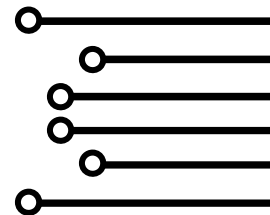
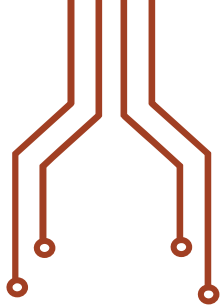




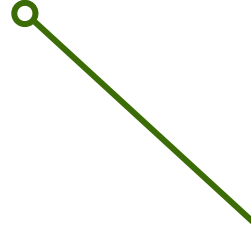
# C Embebido





```
typedef struct
{
    int32_t id_carro;
    char color[1];
    char conductor[10];
}uber_t;

uber_t uberX;
```



```
typedef struct{
    ....
}nombre_struct_t;
```

## Struct

## Bit Fields

Son variables que se definen utilizando un tamaño predefinido.

```
typedef struct
{
    uint8_t trueData    :1;
    uint8_t falseData   :1;
}data_t;
```

*tipo\_de\_dato* nombre\_miembro : ancho\_bits;

## Bit Fields

trueData	falseData	0	0	0	0	0	0
----------	-----------	---	---	---	---	---	---

data\_t

```
data_t X;  
  
X.falseData = 0;  
X.trueData = 1;
```

## Bit Fields

```
typedef struct
{
    uint32_t trueData    :4;
    uint32_t falseData   :4;
    uint32_t valueData    :8;
    uint32_t crcData      :8;
    uint32_t signalData   :4;
    uint32_t endData      :4;
}data_t;
```

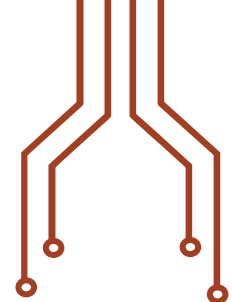
**NOTA:** Los miembros deben tener como máximo la cantidad de bits del tipo de dato.

# Que no puedo hacer

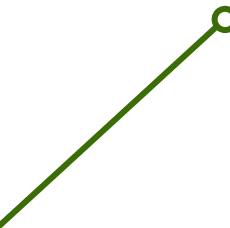
Wels

```
typedef struct
{
    uint8_t x[10] :4;
    uint8_t y :4;
}data_t;
```

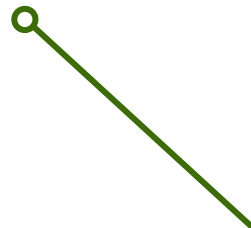
- No podemos tener punteros a los miembros de la estructura.
- El valor asignado a un miembro no puede estar fuera de rango.
- No se pueden utilizar arreglos como miembros en bit fields.



```
typedef union
{
    char  dato1;
    float dato2;
}ejemplo_union_t;
```



```
typedef union{
    ....
}nombre_union_t;
```



# UNION

# Uniones

```
typedef union
{
    char dato1;
    float dato2;
}ejemplo_union_t;
```

- Las uniones son un tipo de estructura de datos utilizada para crear tipos de datos definidos por el usuario.
- Todos los miembros se almacenan en la misma posición de memoria.
- El tamaño de la unión depende del tamaño del miembro más grande de la unión.
- Y sólo se puede obtener el valor de un miembro.



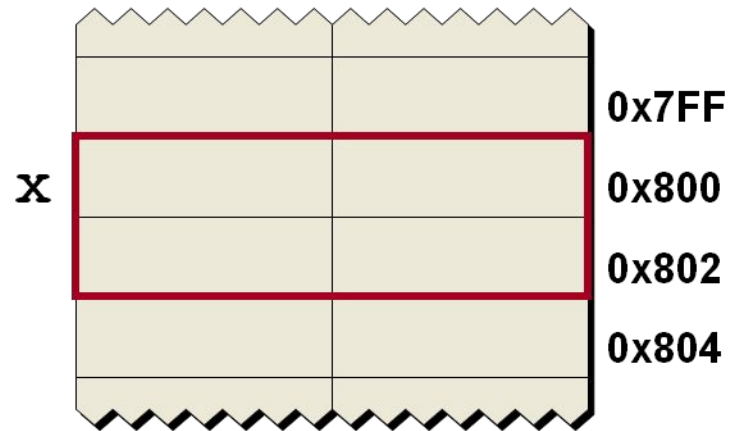
# Uniones

```
typedef union  
{  
    char a;  
    int b;  
    float c;  
} mixedBag;
```

```
mixedBag x;
```

Space allocated  
for x is size of  
float

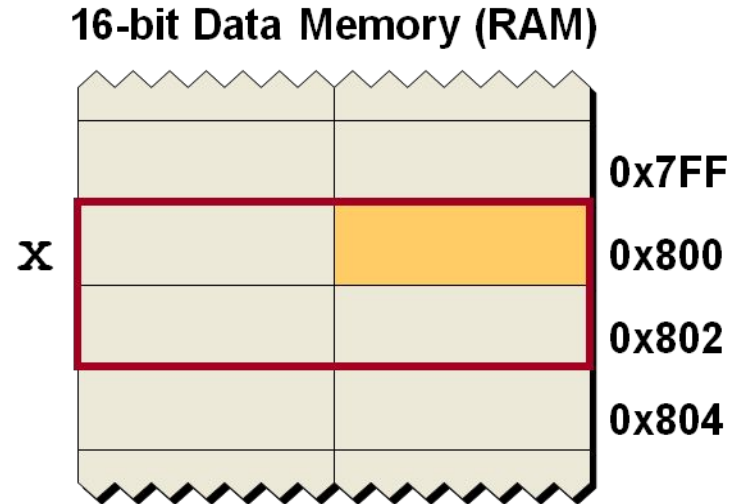
16-bit Data Memory (RAM)



# Uniones

```
typedef union  
{  
    char a;  
    int b;  
    float c;  
} mixedBag;  
  
mixedBag x;
```

`x.a` only  
occupies the  
lowest byte of  
the union

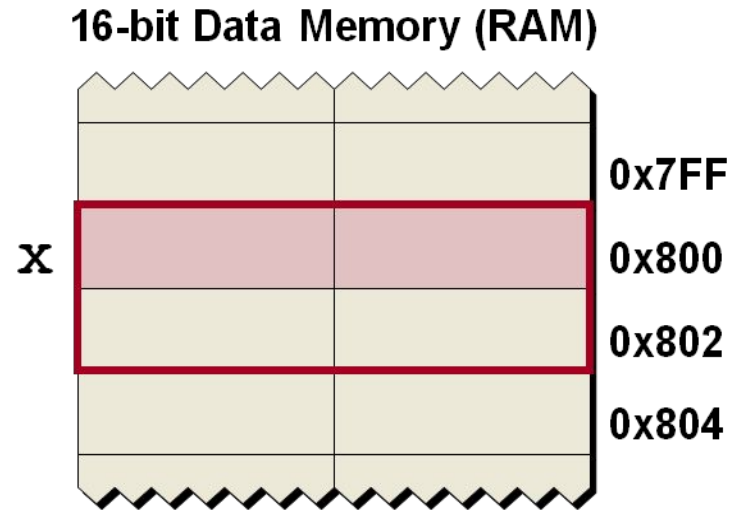


# Uniones

```
typedef union
{
    char a;
    int b;
    float c;
} mixedBag;

mixedBag x;
```

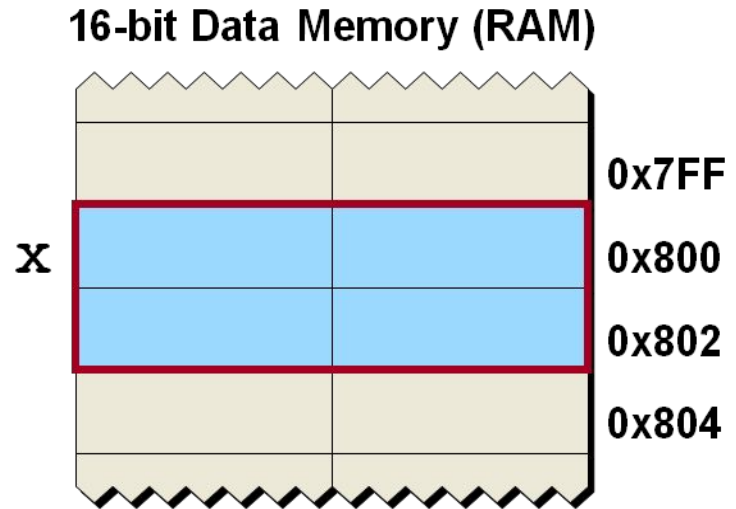
`x.b` only  
occupies the  
lowest two  
bytes of the  
union



# Uniones

```
typedef union  
{  
    char a;  
    int b;  
    float c;  
} mixedBag;  
  
mixedBag x;
```

**x.c occupies  
all four bytes of  
the union**





**Struct**

## TRISD



```
typedef union {  
    struct {  
        unsigned TRISD0      :1;  
        unsigned TRISD1      :1;  
        unsigned TRISD2      :1;  
        unsigned TRISD3      :1;  
        unsigned TRISD4      :1;  
        unsigned TRISD5      :1;  
        unsigned TRISD6      :1;  
        unsigned TRISD7      :1;  
    };  
    struct {  
        unsigned RD0         :1;  
        unsigned RD1         :1;  
        unsigned RD2         :1;  
        unsigned RD3         :1;  
        unsigned RD4         :1;  
        unsigned RD5         :1;  
        unsigned RD6         :1;  
        unsigned RD7         :1;  
    };  
} TRISDbits_t;
```

TRISDbits\_t



# Gracias

@welstheory

[hola@welstheory.com](mailto:hola@welstheory.com)

+51 918 899 684

