What seems logical and readable for a non-embedded system does not make sense for an embedded system.

For embedded systems you need to what size everything and what is aligned to. Packing though not portable can help. Packing is not a complete solution.

Recommended code in a lot of circles is not good in an embedded system. Especially while passing data between processors and/or hardware.

```
enum type_of_struct_enum {STR_A, STR_B, STR_C};

struct
{
   type_of_struct_enum struct_type;  /* what size is this?
                                        FYI, it is not dependent on
                                        the number of enums */
   union
   {
      unsigned int reg; /* what size is this? what will the structure
                           be aligned to? */
      struct
      {
         int upperbits : 16; /* is this packed and on what boundary?
                                What are the relative bit locations
                                (endian-ness?) */
         int lowerbits : 16; /* is this packed and on what boundary?
                                What are the relative bit locations
                                (endian-ness?) */
      } bits;
   } reg_ctrl;
} recommended_but_poor_idea;
```

Better for an embedded system.

```
/* now you control the size */

#define uchar unsigned char

#define ulong unsigned int

enum type_of_struct_enum {STR_A, STR_B, STR_C};

struct
{
```

```
   uchar struct_type;  /* store the enums in it an things are
                          better (you will know the size) */
   ulong reg;          /* and or and not the bits to set them
                          (that is what the compiler will be
                          doing anyway) this data is not aligned
                          so may take longer to access and
                          require more assembly code */
} better_idea;
```