9/2/2010

RSS-NIR
Filter/Grating Inserter Motor Drive
Initialization Scripts

Refer to the Magellan Motion Processor Users Guide sections 12.3.6 and 12.3.7 for information regarding the timing requirements for packets to be recognized as complete.

The structure of the command packets are explained in section 12.3.2 of the Magellan Motion Processor Users Guide and summarized for RSS-NIR use here:

Address – 1 byte – Always 0x00 when in "point to point mode"

Checksum – 1 byte - two's complement of the sum of the other bytes in the packet

Axis – 1 byte – Always 0x00 for RSS-NIR

Instruction – 1byte

Data – 0-6 bytes

The checksum is explained in section 12.3.4 of the Magellan Motion Processor Users Guide and copied here:

> The serial checksum is calculated by summing all bytes in the packet (not including the checksum) and negating (i.e., taking the two's complement of) the result. The lower eight bits of this value are used as the checksum. To check for a valid checksum, all bytes of a packet should be summed (including the checksum byte), and if the lower eight bits of the result are zero, then the checksum is valid.
>
> For example, if a command packet is sent to motion processor address 3, containing command 0177h (**SetMotorCommand** for axis 2) with the one-word data value 1234h, then the checksum will be calculated by summing all bytes of the command packet (03h + 01h + 77h + 12h + 34h = C1) and negating this to find the checksum value (3Fh). On receipt, the motion processor will sum all bytes of the packet, and if the lower eight bits of the result are zero, then it will accept the packet (03h + 3Fh + 01h + 77h + 12h + 34h = 100h).

The structure of the response packets are explained in section 12.3.2 of the Magellan Motion Processor Users Guide and summarized for RSS-NIR use here:

Address – 1 byte

Status – 1 byte – 0x00 if command was interpreted correctly as valid

Checksum – 1 byte – two's complement of the sum of the other bytes in the packet

Data – 0-6 bytes

If the status field returns non-zero, the value returned is an instruction error. Refer to the Magellan Motion Processor Users Guide, section 12.2.5 to decode the nature of the instruction error. The command **GetInstructionError** will also return the instruction error but will clear the instruction error fault bit as well.

Communications Initialization

To initialize the serial port mode in the drive, send the commands listed below and verify the responses.

1. **NOP** – to verify physical communications link

   Command Packet

Address	0x??	Address of ION motor drive.
Checksum	0x00	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x00	Command = GetVersion

   Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet

   a. Timeout Exception - Drive may be out of synchronism with host packet timing. Wait 2 ms and transmit another 0x00. Repeat for 10 retries checking for a response packet each time.

   b. Retry Exception - Display an error message and bail. The drive may be communicating in Point to Point Mode. This is the case if the drive is new out of the box. The drive needs to be setup over an RS-232 link using the manufacturer's canned software.

2. **GetVersion** – to verify motor drive device type and firmware version

   Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x8F	Command = GetVersion

Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet.
Data 1	0x99	Product Family = ION, Motor = Any type on ION
Data 2	0x11	# of axis supported = 1, # of chips = 1
Data 3	0x00	Customization = 0,
Data 4	0x15	ProductVersion = 0, MajorSoftwareVersion = 1, MinorSoftwareVersion = 5

c. Timeout Exception – Drive may already be in Multi-drop mode and respond with a leading address. Recheck packet assuming the first byte is an address.

3. **SetOperatingMode** - to disable the drive

Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x65	Command = SetOperatingMode
Data 1	0x00	Reserved
Data 2	0x00	Disable axis and all control modules

Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet

Drive Initialization

This script ensures that the drive is disabled and then sets all the motor drive constants that are not related to a particular motor or axis function, and that do not require motor motion.

1. **SetOperatingMode** - to disable the drive

   Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x65	Command = SetOperatingMode
Data 1	0x00	Reserved
Data 2	0x00	Disable axis and all control modules

   Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet

2. **SetMotorLimit** - to zero to disable the drive

   Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x06	Command = SetOperatingMode
Data 1	0x00	Set motor limit MSB to zero
Data 2	0x00	Set motor limit LSB to zero

   Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error

3. **SetOverTemperatureLimit** – Set the limit for the motor drive baseplate temperature sensor.

   Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x1B	Command = SetOverTemperatureLimit
Data 1	0x32	MSB for limit of 50°C (((Data 1:Data 2) / 256)°C)
Data 2	0x00	LSB for limit of 50°C

   Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Data 1	0x32	MSB for limit of 50°C (((Data 1:Data 2) / 256)°C)
Data 2	0x00	LSB for limit of 50°C

4. **SetBusVoltageLimits** – Set the limit for the maximum motor drive bus voltage.

   Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x62	Command = SetBusVoltageLimits
Data 1	0x00	Always zero
Data 2	0x00	Limit to set is the Overvoltage limit
Data 3	0x56	MSB of 30V limit (30V / 1.3612mV)
Data 4	0x18	LSB of 30V limit

Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Data 1	0x00	Always zero
Data 2	0x00	Limit to set is the Overvoltage limit
Data 3	0x56	MSB of 30V limit (30V / 1.3612mV)
Data 4	0x18	LSB of 30V limit

5. **SetBusVoltageLimits** – Set the limit for the minimum motor drive bus voltage.

Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x62	Command = SetBusVoltageLimits
Data 1	0x00	Always zero
Data 2	0x01	Limit to set is the Undervoltage limit
Data 3	0x39	MSB of 20V limit (20V / 1.3612mV)
Data 4	0x65	LSB of 20V limit

Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Data 1	0x00	Always zero
Data 2	0x01	Limit to set is the Undervoltage limit
Data 3	0x39	MSB of 20V limit (20V / 1.3612mV)

Data 4	0x65	LSB of 20V limit

6. **SetPWMFrequency** – Set the switching frequency of the motor drive bus PWM output.

   Command Packet

Address	0x??	Drive address
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Command	0x00, 0x6C	Command = SetBusVoltageLimits
Data 1	0x13	MSB of value for 20KHz PWM frequency (5000)
Data 2	0x88	LSB of value for 20KHz PWM frequency

   Response Packet

Address	0x??	Drive address
Status	0x00	If non-zero, decode instruction error
Checksum	0x??	Two's complement of the least significant byte of sum of the rest of the packet
Data 1	0x13	MSB of value for 20KHz PWM frequency (5000)
Data 2	0x88	LSB of value for 20KHz PWM frequency

Note: From this point on in the document, only the commands and the desired settings are listed in place of the entire packet detail.

7. SetCommutationMode – Sinusoindal
8. SetPhaseCorrectionMode – Enabled
9. SetCurrentControlMode – FOC
10. ClearPositionError
11. SetSampleTime – 104.2μs (102)
12. ClearDriveFaultStatus
13. ResetEventStatus 0x0000

Motor Initialization

1. SetOperating Mode – Disable Drive (0)
2. SetMotorLimit – 0
3. ClearPositionError

4. SetCurrentFoldback – Continuous Current (0), For filter inserter , 3.8 Amps = 17.9% (5873 = 0x16F1)
5. SetCurrentFoldback – Energy Limit (1), For filter inserter, 350Arms^2-s = 11.6% (3791 = 0x0ECF)
6. SetEncoderSource – Incremental
7. SetSignalSense – Invert Negative Limit, Positive Limit, and the Encoder Index signals(0x0034)
8. SetPhaseCounts – 1200 counts/electrical cycle
9. SetPhaseOffset – Predetermined offset value saved for this motor/encoder pair
10. SetFOC – Both D and Q magnetic axis / Kp (0x0200), Predetermined proportional gain value for this motor – GFIP grating insterter is 148 and GFIP grating latch rotator is 165.
11. SetFOC – Both D and Q magnetic axis / Ki (0x0201), Predetermined integral gain value for this motor – GFIP grating insterter is 18 and GFIP grating latch rotator is 24.
12. SetFOC – Both D and Q magnetic axis / ILim (0x0202), Predetermined integral limit value for this motor - GFIP grating insterter is 1820 and GFIP grating latch rotator is 1365.

Phase Initialization

1. See Phase Initialization Procedure document.

Axis Initialization

1. SetOperating Mode – Disable Drive (0)
2. SetMotorLimit – 0
3. SetActualPosition – Enter value corresponding to the position reported by linear absolute position sensor
4. ClearPositionError
5. SetSignalSense – Invert Encoder Index and Limit Switch signals (0x0034)
6. SetAxisOutMask – Selection Mask - Source register is the Activity Status Register (2), Source axis is this axis, Axis 1 (0), Source register mask is Axis Settled (0x80) , Sense Mask – No inputs inverted (0x0000)
7. SetEventAction – Negative Limit (2), Abrupt Stop (2)
8. SetEventAction – Positive Limit (1), Abrupt Stop (2)
9. SetPositionErrorLimit – 1000 counts
10. SetSettleTime – 0.5seconds (4902)
11. SetSettleWindow – 50 (counts)
12. SetTrackingWindow – 300 (counts)
13. SetEventAction – Motion Error (3), Smooth stop (3)
14. SetPositionLoop – Kp (0), Predetermined proportional gain value for this load (200)
15. SetPositionLoop – Ki (1), Predetermined integral gain value for this load (350)
16. SetPositionLoop – ILim (2), Predetermined integral limit value for this load (80,000)

17. SetPositionLoop – Kd (3), Predetermined derivative gain value for this load (20000)
18. SetPositionLoop – dT (4), Predetermined derivative time value for this load (2)
19. SetPositionLoop – Kout (5), Predetermined total gain value for this load – 5% (3277)
20. SetJerk – 2500080 counts/sec^3 (11395)
21. SetAcceleration – 1000000669 counts/sec^2 (681837)
22. SetVelocity – counts/sec 75000 (501350)
23. SetProfileMode – S-curve (0x0002)
24. SetFaultOutMask – Current Foldback, Commutation Error, Bus Voltage fault, Over Temperature faults, Negative Limit, Positive Limit, Motion Error, Break Point 1 (either limit switch), Wrap Around, 0x1E76
25. SetMotionCompleteMode – Actual (1)

Axis Move commands

1. SetMotorLimit – 40% = 8.48 Amps
2. SetOperatingMode – Enable the Axis and turn on the Motor Output, Current Loop, Position Loop, and Trajectory Generator (0x0037)
3. SetPosition – target
4. GetActivityStatus – Mask for settled
5. When settled or timed out, SetOperatingMode – Enable Axis only (0x0001)
6. Get linear absolute encoder position
7. For next move, go to step 2.