```
// FILE: PC-M-S-1_0a.nxc - Works! Hallelujah!
// DATE: 04/15/20 11:31 - Works!
// AUTH: P.Oh
// DESC: Scilab runs serialPc-M-1_0a.sce on PC to serially send a pair of
//       angles in a string.  Master NXT (running this code) receives and
//       verifies string and extracts angles.  Master NXT then
//       sends Bluetooth message containing these angles, to Slave.  Slave NXT
//       runs btS-R-1_0a.nxc applies these angles to forward kinematics
//       and command the XL-320 servos of the Lego 2-DOF planar manipulator
// VERS: 1_0a: based on btM0_1f.nxc
//       Works with Slave (btS-R-1_0a.nxc) and PC (serialPc-M-1_0a.sce)

#include "protocol0_2a.h"

task main() {

  // Bluetooth related variables
  string stringFromSlave;         // any messages from slave
  int i;                          // dummy index
  string strMaster;               // string to be sent by Master
  string message;                 // string containing message
  string ok = "OK" ;              // OK message for Slave -> Master
  string roger = "ROGER" ;        // ROGER message for Master -> PC

  // Serial port related variables
  byte readBuffer[];              // array to store bytes received from PC
  string charsRead;               // string of ASCII characters read from PC
  int lenCharsRead;               // strlen of charsRead
  byte byteC;                     // ASCII value of character read
  int atPosition;                 // position in string of @ character
  bool atPositionFound;           // @ character found
  int commaPosition;              // position in string of , character
  string strValue01, strValue02;  // extracted numbers as strings
  float value01, value02;         // numeric values of extracted string

  // Set up NXT's serial port
  UseRS485();                               // (1) Configure S4 for RS-485
  RS485Enable();                            // (2) Activate RS-485
  RS485Uart(HS_BAUD_4800, HS_MODE_DEFAULT); // (3) Baud and default parity
  Wait(MS_1);                               // (4) Brief wait for port settings

  TextOut(0, LCD_LINE1, "Master" );
  mastercheck(); // check Master bluetooth connection

  while(true) { // read and display strings received from PC until abort
      while(!RS485DataAvailable()) {
         // if no ASCII chars available, then do nothing
      };
      atPosition = 0;
      atPositionFound = FALSE;

      // Some character(s) is on the serial port, so read and check it
      RS485Read(readBuffer);
      // Convert bytes into ASCII string
      charsRead = ByteArrayToStr(readBuffer);
      message = "PC->M:" ;
      strcat(message, charsRead);
      TextOut(0, LCD_LINE2, message);
      lenCharsRead = strlen(charsRead);
      for(i=0; i<=lenCharsRead; i++) {
          byteC = StrIndex(charsRead, i);
          if(byteC == 64) { // 64 DEC is ASCII character for @
             atPosition = i;
```

```
                atPositionFound = TRUE;
                ClearLine(LCD_LINE5); // clear @: None message from LCD
             }; // end if
        }; // end for loop to check for @ character
        if(atPositionFound != TRUE) {
                TextOut(0, LCD_LINE5, "@: None" );
        };
        if(atPositionFound == TRUE) { // valid message received
            PlayTone(TONE_A3, 100);
            // (1) find comma position
            for(i=0; i<=lenCharsRead; i++) {
                byteC = StrIndex(charsRead, i); // StrIndex returns ASCII value
                if(byteC == 44) { // 44 DEC is ASCII is comma
                    commaPosition = i;
                };
            }; // end for loop checking for comma character
            // (2) Extract first number
            strValue01 = Copy(charsRead, atPosition+1, commaPosition);
            value01 = StrToNum(strValue01);
            // (3) Extract second number.  NB: Format has 1 whitespace after comma
            strValue02 = Copy(charsRead, commaPosition+1, lenCharsRead);
            value02 = StrToNum(strValue02);
            TextOut(0, LCD_LINE3, FormatNum("deg01:%3.2f" , value01) );
            TextOut(0, LCD_LINE4, FormatNum("deg02:%3.2f" , value02) );
            Wait(200);
            // (4) Create proper string to send to Slave
            strMaster = StrCat(strValue01, strValue02);
            message = "M-->S:" ;
            strcat(message, strMaster);
            TextOut(0, LCD_LINE6, message);
            // (5) Send resulting string to Slave
            sendtoslave(strMaster);
            ResetSleepTimer(); // keep Brick awake for Bluetooth connection
            // (6) Wait until Slave says OK
            do {
               stringFromSlave = receivefromslave();
               // keep checking until slave acknowledges with "OK"
               Wait(500);
            } while(strcmp(stringFromSlave, ok) != 0);
            message = "S-->M:" ;
            strcat(message, ok);
            TextOut(0, LCD_LINE7, message);
            // (7) Tell PC ready to receive next message
            RS485Write(roger);
            message = "M->PC:" ;
            strcat(message, roger);
            TextOut(0, LCD_LINE8, message);
        }; // end if atPositionFound
        readBuffer = 0;
        Wait(5000);  // so that user can read LCD
        ClearLine(LCD_LINE8); // clear M->PC roger from LCD
        ClearLine(LCD_LINE7); // clear S->M ok from LCD
        ClearLine(LCD_LINE6); // clear M->S string from LCD

    }; // end while(true)
} // end main
```