Preliminary

Animatics SmartMotor™ *with* DeviceNet Specifications:

Appendix: Communicating between the Motor User Program and DeviceNet

Revision 0	09/18/03		Jason Hyatt
Revision 1	09/25/03	add example	Jason Hyatt
Revision 2	09/26/03		Jason Hyatt
Revision 3	10/06/03	corrections	Jason Hyatt

THE DIFFERENCE BETWEEN DEVICENET ATTRIBUTES AND MOTOR SYSTEM VARIABLES

The SmartMotor™ follows the DeviceNet Position Controller Device Profile. The Device Profile is meant to provide interoperability among position controller motors regardless of vendor. Not all motors have the features of the SmartMotor™, so many of the SmartMotor™ features **are not accessible** through DeviceNet.

Preliminary

Some DeviceNet variables have a Set access, and some Get access. Generally attributes with Set access may be set in the motor through DeviceNet. If you "Get" a DeviceNet attribute with Set access, the value returned will be the DeviceNet default value or the value you "Set" it to in DeviceNet. If you send commands through the serial port, or commands in the downloaded user program in the motor are executed that change the value of the motor system variable reflected by the attribute, this will not be reflected when you "Get" the Set access DeviceNet attribute.

Some DeviceNet variables have Get access. Attributes with Get access may not be set through DeviceNet. If commands received by the motor through the serial port, or commands in the downloaded user program in the motor are executed that change the value of the motor system variable reflected by the attribute, this will be reflected when you "Get" the Set access DeviceNet attribute.

COMMUNICATING FROM DEVICENET TO THE USER PROGRAM

The DeviceNet Position Controller Profile does not provide access to the motor user variables (a through zzz). It does provide Set access to some motor System variables. The user program may query the value of these System variables. System variables unneeded for their designated function by your application may be "taken over" to communicate to the motor's user program.

DeviceNet Attributes Useful to Communicate TO the User Program

Attribute	Size	Available if:	Motor access
Class Decimal 36, Position Controller Supervisor, Instance 1			
27	31 bits	Follow with ratio motion mode not used	term MFDIV, eg. a=MFDIV (number), d=MFDIV&&32 (bit)
Class Decimal 37, Position Controller, Instance 1			
6	32 bits	Absolute Position motion mode not used	term P, eg a=P
7	32 bits	Neither Position nor Velocity motion modes used	term V, eg a=V
8	32 bits	Neither Position nor Velocity motion modes used	term A, eg a=A
25	11 bits	Torque mode not used	term T, eg a=T
30	1-2 bits	application insensitive to KP, allowing low-order bits to be used	term KP
31	1 bit	application insensitive to KI, allowing low-order bit to be used	term KI
32	1-4 bits	application insensitive to KD, allowing low-order bits to be used	term KD

33	1 bit	application insensitive to KL, allowing low-order bit to be used	term KL
35	1-8 bits	application insensitive to KV, allowing low-order bits to be used (Most likely of the Kx to affect the application least.)	term KV
Class 112, SmartMotor™ I/O, Instance 1			
2	1 bit	motor I/O pin A unused (or consistently used)	term UAO, eg a=UAO
Class 112, SmartMotor™ I/O, Instance 2			
2	1 bit	motor I/O pin B unused (or consistently used)	term UBO, eg a=UBO
Class 112, SmartMotor™ I/O, Instance 1			
2	1 bit	motor I/O pin C unused (or consistently used)	term UCO, eg a=UCO
Class 112, SmartMotor™ I/O, Instance 1			
2	1 bit	motor I/O pin D unused (or consistently used)	term UDO, eg a=UDO
Class 112, SmartMotor™ I/O, Instance 1			
2	1 bit	motor I/O pin E unused (or consistently used)	term UEO, eg a=UEO
Class 112, SmartMotor™ I/O, Instance 1			
2	1 bit	motor I/O pin F unused (or consistently used)	term UFO, eg a=UFO
Class 112, SmartMotor™ I/O, Instance 1			
2	1 bit	motor I/O pin G unused (or consistently used)	term UGO, eg a=UGO

Notes regarding the above table:

When the buffered tuning parameters KP, KI, KD, KL, or KV, attributes 30, 31, 32, 33, or 35 are set by DeviceNet, they are also loaded into the filter. The motor F command is issued by DeviceNet.

COMMUNICATING FROM THE USER PROGRAM TO DEVICENET

Generally speaking, only the Get Access DeviceNet attributes are updated when the user program changes the value of the corresponding motor System variable. DeviceNet may query the value of these attributes. System variables unneeded for their designated function by your application may be "taken over" to communicate to DeviceNet.

DeviceNet Attributes Useful to Receive Communication FROM the User Program

Attribute	Size	Available if:	Motor access
Class Decimal 37, Position Controller, Instance 1			
30	1-2 bits	application insensitive to KP, allowing low-order bits to be used	KP=<expression>
	16 bits	application does not dynamically change the PID filter (does not issue F command)	
31	1 bit	application insensitive to KI, allowing low-order bit to be used	KI=<expression>
	16 bits	application does not dynamically change the PID filter (does not issue F command)	
32	1-4 bits	application insensitive to KD, allowing low-order bits to be used	KD=<expression>
	16 bits	application does not dynamically change the PID filter (does not issue F command)	
33	1 bit	application insensitive to KL, allowing low-order bit to be used	KL=<expression>

	16 bits	application does not dynamically change the PID filter (does not issue F command)	
35	1-8 bits	application insensitive to KV, allowing low-order bits to be used (Most likely of the Kx to affect the application least.)	KV=<expression>
	16 bits	application does not dynamically change the PID filter (does not issue F command)	

Class 112, SmartMotor™ I/O, Instance 1

4	1 bit	motor I/O pin A unused (or consistently used)	UAO UAO=<expression>

Class 112, SmartMotor™ I/O, Instance 2

4	1 bit	motor I/O pin B unused (or consistently used)	UBO UBO=<expression>

Class 112, SmartMotor™ I/O, Instance 1

4	1 bit	motor I/O pin C unused (or consistently used)	UCO UCO=<expression>

Class 112, SmartMotor™ I/O, Instance 1

4	1 bit	motor I/O pin D unused (or consistently used)	UDO UDO=<expression>

Class 112, SmartMotor™ I/O, Instance 1

4	1 bit	motor I/O pin E unused (or consistently used)	UEO UEO=<expression>

Class 112, SmartMotor™ I/O, Instance 1

4	1 bit	motor I/O pin F unused (or consistently used)	UFO UFO=<expression>

Class 112, SmartMotor™ I/O, Instance 1

4	1 bit	motor I/O pin G unused (or consistently used)	UGO UGO=<expression>

Notes regarding the above table:

When the BUFFERED tuning parameters KP, KI, KD, KL, or KV, corresponding to DeviceNet attributes 30, 31, 32, 33, or 35, are set by the motor user program, THEY ARE NOT LOADED INTO THE PID FILTER, so long as the motor F command is not issued. This allows them to be safely used to transmit information to DeviceNet, so long as DeviceNet only GETs the attributes and never SETs them. This is because if they are SET by DeviceNet, DeviceNet will also cause the F command to be issued, loading the values into the motor.

When reading the values of the output pins, attribute 4 is used to read the output value on the pin, as set by the motor user program. This is an analog value that ranges from 0 to 1023. If the value is approximately 0, the digital pin value is 0. It the value is approximately 1023, the digital pin value is 1.
Attribute ID 3, Get Digital Input, for Motor DeviceNet version 1.05 and below, is only supported in motors with 4.76 and 4.77 firmware. For these motors, attribute 3 may be used.

ACCESSING DEVICENET ATTRIBUTES

All the attributes may be accessed through DeviceNet Explicit Messaging, as one shot messages, using the GET ATTRIBUTE or the SET ATTRIBUTE service, using the functions provided by your DeviceNet Master.

Attributes in the two classes Position Controller Supervisor Class, Decimal 36, and Position Controller Class, Decimal 37, may be accessed through DeviceNet I/O Polling Messaging, as continuously resent messages, using the handshaking protocol in bit 0 of byte 0 of the command message. Command message types hex 1A and hex 1B may be used to access the class attributes. This is explained in the manual and other appendices.

EXAMPLE

Send 4 bit BCD value to motor user program over DeviceNet
Get 3 bit BCD value from user program over DeviceNet

In this example, Mode Follow motion mode is not being used, so the following ratio divisor, motor variable MFDIV, which is DeviceNet attribute 27 decimal of the object Position Controller Supervisor, class 36 decimal, instance 1, is used to send data to the user program.

Since the PID tuning parameter KV is the least sensitive tuning parameter, and is seldom required, KV is selected to receive information from the motor user program, which is attribute 35 decimal of the object Position Controller, class 37, instance 1. In this example it is assumed KV is not being used simultaneously for tuning. That is, KV is never Set by DeviceNet, but DeviceNet only Gets KV, so DeviceNet is not causing an F to be issued, and the user program is not issuing an F command. Since the F command is never issued, the buffered KV value is never loaded into the filter.

In the PC or PLC

DEVICENET EXPLICIT MESSAGING

In the PLC or PC, using the appropriate DeviceNet master registers or functions, Explicit Messages are sent to Set class 36, instance 1, attribute 27, or Get class 37, instance 1, attribute 35.

Class 36, instance 1, attribute 27 is set to a value from 0 to 15, a value from 0 to 8 is retrieved from Class 37, instance 1, attribute 35.

DEVICENET I/O POLLING MESSAGING

In this example illustrating I/O Polling Messages, the value sent will be 6, and the value retrieved 5. Please refer to other sections for detailed explanations and examples of I/O Polling Messaging and handshaking.

I/O Polling Command Message Setup in PLC registers to **SET MFDIV** (data not loaded) to send MFDIV to the motor

Byte	Value	Meaning
0	x8**0**	enable motor to have power to coils (don't disable) bit 0 is 0 to allow the 0 to 1 handshake transition to load data
1	**x1B**	optionally Get same attribute as Set from class 36 decimal
2	**x1A**	Message Type to access Position Controller Supervisor, class 36
3	**x1B**	Set attribute 27 decimal

4	**x06**	low order data byte, value is 6
5	x00	second data byte 0
6	x00	third data byte 0
7	x00	high order data byte, 0

I/O Polling Response Message

Not Shown

I/O Polling Command Message Handshake in PLC registers **SET MFDIV (loading data)** to **send MFDIV to the motor**

Byte	Value	Meaning
0	x8**1**	**add load data bit, data is loaded**
1	x1B	Get same attribute SET in class 36 decimal
2	x1A	Message Type to access Position Controller Supervisor, class 36
3	x1B	Set attribute 27 decimal, Following Divisor (MFDIV in the motor)
4	x06	low order data byte **SET in DeviceNet and sent to motor, value is 6**
5	x00	second data byte sent to motor
6	x00	third data byte sent to motor
7	x00	high order data byte sent to motor

I/O Polling **Response Message**

Byte	Value	Meaning
0	x80	
1	x1A	no attribute to Get
2	x**8**?	**bit 7 set**, data **SET in DeviceNet**
3	x1A	Response Message Type to access Position Controller Supervisor, class 36
4	**x06**	low order data byte returned from DeviceNet, value is 6
5	x00	second data byte returned from motor
6	x00	third data byte returned from motor
7	x00	high order data byte returned from motor

Not Shown

I/O Polling Command Message Setup in PLC registers **Get KV** (not yet receiving data)

Byte	Value	Meaning
0	x8**0**	bit 0 is 0 to set up to allow the 0 to 1 handshake transition to load data

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 prior message, unused
5	x00	second data byte, unused
6	x00	third data byte, unused
7	x00	high order data byte, unused

I/O Polling Response Message

Not Shown

I/O Polling Command Message Handshake in PLC registers **Get KV (receiving data)**

Byte	Value	Meaning
0	x8**1**	**add load data bit, command complete to act upon**
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	x00	low order data byte, unused
5	x00	second data byte, unused
6	x00	third data byte, unused
7	x00	high order data byte, unused

I/O Polling **Response** Message **returns value of KV**

Byte	Value	Meaning
0	x80	
1	x23	attribute to Get
2	x??	various status
3	x1B	Response Message Type to access Position Controller, class 37
4	**x05**	low order data byte **returned from motor, value 5**
5	x00	second data byte returned from motor
6	x00	third data byte returned from motor
7	x00	high order data byte returned from motor

In the Motor User Program

RECEIVING THE VALUE FROM THE DEVICENET MASTER

Motor variable MFDIV will contain the value being sent by the DeviceNet master received by the motor. If the four bits are meant to be four Booleans, the motor program could be written several ways:

```
x=MFDIV      ‘ capture MFDIV if all bits need to be coherent with each
             ‘ other during a cycle of processing in the program
IF x&1 <commands to execute if bit 0 is 1> ENDIF
IF x&2 <commands to execute if bit 1 is 1> ENDIF
IF x&4 <commands to execute if bit 2 is 1> ENDIF
IF x&8 <commands to execute if bit 3 is 1> ENDIF
```

or, if it was desired to place the binary value of each bit in four separate variables, j, k, l, and m, to be logically treated as Booleans with only values 0 or non-zero:

```
x=MFDIV      ‘ capture MFDIV if all bits need to be coherent with each
             ‘ other during a cycle of processing in the program
j=x&1        ‘ set j to 0 or 1 if bit 0 is 1
k=x&2        ‘ set k to 0 or 2 if bit 1 is 1
l=x&4        ‘ set l to 0 or 4 if bit 2 is 1
m=x&8        ‘ set m to 0 or 8 if bit 3 is 1
```

or, if it was desired to place the binary value of each bit in four separate variables, j, k, l, and m, to be logically treated as Booleans with only values 0 or 1:

```
x=MFDIV      ‘ capture MFDIV if all bits need to be coherent with each
             ‘ other during a cycle of processing in the program
j=x&1        ‘ set j to 0 or 1 if bit 0 is 1
k=x&2        ‘ set k to 0 or 2 if bit 1 is 1
k=k==2       ‘ set k to 0 or 1 if k is equal to 2
l=x&4        ‘ set l to 0 or 4 if bit 2 is 1
l=l==4       ‘ set l to 0 or 1 if l is equal to 4
m=x&8        ‘ set m to 0 or 8 if bit 3 is 1
m=m==8       ‘ set m to 0 or 1 if m is equal to 8
```

SENDING THE VALUE TO THE DEVICENET MASTER

Motor variable buffered KV is used to send the three bit value to the DeviceNet master.

If we suppose that variables p, q, and r govern respectively bits 0, 1, and 2 of the value of KV, we might use motor code:

```
y=0          ‘ zero shadow register used if all bits need to be coherent with one
             ‘ another when sent to the master
IF p y=y|1 ENDIF     ‘ set bit 0 in y if p
IF q y=y|2 ENDIF     ‘ set bit 1 in y if q
```

```
IF r y=y|4 ENDIF        ‘ set bit 2 in y if r
KV=y
```

or, if p, q, and r only have values 0 or, respectively, 1, 2, and 4:

```
y=0             ‘ zero shadow register used if all bits need to be coherent with one
                ‘ another when sent to the master
y=y|p           ‘ set bit 0 in y if p (since p is only 0 or 1)
y=y|q           ‘ set bit 1 in y if q (since q is only 0 or 2)
y=y|r           ‘ set bit 2 in y if r (since r is only 0 or 4)
KV=y
```

or, if p, q, and r all have values 0 or 1:

```
y=0             ‘ zero shadow register used if all bits need to be coherent with one
                ‘ another when sent to the master
y=y|p           ‘ set bit 0 in y if p (since p is only 0 or 1)
z=q*2           ‘ value of q in bit 1 of z (since q is only 0 or 1)
y=y|z           ‘ set bit 1 in y if q
z=r*4           ‘ value of r in bit 2 of z (since r is only 0 or 1)
y=y|z           ‘ set bit 2 in y if r
KV=y
```