DeviceNet™ for SERVOSTAR®
S300/S600/S700

and DeviceNet HMS Simulator

Paul Coughlin
DeviceNet

- DeviceNet is an 8-Byte Field Bus System for medium range industrial I/O control

- Originally created in 1996 for operation of sensors, switches, barcode scanners, AC/DC drives.

- Currently maintained by independent organization ODVS (Open DeviceNet Vendor’s Association)

- Bus connection Controller Area Network (CAN-standard ISO 11898) as base

- Features implemented for Servo Amplifiers
  - Setup and General Functions – homing, jogging, speed/torque control
  - Positioning Functions – execute motion tasks, absolute trajectory
  - Data Transfer Functions – read actual values, errors, inputs, set outputs
DeviceNet

- Controls up to Up to 64 Nodes

- 24 Volts DC input through shielded 4 wire cable
  - V+, CAN-hi, CAN-lo, V-

- Terminating resistor 120 ohms at each end of cable

* according to line impedance about 1200
DeviceNet

- 3 Baud rates available
  - 125k bps
  - 250k bps
  - 500k bps

- Distances
  - Up to 500 meters @ Baud rate of 125k
  - Up to 250 meters @ Baud rate of 250k
  - Up to 100 meters @ Baud rate of 500k
DeviceNet

- Network Status LED
  - Device not powered, Allocated to Master, Recoverable / Non-recoverable faults

- 2 Rotary Switches for 0 – 63 Amplifier Address
  Address 10 - MSD switch = 1, LSD switch = 0

- Single Rotary Switch for Baud Rate
  Switch 0 = 125k bps, 1 = 250k bps, 2 = 500k bps
2 Communication Methods

- **Explicit Messaging (Slower) – Used to Configure Drive Communications:** <50 mSec (Parameter Object <500 mSec)
  - OOT (Object Oriented Technique)
  - Parameters selected through:
    - Class Object Number, Instance ID, Attribute ID, Explicit Message
  - Read or Write single parameter at a time
  - Homing can only be done via Explicit Messaging

- **Polled I/O (Fast) – Used to Control Movement Real-time Communications:** <10 mSec
  - 8 Byte – Control bits & Status bits
  - Command Assembly (write)
  - Response Assembly (read)
  - Limited Parameters (cannot Home via Polled I/O)
Example of Object Oriented Technique:

- Set Jog Velocity to 500 rpm (01F4 hex)
  - Position Controller Object (class = 0x25)
  - Instance ID = 1
  - Attribute ID = 0x16 (dec 22)
  - Explicit Message = 1F4 (dec 500)
  - Data Type = Double Integer
Polled I/O

- Real-time response communications
- Used to control motion
- Combines Control & Status bits into 8-byte message
- Command Assembly (send)
- Response Assembly (read)
- Only certain parameters assessable
  - Less versatile than Explicit Messaging
DeviceNet

Command Assembly Structure

- 8 Bytes with 8 status bits
  - Byte 0 dedicated commands
  - Byte 1 identifies Block number (Motion Task number)
  - Byte 2 Command Axis / Command Assembly Type (send data)
  - Byte 3 Response Axis / Response Assembly Type (read data)
  - Bytes 4 – 7 Data Fields for data to be transmitted, such as velocity, position

(bit 5 is always 1 for Command & Response Axis – it is not Amp address)

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Enable</td>
<td>Reg Arm</td>
<td>Hard Stop</td>
<td>Smooth Stop</td>
<td>Dir</td>
<td>Relative</td>
<td>Start Block</td>
<td>Load / Start</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>Block Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>Command axis = 001</td>
<td>Command Assembly Type (00001)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>Response axis = 001</td>
<td>Response Assembly Type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>Target Position Low Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>Target Position Low Middle Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>Target Position High Middle Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>Target Position High Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DeviceNet

Example of Polled I/O: (need to be in Velocity Opmode)

- **Set Jog Velocity to 20,000 counts per second**
  - Byte 0 - bit 7 Enables drive, bit 0 Starts motion
  - Byte 1 – not use
  - Byte 2 – selects Command Axis (writes) & Command Assembly 0x02 Velocity
  - Byte 3 – selects Response Axis (reads) & Response Assembly 0x01 Actual Position
  - Bytes 4 & 5 loads data 0x00004E20 in hex (dec 20,000)
DeviceNet Simulator

- Polled I/O Screen
  - Output Data
    - (Command Assembly)
  - Input Data
    - (Response Assembly)
- Dongle Port
- Baud Rate
- Slave Address
  - Additional information
  - Vendor ID
  - Serial Number
  - Product Name
Polled I/O – Enabling Drive

- Enable Drive = Byte 0, Bit 7 to 1

![DeviceNet Master Simulator](image)
Changing Velocity (Digital Velocity)

Change Velocity and/or Direction on the fly using Polled I/O

<table>
<thead>
<tr>
<th>Opmode 0 (Digital Velocity)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Bit 7</td>
<td>Bit 6</td>
<td>Bit 5</td>
<td>Bit 4</td>
<td>Bit 3</td>
<td>Bit 2</td>
<td>Bit 1</td>
<td>Bit 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

DeviceNet Master Simulator

DeviceNet Dongle Port: USB
DeviceNet Dongle Status: Bus normal
Baud Rate of DeviceNet Dongle: 500 kBaund
Current Slave Address: 101

Output Data: 76543210
Input Data: 812b 10110001

Vendor ID: 452
Device Type: 16
Product Code: 4
Revision: 1.5
Serial Number: 020214390
Product Name: ServoStar 300
Monitoring Actual Position

- Polled I/O
  Actual Position
  Read from GUI = 4998

Actual Position
Read from DeviceNet
00001387 hex
Input Data Bytes 4 & 5
Write Explicit Messaging

- Set Jog Velocity to 500 RPM
  Class 0x25
  Instance ID 1
  Attribute ID 0x16
  Explicit Message 0x01F4
  (Send Value)

  Data Type Double Integer

- Write Attribute Screen

- Then set Attribute 0x0B to initiate the move.
  - Data field Boolean
  - Send Value 01.
Read Explicit Messaging

- Read Jog Velocity
  Class 0x25
  Instance ID 1
  Attribute ID 0x16
  Explicit Message 0x01F4
  (Result Value)

Data Type Double Integer

- Read Attribute Screen

- Result Value should be 1F4 hex
  (Value of Attribute 0x16)
Explicit Messaging

- **Position Controller Object: Class 0x25**
  - Most commonly used for Operation Modes (Torque, Velocity & Position) and configuring Motion

- **Parameter Object: Class 0x0F**
  - Vendor defined object for amplifier configuration. Uses DPR (dual port RAM) number from ASCII reference guide

- **Additional Objects**
  - **Command Block: Class 027** – creating and storing Motion Tasks
  - **Block Sequencer: Class 0x26** – executing Motion Tasks
  - **Position Object: Class 0x24** – handles errors of amplifiers
DeviceNet Simulator

- Start DeviceNet Master Simulator
- Click on Address Search for slaves

Note:
CommPort Baud Rate Node Address
- Address 10 is identified (no parenthesis)
DeviceNet Simulator

- Establish Communications
- Click on Communications
  - Start
  - Stop
  - Read Attribute
  - Write Attribute
- Polled I/O screen shown
Read Attribute screen shown
Write Attribute screen shown
Position Controller Object: Class – 0x25

- Changing Opmodes
  - Class 0x25
  - Instance 0x01
  - Attribute 0x03
  - Explicit Message 0x00 (Send Value)

- Data Type Short Integer

  Position Mode = 0x00
  Velocity Mode = 0x01
  Torque Mode = 0x02

  Read only = 0x03

  View S300 GUI as you enter Send Values
Position Controller Object: Class – 0x25

- Reading Actual Position
  - Class 0x25
  - Instance 0x01
  - Attribute 0x0D
  - Explicit Message (Result Value)

- Data Type Double Integer

Return Result is 0x135C
= 4956 counts (verify with GUI)
Most drive parameters can be read or written using 0x0F
- Instance ID = DPR from GUI
- Attribute ID = 0x01

Note: Most Objects use Instance 0x01 and special Attribute number. Parameter Object uses Attribute 0x01 and Instance corresponding to DPR from GUI

- Only DPR 1 – 254 can be used in Instance, since Instance is 1 byte
- For DPR numbers greater than 254 we use Instance 255, Attribute 0x64 (Instance 255 to be discussed later)

Response time <500 mSec (all other objects <50mSec)
**Example:**

- **Move Home (MH)**
  - `DPR = 141 (dec)`

- **Class ID = 0x0F**
- **Instance ID = 0x8D**
- **Attribute ID = 0x01**
- **Explicit Message = 0x01**
  (Send Value)

<table>
<thead>
<tr>
<th>ASCII-Command</th>
<th>MH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax Transmit</td>
<td>MH</td>
</tr>
<tr>
<td>Syntax Receive</td>
<td>MH</td>
</tr>
<tr>
<td>Type</td>
<td>Command</td>
</tr>
<tr>
<td>ASCII Format</td>
<td>-</td>
</tr>
<tr>
<td>DIM</td>
<td>-</td>
</tr>
<tr>
<td>Range</td>
<td>-</td>
</tr>
<tr>
<td>Default</td>
<td>-</td>
</tr>
<tr>
<td>OpName</td>
<td>8</td>
</tr>
<tr>
<td>Drive State</td>
<td>Enabled</td>
</tr>
<tr>
<td>Start Firmware</td>
<td>1.0</td>
</tr>
<tr>
<td>Configuration</td>
<td>No</td>
</tr>
<tr>
<td>Function Group</td>
<td>Position Controller</td>
</tr>
<tr>
<td>Short Description</td>
<td>Start Homing</td>
</tr>
</tbody>
</table>
Parameter Object: Class – 0x0F

- **Reading Velocity Gain GV**
  - Class 0x0F
  - Instance 0x48 (DPR value 72 dec)
  - Attribute 0x1
  - Explicit Message (Result Value)

- **Data Type Double Integer**

  Return Result is 0x22
  = 34 dec

  Note: Divide Device Net result by 1000 to obtain GUI result
  34 / 1000 = 0.034
  (GUI has Weighting listed)
### Reading GV Proportional Gain

- **ASCII object reference**
- **GV Proportional gain Velocity Control Loop**
  - DPR 72 (dec) 0x48 hex
  - Weighting = 1000

<table>
<thead>
<tr>
<th>ASCII - Command</th>
<th>GV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax Transmit</td>
<td>GV [Data]</td>
</tr>
<tr>
<td>Syntax Receive</td>
<td>GV &lt;Data&gt;</td>
</tr>
<tr>
<td>Available in</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Variable rw</td>
</tr>
<tr>
<td>MMI</td>
<td>Yes</td>
</tr>
<tr>
<td>CAN (CoE)</td>
<td></td>
</tr>
<tr>
<td>Object Number</td>
<td></td>
</tr>
<tr>
<td>3548 (hex)</td>
<td></td>
</tr>
<tr>
<td>DIM</td>
<td>-</td>
</tr>
<tr>
<td>PROFIBUS PNU</td>
<td></td>
</tr>
<tr>
<td>1672 (dec) IND = 1 (dec)</td>
<td></td>
</tr>
<tr>
<td>ASCII Format</td>
<td>Float</td>
</tr>
<tr>
<td>Float</td>
<td></td>
</tr>
<tr>
<td>0.001 ... 369.2</td>
<td></td>
</tr>
<tr>
<td>DPR</td>
<td>-</td>
</tr>
<tr>
<td>72 (dec)</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>0.046</td>
</tr>
<tr>
<td>SERCOS IDN</td>
<td>100</td>
</tr>
<tr>
<td>Data Type</td>
<td>-</td>
</tr>
<tr>
<td>Bus/DPR</td>
<td></td>
</tr>
<tr>
<td>Integer32</td>
<td></td>
</tr>
<tr>
<td>Opmode</td>
<td>0, 1</td>
</tr>
<tr>
<td>Weighting</td>
<td>1000</td>
</tr>
<tr>
<td>Drive State</td>
<td>-</td>
</tr>
<tr>
<td>Start Firmware</td>
<td>1.0</td>
</tr>
<tr>
<td>Configuration</td>
<td>No</td>
</tr>
<tr>
<td>Last Change of this Object</td>
<td>1.0</td>
</tr>
<tr>
<td>Function Group</td>
<td>Velocity Controller</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Yes</td>
</tr>
<tr>
<td>Short Description</td>
<td>Velocity Control Loop: Proportional Gain</td>
</tr>
</tbody>
</table>
Note
Displays:

GUI = 0.034
DeviceNet = 0x22 (34 dec)
Reading or Writing for DPR greater than 254 can be accomplished through Attribute 0x64: Parameter Number.

Load the desired DPR number into the Parameter Number Attribute, then use the Parameter Object Instance 255 (0xFF) to access the Parameter.

Note since this is a Class Attribute (0x0F) use Instance 0 when setting the Parameter Number.

It really is easy. Just an extra step.
Example to Read VLIMP:
- VLIM DPR = 122 (290 DEC)

Set up to read value first
- Class ID = 0x0F
- Instance ID = 0x00
- Attribute ID = 0x64
- Explicit Message = 0x122 (Send Value)

The Explicit Message tells the device that DPR 0x122 is the parameter of the drive we are interested in reading or writing.
Using Attribute 0x64 to set Parameter we are interested in viewing:
- Class 0x0F
- Instance 0x00
- Attribute 0x64
- Explicit Message 0x122
  - DPR 290 = VLIMP
    (Send Value)
- Data Type Integer
Using Instance 255 (0xFF) read Parameter we are interested in viewing:

- Class 0x0F
- Instance 0xFF
- Attribute 0x01
- Explicit Message 0x2DC6C0
  Dec = 3,000,000 (Result Value)

- Data Type Double Integer

Remember value is weighted by 1000, so VLIMP GUI READS 3000
Note that DeviceNet Simulator allows to change from hex to dec.
Running Motion Tasks (Class 0x26)

- **Block Sequencer Object**
  - Class 0x26
  - Instance 0x01
  - Attribute 0x01
  - Explicit Message 0x05 (Send Value)

  This determines the Block Number (Motion Task) to Execute
Running Motion Tasks (Class 0x26)

- Block Sequencer Object
  - Class 0x26
  - Instance 0x01
  - Attribute 0x02
  - Explicit Message 0x01 (Send Value)

This Executes the Block Number (Motion Task) identified from Attribute 0x01
Creating Motion Tasks (Class 0x27)

- Command Block Object
- Class 0x27
- Command 0x08 Motion Task

REMEMBER TO DISABLE DRIVE!!

- Class 0x27
- Instance 0x05
- Attribute 0x01
- Explicit Message 0x08 (Send Value)

Instance = Block No. (Motion Task)
0x08 is Command Block – Motion Task
Creating Motion Tasks (Class 0x27)

- Command Block Object
  Class 0x27
  - Keep Class 0x27
  - Instance = Block Number
  - Attributes for Profile
  - Explicit Message = Values for Task

- Class 0x27
- Instance 0x05
- Attribute 0x03 (Target Position)
- Explicit Message (Send Value)

REMEMBER TO DISABLE DRIVE!!
Creating Motion Tasks (Class 0x27)

- Command Block Object
  Class 0x27

- Attributes
  0x03 = Target Position
  0x04 = Target Velocity
  0x05 = Incremental Move
  0x65 = Acceleration
  0x66 = Deceleration

Note: 0x65 & 0x66 not supported in S600. Use Class 0x25 Attributes 0x08 & 0x09

REMEMBER TO DISABLE DRIVE!!
Creating Motion Tasks (Class 0x27)

- Take a look at the Motion Task Screen as you enter Values

- Now take a look at Terminal Screen
  - Enter Order # (Motion Task number)

  What is the response?

- Now go back to Motion Task Table

  What values appear in the Table?

- Now try initiating the Motion Task with Block Sequencer
Set up Scan List
DeviceNet PLC Screen

- **On Line**

![DeviceNet PLC Screen](image-url)

- **Scanner**
  - Mac Id: 24
  - BaudRate: 500K baud
  - Scan Interval: 50

- **Connections**
  - Explicit
  - Strobed I/O
  - Change-of-State
  - Polled I/O
  - Cyclic
  - Ack Suppress

- **Explicit Buffer**
  - Request Size: 0
  - Req. Offset (hex): 0
  - Response Size: 0
  - Res. Offset (hex): 0

- **I/O Connection 1**
  - Input Size: 1
  - Input Offset (hex): 1000
  - Output Size: 1
  - Output Offset (hex): 1010

- **I/O Connection 2**
  - Input Size: 10
  - Input Offset (hex): 1020
  - Output Size: 10
  - Output Offset (hex): 1030

- **I/O 1 Interval**: 0
- **I/O 2 Interval**: 0

![DeviceNet PLC Screen Diagram](image-url)
Node Active

- MAC Id | Status
- 10     | Active (0)

500K, Active, I/O Active
DeviceNet PLC Screen

- **Polled I/O**
  - Node address (MAC Id)
  - Output to drive
    - Byte 0 – 7
    - Bits 0 - 7
  - Reply from drive
    - Byte 0 – 7
    - Bits 0 - 7
Explicit Message Screen

- Set GV to 5000 dec (1388 hex)
- Node address (MAC Id)
- Service Code
- Class ID
- Data Size
- Instance ID
- Attribute
- Explicit Message (LSD to MSD)

13 88 = 5000 dec (weight 1000)

- Transmit = 1
- Address = 10
- [Service, Class, Instance]
- Attribute, Explicit Message