802.3dj D2.2
Comment Resolution
Logic Track

Gary Nicholl (Cisco), Logic Track Lead Editor
Matt Brown (Alphawave Semi), 802.3dj Chief Editor
Eugene Opsasnick (Broadcom), Logic Editor

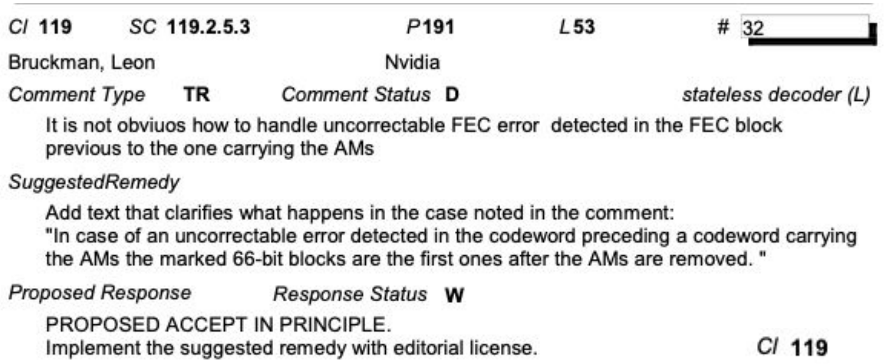
Introduction

- This slide package was assembled by the 802.3dj editorial team to provide background and detailed resolutions to aid in comment resolution.
- Specifically, these slides are for the various logic-track comments.

Stateless Decoder

Comments [#32, #392, #93]

Stateless Decoder
Comments #32, #392



Cl **119** *SC* **119.2.5.3** *P* **191** *L* **53** # 32

Bruckman, Leon Nvidia

Comment Type **TR** *Comment Status* **D** *stateless decoder (L)*

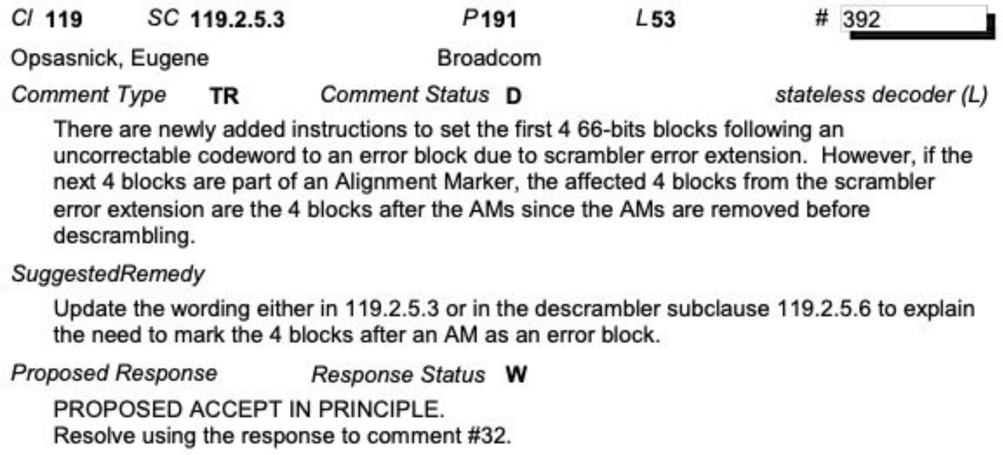
It is not obviuos how to handle uncorrectable FEC error detected in the FEC block previous to the one carrying the AMs

SuggestedRemedy

Add text that clarifies what happens in the case noted in the comment:
"In case of an uncorrectable error detected in the codeword preceding a codeword carrying the AMs the marked 66-bit blocks are the first ones after the AMs are removed. "

Proposed Response *Response Status* **W**

PROPOSED ACCEPT IN PRINCIPLE.
Implement the suggested remedy with editorial license.



Cl **119** *SC* **119.2.5.3** *P* **191** *L* **53** # 392

Opsasnick, Eugene Broadcom

Comment Type **TR** *Comment Status* **D** *stateless decoder (L)*

There are newly added instructions to set the first 4 66-bits blocks following an uncorrectable codeword to an error block due to scrambler error extension. However, if the next 4 blocks are part of an Alignment Marker, the affected 4 blocks from the scrambler error extension are the 4 blocks after the AMs since the AMs are removed before descrambling.

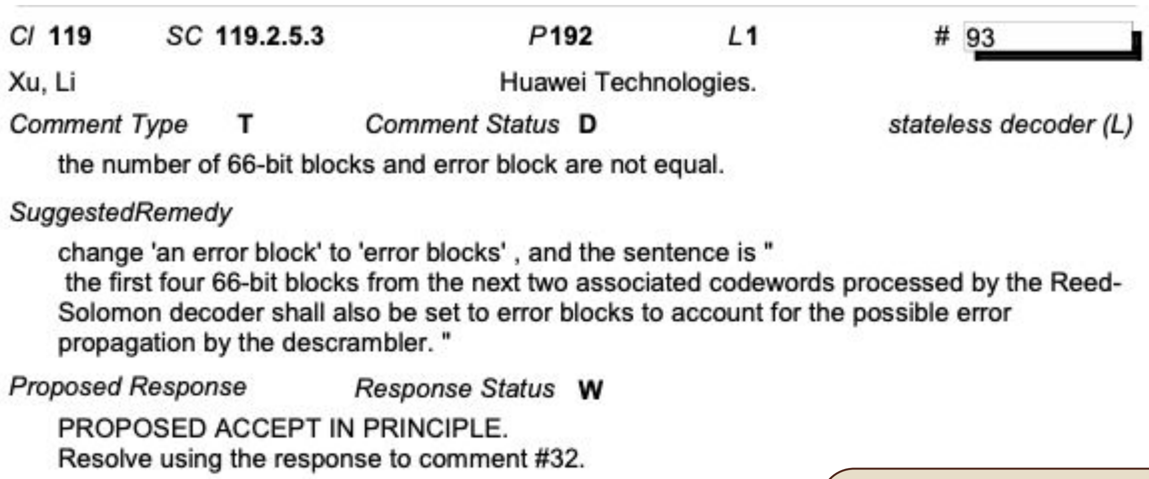
SuggestedRemedy

Update the wording either in 119.2.5.3 or in the descrambler subclause 119.2.5.6 to explain the need to mark the 4 blocks after an AM as an error block.

Proposed Response *Response Status* **W**

PROPOSED ACCEPT IN PRINCIPLE.
Resolve using the response to comment #32.

Stateless Decoder Comments #93



Cl **119** *SC* **119.2.5.3** *P* **192** *L* **1** # 93

Xu, Li Huawei Technologies.

Comment Type **T** *Comment Status* **D** *stateless decoder (L)*

the number of 66-bit blocks and error block are not equal.

SuggestedRemedy

change 'an error block' to 'error blocks' , and the sentence is "
the first four 66-bit blocks from the next two associated codewords processed by the Reed-Solomon decoder shall also be set to error blocks to account for the possible error propagation by the descrambler. "

Proposed Response *Response Status* **W**

PROPOSED ACCEPT IN PRINCIPLE.
Resolve using the response to comment #32.

This grammatical error can be fixed along with the any updates to the wording for comments #32 & #392.

Stateless Decoder
Comments #32, #392

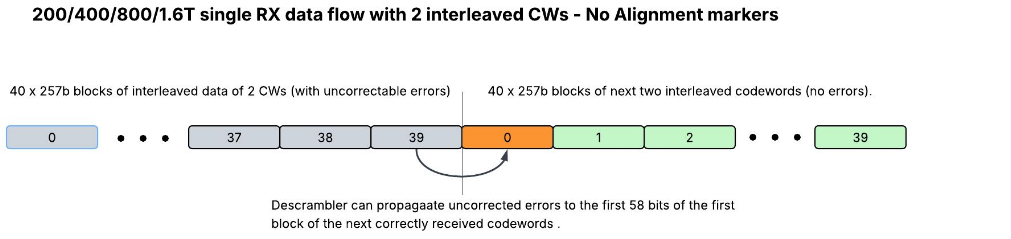
- **Updates from D2.0 to D2.1 to Clause 119 (in July) included:**

Update the 3rd paragraph of **119.2.5.3 “Reed-Solomon decoder”** with an added sentence:

> If bypass error indication is not supported or not enabled, when the Reed-Solomon decoder determines that a codeword contains errors that were not corrected, it shall cause the PCS receive function to set every 66-bit block within the two associated codewords to an error block (set to EBLOCK_R). This may be achieved by setting the synchronization header to 11 for all 66-bit blocks created from these codewords by the 256B/257B to 64B/66B transcoder. <u>When the stateless 64B/66B decoder is used as defined in 119.2.5.8.2, then the first four 66-bit blocks following the uncorrected codewords shall also be set to an error block.</u>

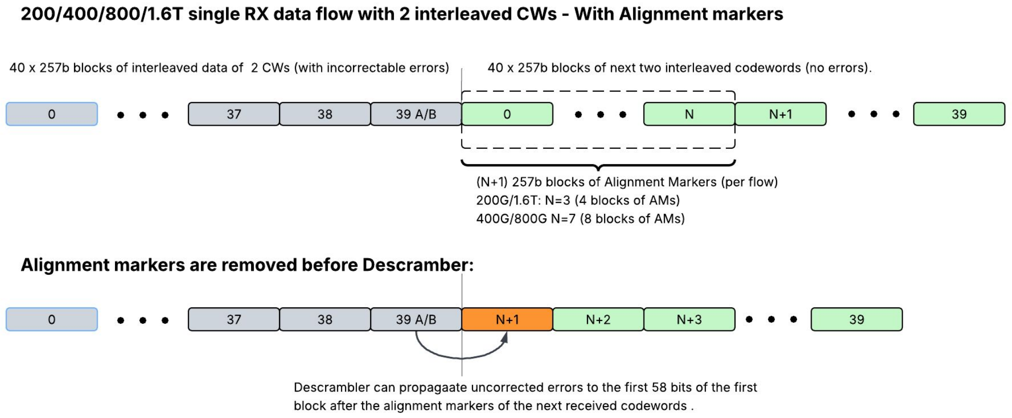
- One condition is not covered or explained well:
    - When the next codeword after an uncorrectable codeword contains the alignment markers, those alignment markers are removed before the descrambler and the descrambler entends the error (past the AMs) to the first 4 66-bit blocks after the AMs.

Stateless Decoder
Comments #32, #392


200/400/800/1.6T single RX data flow with 2 interleaved CWs - No Alignment markers
40 x 257b blocks of interleaved data of 2 CWs (with uncorrectable errors)
40 x 257b blocks of next two interleaved codewords (no errors).
0
37
38
39
0
1
2
39
Descrambler can propagaate uncorrected errors to the first 58 bits of the first block of the next correctly received codewords .

When No AMs:

The 1st 257-bit block of the next codeword after an uncorrectable codword can be corrupted be the descrambler.


200/400/800/1.6T single RX data flow with 2 interleaved CWs - With Alignment markers
40 x 257b blocks of interleaved data of 2 CWs (with incorrectable errors)
40 x 257b blocks of next two interleaved codewords (no errors).
0
37
38
39 A/B
0
N
N+1
39
(N+1) 257b blocks of Alignment Markers (per flow)
200G/1.6T: N=3 (4 blocks of AMs)
400G/800G N=7 (8 blocks of AMs)
Alignment markers are removed before Descramber:
0
37
38
39 A/B
N+1
N+2
N+3
39
Descrambler can propagaate uncorrected errors to the first 58 bits of the first block after the alignment markers of the next received codewords .

With AMs:

The 257-bit block after the AMs of the next codeword can be affected by the descrambler.

Stateless Decoder
Comments #32, #392

Update the 3rd paragraph of paragraph in **119.2.5.3 “Reed-Solomon decoder”** again:

> If bypass error indication is not supported or not enabled, when the Reed-Solomon decoder determines that a codeword contains errors that were not corrected, it shall cause the PCS receive function to set every 66-bit block within the two associated codewords to an error block (set to EBLOCK_R). This may be achieved by setting the synchronization header to 11 for all 66-bit blocks created from these codewords by the 256B/257B to 64B/66B transcoder. When the stateless 64B/66B decoder defined in 119.2.5.8.2 is used, then the first four 66-bit blocks *(accounting for any alignment marker removal)* following the associated codewords shall also be set to error blocks.

Deskew State Diagrams

Comments #366, #374

Deskew State Diagrams - Clause 184 and 186
Comment #366 and #374

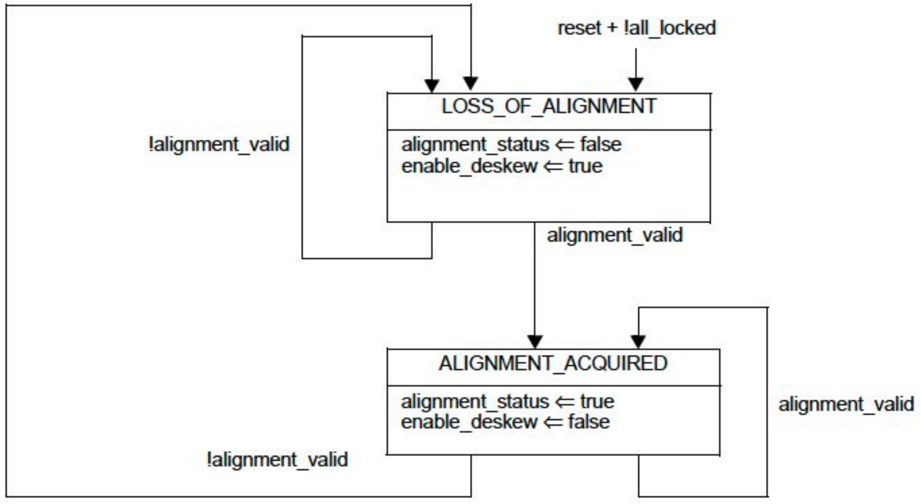


Figure 184–10—Deskew state diagram

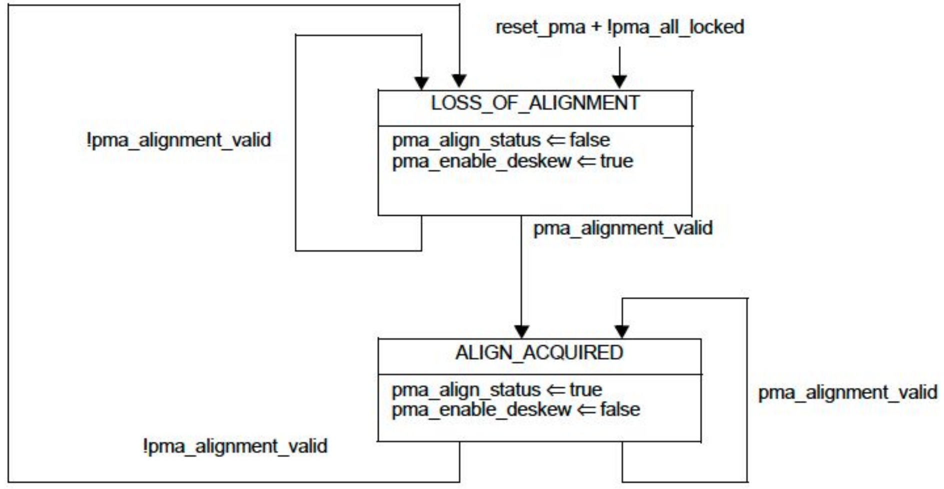


Figure 186–18—800GBASE-ER1 PMA deskew state diagram

- In both state diagrams the value of alignment_status / pma_align_status is exactly the same as alignment_valid / pma_alignment_valid. The [pma_]alignment_valid variables have “complete” definitions, making the “status” variables unnecessary.
- The enable_deskew variables are set but never used.
- Both state machines do not add information in a meaningful way and can be removed.

Deskew State Diagrams - Clause 184 - other changes 1
Comment #366

Delete unused variables in 184.7.2.2 :

~~alignment_status~~

~~A Boolean variable set by the deskew process to reflect the status of the X polarization symbol stream to Y polarization symbol stream alignment. Set to true when the polarization symbol streams are synchronized and aligned and set to false otherwise.~~

alignment_valid

A Boolean variable that is set to true if the polarization symbol streams are aligned. Polarization symbol streams are considered to be aligned when dsp_lock<*x*> is true for both *x*, each polarization symbol stream has a unique identifier (dsp_ps_id<0> is different from dsp_ps_id<1>), and the polarization symbol streams are deskewed. Otherwise, this variable is set to false.

~~all_locked~~

~~A Boolean variable that is set to true when dsp_lock<*x*> is true for both *x* and is set to false when dsp_lock<*x*> is false for either *x*.~~

~~enable_deskew~~

~~A Boolean variable that set to true when deskew is enabled and set to false when deskew is disabled. Received symbols may be discarded whenever deskew is enabled.~~

Deskew State Diagrams - Clause 184 - other changes 2
Comment #366

Replace all instances of *alignment_status* with *alignment_valid*

- In 184.3:

The SIGNAL_OK parameter is set to OK when alignment_status (see 184.7.2.2) is true and FAIL when alignment_status is false.

- In 184.5.7:

Inner_FEC_corrected_cw_counter

A 32-bit counter that counts once for each corrected FEC codeword processed by any of the 32 BCH decoders when alignment_status is true.

Inner_FEC_uncorrected_cw_counter

A 32-bit counter that counts once for each uncorrected FEC codeword processed by any of the 32 BCH decoders when alignment_status is true.

Inner_FEC_total_bits_counter

A 64-bit counter that counts once for each bit processed by any of the 32 BCH decoders when alignment_status is true.

Inner_FEC_corrected_bits_counter

A 64-bit counter that counts once for each bit corrected by any of the 32 BCH decoders when alignment_status is true.

Inner_FEC_cw_counter

A 48-bit counter that counts once for each FEC codeword received when alignment_status is true.

Inner_FEC_codeword_error_bin_*k*

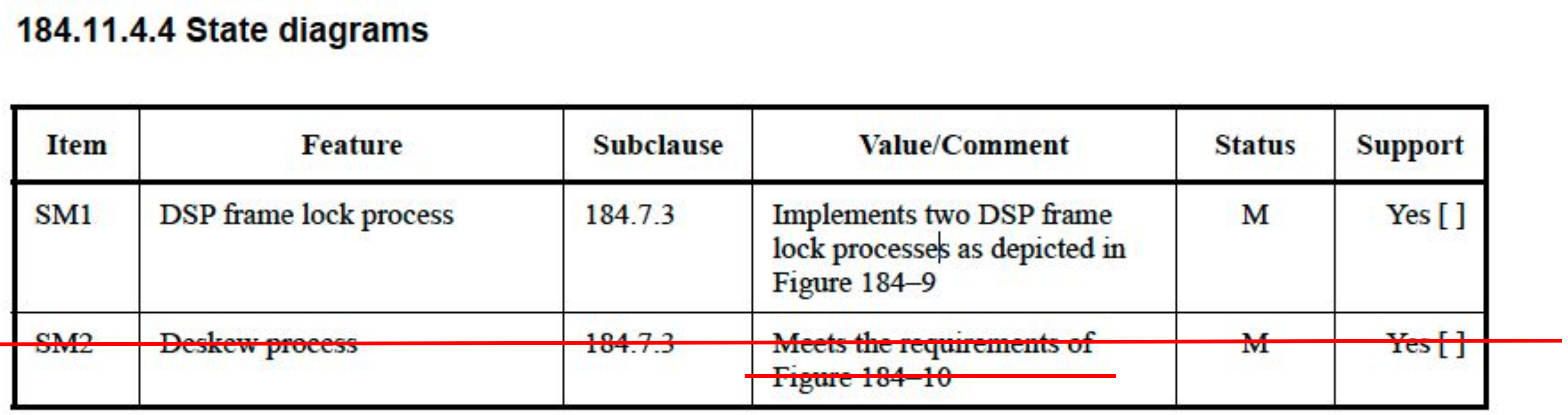
A set of *k*+1 32-bit counters where *k* = 0 to 4. While alignment_status is true, for each Inner FEC codeword received with exactly *k* corrected (flipped) bits ….

Deskew State Diagrams - Clause 184 - other changes 3
Comment #366

- Remove Figure 184-10 and all references to it.
- In 184.7.3:

~~The Inner FEC shall also implement the deskew process as shown in Figure 184–10.~~

- In 184.11.4.4 (PICS):



184.11.4.4 State diagrams

Item	Feature	Subclause	Value/Comment	Status	Support
SM1	DSP frame lock process	184.7.3	Implements two DSP frame lock processes as depicted in Figure 184–9	M	Yes [ ]
~~SM2~~	~~Deskew process~~	~~184.7.3~~	~~Meets the requirements of Figure 184–10~~	~~M~~	~~Yes [ ]~~

Deskew State Diagrams - Clause 186 - changes Comment #374

- Remove Figure 186-18 and all references to it.
- In 186.4.3:

  ~~The 800GBASE-ER1 PMA shall also implement the deskew process as shown in Figure 186–18.~~

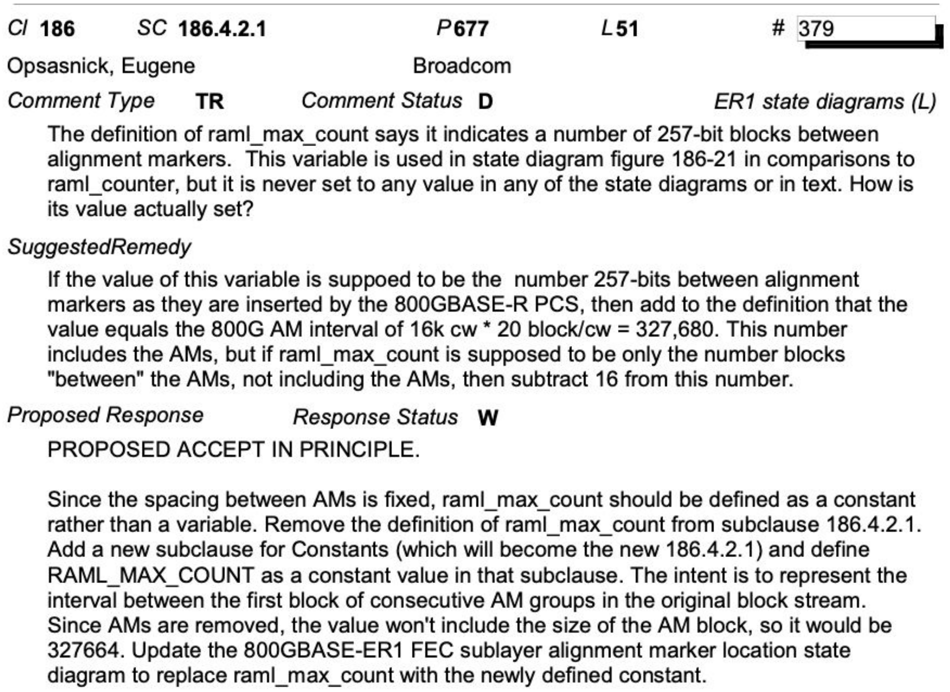
- Delete unused variables in 186.4.2.1 variable definitions:
  - pma_align_status
  - pma_all_locked
  - pma_enable_deskew

- Update text in 186.3.4.3 to clarify the deskew process.

ER1 State Diagrams

Comments #379, #386, #390

RAML_MAX_COUNT Constant Comment #379



Cl **186** *SC* **186.4.2.1** *P* **677** *L* **51** # 379

Opsasnick, Eugene Broadcom

Comment Type **TR** *Comment Status* **D** *ER1 state diagrams (L)*

The definition of raml_max_count says it indicates a number of 257-bit blocks between alignment markers. This variable is used in state diagram figure 186-21 in comparisons to raml_counter, but it is never set to any value in any of the state diagrams or in text. How is its value actually set?

SuggestedRemedy

If the value of this variable is suppoed to be the number 257-bits between alignment markers as they are inserted by the 800GBASE-R PCS, then add to the definition that the value equals the 800G AM interval of 16k cw * 20 block/cw = 327,680. This number includes the AMs, but if raml_max_count is supposed to be only the number blocks "between" the AMs, not including the AMs, then subtract 16 from this number.

Proposed Response *Response Status* **W**

PROPOSED ACCEPT IN PRINCIPLE.

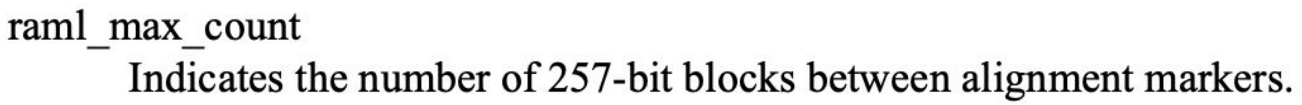
Since the spacing between AMs is fixed, raml_max_count should be defined as a constant rather than a variable. Remove the definition of raml_max_count from subclause 186.4.2.1. Add a new subclause for Constants (which will become the new 186.4.2.1) and define RAML_MAX_COUNT as a constant value in that subclause. The intent is to represent the interval between the first block of consecutive AM groups in the original block stream. Since AMs are removed, the value won't include the size of the AM block, so it would be 327664. Update the 800GBASE-ER1 FEC sublayer alignment marker location state diagram to replace raml_max_count with the newly defined constant.

There are a few pieces to this remedy:

1. Change variable raml_max_count to a be defined as a constant.
2. Add the actual value to the constant definition.
3. Update the state diagram that uses this value.

RAML_MAX_COUNT Constant – Fix - 1

- Current variable definition in 186.4.2.1 “Variables”:



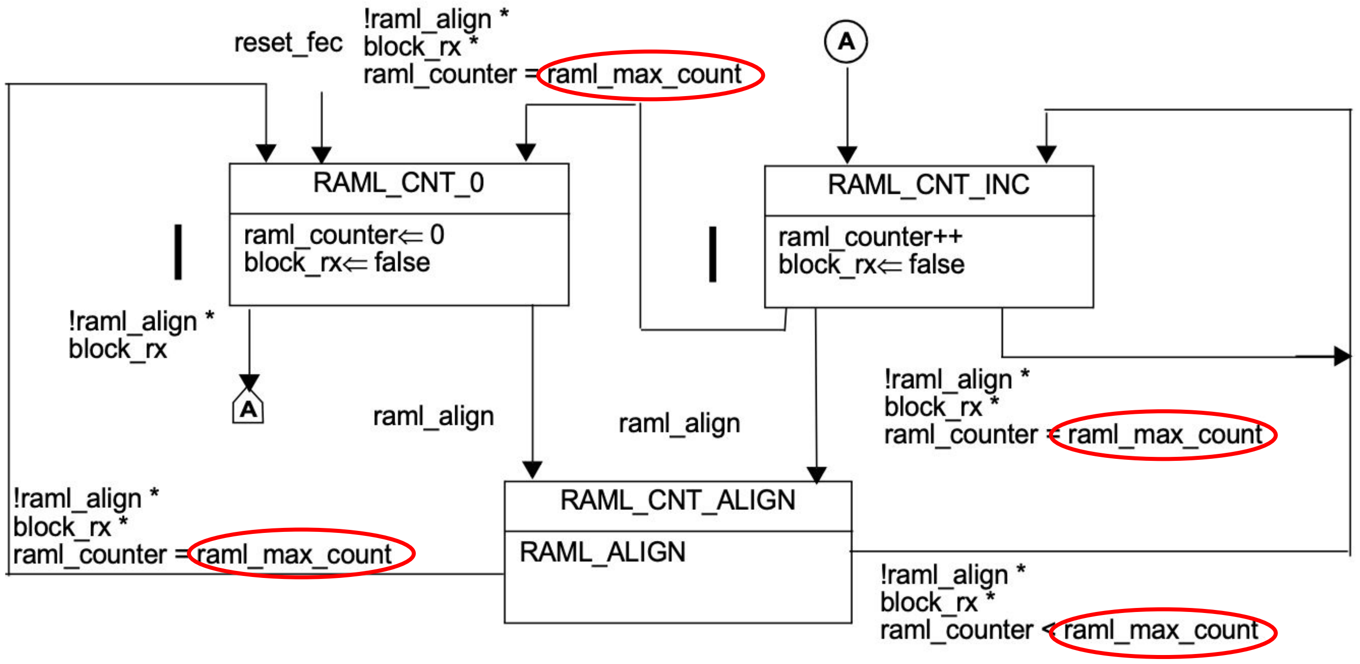
raml_max_count
    Indicates the number of 257-bit blocks between alignment markers.

- Step 1: Remove the above variable definition and add a new Constant definition in a new 186.4.2.1 “Constants” subclause (pushing the “Variables” subclause to 186.4.2.2):

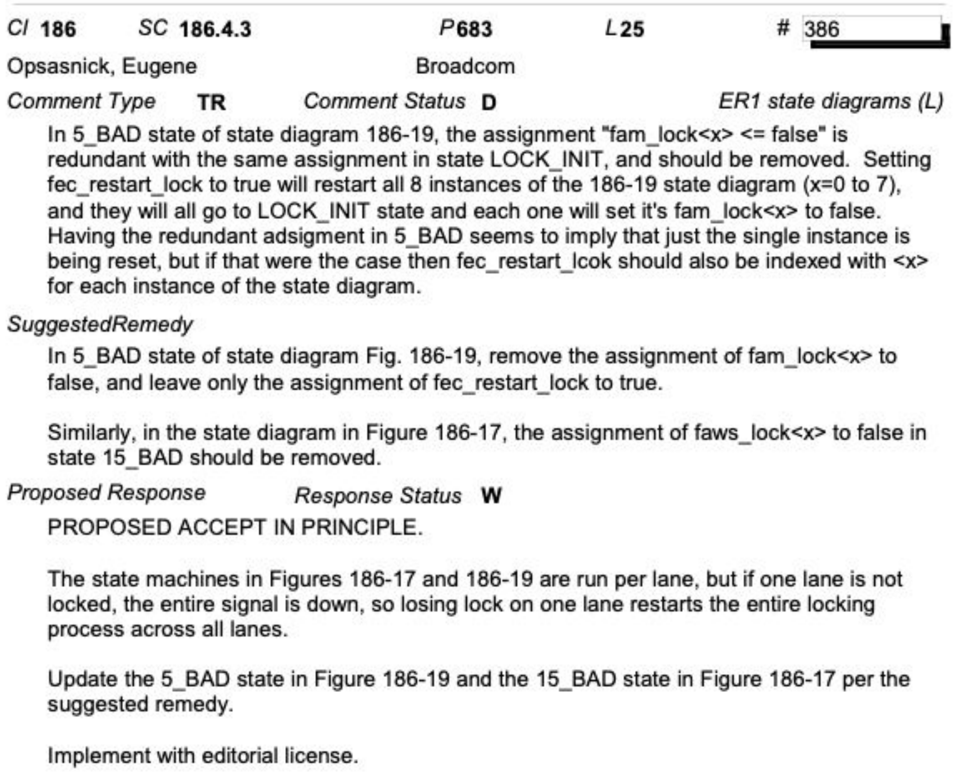
AM_INTERVAL
    The number of 257-bit blocks between alignment markers. For 800GBASE-R it is set to 327 664.

RAML_MAX_COUNT Constant – Fix - 2

- Step 2: Replace the variable raml_max_count with the constant AM_INTERVAL in Figure 186-21.



FAM_lock restart (and FAWS_lock restart) - Fig. 186-19 and Fig. 186-17 Comment #386



Cl **186** *SC* **186.4.3** *P* **683** *L* **25** # 386

Opsasnick, Eugene Broadcom

Comment Type **TR** *Comment Status* **D** *ER1 state diagrams (L)*

In 5_BAD state of state diagram 186-19, the assignment "fam_lock<x> <= false" is redundant with the same assignment in state LOCK_INIT, and should be removed. Setting fec_restart_lock to true will restart all 8 instances of the 186-19 state diagram (x=0 to 7), and they will all go to LOCK_INIT state and each one will set it's fam_lock<x> to false. Having the redundant adsigment in 5_BAD seems to imply that just the single instance is being reset, but if that were the case then fec_restart_lcok should also be indexed with <x> for each instance of the state diagram.

SuggestedRemedy

In 5_BAD state of state diagram Fig. 186-19, remove the assignment of fam_lock<x> to false, and leave only the assignment of fec_restart_lock to true.

Similarly, in the state diagram in Figure 186-17, the assignment of faws_lock<x> to false in state 15_BAD should be removed.

Proposed Response *Response Status* **W**

PROPOSED ACCEPT IN PRINCIPLE.

The state machines in Figures 186-17 and 186-19 are run per lane, but if one lane is not locked, the entire signal is down, so losing lock on one lane restarts the entire locking process across all lanes.

Update the 5_BAD state in Figure 186-19 and the 15_BAD state in Figure 186-17 per the suggested remedy.

Implement with editorial license.

FAM_lock restart - Figure 186-19
Comment #386

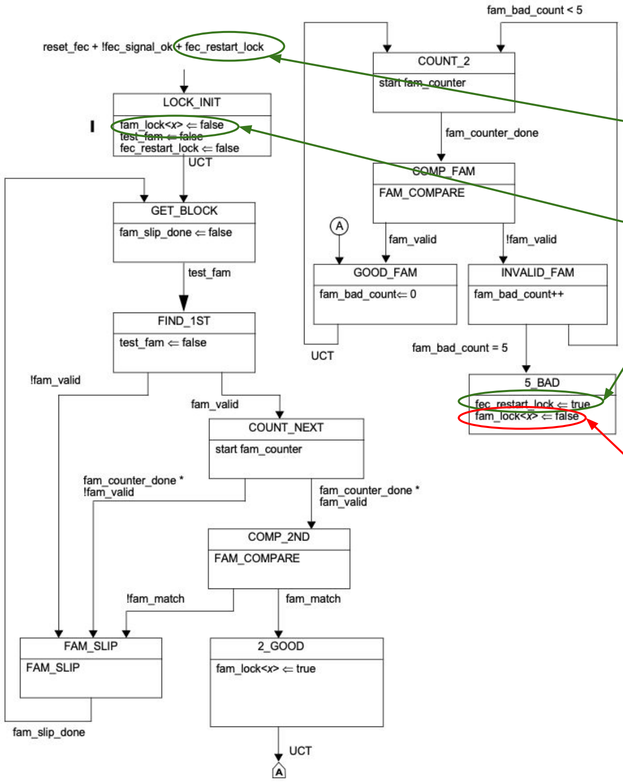


Figure 186–19—800GBASE-ER1 FEC sublayer FAM field lock state diagram

(1) Setting fec_restart_lock to true in 5_BAD state causes all 8 instances of this state diagram to restart the lock process.

(2) Each of the 8 instances resets fam_lock<x> to false in the LOCK_INIT state

Setting fam_lock<x> to false in the 5_BAD state is redundant and a little misleading.

Remove this line from 5_BAD.

FAWS_lock restart - Figure. 186-17 Comment #386

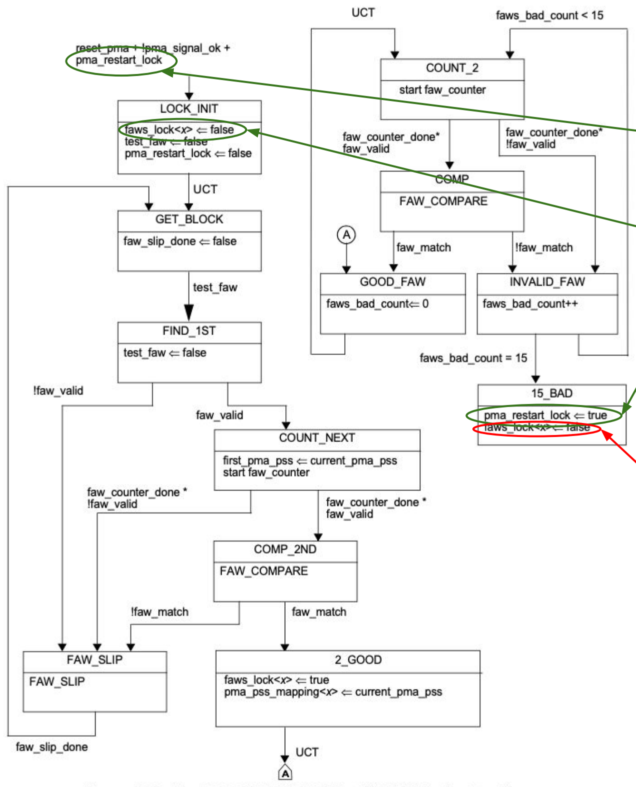


Figure 186–17—800GBASE-ER1 PMA FAW field lock state diagram

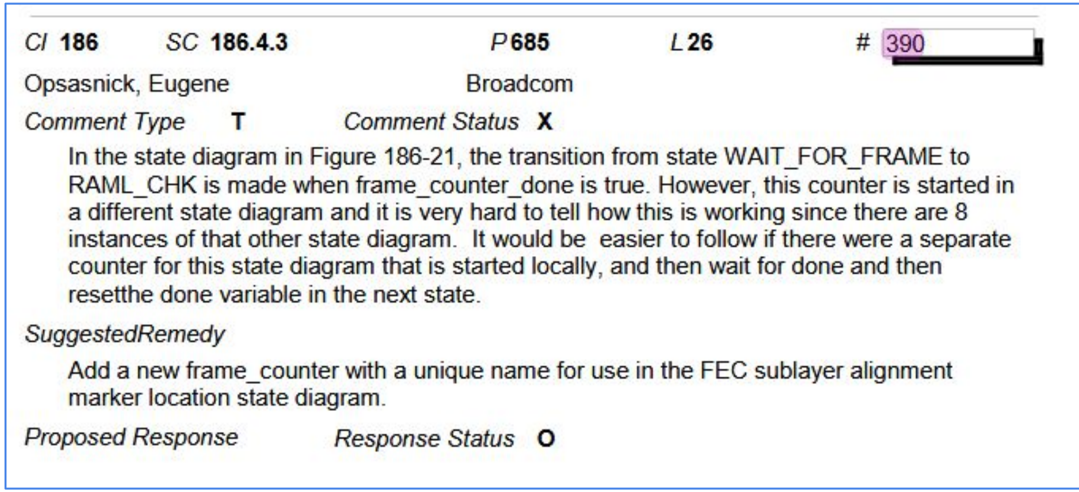
(1) Setting pma_restart_lock to true in 15_BAD state causes all 8 instances of this state diagram to restart the lock process.

(2) Each of the 8 instances resets faws_lock<x> to false in the LOCK_INIT state

Setting faws_lock<x> to false in the 15_BAD state is redundant and a little misleading.

Remove this line from 15_BAD.

Frame_counter across state diagrams
Comment #390



Cl **186**	*SC* **186.4.3**	*P* **685**	*L* **26**	# 390

Opsasnick, Eugene Broadcom

Comment Type **T** *Comment Status* **X**

In the state diagram in Figure 186-21, the transition from state WAIT_FOR_FRAME to RAML_CHK is made when frame_counter_done is true. However, this counter is started in a different state diagram and it is very hard to tell how this is working since there are 8 instances of that other state diagram. It would be easier to follow if there were a separate counter for this state diagram that is started locally, and then wait for done and then resetthe done variable in the next state.

SuggestedRemedy

Add a new frame_counter with a unique name for use in the FEC sublayer alignment marker location state diagram.

Proposed Response *Response Status* **O**

The variable frame_counter appears in the state diagrams in figures 186-20 and 186-21, but the counter that is required is different in each diagram:

- Figure 186-20 is dealing with multiframe alignment for the tributary flows and uses a counter to verify the MFAS overhead is as expected in consecutive frames of the multiframe (i.e., that the MFAS OH follows the expected sequence of values), so it needs to count blocks in a frame. No changes needed to this diagram or variable.
- Figure 186-21 is dealing with the AML overhead, which appears only once per multiframe. As such, this diagram needs a different counter that counts blocks in a multiframe (i.e., between instances of the AML OH).

Frame_counter across state diagrams - Figure 186-20 Comment #390

There are 8 instances of the Fig. 186-20 state diagram process.

Each instance has a "frame_counter<x>"

Comment #385 will update "frame_counter" to "frame_counter<x>" in this state diagram.

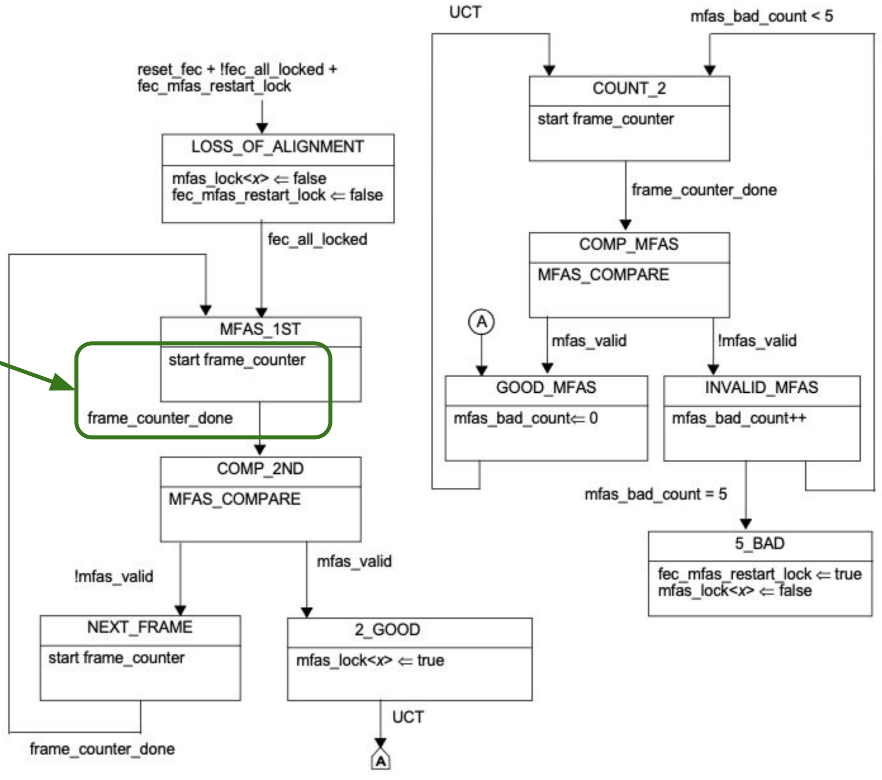


Figure 186–20—800GBASE-ER1 FEC sublayer multi-frame alignment state diagram

Frame_counter across state diagrams - Figure 186-21 Comment #390

There is one instance of the Fig. 186-20 state diagram process.

Change frame_counter_done to multiframe_count_done.

Add new counter multiframe_counter (with editorial license).

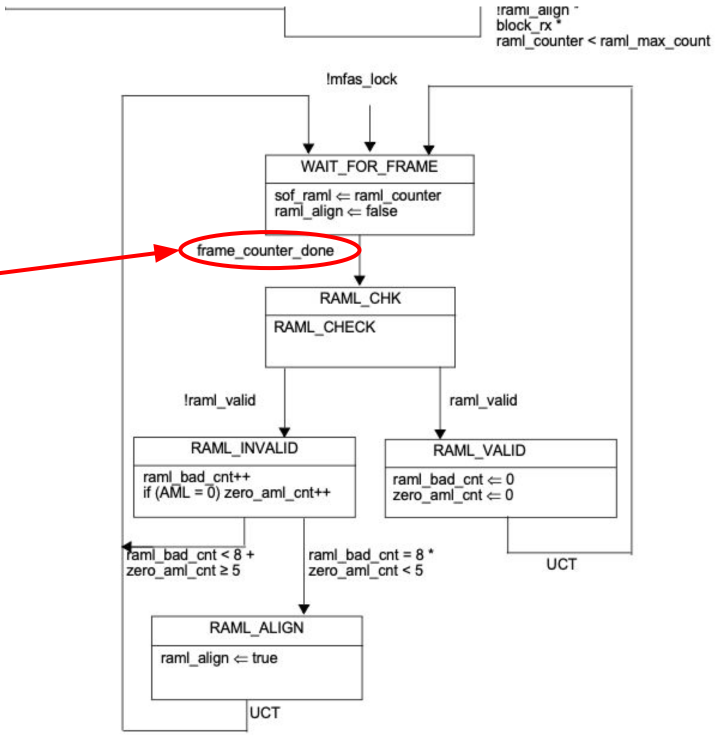


Figure 186–21—800GBASE-ER1 FEC sublayer alignment marker location state diagram

Inner FEC Pad Scrambler

Comment #197

Inner FEC Pad Scrambler - the issue
Comment #197

- 177.4.7.2 defines a 13-bit scrambler with a reference to Figure 94-6 and Equation 94-3.

$$G(x) = 1 + x + x^2 + x^{12} + x^{13} \qquad (94\text{–}3)$$

- Figure 94-6 is a PRBS13 generator, and needs to be modified slightly to be a self-sync scrambler

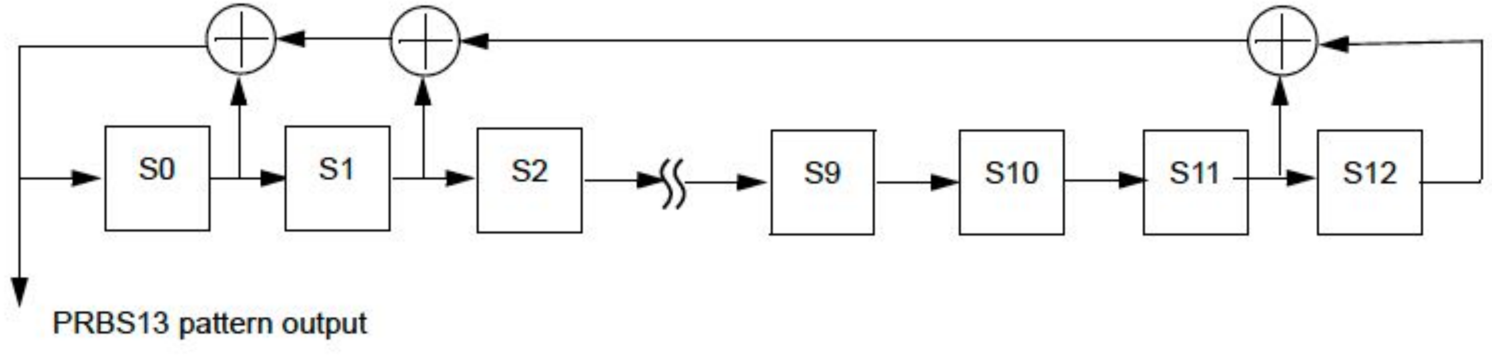


Figure 94–6—PRBS13 pattern generator

Inner FEC Pad Scrambler - new figure for 177.4.7.2 Comment #197

- Add a new figure to 177.4.7.2, based on Figure 94-6, but adding the data input.

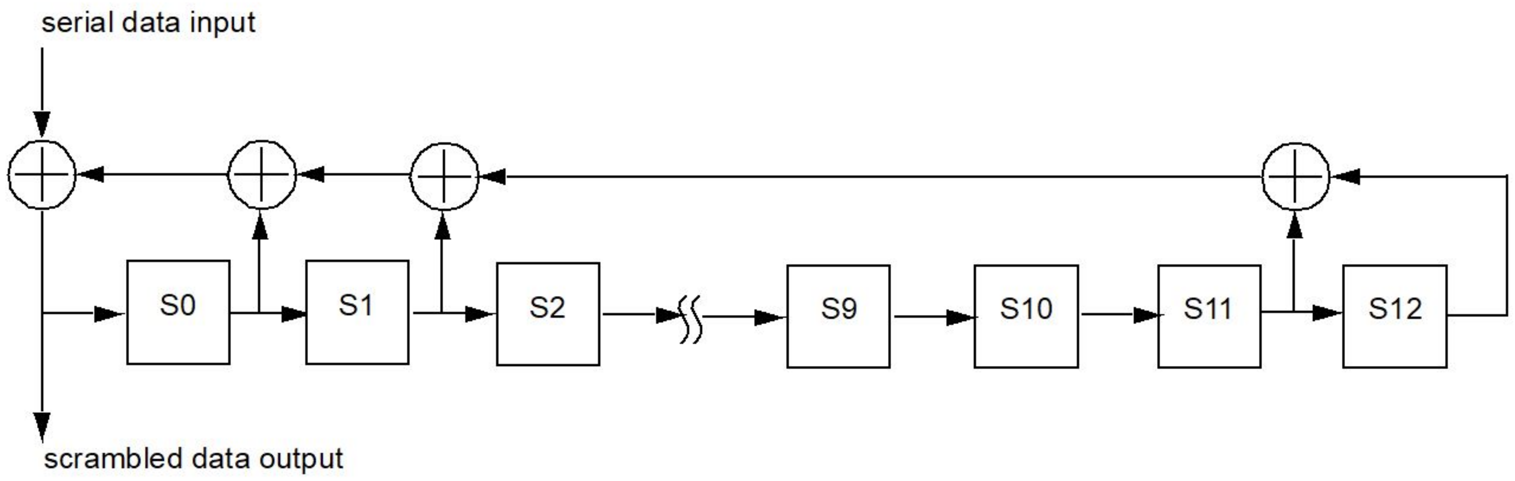


Figure 177–n—PRBS13 scrambler

PCS State Variables

Comment #362

PCS State Variables - 1
Comment #362

Current variables in question:

restart_lock

Boolean variable that is set by the PCS synchronization state diagram (see Figure 175–8) to restart the alignment marker lock process on all PCS lanes. It is set to true in the DESKEW_FAIL state or if restart_lock<*z*> is true for any *z*. It is set to false upon entry into the LOSS_OF_ALIGNMENT state.

restart_lock<*z*>

Boolean variable that is set by the PCS codeword monitor state diagram (see Figure 175–9) to restart the alignment marker lock process on all PCS lanes. It indicates that three consecutive uncorrected codewords are received for FEC codeword *z* where *z* = {A, B, C, or D}.

Two Issues:

1. Two separate variables with same basic name (one is indexed) should be renamed to remove any confusion between them.
2. Restart_lock can be set to true by both a state diagram and by its own definition when another variable (restart_lock<z>) is set to true in a different state diagram.

PCS State Variables - 2
Comment #362

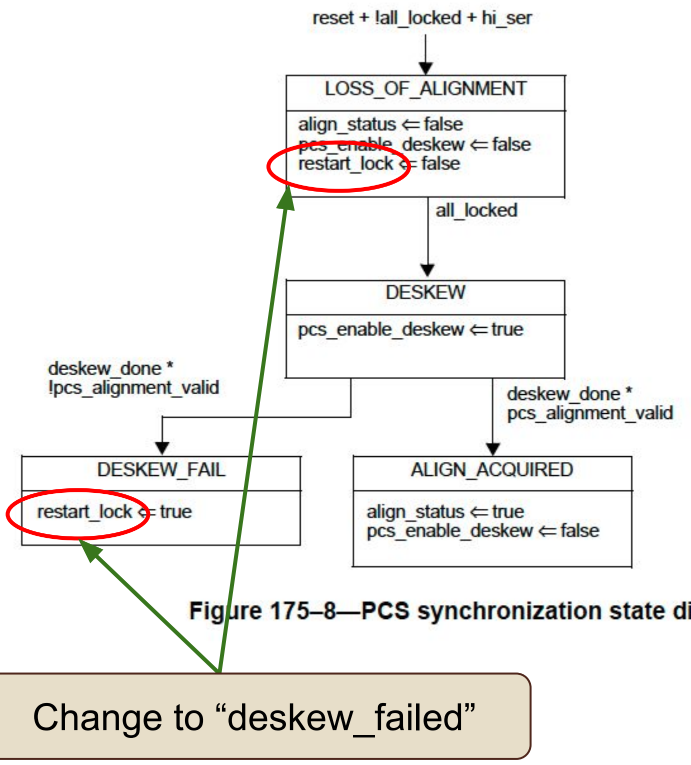

reset + !all_locked + hi_ser
LOSS_OF_ALIGNMENT
align_status ⇐ false
pcs_enable_deskew ⇐ false
restart_lock ⇐ false
all_locked
DESKEW
pcs_enable_deskew ⇐ true
deskew_done * !pcs_alignment_valid
deskew_done * pcs_alignment_valid
DESKEW_FAIL
restart_lock ⇐ true
ALIGN_ACQUIRED
align_status ⇐ true
pcs_enable_deskew ⇐ false
Change to “deskew_failed”

Figure 175–8—PCS synchronization state diagram

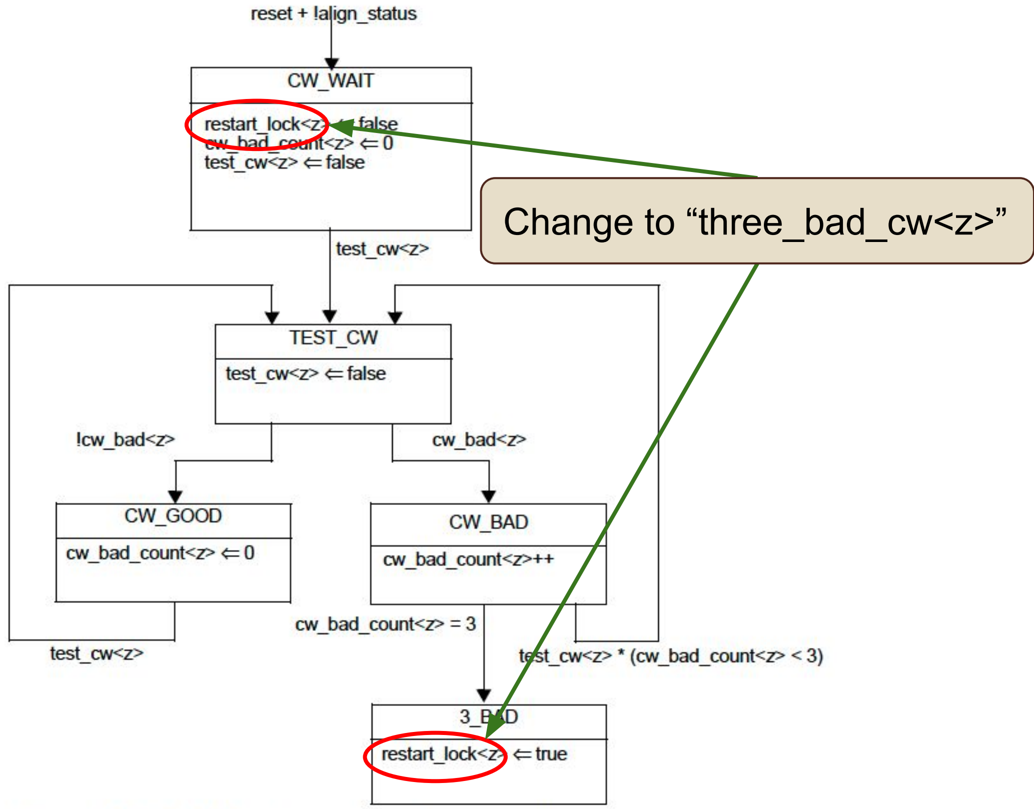


Figure 175–9—PCS codeword monitor state diagram

PCS State Variables - 3
Comment #362

New Variable definitions:

deskew_failed

Boolean variable that indicates the pcs synchronization process failed to identify 16 unique PCS lanes and is used to set the restart_lock variable. The value of deskew_failed is set by the PCS synchronization state diagram (Figure 175–8).

three_bad_cw<*z*>

Boolean variable that is set by the PCS codeword monitor state diagram (Figure 175–9) and is used to set the restart_lock variable. It indicates that three consecutive uncorrected codewords are received for FEC codeword *z* where *z* = {A, B, C, or D}.

restart_lock

Boolean variable that is used to restart the alignment marker lock process on all PCS lanes (see Figure 119-12). Its value is set to true if deskew_failed is true or if three_bad_cw<z> is true for any z. Otherwise, this variable is set to false.

Thank you