```
// FILE: btS-R-1_0a.nxc - Works!
// DATE: 04/15/20 11:49
// AUTH: P.Oh
// DESC: Slave receives Bluetooth string from Master (running PC-M-S-1_0a.nxc).
//       Slave extracts numerical values from string.  The values are angles
//       which are fed into forward kinematics.  The result is XL-320 joint
//       commands.  The Lego-based 2-DOF planar manipulator moves to those
//       joint commands, briefly waits, and then goes back to HOME position.
//       Slave then sends OK message via Bluetooth, back to Master, and
//       waits for the next angle command from Master.
// VERS: 1_0a: based on btS0_2a.nxc
// REFS: Works with Master running PC-M-S-1_0a.nxc and PC running
//       serialPc-M-1_0a.sce.
//       extract0_1e.nxc: used to detect comma and extract numbers from string
//       xl320-2dof-fk-1_0.nxc forward kinematics

#include "protocol0_2a.h"
#include "xl320-defines1_0a.h"   // XL-320 defines from Control Table
#include "xl320-functions1_0d.h" // P.Oh functions written for XL-320

#define ID_ALL_MOTORS 0XFE // 0XFE commands all XL-320 motors
#define ID_MOTOR01    0X03 // Assumes Motor 1 configured with ID = 3
#define ID_MOTOR02    0X07 // Assumes Motor 2 configured with ID = 7
#define mmPerStud     8    // 8 millimeters per LEGO stud

// Global variables
  bool  orangeButtonPushed;      // Detect Brick Center button state
  bool  rightArrowButtonPushed;  // Detect Brick right arrow button state
  bool  leftArrowButtonPushed;   // Detect Brick left arrow button state
  bool  greyButtonPushed;        // Detect Brick Grey/Abort button state

void rotateMotorAbsolutely(float angle01, float angle02) { //------------------
 // Rotates desired the two Dynamixel XL-320 motors to their desired angles
 // Assumes motor count of 512 denotes 0 degrees.  Uses right-hand rule for
 // rotational direction

  float desiredAngle01InDegrees;   // Angle Motor 1 to move to [deg]
  float desiredAngle02InDegrees;   // Angle Motor 2 to move to [deg]
  float degreesPerCount;           // Conversion 0.29 [degrees/count]
  float calculatedCount;           // Count equivalent of desired angle [count]
  int   motor01Offset;             // Motor 1's offset [count]
  float theta01InDegrees;          // Motor 1 angle [counts]
  int   theta01InCounts;           // Motor 1 angle [deg]
  int   motor02Offset;             // Motor 2's offset [count]
  float theta02InDegrees;          // Motor 2 angle [counts]
  int   theta02InCounts;           // Motor 2 angle [deg]
  string msg01, msg02;             // dummy strings to print values to screen

  motor01Offset = 512; // Set Link 1 at 0 deg (i.e. 512 counts)
  motor02Offset = 512; // Set Link 2 at 0 deg (i.e. 512 counts)

  // Note 1: Looking into horn from Top, count > 512 is CCW (i.e. +Z axis)
  // and count < 512 is CW (i.e. -Z axis)
  degreesPerCount = 0.29; // [deg/count] found from XL-320 data sheet

  ClearScreen();
  desiredAngle01InDegrees = angle01;
  theta01InCounts = motor01Offset + desiredAngle01InDegrees/degreesPerCount;
  desiredAngle02InDegrees = angle02;
  theta02InCounts = motor02Offset + desiredAngle02InDegrees/degreesPerCount;

  // Format string so displays nicely on Brick screen
  sprintf(msg01, "Goto [%3.1f, " ,desiredAngle01InDegrees);
  sprintf(msg02, "%3.1f]" , desiredAngle02InDegrees);
  TextOut(0, LCD_LINE2, strcat(msg01, msg02));

  XL320_servo(ID_MOTOR01, theta01InCounts, 200); // motor position at speed 200
  Wait(2000); // wait about 2 seconds before issuing another command
  XL320_servo(ID_MOTOR02, theta02InCounts, 200); // motor position at speed 200
  Wait(2000); // wait about 2 seconds before issuing another command
  PlayTone(TONE_B3,50);

}; // end rotateMotorAbsolutely function ---------------------------------

task main() {

  // Bluetooth related variables
  string stringFromMaster;          // store string from Master
  int lenStringFromMaster;          // store length value of received string

  byte byteC;                       // ASCII value of character read in strData
  int i;                            // dummy counter variable
  int commaPosition;                // Position in strData of comma
  string message;                   // dummy string to display message
  string strValue01, strValue02;    // extracted numbers as strings
  float floatValue01, floatValue02; // floats of extracted string
  string strOkFromSlave = "OK" ;    // OK from slave

  // planar manipulator variables
  float l1, l2; // length of link 1 and link 2 [mm]
  float theta1, theta2; // angle of joint 1 and joint 2 [rad]
  float theta1InDegrees, theta2InDegrees; // angle of joint 1 and 2 [deg]
  float xP0, yP0; // end-effector absolute position i.e. wrt x0y0 frame [mm]
  int xP0InStuds, yP0InStuds; // [studs]

  // calculation and dummy variables
  float C, k1, k2, num, den;

  // initializations
  l1 = 7 * mmPerStud; // [mm] link 1 is 7 studs long
  l2 = 5 * mmPerStud; // [mm] link 2 is 5 studs long

  UseRS485();
  RS485Enable();
  RS485Uart(HS_BAUD_57600, HS_MODE_8N1); //57600 baud, 8bit, 1stop, no parity

  ClearScreen();
  // Prompt user to begin
  TextOut(0, LCD_LINE1, "Start: hit ->");
  do {
     rightArrowButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
  } while(!rightArrowButtonPushed);
  ClearScreen();

  // First go to home position
  ClearScreen();
  TextOut(0, LCD_LINE2, "Homing..." );
  Wait(2000);
  theta1InDegrees = theta2InDegrees = 0.0;
  rotateMotorAbsolutely(theta1InDegrees, theta2InDegrees);
  Wait(2000);
  PlayTone(TONE_E4, 500);

  ClearScreen();
  slavecheck();  // initialize NXT running this program as the Slave
  TextOut(0, LCD_LINE1, "Slave" );
  for(;;) {
    do { // keep checking of Master sent a message
      stringFromMaster = receivefrommaster();
      lenStringFromMaster = StrLen(stringFromMaster);
    } while(lenStringFromMaster == 0);

    // Now Master's message received
    message = "Rec'd: " ;
    strcat(message, stringFromMaster);
    ClearLine(LCD_LINE2); // clear any old Master's string message from LCD
    TextOut(0, LCD_LINE2, message); // display newly received message

    // (1) Find position of comma
    for(i=0; i <= lenStringFromMaster; i++) {
       byteC = StrIndex(stringFromMaster, i); // StrIndex returns ASCII value in
DEC
       if(byteC == 44) { // 44 ASCII is comma
          commaPosition = i;
       }; // end if
     }; // end (1)

    // (2) Extract first number
    strValue01 = Copy(stringFromMaster, 0, commaPosition);
    // ---- message = "str1: " ;
    // ---- strcat(message, strValue01);
    theta1InDegrees = StrToNum(strValue01);
    theta1 = theta1InDegrees * PI/180; // [rad]

    // (3) Extract second number.  NB: Format has 1 whitespace after comma
    strValue02 = Copy(stringFromMaster, commaPosition+2, lenStringFromMaster);
    // --- message = "str2: " ;
    // --- strcat(message, strValue02);
    theta2InDegrees = StrToNum(strValue02);
    theta2 = theta2InDegrees * PI/180; // [rad]

    // Forward Kinematics equations yield end-effector position (xP0, yP0)
    xP0 = l1*cos(theta1) + l2*cos(theta1 + theta2); // [mm]
    yP0 = l1*sin(theta1) + l2*sin(theta1 + theta2); // [mm]
    // End-effector position in LEGO studs
    xP0InStuds = ceil(xP0 / mmPerStud); // round up [stud]
    yP0InStuds = ceil(yP0 / mmPerStud); // round up [stud]

    TextOut(0, LCD_LINE3, "Will go to:" );
    TextOut(0, LCD_LINE4, FormatNum("xP0 = %3d studs" , xP0InStuds) );
    TextOut(0, LCD_LINE5, FormatNum("xP0 = %3.3f mm", xP0) );
    TextOut(0, LCD_LINE6, FormatNum("yP0 = %3d studs" , yP0InStuds) );
    TextOut(0, LCD_LINE7, FormatNum("yP0 = %3.3f mm", yP0) );
    // Prompt user to begin motion
    TextOut(0, LCD_LINE8, "Yes: hit ->");
    do {
       rightArrowButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
    } while(!rightArrowButtonPushed);
    ClearScreen();

    rotateMotorAbsolutely(theta1InDegrees, theta2InDegrees);
    Wait(2000);
    TextOut(0, LCD_LINE2, "Back to Home" );
    theta1InDegrees = theta2InDegrees = 0.0;
    rotateMotorAbsolutely(theta1InDegrees, theta2InDegrees);
    Wait(2000);
    PlaySound(SOUND_DOUBLE_BEEP);
```

```
     // (4) Tell master ready for new message
    sendtomaster(strOkFromSlave);
    ResetSleepTimer(); // don't time out and shut off Brick

  } // end for
} // end main
```