# M1502ADocumentation

TxData[10]:Global Array Variable

RxData[10]:Global Array Variable

Uint8\_t Temp;global variable //Value of Temp

/\*\*

\* @brief CAN Tx message header structure definition

\*/

typedef struct

{

uint32\_t **StdId**; /\*!< Specifies the standard identifier.

uint32\_t **ExtId**; /\*!< Specifies the extended identifier.

uint32\_t **IDE**; /\*!< Specifies the type of identifier for the message that will be transmitted. This parameter can be a value of @ref CAN\_identifier\_type \*/

uint32\_t **RTR**; /\*!< Specifies the type of frame for the message that will be transmitted.

This parameter can be a value of @ref CAN\_remote\_transmission\_request \*/

uint32\_t **DLC**; /\*!< Specifies the length of the frame that will be transmitted.

This parameter must be a number between Min\_Data = 0 and Max\_Data = 8. \*/

} CAN\_TxHeaderTypeDef;

CAN\_TxHeaderTypeDef TxHeader;//global variable

typedef struct report

{

uint8\_t ErrCode;//FeedBack ErrorCode of motor

uint16\_t FBSpeed;//FeedBack Speed of motor

uint16\_t ECurru;//FeedBack electric current of motor

uint16\_t Position;//FeedBack Position of motor

uint8\_t FBMode;//FeedBack Mode of motor

}reporter;

### Important ：Send data to motor

/\*\*

\* @name USER\_CAN\_Send

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

Set stdid before using the function.

\* @param None

\* @retval None

\*/

void USER\_CAN\_Send(void)

{

TxHeader.ExtId = 0x00;

TxHeader.RTR = CAN\_RTR\_DATA;

TxHeader.IDE = CAN\_ID\_STD;

TxHeader.DLC = 8;

TxHeader.TransmitGlobalTime = DISABLE;

HAL\_Delay(1000);

if (HAL\_CAN\_AddTxMessage(&hcan, &TxHeader, TxData, &TxMailbox) != HAL\_OK)

{

/\* Transmission request Error \*/

Error\_Handler();

}

HAL\_Delay(10);

TxData[0]++;

}

First,The identifier needs to be assigned to the corresponding member of the structure.Put data into Txdata, The value of TxHeader.DLC determines the length of data sent.Send data by calling the can send function of Hal library.

### motor drive

/\*\*

\* @name Send

\* @brief This function is used to The driving motor changes the motion state

If id = 2, the second data frame of speed or current position value

is raised by eight bits and the third is lowered by eight bits. ID <= 8

ID determines the data storage location of the data frame.

\* @param Speed ID

\* @retval None

\*/

void Motor\_Drive(uint16\_t Speed,uint8\_t ID)

{

uint16\_t i=0;

while(i<8)

{

TxData[i++]=0;

}

TxData[ID-1]=Speed>>8;

TxData[ID]=Speed&0x00ff;

TxHeader.StdId=0x32;

USER\_CAN\_Send();

}

This function is used to drive different ID motors and set speed, current or position in different modes.According to the M1502A motor communication protocol, the identifier is 0x32.(The motor mode defaults to open-loop mode.)

### **Set motor mode**

/\*\*

\* @name Motor\_SetMode

\* @brief This function is used to Set motor mode

mode 0x00 open loop 0x01 Current loop

0x02 Velocity loop 0x03 Position ring

\* @param Mode

\* @retval FeedBack Mode Value

\*/

uint8\_t Motor\_SetMode(unsigned char Mode)

{

uint16\_t i=0;

while(i<8)

{

TxData[i++]=0;

}

TxData[0]=Mode;

TxHeader.StdId=0x105;

USER\_CAN\_Send();

return RxData[0];

}

This function is used to set the motor mode. The transmitted data frame is 0 except that the first data is mode.The identifier is 0x105. The motor will feed back the data frame after receiving the command The data frame is 0 except that the first data is mode too.

### **Set motor ID**

/\*\*

\* @name ID\_Set

\* @brief This function is used to Set motor ID

\* @param None

\* @retval FeedBack ID

\*/

uint8\_t ID\_Set(uint8\_t ID)

{

uint16\_t i=0;

while(i<8)

{

TxData[i++]=0;

}

TxHeader.StdId=0x108;

TxData[0]=ID;

USER\_CAN\_Send();

return RxData[0];

}

This function is used to set the motor ID. The transmitted data frame is 0 except that the first data is ID.The identifier is 0x108. The motor will feed back the data frame after receiving the command The data frame is 0 except that the first data is ID too.

### **Obtain motor report data**

/\*\*

\* @name Obtain\_Motor\_Report

\* @brief This function is used to Obtain the feedback

information actively reported by the motor

\* @param report

\* @retval None

\*/

void Obtain\_Motor\_Report(reporter\*report)

{

report->ECurru=(RxData[2]<<8)+RxData[3];

report->FBSpeed=(RxData[0]<<8)+RxData[1];

report->Position=(RxData[4]<<8)+RxData[5];

report->ErrCode=RxData[6];

report->FBMode=RxData[7];

}

When the feedback mode value is less than 0x80, the motor will report the data every time it passes the feedback mode value Ms.The reported data frame is fixed as speed, current, position, fault code and mode.(The feedback mode defaults to report feedback.)

**5.Query motor status**

/\*\*

\* @name Ck\_Check

\* @brief This function is used to The feedback method is to query and

specify the values of three objects on the premise of query. 0x01:Query speed 0x02:Query torque current 0x03:Query temperature0x04:Query location 0x05:Query fault code 0x06:Query mode.

\* @param ID Check1 Check2 Check3 reporter

\* @retval None

\*/

void Ck\_Check(uint8\_t ID,uint8\_t Check1,uint8\_t Check2,uint8\_t Check3,reporter\* reporter)

{

TxData[0]=ID;

TxData[1]=Check1;

TxData[2]=Check2;

TxData[3]=Check3;

TxData[4]=0xAA;

TxData[5]=0;

TxData[6]=0;

TxData[7]=0;

TxHeader.StdId=0x107;

USER\_CAN\_Send();

switch(Check1)

{

case 1:

{

reporter->FBSpeed=(RxData[0]<<8)+RxData[1];

break;

}

case 2:

{

reporter->ECurru=(RxData[0]<<8)+RxData[1];

break;

}

case 3:

{

Temp=(RxData[0]<<8)+RxData[1];

break;

}

case 4:

{

reporter->Position=(RxData[0]<<8)+RxData[1];

break;

}

case 5:

{

reporter->ErrCode=(RxData[0]<<8)+RxData[1];

break;

}

case 6:

{

reporter->FBMode=(RxData[0]<<8)+RxData[1];

break;

}

}

switch(Check2)

{

case 1:

{

reporter->FBSpeed=(RxData[2]<<8)+RxData[3];

break;

}

case 2:

{

reporter->ECurru=(RxData[2]<<8)+RxData[3];

break;

}

case 3:

{

Temp=(RxData[2]<<8)+RxData[3];

break;

}

case 4:

{

reporter->Position=(RxData[2]<<8)+RxData[3];

break;

}

case 5:

{

reporter->ErrCode=(RxData[2]<<8)+RxData[3];

break;

}

case 6:

{

reporter->FBMode=(RxData[2]<<8)+RxData[3];

break;

}

}

switch(Check3)

{

case 1:

{

reporter->FBSpeed=(RxData[4]<<8)+RxData[5];

break;

}

case 2:

{

reporter->ECurru=(RxData[4]<<8)+RxData[5];

break;

}

case 3:

{

Temp=(RxData[4]<<8)+RxData[5];

break;

}

case 4:

{

reporter->Position=(RxData[4]<<8)+RxData[5];

break;

}

case 5:

{

reporter->ErrCode=(RxData[4]<<8)+RxData[5];

break;

}

case 6:

{

reporter->FBMode=(RxData[4]<<8)+RxData[5];

break;

}

}

}

Before querying, you need to set the feedback mode to query mode.This function is used to query status values that are not in the reporting mode.Fill in the ID of the transmitted data frame, query target value 1, query target value 2, query target value 3 and self-set value.

The data frame returned by the motor successively returns the high eight bits of data 1, the low eight bits of data 1, the high eight bits of data 2, the low eight bits of data 2, the high eight bits of data 3, the low eight bits of data 3, and the self-set value.

**6. motor Calibration**

/\*\*

\* @name Motor\_Calibration

\* @brief This function is used to Motor calibration before operation

\* @param None

\* @retval None

\*/

void Motor\_Calibration(void)

{

uint16\_t i=0;

while(i<8)

{

TxData[i++]=0;

}

TxHeader.StdId=0x104;

USER\_CAN\_Send();

}

This function is used to Motor calibration before operation .Only blank data frames with identifier 0x104 are sent.It is recommended that the motor be calibrated before each drive.

**7.Set motor feedback mode**

/\*\*

\* @name Motor\_Set\_FeedBack

\* @brief This function is used to set the feedback mode of the motor

\* @param FeedBack If it is greater than 0x80, you need to obtain the

relevant value through the query function,Less than 0x40,

the motor actively reports fixed information, and the value

determines the reporting time interval (default).ID determines

the data storage location of the data frame.

\* @retval FBFeedback

\*/

uint8\_t Motor\_Set\_FeedBack(unsigned char FeedBack,uint8\_t ID)

{

uint16\_t i=0;

while(i++<8)

{

TxData[i]=0;

}

TxData[ID-1]=FeedBack;

TxHeader.StdId=0x106;

USER\_CAN\_Send();

return RxData[0];

}

This function sets the feedback mode of the motor corresponding to the ID.The identifier is 0x106.Assign feedback values to the elements corresponding to Id-1 in the data frame, and set the rest to 0.