```
        memcpy(handler_states[handler_num].tx_buffer + b_ind, data, len);
        b_ind += len;

        crc = crc16(data, len);
        handler_states[handler_num].tx_buffer[b_ind++] = (uint8_t)(crc >> 8);
        handler_states[handler_num].tx_buffer[b_ind++] = (uint8_t)(crc & 0xFF);
        handler_states[handler_num].tx_buffer[b_ind++] = 3;

        if (handler_states[handler_num].send_func) {
            handler_states[handler_num].send_func(handler_states[handler_num].tx_buffer,
b_ind);
        }
}
```

5.3 MIT Mode Communication Protocol

Special CAN Codes

Enter Motor Control Mode: {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,0XFC }

Exit Motor Control Mode: {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0XFD }

Set Current Motor Position as zero position: {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0XFE }

Note: It is necessary to enter Motor Control Mode before controlling the motor using CAN communication!

PS: (If you want to read the current state in a stateless manner, the command to send is {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,0XFC }）

MIT Mode Driver Board Receive Data Definition

Identifier: Set Motor ID (default is 1)　　Frame Type: Standard Frame

Frame Format: DATA　　Data Length Code (DLC): 8 Bytes

Data Field	DATA[0]	DATA[1]	DATA[2]	DATA[3]	
Data Bits	7-0	7-0	7-0	7-4	3-0
Data Content	Motor Position High 8 bits	Motor Position Low 8 bits	Motor Speed High 8 bits	Motor Speed Low 4 bits	KP Value High 4 bits

Data Field	DATA[4]	DATA[5]	DATA[6]		DATA[7]
Data Bits	7-0	7-0	7-4	3-0	0-7
Data Content	KP Value Low 8 bits	KD Value High 8 bits	KD Value Low 4 bits	Current Value High 4 bits	Current Value Low 8 bits

MIT Mode Driver BoardSend Data Definition

Identifier: 0X00+Driver ID　　Frame Type: Standard Frame

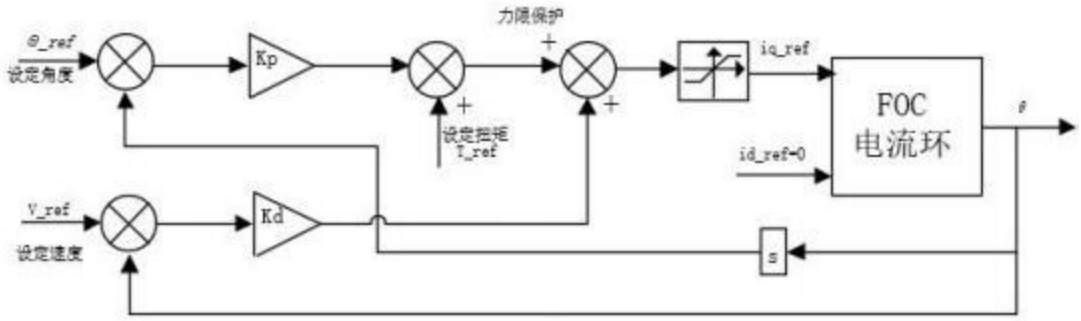
Frame Format: DATA　　Data Length Code (DLC): 8 Bytes

Data Field	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]
Data Bits	7-0	7-0	7-0	7-0	7-4
Data Content	Driver ID Number	Motor Position High 8 bits	Motor Position Low 8 bits	Motor Speed High 8 bits	Motor Speed Low 4 bits

Data Field	DATA[4]	DATA[5]	DATA[6]	DATA[7]
Data Bits	3-0	7-0	7-0	7-0
Data Content	Current Value High 4 bits	Current Value Low 4 bits	Motor Temperature	Motor Error Flag

CAN Speed: 1 MHz

MIT mode simplified control block diagram



Parameter Ranges:

Module	AK10-9	AK60-6	AK70-10	AK80-6	AK80-9	AK80-64	AK80-8
Position (rad)	-12.5f-12.5f						
Speed (rad/s)	-50.0f-50.0f	-45.0f-45.0f	-50.0f-50.0f	-76.0f-76.0f	-50.0f-50.0f	-8.0f-8.0f	-37.5f-37.5f
Torque (N.M)	-65.0f-65.0f	-15.0f-15.0f	-25.0f-25.0f	-12.0f-12.0f	-18.0f-18.0f	-144.0f-144.0f	-32.0f-32.0f
Kp Range	0-500						
Kd Range	0-5						

MIT Mode Sending&Receiving Code Example

Sending Example Code

```
void pack_cmd(CANMessage * msg, float p_des, float v_des, float kp, float kd, float t_ff){
    /// limit data to be within bounds ///
    float P_MIN =-12.5f;
    float P_MAX =12.5f;
    float V_MIN =-30.0f;
    float V_MAX =30.0f;
    float T_MIN =-18.0f;
    float T_MAX =18.0f;
    float Kp_MIN =0;
    float Kp_MAX =500.0f;
    float Kd_MIN =0;
    float Kd_MAX =5.0f;
    float Test_Pos=0.0f;

    p_des = fminf(fmaxf(P_MIN, p_des), P_MAX);
    v_des = fminf(fmaxf(V_MIN, v_des), V_MAX);
    kp = fminf(fmaxf(Kp_MIN, kp), Kp_MAX);
    kd = fminf(fmaxf(Kd_MIN, kd), Kd_MAX);
    t_ff = fminf(fmaxf(T_MIN, t_ff), T_MAX);
    /// convert floats to unsigned ints ///

    int p_int = float_to_uint(p_des, P_MIN, P_MAX, 16);
    int v_int = float_to_uint(v_des, V_MIN, V_MAX, 12);
    int kp_int = float_to_uint(kp, KP_MIN, KP_MAX, 12);
    int kd_int = float_to_uint(kd, KD_MIN, KD_MAX, 12);
    int t_int = float_to_uint(t_ff, T_MIN, T_MAX, 12);

    /// pack ints into the can buffer ///
    msg->data[0] = p_int>>8;              // Position High 8
    msg->data[1] = p_int&0xFF;    // Position Low 8
    msg->data[2] = v_int>>4;              // Speed High 8 bits
    msg->data[3] = ((v_int&0xF)<<4)|(kp_int>>8);   // Speed Low 4 bits KP High 4 bits
    msg->data[4] = kp_int&0xFF;   // KP Low 8 bits
    msg->data[5] = kd_int>>4;     // Kd High 8 bits
    msg->data[6] = ((kd_int&0xF)<<4)|(t_int>>8);        // KP Low 4 bits Torque High 4 bits
    msg->data[7] = t_int&0xff;    // Torque Low 8 bits
}
```

When sending packets, all numbers need to go through the following function to be converted into integer values before being sent to the motor:

```
int float_to_uint(float x, float x_min, float x_max, unsigned int bits){
    /// Converts a float to an unsigned int, given range and number of bits ///
    float span = x_max - x_min;
    if(x < x_min) x = x_min;
    else if(x > x_max) x = x_max;
    return (int) ((x- x_min)*((float)((1<<bits)/span)));
}
```

Receiving Example Code

```
void unpack_reply(CANMessage msg){
    /// unpack ints from can buffer ///
    int id = msg.data[0]; //Driver ID
    int p_int = (msg.data[1]<<8)|msg.data[2];              // Motor Position Data
    int v_int = (msg.data[3]<<4)|(msg.data[4]>>4);         // Motor Speed Data
    int i_int = ((msg.data[4]&0xF)<<8)|msg.data[5];        //Motor Torque Data
    Int T_int = msg.data[6] ;
    /// convert ints to floats ///
    float p = uint_to_float(p_int, P_MIN, P_MAX, 16);
    float v = uint_to_float(v_int, V_MIN, V_MAX, 12);
    float i = uint_to_float(i_int, -I_MAX, I_MAX, 12);
    float T =T_int;
    if(id == 1){
      postion = p;                    // Read corresponding data based on ID
      speed = v;
      torque = i;
      Temperature = T-40;     // Temperature range: -40~215
      }
   }
```

When receiving, convert all values to floating-point numbers using the following function:

```
float uint_to_float(int x_int, float x_min, float x_max, int bits){
    /// converts unsigned int to float, given range and number of bits ///
    float span = x_max - x_min;
    float offset = x_min;
    return ((float)x_int)*span/((float)((1<<bits)-1)) + offset;
}
```