**Hands-on Lab**

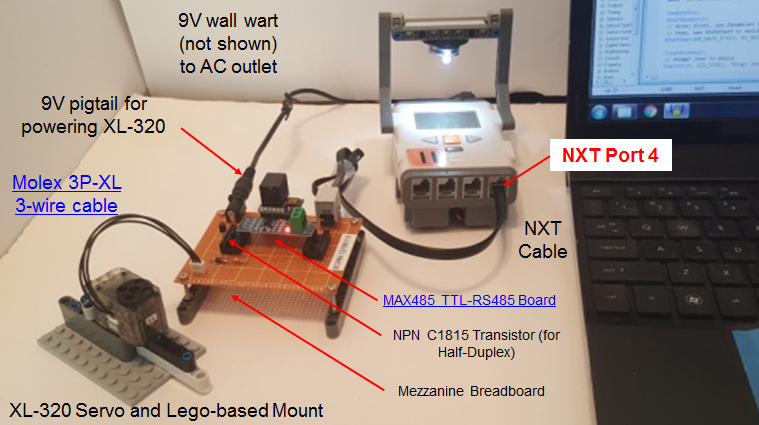
**XL-320 NXC Programming – “Hello World (LED)” Example**

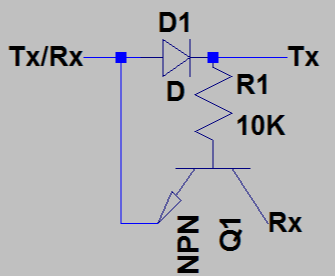
This lab introduces NXC Programming of the Robotis XL-320 Dynamixel servo. The Lego NXT Brick’s Port 4 features a serial interface (RS-485 protocol). This allows the Brick to communicate to the TTL-level serial port on the XL-320. Changing the XL-320’s LED color is a “Hello World” example to introduce RS-485 programming and writing instructions to the XL-320.

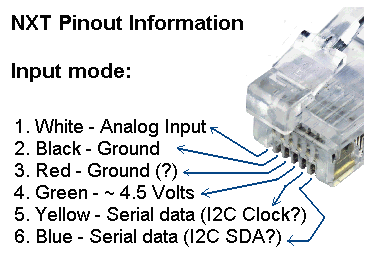
**Preliminary:** Hardware connections and explanation

Hardware Connections

**Figure A:** NXT-to-XL320 connections (left). NPN Transistor for half-duplex (top right) and NXT cable wire description (bottom right)







**Figure A** shows the hardware connections. RS-485 protocol digitizes at -7 to +12 Volts. However, the XL-320 uses transistor-to-transistor logic (TTL) to convert bytes digitally (+5V and Ground). As such, a converter is needed. The Maxim MAX485 is a popular chip for such conversion. Its popularity is underscored by $2 boards complete with supporting passive components. One caveat of the XL-320 is that it uses half-duplex RS-485 interfacing. The Molex connector has wires for power, ground, and data. Thus only 1-wire is used to read and/or write bytes (i.e. half-duplex). As such, a NPN transistor is used to implement half-duplexing (**Figure A top right**). Lastly, the NXT cable’s Yellow (YLW) and Blue (BLU) wires (**Figure A bottom right**) serve serial purposes when RS485 is invoked.

Dynamixel Protocol 2.0 and XL-320 EEPROM

As introduced earlier, the XL-320 is a *smart* servo. Embedded in this servo is random access memory (*RAM*) to hold temporary information like encoder positions and LED states. Also this servo has electronically erasable programmable read-only memory (*EEPROM*) to hold the XL-320’s firmware. Firmware is used to permanently hold device information like identifiers (e.g. model or ID number, communication settings like baud rate, and firmware version).

Robotis’ information on the XL-320 is comprehensive, albeit cryptic:

1. XL-320 specifications <http://emanual.robotis.com/docs/en/dxl/x/xl320/>
2. EEPROM and RAM Control Table <http://emanual.robotis.com/docs/en/dxl/x/xl320/#control-table>
3. Robotis Protocol 2.0 Instruction and Status packets and Packet Processing <http://emanual.robotis.com/docs/en/dxl/protocol2/>

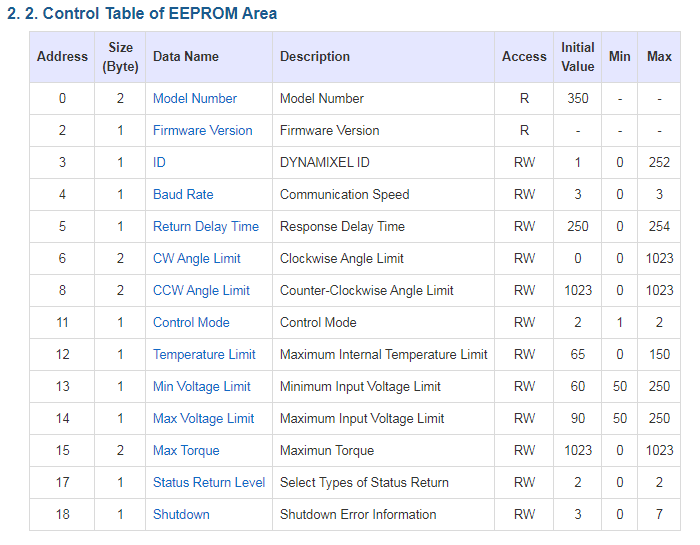
The XL-320 has powerful features and the above links are needed to exploit them.

**Concept 1 Create Definition Header File (H-File) xl320-defines1\_0a.h**

**Step 1:** Create Definition Header File (H-File) – EEPROM Area

A cursory view of links underscores many details of the XL-320. It is helpful to first create a definition header file. This file will #define constants that will be needed to read and/or write Instruction and Status packets. Such packets reference the firmware to command the XL-320.

Section 2.2 of <http://emanual.robotis.com/docs/en/dxl/x/xl320/> details the EEPROM Control Table and shown in **Figure 1A**.



**Figure 1A:** Addresses (in Decimal) for each Data Name in **EEPROM**. This table can be found in [Section 2.2](http://emanual.robotis.com/docs/en/dxl/x/xl320/#control-table) (Control Table) of the Robotis XL-320 E-Manual.

In the C programming language, all-uppercase is conventionally used to define constants. For this and future labs, the prefix EEPROM\_, RAM\_, and INSTRUCTION\_ will be used before the Data Name. Also, the underscore character will be used between each word. Example, for Model Number (top line in **Figure 1A**) would be represented in the H-file as:

#define EEPROM\_MODEL\_NUMBER 0x00 // 2 bytes; motor’s model number

Following this conventional, the EEPROM defines for the H-file (Yellow highlight) would look like **Figure 1B**.

// FILE: xl320-defines1\_0a.h

// AUTH: P.Oh

// DATE: 09/19/19 12:24

// VERS: 1.0a: XL-320 motor defines in Control Table; no functions in this file

// DESC: Refers to Section 2

// http://emanual.robotis.com/docs/en/dxl/x/xl320/#control-table

// Section 2.2 is EEPROM Control Table defines

// Section 2.3 is the RAM Control Table defines

// REFS: F:\nationalInstruments\nxcProjects\rs-485\dynamixel\Dynamixel XL-320\

// paulOhDynamixelXl320HeaderFile-1.0d.h

// Instruction related Defines

#define HEADER\_1 0xFF // For Instruction Packet Header 1

#define HEADER\_2 0xFF // For Instruction Packet Header 2

#define HEADER\_3 0xFD // For Instruction Packet Header 3

#define RESERVED 0x00 // For Instruction Packet Reserved

// EEPROM Address related Defines

// See Robotis Section 2.2 http://emanual.robotis.com/docs/en/dxl/x/xl320/

#define EEPROM\_MODEL\_NUMBER 0x00 // 2 bytes; motor's model number

#define EEPROM\_FIRMWARE\_VERSION 0x02 // 1 byte; motor's firmware version

#define EEPROM\_ID 0x03 // 1 byte; motor's ID number [0-252]

#define EEPROM\_BAUD\_RATE 0x04 // 1 byte; baud [0-3]

#define EEPROM\_RETURN\_DELAY\_TIME 0x05 // 1 byte; instruction packet send time

#define EEPROM\_CW\_ANGLE\_LIMIT 0x06 // 2 bytes; minimum value of Goal Position

#define EEPROM\_CCW\_ANGLE\_LIMIT 0X08 // 2 bytes; maximum value of Goal Position

#define EEPROM\_CONTROL\_MODE 0x0B // 1 byte; Wheel (1) or Joint (2) modes

#define EEPROM\_TEMPERATURE\_LIMIT 0x0C // 1 byte; overheat shutdown value [0-100]

#define EEPROM\_MIN\_VOLTAGE\_LIMIT 0x0D // 1 byte; minimum operational voltage

#define EEPROM\_MAX\_VOLTAGE\_LIMIT 0x0E // 1 byte; maximum operational voltage

#define EEPROM\_MAX\_TORQUE 0x0F // 2 bytes; maximum torque value

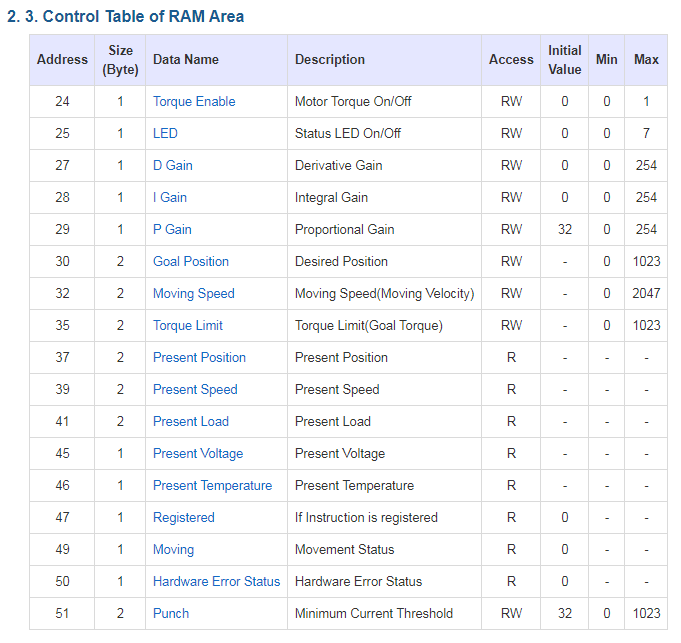
#define EEPROM\_STATUS\_RETURN\_LEVEL 0x11 // 1 byte; how to send status packet

#define EEPROM\_SHUTDOWN 0x12 // 1 byte; when to shutdown motor

**Figure 1B**: #defines for the **EEPROM** Control Table constants

**Step 2:** Create Definition Header File (H-File) – RAM Area

Section 2.3 of <http://emanual.robotis.com/docs/en/dxl/x/xl320/> details the RAM Control Table and shown in **Figure 1C**. Following the aforementioned naming convention, **Figure 1D** shows the #defines to be added to the H-file in **Figure 1B**.



**Figure 1C:** Addresses (in Decimal) for each Data Name in **RAM**. This table can be found in [Section 2.2](http://emanual.robotis.com/docs/en/dxl/x/xl320/#control-table) (Control Table) of the Robotis XL-320 E-Manual.

// RAM Address related Defines

// See Robotis Section 2.3 http://emanual.robotis.com/docs/en/dxl/x/xl320/

#define RAM\_TORQUE\_ENABLE 0x18 // 1 byte; turns on/off torque control

#define RAM\_LED 0x19 // 1 byte; changes motor's LED color

#define RAM\_D\_GAIN 0x1B // 1 byte; motor's derivative gain

#define RAM\_I\_GAIN 0x1C // 1 byte; motor's integral gain

#define RAM\_P\_GAIN 0x1D // 1 byte; motor's proportional gain

#define RAM\_GOAL\_POSITION 0x1E // 2 bytes; destination position value

#define RAM\_MOVING\_SPEED 0x20 // 2 bytes; Wheel or Joint dependent

#define RAM\_TORQUE\_LIMIT 0x23 // 2 bytes; maximum torque limit value

#define RAM\_PRESENT\_POSITION 0x25 // 2 bytes; motor's present position

#define RAM\_PRESENT\_SPEED 0x27 // 2 bytes; Wheel or Joint mode dependent [0-2047]

#define RAM\_PRESENT\_LOAD 0x29 // 2 bytes; currently applied load value is [0-2047]

#define RAM\_PRESENT\_VOLTAGE 0x2D // 1 byte; present supply voltage

#define RAM\_PRESENT\_TEMPERATURE 0x2E // 1 byte; motor's internal temperature in Celsius

#define RAM\_REGISTERED 0x2F // 1 byte; REG\_WRITE instruction received or not

#define RAM\_MOVING 0x31 // 1 byte; Goal Position completed or in-progress

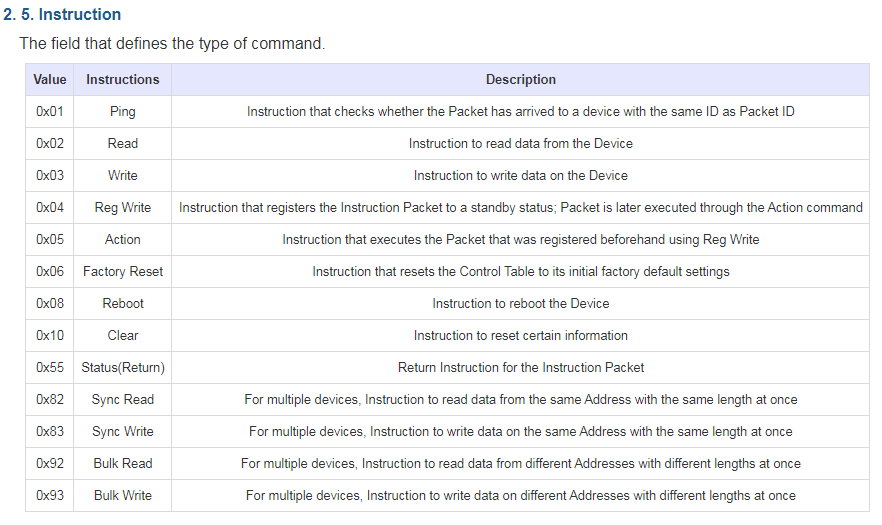
#define RAM\_HARDWARE\_ERROR\_STATUS 0x32 // 1 byte; present hardware error status

#define RAM\_PUNCH 0x33 // 2 bytes; minimum current to drive motor

**Figure 1D**: #defines for the **RAM** Control Table constants

**Step 3:** Create Definition Header File (H-File) – Instruction Area

The XL-320 is one of about a dozen different Dynamixel servos. Their EEPROM and RAM constants may differ but instructions are common. Robotis calls this Protocol 2.0. Section 2.5 of <http://emanual.robotis.com/docs/en/dxl/protocol2/#instruction-packet> details the Instruction Packet and given in **Figure 1E**.



**Figure 1E:** Addresses (in Hex) for each Instruction. This table can be found in [Section 2.5](http://emanual.robotis.com/docs/en/dxl/protocol2/#instruction-packet) (Instruction) of the Robotis Protocol 2.0 E-Manual.

Continuing with the naming convention, the #defines in **Figure 1F** can be added to the H-file.

// Instruction related Defines

// See Section 2.5

// http://emanual.robotis.com/docs/en/dxl/protocol2/#instruction-packet

#define INSTRUCTION\_PING 0x01 // checks if arriving packet ID is same as packet ID

#define INSTRUCTION\_READ 0x02 // read data from device

#define INSTRUCTION\_WRITE 0x03 // write data to device

#define INSTRUCTION\_REG\_WRITE 0x04 // registers instruction packet to set for standby

#define INSTRUCTION\_ACTION 0x05 // executes packet by INSTRUCTION\_ REG\_WRITE

#define INSTRUCTION\_FACTORY\_RESET 0x06 // reset Control Table to factory default

#define INSTRUCTION\_REBOOT 0x08 // reboot device

#define INSTRUCTION\_CLEAR 0x10 // reset certain information

#define INSTRUCTION\_STATUS\_RETURN 0x55 // return instruction for the Instruction packet

#define INSTRUCTION\_SYNC\_READ 0x82 // multiple devices: read all devices

#define INSTRUCTION\_SYNC\_WRITE 0x83 // multiple devices: write all devices

#define INSTRUCTION\_BULK\_READ 0x92 // multiple devices: read different devices

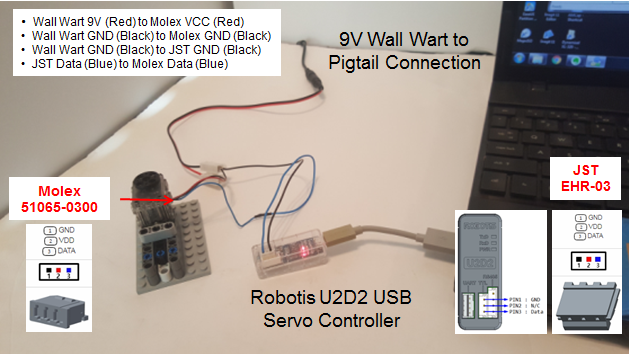
#define INSTRUCTION\_BULK\_WRITE 0x93 // multiple devices: write different devices

**Figure 1D**: #defines for the Robotis Dynamixel Protocol 2.0 Instruction Packet

**Step 4:** Create Definition Header File (H-File) – Packet Headers

Blah see section 3 of Protocol 2.0 <http://emanual.robotis.com/docs/en/dxl/protocol2/#status-packet> add picture of Status Packet – reference top part of code (Figure 1A).

mponents: a 9V wall wart is used to power the XL-320 servo. The pigtail allows the 9V and GND lines to connect to the Robotis U2D2 servo controller using a spring connector. The U2D2 is essentially a microprocessor that allows a computer (USB) to communicate to the XL-320.



**Figure B:** Cable connections.

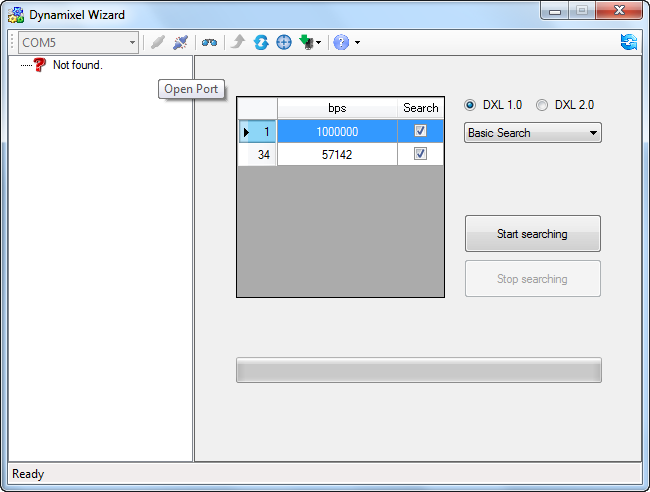
**Figure B** depicts the cable connections. First, tie the GND lines of the XL-320 (Molex connector), the U2D2 (JST connector) and the pigtail using a spring connector. Second, tie the pigtail’s 9V line to the XL-320 VCC line via the spring connector. Lastly, make sure the data lines from the U2D2 and XL-320 are connected (blue wire).

# **Concept 1 – Setting Servo ID and Baudrate**

**Step 1:** Launch the Dynamixel Wizard

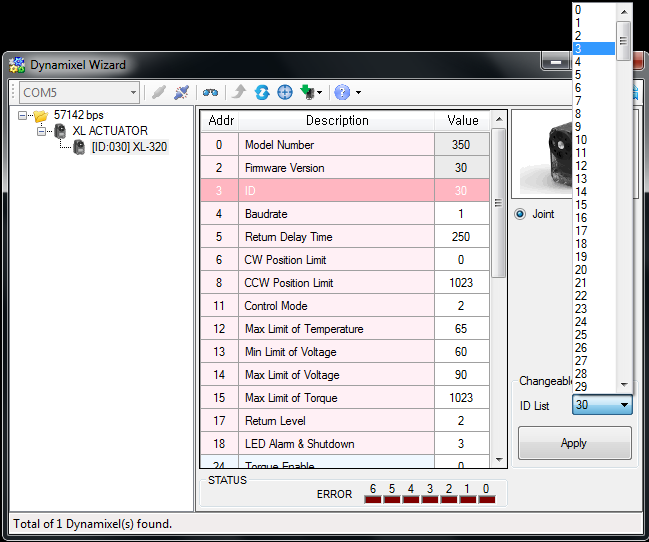
**Figure 1-1** Dynamixel opening screen (left). Clicking Open Port (red circle) and pushing the Start Searching button requests a Dynamixel servo search (right).

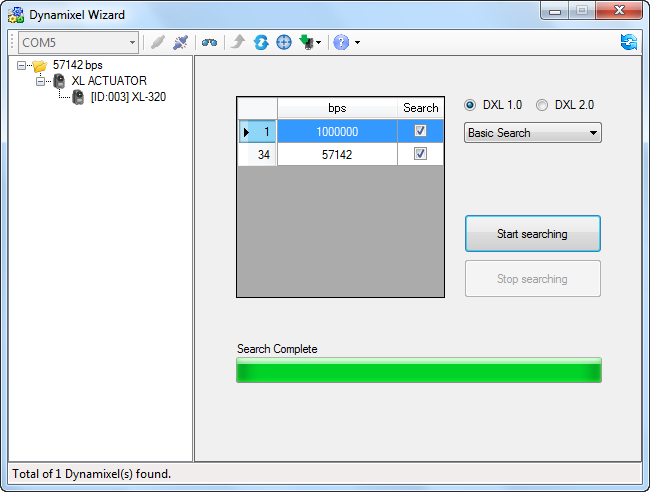




The Dynamixel Wizard is used to setup Robotis servos. **Figure 1-1** (right) allows one to see what Dynamixel servos are connected to the PC’s serial port.

**Figure 1-2:** Results of the search reveals an XL-320 connected to the COM port (left). Clicking on the motor (red circle) reveals its current settings (right).





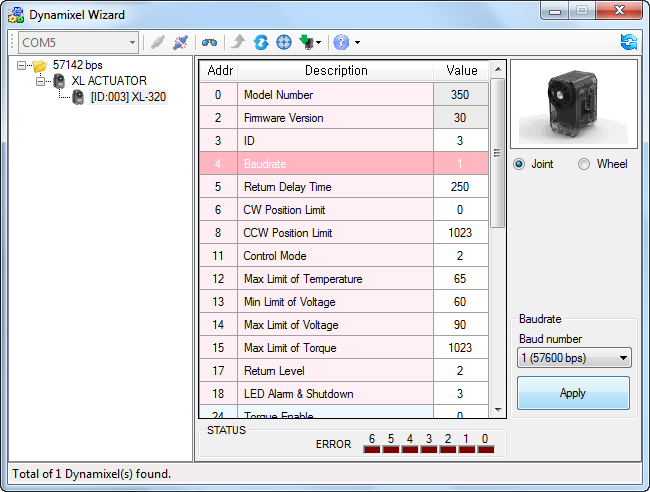
**Step 2:** Change XL-320 Servo ID

**Figure 1-2** shows the properties of the XL-320 servo. Up to 253 different XL-320 servos can be daisy-chained; the servo ID allows one to uniquely identify each one. By default, the servo ID is the set to the label affixed to the XL-320.

For our future Lego NXC programming purposes, let us assign the XL-320 an ID of 1. Select ID and change to 1 and then press the Apply Button. This assigns the connected XL-320 with the ID of 1 (or 0x01 in Hexadecimal).

**Step 3**: Change Baudrate to 57600 bps

By default, factory XL-320 servos are set at a 1 Mbps baud rate. For our future Lego NXC programming purposes, change the baud to 57600.



**Figure 1-3:** Clicking Baudrate allows one to change settings

Click the pull-down menu (**Figure 1-3** red circle), choose 57600 bps, and push Apply.

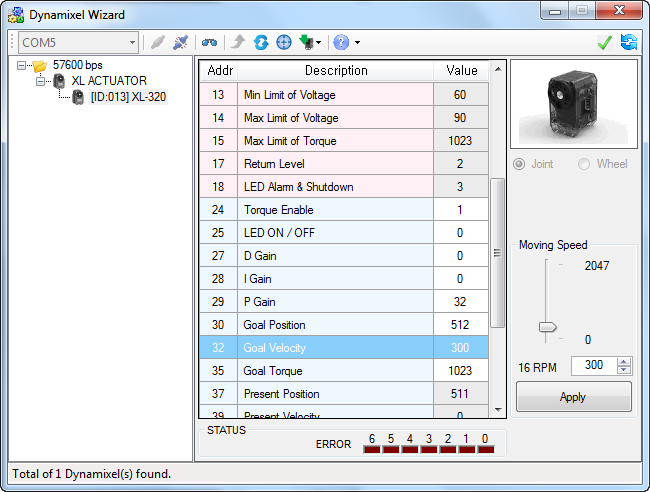
Congratulations! You’ve set the XL-320 servo’s ID and baudrate

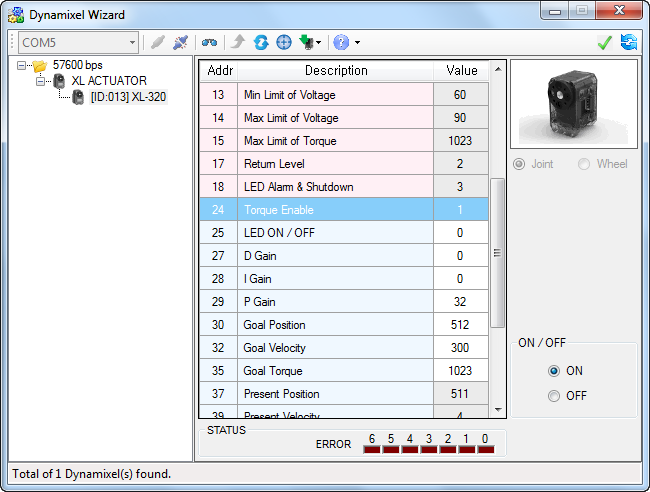
**Concept 2 – Command Angular Position (Joint Mode)**

Dynamixel Wizard offers different modes to command the XL-320. In Joint Mode, one can command the XL-320 to desired angular positions.

**Step 1:** Enable Torque and Set Velocity

**Figure 2-1:** Selecting Torque Enable (left) shows options to turn torque detection ON or OFF. By default, torque is enabled and the motor is in Joint mode (red arrow). Goal Velocity sets the XL-320’s rotational speed using the slider bar or typing values, and hitting the Apply button.

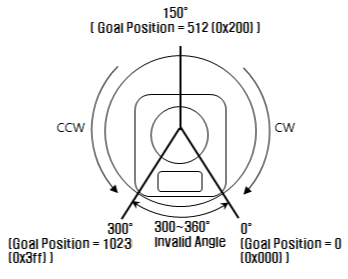


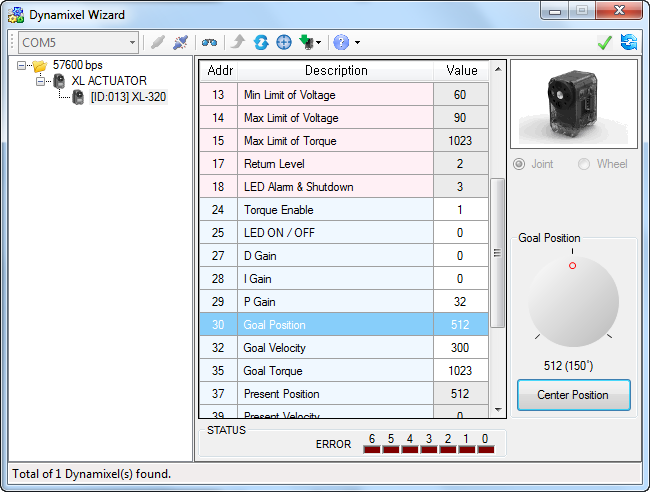


The XL-320 features over-torque protection. This is good to prevent damaging the servo. **Figure 2-1 (left)** shows that by default, Torque Enable is ON. The XL-320 can rotate very quickly and trigger over-torque protection. Thus, one can slow the servo’s velocity; **Figure 2-1 (right)** shows Goal Velocity set to 300 once the Apply button is clicked.

**Step 2:** Command Angular Position

**Figure 2-2:** Goal Position (left) and corresponding degrees and direction (right)





**Figure 2-2** shows Goal Position. Clicking Center Position will command the XL-320 servo to be the middle of its range (0 to 1023). Clicking-and-dragging the dial rotates the XL-320’s horn.

**Exercise 1:**

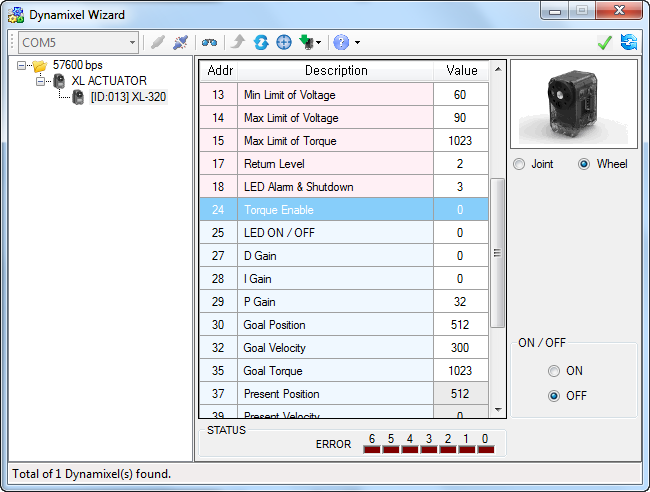
* 1. What Goal Positions corresponds to the XL-320 being at its maximum clockwise and counter-clockwise positions?
  2. What is resolution of the XL-320 servo in degrees? Hint: Maximum angular range is 300 degrees.

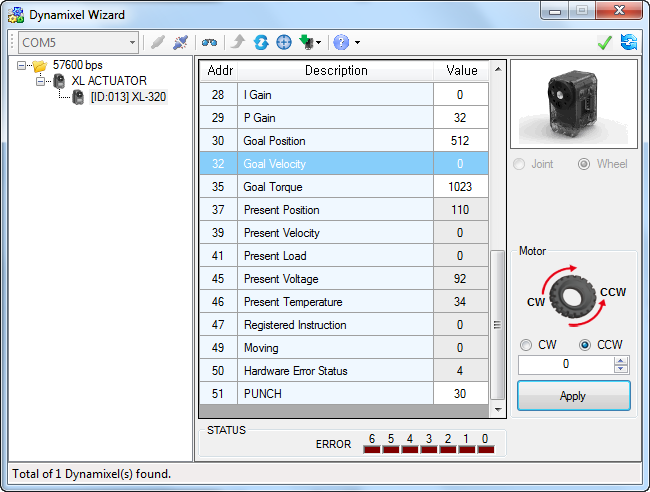
**Concept 3 – Continuous Rotation (Wheel Mode)**

In Wheel Mode, the XL-320’s horn will rotate continuously.

**Step 1:** Disable Torque

**Figure 3-1:** To rotate continuously, first set Torque Enable to OFF (left) and then choose Wheel (red circle). Clicking on Goal Velocity gives options for continuous rotation (right).





**Step 2:** Rotate continuously

Selecting Goal Velocity (**Figure 3-1 right**) one can choose to rotate the XL-320 servo horn clockwise or counter-clockwise. One can also type a numeric value (0 to 2047) and after hitting the Apply button, the XL-320 servo horn will rotate at the speed. Type in 0 and push Apply to stop the servo.

**Exercise 2:**

* 1. What is the maximum angular velocity in degrees/sec?
  2. Set the angular velocity to 150. With a stopwatch, how many seconds does it take for the XL-320 servo horn to rotate once?

Congratulations! You can command XL-320 to desired angular positions and rotate continuously at desired angular velocities