Running the motor in Modbus Mode

Version: 1.0.0

1. Introduction

Modbus gives read/write access to all Cool Muscle registers such as K-parameters, positions, speeds and accelerations. A CML program needs to reside in the motor to execute functions depending on the status of these registers. Below you will find an example program that is used in all Modbus TCP motors. It uses a control word in the R0 register to execute a number of functions. This code is useful for point-to-point motion and speed control. An application may require a significantly more complex program which can replace the example program. The CML is written to be compiled and sent from Control Room. The user does not need to understand the code but only how to use it. The full program code is supplied for those users wanting to change or get a better understanding of how the program works. Modbus TCP motors come standard with the Modbus program loaded.

2. Running the motor

The following list of holding registers can be used to the read and write move data to the motor.

Holding Register	Parameter	Description	R/W
40201	P0	Target position	R/W
40603	S0	Target speed	R/W
40635	A0	Target acceleration	R/W
40301	R0	Control Word	R/W
40003	Motor Position	Motors current position in pulses	R
40009	Motor Status	Motors current status	R

2.1. Position, Speed and Acceleration

The default setting for the motor is K37=3 which sets the following units. This can easily be changed by modifying the value of K37.

Register	Unit/Resolution
P0	1000 pulses/revolution
S0	100 pulse/s
A0	1K pulses/s^2

2.2. Control Word

The R0 register is used for the Control Word. It has the following value options by default:

R0 Value	Description
0	Do nothing
1	Start the position move
2	Stop the motor
3	Enable the motor
4	Disable the motor
5	Home the motor

⚠ These functions can be edited or expanded by changing the default program based on user requirements. Please contact our support team if you need assistance with this.

Some things to note when using the control word

1. Changing the value of the control word immediately executes the operation
2. If the Control Word is left with the value 1 then changing the position once the motor has come to a stop will execute the next move. This allows the Modbus master to only change the position and not need to also toggle the control word to execute the next move.
3. The home routine is by default set to a hardstop search in the CCW direction. Please see K42 to K48 for home routine options.

3. CML Code

The following is the CML code used for motor control in Modbus. It is not required for users to understand the code unless they are looking to change it.

CML Modbus Code

```
//set the logic scan rate to 1ms
K87.1=1
//set logic bank 1 to scan on power up
K85.1=1
//set the modbus register offset to 0
K89.1=0
//switch off all automatic motor event reporting
K23.1=0
//make sure carraige return is not automatic after line feed (legacy setting)
K70.1=0

/*create variables for the old/previous target
control word
position
speed
acceleration
These are used to find a change in the target
Init them to 0
*/
var old_ControlWord R1.1                                //old control word
R1.1=0
var old_TargetPos P1.1                //old position
P1.1=0
var old_TargetSpd S1.1                //old speed
S1.1=0
var old_TargetAcc A1.1                //old acceleration
A1.1=0

/*create variables for the new target
control word
position
speed
acceleration
These are used to find a change in the target
Init them to 0
*/
var ControlWord R0.1                                //control word
R0.1=0
var TargetPos P0.1                //position
P0.1=0
var TargetSpd S0.1                //speed
S0.1=0
var TargetAcc A0.1                //acceleration
A0.1=0

/*
Logic L1 scans for a change in the word or any target value
if a change is detected it call the relevant logic bank
*/
L1.1
ControlWord!= old_ControlWord, CL2.1, T0.1        //scan control word
TargetPos!= old_TargetPos, CL3.1, T0.1                //scan position
TargetAcc!= old_TargetAcc, CL4.1, T0.1                //scan acceleration
TargetSpd!= old_TargetSpd, CL5.1, T0.1                //scan speed
END.1
```

```
/*
Logic L2 is called if there is a change in the control word
1) it saves the new state into the old state
2) It compares the changed value with defined values to
execute the relavant command
*/
L2.1
old_ControlWord= ControlWord;
old_ControlWord== 1, ^.1, T0.1               //run
old_ControlWord== 2, ].1, T0.1               //stop
old_ControlWord== 3, (.1, T0.1               //enable
old_ControlWord== 4, ).1, T0.1               //disable
old_ControlWord== 5,|.1,T0.1               //home
END.1
/*
Logic L3 executes a change in position
If the control word eqauls 1 then it executes the move immediately
*/
L3.1
old_TargetPos= TargetPos;
ControlWord== 1, ^.1, T0.1        //execute move is ControlWord equals 1
END.1
/*
The following 2 logic banks set the speed and acceleration
Writing to the value through modbus only changes the register
it does not process the change.
The change must be processed through CML for it to be
executed immediately
*/
//Logic L4 sets the acceleration
L4.1
old_TargetAcc= TargetAcc;
TargetAcc= TargetAcc;
END.1
//Logic L4 sets the speed
L5.1
old_TargetSpd= TargetSpd;
TargetSpd= TargetSpd;
END.1
$.1
```

Download the complete Control Room project here Default Modbus CML program.crp

4. Sample

4.1. Beijer X2 Base

The following sample uses the Beijer X2 Base 7 HMI to communicate to the CM1:

SingleAxisModbus.zip

The following is the same program configured to run on the CM2 motor:

CM2_SingleAxisModbus.zip

4.2. Mitsubishi GOT1000

The following program configures a Mitsubishi GOT1000 to run the motor over Modbus RTU. The application is written in GT Designer3.

GOT1000.zip