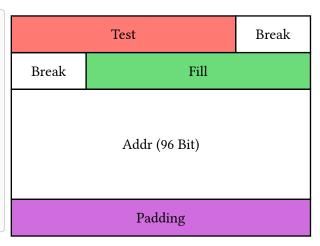
Bytefield

Colored Example

```
#bytefield(
     bitheader(),
3
     bytes(3,
       fill: red.lighten(30%)
5
     )[Test],
     bytes(2)[Break],
6
7
     bits(24,
       fill: green.lighten(30%)
8
9
     )[Fill],
10
     bytes(12)[Addr],
     padding(
       fill: purple.lighten(40%)
13
     )[Padding],
14
```



Show all bits in the bitheader

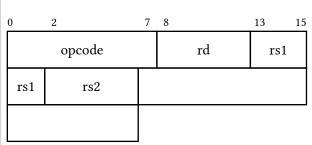
Show all bit headers with bitheader: "all"

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



Bounds bit header

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

```
#bytefield(
bits: 16,
bitheader("bounds"),
bits(8)[opcode],
bits(5)[rd],
bits(5)[rs1],
bits(5)[rs2],
padding()[]

9 )
```

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],
)
```

MSB	Two	
Two	LSB 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		Dı	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],

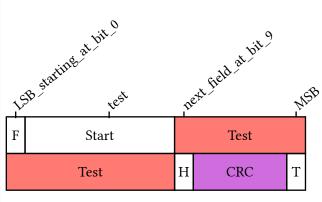
)
```

0	3	6	9	12	15
	First	Duo	Five		Last

Text header instead of numbers [WIP]

Pass an dictionary as bitheader. Example:

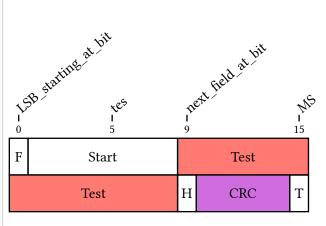
```
#bytefield(
     bits: 16,
2
3
     bitheader(
4
       0, [LSB_starting_at_bit_0],
5
       5, [test],
6
       9, [next_field_at_bit_9],
       15, [MSB],
8
       angle: -40deg,
9
       marker: auto // or none
10
11
     bit[F],
12
     byte[Start],
13
     bytes(2,
       fill: red.lighten(30%)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



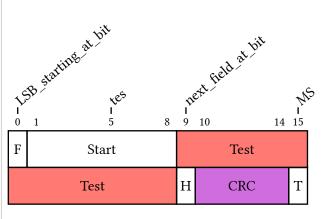
Text header and numbers [WIP]

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
     bits: 16,
3
     bitheader(
       0, [LSB_starting_at_bit],
4
5
       5, [tes],
       9, [next_field_at_bit],
6
7
       15, [MS],
8
        autofill: true,
9
       angle: -40deg,
10
       marker: auto // or none
11
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(
     bits: 16,
3
     bitheader(
       0, [LSB starting at bit],
4
       5, [tes],
       9, [next field at bit],
6
       15, [MS],
7
       autofill: "bounds",
8
9
       angle: -40deg,
10
       marker: auto // or none
     bit[F],
13
     byte[Start],
14
     bytes(2,
       fill: red.lighten(30%)
16
     )[Test],
17
     bit[H],
18
     bits(5,
       fill: purple.lighten(40%)
19
20
     )[CRC],
     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.

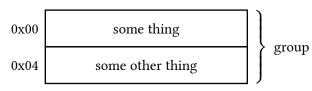
The helper group takes the side it should appear on as first argument, as second argument rowspan for the number of rows it should span and an optional level for the nesting level.

```
#bytefield(
bits:32,

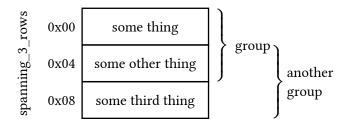
note(left)[0x00],
group(right,2)[group],
bytes(4)[some thing],

note(left)[0x04],
bytes(4)[some other thing],

bytes(4)[some other thing],
```

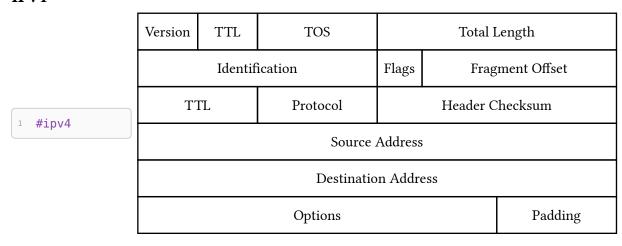


```
#bytefield(
     bits:32,
3
     pre: (1cm, auto),
     post: (auto,1cm),
6
     note(left, rowspan:3, level:1)[
7
      #flagtext[spanning_3_rows]
8
9
     note(left)[0x00],
10
     group(right,2)[group],
11
     bytes(4)[some thing],
13
     note(left)[0x04],
     group(right, 2, level: 1) [another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



Some predefined network protocols

IPv4



IPv6

	Version Traffic Class		Flowlabel				
	Payload Length			Next Header	Hop Limit		
1 #ipv6	Source Address (128 Bit)						
		Dest	ination Ac	ldress (128 Bit)			

ICMP

Type Code Checksum

Identifier Sequence Number

Optional Data

ICMPv6

Type Code Checksum

Internet Header + 64 bits of Original Data Datagram

DNS

	Identification	Flags			
	Number of Questions	Number of answer RRs			
	Number of authority RRs	Number of additional RRs			
	Questions (64 Bit)				
1 #dns	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 #udp

		Source Por	t	Destinatino Port			
	Sequence Number						
	Acknowledgment Number						
1 #tcp	Data Offset Reserved Flags Window						
		Checksum	Urgent :	Pointer			
				Padding			
	DATA						
		ino Port					
	Sequence Number						
	Acknowledgment Number						
1 #tcp_detailed	Data Offset Reserved Offset Reserved Reserved Reserved				Window		
		Pointer					
		Padding					
	DATA						
UDP							
	Source Port Destinatino Port						

Length

...DATA...

Checksum