Modify section 92.2.4.1 to read:
"The codeword synchronization function receives data via 16-bit PMA_UNITDATA.request primitive.

The synchronizer shall form a bit stream from the primitives by concatenating requests with the bits of each primitive in order from rx_data-group<0> to rx_data-group<15> (see Figure 92-##). It obtains lock to the 31*66-bit blocks in the bit stream using the sync headers and outputs 2040 bit codewords to the FEC decoder function. Lock is obtained as specified in the codeword lock state machine shown in Figure 92-## (3av_0803_effenberger_5).

The incoming sync header pattern is 27 conventional (clause 49) sync headers (01 or 10), and then 00, 11, 11, and 00. The state machine performs a search for this pattern, and when it finds a perfect match of two full codewords (62 blocks), it then asserts codeword lock.

When in codeword lock, the state machine accumulates the appropriate contents of the 31 blocks that constitute a codeword in an input buffer. When the codeword is complete, the FEC decoder is triggered, and the input buffer is freed for the next codeword.

When in codeword lock, the state machine continues to check for sync header validity. If 16 or more sync headers in a codeword pair (62 blocks) are invalid, then the state machine deasserts codeword lock. In addition, if the Persistent decode failure signal becomes set, then codeword lock is deasserted (this check insures that certain false-lock cases are not persistent.)"

Insert the following text at the end of first paragraph in section 92.2.4.2:
"The handling of data leaving the FEC decoder and going to the descrambler is specified in the FEC-decoder state machine shown in Figure 92-X.

The synchronizer state machine accumulates a full codeword in a buffer. If the synchronizer is locked, then the FEC decoding process is triggered. The FEC algorithm then processes the buffer. The algorithm produces two outputs: the Decode_success signal and (if successful) the corrected buffer. The data portion of the buffer is then read out to the descrambler logic in 66 bit blocks, as normal. Note that the rate of 66 bit transfers is lower then normal here. This is corrected in the idle insertion step.

If the Decode_success is false, then a counter is incremented. It there are three decoding failures in a row, then the Persist_dec_fail signal is asserted. This signal will then reset the synchronizer."

Add the following variables to section 92.2.4.6.2
decode_success
Boolean indication that is set true if the codeword was successfully decoded by the FEC algorithm, and false otherwise.

decode_failures
Counter that holds the number of consecutive decoding failures.

persist_dec_fail
Boolean indication that is set when three consecutive decoding failures have occured.

decode_done
Boolean indication that is transiently set when the FEC decoder algorithm has completed its processing and the corrected data is present in the output buffer.

input_buffer[]
An array of 2040 bits.

input_buffer_location
An integer that points to the next appending location in the input buffer.

output_buffer[]
An array of 2040 bits.

Delete the "Force()" function from section 92.2.4.6.3

Add the following functions to section 92.2.4.6.3
Flush_inbuffer()
Flushes the input buffer of the FEC decoding algorithm block.

```
Flush_inbuffer()
  {
    for(i=0, i<2040, i++) {
      inbuffer[i]=0
    }
    input_buffer_location = 29
  }
```

Append_inbuffer()
Appends the newly arrived 66b bit block into the input buffer of the FEC decoding algorithm, taking care to only insert the bits to be protected, and discarding the unwanted bits.

```
Append_inbuffer()
  {
    BlockFromGearbox()

    if(rx_coded<0> <> rx_coded<1>) {
      inbuffer[input_buffer_location]=rx_coded<1>
      input_buffer_location++
    }
```

```
   for(i=2, i<66, i++) {
     inbuffer[input_buffer_location]=rx_coded<i>
     input_buffer_location++
   }
   if(rx_coded<0>=1 and rx_coded<1>=1) {
    cword_done=true
   }
  }
```

Decode()
Triggers the FEC decoding algorithm to accept the contents of the input buffer, and do its decoding work. Note that this function is not blocking, and returns immediately. It is assumed that the FEC decoding algorithm copies the input buffer contents into its own internal memory, so that the input buffer is released to accept the next codeword.

DecodeWhenReady()
Determines if the inbuffer contains a full codeword, and if so, it triggers the Decode function, and then clears the inbuffer for the next codeword.

```
DecodeWhenReady()
{
  if (sh_cnt=0 or sh_cnt=31) {
    if (cword_lock) {
      Decode();
    }
    Flush_inbuffer();
  }
}
```

Read_outbuffer(i)
Passes output buffer contents to the descrambler, with the appropriate format.

```
Read_outbuffer[i]
  {
    int offset = 29+i*65
    for(j=0, j<65, j++) {
      rx_coded_corrected<j+1> = out_buffer[j+offset]
    }
    rx_coded_corrected<0>=!rx_coded_corrected<1>
    BlockToDescrambler()
  }
```

BlockFromGearbox
Function that accepts the next rx_coded<0..65> block of data from the gearbox. It does not return until the transfer is completed.

BlockToDescrambler
Function that sends the next rx_coded_corrected<0..65> block to the scrambler. It does not return until the transfer is completed.

Add the attached figure to section 92.2.4.6.6.