AT90CAN128/64/32

CAN Interrupt Library

AVR GCC Platform:

WinAVR-200700525

CAN Interrupt Library

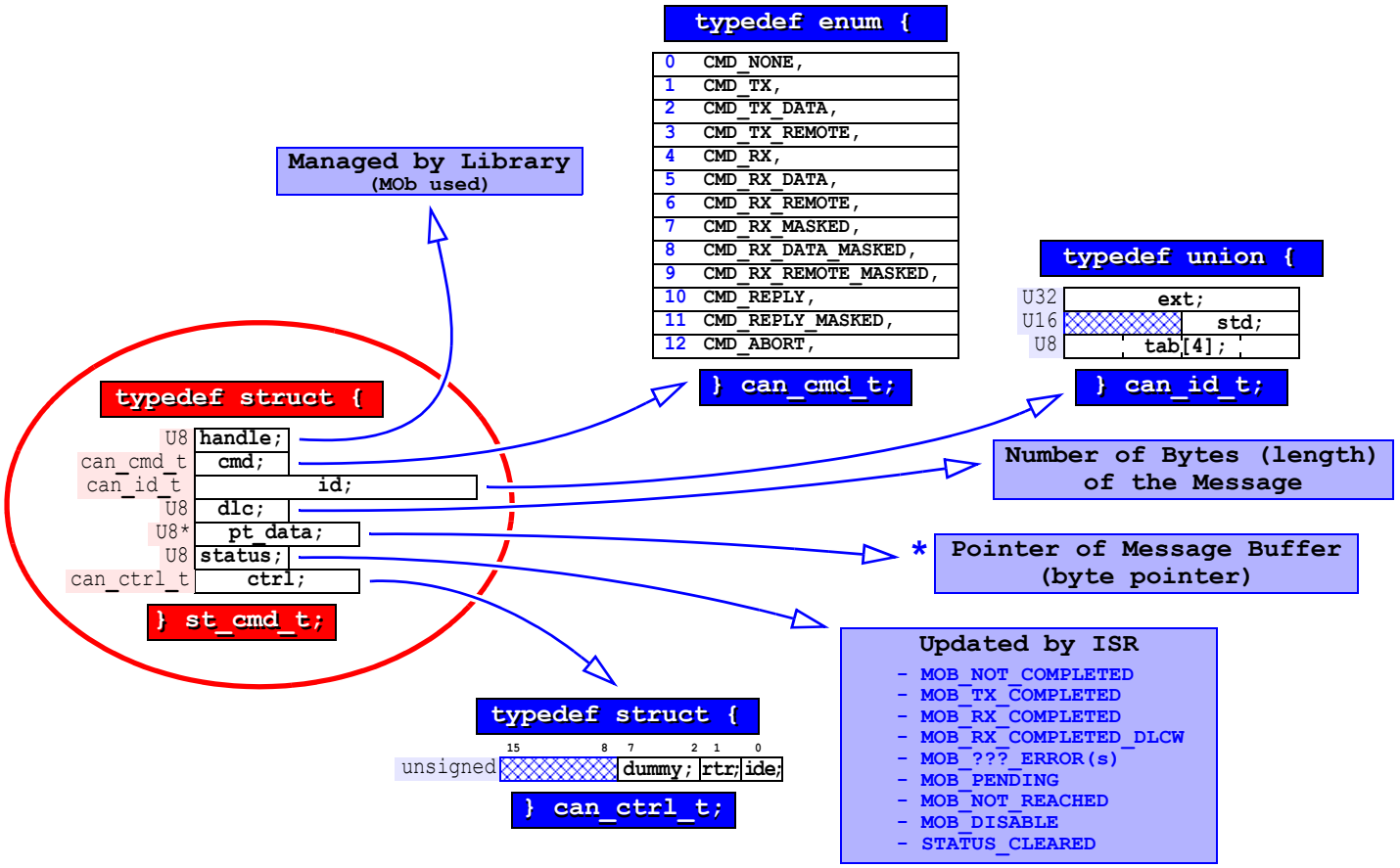
This library uses a 'C' structure to describe a CAN message. This structure is called CAN message descriptor. To handle such a descriptor a first function is provided to indicate what to do with the CAN message, a transmission, a reception, ... Once this command is entered, the CAN message is handled by interrupt. The status element of the CAN message descriptor structure will give information on the message events as transmission or reception performed.

1 - FOUR MAIN ELEMENTS (c.f. "can_lib.h")

1. ***"st_cmd_t"*** structure (CAN message descriptor),
2. ***"U8 can_cmd (st_cmd_t *)"*** function (to perform a command -action- on the CAN bus),
3. ***"ISR (CANIT_vect)"*** interrupt service routine (management of the enabled CAN communication),
4. "***st_cmd_t *can_descript_list []"*** array of pointers to user CAN message descriptor structures.

1.1 - "*st_cmd_t*" Structure

Figure 1. CAN Message Descriptor Structure



1.2 - "*U8 can_cmd (st_cmd_t *)*" Function

- This function needs as argument a pointer on a structure of ***"st_cmd_t"*** type, a CAN message descriptor.
- This function looks for a MOb free and configures it with data given by the CAN message descriptor.

- This function updates the ***"can_descript_list [] "***. The index of this list is the MOb number used. The element of the array is the pointer to the CAN message descriptor passed as function argument.
- This function updates the status of the CAN message descriptor and returns the rating of its action: **-> CAN_CMD_ACCEPTED** or **CAN_CMD_REFUSED**.

1.3 - "*ISR (CANIT_vect)*" Interrupt Service Routine

- This function recovers the MOb number responsible of the interrupt and points to the corresponding CAN message descriptor thanks to the ***"can_descript_list []"***.
- This routine performs the necessary management of the CAN message and updates the status element of the CAN message descriptor.
- If a general interrupt rises, the interrupt service routine updates some global variables:
  - "*can_bus_off_count* ", Bus-Off counter,
  - "*can_general_error_count* ", CAN general error counter,
  - "*can_manag_error_count* ", error counter on descriptor/MOb management.

1.4 - "*st_cmd_t*" *can_description_list []*

- The ***"can_cmd ()"*** function initializes one element of this array (index = MOb number used or "handle" element value of the CAN Message Descriptor structure).
- The ***"CANIT_vect"*** interrupt service routine uses this array to recover the CAN message descriptor associated to the MOb sourcing the interrupt.

2. Example of Using

Reception of any CAN message

2.1 - Simple

```
//.......
st_cmd_t message;
unsigned char buffer[8];

//.......
message.pt_data = &buffer[0];
message.cmd = CMD_RX;
//.......
while(can_cmd(&message) != CAN_CMD_ACCEPTED);
while(message.status == MOB_PENDING);
//.......
```

2.2 - Full

```
//.......
unsigned char buffer[];
st_cmd_t message;

//.......
message.pt_data = &buffer[0];
message.cmd = CMD_RX;
// Initialization of the reception
if(can_cmd(&message) == CAN_CMD_REFUSED)
{
    //....... No MOb available
}
//.......
// Is there a received CAN message ?
```

```
switch (message.status)
{
    case MOB_RX_COMPLETED:
         //.......
         break;
    case MOB_RX_COMPLETED_DLCW:
         //.......
         break;
    case MOB_ACK_ERROR:
    case MOB_FORM_ERROR:
    case MOB_CRC_ERROR:
    case MOB_STUFF_ERROR:
    case MOB_BIT_ERROR:
         //.......
         break;
    case MOB_PENDING:
         //.......
         break;
    default:
         //.......
         break;
}
//.......
```

3 - OTHER FUNCTION(S)

3.1 - "*U8 can_init (U8 mode)*" Function

- This function performs a full initialization of the CAN controller enabling global interrupt ("SEI") and CAN interrupts (except Overrun CAN Timer interrupt). This function also accepts an automatic recognition of the Baud Rate on the CAN bus (c.f. ***"#define CAN_BAUDRATE ..."*** in ***"config.h"*** file).
- The first time, this function needs ***"0"*** as argument.
  If the automatic recognition of the Baud Rate has been used and if some fails are detected while running your program, a new call to this function can be performed. At this moment, if ***"1"*** is passed as argument the function will try other values than those (previouly) used.
- When automatic recognition of the Baud Rate is used, this function returns ***"0x00"*** when **none** Baud Rate has been found, else ***"0x01"*** is returned.

Example:

```
void main (void)
{
unsigned char temp;

temp = 0;
//.......
while(1)
{
    temp = can_init(temp);
    if (temp != 0)
    {
        //.......
        your_program(); // If too many errors, output from "your_program()"
    }
    else
    {
        printf("No synchro, exit.\r\n"); break;
    }
}
//....... Synchro error
//.......
```

Notes: