Structures

```c
struct tag-name {
    type member1;
    type member2;
    type member3;
    ...
    type memberN;
} variable-list;
```

--------------Example--------------

```c
void main(void) {
    struct mystructtag {
        float myfloat;
        int myint;
    } mystruct1,*mystruct2;

    mystruct2 = (struct mystructtag*)malloc(sizeof(struct mystructtag));

    mystruct1.myfloat = 1.2345;
    mystruct1.myint = 347;

    mystruct2->myfloat = mystruct1.myfloat/34.2;
    mystruct2->myint = 412;
}
```

--------------Also can define a new type--------------

```c
typedef struct {
    unsigned int addr;
    int cpr;
    int rollovers;
    int flags;
    unsigned int raw;
    enc_status_t status;
    float init_rad_coord;
    float k, k1;
} enc_t;
```

```c
cenc_t test1,test2;

test1.cpr = 1000;
test2.addr = 22;
```

--------------A function and return a structure--------------

```c
cenc_t myfunc(int val1,float val2);
```
struct tag-name {
    (unsigned or int) member1:size;
    (unsigned or int) member2:size;
    (unsigned or int) member3:size;
    ...
    (unsigned or int) memberN:size;
} variable-list;

--------------Example------------------

void main(void) {
    struct mybitfieldtag {
        unsigned fourbitval:4;
        unsigned eightbitval:8;
        int fourbitsigned:4;
    } mybitfield;

    mybitfield.fourbitval = (a number from 0 to 15);
    mybitfield.eightbitval = (a number from 0 to 255);
    mybitfield.fourbitsigned = (a number from -8 to 7);
}

Union in C

union tag-name {
   type member1;
   type member2;
   type member3;
   .
   .
   .
   type memberN;
} variable-list;

------------Example------------------
void main(void) {
   union myuniontag {
      unsigned char myc[4];
      unsigned int myunsignedint;
      int myint;
   } myunion;

   myunion.myc[0] = 0x23;
   myunion.myc[1] = 0x00;
   myunion.myc[2] = 0x45;
   myunion.myc[3] = 0xc3;

   What does myunion.myunsignedint equal?????
   What does myunion.myint equal ??????  Is it positive or negative

}
struct SPIFFRX_BITS {  // bits   description
  unsigned RXFFIL:5;  // 4:0    Interrupt level
  unsigned RXFFIENA:1;  // 5      Interrupt enable
  unsigned RXFFINTCLR:1;  // 6      Clear INT flag
  unsigned RXFFINT:1;  // 7      INT flag
  unsigned RXFFST:5;  // 12:8    FIFO status
  unsigned RXFIFORESET:1;  // 13    FIFO reset
  unsigned RXFFOVFCLR:1;  // 14    Clear overflow
  unsigned RXFFOVF:1;  // 15    FIFO overflow
};

union SPIFFRX_REG {
  unsigned all;
  struct SPIFFRX_BITS bit;
};

---

**Figure 2-10. SPI FIFO Receive (SPIFFRX) Register - Address 7948h**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>RXFFOF</td>
<td>0: Receive FIFO has not overflowed, read-only bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Receive FIFO has overflowed, read-only bit. More than 15 words have</td>
</tr>
<tr>
<td></td>
<td></td>
<td>been received in the FIFO, and the last received word is out.</td>
</tr>
<tr>
<td>14</td>
<td>RXFFOF</td>
<td>0: Write 0 does not affect RXFFOF flag bit, Bit reads back a zero</td>
</tr>
<tr>
<td></td>
<td>CLR</td>
<td>1: Write 1 to clear RXFFOF flag bit in bit 15.</td>
</tr>
<tr>
<td>13</td>
<td>RXFFIO</td>
<td>0: Write 0 to reset the FIFO pointer to zero, and hold in reset.</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>1: Re-enable transmit FIFO operation</td>
</tr>
<tr>
<td>8-12</td>
<td>RXFFST4-0</td>
<td>00000: Receive FIFO is empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00001: Receive FIFO has 1 word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00010: Receive FIFO has 2 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00011: Receive FIFO has 3 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1XX0: Receive FIFO has X words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: 10000: Receive FIFO has 16 words</td>
</tr>
<tr>
<td>7</td>
<td>RXFFINT</td>
<td>0: RXFFINT is interrupt has not occurred, read only bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: RXFFINT is interrupt has occurred, read only bit</td>
</tr>
<tr>
<td>6</td>
<td>RXFFINT</td>
<td>0: Write 0 has no effect on RXFFINT flag bit, Bit reads back a zero</td>
</tr>
<tr>
<td></td>
<td>CLR</td>
<td>1: Write 1 to clear RXFFINT flag bit in bit 7.</td>
</tr>
<tr>
<td>5</td>
<td>RXFFIENA</td>
<td>0: RX FIFO interrupt based on RXFFIEN match (less than or equal to) will</td>
</tr>
<tr>
<td></td>
<td></td>
<td>be disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: RX FIFO interrupt based on RXFFIEN match (less than or equal to) will</td>
</tr>
<tr>
<td></td>
<td></td>
<td>be enabled</td>
</tr>
<tr>
<td>4-0</td>
<td>RXFFIL4-0</td>
<td>11111: Receive FIFO Interrupt level bits, Receive FIFO will generate update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>when the FIFO status bits (RXFFST4-0) and FIFO level bits (RXFFIL4-0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>match (greater than or equal to). Default value of these bits after reset =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11111. This will avoid frequent interrupts, after reset, as the receive FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>will be empty most of the time.</td>
</tr>
</tbody>
</table>