7 Unpacking the Frame

We have seen how to generate PCM samples from blocks of subband samples that are packaged inside frames in an MPEG audio stream, but we still need to see the details. This chapter will explain most of them.

Each frame has four parts, the header, optional a CRC field for error checking, then the audio data, which is split into side information and subband samples, and finally there might be ancillary data.

Let's examine these parts one after the other.

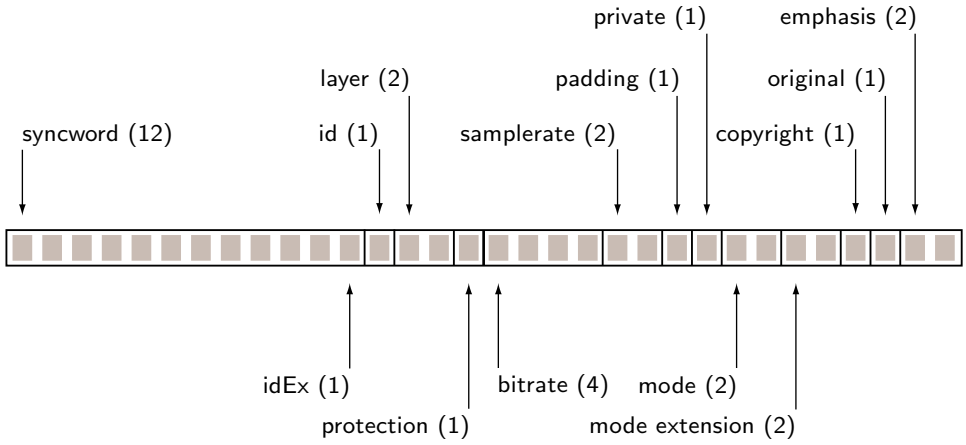

Fig. 23: The header layout

7.1 The Header

The first four byte of a frame are the header.

⟨ private declarations 1 ⟩ +≡ (77)

#**define** HEADER_SIZE 4

It serves two purposes: synchronization, that is finding the beginning of a frame—we deal with that later—, and providing information on the stream. The function

decode_header will extract this information into *info* and return 1. If *frame* does not point to a valid header, it will return 0.

⟨ auxiliary functions 69 ⟩ +≡ (78)

```
int decode_header(mp3_info *info, unsigned char *frame)
{ unsigned int header;
  header = (((((frame[0] ≪ 8) | frame[1]) ≪ 8) | frame[2]) ≪ 8) | frame[3];
  info→header = header;
  ⟨ decode the header 79 ⟩
  ⟨ determine frame_size 90 ⟩
  return 1;
}
```

According to the MPEG standard, the first 12 bit of the header are 1111 1111 1111. They are called the "syncword" and help us to *synchronize* the stream. Later versions (see chapter 14) use, however, only 11 bit for the syncword and set the 12^{th} bit to 0 to distinguish an extended bit stream from a standard version 1 bit stream.

Once we found the syncword, we do not need to keep it.

⟨ decode the header 79 ⟩ ≡ (79)

```
{ int n = 11; ⟨ get the next n header bit 80 ⟩
  if (bit ≠ #7FF) return 0;
}
```

Used in 78.

We read n bit from the *header* by shifting it to the right until only n bit remain and assign the result to *bit*. Then we discard these n bit from the header by shifting it left.

⟨ get the next n header bit 80 ⟩ ≡ (80)

```
int bit = header ≫ (32 − n);
header = header ≪ n;
```

Used in 79, 81, 82, 83, 85, 87, 88, 92, 94, 95, and 98.

We use the 12^{th} bit of the header, also called the ID extension bit, and the next bit of the header, the ID bit, to determine the MPEG *version*.

Constant	ID ext.	ID	Version
MP3_V2_5	0	0	MPEG Version 2.5
MP3_V2_0	1	0	MPEG Version 2 (ISO/IEC 13818-3)
MP3_V1_0	1	1	MPEG Version 1 (ISO/IEC 11172-3)
	0	1	reserved

Tab. 24: Version

⟨ decode the header 79 ⟩ +≡ (81)

```
{ int n = 2; ⟨ get the next n header bit 80 ⟩
  if (bit ≡ 0) info→version = MP3_V2_5;
  else if (bit ≡ 2) info→version = MP3_V2_0;
```