Unions

- A union in 'C' is similar to a structure except that all of its members start at the same location in memory.
- A union variable can represent the value of only one of its members at a time.

structure memory allocation

only difference is writing union instead of struct.

```
struct address
{
    uint16_t    short Addr;
    uint32_t    long Addr;
};
```

```
Union address
{
    uint16_t    short Addr;
    uint32_t    long Addr;
};
```

Unions: use unions instead of structure to save memory when access to its member elements is mutually exclusive. (only for some) → 1. limited to only one person or group of people.

example for usage of unions: for example you have a packet which has long address either short address. When we have this type of situations. We don't need to clarify one of the variable and when we have this situation. We use unions.

unions.c

```c
#include <stdio.h>
#include <stdint.h>

union address
{
    uint16_t shortaddr;
    uint32_t longaddr;
};


int main(){

    union address addr;

    addr.shortaddr = 0xABCD;
    addr.longaddr = 0xEEEECCCC;

    printf("short adress: %X\n",addr.shortaddr);
    printf("long adress: %X\n",addr.longaddr);

    return(0);
}
```

```
PS C:\Users\islam\Desktop\Embedded-C\My_workspace\target\000CFolder> cd "c:\Users\islam\Desktop\Embedded-C\My_workspace\target\000CFolder\" ; if ($?) { gcc unions.c -o unions } ; if ($?) { .\unions }
short adress: CCCC
long adress: EEEECCCC
```

I will explain what is happening in here.

As I describe before our location for our two variables which created in unions block. Here is the good example how its respond.

firstly, it will look like this way. Then.

But after giving value for some address will change what exsited and it will show wrong.

Applicability of Unions in Embedded System Code

1. Bit extraction
2. Storing mutually exclusive data thus saving memory
   - "thus" = "in this way" / "with this result"

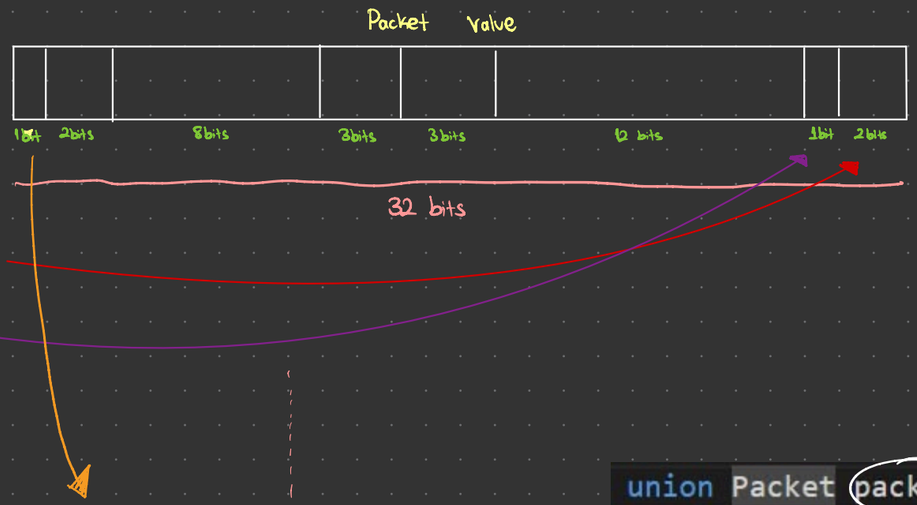
1. Bit extraction:

This usage is so meaningful and life saver. As I explained on above our union uses the same memory location for two different variables. When you know this, it will be more sense. Because for bit extraction we wrote a lot of code and tried to obtain every bits of it. In this part we will assign the value and without writing any code. We will extract the bits. Let's make it more clear.

```
union Packet{

    uint32_t packetValue;

    struct
    {
        uint32_t crc :2; LSB ( Low Significant Bit)
        uint32_t status : 1;
        uint32_t payload :12;
        uint32_t bat :3;
        uint32_t sensor :3;
        uint32_t longAddr :8;
        uint32_t shortAddr :2;
        uint32_t addrMode :1; MSB ( Most Significant Bit)
    } packetFields;
}
```



After doing bit extraction, time to decode.

```
union Packet packet;
printf("Enter the 32 bit packed value:");
scanf("%x",&packet.packetValue);

printf("decode = %#x\n",packet.packetFields.crc);
printf("decode = %#x\n",packet.packetFields.status);
printf("decode = %#x\n",packet.packetFields.payload);
printf("decode = %#x\n",packet.packetFields.bat);
printf("decode = %#x\n",packet.packetFields.sensor);
printf("decode = %#x\n",packet.packetFields.longAddr);
printf("decode = %#x\n",packet.packetFields.shortAddr);
printf("decode = %#x\n",packet.packetFields.addrMode);
```

name of union

struct name inside of union

specific value in our union

Then, the result will be some with other bit extraction.