7th Task in Embedded Systems

◈ What is the purpose of typedef?

typedef is used to create a new name for an existing data type.

- This makes code easier to read and manage, especially for complex types.

```
typedef unsigned int uint;
```

◈ How are bit fields declared and what are their size limitations?

Bit fields are used inside struct to store values in a specific number of bits.

Size limits:

- The size of a bit field can't be larger than the size of its underlying type (usually int, so typically 32 bits).
- Exact limits depend on the compiler and system.

```
struct Flags {
    unsigned int a : 1;        // uses 1 bit
    unsigned int b : 3;        // uses 3 bits
 };
```

◈ What happens if a bit field overflows?

If you store a number that needs more bits than allowed, it will be truncated (some bits are lost), leading to incorrect values.

Example:
If a 3-bit field can store values 0–7, and you try to store 9 → it gets cut to fit (and may become 1).

◈ How is typedef used with complex types like structs and unions?

-It gives a shorter name for a struct or union.

```
typedef struct {
    int x;
    int y;
 } Point;
```

Now instead of writing struct Point, you can just write Point.

◈ What is the default underlying type of an enum?

By default, the underlying type of an enum is int (usually 32 bits), unless specified otherwise.

```
enum Color { RED, GREEN, BLUE };     // all treated as ints by default
```

◈ How is a union different from a struct?

- In a struct, each member has its own space in memory.
- In a union, all members share the same memory (only one can be used at a time).

◈ When is using a union more memory-efficient?

A union is more memory-efficient **when you only need one variable at a time** from a group of variables.

Ex.

If you have a number that could be an int, float, or char, a union saves memory by using just one shared space.