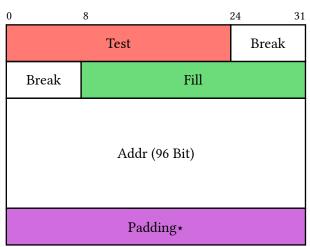
Bytefield

Colored Example

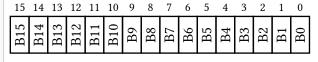
```
#bytefield(
     bytes (3,
2
       fill: red.lighten(30%)
3
     )[Test],
4
5
     bytes(2)[Break],
6
     bits(24,
7
       fill: green.lighten(30%)
8
     )[Fill],
9
     bytes(12)[Addr],
     padding(
10
       fill: purple.lighten(40%)
11
12
     )[Padding],
13
```



Show all bits in the bitheader

Show all bit headers with bitheader: "all"

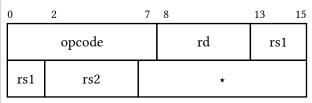
```
1  #bytefield(
2    bits:16,
3    msb_first: true,
4    bitheader: "all",
5    ...range(16).map(
6    i => bit[#flagtext[B#i]]
7    ).rev(),
8 )
```



Smart bit header

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

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



Bounds bit header

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

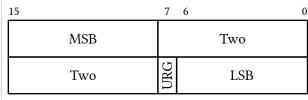
```
1  #bytefield(
2   bits: 16,
3   bitheader: "bounds",
4   bits(8)[opcode],
5   bits(5)[rd],
6   bits(5)[rs1],
7   bits(5)[rs2],
8   padding()[]
```

0 1	2	6	7	8		12	13	15
	opcode				rd		rs1	
rs1	rs2				*			

Reversed bit order

Select msb_first: true for a reversed bit order.

```
#bytefield(
bits: 16,
msb_first: true,
bitheader: "smart",
byte[MSB],
bytes(2)[Two],
bit[#flagtext("URG")],
bits(7)[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		Dι	uo		Five		Last

Pass an integer to show all multiples of this number.

```
#bytefield(
bits:16,
bitheader: 3,
bits(6)[First],
bits(2)[Duo],
bits(5)[Five],
bits(3)[Last],

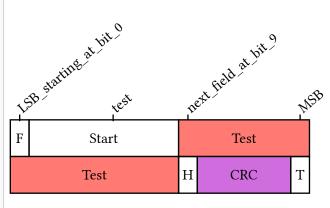
)
```

0	3	6	9	12	15
	First	Duo	Five		Last

Text header instead of numbers [WIP]

Pass an dictionary as bitheader. Example:

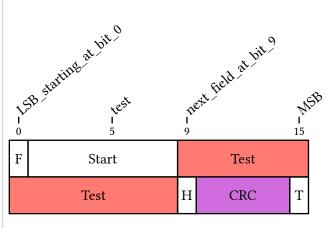
```
#bytefield(
     bitheader: (
2
        "0": "LSB_starting_at_bit_0",
3
        "5": "test",
4
        "9": "next_field_at_bit_9",
5
6
        "15": "MSB",
7
       angle: -40deg,
8
       marker: auto // or none
9
10
     bits: 16,
     bit[F],
11
12
     byte[Start],
13
     bytes(2,
        fill: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



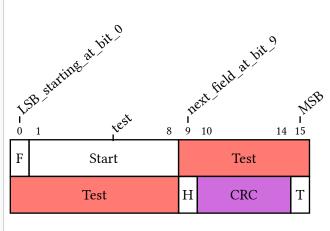
Text header and numbers [WIP]

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.

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



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

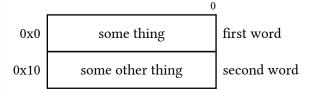


Pre/Post columns

Define additional columns with before the bitfield with pre or behind the bitfield with post and pass any tablex object.

You can use the helpers left_aligned and right_aligned for left and right aligned text.

```
#bytefield(
2
     bits:1,
3
     pre:(auto,),
     post:(auto,),
     right_aligned[0x0],
6
     bit[some thing],
     left_aligned[first word],
     right_aligned[0x10],
8
     bit[some other thing],
     left_aligned[second word],
10
11
```

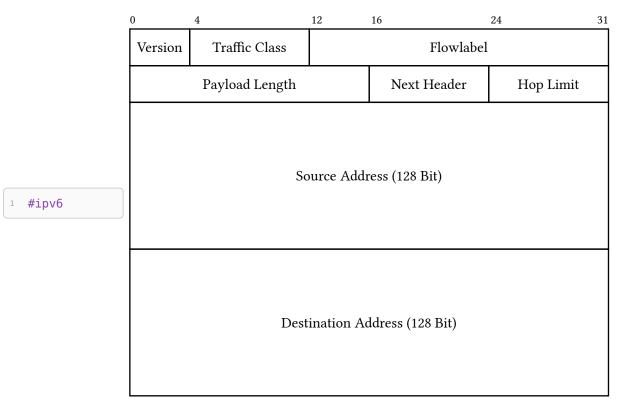


Some predefined network protocols

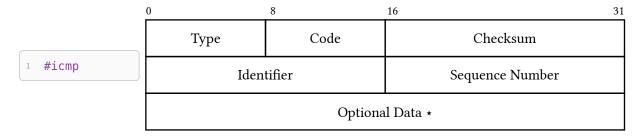
IPv4

	0	4	8	16	19	24 31	
	Version	TTL	TOS	Total I		Length	
		Identif	ìcation	Flags Fragment Offset			
1 #ipv4	TTL		Protocol	Header Checksum		Checksum	
	Source Address						
	Destination Address						
	Options	Padding					

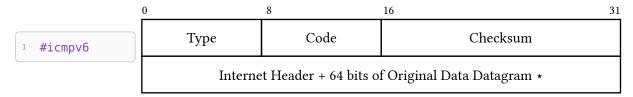
IPv6



ICMP



ICMPv6



DNS

1 #dns

0	16 31						
Identification	Flags						
Number of Questions	Number of answer RRs						
Number of authority RRs	Number of additional RRs						
Questions (64 Bit)							
Answers (variable number of resource records) (64 Bit)							
Authority (variable number of resource records) (64 Bit)							
Additional information (variable number of resource records) (64 Bit)							

1	0	4	10	16	24 31			
	Source Port			Destinatino Port				
1 #tcp	Data Offset Reserved Flags			Window				
		Checksun	1	Urgent Pointer				
			Options		Padding			
			DA	TA*				
	0	4	10 11 12 13 14 15	16	24 31			
		Source Por	rt	Destina	tino Port			
			Sequence	e Number				
			Acknowledg	ment Number				
<pre>#tcp_detailed</pre>	Data Offset	Reserved	URG ACK PSH RST SYN FIN	Wii	ıdow			
		Checksun	1	Urgent Pointer				
			Options		Padding			
		•						
UDP								
	0 16 31							
		Source Por	rt	Destina	tino Port			
¹ #udp		Length		Checksum				

...DATA...*