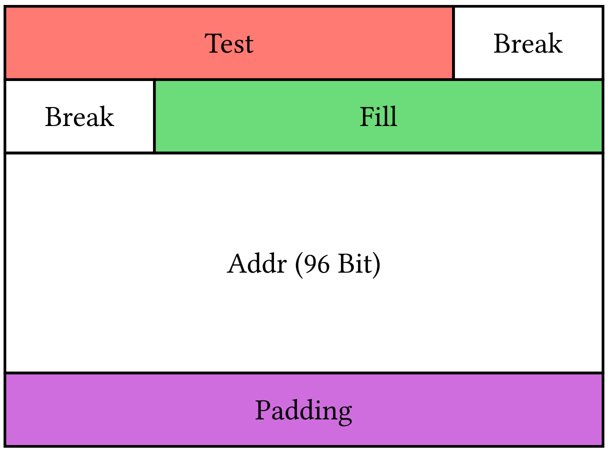
Bytefield

Colored Example

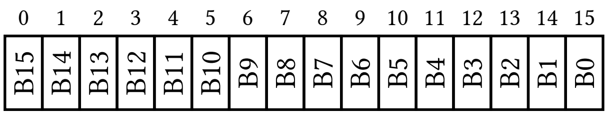
```
#bytefield(
  bitheader(),
  bytes(3,
    fill: red.lighten(30%)
  )[Test],
  bytes(2)[Break],
  bits(24,
    fill: green.lighten(30%)
  )[Fill],
  bytes(12)[Addr],
  padding(
    fill: purple.lighten(40%)
  )[Padding],
)
```



Show all bits in the bitheader

Show all bit headers with `bitheader: "all"`

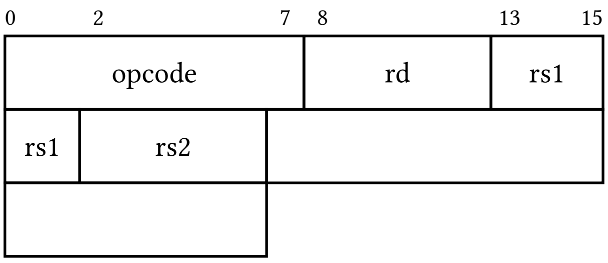
```
#bytefield(
    bits:16,
    msb_first: true,
    bitheader("all"),
    ..range(16).map(
      i => bit[#flagtext[B#i]]
    ).rev(),
)
```



Smart bit header

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

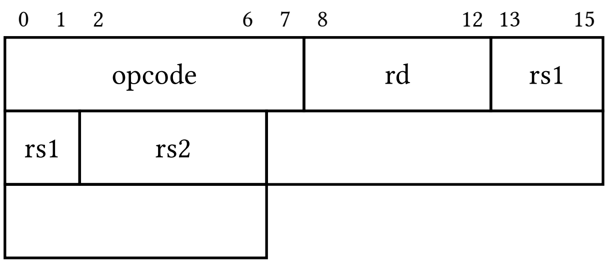
```
#bytefield(
  bits: 16,
  bitheader("smart"),
  // same as
  // bitheader(0,2,7,8,13,15),
  bits(8)[opcode],
  bits(5)[rd],
  bits(5)[rs1],
  bits(5)[rs2],
  padding()[]
)
```



Bounds bit header

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

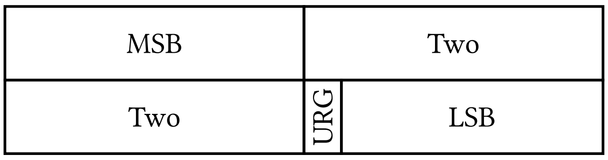
```
#bytefield(
  bits: 16,
  bitheader("bounds"),
  bits(8)[opcode],
  bits(5)[rd],
  bits(5)[rs1],
  bits(5)[rs2],
  padding()[]
)
```



Reversed bit order

Select `msb_first: true` for a reversed bit order.

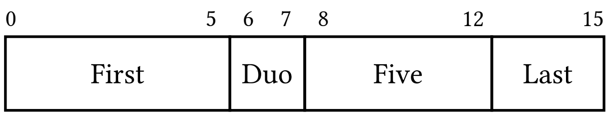
```
#bytefield(
    bits: 16,
    msb_first: true,
    bitheader: "smart",
    byte[MSB],
    bytes(2)[Two],
    bit[#flagtext("URG")],
    bits(7)[LSB],
)
```



Custom bit header

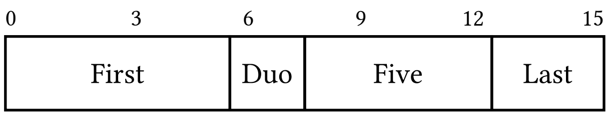
Pass an `array` to specify each number.

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



Pass an `integer` to show all multiples of this number.

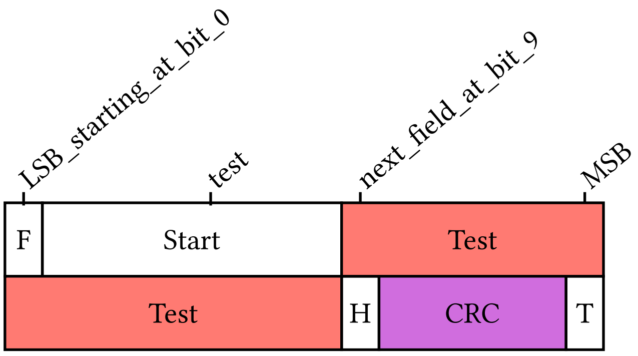
```
#bytefield(
    bits:16,
    bitheader(3),
    bits(6)[First],
    bits(2)[Duo],
    bits(5)[Five],
    bits(3)[Last],
)
```



Text header instead of numbers [WIP]

Pass an `dictionary` as bitheader. Example:

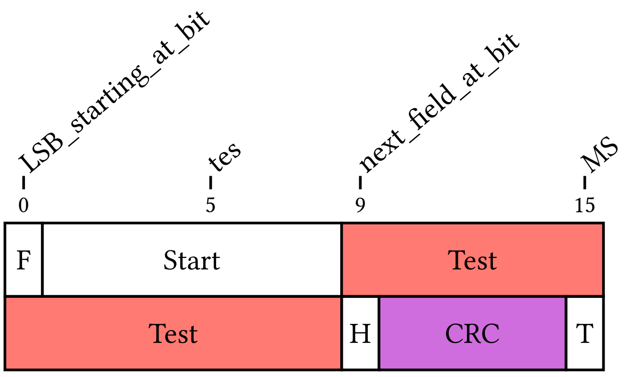
```
#bytefield(
  bits: 16,
  bitheader(
    0, [LSB_starting_at_bit_0],
    5, [test],
    9, [next_field_at_bit_9],
    15,[MSB],
    angle: -40deg,
    marker: auto // or none
  ),
  bit[F],
  byte[Start],
  bytes(2,
    fill: red.lighten(30%)
  )[Test],
  bit[H],
  bits(5,
    fill: purple.lighten(40%)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



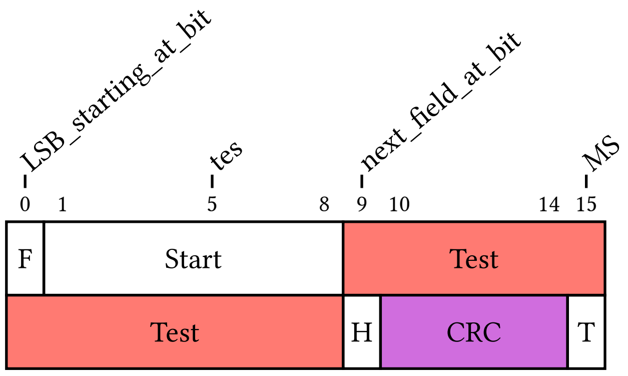
Text header and numbers [WIP]

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(
  bits: 16,
  bitheader(
    0, [LSB_starting_at_bit],
    5, [tes],
    9, [next_field_at_bit],
    15,[MS],
    autofill: true,
    angle: -40deg,
    marker: auto // or none
  ),
  bit[F],
  byte[Start],
  bytes(2,
    fill: red.lighten(30%)
  )[Test],
  bit[H],
  bits(5,
    fill: purple.lighten(40%)
  )[CRC],
  bit[T],
)
```



```
#bytefield(
  bits: 16,
  bitheader(
    0, [LSB_starting_at_bit],
    5, [tes],
    9, [next_field_at_bit],
    15,[MS],
    autofill: "bounds",
    angle: -40deg,
    marker: auto // or none
  ),
  bit[F],
  byte[Start],
  bytes(2,
    fill: red.lighten(30%)
  )[Test],
  bit[H],
  bits(5,
    fill: purple.lighten(40%)
  )[CRC],
  bit[T],
)
```



Annotations

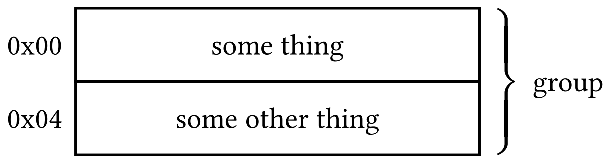
Define annotations in columns left or right of the bitfields current row with the helpers `note` and `group`.

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

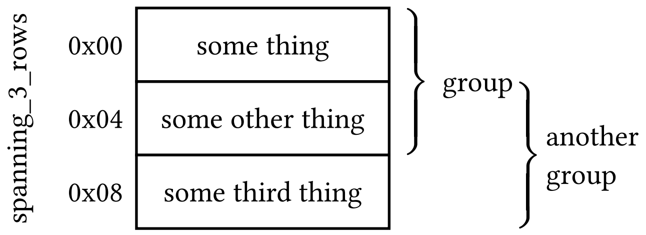
The helper `note` takes the side it should appear on as first argument, an optional `rowspan` for the number of rows it should span and an optional `level` for the nesting level.

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],
)
```



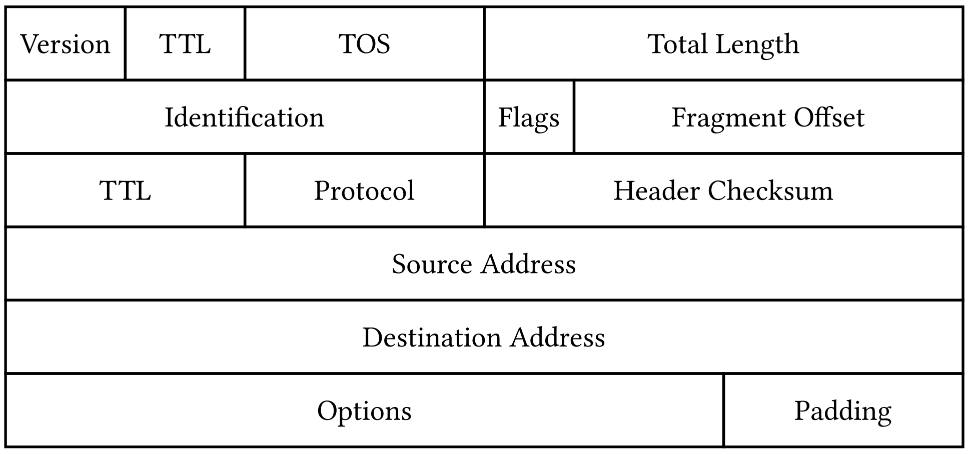
```
#bytefield(
  bits:32,
  pre: (1cm,auto),
  post: (auto,1cm),

  note(left, rowspan:3, level:1)[
    #flagtext[spanning_3_rows]
  ],
  note(left)[0x00],
  group(right,2)[group],
  bytes(4)[some thing],

  note(left)[0x04],
  group(right,2,level:1)[another group],
  bytes(4)[some other thing],
  note(left)[0x08],
  bytes(4)[some third thing],
)
```



Some predefined network protocols

IPv4

```
#ipv4
```



IPv6

```
#ipv6
```

Version	Traffic Class	Flowlabel
Payload Length	Next Header	Hop Limit
Source Address (128 Bit)		
Destination Address (128 Bit)		

ICMP

```
#icmp
```

Type	Code	Checksum
Identifier		Sequence Number
Optional Data		

ICMPv6

```
#icmpv6
```

Type	Code	Checksum
Internet Header + 64 bits of Original Data Datagram		

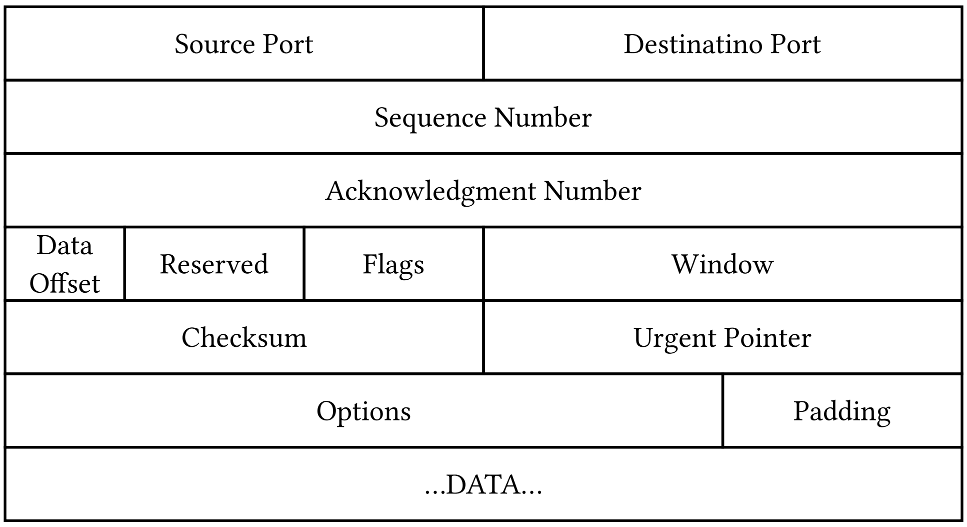
DNS

```
#dns
```

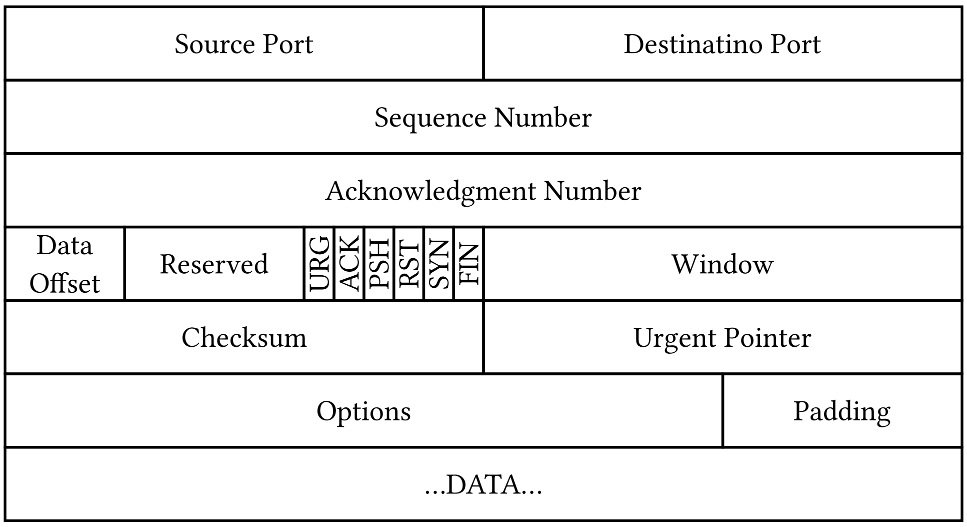
Identification	Flags
Number of Questions	Number of answer RRs
Number of authority RRs	Number of additional RRs
Questions (64 Bit)	
Answers (variable number of resource records) (64 Bit)	
Authority (variable number of resource records) (64 Bit)	
Additional information (variable number of resource records)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

TCP

```
#tcp
```

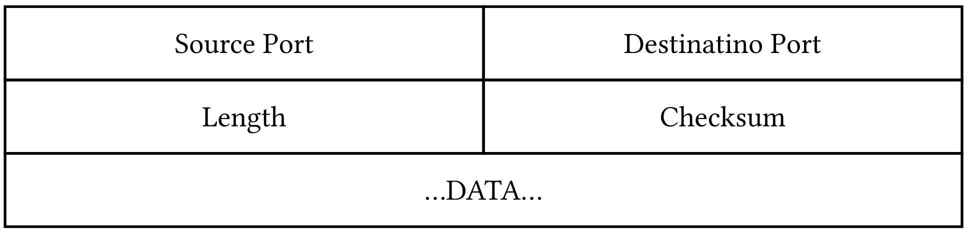


```
#tcp_detailed
```



UDP

```
#udp
```



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