Preliminary

Animatics SmartMotor™ *with* DeviceNet Specifications:

Appendix: I/O Polling Messaging Handshaking Example

Revision 0	09/30/03	Jason Hyatt
Revision 1	10/03/03	Jason Hyatt
Revision 2	10/06/03	Jason Hyatt

The Example

The handshaking example illustrated is the same example as used in the document section **Animatics SmartMotor™ *with* DeviceNet Specifications, Appendix: Communicating between the Motor User Program and Device Net.**

In this example, a 4 bit BCD value is sent to the motor over DeviceNet and a 3 bit BCD value is received from the motor user program over DeviceNet.

The 4 bit value is sent in object Position Controller Supervisor, class 36 decimal, instance 1 as attribute 27 decimal, following divisor, which is otherwise unused for the application. The corresponding motor variable in the motor user program is MFDIV.

The 3 bit value is received from the motor user program in object Position Controller, class 37 decimal, instance 1 as attribute 35 decimal, velocity feed forward, which is otherwise unused for the application. The corresponding motor variable in the user program is KV.

The four bit value sent to the motor will be 6, the three bit value received from the motor will be 5.

I/O Polling Messages

The DeviceNet Command Message is 8 bytes, bytes 0 through 7, bits 0 through 63, set by the PLC and continuously being taken and sent by the DeviceNet master (without regard to how the PLC is changing the bits and bytes of the message).

The DeviceNet Response Message is 8 bytes, bytes 0 through 7, bits 0 through 63, set by the DeviceNet master as received from the motor (without regard to how the PLC may be changing bits in the command message that will affect the contents of the response message).

Handshaking Sequence

COMMAND MESSAGE HANDSHAKING

Clear command message bit 0 to 0. Set command message parameters (first four command message bytes).

Clear command message bit 0 to 0. The first four bytes of the message registers must never produce an invalid message. Depending on the values already set in the first four bytes of the message registers, it may be necessary to set the first four bytes of the command message as a coherent, single, 32-bit word store. This avoids creating invalid messages that would occur if the parameters were inconsistent. Particularly, an invalid message would occur if the attribute to set or get was inconsistent with the message types hex 1A, Position Controller Supervisor Attribute, and hex 1B, Position Controller Attribute. Single bits or bytes may be set, so long as there is never a time that the values of the first four bytes taken together would produce an invalid message.

Optionally, the command message parameters may be set together (coherently) with setting command message bit 0 to 1, below.

Set all command data bits and bytes in any order.

Set the four data bytes in any order.
Verify Response message byte 2 bit 7 is 0 (or wait until it becomes 0).

Verify the data loaded bit, Response message byte 2, bit 7 is 0 (or wait until it becomes 0). This bit being clear indicates that the motor has "seen" that command message bit 0 has become clear, so the motor will be able to recognize the zero to one transition of the bit.

Set command message bit 0 to 1 to say all bytes in the command are valid and coherent.

Set command message bit 0 to 1 to indicate all bytes in the command are valid and coherent, including the data. This bit transitioning from 0 to 1 tells the motor to load the data contained in the message, according to the message parameters.

Optionally, the command message parameters may be set together (coherently) with setting command message bit 0 to 1.

Verify Response message byte 2 bit 7 is 1 (or wait until it becomes 1).

Verify the data loaded bit, Response message byte 2, bit 7 has become 1 (or wait until it becomes 1). This bit set indicates the motor has "seen" that command message bit 0 has become set, and has loaded, or "consumed," the data in the command message. The command message may now be changed.

RESPONSE MESSAGE HANDSHAKING

Response message "handshaking" consists of the meaning of the response message data being determined by the response message type and, if applicable, the Get Attribute number, carried in the response message.

When the command message changes the requested response message, as soon as the response message type and, if applicable, the Get Attribute number, reflect the change, the data in the response message also reflects the change.

Set the Response message parameters in the command message.

Set the response message request parameters in the command message. For command message types hex 01 to hex 05, only set the response message type in byte 3 of the command message. For message types hex 1A, Position Controller Supervisor Attribute, and hex 1B, Position Controller Attribute, the response message type will be identical to, and determined by, the command message type. For these types, hex 1A and hex 1B, set the Attribute to Get number in byte 1 of the command message. The first four bytes of the message registers must never produce an invalid message. Depending on the values already set in the first four bytes of the message registers, it may be necessary to set the first four bytes of the command message as a coherent, single, 32-bit word store. This avoids creating invalid messages that would occur if the parameters were inconsistent. Particularly, an invalid message would occur if the attribute to set or get was inconsistent with the message types hex 1A, Position Controller Supervisor Attribute, and hex 1B, Position Controller Attribute. Single bits or bytes may be set, so long as there is never a time that the values of the first four bytes taken together would produce an invalid message.

Verify Response message type in byte 3 of response message is of type expected, and Attribute to Get is number expected

Verify response message type in byte 3 of response message is of type expected (or wait until it matches the response message type being requested in the command message).

For message types hex 1A, Position Controller Supervisor Attribute, and hex 1B, Position Controller Attribute, verify the Attribute to Get number in byte 1 of the response message is the attribute expected (or wait until it matches the response message type being requested in the command message).

Rely on Response message data bits and bytes to correspond to response message type and attribute to get parameters contained in the response message.

Retrieve the data from the message registers and interpret according to the .

Possible Re-verification

(If a different task were changing the command message, and changing response message requests, one would be concerned with making sure the response message type and attribute requested, if applicable, had not been changed between verification and reading the data.)

Handshaking Sequence Applied to Example

In the example, the handshaking sequence involves two different Polling Message Command Types. Type hex 1A is used to for to send (SET) the 4 bit value to object Position Controller Supervisor, which in turn is received by the motor. Type hex 1B to retrieve (GET) the 3 bit value from object Position Controller.

Only one of them, the SET, involves loading data, and therefore the load data bit.

SEND A COMMAND TO THE MOTOR (value = 6)

Clear command message bit 0 to 0. Optionally set command message parameters.

Command Message

Byte	Value	Meaning
0	x8**0**	bit 0 clear, prepare for latch of command data when bit 0 transitions from 0 to 1 bit 7, x80, enable motor to have power to coils (don't disable)
1	x**1B**	GET attribute for response message can be same as SET
2	**x1A**	Message Type to access Position Controller Supervisor object
3	**x1B**	SET attribute is 27 decimal is hex 1B, follow divisor
4	…	don't care
5	…	don't care
6	…	don't care
7	…	don't care

Set all command data bits and bytes in any order.

Command Message

Byte	Value	Meaning
0	x80	enable motor to have power to coils (don't disable)
1	x1B	GET attribute for response message can be same as SET
2	x1A	Message Type to access Position Controller Supervisor object
3	x1B	SET attribute is 27 decimal is hex 1B, follow divisor
4	**x06**	low order data byte, value is 6 to send to motor (bits 33, 35)
5	x00	second data byte 0
6	x00	third data byte 0
7	x00	high order data byte, 0

Verify Response message byte 2 bit 7 is 0 (or wait until it becomes 0).

Response Message

Byte	Value	Meaning
0	…	
1	…	

2	**b0**xxx xxxx	bit 7 clear, prepared for latch of command data when bit 0 transitions from 0 to 1
3	…	
4	…	
5	…	
6	…	
7	…	

Set command message bit 0 to 1 to say all bytes in the command are valid and coherent. (Optionally coherently set command message parameters if they have not been set before.)

Command Message

Byte	Value	Meaning
0	x8**1**	bit 0 transitions from 0 to 1, load command message data
1	x1B	GET attribute for response message can be same as SET
2	x1A	Message Type to access Position Controller Supervisor object
3	x1B	SET attribute is 27 decimal is hex 1B, follow divisor
4	x06	low order data byte, value is 6 to send to motor (bits 33, 35)
5	x00	second data byte 0
6	x00	third data byte 0
7	x00	high order data byte, 0

Verify Response message byte 2 bit 7 is 1 (or wait until it becomes 1).

Data loaded bit set in The command has now been communicated to the motor. There may be a small delay for the user program in the motor to poll the variable and determine the new value.

Command message type 1A results in response message type 1A, which is reflected back in byte 3. The Attribute to GET number is reflected back in byte 1. The data value retrieved is the value just SET.

Response Message

Byte	Value	Meaning
0	x80	
1	**x1B**	GET attribute for response message data, decimal 27
2	x**8**0	data loaded bit, bit 7, and other status
3	**x1A**	Response Message Type for Position Controller Supervisor object
4	**x06**	low order data byte, GET value is 6 because SET to 6 (bits 33, 35)
5	x00	second data byte 0
6	x00	third data byte 0
7	x00	high order data byte, 0

Preliminary

VALUE SENT TO MOTOR.

CLEAR HANDSHAKE IN ADVANCE TO BE READY TO SEND NEXT VALUE.

MONITOR VALUES SENT BY THE MOTOR UNTIL TIME TO SEND NEXT VALUE TO MOTOR.

Clear command message bit 0 to 0. Set command message parameters.

The command bit 0 is cleared and the response message type and Attribute to Get number are set to GET the value from the motor, contained in attribute 35 decimal, 23 hex of the Position Controller object. Since there is no attribute to SET in the Position Controller Object, the Attribute to Set number in command message byte 3 is set to 0. The first four bytes of the command message are stored as a coherent, single, 32-bit word store. This avoids creating invalid messages that would occur if the parameters were inconsistent.

In this example, if the Message Type byte was changed by itself, the command message would still be saying to SET and GET attribute 27 decimal, 1B hex. However, attribute 27 supported in the Position Controller Supervisor, is not supported in the Position Controller, so an error would result. Similarly, if the

Command Message

Byte	Value	Meaning
0	**x80**	enable motor to have power to coils (don't disable)
1	**x23**	Get attribute 35 decimal from class 37
2	**x1B**	Message Type to access Position Controller, class 37
3	**x00**	no attribute to Set
4	x06	low order data byte left over from previous command, unused
5	x00	second data byte, unused
6	x00	third data byte, unused
7	x00	high order data byte, unused

VALUE RECEIVED FROM MOTOR.

Rely on Response message data bits and bytes to correspond to response message type and attribute to get parameters contained in the response message.

The data for GET attribute x1B, decimal 27 should now be the same as we SET.

Response Message

Byte	Value	Meaning
0	x80	
1	**x23**	GET attribute for response message data, decimal 27
2	x00	miscellaneous status
3	**x1B**	Response Message Type for Position Controller Supervisor object
4	**x05**	low order data byte, GET value from motor is 5
5	x00	second data byte 0
6	x00	third data byte 0
7	x00	high order data byte, 0

WHEN OUTPUT DATA TO BE SENT TO MOTOR CHANGES, LOOP BACK TO BEGINNING OF SEQUENCE.

The PLC will keep sending command messages and receiving response messages from the motor. Each response message will contain the latest value sent from the motor. When the PLC needs to change the value it sent to the motor, the sequence may be begun again.