Myostat Motion Control Inc.

Cool Muscle 1
RT3 Application Note

AN100

Document Version 1.02

COOL MUSCLE

Myostat Motion Control Inc
+1 905 836-4441
www.myostat.ca

Introduction

This application note demonstrates common application examples used in the Cool Muscle 1 (CM1) motor. It is specific to RT3 motors and generally based on firmware version RT3.12 unless otherwise specified.

Each example will have a brief description and then list the CML (Cool Muscle Language) code. The user should be somewhat familiar with the motor application software (CoolWorks Lite) and how to send data.

Contents

1. General Notes

1. Using Comments

Comments are allowed in CML code and are indicated with a single forward slash '/'. There are a few rules regarding comments:

- Comments should be on their own line
- Comments are not allowed inside program banks or logic banks.
- Comments should not exceed 40 characters from '/' to the carriage return.

Examples:

Valid Comments	Invalid Comments
/This comment is not more /than 40 characters long /P1 sets the bottom position P1.1=1000 S1.1=10 A1.1=5	/This comment is more than 40 characters long P1.1=1000 /P1 sets the bottom position S1.1=10 A1.1=5
/Bank1 will move to the bottom position B1.1 A1.1,S1.1,P1.1 END.1	B1.1 /set acc, speed and position A1.1,S1.1,P1.1 /move to the bottom position END.1

2. Calculating Speed (pulses/sec ←→RPM)

Speed values (S0, S1, etc.) are set in the motor in pulses/second with a speed unit defined in K37. Speed units are 100p/s, 10p/s or 1p/s depending on the K37 value.

The general calculation to from pulses/second to RPM is:

$$RPM\left(\frac{revolution}{minute}\right) = \left(SPEED \times SpeedUnit\left(\frac{pulses}{second}\right)\right) \times 60\left(\frac{second}{minute}\right) \div RESOLUTION\left(\frac{pulses}{revolution}\right)$$

$$SPEED = \ RPM\left(\frac{revolution}{minute}\right) \div 60\left(\frac{second}{minute}\right) \times RESOLUTION\left(\frac{pulses}{revolution}\right) \div SpeedUnit\left(\frac{pulses}{second}\right) ($$

Example 1 (RPM→ SPEED):

K37=3 which defines a speed unit of 100 pulses/second and a resolution 1000pulses/revolution. If we want to run at 120 rpm this gives is an S value:

$$SPEED = \ RPM\left(\frac{revolution}{minute}\right) \div 60\left(\frac{second}{minute}\right) \times RESOLUTION\left(\frac{pulses}{revolution}\right) \div SpeedUnit\left(\frac{pulses}{second}\right)$$

$$SPEED = 120 \div 60\left(\frac{second}{minute}\right) \times 1000\left(\frac{pulses}{revolution}\right) \div 100\left(\frac{pulses}{second}\right)$$

$$SPEED = 120 \div 6$$

$$SPEED = 20$$

So for the case of K37=3 you can easily always divide by six to get RPM. E.g. S = 1500rpm/6 = 250.

Example 2 (SPEED→RPM):

K37=30 which defines a speed unit of 10 pulses/second and a resolution of 50000 pulses/revolution. If we have set an S0=33500 what is the RPM equivalent?

$$RPM\left(\frac{revolution}{minute}\right) = \left(SPEED \times SpeedUnit\left(\frac{pulses}{second}\right)\right) \times 60\left(\frac{second}{minute}\right) \div RESOLUTION\left(\frac{pulses}{revolution}\right)$$

$$RPM\left(\frac{revolution}{minute}\right) = \left(33500 \times 10\left(\frac{pulses}{second}\right)\right) \times 60\left(\frac{second}{minute}\right) \div 50000\left(\frac{pulses}{revolution}\right)$$

$$RPM\left(\frac{revolution}{minute}\right) = 402$$

3. Motor ID

Motors on a network auto ID themselves according to their position on the network. A single motor will always be assigned ID 1.

When a motor is powered up you will see the ID in the power up information.

E.g: “ID1 :CM1v3.12CH.5 #01236”

In this case the motor has ID1, firmware version 3.12 and hardware version 5. It has an internal serial number 01236.

The ID is assigned during programming by appending a “.ID” at the end of the command or register name.

Example CML Code:

/sets motor 1 position 1 to 1000.
P1.1=1000
/sets motor 2 speed 3 to 56
S3.1=56

/write a basic bank in motor 1
B1.1
A1.1,S1.1, P1.1
END.1

/write a basic bank in motor 2 calling motor 3.
/Note that the ID is set back on the END.2 and not assumed to be 2.
B1.2
A1.3,S1.3,P1.3
END.2

/set output 2 on motor 1 on
O2.1
/set output 2 on motor 5 off
F2.5

It is good practice to always append the ID on the end of all commands however it is not necessary. Once a motor ID has been called it will be the default until it is changed.

4. Switching off RS485 mode – '{#'

K62 sets the RS485 node ID. If K62=0 RS485 is off. If K62 is set to any other number that number is the node ID.

The motor is in RS485 slave mode if it is streaming '{' with a number following it on power up. The motor is requesting that it have communication access as it has something to report. To switch back to standard serial we need to:

1) give the motor communication access,
2) let it report its message,
3) gain communication access again
4) set K62=0

The process to do this is the following

1) Send the motor bank its request. I.e if the motor is streaming "{1" send back "{1". If it is send "{56" send back "{56".
2) At this point the motor will send some information (power up ID position, etc) and then "{0" to close communication.
3) To gain bank communication access we send the RS485 ID again. I.e. "{1" or "{56" as in the above examples.
4) Send "K62=0" to switch off RS485.

If there is a logic bank or program bank running that is reporting information constantly timing is critical. As soon as the {0 is received from the motor we must send bank {ID. If we don't and the motor has more to report it will start streaming the communication request again.

2. Direct Mode

1. Basic Point to Point Move

The basic move structure in CML is to define a position (P0), speed (S0), acceleration (A0) and optionally a torque (M0). These 4 values form the core of a basic direct mode move. They are set in read/write registers. P0 is always in absolute coordinates and all values are with respect to the current motor resolution. In this example we assume it to be default (K37=3) which defines that the motor has a resolution of 1000 pulses/rev and a speed unit of 100 pulses/sec.

CML Code Used:

P0 → position in pulses in absolute coordinates
S0 → speed in pulses/sec (x 100 as per K37).
A0 → acceleration in kilopulses/sec^2
M0 → torque limit in % of peak torque

^ → execute the direct move

Example CML Code:

```
/set the direct mode registers
P0.1=10000
S0.1=10
A0.1=5

/execute the absolute move
^.1
```

2. Speed control

Speed Control is set by setting P0=1000000000 (1 billion, 1 and 9 0's). To start the motion execute the dynamic move with the '^'. S0, A0 and M0 can now be changed at any time to adjust motor speed, acceleration and torque while the motor is moving.

CML Code Used:

P0 → position set to 1 billion
S0 → speed in pulses/sec (x 100 as per K37).
A0 → acceleration in kilopulses/sec^2
M0 → torque limit in % of peak torque

^ → start the speed control

Example CML Code:

```
/set the direct mode registers
P0.1=1000000000
S0.1=10
A0.1=5
M0.1=100

/execute the absolute move
^.1
```

/change the speed as required
S0.1=100

Myostat Motion Control Inc
+1 905 836-4441
www.myostat.ca

3. Program Banks

1. Basic Program Bank

This example shows how to write a very basic program bank that moves the motor to position 10000 pulses and back to 0.

1) Register values are assigned
2) Program bank 1 in motor 1 (B1.1) is created
3) Bank 1 is executed

CML Code Used:

P1, P2 → position values
S1 → assigned speed
A1 → assigned acceleration

B1 → start of bank1 programming
END → end of bank programming

[1 →execute bank1. The '[' character indicates starting the bank. The number following it indicates which bank is executed

Example CML Code:

```
/initialize the registers
P1.1=10000
P2.1=0
S1.1=50
A1.1=10

/program the bank
B1.1
A1.1,S1.1,P1.1
P2.1
END.1

/execute the bank
[1.1
```

Notes:

1) Inside the bank a speed and acceleration are always called at least once before the 1st position. This ensures a known speed and acceleration is set before a position is executed.
2) All commands on the same line are executed together. Commands on different lines are executed sequentially once the previous line is complete. In this example P1 will complete before P2 is executed.
3) Unless and M value (torque) is called the M0 value is used as the torque. By default this is 100 (%).

2. Starting a Program Bank on Power-Up

This example shows how to start a program bank on power up. It uses both logic banks and program banks. Essentially only a logic bank can be started on power up, so, a logic bank is used to execute a program bank.

In this example the motor will start to run continuously clockwise immediately on power up.

CML Code Used:

P1=1000000000 → position 1 register. If this is set to 1 billion the motor is in speed control.
A1=100 → acceleration 1 register.
S1=30 → speed 1 register.

B1 → beginning of program bank

L1 → beginning of logic bank 1
END → end of logic bank

[L1 → execute/run logic bank 1
]L → stop the logic bank

K87=1 → logic bank execution time in milliseconds.
K85=1 → logic bank number started on power up

Example CML Code:

```
/set the logic scan time
K87=1
/set logic bank on power up
K85=1

/set pos, spd and accel
P1=1000000000
S1=30
A1=100

/Program Bank 1
/running speed control
B1
A1,S1,P1
END

/Logic Bank 1 on power up
/executes program bank 1
/then jumps to Logic 2 to do nothing
L1
[1.1
END

L2.1
END
```

3. Running 2 motors in a program bank

This example runs two motors from a program bank in motor 1. All position moves are absolute. The move sequence is as follows:

1) Move motor 1 to 10000 pulses
2) Move motor 2 to -5000 pulses
3) Move motor 1 to 0 pulses
4) Move motor 2 to 0 pulses

When executing a move for the first time a speed and acceleration must be set for that motor. Careful note should also be taken with the motor ID. Note that the last position called is P2.2. When the END of the bank is called it must have END.1 or it will be associated with motor 2 and will not load correctly.

CML Code Used:

P1.1=10000 → position 1 motor 1 register.
P2.1=0 → position 2 motor 1 register
A1.1=10 → acceleration 1 motor 1 register.
S1.1=30 → speed 1 motor 1 register.

P1.2=5000 → position 1 motor 2 register.
P2.2=0 → position 2 motor 1 register
A1.2=20 → acceleration 1 motor 1 register.
S1.2=10 → speed 1 motor 1 register.

B1.1 → beginning of program bank 1 in motor 1
END.1 → end of program bank 1 in motor 1

[1.1 →start bank 1 in motor 1

Example CML Code:

```
/load motor 1 registers
P1.1=10000
P2.1=0
A1.1=10
S1.1=30

/load motor 2 registers
P1.2=5000
P2.2=0
A1.2=20
S1.2=10

/load bank 1 in motor 1
/note A and S are called before P
B1.1
A1.1,S1.1,P1.1
A1.2,S1.2,P1.2
P2.1
P2.2
END.1

/start the bank 1 motor 1
[1.1
```

4. Logic Banks

1. Basic Logic Bank

This example shows how we can write a very basic logic bank and execute it. This logic bank does a simple mathematic calculation. If for example the motor is connected to a ballscrew actuator that had a 2mm pitch (2mm/rotation), the motor resolution is 1000 pulses per revolution and you want to have a variable that always changes a position from millimeters to pulses.

CML Code Used:

P1 → calculated position in pulses
P2 → entered position in mm
V1 → screw pitch
V2 → motor resolution

L1 → beginning of logic bank 1
END → end of logic bank

[L1 → execute/run logic bank 1
]L → stop the logic bank

K87 → logic bank execution time in milliseconds.

Example CML Code:

```
/set the scan time
K87=200

/set default position to 0
P1=0

/position in mm
/can be changed at will
P2=50

/screw pitch
V1=6
/motor resolution
V2=1000

/calculate the pulses
L1
P1=P2*V2/V6
END

/start the logic bank
[L1
```

Notes:

1) Once started the equation will return P1 on the serial port. To stop this and only see the value if queried we can put a semi-colon ';' at the end of the line. This stops the logic bank from displaying the result. I.e. P1=P2*V2/V6;
2) The logic bank can be stopped by sending]L.

2. Starting and Stopping Logic Banks

A logic bank is started with the "[L#" command, where '#' is the logic bank number, and stopped with the "]L" command.

This example application sets V2 equal to the motor position.

CML Code Used:

V1="Px" → internal motor position state
V2 → value to be set to motor position

L2 → beginning of logic bank 2
END → end of logic bank

[L2→ execute/run logic bank 2
]L → stop the logic bank

K87 → logic bank execution time in milliseconds.

Example CML Code:

```
/set the scan time
K87.1=200

/internal motor position
V1="Px"
/variable to save the position
V2=0

/set V2 to position
L2
V2=V1
END

/start the logic bank
[L2.1
```

Notes:

1) Once the logic bank is executing it will continue to run in the bank ground. In this application example it will execute every 200ms.
2) To stop the logic bank send "]L".

3. Starting a Logic Bank on Power-up

K85 sets the startup/power-up logic bank. E.g. Set K85=1 to run logic bank 1 on power-up.

E.g.

1) Set K85=1
2) Run the Basic Logic Bank example.

4. 'if – then – else' statement in a logic bank

Logical statements can be used with variables, inputs and most other registers inside logic and program banks. This example looks at the status of input 3 and will start or stop the motor depending on if it is on or off.

Logical statement overview:

LOGIC, TRUE, FALSE

E.g. (with V1=0)

CML Code	C Equivalent

I3==V1,].1,^.1	if (IN3==0)
	{
	].1 //stop the motor
	}
	else
	{
	^.1 //start the motor
	}

CML Code Used:

V1.1=0→ variable set to 0 to compare to Input 3
I3.1 → Input 3 bank status
].1→ CML command to stop a motor
^.1 → CML command to start a direct mode move
P0.1=1000000000 → set direct move position to 1 billion for speed control
S0.1=50→ set direct move speed to 50
A0.1=10 →set direct move accel to 10

L1.1→ beginning of logic bank 2
END.1 → end of logic bank

[L1.1→ execute/run logic bank 2

K87.1=1 → logic bank execution time to 1 millisecond.

Example CML Code:

```
/set the scan time
K87.1=1

/internal motor position
V1.1=0

/set direct mode variables
P0.1=1000000000
S0.1=50
A0.1=10

/set V2 to position
L1.1
I3.1==V1.1,].1,^.1
END.1
```

/start the logic bank
[L1.1

Notes:

1) Like most logical statements the compared value can be implied. I.e. you can say I3.1,].1,^.1
2) Logical statements can be used in program and logic banks.

5. Inputs and Outputs

1. Speed Control Using Multiple Digital Inputs as Binary Control

This example uses inputs 1-4 on the motor as digital inputs to set a speed. It uses a binary combination of the inputs to calculate 16 different set-points. All calculations and executions are done in three logic banks. The 1st logic bank calculates the binary combination. If it has changed the 2nd logic bank decides if the motor should be stopped or a speed is set. The 3rd logic bank sets required speed.

CML Code Used:

.1 → sending just the motor ID allows all following commands to be sent to that ID.

K87=5 → 5ms logic bank scan time
K85=1 → Logic bank 1 is started on power-up

P0=1000000000 → set direct mode to speed control
S0=0 → default speed is 0
A0=10 → default acceleration is 10

N10-N25 → N registers define constants to compare the binary combination too.
S1-S15 → S registers set the speed options for the binary combination

V0=0 → used to calculate and hold the binary combination
V1=0 → temporary variable used in calculations

V2=2 → binary multiplier for IN2
V4=4 → binary multiplier for IN3
B8=8 → binary multiplier for IN4

L1-L3 → indicated the start of a logic bank
END → indicates the end of a logic bank

$ → save the data to the eeprom to it is retained on a power cycle

Example CML Code:

```
/set ID to motor 1
.1

/set logic bank scan time
/and startup bank
K87=5
K85=1

/clear program banks
/and logic banks
B100
L100

/set P0 to speed control
/default speed is 0
P0=1000000000
S0=0
A0=10
```

```
/assign N values to be
/used as constants to compare
/to the input combination
N10=0
N11=1
N12=2
N13=3
N14=4
N15=5
N16=6
N17=7
N18=8
N19=9
N20=10
N21=11
N22=12
N23=13
N24=14
N25=15

/define speeds
S1=10
S2=20
S3=30
S4=40
S5=50
S6=60
S7=70
S8=80
S9=90
S10=100
S11=110
S12=120
S13=130
S14=140
S15=150

/set default values for variables
V0=0
V1=0
V3=0
V2=2
V4=4
V8=8

/scan inputs and calculate
/binary combination
/if it has changed call logic 2
L1.1
V0=I1;
V1=I2*V2;
V0=V0+V1;
V1=I3*V4;
V0=V0+V1;
V1=I4*V8;
```

```
V0=V0+V1;
V0!=V3, CL2,T0
END.1

/depending on combination
/stop the motor or set a speed
/by calling L3
L2.1
V3=V0;
V0==N10,],CL3
END.1

/set the speed and start the motor
L3.1
V0==N11,S0=S1,T0
V0==N12,S0=S2,T0
V0==N13,S0=S3,T0
V0==N14,S0=S4,T0
V0==N15,S0=S5,T0
V0==N16,S0=S6,T0
V0==N17,S0=S7,T0
V0==N18,S0=S8,T0
V0==N19,S0=S9,T0
V0==N20,S0=S10,T0
V0==N21,S0=S11,T0
V0==N22,S0=S12,T0
V0==N23,S0=S13,T0
V0==N24,S0=S14,T0
V0==N25,S0=S15,T0
^.1
END.1

/save data to eeprom
$.1
```