x00	Packet count of MIPI PHY3	

HVD Outputs and Counters

The GMSL2 CSI-2 quad deserializers are capable of outputting the VS, HS, DE, or TX-Start signals from the video pipe data streams on four MFP pins. Each of these pins can be selected to output from video pipes 0-3, and the type of signal can be selected for that specified pipe.

All four pulses can be output at the same time on the four different GPIOs.

- MFP1, 3, 7, 8 – VS, HS, DE, or TX-Start output to GPIO

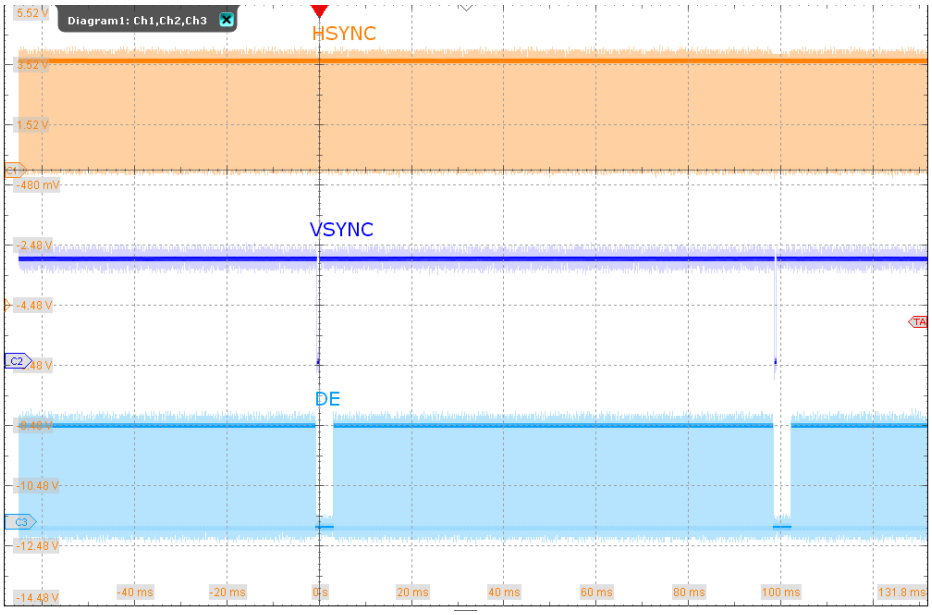

Figure 25 HS, VS, and DE outputs from GPIO's

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. In particular, the MFP7/MFP8 pins are enabled as the second I2C port by default, so this feature must be disabled.

Reference *Table 31* for sync pulse output registers.

Table 31. Sync Pulse Output Registers

Register	Bits	Default Value	Description
0x00FA	3:0	0x00	**HVD_OUT_EN register:** Bit 3: HVD Enable MFP8 Bit 2: HVD Enable MFP7 Bit 1: HVD Enable MFP3 Bit 0: HVD Enable MFP1
0x00FB	7:0	0x00	**HVD_HS_SEL register:** Bit [7:6]: MFP8 pipe select for HS Bit [5:4]: MFP7 pipe select for HS Bit [3:2]: MFP3 pipe select for HS Bit [1:0]: MFP1 pipe select for HS 00 = pipe 0 01 = pipe 1 10 = pipe 2 11 = pipe 3
0x00FC	7:0	0x00	**HVD_VS_SEL register:** Bit [7:6]: MFP8 pipe select for VS Bit [5:4]: MFP7 pipe select for VS Bit [3:2]: MFP3 pipe select for VS Bit [1:0]: MFP1 pipe select for VS 00 = pipe 0

			01 = pipe 1 10 = pipe 2 11 = pipe 3
0x00FD	7:0	0x00	**HVD_DE_SEL register:** Bit [7:6]: MFP8 pipe select for DE Bit [5:4]: MFP7 pipe select for DE Bit [3:2]: MFP3 pipe select for DE Bit [1:0]: MFP1 pipe select for DE 00 = pipe 0 01 = pipe 1 10 = pipe 2 11 = pipe 3
0x00FE	7:0	0x00	**HVD_OUT_SEL register:** Bit [7:6]: MFP8 pipe select for output Bit [5:4]: MFP7 pipe select for output Bit [3:2]: MFP3 pipe select for output Bit [1:0]: MFP1 pipe select for output 00 = HS 01 = VS 10 = DE 11 = TX-Start
0x00FF	7:0	0x00	**HVD_ST_SEL register:** Bit [7:6]: MFP8 pipe select for TX-Start Bit [5:4]: MFP7 pipe select for TX-Start Bit [3:2]: MFP3 pipe select for TX-Start Bit [1:0]: MFP1 pipe select for TX-Start 00 = pipe 0 01 = pipe 1 10 = pipe 2 11 = pipe 3

Alternatively, sync pulse signals can be monitored via registers. Registers *0x11F0/0x11F1/0x11F2* contain status flags indicating the detection of DE/HS/VS signals for each video pipe. The polarity of the HS/VS signals is accessed with registers *0x11F3/0x11F4*. HVD counters may also be read to verify the pipe count values match the expected video input values. Additionally, comparators may be setup and enabled to flag dropped frames or lines as errors. See *Table 32* for additional information.

Table 32. Sync Signal Status Registers

Register	Bits	Default Value	Description
0x11F0	3:0	N/A	**DE_DET registers:** Bits 0 through 3 correspond with video pipes 0 through 3. Bitfield name is appended with video pipe number. Indicates if DE signal is detected in video pipe. 0b0: DE is not detected 0b1: DE is detected
0x11F1	3:0	N/A	**HS_DET registers:** Bits 0 through 3 correspond with video pipes 0 through 3. Bitfield name is appended with video pipe number. Indicates if HS signal is detected in video pipe. 0b0: HS is not detected 0b1: HS is detected
0x11F2	3:0	N/A	**VS_DET registers:** Bits 0 through 3 correspond with video pipes 0 through 3. Bitfield name is appended with video pipe number. Indicates if VS signal is detected in video pipe. 0b0: VS is not detected 0b1: VS is detected
0x11F3	3:0	N/A	**HS_POL registers:** Bits 0 through 3 correspond with video pipes 0 through 3. Bitfield name is appended with video pipe number. Indicates the detected HS signal polarity in video pipe. 0b0: Active low 0b1: Active high
0x11F4	3:0	N/A	**VS_POL registers:** Bits 0 through 3 correspond with video pipes 0 through 3. Bitfield name is appended with video pipe number. Indicates the detected VS signal polarity in video pipe. 0b0: Active low 0b1: Active high
0x11F9	7:0	0x0F	**HVD_CNT_CTRL registers:** Bits [7:4]: Reset counter values for pipes 0-3. Bits [3:0]: HVD count enable for pipes 0-3.
0x11FA	7:0	0x00	**HVD_CNT_OS registers:** Bits [3:0]: Enable VS and frames per line counters in one-shot mode. Must be used when HVD_CNT_EN is disabled.
0x1202, 0x1212, 0x1222, 0x1232	5:0	0x00	**VS_CNT_CMP registers:** Bits [5:0]: VS count comparator value.

0x1203, 0x1213, 0x1223, 0x1233	3:0	0x00	**HS_CNT_CMP Most Significant Bits registers:** Bits [3:0]: HS count comparator value MSB.
0x1204, 0x1214, 0x1224, 0x1234	7:0	0x00	**HS_CNT_CMP Least Significant Bits registers:** Bits [7:0]: HS count comparator value LSB.
0x1205, 0x1215, 0x1225, 0x1235	3:0	0x00	**DE_CNT_CMP Most Significant Bits registers:** Bits [3:0]: HS count comparator value MSB.
0x1206, 0x1216, 0x1226, 0x1235	7:0	0x00	**DE_CNT_CMP Least Significant Bits registers:** Bits [7:0]: HS count comparator value LSB.
0x1207, 0x1217, 0x1227, 0x1237	5:0	N/A	**VS count registers:** Bits [5:0]: VS count value.
0x1208, 0x1218, 0x1228, 0x1238	3:0	N/A	**HS count MSB registers:** Bits [3:0]: HS count value.
0x1209, 0x1219, 0x1229, 0x1239	7:0	N/A	**HS count LSB registers:** Bits [7:0]: HS count value.
0x120A, 0x121A, 0x122A, 0x123A	3:0	N/A	**DE count MSB registers:** Bits [3:0]: DE count value.
0x120B, 0x121B, 0x122B, 0x123B	7:0	N/A	**DE count LSB registers:** Bits [7:0]: DE count value.
0x120C, 0x121C, 0x122C, 0x123C	7:5	0xE0	**Comparator Error Output enable registers:** Bit 7: VS_CNT_?_CMP_ERR_OEN Bit 6: HS_CNT_?_CMP_ERR_OEN Bit 5: DE_CNT_?_CMP_ERR_OEN 0 = Disable 1 = Enable comparator error.
0x120D, 0x121D, 0x122D, 0x123D	7:5	0x00	**Comparator Error Output enable registers:** Bit 7: VS_CNT_?_CMP_ERR_FLAG Bit 6: HS_CNT_?_CMP_ERR_FLAG Bit 5: DE_CNT_?_CMP_ERR_FLAG 0 = No Error Detected

			1 = Error Detected

Programming Examples

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

Output pipe 1's HS on MFP1, VS on MFP3, and DE on MFP7

```
# Disable I2C for MFp7 and MFP8
0x4E,0x0001,0xE0
# Enable HVD_OUT for MFP1, MFP3, and MFP7
0x4E,0x00FA,0x07
# Select Pipe 1 for HS on MFP1
0x4E,0x00FB,0x01
# Select Pipe 1 for VS on MFP3
0x4E,0x00FC,0x04
# Select Pipe 1 for DE on MFP7
0x4E,0x00FD,0x10
# Select output MFP1 = HS, MFP3 = VS, and MFP7 = DE.
0x4E,0x00FE,0x14
```

Output pipe 1 comparator setup for VS monitoring. VS = 30 fps

```
# This example is used to setup VS monitoring, but it is similar for HS and DE.
# Read current VS count
0x4E,0x1217,0x?? → 0x1E = 30 fps
# Set VS comparator to 0x1E = 30 fps
0x4E,0x1212,0x1E
# Enable VS comparator to ERRB pin
0x4E,0x121C,0x80
# Read VS comparator to ERRB flag. Note if VS is stable this will not trip.
0x4E,0x121D,0x?? → 0x00 = no errors, 0x80 = Error with VS count
```

MIPI Error Packet

MIPI Error packets may be enabled to output when uncorrectable header errors occur in tunneling mode. In the case of an uncorrectable header, the entire packet with the errors will be dropped by the GMSL link. Some SOC's may have issues dealing with the loss of a packet. In this case, a created packet is filled with null data and may be inserted in place of the dropped invalid packet.

The MIPI error packet will automatically match the word count and virtual channel of incoming data packets and it is up to the user to define the data type of the inserted packet. In D-PHY mode the supported virtual channels for this packet are 0-3, and in C-PHY mode the supported virtual channels are 0-31. The user may set the data type to any data types that are recognized by MIPI. The user may also define the virtual channel or word count by overriding the value in the MIPI error packet registers.

MIPI Error Packet Registers

Table 33. MIPI Error Packet Register Table

Register	Bits	Default Value	Description
0x08D8	7:0	0x3E	**MIPI Error Packet Enable and Data Type:** Bit 7: Error packet enable for pipe 0. 0 = Disable, 1 = Enable Bit 6: RSVD Bit [5:0]: Data type of error packet.
0x08D9	7:0	0x3E	**MIPI Error Packet Enable and Data Type:** Bit 7: Error packet enable for pipe 1. 0 = Disable, 1 = Enable Bit 6: RSVD Bit [5:0]: Data type of error packet.
0x08DA	7:0	0x3E	**MIPI Error Packet Enable and Data Type:** Bit 7: Error packet enable for pipe 2. 0 = Disable, 1 = Enable Bit 6: RSVD Bit [5:0]: Data type of error packet.
0x08DB	7:0	0x3E	**MIPI Error Packet Enable and Data Type:** Bit 7: Error packet enable for pipe 3. 0 = Disable, 1 = Enable Bit 6: RSVD Bit [5:0]: Data type of error packet.
0x08DC	7:0	0x0F	**MIPI Error Packet Virtual Channel Override:** Bit 7: Override MIPI error packet VC for pipe 0. 0 = Disable (Will use VC on incoming video stream), 1 = Enable Bit [6:5]: RSVD Bit [4:0]: Virtual channel of error packet.
0x08DD	7:0	0x0F	**MIPI Error Packet Virtual Channel Override:** Bit 7: Override MIPI error packet VC for pipe 1. 0 = Disable (Will use VC on incoming video stream), 1 = Enable Bit [6:5]: RSVD Bit [4:0]: Virtual channel of error packet.
0x08DE	7:0	0x0F	**MIPI Error Packet Virtual Channel Override:** Bit 7: Override MIPI error packet VC for pipe 2. 0 = Disable (Will use VC on incoming video stream), 1 = Enable Bit [6:5]: RSVD Bit [4:0]: Virtual channel of error packet.
0x08E0	7:0	0x0F	**MIPI Error Packet Virtual Channel Override:** Bit 7: Override MIPI error packet VC for pipe 3. 0 = Disable (Will use VC on incoming video stream), 1 = Enable Bit [6:5]: RSVD Bit [4:0]: Virtual channel of error packet.
0x08E1	4:0	0x00	**MIPI Error Packet Virtual Channel Value:** Bit [7:5]: RSVD Bit [4:0]: Virtual channel of error packet detected on pipe 0.
0x08E2	4:0	0x00	**MIPI Error Packet Virtual Channel Value:** Bit [7:5]: RSVD Bit [4:0]: Virtual channel of error packet detected on pipe 1.
0x08E3	4:0	0x00	**MIPI Error Packet Virtual Channel Value:** Bit [7:5]: RSVD

			Bit [4:0]: Virtual channel of error packet detected on pipe 2.
0x08E4	4:0	0x00	**MIPI Error Packet Virtual Channel Value:** Bit [7:5]: RSVD Bit [4:0]: Virtual channel of error packet detected on pipe 3.
0x08E5	7:4	0x00	**MIPI Error Packet Word Count Override Enable:** Bit 7: Error Packet WC override EN pipe 3 Bit 6: Error Packet WC override EN pipe 2 Bit 5: Error Packet WC override EN pipe 1 Bit 4: Error Packet WC override EN pipe 0 Bit [3:0]: RSVD
0x08E6	7:0	0x00	**MIPI Error Packet Word Count Value High Bits Pipe 0:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.
0x08E7	7:0	0x00	**MIPI Error Packet Word Count Value Low Bits Pipe 0:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.
0x08E8	7:0	0x00	**MIPI Error Packet Word Count Value High Bits Pipe 1:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.
0x08E9	7:0	0x00	**MIPI Error Packet Word Count Value Low Bits Pipe 1:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.
0x08EA	7:0	0x00	**MIPI Error Packet Word Count Value High Bits Pipe 2:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.
0x08EB	7:0	0x00	**MIPI Error Packet Word Count Value Low Bits Pipe 2:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.
0x08EC	7:0	0x00	**MIPI Error Packet Word Count Value High Bits Pipe 3:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.
0x08ED	7:0	0x00	**MIPI Error Packet Word Count Value Low Bits Pipe 3:** Bit [7:0]: Sets or read the high bits of the word count value. If ERR_PKT_WC_OVRD_EN = 1, this sets the value. If ERR_PKT_WC_OVRD_EN = 0, this represents the value determined from incoming data.

MIPI Error Packet Programming Example

This example enables the MIPI Error Packet with data type set to RAW12 for all pipes and virtual channels are 0-3 for each pipe respectively.

```
# Enable the MIPI Error Packet for Pipe 0. Data type is set to 0x2C = Raw12.
0x4E,0x08D8,0xAC
# Enable the MIPI Error Packet for Pipe 1. Data type is set to 0x2C = Raw12.
0x4E,0x08D9,0xAC
# Enable the MIPI Error Packet for Pipe 2. Data type is set to 0x2C = Raw12.
0x4E,0x08DA,0xAC
# Enable the MIPI Error Packet for Pipe 3. Data type is set to 0x2C = Raw12.
0x4E,0x08DB,0xAC
# MIPI Error Packet virtual channel for pipe 0 is overridden to VC = 0.
0x4E,0x08DC,0x80
# MIPI Error Packet virtual channel for pipe 1 is overridden to VC = 1.
0x4E,0x08DD,0x81
# MIPI Error Packet virtual channel for pipe 2 is overridden to VC = 2.
0x4E,0x08DE,0x82
# MIPI Error Packet virtual channel for pipe 3 is overridden to VC = 3.
0x4E,0x08E0,0x83
```

ERRB Forwarding

The ERRB forwarding is a CSI packet that may be generated inside of the deserializer and sent with each video frame. This packet contains the device status and error information for any errors that are enabled at the time of the packet generation. The ERRB Packet allows for systems to quickly obtain critical SerDes operational information without having to poll multiple registers with I^2C.

The packet may be inserted on each MIPI PHY and may be inserted before or after a frame start or frame end short packet. To differentiate this packet from the video data, it may be sent as the EMB8, or user defined data types (0x12, or 0x31-0x37). The packet may adjust virtual channels, depending on the MIPI output selection of DPHY VC0-15 or CPHY VC0-31.

Additional front or back porch blanking time may be required to support insertion of this packet into a video frame. If inadequate time is allotted for packet generation, it will not be inserted properly into the frame. The required Error packet generation time can be calculated by the following:

$$Error\ Packet\ generation\ time = \frac{bpp}{8} * DE\ width$$

If the required time is met and the ERRB forwarding is enabled, the packet line length will automatically match that of the incoming video stream. For example, if the video line length is 1920 pixels, the inserted ERRB Packet will also be 1920 pixels long. The packet requires 600 data pixels and will add the remaining pixels with null data to match the incoming video data word count.

The packet will also automatically match the incoming virtual channel of the data unless the VC override is enabled. If multiple video streams with different bpp or VC are on the same pipe, the inserted ERRB Packet will automatically match the VC and line length of one of the streams. In complex cases such as this, it may be beneficial for users to specify the VC and word count manually through the ERRB Packet overrides.

ERRB Packet Decode

Table 34. ERRB Packet Decode Table

Data Byte	Register Address	Important Bits	Function
Data Byte 0	0x00		ERRB Status Address 0
Data Byte 1	0x1A		ERRB Status Address 1
Data Byte 2		bit 2	ERRB Status
Data Byte 3	0x00		Remote PHY A ERRB Status Address 0
Data Byte 4	0x30		Remote PHY A ERRB Status Address 1
Data Byte 5		bit 6	Remote PHY A ERRB Status
Data Byte 6	0x00		Remote PHY B ERRB Status Address 0
Data Byte 7	0x31		Remote PHY B ERRB Status Address 1
Data Byte 8		bit 6	Remote PHY B ERRB Status
Data Byte 9	0x00		Remote PHY C ERRB Status Address 0
Data Byte 10	0x32		Remote PHY C ERRB Status Address 1
Data Byte 11		bit 6	Remote PHY C ERRB Status
Data Byte 12	0x00		Remote PHY D ERRB Status Address 0
Data Byte 13	0x33		Remote PHY D ERRB Status Address 1
Data Byte 14		bit 6	Remote PHY D ERRB Status
Data Byte 15	0x00		LOCK Pin Status Address 0
Data Byte 16	0x1A		LOCK Pin Status Address 1
Data Byte 17		bit 0	LOCK Pin Status
Data Byte 18	0x00		PHY A LOCK Status Address 0
Data Byte 19	0x1A		PHY A LOCK Status Address 1
Data Byte 20		bit 3	PHY A LOCK Status
Data Byte 21	0x00		PHY B LOCK Status Address 0
Data Byte 22	0x0A		PHY B LOCK Status Address 1
Data Byte 23		bit 3	PHY B LOCK Status
Data Byte 24	0x00		PHY C LOCK Status Address 0
Data Byte 25	0x0B		PHY C LOCK Status Address 1
Data Byte 26		bit 3	PHY C LOCK Status
Data Byte 27	0x00		PHY D LOCK Status Address 0
Data Byte 28	0x0C		PHY D LOCK Status Address 1
Data Byte 29		bit 3	PHY D LOCK Status
Data Byte 30	0x04		CSIPLL LOCK 0/1/2/3 Status Address 0
Data Byte 31	0x00		CSIPLL LOCK 0/1/2/3 Status Address 1
Data Byte 32		bits 7:4	CSIPLL LOCK 0/1/2/3 Status
Data Byte 33	0x0B		GMSL1 Link A Lock Status Address 0
Data Byte 34	0xCB		GMSL1 Link A Lock Status Address 1
Data Byte 35		bit 0	GMSL1 Link A Lock Status
Data Byte 36	0x0C		GMSL1 Link B Lock Status Address 0
Data Byte 37	0xCB		GMSL1 Link B Lock Status Address 1

Data Byte 38		bit 0	GMSL1 Link B Lock Status
Data Byte 39	0x0D		GMSL1 Link C Lock Status Address 0
Data Byte 40	0xCB		GMSL1 Link C Lock Status Address 1
Data Byte 41		bit 0	GMSL1 Link C Lock Status
Data Byte 42	0x0E		GMSL1 Link D Lock Status Address 0
Data Byte 43	0xCB		GMSL1 Link D Lock Status Address 1
Data Byte 44		bit 0	GMSL1 Link D Lock Status
Data Byte 45	0x08		MIPI Control Select Status Address 0
Data Byte 46	0xCA		MIPI Control Select Status Address 1
Data Byte 47		NA	MIPI Control Select Status
Data Byte 48	0x00		Video Pipe Select 0 Status Address 0
Data Byte 49	0xF0		Video Pipe Select 0 Status Address 1
Data Byte 50		NA	Video Pipe Select 0 Status
Data Byte 51	0x00		Video Pipe Select 1 Status Address 0
Data Byte 52	0xF0		Video Pipe Select 1 Status Address 1
Data Byte 53		NA	Video Pipe Select 1 Status
Data Byte 54	0x00		Video Pipe Select 2 Status Address 0
Data Byte 55	0xF1		Video Pipe Select 2 Status Address 1
Data Byte 56		NA	Video Pipe Select 2 Status
Data Byte 57	0x00		Video Pipe Select 3 Status Address 0
Data Byte 58	0xF1		Video Pipe Select 3 Status Address 1
Data Byte 59		NA	Video Pipe Select 3 Status
Data Byte 60	0x00		Video Pipe Enable Status Address 0
Data Byte 61	0xF4		Video Pipe Enable Status Address 1
Data Byte 62		NA	Video Pipe Enable Status
Data Byte 63	0x01		Video Lock Status Pipe 0 Address 0
Data Byte 64	0xDC		Video Lock Status Pipe 0 Address 1
Data Byte 65		NA	Video Lock Status Pipe 0
Data Byte 66	0x01		Video Lock Status Pipe 1 Address 0
Data Byte 67	0xFC		Video Lock Status Pipe 1 Address 1
Data Byte 68		NA	Video Lock Status Pipe 1
Data Byte 69	0x02		Video Lock Status Pipe 2 Address 0
Data Byte 70	0x1C		Video Lock Status Pipe 2 Address 1
Data Byte 71		NA	Video Lock Status Pipe 2
Data Byte 72	0x02		Video Lock Status Pipe 3 Address 0
Data Byte 73	0x3C		Video Lock Status Pipe 3 Address 1
Data Byte 74		NA	Video Lock Status Pipe 3
Data Byte 75	0x12		Video VS_CNT_0 Count Status Address 0
Data Byte 76	0x07		Video VS_CNT_0 Count Status Address 1
Data Byte 77		NA	Video VS_CNT_0 Count Status
Data Byte 78	0x12		Video HS_CNT_0_MSB Count Status Address 0

Data Byte 79	0x08		Video HS_CNT_0_MSB Count Status Address 1
Data Byte 80		NA	Video HS_CNT_0_MSB Count Status
Data Byte 81	0x12		Video HS_CNT_0_LSB Count Status Address 0
Data Byte 82	0x09		Video HS_CNT_0_LSB Count Status Address 1
Data Byte 83		NA	Video HS_CNT_0_LSB Count Status
Data Byte 84	0x12		Video DE_CNT_0_MSB Count Status Address 0
Data Byte 85	0x0A		Video DE_CNT_0_MSB Count Status Address 1
Data Byte 86		NA	Video DE_CNT_0_MSB Count Status
Data Byte 87	0x12		Video DE_CNT_0_LSB Count Status Address 0
Data Byte 88	0x0B		Video DE_CNT_0_LSB Count Status Address 1
Data Byte 89		NA	Video DE_CNT_0_LSB Count Status
Data Byte 90	0x12		Video VS_CNT_1 Count Status Address 0
Data Byte 91	0x17		Video VS_CNT_1 Count Status Address 1
Data Byte 92		NA	Video VS_CNT_1 Count Status
Data Byte 93	0x12		Video HS_CNT_1_MSB Count Status Address 0
Data Byte 94	0x18		Video HS_CNT_1_MSB Count Status Address 1
Data Byte 95		NA	Video HS_CNT_1_MSB Count Status
Data Byte 96	0x12		Video HS_CNT_1_LSB Count Status Address 0
Data Byte 97	0x19		Video HS_CNT_1_LSB Count Status Address 1
Data Byte 98		NA	Video HS_CNT_1_LSB Count Status
Data Byte 99	0x12		Video DE_CNT_1_MSB Count Status Address 0
Data Byte 100	0x1A		Video DE_CNT_1_MSB Count Status Address 1
Data Byte 101		NA	Video DE_CNT_1_MSB Count Status
Data Byte 102	0x12		Video DE_CNT_1_LSB Count Status Address 0
Data Byte 103	0x1B		Video DE_CNT_1_LSB Count Status Address 1
Data Byte 104		NA	Video DE_CNT_1_LSB Count Status
Data Byte 105	0x12		Video VS_CNT_2 Count Status Address 0
Data Byte 106	0x27		Video VS_CNT_2 Count Status Address 1
Data Byte 107		NA	Video VS_CNT_2 Count Status
Data Byte 108	0x12		Video HS_CNT_2_MSB Count Status Address 0
Data Byte 109	0x28		Video HS_CNT_2_MSB Count Status Address 1
Data Byte 110		NA	Video HS_CNT_2_MSB Count Status
Data Byte 111	0x12		Video HS_CNT_2_LSB Count Status Address 0
Data Byte 112	0x29		Video HS_CNT_2_LSB Count Status Address 1
Data Byte 113		NA	Video HS_CNT_2_LSB Count Status
Data Byte 114	0x12		Video DE_CNT_2_MSB Count Status Address 0
Data Byte 115	0x2A		Video DE_CNT_2_MSB Count Status Address 1
Data Byte 116		NA	Video DE_CNT_2_MSB Count Status
Data Byte 117	0x12		Video DE_CNT_2_LSB Count Status Address 0
Data Byte 118	0x2B		Video DE_CNT_2_LSB Count Status Address 1
Data Byte 119		NA	Video DE_CNT_2_LSB Count Status

Data Byte 120	0x12		Video VS_CNT_3 Count Status Address 0
Data Byte 121	0x37		Video VS_CNT_3 Count Status Address 1
Data Byte 122		NA	Video VS_CNT_3 Count Status
Data Byte 123	0x12		Video HS_CNT_3_MSB Count Status Address 0
Data Byte 124	0x38		Video HS_CNT_3_MSB Count Status Address 1
Data Byte 125		NA	Video HS_CNT_3_MSB Count Status
Data Byte 126	0x12		Video HS_CNT_3_LSB Count Status Address 0
Data Byte 127	0x39		Video HS_CNT_3_LSB Count Status Address 1
Data Byte 128		NA	Video HS_CNT_3_LSB Count Status
Data Byte 129	0x12		Video DE_CNT_3_MSB Count Status Address 0
Data Byte 130	0x3A		Video DE_CNT_3_MSB Count Status Address 1
Data Byte 131		NA	Video DE_CNT_3_MSB Count Status
Data Byte 132	0x12		Video DE_CNT_3_LSB Count Status Address 0
Data Byte 133	0x3B		Video DE_CNT_3_LSB Count Status Address 1
Data Byte 134		NA	Video DE_CNT_3_LSB Count Status
Data Byte 135	0x09		MIPI TX 0 Status Address 0
Data Byte 136	0x02		MIPI TX 0 Status Address 1
Data Byte 137		NA	MIPI TX 0 Status
Data Byte 138	0x09		MIPI TX 1 Status Address 0
Data Byte 139	0x42		MIPI TX 1 Status Address 1
Data Byte 140		NA	MIPI TX 1 Status
Data Byte 141	0x09		MIPI TX 2 Status Address 0
Data Byte 142	0x82		MIPI TX 2 Status Address 1
Data Byte 143		NA	MIPI TX 2 Status
Data Byte 144	0x09		MIPI TX 3 Status Address 0
Data Byte 145	0xC2		MIPI TX 3 Status Address 1
Data Byte 146		NA	MIPI TX 3 Status
Data Byte 147	0x00		Reserved 0 Status Address 0
Data Byte 148	0x00		Reserved 0 Status Address 1
Data Byte 149		NA	Reserved 0 Status
Data Byte 150	0x00		Reserved 1 Status Address 0
Data Byte 151	0x00		Reserved 1 Status Address 1
Data Byte 152		NA	Reserved 1 Status
Data Byte 153	0x00		Reserved 2 Status Address 0
Data Byte 154	0x00		Reserved 2 Status Address 1
Data Byte 155		NA	Reserved 2 Status
Data Byte 156	0x00		Reserved 3 Status Address 0
Data Byte 157	0x00		Reserved 3 Status Address 1
Data Byte 158		NA	Reserved 3 Status
Data Byte 159	0x00		Reserved 4 Status Address 0
Data Byte 160	0x00		Reserved 4 Status Address 1

Data Byte 161		NA	Reserved 4 Status
Data Byte 162	0x00		PHY_INT_DEC_ERR Status (INTR3) Address 0
Data Byte 163	0x26		PHY_INT_DEC_ERR Status (INTR3) Address 1
Data Byte 164		bits 7:4,3:0	PHY_INT_DEC_ERR Status
Data Byte 165	0x00		EOM_ERR/LFLT ERR Status (INTR5) Address 0
Data Byte 166	0x28		EOM_ERR/LFLT ERR Status (INTR5) Address 1
Data Byte 167		bits 7:4,3,2	EOM_ERR/LFLT ERR Status (INTR5)
Data Byte 168	0x00		GMSL1 ERR/G2 LCRC/G2 VPRBS ERR/G2 Remote Err/G2 Frame Sync Err Status (INTR7) Address 0
Data Byte 169	0x2A		GMSL1 ERR/G2 LCRC/G2 VPRBS ERR/G2 Remote Err/G2 Frame Sync Err Status (INTR7) Address 1
Data Byte 170		bits 7:4,3,2,1,0	GMSL1 ERR/G2 LCRC/G2 VPRBS ERR/G2 Remote Err/G2 Frame Sync Err Status (INTR7)
Data Byte 171	0x00		Packet Count/Idle Packet Err Status (INTR9) Address 0
Data Byte 172	0x2C		Packet Count/Idle Packet Err Status (INTR9) Address 1
Data Byte 173		bits 7:4,3:0	Packet Count/Idle Packet Err Status (INTR9)
Data Byte 174	0x00		Retransmission Flag/ Max Retransmission Err Status (INTR11) Address 0
Data Byte 175	0x2E		Retransmission Flag/ Max Retransmission Err Status (INTR11) Address 1
Data Byte 176		bits 7:4,3:0	Retransmission Flag/ Max Retransmission Err Status (INTR11)
Data Byte 177	0x00		Decode Error PHY A Status Address 0
Data Byte 178	0x35		Decode Error PHY A Status Address 1
Data Byte 179		bits 7:0	Decode Error PHY A Status
Data Byte 180	0x00		Decode Error PHY B Status Address 0
Data Byte 181	0x36		Decode Error PHY B Status Address 1
Data Byte 182		bits 7:0	Decode Error PHY B Status
Data Byte 183	0x00		Decode Error PHY C Status Address 0
Data Byte 184	0x37		Decode Error PHY C Status Address 1
Data Byte 185		bits 7:0	Decode Error PHY C Status
Data Byte 186	0x00		Decode Error PHY D Status Address 0
Data Byte 187	0x38		Decode Error PHY D Status Address 1
Data Byte 188		bits 7:0	Decode Error PHY D Status
Data Byte 189	0x00		Idle Error PHY A Status Address 0
Data Byte 190	0x39		Idle Error PHY A Status Address 1
Data Byte 191		bits 7:0	Idle Error PHY A Status
Data Byte 192	0x00		Idle Error PHY B Status Address 0
Data Byte 193	0x3A		Idle Error PHY B Status Address 1
Data Byte 194		bits 7:0	Idle Error PHY B Status
Data Byte 195	0x00		Idle Error PHY C Status Address 0
Data Byte 196	0x3B		Idle Error PHY C Status Address 1
Data Byte 197		bits 7:0	Idle Error PHY C Status
Data Byte 198	0x00		Idle Error PHY D Status Address 0

Data Byte 199	0x3C		Idle Error PHY D Status Address 1
Data Byte 200		bits 7:0	Idle Error PHY D Status
Data Byte 201	0x00		Packet Count Error PHY A Status Address 0
Data Byte 202	0x40		Packet Count Error PHY A Status Address 1
Data Byte 203		NA	Packet Count Error PHY A Status
Data Byte 204	0x00		Packet Count Error PHY B Status Address 0
Data Byte 205	0x41		Packet Count Error PHY B Status Address 1
Data Byte 206		NA	Packet Count Error PHY B Status
Data Byte 207	0x00		Packet Count Error PHY C Status Address 0
Data Byte 208	0x42		Packet Count Error PHY C Status Address 1
Data Byte 209		NA	Packet Count Error PHY C Status
Data Byte 210	0x00		Packet Count Error PHY D Status Address 0
Data Byte 211	0x43		Packet Count Error PHY D Status Address 1
Data Byte 212		NA	Packet Count Error PHY D Status
Data Byte 213	0x00		Retention CRC Error Status Address 0
Data Byte 214	0x28		Retention CRC Error Status Address 1
Data Byte 215		bit 3	Retention CRC Error Status
Data Byte 216	0x00		EFUSE CRC Error Address 0
Data Byte 217	0x4E		EFUSE CRC Error Address 1
Data Byte 218		bit 4	EFUSE CRC Error Status
Data Byte 219	0x01		Video Line CRC Error PHY A Status Address 0 - only 1 flag to ERRB below OR'd together
Data Byte 220	0x00		Video Line CRC Error PHY A Status Address 1
Data Byte 221		bit 7	Video Line CRC Error PHY A Status
Data Byte 222	0x01		Video Line CRC Error PHY B Status Address 0
Data Byte 223	0x12		Video Line CRC Error PHY B Status Address 1
Data Byte 224		bit 7	Video Line CRC Error PHY B Status
Data Byte 225	0x01		Video Line CRC Error PHY C Status Address 0
Data Byte 226	0x24		Video Line CRC Error PHY C Status Address 1
Data Byte 227		bit 7	Video Line CRC Error PHY C Status
Data Byte 228	0x01		Video Line CRC Error PHY D Status Address 0
Data Byte 229	0x36		Video Line CRC Error PHY D Status Address 1
Data Byte 230		bit 7	Video Line CRC Error PHY D Status
Data Byte 231	0x01		Video Pipe 0 Sequence Error Detected Address 0
Data Byte 232	0x08		Video Pipe 0 Sequence Error Detected Address 1
Data Byte 233		bit 4	Video Pipe 0 Sequence Error Detected
Data Byte 234	0x01		Video Pipe 1 Sequence Error Detected Address 0
Data Byte 235	0x1A		Video Pipe 1 Sequence Error Detected Address 1
Data Byte 236		bit 4	Video Pipe 1 Sequence Error Detected
Data Byte 237	0x01		Video Pipe 2 Sequence Error Detected Address 0
Data Byte 238	0x2C		Video Pipe 2 Sequence Error Detected Address 1
Data Byte 239		bit 4	Video Pipe 2 Sequence Error Detected

Data Byte 240	0x01		Video Pipe 3 Sequence Error Detected Address 0
Data Byte 241	0x3E		Video Pipe 3 Sequence Error Detected Address 1
Data Byte 242		bit 4	Video Pipe 3 Sequence Error Detected
Data Byte 243	0x00		Video Pixel CRC Error and Video Masked Flag for Links A/B/C/D Address 0
Data Byte 244	0x45		Video Pixel CRC Error and Video Masked Flag for Links A/B/C/D Address 1
Data Byte 245		bits 7:4,3:0	Video Pixel CRC Error and Video Masked Flag for Links A/B/C/D
Data Byte 246	0x11		Video Pixel CRC Error PHY AX Status Address 0
Data Byte 247	0xD0		Video Pixel CRC Error PHY AX Status Address 1
Data Byte 248		NA	Video Pixel CRC Error PHY AX Status
Data Byte 249	0x11		Video Pixel CRC Error PHY AY Status Address 0
Data Byte 250	0xD1		Video Pixel CRC Error PHY AY Status Address 1
Data Byte 251		NA	Video Pixel CRC Error PHY AY Status
Data Byte 252	0x11		Video Pixel CRC Error PHY AZ Status Address 0
Data Byte 253	0xD2		Video Pixel CRC Error PHY AZ Status Address 1
Data Byte 254		NA	Video Pixel CRC Error PHY AZ Status
Data Byte 255	0x11		Video Pixel CRC Error PHY AU Status Address 0
Data Byte 256	0xE0		Video Pixel CRC Error PHY AU Status Address 1
Data Byte 257		NA	Video Pixel CRC Error PHY AU Status
Data Byte 258	0x11		Video Pixel CRC Error PHY BX Status Address 0
Data Byte 259	0xE1		Video Pixel CRC Error PHY BX Status Address 1
Data Byte 260		NA	Video Pixel CRC Error PHY BX Status
Data Byte 261	0x11		Video Pixel CRC Error PHY BY Status Address 0
Data Byte 262	0xE2		Video Pixel CRC Error PHY BY Status Address 1
Data Byte 263		NA	Video Pixel CRC Error PHY BY Status
Data Byte 264	0x11		Video Pixel CRC Error PHY BZ Status Address 0
Data Byte 265	0xE3		Video Pixel CRC Error PHY BZ Status Address 1
Data Byte 266		NA	Video Pixel CRC Error PHY BZ Status
Data Byte 267	0x11		Video Pixel CRC Error PHY BU Status Address 0
Data Byte 268	0xE4		Video Pixel CRC Error PHY BU Status Address 1
Data Byte 269		NA	Video Pixel CRC Error PHY BU Status
Data Byte 270	0x11		Video Pixel CRC Error PHY CX Status Address 0
Data Byte 271	0xE5		Video Pixel CRC Error PHY CX Status Address 1
Data Byte 272		NA	Video Pixel CRC Error PHY CX Status
Data Byte 273	0x11		Video Pixel CRC Error PHY CY Status Address 0
Data Byte 274	0xE6		Video Pixel CRC Error PHY CY Status Address 1
Data Byte 275		NA	Video Pixel CRC Error PHY CY Status
Data Byte 276	0x11		Video Pixel CRC Error PHY CZ Status Address 0
Data Byte 277	0xE7		Video Pixel CRC Error PHY CZ Status Address 1
Data Byte 278		NA	Video Pixel CRC Error PHY CZ Status

Data Byte 279	0x11		Video Pixel CRC Error PHY CU Status Address 0
Data Byte 280	0xE8		Video Pixel CRC Error PHY CU Status Address 1
Data Byte 281		NA	Video Pixel CRC Error PHY CU Status
Data Byte 282	0x11		Video Pixel CRC Error PHY DX Status Address 0
Data Byte 283	0xE9		Video Pixel CRC Error PHY DX Status Address 1
Data Byte 284		NA	Video Pixel CRC Error PHY DX Status
Data Byte 285	0x11		Video Pixel CRC Error PHY DY Status Address 0
Data Byte 286	0xEA		Video Pixel CRC Error PHY DY Status Address 1
Data Byte 287		NA	Video Pixel CRC Error PHY DY Status
Data Byte 288	0x11		Video Pixel CRC Error PHY DZ Status Address 0
Data Byte 289	0xEB		Video Pixel CRC Error PHY DZ Status Address 1
Data Byte 290		NA	Video Pixel CRC Error PHY DZ Status
Data Byte 291	0x11		Video Pixel CRC Error PHY DU Status Address 0
Data Byte 292	0xEC		Video Pixel CRC Error PHY DU Status Address 1
Data Byte 293		NA	Video Pixel CRC Error PHY DU Status
Data Byte 294	0x01		Video PRBS Error PHY 0 Status Address 0
Data Byte 295	0xDB		Video PRBS Error PHY 0 Status Address 1
Data Byte 296		NA	Video PRBS Error PHY 0 Status
Data Byte 297	0x01		Video PRBS Error PHY 1 Status Address 0
Data Byte 298	0xFB		Video PRBS Error PHY 1 Status Address 1
Data Byte 299		NA	Video PRBS Error PHY 1 Status
Data Byte 300	0x02		Video PRBS Error PHY 2 Status Address 0
Data Byte 301	0x1B		Video PRBS Error PHY 2 Status Address 1
Data Byte 302		NA	Video PRBS Error PHY 2 Status
Data Byte 303	0x02		Video PRBS Error PHY 3 Status Address 0
Data Byte 304	0x3B		Video PRBS Error PHY 3 Status Address 1
Data Byte 305		NA	Video PRBS Error PHY 3 Status
Data Byte 306	0x00		Link A GPIO Retransmission Count Address 0
Data Byte 307	0xA7		Link A GPIO Retransmission Count Address 1
Data Byte 308		bits 6:0	Link A GPIO Retransmission Count
Data Byte 309	0x00		Link B GPIO Retransmission Count Address 0
Data Byte 310	0xAF		Link B GPIO Retransmission Count Address 1
Data Byte 311		bits 6:0	Link B GPIO Retransmission Count
Data Byte 312	0x00		Link C GPIO Retransmission Count Address 0
Data Byte 313	0xB7		Link C GPIO Retransmission Count Address 1
Data Byte 314		bits 6:0	Link C GPIO Retransmission Count
Data Byte 315	0x00		Link D GPIO Retransmission Count Address 0
Data Byte 316	0xBF		Link D GPIO Retransmission Count Address 1
Data Byte 317		bits 6:0	Link D GPIO Retransmission Count
Data Byte 318	0x00		Link A GPIO Max Retransmission Error Address 0
Data Byte 319	0xA7		Link A GPIO Max Retransmission Error Address 1

Data Byte 320		bit 7	Link A GPIO Max Retransmission Error
Data Byte 321	0x00		Link B GPIO Max Retransmission Error Address 0
Data Byte 322	0xAF		Link B GPIO Max Retransmission Error Address 1
Data Byte 323		bit 7	Link B GPIO Max Retransmission Error
Data Byte 324	0x00		Link C GPIO Max Retransmission Error Address 0
Data Byte 325	0xB7		Link C GPIO Max Retransmission Error Address 1
Data Byte 326		bit 7	Link C GPIO Max Retransmission Error
Data Byte 327	0x00		Link D GPIO Max Retransmission Error Address 0
Data Byte 328	0xBF		Link D GPIO Max Retransmission Error Address 1
Data Byte 329		bit 7	Link D GPIO Max Retransmission Error
Data Byte 330	0x00		Line Fault 0 Error Status Address 0
Data Byte 331	0xE1		Line Fault 0 Error Status Address 1
Data Byte 332		bits 2:0	Line Fault 0 Error Status
Data Byte 333	0x00		Line Fault 1 Error Status Address 0
Data Byte 334	0xE1		Line Fault 1 Error Status Address 1
Data Byte 335		bits 6:4	Line Fault 1 Error Status
Data Byte 336	0x00		Line Fault 2 Error Status Address 0
Data Byte 337	0xE1		Line Fault 2 Error Status Address 1
Data Byte 338		bits 2:0	Line Fault 2 Error Status
Data Byte 339	0x00		Line Fault 3 Error Status Address 0
Data Byte 340	0xE1		Line Fault 3 Error Status Address 1
Data Byte 341		bits 6:4	Line Fault 3 Error Status
Data Byte 342	0x00		Line Fault Interrupt ERRB Flags (Sticky) Address 0
Data Byte 343	0xE5		Line Fault Interrupt ERRB Flags (Sticky) Address 1
Data Byte 344		bits 3:0	Line Fault Interrupt ERRB Flags (Sticky)
Data Byte 345	0x04		CMD FIFO / Line Memory Overflow Error Status Address 0
Data Byte 346	0x0A		CMD FIFO / Line Memory Overflow Error Status Address 1
Data Byte 347		bits 7:4,3:0	CMD FIFO / Line Memory Overflow Error Status
Data Byte 348	0x04		Pipe 1(0) Tunneling Header Flags Address 0
Data Byte 349	0x42		Pipe 1(0) Tunneling Header Flags Address 1
Data Byte 350		bits 6:0	Pipe 1(0) Tunneling Header Flags
Data Byte 351	0x04		Pipe 2(1) Tunneling Header Flags Address 0
Data Byte 352	0x43		Pipe 2(1) Tunneling Header Flags Address 1
Data Byte 353		bits 6:0	Pipe 2(1) Tunneling Header Flags
Data Byte 354	0x04		Pipe 3(2) Tunneling Header Flags Address 0
Data Byte 355	0x44		Pipe 3(2) Tunneling Header Flags Address 1
Data Byte 356		bits 6:0	Pipe 3(2) Tunneling Header Flags
Data Byte 357	0x04		Pipe 4(3) Tunneling Header Flags Address 0
Data Byte 358	0x45		Pipe 4(3) Tunneling Header Flags Address 1
Data Byte 359		bits 6:0	Pipe 4(3) Tunneling Header Flags

Data Byte 360	0x04		Frame Sync Error Count Address 0
Data Byte 361	0xB0		Frame Sync Error Count Address 1
Data Byte 362		bits 7:0	Frame Sync Error Count
Data Byte 363	0x05		I2C/UART Port 0 Retry Count Error PHY A Status Address 0
Data Byte 364	0x07		I2C/UART Port 0 Retry Count Error PHY A Status Address 1
Data Byte 365		bits 6:0	I2C/UART Port 0 Retry Count Error PHY A Status
Data Byte 366	0x05		I2C/UART Port 0 Retry Count Error PHY B Status Address 0
Data Byte 367	0x17		I2C/UART Port 0 Retry Count Error PHY B Status Address 1
Data Byte 368		bits 6:0	I2C/UART Port 0 Retry Count Error PHY B Status
Data Byte 369	0x05		I2C/UART Port 0 Retry Count Error PHY C Status Address 0
Data Byte 370	0x27		I2C/UART Port 0 Retry Count Error PHY C Status Address 1
Data Byte 371		bits 6:0	I2C/UART Port 0 Retry Count Error PHY C Status
Data Byte 372	0x05		I2C/UART Port 0 Retry Count Error PHY D Status Address 0
Data Byte 373	0x37		I2C/UART Port 0 Retry Count Error PHY D Status Address 1
Data Byte 374		bits 6:0	I2C/UART Port 0 Retry Count Error PHY D Status
Data Byte 375	0x05		I2C/UART Port 1 Retry Count Error PHY A Status Address 0
Data Byte 376	0x67		I2C/UART Port 1 Retry Count Error PHY A Status Address 1
Data Byte 377		bits 6:0	I2C/UART Port 1 Retry Count Error PHY A Status
Data Byte 378	0x05		I2C/UART Port 1 Retry Count Error PHY B Status Address 0
Data Byte 379	0x77		I2C/UART Port 1 Retry Count Error PHY B Status Address 1
Data Byte 380		bits 6:0	I2C/UART Port 1 Retry Count Error PHY B Status
Data Byte 381	0x05		I2C/UART Port 1 Retry Count Error PHY C Status Address 0
Data Byte 382	0x87		I2C/UART Port 1 Retry Count Error PHY C Status Address 1
Data Byte 383		bits 6:0	I2C/UART Port 1 Retry Count Error PHY C Status
Data Byte 384	0x05		I2C/UART Port 1 Retry Count Error PHY D Status Address 0
Data Byte 385	0x97		I2C/UART Port 1 Retry Count Error PHY D Status Address 1
Data Byte 386		bits 6:0	I2C/UART Port 1 Retry Count Error PHY D Status
Data Byte 387	0x05		I2C/UART Port 0 Max Retry Count Error PHY A Status Address 0
Data Byte 388	0x07		I2C/UART Port 0 Max Retry Count Error PHY A Status Address 1
Data Byte 389		bit 7	I2C/UART Port 0 Max Retry Count Error PHY A Status

Data Byte 390	0x05		I2C/UART Port 0 Max Retry Count Error PHY B Status Address 0
Data Byte 391	0x17		I2C/UART Port 0 Max Retry Count Error PHY B Status Address 1
Data Byte 392		bit 7	I2C/UART Port 0 Max Retry Count Error PHY B Status
Data Byte 393	0x05		I2C/UART Port 0 Max Retry Count Error PHY C Status Address 0
Data Byte 394	0x27		I2C/UART Port 0 Max Retry Count Error PHY C Status Address 1
Data Byte 395		bit 7	I2C/UART Port 0 Max Retry Count Error PHY C Status
Data Byte 396	0x05		I2C/UART Port 0 Max Retry Count Error PHY D Status Address 0
Data Byte 397	0x37		I2C/UART Port 0 Max Retry Count Error PHY D Status Address 1
Data Byte 398		bit 7	I2C/UART Port 0 Max Retry Count Error PHY D Status
Data Byte 399	0x05		I2C/UART Port 1 Max Retry Count Error PHY A Status Address 0
Data Byte 400	0x67		I2C/UART Port 1 Max Retry Count Error PHY A Status Address 1
Data Byte 401		bit 7	I2C/UART Port 1 Max Retry Count Error PHY A Status
Data Byte 402	0x05		I2C/UART Port 1 Max Retry Count Error PHY B Status Address 0
Data Byte 403	0x77		I2C/UART Port 1 Max Retry Count Error PHY B Status Address 1
Data Byte 404		bit 7	I2C/UART Port 1 Max Retry Count Error PHY B Status
Data Byte 405	0x05		I2C/UART Port 1 Max Retry Count Error PHY C Status Address 0
Data Byte 406	0x87		I2C/UART Port 1 Max Retry Count Error PHY C Status Address 1
Data Byte 407		bit 7	I2C/UART Port 1 Max Retry Count Error PHY C Status
Data Byte 408	0x05		I2C/UART Port 1 Max Retry Count Error PHY D Status Address 0
Data Byte 409	0x97		I2C/UART Port 1 Max Retry Count Error PHY D Status Address 1
Data Byte 410		bit 7	I2C/UART Port 1 Max Retry Count Error PHY D Status
Data Byte 411	0x08		MIPI PHY CP Error Address 0
Data Byte 412	0xAB		MIPI PHY CP Error Address 1
Data Byte 413		bits 7:4	MIPI PHY CP Error
Data Byte 414	0x08		Tunnel Mode Header ECC Error Correctable Error Controller 0 Status Address 0
Data Byte 415	0xB1		Tunnel Mode Header ECC Error Correctable Error Controller 0 Status Address 1
Data Byte 416		bits 6:3	Tunnel Mode Header ECC Error Correctable Error Controller 0 Status
Data Byte 417	0x09		Video Masked Error PHY A Status Address 0
Data Byte 418	0x34		Video Masked Error PHY A Status Address 1
Data Byte 419		bits 7:4	Video Masked Error PHY A Status
Data Byte 420	0x09		Video Masked Error PHY B Status Address 0

Data Byte 421	0x74		Video Masked Error PHY B Status Address 1
Data Byte 422		bits 7:4	Video Masked Error PHY B Status
Data Byte 423	0x09		Video Masked Error PHY C Status Address 0
Data Byte 424	0xB4		Video Masked Error PHY C Status Address 1
Data Byte 425		bits 7:4	Video Masked Error PHY C Status
Data Byte 426	0x09		Video Masked Error PHY D Status Address 0
Data Byte 427	0xF4		Video Masked Error PHY D Status Address 1
Data Byte 428		bits 7:4	Video Masked Error PHY D Status
Data Byte 429	0x0B		GMSL1 Link A Detect Error Address 0
Data Byte 430	0x15		GMSL1 Link A Detect Error Address 1
Data Byte 431		bits 7:0	GMSL1 Link A Detect Error
Data Byte 432	0x0B		GMSL1 Link A Max Retransmission Error Address 0
Data Byte 433	0x17		GMSL1 Link A Max Retransmission Error Address 1
Data Byte 434		bits 6,3	GMSL1 Link A Max Retransmission Error Flag
Data Byte 435	0x0B		GMSL1 Link A I2C Packet Retransmission Count Address 0
Data Byte 436	0x18		GMSL1 Link A I2C Packet Retransmission Count Address 1
Data Byte 437		bits 7:0	GMSL1 Link A I2C Packet Retransmission Count
Data Byte 438	0x0B		GMSL1 Link A CC CRC Error Count Address 0
Data Byte 439	0x19		GMSL1 Link A CC CRC Error Count Address 1
Data Byte 440		bits 7:0	GMSL1 Link A CC CRC Error Count
Data Byte 441	0x0B		GMSL1 Link A Line CRC Error Flag Address 0
Data Byte 442	0x1B		GMSL1 Link A Line CRC Error Flag Address 1
Data Byte 443		bit 2	GMSL1 Link A Line CRC Error Flag
Data Byte 444	0x0B		GMSL1 Link A ERRB output flags Address 0
Data Byte 445	0xD3		GMSL1 Link A ERRB output flags Address 1
Data Byte 446		bits 7:0	GMSL1 Link A ERRB output flags
Data Byte 447	0x0B		GMSL1 Link A EOM Eye Width Sticky Data Address 0
Data Byte 448	0xD4		GMSL1 Link A EOM Eye Width Sticky Data Address 1
Data Byte 449		bits 5:0	GMSL1 Link A EOM Eye Width Sticky Data
Data Byte 450	0x0C		GMSL1 Link B Detect Error Address 0
Data Byte 451	0x15		GMSL1 Link B Detect Error Address 1
Data Byte 452		bits 7:0	GMSL1 Link B Detect Error
Data Byte 453	0x0C		GMSL1 Link B Max Retransmission Error Address 0
Data Byte 454	0x17		GMSL1 Link B Max Retransmission Error Address 1
Data Byte 455		bits 6,3	GMSL1 Link B Max Retransmission Error Flag
Data Byte 456	0x0C		GMSL1 Link B I2C Packet Retransmission Count Address 0
Data Byte 457	0x18		GMSL1 Link B I2C Packet Retransmission Count Address 1
Data Byte 458		bits 7:0	GMSL1 Link B I2C Packet Retransmission Count
Data Byte 459	0x0C		GMSL1 Link B CC CRC Error Count Address 0

Data Byte 460	0x19		GMSL1 Link B CC CRC Error Count Address 1
Data Byte 461		bits 7:0	GMSL1 Link B CC CRC Error Count
Data Byte 462	0x0C		GMSL1 Link B Line CRC Error Flag Address 0
Data Byte 463	0x1B		GMSL1 Link B Line CRC Error Flag Address 1
Data Byte 464		bit 2	GMSL1 Link B Line CRC Error Flag
Data Byte 465	0x0C		GMSL1 Link B ERRB output flags Address 0
Data Byte 466	0xD3		GMSL1 Link B ERRB output flags Address 1
Data Byte 467		bits 7:0	GMSL1 Link B ERRB output flags
Data Byte 468	0x0C		GMSL1 Link B EOM Eye Width Sticky Data Address 0
Data Byte 469	0xD4		GMSL1 Link B EOM Eye Width Sticky Data Address 1
Data Byte 470		bits 5:0	GMSL1 Link B EOM Eye Width Sticky Data
Data Byte 471	0x0D		GMSL1 Link C Detect Error Address 0
Data Byte 472	0x15		GMSL1 Link C Detect Error Address 1
Data Byte 473		bits 7:0	GMSL1 Link C Detect Error
Data Byte 474	0x0D		GMSL1 Link C Max Retransmission Error Address 0
Data Byte 475	0x17		GMSL1 Link C Max Retransmission Error Address 1
Data Byte 476		bits 6,3	GMSL1 Link C Max Retransmission Error Flag
Data Byte 477	0x0D		GMSL1 Link C I2C Packet Retransmission Count Address 0
Data Byte 478	0x18		GMSL1 Link C I2C Packet Retransmission Count Address 1
Data Byte 479		bits 7:0	GMSL1 Link C I2C Packet Retransmission Count
Data Byte 480	0x0D		GMSL1 Link C CC CRC Error Count Address 0
Data Byte 481	0x19		GMSL1 Link C CC CRC Error Count Address 1
Data Byte 482		bits 7:0	GMSL1 Link C CC CRC Error Count
Data Byte 483	0x0D		GMSL1 Link C Line CRC Error Flag Address 0
Data Byte 484	0x1B		GMSL1 Link C Line CRC Error Flag Address 1
Data Byte 485		bit 2	GMSL1 Link C Line CRC Error Flag
Data Byte 486	0x0D		GMSL1 Link C ERRB output flags Address 0
Data Byte 487	0xD3		GMSL1 Link C ERRB output flags Address 1
Data Byte 488		bits 7:0	GMSL1 Link C ERRB output flags
Data Byte 489	0x0D		GMSL1 Link C EOM Eye Width Sticky Data Address 0
Data Byte 490	0xD4		GMSL1 Link C EOM Eye Width Sticky Data Address 1
Data Byte 491		bits 5:0	GMSL1 Link C EOM Eye Width Sticky Data
Data Byte 492	0x0E		GMSL1 Link D Detect Error Address 0
Data Byte 493	0x15		GMSL1 Link D Detect Error Address 1
Data Byte 494		bits 7:0	GMSL1 Link D Detect Error
Data Byte 495	0x0E		GMSL1 Link D Max Retransmission Error Address 0
Data Byte 496	0x17		GMSL1 Link D Max Retransmission Error Address 1
Data Byte 497		bits 6,3	GMSL1 Link D Max Retransmission Error Flag
Data Byte 498	0x0E		GMSL1 Link D I2C Packet Retransmission Count Address 0

Data Byte 499	0x18		GMSL1 Link D I2C Packet Retransmission Count Address 1
Data Byte 500		bits 7:0	GMSL1 Link D I2C Packet Retransmission Count
Data Byte 501	0x0E		GMSL1 Link D CC CRC Error Count Address 0
Data Byte 502	0x19		GMSL1 Link D CC CRC Error Count Address 1
Data Byte 503		bits 7:0	GMSL1 Link D CC CRC Error Count
Data Byte 504	0x0E		GMSL1 Link D Line CRC Error Flag Address 0
Data Byte 505	0x1B		GMSL1 Link D Line CRC Error Flag Address 1
Data Byte 506		bit 2	GMSL1 Link D Line CRC Error Flag
Data Byte 507	0x0E		GMSL1 Link D ERRB output flags Address 0
Data Byte 508	0xD3		GMSL1 Link D ERRB output flags Address 1
Data Byte 509		bits 7:0	GMSL1 Link D ERRB output flags
Data Byte 510	0x0E		GMSL1 Link D EOM Eye Width Sticky Data Address 0
Data Byte 511	0xD4		GMSL1 Link D EOM Eye Width Sticky Data Address 1
Data Byte 512		bits 5:0	GMSL1 Link D EOM Eye Width Sticky Data
Data Byte 513	0x14		Eye Opening Monitor Error PHY A Status Address 0
Data Byte 514	0x07		Eye Opening Monitor Error PHY A Status Address 1
Data Byte 515		bits 7:0	Eye Opening Monitor Error PHY A Status
Data Byte 516	0x15		Eye Opening Monitor Error PHY B Status Address 0
Data Byte 517	0x07		Eye Opening Monitor Error PHY B Status Address 1
Data Byte 518		bits 7:0	Eye Opening Monitor Error PHY B Status
Data Byte 519	0x16		Eye Opening Monitor Error PHY C Status Address 0
Data Byte 520	0x07		Eye Opening Monitor Error PHY C Status Address 1
Data Byte 521		bits 7:0	Eye Opening Monitor Error PHY C Status
Data Byte 522	0x17		Eye Opening Monitor Error PHY D Status Address 0
Data Byte 523	0x07		Eye Opening Monitor Error PHY D Status Address 1
Data Byte 524		bits 7:0	Eye Opening Monitor Error PHY D Status
Data Byte 525	0x12		BackTop Tunnel 1 Error Flags Address 0
Data Byte 526	0x64		BackTop Tunnel 1 Error Flags Address 1
Data Byte 527		NA	BackTop Tunnel 1 Error Flags
Data Byte 528	0x12		BackTop Tunnel 2 Error Flags Address 0
Data Byte 529	0x65		BackTop Tunnel 2 Error Flags Address 1
Data Byte 530		NA	BackTop Tunnel 2 Error Flags
Data Byte 531	0x12		BackTop Tunnel 3 Error Flags Address 0
Data Byte 532	0x66		BackTop Tunnel 3 Error Flags Address 1
Data Byte 533		NA	BackTop Tunnel 3 Error Flags
Data Byte 534	0x12		BackTop Tunnel 4 Error Flags Address 0
Data Byte 535	0x67		BackTop Tunnel 4 Error Flags Address 1
Data Byte 536		NA	BackTop Tunnel 4 Error Flags
Data Byte 537	0x00		VDD18/VDDIO/CAPVDD/VDD Supply Error Status Address 0

Data Byte 538	0x12		VDD18/VDDIO/CAPVDD/VDD Supply Error Status Address 1
Data Byte 539		NA	VDD18/VDDIO/CAPVDD/VDD Supply Error Status (CMP_STATUS)
Data Byte 540	0x00		VTERM Supply Error Status Address 0
Data Byte 541	0x12		VTERM Supply Error Status Address 1
Data Byte 542		NA	VTERM Supply Error Status (VDDBAD_STATUS)
Data Byte 543	0x00		VDD18/VDDIO POR Supply Error Status Address 0
Data Byte 544	0x13		VDD18/VDDIO POR Supply Error Status Address 1
Data Byte 545		NA	VDD18/VDDIO POR Supply Error Status (PORZ_STATUS)
Data Byte 546	0x00		VDD18/VDDIO/VTERM/VDD/CAPVDD Overvoltage Supply Error Interrupt Address 0
Data Byte 547	0x49		VDD18/VDDIO/VTERM/VDD/CAPVDD Overvoltage Supply Error Interrupt Address 1
Data Byte 548		bits 4:0	VDD18/VDDIO/VTERM/VDD/CAPVDD Overvoltage Supply Error Interrupt
Data Byte 549	0x00		VTERM/PORZ Supply Error Interrupt Address 0
Data Byte 550	0x49		VTERM/PORZ Supply Error Interrupt Address 1
Data Byte 551		bits 7:6	VTERM/PORZ Supply Error Interrupt Data
Data Byte 552	0x00		CMP_STATUS Combined Interrupt/ VTERM Status Error Address 0
Data Byte 553	0x4B		CMP_STATUS Combined Interrupt/ VTERM Status Error Address 1
Data Byte 554		bit 7	CMP_STATUS Combined Interrupt/ VTERM Status Error
Data Byte 555	0x11		I2C / UART CRC Value Address 0
Data Byte 556	0x02		I2C / UART CRC Value Address 1
Data Byte 557		bits 7:0	I2C / UART CRC Value
Data Byte 558	0x11		I2C / UART CRC Error Count Address 0
Data Byte 559	0x05		I2C / UART CRC Error Count Address 1
Data Byte 560		bits 7:0	I2C / UART CRC Error Count
Data Byte 561	0x11		I2C / UART CRC Error Flag Address 0
Data Byte 562	0x09		I2C / UART CRC Error Flag Address 1
Data Byte 563		bit 6	I2C / UART CRC Error Flag
Data Byte 564	0x11		I2C / UART Message Counter LSByte Address 0
Data Byte 565	0x03		I2C / UART Message Counter LSByte Address 1
Data Byte 566		bits 7:0	I2C / UART Message Counter LSByte
Data Byte 567	0x11		I2C / UART Message Counter MSByte Address 0
Data Byte 568	0x04		I2C / UART Message Counter MSByte Address 1
Data Byte 569		bits 7:0	I2C / UART Message Counter MSByte
Data Byte 570	0x11		I2C / UART Message Counter Error Count Address 0
Data Byte 571	0x06		I2C / UART Message Counter Error Count Address 1
Data Byte 572		bits 7:0	I2C / UART Message Counter Error Count
Data Byte 573	0x11		I2C / UART Message Counter Error Count Flag Address 0
Data Byte 574	0x09		I2C / UART Message Counter Error Count Flag Address 1

Data Byte 575		bit 7	I2C / UART Message Counter Error Count Flag
Data Byte 576	0x04		Backtop SRAM LCRC ERR Flags (4:1) Address 0
Data Byte 577	0x58		Backtop SRAM LCRC ERR Flags (4:1) Address 1
Data Byte 578		bits 3:0	Backtop SRAM LCRC ERR Flags (4:1)
Data Byte 579	0x04		CSIPLL Loss of Lock Sticky ERRB Address 0
Data Byte 580	0x55		CSIPLL Loss of Lock Sticky ERRB Address 1
Data Byte 581		bits 7:4	CSIPLL Loss of Lock Sticky ERRB
Data Byte 582	0x00		Reserved 5 Error Status Address 0
Data Byte 583	0x00		Reserved 5 Error Status Address 1
Data Byte 584		NA	Reserved 5 Error Status
Data Byte 585	0x00		Reserved 6 Error Status Address 0
Data Byte 586	0x00		Reserved 6 Error Status Address 1
Data Byte 587		NA	Reserved 6 Error Status
Data Byte 588	0x00		Reserved 7 Error Status Address 0
Data Byte 589	0x00		Reserved 7 Error Status Address 1
Data Byte 590		NA	Reserved 7 Error Status
Data Byte 591	0x00		Reserved 8 Error Status Address 0
Data Byte 592	0x00		Reserved 8 Error Status Address 1
Data Byte 593		NA	Reserved 8 Error Status
Data Byte 594	0x00		Reserved 9 Error Status Address 0
Data Byte 595	0x00		Reserved 9 Error Status Address 1
Data Byte 596		NA	Reserved 9 Error Status
Data Byte 597	0x00		ERRB Status Address 0
Data Byte 598	0x1A		ERRB Status Address 1
Data Byte 599		bit 2	ERRB Status

ERRB Packet Registers

Table 35. ERRB Packet Register Table

Register	Bits	Default Value	Description
0x0420	7:4	0x01	**ERRB Packet Enable:** Bit 7: ERRB Packet Enable for PHY 3. Bit 6: ERRB Packet Enable for PHY 2. Bit 5: ERRB Packet Enable for PHY 1. Bit 4: ERRB Packet Enable for PHY 0. 0 = Disable, 1 = Enable.
0x0421	7:0	0x55	**ERRB Packet Edge Select:** Bit [7:6]: Insert before or after short packet for pipe 3. Bit [5:4]: Insert before or after short packet for pipe 2. Bit [3:2]: Insert before or after short packet for pipe 1. Bit [1:0]: Insert before or after short packet for pipe 0. 00 = No insertion 01 = Insert before selected short packet 10 = Insert after selected short packet 11 = Reserved, do not use this value.
0x0422	6,4,2,0	0x00	**Short packet insertion Select:** Bit 6: Insert at frame end or frame start for pipe 3. Bit 4: Insert at frame end or frame start for pipe 2. Bit 2: Insert at frame end or frame start for pipe 1. Bit 0: Insert at frame end or frame start for pipe 0. 0 = ERRB_PKT comes out at Frame End. 1 = ERRB_PKT comes out at Frame Start.
0x0423	7:0	0x12	**Error Packet Data Type and Double mode:** Bit 7: ERRB_PKT_DBL_MODE_1. For pixel mode only. Double mode allows the ERRB packet to be packed more efficiently into memory. Must be used for data types with bpp > 16 and pixels per line count over 2700. Note: Double mode can only be used if the word count of the incoming video is an even number. 0 = Double mode disabled 1 = Double mode enabled. Bit 6: RSVD. Bit [5:0]: Error Packet Data Type. Value = 0x12 or 0x31-37.
0x0424	7:0	0x12	**Error Packet Data Type and Double mode:** Bit 7: ERRB_PKT_DBL_MODE_2. For pixel mode only. Double mode allows the ERRB packet to be packed more efficiently into memory. Must be used for data types with bpp > 16 and pixels per line count over 2700. Note: Double mode can only be used if the word count of the incoming video is an even number. 0 = Double mode disabled 1 = Double mode enabled. Bit 6: RSVD. Bit [5:0]: Error Packet Data Type. Value = 0x12 or 0x31-37.
0x0425	7:0	0x12	**Error Packet Data Type and Double mode:** Bit 7: ERRB_PKT_DBL_MODE_3.

			For pixel mode only. Double mode allows the ERRB packet to be packed more efficiently into memory. Must be used for data types with bpp > 16 and pixels per line count over 2700. Note: Double mode can only be used if the word count of the incoming video is an even number. 0 = Double mode disabled 1 = Double mode enabled. Bit 6: RSVD. Bit [5:0]: Error Packet Data Type. Value = 0x12 or 0x31-37.
0x0426	7:0	0x12	**Error Packet Data Type and Double mode:** Bit 7: ERRB_PKT_DBL_MODE_4. For pixel mode only. Double mode allows the ERRB packet to be packed more efficiently into memory. Must be used for data types with bpp > 16 and pixels per line count over 2700. Note: Double mode can only be used if the word count of the incoming video is an even number. 0 = Double mode disabled 1 = Double mode enabled. Bit 6: RSVD. Bit [5:0]: Error Packet Data Type. Value = 0x12 or 0x31-37.
0x0427	7:0	0x0F	**Error Packet Virtual Channel Override:** Bit 7: Error Packet Virtual Channel Override pipe 0. 0 = Disable, 1 = Enable. Bit [4:0]: Virtual channel number of ERRB packet. DPHY = 0-16 or CPHY = 0-31 (Decimal Values).
0x0428	7:0	0x0F	**Error Packet Virtual Channel Override:** Bit 7: Error Packet Virtual Channel Override pipe 1. 0 = Disable, 1 = Enable. Bit [4:0]: Virtual channel number of ERRB packet. DPHY = 0-16 or CPHY = 0-31 (Decimal Values).
0x0429	7:0	0x0F	**Error Packet Virtual Channel Override:** Bit 7: Error Packet Virtual Channel Override pipe 2. 0 = Disable, 1 = Enable. Bit [4:0]: Virtual channel number of ERRB packet. DPHY = 0-16 or CPHY = 0-31 (Decimal Values).
0x042A	7:0	0x0F	**Error Packet Virtual Channel Override:** Bit 7: Error Packet Virtual Channel Override pipe 3. 0 = Disable, 1 = Enable. Bit [4:0]: Virtual channel number of ERRB packet. DPHY = 0-16 or CPHY = 0-31 (Decimal Values).
0x042B	4:0	0x00	**Error Packet Used VC:** Bit [4:0]: Virtual channel detected on pipe 0 for ERRB packet.
0x042C	4:0	0x00	**Error Packet Used VC:** Bit [4:0]: Virtual channel detected on pipe 1 for ERRB packet.
0x042D	4:0	0x00	**Error Packet Used VC:** Bit [4:0]: Virtual channel detected on pipe 2 for ERRB packet.
0x042E	4:0	0x00	**Error Packet Used VC:** Bit [4:0]: Virtual channel detected on pipe 3 for ERRB packet.
0x0437	7:4	0x00	**Error Packet Word Count Override:** Bit 7: Word count override pipe 3.

			Bit 6: Word count override pipe 2. Bit 5: Word count override pipe 1. Bit 4: Word count override pipe 0. 0 = Disable, 1 = Enable.
0x0438	7:0	0x00	**Error Packet Word Count Value High Bits for Pipe 0:** Bits [7:0]: Error packet word count, high bits. If ERRB_PKT_WC_OVRD_EN_1 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.
0x0439	7:0	0x00	**Error Packet Word Count Value Low Bits for Pipe 0:** Bits [7:0]: Error packet word count, Low bits. If ERRB_PKT_WC_OVRD_EN_1 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.
0x043A	7:0	0x00	**Error Packet Word Count Value High Bits for Pipe 1:** Bits [7:0]: Error packet word count, high bits. If ERRB_PKT_WC_OVRD_EN_2 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.
0x043B	7:0	0x00	**Error Packet Word Count Value Low Bits for Pipe 1:** Bits [7:0]: Error packet word count, Low bits. If ERRB_PKT_WC_OVRD_EN_2 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.
0x043C	7:0	0x00	**Error Packet Word Count Value High Bits for Pipe 2:** Bits [7:0]: Error packet word count, high bits. If ERRB_PKT_WC_OVRD_EN_3 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.
0x043D	7:0	0x00	**Error Packet Word Count Value Low Bits for Pipe 2:** Bits [7:0]: Error packet word count, Low bits. If ERRB_PKT_WC_OVRD_EN_3 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.
0x043E	7:0	0x00	**Error Packet Word Count Value High Bits for Pipe 3:** Bits [7:0]: Error packet word count, high bits. If ERRB_PKT_WC_OVRD_EN_4 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.
0x043F	7:0	0x00	**Error Packet Word Count Value Low Bits for Pipe 3:** Bits [7:0]: Error packet word count, Low bits. If ERRB_PKT_WC_OVRD_EN_4 is set, this register sets the word count for ERRB packet. When not in WC override mode, this register reads back detected word count from incoming video used for ERRB packet.

ERRB Packet Programming Example

This example enables the ERRB Packet and inserts it before frame end short packets. Its data type is set to EMB8, and it will be virtual channel 7.

```
# Enable the Error Packet for Pipe 1. Only one Error packet is needed per PHY
output.
0x4E,0x0420,0x21
#Select Insertion of the Error packet before the short packet.
0x4E,0x0421,0x55
#Select Error packet insertion point at the Frame End packet.
0x4E,0x0422,0x00
#Set Error Packet Data Type to EMB8 0x12.
0x4E,0x0424,0x12
#Set Error Packet virtual channel to VC7.
0x4E,0x0428,0x87
```

Error Flags

The device has many error detection mechanisms and flags to verify when errors have occurred. Many of these errors may be sent to the ERRB pin if they occur, through the output enable (OEN) bit fields. Most error OEN's are enabled by default and are similarly named as the flag bits described in this section. If an error occurs, the errors must be cleared through reading the specific error flag or by writing to the master error reset *ERRB_MST_RST* bit in register *0x0005*. Table 36. Error Flags Table has a list of errors and the descriptions.

Table 36. Error Flags Table

Error Flag	Error Description
BACKTOP*1_TMD_CRC_1L_ERR_FLAG	1-lane CPHY mode CRC error detected. *May be replaced with the pipe 0-3.
BACKTOP*1_TMD_CRC_2L_ERR_FLAG	2-lane CPHY mode CRC error detected. *May be replaced with the pipe 0-3.
BACKTOP*1_TMD_ECC_ERR_FLAG	DPHY mode uncorrectable ECC error detected. *May be replaced with the pipe 0-3.
BACKTOP*1_TMD_ECC_FLAG	DPHY mode ECC error (1-bit correctable) detected. *May be replaced with the pipe 0-3.
BACKTOP*1_TMD_SP_DET_ERR	Detected header error in short packet detect state. Short packet header after detected three consecutive matching headers did not match in mode (DPHY/CPHY 1-lane/CPHY 2-lane).
CAPVDD_OV_FLAG	CAPVDD overvoltage indication.
cmd_overflow*0	Command FIFO overflow occurred. *May be replaced with the pipe 0-3.
CMP_STATUS_VDD_OV	VDD measured above overvoltage comparator threshold.
CMP_STATUS_VDD12_OV	VTERM measured above overvoltage comparator threshold.
CMP_STATUS_VDD18_OV	VDD18 measured above overvoltage comparator threshold.
CMP_STATUS_vddio_ov	VDDIO measured above overvoltage comparator threshold.
CSIPLL*3_LOL_STICKY_FLAG	CSIPLL Loss of lock ERRB Flag. *May be replaced with the PHY 0-3.

Csipll*3_PLLORangeH_flag	Indicates csipll*3 was above range at some point since last cleared. *May be replaced with the PHY 0-3.
Csipll*3_PLLORangeL_flag	Indicates csipll*3 was below range at some point since last cleared. *May be replaced with the PHY 0-3.
DE_CNT_*0_CMP_ERR_FLAG	DE count comparison error flag. Refer to DE_CNT_x_CMP register. *May be replaced with the pipe 0-3.
DEC_ERR_FLAG_*A	Errors have been detected in a GMSL data packet. This could indicate poor link quality. *May be replaced with the link A/B/C/D.
DESKEW_START_OVERLAP_FLAG_*3	Error flag that indicates that the MIPI controller tried to start outputting data while initial deskew was still running. Either speed up the MIPI output so the deskew finishes sooner or increase n_vs_block. *May be replaced with the PHY 0-3.
EFUSE_CRC_ERR	An issue with the device EFUSE has occurred, the device should no longer be used.
EFUSE_LDR_TIMED_OUT	EFUSE did not load properly. Device needs to be power cycled or reset.
EOM_ERR_FLAG_*A	Indicates that receiver eye-opening is below configured threshold. *May be replaced with the link A/B/C/D.
FSYNC_ERR_FLAG	An error in the frame synchronization has occurred.
G1_*A_ERR_FLAG	An error on the GMSL1 Link was detected. *May be replaced with the link A/B/C/D.
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 I2C Error flag.
G1_PRBS_ERR_FLAG	GMSL1 PRBS error flag.
HS_CNT_*0_CMP_ERR_FLAG	HS count comparison error flag. Refer to HS_CNT_x_CMP register. *May be replaced with the pipe 0-3.
I2C_UART_CRC_ERR_INT	CRC error occurred for I2C packet.
I2C_UART_MSGCNTR_ERR_INT	Error in the I2C message counter.
IDLE_ERR_FLAG_*A	Errors have been detected in a GMSL Idle packet. This could indicate poor link quality.
LCRC_ERR	A video line CRC error has been detected. This is error is for each video pipe.
LCRC_ERR_FLAG	A video line CRC error has been 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_*0	Line memory overflow occurred. *May be replaced with the pipe 0-3.
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. *May be replaced with the link A/B/C/D.
phy_cp_err[0]	PHY copy 0 FIFO overflow.
phy_cp_err[1]	PHY copy 0 FIFO underflow.
phy_cp_err[2]	PHY copy 1 FIFO overflow.
phy_cp_err[3]	PHY copy 1 FIFO underflow.
PHY_INT_*A	PHY interruption has occurred. *May be replaced with the PHY A/B/C/D.
PKT_CNT_FLAG_*A	Packet counter has reached packet count threshold. *May be replaced with the link A/B/C/D.
READ_FLAG	Packet GPIO RX Aggregation Output, Active Low.
REM_ERR_FLAG	An error on the serializer has been flagged. This is the status of the serializer ERRB level.
RT_CNT_FLAG_*A	One or more of the selected channels has done at least one ARQ retransmission. *May be replaced with the link A/B/C/D.
RTTN_CRC_INT	Retention Memory Restore CRC Error Interrupt. Asserted when an error is detected in the CRC calculation during a retention memory restore.
SRAM_LCRC_ERR_*0	SRAM Line CRC Error Detection. Compares CRC value of data written to the Video Line SRAM vs CRC value of data read out of the Video Line SRAM. *May be replaced with the pipe 0-3.
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 or CPHY header CRC error (correctable). Bit [4] Tunneling mode: DPHY ECC or CPHY header CRC error (uncorrectable). Bit [5] Tunneling mode: DPHY/CPHY data CRC error. Bit [6] Tunneling mode: DPHY to CPHY.
TUN_CONV_DATA_CRC_ERR	For tunneling mode, DPHY to CPHY conversion (header) data CRC errors
TUN_DATA_CRC_ERR	For tunneling mode, DPHY/CPHY data CRC errors
TUN_ECC_CORR_ERR	For tunneling mode, correctable errors on DPHY ECC or CPHY header CRC.
TUN_ECC_UNCORR_ERR	For tunneling mode, uncorrectable errors on DPHY ECC or CPHY header CRC
TUN_HDR_CRC_ERR_0_FLAG_*0	Tunneling mode CPHY first header CRC error flag. *May be replaced with the pipe 0-3.
TUN_HDR_CRC_ERR_1_FLAG_*0	Tunneling mode CPHY second header CRC error flag. *May be replaced with the pipe 0-3.
TUN_HDR_CRC_ERR_2_FLAG_*0	Tunneling mode CPHY third header CRC error flag. Obsolete when SER is in 1-lane CPHY mode. *May be replaced with the pipe 0-3.

TUN_HDR_CRC_ERR_3_FLAG_*0	Tunneling mode CPHY fourth header CRC error flag. Obsolete when SER is in 1-lane CPHY mode. *May be replaced with the pipe 0-3.
TUN_HDR_ECC_ERR_FLAG_*0	Tunneling mode DPHY header 2-bit (uncorrectable) ECC error flag. *May be replaced with the pipe 0-3.
TUN_HDR_ECC_FLAG_*0	Pipe 0 tunneling mode DPHY header 1-bit (correctable) ECC error flag. *May be replaced with the pipe 0-3.
TUN_HDR_ERR_FLAG_*0	Tunneling mode packet header error flag for pipe *0. *May be replaced with the pipe 0-3.
VDDBAD_INT_FLAG	Combined VDDBAD indicator, indicates an issue with VDD level.
VDDCMP_INT_FLAG	Combined VDD comparator output. See CMP_STATUS and VDDCMP_MASK registers.
VID_PXL_CRC_ERR_FLAG_*A	Video pixel CRC error detected. *May be replaced with the link A/B/C/D.
VID_SEQ_ERR	Video Rx sequence error detection.
VIDEO_MASKED_*0_FLAG	Flag will be set if any of Video Pipes 0-3 apart of 4WxH or Wx4H synchronous aggregation of Controller *0 lose video lock. * May be replaced with the pipe 0-3
VPRBS_ERR_FLAG	Video PRBS errors have been detected. This is for testing purposes only.
VS_CNT_*0_CMP_ERR_FLAG	VS count comparison error flag. Refer to VS_CNT_x_CMP register. *May be replaced with the pipe 0-3.

General-Purpose Input and Output (GPIO)

Overview

The devices have 9 multi-function pins (MFP) that can be used as general-purpose input and output (GPIO) pins or for other functionality (e.g., I2C, LOCK, ERRB, etc.). This section explains the GPIO function of MFP pins. Note that the I/O Matrix (MFP) is device specific. Refer to device datasheets for specific device support and MFP priority.

The GPIO blocks of GMSL2 devices 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 tunneled to any of the GPIO outputs on the opposite side of the link.

Operation

GPIO pin mapping is coordinated across the serial link via 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–GPIO0, GPIO1–GPIO1, GPIO2–GPIO2, etc. The GPIO mappings can be changed via registers.

GMSL2 devices use 5-bit pin IDs that can support mapping up to 32 GPIO pins. Note that the usable number of GPIOs is limited by the device 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.) GPIO registers *GPIO_B* and *GPIO_C*, the identification registers, are separated by GMSL links A-D, where *GPIO_A* registers are used for general GPIO setup.

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 will be ignored by Rx since Rx is already high. All subsequent transitions will be correctly observed.

GPIO Pull-Up and Pull-Down Resistor Setup

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

The resistor is configurated 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: 40kΩ
- 1: 1MΩ

GPIO Output Driver Setup

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

Configuring GPIO Forwarding

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

Configuring Input GPIO:

- Set *GPIO_TX_ID* with a value from 0–31 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 0–31 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. Transitions on a single GPIO input can also be mapped to multiple GPIO outputs (broadcasting). This is configured by setting multiple GPIOs to the same *GPIO_RX_ID*.

Toggling GPIO Manually with Registers

GPIO pins can be manually controlled via 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 and configure output driver by setting *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 37 GPIO Registers

Register	Bits	Default Value	Description
0x0300	7:0	0x81	**GPIO0 A:** Bit 7: RES_CFG 0 = 40KΩ, 1 = 1MΩ 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.
0x0301	7:0	0xA0	**Link A, GPIO0 B:** Bit [7:6]: Resistor configuration 00 = None 01 = Pull-up 10 = Pull-down Bit 5: OUT_TYPE 0 = Open-drain, 1 = Push-pull Bit [4:0]: GPIO_TX_ID, Address = 0-31
0x0302	6:0	0x00	**Link A, GPIO0 C:** Bit 6: GPIO_RECVED, Received GPIO Value 0 or 1. Bit [4:0]: GPIO_RX_ID, Address = 0-31
0x0303 - 0x031B	…	…	**Repeat Registers A, B, C for GPIO1-8**
0x0336	7:0	0x00	**Link B, GPIO0 A:** Bit [7:0]: RSVD
0x0337	5:0	0x00	**Link B, GPIO0 B:** Bit 5: GPIO_TX_EN_B 0 = Disable transmitting to link B. 1 = Enable transmitting to link B. Bit [4:0]: GPIO_TX_ID, Address = 0-31
0x0338	6:0	0x00	**Link B, GPIO0 C:** Bit 6: GPIO_RECVED_B, Received GPIO Value 0 or 1. Bit 5: GPIO_RX_EN_B 0 = Disable receiving from link B. 1 = Enable receiving from link B. Bit [4:0]: GPIO_RX_ID, Address = 0-31
0x0339 – 0x03C5	…	…	**Repeat Registers A, B, C for GPIO1-8, for each link B-D.**

GPIO Programming Example

This example enables deserializer GPIO0, 1, 2, and 3 forwarding from MAX96717 serializers GPIO0 inputs. GPIO0 comes from link A, GPIO1 comes from Link B, GPIO2 comes from link C, and GPIO3 comes from link D. The MAX96724 deserializer is connected to four serializers in this example, and all have the unique addresses 0x80, 0x82, 0x84, and 0x86.

Deserializer Link A GPIO Forwarding:

```
#Setup deserializer GPIO0 RX enable and enable output.
0x4E,0x0300,0x84
#Setup deserializer GPIO0 Output type to push-pull and pull-down register.
0x4E,0x0301,0xA0
#Setup deserializer GPIO0 receive ID GPIO_RX_ID to be 0.
0x4E,0x0302,0x00
```

Deserializer Link B GPIO Forwarding:

```
#Setup deserializer GPIO1 RX enable and enable output.
0x4E,0x0303,0x84
#Setup deserializer GPIO1 Output type to push-pull and pull-down register.
0x4E,0x0304,0xA0
#Setup deserializer GPIO1 receive ID GPIO_RX_ID to be 0 and GPIO_RX_EN.
0x4E,0x033B,0x20
```

Deserializer Link C GPIO Forwarding:

```
#Setup deserializer GPIO2 RX enable and enable output.
0x4E,0x0306,0x84
#Setup deserializer GPIO2 Output type to push-pull and pull-down register.
0x4E,0x0307,0xA0
#Setup deserializer GPIO2 receive ID GPIO_RX_ID to be 0 and GPIO_RX_EN.
0x4E,0x0375,0x20
```

Deserializer Link D GPIO Forwarding:

```
#Setup deserializer GPIO3 RX enable and enable output.
0x4E,0x0309,0x84
#Setup deserializer GPIO3 Output type to push-pull and pull-down register.
0x4E,0x030A,0xA0
#Setup deserializer GPIO3 receive ID GPIO_RX_ID to be 0 and GPIO_RX_EN.
0x4E,0x03AE,0x20
```

Serializer Link A GPIO Forwarding:

```
#Setup serializer GPIO0 TX enable.
0x80,0x02BE,0x83
#Setup serializer GPIO0 transmit ID GPIO_TX_ID to be 0.
0x80,0x02BF,0xA0
```

Serializer Link B GPIO Forwarding:

```
#Setup serializer GPIO0 TX enable.
0x82,0x02BE,0x83
#Setup serializer GPIO0 transmit ID GPIO_TX_ID to be 0.
0x82,0x02BF,0xA0
```

Serializer Link C GPIO Forwarding:

```
#Setup serializer GPIO0 TX enable.
0x84,0x02BE,0x83
#Setup serializer GPIO0 transmit ID GPIO_TX_ID to be 0.
```

```
0x84,0x02BF,0xA0
```

Serializer Link D GPIO Forwarding:

```
#Setup serializer GPIO0 TX enable.
0x86,0x02BE,0x83
#Setup serializer GPIO0 transmit ID GPIO_TX_ID to be 0.
0x86,0x02BF,0xA0
```

GPIO Aggregation

Some use cases may require users to logically aggregate multiple deserializer GPIO pins to single output. A limited number of deserializer GPIO pins may be selected to aggregate and output to ERRB pin and/or deserializer MFP6. The GPIO aggregation is a logical circuit with each input value representing the deserializer multifunction pin *GPIO_RECVED* bit fields. For the aggregation circuit to receive inputs, GPIO forwarding must be enabled from the selected serializer GPIO input pins to the selected deserializer MFP pins. The aggregation function will allow for up to 8 deserializer MFP pin values to logically connect. The GPIO aggregation circuit is represented in Figure 26 GPIO Aggregation Logic Circuit.

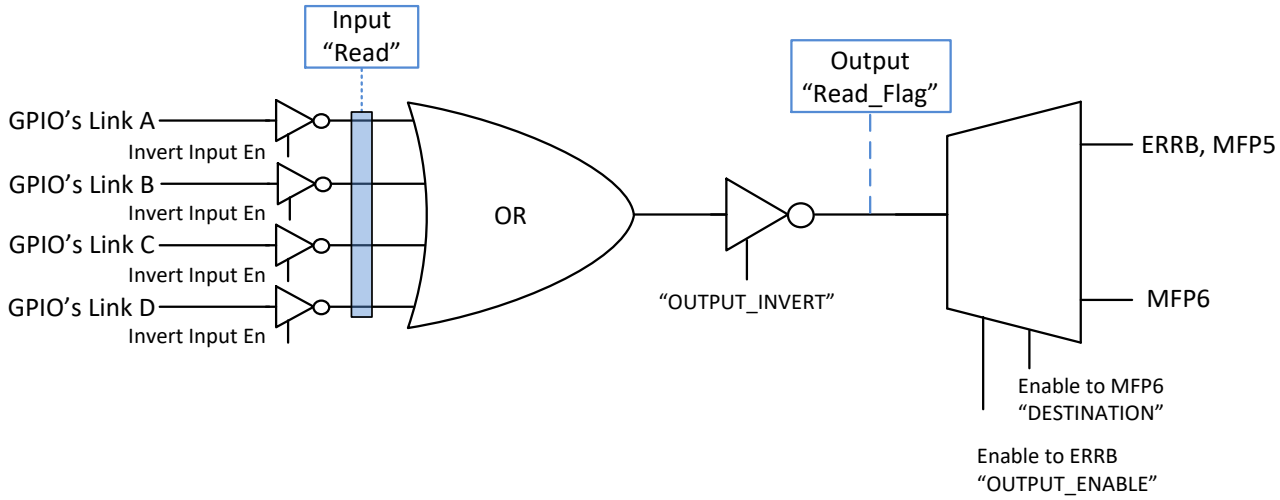

Figure 26 GPIO Aggregation Logic Circuit

The GPIO Aggregation circuit may take inputs from any GMSL link and can invert each input value. These input values are logically ORed together, where the final output may be inverted. The output can then be routed to the ERRB pin through the *OUTPUT_ENABLE* bit field. The output of the circuit may additionally be routed to the deserializer MFP6 through the *DESTINATION* bit field. Both outputs may be enabled if system designers wish to have redundant outputs.

GPIO Aggregation Registers

Table 38 GPIO Aggregation Registers

Register	Bits	Default Value	Description
0x02E0	7:0	0x00	**Polarity Inversion Enable GPIO's Link A:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = True State, default 1 = Invert State
0x02E1	7:0	0x00	**Polarity Inversion Enable GPIO's Link B:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = True State, default 1 = Invert State
0x02E2	7:0	0x00	**Polarity Inversion Enable GPIO's Link C:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = True State, default 1 = Invert State
0x02E3	7:0	0x00	**Polarity Inversion Enable GPIO's Link D:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = True State, default 1 = Invert State
0x02E4	7:0	0x00	**Polarity Inversion Enable High MFPs GPIO's Link A and B:** Bit [6:4]: Each bit represents GPIO[10:8] pins Link B. Bit [2:0]: Each bit represents GPIO[10:8] pins Link A. 0 = True State, default 1 = Invert State
0x02E5	7:0	0x00	**Polarity Inversion Enable High MFPs GPIO's Link C and D:** Bit [6:4]: Each bit represents GPIO[10:8] pins Link D. Bit [2:0]: Each bit represents GPIO[10:8] pins Link C. 0 = True State, default 1 = Invert State
0x02E6	7:0	0x00	**Enable Aggregation GPIO's Link A:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Disabled 1 = Enabled
0x02E7	7:0	0x00	**Enable Aggregation GPIO's Link B:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Disabled 1 = Enabled
0x02E8	7:0	0x00	**Enable Aggregation GPIO's Link C:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Disabled 1 = Enabled
0x02E9	7:0	0x00	**Enable Aggregation GPIO's Link D:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Disabled 1 = Enabled
0x02EA	7:0	0x00	**Enable Aggregation High MFPs GPIO's Link A and B:** Bit [6:4]: Each bit represents GPIO[10:8] pins Link B. Bit [2:0]: Each bit represents GPIO[10:8] pins Link A. 0 = Disabled 1 = Enabled
0x02EB	7:0	0x00	**Enable Aggregation High MFPs GPIO's Link C and D:**

			Bit [6:4]: Each bit represents GPIO[10:8] pins Link D. Bit [2:0]: Each bit represents GPIO[10:8] pins Link C. 0 = Disabled 1 = Enabled
0x02EC	7:0	0x00	**Read State GPIO's Link A:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Value is low 1 = Value is high
0x02ED	7:0	0x00	**Read State GPIO's Link B:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Value is low 1 = Value is high
0x02EE	7:0	0x00	**Read State GPIO's Link C:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Value is low 1 = Value is high
0x02EF	7:0	0x00	**Read State GPIO's Link D:** Bit [7:0]: Each bit represents GPIO[7: 0] pins 0 = Value is low 1 = Value is high
0x02F0	7:0	0x00	**Read State High MFPs GPIO's Link A and B:** Bit [6:4]: Each bit represents GPIO[10:8] pins Link B. Bit [2:0]: Each bit represents GPIO[10:8] pins Link A. 0 = Value is low 1 = Value is high
0x02F1	7:0	0x00	**Read State High MFPs GPIO's Link C and D:** Bit [6:4]: Each bit represents GPIO[10:8] pins Link D. Bit [2:0]: Each bit represents GPIO[10:8] pins Link C. 0 = Value is low 1 = Value is high
0x02F2	3:0	0x00	**GPIO Aggregation Output:** Bit 3: OUTPUT_INVERT, Aggregation output invert. 0 = True State, default 1 = Invert State Bit 2: OUTPUT_ENABLE, Output enable to ERRB pin. 0 = Disabled 1 = Enabled Bit 1: DESTINATION, Output enable to MFP6. 0 = Disabled 1 = Enabled Bit 0: READ_FLAG, Aggregation output. 0 = Output Low 1 = Output high

GPIO Aggregation Programming Example

This example enables GPIO0, 1, 2 aggregations to the ERRB pin. GPIO0 comes from link A, GPIO1 comes from Link B, and GPIO2 comes from link C. The MAX96724 deserializer is connected to three MAX96717 serializers in this example, and all have the same address 0x80.

```
# Enable the aggregation of link A GPIO0.
0x4E,0x02E6,0x01
# Enable the aggregation of link B GPIO1.
0x4E,0x02E7,0x02
# Enable the aggregation of link C GPIO2.
0x4E,0x02E8,0x04
# Enable OUTPUT_ENABLE bit for routing to ERRB pin.
0x4E,0x02F2,0x05

#Setup deserializer GPIO0 to receive value from the link.
0x4E,0x0300,0x84
#Setup deserializer GPIO0 output type to be push-pull with pulldown resistor.
0x4E,0x0301,0xA0
#Setup deserializer GPIO0 receive ID GPIO_RX_ID to be 0.
0x4E,0x0302,0x00

#Setup deserializer GPIO1 to receive value from the link.
0x4E,0x0303,0x84
#Setup deserializer GPIO1 to receive from Link B serializer,
#GPIO_RX_ID set to be 0.
0x4E,0x033B,0x20

#Setup deserializer GPIO2 to receive value from the link.
0x4E,0x0306,0x84
#Setup deserializer GPIO2 to receive from Link C serializer,
#GPIO_RX_ID set to be 0.
0x4E,0x0375,0x20

# MAX96717 Serializer GPIO Setup for Links A-C. GPIO0 is used on all serializers as
an input flag.
#Setup MAX96717 serializer GPIO0 to transmit input pin value to the link.
0x80,0x02BE,0x83
#Setup MAX96717 GPIO transmission ID GPIO_TX_ID to be 0.
0x80,0x02BF,0xA0
#Setup MAX96717 GPIO transmission ID GPIO_TX_ID to be 0.
```

Video PRBS Generator and Checker

Serial link systems incorporating the quad deserializer and GMSL2 serializers have several options for generating and checking video PRBS packets to verify that the data path between a serializer and the quad deserializer is error-free.
The method of conducting VPRBS testing is dependent on the system and serializer connected. For example, not all serializers can create the required PCLK signal to transmit a VPRBS signal. Serializers like the MAX96717 can create an internal PCLK signal, where MAX9295A serializers must receive an external PCLK signal from the system. A common system variation that causes a difference in the VPRBS testing method is whether image sensor data is active and transmitting though the serializer or not. An example of VPRBS testing for serializers that internally generate PCLK are included in this document as well as an example of conducting VPRBS testing as video is running.

It is not recommended to conduct video PRBS testing in some quad deserializer systems as external connections may be required to obtain the required serializer PCLK. The test can be useful for bringups and testing where the external connections can be made to the serializer. The MIPI input to the serializer must also be disabled prior to running this test.

Each of the four quad deserializer video pipes have independent control over packet generation and checking parameters and dedicated error count registers. Reference Table 39 for register information.

Video PRBS Registers

Table 39 MIPI Video PRBS Generator and Checker Registers

Register	Bits	Default Value	Description	Decode
0x01DC 0x01FC 0x021C 0x023C	7	0x1	Pattern generator clock source for video PRBS7, PRBS9, PRBS24, checkerboard, and gradient patterns.	0b0: 150MHz 0b1: 375MHz (default)
	4	0b0	Enables video PRBS24 generator/checker	0b0: Disabled 0b1: Enabled
	3	0b0	Enables video PRBS7 generator/checker	0b0: Disabled 0b1: Enabled
	2	0b0	Enables video PRBS9 generator/checker	0b0: Disabled 0b1: Enabled
0x01DB 0x01FB 0x021B 0x023B	7:0	0x00	Video PRBS error counter, clears on read	0xXX: Number of video PRBS errors since last read
0x0029	2	0b1	Enables reporting of video PRBS errors (VPRBS_ERR_FLAG—0x2A) at ERRB pin.	0b0: Disable video PRBS error reporting 0b1: Enable video PRBS error reporting
0x002A	2	0b0	Video PRBS Error Flag. Asserted when VPRBS_ERR (0x1DB) > 0.	0b0: VPRBS_ERR = 0 0b1: VPRBS_ERR > 0

Video PRBS Programming Examples

Serializer Generated VPRBS

The following script will configure a single MAX96717 and a MAX96724 to conduct the standard VPRBS test. In this test the serializer will generate a PRBS pattern and the deserializer will check it. This test requires that the serializer has PCLK and that no video is running into the serializer.

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

#Enable Deserializer link B only
0x4E,0x0006,0xF2
#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(2^24) checker for deserializer pipe 1
0x4E,0x01FC,0x10
#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
0x4E,0x01FC,0x11
#Verify deserializer does not have PRBS errors VPBRB_ERR = 0x00
0x4E,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 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.
0x4E,0x01FB,0xFF
```

Pairing with GMSL1 Serializers

Overview

Quad deserializers are GMSL1 backwards compatible and may be paired with GMSL1 serializers on any of the four GMSL links. 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 via the reverse channel.
- Program the deserializer following the steps described above in the Configuration section.
- Enable image sensor streaming and enable the forward channel.

When using GMSL1 mode, use only GMSL1 camera serializers that are equipped with high immunity mode (HIM). HIM provides robust control channel 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 quad 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 I2C master 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 quad deserializer. This is achieved by adding a pull-up resistor at the HIM pin of the serializer. Refer to the MAX96705A datasheet for additional details.

There are two methods to build the CLINK. Power up the GMSL2 CSI-2 quad deserializer with GMSL1 HIM mode enabled by setting the CFG pins or enable HIM mode with register write *HIGHHIMM* (bit 7 in register *0x0B06/0x0C06/0x0D06/0x0E06* for link A/B/C/D respectively).

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 below (Figure 27 and Table 40). Note that before HIM mode is established on both sides, it is recommended to individually enable and program each link to avoid potential I2C command race conditions. This can be done by only having one link enabled on the deserializer at a time.

Programming Examples

This example demonstrates how to set up the communication channel for a GMSL 1 link. Note that it is important to enable and program each link individually.

```
# Turn on HIM on MAX96724 Link A
0x4E,0x0B06,0xEF
# Turn on local I2C acknowledge as link is not established yet
0x4E,0x0B0D,0x80
# Enable CLINK in MAX96705A that is connected on Link A
0x80,0x0004,0x43
# Delay 5ms
# Turn off local I2C acknowledge
0x4E,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 Link A only
0x4E,0x0006,0x01
# Set deserializer link A: HIM Mode Enable
0x4E,0x0B06,0xEF
# Turn on local I2C acknowledge as link is not established yet
0x4E,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
0x4E,0x0B07,0x84
# Deserializer Disable Remote master
0x4E,0x0B05,0x79
# Deserializer Enable SHIFT_VID_HVD
0x4E,0x0BA7,0x45
# Disable I2C_LOC_ACK: This should be disabled once the forward channel is
established.
0x4E,0x0B0D,0x00
# Deserializer, Set datatype to YUV422. This can be changed depending on the data
type used.
0x4E,0x0B96,0x1B
# Deserializer, DE_EN = 0; This is dependent on the camera used.
0x4E,0x0B0F,0x01
```

Configuration Flow Chart and Detailed Steps

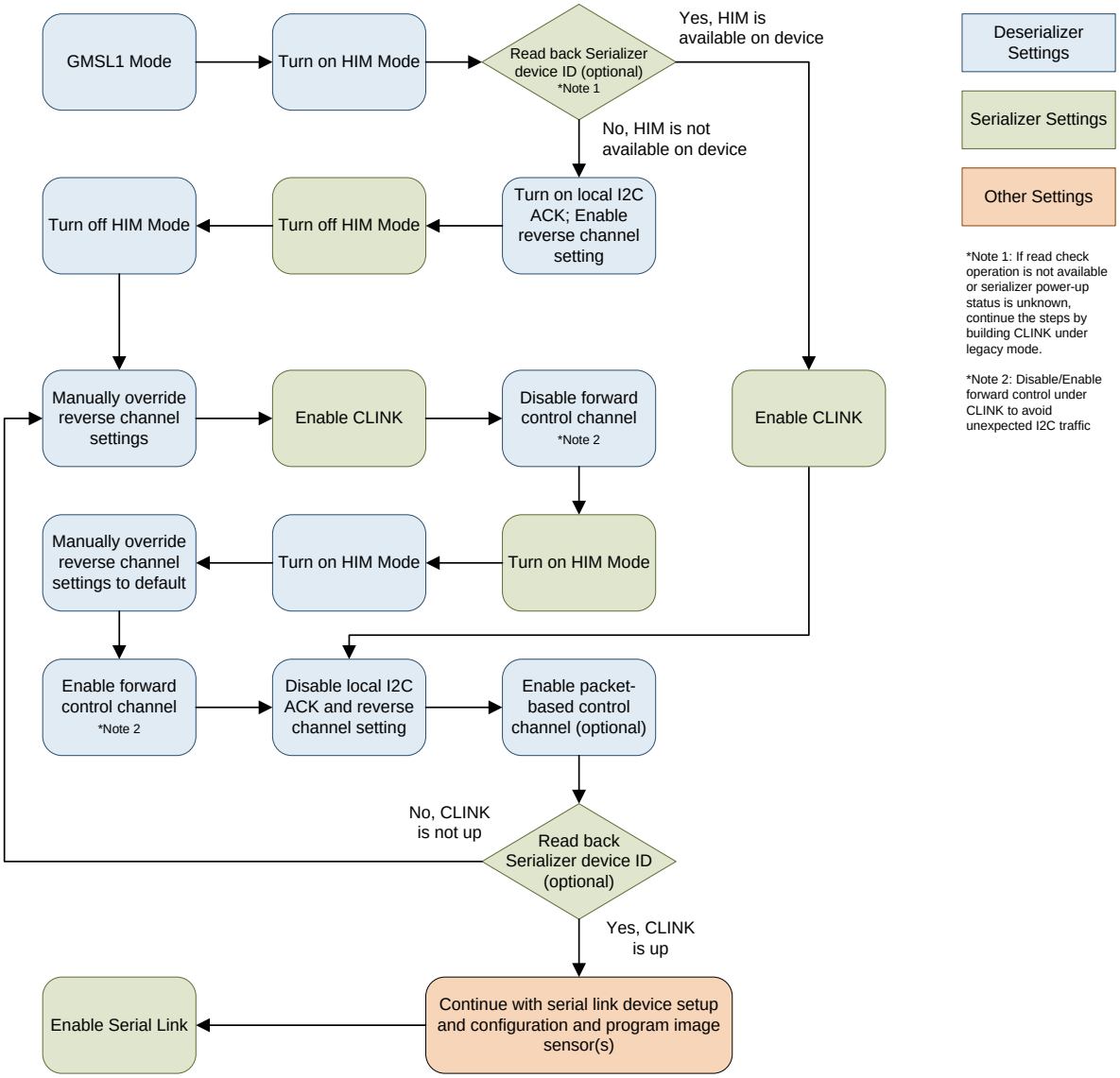

Figure 27 Building the CLINK on HIM-Capable GMSL1 Serializer

Table 40 Configuration Steps for Pairing HIM-Capable GMSL1 Serializers

Step	Device	Slave ID	Register(s)	Value	Description
1	Quad DES	0x4E	0x0006	0x01/ 0x02/ 0x04/ 0x08	Enable Link A/B/C/D in GMSL1 mode. Each link being programmed must be enabled.
Delay 5ms					
2	Quad DES	0x4E	0x0B06/ 0x0C06/ 0x0D06/	0xEF	Turn on HIM mode on Link A/B/C/D. *This step should be skipped if the serializer is not in HIM mode.

			0x0E06		
Note					**If Ser is powered up with HIM by default**, CLINK is established with one more write on Ser (0x04=0x43). User can proceed to step 17. Otherwise continue.
3	Quad DES	0x4E	0x0B0D/ 0x0C0D/ 0x0D0D/ 0x0E0D	0x81	Enable reverse channel config on Link A/B/C/D. Turn on local I2C acknowledge on Link A/B/C/D.
4	SER	0x80	0x004D	0x40	Turn off HIM on Ser.
Delay 10ms					
5	Quad DES	0x4E	0x0B06/ 0x0C06/ 0x0D06/ 0x0E06	0x6F	Turn off HIM mode on Link A/B/C/D.
6	Quad DES	0x4E	0x14C5/ 0x15C5/ 0x16C5/ 0x17C5	0xAA	Manually override reverse channel pulse length for Link A/B/C/D.
7	Quad DES	0x4E	0x14C4/ 0x15C4/ 0x16C4/ 0x17C4	0x80	Manually override reverse channel rise/fall time for Link A/B/C/D.
8	Quad DES	0x4E	0x1495/ 0x1595/ 0x1695/ 0x1795	0xC8	Manually override reverse channel Tx amplitude for Link A/B/C/D.
9	SER	0x80	0x0004	0x43	Enable control link only; Disable serial link.
Delay 5ms					Control link should be built up successfully.
10	Quad DES	0x4E	0x0B04/ 0x0C04/ 0x0D04/ 0x0E04	0x02	Disable forward control channel transmitter for Link A/B/C/D.
11	SER	0x80	0x004D	0xC0	Enable HIM mode on Ser (i.e., remote side).
Delay 5ms					
12	Quad DES	0x4E	0x0B06/ 0x0C06/ 0x0D06/ 0x0E06	0xEF	Turn on HIM mode on Link A/B/C/D.
Repeat above steps for all other links. **CLINK should be established on all links with HIM enabled.**					

13	Quad DES	0x4E	0x14C5/ 0x15C5/ 0x16C5/ 0x17C5	0x40	Manually override reverse channel pulse length to default value for Link A/B/C/D.
14	Quad DES	0x4E	0x14C4/ 0x15C4/ 0x16C4/ 0x17C4	0x40	Manually override reverse channel rise/fall time to default value for Link A/B/C/D.
15	Quad DES	0x4E	0x1495/ 0x1595/ 0x1695/ 0x1795	0x69	Manually override reverse channel Tx amplitude to default value for Link A/B/C/D.
16	Quad DES	0x4E	0x0B04/ 0x0C04/ 0x0D04/ 0x0E04	0x03	Enable forward control channel transmitter for Link A/B/C/D.
17	Quad DES	0x4E	0x0B0D/ 0x0C0D/ 0x0D0D/ 0x0E0D	0x00	Disable local I2C acknowledge. Disable reverse channel setting.
18 (optional)	Quad DES	0x4E	0x0B08	0x25	Enable packet-based control channel mode for Link A.
Delay 10ms					
			0x0C08	0x25	Enable packet-based control channel mode for Link B.
Delay 10ms					
			0x0D08	0x25	Enable packet-based control channel mode for Link C.
Delay 10ms					
			0x0E08	0x25	Enable packet-based control channel mode for Link D.
Delay 10ms					
Continue programming the quad deserializer configuration.					
Continue programming the Serializer configuration and camera.					
Delay 10ms					
19	SER	0x80	0x0004	0x83	Enable serial link.
Last	DES	0x4E	0x0006	0x0F	Enable all four links (A, B, C, and D).

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 are annotated to give a broad picture of the requirements users can expect to get a system working with the MAX96724 deserializer and MAX96717 serializers.

The format of the programming examples throughout this section will follow the format allowable by the GMSL GUI for .CPP files, so that users may copy them for use immediately.

The format for the programming is adjusted as follows:
0x80,0x0383,0x00 → 0x04, 0x80, 0x03, 0x83, 0x00, // Disable tunneling mode

Table 41 Explanation of GUI Programming for .cpp files

Number of I2C transactions	Device Address	High Register Addr.	Low Register Addr.	Register value	Description
0x04	0x80	0x03	0x83	0x00	//Description

Use Case Example 1

This use case connects three MAX96717 serializers to GMSL2 links B, C, and D of the MAX96724 deserializer. The MAX96724 uses MIPI profiles to aggregate Link B,C,D data to PHY A. The aggregated data is then copied to PHY B as a replicated output. The cameras connected to the MAX96717 serializer use RAW12 as the data type format. The SOC sends a frame synchronization signal to the serializer and has complete control over the signal.

Use Case Example 1 Block Diagram

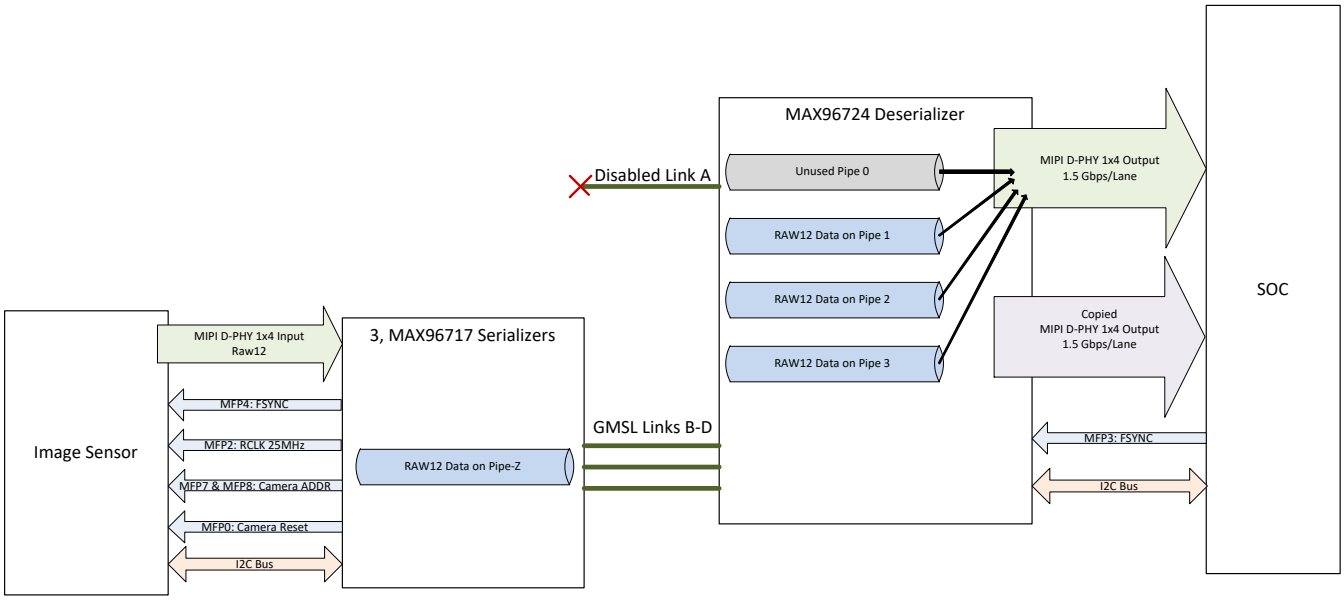

Figure 28 Use Case Example 1 Block Diagram

Use Case Example 1 Programming Script

```
//************************************************************************
// Errata: The latest device errata should be added to the program for both
// serializer and deserializer. The errata are very important.
//************************************************************************

//************************************************************************************
// MAX96717 Broadcasting Setup
//************************************************************************************
0x04, 0x80, 0x03, 0x83, 0x00, // Disable tunneling mode
0x04, 0x80, 0x03, 0x13, 0x40, // Double 12-bit data on pipe Z
0x04, 0x80, 0x03, 0x1E, 0x38, // BPP = 24 on pipe Z
0x04, 0x80, 0x03, 0x18, 0x6C, // RAW12 to pipe Z

// MFP4: FSYNC Output Signal
0x04, 0x80, 0x02, 0xCA, 0x84, // Enable receiving GPIO signal from the link.
0x04, 0x80, 0x02, 0xCB, 0x24, // Output is push-pull
0x04, 0x80, 0x02, 0xCC, 0x03, // Receive ID = 3

// MFP2: 25MHz RCLK to Camera
0x04, 0x80, 0x05, 0x6F, 0x0E, // Slew rate is maximum
0x04, 0x80, 0x00, 0x03, 0x04, // RCLK = 25MHz, use MFP2
0x04, 0x80, 0x00, 0x06, 0xB0, // Enable RCLK
```

```
// MFP7: Camera SADDR0
0x04, 0x80, 0x02, 0xD4, 0xA0, // Output is push-pull
0x04, 0x80, 0x02, 0xD3, 0x00, // Drive output low

// MFP8: Camera SADDR1
0x04, 0x80, 0x02, 0xD7, 0xA0, // Output is push-pull
0x04, 0x80, 0x02, 0xD6, 0x00, // Drive output low

// MFP0: Camera reset (active-low)
0x04, 0x80, 0x02, 0xBF, 0xA0, // Output is push-pull
0x04, 0x80, 0x02, 0xBE, 0x00, // Drive output low (Camera disabled)
0x00, 0x0A, // 10ms delay
0x04, 0x80, 0x02, 0xBE, 0x10, // Drive output high (Camera enabled)
0x00, 0x0A, // 10ms delay

//*************************************************************************
// MAX96724 Setup
//*************************************************************************
0x04, 0x4E, 0x06, 0xE1, 0x05, // Mipi Profile setup
0x04, 0x4E, 0x00, 0x06, 0xFE, // Enable Links B-D
//0x04, 0x4E, 0x08, 0xCA, 0xAA, // Controller Mapping using Method 1. Unused.
0x04, 0x4E, 0x04, 0x24, 0x00, //Disable ERRB Forward Packet output

//0x04, 0x4E, 0x09, 0x39, 0x40, // Dis auto tun detect on Pipe 0. Unused.
//0x04, 0x4E, 0x09, 0x79, 0x40, // Dis auto tun detect on Pipe 1. Unused.
//0x04, 0x4E, 0x09, 0xB9, 0x40, // Dis auto tun detect on Pipe 2. Unused.
//0x04, 0x4E, 0x09, 0xF9, 0x40, // Dis auto tun detect on Pipe 3. Unused.

0x04, 0x4E, 0x09, 0x33, 0x01, // Un-double 12-bit data on ctrl 0
0x04, 0x4E, 0x09, 0x73, 0x01, // Un-double 12-bit data on ctrl 1
0x04, 0x4E, 0x09, 0xB3, 0x01, // Un-double 12-bit data on ctrl 2
0x04, 0x4E, 0x09, 0xF3, 0x01, // Un-double 12-bit data on ctrl 3

//PHY Copy of PHY A to PHY B
0x04, 0x4E, 0x08, 0xA9, 0xC8,

// Frame Sync is disabled, SOC will send FSYNC through GPIO.
0x04, 0x4E, 0x04, 0xA0, 0x0D,
//MFP3: FSYNC Input Signal
0x04, 0x4E, 0x03, 0x09, 0x83, // Set MFP pin to transmit FSNYC to the GMSL2 link.
0x04, 0x4E, 0x03, 0x0A, 0x83, // Set Transmit ID = 3 for link A
0x04, 0x4E, 0x03, 0x41, 0x23, // Set Transmit ID = 3 for link B
0x04, 0x4E, 0x03, 0x77, 0x23, // Set Transmit ID = 3 for link C
0x04, 0x4E, 0x03, 0xAD, 0x23, // Set Transmit ID = 3 for link D

//*************************************************************************
// Camera Setup: This is where the camera startup sequence can be placed.
//*************************************************************************
```

Use Case Example 2

This use case connects four MAX96717 serializers in tunnel mode to GMSL2 links A, B, C, and D of the MAX96724 deserializer. The MAX96724 aggregates all the data to PHY A. The aggregated data is then copied to PHY B as a replicated output. The cameras connected to the MAX96717 serializer use RAW12 and EMB8 data type formats. Each camera is programmed to have a unique virtual channel 0-3. The SOC sends a frame synchronization signal to the serializer and has complete control over the signal. MIPI profiles are not used in this example, so all registers are configured independently.

Use Case Example 2 Block Diagram

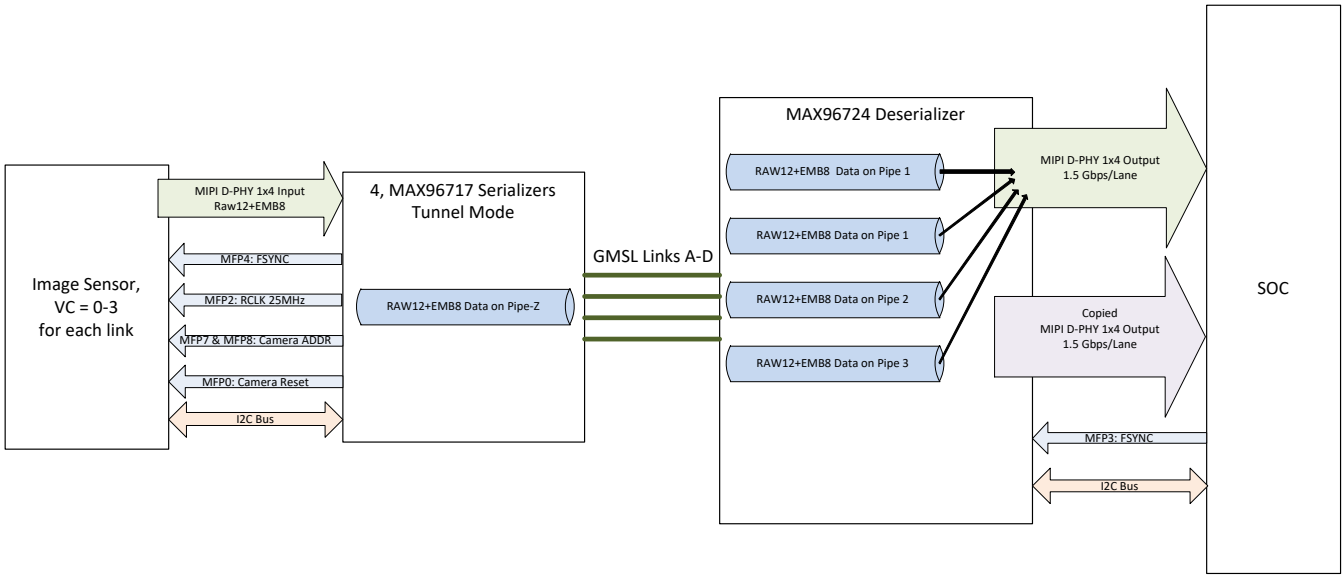

Figure 29 Use Case Example 1 Block Diagram

Use Case Example 2 Programming Script

```
//************************************************************************
// Errata: The latest device errata should be added to the program for both
// serializer and deserializer. The errata are very important.
//************************************************************************

//*****************************************************************************
// MAX96717 Broadcasting Setup
//*****************************************************************************
0x04, 0x80, 0x03, 0x83, 0x80, // Enable tunneling mode
// MFP4: FSYNC Output Signal
0x04, 0x80, 0x02, 0xCA, 0x84, // Enable receiving GPIO signal from the link.
0x04, 0x80, 0x02, 0xCB, 0x24, // Output is push-pull
0x04, 0x80, 0x02, 0xCC, 0x03, // Receive ID = 3

// MFP2: 25MHz RCLK to Camera
0x04, 0x80, 0x05, 0x6F, 0x0E, // Slew rate is maximum
0x04, 0x80, 0x00, 0x03, 0x04, // RCLK = 25MHz, use MFP2
0x04, 0x80, 0x00, 0x06, 0xB0, // Enable RCLK

// MFP7: Camera SADDR0
0x04, 0x80, 0x02, 0xD4, 0xA0, // Output is push-pull
0x04, 0x80, 0x02, 0xD3, 0x00, // Drive output low
```

```
// MFP8: Camera SADDR1
0x04, 0x80, 0x02, 0xD7, 0xA0, // Output is push-pull
0x04, 0x80, 0x02, 0xD6, 0x00, // Drive output low

// MFP0: Camera reset (active-low)
0x04, 0x80, 0x02, 0xBF, 0xA0, // Output is push-pull
0x04, 0x80, 0x02, 0xBE, 0x00, // Drive output low (Camera disabled)
0x00, 0x0A, // 10ms delay
0x04, 0x80, 0x02, 0xBE, 0x10, // Drive output high (Camera enabled)
0x00, 0x0A, // 10ms delay

//*****************************************************************************
// MAX96724 Setup
//*****************************************************************************
0x04, 0x4E, 0x04, 0x24, 0x00, // Disable ERRB Forward Packet output
0x04, 0x4E, 0x09, 0x39, 0x10, // Auto tun. detect used and Tun. Dest. set to PHYA
0x04, 0x4E, 0x09, 0x79, 0x10, // Auto tun. detect used and Tun. Dest. set to PHYA
0x04, 0x4E, 0x09, 0xB9, 0x10, // Auto tun. detect used and Tun. Dest. set to PHYA
0x04, 0x4E, 0x09, 0xF9, 0x10, // Auto tun. detect used and Tun. Dest. set to PHYA

//Registers to manually enable tunneling mode, note: auto tunnel detection must be
//disabled for this to function properly. Unused in this example.
//0x04, 0x4E, 0x09, 0x36, 0x29, //Enable Tunneling mode
//0x04, 0x4E, 0x09, 0x76, 0x29, //Enable Tunneling mode
//0x04, 0x4E, 0x09, 0xB6, 0x29, //Enable Tunneling mode
//0x04, 0x4E, 0x09, 0xF6, 0x29, //Enable Tunneling mode

// Enable D-PHY, and disable extended VC
0x04,0x4E,0x09,0x0A,0xC0,
0x04,0x4E,0x09,0x4A,0xC0,
0x04,0x4E,0x09,0x8A,0xC0,
0x04,0x4E,0x09,0xCA,0xC0,

//Mipi Output Rate
0x04,0x4E,0x04,0x15,0x2F, // 1500 Mbps/lane on port C (D-PHY)
0x04,0x4E,0x04,0x18,0x2F, // 1500 Mbps/lane on port A/D (D-PHY)
0x04,0x4E,0x04,0x1B,0x2F, // 1500 Mbps/lane on port B/E (D-PHY)
0x04,0x4E,0x04,0x1E,0x2F, // 1500 Mbps/lane on port F (D-PHY)

//Setup for 2x4 mode MIPI Output
0x04, 0x4E, 0x08, 0xA0, 0x04,

//Lane Mapping for 2x4 mode
0x04, 0x4E, 0x08, 0xA3, 0xE4,
//Lane Mapping for 2x4 mode
0x04, 0x4E, 0x08, 0xA4, 0xE4,

//PHY Copy of PHY A to PHY B
0x04, 0x4E, 0x08, 0xA9, 0xC8,

//*****************************************************************************
// Camera Setup Camera Link A
//*****************************************************************************
0x04, 0x4E, 0x00, 0x03, 0xFE, // Disable Remote control channels B,C,D
0x05, 0x20, 0x33, 0x44, 0x00, 0x03, // Change VC on camera = 0
```

```
//*****************************************************************************
// Camera Setup Camera Link B
//*****************************************************************************
0x04, 0x4E, 0x00, 0x03, 0xFB, // Disable Remote control channels A,C,D
0x05, 0x20, 0x33, 0x44, 0x01, 0x03, // Change VC on camera = 1

//*****************************************************************************
// Camera Setup Broadcast Camera Link C
//*****************************************************************************
0x04, 0x4E, 0x00, 0x03, 0xEF, // Disable Remote control channels A,B,D
0x05, 0x20, 0x33, 0x44, 0x02, 0x03, // Change VC on camera = 2

//*****************************************************************************
// Camera Setup Broadcast Camera Link D
//*****************************************************************************
0x04, 0x4E, 0x00, 0x03, 0xBF, // Disable Remote control channels A,B,C
0x05, 0x20, 0x33, 0x44, 0x03, 0x03, // Change VC on camera = 3

//Enable All I2C communications. Enable all remote channels.
0x04, 0x4E, 0x00, 0x03, 0xAA,

// Frame Sync is disabled, SOC will send FSYNC through GPIO.
0x04, 0x4E, 0x04, 0xA0, 0x0D,
//MFP3: FSYNC Input Signal
0x04, 0x4E, 0x03, 0x09, 0x83, // Set MFP pin to transmit FSNYC to the GMSL2 link.
0x04, 0x4E, 0x03, 0x0A, 0x83, // Set Transmit ID = 3 for link A
0x04, 0x4E, 0x03, 0x41, 0x23, // Set Transmit ID = 3 for link B
0x04, 0x4E, 0x03, 0x77, 0x23, // Set Transmit ID = 3 for link C
0x04, 0x4E, 0x03, 0xAD, 0x23, // Set Transmit ID = 3 for link D

//*****************************************************************************
// Camera Setup: This is where the camera startup sequence can be placed.
//*****************************************************************************
```

Appendix

Figures

Tables

Revision History

Revision	Changes	Date
1	Initial Revision	6 September 2022
2	1. Corrected Minor typographical errors. 2. Added Document Overview Section. 3. Updated Frame Sync Examples to erase an error in register write.	1 March 2023