MIT Driver board documentation V1.3

Jan, 2020

1. Hardware Description of the Driver Board

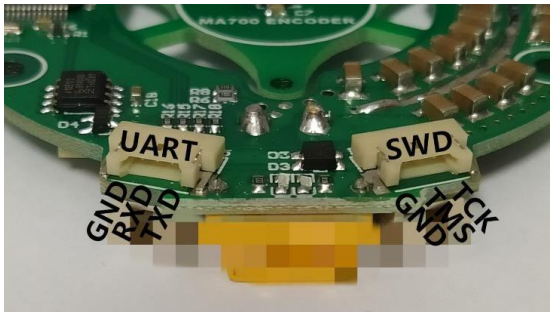
(1) The driver boards have been tested before leaving factory.

(2) The red light of the driver is the power indicator, and the green light is the system status light.

(3) The encoder model used in the latest open source hardware is MA700. The current V3 version motor uses MA702 encoder (the motor is labeled with Ma702 Encoder). (See the attachment for the data manual of the two types of encoders).

(4) Important note: Because of the large current of the motor, the current version of the driver board does not provide anti-reverse connection of the power supply. Please supply power to the motor correctly through the XT30 interface. The normal supply voltage is 24V.

(5) The description of each interface of the driver board is shown in the figure below. The serial port terminal model is molex51146-3pin, the SWD program burning port terminal model is molex51146-3pin, the 5V power input port terminal model is GH1.25-2pin, and the CAN bus interface terminal model is GH1.25-2pin. The specific pins of each interface are shown in the figure below:



(6) About the 5V power interface description: The driver board reserves a 5V power interface to facilitate program debugging and burning. The drive board provides 24V power supply through XT30, so there is no need to supply 5V power.

2.Software debugging tools

(1) The serial port is required for log output of the driver board, encoder calibration, parameter setting, etc. You can use SSCOM (http://www.daxia.com/download/sscom.rar).

(2) As shown below. USB to TTL is connected to the driver board through molex51146-3Pin interface, plug in USB to TTL to the computer. Open the SSCOM serial port debugging assistant, select the correct serial port, and set the serial port baud rate to 921600. Connect 24V power to the driver board, SSCOM serial debugging assistant can print the following content normally, you can enter the corresponding command according to the prompt, and press the ESC key of the keyboard to exit the corresponding command.

【m-Motor Mode】Enter the m command, the motor enters the FOC control mode. The control parameters will be mentioned below.

【c-Calibrate Encoder】Enter the c command, the motor will enter phase sequence and encoder calibration. After inputting the c command, the motor first performs phase sequence calibration. The motor will rotate an angle, and a prompt message will be output

when the phase sequence calibration is completed. After the phase sequence calibration is completed, the motor will enter the encoder calibration. The motor will rotate forward and reverse during the calibration process.

【 s-Setup 】 Enter the s command, the motor enters the parameter setting. Set the corresponding parameters according to the printout prompts

【 e-Display Encoder 】 Enter the e command, the serial port will output the encoder information in real time.

【z-Set Zero Position】Set the zero position of the motor

【ESC-Exit To Menu】Press the ESC key on the keyboard to return to the previous level of exit command and return to the previous level of menu

About the Source Code

Source code address: https://os.mbed.com/users/benkatz/code/Hobbyking_Cheetah/

In the debugging process, after calibrating the encoder, you can enter the motor FOC control mode through the m command. Send motor control parameters to control the motor through the CAN interface. The entire motor control consists of 5 parameters: Position Command, Velocity Command, Kp, Kd, Feed Forward Torque. The specific content of parameters can be viewed in the void unpack_cmd(CANMessage msg, ControllerStruct * controller) function of CAN_com.cpp file.

It can also switch the motor mode and calibrate the encoder by sending commands through the CAN bus. The specific CAN communication protocol can be viewed in the void onMsgReceived() function of main.cpp file.

Based on the actual test, using the MA700 magnetic encoder, the void PositionSensorAM5147::Sample(float dt) function of the PositionSensor.cpp file in the source code needs to be modified. As shown below:

```
void PositionSensorAM5147::Sample(float dt){
    GPIOA->ODR &= ~(1 << 15);
    //raw = spi->write(readAngleCmd);
    //raw &= 0x3FFF;
    raw = spi->write(0);
    raw &= 0x7FFF;
    raw = raw>>1;
    GPIOA->ODR |= (1 << 15);
    int off_1 = offset_lut[raw>>7];
    int off_2 = offset_lut[((raw>>7)+1)%128];
```

MA702 encoder version motor driver does not need to modify the driver part source code of the encode, just use it directly! !

The relationship between CAN command control parameters is as follows:

Drive reference torque = kp*(mechanical position difference) + t + kd*(mechanical speed difference)

Mechanical position difference = (P-current mechanical position of the motor)

Mechanical speed difference = (V-current mechanical speed of the motor)

By the above expression, the unit of the final calculated drive reference torque is N-m.

Among them: the unit of each parameter is as follows:

①P is the target position, the unit is radians (rad);

②V is the target speed, the unit is rad/s;

③kp is the position gain, the unit is N-m/rad;

④kd is the speed gain, the unit is N-m*s/rad;

⑤t is the torque, the unit is N-m;

5) For the CAN tools and codes for motor testing, please refer to the content mentioned in the control MIT document.

1. Question and Answer

1. Q: After the motor enters the motor mode and runs for a period of time, the motor does not respond by sending parameter control command. But the motor can respond normally when sending MotorMode and RestMode. Why?

   A: In the case of UPS, connect the serial port of the motor and connect to the computer serial debugging assistant via USB to TTL to view the motor printout content. If you see the debugging assistant prints out the content of Fault in real time, it means that the motor has a fault during operation. The reason for this fault is that the parameters of P, V, Kp, Kd and t input to the motor are unreasonable, which causes the motor to respond instantly, and the motor driver enters self-protection and prints out the fault. In this case, it can be solved by re-powering on the motor driver. If the serial port keeps inputting Fault information after power on again, it means the driver board has been damaged.