Hands-on Lab

NXC Programming – RS-485 Communications

The Lego NXT Brick can communicate with other peripherals via RS-485 serial communications. Port 4 on the Brick provides this high-speed full-duplex capability. Many peripherals like scanners, joysticks, and keypads use this serial communication protocol. RS-485 can also connect the NXT Brick to other micro-processors and computers that have this port. This lab introduces RS-485 NXC programming.

Concept 1 ~~Master-Slave~~ Leader-Follower Communications

An NXT cable is connected on Port 4 of two NXT Bricks. This allows the two Bricks to communicate via RS-485; one will be called the ~~Master~~ Leader and the other, a ~~Slave~~ Follower.

```
// FILE: 485Master1_0.nxc - Works!
// DATE: 09/26/16 12:45
// AUTH: P.Oh
// DESC: Two NXT bricks connected together on their Port S4 (i.e. RS-485 communications)
//       This code runs on Master brick.  485Slave1_0.nxc runs on Slave brick.
//       As long as Slave is on and sending messages, Master iterates and displays number

inline void WaitForMessageToBeSent()
{
  while(RS485SendingData())
  Wait(MS_1);
}

task main() {
  UseRS485(); // (1) Port S4 configured for RS485
  RS485Enable(); // (2) turn on RS485
  RS485Uart(HS_BAUD_DEFAULT, HS_MODE_DEFAULT); // (3) initialize UART to default values
  Wait(MS_1); // (4) wait a bit so all's activated

  int i;
  byte buffer[];
  string msg;
  byte cnt;

  while (true) {
    msg = "Master " + NumToStr(i);
    TextOut(0, LCD_LINE1, msg);
    // send the # of bytes (5 bytes)
    cnt = ArrayLen(msg);
    SendRS485Number(cnt);
    WaitForMessageToBeSent();

    // wait for ACK from recipient
    until(RS485DataAvailable());
    RS485Read(buffer);

    // now send the message
    SendRS485String(msg);
    WaitForMessageToBeSent();

    // wait for ACK from recipient
    until(RS485DataAvailable());
    RS485Read(buffer);

    i++;
  }

  // disable RS485 (not usually needed)
  RS485Disable();
} // end of main
```

Figure 1A: Listing for `485Master1_0.nxc`

The first Brick will be deemed ~~Master~~ Leader and will execute `485Master1_0.nxc`. The NXC program begins by configure Port 4 for RS485 communications at the Brick's default settings. The NXC constant `HS_BAUD_DEFAULT` represents 921,600 BPS, the fastest rate available.

An endless `while` loop increments a variable (`i`). The string `msg` contains the characters "`Master` " plus the value of the variable (`i`). The NXC function `ArrayLen` calculates the number of bytes for the resulting string (`msg`) and stores it in the variable cnt. `SendRS485Number` transmits this number to the ~~Slave~~ Follower.

To confirm that the ~~Slave~~ Follower received the transmission, the NXC function `RS485DataAvailable` is called. Once confirmed, the ~~Master~~ Leader sends `msg` via a call to `SendRS485String`.

```
// FILE: 485Slave1_0.nxc - Works!
// DATE: 09/26/16 12:47
// AUTH: P.Oh
// DESC: Two NXT bricks connected together on their Port S4 (i.e. RS-485
communications)
//       This code runs on Slave brick.  485Master1_0.nxc runs on Master brick.
//       When Slave is off, then Master stops.  When Slave is on, the Master iterates

inline void WaitForMessageToBeSent()
{
  while(RS485SendingData())
  Wait(MS_1);
}

task main() {
  UseRS485(); // (1) Port S4 configured for RS485
  RS485Enable(); // (2) turn on RS485
  RS485Uart(HS_BAUD_DEFAULT, HS_MODE_DEFAULT); // (3) initialize UART to default values
  Wait(MS_1); // (4) wait a bit so all's activated

  int i;
  byte buffer[];
  string msg;
  byte cnt;

  while (true) {
    msg = "Slave " + NumToStr(i);
    TextOut(0, LCD_LINE1, msg);
    // send the # of bytes (5 bytes)
    cnt = ArrayLen(msg);
    SendRS485Number(cnt);
    WaitForMessageToBeSent();

    // wait for ACK from recipient
    until(RS485DataAvailable());
    RS485Read(buffer);

    // now send the message
    SendRS485String(msg);
    WaitForMessageToBeSent();

    // wait for ACK from recipient
    until(RS485DataAvailable());
    RS485Read(buffer);

    i++;
  }

  // disable RS485 (not usually needed)
  RS485Disable();
} // end of main
```

Figure 1B: Listing for `485Slave1_0.nxc`

A second Brick runs `485Slave1_0.nxc` given in **Figure 1B** and acts as a ~~Slave~~ Follower. Here, the program begins by configure Port 4 for RS-485 communications at default settings. Like **Figure 1A**, an endless while loop sends to the ~~Master~~ Leader, the number of bytes for the string it will send, waits for acknowledgement from the ~~Master~~ Leader, and then transmits strings.

Congratulations! You can program the NXT Brick for RS-485 communications.

Exercises

1.1 Write the following NXC program. The ~~Master~~ Leader Brick iterates incrementally by 1, from 1 to 20 (e.g. using a for-loop) and displays this value on its screen. At each iteration this Brick also transmits via RS-485, the value to the ~~Slave~~ Follower Brick. The ~~Slave~~ Follower Brick upon receiving this value displays the corresponding value squared.