I Wrote an FFV1 Decoder in Go for Fun:

What I Learned Going from Spec to Implementation

Derek Buitenhuis

@daemon404

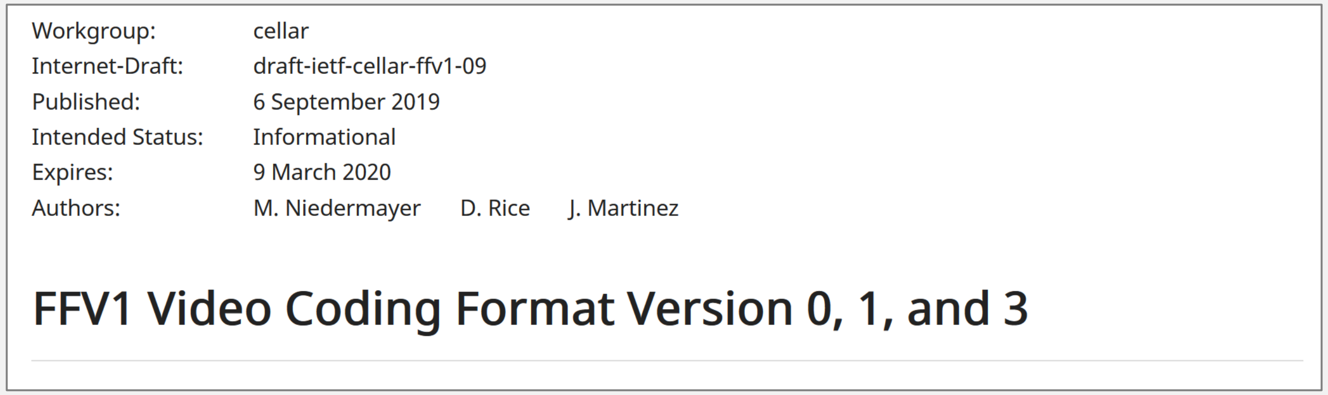
derek@videolan.org

NTTW4

5 December 2019
Budapest, Hungary

But… why?

- Because why not?
  - It seemed fun.
- I wanted to see how hard it was to write something while trying not to use anything but a spec.
  - Why? How do *you* spend your weekends? This is normal.
- But mostly: To make the spec better and aid in adoption of free and open codecs and standards.
  - More implementations means a better spec.
  - Doing it while trying not to reference anything else would expose anything missing or confusing.



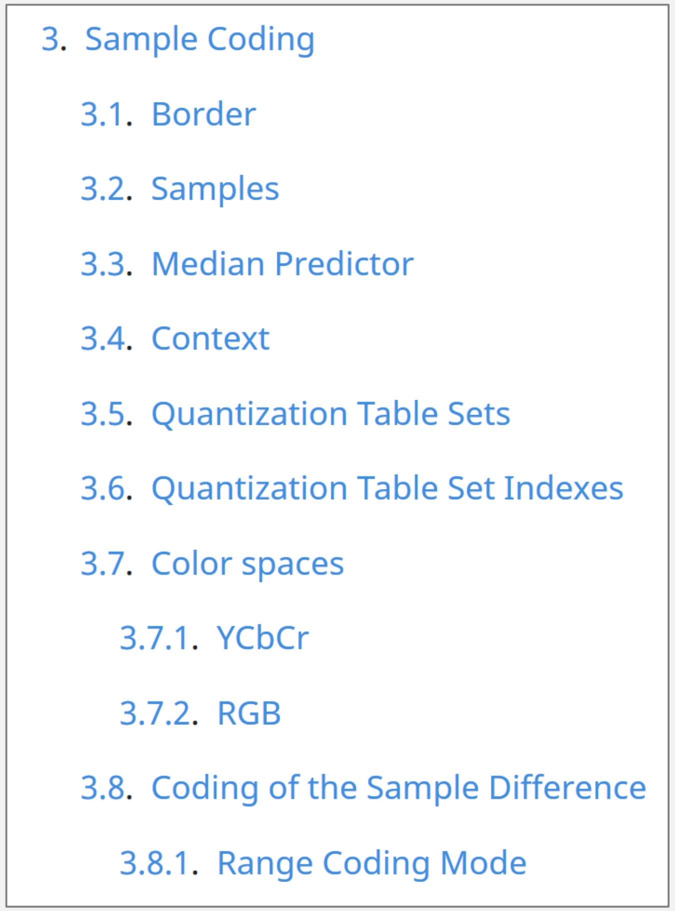
Workgroup:	cellar
Internet-Draft:	draft-ietf-cellar-ffv1-09
Published:	6 September 2019
Intended Status:	Informational
Expires:	9 March 2020
Authors:	M. Niedermayer D. Rice J. Martinez

FFV1 Video Coding Format Version 0, 1, and 3

NTTW4

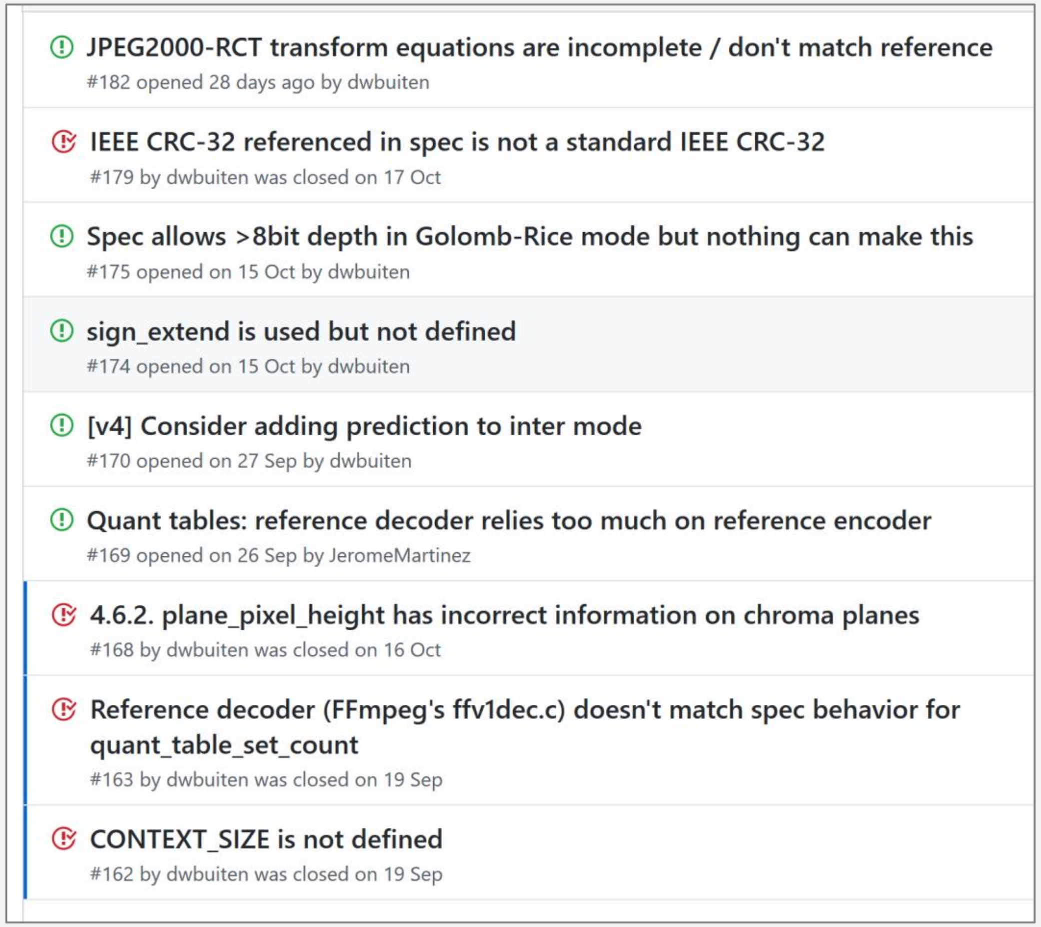
You’ll end up reading sections out-of-order and multiple times

- Specs are not written in the order you actually have to perform the steps in, or the order you’ll write them in.
- A second monitor really helps with all the back-and-forth referencing between your code and the spec.
- Pseudo-code in one section will use variables defined in other sections / scopes.



Third party implementation is important!

- There can be many things missing or unclear in a spec until at least one non-author implements it.
    - Things that may be obvious or clear to the author may not be to everyone else.
- The end result is a much more robust, tested spec.



JPEG2000-RCT transform equations are incomplete / don't match reference
#182 opened 28 days ago by dwbuiten

IEEE CRC-32 referenced in spec is not a standard IEEE CRC-32
#179 by dwbuiten was closed on 17 Oct

Spec allows >8bit depth in Golomb-Rice mode but nothing can make this
#175 opened on 15 Oct by dwbuiten

sign_extend is used but not defined
#174 opened on 15 Oct by dwbuiten

[v4] Consider adding prediction to inter mode
#170 opened on 27 Sep by dwbuiten

Quant tables: reference decoder relies too much on reference encoder
#169 opened on 26 Sep by JeromeMartinez

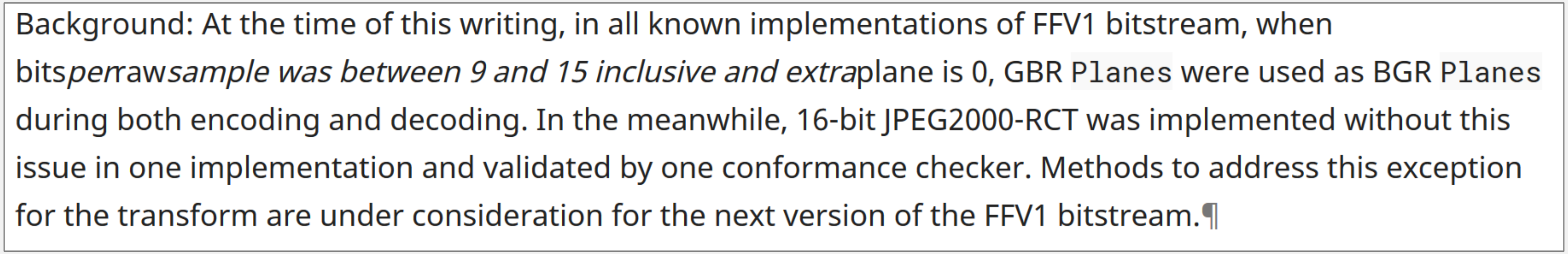
4.6.2. plane_pixel_height has incorrect information on chroma planes
#168 by dwbuiten was closed on 16 Oct

Reference decoder (FFmpeg's ffv1dec.c) doesn't match spec behavior for quant_table_set_count
#163 by dwbuiten was closed on 19 Sep

CONTEXT_SIZE is not defined
#162 by dwbuiten was closed on 19 Sep

Having background and context in the spec is extremely useful.

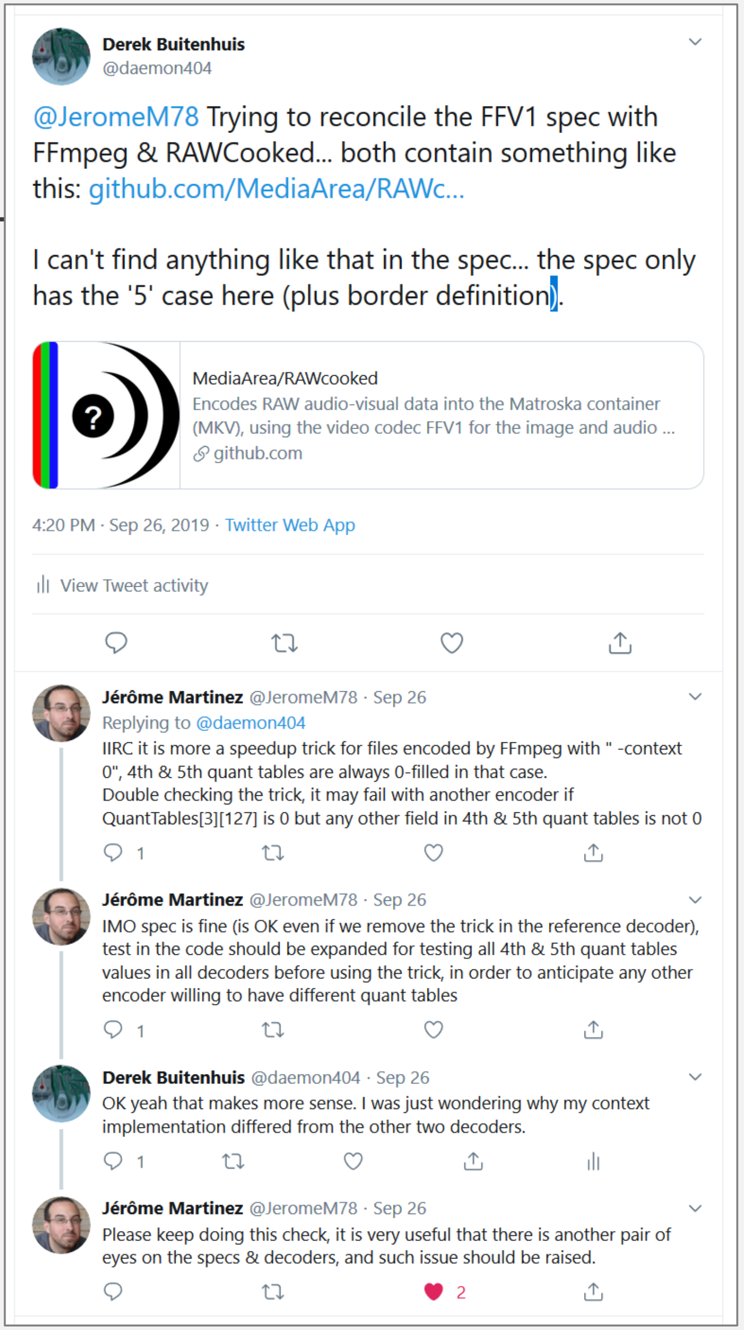
- Not many ‘professional’ specs have context or background in them, so why something is the way it is, or important implementation consequences may not be obvious.
- The FFV1 spec has lots of background, and it came in very useful.



Background: At the time of this writing, in all known implementations of FFV1 bitstream, when bits*per*raw*sample was between 9 and 15 inclusive and extra*plane is 0, GBR `Planes` were used as BGR `Planes` during both encoding and decoding. In the meanwhile, 16-bit JPEG2000-RCT was implemented without this issue in one implementation and validated by one conformance checker. Methods to address this exception for the transform are under consideration for the next version of the FFV1 bitstream.¶

Having contact with the spec author(s) is extremely useful.

- It's good to confirm you're not going crazy trying to figure something out.
- Lean on the knowledge of those who came before you.



Derek Buitenhuis @daemon404

@JeromeM78 Trying to reconcile the FFV1 spec with FFmpeg & RAWCooked... both contain something like this: github.com/MediaArea/RAWc...

I can't find anything like that in the spec... the spec only has the '5' case here (plus border definition).

MediaArea/RAWcooked
Encodes RAW audio-visual data into the Matroska container (MKV), using the video codec FFV1 for the image and audio ...
github.com

4:20 PM · Sep 26, 2019 · Twitter Web App

View Tweet activity

Jérôme Martinez @JeromeM78 · Sep 26
Replying to @daemon404
IIRC it is more a speedup trick for files encoded by FFmpeg with " -context 0", 4th & 5th quant tables are always 0-filled in that case.
Double checking the trick, it may fail with another encoder if QuantTables[3][127] is 0 but any other field in 4th & 5th quant tables is not 0
1

Jérôme Martinez @JeromeM78 · Sep 26
IMO spec is fine (is OK even if we remove the trick in the reference decoder), test in the code should be expanded for testing all 4th & 5th quant tables values in all decoders before using the trick, in order to anticipate any other encoder willing to have different quant tables
1

Derek Buitenhuis @daemon404 · Sep 26
OK yeah that makes more sense. I was just wondering why my context implementation differed from the other two decoders.
1

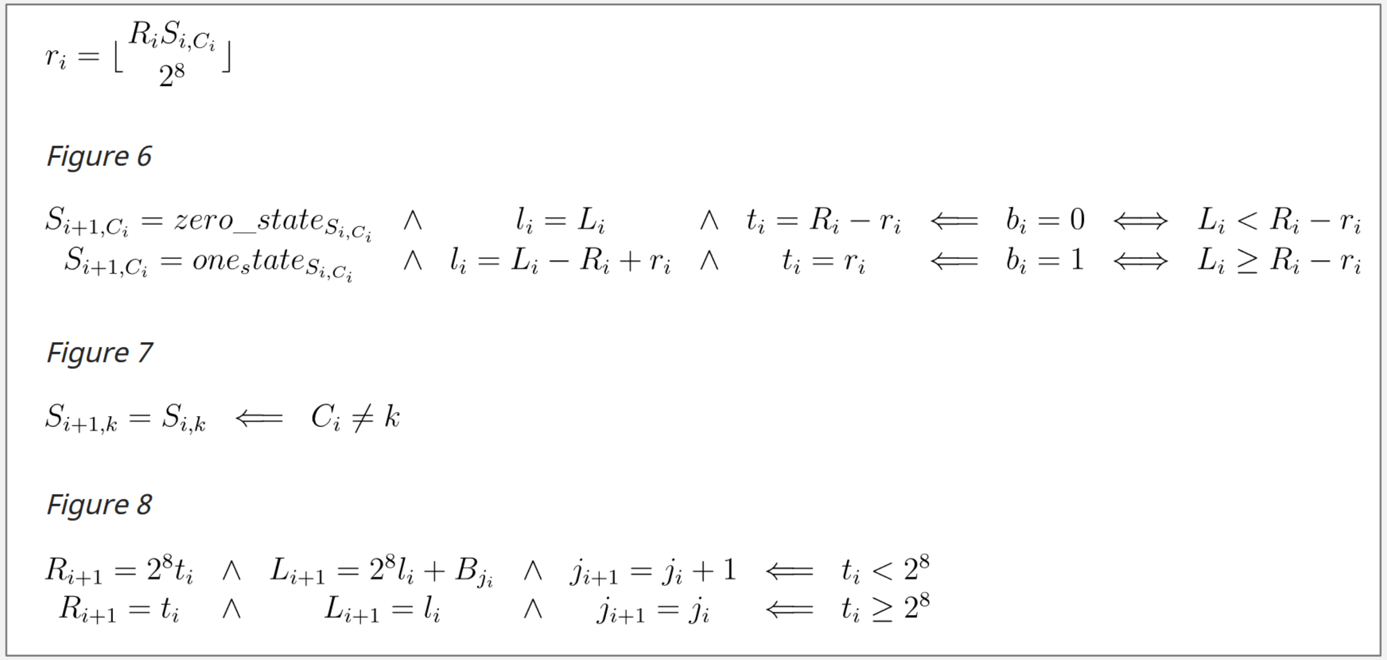
Jérôme Martinez @JeromeM78 · Sep 26
Please keep doing this check, it is very useful that there is another pair of eyes on the specs & decoders, and such issue should be raised.
2

Writing a spec based on an existing codebase makes for “interesting” specs

- Lots of the spec bugs I found were assumptions based on FFmpeg’s encoder or decoder behavior.
- Annoying implications like requiring 17-bit buffers for predicting 16-bit RGB, which are not spelled out in the spec at all. I hit this right as I finished up.
- On the flip side, the spec also had advice for how to multithread based off of the slice footers, and other real-world tips.

A bunch of LaTeX math and a paper reference alone aren’t as good as pseudocode

- Looking at you range coder.
  - Paper was OK, but implementation details were annoying.
  - Not gonna lie, I based my implementation off of Wikipedia + FFV1 spec constants.



$$r_i = \lfloor \frac{R_i S_{i,C_i}}{2^8} \rfloor$$

Figure 6

$$\begin{array}{ccccccccc} S_{i+1,C_i} = zero\_state_{S_{i,C_i}} & \wedge & l_i = L_i & \wedge & t_i = R_i - r_i & \Longleftarrow & b_i = 0 & \Longleftrightarrow & L_i < R_i - r_i \\ S_{i+1,C_i} = one_s tate_{S_{i,C_i}} & \wedge & l_i = L_i - R_i + r_i & \wedge & t_i = r_i & \Longleftarrow & b_i = 1 & \Longleftrightarrow & L_i \geq R_i - r_i \end{array}$$

Figure 7

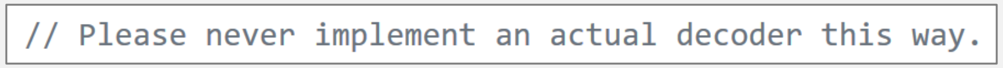
$$S_{i+1,k} = S_{i,k} \quad \Longleftarrow \quad C_i \neq k$$

Figure 8

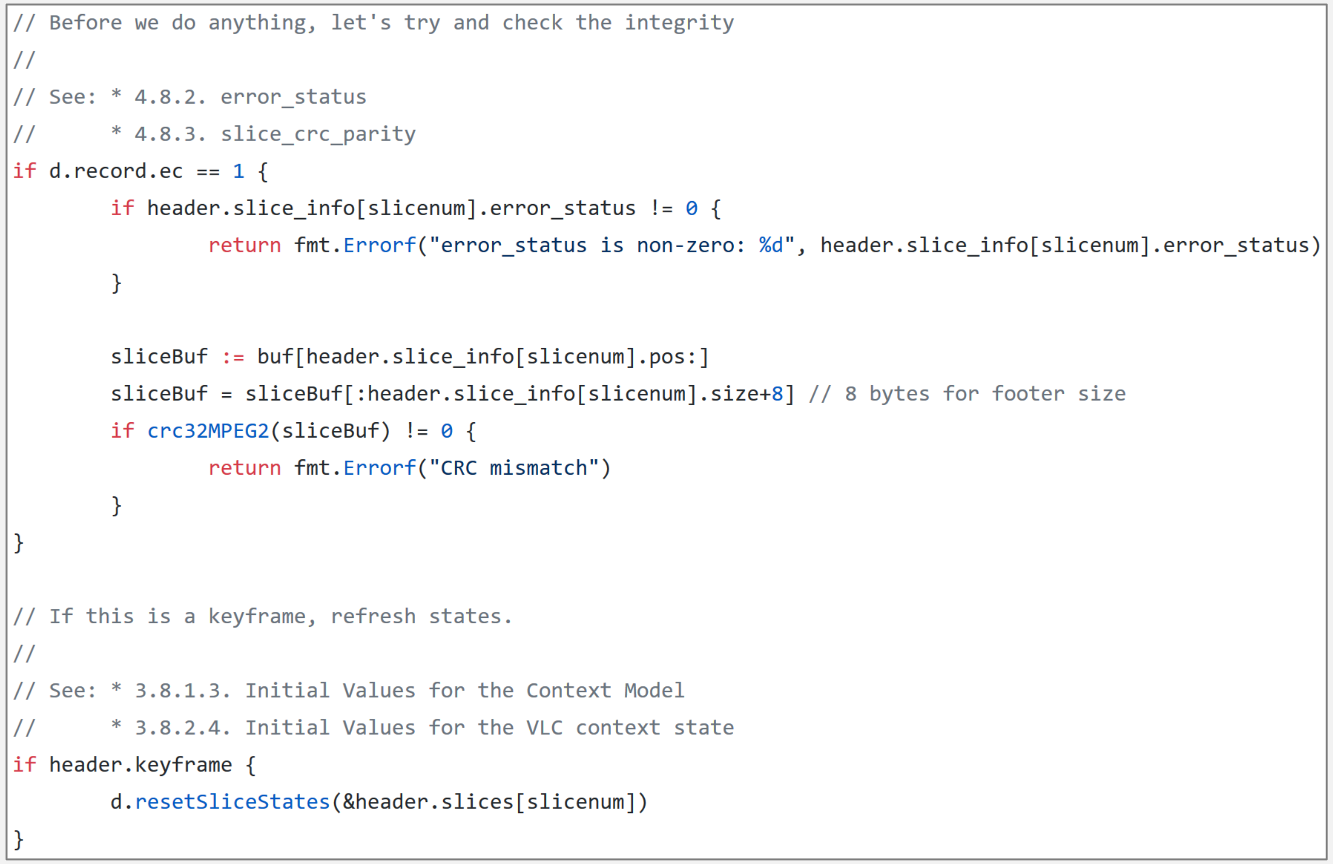
$$\begin{array}{ccccccc} R_{i+1} = 2^8 t_i & \wedge & L_{i+1} = 2^8 l_i + B_{j_i} & \wedge & j_{i+1} = j_i + 1 & \Longleftarrow & t_i < 2^8 \\ R_{i+1} = t_i & \wedge & L_{i+1} = l_i & \wedge & j_{i+1} = j_i & \Longleftarrow & t_i \geq 2^8 \end{array}$$

Some Quick Notes on the Code

- Not exactly idiomatic Go, in order to remain as close as possible to the spec.
    - Goal is be a good reference.
- Everything is annotated with references to spec sections.
- I used a bit of Perl for code generation…



```
// Please never implement an actual decoder this way.
```



```go
// Before we do anything, let's try and check the integrity
//
// See: * 4.8.2. error_status
//      * 4.8.3. slice_crc_parity
if d.record.ec == 1 {
	if header.slice_info[slicenum].error_status != 0 {
		return fmt.Errorf("error_status is non-zero: %d", header.slice_info[slicenum].error_status)
	}

	sliceBuf := buf[header.slice_info[slicenum].pos:]
	sliceBuf = sliceBuf[:header.slice_info[slicenum].size+8] // 8 bytes for footer size
	if crc32MPEG2(sliceBuf) != 0 {
		return fmt.Errorf("CRC mismatch")
	}
}

// If this is a keyframe, refresh states.
//
// See: * 3.8.1.3. Initial Values for the Context Model
//      * 3.8.2.4. Initial Values for the VLC context state
if header.keyframe {
	d.resetSliceStates(&header.slices[slicenum])
}
```

Links

- FFV1 Go Implementation: https://github.com/dwbuiten/go-ffv1
  - godoc: https://godoc.org/github.com/dwbuiten/go-ffv1/ffv1
- Simple Matroska Go Package: https://github.com/dwbuiten/matroska
  - godoc: https://godoc.org/github.com/dwbuiten/matroska
- FFV1 Spec: https://tools.ietf.org/id/draft-ietf-cellar-ffv1-10.html
  - FFV1 Repo: https://github.com/FFmpeg/FFV1/

Questions?