CommentID 1036
Proposed response:
Insert the following text at the end of first paragraph in section 92.2.4.2: **FEC Decoder**
"The exact handling of data through the FEC decoder is specified in the FEC-decoder state machine shown in Figure 92-X. It should be noted that there are two separate threads of execution in this state machine, to reflect the fact that the FEC decoding process takes considerable time.

When the synchronizer is in the unlocked state, the FEC decoder is inactive. When the synchronizer is in the locked state, the 66 bit blocks that are arriving from the synchronizer are added to a buffer that accumulates only the bits that are considered by the FEC algorithm (see figure 92-Z). [Editor to create inverted version of figure 92-6] 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. [Editor to provide cross reference to new idle insertion state diagram, see comment 931]

If the Decode_success is false, then a counter is incremented. If 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 Variables**
decode_success
Type: Boolean.
Set to true if the codeword was successfully decoded by the FEC algorithm, otherwise false.
Default: na

persist_dec_fail
Type: Boolean
Set when three consecutive decoding failures have occured.
Default: set

decode_done
Type: Boolean
Transiently set when the FEC decoder algorithm has completed its processing and the corrected data is present in the output buffer.
Default: cleared.

input_buffer[]
Type: Array<0..2039>
An array of 2040 bits.
Default: na

input_buffer_location
Type: unsigned 16 bit
An integer that points to the next appending location in the input buffer.
Default: na

output_buffer[]
Type: Array<0..2039>
An array of 2040 bits.
Default: na

Add the following to counters to section 92.2.4.6.5 Counters
decode_failures
Type: 2 bit counter
Counter that holds the number of consecutive decoding failures.
Default: binary 00

Add the following functions to section **92.2.4.6.3 Functions**
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()
  {
    BlockFromSynchronizer()

    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.

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()
  }
```

BlockFromSyncronizer

Function that accepts the next rx_coded<0..65> block of data from the synchronizer. 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 figure 3av_0803_effenberger_1.pdf to section 92.2.4.6.6 State Diagrams