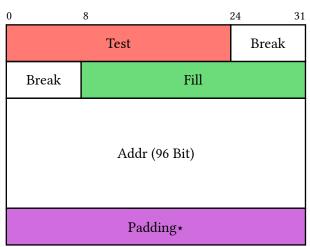
# **Bytefield**

## **Colored Example**

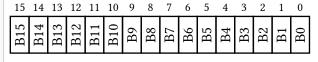
```
#bytefield(
     bytes (3,
2
       fill: red.lighten(30%)
3
     )[Test],
4
5
     bytes(2)[Break],
6
     bits(24,
7
       fill: green.lighten(30%)
8
     )[Fill],
9
     bytes(12)[Addr],
     padding(
10
       fill: purple.lighten(40%)
11
12
     )[Padding],
13
```



### Show all bits in the bitheader

Show all bit headers with bitheader: "all"

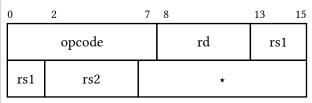
```
1  #bytefield(
2    bits:16,
3    msb_first: true,
4    bitheader: "all",
5    ...range(16).map(
6    i => bit[#flagtext[B#i]]
7    ).rev(),
8 )
```



#### 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".

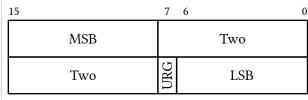
```
1  #bytefield(
2   bits: 16,
3   bitheader: "bounds",
4   bits(8)[opcode],
5   bits(5)[rd],
6   bits(5)[rs1],
7   bits(5)[rs2],
8   padding()[]
```

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	10		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],

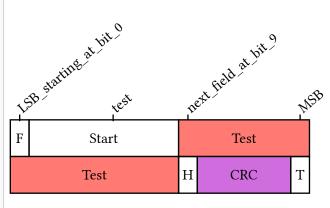
)
```

0	3	6	9	12	15
	First	Duo	Five		Last

### Text header instead of numbers [WIP]

Pass an dictionary as bitheader. Example:

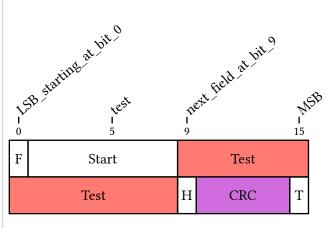
```
#bytefield(
     bitheader: (
2
        "0": "LSB_starting_at_bit_0",
3
        "5": "test",
4
        "9": "next_field_at_bit_9",
5
6
        "15": "MSB",
7
       angle: -40deg,
8
       marker: auto // or none
9
10
     bits: 16,
     bit[F],
11
12
     byte[Start],
13
     bytes(2,
        fill: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



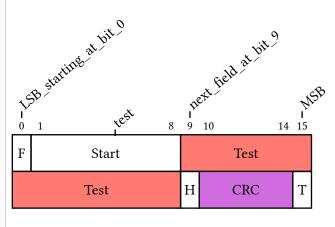
## Text header and numbers [WIP]

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: (
3
        "0": "LSB starting at bit 0",
       "5": "test",
4
       "9": "next_field_at_bit_9",
       "15": "MSB",
6
       numbers: "bounds",
7
8
       angle: -40deg,
9
       marker: auto // or none
10
     bits: 16,
     bit[F],
     byte[Start],
     bytes(2,
14
       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.

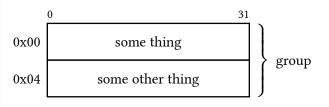
```
#bytefield(
bits:32,

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

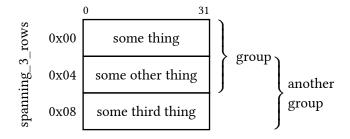
note(left)[0x04],
bytes(4)[some other thing],

bytes(4)[some other thing],

)
```

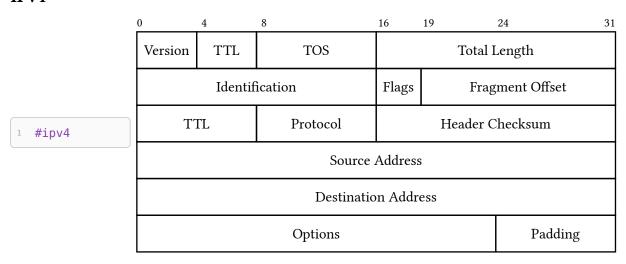


```
#bytefield(
     bits:32,
3
     pre: (1cm, auto),
     post: (auto,1cm),
6
     note(left, rowspan:3, level:1)[
7
       #flagtext[spanning_3_rows]
8
9
     note(left)[0x00],
10
     group(right,2)[group],
11
     bytes(4)[some thing],
13
     note(left)[0x04],
     group(right, 2, level: 1) [another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

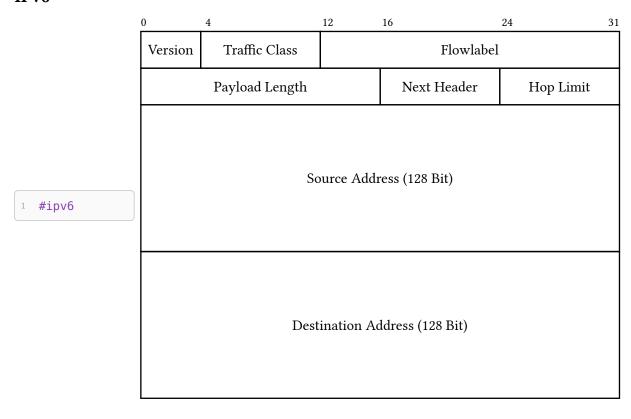


# Some predefined network protocols

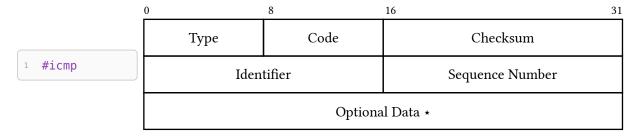
### IPv4



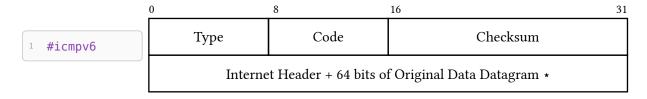
## IPv6



## **ICMP**



## ICMPv6



# DNS

	0	16 31				
	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)					

	0	4	10	16	24 31				
	Source Port			Destinat	ino Port				
	Sequence Number								
	Acknowledgment Number								
1 #tcp	Data Offset Reserved Flags Window				dow				
		Checksun	n	Urgent Pointer					
			Options	Padding					
			DA	ГА∗					
	0	4	24 31						
		Source Po	rt	Destinatino Port					
	Sequence Number								
	Acknowledgment Number								
1 #tcp_detailed	Data Offset	Reserved	URG ACK PSH RST SYN FIN	Win	Window				
		Checksun	n	Urgent	Pointer				
		Padding							
	DATA*								
UDP									
	0			16	31				
	Source Port Destinatino Port								
¹ #udp		Length		Checksum					
	DATA*								