# M0601Instructions

typedef struct Received

{

uint8\_t Temp;//Temp of motor

uint16\_t ECurru;//electric current of motor

int16\_t BSpeed;//FeedBack Speed of motor

uint16\_t ErrCode;//Fault code of motor

uint8\_t BMode;//Feedback Mode of motor

uint16\_t Position;//Feedback Position of motor

}Receiver;

The structure is used to save the received motor state.

uint8\_t Tx[10];

uint8\_t Rx[10];

Receive and send data cache.

1.Send Data to Motor

/\*\*

\* @name Send\_Motor

\* @brief This function is used to Send data to motor.

\* @param None

\* @retval None

\*/

void Send\_Motor(void)

{

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_8,GPIO\_PIN\_SET);

uint16\_t err=HAL\_UART\_Transmit(&huart1,Tx,10,0xffff);

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_8,GPIO\_PIN\_RESET);

HAL\_Delay(10);

}

The motor is based on RS485 communication.This function sends data to the motor.

2.Calculate CRC

/\*\*

\* @name CRC8\_Table

\* @brief This function is used to Get CRC8 check value.

\* @param p :Array pointer. counter :Number of array elements.

\* @retval crc8

\*/

unsigned char CRC8\_Table(unsigned char \*p, int counter)

{

unsigned char crc8 = 0;

for( ; counter > 0; counter--)

{

crc8 = CRC8Table[crc8^\*p];

p++;

}

return(crc8);

}

Get CRC8 check value.CRC verification is required for data frames sent by most function functions.

3.Control Motor

/\*\*

\* @name Control\_Motor

\* @brief This function is used to Set the speed,acceleration

or braking of motor.The end of the data

frame shall be CRC8 verified.

(the default speed is 0, which can also be braked)

\* @param Speed, ID, Acce, Brake\_P, Receiver

\* @retval None

\*/

void Control\_Motor(uint16\_t Speed,uint8\_t ID,uint8\_t Acce,uint8\_t Brake\_P,Receiver\* Receiver)

{

Tx[0]=ID;

Tx[1]=0x64;

Tx[2]=Speed>>8;

Tx[3]=Speed&0x00FF;

Tx[4]=0;

Tx[5]=0;

Tx[6]=Acce;

Tx[7]=Brake\_P;

Tx[8]=0;

Tx[9]=CRC8\_Table(Tx,9);

Send\_Motor();

HAL\_Delay(10);

Receiver->BMode=Rx[1];

Receiver->ECurru=(Rx[2]<<8)+Rx[3];

Receiver->BSpeed=(Rx[4]<<8)+Rx[5];

Receiver->Position=(Rx[6]<<8)+Rx[7];

Receiver->ErrCode=Rx[8];

}

4.Get other feedback from the motor

/\*\*

\* @name Get\_Motor

\* @brief This function is used Get other feedback from the motor

\* @param ID,Receiver

\* @retval None

\*/

void Get\_Motor(uint8\_t ID,Receiver\*Receiver)

{

Tx[0]=ID;

Tx[1]=0x74;

Tx[2]=0;

Tx[3]=0;

Tx[4]=0;

Tx[5]=0;

Tx[6]=0;

Tx[7]=0;

Tx[8]=0;

Tx[9]=CRC8\_Table(Tx,9);

Send\_Motor();

HAL\_Delay(10);

Receiver->BMode=Rx[1];

Receiver->ECurru=(Rx[2]<<8)+Rx[3];

Receiver->BSpeed=(Rx[4]<<8)+Rx[5];

Receiver->Temp=Rx[6];

Receiver->Position=Rx[7];

Receiver->ErrCode=Rx[8];

}

The difference between this function and the data frame obtained after transmitting the drive motor function lies in the feedback temperature.

5.Set Motor Mode

/\*\*

\* @name Set\_MotorMode

\* @brief This function is used Set Mode of motor

\* @param ID Mode

\* @retval None

\*/

void Set\_MotorMode(uint8\_t Mode,uint8\_t ID)

{

Tx[0]=ID;

Tx[1]=0xA0;

Tx[2]=0;

Tx[3]=0;

Tx[4]=0;

Tx[5]=0;

Tx[6]=0;

Tx[7]=0;

Tx[8]=0;

Tx[9]=Mode;

Send\_Motor();

}

6.Set Motor ID

/\*\*

\* @name Set\_MotorID

\* @brief This function is used Set Mode of motor

\* @param ID

\* @retval None

\*/

void Set\_MotorID(uint8\_t ID)

{

Tx[0]=0xAA;

Tx[1]=0x55;

Tx[2]=0x53;

Tx[3]=ID;

Tx[4]=0;

Tx[5]=0;

Tx[6]=0;

Tx[7]=0;

Tx[8]=0;

Tx[9]=0;

Send\_Motor();

}

7.Check Motor

/\*\*

\* @name Check\_Motor

\* @brief This function is used to Query motor.

\* @param Receiver

\* @retval None

\*/

void Check\_Motor(Receiver\*Receiver)

{

Tx[0]=0xc8;

Tx[1]=0x64;

Tx[2]=0;

Tx[3]=0;

Tx[4]=0;

Tx[5]=0;

Tx[6]=0;

Tx[7]=0;

Tx[8]=0;

Tx[9]=CRC8\_Table(Tx,9);

Send\_Motor();

HAL\_Delay(10);

Receiver->BMode=Rx[1];

Receiver->ECurru=(Rx[2]<<8)+Rx[3];

Receiver->BSpeed=(Rx[4]<<8)+Rx[5];

Receiver->Position=(Rx[6]<<8)+Rx[7];

Receiver->ErrCode=Rx[8];

}