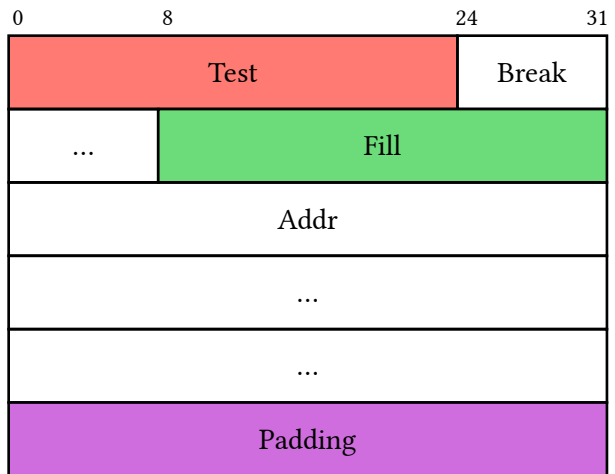# Bytefield

Colored Example
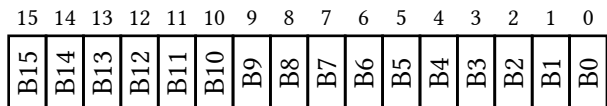
```
1  #bytefield(
2    bytes(3,
3      fill: red.lighten(30%)
4    )[Test],
5    bytes(2)[Break],
6    bits(24,
7      fill: green.lighten(30%)
8    )[Fill],
9    bytes(12)[Addr],
10   padding(
11     fill: purple.lighten(40%)
12   )[Padding],
13 )
```


Show all bits in the bitheader

Show all bit headers with `bitheader: "all"`
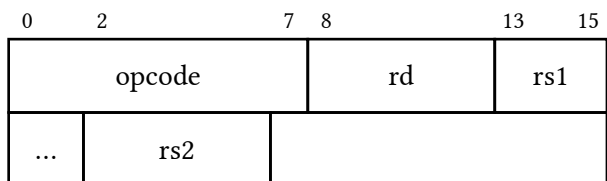
```
1  #bytefield(
2    bits:16,
3    msb_first: true,
4    bitheader: "all",
5    ..range(16).map(
6      i => bit[#flagtext[B#i]]
7    ).rev(),
8  )
```


Smart bit header

Show start and end bit of each bitbox with `bitheader: "smart"`.

```
1  #bytefield(
2    bits: 16,
3    // same as
4    // bitheader: (0,2,7,8,13,15),
5    bitheader: "smart",
6    bits(8)[opcode],
7    bits(5)[rd],
8    bits(5)[rs1],
9    bits(5)[rs2],
10   padding()[]
11 )
```
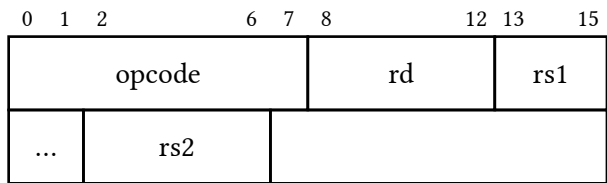

Bounds bit header

Show start bit of each bitbox with `bitheader: "bounds"`.

```
1  #bytefield(
2    bits: 16,
3    bitheader: "bounds",
4    bits(8)[opcode],
5    bits(5)[rd],
6    bits(5)[rs1],
7    bits(5)[rs2],
8    padding()[]
9  )
```
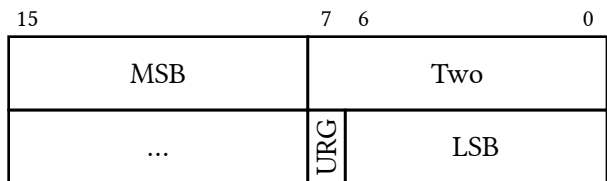
0	1	2				6	7	8				12	13		15
		opcode						rd					rs1		
...		rs2													

Reversed bit order

Select msb_first: true for a reversed bit order.

```
1  #bytefield(
2    bits: 16,
3    msb_first: true,
4    bitheader: "smart",
5    byte[MSB],
6    bytes(2)[Two],
7    bit[#flagtext("URG")],
8    bits(7)[LSB],
9  )
```
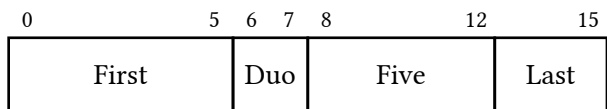
15			7	6			0
MSB				Two			
...			URG	LSB			

Custom bit header

Pass an array to specify each number.

```
1  #bytefield(
2    bits:16,
3    bitheader: (0,5,6,7,8,12,15),
4    bits(6)[First],
5    bits(2)[Duo],
6    bits(5)[Five],
7    bits(3)[Last],
8  )
```

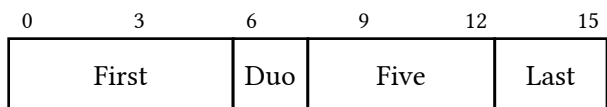| 0					5	6	7	8				12			15
First						Duo		Five					Last		

Pass an integer to show all multiples of this number.

```
1  #bytefield(
2    bits:16,
3    bitheader: 3,
4    bits(6)[First],
5    bits(2)[Duo],
6    bits(5)[Five],
7    bits(3)[Last],
8  )
```
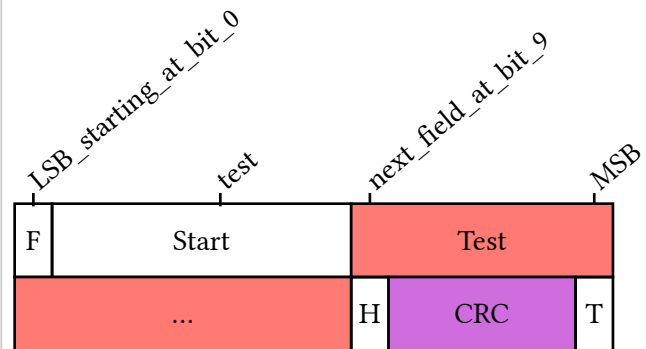
0			3			6			9			12			15
First						Duo		Five					Last		

Text header instead of numbers [WIP]

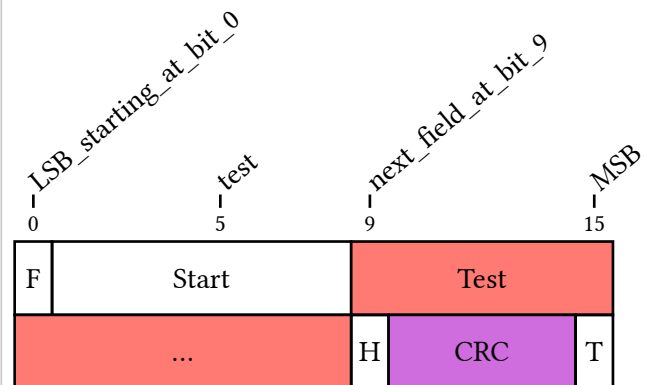Pass an `dictionary` as bitheader. Example:

```
1  #bytefield(
2    bitheader: (
3      "0": "LSB_starting_at_bit_0",
4      "5": "test",
5      "9": "next_field_at_bit_9",
6      "15":"MSB",
7      angle: -40deg,
8      marker: auto // or none
9    ),
10   bits: 16,
11   bit[F],
12   byte[Start],
13   bytes(2,
14     fill: red.lighten(30%)
15   )[Test],
16   bit[H],
17   bits(5,
18     fill: purple.lighten(40%)
19   )[CRC],
20   bit[T],
21 )
```


Text header and numbers [WIP]

You can also show labels and indexes by specifying `numbers`. `numbers` accepts the same string arguments as `bitheader`. You may also specify an array of indexes to show or simply `true` to show the index for each specified label.

```
1  #bytefield(
2    bitheader: (
3      "0": "LSB_starting_at_bit_0",
4      "5": "test",
5      "9": "next_field_at_bit_9",
6      "15":"MSB",
7      numbers: true,
8      angle: -40deg,
9      marker: auto // or none
10   ),
11   bits: 16,
12   bit[F],
13   byte[Start],
14   bytes(2,
15     fill: red.lighten(30%)
16   )[Test],
17   bit[H],
18   bits(5,
19     fill: purple.lighten(40%)
20   )[CRC],
21   bit[T],
22 )
```

```
1   #bytefield(
2     bitheader: (
3       "0": "LSB_starting_at_bit_0",
4       "5": "test",
5       "9": "next_field_at_bit_9",
6       "15":"MSB",
7       numbers: "bounds",
8       angle: -40deg,
9       marker: auto // or none
10    ),
11    bits: 16,
12    bit[F],
13    byte[Start],
14    bytes(2,
15      fill: red.lighten(30%)
16    )[Test],
17    bit[H],
18    bits(5,
19      fill: purple.lighten(40%)
20    )[CRC],
21    bit[T],
22  )
```
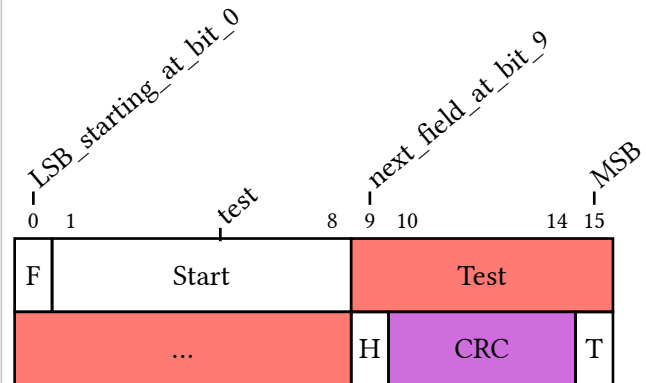

Annotations

Define annotations in columns left or right of the bitfields current row with the helpers `note` and `group`.

The needed number of columns is determined automatically, but can be forced with the `pre` and `post` arguments.

The helper `note` takes the side it should appear on as first argument, an optional `rowspan` for the number of rows it should span and an optional `level` for the nesting level.

The helper `group` takes the side it should appear on as first argument, as second argument `rowspan` for the number of rows it should span and an optional `level` for the nesting level.

```
1   #bytefield(
2     bits:32,
3
4     note(left)[0x00],
5     group(right,2)[group],
6     bytes(4)[some thing],
7
8     note(left)[0x04],
9     bytes(4)[some other thing],
10  )
```
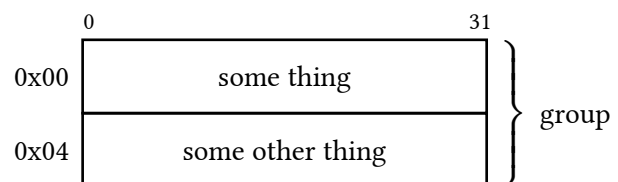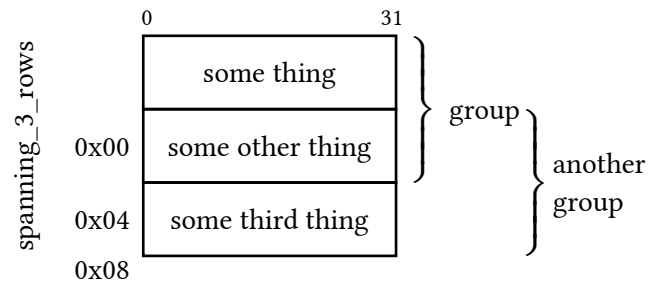
```
1  #bytefield(
2    bits:32,
3    pre: (1cm,auto),
4    post: (auto,1cm),
5
6    note(left, rowspan:3, level:1)[
7      #flagtext[spanning_3_rows]
8    ],
9    note(left)[0x00],
10   group(right,2)[group],
11   bytes(4)[some thing],
12
13   note(left)[0x04],
14   group(right,2,level:1)[another
   group],
15   bytes(4)[some other thing],
16   note(left)[0x08],
17   bytes(4)[some third thing],
18 )
```


Some predefined network protocols
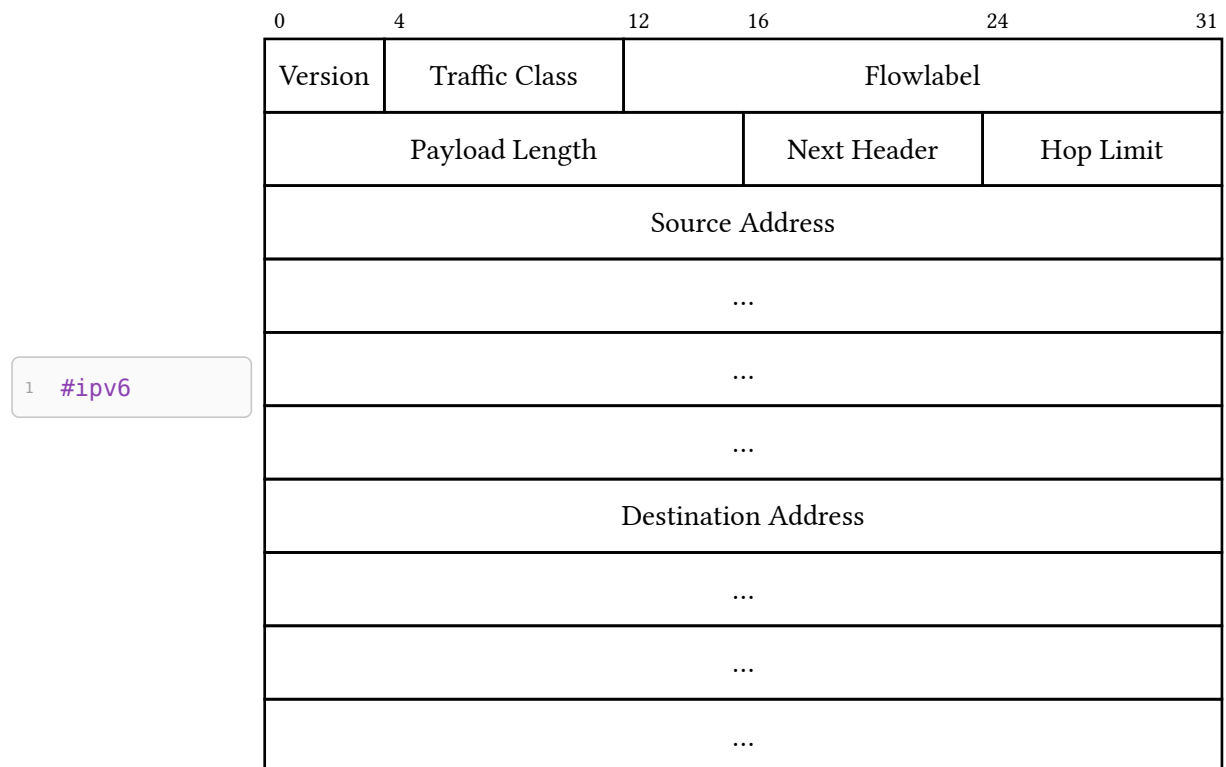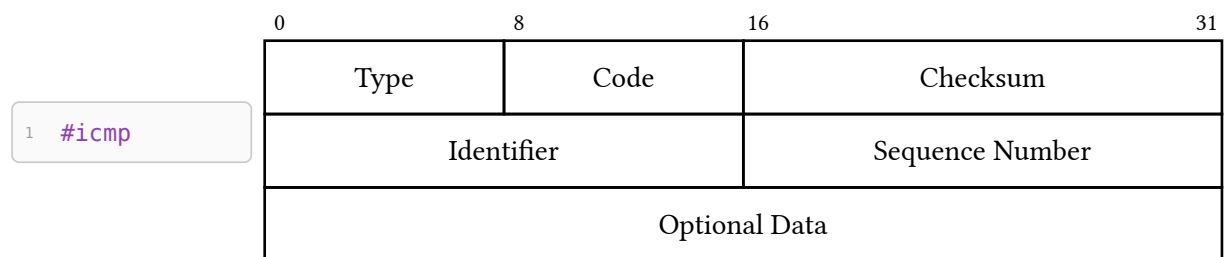
IPv4


```
1  #ipv4
```

IPv6

0	4	12	16	24	31

Version	Traffic Class	Flowlabel			
Payload Length			Next Header	Hop Limit	
Source Address					
...					
...					
...					
Destination Address					
...					
...					
...					

`1  #ipv6`

ICMP

0	8	16	31
Type	Code	Checksum	
Identifier		Sequence Number	
Optional Data			

`1  #icmp`

ICMPv6

0	8	16	31
Type	Code	Checksum	
Internet Header + 64 bits of Original Data Datagram			

`1  #icmpv6`
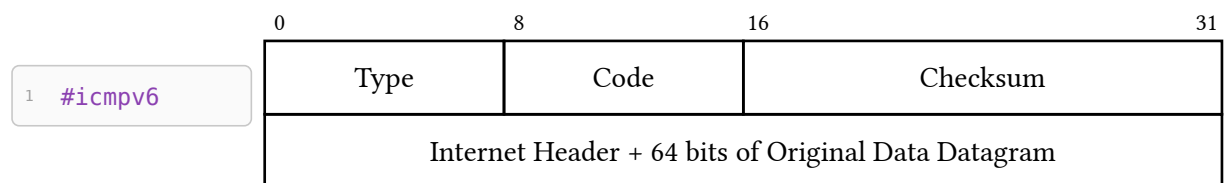
DNS

0	16	31
Identification		Flags
Number of Questions		Number of answer RRs
Number of authority RRs		Number of additional RRs
Questions		
...		
Answers (variable number of resource records)		
...		
Authority (variable number of resource records)		
...		
Additional information (variable number of resource records)		
...		

```
1   #dns
```

TCP

| 0 | 4 | 10 | 16 | 24 | 31 |

1 #tcp

Source Port	Destinatino Port
Sequence Number	
Acknowledgment Number	

Data Offset	Reserved	Flags	Window
Checksum	Urgent Pointer
Options	Padding
...DATA...	

| 0 | 4 | 10 11 12 13 14 15 16 | 24 | 31 |

1 #tcp_detailed

Source Port	Destinatino Port
Sequence Number	
Acknowledgment Number	

Data Offset	Reserved	URG ACK PSH RST SYN FIN	Window
Checksum	Urgent Pointer
Options	Padding
...DATA...	

UDP

| 0 | 16 | 31 |

1 #udp

Source Port	Destinatino Port
Length	Checksum
...DATA...	