Fisher & Paykel / Cherokee Washer Motor Controller Software Specifications

Version 1.0

By Jacky Cheung

Last Update: 30/10/2015 JT

Table of Contents

1.0 General Operation

1.1 Overview

The F&P washer motor controller accepts two groups of command – control commands and feedback commands. The control commands are used for specifying the control parameters and the behaviour of the motor. The feedback commands allow access to the status of the controller and the motor, which can be used for fault detection, sensing purposes or getting feedback from the available peripherals etc.

Figure 1-1 shows the different parts of a motor profile. All wash action, sensing strokes and spin actions can be generated by specifying different motor profiles in sequence.

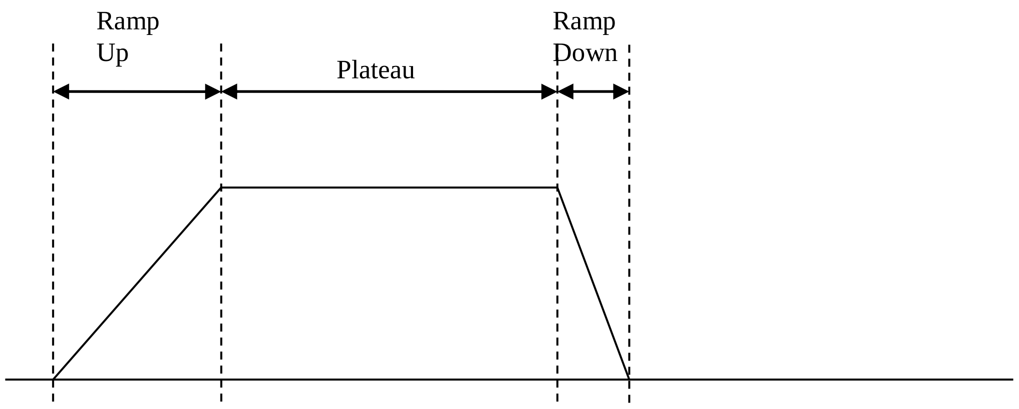

Figure 1-1 Motor Motion Profile

1.2 Running the Motor

This section describes the basic steps for starting and stopping the motor. Spin and agitate action are discussed here to illustrate how different actions can be achieved. Other actions can be generated in similar ways by sequencing the motor through different spin profiles.

Appendix C shows some sample sequences for sense, agitate and spin.

1.2.1 Spin

Spin can be started by using the ***PS*** command with the number of cycles set to 1 to set up the motion profile and then issue a ***GS*** command. After the **GS** command is issued, the motor starts accelerating towards the target speed at a rate specified in the ***PR*** command (see 'Ramp Up' in Figure 1-1). Once the target speed is reached, the motor stays at the target speed (see 'Plateau' in Figure 1-1) until the plateau time has expired. The Ramp down profile is not controlled and the controller simply stops driving the motor. The ramp down rate achieved depends upon the system dynamics.

1.2.2 Agitate

Agitate can be achieved by repeating the spin profiles in the clockwise and counter-clockwise directions. Figure 1-2 illustrates how to generate an agitate action.

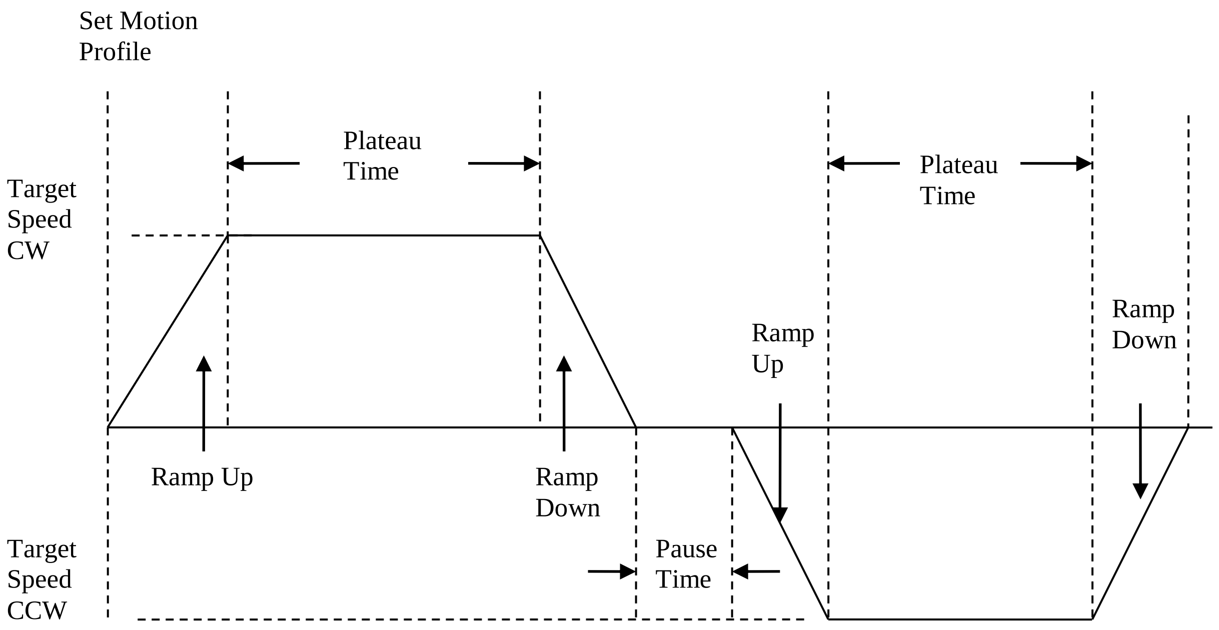

Figure 1-2 Generating Agitate

After motion profile is set (and the **GA** command issued), the motor starts accelerating towards Target Speed. It stays at that speed until the plateau time has expired, at which point the motor starts decelerating to 0RPM. After the motor is stopped and the pause time has expired, it accelerates in the other direction with the same profile and time. The number of agitate cycles is set by NumberOfCycles within the GA command

2.0 Communication

2.1 Serial Communication Protocol

The communication to the motor controller is as follows:

- Baud Rate: 4800 baud
- Data Bits: 8
- Stop Bits: 1
- Parity: None

Timeout: 10Sec(Go idle state if no valid comms message received in 10Sec)

In this document **[CS]** refers to the data checksum, and **[CR]** denotes a carriage return 0D hexadecimal (0Dh).

The checksum is the sum of all the bytes in the message prior to the checksum expressed in ASCII format. For example, if the checksum is 5Dh this will be sent as 35h 44h as shown in the example below:

Example:
To enable checksum checking on the controller requires the following message:

```
>CS,1,5D[CR]
```

in ASCII the message would be expressed as:

```
3E 43 53 2C 31 2C 35 44 0D
 >  C  S  ,  1  ,  5  D [CR]
```

All messages from the main controller (master) are preceded by **'>'** (3Eh), and all replies from the motor controller are preceded by **'<'** (3Ch).

In the event of an error in processing the message, for example a parameter out of range, the reply will include the string **ERR.**

Example:

```
>GA,3,543,[CS][CR]
```

Reply:

```
<GA,ERR,[CS][CR]
```

2.2 Software Commands

2.2.1 Control Commands

CS – checksum on/off

The **CS** command enables/disables Checksum checking in the Motor Controller.

- Syntax: >**CS,ChecksumEnable,[CS][CR]**
- Reply: <CS,[CS][CR]
- Range: ChecksumEnable = 0 Checksum disabled
  ChecksumEnable = 1 Checksum enabled
- Default: 1 (Checksum Enabled)

GA – move using profile agitate parameters

The **GA** command instructs the motor to move using the profile agitate (**PA**) parameters. The **Direction** parameter specifies the start direction, Clockwise (CW) or Counter-clockwise (CCW). **NumOfCycles** specifies the maximum running time for the motion profile.

- Syntax: **>GA,Direction,MaxRunTime,[CS][CR]**
- Reply: <GA,[CS][CR]
- Range: Direction = 0 CW
  Direction = 1 CCW (not implemented)
  MaxRunTime = 1 – 65535 in 1S increments (0 – no limit)

GS – move using profile spin parameters

The **GS** command instructs the motor to move using the profile (**PS**) parameters. The **Direction** parameter specifies the spin direction, Clockwise (CW) or Counter-clockwise (CCW). **NumOfCycles** specifies the maximum running time for the motion profile.

- Syntax: **>GS,Direction,MaxRunTime,[CS][CR]**
- Reply: <GS,[CS][CR]
- Range: Direction = 0 CW
  Direction = 1 CCW
  MaxRunTime = 1 – 65535 in 1S increments (0 – no limit)

ILM – set current limit

The **ILM** command sets the motor current limit in 100mA increments. The current limit will be reset back to the default value after motor stop.

- Syntax: **>ILM,CurrentLimit,[CS][CR]**
- Reply: <ILM,[CS][CR]
- Range: CurrentLimit = 0 – 50 A/10 (5A max, 0 -use the default value)
- Default: CurrentLimit = 0 (Default value – 2.8A)

PA – set agitate motion profile

The **PA** command is used to specify the motion profile for the motor, including target speed, ramp up rate and pause time between CW and CCW strokes. Note that this command only loads the profile to the memory of the controller, but does not instruct the controller to use it. The **GA** command must be used after this command to instruct the controller to start using these new parameters.

Although the controller will try and follow the profile, the actual dynamic response will depend on the load it is driving.

- Syntax:

>PA,RampUpTime,TargetSpeed,PlateauTime,PauseTime,[CS][CR]

- Reply: <PA,[CA][CR]
- Range: RampUpTime = 1 – 600 in 1ms increments
  TargetSpeed = 0 - 200 rpm
  PlateauTime = 0 - 1800 in 10ms increment
  PauseTime = 0 - 1800 in 10ms increments
- Default: RampUpRate = 550
  TargetSpeed = 38
  PlateauTime = 80
  PauseTime = 0

PS – set spin motion profile

The **PS** command is used to specify the motion profile for the motor, including target spin speed, direction and ramp up rate. Note that this command only loads the profile to the memory of the controller, but does not instruct the controller to use it. The **GS** command must be used after this command to instruct the controller to start using these new parameters.

Although the controller will try and follow the profile, the actual dynamic response will depend on the load it is driving.

- Syntax: **>PS,RampUpRate,TargetSpeed,[CS][CR]**
- Reply: <PS,[CA][CR]
- Range: RampUpRate = 10 – 1000 (10-slowest, 1000-max)
  TargetSpeed = 23 to 1600 rpm
- Default: RampUpRate = 10
  TargetSpeed = 23

RSF – reset fault flags

The **RSF** command resets all the fault flags in the controller to 0

- Syntax: **>RSF,[CS][CR]**
- Reply: <RSF,[CS][CR]

SNS – set sense profile

The **SNS** Profile is used to set the profile for sensing. The sense is performed using a fixed rotation and torque, and the time taken can be used to determine clutch engagement.

- Syntax:
  >SNS,MaxSpeed,MaxRotation,%Torque,MaxRunTime,[CS][CR]
- Reply: <SNS,[CS][CR]
- Range: MaxSpeed = 0 – 100 rpm
  MaxRotation = 0 - 90 degrees
  %Torque = 0 – 100 (not implemented)
  MaxRunTime = 1 – 65535 in 1S increments (0 – no limit)

STP – stop motion

The **STP** command is used to stop the motor immediately.

- Syntax: **>STP,[CS][CR]**
- Reply <STP,[CS][CR]

MR – Read the memory value

The **MR** command is used to read the value of a memory address. The memory allocation table is shown below

```
/* 0 */   &motor_fb.current_hall_pattern,
/* 1 */   &global_wv.raw_adc.motorIA8,
/* 2 */   &global_wv.raw_adc.motorIB8,
/* 3 */   &global_wv.raw_adc.motorIC8,
/* 4 */   &global_wv.raw_adc.ad_check_2,
/* 5 */   &global_wv.raw_adc.digDisable,
/* 6 */   mtc_U16ToU8Lsb(global_wv.rom_checksum),
/* 7 */   mtc_U16ToU8Msb(global_wv.rom_checksum)
```

- Syntax:

>MR,Address,[CS][CR]

- Reply: `<MR,CurrentValue,[CA][CR]`
- Range: Address = 0 - 6

GF – Move with friction test profile

The **GF** command is used to initiate the friction test sequence. The friction test is used to determine the total friction level of the assembly. It uses the coast down time to determine the friction, the bowl mass should be consistent in this case. This command specifies the start speed and end speed for the test. The **FR?** command can be used after this command to read the friction test result. The maximum time duration for the test is 10 seconds, after this time duration, if the motor still has not reached the end speed, the motor will be forced to stop.

- Syntax:

>GF,StartSpeed,EndSpeed,[CS][CR]

- Reply: `<GF,[CA][CR]`
- Range:
  StartSpeed = 0 - 200 rpm
  EndSpeed = 0 - 200 rpm
  StartSpeed > EndSpeed

GB – Move with Bridge test profile

The **GB** command is used to set the state for the output switchers for controller test. It sets the output pattern number, current and PWM limit. Table below list the output pattern number and actual switch state.

1: A+ C-

2: B+ A-

3: B+ C-

4: C+ B-

5: A+ B-

6: C+ A-

- Syntax:

>GB,PatternNumber,CurrentLimit,PwmLimit,[CS][CR]

- Reply: <GB,[CA][CR]
- Range: PatternNumber = 1 - 6
  CurrentLimit = 1 – 50(0.1 – 5A)
  PwmLimitLimit = 1 – 100(in percent)

GT – Move with Stepper profile

The **GT** command is used to initiate the stepper sequence. The stepper sequence step through 6 steps and go idle state.

- Syntax:

>GT,[CS][CR]

- Reply: <GT,[CA][CR]

SMN – Set motor number

The **SMN** command is used to set the motor type. Table below list the motor type number.

0: 6Kg Motor

1: 7/8Kg Motor

...

- Syntax:

>SMN,MotorNumber,[CS][CR]

- Reply: <SMN,[CA][CR]
- Range: MotorNumber = 0 – (Available motors - 1)
- Default: MotorNumber = 0

2.2.2 Feedback Commands

BT? – read motor bridge temperature

The **BT?** command returns the temperature of the motor bridge.

- Syntax: **>BT?,[CS][CR]**
- Reply: <BT?,BridgeTemp,[CS][CR]
- Range: BridgeTemp 0 – 255 degC

F? – read fault flags

The **F?** Command returns the Fault Flags.

- Syntax: **>F?,[CS][CR]**
- Reply: <F?,FaultFlags,[CS][CR]
- Range: FaultFlags= 16bit (see Appendix B)

HV? – read hvdc

The **HV?** Command returns the High Voltage DC (HVDC) measurement.

- Syntax: **>HV?,[CS][CR]**
- Reply: <HV?,HighVoltageDC,[CS][CR]
- Range: HighVoltageDC= 0 – 400 VDC

I? – read motor current

The **I?** command reads the instantaneous motor current.

- Syntax: **>I?,[CS][CR]**
- Reply: <I?,MotorCurrent,[CS][CR]
- Range: MotorCurrent= 0 – 10000 (10A, 1mA/Count)

IA? – read motor average current

The **IA?** command reads the average motor current(2sec average).

- Syntax: **>IA?,[CS][CR]**
- Reply: <IA?,MotorCurrent,[CS][CR]
- Range: MotorCurrent= 0 – 10000 (10A, 1mA/Count)

IS? – read interlock status

The **IS?** Command returns the current state of the interlock input. The motor will only run if the interlock is High (1)

- Syntax: **>IS?,[CS][CR]**
- Reply: <IS?,InterlockStatus,[CS][CR]
- Range: High=1, Low=0

MS? – read motor status

The **MS?** command returns the current status of the motor.

- Syntax: **>MS?,[CS][CR]**
- Reply: <MS?,MotorStatus,[CS][CR}
- Range:

MotorStatus=0	Idle
MotorStatus=1	Spin
MotorStatus=2	Agitate
MotorStatus=3	Sensing
MotorStatus=4	Coasting Down
MotorStatus=5	Rail Pump
MotorStatus=6	Stepper
MotorStatus=7	Bridge Test
MotorStatus=8	Friction Test
MotorStatus=20	Other state

MT? – read motor temperature

The **MT?** command returns the temperature of the thermistor on the rotor position sensor (RPS) PCB.

- Syntax: **>MT?,[CS][CR]**
- Reply: <MT?,Motortemp,[CS][CR}
- Range: BridgeTemp= 0 to 255 degC

SP? – read motor speed

The **SP?** Command returns the current motor speed.

- Syntax: **>SP?,[CS][CR]**
- Reply: <SP?,MotorSpeed,[CS][CR]
- Range: MotorSpeed= 0 to 1000 rpm

SR? – read sense profile time

The **SR?** command returns the most recent sense profile times. A Clockwise and Counter-clockwise value is returned.

- Syntax: **>SR?,[CS][CR]**
- Reply: <SR?,SenseTimeCW,SenseTimeCCW,[CS][CR]
- Range: SenseTimeCW= 0 – 65535 ms
  SenseTimeCCW= 0 – 65535 ms

SW? – read software version

The **SW?** command returns the motor controller software version. The value is returned as a 16-bit number, the first byte is the major version number, the second byte the minor version number.

- Syntax: **>SW?,[CS][CR]**

- Reply: <SW?,MajorVersion,MinorVersion,[CS][CR]
- Range: *MajorVersion* (range: 0 – 255)
  MinorVersion (range: 0 – 255)

FR? – read friction test result

The **FR?** Command returns the result(time duration) for the last friction test

- Syntax: **>FR?,[CS][CR]**
- Reply: <FR?,CoastDownTime,[CS][CR]
- Range: CoastDownTime = 0 to 10000 ms

PW? – read current PWM

The **PW?** Command returns the current output PWM value in percentage.

- Syntax: **>PW?,[CS][CR]**
- Reply: <PW?,CurrentPwm,[CS][CR]
- Range: CurrentPwm= 0 to 100

MN? – read current motor number

The **MN?** Command returns the current motor type number.

- Syntax: **>MN?,[CS][CR]**
- Reply: <MN?,CurrentMotorNumber,[CS][CR]
- Range: CurrentMotorNumber= 0 to (Available motors – 1)

3.0 Appendix A – Summary of Commands

3.1 Control Commands

CS	-	Checksum
GA	-	Go Agitate
GS	-	Go Spin
ILM	-	Current Limit
PA	-	Profile Agitate
PS	-	Profile Spin
RSF	-	Reset Fault Register
SNS	-	Sense profile
STP	-	Stop
GB	-	Go Bridge Test
GF	-	Go Friction Test
MR	-	Read Memory
GT	-	Go Stepper
SMN	-	Set Motor Number

3.2 Feedback Commands

BT?	-	Read Motor Bridge Temperature
F?	-	Read Fault Register
HV?	-	Read HVDC
I?	-	Read Instantaneous Motor Current
IA?	-	Read Average Motor Current
IS?	-	Read Interlock Status
MS?	-	Read Motor Status
MT?	-	Read Motor Temperature
SP?	-	Read Motor Speed
SR?	-	Read Sense Profile Register
SW?	-	Read Software Version
FR?	-	Read Friction Result
PW?	-	Read Current Pwm Value
MN?	-	Read Current Motor Number

4.0 Appendix B – Fault Flags

The fault flags are contained in a 16-bit status register as show below:

Bit#	**Type**	**Description**
0	MotorStall	Motor stall
1	MotorLossOfPhase	One or more stator windings open circuit
2	Motor_Overcurrent	Motor current over set ILM (usually due to motor bridge failure)
3	HallSensorOutOfSequence	Invalid hall sequence detected
4	SingleHallError	Invalid hall pattern
5	CurrentSenseError	Current sense malfunction
6	SelfCheckError	MCU failed self check
7	Brownout	HVDC too low Limit: 80v HVDC for agitate. 160v HVDC for spin.
8	MotorOverTemp	Stator winding over limit
9	BridgeOverTemp	IPM over temperature
10	BowlReengagedDuringAgitate	Detect clutch reengaged, can be caused by insufficient water volume
11	FunctionTimedOut	Reached max run time
12	CommTimeoutOccured	Comm timeout occurred, force motor stop
13	LossInterlockDuringRunning	Interlock signal lost during the motor running
14		
15	OtherFault	

5.0 Example Commands

5.1 Sensing

The **SNS** command is used to sense if the clutch is engaged. The values returned will depend on the system.

```
Example where:
               MaxSpeed=60rpm
               MaxRotation=20degrees
               %Torque=30%
               NumOfCycles=2
               Checksum= 6Bh

Message:  >SNS,60,20,30,2,6B[CR]

          3E 53 4E 53 2C 36 30 2C 32 30 2C 33 30 2C 32 2C 36
42 0D

Reply:         <SNS,5C[CR]

          3C 53 4E 53 2C 35 43 0D
```

The sense profile response is read using the **SR?** command:

```
Message:  >SR?,4E[CR]

          3E 53 52 3F 2C 34 45 0D

Reply:         <SR?,500,500,CE[CR]

          3C 53 52 3F 2C 35 30 30 2C 35 30 30 2C 43 45 0D
```

5.2 Agitate Cycle

In this example the profile is set up using the **PA** command:

```
Message:  >PA,100,100,30,20,92[CR]

3E 50 41 2C 31 30 30 2C 31 30 30 2C 33 30 2C 32 30 2C 39 32 0D

Reply:         <PA,FB[CR]

          3E 50 41 2C 46 42 0D
```

The **GA** command is then used to start the agitate cycle:

```
Message: >GA,0,1200,3D[CR]

3E 47 41 2C 30 2C 31 32 30 30 2C 33 44 0D

Reply: <GA,F0[CR]

3C 47 41 2C 46 30 0D
```

5.3 Spin Cycle

In this example the profile is set up using the **PS** command:

```
Message:  >PS,10,500,5B[CR]

3E 50 53 2C 31 30 2C 35 30 30 2C 35 42 0D


Reply:         <PR,0A [CR]

          3E 50 53 2C 30 44 0D
```

The **GS** command is then used to start the spin cycle:

```
Message:  >GS,1,1,BE[CR]

3E 47 53 2C 31 2C 31 2C 42 45 0D

Reply:    <GS,04[CR]

3E 47 53 2C 30 34 0D
```