**RX CAN**

The library is implemented for Standard CAN frame type and the address, i.e. The packet ID is specified in the decimal number system and not in HEX. All the functions work on the principle of binary vector and are defined only for IDs from 0 to 15 decimal (that is from 0h to Fh), for only 16 ID addresses.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Position | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

To make it easier to implement data retrieval, the data is divided into 4 packets of 2Byte width instead of 8 packets of 1 Byte. The CAN protocol framework is organized so that a maximum of 8 Bytes of data can be sent for one ID value. Testing concluded that 1Byte is not enough for each data, so a new standard of 2Byte was adopted for each data. This means that a maximum of four data can be sent per ID. The modified CAN frame for custom CAN Library would look like the follows:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Position | 7 | 6 | 5 | 4 | 3. | 2 | 1 | 0 |
| Size: | 1Byte | 1Byte | 1Byte | 1Byte | 1Byte | 1Byte | 1Byte | 1Byte |
| Position | 3 | | 2 | | 1 | | 0 | |
| Size: | 2 Byte | | 2 Byte | | 2 Byte | | 2 Byte | |

**TX CAN**

One of the main tasks of the library was to make a fixed sending of messages by CAN without doing so using the while (1) loop in the main () function. Considering this problem, a decision was made to join the library with timers.

This library provides the ability to send CAN messages simultaneously at three different time intervals. It was implemented using three timers, so when using it, the question arose how to call and initialize these timers. If the requests of the whole program can be realized at only one fixed interval of sending messages for different or the same ID packets, then the program uses functions that are NOT indexed with - [1,2, 3]. If the requirements of the entire program do not meet the needs of one-time interval for sending data, then you should, exclusively, use the indexed functions - Note \* and Note \*\* (see table).

***CAN Library***

|  |  |
| --- | --- |
| Functions | Description |
| void can\_initRX(); | Initialization of the protocol for receiving messages by CAN. Be sure to define at the beginning of the main () function, outside the while loop. |
| void can\_msg\_receive(); | Interrupt function called when receiving messages via CAN. Do not use in user mode! |
| uint16\_t get\_data(char ID, char Position); | Retrieves 2Byte size data from ID - ID and position (0-3). |
| bool check\_flag(char ID); | Returns true if a CAN packet arrived from the given ID as a function argument. Otherwise returns false. |
| void canTX\_set\_Interval(float); | Function for setting a fixed CAN message interval. The value of the real parameter is in seconds. |
| void canTX\_set\_Interval\_1(float);  void canTX\_set\_Interval\_2(float);  void canTX\_set\_Interval\_3(float); | Function to set the interval for sending messages to a fixed time interval. The value of the parameter is in seconds. |
| void can\_initTX(char); | Initialize sending CAN messages. The function parameter is the ID for sending CAN messages. The ID is specified in the decimal number system. |
| void can\_initTX1(char);  void can\_initTX2(char);  void can\_initTX3(char); | Initialize sending CAN messages. The function parameter is the ID for sending CAN messages. The ID is specified in the decimal number system. Note \*. |
| void can\_msg\_send(); | Function for sending CAN messages to the specified sending ID. It is not sent periodically (setting the interval of sending a message has no effect on this function). The message length is set to 8B. |
| void can\_msg\_send1();  void can\_msg\_send2();  void can\_msg\_send3(); | Function for sending CAN messages to the specified sending ID. It is not sent periodically (setting the interval of sending a message has no effect on this function). The message length is fixed to 8B.Note  \*\*. |
| void can\_msg\_send\_tick(); | Function for sending CAN messages to the specified sending ID and the interval specified by the canTX\_set\_Interval (float) function and initialized can\_initTX (char). It is not sent periodically (setting the message interval has no effect on this function). The message length is set to 8B. |
| void can\_msg\_send\_tick1();  void can\_msg\_send\_tick2();  void can\_msg\_send\_tick3(); | Function for sending CAN messages to the specified sending ID and the interval specified by the canTX\_set\_Interval (float) function and initialized can\_initTX (char). It is sent periodically. The message length is set to 8B. |
| void pack\_data(uint16\_t data, char pos); | The first argument of the function is 2B data, which is sent at position pos in the CAN message. |
| void pack\_data(char data, char pos); | The first argument of the function is 1B data, which is sent at position pos in the CAN message. |
| void pack\_data1(uint16\_t, char);  void pack\_data1(char, char);  void pack\_data2(uint16\_t, char);  void pack\_data2(char, char);  void pack\_data3(uint16\_t, char);  void pack\_data3(char, char); | The first argument of the function is 2B / 1B data, which is sent at position pos in the CAN message. To apply these functions, see Note \*. |