Controlling Animatronic Mechanisms with Various Interfaces

Included Files:
- QCI-AN079_0to10v_Interface.qcp
- QCI-AN079_DMX_Interface.qcp
- QCI-AN079_CAN_Interface.qcp

This application note is to demonstrate the programming flexibly of QuickSilver Controls’ SilverMax™ X-series fully integrated hybrid servo motors and their conventional non-integrated SilverSterling™ series servo controller/drivers in animatronic and show applications.

SilverMax X-series servo motors and SilverSterling series servo controllers both support various interfaces common in the animatronic and show industries. With multiple interface options, the user is free to select the desired servo motor controller interface and not be constrained by product capabilities.

This application note provides example programs for controlling a rotary axis with the following common animatronic/show interfaces:

1. 0 to 10v Analog Input*
2. DMX-512
3. CANopen

*SilverMax X-series servo motors include circuitry to directly interface with a 0 to 10v analog input. For SilverSterling series controller/driver, contact factory for 0 to10v analog interface options.

Mechanism

In this application note example, the mechanism consists of a NEMA 23 SilverMax X-series servo motor (not shown) driving a 50:1 gearhead with a home limit switch. The NEMA 23 SilverMax X-series has an encoder resolution of 8,000 counts/revolution. The clear blue piece is used to illustrate an animatronic rotary joint, such as an elbow, shoulder, knee, etc.

Alternatively, in applications with size and/or weight constraints, the integrated SilverMax X-series servo motor and gearhead may be replaced with a more compact 3rd party geared motor, such as a Harmonic Drive 3-phase geared servo motor with position feedback, paired with a SilverSterling servo controller/driver. Contact Factory for options.
Homing to Sensor

All program examples provided in this application note include a homing routine that homes the motor to a sensor wired into I/O #1. The home sensor is assumed to be active LOW.

Additional homing methods are discussed in Application Note AN001 - Homing Techniques.

On power-up, the motor is programmed to find the home sensor. If the home sensor is active on power-up, logic is included to move motor away from the home sensor until inactive before reversing direction towards home sensor.

After the sensor is found, software stop limits are implemented as a safe guard to only allow commanded motions within a safe travel range.

Once homed, the servo motor program will enter one of the following three interfaces described in this application note:

1. 0 to 10v Analog Input
2. DMX512
3. CAN Bus

Total Range of Travel

For the NEMA 23 SilverMax X-series servo motor, from the home sensor, the total range of travel is ¼ turn (90°) of the 50:1 gearhead output shaft. This equates to:

\[
50 \text{ revs} \times \frac{1}{4} \times 8,000 \text{ counts/rev} = 100,000 \text{ counts}
\]

Total Range of Travel = 100,000 counts

The maximum interface input (10v for analog input and 65535 for DMX/CAN 16-bit input data) will be scaled to the total range of travel, which is 100,000 counts in our application.
**0 to 10v Analog Input**

The 0 to 10v analog input is scaled from 0 counts (home position) to 100,000 counts (Total Range of Travel), respectively.

To achieve this, the Position Input Mode (PIM) command function is used to automatically control and scale motor position proportional to the analog input.

Line 3: ARX command configures the ADC continuously convert the 0 to 10v analog input and copy the ADC result to Input Source Data[12] register.

Lines 5-15: Define registers used by PIM command:
- Input Offset[13]
- Input Dead Band[14]
- Maximum Scale[15]
- Maximum Scale Output[16]
- Output Offset[17]
- Output Rate of Change[18]

Line 11: Sets total range of travel. If modifying this line, note to also modify the software stop limits in the homing program.

Line 15: Sets motor velocity to control how fast motor can respond to changes in commanded position. For immediate motor response to commanded position, increase velocity. For softer starts and stops, reduce velocity. Note: Initially, motor velocity is at a reduced valued for 5 seconds to allow motor to wind-up to target position. After a brief delay, velocity is increased.

Once program is downloaded, registers may be modified via the Register Watch tool (Tools → Register Watch) with changes to registers being applied on the fly. This allows the programmer to test register parameters without having to redownload program.

**Suggestions**

If Input Source Data[12] ADC result doesn’t zero out with a 0v input, this offset can be cancelled by adjusting the Input Offset[13] register. On the opposite end, if motor doesn’t reach the full length of travel for 10v input, adjust the Maximum Scale/Limit[15] register.

For velocity "jog" control, rather than positional control, the Velocity Input Mode (VIM) command function is available. For more information on Input Mode command functions, refer to Application Note AN047 - Input Mode.pdf
DMX512

The DMX protocol transmits data in bursts (DMX packet). A DMX packet begins with a start code which identifies the data type followed by up to 512 individual bytes of data called slots or channels. Each slot carries 8-bits of information and can hold a value from 0 to 255. QCI’s devices can map a single 8-bit slot, up to four contiguous 8-bit slots, providing 32-bits of DMX data.

The application program will map 2 contiguous 8-bit slots (16-bits) for a total incoming DMX data range of 0 to 65535. DMX slots 10 and 11 are mapped to user register [41], then the contents of register [41] is copied to the Input Source Data[12] register. The program uses the same PIM command function as the 0 to 10v application example above. However, instead converting analog input to ADC counts into Input Source Data[12] register, the program will map DMX data into Input Source Data[12] register.

Lines 3-5: Configures communication for DMX protocol.

Line 7: DMRM command defines a single register map. The DMRM command can define up to 6 separate register maps to map additional DMX slots within the same DMX packet.

Lines 19-32: Define registers used by PIM command:
- Input Offset [13]
- Input Dead Band[14]
- Maximum Scale[15]
- Maximum Scale Output[16]
- Output Offset[17]
- Output Rate of Change[18]

Line 32: Sets motor velocity to control how fast motor can respond to changes in command position.

Line 46: Copy mapped DMX data from register [41] into Input Source Data[12].

Refer to Application Note AN045 – DMX512 Protocol.pdf for more information on advanced DMX-512 configuration options.
CANopen

The CANopen program example configures an Receive Process Data Object (RPDO) to have the unit to receive 16-bits of data transmitted from a remote CAN device with CAN ID of 1. The data is assumed to be transmitted and received via CAN Channel #1. The data is mapped to Input Data Source [12] register, where the PIM command uses to scale motor position.

Lines 3-5: Configures CAN baud rate and CAN ID.

Lines 7-18: Configures the unit to map CAN data to its local register, Input Source Data[12] register.

Lines 20-32: Define registers used by PIM command:
- Input Offset [13]
- Input Dead Band[14]
- Maximum Scale[15]
- Maximum Scale Output[16]
- Output Offset[17]
- Output Rate of Change[18]

Line 32: Sets the initial motor velocity. The initial motor velocity is reduced to prevent the motor from taking off at a high speed when first entering PIM.

Line 40: After an initial delay of 5 seconds, the final velocity is set.

Refer to SilverLode CANopen User Manual for more information on CANopen.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Label</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 REM</td>
<td>QCI_AN079_CAN_Interface.cnp</td>
<td>Interface: CANopen</td>
</tr>
<tr>
<td>2 REM</td>
<td>CAN_BaudRate = 1 Mbaud</td>
<td></td>
</tr>
<tr>
<td>3 REM</td>
<td>CAN_BaudRate = 1 Mbaud</td>
<td></td>
</tr>
<tr>
<td>4 REM</td>
<td>CAN_ID = 0x01</td>
<td></td>
</tr>
<tr>
<td>5 REM</td>
<td>CAN_ID = 0x01</td>
<td></td>
</tr>
<tr>
<td>6 REM</td>
<td>Configures Receive Process Data Object (RPDO) to have unit map CANopen data to its local register.</td>
<td></td>
</tr>
<tr>
<td>7 REM</td>
<td>Line 7-18: Configures the unit to map CAN data to its local register, Input Source Data[12] register.</td>
<td></td>
</tr>
<tr>
<td>8 REM</td>
<td>Lines 20-32: Define registers used by PIM command:</td>
<td></td>
</tr>
<tr>
<td>9 REM</td>
<td>Lines 32: Sets the initial motor velocity. The initial motor velocity is reduced to prevent the motor from taking off at a high speed when first entering PIM.</td>
<td></td>
</tr>
<tr>
<td>10 REM</td>
<td>Line 40: After an initial delay of 5 seconds, the final velocity is set.</td>
<td></td>
</tr>
<tr>
<td>11 REM</td>
<td>Refer to SilverLode CANopen User Manual for more information on CANopen.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7 – QCI-AN079_CAN_Interface.cnp