Lamarr JESD204B

Debugging Guide

Texas Instruments

ASP – DSPS – CI – Digital Radio (Graychip)

Revision History

Date	Revision	Changes
2014-Feb-20	0.0	initial release
2014-Mar-10	0.1	FIFO errors
2014-Mar-14	0.2	Initialization sequence
2014-Mar-27	0.3	RX lane errors

If you have any questions regarding this document, please contact:

Geo Tu

geo@ti.com

(408) 543-5413

Table of Contents

Introduction

While the JESD IP is customizable with compile-time parameters for integration into multiple chips, this document is specific to Lamarr's JESD block.

This document describes issues regarding board bringup, SYSREF, asynchronous FIFOs, initialization, status signals, and error signals.

Board Bringup

Bringing up a board with a JESD link has a few challenges and things to verify. This section offers a list of things to check.

SERDES

JESD data lanes are transmitted via SERDES, which typically is implemented using differential pair transmission lines. Getting the SERDES link operational, running at the desired rates, and passing BER tests is a set of prerequisites to verifying the JESD link.

Another issue that affects the SERDES RX is clock recovery. JESD RX logic has asynchronous FIFOs that transfer the data from the SERDES RX byte clocks to the JESD/DFE clock domain. If the SERDES RX byte clocks are not stable when the JESD RX is initialized, the FIFOs will overflow or underflow. In order to successfully recover the byte clock, the SERDES RX needs to see a stream of comma codes at its input. This means any JESD TX device should be initialized before any JESD RX device so that it can send that stream of comma codes. If done in reverse order, then the JESD RX block in the JESD RX device needs to be reinitialized by software by reasserting the *init_state* signal in order to reset the asynchronous FIFOs.

SYNC

Any JESD link has one or more data lanes driven by the transmitting device to the receiving device, for example from a DSP to a DAC or from an ADC to a DSP. Every JESD link also has one SYNC signal driven in the opposite direction by the receiving device to the transmitting device. This signal must also be connected for the JESD link to work, whether it is connected physically on the board or connected internally in loopback testing by link programming.

Lanes

If a lane is enabled by programming but not physically connected, the JESD link will not successfully initialize. In other words, be sure to disable any unconnected lanes in the configuration file.

SYSREF

In any JESD204B Subclass 1 link, the local multiframe clock (LMFC) generated within each device is aligned to an external input signal called SYSREF to achieve deterministic latency. With both LMFC's in the transmit and receive devices aligned to an external reference signal, a known relationship can be determined between the LMFC's in the two devices. This allows for a known alignment between the frame counters in each device, which ultimately allows for deterministic latency to be achieved across the link. There are a few important things to note about using the SYSREF signal.

The SYSREF signal's period must be an integer multiple of the LMFC period. The LMFC period varies across configurations since it is calculated from the product of the frame length (F octets per frame) and the multiframe length (K frames per multiframe) divided by the number of octets processed per clock cycle (2 in Lamarr). In most Lamarr use cases, we've simply set the SYSREF period to a large 2^n integer multiple of the DFE clock rate.

However, this solution may not always work. If the JESD logic is running on a gated clock that is not gated by a factor of 2^n, you may need to program SYSREF to operate at a different frequency. For example, if the DFE clock is set to 368.64 MHz, and the JESD clock gating logic is gating off 1 of every 3 clocks to operate JESD an effective 245.76 MHz, then the SYSREF period may need to be changed by a factor of 1.5.

The JESD logic has the ability to trigger an interrupt due to a "SYSREF error." This error bit can indicate different things. In normal operation, it would indicate that a sampled SYSREF edge does not align with previously sampled SYSREF edges. This could be caused by an alignment drift, an asynchronous sampling of SYSREF that is out of the margin of error, an incorrect programming of the SYSREF frequency given the multiframe length, or an invalid programming of the multiframe length given available SYSREF frequencies.

Depending on the reliability of the SYSREF generating device, the SYSREF signal may glitch when it is initially turned on. To avoid sampling a potential glitch or glitches, the JESD may be programmed to ignore the first one or two SYSREF edges. This is done by programming the control signal "sysref_mode" in the JESD link layers.

By programming the JESD link layer to sample SYSREF continuously (sysref_mode = 1, 4, or 6), the SYSREF programming can be verified by simply checking for no SYSREF errors.

Once SYSREF operation has been verified, then deterministic latency can be achieved by programming the JESD RX Elastic Buffers buffer delay (RBD). Instructions for this can be found in the section called "RX Elastic Buffers" in the JESD204B Transmit and Receive Link Layer IP Cores integration spec named TI_GC_JESD204B.docx.

JESD-SERDES FIFOs

One of the first potential problem causes to check is whether the JESD-SERDES FIFOs are operating normally. Each JESDTX lane has a FIFO to transfer data from the JESD (gated) clock domain to the SERDES byte clock domain. Each JESDRX lane has a FIFO to transfer data from the SERDES byte clock domain to the JESD (gated) clock domain. These clock frequencies must precisely match for the FIFOs to run. If they don't match, the FIFOs could overflow or underflow as indicated by the write error and read error bits for each FIFO.

Each lane has four FIFO flag bits:
[0] *read_empty* - empty flag
[1] **read_error** - asserted if read request when empty (cleared by syncing FIFO with init_state)
[2] *write_full* - full flag
[3] **write_error** - asserted if write request when full (cleared by syncing FIFO with init_state)

The FIFO flags are all connected to interrupt logic. Since the FIFO empty flag is asserted by default out of reset, the interrupt bit may be high the first time it is read. To confirm that a FIFO truly is empty, you must write a 0 to the register bit to clear the interrupt and read it again.

The FIFO interrupt register addresses are documented in the tables towards the end of this document.
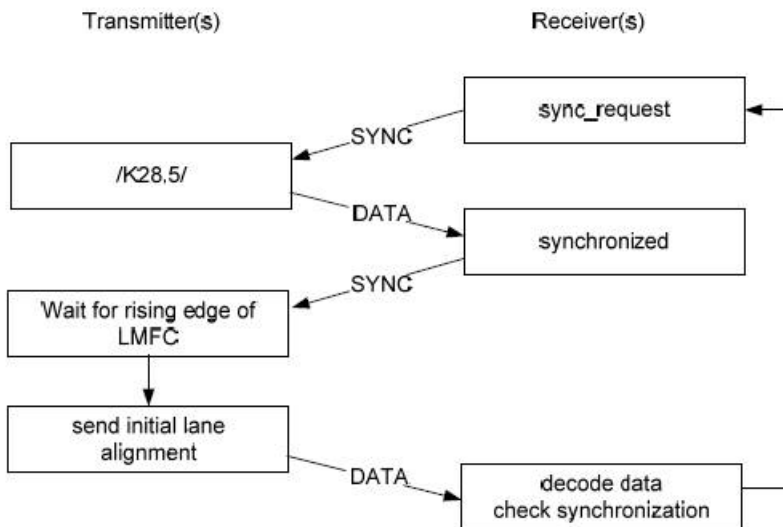
JESD Initialization

Figure 32 — Synchronization process for Subclass 1 or 2

Figure 32 from the JESD204B standard shows the order of events for successful initialization of a JESD lane. The TX has one state machine whose status register is called *sync_state*, and the RX has two state machines called *cs_state* for code group synchronization and *fs_state* for frame synchronization. To elaborate on this figure, the following is a list of the state machine status registers to check for each event during initialization.

sync_request

RX *cs_state* = 0 = CS_INIT, waiting for commas, assert sync request
RX *fs_state* = 0 = FS_INIT, sync request asserted or still receiving commas
TX *sync_state* = 0 = SYNC, waiting for sync request, transmitting commas

Note that a register to check in the TX is *first_sync_request*. This is a sticky bit that is asserted when the first sync request is received from the JESD receiver. If DFE initialization successfully finishes, this should be high and remain high.

/K28.5/

TX *sync_state* = 0 = SYNC, waiting for sync request, transmitting commas

If the TX state machine is stuck in this state, it means either it has not yet received a sync request from the RX or the RX is asserting and holding the sync request and has not deasserted it yet.

synchronized

RX *cs_state* = 2 = CS_DATA, enough commas received, deassert sync request (operational state)

If the RX state machine is stuck in this state, it means that it has received enough commas from the TX to achieve code group synchronization, but the TX has not started sending an initial lane alignment (ILA) sequence or data.

send initial lane alignment (ILA)

TX *sync_state* = 1 = INIT_LANE, transmitting initial lane alignment (ILA) sequence

Chances are low for reading back this value for the TX state machine, because it should only be in this state momentarily while it is transmitting the initial lane alignment (ILA) sequence.

decode data check synchronization

TX *sync_state* = 2 = DATA_ENC, transmitting data (operational state)
RX *fs_state* = 1 = FS_DATA, lanes have been aligned, receiving data (operational state)

Once the link has been successfully initialized, the state machines should remain in the following states:
TX *sync_state* = 2 = DATA_ENC, transmitting data (operational state)
RX *cs_state* = 2 = CS_DATA, enough commas received, deassert sync request (operational state)
RX *fs_state* = 1 = FS_DATA, lanes have been aligned, receiving data (operational state)

Breaking the Link

One way to purposely break the link is to transmit invalid 8b/10b codes. An easy way to do this is to assert the *clear_data* signal in the JESD TX, which clears the output to the SERDES. Since a value of all zeros is an invalid 10b code, this would throw the JESD RX code group synchronization state machine back to its initial state (cs_state = 0), which would cause the JESD RX to assert a sync request on the SYNC signal.

JESD TX

JESD TX Status and Error Signals
The following table includes the JESD TX status and error signals.

Signal	Address	Bits	For	Description
sync_state (read only)	0x25D00030 0x25D00030 0x25D00030 0x25D00030	1:0 5:4 9:8 13:12	lane 0 lane 1 lane 2 lane 3	alignment state machine status. All lanes on the same link should be in the same state. 0 = SYNC, waiting for sync request or waiting for sync request to deassert, transmitting commas 1 = INIT_LANE, transmitting initial lane alignment (ILA) sequence (only in this state momentarily) 2 = DATA_ENC, transmitting data – operational state (verify with this value) 3 = UNDEFINED
fifo_flags (interrupt, write 0 to clear)	0x25D01C08 0x25D01C48 0x25D01C88 0x25D01CC8	11:8 11:8 11:8 11:8	lane 0 lane 1 lane 2 lane 3	SerDes FIFO flags and errors [3] write_error - asserted if write request when full (cleared by syncing FIFO with init_state) [2] write_full - full flag [1] read_error - asserted if read request when empty (cleared by syncing FIFO with init_state) [0] read_empty - empty flag These are interrupt bits, so you must write a 0 to the bit to clear it.
first_sync_request (read only)	0x25D00034 0x25D00034	0 4	link 0 link 1	sticky bit that is asserted when the first sync request is received from the JESD receiver, cleared with init_state
sysref_err (interrupt, write 0 to clear)	0x25D02008 0x25D02008	8 9	link 0 link 1	if SYSREF comes at an unexpected time. When the core is requesting the SYSREF signal and it arrives misaligned to the LMFC, this error will be asserted high. The first SYSREF edge sampled according to the *sysref_mode* will not trigger this error. This only applies when the *sysref_mode* is one that continuously samples the SYSREF. These are interrupt bits, so you must write a 0 to the bit to clear it.

err_cnt	0x25D01828	15:0	link 0	error count as reported by the RX via the SYNC~ interface. The RX can report errors that do not require re-initialization to the TX by asserting the SYNC~ signal for two frame periods. The error count maxes out at 16 bits and does not overflow. The error count is cleared automatically whenever there is a synchronization request, and it can be cleared manually by writing a 1 to the register. Note that the TX error count is not as accurate as the RX error count, because multiple errors that are detected in the RX within a short period may be reported as only one error over the SYNC~ interface.
(read only, write 1 to clear)	0x25D01868	15:0	link 1	

JESD RX

JESD RX Status and Error Signals
The following table includes the JESD RX status and error signals.

Signal	Address	Bits	Per	Description
cs_state (read only)	0x25D40030 0x25D40030 0x25D40030 0x25D40030	1:0 5:4 9:8 13:12	lane 0 lane 1 lane 2 lane 3	code group synchronization state machine status 0 = CS_INIT, waiting for commas, assert sync request 1 = CS_CHECK, temporary state triggered by 8b/10b error 2 = CS_DATA, enough commas received, deassert sync request – operational state (verify with this value) 3 = UNDEFINED
fs_state (read only)	0x25D40034 0x25D40034 0x25D40034 0x25D40034	1:0 5:4 9:8 13:12	lane 0 lane 1 lane 2 lane 3	frame synchronization state machine status 0 = FS_INIT, sync request asserted or still receiving commas 1 = FS_DATA, lanes have been aligned, receiving data – operational state (verify with this value) 2 = FS_CHECK, temporary state triggered by receiving a comma after initialization 3 = UNDEFINED
lane_errors (interrupts, write 0 to clear)	0x25D41C08 0x25D41C48 0x25D41C88 0x25D41CC8	7:0 7:0 7:0 7:0	lane 0 lane 1 lane 2 lane 3	lane errors [7] multiframe alignment error [6] frame alignment error [5] link configuration error [4] elastic buffer overflow error (bad RBD value) [3] elastic buffer match error (first non-/K/ doesn't match match_ctrl and match_data) [2] code group synchronization error [1] 8B/10B not-in-table code error [0] 8B/10B disparity error These are interrupt bits, so you must write a 0 to the bit to clear it.
fifo_flags (interrupts, write 0 to clear)	0x25D41C08 0x25D41C48 0x25D41C88 0x25D41CC8	11:8 11:8 11:8 11:8	lane 0 lane 1 lane 2 lane 3	SerDes FIFO flags and errors [3] write_error - asserted if write request when full (cleared by syncing FIFO with init_state) [2] write_full - full flag [1] read_error - asserted if read request when empty (cleared by syncing FIFO with init_state) [0] read_empty - empty flag These are interrupt bits, so you must write a 0 to the bit to clear it.

test_seq_err (interrupts, write 0 to clear)	0x25D41C08 0x25D41C48 0x25D41C88 0x25D41CC8	12 12 12 12	lane 0 lane 1 lane 2 lane 3	test sequence error. When *test_seq_sel* for is programmed to a non-zero value to verify a specific test sequence and the test sequence verification fails, the *test_seq_err* bit will be asserted. This is not a sticky bit. *test_seq_sel* = 1 – *test_seq_err* asserted whenever input is not /D.21.5/ *test_seq_sel* = 2 – *test_seq_err* asserted whenever input is not /K.28.5/ *test_seq_sel* = 3 – *test_seq_err* asserted whenever code group synchronization has not been achieved or when input does not match the repeating initial lane alignment (ILA) sequence These are interrupt bits, so you must write a 0 to the bit to clear it.
lane_skew (read only)	0x25D4182C 0x25D4186C	4:0 4:0	link 0 link 1	lane-to-lane skew measurement in units of gated clocks. This measurement automatically updates and holds the output value whenever the elastic buffers are released. This could happen repeatedly if the skew causes buffer overflow, which subsequently causes repeated synchronization requests. Only a reset or *init_state* assertion will clear the register.
sysref_err (interrupts, write 0 to clear)	0x25D42008 0x25D42008	8 9	link 0 link 1	if SYSREF comes at an unexpected time. When the core is requesting the SYSREF signal and it arrives misaligned to the LMFC, this error will be asserted high. The first SYSREF edge sampled according to the sysref_mode will not trigger this error. This only applies when the sysref_mode is one that continuously samples the SYSREF. These are interrupt bits, so you must write a 0 to the bit to clear it.
err_cnt (read only, write 1 to clear)	0x25D41830 0x25D41870	15:0 15:0	link 0 link 1	error count. The control signal *error_ena* determines which type of errors to count. The error count maxes out at 16 bits and does not overflow. The error count is cleared automatically whenever there is a synchronization request, and it can be cleared manually by writing a 1 to the register.

JESD RX Lane Errors

Each RX lane provides eight "lane error" bits. Like all the error bits, these are connected to interrupt logic, so you must write a 0 to the bit to clear it.

 [0] 8B/10B disparity error

This error triggers whenever the input encoded 10b character is able to be decoded into an 8b value but does not match the expected running disparity based on the previous input. In other words, if the current disparity is negative, and a negative disparity code (containing 6 0's and 4 1's) was the next input, this error would be triggered. Conversely if the current disparity were positive, and a positive disparity code (containing 6 1's and 4 0's) was the next input, this error would be triggered.

[1] 8B/10B not-in-table code error

This error triggers whenever the input encoded 10b character is **not** able to be decoded into an 8b value. It is named "not-in-table code error" because there is no lookup value for the 10b code. When this occurs, the decoder outputs an 8b value of 0.

NOTE: The control signal *sync_request_ena* is a mask register that determines which error bits trigger a re-initialization sync request, which is sent to the transmitter device by asserting SYNC~ signal for a "long" period. By default, this mask enables all error bits except 0 and 1 – the 8B/10B disparity error and the 8B/10B not-in-table code error. By default, these two types of errors are reported to the transmitter device by asserting the SYNC~ signal for a "short" period, which is also known as the "error reporting" mechanism.

[2] code group synchronization error

This error occurs when the code group synchronization state machine (*cs_state*) returns to its initial state CS_INIT = 0, which occurs when the 8b/10b decoder outputs too many invalid decoded values within a short period. To debug this error, ensure the SERDES connection is stable and reliable.

[3] elastic buffer match error (first non-/K/ doesn't match match_ctrl and match_data)

Each lane's elastic buffer is triggered to start buffering data when it detects a transition from receiving commas (/K/ codes) to a non-comma. Usually, the very first octet received is an /R/ code to mark the start of the initial lane alignment (ILA) sequence. However, it could be programmed to expect something different. The elastic buffer match error occurs when the first octet received does not match what was programmed/expected. To debug this error, verify whether the TX device is sending an ILA or transitioning directly from sending commas to sending data.

[4] elastic buffer overflow error (too much lane skew or bad RBD value)

This error occurs when the elastic buffer overflows. This can happen for two reasons. If the skew between the earliest arriving lane and the latest arriving lane is too large, the elastic buffer can overflow. If the RX buffer delay or RBD value (*rbd_m1*) is selected poorly, the delay between when the buffer begins to buffer data and the next release opportunity might be too large.

[5] link configuration error

This error occurs if the initial lane alignment (ILA) sequence received from the TX device does not match what was expected. This can be due to either a mismatch in the counting sequence or a mismatch in the link configuration parameters sent in the second multiframe of the ILA. To debug this error, double check the link programming in both devices.

[6] frame alignment error

The TX device will occasionally replace the octet at the end of a frame with an /F/ code in a manner that allows for the replaced data to be restored in the RX device. This allows the RX device to continuously monitor frame alignment. The frame alignment error occurs if /F/ frame alignment replacement codes are consistently received in a position that is not at the end of a frame.

[7] multiframe alignment error

The TX device will occasionally replace the octet at the end of a multiframe with an /A/ code in a manner that allows for the replaced data to be restored in the RX device. This allows for the RX device to continuously monitor multiframe alignment. The multiframe alignment error occurs if /A/ multiframe alignment replacement codes are consistently received in a frame that is not the last frame of a multiframe.